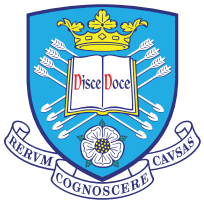


The University Of Sheffield.

Using Bayesian Statistics, Random Fields, and Neural Networks to Process Images from Single Molecule Localisation Microscopy

Nicola Connell

Submitted for the degree of Doctor of Philosophy
Department of Physics and Astronomy
University of Sheffield

August 2020

ABSTRACT

Over time, optical microscopy has improved to allow for increasing resolution. Traditional light microscopes can resolve distances down to 200nm due to the diffraction limit of light. Significant challenges arise when trying to see structures smaller than this.

Single Molecule Localisation Microscopy (SMLM) techniques, including stochastic optical reconstruction microscopy (STORM) [1], photo-activated localisation microscopy (PALM) [2] and fluorescence photo-activated localisation microscopy (FPALM) [3] have brought nanoscale resolution into biology, circumventing the diffraction limit on microscopy by exploiting the photoblinking ability of some fluorophores.

There remain a number of technical challenges involved with SMLM, in both acquisition and analysis of the data. Extracting useful information from reconstructed images (particularly counting the number of molecules of interest in the sample) is still difficult in samples which are densely-labelled with many overlapping fluorophores. Datasets are also typically very large so simply processing and storing data can be a technical challenge itself.

This thesis postulates that a combination of Gibbs sampling by means of a Mixture Model and a Modified Ising Model (MIM) could be used to improve the accuracy of SMLM reconstructions by characterising each pixel of the image into two populations: signal or noise. Through testing on simulated datasets, it was found that this method of classifying pixels can successfully denoise images, and reduce the amount of disk space required to save them by, on average, 95%. Statistical knowledge about the two populations is also determined using a parameterless system.

It was found that using the processed data in combination with machine learning algorithms [4], could provide an improved method for accurately counting fluorophores of interest in sample sections. The accuracy reached values of 88.6%, 89.3%, 90.8% and 98.9%, and loss values of 0.229, 0.213, 0.191 and 0.017 for the non-processed datasets, the Mixture Model output, the MIM output and the product of the Mixture Model and MIM outputs respectively. Using the processed datasets also decreased the number of iterations required for the neural network (NN) to reach a high accuracy/ low loss. These accuracies

and loss values were reached on the training dataset. This method could be used on real SMLM datasets, potentially providing a way to determine the absolute numbers of biological molecules of interest in cells, allowing biologists to extract more quantitative information from SMLM techniques.

The NN was adapted to take in three dimensional data instead of just single images. This data included the previous, current and following images. Unfortunately this only managed to get to an accuracy of 42.4% using the 4 different datasets as before: non-processed data, the Mixture Model output, the MIM output and the product of the Mixture Model and MIM outputs

REFERENCES

[1]M. Rust, M. Bates and X. Zhuang, 'Stochastic optical reconstruction microscopy (STORM) provides sub-diffraction-limit image resolution', Nature methods **3**, 793–795 (2006).

[2]E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz and H. F. Hess, 'Imaging intracellular fluorescent proteins at nanometer resolution.', Science (New York, N.Y.) **313**, 1642–5 (2006).

[3]S. T. Hess, T. P. K. Girirajan and M. D. Mason, 'Ultra-high resolution imaging by fluorescence photoactivation localization microscopy.', Biophysical journal **91**, 4258–72 (2006).

[4]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, http://www.deeplearningbook.org (MIT Press, 2016).

ACKNOWLEDGEMENTS

I would like to thank the multitude of people who, without their help and support, I would not have been able to complete this thesis. I'd like to apologise in advance for any names I may have missed, you still have my gratitude!

First and foremost I would like to thank my two supervisors: Ashley Cadby and Miguel Juárez who guided me in this project from start to finish. I would never have been able to complete this without you.

To my family, thank you so much. You were always there to help and you are amazing. Especially my parents for allowing me to come back and live with them for the COVID pandemic where I wrote up the majority of my thesis. I could not have better parents. To both my brothers, my 'sister-in-law', and my little niece, you're all fantastic.

I would also like to extend my gratitude to the people in my physics group ("Group Ash"), both past and present: Amy, Tania, Sam and James ("Sames"), Liyana, Olivier, Elliot, Dylan and Yin. You were always there to help, with either the academic or as just a friend, especially with the trips to FOM.

My personal tutor, Simon Goodwin was immensely helpful towards the end when I was really struggling to write my thesis and losing a lot of motivation.

I would not have been able to do this without the help and support of my friends. There are numerous friends that I would like to thank, and I apologise if I have missed any out of this section, but your help has not been forgotten! I would not have been able to make it through without you: Chris, Josh, Dan, Dave, Alastair, Stevie G, Bev and Jason, Amy, Seddon, Mike, Batt, Zsoka, Nikki and Johnny. Also to members of both my hockey club (SUBHC) and badminton club (Oakbrook badminton club). I enjoyed all the socials, the pub trips, the walks and camping and everything else we did together.

I would like to extend specific thanks to Chris for fixing my laptop and being a beautiful and wonderful man! He was always there when I needed a moan and a natter (and maybe a takeaway with a bottle of wine).

Also, special thanks to Dan for lending me his laptop and being an awesome, lovely person. Again, a great shoulder to cry and on always there when I needed someone.

To all the people that helped proof read my thesis, no matter how small, many thanks are given. These include Dan, Chris, Steve, Pete, my mum and my dad. Extra special thanks go out to Amy who gave very detailed and constructive comments.

The assorted menagerie of animals who have helped me cope go without saying; Chico and Alfie, Mr Marmalade and Boo, Basil (RIP), Pippa and Pluto, Rory, and Shady (RIP) to name but a few.

Finally, I would like to thank The University of Sheffield and the ESPRC for their help and funding.

CONTENTS

Abstract iii

1 LIGHT, LENSES: PHYSICS AND BIOLOGY 1
- 1.1 Microscopes - an early history . . . 1
- 1.2 Lenses . . . 2
- 1.3 Electron microscopy (EM) . . . 8
- 1.4 Defying the limits . . . 9
  - 1.4.1 Deconvolution . . . 9
  - 1.4.2 Confocal Microscope . . . 10
- 1.5 Fluorophores . . . 10
- 1.6 High-Resolution Fluorescence Microcopy . . . 12
- 1.7 Super-resolution Microscopy . . . 13
  - 1.7.1 Single Molecule Localisation Microscopy (SMLM) 16
  - 1.7.2 Super Resolution Optical Fluctuation Microscopy (SOFI) . . . 20
- 1.8 Outline and scope of this thesis . . . 20

2 METHODS AND MATERIALS 35
- 2.1 Introduction . . . 35
- 2.2 Reconstruction Algorithms . . . 35
  - 2.2.1 Finding regions of interest (ROI) . . . 35
  - 2.2.2 Localising using fitting . . . 36
  - 2.2.3 Least-squares (LS) criterion . . . 36
  - 2.2.4 Maximum likelihood estimate (MLE) . . . 37
  - 2.2.5 Localising without fitting . . . 37
  - 2.2.6 Localising multiple emitters simultaneously . . . 38
  - 2.2.7 Image estimation . . . 39
  - 2.2.8 Testing localisation algorithms . . . 40
- 2.3 Statistics . . . 41
  - 2.3.1 Bayesian Inference . . . 42
  - 2.3.2 Markov Chain Monte Carlo (MCMC) . . . 43
  - 2.3.3 Simple Example of Markov Chain Monte Carlo (MCMC) . . . 44
  - 2.3.4 Pseudocode . . . 49
- 2.4 Photon Detection . . . 52
- 2.5 Artificial intelligence (AI) . . . 54
  - 2.5.1 A neural network (NN) . . . 54
  - 2.5.2 Deep learning (DL) . . . 56
- 2.6 Acronym List . . . 57

3 SIMULATIONS 71
3.1 Summary . . . 71
3.2 Introduction . . . 71
3.3 Real Data Production . . . 74
3.4 Noise . . . 76
3.4.1 Camera Noise . . . 76
3.4.2 Noise within the signal . . . 79
3.4.3 Autofluorescence . . . 79
3.5 Simulations . . . 80
3.6 Chosen Statistics . . . 82
3.7 Flowchart . . . 83
3.8 Differences . . . 85
3.9 Results . . . 86
3.9.1 ThunderSTORM results . . . 86
3.9.2 Comparison with real images . . . 88
3.9.3 Time traces of images . . . 89
3.10 Discussion . . . 92
3.11 The Algorithm . . . 93
4 MIXTURE MODEL AND THE MODIFIED ISING MODEL 105
4.1 Summary . . . 105
4.2 Introduction . . . 105
4.3 The First Stage: Denoising (The Mixture Model) . . . 106
4.3.1 The Likelihood . . . 107
4.3.2 The Priors . . . 107
4.3.3 Values chosen for priors . . . 109
4.3.4 The Posterior . . . 111
4.3.5 Full Conditionals . . . 111
4.3.6 MCMC method . . . 112
4.3.7 Sensitivity Test . . . 121
4.3.8 Pseudocode for this Mixture Model . . . 125
4.3.9 Limitations of the Mixture Model . . . 128
4.4 The Second Stage: The Modified Ising Model (MIM) . . . 130
4.4.1 Markov Random Fields (MRF) . . . 130
4.4.2 Metropolis-Hastings Ratio . . . 132
4.4.3 Modified Ising Model (MIM) . . . 132
4.4.4 Sensitivity Test . . . 136
4.5 Applications . . . 139
4.5.1 Data Reduction . . . 140
4.5.2 Measuring Signal and Noise (signal-to-noise ratio (SNR)) . . . 143
4.5.3 Localisation . . . 144
4.5.4 Preprocessing - Counting . . . 144

4.6 Discussion . . . 144
4.7 Further Work . . . 146
4.8 The Algorithm . . . 147
5 COUNTING 159
5.1 Summary . . . 159
5.2 Introduction . . . 160
5.3 Background . . . 161
5.4 The Data Used . . . 163
5.5 Approaches/ Methods . . . 164
5.5.1 Two-Staged Mixture Model . . . 165
5.5.2 Using the Mixture Model Output . . . 165
5.5.3 Machine Learning (ML) . . . 170
5.5.4 Neural networks (NNs) . . . 175
5.5.5 Results from the neural network (NN) . . . 184
5.6 Results and Discussion . . . 195
5.7 Further work . . . 198
5.8 The Algorithm . . . 198
6 DISCUSSION 211
6.1 Simulations . . . 211
6.2 Mixture Model, Modified Ising Model (MIM) . . . 212
6.3 Counting . . . 213
6.4 Further work . . . 215
A APPENDIX A: INTRODUCTION 231
A.1 Example MCMC code . . . 231
A.2 Continuous Distributions . . . 233
B APPENDIX B: SIMULATIONS 235
B.1 Simulations Code . . . 235
B.1.1 Adding molecules to the image . . . 238
B.1.2 Creating a mask . . . 238
B.1.3 Binning the data in space . . . 238
C APPENDIX C: GIBBS ISING 241
C.1 Gibbs Ising Code . . . 241
C.1.1 Gibbs Sampler for two populations . . . 242
C.1.2 Separating the signal and noise . . . 244
C.1.3 Gibbs Sampler for one population . . . 244
C.2 Sensitivity Test Results . . . 249
D APPENDIX D: COUNTING 253
D.1 The two stage Gibbs Algoirhtm . . . 253
D.2 Using the Gibbs Statistics to calculate the change in 'on' fluorophores . . . 256
D.3 The support vector machines (SVM) classification learner 268

D.4 The K-nearest neighbours (KNN) classification learner . 271
D.5 The neural network (NN) 274
D.6 The 3D neural network (NN) 279
D.6.1 Main Code 279
D.6.2 3D neural network (NN) - the main code 283
D.6.3 3D neural network (NN) - the NN 300

1

LIGHT, LENSES: PHYSICS AND BIOLOGY

For nearly 2000 years, [1] humans have been using lenses to image objects which are inaccessible to the naked eye. For example, we use imaging techniques in medicine to look inside objects, in microbiology to look at very small objects, and in astronomy to look at objects which are very far away. This thesis focuses on imaging in the field of microscopy, particularly for biological applications.

The human eye can discern objects down to around 0.07mm in diameter [2] which is roughly the size of a single strand of hair. In order to image objects smaller than this, for example organelles inside cells, a magnified image of the object in question must be produced using combinations of lenses.

1.1 MICROSCOPES - AN EARLY HISTORY

Although imaging can be a huge challenge for scientists, lenses have been used for millennia to make objects appear larger. The earliest microscopic observation was circa A.D. 63, when Seneca noticed that a glass globe filled with water made letters larger and clearer [1]. The Romans also used curved planes of glass to magnify objects by around 2.5 times in the first century [3]. Some optical properties of lenses were known by medieval writers such as Roger Bacon, and the first spectacles with convex lenses were created in the 1300s by Salvino d'Amarto [3].

The first publication that used a magnifying glass was in 1592, that contains hand drawn illustrations that go into more detail than could be seen by the naked eye. The illustrations are of common objects in nature. However, they are drawn with minute accuracy with enlarged details which would have been indistinguishable to the unaided eye. [1]

Up to this point is often referred to as the Pioneering Period of Microscopical discovery and it brought together the understanding that lenses could be used in both microscopes and telescopes to make objects look bigger. Although telescopes were generally accepted straight away and used to further research in astronomy, microscopes took around half a century before they started being used for scientific research and the observations in the beginning were unrelated and scattered across many different areas [1]. Before this they were seen as mere amusements [4].

Optical or light microscopes, use visible light and lenses to magnify objects. Objects can be magnified using just one lens, but modern-day microscopes use a combination of lenses to achieve higher magnification power [5]. The compound microscope uses two or more lenses, both an objective lens and an eyepiece, to further magnify objects. It is unknown who specifically came up with this idea however credit is often given to Hans and Zachcrias Hanssen in the 1590's [6, 7] due to letters that were sent in this period describing this particular set up.

In 1661, Marzello Malpighi was the first man to publish a scientific discovery using a microscope, thus starting the Classical period of microscopical discovery. He looked at dried lungs of frogs and postulated that blood flows in a closed system. He also discovered the existence of capillaries. [6].

Robert Hooke and Antoni van Leeuwenhoek advanced microscopy and discovered the existence of microscopic organisms between 1665-83 [5]. Often Leeuwenhoek is described as the 'first of the microbe hunters' due to his letter from 1676 [6]. However, Hooke had previously discovered several varieties of fungi, and published "Micrographia", an illustrated book on microscopy in 1665 [8]. Hooke also coined the term 'cell' after looking at cork through a microscope and seeing a regular series of shapes and likening them to a monk's cell [6].

Many more discoveries were found using compound microscopes from this point, but it was not until the late 1800's that the resolving power of these microscopes were fundamentally limited by the diffraction of light. It was at this point that new microscopes and techniques were starting to be developed.

1.2 LENSES

All waves, for example electromagnetic waves, travel at different speeds through different media. This leads to a change in the direction that the light is travelling when transitioning a boundary between two different

materials. This property is known as refraction and is described by Snell's law [9, 10] in equation (1.1):

$$n_1 \sin\theta_i = n_2 \sin\theta_r, \tag{1.1}$$

where θ_i and θ_r are the angle of incidence and refraction respectively and n_1 and n_2 are the refractive indices of materials 1 and 2 respectively. The refractive index of a medium is the ratio of the speed the wave travels in a vacuum, c, to the speed it travels in the medium, v_i:

$$n_i = \frac{c}{v_i}. \tag{1.2}$$

Although Snell's law was first realised by Ibn Sahl in 984A.D. [9, 11], it is credited to Astronomer Snellius and his experimental work in 1621 [9]. Snell's law shows that light travelling from a medium with a low refractive index to a medium with a higher refractive index will bend towards the normal. This phenomena is shown in figure 1.1.

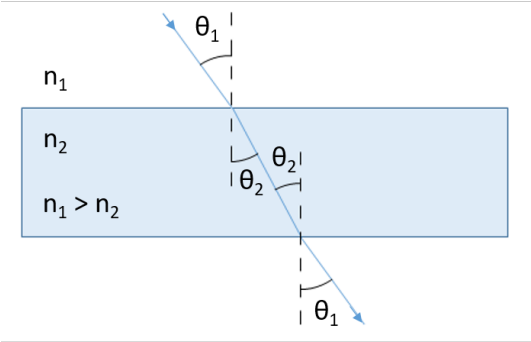


Figure 1.1: A wave transitioning a boundary from a material with a lower refractive index to a material with a higher refractive index will be refracted towards the normal ($\theta_2 < \theta_1$).

By changing the shape of the interface between materials, refraction can be exploited and light can be bent to a focal point, f [12]. There exists many different shapes of lenses with straight, concave or convex edges which changes the position of the focal point. Common thin lenses can be seen in figure 1.2. The lens-maker formula or thin-lens equation below (equation (1.3)) [12] can be used to calculate the focal point of a lens.

$$\frac{1}{s_o} + \frac{1}{s_i} = \frac{1}{f} = (n_l - n)\left(\frac{1}{R_1} - \frac{1}{R_2}\right), \tag{1.3}$$

where s_o and s_i are the distances from the object and image respectively, n_i and n are the refractive index's of the lens and external media

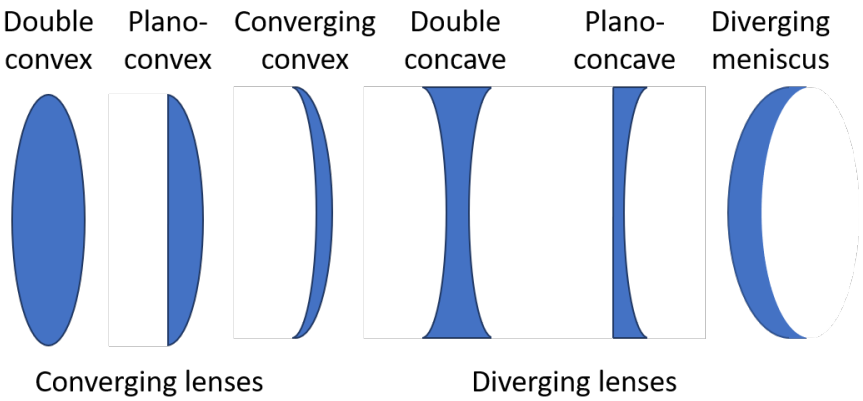


Figure 1.2: Shapes of common thin lenses. The types shown are: a) double convex, b) plano-convex, c) converging meniscus, d) double concave, e) plano-concave, f) diverging meniscus. a-c) are all converging lenses, which d-f) being diverging lenses (adapted from [13])

respectively (e.g. if the lens was in air then $n \approx 1$). R_1 and R_2 are the radii of curvature of the first and second edge of the lens.

The most commonly used lenses in microscopes are double convex, or biconvex lenses and plano-convex, or straight-convex, lenses and they are often put together in the form of compound lenses. Compound lenses are simply a combination of different lenses that fit together to make a more complicated lens and change the direction of the light in question.

Another phenomenon that occurs in waves is diffraction [10]. All transverse waves, from water waves to electromagnetic waves, undergo diffraction when reaching a boundary or going through an aperture. This causes the waves to spread out, changing the wavefront. When going through a small aperture, for example a small slit or a circular hole that is comparable to the wavelength, diffraction causes constructive and destructive interference to occur [10, 12]. Light propagating after the aperture has a very different wavefront to the light before. When light passes through a small slit, a double slit, or even a series of slits, the pattern of light on a screen would be a series of lines starting the brightest in the central maxima and decreasing in intensity as you move away from the centre. The diffraction pattern from a single slit experiment is shown in figure 1.3.

When going through a small circular aperture the light diffracts to create a central bright spot surrounded by concentric rings which decrease in intensity the further from the centre you move [15]. This pattern is called an Airy disk [12], named after George Airy [15], and is

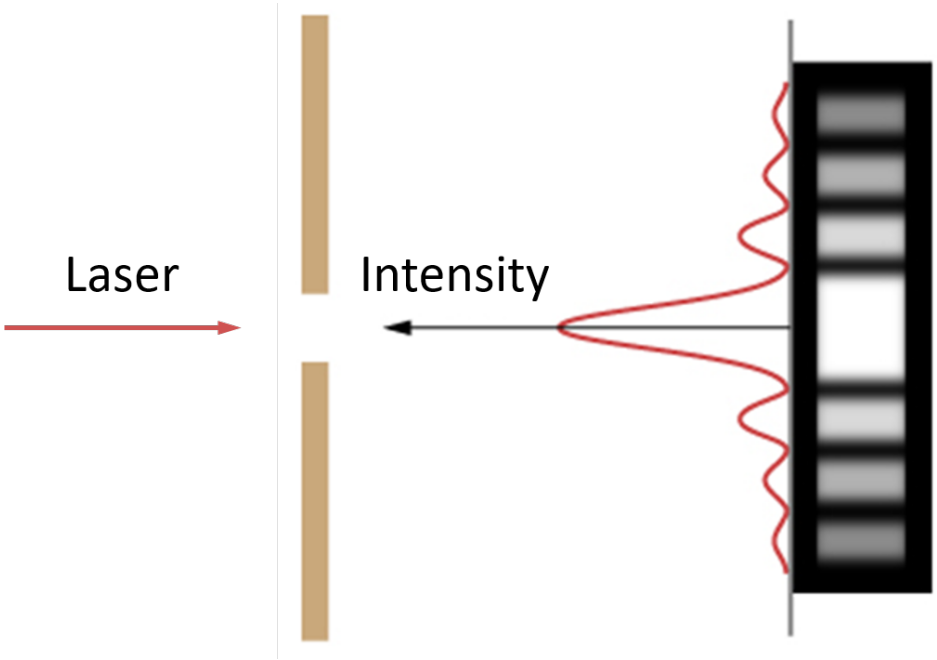


Figure 1.3: Incident monochromatic light enters a single thin slit from the left, creating a diffraction pattern on the right. A visual representation of the diffraction pattern can be seen, along with an intensity profile (red), showing a central intensity maximum with thinner, dimmer maxima on either side of the maximum. [14]

shown in figure 1.4. When light from a point source is imaged through a lens onto a screen or camera, this same pattern, called a point spread function (PSF), is formed.

The diffractive property of light makes it fundamentally impossible to focus light from a point source back down to a single point and makes traditional optical microscopes have a resolution limit which is diffraction limited.

When point sources are very close to each other in object space, their PSFs can overlap in image space, as shown in figure 1.4. If the physical separation of these sources is less than half of the size of the microscope's PSF, it is no longer possible to differentiate between the sources in the image. In 1873 Ernst Abbe developed the first equation to describe this resolution in the x and y direction [17]:

$$\text{Resolution}(x, y) = \frac{\lambda}{2\text{NA}} = \frac{\lambda}{2n\sin(\theta)}, \tag{1.4}$$

where λ is the wavelength of light, $\text{NA} = n\sin\theta$ is the numerical aperture (NA) [18] of the objective lens, with n being the refractive

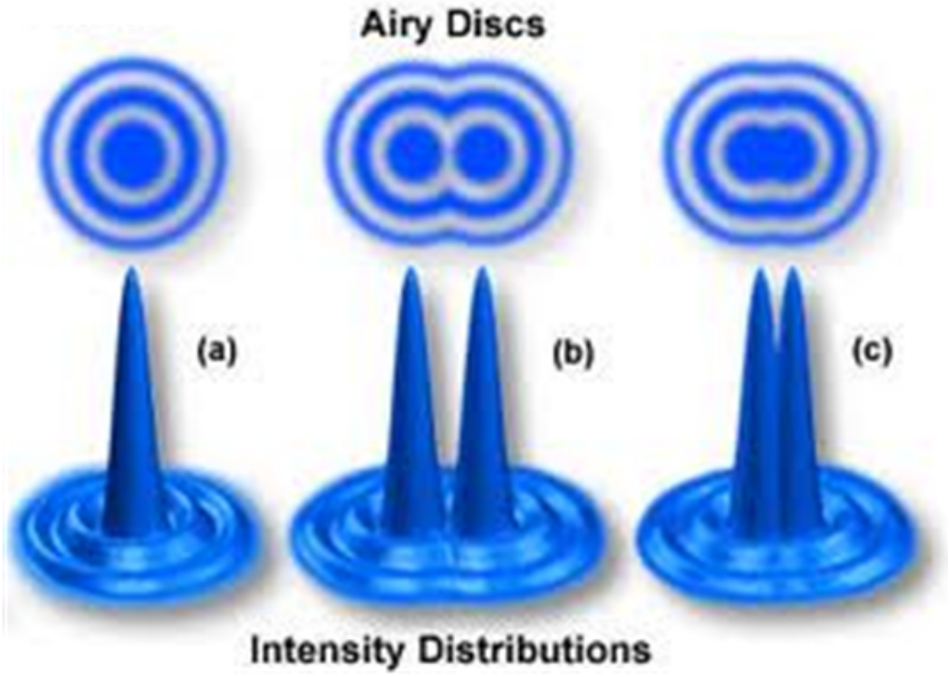


Figure 1.4: Airy disks and intensity distributions for a) a point spread function (PSF), b) two PSFs that are just resolvable and c) two PSFs that are unresolvable. [16]

index of the lens and θ the collection angle of the lens. This is shown in figure 1.5.

There are several other different equations to describe the resolution limit. They all have slightly different constants in the equation. Another popular resolution equation is the Rayleigh Criteria [19]:

$$\text{Resolution}(x, y) = 0.61\frac{\lambda}{NA} = 0.61\frac{\lambda}{n\sin\theta}. \tag{1.5}$$

Here you can see that the constant in front of the fraction has been changed from $\frac{1}{2}$ (or 0.5) to 0.61.

Using equations (1.4) and (1.5), the best achievable resolution with a typical oil objective lens (NA = 1.4) and visible light ($\lambda \approx 500$ nm) is approximately 200 nm. This is sometimes referred to as Abbe's diffraction limit [20].

At this resolution, only large organelles separated by significant distances within the cells will be discernible. Finer cellular structures cannot be determined, so it would not be possible to perform co-localisation studies to investigate the relationship between particular

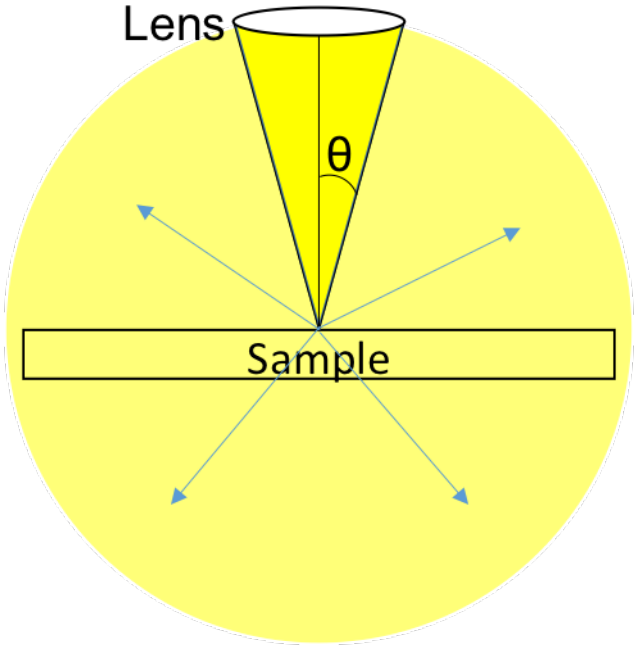


Figure 1.5: The collection angle of a lens. A point source in the sample emits light in all directions. Only light collected by the lens can contribute to the final image. Using a lens material with a higher refractive index can increase the collection angle of the lens, leading to higher resolution images.

cell components to help determine how the cell functions. Figure 1.6 shows some examples of structures at different sizes and what can be seen with this resolution.

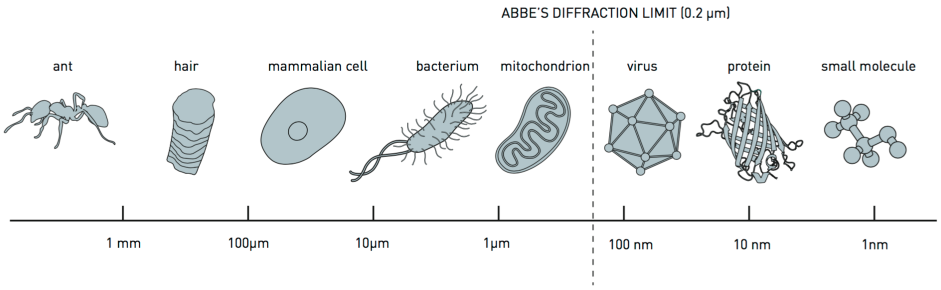


Figure 1.6: An array of biological specimens and their relative sizes. At the typical Abbe resolution limit of 200 nm, viruses and smaller internal molecules of cells will not be visible. [20]

Equations (1.4) and (1.5) indicate that the resolving power of a microscope can be increased by reducing the wavelength of light used for imaging, or by increasing the NA of the objective lens. In 1910 the first

UV microscopes were created, however it was quickly seen that UV light is toxic to cells [21]. It also causes more background noise due to their higher energy, decreasing the signal-to-noise ratio (SNR).

Although increasing the NA is possible, there is a limit. From equation (1.4) it can be seen that

$$NA = n \sin(\theta). \tag{1.6}$$

Therefore the main limiting factor is n, the refractive index of the imaging medium [22]. This value can range between 1 for air and 1.51 for specialised immersion oils. [22]

Wave-particle duality is a quantum mechanical concept in which all particles or quantum entity can be described as either a wave, or a particle. In 1929, Louis-Victor de Broglie received a Nobel Prize in Physics for the wavelike property of electrons. [23] Due to this phenomenon, high-energy electrons can be used to image samples at an equivalent wavelength which is shorter than that of visible light, leading to a higher resolving power. Electron microscopy (EM) is described in more detail in section 1.3.

1.3 Electron microscopy (EM)

As stated above in equations (1.4) and (1.5) the resolution of microscopes can be increased by using a shorter wavelength and due to wave-particle duality, electrons can be used to achieve this and hence EM was created. The wavelength of the electrons is determined by momentum which can be fine tuned by changing the voltage used to accelerate the electrons.

The first electron lens and EM prototype were produced in 1931 with a magnification of around 400x [24]. There are many different types of EM including the scanning transmission EM which has achieved a resolution better than 50pm and magnifications of around 10,000,000x [25]. This is around a 4000x better resolution than traditional light microscopy is capable of. In 1986 Ernst Ruska received a Nobel Prize in physics for "The Development of the Electron Microscope and of Electron Microscopy". [26]

There are many drawbacks and limitations to using EM. As particles (electrons) are used instead of electromagnetic waves, the sample needs to be imaged in a vacuum so that the electrons are not scattered by air particles before reaching the sample [27]. To increase the contrast of the images, the sample is coated in metal and, as electrons are not very penetrative, only the surface or very thin samples can be imaged.

Due to all of these preparatory measures it is impossible to image live samples using this technique. The cell fixation methods required for EM can cause artefacts to occur, such as membrane deformation/shrinkage. The sample is also badly damaged by ionisation due to the high energy electrons being fired at it. [27]

1.4 DEFYING THE LIMITS

Light microscopy techniques offer the best alternative and are a natural choice to image live samples as they are the least invasive. As the resolution of these techniques is limited to around 200nm, novel methods are required in order to circumvent the fundamental limit of light microscopes, such that smaller structures become visible, thus moving microscopy into nanoscopy.

1.4.1 *Deconvolution*

Deconvolution is an algorithm based program used after images have been taken to try and reverse convolution [28]. It is used in many different areas including seismology, radio astronomy and optical imaging. In optical systems such as fluorescence microscopy (as described in sections 1.6 and 1.7), it corrects the systematic error of blur and helps to bring back the contrast in smaller features.

An image taken with SMLM (described in section 1.7.1) can be thought of as a convolution between the actual photons produced by the sample and the Probability Density Function (PDF) of the microscope:

$$f * PDF = i, \tag{1.7}$$

where f is the ground truth, i is the image recorded and $*$ means to convolute. Deconvolution is performed in the frequency domain by computing the fourier transform (FT) of both the PSF (described in section 1.2) and the image. These are then divided and the result is the FT of the ground truth. The inverse fourier transform (IFT) is computed rendering the final result [28, 29].

Unfortunately the image also includes some noise (described in detail in section 3.4), ϵ:

$$(f * PSF) + \epsilon = i. \tag{1.8}$$

This can make the deconvolution very difficult to perform. The PSF may not be known precisely and then would have to be estimated. This would decrease the accuracy of the deconvolution result. [29]

1.4.2 *Confocal Microscope*

The confocal microscope was invented in 1955 by Marvin Minsky [30], with a patent being filed and accepted in 1961 [31]. Although this microscope uses simple objectives, a higher resolving power can be produced than in a simple microscope. It uses two apertures, one to create point illumination upon a sample, and the second to reject all scattered light except that coming from the central focal point. As the light is focused onto a single point of the sample, this point is then raster scanned across the sample and an image is reconstructed. A simple diagram of the confocal setup is shown in figure 1.7.

[32].

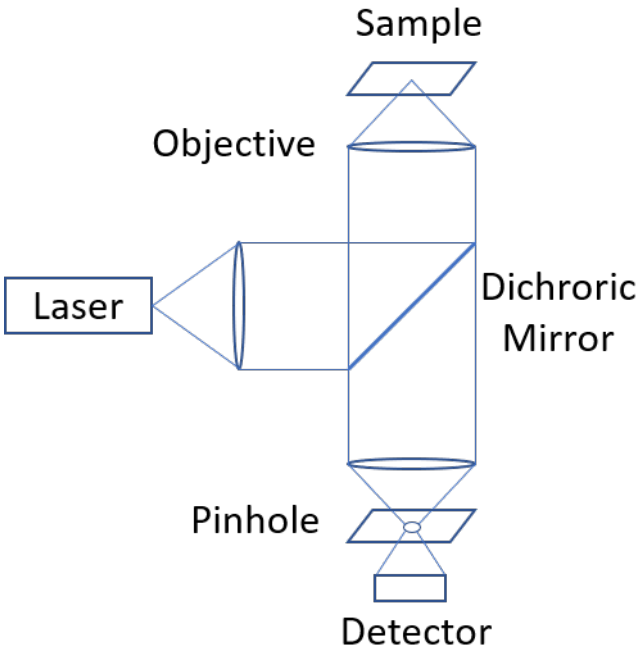


Figure 1.7: The simplified workings of a confocal microscope. (Adapted from [32])

1.5 FLUOROPHORES

As early as 1874 it was postulated that clearer images could be produced if the subject itself were to emit light, rather than to illuminate it by an outside source [32, 33]. This technique uses fluorophores.

Although fluorescence was observed as early as 1560 [34], the name fluorescence was not coined until the early 1800's by George Stokes [35, 36]. A fluorophore was not termed until 1897 by Richard Meyer [37].

Fluorescence is a form of luminescence whereupon a substance emits photons after absorbing electromagnetic radiation [38]. Although some things naturally fluoresce (intrinsic fluorescence or auto-fluorescence), specific molecules that are known to fluoresce are called fluorophores.

Fluorophores contain electrons which are usually in the ground state, S_0. When these molecules are irradiated with photons with an energy equal to the difference between two energy states, the electrons absorb the energy and enter an excited state, S_1:

$$S_0 + h\nu_{ex} \rightarrow S_1, \tag{1.9}$$

where h is Planck's constant and ν_{ex} is the frequency of light. The electron then disperses energy non-radiately to drop down to the first excited singlet state, giving off heat. The molecule then fluoresces by dropping back down to the ground state spontaneously and emitting a photon. This usually has a smaller frequency, or longer wavelength, than the excitation photon:

$$S_1 \rightarrow S_0 + h\nu_{em} + \text{phonon}, \tag{1.10}$$

where ν_{em} is the frequency of emitted light (usually $\nu_{em} < \nu_{ex}$). This process can be described by a Jablonski diagram as shown in figure 1.8.

Fluorophores have many different characteristics including maximum excitation and emission wavelength, quantum yield (QY), lifetime and Stokes shift [39]. These characteristics are all fluorophore specific and vary substantially.

The maximum excitation wavelength is the maximum wavelength (minimum energy) a photon can have that can be used to excite the fluorophore into its excited state. The maximum emission wavelength is the maximum wavelength (minimum energy) a photon can have that is emitted when the fluorophore drops back down to its ground state. [39]

The QY is the efficiency of the fluorescent process; the ratio of the energy from incident light to emitted light. QY is calculated using equation (1.11). [39]

$$QY = \frac{\text{number of photons emitted}}{\text{number of photons absorbed}} = \frac{k_f}{\sum_i k_i}, \tag{1.11}$$

where k_f is the rate constant of spontaneous emission and $\sum_i k_i$ is the sum of all rates of excited decay.

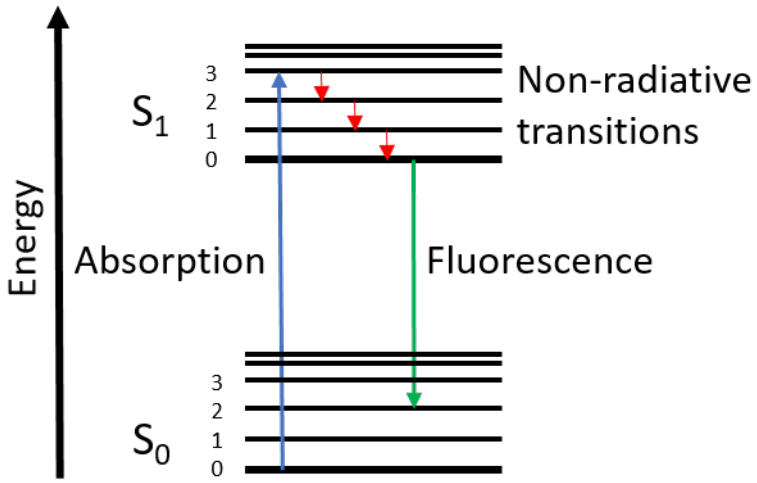


Figure 1.8: A simplified Jablonski diagram showing the fluorescence process. An electron absorbs photon energy and is excited both electronically and vibrationally (blue arrow). The system then relaxes non-radiatively (red arrows) emitting heat before fluorescing. A photon is released with a longer wavelength (green arrow).

The lifetime refers to how long the fluorophore remains in its excited state, S_1, before emitting a photon and returning back to the ground state, S_0 [39]. This process generally follows first order kinetics as follows:

$$[S_t] = [S_0]exp(-\Gamma t), \tag{1.12}$$

where $[S_t]$ is the concentration of molecules in the excited state at time t, $[S_0]$ is the initial concentration and Γ is the decay rate, which is equal to the inverse of the fluorophore lifetime.

Fluorescence is also dependent on the orientation of the incoming light relative to its transition moment and the nature of the sample [38]. The relative orientation of the light can be overcome by using circularly polarised light so that all different orientations of light hit the fluorophores. The nature of the sample can also affect fluorescence: whether the sample is fixed, moveable or where the fluorophores are in the sample (i.e next to the glass slide or deep in the sample).

1.6 HIGH-RESOLUTION FLUORESCENCE MICROCOPY

Although optical microscopes are favoured for imaging live cells due to their non-invasive property, using the above approaches requires high

energy photons/lasers which causes damage to the cells (phototoxicity) and still limits their resolution to around 200nm [40]. Therefore new ways to increase the resolution without damaging the cells were researched.

Three different microscopes that exploit some of the properties of light to produce images are: 2-photon microscopy [41]; Light Sheet Fluorescence Microscopy (LSFM) [41] and total internal reflection fluorescence (TIRF) [42]. The first two of these techniques vary the incident light whereas TIRF utilises the refractive property of light to increase the resolution of the images produced. These techniques all use lasers and therefore phototoxicity can occur in the form of photobleaching and photodamage.

Photobleaching is the photochemical alteration that occurs to the fluorophores due to the laser energy. This causes an irreversible reaction that stops fluoresence from being able to occur by cleaving some of the covalent bonds. [43, 44]

Photodamage occurs to the cells being imaged and is also due to the high energy incoming photons. Again the high energy photons cleave covalent bonds. The higher the energy of the photons and the longer the cells are exposed the more damage that occurs and this can kill cells. [43, 44]

1.7 SUPER-RESOLUTION MICROSCOPY

Super-resolution microscopy allows images to be taken with a resolution that is not limited by the diffraction of light. There are two major groups of functional super resolution: deterministic super-resolution and stochastic super-resolution. Deterministic super-resolution exploits fluorophores' non-linear response to excitation whereas stochastic super-resolution uses fluorophores with complex temporal behaviour and and resolves fluorophores in time. Deterministic super-resolution techniques include stimulated emission depletion (STED) [45], near-field scanning optical microscopy (NSOM) [46] and structured illumination microscopy (SIM) [47]. Stochastic super-resolution techniques include SMLM [48–50] and SOFI [51].

The first super-resolution microscope developed was STED [45]. In 2014 Stefan Hell received a Nobel Prize in Chemistry along with Eric Betzig and William Moerner "for the development of super-resolved microscopy". [20]

The STED microscope exploits the non-linear response of fluorophores and minimises the area of illumination at the focal point [45, 52]. Two lasers are used as shown in figure 1.9. The excitation laser (green) is used to excite the fluorophores into their 'on' state. These can then return to the ground state via several different processes, mainly spontaneous fluorescence emission, or stimulated emission. Stimulated emission is caused by the second laser (red), sometimes called the STED laser. This laser is a similar wavelength to the emitted fluorescence light and it is incident on the fluorophore before spontaneous emission can occur, forcing it to relax into a higher vibrational state as shown in the Jablonski diagram in figure 1.10. It is a doughnut shape that is overlaid on the first laser, and reduces the effective excitation volume as shown in figure 1.9. The photons emitted are red-shifted and therefore can be differentiated from spontaneous emission. Typically this microscope has a resolution between 30-80nm however in 2012 a resolution of 2.4nm was reported [53].

Although a good resolution can be found with this microscope, it requires a complex optical set up which is expensive. It can also take a long time to produce images as the lasers are raster scanned across the sample. Due to this long acquisition time, it is difficult to analyse live processes.

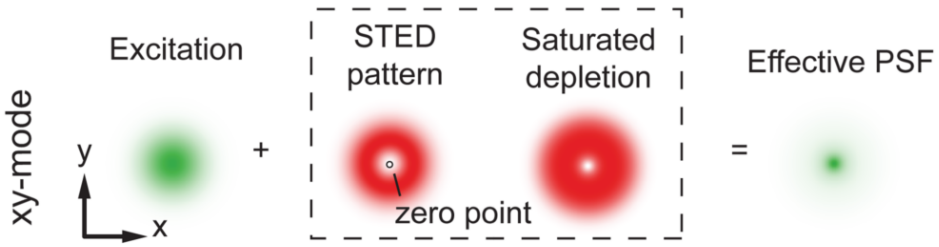


Figure 1.9: STED illumination strategy. The excitation laser (left, green) is overlaid with the STED laser (centre, red), causing stimulated emission at the edges of the excitation beam. Together, these create a smaller effective PSF (right, green), exciting a much smaller volume than the original excitation laser. [52]

NSOM [46], sometimes known as scanning near-field optical microscopy (SNOM), is a type of scanning probe microscope which exploits the nature of evanescent waves. The excitation laser is focused through an aperture that has a diameter smaller than the wavelength. This causes an evanescent field to be created on the other side of the aperture.

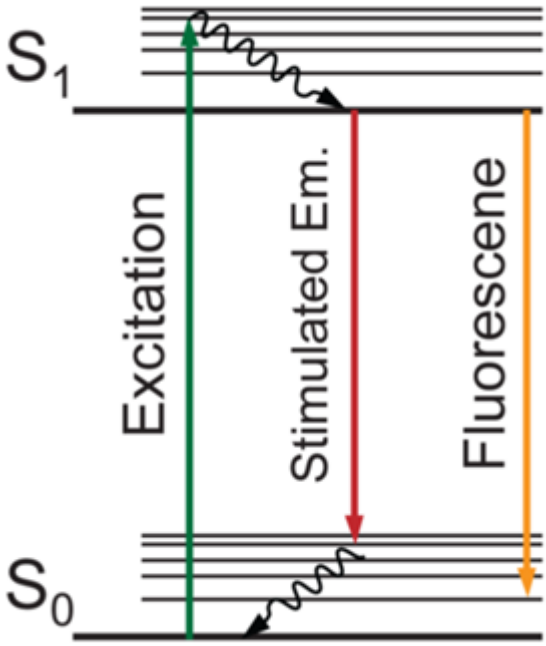


Figure 1.10: A simplified Jablonski diagram of both stimulated emission and fluorescence. The excitation laser excites a fluorophore from the ground state (S_0) into the first excited state (S_1). If this falls back down to the ground state spontaneously then this is fluorescence. If, however, the fluorophore encounters a photon with an energy comparable to the energy difference between the ground and excited state then stimulated emission occurs. [52]

The sample is scanned very close to the aperture and the detector is also placed a few nm from the surface so that everything is within the evanescent field. As an evanescent field is very localised to the aperture (it drops off proportionally to the square distance), this technique is primarily for surface inspection. It also takes a long time for large areas to be imaged and is also vulnerable to artefacts which typically comes from the tip breaking [46].

SIM uses patterned light to obtain high order spatial resolution, increasing the lateral and axial resolution two-fold compared to conventional light microscopy [47]. The first results were published in 1995. When objects containing fine structures are illuminated with this patterned, or structured, light, Moiré fringes arise. These shift the high frequency features to lower frequencies that can then be detected by the camera. The illumination pattern is shifted relative to the sample to

produce images at different phases as shown in figure 1.11. Computer software is then used to analyse the images and reconstruct an image by using superimposed additional information that is found in reciprocal space.

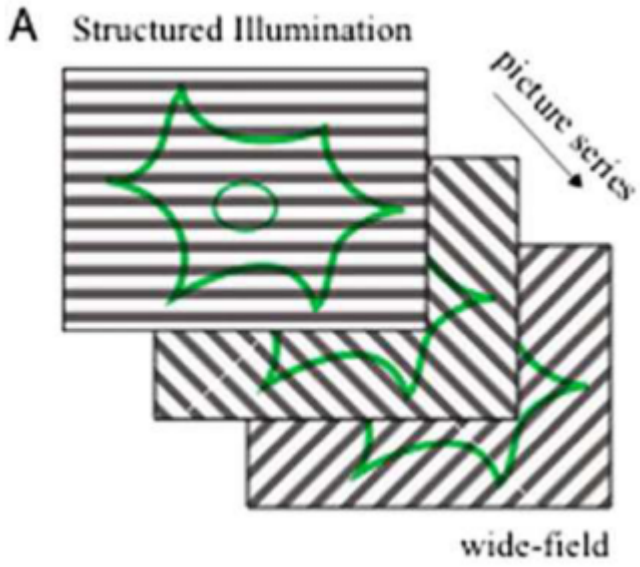


Figure 1.11: Simplified diagram of the illuminated samples in Structured Illumination Microscopy. Patterned light is used to illuminate the sample in widefield. The phase between the sample and illumination pattern is then changed and another image is taken. The images for the differently oriented images are then used to create a image. [32]

Although in general the sample needs to be stable in order to take the multiple images needed with light at different angles, this technique has been used with live cell measurements. This technique can also be enhanced by using saturated light: saturated structured illumination microscopy (SSIM) [54], to enable a resolution of up to 50nm [55].

1.7.1 *Single Molecule Localisation Microscopy (SMLM)*

SMLM summarises all stochastic super-resolution techniques that isolate emitters and fit a PSF. In 1995, Betzig proposed an idea to create super resolved images using blinking molecules [56]. The first observation of single molecules was W.E. Moerner in 1989 [57]. He observed them in solids at very low temperatures, however the field has advanced to low [58] and room temperatures [59]. It was later found that when single molecules of mutants of the green fluorescent protein

(GFP) were irradiated with 488nm light, they were seen to blink for several cycles before entering a stable dark state [60]. This detection of single molecules and then blinking single molecules was the pivotal achievement that led to SMLM and earned Moerner a Nobel prize in Chemistry in 2014 along with Eric Betzig and Stefan Hell (mentioned previously in section 1.7) [20].

After discovering that fluorophores were capable of photoswitching and photoblinking, three different research groups developed similar methods to exploit these properties in 2006. These were photo-activated localisation microscopy (PALM) [48], stochastic optical reconstruction microscopy (STORM) [49] and fluorescence photo-activated localisation microscopy (FPALM) [50].

These methods allow a few sparsely distributed fluorophores to be stochastically photoactivated while their remaining neighbouring molecules remain dark. This allows spatially inseparable fluorophores, when viewed together, to be resolved in time. These sparse fluorophores are imaged, localised and bleached/ turned off and then the process is repeated with a different, sparsely distributed, group of fluorophores. Merging all the obtained single-molecule localisations yields the final pointillistic super-resolution image. This process can be seen in figure 1.12

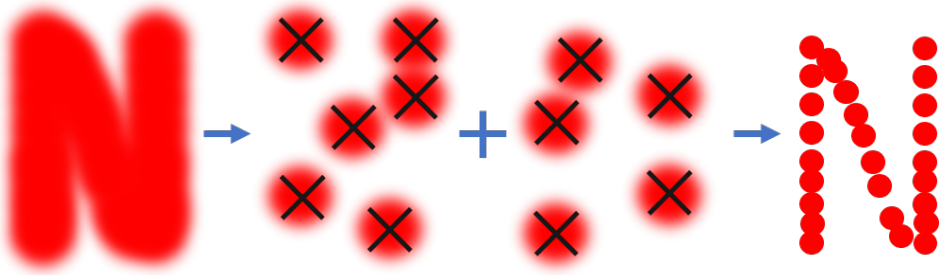


Figure 1.12: A simplified example of how SMLM works. The image on the far left shows a widefield image where all the fluorophores are emitting at the same time and the PSFs are overlapping. The middle two show isolated subsets of fluorophores emitting in multiple stochastic rounds of activation and localisation. As the PSFs are isolated, the fluorophores can be localised with a high precision by finding the centre. The final image on the right shows a computer-rendered super-resolution image creating using all the locations found in the images taken.

Fluorophores with complex temporal properties allow them to have two distinct states: 'on', and 'off' or 'dark'. These particular fluorophores are used in these methods. Although the complex process is not

completely understood, the most accepted theory is as follows, and the transitions are shown in the simplified Jablonski diagram figure 1.13. The fluorophore is 'on' when it is transitioning between its ground state, S_0, its first excited state, S_1 and its first triple state T_1. It is 'off' when it is in its charge transfer state (CTS) and it can also become bleached and 'die', meaning that it can no longer transition back to its 'on' state and release any more photons. Although this means there are three different states, there is no way to differentiate between 'off' and bleached/dead in the images produced and when they are not fluorescing ('on') it is not known where where fluorophores are located. [61, 62]

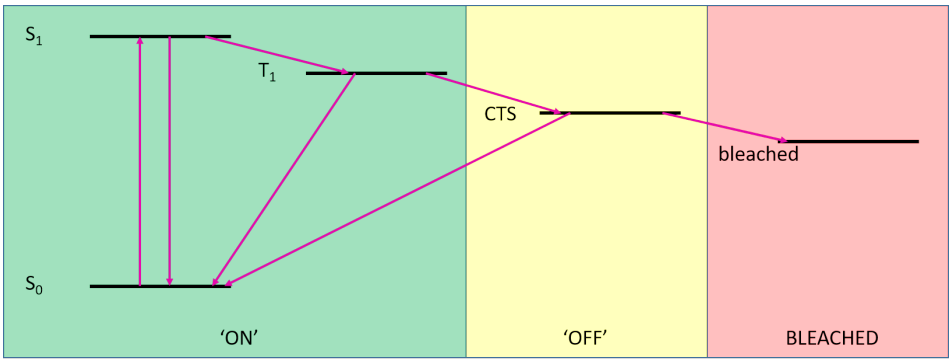


Figure 1.13: A simplified Jablonski diagram showing the possible transitions a fluorophore can undergo during a SMLM experiment.

It should be noted that the transition from T_1 to S_0 is non-radiative and does not emit a photon. This is a dark transition and is sometimes thought of as 'off'. However, as the lifetime in this state is very small (~100's nm), it can be considered as still being 'on' during this time.

A fluorophore is naturally in S_0 and is excited by the incident laser to S_1 whose lifetime is generally in the region of nanoseconds long. When it drops back down to S_0 a photon is emitted which is detected by the camera (or photon collector) and therefore it can be 'seen'. If the fluorophore transitions non-radiatively to its CTS instead, it will enter its 'off' state which has a lifetime in the range of ns-µs. From this state the fluorophore can either drop back to its ground state and turn back 'on', or it can bleach. Bleaching is often caused by the fluorophore oxidising. If the fluorophore enters this state then it can no longer enter its 'on' state and fluoresce. Although this is a third state, it is not distinct. In the images we cannot tell the difference between the fluorophore being 'off' or 'bleached'. [61, 62]

When a photon (of the correct energy) from the incident laser strikes a fluorophore that is in its ground state, it transitions to an excited

state such as S_1. When a fluorophore (or any molecule) is in an excited state it can release energy through several different means. These can be split into two different groups: intramolecular and intermolecular. For fluorescence the process is intramolecular and the fluorophore drops back down to S_0 emitting a photon which can be detected. If the fluorophore only transitioned between S_0 and S_1 then the power output from the fluorophore would remain roughly constant as the lifetime of S_1 is very short (nanoseconds). However, due to other means that a fluorophore can de-excite, the power output has a very large range. All the methods that reduce the photon yield are known as quenching.

The excited fluorophore can enter into a triplet state, T_1, non-radiatively, without emitting a photon, and stay there for a short period of time (ns to µs) before dropping down to S_0. The electron flips its spin state which is a quantum-mechanically forbidden transition and therefore has a very small probability of occurring. This is known as intersystem crossing and is an intramolecular de-excitation.

The fluorophore can also de-excite through non-radiative internal conversion, conformational changes, or intermolecular changes [38]. The conformational changes could come from collisions with, for example, oxygen (oxidation), halogens and anions. These are known as collisional quenching. Some intermolecular transfers can include photo induced electron transfer (PET) and forster resonance energy transfer (FRET) [63], both of which can be exploited.

The light liberated from individual fluorophores is observed in the images as an intensity spot, which is blurred due to diffraction, and is generally around 250-300nm laterally and 500-800nm axially [32]. Although the details of this spot cannot be resolved, the knowledge that only one fluorophore is present is utilised, and a location can be determined using the xy intensity centre of the spot.

To determine the localisations of these emitters, a variety of different computational algorithms can be used. Several of these are described in section 2.2 but a directory of these can be found here [64]. There are many factors that contribute to the localisation precision including; the number of detected photons per switching event, the pixel size of the camera and the magnification of the microscope, the background noise, and the labelling density [32]. It has been found that the precision is inversely proportional to the square root of the number of photons detected [65]:

$$< (\Delta x)^2 >= \frac{s^2}{\sqrt{N}}, \tag{1.13}$$

where N is the number of photons detected and s is the spot size.

If the detector has very large pixels then the precision will be limited by this and it will not be possible to determine where in this pixel that the fluorophore is situated. The Nyquist criterion also states that the average distance between two neighbouring molecules is roughly half the resolution that can be produced [65].

SMLM is favoured as it does not require any specialised equipment or a complicated set up as is needed in both STED or NSOM, and unlike EM live cells can be imaged using the technique. It can also achieve a higher resolution than SIM.

Unfortunately, this technique can be slow to acquire the raw images. The first PALM image took several hours [50] and the first STORM image taking 5 minutes to take [49]. Also some of the algorithms that are used to reconstruct the final image can take a long time to compute the final image. Despite this, the imaging times of just seconds have been reported [66] with acquisition times constantly reducing with new developments in the field.

1.7.2 *Super Resolution Optical Fluctuation Microscopy (SOFI)*

SOFI is similar to SMLM, however it is more of a post processing method. [51] This method requires fluorophores with two distinct states, either like in SMLM, with 'on' and 'off' states or with two different intensities. It uses the temporal correlations of the independently fluctuating emitters to create a 5-fold improvement in spatial resolution with a large background reduction.

1.8 OUTLINE AND SCOPE OF THIS THESIS

The aim of this thesis was to denoise images from SMLM and use this output to determine the number of fluorophores emitting in different regions of the images. Simulations were produced to enable precise analysis of post processes. They also enabled images with different qualities to be produced, for example images where all fluorophores are overlapping, or where they are very sparse. A two staged denoising algorithm was written that uses Bayesian Statistics and Markov random fields to determine which pixels are signal and which are noise. The first stage has all the pixels independent, whereas the second stage uses this output and a spatial dependence to further classify the pixels. Various different methods were used to try and determine the number of fluorophores in the images including a AI classifier.

Chapter 2 gives an overview of the previous methods used. It also introduces the different statistics and AI used.

Chapter 3 describes how the simulations were produced and explains the physics behind how SMLM works. Appendix B contains the code used in this chapter.

Chapter 4 explains how the denoising algorithm works and how the two different sections of the code can be altered and used. Appendix C contains all the code used in this chapter.

Chapter 5 describes the methods used to try and count the number of fluorophores and therefore molecules of interest in the sample. Appendix D contains all the code used in this chapter.

Chapter 6 contains a discussion and conclusions about all the previous chapters.

ACRONYMS

ADC analog to digital converter
AF Alexa Fluor
AI artificial intelligence
sCMOS scientific Complementary Metal-Oxide Semiconductor
CNN convolutional neural network
CTS charge transfer state
DL deep learning
e-h electron-hole
EM electron microscopy
EMCCD Electronic Multiplying Charge-Coupled Devices
FISH fluorescence in situ hybridisation
FNN feedforward neural network
FP fixed pattern
FPALM fluorescence photo-activated localisation microscopy
FRET forster resonance energy transfer
FT fourier transform
GFP green fluorescent protein
GPU graphics processing unit
IFT inverse fourier transform
i.i.d. independent, identically distributed
KNN K-nearest neighbours
LM localisation microscopy
LS least-squares
LSFM Light Sheet Fluorescence Microscopy
LSM Light Sheet Microscopy
MCMC Markov Chain Monte Carlo
MH Metropolis-Hastings
MIM Modified Ising Model
ML Machine Learning
MLE maximum likelihood estimate
MNIST modified national institute of standards and technology
MRF markov random field

NA numerical aperture
NN neural network
NSOM near-field scanning optical microscopy
PALM photo-activated localisation microscopy
PDF Probability Density Function
PET photo induced electron transfer
PSF point spread function
QE quantum efficiency
QW quantum well
QY quantum yield
RL Richardson-Lucy
RMS root mean squared
RNN recurrent neural network
ROI regions of interest
SIM structured illumination microscopy
SMLM Single Molecule Localisation Microscopy
SNOM scanning near-field optical microscopy
SNR signal-to-noise ratio
SOFI Super Resolution Optical Fluctuation Microscopy
SSIM saturated structured illumination microscopy
STED stimulated emission depletion
STORM stochastic optical reconstruction microscopy
SVM support vector machines
TIRF total internal reflection fluorescence
VWCM virtual window center of mass

REFERENCES

[1] C. SINGER, 'The Dawn of Microscopical Discovery', Journal of the Royal Microscopical Society **35**, 317–340 (1915).

[2] S. Bradbury, *An Introduction to the Optical Microscope* (Oxford University Press, 1989), p. 5.

[3] G. Sines and Y. A. Sakellarakis, 'Lenses in Antiquity', American Journal of Archaeology **91**, 191 (1987).

[4] S. Bradbury, *The Evolution of the Microscope* (Pergamon Press, 1967).

[5] H. Gest, 'The discovery of microorganisms by Robert Hooke and Antoni van Leeuwenhoek, Fellows of The Royal Society', Notes and Records of the Royal Society **58**, 187–201 (2004).

[6] C. S. Ball, 'The early history of the compound microscope', Bios **37**, 51–60 (1966).

[7] D. Bardell, 'The First Record of Microscopic Observations', BioScience **33**, 36–38 (1983).

[8] R. Hooke, *Micrographia* (Royal Society, 1665).

[9] K. B. Wolf and G. Krötzsch, 'Geometry and dynamics in refracting systems', European Journal of Physics **16**, 14–20 (1995).

[10] M. Hausner, *Optics Inspections and Tests: A Guide for Optics Inspectors and Designers* (2017), pp. 7–32.

[11] A. Kwan, J. Dudley and E. Lantz, 'Who really discovered snell's law?', Physics World **15**, 64 (2002).

[12] R. D. Fiete, 'Optics', in *Modeling the imaging chain of digital cameras* (SPIE Press, 2010), pp. 49–72.

[13] L. Davis, C, *Thin Lenses*, http://www.physics.louisville.edu/cldavis/phys299/notes/lo_lense Accessed: 2020-14-04.

[14] S. J. Ling, J. Sanny and B. Moebs, *4.1: single-slit diffraction*, https://phys.libretexts.org/?title=T _Optics_and_Modern_Physics_(OpenStax)/4:_Diffraction/4.1:_Single-Slit_Diffraction%0A, Visited on 2018-05-23.

[15] G. B. Airy, 'On the Diffraction of an Object-glass with Circular Aperture', Transactions of the Cambridge Philosophical Society **5**, 283–291 (1835).

[16] *Greenfluorescentblog*, `https://greenfluorescentblog.wordpress.com/2012/04/01/numerical-aperture-and-resolution/`, Accessed: 2018-01-29.

[17]E. Abbe, 'Beitrage zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung', Arch. Mikroscop Anat **9**, 413–420 (1873).

[18]T. S. Tkaczyk, 'Numerical Aperture', in *Field guide to microscopy* (2010), p. 38.

[19]B. R. Masters, 'Optical Resolution and Resolving Power : What It Is , How to Measure It , and What Limits It', in *Confocal microscopy and multiphoton excitation microscopy: the genesis of live cell imaging* (SPIE Press, 2006), pp. 49–54.

[20]N. Foundation, *The nobel prize in chemistry*, 2014.

[21]S. Balaiya, R. K. Murthy, V. S. Brar and K. V. Chalam, 'Evaluation of ultraviolet light toxicity on cultured retinal pigment epithelial and retinal ganglion cells', Clinical Ophthalmology, 33–39 (2010).

[22]M. Abramowitz and M. W. Davidson, *Anatomy of a Microscope - Numerical Aperture and Resolution*, Visited on 2020-04-16.

[23]N. Foundation, *The nobel prize in physics*, 1929.

[24]W. J. Croft, *Under the microscope [electronic resource] : a brief history of microscopy* (World Scientific Publishing Co. Pte. Ltd., 2006), pp. 57–72.

[25]R. Erni, M. D. Rossell, C. Kisielowski and U. Dahmen, 'Atomic Resolution Imaging with a sub-50 pm Electron Probe', Physical Review Letters **102**, 96–101 (2009).

[26]N. Foundation, *The nobel prize in physics*, 1986.

[27]J. Ayache, L. Beaunier, J. Boumendil, G. Ehret and D. Laub, 'Artifacts in Transmission Electron Microscopy', in *Sample preparation handbook for transmission electron microscopy* (Springer, New York, NY, 2010), pp. 125–170.

[28]J. G. Mcnally, T. Karpova, J. Cooper and J. A. Conchello, 'Three-Dimensional Imaging by Deconvolution Microscopy', Methods **19**, 373–385 (1999).

[29]G. Sparacino, G. De Nicolao, G. Pillonetto and C. Cobelli, 'Deconvolution', in *Modelling methodology for physiology and medicine* (Elsevier, 2014), pp. 45–68.

[30]M. Minsky, 'Memoir on inventing the confocal scanning microscope', Scanning **10**, 128–138 (1988).

[31]M. Minsky, *Microscopy Apparatus*, 1961.

[32]M. Renz, 'Fluorescence microscopy-A historical and technical perspective', Cytometry Part A **83**, 767–779 (2013).

[33]H. Helmholtz, 'Die theoretischen Grenzen für die Leistungsfähigkeit der Mikroskope.', Annalen der Physik, 557–584 (1874).

[34]M. Muyskens and Ed Vitz, 'The Fluorescence of Lignum nephriticum: A Flash Back to the Past and a Simple Demonstration of Natural Substance Fluorescence', Journal of Chemical Education **83**, 765 (2006).

[35]G. G. Stokes, 'On the Change of Refrangibility of Light', Philosophical Transactions of the Royal Society of London **142**, 463–562 (1852).

[36]G. G. Stokes, 'On the Change of Refrangibility of Light. No. II', Philosophical Transactions of the Royal Society of London **143**, 385–396 (1853).

[37]L. J. Kricka and P. Fortina, 'Analytical ancestry: "firsts" in fluorescent labeling of nucleosides, nucleotides, and nucleic acids', Clinical Chemistry **55**, 670–683 (2009).

[38]D. J. S. Birch, Y. Chen and O. J. Rolinski, 'Fluorescence', in *Photonics, volume 4 : biomedical photonics, spectroscopy, and microscopy* (John Wiley & Sons, Incorporated, 2015), pp. 1–58.

[39]G. T. Dempsey, J. C. Vaughan, K. H. Chen, M. Bates and X. Zhuang, 'Evaluation of fluorophores for optimal performance in localization-based super-resolution imaging', Nature Methods **8**, 1027–1036 (2011).

[40]R. Smallman and A. Ngan, 'Characterization and Analysis', in *Modern physical metallurgy* (Elsevier, 2014), pp. 159–250.

[41]H. Siedentopf and R. Zsigmondy, 'Uber Sichtbarmachung und Größenbestimmung ultramikoskopischer Teilchen, mit besonderer Anwendung auf Goldrubingläser', Annalen der Physik **315**, 1–39 (1902).

[42]D. Axelrod, 'Cell-substrate contacts illuminated by total internal reflection fluorescence.', The Journal of Cell Biology **89**, 141–145 (1981).

[43]Y. Markaki and H. Harz, eds., *Light Microscopy*, Vol. 1563, Methods in Molecular Biology (Springer New York, New York, NY, 2017).

[44]B. O. Leung and K. C. Chou, 'Review of super-resolution fluorescence microscopy for biology.', Applied spectroscopy **65**, 967–80 (2011).

[45]S. W. Hell and J. Wichmann, 'Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy', Optics Letters **19**, 780 (1994).

[46]R. C. Dunn, 'Near-field scanning optical microscopy.', Chemical reviews **99**, 2891–928 (1999).

[47]M. G. L. Gustafsson, 'Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution', Proceedings of the National Academy of Sciences **102**, 13081–13086 (2005).

[48]E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz and H. F. Hess, 'Imaging intracellular fluorescent proteins at nanometer resolution.', Science (New York, N.Y.) **313**, 1642–5 (2006).

[49]M. Rust, M. Bates and X. Zhuang, 'Stochastic optical reconstruction microscopy (STORM) provides sub-diffraction-limit image resolution', Nature methods **3**, 793–795 (2006).

[50]S. T. Hess, T. P. K. Girirajan and M. D. Mason, 'Ultra-high resolution imaging by fluorescence photoactivation localization microscopy.', Biophysical journal **91**, 4258–72 (2006).

[51]T. Dertinger, R. Colyer, G. Iyer, S. Weiss and J. Enderlein, 'Fast, background-free, 3D super-resolution optical fluctuation imaging (SOFI).', Proceedings of the National Academy of Sciences of the United States of America **106**, 22287–22292 (2009).

[52]B. Huang, M. Bates and X. Zhuang, 'Super-Resolution Fluorescence Microscopy', Annual Review of Biochemistry **78**, 993–1016 (2009).

[53]D. Wildanger et al., 'Solid immersion facilitates fluorescence microscopy with nanometer resolution and sub-Ångström emitter localization', Advanced Materials **24**, 309–313 (2012).

[54]R. Heintzmann, T. M. Jovin and C. Cremer, 'Saturated patterned excitation microscopy—a concept for optical resolution improvement', Optical Society of America **19**, 1599–1609 (2002).

[55]E. H. Rego, L. Shao, J. J. Macklin, L. Winoto, G. A. Johansson, N. Kamps-Hughes, M. W. Davidson and M. G. Gustafsson, 'Nonlinear structured-illumination microscopy with a photoswitchable protein reveals cellular structures at 50-nm resolution', Proceedings of the National Academy of Sciences of the United States of America **109**, 13–15 (2012).

[56]E. Betzig, 'Proposed method for molecular optical imaging', Optics Letters **20**, 237 (1995).

[57]W. Moerner and L. Kador, 'Optical detection and spectroscopy of single molecules in a solid', Physical Review Letters **62**, 2535–2538 (1989).

[58]M. Orrit and J. Bernard, 'Single pentacene molecules detected by fluorescence excitation in a p-terphenyl crystal', Physical Review Letters **65**, 2716–2719 (1990).

[59]W. P. Ambrose and W. E. Moerner, 'Fluorescence spectroscopy and spectral diffusion of single impurity molecules in a crystal', Nature **349**, 225–227 (1991).

[60]R. M. Dickson, A. B. Cubitt, R. Y. Tsien and W. E. Moerner, 'On / off blinking and switching behaviour of single molecules of green fluorescent protein', **388**, 355–358 (1997).

[61]M. Heilemann, S. van de Linde, A. Mukherjee and M. Sauer, 'Super-resolution imaging with small organic fluorophores.', Angewandte Chemie (International ed. in English) **48**, 6903–8 (2009).

[62]M. Heilemann, S. van de Linde, M. Schüttpelz, R. Kasper, B. Seefeldt, A. Mukherjee, P. Tinnefeld and M. Sauer, 'Subdiffraction-resolution fluorescence imaging with conventional fluorescent probes.', Angewandte Chemie (International ed. in English) **47**, 6172–6 (2008).

[63]L. Stryer and R. P. Haugland, 'Energy transfer: a spectroscopic ruler.', Proceedings of the National Academy of Sciences **58**, 719–726 (1967).

[64]*Single-molecule localization microscopy: directory of smlm software*, http://bigwww.epfl.ch/smlm, Visited on 2020-06-19, 2017.

[65]P. D. Simonson, E. Rothenberg and P. R. Selvin, 'Single-molecule-based super-resolution images in the presence of multiple fluorophores.', Nano letters **11**, 5090–6 (2011).

[66]S. A. Jones, S.-H. Shim, J. He and X. Zhuang, 'Fast, three-dimensional super-resolution imaging of live cells', Nature Methods **8**, 499–505 (2011).

[67]A. Small and S. Stahlheber, 'Fluorophore localization algorithms for super-resolution microscopy.', Nature methods **11**, 267–79 (2014).

[68]A. V. Abraham, S. Ram, J. Chao, E. S. Ward and R. J. Ober, 'Quantitative study of single molecule location estimation techniques', Optics Express **17**, 23352 (2009).

[69]R. E. Thompson, D. R. Larson and W. W. Webb, 'Precise nanometer localization analysis for individual fluorescent probes.', Biophysical journal **82**, 2775–2783 (2002).

[70]S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice Hall, 1993).

[71]R. Henriques, M. Lelek, E. F. Fornasiero, F. Valtorta, C. Zimmer and M. M. Mhlanga, 'QuickPALM: 3D real-time photoactivation nanoscopy image processing in ImageJ.', Nature methods **7**, 339–40 (2010).

[72]R. Parthasarathy, 'Rapid, accurate particle tracking by calculation of radial symmetry centers', Nature Methods **9**, 724–726 (2012).

[73]A. J. Berglund, M. D. McMahon, J. J. McClelland and J. A. Liddle, 'Fast, bias-free algorithm for tracking single particles with variable size and shape', Optics Express **16**, 14064 (2008).

[74]H. Ma, F. Long, S. Zeng and Z.-L. Huang, 'Fast and precise algorithm based on maximum radial symmetry for single molecule localization', Optics Letters **37**, 2481 (2012).

[75]T. Quan, H. Zhu, X. Liu, Y. Liu, J. Ding, S. Zeng and Z.-L. Huang, 'High-density localization of active molecules using Structured Sparse Model and Bayesian Information Criterion', Optics Express **19**, 16963 (2011).

[76]Y. Wang, T. Quan, S. Zeng and Z.-L. Huang, 'PALMER: a method capable of parallel localization of multiple emitters for high-density localization microscopy', Optics Express **20**, 16039 (2012).

[77]F. Huang, S. L. Schwartz, J. M. Byars and K. a. Lidke, 'Simultaneous multiple-emitter fitting for single molecule super-resolution imaging.', Biomedical optics express **2**, 1377–1393 (2011).

[78]F. Huang et al., 'Video-rate nanoscopy using sCMOS camera-specific single-molecule localization algorithms.', Nature methods **10**, 653–8 (2013).

[79]S. J. Holden, S. Uphoff and A. N. Kapanidis, 'DAOSTORM: an algorithm for high-density super-resolution microscopy', Nature methods **8**, 279–280 (2011).

[80]D. R. Tobergte and S. Curtis, 'Daophot: a Computer Program for Crowded-Field Stellar Photometry (Psf Photometry)', Journal of Chemical Information and Modeling **53**, 1689–1699 (2013).

[81]L. B. Lucy, 'An iterative technique for the rectification of observed distributions', Astronomical Journal **79**, 745–754 (1974).

[82]W. Richardson, 'Bayesian-based iterative method of image restoration', Optical Society of America **62**, 55–59 (1972).

[83]L. Zhu, W. Zhang, D. Elnatan and B. Huang, 'Faster STORM using compressed sensing', Nature Methods **9**, 721–723 (2012).

[84]S. Cox, E. Rosten, J. Monypenny, T. Jovanovic-Talisman, D. T. Burnette, J. Lippincott-Schwartz, G. E. Jones and R. Heintzmann, 'Bayesian localization microscopy reveals nanoscale podosome dynamics.', Nature methods **9**, 195–200 (2012).

[85]S. Wolter, U. Endesfelder, S. van de Linde, M. Heilemann and M. Sauer, 'Measuring localization performance of super-resolution algorithms on very active samples.', Optics express **19**, 7020–33 (2011).

[86]S. Wang, J. R. Moffitt, G. T. Dempsey, X. S. Xie and X. Zhuang, 'Characterization and development of photoactivatable fluorescent proteins for single-molecule-based superresolution imaging', Proceedings of the National Academy of Sciences of the United States of America **111**, 8452–8457 (2014).

[87]N. Durisic, L. Laparra-Cuervo, Á. Sandoval-Álvarez, J. S. Borbely and M. Lakadamyali, 'Single-molecule evaluation of fluorescent protein photoactivation efficiency using an in vivo nanotemplate', Nature Methods **11**, 156–162 (2014).

[88]P. J. Green, 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', Biometrika **82**, 711 (1995).

[89]H. Jeffreys, *Theory of probability*, 3rd (Press, Oxford: University, 1961).

[90]L. Savage, *The foundations of Statistics* (New York: Wiley, 1972).

[91]B. De Finetti, *Theory of probability* (London: Wiley, 1974).

[92]R. Christensen, T. E. Hanson, W. Johnson, Branscum and Adam, *Bayesian Ideas and Data Analysis : An Introduction for Scientists and Statisticians*, 1st ed. (Taylor & Francis Group, 2010).

[93]S. Chib and E. Greenberg, 'Understanding the Metropolis-Hastings Algorithm', The American Statistician **49**, 327–335 (1995).

[94]M. Jesper, ed., *Spactial Statistics and Computational Methods* (Springer, 2003), pp. 1–13.

[95]K. I. Mortensen, L. S. Churchman, J. A. Spudich and H. Flyvbjerg, 'Optimized localization analysis for single-molecule tracking and super-resolution microscopy', Nature Methods **7**, 377–381 (2010).

[96]J. R. Janesick, *Photon Transfer* (SPIE.Digital Library, 2007).

[97]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, `http://www.deeplearningbook.org` (MIT Press, 2016).

[98]S. Mor-Yosef, A. Samueloff, B. Modan, D. Navot and J. G. Schenker, 'Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study.', Obstetrics and gynecology **75**, 944–7 (1990).

[99]V. Metsis, I. Androutsopoulos and G. Paliouras, 'Spam Filtering with Naive Bayes – Which Naive Bayes?', In CEAS (2006).

[100]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, http://www.deeplearningbook.org (MIT Press, 2016).

[101]*Neural Network Programming - Deep Learning with PyTorch*, https://deeplizard.com/learn/ Accessed: 2020-07-08.

[102]D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', Nature **323**, 533–536 (1986).

[103]M. Torrisi, G. Pollastri and Q. Le, 'Deep learning methods in protein structure prediction', Computational and Structural Biotechnology Journal **18**, 1301–1310 (2020).

[104]*Single-molecule localization microscopy: collection of reference datasets*, http://bigwww.epfl.ch/smlm/datasets/index.html, Visited on 2020-06-19, 2017.

[105]H. Li and J. C. Vaughan, 'Switchable Fluorophores for Single-Molecule Localization Microscopy', Chemical Reviews **118**, 9412–9454 (2018).

[106]B. A. Griffin, S. R. Adams, R. Y. Tsien, B. A. Griffin, S. R. Adams and R. Y. Tsient, 'Specific Covalent Labeling of Recombinant Protein Molecules Inside Live Cells Published by : American Association for the Advancement of Science Stable URL : http://www.jstor.org/stable/2896025 REFERENCES Linked references are available on JSTOR for this', Science **281**, 269–272 (1998).

[107]L. W. Miller, Y. Cai, M. P. Sheetz and V. W. Cornish, 'In vivo protein labeling with trimethoprim conjugates: A flexible chemical tag', Nature Methods **2**, 255–257 (2005).

[108]A. Keppler, S. Gendreizig, T. Gronemeyer, H. Pick, H. Vogel and K. Johnsson, 'A general method for the covalent labeling of fusion proteins with small molecules in vivo', Nature Biotechnology **21**, 86–89 (2003).

[109]A. Gautier, A. Juillerat, C. Heinis, I. R. Corrêa, M. Kindermann, F. Beaufils and K. Johnsson, 'An Engineered Protein Tag for Multiprotein Labeling in Living Cells', Chemistry and Biology **15**, 128–136 (2008).

[110]G. V. Los et al., 'HaloTag: A novel protein labeling technology for cell imaging and protein analysis', ACS Chemical Biology **3**, 373–382 (2008).

[111]I. Y. Iourov, *Fluorescence In Situ Hybridization (FISH)*, edited by T. Liehr, Springer Protocols Handbooks (Springer Berlin Heidelberg, Berlin, Heidelberg, 2017), pp. 17–25.

[112]M. E. Brezinski, *Noise and system performance with td-oct and sd-oct* (Academic Press, Amsterdam ; 2006), pp. 175–195.

[113]T. Bushnell, *What is autofluorescence*, https://expert.cheekyscientist.com/what-is-autofluorescence/, Visited on 2020-06-09.

[114]B. Huang, W. Wang, M. Bates and X. Zhuang, 'Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy.', Science (New York, N.Y.) **319**, 810–3 (2008).

[115]M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych and G. M. Hagen, 'ThunderSTORM: A comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging', Bioinformatics **30**, 2389–2390 (2014).

[116]B. A. Cipra, 'An Introduction to the Ising Model', The American Mathematical Monthly **94**, 937–959 (1987).

[117]I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, 'Chapter 9 - Probabilistic methods', in *Data mining practical machine learning tools and techniques*, edited by I. H. Witten, E. Frank, M. A. Hall and C. J. B. T. D. M. (E. Pal (Morgan Kaufmann, 2017), pp. 335–416.

[118]B. A. Landman, I. Lyu, Y. Huo and A. J. Asman, 'Chapter 6 - Multiatlas segmentation', in *Handbook of medical image computing and computer assisted intervention*, edited by S. K. Zhou, D. Rueckert, G. B. T. H. o. M. I. C. Fichtinger and C. A. Intervention (Academic Press, 2020), pp. 137–164.

[119]Orchard, Peter, *Markov random field optimisation*, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0809/ORCHARD/, Accessed 2014-08-28.

[120]Q. Jackson and D. Landgrebe, 'Adaptive Bayesian contextual classification based on Markov random fields', Geoscience and Remote Sensing, ... **40**, 2454–2463 (2002).

[121]A. Sonnleitner, G. Schütz and T. Schmidt, 'Free Brownian Motion of Individual Lipid Molecules in Biomembranes', Biophysical Journal **77**, 2638–2642 (1999).

[122]M. K. Cheezum, W. F. Walker and W. H. Guilford, 'Quantitative comparison of algorithms for tracking single fluorescent particles', Biophysical Journal **81**, 2378–2388 (2001).

[123]U. Kubitscheck, O. Kückmann, T. Kues and R. Peters, 'Imaging and Tracking of Single GFP Molecules in Solution', Biophyiscal Journal **78**, 2170–2179 (2000).

[124]*Markov random fields*, https://www2.isye.gatech.edu/isyebayes/bank/handout16.pdf, Accessed August 11, 2022.

[125] *Classification Learner*, https://uk.mathworks.com/help/stats/classificationlearner-app.html, Accessed: 2020-08/-8.

[126] M. J. Rust, M. Bates and X. Zhuang, 'Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM).', Nature methods **3**, 793–5 (2006).

[127] R. P. Nieuwenhuizen, M. Bates, A. Szymborska, K. A. Lidke, B. Rieger and S. Stallinga, 'Quantitative localization microscopy: Effects of photophysics and labeling stoichiometry', PLoS ONE **10**, 1–18 (2015).

[128] D. Gross and W. Webb, 'Molecular counting of low-density lipoprotein particles as individuals and small clusters on cell surfaces', Biophysical Journal **49**, 901–911 (1986).

[129] B. M. Burton, K. A. Marquis, N. L. Sullivan, T. A. Rapoport and D. Z. Rudner, 'The ATPase SpoIIIE Transports DNA across Fused Septal Membranes during Sporulation in Bacillus subtilis', Cell **131**, 1301–1312 (2007).

[130] M. C. Leake, J. H. Chandler, G. H. Wadhams, F. Bai, R. M. Berry and J. P. Armitage, 'Stoichiometry and turnover in single, functioning membrane protein complexes', Nature **443**, 355–358 (2006).

[131] S. K. Das, M. Darshi, S. Cheley, M. I. Wallace and H. Bayley, 'Membrane protein stoichiometry determined from the step-wise photobleaching of dye-labelled subunits', ChemBioChem **8**, 994–999 (2007).

[132] K. Tsekouras, T. C. Custer, H. Jashnsaz, N. G. Walter and S. Pressé, 'A novel method to accurately locate and count large numbers of steps by photobleaching', Molecular Biology of the Cell **27**, edited by D. Lidke, 3601–3615 (2016).

[133] A. Lee, K. Tsekouras, C. Calderon, C. Bustamante and S. Pressé, 'Unraveling the Thousand Word Picture: An Introduction to Super-Resolution Data Analysis', Chemical Reviews **117**, 7276–7330 (2017).

[134] E. Alpaydin, 'Introduction', in *Introduction to machine learning* (2010), pp. 1–19.

[135] C. Campbell and Y. Ying, 'Learning with Support Vector Machines', Synthesis Lectures on Artificial Intelligence and Machine Learning **5**, 1–95 (2011).

[136] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (Cambridge University Press, Mar. 2000).

[137]V. Hodge and J. Austin, 'A Survey of Outlier Detection Methodologies', Artificial Intelligence Review **22**, 85–126 (2004).

[138]M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks', Neural Networks **106**, 249–259 (2018).

[139]L. Deng, 'The mnist database of handwritten digit images for machine learning research [best of the web]', IEEE Signal Processing Magazine **29**, 141–142 (2012).

[140]S. L. Smith, P.-j. Kindermans, C. Ying, Q. V. Le and G. Brain, 'D on ' T D Ecay the L Earning R Ate , I Ncrease the B Atch S Ize', 1–11 (2018).

[141]*Cross entropy loss*, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, Accessed April 02, 2021.

[142]Z. Zhang and M. R. Sabuncu, 'Generalized cross entropy loss for training deep neural networks with noisy labels', Advances in Neural Information Processing Systems **2018-December**, 8778–8788 (2018).

[143]*Cross-entropy loss function*, https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e, Accessed April 02, 2021.

2

METHODS AND MATERIALS

2.1 INTRODUCTION

Many different methods were used throughout this thesis. This Chapter gives an overview of both the traditional methods used and a brief introduction to the statistics and artificial intelligence artificial intelligence (AI) used in chapters 4 and 5 respectively.

2.2 RECONSTRUCTION ALGORITHMS

There are many different algorithms that can be used to localise fluorophores in a stack of images created using localisation microscopy (LM). A comprehensive list can be found here [64]. These algorithms use different methods to find and localise the fluorophores. The following will describe some of the methods used.

These algorithms are designed to localise the different fluorophores, $\boldsymbol{x_i} = (x_i, y_i)$, where $i = \{0, .., N\}$, N being the number of fluorophores, from an image, $\boldsymbol{I}$. $\boldsymbol{I}$ can be thought of as a matrix where each element represents the intensity for each pixel of the camera.

Localising the fluorophores can be done in two different ways; either by fitting the whole image instantaneously, or by finding areas where it is believed a single (or multiple) fluorophore is active and doing further analysis on these smaller areas. These areas are often referred to as regions of interest (ROI). Generally, the latter is used as fitting to the entire image is very difficult and usually computationally expensive.

2.2.1 *Finding regions of interest (ROI)*

Finding the ROI can be achieved using several different methods. A manually chosen threshold can be used to find areas which have intens-

ity values larger than this. This method is often dismissed as it does not achieve high enough identification levels due to the wide variation in intensity across different fluorophores.

Generally, deconvolution is used to enhance features of a characteristic shape; σ. This causes areas that have this shape to have sharper peaks. Using a threshold on this enhanced image finds ROI with a much higher precision. This precision can be found by using simulations and finding a user defined false positive p-value. [67]

Often, once the ROI have been found, they will be checked to ensure that they only contain one fluorophore. This is often done by a shape test and rejecting shapes that do not follow a point spread function (PSF), for example if they are too elliptical. Rejecting ROI that have too high a photon count (intensity) may also be used as this could indicate that there are two fluorophores emitting in a close region. [67]

2.2.2 *Localising using fitting*

Fitting images to a PSF can be a very rigorous approach to localising fluorophores. In this case, to allow the parameters of each fluorophore to be estimated, a physical model describing the mean number of photons collected at each pixel, given that a fluorophore is at $\boldsymbol{x_o}$; $\lambda(\boldsymbol{x}; \boldsymbol{x_o})$ is required. The PSF of the system is often used for this. For more complicated systems more parameters about the fluorophore are included which can include brightness, orientation and even velocity [67, 68]. This updates the model to $\lambda(\boldsymbol{x}; \boldsymbol{\Lambda})$ where $\boldsymbol{\Lambda}$ encompasses all of the parameters.

2.2.3 *Least-squares (LS) criterion*

The LS fit varies parameters to minimise the following equation:

$$S = \sum_{\text{pixels}} \frac{(\text{data} - \text{model prediction})^2}{\text{expected variance of data}}. \tag{2.1}$$

Here, S is the weighted sum of squared errors. It aims to find the set of model parameters that produces the least difference between the model and the observed data. The expected variance of data, depicted in the denominator, is the variance of the signal on that pixel. LS requires no detailed knowledge of the camera's noise.

This model is robust against the misspecification of the shape of PSF, as long as it is symmetrical. However, it needs the width to be well defined or depicted as an unknown parameter. [67, 68]

Weighting can be understood as comparing actual errors to expected errors. When mismatching these values, the accuracy of this model is reduced. The PSF can be constructed from experimental data or from an approximate formula.

There is a fast available algorithm that calculates only the x and y positions and no other parameters. [69]

2.2.4 *Maximum likelihood estimate (MLE)*

One of the best methods would be to use MLE as theoretically it achieves the lowest mean-squared error. However it needs accurate characteristics of the imaging system. If these are incorrect, the localisation accuracy is severely diminished. [67]

MLE returns values for the model parameters that are most likely to produce the observed data [68]. It requires three different theorems which each state a key result [70]. The first one states that, for every estimated parameter, there is a theoretical limit to the variance of unbiased estimators. This limit is the maximum achievable precision. Theorem two states that when this is attainable by an unbiased estimator, MLE will achieve this limit. Finally, theorem three states that, for a large data set, the variance of an MLE estimate will approach this limit.

The model works by varying the parameters and calculating the likelihood of obtaining the observed signal. It keeps the values of the parameters that provide the largest likelihood. It can also calculate a prediction of the variance of the estimate. [67]

2.2.5 *Localising without fitting*

It can be advantageous to use algorithms that are less dependent on a model of the system. These include centroid localisation method [71] and finding the point of radial symmetry [72].

The centroid localisation method is also known as the centre of mass method. One well known algorithm that uses this method is called QuickPALM [71]. In this method, the background fluorescence needs to be removed. If this is not removed then the result becomes a weighted average between the geometric centre and the true centre. One

problem with this method is that the actual PSF should be symmetrical however as the camera is pixelated and has a discrete grid (rather than a continuous), the image is distorted and the centres of the camera image and the actual image may not coincide. There are advantages of this method as there are very few assumptions of the image shape however caution needs to be used when using this method alone.

This method can be improved by using virtual window center of mass (VWCM) [73] as it means that it is unaffected by background noise. It works by ensuring the fluorophore is in the centre of the ROI by trimming and reshaping it.

Another method is to find the point of radial symmetry [72, 74]. This method works by calculating the gradient at each pixel, or at each corner, to define a line. Using these lines, the centre of the radial symmetry is determined by finding the point which minimises the total distance for all these lines meeting. Sometimes a weighting factor is used. This can be the inverse of the distance of each line.

This method is very fast and very good at finding locations when the fluorophores are sparsely distributed and there is only one fluorophore in each ROI. However, reconstruction accuracy is degraded for datasets with densely populated fluorophores, with the method often identifying a single emitter where there should be multiple fluorophores.

2.2.6 *Localising multiple emitters simultaneously*

There are several different methods that have been created to try and fit multiple emitters simultaneously. Many of these extensions of the fitting methods were mentioned earlier and produce a list of positions however, some of them produce a density profile instead. As before, when using small sections of the image, there are both fitting and non-fitting based methods. [67]

Some issues that occur when fitting multiple fluorophores is that the model can favour more fluorophores than there actually are. Often a user-determined threshold is created for the residual or log likelihood and the algorithm will stop adding fluorophores when this threshold is reached.

Fitting based methods often work by cutting the image into ROI as before, and assuming that each of these regions are independent of each other. The theory is that if the PSF of the fluorophore is well known, then it is possible to create models that describe how they would behave if they overlap. These methods assume that there is

no interaction between close emitting fluorophores. They are also more computationally expensive as there are a lot more parameters to estimate.

The most rigorous approach is to calculate the MLE [75]. These have also been extended into using a graphics processing unit (GPU) which increases the speed of the algorithms [76, 77]. Huang *et al.* have created a method that is optimised for scientific Complementary Metal-Oxide Semiconductor (sCMOS) cameras. [78]

One popular LS approach is called DAOSTORM [79] which is based on an astronomy algorithm [80]. It uses a fixed shape, multiple model PSF to localise the fluorophores. It has been shown to maintain high performance in high density images.

2.2.7 *Image estimation*

These methods estimate the local density/ concentration of fluorophores. Two different approaches use either the Richardson-Lucy (RL) deconvolution [79, 81, 82] or compressed sensing [83].

Richardson-Lucy deconvolution uses an MLE approach and requires a density map, a PSF and a noise model. It computes the probability of obtaining the data set chosen and then picks the one with the highest probability. One advantage is that it can incorporate blinking dynamics into the model.

Compressed sensing requires no prior information on noise or blinking dynamics. It works by subdividing the pixels so that the number of unknown variables is greater than the number of input variables. Here, the unknown variables are the places on the density map and the input variables are the photon values at each pixel. There are then more equations than unknown variables and therefore this problem can be solved. For each data there will be multiple maps that would fit. Compressed sensing then chooses the sparsest of these maps so as not to overestimate the number of fluorophores. [83]

Another method uses Bayesian Statistics (described in section 2.3.1) to compute a conditional probability of a density map [84]. It is called 3B, short for Bayesian analysis of blinking and bleaching method. It uses the whole sequence of images and the blinking dynamics. The final image created is the average over many different density maps. This method is very computationally intensive and the effort is linear to the number of fluorophores multiplied by the number of pixels. It also trades off temporal resolution for spatial resolution. Each reappearance

of a fluorophore (as it blinks) improves the localisation accuracy and it can achieve a 50nm resolution after only a few seconds of acquisition. [84]

2.2.8 *Testing localisation algorithms*

To determine how good a localisation algorithm is, synthetic data is generally used. There are several possible errors in these algorithms. They can find the location of a fluorophore that is not there, a false positive, or they can miss the location of fluorophore, a false negative. They can also determine that a fluorophore is there, but localise it in the incorrect location; spatial precision. There are two fundamental values that are used to describe how good the algorithms are, the precision p and the recall r. These are described as follows:

$$p = \frac{TP}{FP + TP} \tag{2.2}$$

and

$$r = \frac{TP}{FN + TP} \tag{2.3}$$

where TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives. The precision is the percentage of locations found that are correct and the recall is the percentage of all the real fluorophores that are found.

These values are sometimes combined as either the Jaccard Index, JAC or the F_1-score;

$$JAC = \frac{TP}{FN + FP + TP} \tag{2.4}$$

and

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}. \tag{2.5}$$

Modern localisation algorithms typically have quite good precision but varied recall, however these values will very drastically depending on the density of the emitters per image [85], and generally there is a peak performance for different fluorophores.

Although these values give a good idea of how many fluorophores enter the 'on' state, a good image of what is occurring in the cell cannot be produced if the object of interest is poorly labeled, or if the

fluorophores never turn 'on'. In certain circumstances it is possible to label everything (by using genome editing systems), however it is not possible to ensure that every fluorophore in the sample photoconverts into its 'on' state. If the fluorophore never emits any photons, then it is fundamentally impossible for any algorithm to determine its location.

The algorithms, however, could have very good recall and precision, but fail to determine the location of the fluorophores accurately.

The algorithm could be perfect at counting all the fluorophores that emit any photons, but this will only show a good representation of the sample if the method is not limited by experimental difficulties.

Several different groups have looked into the percentage of fluorophores that mature (fluorescent proteins only) [86] and photoconvert into their 'on' state. These vary substantially from 1-20% [86] to 40-80% [87].

2.3 STATISTICS

Statistics are heavily used in this thesis to both create simulated images (chapter 3) and analyse them (chapter 4). In general, statistics are used across a very broad range of disciplines to extract and infer conclusions from noisy data with uncertainty. The field uses probabilities to draw inferences about unknown parameters, such as means and variances, or to predict future observations. To do this the uncertainty/variability of the data is formalised by treating the data as realisations of random variables whose joint distribution comprises the data generation mechanism, also known as the likelihood. As "true models" are only available in very limited situations, several different likelihoods can be chosen for each situation and this leads to shades of subjectivity on the model [88].

The Bayesian approach was pioneered by Reverend Thomas Bayes and Pierre-Simon Laplace in the 18th century and developed by Harold Jeffreys [89],Leonard Jimmie Savage [90] and Bruno de Finette [91] in the 1950's to the 1970's. This approach can be used for any uncertain event even if it cannot be repeated (e.g. the speed of light or if England will win the Women's hockey in the next Olympic games). Bayesian, or Bayes Theorem is expressed in the statement "The posterior is proportional to the likelihood times the prior". If follows Dennis Lindley's two rules [92]:

1. they always obey the laws of probability,
2. all uncertainty is to be modelled using probability.

Simply, the difference between a Frequentest and a Bayesian is that to a frequentist, data are repeatable and they will consider what values for the data are plausible conditional on a particular value of the parameters: P[data | parameters], whereas to a Bayesian the parameters are uncertain and the observed data are not. They consider their probability distribution of the parameters conditional on the observed data P[parameters | data].

Probability-based models are used in all the algorithms in this thesis. We refer to these by name and the distribution are shown in table A.1.

2.3.1 *Bayesian Inference*

As stated before, Bayes allows statistical inference to be performed on any data, whether it is replicable or not. Bayesian Inference states that the posterior is proportional to the likelihood multiplied by the prior. This statement can be written as:

$$\pi(\boldsymbol{\theta} \mid D) \propto L(\boldsymbol{\theta}; D)\pi(\boldsymbol{\theta}), \tag{2.6}$$

which is simplified from:

$$\pi(\boldsymbol{\theta} \mid D, A, K) = \frac{p(D \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid K)}{f(D)}. \tag{2.7}$$

In these equations (equations (2.6) and (2.7)), $\pi(\boldsymbol{\theta} \mid D)$ and $\pi(\boldsymbol{\theta} \mid D, A, K)$ are the posteriors, $L(\boldsymbol{\theta}; D)$ and $p(D \mid \boldsymbol{\theta})$ are the likelihoods and $\pi(\boldsymbol{\theta})$ and $\pi(\boldsymbol{\theta} \mid K)$ are the priors. $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ are the unknown parameters, D is the data, A and K are the assumptions and previous knowledge respectively which are omitted from equation (2.6). $f(D)$ is a normalising constant.

The prior, or prior (probability) distribution, is a mathematical construct which expresses the decision maker's beliefs about the unknown quantities. Every unknown parameter has its own separate prior which describes its shape. This can often be quite difficult to decide and is very subjective. There are many different choices for each prior, and different people tackling the problem will choose different priors which adds to the subjectivity of the model. These priors reflect all the relevant information about the parameter, not depending on the data at hand. If there is little to no knowledge about the parameters then uninformative or 'flat' priors can be used that reflect a balance among all possible values of the parameter. The prior can also be produced using data from previous experiments or it can be elicited subjectively by an expert. Often conjugate priors are used, which are distributions

from the same family as the posterior. This simplifies the posterior considerably.

The likelihood, or likelihood function, is the chosen model for the data generation mechanism. This function is used in both Bayesian and frequentest statistics and is a function of both the data and the unknown parameters. The numerical observations, or data, are taken as realisations of random variables and the joint distribution of these observations is taken to be the likelihood. This is the main assumption in any statistical approach and is a Probability Density Function (PDF) which depends on fixed unknown parameters ($\boldsymbol{\theta}$). Once the data has been observed it can be fixed and used to find the unknown parameters ($\boldsymbol{\theta}$).

The posterior can be thought of as the updated distribution of $\boldsymbol{\theta}$ using the prior information. It is a PDF of the unknown quantities in question. Each of these unknown quantities are treated as random distributions. It is conditional on the data that was provided in the likelihood and the prior distributions elicited. Although the posterior is dependent on both the prior and the likelihood, there are many different factors that sway it towards either one. The posterior is driven towards the prior distribution if: there is high confidence in it and it has been given a high precision, if there is little data, or if the likelihood has low precision. It will be driven more towards the likelihood if it has a high precision, there are lots of data, or if an uninformative prior is used.

The proportionality constant ensures that the posterior is a PDF by normalising the function:

$$f(D) = \int_{\Theta} p(D \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid K)d\boldsymbol{\theta}, \tag{2.8}$$

ensuring that the posterior integrates over all space to one. If this easily calculated then inference can begin straight away as the posterior is simple to find [92]. When this is difficult to calculate, which is often the case, more complicated methods must be used such as Markov Chain Monte Carlo (MCMC), described below in section 2.3.2.

2.3.2 *Markov Chain Monte Carlo (MCMC)*

When the normalising constant, $f(D)$, cannot be calculated then different methods such as MCMC or numerical integration have to be used to complete Bayesian Inference. [92] Numerical integration can be very difficult and complicated to implement, especially if using both

discrete and continuous quantities and high dimensions. It also is not very numerically stable and therefore MCMC is preferable.

A sequence $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \boldsymbol{\theta}^3, ..., \boldsymbol{\theta}^N$ is a Markov Chain if, for any set A, $\boldsymbol{\theta}^k \in \boldsymbol{\Theta} \subseteq \mathbb{R}^p$ and $\{k \in \mathbb{N} : k > 1\}$:

$$\Pr(\boldsymbol{\theta}^k \in A \mid \boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^{k-1}) = \Pr(\boldsymbol{\theta}^k \in A \mid \boldsymbol{\theta}^{k-1}), \tag{2.9}$$

so each state only relies on the previous state. This dependence is known as the Markov Property. The sequence must be (p-)irreducible, aperiodic and (Harris/positive) recurrent, meaning that it must be possible to get from any state to any other state in a number of steps. [92] It must also be non-cyclic and able to repeat values. These characteristics are exploited to build a scheme that will provide a means to explore the posterior distribution.

Monte Carlo is a traditional name for simulation methods which rely on random sampling to obtain approximate numerical results. So together an MCMC is just a Markov chain which uses random sampling.

The initial values: $\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^T$, where $T < N$, are known as the burn-in period and are discarded. Although the value of θ^1 is arbitrary and not entirely important, the closer it is to the stationary distribution then the smaller the burn-in period will be and therefore the cheaper (and faster) the process will be. The number of values to discard can be determined by looking at a plot of the values of one of the parameters. An example of this can be seen in section 2.3.3 in figure 2.1.

2.3.3 *Simple Example of Markov Chain Monte Carlo (MCMC)*

This section will describe a simple example of Bayesian Inference using MCMC. Imagine that we have sections from raw images in Single Molecule Localisation Microscopy (SMLM) in which we know there are only single events. From these data we want to calculate the mean intensity of an event. Although this can be found using a frequentist approach it would not be possible to incorporate any prior knowledge about the events in it and therefore a Bayesian approach will be used.

The data is modelled with a Gaussian distribution:

$$X_i \sim \mathcal{N}(x_i \mid \mu, \frac{1}{\lambda}), \tag{2.10}$$

with unknown mean μ, and precision λ. Each data point is said to be independent, identically distributed (i.i.d.) and therefore the likelihood is:

$$\pi(\boldsymbol{\theta}; D) = \prod_{i=1}^{N} \mathcal{N}(x_i \mid \mu, \frac{1}{\lambda}), \tag{2.11}$$

where N is the number of data points, $\boldsymbol{\theta}$ is a place-holder for the unknown parameters and D is the data.

A prior is put on both the unknown parameters μ and λ. A Gaussian distribution is chosen on the mean as it is known that the distribution is bell shaped and also because this can be made quite 'flat' and uninformative. A Gamma distribution is put on the precision because it has to be greater than zero, and again they can be quite uninformative. The two prior distributions are:

$$\pi(\mu) = \mathcal{N}(\mu \mid \eta, \frac{1}{\tau}) \tag{2.12}$$

and

$$\pi(\lambda) = \mathrm{Ga}(\lambda \mid a, b), \tag{2.13}$$

with known a and b; the shape and scale parameter of λ's Gamma distribution; but unknown η and τ; mean and precision of μ's Gaussian distribution. As there are unknown parameters in the priors, often called hyperparamters, this is known as a hierarchal model and a second layer of priors (one for each unknown in the first layer of priors), often called hyperpriors, need to be included:

$$\pi(\eta) = N(\eta \mid m, b) \tag{2.14}$$

and

$$\pi(\tau) = \mathrm{Ga}(\tau \mid c, d). \tag{2.15}$$

Where m and b are the mean and precision of η's Gaussian distribution, and c and d are the shape and scale parameters for τ's gamma distribution. In this example, all the prior distributions are kept quite flat and uninformative.

All the priors (equations (2.12) and (2.13)) and hyperpriors (equations (2.14) and (2.15)) are multiplied together with the likelihood (equation (2.11)) to compute the posterior (equations (2.6) and (2.7)). This is of unknown form so the probability constant cannot be calculated analytically and therefore inference cannot be made directly.

Instead an MCMC method is used. To do this, the full conditional of each unknown parameter (μ, λ, η and τ) is found using the posterior. This is done by setting all the other parameters, other than the one in question, as constants and simplifying it to find the known distribution. From these distributions values can be drawn. As the full conditionals are dependant on the other parameters and hyper-parameters, they need to be constantly updated. Values are randomly produced from each of the full conditionals sequentially and recorded, with the distributions being updated before each value is drawn. After a period of time the values will start to converge and once this occurs each drawn value will be a realisation from the posterior distribution. Up to this point is called the burn-in period and these initial values are discarded as they have not been drawn from posterior. The number of data points that are discarded is determined by plotting the data points as shown in figure 2.1. Using a plot like this it is easy to see when the chains start to converge. In this particular case the chains converge almost instantly. Although drawing this graph and using it to determine the burn-in values is very useful, it is not practical to use this method in practice when doing inference over many different images. Instead, a relatively large number of values are dropped to ensure that all of the burn-in period is discarded. Once a sufficient number of values have been collected, inference can be made on the collected data. Bayesian Inference is useful as it also gives an uncertainty on the predicted values. A pseudocode is shown in section 2.3.4 and the actual code is shown in appendix A.1.

How close the values in posterior are to the actual values is dependent on several different factors: how many iterations the algorithm goes through (up to the uncertainty of the model), how much data is available and how informative the priors are. Below is an example showing how the the posterior changes depending on how much data is available.

There are three data sets. Each one consists of random numbers taken from a Gaussian distribution with mean 10 and precision 1. The first data set has 10 data points, the second has 100 and the final has 1000 data points. Each of these data sets was put through the same algorithm with the same number of iterations, starting points and priors. The priors chosen in this example were taken to be quite flat and uninformative. Figure 2.2 shows the posteriors results for the three calculated means (μ's). It can be seen from this figure that the more data is available the more precise the value of μ is. The pseudocode for this process is in section 2.3.4. The code written in MATLAB (by myself) is in appendix A.

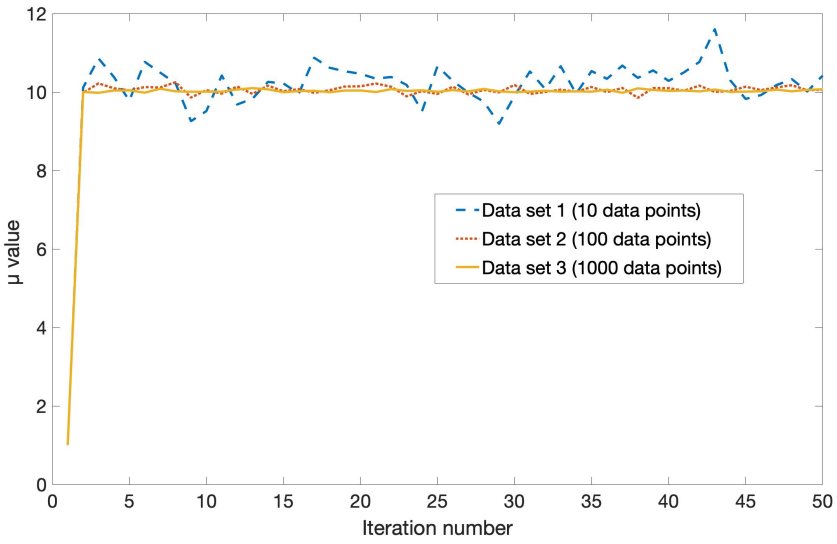


Figure 2.1: Burn-in for each data set. It can be seen that in these particular data sets there is a very small burn-in and the values converge almost instantly. It can be seen that the smallest dataset has the largest variability.

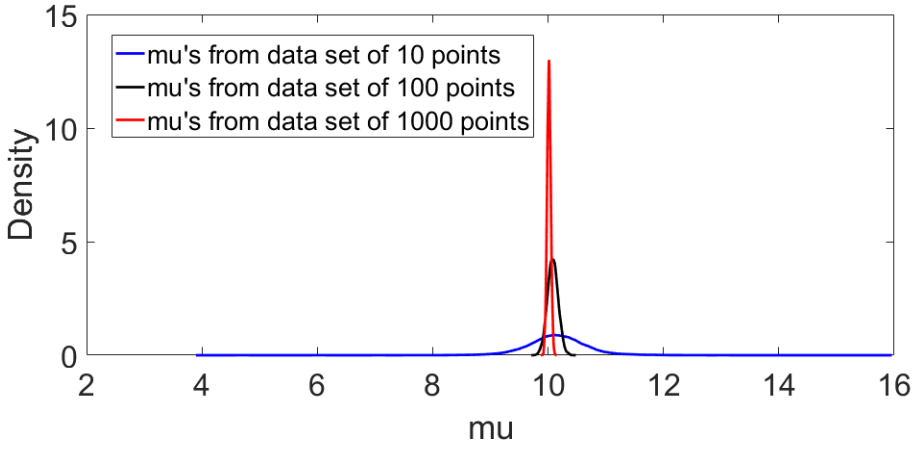


Figure 2.2: Graph shows the probability densities of the μ's calculated using Bayesian Inference for three different data sets. The data for each set come from the same distribution, however the size varies from 10 points to a 1000 points. It can be seen that the μ's calculated from the larger data set is more precise.

Table 2.1 below shows some numerical summaries taken from the Bayesian Inference of this problem. It can be seen from this table that

the more data points there are, the closer the values are to the exact value and the more precise the values are.

	Actual Value	Data Set 1	Data Set 2	Data Set 3
number of data points		10	100	1000
mean	10	10.2886	10.0375	10.0309
variance of mean		0.3106	0.0097	0.00096
precision	1	0.5392	1.0397	1.02357

Table 2.1: Table shows the results from the Bayesian Inference for three different data sets. The data are taken randomly from the same distribution of mean 10 and precision 1. It can be seen that the more data points in the set, the more accurate the final value of the mean and precision can be found.

If the full distributions are of an unknown form then random values cannot be drawn. Instead, values are drawn from a proposal distribution; $g(x'|x_t)$ where x' is the new value and x_t is the previous value. [93] In principle, this proposal can be any distribution, however in practice a distribution that is similar to the full distribution is chosen. These values are then kept with probability α, calculated from the Metropolis-Hastings (MH) ratio [94]:

$$\alpha = \min\left\{\frac{\pi(x')g(x_t|x')}{\pi(x_t)g(x'|x_t)}, 1\right\}. \tag{2.16}$$

This determines how compatible the new value is compared to the current value. If the new value, x', is rejected then the previous value, x_t, is used as the next value, x_{t+1}. The proposal distribution is usually chosen so that between 20% and 40% of values are kept. In a Gibbs sampler the proposal distribution is the same as the full distribution and therefore the MH ratio is always equal to 1 so every value is kept.

As stated earlier, several additional factors other than the amount of data available determine the accuracy of the results found from Bayesian Inference. The more iterations performed the higher the accuracy and precision of the results, up to the uncertainty of the model. A compromise has to be found between the number of iterations and computational power used. The priors chosen can affect the results considerably, and as stated before, as the priors are subjective they can vary substantially depending on the person who chooses these. If little

is known about the variable then a flat, or non-informative, prior can be used. This means that the results will tend more towards the data. If a high precision prior is used then the posterior will be pushed more to the prior.

The prior is more important if the sample is small, the data are complex or variable, or if the data are heterogeneous; coming from several different sources. In these cases, the prior chosen will have a much higher impact on the resulting posterior.

2.3.4 *Pseudocode*

This section describes the pseudocode for the process described in section 2.3.3. The likelihood is described in equation (2.11) and the priors for μ, λ, η and τ are shown in equations (2.12) to (2.15). The posterior is calculated as:

$$\pi(\boldsymbol{\theta} \mid D) = \prod_{i=1}^{N} \mathcal{N}(x_i \mid \mu, \frac{1}{\lambda}) \mathcal{N}(\mu \mid \eta, \frac{1}{\tau}) \mathrm{Ga}(\lambda \mid a, b) \mathrm{N}(\eta \mid m, p) \mathrm{Ga}(\tau \mid c, d). \tag{2.17}$$

From this the full conditionals can be determined as follows and are shown in equations (2.18), (2.21), (2.26) and (2.29)

$$\Pi(\mu) = \mathcal{N}(\mu \mid m^*, p^*) \tag{2.18}$$

with

$$m^* = \frac{n\lambda}{n\lambda + p} \bar{x} + \frac{p}{n\lambda + p} m \tag{2.19}$$

as a weighted mean and

$$p^* = n\lambda + p. \tag{2.20}$$

From this it can be seen that if the precision of the prior is very small, then m^* would be approximately equal to $\bar{x}$. Also, if n, the number of data points, is very large, then the same would also occur. This means that the prior will only have a large effect if it has a large precision, or only a small amount of data is collected.

The full conditional for λ is:

$$\Pi(\lambda) = \Gamma(\lambda \mid a^*, b^*) \tag{2.21}$$

where

$$a^* = \frac{n}{2} + a \tag{2.22}$$

and

$$b^* = b + \frac{n}{2}(s^2 + (\bar{x} - \mu)^2) \tag{2.23}$$

with s^2 being the sample variance:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{2.24}$$

and $\bar{x}$ is the average of x given by:

$$\bar{x} = \frac{1}{n}\sum_{i} x_i. \tag{2.25}$$

The full conditionals of η

$$\Pi(\eta) = \mathcal{N}(\eta \mid \alpha^*, \beta^*) \tag{2.26}$$

where

$$\alpha^* = \frac{\tau\mu + pm}{\tau + p} \tag{2.27}$$

and

$$\beta^* = \frac{1}{\tau + p} \tag{2.28}$$

The full conditional of τ

$$\Pi(\tau) = \Gamma(\tau \mid c^*, d^*) \tag{2.29}$$

where

$$c^* = c + \frac{1}{2} \tag{2.30}$$

and

$$d^* = d + \frac{(\mu - \eta)^2}{2} \tag{2.31}$$

The Pseudocode is as follows:

1. Calculate values for $\bar{x}$ and s^2 (equations (2.24) and (2.25))

2. Fix values for m, p, a and b
3. Calculate a^* (equation (2.22))
4. Arbitrarily choose a value for $\lambda > 0$
5. Use λ to calculate/update m^* and p^* (equations (2.19) and (2.20))
6. Generate a random number from the distribution of μ (equation (2.18)) using current values of m^* and p^*
7. Use current value of μ to calculate/update b^* (equation (2.23))
8. Generate a random number from the distribution of λ (equation (2.21)) using current value of b^*
9. Repeat steps 5-8 m times (m is the number of iterations)

2.4 PHOTON DETECTION

Before cameras, the only way to record images was to hand draw what was seen through a microscope. Although some very good images were produced in this manner, it is very impractical and quite impossible to complete at the speed the images change when using any wide-field super-resolution technique or any technique that uses raster scanning. Hand drawing has very low precision and accuracy and visually seeing the images from LM would be near impossible. Currently, either an Electronic Multiplying Charge-Coupled Devices (EMCCD) or a sCMOS camera is used when imaging in wide field. When imaging by raster scanning across a sample, often an optical fibre is used to collect the photons which are then transported to a single counting module which is similar to a one pixel camera.

EMCCD and sCMOS cameras both have a sensory array that detects and collects incident photons. Each pixel in the sensory array consists of a photodiode which is a semiconductor p-n junction. Electrons diffuse from the n- to the p-region leaving holes and inducing an electronic field which opposes this diffusion until an equilibrium is created.

This natural potential difference forms a quantum potential or quantum well (QW): a charge depletion region. The region is increased by putting a reverse bias upon the photodiode which also reduces any current being produced when there are no photons hitting the pixel. This is known as dark current and will be explained further later in this section.

When a photon with sufficient energy hits the pixel, an electron is excited from the valence band to the conduction band and is then described as a photoelectron. These photoelectrons then migrate to the quantum well, where they stay until they are counted. In one frame, each pixel will collect a number of photoelectrons proportional to the number of photons that are incident upon it.

Both the EMCCD and the sCMOS collect the photoelectrons in this way however they readout the values using different methods. In the EMCCD camera a sinusoidal voltage is used to shift the electrons from each pixel down into a holding area. Each pixel, row by row, then goes through a read-out register and an electron multiplication register which turns the electrons into an electrical signal which is measured.

In sCMOS cameras the photoelectrons are converted into a voltage at the pixel. Immediately after the image has been taken, each row of pixels is read out separately with its own charge amplifier which digitises the signal. As this does not have to be grown on one chip like

an EMCCD it is much cheaper to produce and due to there being an amplifier in each row the sCMOS camera can have much faster frame rates. However, having a different amplifier for each row means that there are slight differences to each read out and the values will not be as uniform. The quantum efficiencies will be slightly different for each amplifier and therefore there will be more variation over the image than the more uniform EMCCD camera.

Although, after the readout register the EMCCD is more uniform, the EMCCD's electron multiplication register increases the variation of the number of electrons counted. This multiplication factor is different each time and changes the values non-uniformly across the pixels, both spatially and through time. This multiplication factor essentially halves the quantum efficiency (QE) of an EMCCD camera from 95% to <48%. [78, 95]

Both cameras are susceptible to thermal noise. Some electrons acquire enough thermal energy to escape the semiconductors sea of electrons. These electrons move into the QW and are read out by the cameras amplifiers. This is proportional to the temperature and some cameras are cooled to minimise this. This noise is often called thermal dark current as it is produced even when there is no light incident upon the camera.

As the QW is not finite, some electrons will also be able to naturally tunnel out. This would reduce the amount of detected photons that would be in the final image.

Pixels in both cameras can saturate, as the QW can only hold so many electrons. If this occurs then there is a loss of information and the images produced are less accurate. Also, if the pixels saturate, sometimes a 'bleeding effect' may occur and the stored electrons can move into neighbouring pixels, further reducing the accuracy of the produced image. This affect is reduced by having smaller exposure times, i.e. increasing the frame rate.

A compromise has to be found between read noise and saturation. If the frame rate is very quick then read noise will have a high impact on the images produced. If it is too slow then the pixels could saturate.

Cameras have a lot of different sources of noises that they add to the images. These include, read noise, shot noise, fixed pattern (FP) noise and Fano noise [96]. These are described in more detail in section 3.4.

2.5 Artificial intelligence (AI)

AI refers to machines (or robots etc) that mimic human intelligence or the human mind. This could be in the form of problem solving; playing games; recognising objects, animals or people; or understanding and responding to a language. The machine may utilise one or more of these abilities. It is used regularly in day-to-day life; in phones such as 'Siri' and 'Google assistant', to recommend what to watch next on Netflix etc, and when asking for directions.

Early AI solved problems that were intellectually difficult for humans, that followed formal mathematical rules. These were easy for computers to resolve and comparatively simple to program. The true challenge of programming AI is solving problems that are intuitive and simple for humans, for example, recognising speech, faces or animals. Although these tasks are easy for humans to complete, they are difficult to describe, especially mathematically. To overcome this, the computer is programmed to learn from experience (data). It then learns in terms of a hierarchy of concepts, using simple concepts to learn more complicated ones. This negates the need to formally specify all knowledge from the offset, which would be described as a knowledge based approach. [97]

Machine Learning (ML) is a subset of AI that that learns by itself using data and looking for patterns within it. It uses the data to reprogram itself and, generally, the more data it has to learn from, the more accurate it becomes at solving a problem. This depends heavily on the representation of the data. Some simple examples include logistic regression, which can recommend a cesarean delivery [98], and naive Bayes which aims to separate e-mails into legitimate and spam [99].

Deep learning (DL) is a further subset of ML that, without human intervention, teaches itself to perform a specific task. They are based on neural networks (NNs) which are described in section 2.5.1.

AI is a huge topic and is used in many different applications. This section will focus on the tools and methods used in chapter 5.

2.5.1 *A neural network (NN)*

A NN is a subsection of ML that uses algorithms inspired by a brain's neural network structure and function [100–102]. It is more precisely called an artificial NN. In this thesis, the 'artificial' will be omitted.

NNs are computing systems that comprise of connected units organised into layers. These units are called neurons, or nodes, and the connections between them form the network. As stated earlier, these neurons are organised into layers and data flows through these, being transformed as they transition between each one. A pictorial representation is shown in figure 2.3. [100, 101]

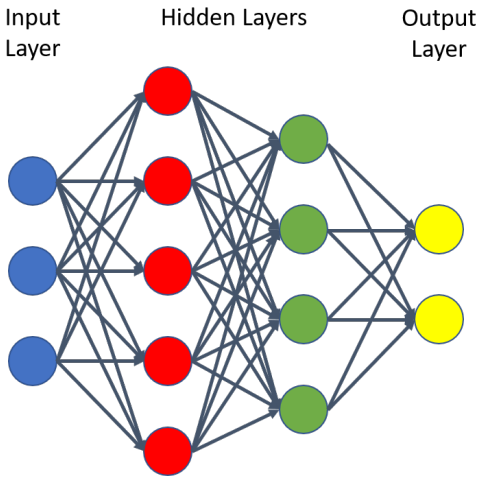


Figure 2.3: Example illustration of a neural network. The circles represent each neuron which are arranged into layers. In this example there is one input layer represented by blue circles, two hidden layers which are red and green, and one output layer in yellow. There is always one input and one output layer. The number of hidden layers can vary.

There are different types of layers in every NN; an input layer, an output layer and hidden layers. If there is more than one hidden layer, then the algorithm is known as deep learning [101]. The input layer comprises of one neuron for each component of the input data. The output layer has one neuron for each of the possible outputs. The hidden layers however, can have any number of neurons, that can be arbitrarily chosen. These apply weights, biases and thresholds to the inputs. Each layer can be thought of as a different mathematical function that defines a new representation of the input data. The number of layers in the system is often known as the depth of the model. For example, the NN shown in figure 2.3 has a depth of 3. Some approaches use the depth of the graph which describes how concepts are related to each other.

2.5.2 *Deep learning (DL)*

DL uses NN with multiple hidden layers which are sometimes referred to as deep neural networks. These can work with both labelled and unlabelled data and are capable of both supervised and unsupervised learning. They determine complicated concepts by combining simple representations together, for examples patterns and structures in the data.

For supervised learning algorithms data is supplied to the algorithm that contains features, however each data is associated with a label or target. It aims to define a network that matches incoming data to the correct label or target [97]. Supervised learning is used in this thesis.

In the case of unsupervised learning algorithms, the data is not linked to a label or target. The algorithm attempts to learn properties of the structure of the data set either implicitly or explicitly [97]. These two terms are not formally defined and often there are examples that cross-over, however they help to categorise some of the different ML algorithms.

When performing DL, several different types of NN can be used. Feedforward neural networks (FNNs) [97], recurrent neural networks (RNNs) [97, 103] and convolution neural networks (CNNs) [97, 103] are all examples. FNN, also known as multilayer perceptions, are so called as they only allow data to flow in one direction through the network. RNNs allow for cycles within the network. CNNs are NNs that, for at least one layer transition, use convolution in place of using simple multiplication/ manipulation. This thesis uses both FNN and CNN.

Feedforward neural network (FNN)

The aim of a FNN is to approximate some function f^* that classifies an input, $\boldsymbol{x}$, to a category, y:

$$y = f^*(\boldsymbol{x}). \tag{2.32}$$

It defines a mapping:

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}), \tag{2.33}$$

where $\boldsymbol{\theta}$ are all the parameters learnt by the NN that defines the best function approximation so that $f(\boldsymbol{x}; \boldsymbol{\theta})$ is the closest approximation of $f^*(\boldsymbol{x})$ as possible.

It is defined as a network since it combines multiple different functions together. For example, if there were three hidden layers then $f(\boldsymbol{x})$ would be defined as:

$$f(\boldsymbol{x}) = f^1(f^2(f^3(\boldsymbol{x}))), \tag{2.34}$$

where $f^i, i \in \{1, 2, 3\}$ are three different functions used between the layers.

During the training process, the raw data is used to drive $f(\boldsymbol{x})$ as close to $f^*(\boldsymbol{x})$ as possible. Each data, $\boldsymbol{x}$, has a label, y, and the learning algorithm decides how to alter the hidden functions (layers) to try and reproduce the closest value to this label for the output.

Convolutional neural network (CNN)

As stated earlier, a CNN uses convolution in at least one of the layer transitions. It is generally used when there is a spatial dependency within the data. CNN only go one way through the model and are therefore are a specialist form of FNNs.

Performance measure

When performing any kind of ML, in order to evaluate its abilities, a quantitive measure of performance must be derived. How this is measured is very dependent on the task required by the machine. When creating a classification algorithm, the accuracy (the proportion of correct classifications) or error rate (the proportion of incorrect classifications) can be used and implemented quite easily. For some tasks, for example density estimation, these performance measures, or any that give a value between 0 and 1, do not make sense. The most common approach is to use the average log-probability [97].

2.6 ACRONYM LIST

ACRONYMS

ADC analog to digital converter

AF Alexa Fluor

AI artificial intelligence

sCMOS scientific Complementary Metal-Oxide Semiconductor

CNN convolutional neural network
CTS charge transfer state
DL deep learning
E-H electron-hole
EM electron microscopy
EMCCD Electronic Multiplying Charge-Coupled Devices
FISH fluorescence in situ hybridisation
FNN feedforward neural network
FP fixed pattern
FPALM fluorescence photo-activated localisation microscopy
FRET forster resonance energy transfer
FT fourier transform
GFP green fluorescent protein
GPU graphics processing unit
IFT inverse fourier transform
I.I.D. independent, identically distributed
KNN K-nearest neighbours
LM localisation microscopy
LS least-squares
LSFM Light Sheet Fluorescence Microscopy
LSM Light Sheet Microscopy
MCMC Markov Chain Monte Carlo
MH Metropolis-Hastings
MIM Modified Ising Model
ML Machine Learning
MLE maximum likelihood estimate
MNIST modified national institute of standards and technology
MRF markov random field
NA numerical aperture
NN neural network
NSOM near-field scanning optical microscopy
PALM photo-activated localisation microscopy
PDF Probability Density Function
PET photo induced electron transfer

PSF point spread function
QE quantum efficiency
QW quantum well
QY quantum yield
RL Richardson-Lucy
RMS root mean squared
RNN recurrent neural network
ROI regions of interest
SIM structured illumination microscopy
SMLM Single Molecule Localisation Microscopy
SNOM scanning near-field optical microscopy
SNR signal-to-noise ratio
SOFI Super Resolution Optical Fluctuation Microscopy
SSIM saturated structured illumination microscopy
STED stimulated emission depletion
STORM stochastic optical reconstruction microscopy
SVM support vector machines
TIRF total internal reflection fluorescence
VWCM virtual window center of mass

REFERENCES

[1] C. SINGER, 'The Dawn of Microscopical Discovery', Journal of the Royal Microscopical Society **35**, 317–340 (1915).

[2] S. Bradbury, *An Introduction to the Optical Microscope* (Oxford University Press, 1989), p. 5.

[3] G. Sines and Y. A. Sakellarakis, 'Lenses in Antiquity', American Journal of Archaeology **91**, 191 (1987).

[4] S. Bradbury, *The Evolution of the Microscope* (Pergamon Press, 1967).

[5] H. Gest, 'The discovery of microorganisms by Robert Hooke and Antoni van Leeuwenhoek, Fellows of The Royal Society', Notes and Records of the Royal Society **58**, 187–201 (2004).

[6] C. S. Ball, 'The early history of the compound microscope', Bios **37**, 51–60 (1966).

[7] D. Bardell, 'The First Record of Microscopic Observations', BioScience **33**, 36–38 (1983).

[8] R. Hooke, *Micrographia* (Royal Society, 1665).

[9] K. B. Wolf and G. Krötzsch, 'Geometry and dynamics in refracting systems', European Journal of Physics **16**, 14–20 (1995).

[10] M. Hausner, *Optics Inspections and Tests: A Guide for Optics Inspectors and Designers* (2017), pp. 7–32.

[11] A. Kwan, J. Dudley and E. Lantz, 'Who really discovered snell's law?', Physics World **15**, 64 (2002).

[12] R. D. Fiete, 'Optics', in *Modeling the imaging chain of digital cameras* (SPIE Press, 2010), pp. 49–72.

[13] L. Davis, C, *Thin Lenses*, http://www.physics.louisville.edu/cldavis/phys299/notes/lo_lense Accessed: 2020-14-04.

[14] S. J. Ling, J. Sanny and B. Moebs, *4.1: single-slit diffraction*, https://phys.libretexts.org/?title=T _Optics_and_Modern_Physics_(OpenStax)/4:_Diffraction/4.1:_Single-Slit_Diffraction%0A, Visited on 2018-05-23.

[15] G. B. Airy, 'On the Diffraction of an Object-glass with Circular Aperture', Transactions of the Cambridge Philosophical Society **5**, 283–291 (1835).

[16] *Greenfluorescentblog*, `https://greenfluorescentblog.wordpress.com/2012/04/01/numerical-aperture-and-resolution/`, Accessed: 2018-01-29.

[17]E. Abbe, 'Beitrage zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung', Arch. Mikroscop Anat **9**, 413–420 (1873).

[18]T. S. Tkaczyk, 'Numerical Aperture', in *Field guide to microscopy* (2010), p. 38.

[19]B. R. Masters, 'Optical Resolution and Resolving Power : What It Is , How to Measure It , and What Limits It', in *Confocal microscopy and multiphoton excitation microscopy: the genesis of live cell imaging* (SPIE Press, 2006), pp. 49–54.

[20]N. Foundation, *The nobel prize in chemistry*, 2014.

[21]S. Balaiya, R. K. Murthy, V. S. Brar and K. V. Chalam, 'Evaluation of ultraviolet light toxicity on cultured retinal pigment epithelial and retinal ganglion cells', Clinical Ophthalmology, 33–39 (2010).

[22]M. Abramowitz and M. W. Davidson, *Anatomy of a Microscope - Numerical Aperture and Resolution*, Visited on 2020-04-16.

[23]N. Foundation, *The nobel prize in physics*, 1929.

[24]W. J. Croft, *Under the microscope [electronic resource] : a brief history of microscopy* (World Scientific Publishing Co. Pte. Ltd., 2006), pp. 57–72.

[25]R. Erni, M. D. Rossell, C. Kisielowski and U. Dahmen, 'Atomic Resolution Imaging with a sub-50 pm Electron Probe', Physical Review Letters **102**, 96–101 (2009).

[26]N. Foundation, *The nobel prize in physics*, 1986.

[27]J. Ayache, L. Beaunier, J. Boumendil, G. Ehret and D. Laub, 'Artifacts in Transmission Electron Microscopy', in *Sample preparation handbook for transmission electron microscopy* (Springer, New York, NY, 2010), pp. 125–170.

[28]J. G. Mcnally, T. Karpova, J. Cooper and J. A. Conchello, 'Three-Dimensional Imaging by Deconvolution Microscopy', Methods **19**, 373–385 (1999).

[29]G. Sparacino, G. De Nicolao, G. Pillonetto and C. Cobelli, 'Deconvolution', in *Modelling methodology for physiology and medicine* (Elsevier, 2014), pp. 45–68.

[30]M. Minsky, 'Memoir on inventing the confocal scanning microscope', Scanning **10**, 128–138 (1988).

[31]M. Minsky, *Microscopy Apparatus*, 1961.

[32]M. Renz, 'Fluorescence microscopy-A historical and technical perspective', Cytometry Part A **83**, 767–779 (2013).

[33]H. Helmholtz, 'Die theoretischen Grenzen für die Leistungsfähigkeit der Mikroskope.', Annalen der Physik, 557–584 (1874).

[34]M. Muyskens and Ed Vitz, 'The Fluorescence of Lignum nephriticum: A Flash Back to the Past and a Simple Demonstration of Natural Substance Fluorescence', Journal of Chemical Education **83**, 765 (2006).

[35]G. G. Stokes, 'On the Change of Refrangibility of Light', Philosophical Transactions of the Royal Society of London **142**, 463–562 (1852).

[36]G. G. Stokes, 'On the Change of Refrangibility of Light. No. II', Philosophical Transactions of the Royal Society of London **143**, 385–396 (1853).

[37]L. J. Kricka and P. Fortina, 'Analytical ancestry: "firsts" in fluorescent labeling of nucleosides, nucleotides, and nucleic acids', Clinical Chemistry **55**, 670–683 (2009).

[38]D. J. S. Birch, Y. Chen and O. J. Rolinski, 'Fluorescence', in *Photonics, volume 4 : biomedical photonics, spectroscopy, and microscopy* (John Wiley & Sons, Incorporated, 2015), pp. 1–58.

[39]G. T. Dempsey, J. C. Vaughan, K. H. Chen, M. Bates and X. Zhuang, 'Evaluation of fluorophores for optimal performance in localization-based super-resolution imaging', Nature Methods **8**, 1027–1036 (2011).

[40]R. Smallman and A. Ngan, 'Characterization and Analysis', in *Modern physical metallurgy* (Elsevier, 2014), pp. 159–250.

[41]H. Siedentopf and R. Zsigmondy, 'Uber Sichtbarmachung und Größenbestimmung ultramikoskopischer Teilchen, mit besonderer Anwendung auf Goldrubingläser', Annalen der Physik **315**, 1–39 (1902).

[42]D. Axelrod, 'Cell-substrate contacts illuminated by total internal reflection fluorescence.', The Journal of Cell Biology **89**, 141–145 (1981).

[43]Y. Markaki and H. Harz, eds., *Light Microscopy*, Vol. 1563, Methods in Molecular Biology (Springer New York, New York, NY, 2017).

[44]B. O. Leung and K. C. Chou, 'Review of super-resolution fluorescence microscopy for biology.', Applied spectroscopy **65**, 967–80 (2011).

[45]S. W. Hell and J. Wichmann, 'Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy', Optics Letters **19**, 780 (1994).

[46]R. C. Dunn, 'Near-field scanning optical microscopy.', Chemical reviews **99**, 2891–928 (1999).

[47]M. G. L. Gustafsson, 'Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution', Proceedings of the National Academy of Sciences **102**, 13081–13086 (2005).

[48]E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz and H. F. Hess, 'Imaging intracellular fluorescent proteins at nanometer resolution.', Science (New York, N.Y.) **313**, 1642–5 (2006).

[49]M. Rust, M. Bates and X. Zhuang, 'Stochastic optical reconstruction microscopy (STORM) provides sub-diffraction-limit image resolution', Nature methods **3**, 793–795 (2006).

[50]S. T. Hess, T. P. K. Girirajan and M. D. Mason, 'Ultra-high resolution imaging by fluorescence photoactivation localization microscopy.', Biophysical journal **91**, 4258–72 (2006).

[51]T. Dertinger, R. Colyer, G. Iyer, S. Weiss and J. Enderlein, 'Fast, background-free, 3D super-resolution optical fluctuation imaging (SOFI).', Proceedings of the National Academy of Sciences of the United States of America **106**, 22287–22292 (2009).

[52]B. Huang, M. Bates and X. Zhuang, 'Super-Resolution Fluorescence Microscopy', Annual Review of Biochemistry **78**, 993–1016 (2009).

[53]D. Wildanger et al., 'Solid immersion facilitates fluorescence microscopy with nanometer resolution and sub-Ångström emitter localization', Advanced Materials **24**, 309–313 (2012).

[54]R. Heintzmann, T. M. Jovin and C. Cremer, 'Saturated patterned excitation microscopy—a concept for optical resolution improvement', Optical Society of America **19**, 1599–1609 (2002).

[55]E. H. Rego, L. Shao, J. J. Macklin, L. Winoto, G. A. Johansson, N. Kamps-Hughes, M. W. Davidson and M. G. Gustafsson, 'Nonlinear structured-illumination microscopy with a photoswitchable protein reveals cellular structures at 50-nm resolution', Proceedings of the National Academy of Sciences of the United States of America **109**, 13–15 (2012).

[56]E. Betzig, 'Proposed method for molecular optical imaging', Optics Letters **20**, 237 (1995).

[57]W. Moerner and L. Kador, 'Optical detection and spectroscopy of single molecules in a solid', Physical Review Letters **62**, 2535–2538 (1989).

[58]M. Orrit and J. Bernard, 'Single pentacene molecules detected by fluorescence excitation in a p-terphenyl crystal', Physical Review Letters **65**, 2716–2719 (1990).

[59]W. P. Ambrose and W. E. Moerner, 'Fluorescence spectroscopy and spectral diffusion of single impurity molecules in a crystal', Nature **349**, 225–227 (1991).

[60]R. M. Dickson, A. B. Cubitt, R. Y. Tsien and W. E. Moerner, 'On / off blinking and switching behaviour of single molecules of green fluorescent protein', **388**, 355–358 (1997).

[61]M. Heilemann, S. van de Linde, A. Mukherjee and M. Sauer, 'Super-resolution imaging with small organic fluorophores.', Angewandte Chemie (International ed. in English) **48**, 6903–8 (2009).

[62]M. Heilemann, S. van de Linde, M. Schüttpelz, R. Kasper, B. Seefeldt, A. Mukherjee, P. Tinnefeld and M. Sauer, 'Subdiffraction-resolution fluorescence imaging with conventional fluorescent probes.', Angewandte Chemie (International ed. in English) **47**, 6172–6 (2008).

[63]L. Stryer and R. P. Haugland, 'Energy transfer: a spectroscopic ruler.', Proceedings of the National Academy of Sciences **58**, 719–726 (1967).

[64]*Single-molecule localization microscopy: directory of smlm software*, http://bigwww.epfl.ch/smlm, Visited on 2020-06-19, 2017.

[65]P. D. Simonson, E. Rothenberg and P. R. Selvin, 'Single-molecule-based super-resolution images in the presence of multiple fluorophores.', Nano letters **11**, 5090–6 (2011).

[66]S. A. Jones, S.-H. Shim, J. He and X. Zhuang, 'Fast, three-dimensional super-resolution imaging of live cells', Nature Methods **8**, 499–505 (2011).

[67]A. Small and S. Stahlheber, 'Fluorophore localization algorithms for super-resolution microscopy.', Nature methods **11**, 267–79 (2014).

[68]A. V. Abraham, S. Ram, J. Chao, E. S. Ward and R. J. Ober, 'Quantitative study of single molecule location estimation techniques', Optics Express **17**, 23352 (2009).

[69]R. E. Thompson, D. R. Larson and W. W. Webb, 'Precise nanometer localization analysis for individual fluorescent probes.', Biophysical journal **82**, 2775–2783 (2002).

[70]S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice Hall, 1993).

[71]R. Henriques, M. Lelek, E. F. Fornasiero, F. Valtorta, C. Zimmer and M. M. Mhlanga, 'QuickPALM: 3D real-time photoactivation nanoscopy image processing in ImageJ.', Nature methods **7**, 339–40 (2010).

[72]R. Parthasarathy, 'Rapid, accurate particle tracking by calculation of radial symmetry centers', Nature Methods **9**, 724–726 (2012).

[73]A. J. Berglund, M. D. McMahon, J. J. McClelland and J. A. Liddle, 'Fast, bias-free algorithm for tracking single particles with variable size and shape', Optics Express **16**, 14064 (2008).

[74]H. Ma, F. Long, S. Zeng and Z.-L. Huang, 'Fast and precise algorithm based on maximum radial symmetry for single molecule localization', Optics Letters **37**, 2481 (2012).

[75]T. Quan, H. Zhu, X. Liu, Y. Liu, J. Ding, S. Zeng and Z.-L. Huang, 'High-density localization of active molecules using Structured Sparse Model and Bayesian Information Criterion', Optics Express **19**, 16963 (2011).

[76]Y. Wang, T. Quan, S. Zeng and Z.-L. Huang, 'PALMER: a method capable of parallel localization of multiple emitters for high-density localization microscopy', Optics Express **20**, 16039 (2012).

[77]F. Huang, S. L. Schwartz, J. M. Byars and K. a. Lidke, 'Simultaneous multiple-emitter fitting for single molecule super-resolution imaging.', Biomedical optics express **2**, 1377–1393 (2011).

[78]F. Huang et al., 'Video-rate nanoscopy using sCMOS camera-specific single-molecule localization algorithms.', Nature methods **10**, 653–8 (2013).

[79]S. J. Holden, S. Uphoff and A. N. Kapanidis, 'DAOSTORM: an algorithm for high-density super-resolution microscopy', Nature methods **8**, 279–280 (2011).

[80]D. R. Tobergte and S. Curtis, 'Daophot: a Computer Program for Crowded-Field Stellar Photometry (Psf Photometry)', Journal of Chemical Information and Modeling **53**, 1689–1699 (2013).

[81]L. B. Lucy, 'An iterative technique for the rectification of observed distributions', Astronomical Journal **79**, 745–754 (1974).

[82]W. Richardson, 'Bayesian-based iterative method of image restoration', Optical Society of America **62**, 55–59 (1972).

[83]L. Zhu, W. Zhang, D. Elnatan and B. Huang, 'Faster STORM using compressed sensing', Nature Methods **9**, 721–723 (2012).

[84]S. Cox, E. Rosten, J. Monypenny, T. Jovanovic-Talisman, D. T. Burnette, J. Lippincott-Schwartz, G. E. Jones and R. Heintzmann, 'Bayesian localization microscopy reveals nanoscale podosome dynamics.', Nature methods **9**, 195–200 (2012).

[85]S. Wolter, U. Endesfelder, S. van de Linde, M. Heilemann and M. Sauer, 'Measuring localization performance of super-resolution algorithms on very active samples.', Optics express **19**, 7020–33 (2011).

[86]S. Wang, J. R. Moffitt, G. T. Dempsey, X. S. Xie and X. Zhuang, 'Characterization and development of photoactivatable fluorescent proteins for single-molecule-based superresolution imaging', Proceedings of the National Academy of Sciences of the United States of America **111**, 8452–8457 (2014).

[87]N. Durisic, L. Laparra-Cuervo, Á. Sandoval-Álvarez, J. S. Borbely and M. Lakadamyali, 'Single-molecule evaluation of fluorescent protein photoactivation efficiency using an in vivo nanotemplate', Nature Methods **11**, 156–162 (2014).

[88]P. J. Green, 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', Biometrika **82**, 711 (1995).

[89]H. Jeffreys, *Theory of probability*, 3rd (Press, Oxford: University, 1961).

[90]L. Savage, *The foundations of Statistics* (New York: Wiley, 1972).

[91]B. De Finetti, *Theory of probability* (London: Wiley, 1974).

[92]R. Christensen, T. E. Hanson, W. Johnson, Branscum and Adam, *Bayesian Ideas and Data Analysis : An Introduction for Scientists and Statisticians*, 1st ed. (Taylor & Francis Group, 2010).

[93]S. Chib and E. Greenberg, 'Understanding the Metropolis-Hastings Algorithm', The American Statistician **49**, 327–335 (1995).

[94]M. Jesper, ed., *Spactial Statistics and Computational Methods* (Springer, 2003), pp. 1–13.

[95]K. I. Mortensen, L. S. Churchman, J. A. Spudich and H. Flyvbjerg, 'Optimized localization analysis for single-molecule tracking and super-resolution microscopy', Nature Methods **7**, 377–381 (2010).

[96]J. R. Janesick, *Photon Transfer* (SPIE.Digital Library, 2007).

[97]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, `http://www.deeplearningbook.org` (MIT Press, 2016).

[98]S. Mor-Yosef, A. Samueloff, B. Modan, D. Navot and J. G. Schenker, 'Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study.', Obstetrics and gynecology **75**, 944–7 (1990).

[99]V. Metsis, I. Androutsopoulos and G. Paliouras, 'Spam Filtering with Naive Bayes – Which Naive Bayes?', In CEAS (2006).

[100]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, http://www.deeplearningbook.org (MIT Press, 2016).

[101]*Neural Network Programming - Deep Learning with PyTorch*, https://deeplizard.com/learn/ Accessed: 2020-07-08.

[102]D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', Nature **323**, 533–536 (1986).

[103]M. Torrisi, G. Pollastri and Q. Le, 'Deep learning methods in protein structure prediction', Computational and Structural Biotechnology Journal **18**, 1301–1310 (2020).

[104]*Single-molecule localization microscopy: collection of reference datasets*, http://bigwww.epfl.ch/smlm/datasets/index.html, Visited on 2020-06-19, 2017.

[105]H. Li and J. C. Vaughan, 'Switchable Fluorophores for Single-Molecule Localization Microscopy', Chemical Reviews **118**, 9412–9454 (2018).

[106]B. A. Griffin, S. R. Adams, R. Y. Tsien, B. A. Griffin, S. R. Adams and R. Y. Tsient, 'Specific Covalent Labeling of Recombinant Protein Molecules Inside Live Cells Published by : American Association for the Advancement of Science Stable URL : http://www.jstor.org/stable/2896025 REFERENCES Linked references are available on JSTOR for this', Science **281**, 269–272 (1998).

[107]L. W. Miller, Y. Cai, M. P. Sheetz and V. W. Cornish, 'In vivo protein labeling with trimethoprim conjugates: A flexible chemical tag', Nature Methods **2**, 255–257 (2005).

[108]A. Keppler, S. Gendreizig, T. Gronemeyer, H. Pick, H. Vogel and K. Johnsson, 'A general method for the covalent labeling of fusion proteins with small molecules in vivo', Nature Biotechnology **21**, 86–89 (2003).

[109]A. Gautier, A. Juillerat, C. Heinis, I. R. Corrêa, M. Kindermann, F. Beaufils and K. Johnsson, 'An Engineered Protein Tag for Multiprotein Labeling in Living Cells', Chemistry and Biology **15**, 128–136 (2008).

[110]G. V. Los et al., 'HaloTag: A novel protein labeling technology for cell imaging and protein analysis', ACS Chemical Biology **3**, 373–382 (2008).

[111]I. Y. Iourov, *Fluorescence In Situ Hybridization (FISH)*, edited by T. Liehr, Springer Protocols Handbooks (Springer Berlin Heidelberg, Berlin, Heidelberg, 2017), pp. 17–25.

[112]M. E. Brezinski, *Noise and system performance with td-oct and sd-oct* (Academic Press, Amsterdam ; 2006), pp. 175–195.

[113]T. Bushnell, *What is autofluorescence*, https://expert.cheekyscientist.com/what-is-autofluorescence/, Visited on 2020-06-09.

[114]B. Huang, W. Wang, M. Bates and X. Zhuang, 'Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy.', Science (New York, N.Y.) **319**, 810–3 (2008).

[115]M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych and G. M. Hagen, 'ThunderSTORM: A comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging', Bioinformatics **30**, 2389–2390 (2014).

[116]B. A. Cipra, 'An Introduction to the Ising Model', The American Mathematical Monthly **94**, 937–959 (1987).

[117]I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, 'Chapter 9 - Probabilistic methods', in *Data mining practical machine learning tools and techniques*, edited by I. H. Witten, E. Frank, M. A. Hall and C. J. B. T. D. M. (E. Pal (Morgan Kaufmann, 2017), pp. 335–416.

[118]B. A. Landman, I. Lyu, Y. Huo and A. J. Asman, 'Chapter 6 - Multiatlas segmentation', in *Handbook of medical image computing and computer assisted intervention*, edited by S. K. Zhou, D. Rueckert, G. B. T. H. o. M. I. C. Fichtinger and C. A. Intervention (Academic Press, 2020), pp. 137–164.

[119]Orchard, Peter, *Markov random field optimisation*, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0809/ORCHARD/, Accessed 2014-08-28.

[120]Q. Jackson and D. Landgrebe, 'Adaptive Bayesian contextual classification based on Markov random fields', Geoscience and Remote Sensing, ... **40**, 2454–2463 (2002).

[121]A. Sonnleitner, G. Schütz and T. Schmidt, 'Free Brownian Motion of Individual Lipid Molecules in Biomembranes', Biophysical Journal **77**, 2638–2642 (1999).

[122]M. K. Cheezum, W. F. Walker and W. H. Guilford, 'Quantitative comparison of algorithms for tracking single fluorescent particles', Biophysical Journal **81**, 2378–2388 (2001).

[123]U. Kubitscheck, O. Kückmann, T. Kues and R. Peters, 'Imaging and Tracking of Single GFP Molecules in Solution', Biophyiscal Journal **78**, 2170–2179 (2000).

[124]*Markov random fields*, https://www2.isye.gatech.edu/isyebayes/bank/handout16.pdf, Accessed August 11, 2022.

[125]*Classification Learner*, https://uk.mathworks.com/help/stats/classificationlearner-app.html, Accessed: 2020-08/-8.

[126]M. J. Rust, M. Bates and X. Zhuang, 'Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM).', Nature methods **3**, 793–5 (2006).

[127]R. P. Nieuwenhuizen, M. Bates, A. Szymborska, K. A. Lidke, B. Rieger and S. Stallinga, 'Quantitative localization microscopy: Effects of photophysics and labeling stoichiometry', PLoS ONE **10**, 1–18 (2015).

[128]D. Gross and W. Webb, 'Molecular counting of low-density lipoprotein particles as individuals and small clusters on cell surfaces', Biophysical Journal **49**, 901–911 (1986).

[129]B. M. Burton, K. A. Marquis, N. L. Sullivan, T. A. Rapoport and D. Z. Rudner, 'The ATPase SpoIIIE Transports DNA across Fused Septal Membranes during Sporulation in Bacillus subtilis', Cell **131**, 1301–1312 (2007).

[130]M. C. Leake, J. H. Chandler, G. H. Wadhams, F. Bai, R. M. Berry and J. P. Armitage, 'Stoichiometry and turnover in single, functioning membrane protein complexes', Nature **443**, 355–358 (2006).

[131]S. K. Das, M. Darshi, S. Cheley, M. I. Wallace and H. Bayley, 'Membrane protein stoichiometry determined from the step-wise photobleaching of dye-labelled subunits', ChemBioChem **8**, 994–999 (2007).

[132]K. Tsekouras, T. C. Custer, H. Jashnsaz, N. G. Walter and S. Pressé, 'A novel method to accurately locate and count large numbers of steps by photobleaching', Molecular Biology of the Cell **27**, edited by D. Lidke, 3601–3615 (2016).

[133]A. Lee, K. Tsekouras, C. Calderon, C. Bustamante and S. Pressé, 'Unraveling the Thousand Word Picture: An Introduction to Super-Resolution Data Analysis', Chemical Reviews **117**, 7276–7330 (2017).

[134]E. Alpaydin, 'Introduction', in *Introduction to machine learning* (2010), pp. 1–19.

[135]C. Campbell and Y. Ying, 'Learning with Support Vector Machines', Synthesis Lectures on Artificial Intelligence and Machine Learning **5**, 1–95 (2011).

[136]N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (Cambridge University Press, Mar. 2000).

[137]V. Hodge and J. Austin, 'A Survey of Outlier Detection Methodologies', Artificial Intelligence Review **22**, 85–126 (2004).

[138]M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks', Neural Networks **106**, 249–259 (2018).

[139]L. Deng, 'The mnist database of handwritten digit images for machine learning research [best of the web]', IEEE Signal Processing Magazine **29**, 141–142 (2012).

[140]S. L. Smith, P.-j. Kindermans, C. Ying, Q. V. Le and G. Brain, 'D on ' T D Ecay the L Earning R Ate , I Ncrease the B Atch S Ize', 1–11 (2018).

[141]*Cross entropy loss*, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, Accessed April 02, 2021.

[142]Z. Zhang and M. R. Sabuncu, 'Generalized cross entropy loss for training deep neural networks with noisy labels', Advances in Neural Information Processing Systems **2018-December**, 8778–8788 (2018).

[143]*Cross-entropy loss function*, https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e, Accessed April 02, 2021.

3

SIMULATIONS

3.1 SUMMARY

A simulation software has been developed that creates realistic Single Molecule Localisation Microscopy (SMLM) images. Although the simulation software replicates the photophysics seen from Alexa Fluor (AF) 647 in a typical SMLM experiment, it could be altered to include any other fluorophore for which the emission statistics are well characterised. This software gives the user control over the placement and photophysics of fluorophores, allowing different population densities, and images with varied signal-to-noise ratio (SNR) to be created.

These simulations are used in chapter 4 to test the limits of the denoising algorithm. Having the flexibility over all the characteristics of the simulations are used in chapter 5 when looking at how images look and change when fluorophores are placed in close proximity.

3.2 INTRODUCTION

Simulations are used in many different fields including biology, chemistry, physics, performance engineering, economics and the social sciences. They are used to imitate the operation of a real process or system. In order to work, a model has to be created which represents the key characteristics/ behaviours/ functions of the system or process that wants to be investigated.

When looking at images that have been reconstructed from real SMLM datasets, it is not possible to know the ground truth; the exact locations of the fluorescent molecules, how many molecules are in the sample, and whether the molecules have undergone multiple emitting events are all unknown. Using simulations, it is possible to create test datasets where this information is known, which can then be used to test the limitations of reconstruction and analysis algorithms.

Although there are simulated data available online [104], this algorithm was created so that the simulated images could be more customisable. The simulation algorithm was created to understand the science, to enable both standard and high/low density images to be produced, and to try and count the number of fluorophores in a sample (chapter 5). Having more flexibility in the production process allows as many fluorophores to be in close proximity of each other as needed, as well as knowing exactly how many are there and their exact location. By doing this, it allows more scenarios to be explored and the distribution of the different populations (signal and noise) can be determined and used to help count the number of fluorophores. The images can be specifically tailored to the needs required for each step in the process.

Previously, the photophysics of several popular fluorescent dyes [39] and fluorescent proteins [86] have been characterised, in order to evaluate their performance in room-temperature SMLM experiments. Some photophysical properties which are crucial to SMLM are:

1. the number of photons per switching event,
2. the on-off duty cycle,
3. photo-stability,
4. the total number of photoswitching cycles/ survival fraction,
5. the ability to control the fluorophores activation rate and
6. how the buffer affects the fluorophore.

The values for these parameters all vary substantially between the different fluorophores, and are also highly dependent upon the environment that the fluorophore is placed in. For example, when AF 488 is in a β-mercaptoethanol environment, its mean number of detected photons per switching event is 427 [39]. However, when in a mercaptoethylamine environment, its mean number of detected photons per switching event is 1,193 [39].

The number of detected photons per switching event for a particular type of fluorescent molecule follows an exponential distribution [39]. Figure 3.1 shows this distribution for AF 647 in a β-mercaptoethanol environment. This particular fluorophore, in this environment, yields a mean number of 5,202 photons per switching event. As the process is stochastic, the value varies for each event. The probability that a fluorophore will enter the 'off' state is the same at any given time throughout the process, but the longer it has been in the 'on ' state the more likely it is to change state. Due to the localisation precision being inversely proportional to the square root of the number of photons (equation (1.13)),

a high photon yield is desirable to obtain an accurate position of the probes. The photon number varies substantially between different fluorescent molecules; for example, AF 488 in β-mercaptoethanol environment yields an average of 427 photons per switching cycle, whereas Atto 565 in a mercaptoethylamine environment can yield an average of 19,714 photons. [39]

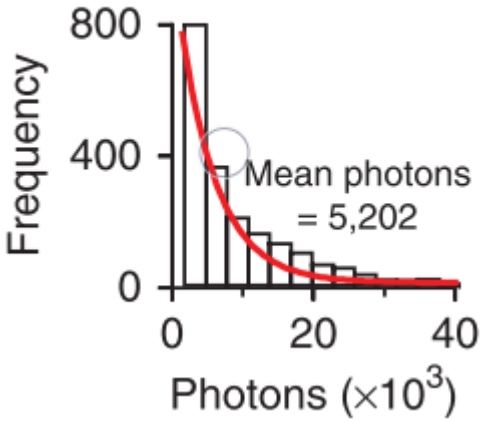


Figure 3.1: Dempsey *et al.* [39] determined the number of detected photons for many different switching events using single-molecule fluorescence time traces for different fluorophores. This histogram was constructed from many events from hundreds of molecules for AF 647 in a β-mercaptoethanol environment. This can be fit with an exponential function whose mean is 5,202 photons. This can vary substantially, from 427 photons for AF 488 in a β-mercaptoethanol environment and 19,714 photons for Atto 565 in a mercaptoethylamine environment.

The on-off duty cycle describes the average fraction of time that a fluorophore spends in the 'on' state relative to the time it spends in its 'off' state. For example, if the fluorophore is 'on' for 1ms every second, then the duty cycle would be 0.001. A low duty cycle is desirable as the maximum number of fluorophores that are able to be localised in a diffraction limited area is inversely proportional to the duty cycle. [39] However, the lower the duty cycle the more raw data has to be taken to create a full reconstructed image as there will be fewer fluorophores 'on' per image, and therefore fewer switching events for image reconstruction. Therefore a compromise needs to be found. Ideally, the duty cycle and the frame rate would be in sync, however, this is near impossible due to the stochastic nature of the fluorophores.

Photo-stability describes how the fluorophore is resistant to chemical changes under illumination [39]. If it is very photostable then it will less less susceptible to photobleaching. If it is unstable, it will bleach rapidly due to the laser illumination and no longer be able to fluoresce. This can vary depending on the buffer used and the environment the samples are in when imaged.

The total number of photoswitching cycles/ survival fraction is the average number of times the fluorophore oscillates between its 'on' and 'off' state before it bleaches [39]. As SMLM can be used for a variety of applications, the desired number of times a fluorophore switches between each state varies. If using SMLM to localise molecules, once a fluorophore has been located then it would be ideal if it would bleach. This would help to combat double counting. However, if SMLM is being used to track molecules, then ideally the fluorophore would be able to switch an infinite number of times and never bleach. This statistic is related quite closely to the photostability.

The ability to control the fluorophores activation rate is how well the fluorophore can be controlled using a light source independent from the imaging light, and if it can be forced to stay in its 'off' state for longer periods of time. This allows the density of 'on' fluorophores to be controlled. [39]

Many different buffers are used in SMLM. These can strongly affect the fluorophores switching properties and change the resulting image quality. Currently it is not fully understood how these buffers work, nor is it well documented, however, they are often used to stop the fluorophores from reacting with oxygen (oxidising) which often causes them to bleach more quickly. Choosing the correct buffer composition is essential for robust on-off switching of dyes [39]. They often include a thiol as some fluorophores, including AF 647 require this for robust switching as it is thought they form a covalent conjugate with a thiol in its dark state. Some fluorophores can still perform without a thiol which suggests that thiol-independent pathways can exist for some dyes [39].

3.3 REAL DATA PRODUCTION

After the type of fluorophore has been chosen (which depends upon the structure of interest, type of sample etc.) they are attached to the structure of interest. This can be done in a variety of different ways that will only be touched on briefly here. A more detailed overview can be found in reference [105].

The range of methods for attaching molecules spans both covalently or non-covalently bonding of small molecules, fluorescent proteins, antibodies, oligonucleotides, proteins or even peptides [105]. Methods include immunostaining [105], protein fusion tags [106–110], small-molecule affinity probes, fluorescence in situ hybridisation (FISH) probes [111] and metabolically incorporated probes.

Choosing the correct method to attach the fluorophore depends upon many different factors including, but not limited, to the particle/ structure of interest, if imaging live cells and whether using multiple colour channels. Two factors that need to be considered are the size of the fluorophores and the distance they are from the particle/ structure of interest. Both these factors will affect the precision of the localisations.

When taking SMLM images, at any given time there is only a subset of fluorophores emitting light due to the chemical properties of the molecules. They each emit light from a point source at the sample. If the frame is taken over 3ms, the amount of photons that any one of these fluorophores emits during this time depends on a variety of different factors including, but not limited to, the intensity distribution, on/off rates, bleaching rates of the specific fluorophore being used, and the laser intensity at that specific point of the sample.

If a fluorophore is 'on' for the whole of the frame then it is more likely to emit more photons than if it only turns on half way through or at the end of the frame, especially if it has a constant rate of emission. This increases the variation of the signal and widens the distribution of the detected photons per molecule for each frame. This is also referred to as noise within the signal and will be for the rest of this thesis.

Photons will leave the fluorophores in all different directions. Photons that reach and pass through the coverslip then enter the microscope before being collected by the camera. There are often many optical components which manipulate the photons around the microscope including filters, lenses and mirrors. Filters stop certain wavelengths from being transmitted however they often have a band pass and let a small range of wavelengths through rather than one specific one. They will also cause some photons to be reflected and therefore reduce the amount of emitted photons that are detected. Lenses and glass components will also reflect some of the photons back due to the change of refractive index the photon is traveling through. This will again reduce the number of photons that reach the camera. All of these components will slightly alter the amount of photons that reach the camera.

When the collected light reaches the camera, the individual photons are binned in space. This depends on both the size of the pixels within

the camera and on the magnification of the microscope. The camera converts the majority of these photons into photoelectrons which are read and used to create an image depending on the quantum efficiency (QE) of the camera. More of this is discussed in section 2.4.

The camera also adds a variety of different sources of noise including, but not limited to, shot noise, read noise and fixed pattern (FP) noise. These are discussed in more detail in section 3.4.1.

3.4 NOISE

There are a variety of different sources of noise when using SMLM. It can arise from external sources, the camera, autofluorescence, and is even within the signal itself. Noise coming from external sources are very unlikely if the set up of the microscope is optimal. They are indistinguishable and therefore still convert electrons into photoelectrons within the camera and therefore add noise to the final images. As the camera and microscope are (usually) set up within a dark room and/or box, this is quite unlikely and, compared to other sources of noise, causes minimal effect.

3.4.1 *Camera Noise*

There are several potential sources of noise in cameras. Four fundamental noise sources are signal shot noise, Fano noise, FP noise and read noise [96]. Signal shot noise and Fano noise are related to photon interaction, FP noise is due to non-uniformity between pixels (mainly in scientific Complementary Metal-Oxide Semiconductor (sCMOS) cameras) and read noise is a compound noise that encompasses all sources that do not depend on signal strength.

Shot noise

Shot noise is known by many different names including photon noise, Bose noise and Poisson noise. It occurs because light comes in discrete, or 'quantised' packets: photons. There are also spontaneous fluctuations due to position-momentum uncertainty, as well as fluctuations caused by the uncertainty of electric- and magnetic fields. The variance of photons arriving at the camera is known as shot noise. [96]

As described in section 1.7 and shown in the Jablonski diagram in figure 1.13, when a fluorophore is in its 'on' state it is transitioning

between three different states; its ground state (S_0), its first excited singlet state (S_1) and triplet state (T_1). The different transitions can be seen in figure 1.13 and they have probabilities specific to the fluorophore used. Every time it drops from S_1 down to S_0 it releases a photon which can be detected. The rate at which this occurs is different for each fluorophore and varies both spatially and temporally. These fluctuations can also be thought of as shot noise.

Fano Noise

Fano noise is fundamentally related to the charge generated by a photon's interaction with a semiconductor. As not all the energy of an interacting photon is spent in the production of an electron-hole (e-h) pair, there is a variation in the number produced by each photon. This variance is the Fano noise. This is not as prevalent in modern cameras and is insignificant compared to other sources of noise in the cameras used for SMLM

Fixed pattern Noise

Although present in both Electronic Multiplying Charge-Coupled Devices (EMCCD) and sCMOS cameras, FP noise is more prevalent in sCMOS cameras due to how they are made. Some pixels will be able to collect charge more efficiently than others due to slight differences in electrical components. If all the pixels are illuminated with the same intensity there will be a pattern of 'hot' (brighter) and 'cold' (darker) pixels [112]. It is called 'fixed' as it is the non-uniformity in each pixel is temporally constant. Therefore the spatial pattern is the same image to image for the same camera.

As EMCCD cameras use the same read-out register for all pixels, the FP comes only from the ability of individual pixels to collect charge. In sCMOS cameras each pixel has its own read-out register and therefore the FP noise will be more varied. It can be mostly corrected for depending on the camera in question, and some sophisticated cameras correct for this themselves.

FP noise is often removed by using a process called flat fielding [96]. A computer is used to adjust pixel sensitivity so that they are all equal. The camera is presented with extended light source that has nearly uniform radiance. The images the camera collects will have pixel-to-pixel non-uniformities. As the light source is uniform across all pixels, these must be inherent to the camera itself.

FP noise is proportional to signal and also the cleanliness of cameras can also affect it. For example, if there are any dust particles or scratches on the camera then these will also contribute to the FP noise [112].

Read Noise

Read noise is a compound noise that encompasses all sources that do not depend on signal strength. These include, but are not limited to, pixel source follower noise, sense node reset noise, thermal dark current shot noise, dark current FP noise, analog to digital converter (ADC) and system noise. [96]

Pixel source follower noise is caused by the residual electrons left after the capacitor has been emptied. There will be some electrons left in the capacitor when it is emptied and these will contribute towards the subsequent image. It is both spatially and temporally systematic, and therefore will affect each pixel differently at each new image.

Sense node reset noise is generated thermally by the channel resistance associated with the reset of the capacitor [112]. Every time a frame is read, energy is added to the system to enable the electrons to flow and be counted. This addition of energy adds electrons to the system which are counted by the detector and is related to the frequency or frame rate. If the frame rate is faster, fewer photons arising from the true source will have been detected, while the noise will contribute highly, and therefore the signal to noise ratio will be lower. A compromise has to be made for frame rate so that this effect can be minimised.

As mentioned before in section 2.4, thermal dark current comes from electrons that acquire enough thermal energy to jump over the band gap into the quantum well (QW) and be registered by the pixel. This is temperature dependent and many of the high end cameras used are now cooled to minimise this effect. Often the room is temperature controlled as well [112]. It consists of both dark current shot noise and dark FP noise. The latter dominates the former and can be determined by taking images with no light source.

ADC quantising noise occurs due to the uncertainty from converting pixel signal into a digital value.

All remaining sources of noise are often combined, and collectively known as "system noise". Sources of system noise include, but are not limited to: transient noise, luminescence, ADC feedback noise, power supply noise and electromagnetic noise. [112]

3.4.2 *Noise within the signal*

The noise in the signal is related to the stochastic nature of the fluorophores and can be confused with shot noise. Per frame, the number of photons collected per fluorophore depends upon when the fluorophore turns on within that frame and the rate at which it emits photons.

The rate that a fluorophore emits photons when it is 'on' is related to the intensity of the laser. When the fluorophore absorbs a photon from the laser it transitions from S_0 to S_1 (see figure 1.13). The higher the intensity of the laser the greater the probability that a (laser) photon will hit the fluorophore causing it to jump to S_1. The lower this intensity the lower the probability the fluorophore will be excited and therefore it will stay longer in the ground state. The wave-front for a laser generally has a Gaussian Distribution and therefore the fluorophores in the centre of the sample will receive a higher intensity of photons than fluorophores nearer the edge. Although lasers with a flat wave-front are available, they are often very expensive and not widely used.

Transitions from both S_1 and T_1 down to S_0 (see figure 1.13) are very difficult to control and can vary substantially both across a sample using the same fluorophores, as well as between different types of fluorophores. The ratio between time spent in S_1 and T_1 affects the average rate of photon emission. Due to these factors, two fluorophores that are in their 'on' state for the same amount of time may still emit a different number of photons.

The number of photons ultimately detected per frame will also vary depending upon when the fluorophore leaves the 'off' state and enters the 'on' state. If a fluorophore begins emitting at the start of a frame, then more photons will be detected in that frame compared to a fluorophore which has started to emit photons part way through the frame. This variation in the number of detected photons affects the ability to measure the location of the fluorophores, and contributes to noise within the signal.

3.4.3 *Autofluorescence*

Autofluorescence describes the natural fluorescence that occurs in biological samples. Some compounds and amino acids inside cells fluoresce naturally when exposed to certain wavelengths. Autofluorescence increases the variance in the noise and can add a further population into the system. This would increase the number of populations from two; signal and noise, to three; signal, autofluorescence and noise. [113]

3.5 SIMULATIONS

The simulation software described in this chapter creates a stack of images which replicate typical datasets acquired from a SMLM experiment using the fluorophore AF 647 as the fluorescent label. The photophysic characteristics of AF 647 were obtained from literature [39], and these values could easily be changed to adapt the model to one of the other commonly-used and well-characterised fluorophores.

The number of fluorophores and frames can be chosen by the user. The frame rate is set to 100 frames per second (10ms long exposure times) and is related to the probabilities chosen so is fixed. In order to change this, all the other variables would have to be recalculated. For every nanosecond in time, the state of each molecule is simulated using probabilities and statistics. Although the actual transitions are very complicated, shown in figure 1.13, in this simulation it is simplified to the transitions shown in figure 3.2. The main differences between these two are that the triplet state, T_1 has been omitted. When a fluorophore is 'on' it is only transitioning between S_0 and S_1. As before it is 'off' when it is in its charge transfer state (CTS), and bleached when in its bleached state. The probabilities are altered so that this follows the actual process as close as possible. The transition probabilities used are:

$$\begin{array}{lc} \text{from}\backslash\text{to} & \begin{array}{ccc} \text{on} & \text{off} & \text{bleached} \end{array} \\ \begin{array}{l} \text{on} \\ \text{off} \\ \text{bleached} \end{array} & \begin{pmatrix} 0.9 & 0.1 & 0 \\ 0.00999910 & 0.99 & 0.00000090 \\ 0 & 0 & 1 \end{pmatrix}. \end{array}$$

These chosen numbers are explained in section 3.6.

Using this data and the frame rate, the fraction of the frame the fluorophore is emitting for is determined. This is calculated for the entire duration of the imagery. For each frame and fluorophore this fraction is then converted into a photon number using statistics. The statistics chosen for this process are described in more detail in section 3.6.

The fluorophores are simulated as being 1nm^2 in size and their position can be chosen either manually or randomly. These positions are stored so that that the ground truth is always known. This ground truth matrix has elements that represent a nm^2 area. If a molecule is 'on' then the corresponding element is given the number of photons this fluorophore emits in this frame. This is now a simulated sample plane, detailing the locations of the fluorophores and how many photons they

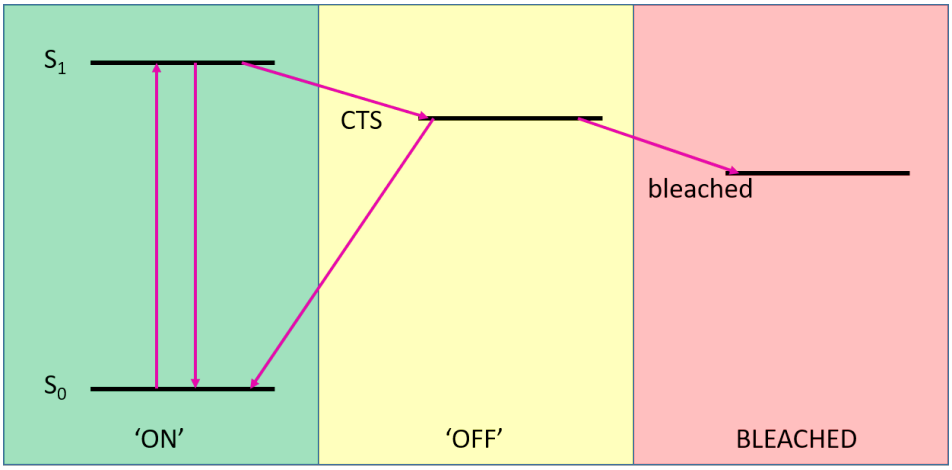


Figure 3.2: A simplified Jablonski diagram showing the transitions used in this simulation.

have emitted, without accounting for any effects that would arise from diffraction, background noise, or imaging with a microscope.

The next stage of the simulation software is to apply transformations to the ground truth matrix which would replicate the process of imaging with a microscope. The ground truth matrix is Fourier transformed to put it into reciprocal space, and then a mask is applied to the resulting matrix which removed high frequency components. The mask is chosen to replicate the theoretical point spread function (PSF) of the imaging system and can be altered if the resultant microscope has a smaller or larger PSF. The filtered matrix is then inversely Fourier transformed back into real space. This process simulates the action of lenses in the theoretical optical system, which act as a low frequency filter. This updated ground truth matrix is now effectively a simulated image where the emission of photons from a fluorophores have been spread over a group of pixels; the PSF.

The pixels of the camera are larger than $1nm^2$ in size and therefore the data is binned in space. This can be changed to be both camera and microscope specific, so that it includes both the magnification of the microscope and the size of the pixels in the camera. For example, if the camera pixels are $10\mu m^2$ ($10000nm^2$), and the microscope has a magnification of 100, the effective size of the pixels are $100nm^2$. Therefore the 'real space' pixels are binned into 100 by 100 'camera space' pixels.

This image is now a noise free version of the image received by the camera. It consists of pixels that are the same size as real data, and contains only photons from the fluorophores themselves, spread out

into the PSF of the microscope. Noise is now added to the image. This noise can be made camera specific by collecting data from the camera when there is no sample or light source (dark data). It can also be simulated using information found in the literature.

A flow chart of how both the real and simulated data are created is shown in section 3.7. The main differences between these two are discussed in section 3.8.

3.6 CHOSEN STATISTICS

The chosen probabilities were used to simulate AF 647. Properties of this fluorophore can be seen in Dempsey *et al.*'s paper [39].

Initially the chosen probabilities were:

$$\begin{array}{lc} \text{from}\backslash\text{to} & \begin{array}{ccc} \text{on} & \text{off} & \text{bleached} \end{array} \\ \begin{array}{l} \text{on} \\ \text{off} \\ \text{bleached} \end{array} & \begin{pmatrix} 0.9 & 0.099 & 0.001 \\ 0.05 & 0.95 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \end{array}$$

As it cannot be physically seen when a fluorophore bleaches it was taken so that the fluorophore could only enter the bleached state from the 'on' state. These values yielded some good results. However, using the Jablonski diagrams, figures 1.13 and 3.2, it can be seen that the fluorophore can only bleach from its 'off' state in both the actual and simplified versions. Due to this, the probabilities were altered to the following:

$$\begin{array}{lc} \text{from}\backslash\text{to} & \begin{array}{ccc} \text{on} & \text{off} & \text{bleached} \end{array} \\ \begin{array}{l} \text{on} \\ \text{off} \\ \text{bleached} \end{array} & \begin{pmatrix} 0.9 & 0.1 & 0 \\ 0.00999910 & 0.99 & 0.00000090 \\ 0 & 0 & 1 \end{pmatrix}. \end{array}$$

The value 0.9 was chosen for the on-on transition. This means that fluorophores are 'on' they are emitting, on average, for nine 'sections' of a frame which would be around 9ms. A lot of the fluorophores will emit for a lot less time that this. Fluorophores that are only 'on' for one 'section' will probably be too dim to be detected.

The value for the on-off transition was calculated using: $1 -$ on-on transition. This is because it cannot go from 'on' to bleached and the values in each row must add up to one as they are probabilities.

The equilibrium on-off duty cycle for AF 647 is 0.0012 [39]. This is the fraction of time spent in the on state. Therefore the fraction of time

spent in the off state is 0.9988. Therefore the off to off transition was kept very high. This also keeps the number of 'on' fluorophores sparse in each frame as the majority of the pixels will be 'off'.

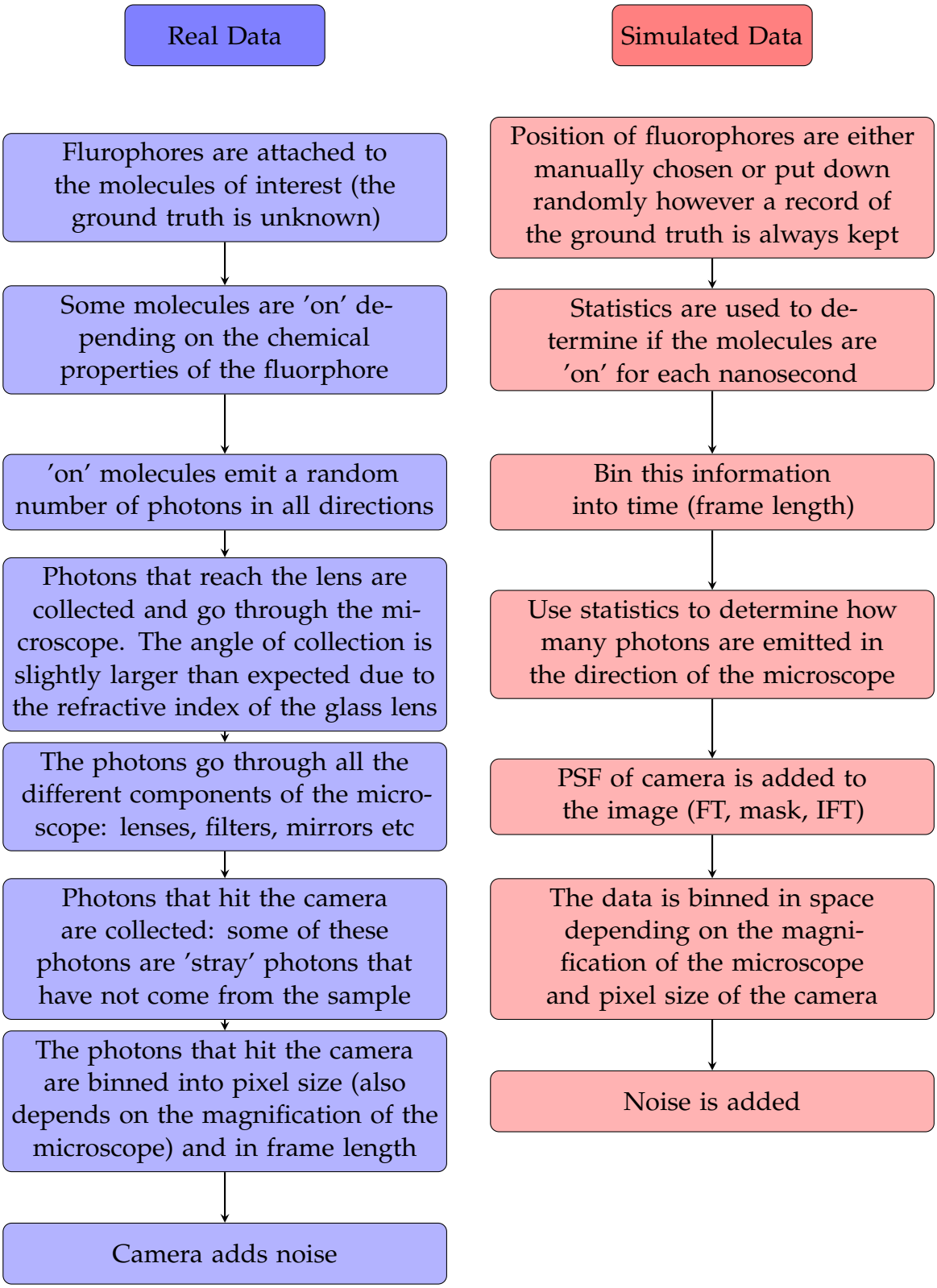
The off-bleach transition was kept very low at 0.00000090. The survival fraction for AF after 400s of illumination is 0.73 [39]. This requires a very low value for the off-bleach transition. Stacks of images were created for lengths of 400s (40,000 images). The off-bleach transition was altered until the percentage of fluorophores that had bleached was, on average, 27%.

The on-bleach transition is zero as there is no transition available for a molecule to go from an 'on' state to a bleached state. The bleach-off and bleach-on transitions are also zero as it is not possible for a fluorophore to make these transitions. The bleach-bleach transition is one as once the fluorophore has entered that state it is unable to turn back 'on' or go into the 'off' state.

The average number of photons that are detected from each AF 647 is described by an exponential fit with a mean value of 5,202 [39]. The number of photons emitted from each 'on' fluorophore was chosen randomly from an Poisson distribution, however it did not have a mean value of 5,202. A single image was created with one single 'on' fluorophore and no noise. The number of photons that the fluorophore emitted was altered until, when the final image was summed across all pixels, a mean value of 5,202 was found. This value was 6019. It should be noted that the simulation only produces the number of photons that are emitted towards the microscope.

3.7 FLOWCHART

The following flowchart shows the main steps of how both real data and the simulated data are created.


Real Data
Flurophores are attached to the molecules of interest (the ground truth is unknown)
Some molecules are 'on' depending on the chemical properties of the fluorphore
'on' molecules emit a random number of photons in all directions
Photons that reach the lens are collected and go through the microscope. The angle of collection is slightly larger than expected due to the refractive index of the glass lens
The photons go through all the different components of the microscope: lenses, filters, mirrors etc
Photons that hit the camera are collected: some of these photons are 'stray' photons that have not come from the sample
The photons that hit the camera are binned into pixel size (also depends on the magnification of the microscope) and in frame length
Camera adds noise
Simulated Data
Position of fluorophores are either manually chosen or put down randomly however a record of the ground truth is always kept
Statistics are used to determine if the molecules are 'on' for each nanosecond
Bin this information into time (frame length)
Use statistics to determine how many photons are emitted in the direction of the microscope
PSF of camera is added to the image (FT, mask, IFT)
The data is binned in space depending on the magnification of the microscope and pixel size of the camera
Noise is added

3.8 DIFFERENCES

There are several differences between the production of the simulation images and the real data. These include: the orders that different steps are completed in; the transitions that occur; and that the simulations are done in discrete time steps whereas the real data production occurs in real, continuous time.

In both the simulated and the real images, the fluorophores positions are chosen initially. In real data these are attached to specific molecules/ structures of interest, whereas in the simulations they are placed randomly. These can be placed specifically in certain places if desired, however there are many molecules in real data and choosing all these places manually could end up being very time consuming. Specific regions can be chosen where the molecules can be placed, for example away from the edges of the simulated images.

In real data, some of the fluorophores may never turn 'on' and emit any photons. In the simulations every fluorophore starts 'on' and then starts to photoblink from here, although there is a probability they will bleach straight away. In these simulations the first few images are often ignored as it is not physical to have all of the molecules 'on'.

When a fluorophore is 'on' it emits photons. The number of photons detected for each event comes from an exponential distribution. In real data it is possible that photons coming from each individual fluorophore come from different distributions due to depth in the sample or polarisation etc. This is not accounted for in the simulations and currently the simulations only create 2D data. It is possible in the future, that this could be extended to create 3D images, for example when the microscope includes an extra lens to introduces astigmatism into the PSFs [114].

The fluorophore within the sample will have specific variations, for example a fluorophore that is deeper within a sample may emit the same amount of fluorophores as one near the top, due to scattering, refraction and absorption less photons will reach the camera. This is not accounted for within the simulations however could be included in the future.

In real data the photons go through different optical components including filters and lenses when inside the microscope. Any time that photons transition between different materials there is a probability of being reflected, and therefore there will be some losses in the system. Although the simulations account for this, it does not take every single

component separately and instead takes them all together including the QE of the camera.

In real data noise can come from a variety of different sources at different stages of the process. This is discussed more in section 3.4. In the simulated images, all noise is added at the end of the process.

When real images are taken the data is binned in both space and time at the camera. The pixel size of the camera and the magnification power of the microscope determine how the data is binned in space. In the simulations these occur at different times. The data is binned in time near the beginning of the process and the data is binned in space just before the noise is added.

There are differences in the transitions that are allowed in both the real data and simulations. The transitions that are allowed in the simulations are shown in the Jablonski diagram in figure 3.2. Real fluorophores can go through many different transitions. The main transitions are shown in figure 1.13, however this is still a simplified version of what actually happens, which is still not fully understood.

As stated before, when creating real data, this occurs in real, continuous time. During the simulations, all the transitions for the whole stack of images are created at the beginning and in discrete sections. These are then binned into bigger time frame sections.

3.9 RESULTS

The simulated images show a strong resemblance to real data both spatially and temporally. Section 3.9.1 looks at a stack of images that were created and sent through ThunderSTORM [115]. It describes the differences between the locales of the molecules that were found, and the molecules that are put down by the simulations.

Section 3.9.2 shows some simulated images and real images side be side and discusses the differences.

Section 3.9.3 looks at how the simulated images vary through time as the fluorophores blink on and off. These are compared to a time trace of real data taken with SMLM.

3.9.1 *ThunderSTORM results*

A stack of 500 images were simulated with 5000 fluorophores. These images were run through the ThunderSTORM [115] to determine how many fluorophores could be localised.

As ThunderSTORM [115] is more of a localisation algorithm rather than a counting algorithm an individual image was first looked at to see the differences in the locations of the fluorophores that were actually 'on' in the simulations and the locations that ThunderSTORM predicted. This is shown in figure 3.3 where an image from the middle of the stack is shown (image 300). Figure 3.3a shows the simulated image.

Figure 3.3b shows the actual positions of all the fluorophores. It should be noted that this contains all the fluorophores that are 'on', even if they only emit for 1ms (a tenth of the frame). Also, as the output of photons is randomised from a poisson distribution, it could be that some fluorophores emit very few photons and will still be classified as 'on'. In this frame there are 99 fluorophores 'on'.

Figure 3.3c shows the positions that ThunderSTORM [115] predicted there to be a fluorophore and figure 3.3d shows both this and the actual positions for comparison. Looking at figure 3.3d it can be seen that ThunderSTORM has failed to find several fluorophores on the edge of the image. As ThunderSTORM ignores the edges of images this was expected. There are several spots where there are fluorophores with overlapping PSF and ThunderSTORM has only fitted one fluorophore to these spots. This was also expected to occur while using this algorithm as it works better with sparse images. Finally, there are some fluorophores that are not near the edge of the simulated images that ThunderSTORM has missed entirely. These are fluorophores that are dim, and not emitting many photons. As the number of photons that each fluorophore emits is generated from an poisson function, this is to be expected.

Looking at the total number of fluorophores ThunderSTORM registers, the number is an overestimate. Due to small fluctuations in the background noise this is thought to be due to multiple counting; i.e. if a fluorophore turns back on then it is recounted. Instead the difference in the number of fluorophores simulated as 'on' in the algorithm, and the number of fluorophores localised by ThunderSTORM per image was looked at. This is shown in figure 3.4. Although the initial 200 frames show a large disagreement in the number of fluorophores, after this the difference in the number of fluorophores is very low. As each frame is 2ms, it takes around 2s for the the difference to reach an equilibrium. Looking at figure 3.4b the mode is between -4 and 7 and it looks to follow a gamma distribution shape.

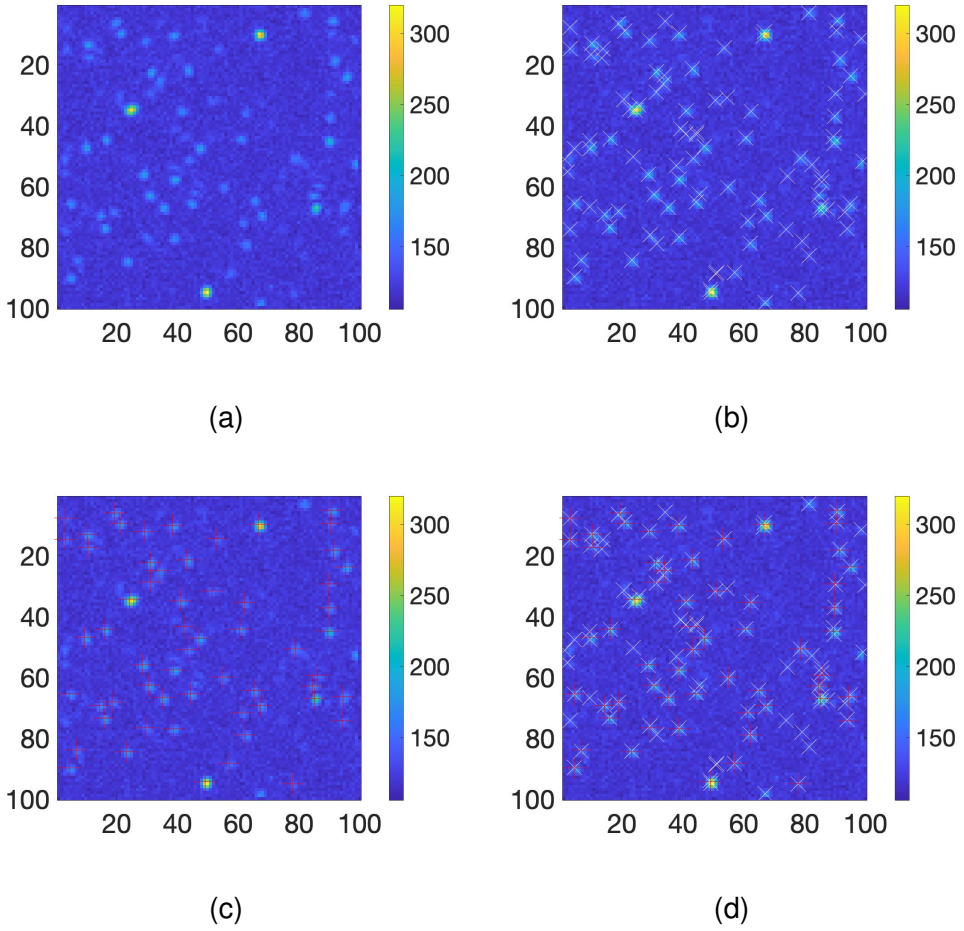
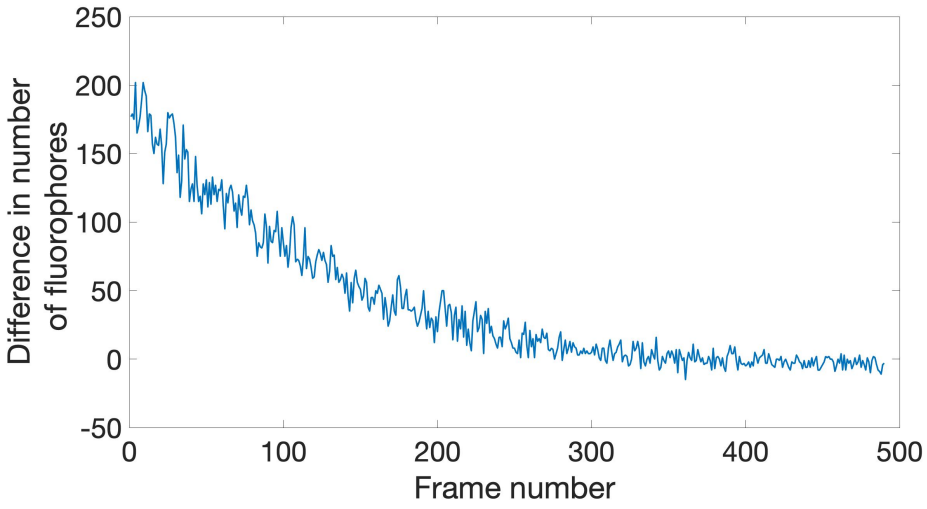


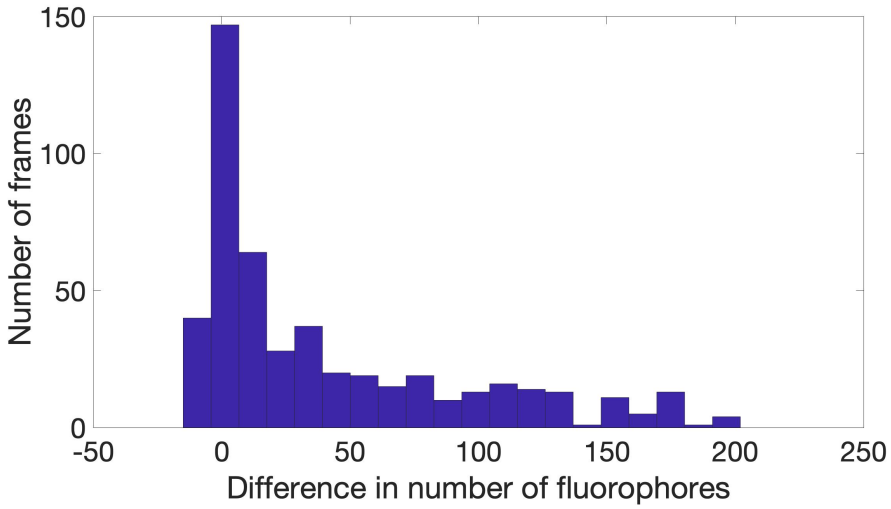
Figure 3.3: (a) An image from the middle of a stack of 500 simulated images (image 300) with a total of 5000 switching fluorophores, (b) shows the locations of all the emitting fluorophores for that timeframe. There are 99 'on' fluorophores. (c) the positions that ThunderSTORM predicted and (d) shows both the actual locations as well as the positions found by ThunderSTORM.

3.9.2 *Comparison with real images*

Figure 3.5 shows both a simulated image and a real data image taken with SMLM. It can be seen that these images have a good resemblance with similar PSFs and noise areas.



(a)



(b)

Figure 3.4: The difference between the number of fluorophores that are 'on' in the simulated images and then number of fluorophores that are localised by the ThunderSTORM [115] algorithm. Fluorophores are classed as 'on' if they are emitting for over half of the frame. (a) shows the difference through time whereas (b) shows a histogram of the whole stack.

3.9.3 *Time traces of images*

Figures 3.6 and 3.7 show a time trace through a stack of images. They both show the mean value of a 3x3 pixel section through time. For

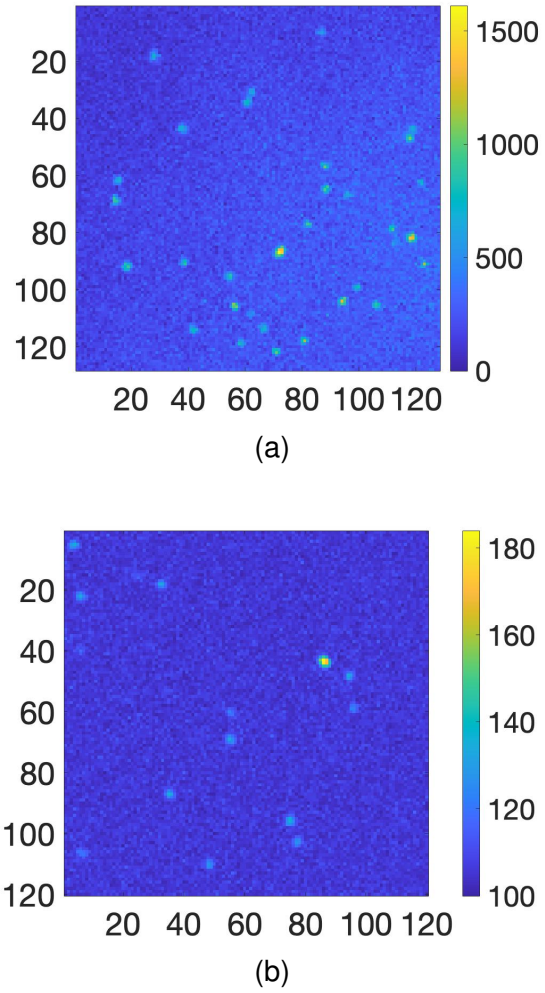


Figure 3.5: (a) An image taken using SMLM and (b) a simulated image produced with the simulation described in this chapter.

each time trace there are nine images showing the 3x3 area at that time. Figure 3.6 is a stack of 500 images that were simulated using the algorithm described above whereas figure 3.7 shows a stack of 100 images of 3T3 cells taken using SMLM.

From these two figures (figures 3.6 and 3.7) it can be seen that the time trace of the simulated images follow a similar trend to the real images. As all the fluorophores are 'on' in the first image, the first few should be discarded when analysing the stack. The noise values have a similar trend and the variance is similar.

Looking at figure 3.6, the peaks in intensity are when the fluorophore is emitting photons. The noise values appear to come from fluctuations in the noise, however a few small peaks can be seen when neighbouring

fluorophores are 'on' and the edges of the PSFs are over the pixels. This can be seen in the sixth image.

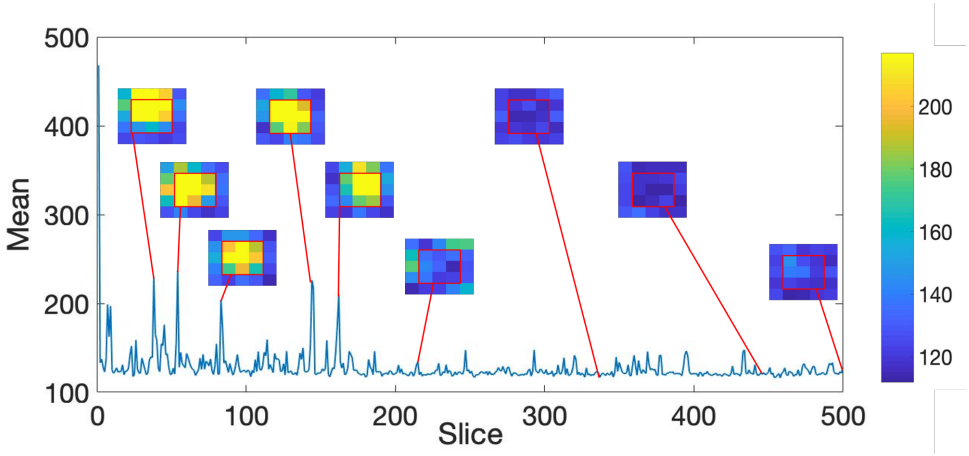


Figure 3.6: Figure shows the mean of a 3x3 area through an entire stack of 5000 simulated images. There are nine different sub-images that show the 3x3 area at that time. Each of these images has the same scale which is shown to the right of the figure.

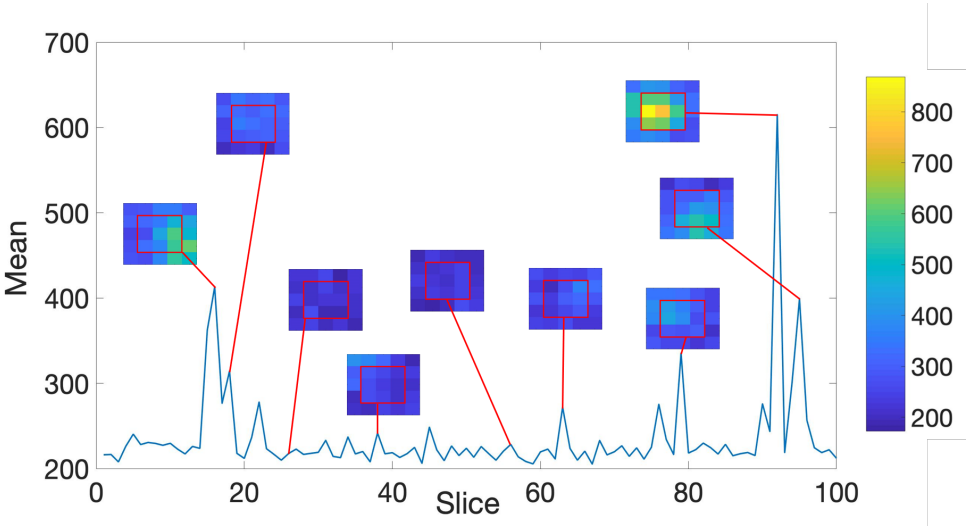


Figure 3.7: Figure shows the mean of a 3x3 area through a stack of 100 real images from 3T3 cells. There are nine different sub-images that show the 3x3 area at that time. Each of these images has the same scale which is shown to the right of the figure.

3.10 DISCUSSION

Software that successfully creates simulated images from SMLM has been developed. It is very flexible in terms of of fluorophore location and fluorophore photophysics. It currently simulates the photophysics of AF 647, however this could be adapted to include any other fluorophore with well documented emission statistics such as any in [39].

The simulated images were run through ThunderSTORM [115] as described in section 3.9.1. As ThunderSTORM is more of a localisation algorithm rather than a counting algorithm, the difference in the total number of fluorophores was expected to be different. When looking at individual images the locations were well matched as shown in figure 3.3. As was expected, ThunderSTORM failed to pick up fluorophores that were very dim. Another limitation with ThunderSTORM is that is unable to discern between fluorophores when their PSF are overlapping. This can also be seen in figure 3.3.

The difference in the number of fluorophores localised by ThunderSTORM and the number of fluorophores simulated is plotted in figure 3.4. Images at the beginning of the simulations are more densely populated and therefore there will be more overlapping PSFs. This will inhibit ThunderSTORM's ability to correctly localise the correct number of fluorophores which can be seen in the plot. After 2s (200 images) the difference in the number of fluorophores counted by ThunderSTORM was very well matched. This shows that the images are similar to real images taken with SMLM. The mode of the difference in fluorophores localised by ThunderSTORM and the number of fluorophores simulated as 'on' is between -4 and 7.

The simulated images were compared to a real stack of images. A time trace of the mean of a 3x3pixel area of the simulations and real images are shown in section 3.9.2 and figure 3.6. The fluctuations in both of these time traces are comparable showing that the simulations follow the same trends.

Within these figures (section 3.9.2 and figure 3.6) are some inlet pictures showing the pixel data at that point in time. From these it can be seen that the PSFs are also comparable showing that the simulations are a good approximation of the real data.

All of these factors show that the simulated images are comparable to the real images. They have similar time traces, PSFs and noise.

When photons are emitted from fluorophore they leave in all directions and start to diffract. These photons are more likely to leave in

certain directions depending on the polarisation of the fluorophore, however this orientation will vary for each fluorophore that is fixed and if the fluorophore is not fixed then it will fluctuate with time. The number of collected photons will also vary diversify depending on how deep the fluorophores are within the sample. The deeper the fluorophores are, the more likely the photons are going to be diffracted, absorbed or refracted before reaching the coverslip/ lens. These factors are not included in the simulation as the photon number for every fluorophore is chosen from the same distribution. This could be incorporated into the code in the future so that fluorophores at different levels/ orientations have photon numbers coming from different distributions.

Currently the algorithm produces only 2-dimensional SMLM images. There are many different ways of expanding SMLM into 3-dimensions. A lot of these work by changing the PSF of the fluorophores. One such method introduces a cylindrical lens into the optical path causing optical astigmatism [114]. When fluorophores are are on either side of the focal plane their PSF changes from circular to elliptical. It widens in the x-direction when it is above the focal plane and in the y-direction when below. This could be incorporated into the simulations in the future.

3.11 THE ALGORITHM

The code that was written and used in this chapter can be found in appendix B.

The code that simulates individual images was adapted from software written by Ashley Cadby. The rest of the code was written by myself.

REFERENCES

[1] C. SINGER, 'The Dawn of Microscopical Discovery', Journal of the Royal Microscopical Society **35**, 317–340 (1915).

[2] S. Bradbury, *An Introduction to the Optical Microscope* (Oxford University Press, 1989), p. 5.

[3] G. Sines and Y. A. Sakellarakis, 'Lenses in Antiquity', American Journal of Archaeology **91**, 191 (1987).

[4] S. Bradbury, *The Evolution of the Microscope* (Pergamon Press, 1967).

[5] H. Gest, 'The discovery of microorganisms by Robert Hooke and Antoni van Leeuwenhoek, Fellows of The Royal Society', Notes and Records of the Royal Society **58**, 187–201 (2004).

[6] C. S. Ball, 'The early history of the compound microscope', Bios **37**, 51–60 (1966).

[7] D. Bardell, 'The First Record of Microscopic Observations', BioScience **33**, 36–38 (1983).

[8] R. Hooke, *Micrographia* (Royal Society, 1665).

[9] K. B. Wolf and G. Krötzsch, 'Geometry and dynamics in refracting systems', European Journal of Physics **16**, 14–20 (1995).

[10] M. Hausner, *Optics Inspections and Tests: A Guide for Optics Inspectors and Designers* (2017), pp. 7–32.

[11] A. Kwan, J. Dudley and E. Lantz, 'Who really discovered snell's law?', Physics World **15**, 64 (2002).

[12] R. D. Fiete, 'Optics', in *Modeling the imaging chain of digital cameras* (SPIE Press, 2010), pp. 49–72.

[13] L. Davis, C, *Thin Lenses*, http://www.physics.louisville.edu/cldavis/phys299/notes/lo_lense Accessed: 2020-14-04.

[14] S. J. Ling, J. Sanny and B. Moebs, *4.1: single-slit diffraction*, https://phys.libretexts.org/?title=T _Optics_and_Modern_Physics_(OpenStax)/4:_Diffraction/4.1:_Single-Slit_Diffraction%0A, Visited on 2018-05-23.

[15] G. B. Airy, 'On the Diffraction of an Object-glass with Circular Aperture', Transactions of the Cambridge Philosophical Society **5**, 283–291 (1835).

[16] *Greenfluorescentblog*, `https://greenfluorescentblog.wordpress.com/2012/04/01/numerical-aperture-and-resolution/`, Accessed: 2018-01-29.

[17]E. Abbe, 'Beitrage zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung', Arch. Mikroscop Anat **9**, 413–420 (1873).

[18]T. S. Tkaczyk, 'Numerical Aperture', in *Field guide to microscopy* (2010), p. 38.

[19]B. R. Masters, 'Optical Resolution and Resolving Power : What It Is , How to Measure It , and What Limits It', in *Confocal microscopy and multiphoton excitation microscopy: the genesis of live cell imaging* (SPIE Press, 2006), pp. 49–54.

[20]N. Foundation, *The nobel prize in chemistry*, 2014.

[21]S. Balaiya, R. K. Murthy, V. S. Brar and K. V. Chalam, 'Evaluation of ultraviolet light toxicity on cultured retinal pigment epithelial and retinal ganglion cells', Clinical Ophthalmology, 33–39 (2010).

[22]M. Abramowitz and M. W. Davidson, *Anatomy of a Microscope - Numerical Aperture and Resolution*, Visited on 2020-04-16.

[23]N. Foundation, *The nobel prize in physics*, 1929.

[24]W. J. Croft, *Under the microscope [electronic resource] : a brief history of microscopy* (World Scientific Publishing Co. Pte. Ltd., 2006), pp. 57–72.

[25]R. Erni, M. D. Rossell, C. Kisielowski and U. Dahmen, 'Atomic Resolution Imaging with a sub-50 pm Electron Probe', Physical Review Letters **102**, 96–101 (2009).

[26]N. Foundation, *The nobel prize in physics*, 1986.

[27]J. Ayache, L. Beaunier, J. Boumendil, G. Ehret and D. Laub, 'Artifacts in Transmission Electron Microscopy', in *Sample preparation handbook for transmission electron microscopy* (Springer, New York, NY, 2010), pp. 125–170.

[28]J. G. Mcnally, T. Karpova, J. Cooper and J. A. Conchello, 'Three-Dimensional Imaging by Deconvolution Microscopy', Methods **19**, 373–385 (1999).

[29]G. Sparacino, G. De Nicolao, G. Pillonetto and C. Cobelli, 'Deconvolution', in *Modelling methodology for physiology and medicine* (Elsevier, 2014), pp. 45–68.

[30]M. Minsky, 'Memoir on inventing the confocal scanning microscope', Scanning **10**, 128–138 (1988).

[31]M. Minsky, *Microscopy Apparatus*, 1961.

[32]M. Renz, 'Fluorescence microscopy-A historical and technical perspective', Cytometry Part A **83**, 767–779 (2013).

[33]H. Helmholtz, 'Die theoretischen Grenzen für die Leistungsfähigkeit der Mikroskope.', Annalen der Physik, 557–584 (1874).

[34]M. Muyskens and Ed Vitz, 'The Fluorescence of Lignum nephriticum: A Flash Back to the Past and a Simple Demonstration of Natural Substance Fluorescence', Journal of Chemical Education **83**, 765 (2006).

[35]G. G. Stokes, 'On the Change of Refrangibility of Light', Philosophical Transactions of the Royal Society of London **142**, 463–562 (1852).

[36]G. G. Stokes, 'On the Change of Refrangibility of Light. No. II', Philosophical Transactions of the Royal Society of London **143**, 385–396 (1853).

[37]L. J. Kricka and P. Fortina, 'Analytical ancestry: "firsts" in fluorescent labeling of nucleosides, nucleotides, and nucleic acids', Clinical Chemistry **55**, 670–683 (2009).

[38]D. J. S. Birch, Y. Chen and O. J. Rolinski, 'Fluorescence', in *Photonics, volume 4 : biomedical photonics, spectroscopy, and microscopy* (John Wiley & Sons, Incorporated, 2015), pp. 1–58.

[39]G. T. Dempsey, J. C. Vaughan, K. H. Chen, M. Bates and X. Zhuang, 'Evaluation of fluorophores for optimal performance in localization-based super-resolution imaging', Nature Methods **8**, 1027–1036 (2011).

[40]R. Smallman and A. Ngan, 'Characterization and Analysis', in *Modern physical metallurgy* (Elsevier, 2014), pp. 159–250.

[41]H. Siedentopf and R. Zsigmondy, 'Uber Sichtbarmachung und Größenbestimmung ultramikoskopischer Teilchen, mit besonderer Anwendung auf Goldrubingläser', Annalen der Physik **315**, 1–39 (1902).

[42]D. Axelrod, 'Cell-substrate contacts illuminated by total internal reflection fluorescence.', The Journal of Cell Biology **89**, 141–145 (1981).

[43]Y. Markaki and H. Harz, eds., *Light Microscopy*, Vol. 1563, Methods in Molecular Biology (Springer New York, New York, NY, 2017).

[44]B. O. Leung and K. C. Chou, 'Review of super-resolution fluorescence microscopy for biology.', Applied spectroscopy **65**, 967–80 (2011).

[45]S. W. Hell and J. Wichmann, 'Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy', Optics Letters **19**, 780 (1994).

[46]R. C. Dunn, 'Near-field scanning optical microscopy.', Chemical reviews **99**, 2891–928 (1999).

[47]M. G. L. Gustafsson, 'Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution', Proceedings of the National Academy of Sciences **102**, 13081–13086 (2005).

[48]E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz and H. F. Hess, 'Imaging intracellular fluorescent proteins at nanometer resolution.', Science (New York, N.Y.) **313**, 1642–5 (2006).

[49]M. Rust, M. Bates and X. Zhuang, 'Stochastic optical reconstruction microscopy (STORM) provides sub-diffraction-limit image resolution', Nature methods **3**, 793–795 (2006).

[50]S. T. Hess, T. P. K. Girirajan and M. D. Mason, 'Ultra-high resolution imaging by fluorescence photoactivation localization microscopy.', Biophysical journal **91**, 4258–72 (2006).

[51]T. Dertinger, R. Colyer, G. Iyer, S. Weiss and J. Enderlein, 'Fast, background-free, 3D super-resolution optical fluctuation imaging (SOFI).', Proceedings of the National Academy of Sciences of the United States of America **106**, 22287–22292 (2009).

[52]B. Huang, M. Bates and X. Zhuang, 'Super-Resolution Fluorescence Microscopy', Annual Review of Biochemistry **78**, 993–1016 (2009).

[53]D. Wildanger et al., 'Solid immersion facilitates fluorescence microscopy with nanometer resolution and sub-Ångström emitter localization', Advanced Materials **24**, 309–313 (2012).

[54]R. Heintzmann, T. M. Jovin and C. Cremer, 'Saturated patterned excitation microscopy—a concept for optical resolution improvement', Optical Society of America **19**, 1599–1609 (2002).

[55]E. H. Rego, L. Shao, J. J. Macklin, L. Winoto, G. A. Johansson, N. Kamps-Hughes, M. W. Davidson and M. G. Gustafsson, 'Nonlinear structured-illumination microscopy with a photoswitchable protein reveals cellular structures at 50-nm resolution', Proceedings of the National Academy of Sciences of the United States of America **109**, 13–15 (2012).

[56]E. Betzig, 'Proposed method for molecular optical imaging', Optics Letters **20**, 237 (1995).

[57]W. Moerner and L. Kador, 'Optical detection and spectroscopy of single molecules in a solid', Physical Review Letters **62**, 2535–2538 (1989).

[58]M. Orrit and J. Bernard, 'Single pentacene molecules detected by fluorescence excitation in a p-terphenyl crystal', Physical Review Letters **65**, 2716–2719 (1990).

[59]W. P. Ambrose and W. E. Moerner, 'Fluorescence spectroscopy and spectral diffusion of single impurity molecules in a crystal', Nature **349**, 225–227 (1991).

[60]R. M. Dickson, A. B. Cubitt, R. Y. Tsien and W. E. Moerner, 'On / off blinking and switching behaviour of single molecules of green fluorescent protein', **388**, 355–358 (1997).

[61]M. Heilemann, S. van de Linde, A. Mukherjee and M. Sauer, 'Super-resolution imaging with small organic fluorophores.', Angewandte Chemie (International ed. in English) **48**, 6903–8 (2009).

[62]M. Heilemann, S. van de Linde, M. Schüttpelz, R. Kasper, B. Seefeldt, A. Mukherjee, P. Tinnefeld and M. Sauer, 'Subdiffraction-resolution fluorescence imaging with conventional fluorescent probes.', Angewandte Chemie (International ed. in English) **47**, 6172–6 (2008).

[63]L. Stryer and R. P. Haugland, 'Energy transfer: a spectroscopic ruler.', Proceedings of the National Academy of Sciences **58**, 719–726 (1967).

[64]*Single-molecule localization microscopy: directory of smlm software*, http://bigwww.epfl.ch/smlm, Visited on 2020-06-19, 2017.

[65]P. D. Simonson, E. Rothenberg and P. R. Selvin, 'Single-molecule-based super-resolution images in the presence of multiple fluorophores.', Nano letters **11**, 5090–6 (2011).

[66]S. A. Jones, S.-H. Shim, J. He and X. Zhuang, 'Fast, three-dimensional super-resolution imaging of live cells', Nature Methods **8**, 499–505 (2011).

[67]A. Small and S. Stahlheber, 'Fluorophore localization algorithms for super-resolution microscopy.', Nature methods **11**, 267–79 (2014).

[68]A. V. Abraham, S. Ram, J. Chao, E. S. Ward and R. J. Ober, 'Quantitative study of single molecule location estimation techniques', Optics Express **17**, 23352 (2009).

[69]R. E. Thompson, D. R. Larson and W. W. Webb, 'Precise nanometer localization analysis for individual fluorescent probes.', Biophysical journal **82**, 2775–2783 (2002).

[70]S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice Hall, 1993).

[71]R. Henriques, M. Lelek, E. F. Fornasiero, F. Valtorta, C. Zimmer and M. M. Mhlanga, 'QuickPALM: 3D real-time photoactivation nanoscopy image processing in ImageJ.', Nature methods **7**, 339–40 (2010).

[72]R. Parthasarathy, 'Rapid, accurate particle tracking by calculation of radial symmetry centers', Nature Methods **9**, 724–726 (2012).

[73]A. J. Berglund, M. D. McMahon, J. J. McClelland and J. A. Liddle, 'Fast, bias-free algorithm for tracking single particles with variable size and shape', Optics Express **16**, 14064 (2008).

[74]H. Ma, F. Long, S. Zeng and Z.-L. Huang, 'Fast and precise algorithm based on maximum radial symmetry for single molecule localization', Optics Letters **37**, 2481 (2012).

[75]T. Quan, H. Zhu, X. Liu, Y. Liu, J. Ding, S. Zeng and Z.-L. Huang, 'High-density localization of active molecules using Structured Sparse Model and Bayesian Information Criterion', Optics Express **19**, 16963 (2011).

[76]Y. Wang, T. Quan, S. Zeng and Z.-L. Huang, 'PALMER: a method capable of parallel localization of multiple emitters for high-density localization microscopy', Optics Express **20**, 16039 (2012).

[77]F. Huang, S. L. Schwartz, J. M. Byars and K. a. Lidke, 'Simultaneous multiple-emitter fitting for single molecule super-resolution imaging.', Biomedical optics express **2**, 1377–1393 (2011).

[78]F. Huang et al., 'Video-rate nanoscopy using sCMOS camera-specific single-molecule localization algorithms.', Nature methods **10**, 653–8 (2013).

[79]S. J. Holden, S. Uphoff and A. N. Kapanidis, 'DAOSTORM: an algorithm for high-density super-resolution microscopy', Nature methods **8**, 279–280 (2011).

[80]D. R. Tobergte and S. Curtis, 'Daophot: a Computer Program for Crowded-Field Stellar Photometry (Psf Photometry)', Journal of Chemical Information and Modeling **53**, 1689–1699 (2013).

[81]L. B. Lucy, 'An iterative technique for the rectification of observed distributions', Astronomical Journal **79**, 745–754 (1974).

[82]W. Richardson, 'Bayesian-based iterative method of image restoration', Optical Society of America **62**, 55–59 (1972).

[83]L. Zhu, W. Zhang, D. Elnatan and B. Huang, 'Faster STORM using compressed sensing', Nature Methods **9**, 721–723 (2012).

[84]S. Cox, E. Rosten, J. Monypenny, T. Jovanovic-Talisman, D. T. Burnette, J. Lippincott-Schwartz, G. E. Jones and R. Heintzmann, 'Bayesian localization microscopy reveals nanoscale podosome dynamics.', Nature methods **9**, 195–200 (2012).

[85]S. Wolter, U. Endesfelder, S. van de Linde, M. Heilemann and M. Sauer, 'Measuring localization performance of super-resolution algorithms on very active samples.', Optics express **19**, 7020–33 (2011).

[86]S. Wang, J. R. Moffitt, G. T. Dempsey, X. S. Xie and X. Zhuang, 'Characterization and development of photoactivatable fluorescent proteins for single-molecule-based superresolution imaging', Proceedings of the National Academy of Sciences of the United States of America **111**, 8452–8457 (2014).

[87]N. Durisic, L. Laparra-Cuervo, Á. Sandoval-Álvarez, J. S. Borbely and M. Lakadamyali, 'Single-molecule evaluation of fluorescent protein photoactivation efficiency using an in vivo nanotemplate', Nature Methods **11**, 156–162 (2014).

[88]P. J. Green, 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', Biometrika **82**, 711 (1995).

[89]H. Jeffreys, *Theory of probability*, 3rd (Press, Oxford: University, 1961).

[90]L. Savage, *The foundations of Statistics* (New York: Wiley, 1972).

[91]B. De Finetti, *Theory of probability* (London: Wiley, 1974).

[92]R. Christensen, T. E. Hanson, W. Johnson, Branscum and Adam, *Bayesian Ideas and Data Analysis : An Introduction for Scientists and Statisticians*, 1st ed. (Taylor & Francis Group, 2010).

[93]S. Chib and E. Greenberg, 'Understanding the Metropolis-Hastings Algorithm', The American Statistician **49**, 327–335 (1995).

[94]M. Jesper, ed., *Spactial Statistics and Computational Methods* (Springer, 2003), pp. 1–13.

[95]K. I. Mortensen, L. S. Churchman, J. A. Spudich and H. Flyvbjerg, 'Optimized localization analysis for single-molecule tracking and super-resolution microscopy', Nature Methods **7**, 377–381 (2010).

[96]J. R. Janesick, *Photon Transfer* (SPIE.Digital Library, 2007).

[97]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, `http://www.deeplearningbook.org` (MIT Press, 2016).

[98]S. Mor-Yosef, A. Samueloff, B. Modan, D. Navot and J. G. Schenker, 'Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study.', Obstetrics and gynecology **75**, 944–7 (1990).

[99] V. Metsis, I. Androutsopoulos and G. Paliouras, 'Spam Filtering with Naive Bayes – Which Naive Bayes?', In CEAS (2006).

[100] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, http://www.deeplearningbook.org (MIT Press, 2016).

[101] *Neural Network Programming - Deep Learning with PyTorch*, https://deeplizard.com/learn/ Accessed: 2020-07-08.

[102] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', Nature **323**, 533–536 (1986).

[103] M. Torrisi, G. Pollastri and Q. Le, 'Deep learning methods in protein structure prediction', Computational and Structural Biotechnology Journal **18**, 1301–1310 (2020).

[104] *Single-molecule localization microscopy: collection of reference datasets*, http://bigwww.epfl.ch/smlm/datasets/index.html, Visited on 2020-06-19, 2017.

[105] H. Li and J. C. Vaughan, 'Switchable Fluorophores for Single-Molecule Localization Microscopy', Chemical Reviews **118**, 9412–9454 (2018).

[106] B. A. Griffin, S. R. Adams, R. Y. Tsien, B. A. Griffin, S. R. Adams and R. Y. Tsient, 'Specific Covalent Labeling of Recombinant Protein Molecules Inside Live Cells Published by : American Association for the Advancement of Science Stable URL : http://www.jstor.org/stable/2896025 REFERENCES Linked references are available on JSTOR for this', Science **281**, 269–272 (1998).

[107] L. W. Miller, Y. Cai, M. P. Sheetz and V. W. Cornish, 'In vivo protein labeling with trimethoprim conjugates: A flexible chemical tag', Nature Methods **2**, 255–257 (2005).

[108] A. Keppler, S. Gendreizig, T. Gronemeyer, H. Pick, H. Vogel and K. Johnsson, 'A general method for the covalent labeling of fusion proteins with small molecules in vivo', Nature Biotechnology **21**, 86–89 (2003).

[109] A. Gautier, A. Juillerat, C. Heinis, I. R. Corrêa, M. Kindermann, F. Beaufils and K. Johnsson, 'An Engineered Protein Tag for Multiprotein Labeling in Living Cells', Chemistry and Biology **15**, 128–136 (2008).

[110] G. V. Los et al., 'HaloTag: A novel protein labeling technology for cell imaging and protein analysis', ACS Chemical Biology **3**, 373–382 (2008).

[111] I. Y. Iourov, *Fluorescence In Situ Hybridization (FISH)*, edited by T. Liehr, Springer Protocols Handbooks (Springer Berlin Heidelberg, Berlin, Heidelberg, 2017), pp. 17–25.

[112]M. E. Brezinski, *Noise and system performance with td-oct and sd-oct* (Academic Press, Amsterdam ; 2006), pp. 175–195.

[113]T. Bushnell, *What is autofluorescence*, https://expert.cheekyscientist.com/what-is-autofluorescence/, Visited on 2020-06-09.

[114]B. Huang, W. Wang, M. Bates and X. Zhuang, 'Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy.', Science (New York, N.Y.) **319**, 810–3 (2008).

[115]M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych and G. M. Hagen, 'ThunderSTORM: A comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging', Bioinformatics **30**, 2389–2390 (2014).

[116]B. A. Cipra, 'An Introduction to the Ising Model', The American Mathematical Monthly **94**, 937–959 (1987).

[117]I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, 'Chapter 9 - Probabilistic methods', in *Data mining practical machine learning tools and techniques*, edited by I. H. Witten, E. Frank, M. A. Hall and C. J. B. T. D. M. (E. Pal (Morgan Kaufmann, 2017), pp. 335–416.

[118]B. A. Landman, I. Lyu, Y. Huo and A. J. Asman, 'Chapter 6 - Multiatlas segmentation', in *Handbook of medical image computing and computer assisted intervention*, edited by S. K. Zhou, D. Rueckert, G. B. T. H. o. M. I. C. Fichtinger and C. A. Intervention (Academic Press, 2020), pp. 137–164.

[119]Orchard, Peter, *Markov random field optimisation*, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0809/ORCHARD/, Accessed 2014-08-28.

[120]Q. Jackson and D. Landgrebe, 'Adaptive Bayesian contextual classification based on Markov random fields', Geoscience and Remote Sensing, ... **40**, 2454–2463 (2002).

[121]A. Sonnleitner, G. Schütz and T. Schmidt, 'Free Brownian Motion of Individual Lipid Molecules in Biomembranes', Biophysical Journal **77**, 2638–2642 (1999).

[122]M. K. Cheezum, W. F. Walker and W. H. Guilford, 'Quantitative comparison of algorithms for tracking single fluorescent particles', Biophysical Journal **81**, 2378–2388 (2001).

[123]U. Kubitscheck, O. Kückmann, T. Kues and R. Peters, 'Imaging and Tracking of Single GFP Molecules in Solution', Biophyiscal Journal **78**, 2170–2179 (2000).

[124]*Markov random fields*, https://www2.isye.gatech.edu/isyebayes/bank/handout16.pdf, Accessed August 11, 2022.

[125] *Classification Learner*, https://uk.mathworks.com/help/stats/classificationlearner-app.html, Accessed: 2020-08/-8.

[126] M. J. Rust, M. Bates and X. Zhuang, 'Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM).', Nature methods **3**, 793–5 (2006).

[127] R. P. Nieuwenhuizen, M. Bates, A. Szymborska, K. A. Lidke, B. Rieger and S. Stallinga, 'Quantitative localization microscopy: Effects of photophysics and labeling stoichiometry', PLoS ONE **10**, 1–18 (2015).

[128] D. Gross and W. Webb, 'Molecular counting of low-density lipoprotein particles as individuals and small clusters on cell surfaces', Biophysical Journal **49**, 901–911 (1986).

[129] B. M. Burton, K. A. Marquis, N. L. Sullivan, T. A. Rapoport and D. Z. Rudner, 'The ATPase SpoIIIE Transports DNA across Fused Septal Membranes during Sporulation in Bacillus subtilis', Cell **131**, 1301–1312 (2007).

[130] M. C. Leake, J. H. Chandler, G. H. Wadhams, F. Bai, R. M. Berry and J. P. Armitage, 'Stoichiometry and turnover in single, functioning membrane protein complexes', Nature **443**, 355–358 (2006).

[131] S. K. Das, M. Darshi, S. Cheley, M. I. Wallace and H. Bayley, 'Membrane protein stoichiometry determined from the step-wise photobleaching of dye-labelled subunits', ChemBioChem **8**, 994–999 (2007).

[132] K. Tsekouras, T. C. Custer, H. Jashnsaz, N. G. Walter and S. Pressé, 'A novel method to accurately locate and count large numbers of steps by photobleaching', Molecular Biology of the Cell **27**, edited by D. Lidke, 3601–3615 (2016).

[133] A. Lee, K. Tsekouras, C. Calderon, C. Bustamante and S. Pressé, 'Unraveling the Thousand Word Picture: An Introduction to Super-Resolution Data Analysis', Chemical Reviews **117**, 7276–7330 (2017).

[134] E. Alpaydin, 'Introduction', in *Introduction to machine learning* (2010), pp. 1–19.

[135] C. Campbell and Y. Ying, 'Learning with Support Vector Machines', Synthesis Lectures on Artificial Intelligence and Machine Learning **5**, 1–95 (2011).

[136] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (Cambridge University Press, Mar. 2000).

[137]V. Hodge and J. Austin, 'A Survey of Outlier Detection Methodologies', Artificial Intelligence Review **22**, 85–126 (2004).

[138]M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks', Neural Networks **106**, 249–259 (2018).

[139]L. Deng, 'The mnist database of handwritten digit images for machine learning research [best of the web]', IEEE Signal Processing Magazine **29**, 141–142 (2012).

[140]S. L. Smith, P.-j. Kindermans, C. Ying, Q. V. Le and G. Brain, 'D on ' T D Ecay the L Earning R Ate , I Ncrease the B Atch S Ize', 1–11 (2018).

[141]*Cross entropy loss*, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, Accessed April 02, 2021.

[142]Z. Zhang and M. R. Sabuncu, 'Generalized cross entropy loss for training deep neural networks with noisy labels', Advances in Neural Information Processing Systems **2018-December**, 8778–8788 (2018).

[143]*Cross-entropy loss function*, https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e, Accessed April 02, 2021.

4

MIXTURE MODEL AND THE MODIFIED ISING MODEL

4.1 SUMMARY

In this chapter a two-part algorithm which successfully denoises images from Single Molecule Localisation Microscopy (SMLM) has been developed in MATLAB. The first section of the algorithm separates the pixels into signal and noise on a pixel-wise basis and produces the probabilities that a pixel is signal (or noise). These probabilities are then passed to the second part of the code which incorporates a spatial structure to further determine how likely the pixels are signal or noise.

4.2 INTRODUCTION

As described in section 1.7.1, SMLM is a fluoresence microscopy technique that utilises specific fluorophores that can enter a stabilised dark state. Using fluorophores that turn 'on' and 'off' stochastically means that at any point in time there is only a small subset of fluorophores that are 'on'. A video, or stack of images, are taken of the sample, and in each frame the fluorophores can be localised with a much higher precision than if they were all fluorescing at the same time. This is due to the point spread function (PSF) (figure 1.4) of the emitting molecules no longer overlapping and fluorophores are separated temporally.

One of the difficulties of using SMLM is analysing the image. In most processing algorithms the priority is to identify the location of fluorophores. However, as SMLM has been adapted for different uses, counting the number of fluorophores in the sample has become more prevalent. Especially when it is used for stoichiometry and higher densely populated images. The very nature of stoichiometry requires the number of fluorophores to be known. When analysing higher density images, knowing the number of fluorophores would make it

easier to localise and fit as it would reduce the number of unknowns, allowing for a much higher precision of localisation.

To separate out the necessary data, the pixels are split into two different populations; signal and noise. If the noise pixels can be identified and set to zero, an image showing only the signal can be produced. Having the noise pixels removed will allow for clearer molecule identification.

This chapter describes a two stage algorithm that has been developed that uses Bayesian Inference and random fields to determine the probability that each pixel in a single image is signal. These probabilities can be used to classify the pixels into each population using a 'soft classification'. It is a soft classification as a threshold is applied on probabilities and not upon the raw data.

The first section (section 4.3), the Mixture Model, gives a good estimation of signal and noise, however it is done using a pixel-wise manner and therefore there are some individual pixels that are incorrectly classified as signal/noise.

The second part of the algorithm, described in section 4.4, takes the probabilities from the Mixture Model and uses Markov random fields to incorporate the spatial knowledge into the model. It uses the knowledge that PSFs spread over several pixels and uses the neighbouring pixels to help reclassify the pixels creating a new matrix of probabilities.

4.3 THE FIRST STAGE: DENOISING (THE MIXTURE MODEL)

The first part of the algorithm uses Gibbs sampling as a means of Bayesian Inference to separate the pixels into two populations; signal and noise. This will be called the Mixture Model from this point forward. The process uses a pixel-wise approach and has several limitations. One of the main problems is that, because its mathematical model imposes no physical constraints and treats every pixel as independent, pixels can be incorrectly classified. However, it can be used as a quick and easy de-noising mechanism if that is all that is required.

Bayesian Statistics is described in section 2.3.1 and requires the posterior to be determined (equations (2.6) and (2.7)). For this to be to be calculated, both a likelihood and its priors need to be chosen.

4.3.1 *The Likelihood*

The algorithm assumes that each population comes from a Gaussian distribution with unknown mean, μ_j, and precision, λ_j where $j \in \{n, s\}$ with n referring to the noise population and s referring to the signal population. The following population distribution is used as the likelihood for each individual pixel:

$$f(x_i \mid \boldsymbol{\theta}) = P_s \mathcal{N}(x_i \mid \mu_s, \lambda_s) + P_n \mathcal{N}(x_i \mid \mu_n, \lambda_n), \tag{4.1}$$

where $\boldsymbol{\theta}$ is a place-holder for the unknown parameters and x_i is the intensity of pixel i in the image and $i \in \mathbb{N} = \{1, 2, ..., N\}$ where N is the number of pixels in the image. P_s and P_n are the probability of a pixel being signal or noise respectively over the entire image and $\mu_{s/n}$ and $\lambda_{s/n}$, being the mean and precision of the distribution respectively, with subscript s referring to signal, and n to noise. As each pixel is taken to be independent, identically distributed (i.i.d.) this can be multiplied over i to get the full likelihood for the entire image as follows:

$$f(\boldsymbol{x} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} f(x_i \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \left[P_s \mathcal{N}(x_i \mid \mu_s, \lambda_s) + P_n \mathcal{N}(x_i \mid \mu_n, \lambda_n) \right]. \tag{4.2}$$

4.3.2 *The Priors*

From the likelihood (equation (4.2)) it can be seen that there are five unknown variables: P_n, μ_n, λ_n, μ_s and λ_s. As P_n can be described in terms of P_s: $P_s = 1 - P_n$, only one of these parameters is needed. As there are five unknown variables, five different prior distributions are required. These were chosen as follows:

$$\pi(P_n) = Be(P_n \mid \alpha, \beta), \tag{4.3}$$

$$\pi(\mu_s) = \mathcal{N}(\mu_s \mid m_s, p_s), \tag{4.4}$$

$$\pi(\lambda_s) = \Gamma(\lambda_s \mid a_s, b_s), \tag{4.5}$$

$$\pi(\mu_n) = \mathcal{N}(\mu_n \mid m_n, p_n) \quad \text{and} \tag{4.6}$$

$$\pi(\lambda_n) = \Gamma(\lambda_n \mid a_n, b_n). \tag{4.7}$$

Where α and β are the shape parameters for the P_n Beta distribution, $m_{s/n}$ and $p_{s/n}$ are mean and precision for the $\mu_{s/n}$ Gaussian distribution and $a_{s/n}$ and $b_{s/n}$ are the shape and scale parameters for the λ_s Gamma distribution.

The following constraints can be applied; $P_s + P_n = 1$, $0 \leqslant P_s, P_n \leqslant 1$ and $\mu_s > \mu_n$. The pixels must belong to either the signal or the noise distribution and probabilities have to between zero and one. Also, due to the nature of the system the signal values will be higher than the noise values.

Information about the precision is unknown, however it is thought that the distribution of the noise would be better determined and henceforth $\lambda_n > \lambda_s$. This is due to the noise population being more homogeneous unless there is a lot of autofluorescence coming from the sample. Autofluorescence would decrease the precision of the noise population and may, in the future and in certain images, require its own population. The algorithm can also be tailored more specifically to the system it is trying to denoise. For example, Electronic Multiplying Charge-Coupled Devices (EMCCD) and scientific Complementary Metal-Oxide Semiconductor (sCMOS) cameras will have different types of noise. Although all pixels in an EMCCD camera go through the same read-out register and have have noise coming from the same distribution, they then use an electron multiplication register that uses a stochastic multiplication. This causes the distribution to be changed in a non-uniform matter increasing the variance of the distribution. sCMOS cameras readout at the pixel and therefore the noise associated with each pixel is different: fixed pattern noise. This is explained more in section 2.4. The algorithm here copes with these differences well as the variance is not fixed in the code and, instead, the data is used (along with some prior knowledge) to determine this.

The signal population should have a larger variance as it can come from a single fluorophore or multiple overlapping fluorophores. Also, each of the pixels containing signal will also have background noise contained within them and, therefore, they are expected to have more variance (precision = 1/variance). Furthermore, due to the PSF of the microscope there will be variations in values across the pixels that

represent one fluorophore. For example, the centre of the fluorophore will have a larger value than the ones nearer the edges.

The prior distribution chosen for P_n is a beta distribution. This is because the value has to be between 0 and 1 as it is a probability. The Beta distribution is also in the same family as the Gaussian distribution and will therefore make the equation easier to simplify and calculate the full conditionals.

Both means, μ_n and μ_s, are given a Gaussian distribution as this is a good approximation of each population. A t-distribution could be used for the signal population as it would add some extra information about the tails of the distribution; it would allow the extremes to hold more weight (be more probable). However, as an average value cannot be calculated for a t-distribution, a Gaussian was ultimately chosen. Gaussian distributions are also favoured as they can be made quite flat and uninformative if little information is known about the populations. The distribution for the noise could also be changed to be more physical, however using this Gaussian fit yields good results for images taken using SMLM as shown in the rest of this chapter specifically in sections 4.3.6 and 4.3.7.

Both precisions λ_n and λ_s are given Gamma distributions to ensure that they are always positive. These distributions can also be made flat and uninformative so that the data is used more than the prior.

4.3.3 *Values chosen for priors*

In the priors (equations (4.3) to (4.7)) there are ten different parameters to be fixed: α, β, m_j, p_j, a_j and b_j where subscript $j \in \{n, s\}$. The values chosen for these values are shown in table 4.1.

The vales for α and β were both chosen as $\frac{1}{2}$ in the prior equation for P_n (equation (4.3)). This means that there is an equal probability of the pixel being in either population; signal or noise. This keeps the prior flat and uninformative so that the data drives the result more than the priors. In the future, this could be changed so there is a higher probability that the pixel is noise. This would be more physical as there are more pixels in the noise population. However, it was kept as $\frac{1}{2}$ so that if the signal population is very small the algorithm was less likely to fail (explained more in section 4.3.9).

The values of m_s and m_n were chosen as the 97.5^{th} and 2.5^{th} quantile of the data respectively. Initially, specific values were chosen for these but as the signal value varies across all SMLM images drastically due

Table 4.1: Table shows the chosen values for all the unknown quantities in the prior distributions described in equations (4.3) to (4.7)

Variable	Value
α	$\frac{1}{2}$
β	$\frac{1}{2}$
m_s	97.5^{th} quantile of the data
p_s	0.05
m_n	2.5^{th} quantile of the data
p_n	0.01
a_s	0.01
b_s	0.01
a_n	0.01
b_n	0.01

to the large variety of fluorophores, using a constant number would not be physical for all images. Although this is not a completely Bayesian approach, it is a good approximation. Another method, that is more Bayesian, would be to use the quantile values in only the initial image. In all subsequent images after, the value for μ_s and μ_n that is computed for previous image can be used as the prior values m_s and m_n respectively in the next image. Although this works very well, it stops the algorithm from being parallelised due to the dependence on the previous image. Currently the algorithm can be parallelised across images so that a full stack can be done at a quicker rate. This requires each image to be independent and therefore including this dependence would inhibit this ability.

A further method, would be to alter the prior for each specific fluorophore, environment and microscope used. The user could either pick the prior themselves, or there could be list of prior values for each fluorophore and its environment. This was discarded as the aim was to make the code as independent as possible, so it could work with a large variety of images without having to change the input values.

Values for p_s and p_n where chosen as 0.05 and 0.01. Choosing a low precision for the mean value allows both populations to rely more on the the data. A slightly higher precision was for the signal population to try and ensure that it kept the higher valued pixels in its population rather than them being classified as noise.

The prior values for $a_{s/n}$ and $b_{s/n}$ were all chosen to be 0.01. This creates a distribution with a peak near zero. This allows the two populations to have a large variation and therefore will enable populations to include values that are far away from the mean. Little information is known about the precisions and although it is thought that the noise may be more precise when there is little to no autofluorescence.

4.3.4 *The Posterior*

To calculate the posterior all the priors (equations (4.3) to (4.7)) are multiplied together with the likelihood (equation (4.2)) as described in equations (2.6) and (2.7). This renders a very complicated and unknown distribution (equation (4.8)):

$$\pi(\boldsymbol{\theta} \mid \mathbf{x}) \propto \prod_{i=1}^{N} f(x_i \mid \boldsymbol{\theta}) Be(P_n \mid \alpha, \beta) \mathcal{N}(\mu_s \mid m_s, p_s) \Gamma(\lambda_s \mid a_s, b_s) \mathcal{N}(\mu_n \mid m_n, p_n) \Gamma(\lambda_n \mid a_n, b_n). \quad (4.8)$$

As this is unknown, inference cannot be made directly. Instead a Markov Chain Monte Carlo (MCMC) method is utilised using the full conditionals of each unknown variable. MCMC methods are described in section 2.3.2, and the full distributions are calculated and shown in section 4.3.5.

4.3.5 *Full Conditionals*

As described in section 2.3.3 the full distributions are found using the complicated posterior. All the unknown parameters, bar the one of interest, are set as constants and then the equation is simplified to find a known distribution.

The distribution is found up to its proportionality. The type of distribution is determined and then the constant of normalisation is found using the type of distribution.

From the posterior, described in equation (4.8), it can be seen that there are five unknowns (P_n, $\mu_{s/n}$ and $\sigma_{s/n}$) which come from the five different priors (equations (4.3) to (4.7)). Each unknown requires its own full conditional and therefore there need to be five different full conditionals. These are described as follows:

$$\pi(P_n) = Be(n_n + \alpha, n_s + \beta), \quad (4.9)$$

$$\pi(\mu_j) = \mathcal{N}(\mu_j \mid m_j^*, p_j^*), \text{ and} \tag{4.10}$$

$$\pi(\lambda_j) = \Gamma(\lambda_j \mid a_j^*, b_j^*). \tag{4.11}$$

Where

$$m^* = \frac{n_j \lambda_j \bar{x} + p_j m_j}{n_j \lambda_j + p_j}, \tag{4.12}$$

$$p_j^* = n_j \lambda_j + p_j, \tag{4.13}$$

$$a_j^* = \frac{n_j}{2} + a_j, \tag{4.14}$$

$$b_j^* = b_j + \frac{n_j}{2}(s_j^2 + (\bar{x}_j - \mu_j)^2), \tag{4.15}$$

and

$$s_j^2 = \frac{1}{n_j} \sum_i (x_i - \bar{x}_j), \tag{4.16}$$

$$\bar{x}_j = \frac{1}{n_j} \sum_i x_i. \tag{4.17}$$

Here, subscript $j \in \{n, s\}$ and n_j is the number of pixels in population j.

All of these full conditionals are of known distributions; Beta, Gaussian and Gamma (see appendix A.2). Therefore random numbers can be drawn from these and inference can be performed using MCMC.

4.3.6 *MCMC method*

The data from each image is first split arbitrarily using the 95th percentile so that everything above this value is primarily counted as signal, and anything below is classed as noise. This value is used as it is known that a much larger proportion of pixels will be noise rather than signal. Although any value can be used to initially split the data,

the closer the initial populations are to the final ones, the quicker the algorithm will reach an equilibrium.

There are now two data sets: signal and noise. These two data sets are treated separately and considered independent from each other. For each data set, the parameters m^*, p^*, a^* and b^* are calculated (equations (4.12) to (4.15)). These are then used to draw a random number from the full conditionals of $\mu_{n/s}$ and $\lambda_{n/s}$ (equations (4.10) and (4.11)). These are stored and used as the new values of $\mu_{n/s}$ and $\lambda_{n/s}$.

The final full conditional, P_n (equation (4.9)), is used to calculate an updated version of P_n. This number describes the overall probability that a pixel is noise. The full conditional uses the number of pixels in each population, and a number is drawn from this distribution to update P_n. This value is then used to determine the probability that each individual pixel is noise using equation (4.18).

$$P_{in} = \frac{P_n \mathcal{N}(x_i \mid \mu_n, \lambda_n)}{P_n \mathcal{N}(x_i \mid \mu_n, \lambda_n) + (1 - P_n)\mathcal{N}(x_i \mid \mu_s, \lambda_s)}. \tag{4.18}$$

These probabilities show how compatible each pixel is with the noise distribution. How compatible a pixel is with the signal distribution is given by $P_{is} = 1 - P_{in}$. If there are very few pixels defined as signal then $P_{in} \to 1$ for all i and therefore all pixels will be classified as noise. $P_{in} \to 0$ for all i if most of the pixels are defined as signal. In these two cases, the model breaks down.

Using a Bernoulli distribution, these probabilities are used to re-classify the pixels as signal or noise. A Bernoulli distribution returns either a one or a zero. If a one is returned, the pixel is classified in the noise population, and if a zero is returned, the pixel is classified in the signal population. These values are all stored in the Z matrix. The populations for signal and noise have now been updated and some pixels will have changed which population they are in. This process is now repeated until an equilibrium is reached. All values that are drawn after the equilibrium are an estimate from the posterior.

The output of the Mixture model includes:

- a matrix stating if each pixel is signal (0) or noise (1) for each iteration: the Z matrix, and
- the values drawn for all five of the priors for each iteration:
  - μ_n,
  - μ_s,
  - λ_n,

- λ_s, and
- P_n.

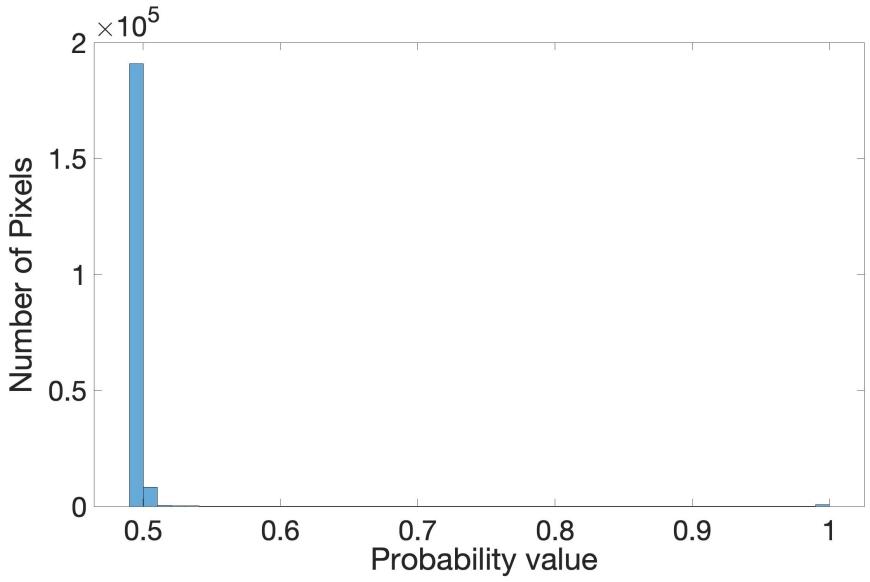
All of this data will include the burn-in period (see section 2.3.2). Before the algorithm reaches equilibrium, the values being drawn are not from the posterior, and therefore have to be discarded. The number of values that need to be discarded can be determined by plotting a graph of the the unknowns and finding where they reach equilibrium. Although this is the most accurate method, it can be very time consuming to do this for a large stack of images. Therefore, instead, a large number of values are discarded to ensure that all of the burnin is removed.

The Z matrix is of size n (number of pixels) by m (the number of iterations) and is used to calculate the probability that each pixel is signal. This is done by finding the average value over all the iterations, after the burn-in period has been removed. The matrix contains only zeros and ones, where a zero denotes it has been classified as noise, and a one denotes it being classified as signal. Therefore, the average over all these values will be the probability that a pixel is signal. An example of these probabilities are shown in figure 4.1. In this figure, figure 4.1a shows the whole range of probabilities, and figure 4.1b shows a close up of the middle range (from 0.53 to 0.99). These values can be used to denoise the images. A threshold is chosen and pixels with probabilities above this are classed as signal, and those below are classified as noise. As this threshold is done on probabilities and not on the actual intensity values from the original image, it can be thought of as a 'soft threshold'. The noise pixels can then be removed or set to zero.

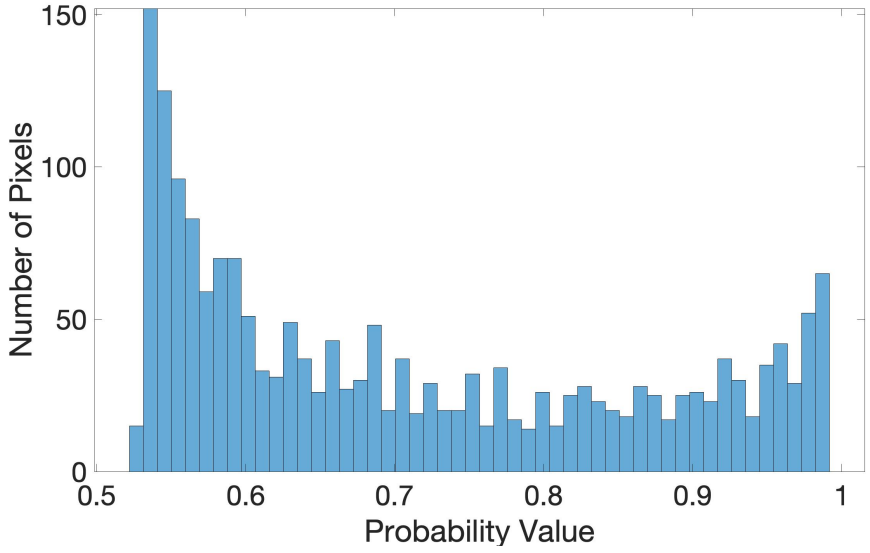
Using a soft threshold on the probabilities is preferred over a 'hard threshold' on the intensity values due to the large variation in intensities. This variation occurs not only across SMLM images taken for one stack, but also between samples. A 'hard threshold' would have to be individually chosen for each image, whereas a soft threshold can be the same across all different images.

Figures 4.2a and 4.3a show images taken using SMLM. These image have been processed using the algorithm discussed above and the output probabilities are shown in figures 4.2b and 4.3b. As described earlier, these probabilities can be used to denoise the images using a soft threshold to create a mask. A mask for figure 4.3a is shown in figure 4.3c. Multiplying this mask with the original image creates a desnoised image as shown in figure 4.3d.

Figure 4.4 shows figure 4.2a after it has been denoised using different 'soft thresholds'. There is a large difference between figure 4.4b and



(a)



(b)

Figure 4.1: The probabilities output from the Mixture Model for figure 4.2a, showing the likelihood a pixel is signal. These are also shown pictorially in figure 4.2b. (a) shows all the values where as (b) shows only the values between 0.52 and 0.99.

the others however there is very little difference between figures 4.4c to 4.4f. For different images there will be different ranges, for some

there will be very little difference between 'soft thresholds' of 0.3 and 0.9. However, generally there is very little difference between 0.6 and 0.95.

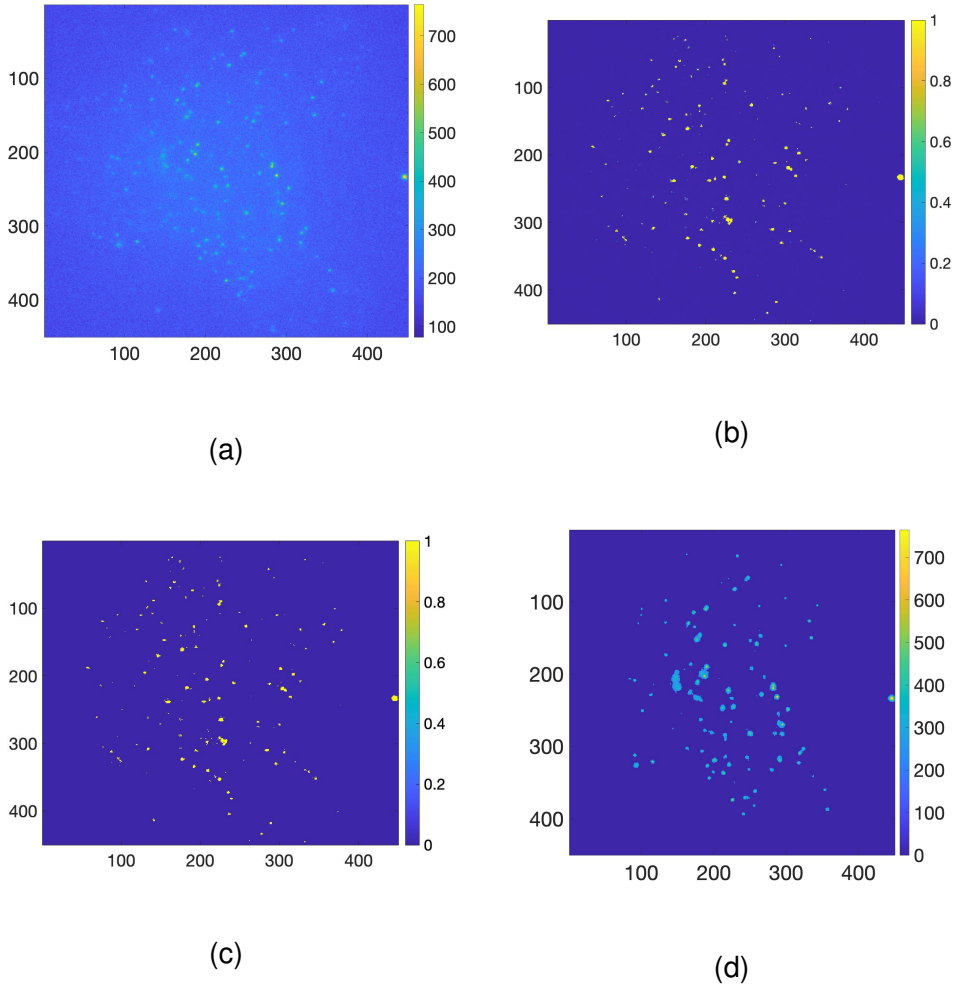


Figure 4.2: (a) An image taken with SMLM of a 3T3 cell labelled with AF 647, (b) the probabilities produced from the Mixture Model showing how likely the pixel is to be signal. (c) A mask produced from (b) using a soft threshold of 0.7. (d) A denoised version of (a) using the mask in (c).

The values of $\mu_{n/s}$ and $\lambda_{n/s}$ describe both populations; signal and noise. In each iteration, a $\mu_{n/s}$ and $\lambda_{n/s}$ are drawn and saved. The mean of these can be used to determine the final value, but also the confidence in this can be calculated by computing the variance. These can be seen in figures 4.5 and 4.6 with figure 4.5 showing the predicted values of $\mu_{n/s}$ and $\lambda_{n/s}$ for figure 4.2a and figure 4.6 shows the populations for figure 4.3a. Figures 4.5a and 4.6a show a histogram of all the

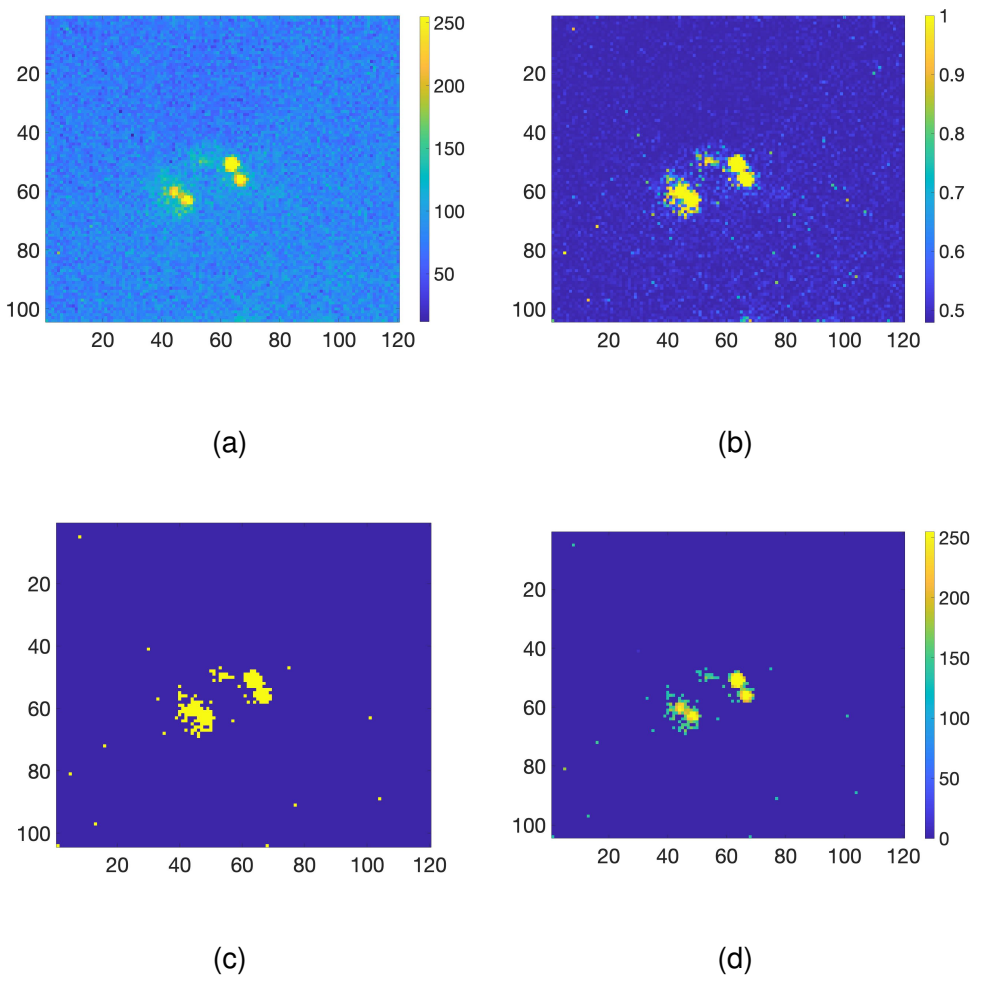


Figure 4.3: a) A sparsely populated image of *Staphylococcus* labeled with AF 647 taken with SMLM. (b) The probabilities produced from the Mixture Model showing how likely a pixel is to be signal. (c) A mask produced from the probabilities in (b) using a soft threshold of 0.7, The pixels with a value of 1 have been classified as signal. (d) A denoised version of (a) using the mask in (c).

pixel intensities in figures 4.2a and 4.3a respectively. Figures 4.5b and 4.6b show histograms of the values drawn for μ_n and μ_s and figures 4.5c and 4.6c show histograms of the values drawn for λ_n and λ_s. Figures 4.5d and 4.6d show the distributions of both the signal and the noise populations calculated using the mean values of $\mu_{n/x}$ and $\lambda_{n/s}$.

These values can be used for various situations including to calculate the signal-to-noise ratio (SNR) (section 4.5.2) and when using the algorithm for data reduction section 4.5.1. These values are also used in

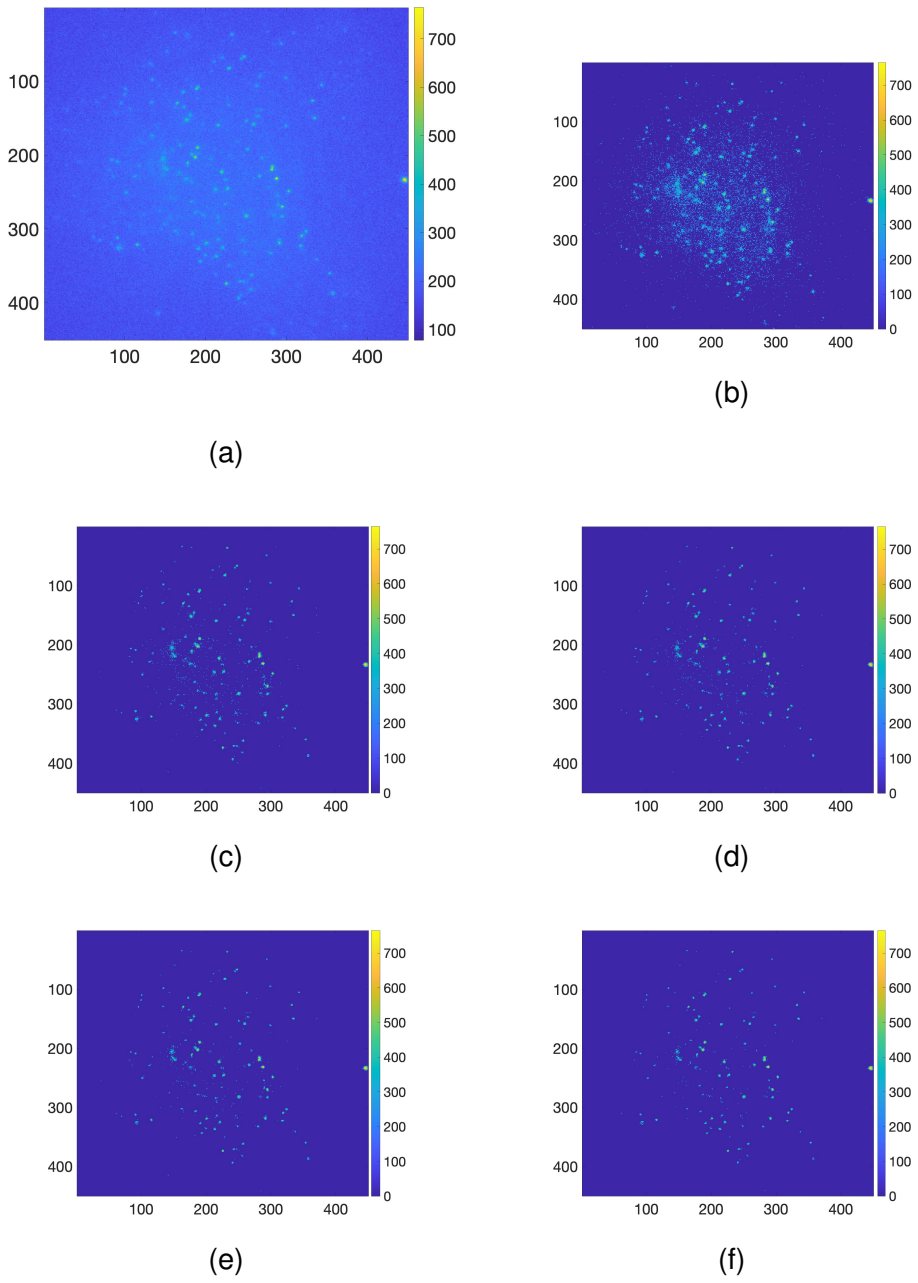


Figure 4.4: (a) The original image that was taken using SMLM and is a repeat of figure 4.2a. Figures (b)-(f) show this image after it has been denoised using soft threshold values of (b) 0.5, (c) 0.6, (d) 0.7, (e) 0.8 and (f) 0.9.

chapter 5 when trying to count the number of fluorophores in a stack of SMLM images.

In figures 4.5 and 4.6 it can be seen that the values of $\mu_{n/s}$ and $\lambda_{s/n}$ look to come from two different populations. As the code redistributes

the pixels into signal and noise populations at every iteration using random numbers and probabilities, the distribution will alter for each iteration. As the two distributions of signal and noise overlap, some pixels will switch between the two groups causing the $\mu_{n/s}$ and $\lambda_{n/s}$ values to alter each time. This causes the posterior to be multimodal as can be seen in figures 4.5b and 4.5c.

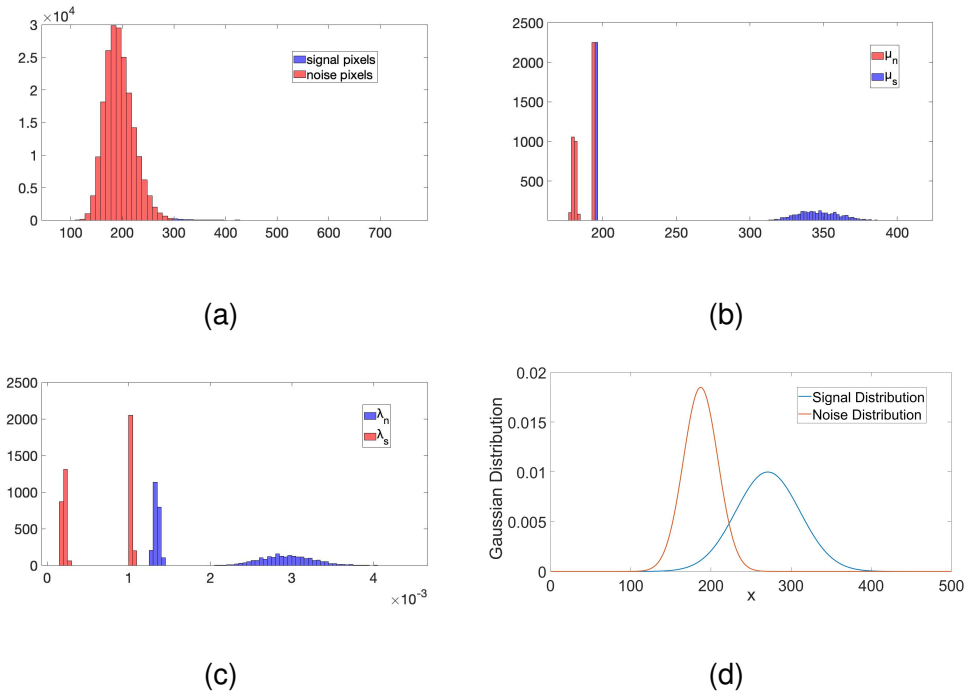


Figure 4.5: (a) A histogram of the intensity values for each pixel in figure 4.2a. (b) Shows a histogram of the drawn μ_s and μ_n values for each iteration (the mean values of the noise and signal populations). (c) Shows a histogram of the drawn λ_s and λ_n values for each iteration (the precision values of the noise and signal populations). (d) Shows the distributions of the noise and signal populations using the mean values of μ_i and λ_i where ($i \in n/s$). In all of these figures, red represents the noise population and blue represents the signal population. It can be seen in both (b) and (c) that the posterior is multimodal. This is due to the overlapping of the two distributions (signal and noise) and the pixels switching between them.

If more accurate values of $\mu_{n/s}$ and $\lambda_{n/s}$ are required, then the fixed populations, determined using a soft threshold, can be sent through a section of the Mixture Model using Bayesian Inference to determine the values from this new data. The priors used in this were calculated using the values from the first iteration through the Mixture Model; the mean

of μ_n and μ_s. These updated values can be used to calculate a more accurate value of the SNR (see section 4.5.2) or to help when trying to count the number of fluorophores in the sample (see chapter 5).

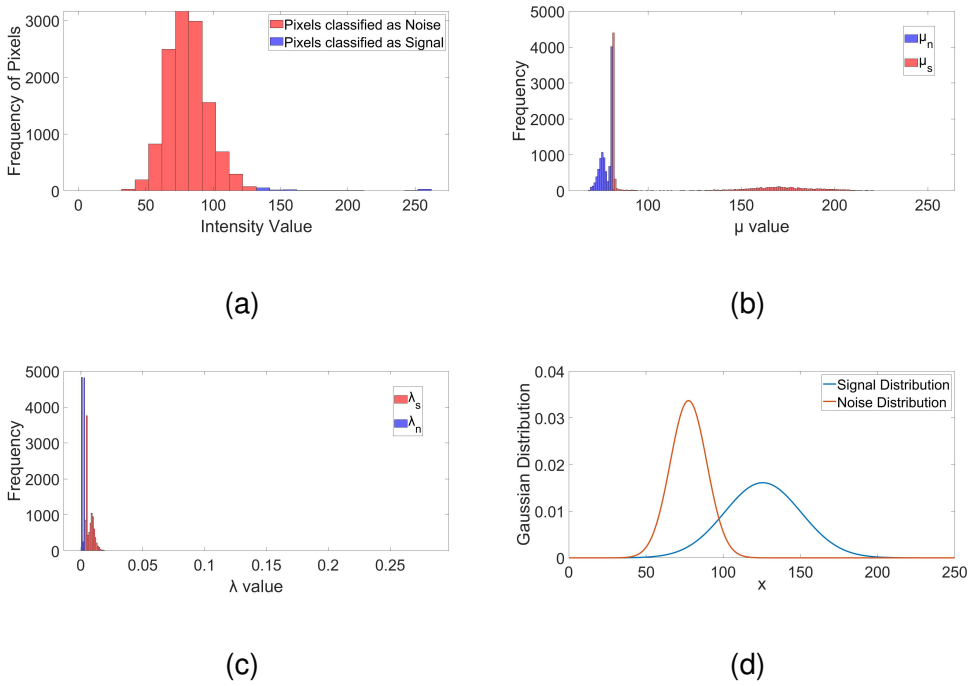


Figure 4.6: (a) A histogram of the intensity values for each pixel in figure 4.3a. (b) Shows a histogram of the drawn μ_s and μ_n values for each iteration (the mean values of the noise and signal populations). (c) Shows a histogram of the drawn λ_s and λ_n values for each iteration (the precision values of the noise and signal populations). (d) Shows the distributions of the noise and signal populations using the mean values of μ_i and λ_i where ($i \in n/s$). In all of these figures, red represents the noise population and blue represents the signal population. It can be seen in both (b) and (c) that the posterior is multimodal. This is due to the overlapping of the two distributions (signal and noise) and the pixels switching between them.

A sensitivity test has been conducted on the Mixture Model. This test showed that the code was robust and the values of m p a and b can be changed without having much difference on the outcome, however it does depend upon the type of images being analysed. This is discussed more in section 4.3.7.

Although this method can be used independently, as each pixel is taken to be independent, the output can have some incorrectly classified pixels. It is known that the PSF of a fluorophore is spread over several

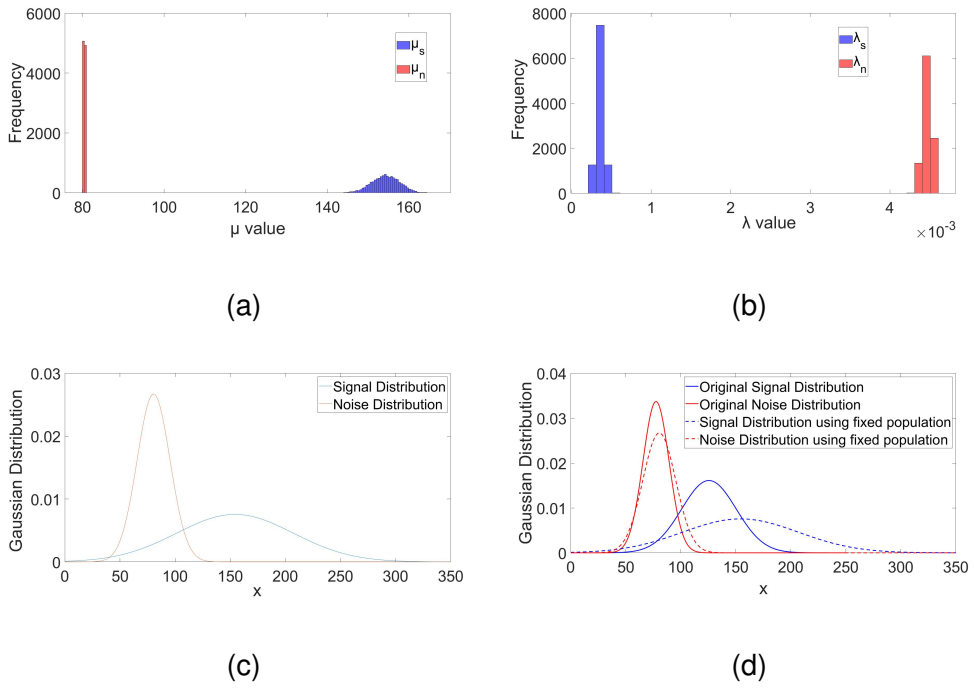


Figure 4.7: The pixels in figure 4.3a were split into fixed noise and signal populations using a soft threshold of 0.7. These populations were sent through a Mixture Model to provide updated values of μ_j (the mean) and λ_j (the precision) where $j \in n, s$. (a) A histogram of the drawn μ_j for each iteration. (b) A histogram of the drawn λ_j for each iteration. (c) The distributions of the fixed noise and signal populations using the mean values of μ_i and λ_i. (d) The distributions of the noise and signal populations from before (figure 4.6d) and using the fixed populations. In all of these figures, red represents the noise population and blue represents the signal population.

pixels and this model can classify isolated pixels as signal (that are surrounded by noise). To incorporate this spatial information into the model, the probabilities calculated in the Mixture Model are passed to a second algorithm which incorporates a spatial structure using the four nearest neighbours to further classify the pixels.

4.3.7 *Sensitivity Test*

The Mixture Model has been sent through a sensitivity test to determine how robust it is. When large, sparse, and clean images are analysed with the algorithm it is robust when choosing the prior values. When

the images have autofluorescence in them then the priors need to be chosen with a lot more thought. If they are not chosen with thought then the algorithm can break, with all pixels being classified into the same population.

Also when analysing smaller data, such as figure 4.3a, the algorithm is less robust and some prior values will cause the code to fail. Prior values for μ_i are the least robust, however, their is more prior knowledge for these unknown parameters.

When changing the values for $a_{n/s}$ and $b_{n/s}$ and $\lambda_{n/s}$ the algorithm is robust. The values for $\mu_{n/s}$ have to be chosen with more thought or the algorithm fails to separate the two different populations. There is, however, typically good knowledge about the values of μ_i and μ_s high comparable to the intensity values of the specific image.

Figure 4.8 shows an image taken using SMLM and then the output of the Mixture Model using different prior values. The values used are shown in table 4.2. From these images it can be seen that the algorithm is robust when changing the prior values as the resultant probabilities are very similar with a mean difference of 4.4958×10^{-5}. Appendix C.2 shows more results from the sensitivity test.

Table 4.2: The priors values chosen for the results shown in figure 4.8

Image	Signal Priors				Noise Priors			
	m	p	a	b	m	p	a	b
a	original image							
b	2.5th quantile	0.01	2	0.6	97.5th quantile	0.01	2	0.6
c	2.5th quantile	0.01	0.01	0.01	97.5th quantile	0.01	0.01	0.01
d	2.5th quantile	0.01	2	0.6	97.5th quantile	0.5	2	0.6
e	2.5th quantile	0.01	0.01	0.01	97.5th quantile	0.5	0.01	0.01
f	2.5th quantile	0.5	0.01	0.01	97.5th quantile	0.01	2	0.6

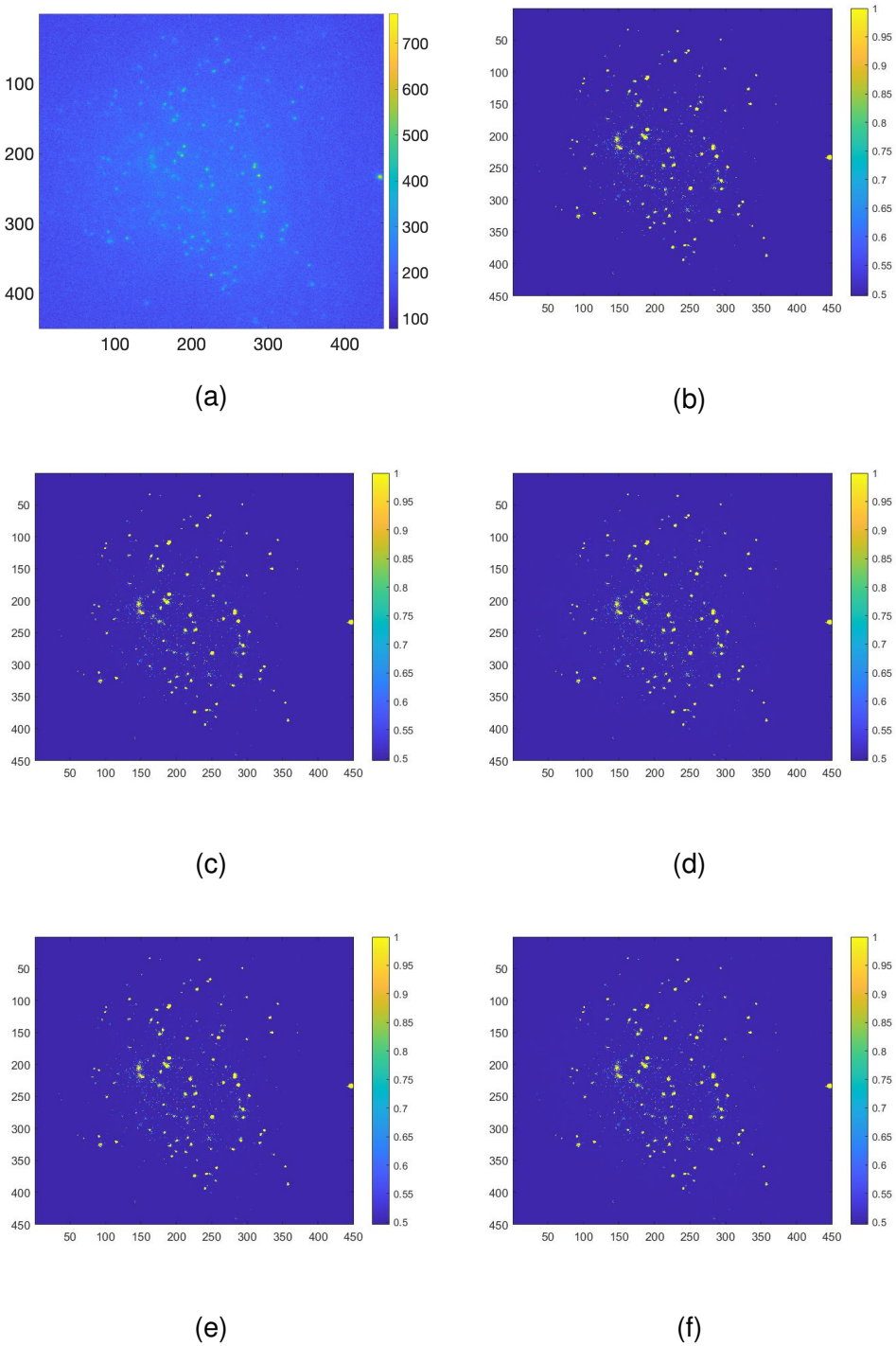


Figure 4.8: (a) The original image (a repeat of figure 4.2a) and then five different (b)-(f) examples of the output using different prior values (More results can be seen in appendix C.2). The prior values used are shown in table 4.2.

4.3.8 *Pseudocode for this Mixture Model*

Given a data set D, which consists of both signal and noise, the likelihood is given by equation (4.2) and the priors are given by equations equations (4.3) to (4.7). equations (4.3) to (4.7). The full posteriors are calculated to be equations (4.9) to (4.11).

1. Split D arbitrarily into signal and noise populations using 95th percentile.
2. Fix values for m_j, p_j, a_j and b_j.
3. For each population, calculate values for $\bar{x}$ and s^2 (equations (4.16) and (4.17)).
4. Calculate a_j^* (equation (4.14)).
5. Arbitrarily choose a value for $\lambda_j > 0$.
6. Use λ_j to calculate/update m_j^* and p_j^* (equations (4.12) and (4.13)).
7. Generate a random number from the distribution of μ_j (equation (4.10)) using current values of m_j^* and p_j^*.
8. Use current value of μ_j to calculate/update b_j^* (equation (4.15)).
9. Generate a random number from the distribution of λ_j (equation (4.11)) using current value of b_j^*.
10. Use equation (4.9) to calculate the probability that a pixel is signal or noise.
11. Use this value and a Bernoulli distribution to classify the pixels as signal or noise.
12. Repeat steps 6-11 m times where m is the number of iterations.

Finally, the code was run on the same data using different seeds to check convergence. figure 4.17 shows the data from figure 4.2a. table 4.3 shows the different quantiles for the different runs using different seeds. From both of these it can be seen that when different seeds are used, the algorithm still converges at the same place, producing the same results.

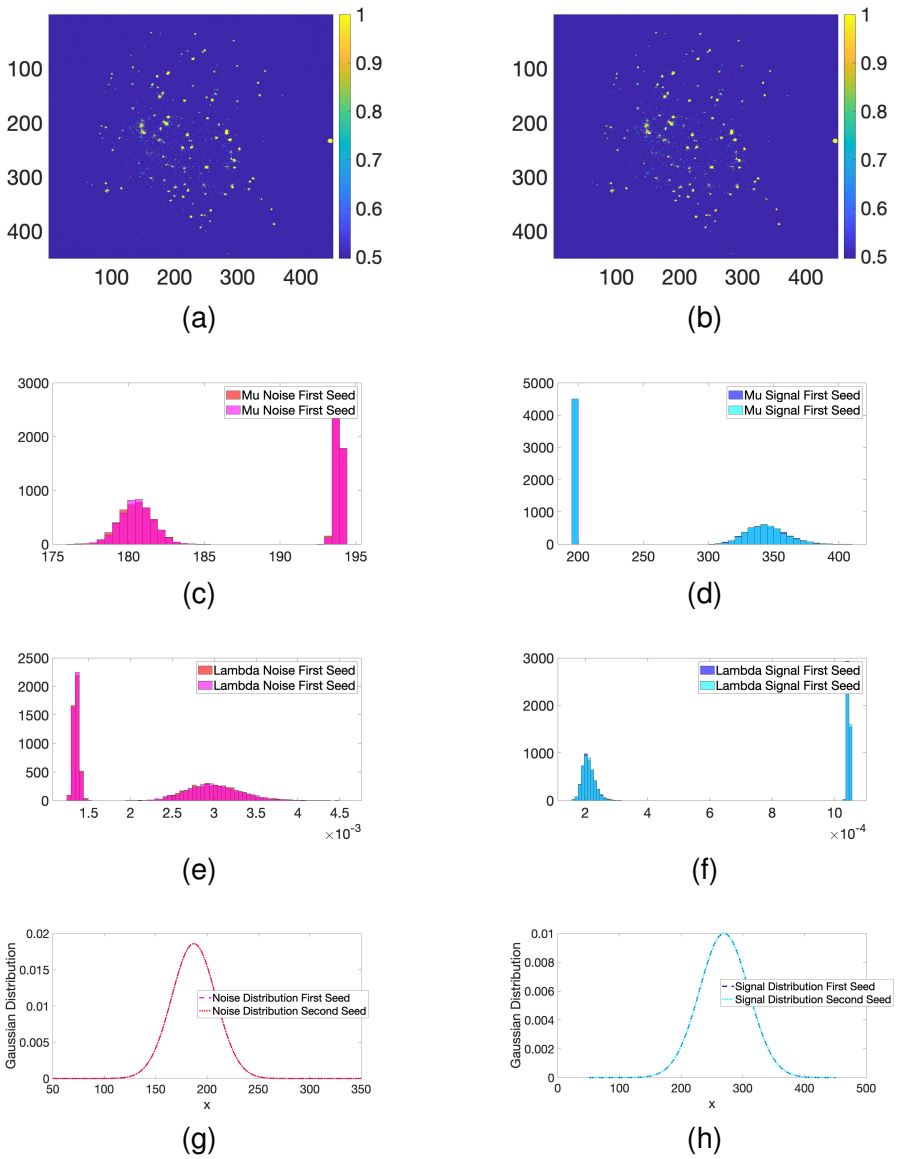


Figure 4.9: The results from using the same data (as shown in figure 4.2a) through the Mixture Model using two different seeds. (a) The probability outputs using the first seed. (b) The probability outputs using the second seed. (c) A histogram showing the values of μN from the two different seeds. (d) A histogram showing the values of μS from the two different seeds. (e) A histogram showing the values of λN from the two different seeds. (f) A histogram showing the values of λS from the two different seeds. (g) The distribution of the two noise distributions. (h) The distribution of the two signal signals

Table 4.3: Quantile values from the results shown in figure 4.9. The Mixture Model was used on the same data produced using different seeds to check convergence.

	Probabilities		mun		mus		lamn		lams	
Quantile	Seed 1	Seed 2	Seed 1	Seed 2	Seed 1	Seed 2	Seed 1	Seed 2	Seed 1	Seed 2
0	0.4968	0.4966	174.5731	174.5745	195.0949	195.0961	0.0013	0.0012	0.0002	0.0002
0.25	0.4987	0.4987	180.4918	180.5138	195.1123	195.1123	0.0013	0.0013	0.0002	0.0002
0.5	0.4992	0.4992	184.2015	184.1701	209.8690	209.9269	0.0021	0.0021	0.0008	0.0008
0.75	0.4996	0.4996	193.8716	193.8730	343.1587	343.3808	0.0030	0.0030	0.0010	0.0010
1	1.0000	1.0000	194.4381	194.4516	407.8976	410.4297	0.0050	0.0050	0.0014	0.0014

174.5731 180.4918 184.2015 193.8716 194.4381

4.3.9 *Limitations of the Mixture Model*

There are several different limitations of this Mixture Model algorithm. It is computationally intensive and it requires there to be enough data in each population. The priors should work for the majority of SMLM images, however, as there are such a diverse number of fluorophores, environments and cameras, the data varies widely. There may be some scenarios where the prior values will need to be altered.

The computational intensity increases linearly with the number of pixels in the image and how many times the process is iterated through. The code, however, is parallelised across images so it can iterate through several images at once.

This will vary from data set to data set depending on the data itself. If there is a lot of data in each population then the result will depend more on the data than the priors. This means that the noise population will tend more towards the data rather than the prior. The signal population is significantly smaller in size than the noise population and, therefore, will tend more towards the prior than the data. If the precision of the data (in each population) is small, then then mean of the posterior will tend towards the prior. The larger the precision the more it will tend to rely on the data. This can be seen from equation (4.12). Therefore the signal population, the more interesting of the two, will depend more on the prior values chosen than the noise population.

If there are not enough data in each population then the algorithm classifies them all into one group and the algorithm fails. This is due to equation equation (4.18). If P_n is large, then all pixels are classified as noise as $P_{in} \to 0$ for all i. If all the pixels are classified into the same group then the algorithm draws a new set of probabilities to try and redistribute the pixels so that there are pixels in both groups. A limiter is in the code that breaks the algorithm if it loops through this step 1000 times.

Synthetic images were created using the simulation algorithm described in chapter 3. The number of fluorophores 'on' in the image were reduced until the resultant data was unable to be split by the Mixture Model. When looking at synthetic images with a SNR of 3.57, the algorithm needs around 1.38% of pixels to be in the signal population. The SNR was calculated using the method described in section 4.5.2 and the number of pixels in the signal population was calculated using a soft threshold of 0.8. Using images with higher SNR yielded a similar

result of 1.38%. As the SNR decreases, the percentage of pixels needed increases, also the number of incorrectly classified pixels increases as the higher values for noise become comparable with the lower values of the signal. A SNR value of 2.22 requires around 4% of pixels to be signal.

The code used to swap the populations around and classify signal as noise and vice versa. This is known as label switching. Figure 4.10 shows an image that has been denoised incorrectly in this manner. This, however, has been rectified by adding in an extra step to ensure that the signal population always has a higher mean than the noise population. If the values were reversed then the code automatically swapped them to ensure that the correct values were allocated to the correct populations.

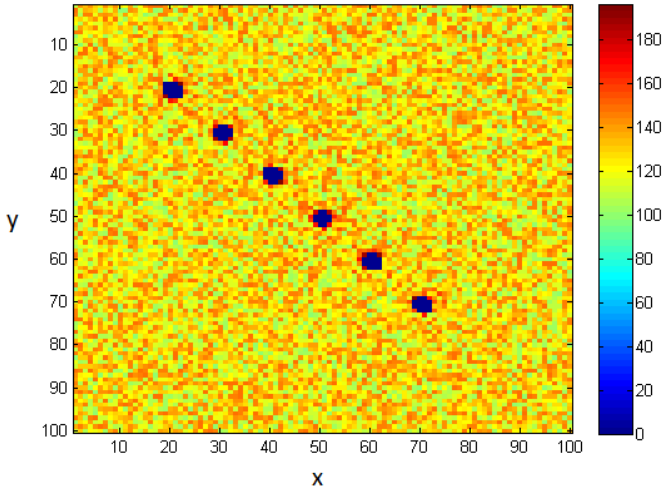


Figure 4.10: Data produced from Mixture Model from simulated data. Algorithm has identified the two populations successfully, however has not managed to denoise the image. The colour scale shows the intensity of the pixels

4.4 THE SECOND STAGE: THE MODIFIED ISING MODEL (MIM)

This section of the algorithm is an adaptation of the Ising Model; a mathematical model of ferromagnetism in statistical mechanics [116]. This has been modified by having the different spins as the two different populations. The algorithm uses a markov random field (MRF) and the four nearest neighbours to further determine the classification of the pixels. It uses both MRF [117] and the Metropolis-Hastings (MH) ratio [93].

4.4.1 *Markov Random Fields (MRF)*

A MRF is an undirected graphical model for a set of random variables, called nodes, which are divided into cliques [117]. They provide a mechanism for enforcing spatial consistency across images, incorporating both the previous estimate and the relationship between neighbouring nodes. [118]

Images are split up into nodes which are connected to their neighbours via edges. The notation for these image graphs is:

$$G = (\mathcal{V}, \mathcal{E}), \tag{4.19}$$

where $\mathcal{V} = (1, 2, ..., i, ..., N)$ is the number of nodes or vertices, N being the maximum number or nodes, and $\mathcal{E}$ represents the edges connecting them. A typical edge is given by: (i, j) with $i, j \in V$, and the graph is undirected meaning $(i, j) = (j, i)$. This is represented in figure 4.11. All the circles (shaded or otherwise) are the nodes, and all the lines connecting them are edges. MRFs are only interested in local interactions and use a neighbourhood system to achieve this. The neighbourhood model can be extended (or reduced) to include more (or less) neighbours as required. Figure 4.11 shows the most simple of local interactions and figure 4.12 shows how it can be extended. Figure 4.12 also shows cliques which are a subset, C, of S. A subset is a clique if two different elements of C are always neighbours.

In graph interpretation, if two nodes are linked by an edge, then they are neighbours. Mathematically this is defined as follows. Let S be a finite set of pixels with elements, or sites, denoted by s. Λ is the finite phase space. A random field on S with phases in Λ is a collection $X = \{X(s)\}_{s\in S}$ of random variables $X(s)$ with values in Λ. A neighbourhood system on S is a family $N = \{\mathcal{N}\}_{s\in S}$ of subsets of S such that for all $s \in S$: $s \notin \mathcal{N}_s$, and if $t \in \mathcal{N}_s \rightarrow s \in \mathcal{N}_t$. That is, s cannot be a neighbour of itself, and if t is a neighbour of s then s is a neighbour of t.

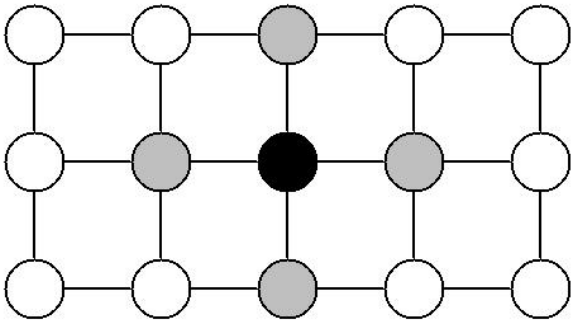


Figure 4.11: Graphical representation of data. Nodes are represented by circles (shaded or not) and edges are represented by straight lines. Given the grey nodes, the black node is conditionally independent of the white nodes. [119]

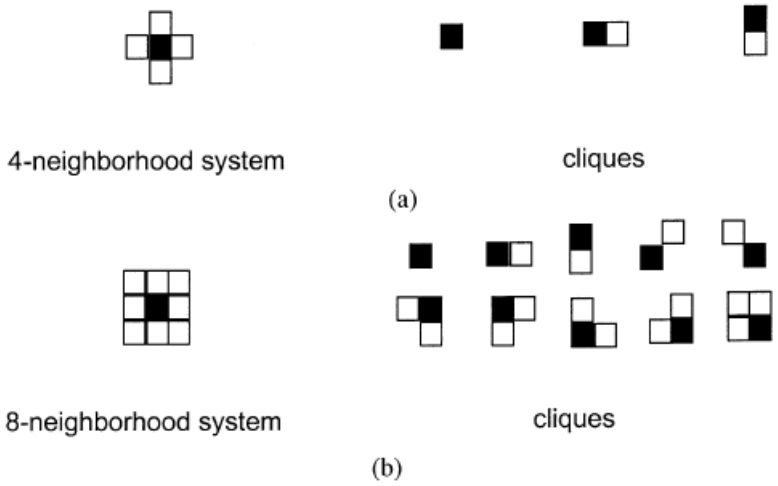


Figure 4.12: Two different neighbourhood systems and their corresponding cliques. [120]

$\mathcal{N}_s$ is called the neighbourhood of site s, and if s and t are neighbours it is denoted $s \sim t$.

A random field is an MRF if:

$$P\Big[X(s) = x(s) \mid X(S\backslash s) = s(S\backslash s)\Big] = P\Big[X(s) = x(s) \mid X(\mathcal{N}_s) = s(\mathcal{N}_s)\Big] \quad (4.20)$$

That is, an element is only dependent on its neighbourhood and not the rest of the data. This is shown in figure 4.11, the black node is conditionally independent of the white nodes, given the grey nodes.

4.4.2 *Metropolis-Hastings Ratio*

The Mixture Model works by generating random numbers from the full distributions as they are of known distributions. If the distributions are unknown and direct sampling is not possible then the MH algorithm, or Metropolis algorithm, is used. [94]

Instead of drawing from the full distribution, a proposal distribution, or candidate-generating density is used. This is a Probability Density Function (PDF) and is permitted to depend upon the current state of the process. If the process is at the point x, and a value y is to be generated, then the proposal is denoted:

$$q(x, y). \tag{4.21}$$

Instead of drawing a random number from the full distribution, a number is generated from this proposal [93]. In principle this proposal can be any distribution, however in practice a distribution that fits as closely as possible to the full distribution is chosen. [94]

This new value, y is kept with probability $\alpha(x, y)$. This is known as the Metropolis Ratio. It determines how compatible the new value is compared to the current value. It is defined as:

$$\alpha(x, y) = \begin{cases} \min\{\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\}, & \text{if } \pi(x)q(x, y) > 0. \\ 1, & \text{if } \pi(x)q(x, y) = 0, \end{cases} \tag{4.22}$$

where $\pi(x)$ is the full distribution [93]. This ratio is sometimes referred to as the probability of move. If y is rejected, then the previous value, x, is repeated and used as the next value. The Mixture Model can be thought of a special case of the MH algorithm where $\alpha(x, y)$ is always equal to one. [94]

4.4.3 *Modified Ising Model (MIM)*

The Ising model is a mathematical model of ferromagnetism in statistical mechanics. Here it has been adapted for use in de-noising with the different spins being the two populations. [116]

In the Ising model, the total energy of the system, the Hamiltonian ($\mathcal{H}$), is

$$E(= \mathcal{H}) = J \sum_{i \sim j} z_i z_j, \tag{4.23}$$

where z_i denotes the spin of particle i, and the sum is over all the neighbouring particles, $i \sim j$, as defined in section 4.4.1. J is a constant that represents the strength of interaction between the particles. If this is positive then neighbouring particles having the same spin decreases the energy.

The Ising model has an inverse temperature dependence which is related to the amount of energy the system contains and the amount of energy it requires to change spin. In this adaptation, this parameter is also denoted as J, and can be thought of the inverse of temperature. The lower this value, the higher the temperature and therefore the more energy is in the system, meaning that more, smaller clusters are favoured. Therefore, pixels with neighbours in the opposite population are unfavourable. The temperature is treated as an unknown parameter and is inferred from the data. [116]

Statistical mechanics states that, if set C represents all possible configurations, and $E(c)$ is the energy of configuration of $c \in C$, then the probability of any particular configuration c, $P(c)$, is:

$$P(c) = \frac{1}{Z} \exp\left\{ -\frac{E(c)}{kT} \right\}. \tag{4.24}$$

Where Z is the normalisation constant, T is the temperature in Kelvin, and k is Boltzmann's constant. Using this and the Hamiltonian from before (equation (4.23)), then

$$P(y) = \frac{1}{Z} \exp\left\{ J \sum_{i \sim j} y_i y_j \right\}, \tag{4.25}$$

where y_i is now the output probability value from the Mixture model at position i.

As

$$\sum_{i \sim j} y_i y_j = 2N(N-1) - 2d_y, \tag{4.26}$$

where N is the number of pixels and d_y is the number of disagreeing edges, the local structure can be measured using:

$$P(y) = \frac{1}{Z} \exp\left\{ 2J\big(N(N-1) - d_y\big) \right\}. \tag{4.27}$$

As $N(N-1)$ and Z are constants, this can be written as:

$$P(y) \propto \exp(-2Jd_y). \tag{4.28}$$

In this section we treat the neighbourhood structure, θ, as unknown, formalised as the Ising model, thus the prior for the neighbourhood is given by:

$$\pi(\theta) \propto \exp(-2Jd_\theta) \tag{4.29}$$

Given that the inverse temperature J has a strong influence on the number of cluster in the filed, a prior distribution was used to allow the data to determine a value for J. As J can be thought of as the inverse of temperature, it must be positive value. Therefore, a Gamma distribution has been chosen as its prior:

$$\pi(J) = \Gamma(J \mid a_J, b_J). \tag{4.30}$$

Here, a_J and b_J are the shape and scale parameters for the Gamma distribution.

The data in this model can be represented with a Gaussian distribution:

$$L(y, \theta) \propto \exp\left(\frac{-1}{2\sigma^2}\sum_{m,n}(y_{mn} - \theta_{mn})^2\right). \tag{4.31}$$

As before, $y_{m,n}$ is the output data (probabilities) from the Mixture model at position m, n. $\theta_{m,n} = \{1, -1\}$ is the classification of the pixel into either the signal (1) or noise (-1) population. σ is the variance of this likelihood.

Combining equations (4.29) to (4.31) and using equations (2.6) and (2.7) the posterior is:

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{D}) \propto \exp\left(\frac{-1}{2\sigma^2}\sum_{m,n}(y_{mn} - \theta_{m,n})^2\right) J^{a_J - 1} \exp\Big(-J(b_J + 2d_\theta)\Big). \tag{4.32}$$

This posterior (equation (4.32)) is very complicated and only one full conditional of the two unknowns is a known distribution: J equation (4.33). The full conditional of the neighbourhood is an unknown distribution and therefore the MH algorithm is used.

The full distribution of J is given by:

$$\Gamma(J|a_J, b_J^*), \tag{4.33}$$

where $b_J^* = b_J + 2d_\theta$. As this full conditional is a known distribution then random values can be drawn from it. Therefore there is no need to use a MH ratio for this parameter.

For the neighbourhood, the full distribution is of an unknown form and therefore numbers cannot be sampled from this. Instead a proposal is used that changes the value of one pixel to the opposite population, for example, if it is classified as signal it changes it to noise (-1 goes to 1, and 1 goes to -1). The proposal is denoted $q(\theta, \theta')$ where θ is the current neighbourhood, and θ' is the proposed neighbourhood with the pixel in question in the other population. The ratio of the proposal equations is given by:

$$\frac{q(\theta', \theta)}{q(\theta, \theta')} = e^{\frac{-2y(pos)}{\sigma^2}}. \tag{4.34}$$

Here, $y(pos)$ is the probability value of the pixel in question that came from the Mixture Model, and σ is related to the amount of noise in the system as described earlier. This value is fixed, however in the future it is possible to treat it as an unknown parameter and infer it from the data.

The full potential is given by:

$$\pi(\theta) \propto \exp(-2Jd_\theta). \tag{4.35}$$

The ratio of the full potentials is given by:

$$\frac{\pi(\theta')}{\pi(\theta)} = e^{2J(d_\theta - d_{\theta'})}, \tag{4.36}$$

where J is the inverse parameter related to temperature and d_θ and $d_{\theta'}$ are the number of neighbourhood pixels that are in the opposite population originally, and when it has been changed to the other population in the proposal respectively. For example, if there is a pixel classified as signal surrounded by four noise pixels, $d_\theta = 4$ and $d_{\theta'} = 0$.

Multiplying equations (4.34) and (4.36) gives the Metropolis ratio; α. A random number between 0 and 1 is generated. If this generated number is less or equal to α then the proposed value is kept. If it is greater than α then the proposed value is rejected and the previous value is reused.

As with the Mixture Model, the MIM goes through many iterations. The more iterations are completed the more accurate the final result will be, up to the limitations of the model. In every iteration, the algorithm goes through each pixel in the image in a random order. The state of the pixel is inverted (from signal to noise or vice versa) and the MH ratio is used to determine which population the pixel should be in. How many times it is classified into each population determines the probability it should be signal (or noise).

Results from the MIM are shown in figures 4.13 and 4.14 with figure 4.13 showing the results from figure 4.2a and figure 4.14 showing the results from figure 4.3a. The probabilities that were inputted into the code are were the the outputs from the Mixture Model shown in figures 4.2b and 4.3b.

4.4.4 *Sensitivity Test*

As with the Mixture Model, a sensitivity test has been performed on the MIM. The values that can be changed in this section are: the prior values for J, a_J and b_J, and σ. J is the inverse temperature dependence and σ which is related to how noisy the image is. In this algorithm J is treated as an unknown and therefore has its own distribution, whereas σ is taken as a constant. However, in the future this could be changed and inferred from the data.

Figures 4.15 and 4.16 show the output of the Ising section of the algorithm for several different prior values for images figures 4.2a and 4.3a. For figure 4.2a (figure 4.15), only the centre of the image has been shown so it can be easier to see the pixel values. Figure 4.15a shows centre of the original image, figure 4.15b shows the Gibbs probabilities and figures 4.15c to 4.15f show four outputs from the MIM. The priors for these results and the results shown in figures 4.16c to 4.16f are shown in table 4.4.

Here is only a selection of the results and their priors, a more comprehensive test has been completed however only a selection are shown here. It can be seen that the Ising section is a lot less robust than the Mixture Model and therefore the values need to be chosen with more care. For both these images the prior values of $a_J = 2$, $b_J = 2$ and $\sigma = 2$ yield the best results and therefore these were the values chosen for future images.

Table 4.4: Prior values used to produce results in figures 4.15 and 4.16

	Prior values		
Images	a_J	b_J	σ
c	1	1	0.5
d	2	2	2
e	3	3	0.75
f	6	4	1

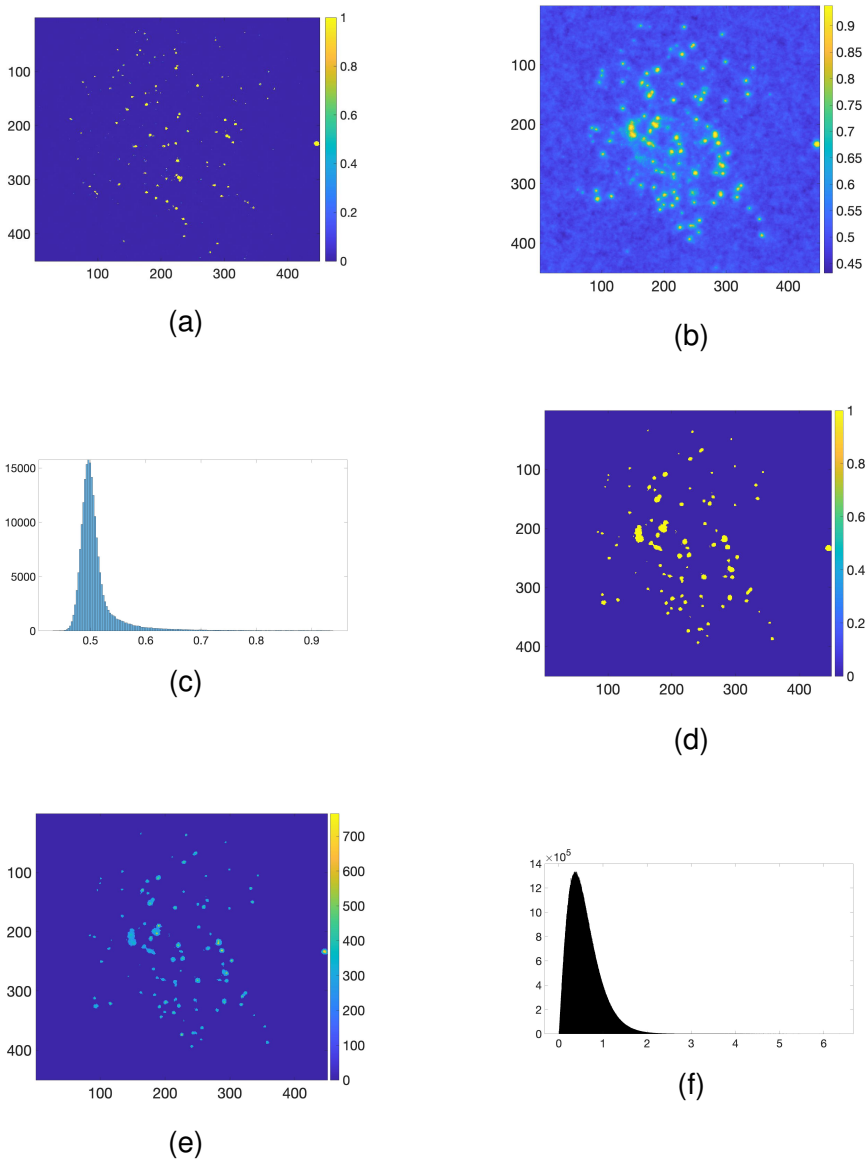


Figure 4.13: (a) The probabilities produced from the Mixture Model for figure 4.2a showing the likelihood each pixel is signal (a repeat of figure 4.2b). (b) The updated probabilities output from the MIM where the input was (a). (c) A histogram showing the same probabilities in (b). (d) A mask created using a soft threshold of 0.7 on the Ising probabilities. (e) A denoised version of figure 4.2a using the mask shown in (d). (f) A histogram showing the drawn values of J from each iteration of the MIM.

Finally, the code was run using different seeds to ensure that the algorithm converged the same place each time. The results of this can be seen in figure 4.17. A histogram showing all of these is shown in

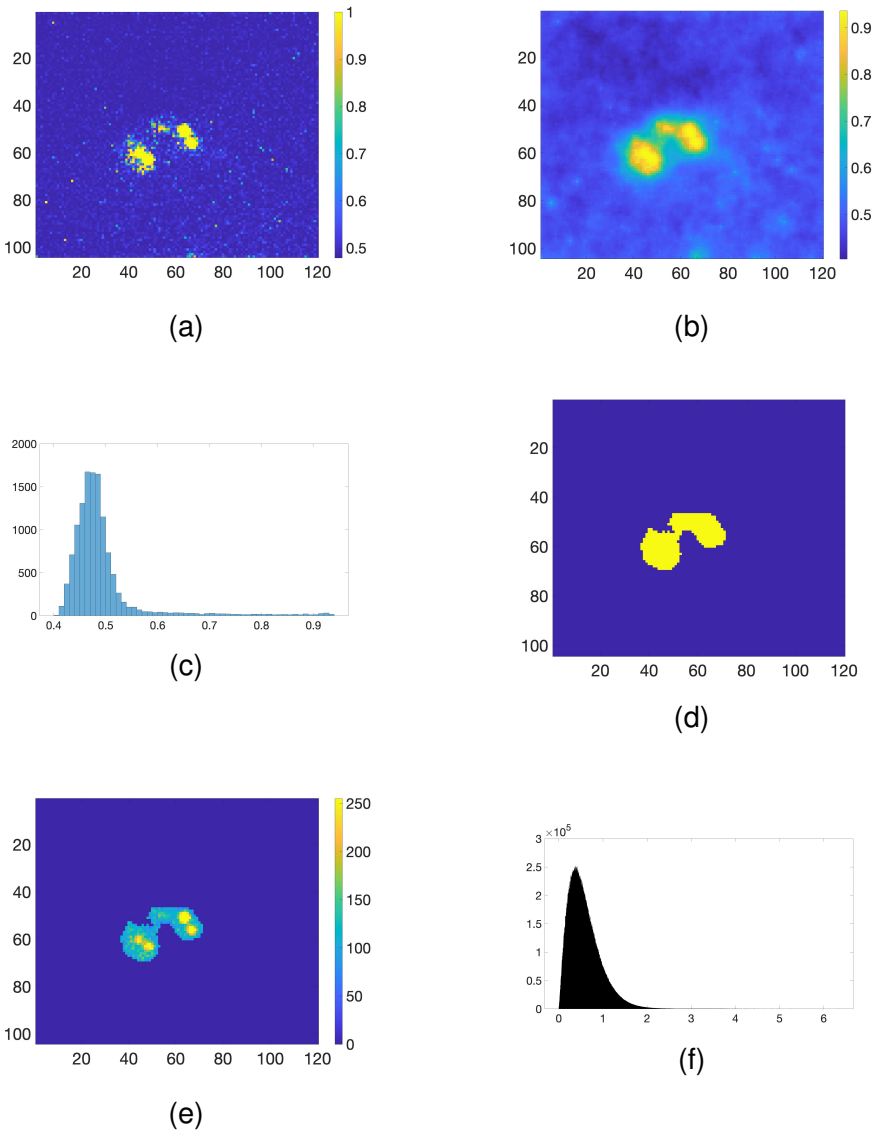


Figure 4.14: (a) The probabilities produced from the Mixture Model for figure 4.3a showing the likelihood each pixel is signal (a repeat of figure 4.3b). (b) The updated probabilities output from the MIM where the input was (a). (c) A histogram showing the same probabilities in (b). (d) A mask created using a soft threshold of 0.7 on the Ising probabilities. (e) A denoised version of figure 4.2a using the mask shown in (d). (f) A histogram showing the drawn values of J from each iteration of the MIM.

figure 4.18. The quantiles of these results can be seen in table 4.5. From this it can be seen that the algorithm converges when different seeds are used.

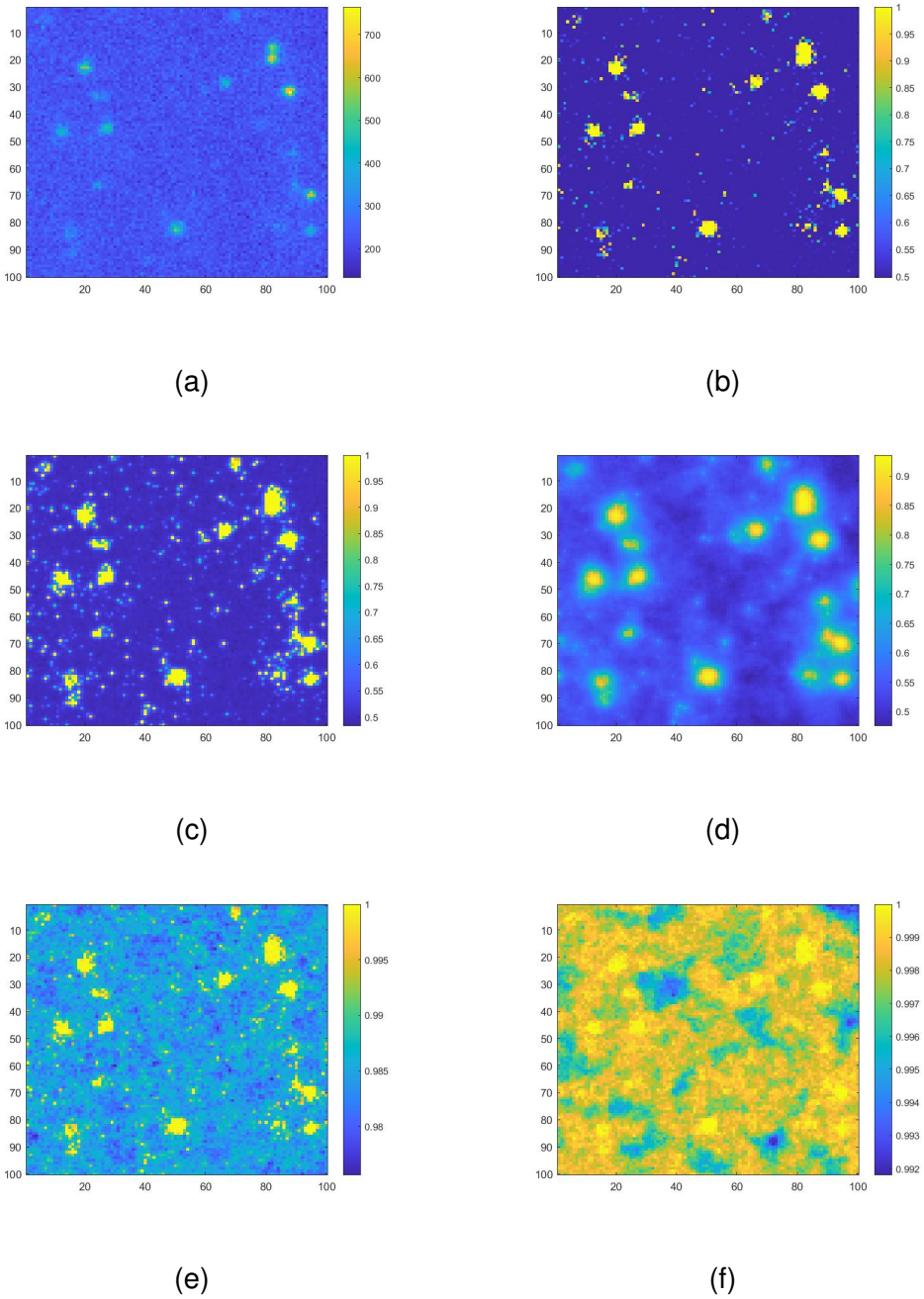


Figure 4.15: (a) The middle area from figure 4.2a, an image taken using SMLM and the (b) the middle area of figure 4.2b, the output from the Mixture Model. (c)-(f) Different outputs from the MIM using different prior values. The prior values are shown in table 4.4

4.5 APPLICATIONS

This code has several different applications, including data reduction, measuring SNR and as a preprocessing algorithm.

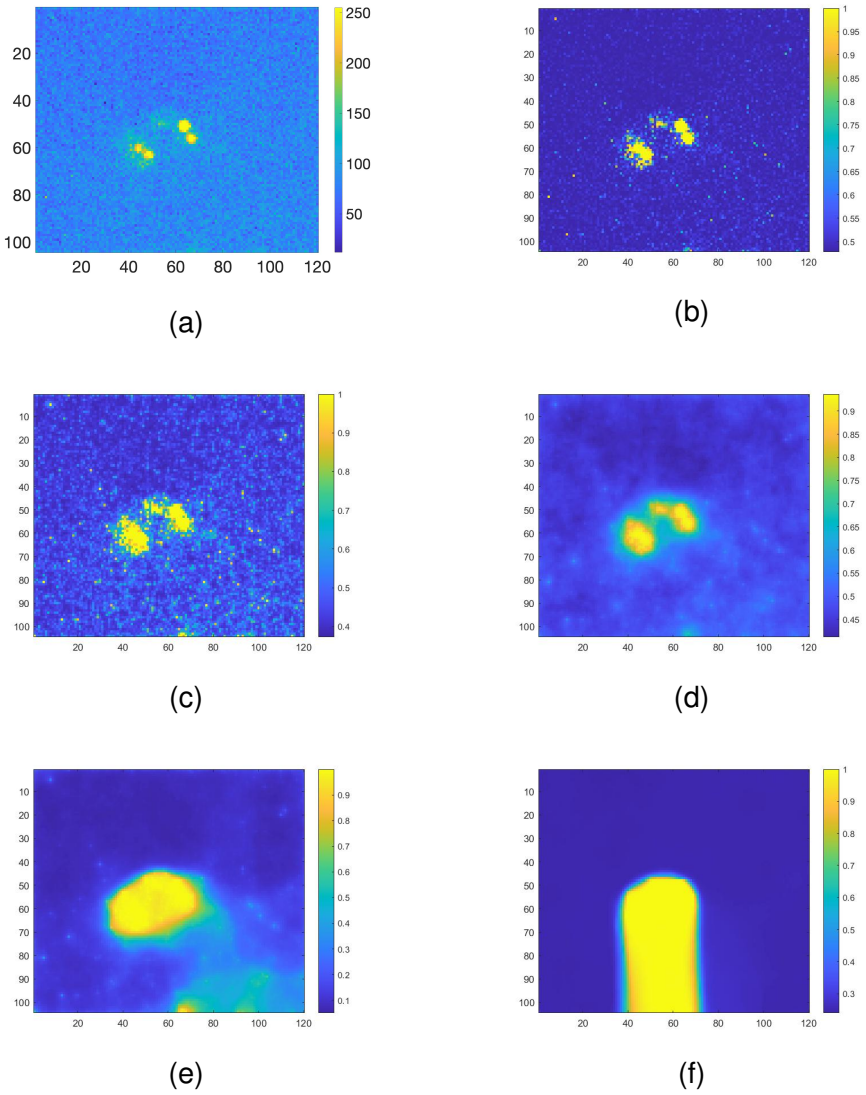


Figure 4.16: (a) A copy of figure 4.3a, an image taken using SMLM and the (b) the copy of figure 4.3b, the output from the Mixture Model for (a). (c)-(f) Different outputs from the MIM using different prior values. The prior values are shown in table 4.4

4.5.1 *Data Reduction*

The algorithm can be used to reduce the amount of memory needed to save the same amount of data. As most of the pixels are only noise, and contain no information about the location of the fluorophores, these pixels can be discarded. Depending on how many fluorophores are emitting in each frame and how dense they are changes the amount of pixels that can be discarded.

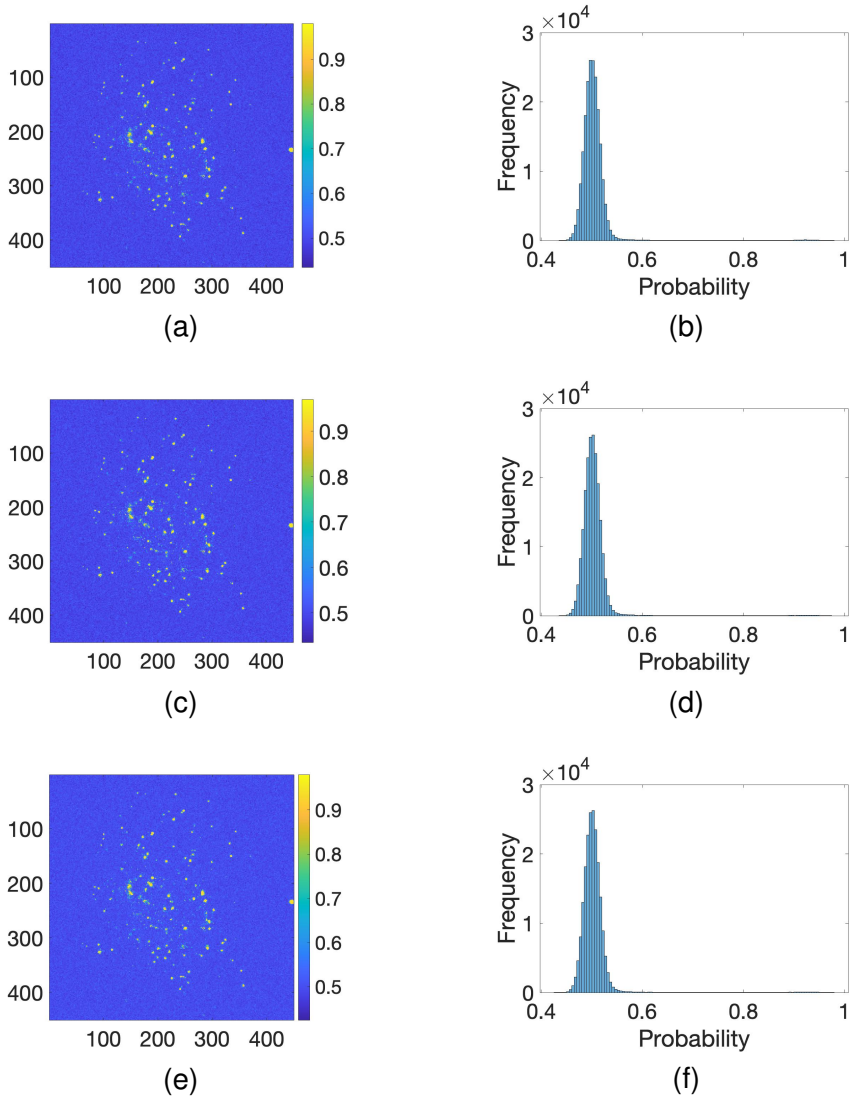


Figure 4.17: (a, c, e) Images showing the probability results using the MIM using different seeds to test that the algorithm converges in the same place. The probabilities going in to this are shown in figure 4.16a. (b, d, f) Histogram of the probabilities that can be seen in (a), (c), and (e) respectively.

Denoising the images can be done using either just the Mixture Model or after both the Mixture Model and the MIM. The data is saved as a as a sparse matrix which contains only the positions and the intensity values of the signal data. As images taken with SMLM have such a high proportion of pixels as noise, if only the signal data is saved then a large amount of memory can be removed. This can be achieved by denoising the images using the Mixture Model and MIM

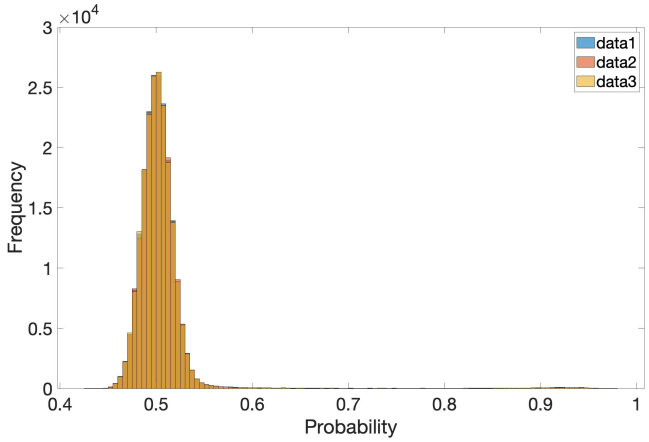


Figure 4.18: A histogram showing the probabilities from the MIM when three different seeds are used. This contains the results shown in figures 4.17b, 4.17d and 4.17f

Table 4.5: Quantile values from the results shown in figure 4.17. MIM was used on the same data produced in the Mixture Model (figure 4.16a) using different seeds to check convergence.

	MIM Results		
Quantile	Seed 1.	Seed 2	Seed 3
0	0.4350	0.4350	0.4250
0.25	0.4900	0.4900	0.4900
0.5	0.5000	0.5000	0.5000
0.75	0.5100	0.5100	0.5100
1	0.9800	0.9700	0.9800

stated above and then saving these images as sparse matrices. This can reduce the amount of memory required to save images by, on average, 95%. This reduces an 8Gb stack down to 400Mb.

When images are sparsely populated, for example in figure 4.3a then this works exceedingly well. The image was sent through the ThunderSTORM algorithm to determine the positions of the fluorophores. These positions are shown in figure 4.19. Next the image was run through the Mixture Model and denoised using a threshold of 0.8. ThunderSTORM uses the background noise to aid in its localisation algorithm and, therefore, the images were re-noised using values randomly chosen from a Gaussian of mean m_n and λ_n. It can be seen that the re-noised Gibbs image finds the locations of the same fluorophores

as in the original image. The Ising model however finds the location of two more fluorophores.

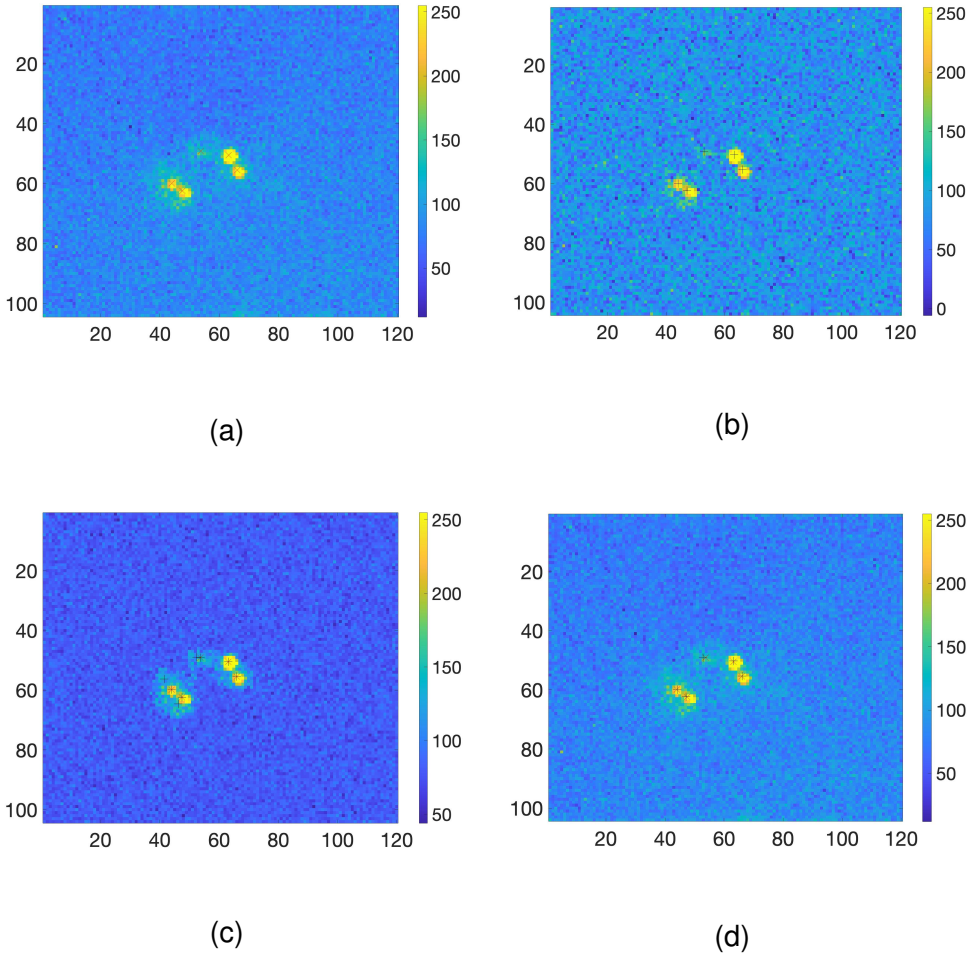


Figure 4.19: (a) Original SMLM image and the locations ThunderSTORM predicted. (b) Renoised image from Mixture Model and the locations predicted by ThunderSTORM on this new image. (c) Renoised image from MIM and the locations predicted by ThunderSTORM as well as the locations from the original image. (d) Original SMLM with both the locations calculated from Gibbs renoised image and the original image.

4.5.2 *Measuring Signal and Noise (signal-to-noise ratio (SNR))*

SNR is an important factor that quantifies the quality of images. If the images have a low SNR then there may be losses of important information. The definition and calculation of SNR varies widely

depending on the field and it is calculated in several different manners when using SMLM.

An equation used to calculate SNR is as follows:

$$\text{SNR} = \frac{I_o - I_b}{\sigma}, \tag{4.37}$$

where I_o is the object intensity, I_b is the background intensity and σ is a representative noise level. The value of σ varies and there are several different methods for calculating it. It can be calculated from the background root mean squared (RMS) [121], the noise over the object (σ_o) [122] and a mixture of the two $\sqrt{\sigma_b^2 + \sigma_o^2}$ [123].

Here, the equation was adapted as follows:

$$\text{SNR} = \frac{\mu_s - \mu_n}{\sigma_n} \tag{4.38}$$

where

$$\sigma_n = \frac{1}{\sqrt{\lambda_n}}. \tag{4.39}$$

μ_s, μ_n and λ_n are all calculated using the Mixture Model, and therefore it can be calculated with no user input.

4.5.3 *Localisation*

Finding the sections using this method and using a soft threshold rather than a hard threshold will enable localisation of fluorophores to occur. This could be used to help find the regions of interests (ROIs) needed for other localisation algorithms. It has many advantages as it requires no input (the prior values are pre-chosen) and using a soft threshold provides better images than using a hard threshold on the intensities of the images.

4.5.4 *Preprocessing - Counting*

The output of this algorithm is used chapter 5. Both the output of the Mixture Model and the MIM are used to try and quantify the amount of fluorophores within the sample.

4.6 DISCUSSION

A two-stage algorithm has been developed that successfully denoises images taken using SMLM with no user input. Both of these stages

create a matrix of probabilities. These show the likelihood that the pixel is in the signal population.

The first stage uses a Mixture Model as a means of Bayesian Inference to split the data into two different populations, signal and noise, on a pixel-wise manner. This can be used as a standalone or the probabilities output can be fed into the MIM.

Due to the nature of SMLM, the PSF of the emitting fluorophores is known to spread over several pixels. Therefore, as pixels are denoised independently of each other, there may be some incorrectly classified pixels. Isolated pixels that are classified as signal, surrounded by noise, have been incorrectly classified. The second section of the algorithm, the MIM, takes the output from the Mixture Model and incorporates a spatial dependence using nearest neighbours.

The Mixture Model calculates a lot of statistical information about the two different populations of the data: signal and noise. These two populations were taken to be Gaussian distributions and the mean and precision were determined for each. As a value for these is calculated at each iteration a distribution of the mean and precision are found and therefore the confidence in them can be determined. These values are all produced with no user input, using the priori and the data alone. Although a soft threshold is chosen by the user, the results are not sensitive to this value and 0.7 yields good results for most images.

Using a soft threshold on the probabilities from the Mixture Model, obtains better results than using a 'hard threshold' on the raw intensities in the original images. The intensities in SMLM images vary substantially both throughout a single stack and when imaging different samples. Therefore, a 'hard threshold' would have to be selected for each individual image. When using a soft threshold however, the same value can be used across a variety of different images and yield a consistent results with all of them.

The Mixture Model was put through a rigorous sensitivity test. The values of μ_i are the least robust out of all the unknown prior values, however more prior information is known about these. The values for all the other unknowns are all robust when using various values.

The Mixture Model, used alone, has several limitations. The data input into the Mixture Model needs to have enough data in each population otherwise the algorithm fails. Depending on the SNR, the images need between 1.38% and 4% of the pixels to be classified as signal or the algorithm fails. Images taken with SMLM should have more than this percentage especially as images are being taken at higher densities in order to reduce images times.

As stated earlier, each pixel is taken independently. This is not physically correct and therefore using the Mixture Model as a standalone may have some anomalies in the final result. Due to the nature of SMLM images, the signal data tends to congregate into 'islands' of signal surrounded by a sea of noise. The output from the Mixture Model has islands with rough edges and isolated pixels that are incorrectly classified. Feeding the probabilities output from the Mixture Model into the MIM helps to mitigate these limitations, helping to remove isolated pixels and smooth the edges of the islands.

The MIM incorporates a spatial structure into the method by using the four nearest neighbours. Any pixels without four neighbours: the outer most edges, are therefore ignored. For most images taken with SMLM, the main interest is in the centre, so little to no data is lost. As it is known that the PSF spreads over several pixels, isolated pixels classified as signal, and surrounded by pixels classified as noise, are incorrectly classified.

The MIM is less robust than the Mixture Model and the prior values have to be chosen with a lot of thought. The values chosen work for a variety of different types of images, and were determined from the results of the sensitivity test.

This method has many different and useful applications. It can reduce the amount of data needed to save the images, measure the SNR, find the areas of interest and is used as a preprocessing algorithm for counting in chapter 5. This is in addition to its ability to denoise data.

When using SMLM, a large stack of images are needed to localise all the fluorophores. This requires a lot of disk space to save the images. The method described in this chapter offers a way to denoise the images and save them in a manor that can save up to 95% of disk space. As the statistical properties about the noise are also determined and saved (μ_s and λ_s), the images can also be renoised at a later stage using only these two values. Most location algorithms use the noise when locating the fluorophores positions and therefore it needs to be added back in.

4.7 FURTHER WORK

Further work the could be completed on this work would be to combine the two stages of the code into one. This would make the method more efficient. Ways into speeding up the algorithm could also be looked into, such as using a graphics processing unit (GPU).

The Mixture Model models both populations as Gaussian distributions, however, it is known that the noise population is not of this type. This population could be updated to see if different populations yield better results.

4.8 THE ALGORITHM

The code that was written and used in this chapter can be found in appendix C.

The first section of this algorithm, the Mixture Model, was written in MATLAB by myself. The second section, the MIM was also written in MATLAB by myself, but adapted from code found here [124].

REFERENCES

[1] C. SINGER, 'The Dawn of Microscopical Discovery', Journal of the Royal Microscopical Society **35**, 317–340 (1915).

[2] S. Bradbury, *An Introduction to the Optical Microscope* (Oxford University Press, 1989), p. 5.

[3] G. Sines and Y. A. Sakellarakis, 'Lenses in Antiquity', American Journal of Archaeology **91**, 191 (1987).

[4] S. Bradbury, *The Evolution of the Microscope* (Pergamon Press, 1967).

[5] H. Gest, 'The discovery of microorganisms by Robert Hooke and Antoni van Leeuwenhoek, Fellows of The Royal Society', Notes and Records of the Royal Society **58**, 187–201 (2004).

[6] C. S. Ball, 'The early history of the compound microscope', Bios **37**, 51–60 (1966).

[7] D. Bardell, 'The First Record of Microscopic Observations', BioScience **33**, 36–38 (1983).

[8] R. Hooke, *Micrographia* (Royal Society, 1665).

[9] K. B. Wolf and G. Krötzsch, 'Geometry and dynamics in refracting systems', European Journal of Physics **16**, 14–20 (1995).

[10] M. Hausner, *Optics Inspections and Tests: A Guide for Optics Inspectors and Designers* (2017), pp. 7–32.

[11] A. Kwan, J. Dudley and E. Lantz, 'Who really discovered snell's law?', Physics World **15**, 64 (2002).

[12] R. D. Fiete, 'Optics', in *Modeling the imaging chain of digital cameras* (SPIE Press, 2010), pp. 49–72.

[13] L. Davis, C, *Thin Lenses*, http://www.physics.louisville.edu/cldavis/phys299/notes/lo_lense Accessed: 2020-14-04.

[14] S. J. Ling, J. Sanny and B. Moebs, *4.1: single-slit diffraction*, https://phys.libretexts.org/?title=T _Optics_and_Modern_Physics_(OpenStax)/4:_Diffraction/4.1:_Single-Slit_Diffraction%0A, Visited on 2018-05-23.

[15] G. B. Airy, 'On the Diffraction of an Object-glass with Circular Aperture', Transactions of the Cambridge Philosophical Society **5**, 283–291 (1835).

[16] *Greenfluorescentblog*, `https://greenfluorescentblog.wordpress.com/2012/04/01/numerical-aperture-and-resolution/`, Accessed: 2018-01-29.

[17]E. Abbe, 'Beitrage zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung', Arch. Mikroscop Anat **9**, 413–420 (1873).

[18]T. S. Tkaczyk, 'Numerical Aperture', in *Field guide to microscopy* (2010), p. 38.

[19]B. R. Masters, 'Optical Resolution and Resolving Power : What It Is , How to Measure It , and What Limits It', in *Confocal microscopy and multiphoton excitation microscopy: the genesis of live cell imaging* (SPIE Press, 2006), pp. 49–54.

[20]N. Foundation, *The nobel prize in chemistry*, 2014.

[21]S. Balaiya, R. K. Murthy, V. S. Brar and K. V. Chalam, 'Evaluation of ultraviolet light toxicity on cultured retinal pigment epithelial and retinal ganglion cells', Clinical Ophthalmology, 33–39 (2010).

[22]M. Abramowitz and M. W. Davidson, *Anatomy of a Microscope - Numerical Aperture and Resolution*, Visited on 2020-04-16.

[23]N. Foundation, *The nobel prize in physics*, 1929.

[24]W. J. Croft, *Under the microscope [electronic resource] : a brief history of microscopy* (World Scientific Publishing Co. Pte. Ltd., 2006), pp. 57–72.

[25]R. Erni, M. D. Rossell, C. Kisielowski and U. Dahmen, 'Atomic Resolution Imaging with a sub-50 pm Electron Probe', Physical Review Letters **102**, 96–101 (2009).

[26]N. Foundation, *The nobel prize in physics*, 1986.

[27]J. Ayache, L. Beaunier, J. Boumendil, G. Ehret and D. Laub, 'Artifacts in Transmission Electron Microscopy', in *Sample preparation handbook for transmission electron microscopy* (Springer, New York, NY, 2010), pp. 125–170.

[28]J. G. Mcnally, T. Karpova, J. Cooper and J. A. Conchello, 'Three-Dimensional Imaging by Deconvolution Microscopy', Methods **19**, 373–385 (1999).

[29]G. Sparacino, G. De Nicolao, G. Pillonetto and C. Cobelli, 'Deconvolution', in *Modelling methodology for physiology and medicine* (Elsevier, 2014), pp. 45–68.

[30]M. Minsky, 'Memoir on inventing the confocal scanning microscope', Scanning **10**, 128–138 (1988).

[31]M. Minsky, *Microscopy Apparatus*, 1961.

[32]M. Renz, 'Fluorescence microscopy-A historical and technical perspective', Cytometry Part A **83**, 767–779 (2013).

[33]H. Helmholtz, 'Die theoretischen Grenzen für die Leistungsfähigkeit der Mikroskope.', Annalen der Physik, 557–584 (1874).

[34]M. Muyskens and Ed Vitz, 'The Fluorescence of Lignum nephriticum: A Flash Back to the Past and a Simple Demonstration of Natural Substance Fluorescence', Journal of Chemical Education **83**, 765 (2006).

[35]G. G. Stokes, 'On the Change of Refrangibility of Light', Philosophical Transactions of the Royal Society of London **142**, 463–562 (1852).

[36]G. G. Stokes, 'On the Change of Refrangibility of Light. No. II', Philosophical Transactions of the Royal Society of London **143**, 385–396 (1853).

[37]L. J. Kricka and P. Fortina, 'Analytical ancestry: "firsts" in fluorescent labeling of nucleosides, nucleotides, and nucleic acids', Clinical Chemistry **55**, 670–683 (2009).

[38]D. J. S. Birch, Y. Chen and O. J. Rolinski, 'Fluorescence', in *Photonics, volume 4 : biomedical photonics, spectroscopy, and microscopy* (John Wiley & Sons, Incorporated, 2015), pp. 1–58.

[39]G. T. Dempsey, J. C. Vaughan, K. H. Chen, M. Bates and X. Zhuang, 'Evaluation of fluorophores for optimal performance in localization-based super-resolution imaging', Nature Methods **8**, 1027–1036 (2011).

[40]R. Smallman and A. Ngan, 'Characterization and Analysis', in *Modern physical metallurgy* (Elsevier, 2014), pp. 159–250.

[41]H. Siedentopf and R. Zsigmondy, 'Uber Sichtbarmachung und Größenbestimmung ultramikoskopischer Teilchen, mit besonderer Anwendung auf Goldrubingläser', Annalen der Physik **315**, 1–39 (1902).

[42]D. Axelrod, 'Cell-substrate contacts illuminated by total internal reflection fluorescence.', The Journal of Cell Biology **89**, 141–145 (1981).

[43]Y. Markaki and H. Harz, eds., *Light Microscopy*, Vol. 1563, Methods in Molecular Biology (Springer New York, New York, NY, 2017).

[44]B. O. Leung and K. C. Chou, 'Review of super-resolution fluorescence microscopy for biology.', Applied spectroscopy **65**, 967–80 (2011).

[45]S. W. Hell and J. Wichmann, 'Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy', Optics Letters **19**, 780 (1994).

[46]R. C. Dunn, 'Near-field scanning optical microscopy.', Chemical reviews **99**, 2891–928 (1999).

[47]M. G. L. Gustafsson, 'Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution', Proceedings of the National Academy of Sciences **102**, 13081–13086 (2005).

[48]E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz and H. F. Hess, 'Imaging intracellular fluorescent proteins at nanometer resolution.', Science (New York, N.Y.) **313**, 1642–5 (2006).

[49]M. Rust, M. Bates and X. Zhuang, 'Stochastic optical reconstruction microscopy (STORM) provides sub-diffraction-limit image resolution', Nature methods **3**, 793–795 (2006).

[50]S. T. Hess, T. P. K. Girirajan and M. D. Mason, 'Ultra-high resolution imaging by fluorescence photoactivation localization microscopy.', Biophysical journal **91**, 4258–72 (2006).

[51]T. Dertinger, R. Colyer, G. Iyer, S. Weiss and J. Enderlein, 'Fast, background-free, 3D super-resolution optical fluctuation imaging (SOFI).', Proceedings of the National Academy of Sciences of the United States of America **106**, 22287–22292 (2009).

[52]B. Huang, M. Bates and X. Zhuang, 'Super-Resolution Fluorescence Microscopy', Annual Review of Biochemistry **78**, 993–1016 (2009).

[53]D. Wildanger et al., 'Solid immersion facilitates fluorescence microscopy with nanometer resolution and sub-Ångström emitter localization', Advanced Materials **24**, 309–313 (2012).

[54]R. Heintzmann, T. M. Jovin and C. Cremer, 'Saturated patterned excitation microscopy—a concept for optical resolution improvement', Optical Society of America **19**, 1599–1609 (2002).

[55]E. H. Rego, L. Shao, J. J. Macklin, L. Winoto, G. A. Johansson, N. Kamps-Hughes, M. W. Davidson and M. G. Gustafsson, 'Nonlinear structured-illumination microscopy with a photoswitchable protein reveals cellular structures at 50-nm resolution', Proceedings of the National Academy of Sciences of the United States of America **109**, 13–15 (2012).

[56]E. Betzig, 'Proposed method for molecular optical imaging', Optics Letters **20**, 237 (1995).

[57]W. Moerner and L. Kador, 'Optical detection and spectroscopy of single molecules in a solid', Physical Review Letters **62**, 2535–2538 (1989).

[58]M. Orrit and J. Bernard, 'Single pentacene molecules detected by fluorescence excitation in a p-terphenyl crystal', Physical Review Letters **65**, 2716–2719 (1990).

[59]W. P. Ambrose and W. E. Moerner, 'Fluorescence spectroscopy and spectral diffusion of single impurity molecules in a crystal', Nature **349**, 225–227 (1991).

[60]R. M. Dickson, A. B. Cubitt, R. Y. Tsien and W. E. Moerner, 'On / off blinking and switching behaviour of single molecules of green fluorescent protein', **388**, 355–358 (1997).

[61]M. Heilemann, S. van de Linde, A. Mukherjee and M. Sauer, 'Super-resolution imaging with small organic fluorophores.', Angewandte Chemie (International ed. in English) **48**, 6903–8 (2009).

[62]M. Heilemann, S. van de Linde, M. Schüttpelz, R. Kasper, B. Seefeldt, A. Mukherjee, P. Tinnefeld and M. Sauer, 'Subdiffraction-resolution fluorescence imaging with conventional fluorescent probes.', Angewandte Chemie (International ed. in English) **47**, 6172–6 (2008).

[63]L. Stryer and R. P. Haugland, 'Energy transfer: a spectroscopic ruler.', Proceedings of the National Academy of Sciences **58**, 719–726 (1967).

[64]*Single-molecule localization microscopy: directory of smlm software*, http://bigwww.epfl.ch/smlm, Visited on 2020-06-19, 2017.

[65]P. D. Simonson, E. Rothenberg and P. R. Selvin, 'Single-molecule-based super-resolution images in the presence of multiple fluorophores.', Nano letters **11**, 5090–6 (2011).

[66]S. A. Jones, S.-H. Shim, J. He and X. Zhuang, 'Fast, three-dimensional super-resolution imaging of live cells', Nature Methods **8**, 499–505 (2011).

[67]A. Small and S. Stahlheber, 'Fluorophore localization algorithms for super-resolution microscopy.', Nature methods **11**, 267–79 (2014).

[68]A. V. Abraham, S. Ram, J. Chao, E. S. Ward and R. J. Ober, 'Quantitative study of single molecule location estimation techniques', Optics Express **17**, 23352 (2009).

[69]R. E. Thompson, D. R. Larson and W. W. Webb, 'Precise nanometer localization analysis for individual fluorescent probes.', Biophysical journal **82**, 2775–2783 (2002).

[70]S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice Hall, 1993).

[71]R. Henriques, M. Lelek, E. F. Fornasiero, F. Valtorta, C. Zimmer and M. M. Mhlanga, 'QuickPALM: 3D real-time photoactivation nanoscopy image processing in ImageJ.', Nature methods **7**, 339–40 (2010).

[72]R. Parthasarathy, 'Rapid, accurate particle tracking by calculation of radial symmetry centers', Nature Methods **9**, 724–726 (2012).

[73]A. J. Berglund, M. D. McMahon, J. J. McClelland and J. A. Liddle, 'Fast, bias-free algorithm for tracking single particles with variable size and shape', Optics Express **16**, 14064 (2008).

[74]H. Ma, F. Long, S. Zeng and Z.-L. Huang, 'Fast and precise algorithm based on maximum radial symmetry for single molecule localization', Optics Letters **37**, 2481 (2012).

[75]T. Quan, H. Zhu, X. Liu, Y. Liu, J. Ding, S. Zeng and Z.-L. Huang, 'High-density localization of active molecules using Structured Sparse Model and Bayesian Information Criterion', Optics Express **19**, 16963 (2011).

[76]Y. Wang, T. Quan, S. Zeng and Z.-L. Huang, 'PALMER: a method capable of parallel localization of multiple emitters for high-density localization microscopy', Optics Express **20**, 16039 (2012).

[77]F. Huang, S. L. Schwartz, J. M. Byars and K. a. Lidke, 'Simultaneous multiple-emitter fitting for single molecule super-resolution imaging.', Biomedical optics express **2**, 1377–1393 (2011).

[78]F. Huang et al., 'Video-rate nanoscopy using sCMOS camera-specific single-molecule localization algorithms.', Nature methods **10**, 653–8 (2013).

[79]S. J. Holden, S. Uphoff and A. N. Kapanidis, 'DAOSTORM: an algorithm for high-density super-resolution microscopy', Nature methods **8**, 279–280 (2011).

[80]D. R. Tobergte and S. Curtis, 'Daophot: a Computer Program for Crowded-Field Stellar Photometry (Psf Photometry)', Journal of Chemical Information and Modeling **53**, 1689–1699 (2013).

[81]L. B. Lucy, 'An iterative technique for the rectification of observed distributions', Astronomical Journal **79**, 745–754 (1974).

[82]W. Richardson, 'Bayesian-based iterative method of image restoration', Optical Society of America **62**, 55–59 (1972).

[83]L. Zhu, W. Zhang, D. Elnatan and B. Huang, 'Faster STORM using compressed sensing', Nature Methods **9**, 721–723 (2012).

[84]S. Cox, E. Rosten, J. Monypenny, T. Jovanovic-Talisman, D. T. Burnette, J. Lippincott-Schwartz, G. E. Jones and R. Heintzmann, 'Bayesian localization microscopy reveals nanoscale podosome dynamics.', Nature methods **9**, 195–200 (2012).

[85]S. Wolter, U. Endesfelder, S. van de Linde, M. Heilemann and M. Sauer, 'Measuring localization performance of super-resolution algorithms on very active samples.', Optics express **19**, 7020–33 (2011).

[86]S. Wang, J. R. Moffitt, G. T. Dempsey, X. S. Xie and X. Zhuang, 'Characterization and development of photoactivatable fluorescent proteins for single-molecule-based superresolution imaging', Proceedings of the National Academy of Sciences of the United States of America **111**, 8452–8457 (2014).

[87]N. Durisic, L. Laparra-Cuervo, Á. Sandoval-Álvarez, J. S. Borbely and M. Lakadamyali, 'Single-molecule evaluation of fluorescent protein photoactivation efficiency using an in vivo nanotemplate', Nature Methods **11**, 156–162 (2014).

[88]P. J. Green, 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', Biometrika **82**, 711 (1995).

[89]H. Jeffreys, *Theory of probability*, 3rd (Press, Oxford: University, 1961).

[90]L. Savage, *The foundations of Statistics* (New York: Wiley, 1972).

[91]B. De Finetti, *Theory of probability* (London: Wiley, 1974).

[92]R. Christensen, T. E. Hanson, W. Johnson, Branscum and Adam, *Bayesian Ideas and Data Analysis : An Introduction for Scientists and Statisticians*, 1st ed. (Taylor & Francis Group, 2010).

[93]S. Chib and E. Greenberg, 'Understanding the Metropolis-Hastings Algorithm', The American Statistician **49**, 327–335 (1995).

[94]M. Jesper, ed., *Spactial Statistics and Computational Methods* (Springer, 2003), pp. 1–13.

[95]K. I. Mortensen, L. S. Churchman, J. A. Spudich and H. Flyvbjerg, 'Optimized localization analysis for single-molecule tracking and super-resolution microscopy', Nature Methods **7**, 377–381 (2010).

[96]J. R. Janesick, *Photon Transfer* (SPIE.Digital Library, 2007).

[97]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, `http://www.deeplearningbook.org` (MIT Press, 2016).

[98]S. Mor-Yosef, A. Samueloff, B. Modan, D. Navot and J. G. Schenker, 'Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study.', Obstetrics and gynecology **75**, 944–7 (1990).

[99] V. Metsis, I. Androutsopoulos and G. Paliouras, 'Spam Filtering with Naive Bayes – Which Naive Bayes?', In CEAS (2006).

[100] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, http://www.deeplearningbook.org (MIT Press, 2016).

[101] *Neural Network Programming - Deep Learning with PyTorch*, https://deeplizard.com/learn/ Accessed: 2020-07-08.

[102] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', Nature **323**, 533–536 (1986).

[103] M. Torrisi, G. Pollastri and Q. Le, 'Deep learning methods in protein structure prediction', Computational and Structural Biotechnology Journal **18**, 1301–1310 (2020).

[104] *Single-molecule localization microscopy: collection of reference datasets*, http://bigwww.epfl.ch/smlm/datasets/index.html, Visited on 2020-06-19, 2017.

[105] H. Li and J. C. Vaughan, 'Switchable Fluorophores for Single-Molecule Localization Microscopy', Chemical Reviews **118**, 9412–9454 (2018).

[106] B. A. Griffin, S. R. Adams, R. Y. Tsien, B. A. Griffin, S. R. Adams and R. Y. Tsient, 'Specific Covalent Labeling of Recombinant Protein Molecules Inside Live Cells Published by : American Association for the Advancement of Science Stable URL : http://www.jstor.org/stable/2896025 REFERENCES Linked references are available on JSTOR for this', Science **281**, 269–272 (1998).

[107] L. W. Miller, Y. Cai, M. P. Sheetz and V. W. Cornish, 'In vivo protein labeling with trimethoprim conjugates: A flexible chemical tag', Nature Methods **2**, 255–257 (2005).

[108] A. Keppler, S. Gendreizig, T. Gronemeyer, H. Pick, H. Vogel and K. Johnsson, 'A general method for the covalent labeling of fusion proteins with small molecules in vivo', Nature Biotechnology **21**, 86–89 (2003).

[109] A. Gautier, A. Juillerat, C. Heinis, I. R. Corrêa, M. Kindermann, F. Beaufils and K. Johnsson, 'An Engineered Protein Tag for Multiprotein Labeling in Living Cells', Chemistry and Biology **15**, 128–136 (2008).

[110] G. V. Los et al., 'HaloTag: A novel protein labeling technology for cell imaging and protein analysis', ACS Chemical Biology **3**, 373–382 (2008).

[111] I. Y. Iourov, *Fluorescence In Situ Hybridization (FISH)*, edited by T. Liehr, Springer Protocols Handbooks (Springer Berlin Heidelberg, Berlin, Heidelberg, 2017), pp. 17–25.

[112]M. E. Brezinski, *Noise and system performance with td-oct and sd-oct* (Academic Press, Amsterdam ; 2006), pp. 175–195.

[113]T. Bushnell, *What is autofluorescence*, https://expert.cheekyscientist.com/what-is-autofluorescence/, Visited on 2020-06-09.

[114]B. Huang, W. Wang, M. Bates and X. Zhuang, 'Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy.', Science (New York, N.Y.) **319**, 810–3 (2008).

[115]M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych and G. M. Hagen, 'ThunderSTORM: A comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging', Bioinformatics **30**, 2389–2390 (2014).

[116]B. A. Cipra, 'An Introduction to the Ising Model', The American Mathematical Monthly **94**, 937–959 (1987).

[117]I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, 'Chapter 9 - Probabilistic methods', in *Data mining practical machine learning tools and techniques*, edited by I. H. Witten, E. Frank, M. A. Hall and C. J. B. T. D. M. (E. Pal (Morgan Kaufmann, 2017), pp. 335–416.

[118]B. A. Landman, I. Lyu, Y. Huo and A. J. Asman, 'Chapter 6 - Multiatlas segmentation', in *Handbook of medical image computing and computer assisted intervention*, edited by S. K. Zhou, D. Rueckert, G. B. T. H. o. M. I. C. Fichtinger and C. A. Intervention (Academic Press, 2020), pp. 137–164.

[119]Orchard, Peter, *Markov random field optimisation*, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0809/ORCHARD/, Accessed 2014-08-28.

[120]Q. Jackson and D. Landgrebe, 'Adaptive Bayesian contextual classification based on Markov random fields', Geoscience and Remote Sensing, ... **40**, 2454–2463 (2002).

[121]A. Sonnleitner, G. Schütz and T. Schmidt, 'Free Brownian Motion of Individual Lipid Molecules in Biomembranes', Biophysical Journal **77**, 2638–2642 (1999).

[122]M. K. Cheezum, W. F. Walker and W. H. Guilford, 'Quantitative comparison of algorithms for tracking single fluorescent particles', Biophysical Journal **81**, 2378–2388 (2001).

[123]U. Kubitscheck, O. Kückmann, T. Kues and R. Peters, 'Imaging and Tracking of Single GFP Molecules in Solution', Biophyiscal Journal **78**, 2170–2179 (2000).

[124]*Markov random fields*, https://www2.isye.gatech.edu/isyebayes/bank/handout16.pdf, Accessed August 11, 2022.

[125] *Classification Learner*, https://uk.mathworks.com/help/stats/classificationlearner-app.html, Accessed: 2020-08/-8.

[126] M. J. Rust, M. Bates and X. Zhuang, 'Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM).', Nature methods **3**, 793–5 (2006).

[127] R. P. Nieuwenhuizen, M. Bates, A. Szymborska, K. A. Lidke, B. Rieger and S. Stallinga, 'Quantitative localization microscopy: Effects of photophysics and labeling stoichiometry', PLoS ONE **10**, 1–18 (2015).

[128] D. Gross and W. Webb, 'Molecular counting of low-density lipoprotein particles as individuals and small clusters on cell surfaces', Biophysical Journal **49**, 901–911 (1986).

[129] B. M. Burton, K. A. Marquis, N. L. Sullivan, T. A. Rapoport and D. Z. Rudner, 'The ATPase SpoIIIE Transports DNA across Fused Septal Membranes during Sporulation in Bacillus subtilis', Cell **131**, 1301–1312 (2007).

[130] M. C. Leake, J. H. Chandler, G. H. Wadhams, F. Bai, R. M. Berry and J. P. Armitage, 'Stoichiometry and turnover in single, functioning membrane protein complexes', Nature **443**, 355–358 (2006).

[131] S. K. Das, M. Darshi, S. Cheley, M. I. Wallace and H. Bayley, 'Membrane protein stoichiometry determined from the step-wise photobleaching of dye-labelled subunits', ChemBioChem **8**, 994–999 (2007).

[132] K. Tsekouras, T. C. Custer, H. Jashnsaz, N. G. Walter and S. Pressé, 'A novel method to accurately locate and count large numbers of steps by photobleaching', Molecular Biology of the Cell **27**, edited by D. Lidke, 3601–3615 (2016).

[133] A. Lee, K. Tsekouras, C. Calderon, C. Bustamante and S. Pressé, 'Unraveling the Thousand Word Picture: An Introduction to Super-Resolution Data Analysis', Chemical Reviews **117**, 7276–7330 (2017).

[134] E. Alpaydin, 'Introduction', in *Introduction to machine learning* (2010), pp. 1–19.

[135] C. Campbell and Y. Ying, 'Learning with Support Vector Machines', Synthesis Lectures on Artificial Intelligence and Machine Learning **5**, 1–95 (2011).

[136] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (Cambridge University Press, Mar. 2000).

[137]V. Hodge and J. Austin, 'A Survey of Outlier Detection Methodologies', Artificial Intelligence Review **22**, 85–126 (2004).

[138]M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks', Neural Networks **106**, 249–259 (2018).

[139]L. Deng, 'The mnist database of handwritten digit images for machine learning research [best of the web]', IEEE Signal Processing Magazine **29**, 141–142 (2012).

[140]S. L. Smith, P.-j. Kindermans, C. Ying, Q. V. Le and G. Brain, 'D on ' T D Ecay the L Earning R Ate , I Ncrease the B Atch S Ize', 1–11 (2018).

[141]*Cross entropy loss*, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, Accessed April 02, 2021.

[142]Z. Zhang and M. R. Sabuncu, 'Generalized cross entropy loss for training deep neural networks with noisy labels', Advances in Neural Information Processing Systems **2018-December**, 8778–8788 (2018).

[143]*Cross-entropy loss function*, https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e, Accessed April 02, 2021.

5

COUNTING

5.1 SUMMARY

In this chapter several different approaches were used to try and count the number of fluorophores in the sample with varying degrees of success. All tests used simulated data produced by the algorithm outlined in chapter 3. The methods used were: a two staged Mixture Model, machine learning and neural network (NN).

The two-staged Mixture Model was used to split the signal population further into two new distributions; S_1 and S_2. Due to the nature of images taken with Single Molecule Localisation Microscopy (SMLM), there was insufficient data in a lot of the images to produce consistent results across a range of images.

Using consolidated results from the Mixture Model and inputting into MATLABS's Classification learner [125] to train various different models yielded promising results. These models obtained an accuracy of 87.2%.

Training data sets were created in order to train a NN. Four different training sets were used; the raw image data, the probabilities output from the Mixture Model and the Modified Ising Model (MIM), and the product of these two probabilities. The accuracy reached from these datasets was 88.6%, 89.3%, 90.8% and 89.9%, and loss values were 0.238, 0.213, 0.190 and 0.017 respectively.

The NN was adapted to take in 3-dimensional data; the previous, current, and following image. A new set of training data was created, and the four different sets were used as before; the raw image data, the probabilities output from the Mixture Model and the MIM, and the product of these two probabilities. The accuracy reached from these datasets was 40.4%, 42.4%, 40.4%, and 42.4% respectively. The loss values for these were 0.481, 19.8, 2.00, and 42.4 respectively.

5.2 INTRODUCTION

SMLM can be used for many different applications including; to determine the positions of a molecule within a sample; to find the stoichiometry of a molecule or to find the number of a specific structure of interest within the sample. All of these are related but there are different considerations depending on the final outcome required.

Finding the exact number of a specific structure of interest within a sample is extremely difficult and many different problems are encountered when trying to determine this. To count a specific structure, it is first labelled with a fluorophore. There are many different methods to attach fluorophores to a structure of interest. These are touched on briefly in section 3.3 and a more in depth description can be found in [105]. As described in section 1.7.1, fluorophores used in SMLM have two distinct states, usually a 'light' and a 'dark' state. This is exploited so that at any point during the imaging process there is a subset of fluorophores 'on' and emitting photons. The enables the diffraction limit to be circumvented and a higher precision of location to be found than in traditional light microcopy as the point spread functions (PSFs) (shown in figure 1.4) are less likely to overlap (see figure 1.12).

For this process to be optimal there would never be any fluorophores with overlapping PSFs 'on' at any one time. To insure this, sparse images can be taken by activating only a very small subset of the total number of fluorophores. However, this method requires a large number of images to be taken over a long period of time to ensure that all the fluorophores are activated at least once, allowing them to be localised. This requires a lot of image frames and the analysis would be computationally intensive.

One of main difficulties with counting the precise number of fluorophores and therefore structure of interest, is the associated noise. This is described in detail in section 3.4 and includes camera noise, background noise and noise within the signal. The noise within the signal makes it very difficult to count the number of fluorophores, especially in sections with overlapping PSFs. Due to the variability in the rate at which fluorophores emit photons (related to shot noise and the laser intensity) and the time in which they spend in the 'on' state per frame, it can be very difficult to determine how many are on in each frame. Two fluorophores with overlapping PSFs could be mistaken for a brighter single fluorophore.

Fluorophores can photobleach due to the lasers that are incident upon them. Once they have photobleached it is no longer possible

for a fluorophore to return to its 'on' state. Some fluorophores may have bleached before the imaging takes place which would lead to undercounting of the structure of interest.

One major difficulty with counting the number of fluorophores in a sample is that they can enter the 'on' state multiple times during the imaging process. For every frame that they are 'on', they could be localised as a separate emitter. The could ultimately look like a cluster of fluorophores rather than one single emitter.

As all transitions are dependent on quantum properties, some fluorophores may not enter the 'on' state throughout the entire imaging process. This will also lead to undercounting. This problem will become more prominent as acquisition times are reducing constantly.

As acquisition times are being reduced, the density of images is increasing in order to image all of the fluorophores in the sample. This could lead to undercounting as fluorophores with overlapping PSFs may be counted as one fluorophore.

Being able to count the number of molecules within a sample would be very useful. If the exact number of molecules is known then these molecules could be fit with Gaussians with a high precision. This would mean that the structure could be more easily studied and calculated.

5.3 BACKGROUND

Being able count the number of fluorophores can be very useful. Inferring the stoichiometry of structures can aid in the understanding of its operation.

Counting the number of fluorophores in order to determine the number of a specific structure of interest in a sample has different difficulties depending on many different factors. This includes but is not limited to, how the fluorophores are attached, what fluorophores are used and the laser intensity.

If the number of fluorophores in each images can be determined, then localising these fluorophores would be simplified. There would be less parameters to change, the x and y positions as well as the standard deviations of the 3D gaussian.

When using dye-labelled antibodies to attach the fluorophores, there arrises a new difficulty. Typically multiple antibodies attach to the target [126] and therefore, if the number of fluorophores could be counted with high accuracy, relating this to the number of structures is near impossible [127]. Therefore, when using SMLM in order to try

and count the number of a specific target in a sample, the labelling process needs to be tailored to try and ensure that there are no multiple bound fluorophores.

Several different methods have been used in order to count the number of fluorophores in either a sample, or in specific areas of the image. These include counting from the fluorescent intensity [128, 129] and photobleaching in diffraction-limited data [130–132].

Using the fluorescent intensity to try and count the number of fluorophores in a specific area can be done in several different manners. When using this technique, fluorophores do not need to be spatially resolved, they can be found in a cluster. The intensity of this cluster is calculated and divided by the estimated number of photons that would be collected by one emitter. This is known as the photon budget of a fluorophore. Unfortunately this photon budget varies due to excitation, local cellular conditions and possible fluorophore interaction. An early approach of this, using low density data, can be seen here [128]. Although this techniques is not used for SMLM the method is transferable.

Another method would be to create a calibration curve as seen in [129]. This accounts for the non-linearity of photons collected as the number of fluorophores increases. A linear relationship was seen for a small number of fluorophores, but the expected number decreased as the number of fluorophores increased. This could be to do with some fluorophore-fluorophore interaction or self quenching [129]. Generally this approach has low precision [133] and is not typically used.

Counting using photobleaching in a diffraction limited area relies on the stochastic bleaching of fluorophores. Das *et al.* [131] estimated that they could determine up to 15 photobleaching steps, depending on the signal-to-noise ratio (SNR), without the need for extrapolation or maximum likelihood estimate (MLE) estimation. They look at individual complexes where every subunit is labelled. The total number of photobleaching steps determines the number of subunits in the complex.

Counting photobleaching steps is a conceptually straight forward idea, however, inherent, and non-constant noise within the data makes it difficult to identify every step. The noise changes stochastically depending on the the number of fluorophores and therefore using a constant noise within a model would degrade the results. Tsekouras *et al.* [132] use a Bayesian approach in order to calculate the step number. When using data with high SNR they can count upwards of

100 fluorophores. With low SNR they can still count upwards of 50 fluorophores. They eliminate unlikely models and allow for blinking.

Counting the number of fluorophores in a pure SMLM dataset is very difficult. DAOSTORM [79] is an algorithm designed to analyse high density SMLM. This algorithm provides a more quantitative report on spatial distribution of fluorophores. It is based on an astronomy algorithm [80] and uses a fixed shape, multiple model PSF to localise the fluorophores. It has been shown to maintain high performance in high density images.

5.4 THE DATA USED

In order to know the ground truth detailing where the molecules are in each image, simulated data was used. The data was produced using the method described in chapter 3, however it will be summarised here.

The number of fluorophores are chosen and for each 100nm of time, it is modelled if each of these are 'on', 'off' or 'bleached'. A simplified model is used for the quantum properties to reduce the simulation time. Figure 1.13 is a Jablonski diagram showing the actual quantum steps that the fluorophores can take, and figure 3.2 shows the simplified Jablonksi diagram used in this model. The statistics have been altered to account for this variation.

These data are binned in time to calculate how long each fluorophore is 'on' for in each frame. This is then converted into a photon number using statistics.

An empty 'ground truth' matrix is created where each element represents a square nanometer area for each image. The position of each fluorophore can either be chosen manually or randomly. These are then taken to occupy one nanometer square and the photon number is added to the corresponding element in the matrix.

The PSF of the microscope is then added by using a Fourier transform, cutting out the high frequencies and then inverse fourier transforming back. This essentially blurs the images, changing the point emitters into refracted spots.

This data is then binned in space, where the size relates to the magnification of the microscope and the pixel size of the camera. Finally, noise is added to each pixel to create the final image. An example of one of these simulated images is shown in figure 5.1.

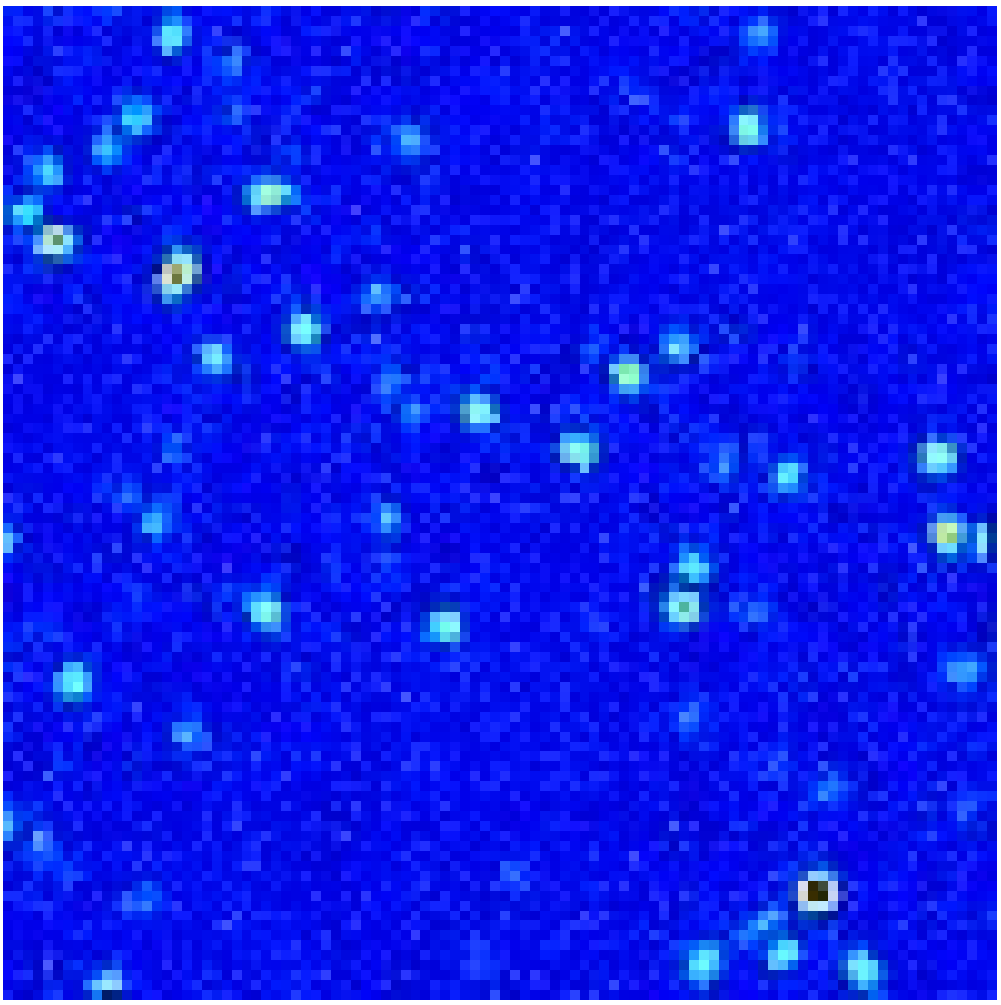


Figure 5.1: An example of a simulated image

5.5 APPROACHES/ METHODS

Several different methods were used to try and count the number of fluorophores. These include: two-staged Mixture Model processing, using the Mixture Model output, and several variations of Machine Learning (ML) [100]. All these different methods yielded varying degrees of success.

The final objective is to be able to count the number of fluorophores in high density images where the PSFs are overlapping each other. Low density images were used initially to check proof of concept.

5.5.1 *Two-Staged Mixture Model*

In SMLM images, the majority of pixels are noise, with 'islands' of signal pixels. It is unknown how many fluorophores there are in the sample in each of these 'islands'; there could be an individual fluorophore, or multiple emitters with overlapping PSFs.

Various images were created with different numbers of fluorophores. These simulated datasets were processed using the Mixture Model algorithm described in chapter 4. A 'soft threshold' was then used to split the data into signal and noise and create a mask were 0 represents noise and 1 represents signal. This mask shows all the 'islands' where the fluorophores are emitting.

To try and determine how many fluorophores were in each 'island' the signal data was passed to the Mixture Model a second time. This splits the data further and adds a third population. Before, there was noise (N), and signal (S), whereas now there is noise (N, as before), but there are two signal populations: S_1 and S_2, where S_1 typically has a higher intensity value than S_2.

It was hoped that the ratio in intensity between these two different populations, or the ratio of number of pixels in each 'island' could be used to discern if there were single or multiple fluorophores with overlapping PSFs. Unfortunately, in a majority of images, there was not enough data in the original signal population (S). and one of the limitations of the Mixture Model algorithm arises if there are not enough data points in either one of the populations. This is due to equation (4.18) and is described in section 4.3.9.

When there was enough datapoints in the signal population, generally small 'islands' were classified purely as S_2 and larger 'islands' had the centre as S_1 and the surrounding pixels as S_2. Unfortunately, due to the large variation in the number of photons emitted per frame when a fluorophore is 'on', this was not consistent. If every 'island' was split so there were some pixels in either population, then this could have been used to aid in finding a more precise location of the emitters, however this was not the case.

5.5.2 *Using the Mixture Model Output*

It was hypothesised that the Mixture Model could be used to observe steps in intensity as the fluorophores turn 'on' and 'off', and then model

how many fluorophores have switched states between neighbouring frames.

A set of fifty simulated images containing 150 fluorophores were created as described in chapter 3. Looking at the sum of the intensity across the whole image, figure 5.2, it can be seen that the intensity changes widely and steps in this data would not be sufficient enough to calculate the number of fluorophores in the system.

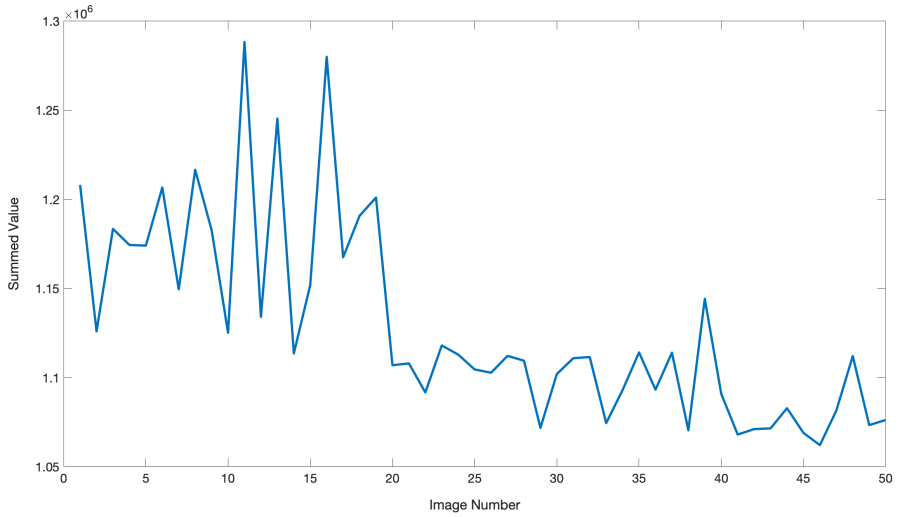


Figure 5.2: The sum of the intensity of entire simulated images through time. These images were simulated to have 150 fluorophores using the process described in chapter 3.

The simulated data were processed using the Mixture Model, which is described in detail in chapter 4. The output of this was used to denoise the images. A soft threshold was used so that pixels with a probability below this threshold are classed as noise and set to zero. Values of 0.7, 0.8 and 0.9 were used as the soft threshold for these images. The sum of these denoised images was then calculated and plotted against time. This is shown in figure 5.3. The step changes in intensity are now more visible and it is easier to determine where the fluorophores turn change state, but difficult to see the number of fluorophores.

As described in chapter 4, the output of the Mixture Model also includes statistics about the noise and signal distributions. These distributions are taken to be Gaussian and the mean and precision of these distributions are determined from the posterior. It is known that

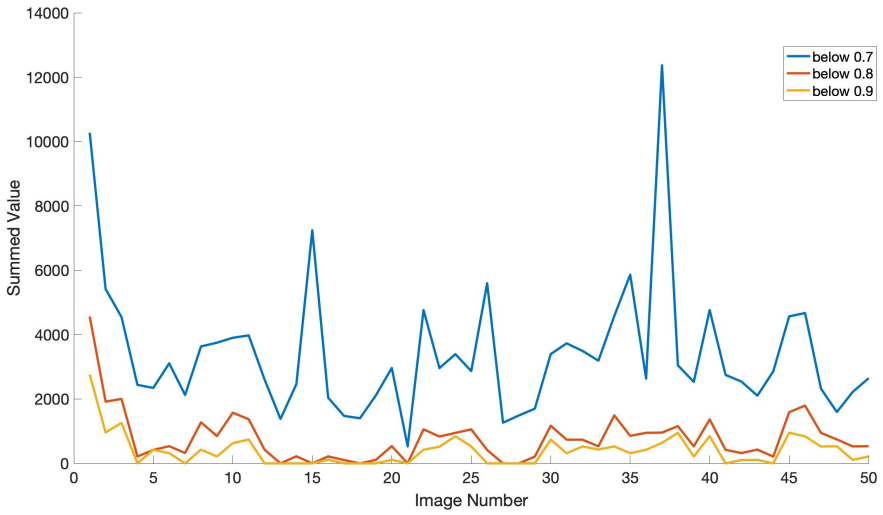


Figure 5.3: The sum of intensity of only the signal pixels of the entire images through time. These images were denoised using the Mixture Model code described in chapter 4 and a soft threshold of 0.7 (blue line), 0.8 (red line) and 0.9 (yellow line).

for one fluorophore, the number of photons at each pixel, X_i, is given by:

$$X_i \sim \mathcal{N}(\mu_s, \sigma_s^2) \tag{5.1}$$

where μ_s and σ_s^2 are the mean and variance of the signal. Therefore, each pixel will be given by

$$y_t \sim \mathcal{N}(n_t\mu_s, n_t\sigma_s^2) + noise \tag{5.2}$$

where $t = 1, ..., T$ and refers to the number of fluorophores present. This can then be used to estimate the change in the number of fluorophores that are 'on' between each frame using the following equations:

$$\frac{E_t[y_{t+1} - y_t]}{\hat{\mu}} = \frac{\mu(n_{t+1} - n_t)}{\hat{\mu}} \approx \Delta n. \tag{5.3}$$

and

$$\mu \approx \hat{\mu} \tag{5.4}$$

where μ is the actual mean of the distribution and $\hat{\mu}$ is the estimated value taken from the Mixture Model (μ_s). E_t is the expected value at time t.

The difference between the sum of the signal data was calculated and divided by μ_s in neighbouring images. As μ_s is calculated for each image, the mean of consecutive images was used. This value should then be Δn, the difference in the number of 'on' fluorophores between the two images. Figure 5.4a shows a plot of this value over a stack of 47 frames as well as the actual difference in the number of fluorophores that are 'on' in each frame. For the actual difference in the number of fluorophores that are 'on', there are three different values. The simulations split each frame into ten different sections. These plots show the difference when fluorophores are classed as 'on' when they are emitting for at least 0.3, 0.5 and 0.7 for each frame. There are also three different values for estimated change (Δn) in fluorophores. Using the precision of the signal population, λ_s, error bars can be calculated using the following equations.

$$\frac{E_t[y_{t+1} - y_t]}{\hat{\mu} + \sigma_s} = \frac{\mu(n_{t+1} - n_t)}{\hat{\mu} + \sigma_s} \approx \Delta_{min} n, \tag{5.5}$$

$$\frac{E_t[y_{t+1} - y_t]}{\hat{\mu} - \sigma_s} = \frac{\mu(n_{t+1} - n_t)}{\hat{\mu} - \sigma_s} \approx \Delta_{max} n, \tag{5.6}$$

where

$$\sigma_s = \frac{1}{\sqrt{\lambda_s}}. \tag{5.7}$$

Looking at figure 5.4a it can be seen that these error bars are very close to the value and therefore, for clarity, they will be omitted from future graphs. It can be seen that this value for the number of 'on' fluorophores is inconsistent with the actual number. Equation (5.3) is for individual pixels and several pixels contribute to one fluorophore related to the PSF of the microscope. Therefore the equation has been modified so that the PSF of the fluorophore is incorporated. Using the calculated mask the average area of the 'islands' was calculated. 'Islands' of only one or two pixels were discarded. Equation (5.3) is therefore adapted to become:

$$\frac{E_t[y_{t+1} - y_t]}{\hat{\mu} \text{ x mean PSF area}} = \frac{\mu(n_{t+1} - n_t)}{\hat{\mu} \text{ x mean PSF area}} \approx \Delta n. \tag{5.8}$$

In this particular stack of images, the average PSF size was calculated to be 9.797 pixels. Figure 5.4b shows the results from this equation along with the actual number of fluorophores that are 'on'. In this image, a fluorophore is classified as 'on' if it is emitting for more than half of the frame during the simulation process (see chapter 3 for more detail).

The difference between the actual and the expected values for the number of 'on' fluorophores is shown in figure 5.4c. The mean of the absolute difference of these is calculated to be 9.95.

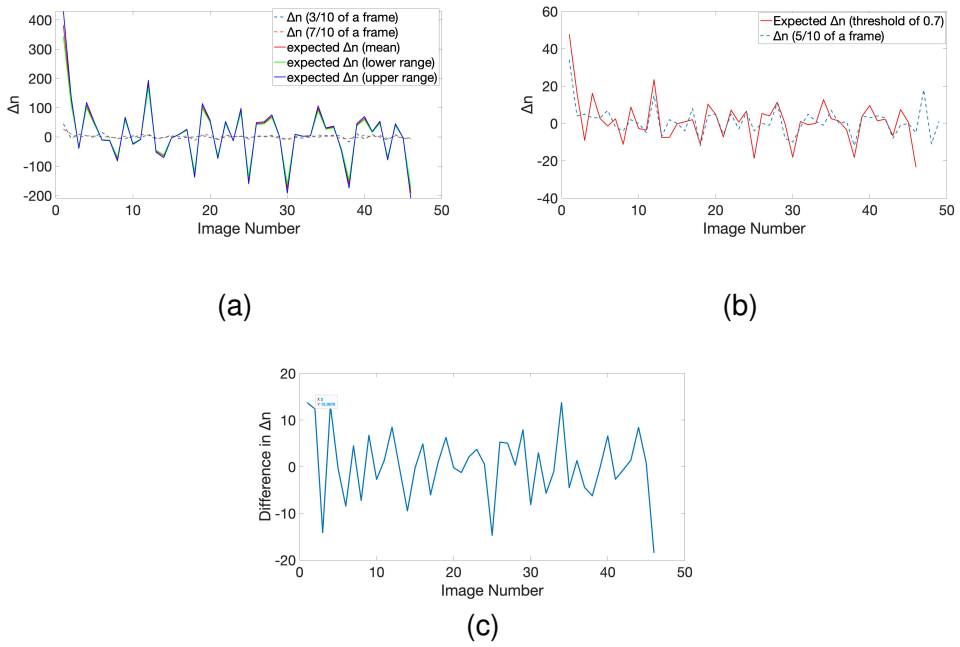
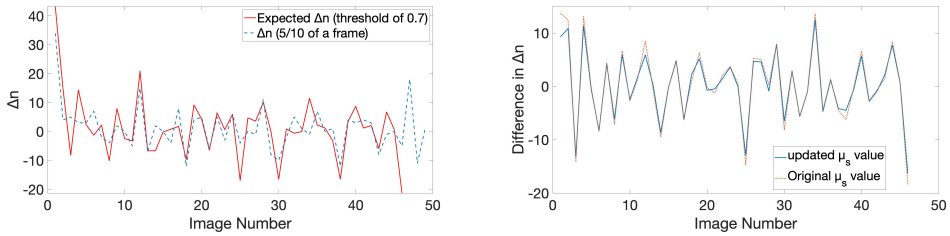


Figure 5.4: (a) The expected change in number of fluorophores (solid red line) and the upper (solid blue line) and lower (solid green line) limits. The actual change in the number of fluorophores is shown by the two dashed lines. These show the actual change when the fluorophore is simulated as 'on' when emitting for more than 0.3 (dashed blue line) and 0.7 of the frame (dashed orange line). (b) The expected change in the number of fluorophores when incorporating the size of the PSF and a soft threshold of 0.7 (red solid line). The actual change in the number of fluorophores is also shown (dashed blue line). Here, a fluorophore is classed as 'on' if it emits for at least half of a frame. (c) The difference between the actual and the expected change in fluorophores for the values shown in (b).

In order to try and improve upon this model, the values of μ_s and λ_s were updated using the process explained in section 4.3.6. A soft

threshold of 0.7 was used to split the data into fixed populations. These fixed populations are sent through a Mixture Model to give more accurate values of μ_s and λ_s. These updated values of μ_s and λ_s were then used to recalculate Δn. This is shown in figure 5.5.

Figure 5.5a shows the results from equation (5.8) for the updated values of μ_s and λ_s and the difference between the actual number of fluorophores. Here, the fluorophores are classified as 'on' if it is emitting for at least half the frame during the simulation process. Figure 5.5b shows the difference between the number of fluorophores that are 'on' and the expected difference calculated for both the original and updated values of mu_s and λ_s. The mean of the absolute difference for the absolute values of these is calculated to be 4.89, showing that updating the values using fixed populations increases the accuracy of the results.



(a) (b)

Figure 5.5: (a) The expected change in the number of fluorophores when incorporating the size of the PSF, a soft threshold of 0.7 and the updated value of μ_s (red solid line). The actual change in the number of fluorophores is also shown (dashed blue line). Here, a fluorophore is classed as 'on' if it emits for at least half of a frame. (b) The difference between the actual and the expected change in fluorophores for both the original value (blue solid line) and updated value (dotted red line) of μ_s.

5.5.3 *Machine Learning (ML)*

An algorithm is a set of instructions that a computer uses to solve a specific problem. It takes an input and performs computations sequentially to create an output. For some tasks an algorithm cannot be produced, and therefore different methods are required. ML is a study of computer algorithms that computes a model based on 'training data'. [134]

ML is a subset of artificial intelligence that completes a task without being explicitly told how to. It optimises parameters using 'training data' and can make predictions about the future, gain knowledge about the data, or both of these. [134]

ML can be used for learning associations, unsupervised learning, reinforcement learning and supervised learning in the case of classifications and regression [134]. Here, supervised learning is used for classification.

Supervised learning uses a set of training data and the desired outputs. It trains a model to try and compute the desired output from input data.

Matlab Classification Learner

Matlab includes a classification learner application [125] which can train models to classify data into groups. Different supervised ML can be explored to determine which classification model type is best for your data. The different models include support vector machines (SVM) [135, 136] and K-nearest neighbours (KNN) [137].

Simulated data was created using the algorithm described in chapter 3. Here 50 images were produced with 150 fluorophores. These were processed using the Mixture Model described in chapter 4 and a soft threshold of 0.7 was used to denoise the images. Any 'islands' that were smaller than three pixels were disregarded and taken as noise as the PSF is known to be greater than this.

Data relating to the 'islands' of signal groups was input to the MATLAB classification learner to determine if the number of fluorophores emitting in each island can be accurately counted.

Firstly the area of the 'islands', the sum and standard deviation of the intensity of the pixels in the 'islands' were input into the classification learner. Next, the area of the 'islands', the sum of the intensity of the pixels in the current image, the previous and the next frame were input into the classification learner.

Care had to be taken as the majority of the regions of interest (ROI) contained only one fluorophore. Therefore if the model classified all areas as containing only one fluorophores then the accuracy percentage would be very high (81.1%). Using a balanced training data set would help to rectify this in the future [138].

Area, Intensity, Standard Deviation

The area of the 'islands', the sum and standard deviation of the intensity of the pixels in the 'islands' were input into the classification learner and trained on all the various classification models. There were two different models that yielded the two highest results of 87.7% and 87.5% accuracy; A KNN classification and a SVM classification respectively.

A KNN classification uses the K-nearest neighbours to categorise query points. It can also incorporate a distance dependence. It uses these to find clusters within the data. [137] The number of neighbours used in this classification was 10 with no weight on the distance.

The data from this is shown in figure 5.6. Figure 5.6a shows a scatter plot of the area size against the standard deviation of the intensities, figure 5.6b shows the area size against the sum of the intensities and figure 5.6c shows the sum of intensities agains the standard deviation of intensities. The points in all of these graphs show the model predictions with dots representing correct predictions and crosses being incorrect. The colours correspond to the different predictions; orange for one, yellow for two and purple for three. Figure 5.6d shows a confusion matrix of the true classes against the predicted classes.

When held out data was used on this model (a KNN classification), the accuracy was 87.3%. This was using data of length 469. The majority of errors were only out by one fluorophore, however there were three with two fluorophores difference. This is very similar to the accuracy found from the training data.

An SVM is an abstract learning machine that attempts to generalise and make predictions using training data. It intuitively introduces directed hyperplanes that separate the training datas features to the corresponding labels. The points closest to these hyperplanes are known as support vectors. These impact the most influence on the position of the hyperplane. If the data cannot be linearly separated then kernels can be used to map the data to feature space. The directed hyperplanes are then used in this space. [135]

Here, a Gaussian kernel was used to map the data into feature space. The results are shown in figure 5.7. This figure shows the same information as that in figure 5.6 but for the new SVM model. The accuracy of this model on the training data was 87.3%.

When held out data was used on this model (a SVM model), an accuracy of 85.3% was achieved. This was using data of length 469. The majority of these errors were only one fluorophore out, however there

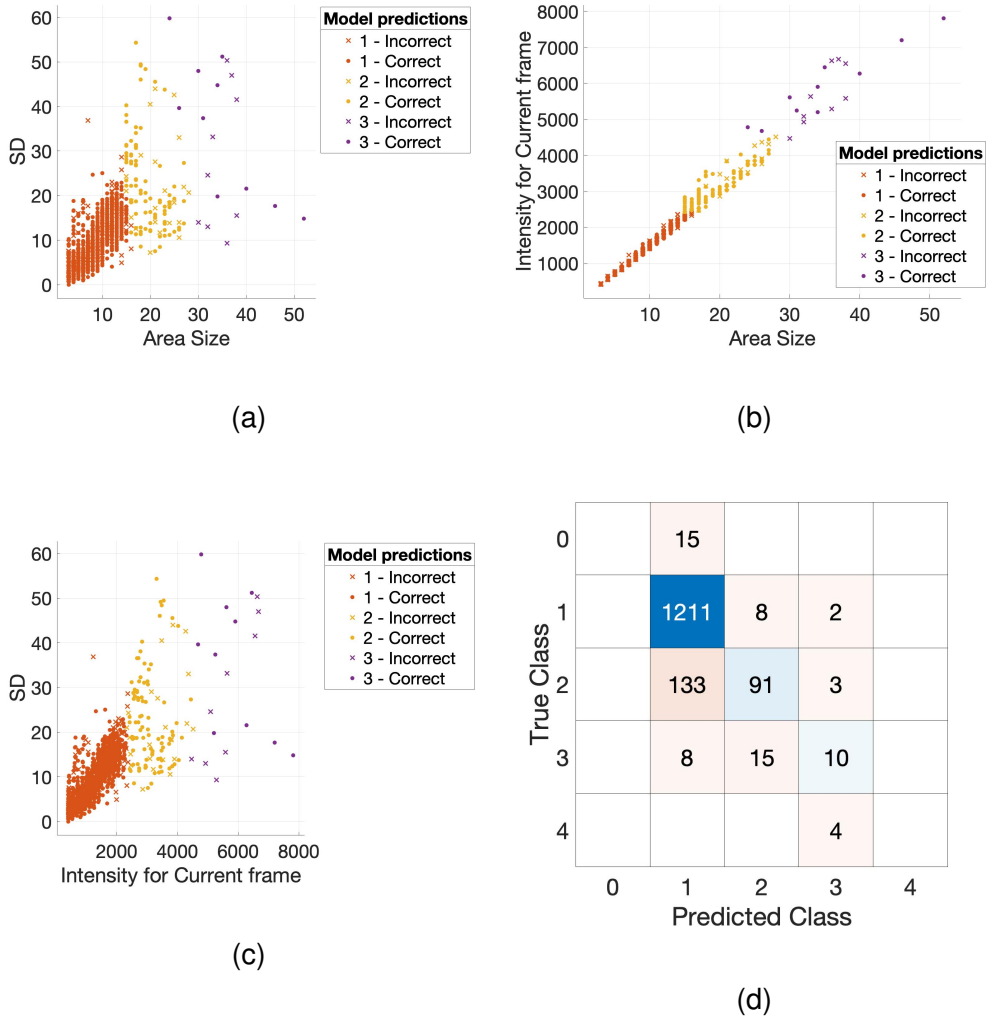


Figure 5.6: KNN classification using 10 nearest neighbours. (a) shows a scatter plot of the area size against the standard deviation of the intensities, (b) shows the Area size against the sum of the intensities and (b) shows the sum of intensities agains the standard deviation of intensities. The points in all of these graphs show the model predictions with dots representing correct predictions and crosses being incorrect. The colours correspond to the different predictions; orange for one, yellow for two and purple for three. (d) Shows a confusion matrix of the true classes against the predicted classes.

were three that were two out. This is slightly less than the accuracy of the KNN model that was created before.

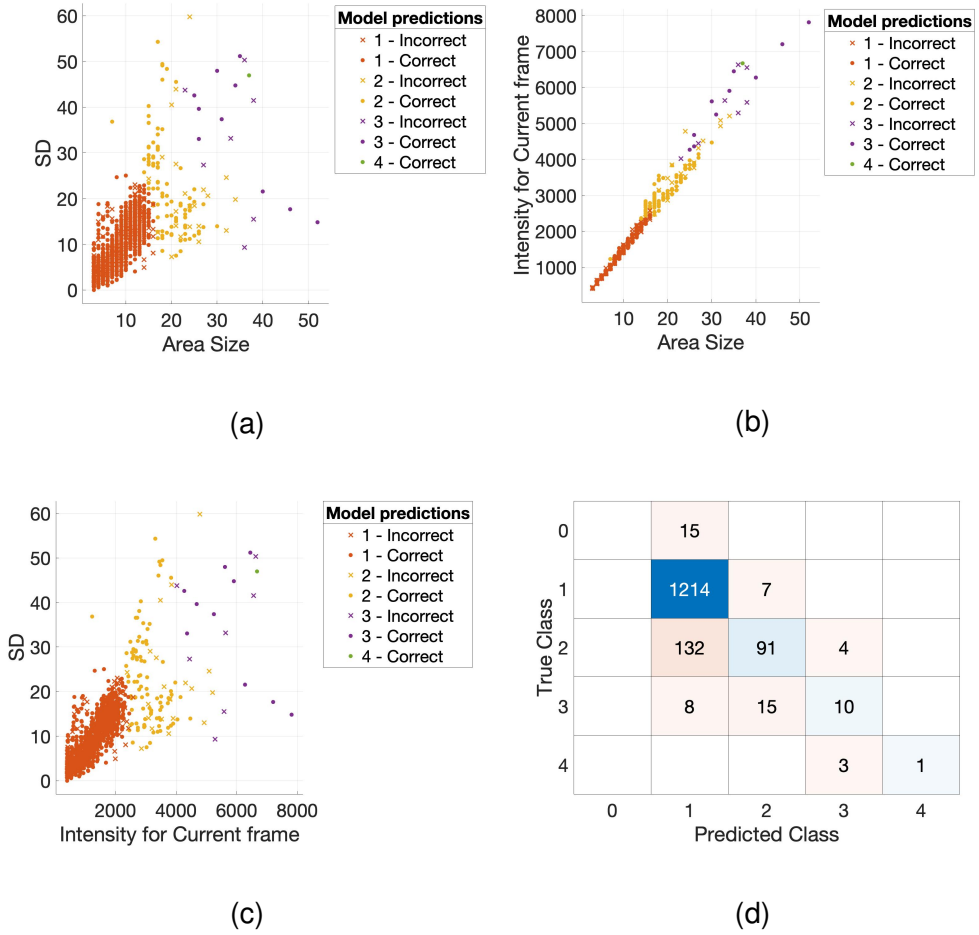


Figure 5.7: SVM classification. (a) shows a scatter plot of the area size against the standard deviation of the intensities, (b) shows the Area size against the sum of the intensities and (b) shows the sum of intensities agains the standard deviation of intensities. The points in all of these graphs show the model predictions with dots representing correct predictions and crosses being incorrect. The colours correspond to the different predictions; orange for one, yellow for two and purple for three. (d) Shows a confusion matrix of the true classes against the predicted classes.

Area, Intensity before, during and after

Using the size of the area, and the sum of the pixel intensities in the current, previous and next image yielded an accuracy percentage of 87.7%. This is a small increase in the percentage error from the SVM model, and the same as the results from the KNN model. The best

model here was a decision tree which is shown in figure 5.8a and the confusion matrix for this classification shown in figure 5.8b

Although this yields a higher percentage than the result from the SVM model, and the same result as the KNN model, it is thought that this type of classification for determining the number of fluorophores in an area is too coarse and therefore should not be used. It would not be able to be used for different sets of data. Looking at figure 5.8a it is clear that this method should not be used. It can only classify an 'island' to have 1, 2 or 3 fluorophores. It only uses the sum of the current and next images intensities.

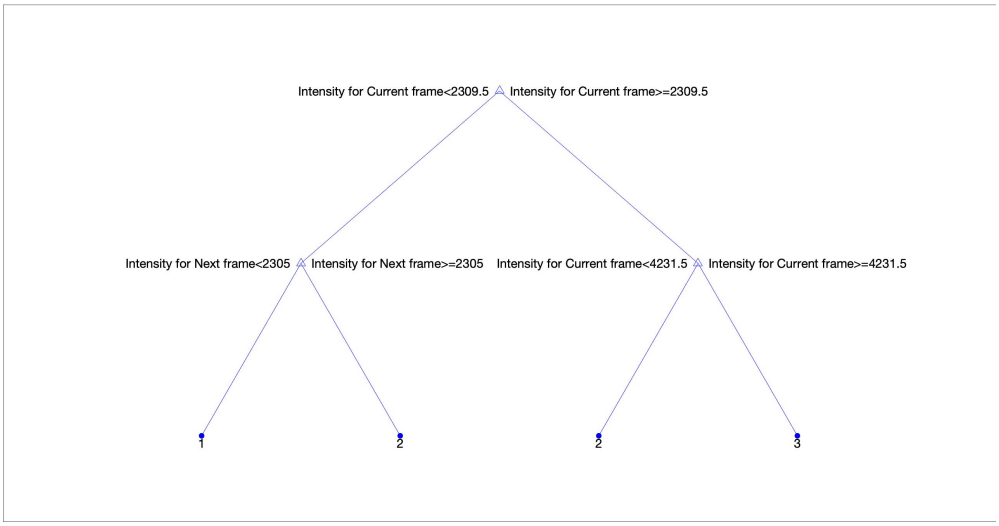
When using held out data for this decision tree, an accuracy of 86.6% was found. 343 data were put through the trained model, the majority of errors were only off by one fluorophore, however there was one that was off by two. Although for this data, it seems like the decision tree is reliable, due to its course data, I do not believe that this can be used for real data, especially due to the variability of intensities across both stacks of images, and between stacks. This is also a lower accuracy than the two previous models (the KNN and SVM) with the different data inputs (the size, intensity, and standard deviation).

Looking at the confusion matrices for all of these different models; KNN, SVM and the decision tree, figures 5.6d, 5.7d and 5.8b respectively, it can be seen that the KNN and SVM models compute very similar results, producing better results on the lower predictions. The decision tree model is the most different and produces better results for higher predictions.

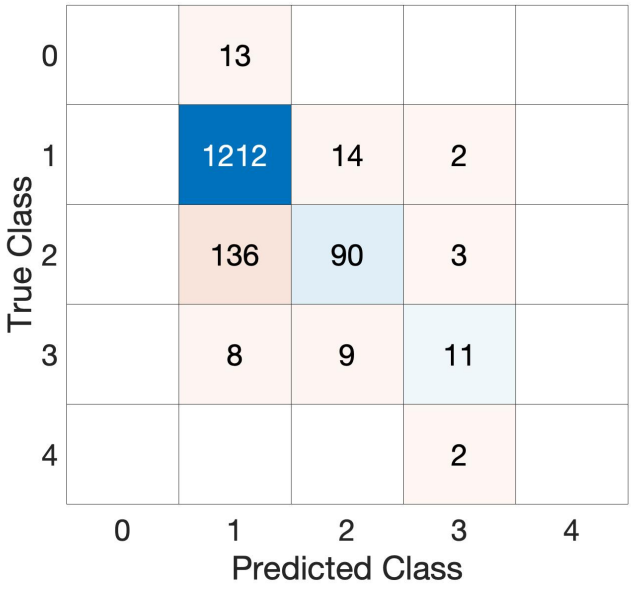
Using the ML system and just taking some consolidating results from the Mixture Model was promising. However, I believe that a model will need to have a success rate of around 90% or above for it to be useable to a high enough accuracy. Also, the course nature of the data included in this model means that it would need a lot more training to be able to treat this as a viable model. These models also seem too course to be able to differentiate the wide variety of different images that come in due to there being different fluorophores with different quantum yield (QY) and due to bleaching effects.

5.5.4 *Neural networks (NNs)*

Different data sets were used to train a NN. Simulated data was created with either 0, 1, 2, 3, 4, or 5 emitting molecules in. These are small areas aiming to imitate a 'cut-out' section of a full image.



(a)



(b)

Figure 5.8: (a) The decision tree created by the Classification learner in MATLAB when the input was area size, and the sum of the intensities for the current, previous and next image. (b) A confusion matrix of the true classes against the predicted classes.

As only the signal population is relevant to the neural network, it was proposed that the Mixture Model or MIM would highlight the valuable data, allowing the NN to be trained more quickly and/or more accurately. Therefore the different datasets used to train the NN

were; the original simulated data, the output of the Mixture Model, the output of the MIM, and the product of the results from both the Mixture Model and the MIM.

The Training Data

A NN was written and trained to take image sections of size 32x32 pixels and determine how many molecules are 'on' in that section. Images were created that contained either 0, 1, 2, 3, 4, or 5 emitting fluorophores in. A balanced dataset was created, ensuring that there were an equal number of images attached to each label. 1000 of each type of image were created. This NN was written using PyTorch [101].

As a sixth of the images contain zero fluorophores emitting in them, they could not be individually sent through the Mixture Model (see section 4.3.9 for more detail). Therefore the data was amalgamated into larger images and all different image types were processed simultaneously. This ensured that there was enough data in the signal population and the algorithm did not fail. These were partitioned back into data of the original size, 32x32 pixels, and used as a second set of data.

The probabilities from the Mixture Model were then processed using the MIM and these probabilities were used as a third data set (see section 4.4 for more detail). Finally the probabilities from both the Mixture Model and the MIM were multiplied together to create a fourth data set.

Figures 5.9 to 5.14 show example frames for the training data used. These figures show examples of 0-5 'on' fluorophores, respectively. They contain the (a) the raw data, (b) the probabilities output from the Mixture Model, (c) the probabilities from the MIM and (d) the probabilities from both the Mixture Model and the MIM multiplied together.

For the NN, a classification model similar to one used for the modified national institute of standards and technology (MNIST) [139] was used. This is a database used for handwriting recognition that is commonly studied using ML. These problems are similar; an image is trying to be classified into groups. In MNIST, the output are letters and numbers, here, the output is a number of fluorophores.

The neural network (NN) created

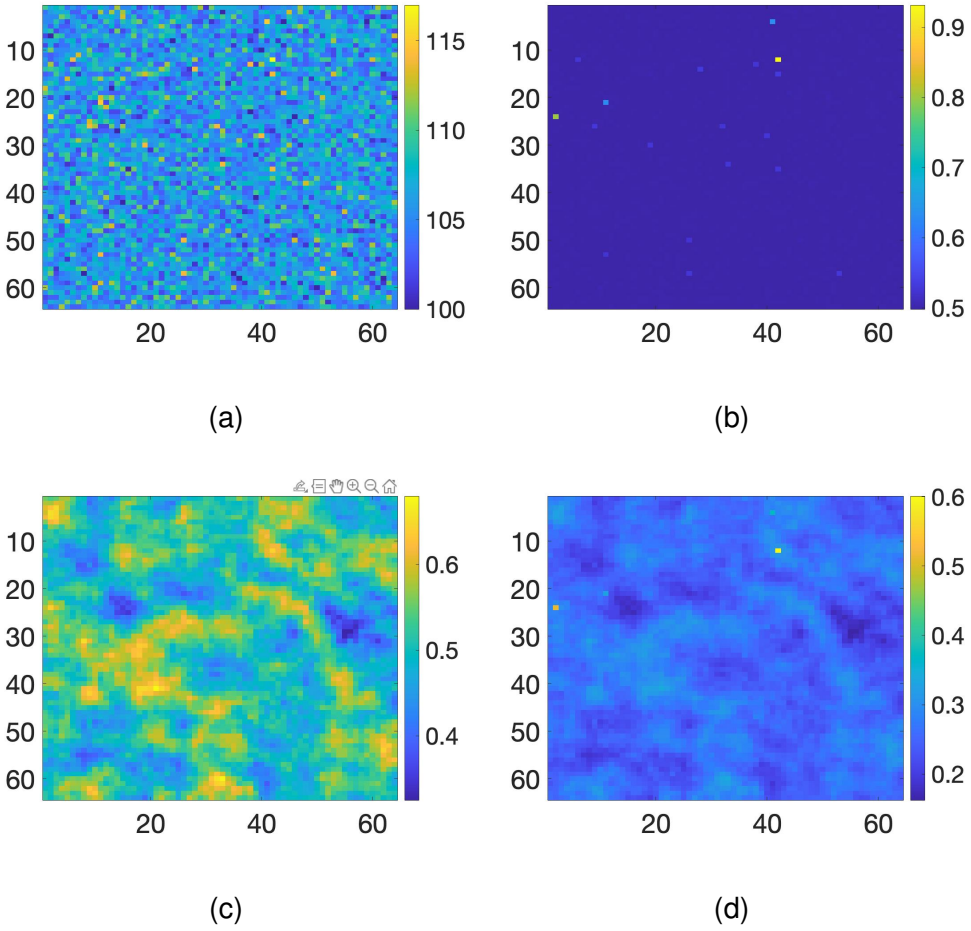


Figure 5.9: Examples of the input data to the NN for an image with no fluorophores emitting. (a) Shows the raw simulated image, (b) after it has been processed by the Mixture Model, (c) after it has been processed with the MIM and (d) is (b) and (c) multiplied together.

A NN with five hidden layers was created. As there are no cycles in the NN it is known as a feedforward neural network (FNN). These layers comprise of two convolution layers and three linear layers. As there are convolution layers it is also known as a convolutional neural network (CNN). The input layer takes the data from each image. The output layer comprises of the five values the NN can predict. For the rest of this chapter the 'C' and 'F' will be omitted, and it will be referred to as a NN (rather than either a FNN or CNN).

As described in section 2.5, input data is manipulated by the NN to predict how many fluorophores are emitting in each image. Each hidden layer in the NN contains specific weight tensors that manipulate

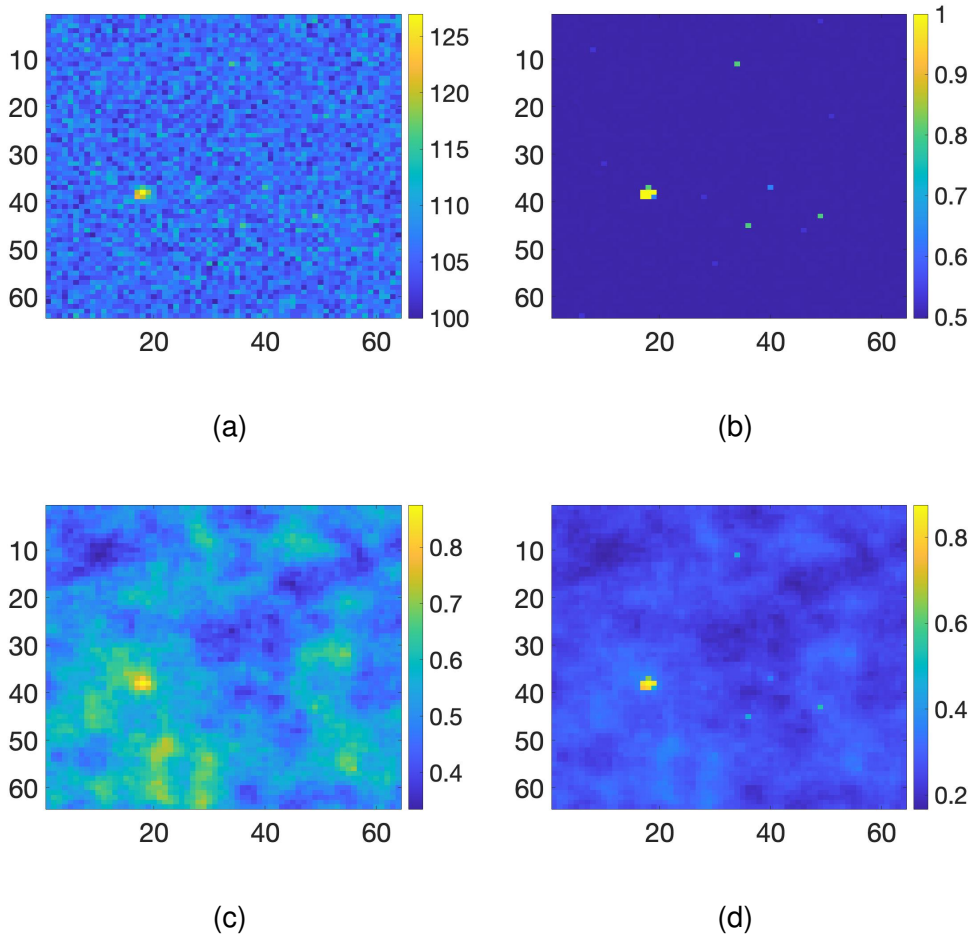


Figure 5.10: Examples of the input data to the NN for an image with one 'on' fluorophore. (a) Shows the raw simulated image, (b) after it has been processed by the Mixture Model, (c) after it has been processed with the MIM and (d) is (b) and (c) multiplied together.

the data. As the NN learns from the data these weights are updated to minimise the loss function. The loss function is related to the difference between the predicted outputs and the true labels. Here a cross-entropy function is used.

The data goes through the NN in batches. An epoch is defined as when all the data has been processed. The larger the batch size the quicker each epoch is completed. However, the unit being used may not have the processing power to compute all of the images simultaneously. Also, if the batch size is too large, the model will degrade and not be as accurate. A large batch size will lead to a poor generalisation, however it is not currently known why [140]. It can also cause instabilities in

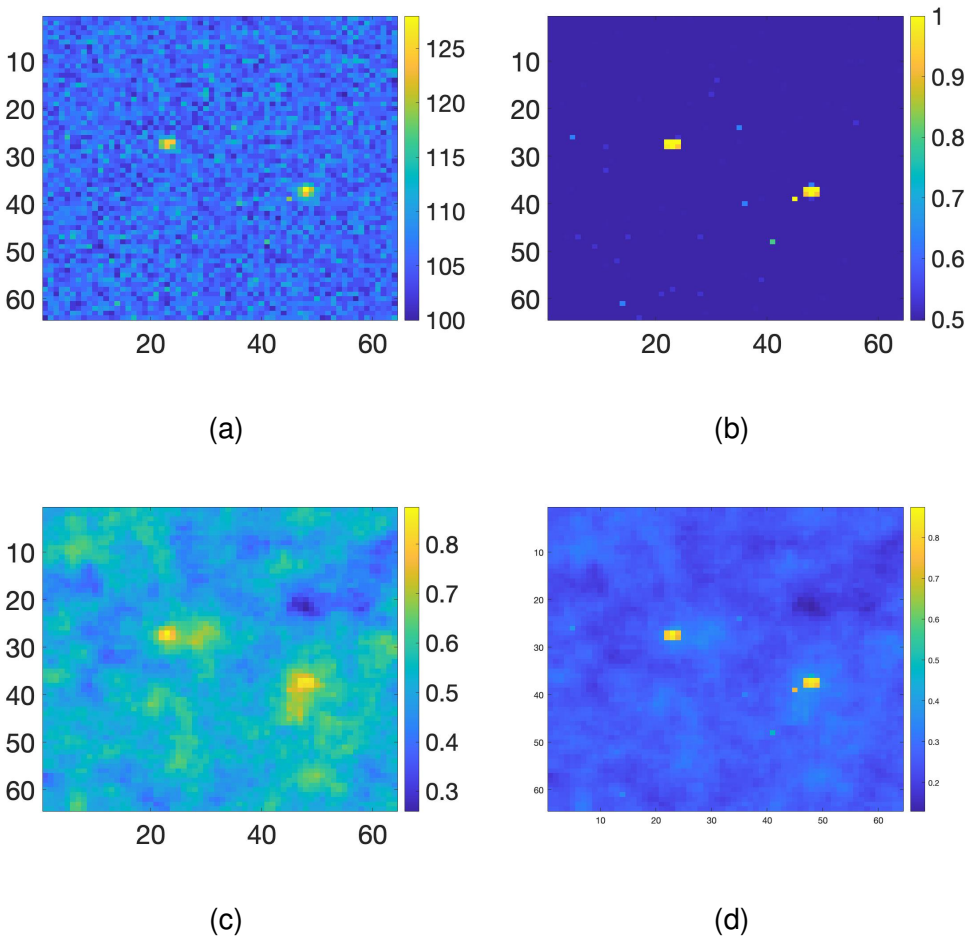


Figure 5.11: Examples of the input data to the NN for an image with two fluorophores emitting. (a) Shows the raw simulated image, (b) after it has been processed by the Mixture Model, (c) after it has been processed with the MIM and (d) is (b) and (c) multiplied together.

the early stages of training. It will also take longer to run and may not be able to process new data. Various different batch sizes were used when training this NN.

The two convolution layers have a kernel size of five, and therefore perform a convolution on the input data using a matrix of size 5x5. This is completed six times so the weight matrix for each convolution matrix can be thought of as size 5x5x6. The values of these matrices are updated in order to minimise the loss function at the end of each batch.

After the two convolution layers there are three linear layers. The data is first converted into a 1-dimensional array and then matrix

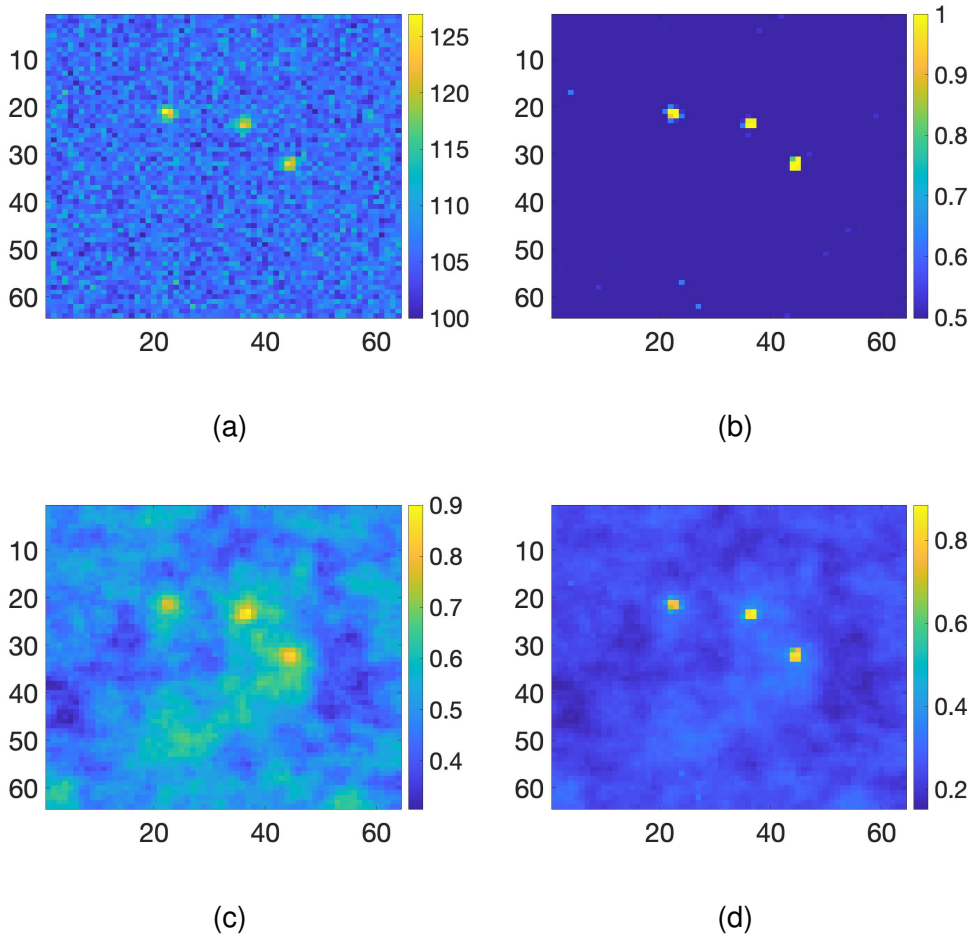


Figure 5.12: Examples of the input data to the NN for an image with three fluorophores emitting. (a) Shows the raw simulated image, (b) after it has been processed by the Mixture Model, (c) after it has been processed with the MIM and (d) is (b) and (c) multiplied together.

multiplication is conducted using new weight matrices. These work to reduce the data by specified amounts defined by the user. The final linear layer reduces the data down to five neurons, which correspond to the number of molecules that are on: 0, 1, 2, 3, 4 or 5. A value for each of these numbers is computed, and the largest is taken to be the predicted value. For example, if the computed values were: 0.1, 0.1, 0.2, 0.3, 0.1, 0.1 for the labels 0-5 respectively, the NN would predict that three fluorophores are emitting in this image as it has the highest probability.

As stated before, the values in all of the weight matrices are updated sequentially in order to minimise the loss function. The rate by which

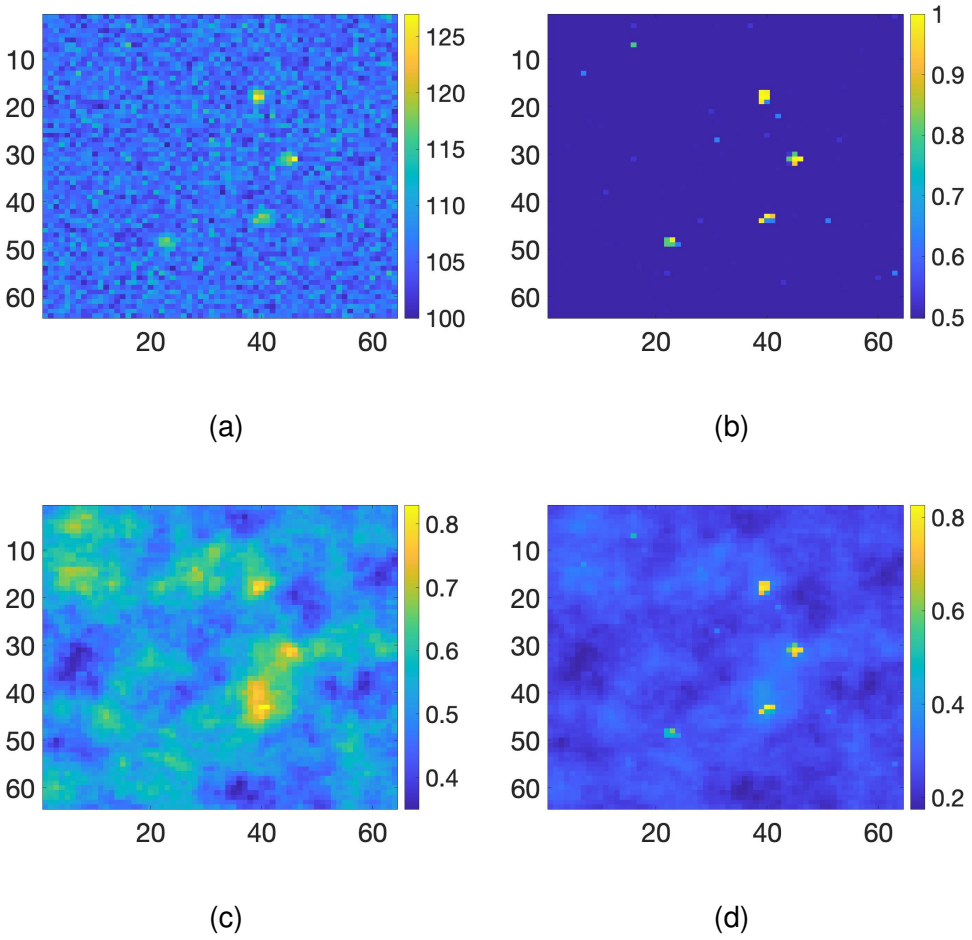


Figure 5.13: Examples of the input data to the NN for an image with four fluorophores emitting. (a) Shows the raw simulated image, (b) after it has been processed by the Mixture Model, (c) after it has been processed with the MIM and (d) is (b) and (c) multiplied together.

they change is related to the learning ratio, which is chosen by the user. Many different learning ratios were used when training this data to determine which value would optimise the NN. The loss function is calculated at the end of each batch. The gradient for each weight, with respect to the loss function is calculated and multiplied by the learning ratio. The weights are then updated by subtracting this value from the previous value.

The learning ratio is chosen by the user. A compromise has to be chosen when selecting this. The smaller the learning ratio, the longer the NN will take to learn. As well as this, the algorithm is more likely to settle on an incorrect local maximum rather than the global maximum.

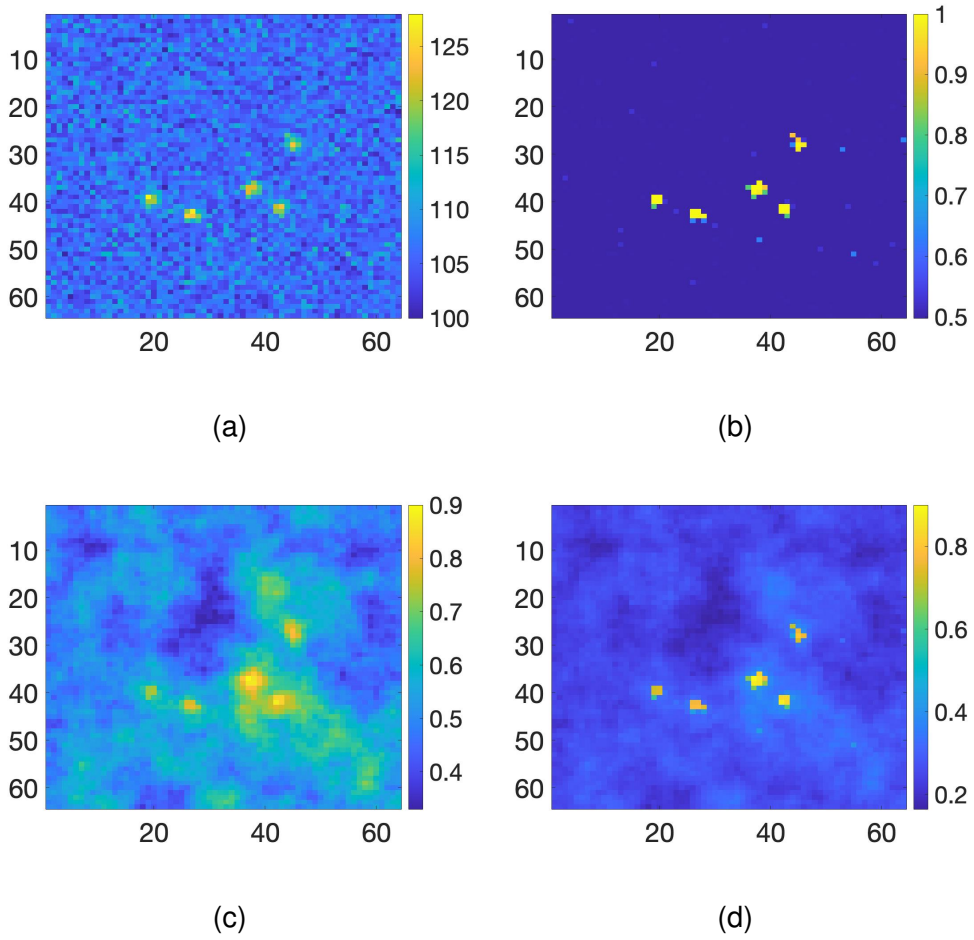


Figure 5.14: Examples of the input data to the NN for an image with five fluorophores emitting. (a) Shows the raw simulated image, (b) after it has been processed by the Mixture Model, (c) after it has been processed with the MIM and (d) is (b) and (c) multiplied together.

On the other hand, if the learning ratio is too large, then the values that minimise the loss function the most may be missed.

The NN computes both the accuracy and the loss of the model after each batch. The accuracy describes how well the NN works at classifying the images to the number of fluorophores that are 'on'.

The loss is calculated using a cross entropy function [141] and is related to the difference between the classification number and the actual label number [142]. This function is commonly used in ML as a loss function.

The cross entropy loss function, L_{CE}, is sometimes known as the logarithmic loss, log loss or logistic loss function. As stated earlier

the NN calculates the probability for each label, and the label with the highest probability is the final output. The loss compares the predicted class probability to the desired output and is calculated using equation (5.9):

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i). \tag{5.9}$$

Where n is the number of classes, t_i is the truth label for the i^{th} class and p_i is the probability calculated by the NN for the i^{th} class. The perfect model would have a cross entropy of zero. Due to the logarithmic scale, there is a larger penalty for differences close to one and a small penalty for differences close to zero. [143]

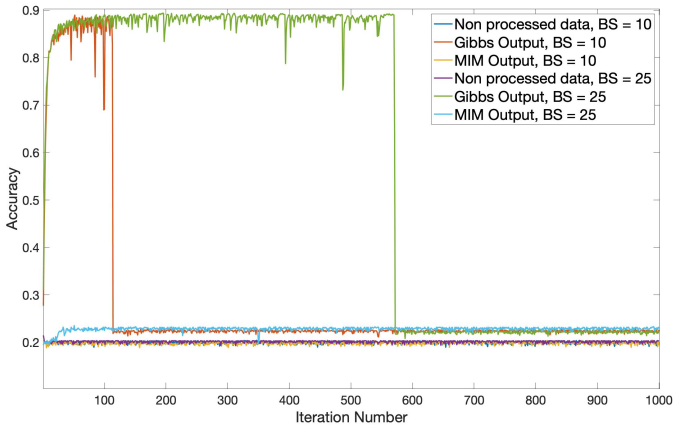
5.5.5 *Results from the neural network (NN)*

Many different values of the learning rate and batch size were used to train the NN and determine the best values to use. Figure 5.15 shows the accuracy and figure 5.16 the loss for the non-processed data, the Mixture Model output and the MIM output with learning rates of $1x10^{-3}$ and $1x10^{-5}$, and batch sizes of both 10 and 25. It can be seen from this data that a learning rate of $1x10^{-3}$ is too small and the NN often fails, being unable to get results. When using a learning rate of $1x10^{-5}$ the highest accuracy reached was 90.8% for both MIM output datasets. The Mixture Model output datasets yielded the next highest accuracies with 89.3% and 89.2% for batch sizes 10 and 25 respectively. Using the non-processed dataset, the accuracy reached was 88.5% and 88.6% for the batch sizes of 10 and 25 respectively.

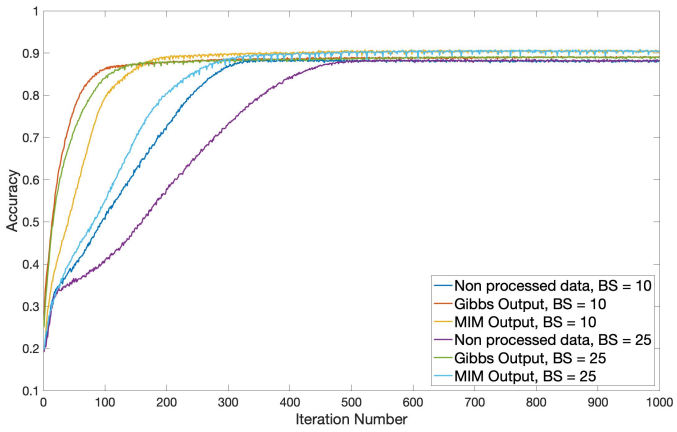
It can be seen from these data that the non processed dataset takes the longest to learn and reaches the lowest accuracy. The MIM datasets learn the second quickest, but to the highest accuracy. The Mixture Model datasets learnt the fastest, however it was not to the highest accuracy. In each case the batch size of 10 learnt the fastest, however it did not always reach the highest accuracy (for each dataset).

It was postulated that removing the images with the label 'zero' would enable the NN to learn to a higher accuracy and lower loss. A learning rate of $1x10^{-5}$ and a batch size of 10 were used. The accuracy results determined were 88.5%, 89.26% and 90.82% for the original data, Mixture Model output and MIM output respectively. The accuracy and loss for this data are shown in figure 5.17.

The Mixture Model and the MIM output were then multiplied together to incorporate the structures from both outputs. The results



(a)

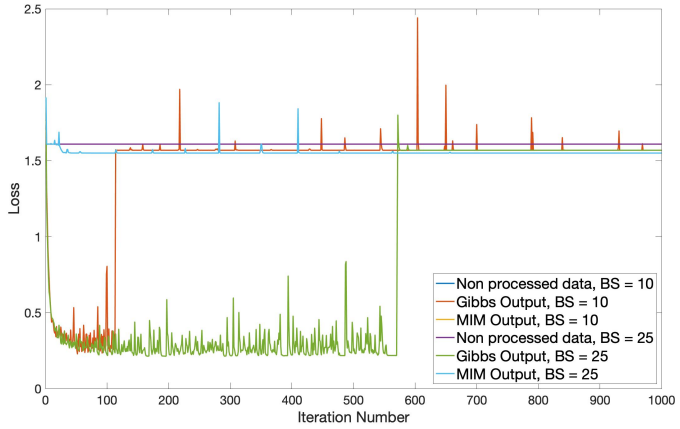


(b)

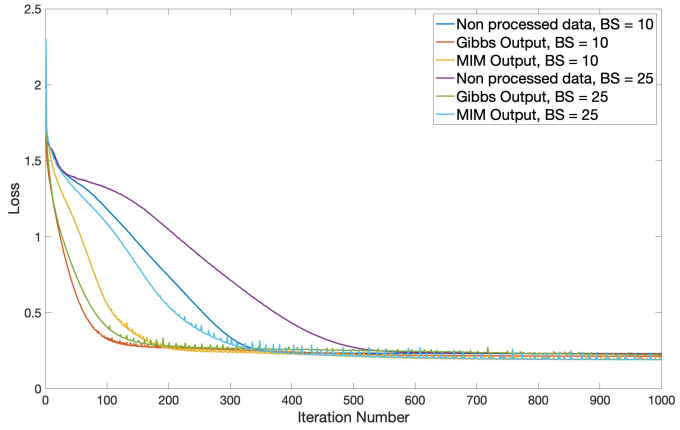
Figure 5.15: Accuracy against iteration number for non processed data, Mixture Model output and MIM output for batch sizes of 10 and 25. a) All datasets use a learning rate of $1x10^{-3}$ and b) uses a learning rate of $1x10^{-5}$.

from this are shown in figure 5.18 with a comparison to the Mixture Model and MIM outputs. These have a batch size of 25 and a learning rate of either $1x10^{-4}$ or $1x10^{-5}$. It can be seen that this data learns faster and to a higher accuracy. The highest accuracy reached by the product of the Mixture Model and MIM outputs are 98.75% and 98.39% for the learning rates $1x10^{-4}$ and $1x10^{-5}$ respectively. The Loss gets to a value of 0.0199 and 0.0270 for the learning rates $1x10^{-4}$ and $1x10^{-5}$ respectively.

Table 5.1 shows a summary of the results from the NN. It shows the highest accuracy reached for each of the different datasets and the



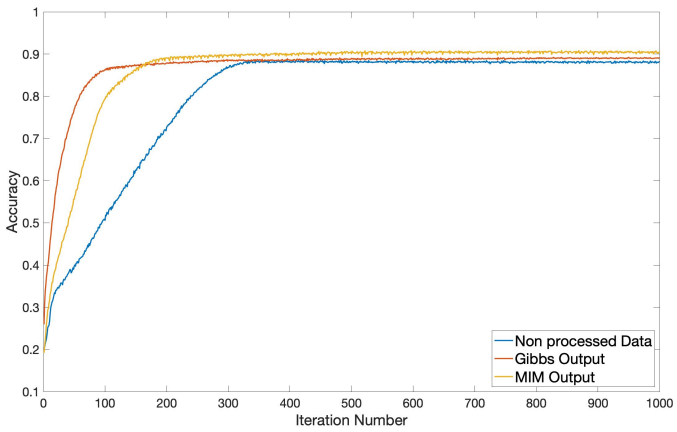
(a)



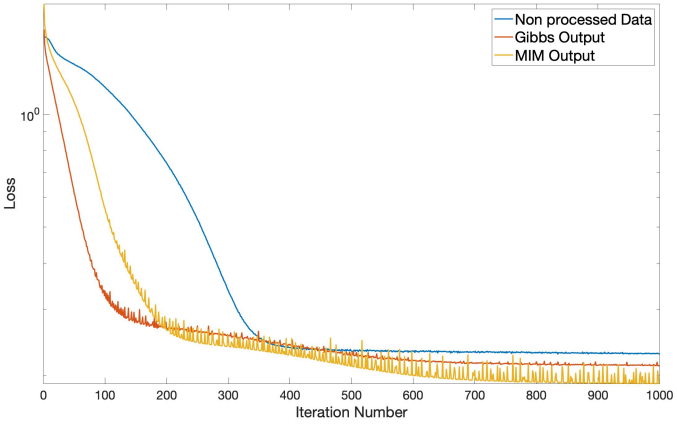
(b)

Figure 5.16: Loss against iteration number for non processed data, Mixture Model output and MIM output for batch sizes of 10 and 25. a) All datasets use a learning rate of $1x10^{-3}$ and b) uses a learning rate of $1x10^{-5}$.

associated loss value. It also shows the number of iterations required for the NN to reach an accuracy of 80%. Different batch sizes and learning rates were used to achieve these results, and it should be noted that, although an accuracy of 80% is reached by this iteration number, due to the nature of how the NN learns, the accuracy may dip below this on subsequent iterations. Different runs of the non-processed, Mixture Model output, and MIM output datasets did learn to 80% in fewer iterations, however they did not reach as high an accuracy, nor were they quicker than the final dataset; the product of the Mixture Model and MIM outputs. From these results it can be seen that the product of



(a)



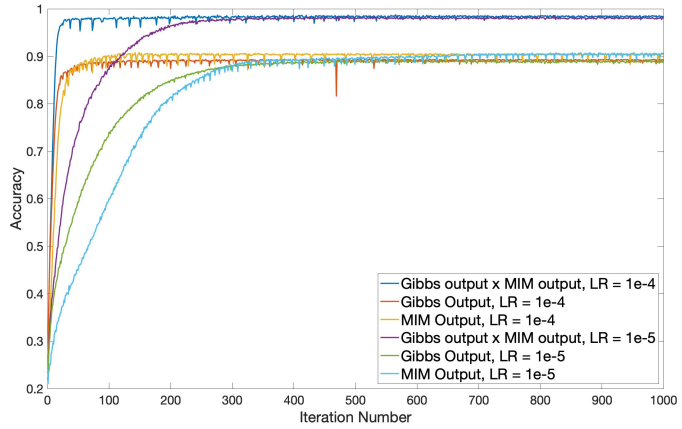
(b)

Figure 5.17: The a) accuracy and b) loss against iteration number for the non processed dataset, the Mixture Model output, and MIM output using labels 1-5. Here a learning rate of $1x10^{-5}$ and a batch size of 10 was used.

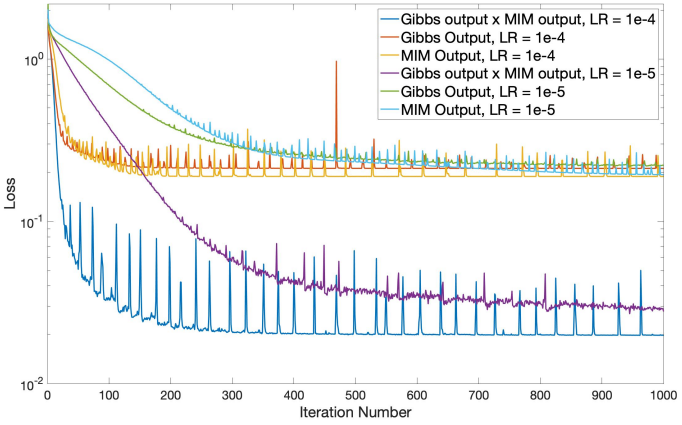
the Mixture Model and MIM output has a significantly higher accuracy, lower loss, and learns the fastest.

The next step would be to benchmark this with real data. This gives rise to many different challenges including a limited ground truth, heterogeneity of real data, experimental variability, and code limitations.

When using real data, establishing a reliable ground truth for benchmarking the algorithm's performance in counting fluorophores is challenging. One way to test it would be to compare its performance to other counting algorithms, for example, ThunderSTORM. Although this



(a)



(b)

Figure 5.18: The a) accuracy and b) loss against iteration number for the product of the Mixture Model and MIM outputs, the Mixture Model output, and MIM. Here a learning rates of both $1x10^{-4}$ and $1x^{-5}$ and a batch size of 25 was used.

serves as a valuable benchmark, it may not fully reveal the algorithms proficiency in counting fluorophores.

Despite the efforts made for the simulated code to create images as realistic as possible, there are certain elements that are absent in the images. For example there is no autofluorescence within the datasets.

The NN has been trained exclusively on data resembling Alexa Fluor (AF) 647. Real experiments introduce numerous variables that may not be fully represented in the training data, although best efforts have been made to make it as real as possible. For example, biological variations, drift, sample preparation techniques, and environmental conditions.

This would potentially limit the NNs ability to generalise effectively to real-world scenarios.

An inherent challenge when analysing real data with the NN lies on its training on sparse images with emitters situated solely at the centre. Finding a stack of data that only has a small amount of fluorophores in, with all of these situated in the centre of the images would be very difficult.

All these factors have prevented the NN from being used on real data in its current form.

Table 5.1: A summary of the results from the NN. This shows the highest accuracy reached, with the associated lowest loss, and the number of iterations to reach an accuracy of 80%. It should be noted that each of these datasets uses different learning rates and batch sizes to reach the highest accuracies. Also, although the accuracy has reached 80% by this number of iterations, due to the nature of the learning process, the accuracy may dip below 80% after this.

Dataset	Number of iterations needed to reach accuracy of 80%	Highest Accuracy reached	Lowest loss Reached
Non-processed	353	89.6%	0.229
Mixture Model Output	59	89.3%	0.213
MIM Output	198	90.8%	0.191
Mixture Model Output x MIM Output	14	98.9%	0.017

Comparing the neural network (NN) with ThunderSTORM

The simulated images created to train the NN were analysed using ThunderSTORM to test how they compared. ThunderSTORM was used to count how many fluorophores were in each image. This was then compared to how many were simulated within the image. If ThunderSTORM counted the same number of fluorophores as was simulated, then this was classed as correctly classified. If any other number of fluorophores were detected then this was classed as an incorrect measurement. In addition to this, it was also looked into how many images ThunderSTORM miscounted by one fluorophore. For example, if the image simulated three fluorophores emitting and ThunderSTORM counted either two or four fluorophores.

This analysis was completed for each stack of images: the raw images, the probability outputs from the Mixture Model, the probability outputs from the MIM, and the latter two multiplied together. The results from this analysis are presented in table 5.2. The table displays the highest accuracy reached using the NN, the percentage of images that were correctly classified by ThunderSTORM, and the percentage of images correctly classified within one fluorophore for ThunderSTORM.

The highest accuracy in table 5.2 refers to the highest possible accuracy from the NN. This is not necessarily the final accuracy. Due to the nature of the learning process within the NN the accuracy may dip below this upon further iterations.

From this it can be seen that NN outperformed ThunderSTORM for all the different types of data input, especially the processed data. It was expected that ThunderSTORM would not perform as well when processed data was inputted due to the nature of the images. The processed images are comprised of probabilities, rather than the larger whole numbers that would be expected from real data. In addition to this, the shape of the fluorophores within these images are going to be different from what ThunderSTORM is programmed to take in.

Looking at the percentage of images correctly classified to within one fluorophore using ThunderSTORM, it can be seen that using the raw simulated images produces an extremely high probability of near 97%. It is possible that some of the fluorophores were too 'dim' for the algorithm to count them as emitters. It is also possible that the algorithm counted 'phantom' fluorophores where fluctuations in the background noise produced a shape similar to an emitter.

Table 5.2: Both the NN and ThunderSTORM were used to classify the same four stacks of simulated images; the raw images, the probabilities outputted from the Mixture Model, the probabilities outputted from the MIM, and the latter two multiplied together. The table shows the highest accuracy achieved when using the NN on the training dataset (the percentage of images in which the number of flourophores that were detected was equal to the number of fluorophores that were simulated in that image of the training data), the percentage of images correctly classified using ThunderSTORM, and the percentage of images correctly classified to within one fluorophore using ThunderSTORM.

Data	Percentage of images correctly classified using the NN with the number of fluorophores simulated using the training data	Percentage of images correctly classified using ThunderSTORM with the number of fluorophores simulated	Percentage of images correctly classified using ThunderSTORM to within one fluorophore of the fluorophores simulated
Raw Simulated Images	89.6%	72.35%	96.69%
Mixture Model output	89.3%	17.20%	32.94%
MIM output	90.8%	9.67%	21.64%
Mixture Model Output x MIM Output	98.9%	24.06%	47.75%

Expanding the neural network (NN) into the time dimension

The majority of the previous codes have analysed the stack of SMLM images individually, initially with the pixels independent, identically distributed (i.i.d.) (the Mixture Model; section 4.3), and then incorporating the neighbouring pixels to include a spatial dependency (using several different methods including the MIM; section 4.4, and the NN; section 5.5.4). Although this approach has yielded good results, due to the blinking nature of fluorophores within SMLM, incorporating a temporal element within the algorithm should theoretically result in a greater accuracy. By tracking fluorophores over time and using their blinking characteristic of turning 'off' and 'on' again, the precision of the localisation could be increased (if the exact emitter can be determined in subsequent images). This temporal integration could help identify more patterns, trends, or even anomalies that might not be evident when taking a frame by frame analysis.

The nature of fluorophores has been discussed in section 3.2. As fluorophores turn 'on' and 'off' stochastically, they are often 'on' for more than one consecutive frame, as well as in multiple frames distributed throughout the stack. The shorter the frame, the more likely that they will be 'on' for multiple consecutive frames, however this would also increase the noise within the image (see section 3.4) as well as reducing the photons released by the emitter for each frame making it more difficult to localise the fluorophores. Different fluorophores have different average 'on' times, and would be more, or less, likely to be 'on' for consecutive frames. This could be utilised by using data in neighbouring frames to localise fluorophores to a higher precision.

To try and exploit this temporal element, the NN was adapted to take in 3-dimensional data. Ideally, the whole stack of images would be analysed as one dataset. Unfortunately, the amount of available computing power is limited, and therefore, at this moment in time, this approach is not possible. Instead, the adapted 3D NN receives three consecutive frames as its input, rather than the individual images independently.

To modify the NN to allow for the input of 3D data, all the convolutions were changed to 3D convolutions, and a new set of training data was created. The code can be found in appendix D.6.3. The training data this time was a stack of images created using the the simulation code described in chapter 3 that changes through time, with the fluorophores being simulated as AF 647.

This stack of simulated images was used to train the 3D NN. The simulation code was developed to simulate 10 fluorophores in a small area (either 'on' or 'off'). The maximum number of fluorophores 'on' in each image is 8 fluorophores, the minimum was 0. These images were also analysed using ThunderSTORM with the results from each method compared. If ThunderSTORM found the correct number of fluorophores then the image was classified as being correctly identified.

When using the 3D NN, the highest accuracy reached when using the raw images was 40.4%, when using the output from the Mixture Model the highest accuracy was 42.4%, for the output of the MIM, the highest accuracy was 40.4%, and for the output of Mixture Model and MIM multiplied together the highest accuracy was 42.4%. The loss for these datasets at this accuracy are 0.481, 19.8, 2.00, and 42.4 respectively. The loss and accuracy for these are shown in figure 5.19.

It can be seen that the 3D NN is struggling to learn past around 40% for all of the different stacks of images. As the output can go up to 8 (or even 10), if the NN was just guessing then there would be an accuracy between 10 and 14%. Further investigation is needed to understand way the 3D NN is struggling to learn. It is possible that is not enough correlation between the neighbouring images, or it could be that there is not enough training data for the NN to work on.

One main concern, that could be the reason for the low accuracy of this NN, is that the dataset is not balanced; there is an unequal number of images with the same number of fluorophores within them. The inherent unpredictability in the generation of simulated images, owing to the stochastic nature of the image generation process, complicates the creation of a time-dependent stack of data that maintains balance across different classes.

Due to limited computing power, only a small batch size could be utilised. Consequently, this could have resulted in a restricted representation of the data during the training process. This could stop the NN from training correctly, and could end up 'memorising' sections rather than learning the overall pattern, compromising the learning process of the NN. If a larger batch size could be used, then this would provide a more comprehensive view of the dataset. In theory, the NN should exhibit a greater accuracy than when using the batch size of 3. Caution should be used when increasing the batch size as using an excessively large number may lead to memory constraints and other consequences.

A much higher learning rate was used for the training data of this 3D NN. A variety of different learning rates were tested, however, if

the a value of less than 0.1 was used the algorithm got 'stuck' and stayed at the same accuracy and loss. It can be seen from the images in figure 5.19 that the results are not as good as the 2-dimensional NN. The NN is struggling to learn, and only accuracies of around 40% are found for each of the different data sets.

Another strategy to enhance the efficiency of this 3D NN would be to increase the volume of training data. Having a limited amount of training data could lead to several detrimental impacts on the performance of the NN. It could lead to overfitting, or bias when applied to new data.

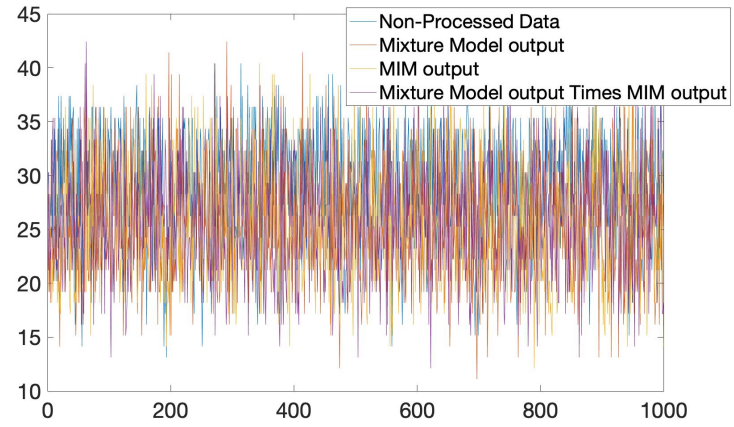
The loss of the 3-dimensional NN is also significantly larger than the loss of the 2-dimensional NN (see figures 5.16 to 5.18). Although this attempt has not yielded results of a high accuracy, further work could be done using a 3-dimensional NN to attempt to improve this.

The results from this 3D NN were then compared to the percentage of images that ThunderSTORM identified correctly. When using the raw images, the percentage of correctly analysed images from ThunderSTORM was 55%, with an extra 33% being only one fluorophore off. Images analysed using the Mixture Model yielded a percentage of 0% (if failed to locate any fluorophores). Using the output from MIM, the percentage of correctly analysed images from ThunderSTORM was 2%. Finally, when using the Mixture Model output multiplied by the MIM, the percentage of correctly identified images from ThunderSTORM was 33%. It was expected that the processed images would have low accuracy when using ThunderSTORM to locate and count the fluorophores as the images are very different to what it is coded for and have values between 0 and 1 as they are probabilities.

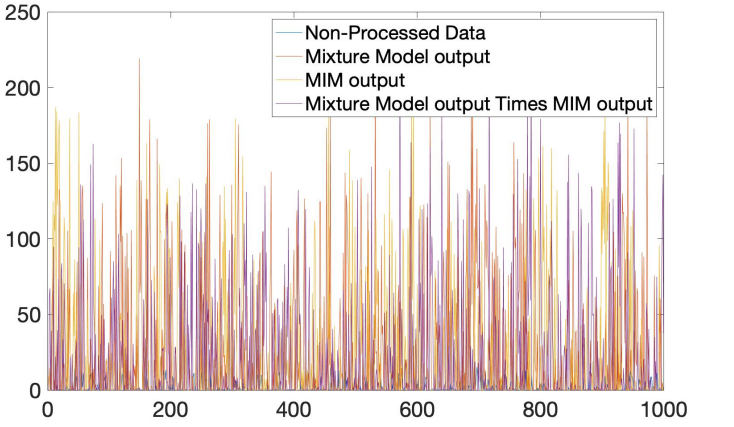
5.6 RESULTS AND DISCUSSION

Several different methods were used to try and determine the number of fluorophores in a sample to varying degrees of success. For all of these methods, simulated data was used.

First, a two staged Mixture Model was used to try and split the signal data into two different populations S_1 and S_2. Unfortunately, due to the lack of signal data in each image this was unsuccessful. The algorithm failed for many images, and was inconsistent across others. It is possible that this could be useful across the entire stack of images, but it is unlikely due to the change in intensity throughout the images.



(a)



(b)

Figure 5.19: The a) accuracy and b) loss against iteration number for the raw images, the Mixture model output, the MIM output, and the product of the Mixture Model and MIM outputs when using the 3D NN. Here a learning rate of 0.2 was used and a batch size of 3 was used.

Next, the Mixture Model statistical output was used to try and determine the change in the number of emitting fluorophores between images using the change in intensity and equation (5.3). The result was inconsistent with the actual difference due to the equation not taking into account the PSF of the fluorophore and working on a pixel wise manner. Incorporating the PSF of the fluorophore, equation (5.3), yielded more promising results with the results and was more consistent with the actual results. A mean absolute difference was calculated to be 9.95.

When the value of μ_s, the mean of the signal population calculated from the Mixture Model, was updated using the process described in section 4.3.6 (using the fixed signal data), the mean absolute difference was calculated to be 4.89. This decreases the error almost 2-fold. There are, on average, 51 molecules 'on' in each frame of this dataset. Therefore the error is around 10%. This result is promising, however, using this to calculate the actual number of fluorophores in each frame would generate too much error. It would be possible that this could be used in conjunction with another method to determine the total number of fluorophores.

MATLAB's classification learner [125] was next used to try and determine how many fluorophores were 'on' in each 'island' of signal. Two different sets of data were used and several different models were trained to try and calculate this. First, the area of the 'islands', the sum and standard deviation of the intensity of the pixels in the 'islands' were used. Next, the area of the 'islands', the sum of the intensity of the pixels in the current image, the previous and the next frame.

Although the second data set produced an accuracy of 87.7% (the same as with the KNN with the first dataset), the modelled train was a decision tree which only took into account the intensity of the current frame and the next frame. This is too coarse a classification and therefore is dismissed as being used as a model for future datasets.

The first data set produced two trained models with accuracies of 87.7% and 87.5%. These were a KNN using ten nearest neighbours and an SVM using a Gaussian kernel respectively. It is thought that the SVM is a better model as it uses a more rigorous and sophisticated approach. Both of these models could be used as viable approaches to count the number of fluorophores in 'islands' of real data. When using held out data, these two models had an accuracy of 87.3% and 85.3% for the KNN and SVM respectively.

A NN was written with the aim to count the number of fluorophores in image sections. Training data was produced that contained between 0 and 4 fluorophores 'on' in each images. These images were processed by the Mixture Model and the MIM to produce two new datasets; the probability outputs. These were then multiplied together to create a fourth and final dataset.

After training the NN on these datasets, the accuracy reached values of 88.6%, 89.3%, 90.8% and 98.9%, and loss values of 0.238, 0.213, 0.190 and 0.017 for the non-processed data, the Mixture Model output, the MIM output and the product of the Mixture Model and MIM outputs respectively. This can be seen in table 5.1. Using processed data also

decreased the number of iterations needed for the NN to learn to a high accuracy. This method potentially provides a way to determine the absolute numbers of fluorophores emitting in a sample, and therefore the number of the structure of interest.

The NN was then adapted to take in 3-dimensional data; the previous, current, and following images. Unfortunately, this NN struggled to learn and the highest accuracy reached was 42.2%. Although this particular NN was unsuccessful, this approach should not be ruled out in the future, and it is thought that, with a lot more work, this could end up having a much higher accuracy than what was found here.

From all these results it can be seen the best method used to count fluorophores is to use NN and the processed data, specifically the final dataset; the product of the Mixture Model and MIM outputs.

5.7 FURTHER WORK

There are many different paths that could be taken using the research that has been completed in this chapter. Larger datasets could be used for the Mixture Model statistics, where more fluorophores are emitting. Equation (5.3) could be used on individual pixels intensity time traces throughout the stack to see if this is a more viable technique than using all the signal pixels in frame.

All of these models could be tested on different simulated data, for example the datasets in [104].

For the NN, the next step would be to use sections of real data to determine if the NN still determines the number to a high accuracy. The 3-dimensional NN needs more work to see if this could get to a higher accuracy. This work could include creating a balanced training data set, using a larger data set, and utilising a supercomputer to enable a larger batch size for training.

The Matlab Classification learner technique could be used in conjunction with the Mixture Model statistical information to calculate the difference in the number of fluorophores in neighbouring frames. Instead of using the intensities in both the neighbouring frames, the estimated change in fluorophore number could be inputted.

5.8 THE ALGORITHM

The code that was written and used in this chapter can be found in appendix D.

The two-staged Mixture Model (section 5.5.1) was written by myself in MATLAB, as was the code that uses the output from these (section 5.5.2). All the ML algorithms in section 5.5.3 were written by myself using the Matlab classification learner. Both the 2D and 3D NNs (section 5.5.4) were written in collaboration with Ashley Cadby, and was adapted from a standard NN that classifies handwritten letters.

REFERENCES

[1] C. SINGER, 'The Dawn of Microscopical Discovery', Journal of the Royal Microscopical Society **35**, 317–340 (1915).

[2] S. Bradbury, *An Introduction to the Optical Microscope* (Oxford University Press, 1989), p. 5.

[3] G. Sines and Y. A. Sakellarakis, 'Lenses in Antiquity', American Journal of Archaeology **91**, 191 (1987).

[4] S. Bradbury, *The Evolution of the Microscope* (Pergamon Press, 1967).

[5] H. Gest, 'The discovery of microorganisms by Robert Hooke and Antoni van Leeuwenhoek, Fellows of The Royal Society', Notes and Records of the Royal Society **58**, 187–201 (2004).

[6] C. S. Ball, 'The early history of the compound microscope', Bios **37**, 51–60 (1966).

[7] D. Bardell, 'The First Record of Microscopic Observations', BioScience **33**, 36–38 (1983).

[8] R. Hooke, *Micrographia* (Royal Society, 1665).

[9] K. B. Wolf and G. Krötzsch, 'Geometry and dynamics in refracting systems', European Journal of Physics **16**, 14–20 (1995).

[10] M. Hausner, *Optics Inspections and Tests: A Guide for Optics Inspectors and Designers* (2017), pp. 7–32.

[11] A. Kwan, J. Dudley and E. Lantz, 'Who really discovered snell's law?', Physics World **15**, 64 (2002).

[12] R. D. Fiete, 'Optics', in *Modeling the imaging chain of digital cameras* (SPIE Press, 2010), pp. 49–72.

[13] L. Davis, C, *Thin Lenses*, http://www.physics.louisville.edu/cldavis/phys299/notes/lo_lense Accessed: 2020-14-04.

[14] S. J. Ling, J. Sanny and B. Moebs, *4.1: single-slit diffraction*, https://phys.libretexts.org/?title=T _Optics_and_Modern_Physics_(OpenStax)/4:_Diffraction/4.1:_Single-Slit_Diffraction%0A, Visited on 2018-05-23.

[15] G. B. Airy, 'On the Diffraction of an Object-glass with Circular Aperture', Transactions of the Cambridge Philosophical Society **5**, 283–291 (1835).

[16] *Greenfluorescentblog*, `https://greenfluorescentblog.wordpress.com/2012/04/01/numerical-aperture-and-resolution/`, Accessed: 2018-01-29.

[17]E. Abbe, 'Beitrage zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung', Arch. Mikroscop Anat **9**, 413–420 (1873).

[18]T. S. Tkaczyk, 'Numerical Aperture', in *Field guide to microscopy* (2010), p. 38.

[19]B. R. Masters, 'Optical Resolution and Resolving Power : What It Is , How to Measure It , and What Limits It', in *Confocal microscopy and multiphoton excitation microscopy: the genesis of live cell imaging* (SPIE Press, 2006), pp. 49–54.

[20]N. Foundation, *The nobel prize in chemistry*, 2014.

[21]S. Balaiya, R. K. Murthy, V. S. Brar and K. V. Chalam, 'Evaluation of ultraviolet light toxicity on cultured retinal pigment epithelial and retinal ganglion cells', Clinical Ophthalmology, 33–39 (2010).

[22]M. Abramowitz and M. W. Davidson, *Anatomy of a Microscope - Numerical Aperture and Resolution*, Visited on 2020-04-16.

[23]N. Foundation, *The nobel prize in physics*, 1929.

[24]W. J. Croft, *Under the microscope [electronic resource] : a brief history of microscopy* (World Scientific Publishing Co. Pte. Ltd., 2006), pp. 57–72.

[25]R. Erni, M. D. Rossell, C. Kisielowski and U. Dahmen, 'Atomic Resolution Imaging with a sub-50 pm Electron Probe', Physical Review Letters **102**, 96–101 (2009).

[26]N. Foundation, *The nobel prize in physics*, 1986.

[27]J. Ayache, L. Beaunier, J. Boumendil, G. Ehret and D. Laub, 'Artifacts in Transmission Electron Microscopy', in *Sample preparation handbook for transmission electron microscopy* (Springer, New York, NY, 2010), pp. 125–170.

[28]J. G. Mcnally, T. Karpova, J. Cooper and J. A. Conchello, 'Three-Dimensional Imaging by Deconvolution Microscopy', Methods **19**, 373–385 (1999).

[29]G. Sparacino, G. De Nicolao, G. Pillonetto and C. Cobelli, 'Deconvolution', in *Modelling methodology for physiology and medicine* (Elsevier, 2014), pp. 45–68.

[30]M. Minsky, 'Memoir on inventing the confocal scanning microscope', Scanning **10**, 128–138 (1988).

[31]M. Minsky, *Microscopy Apparatus*, 1961.

[32]M. Renz, 'Fluorescence microscopy-A historical and technical perspective', Cytometry Part A **83**, 767–779 (2013).

[33]H. Helmholtz, 'Die theoretischen Grenzen für die Leistungsfähigkeit der Mikroskope.', Annalen der Physik, 557–584 (1874).

[34]M. Muyskens and Ed Vitz, 'The Fluorescence of Lignum nephriticum: A Flash Back to the Past and a Simple Demonstration of Natural Substance Fluorescence', Journal of Chemical Education **83**, 765 (2006).

[35]G. G. Stokes, 'On the Change of Refrangibility of Light', Philosophical Transactions of the Royal Society of London **142**, 463–562 (1852).

[36]G. G. Stokes, 'On the Change of Refrangibility of Light. No. II', Philosophical Transactions of the Royal Society of London **143**, 385–396 (1853).

[37]L. J. Kricka and P. Fortina, 'Analytical ancestry: "firsts" in fluorescent labeling of nucleosides, nucleotides, and nucleic acids', Clinical Chemistry **55**, 670–683 (2009).

[38]D. J. S. Birch, Y. Chen and O. J. Rolinski, 'Fluorescence', in *Photonics, volume 4 : biomedical photonics, spectroscopy, and microscopy* (John Wiley & Sons, Incorporated, 2015), pp. 1–58.

[39]G. T. Dempsey, J. C. Vaughan, K. H. Chen, M. Bates and X. Zhuang, 'Evaluation of fluorophores for optimal performance in localization-based super-resolution imaging', Nature Methods **8**, 1027–1036 (2011).

[40]R. Smallman and A. Ngan, 'Characterization and Analysis', in *Modern physical metallurgy* (Elsevier, 2014), pp. 159–250.

[41]H. Siedentopf and R. Zsigmondy, 'Uber Sichtbarmachung und Größenbestimmung ultramikoskopischer Teilchen, mit besonderer Anwendung auf Goldrubingläser', Annalen der Physik **315**, 1–39 (1902).

[42]D. Axelrod, 'Cell-substrate contacts illuminated by total internal reflection fluorescence.', The Journal of Cell Biology **89**, 141–145 (1981).

[43]Y. Markaki and H. Harz, eds., *Light Microscopy*, Vol. 1563, Methods in Molecular Biology (Springer New York, New York, NY, 2017).

[44]B. O. Leung and K. C. Chou, 'Review of super-resolution fluorescence microscopy for biology.', Applied spectroscopy **65**, 967–80 (2011).

[45]S. W. Hell and J. Wichmann, 'Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy', Optics Letters **19**, 780 (1994).

[46]R. C. Dunn, 'Near-field scanning optical microscopy.', Chemical reviews **99**, 2891–928 (1999).

[47]M. G. L. Gustafsson, 'Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution', Proceedings of the National Academy of Sciences **102**, 13081–13086 (2005).

[48]E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz and H. F. Hess, 'Imaging intracellular fluorescent proteins at nanometer resolution.', Science (New York, N.Y.) **313**, 1642–5 (2006).

[49]M. Rust, M. Bates and X. Zhuang, 'Stochastic optical reconstruction microscopy (STORM) provides sub-diffraction-limit image resolution', Nature methods **3**, 793–795 (2006).

[50]S. T. Hess, T. P. K. Girirajan and M. D. Mason, 'Ultra-high resolution imaging by fluorescence photoactivation localization microscopy.', Biophysical journal **91**, 4258–72 (2006).

[51]T. Dertinger, R. Colyer, G. Iyer, S. Weiss and J. Enderlein, 'Fast, background-free, 3D super-resolution optical fluctuation imaging (SOFI).', Proceedings of the National Academy of Sciences of the United States of America **106**, 22287–22292 (2009).

[52]B. Huang, M. Bates and X. Zhuang, 'Super-Resolution Fluorescence Microscopy', Annual Review of Biochemistry **78**, 993–1016 (2009).

[53]D. Wildanger et al., 'Solid immersion facilitates fluorescence microscopy with nanometer resolution and sub-Ångström emitter localization', Advanced Materials **24**, 309–313 (2012).

[54]R. Heintzmann, T. M. Jovin and C. Cremer, 'Saturated patterned excitation microscopy—a concept for optical resolution improvement', Optical Society of America **19**, 1599–1609 (2002).

[55]E. H. Rego, L. Shao, J. J. Macklin, L. Winoto, G. A. Johansson, N. Kamps-Hughes, M. W. Davidson and M. G. Gustafsson, 'Nonlinear structured-illumination microscopy with a photoswitchable protein reveals cellular structures at 50-nm resolution', Proceedings of the National Academy of Sciences of the United States of America **109**, 13–15 (2012).

[56]E. Betzig, 'Proposed method for molecular optical imaging', Optics Letters **20**, 237 (1995).

[57]W. Moerner and L. Kador, 'Optical detection and spectroscopy of single molecules in a solid', Physical Review Letters **62**, 2535–2538 (1989).

[58]M. Orrit and J. Bernard, 'Single pentacene molecules detected by fluorescence excitation in a p-terphenyl crystal', Physical Review Letters **65**, 2716–2719 (1990).

[59]W. P. Ambrose and W. E. Moerner, 'Fluorescence spectroscopy and spectral diffusion of single impurity molecules in a crystal', Nature **349**, 225–227 (1991).

[60]R. M. Dickson, A. B. Cubitt, R. Y. Tsien and W. E. Moerner, 'On / off blinking and switching behaviour of single molecules of green fluorescent protein', **388**, 355–358 (1997).

[61]M. Heilemann, S. van de Linde, A. Mukherjee and M. Sauer, 'Super-resolution imaging with small organic fluorophores.', Angewandte Chemie (International ed. in English) **48**, 6903–8 (2009).

[62]M. Heilemann, S. van de Linde, M. Schüttpelz, R. Kasper, B. Seefeldt, A. Mukherjee, P. Tinnefeld and M. Sauer, 'Subdiffraction-resolution fluorescence imaging with conventional fluorescent probes.', Angewandte Chemie (International ed. in English) **47**, 6172–6 (2008).

[63]L. Stryer and R. P. Haugland, 'Energy transfer: a spectroscopic ruler.', Proceedings of the National Academy of Sciences **58**, 719–726 (1967).

[64]*Single-molecule localization microscopy: directory of smlm software*, http://bigwww.epfl.ch/smlm, Visited on 2020-06-19, 2017.

[65]P. D. Simonson, E. Rothenberg and P. R. Selvin, 'Single-molecule-based super-resolution images in the presence of multiple fluorophores.', Nano letters **11**, 5090–6 (2011).

[66]S. A. Jones, S.-H. Shim, J. He and X. Zhuang, 'Fast, three-dimensional super-resolution imaging of live cells', Nature Methods **8**, 499–505 (2011).

[67]A. Small and S. Stahlheber, 'Fluorophore localization algorithms for super-resolution microscopy.', Nature methods **11**, 267–79 (2014).

[68]A. V. Abraham, S. Ram, J. Chao, E. S. Ward and R. J. Ober, 'Quantitative study of single molecule location estimation techniques', Optics Express **17**, 23352 (2009).

[69]R. E. Thompson, D. R. Larson and W. W. Webb, 'Precise nanometer localization analysis for individual fluorescent probes.', Biophysical journal **82**, 2775–2783 (2002).

[70]S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice Hall, 1993).

[71]R. Henriques, M. Lelek, E. F. Fornasiero, F. Valtorta, C. Zimmer and M. M. Mhlanga, 'QuickPALM: 3D real-time photoactivation nanoscopy image processing in ImageJ.', Nature methods **7**, 339–40 (2010).

[72]R. Parthasarathy, 'Rapid, accurate particle tracking by calculation of radial symmetry centers', Nature Methods **9**, 724–726 (2012).

[73]A. J. Berglund, M. D. McMahon, J. J. McClelland and J. A. Liddle, 'Fast, bias-free algorithm for tracking single particles with variable size and shape', Optics Express **16**, 14064 (2008).

[74]H. Ma, F. Long, S. Zeng and Z.-L. Huang, 'Fast and precise algorithm based on maximum radial symmetry for single molecule localization', Optics Letters **37**, 2481 (2012).

[75]T. Quan, H. Zhu, X. Liu, Y. Liu, J. Ding, S. Zeng and Z.-L. Huang, 'High-density localization of active molecules using Structured Sparse Model and Bayesian Information Criterion', Optics Express **19**, 16963 (2011).

[76]Y. Wang, T. Quan, S. Zeng and Z.-L. Huang, 'PALMER: a method capable of parallel localization of multiple emitters for high-density localization microscopy', Optics Express **20**, 16039 (2012).

[77]F. Huang, S. L. Schwartz, J. M. Byars and K. a. Lidke, 'Simultaneous multiple-emitter fitting for single molecule super-resolution imaging.', Biomedical optics express **2**, 1377–1393 (2011).

[78]F. Huang et al., 'Video-rate nanoscopy using sCMOS camera-specific single-molecule localization algorithms.', Nature methods **10**, 653–8 (2013).

[79]S. J. Holden, S. Uphoff and A. N. Kapanidis, 'DAOSTORM: an algorithm for high-density super-resolution microscopy', Nature methods **8**, 279–280 (2011).

[80]D. R. Tobergte and S. Curtis, 'Daophot: a Computer Program for Crowded-Field Stellar Photometry (Psf Photometry)', Journal of Chemical Information and Modeling **53**, 1689–1699 (2013).

[81]L. B. Lucy, 'An iterative technique for the rectification of observed distributions', Astronomical Journal **79**, 745–754 (1974).

[82]W. Richardson, 'Bayesian-based iterative method of image restoration', Optical Society of America **62**, 55–59 (1972).

[83]L. Zhu, W. Zhang, D. Elnatan and B. Huang, 'Faster STORM using compressed sensing', Nature Methods **9**, 721–723 (2012).

[84]S. Cox, E. Rosten, J. Monypenny, T. Jovanovic-Talisman, D. T. Burnette, J. Lippincott-Schwartz, G. E. Jones and R. Heintzmann, 'Bayesian localization microscopy reveals nanoscale podosome dynamics.', Nature methods **9**, 195–200 (2012).

[85]S. Wolter, U. Endesfelder, S. van de Linde, M. Heilemann and M. Sauer, 'Measuring localization performance of super-resolution algorithms on very active samples.', Optics express **19**, 7020–33 (2011).

[86]S. Wang, J. R. Moffitt, G. T. Dempsey, X. S. Xie and X. Zhuang, 'Characterization and development of photoactivatable fluorescent proteins for single-molecule-based superresolution imaging', Proceedings of the National Academy of Sciences of the United States of America **111**, 8452–8457 (2014).

[87]N. Durisic, L. Laparra-Cuervo, Á. Sandoval-Álvarez, J. S. Borbely and M. Lakadamyali, 'Single-molecule evaluation of fluorescent protein photoactivation efficiency using an in vivo nanotemplate', Nature Methods **11**, 156–162 (2014).

[88]P. J. Green, 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', Biometrika **82**, 711 (1995).

[89]H. Jeffreys, *Theory of probability*, 3rd (Press, Oxford: University, 1961).

[90]L. Savage, *The foundations of Statistics* (New York: Wiley, 1972).

[91]B. De Finetti, *Theory of probability* (London: Wiley, 1974).

[92]R. Christensen, T. E. Hanson, W. Johnson, Branscum and Adam, *Bayesian Ideas and Data Analysis : An Introduction for Scientists and Statisticians*, 1st ed. (Taylor & Francis Group, 2010).

[93]S. Chib and E. Greenberg, 'Understanding the Metropolis-Hastings Algorithm', The American Statistician **49**, 327–335 (1995).

[94]M. Jesper, ed., *Spactial Statistics and Computational Methods* (Springer, 2003), pp. 1–13.

[95]K. I. Mortensen, L. S. Churchman, J. A. Spudich and H. Flyvbjerg, 'Optimized localization analysis for single-molecule tracking and super-resolution microscopy', Nature Methods **7**, 377–381 (2010).

[96]J. R. Janesick, *Photon Transfer* (SPIE.Digital Library, 2007).

[97]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, `http://www.deeplearningbook.org` (MIT Press, 2016).

[98]S. Mor-Yosef, A. Samueloff, B. Modan, D. Navot and J. G. Schenker, 'Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study.', Obstetrics and gynecology **75**, 944–7 (1990).

[99] V. Metsis, I. Androutsopoulos and G. Paliouras, 'Spam Filtering with Naive Bayes – Which Naive Bayes?', In CEAS (2006).

[100] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, http://www.deeplearningbook.org (MIT Press, 2016).

[101] *Neural Network Programming - Deep Learning with PyTorch*, https://deeplizard.com/learn/ Accessed: 2020-07-08.

[102] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', Nature **323**, 533–536 (1986).

[103] M. Torrisi, G. Pollastri and Q. Le, 'Deep learning methods in protein structure prediction', Computational and Structural Biotechnology Journal **18**, 1301–1310 (2020).

[104] *Single-molecule localization microscopy: collection of reference datasets*, http://bigwww.epfl.ch/smlm/datasets/index.html, Visited on 2020-06-19, 2017.

[105] H. Li and J. C. Vaughan, 'Switchable Fluorophores for Single-Molecule Localization Microscopy', Chemical Reviews **118**, 9412–9454 (2018).

[106] B. A. Griffin, S. R. Adams, R. Y. Tsien, B. A. Griffin, S. R. Adams and R. Y. Tsient, 'Specific Covalent Labeling of Recombinant Protein Molecules Inside Live Cells Published by : American Association for the Advancement of Science Stable URL : http://www.jstor.org/stable/2896025 REFERENCES Linked references are available on JSTOR for this', Science **281**, 269–272 (1998).

[107] L. W. Miller, Y. Cai, M. P. Sheetz and V. W. Cornish, 'In vivo protein labeling with trimethoprim conjugates: A flexible chemical tag', Nature Methods **2**, 255–257 (2005).

[108] A. Keppler, S. Gendreizig, T. Gronemeyer, H. Pick, H. Vogel and K. Johnsson, 'A general method for the covalent labeling of fusion proteins with small molecules in vivo', Nature Biotechnology **21**, 86–89 (2003).

[109] A. Gautier, A. Juillerat, C. Heinis, I. R. Corrêa, M. Kindermann, F. Beaufils and K. Johnsson, 'An Engineered Protein Tag for Multiprotein Labeling in Living Cells', Chemistry and Biology **15**, 128–136 (2008).

[110] G. V. Los et al., 'HaloTag: A novel protein labeling technology for cell imaging and protein analysis', ACS Chemical Biology **3**, 373–382 (2008).

[111] I. Y. Iourov, *Fluorescence In Situ Hybridization (FISH)*, edited by T. Liehr, Springer Protocols Handbooks (Springer Berlin Heidelberg, Berlin, Heidelberg, 2017), pp. 17–25.

[112]M. E. Brezinski, *Noise and system performance with td-oct and sd-oct* (Academic Press, Amsterdam ; 2006), pp. 175–195.

[113]T. Bushnell, *What is autofluorescence*, https://expert.cheekyscientist.com/what-is-autofluorescence/, Visited on 2020-06-09.

[114]B. Huang, W. Wang, M. Bates and X. Zhuang, 'Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy.', Science (New York, N.Y.) **319**, 810–3 (2008).

[115]M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych and G. M. Hagen, 'ThunderSTORM: A comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging', Bioinformatics **30**, 2389–2390 (2014).

[116]B. A. Cipra, 'An Introduction to the Ising Model', The American Mathematical Monthly **94**, 937–959 (1987).

[117]I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, 'Chapter 9 - Probabilistic methods', in *Data mining practical machine learning tools and techniques*, edited by I. H. Witten, E. Frank, M. A. Hall and C. J. B. T. D. M. (E. Pal (Morgan Kaufmann, 2017), pp. 335–416.

[118]B. A. Landman, I. Lyu, Y. Huo and A. J. Asman, 'Chapter 6 - Multiatlas segmentation', in *Handbook of medical image computing and computer assisted intervention*, edited by S. K. Zhou, D. Rueckert, G. B. T. H. o. M. I. C. Fichtinger and C. A. Intervention (Academic Press, 2020), pp. 137–164.

[119]Orchard, Peter, *Markov random field optimisation*, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0809/ORCHARD/, Accessed 2014-08-28.

[120]Q. Jackson and D. Landgrebe, 'Adaptive Bayesian contextual classification based on Markov random fields', Geoscience and Remote Sensing, ... **40**, 2454–2463 (2002).

[121]A. Sonnleitner, G. Schütz and T. Schmidt, 'Free Brownian Motion of Individual Lipid Molecules in Biomembranes', Biophysical Journal **77**, 2638–2642 (1999).

[122]M. K. Cheezum, W. F. Walker and W. H. Guilford, 'Quantitative comparison of algorithms for tracking single fluorescent particles', Biophysical Journal **81**, 2378–2388 (2001).

[123]U. Kubitscheck, O. Kückmann, T. Kues and R. Peters, 'Imaging and Tracking of Single GFP Molecules in Solution', Biophyiscal Journal **78**, 2170–2179 (2000).

[124]*Markov random fields*, https://www2.isye.gatech.edu/isyebayes/bank/handout16.pdf, Accessed August 11, 2022.

[125] *Classification Learner*, https://uk.mathworks.com/help/stats/classificationlearner-app.html, Accessed: 2020-08/-8.

[126] M. J. Rust, M. Bates and X. Zhuang, 'Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM).', Nature methods **3**, 793–5 (2006).

[127] R. P. Nieuwenhuizen, M. Bates, A. Szymborska, K. A. Lidke, B. Rieger and S. Stallinga, 'Quantitative localization microscopy: Effects of photophysics and labeling stoichiometry', PLoS ONE **10**, 1–18 (2015).

[128] D. Gross and W. Webb, 'Molecular counting of low-density lipoprotein particles as individuals and small clusters on cell surfaces', Biophysical Journal **49**, 901–911 (1986).

[129] B. M. Burton, K. A. Marquis, N. L. Sullivan, T. A. Rapoport and D. Z. Rudner, 'The ATPase SpoIIIE Transports DNA across Fused Septal Membranes during Sporulation in Bacillus subtilis', Cell **131**, 1301–1312 (2007).

[130] M. C. Leake, J. H. Chandler, G. H. Wadhams, F. Bai, R. M. Berry and J. P. Armitage, 'Stoichiometry and turnover in single, functioning membrane protein complexes', Nature **443**, 355–358 (2006).

[131] S. K. Das, M. Darshi, S. Cheley, M. I. Wallace and H. Bayley, 'Membrane protein stoichiometry determined from the step-wise photobleaching of dye-labelled subunits', ChemBioChem **8**, 994–999 (2007).

[132] K. Tsekouras, T. C. Custer, H. Jashnsaz, N. G. Walter and S. Pressé, 'A novel method to accurately locate and count large numbers of steps by photobleaching', Molecular Biology of the Cell **27**, edited by D. Lidke, 3601–3615 (2016).

[133] A. Lee, K. Tsekouras, C. Calderon, C. Bustamante and S. Pressé, 'Unraveling the Thousand Word Picture: An Introduction to Super-Resolution Data Analysis', Chemical Reviews **117**, 7276–7330 (2017).

[134] E. Alpaydin, 'Introduction', in *Introduction to machine learning* (2010), pp. 1–19.

[135] C. Campbell and Y. Ying, 'Learning with Support Vector Machines', Synthesis Lectures on Artificial Intelligence and Machine Learning **5**, 1–95 (2011).

[136] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (Cambridge University Press, Mar. 2000).

[137]V. Hodge and J. Austin, 'A Survey of Outlier Detection Methodologies', Artificial Intelligence Review **22**, 85–126 (2004).

[138]M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks', Neural Networks **106**, 249–259 (2018).

[139]L. Deng, 'The mnist database of handwritten digit images for machine learning research [best of the web]', IEEE Signal Processing Magazine **29**, 141–142 (2012).

[140]S. L. Smith, P.-j. Kindermans, C. Ying, Q. V. Le and G. Brain, 'D on ' T D Ecay the L Earning R Ate , I Ncrease the B Atch S Ize', 1–11 (2018).

[141]*Cross entropy loss*, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, Accessed April 02, 2021.

[142]Z. Zhang and M. R. Sabuncu, 'Generalized cross entropy loss for training deep neural networks with noisy labels', Advances in Neural Information Processing Systems **2018-December**, 8778–8788 (2018).

[143]*Cross-entropy loss function*, https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e, Accessed April 02, 2021.

6

DISCUSSION

Optical microscopy is a non-invasive technique to allow a user to look at small structures. The resolution for optical microscopes has increased and traditional light microscopes can resolve distances down to 200nm. This is limited by the diffraction limit of light. Single Molecule Localisation Microscopy (SMLM) techniques, including stochastic optical reconstruction microscopy (STORM) [49], photo-activated localisation microscopy (PALM) [48] and fluorescence photo-activated localisation microscopy (FPALM) [50], circumvent this diffraction limit by exploiting the photoblinking ability of some fluorophores allowing nanoscale resolution.

This thesis has developed an adaptable method to create realistic simulations of SMLM images (chapter 3), a two-stage algorithm to denoise SMLM images (chapter 4), and several different methods in order to count the number of fluorophores in a sample (chapter 5).

6.1 SIMULATIONS

A method that successfully simulates images from SMLM was developed. Although the method is tailored to create images using the photophysics of Alexa Fluor (AF) 647, this adaptable algorithm could be altered to include any fluorophore with well documented emission statistics such as any in [39]. Due to the simulation process, the initial frames should be discarded as all the fluorophores start as 'on'.

These images were tested using ThunderSTORM [115] showing very promising results. Although not every fluorophore was found, this was expected. The dim fluorophores, and those localised by the edges were missed. Some fluorophores with overlapping point spread functions (PSFs) were localised as a single emitter.

These simulated images were comparable to real data both spatially and temporally, showing that the model is a success.

6.2 MIXTURE MODEL, MODIFIED ISING MODEL (MIM)

A two-stage algorithm has been developed that successfully denoises images taken using SMLM with no user input. This algorithm also has several other useful applications that can be exploited.

The first stage of the algorithm, the Mixture Model, uses Bayesian Inference to split the data into two distinct populations on a pixel-wise manner. It is known that there is a spatial dependence in these images due to the PSF of the fluorophores, and therefore, this method is known to have several limitations. This can be used as a standalone algorithm or the output can be fed into the second stage; the MIM.

The Mixture Model outputs the probability that a pixel is signal, and several different statistics about the populations. The two populations are modelled as Gaussian distributions and the means and precisions of these are determined using the data and the priori. Priori is information that is previously known about the data before the data is looked at.

As the probability that a pixel is signal is calculated, a 'soft classification' of signal and noise can be determined using a 'soft threshold'. This method is preferred due to the variation in intensity values across all images. The same soft threshold can be used for all SMLM, however a 'hard threshold' on the intensity values of the raw data would have to be tailored for every image.

The main limitation of the Mixture Model is that enough pixels need to be classified in each population otherwise the algorithm fails. The images require upwards of 1.38% to 4% of pixels to be classified as signal, depending on the signal-to-noise ratio (SNR) of the images.

A rigorous sensitivity test was completed on the Mixture Model. The majority of the unknowns were very robust with regards to changing, however the means of the two populations need to be chosen with more care. Fortunately, more prior information is known about these values.

SMLM images consist of mainly noise pixels, with 'islands' of signal at the locations of the emitting fluorophores. For a low density images, these 'islands' should look like the PSF of fluorophores and be circular and any fluorophores with overlapping PSFs should be formed from a summation of circles. As the Mixture Model takes every pixel to be independent, these 'islands' can have irregular shapes. This limitation is mitigated by further processing the probability output using a MIM.

The MIM processes the probabilities further by using the nearest neighbours to incorporate a spatial dependence. Neighbours of the

same classification are preferred, whereas neighbours of the different classification are penalised.

The MIM is less robust than the Mixture Model and the prior values have to be chosen with a lot of thought. The values chosen work for a variety of different types of images, and were determined from the results of the sensitivity test.

There are several different applications from this two-staged algorithm:

- denoiseing the images,
- calculating the SNR,
- reducing the amount of disk space needed to save the useful data, and
- as preprocessing to aid in the determination of the number of fluorophores in an image, or image section, as detailed in chapter 5.

This method has very useful applications and can be used to help further research using SMLM This method could also be adapted to other data that consists of two distinct populations, for example Light Sheet Microscopy (LSM).

6.3 COUNTING

Counting the number of fluorophores in a sample is very challenging. Difficulties include; the variation in the number of photons emitted, and the noise in the images.

Several different methods were explored and developed in the quest to count the number of fluorophores in a sample, or sample section. For all these different methods, simulated data was used.

Different methods explored were:

- a two staged Mixture Model,
- using the Statistical data from the Mixture Model to try and count the difference in emitting fluorophores in neighbouring images,
- using two different consolidated results from the Mixture Model and using to train models in MATLAB's Classification learner, and
- training a neural network (NN) to determine the number of fluorescent molecules in SMLM image sections. A training dataset was designed specifically for this technique.

- adapting the NN from before to take in 3-dimensional data; the previous, current, and following image, to determine the number of fluorescent molecules in SMLM image sections. Again, a training dataset was designed specifically for this

Using a two-staged Mixture Model was dismissed as a viable method due to the limitations of the Mixture Model and the inconsistency of the results. The method was to process the images using the Mixture Model outlined in chapter 4 and then take the signal pixels only, and reclassify these into two new populations. Not enough data was in the signal population to consistently produce results that could be used to aid in the calculation of the number of fluorophores.

Using the statistical information about the signal population calculated using the Mixture Model, the change in the number of emitting fluorophores between images was calculated. This result was inconsistent with the actual difference due to this system not incorporating the PSF of the fluorophores. This method was was then updated to include the PSF and the mean absolute difference was calculated to be 9.95. This value was improved to 4.89 when updating the statistical output from the Mixture Model using fixed data populations. This shows a very promising result and could be developed further to aid in the total number calculation. It could also be used in conjunction with other algorithms.

A consolidation of results from the Mixture Model and MIM were used to train various different models in MATLAB's classification learner [125]. Two different dataset were used:

1. the area of 'islands' classified as signal, the sum of the intensities in these pixels and the standard deviation of these intensities, and
2. the area of 'islands' classified as signal, and the sum of the intensities in these pixels in the current image, the previous image and the next image.

Although the second dataset produced a model with a higher accuracy, 87.3% up from 87.2%, a decision tree was used for this classification which only used two pieces of information. This classification techniques was dismissed due to the variation of values across different images and this classification method being too coarse.

Two different models were trained to the same accuracy for the first dataset: a K-nearest neighbours (KNN) [137] using 10 nearest neighbours, and a support vector machines (SVM) [135, 136] using a Gaussian kernel. Due SVM using a more rigours approach it is thought

that this model would be more viable in the future to count the number of fluorophores in 'islands' of real data.

The most promising result came from the results of a NN. Training data was produced that contained between zero and four fluorophores in the images. This data was processed using the Mixture Model and the MIM. Four different training datasets were used to train the NN;

1. the raw images,
2. the probabilities output from the Mixture Model,
3. the probabilities output from the MIM, and
4. the product of the probabilities output from both the Mixture Model and the MIM.

The results produced from the NN were 93% 96% 65% and 95% accuracy rates for the for the raw image data, Mixture Model probabilities, MIM probabilities and Mixture Model multiplied by the MIM respectively. The Mixture Model probabilities both trained faster and had lower losses, potentially providing a way to determine the absolute numbers of fluorophores emitting in a sample, and therefore the number of the structure of interest.

When the NN was adapted to take in 3-dimensional data; the previous, current, and following images, it struggled to learn and the highest accuracy reached was 42.4%. This accuracy was found using the outputs from the Mixture Model, and when the output of Mixture Model and MIM were multiplied together. When using either the raw images, or the output from the MIM, the highest accuracy reached was 40.4%.

6.4 FURTHER WORK

The simulation method could be extended to produce 3-dimensional data [114], or by allowing individual, or groups of, fluorophores to have different distributions of photon emission. By allowing different fluorophores, or groups of fluorophores to have different emission statistics would allow for different characteristics in the final images. For example, photons emitted from fluorophores that are deeper in the sample will undergo more scatter and absorption before entering the microscope.

Although the Mixture Model produces very effective results, both the signal and the noise populations are modelled to be Gaussian distributions. This is unphysical and these could be updated to different populations to determine if this improved upon these results.

There are many different paths that could be taken using the research completed in chapter 5. Larger datasets could be used for the Mixture Model statistics, where more fluorophores are emitting. Or equation (5.3) could be used on individual pixels intensity time traces throughout the stack to see if this is a more viable technique than using all the signal pixels in frame.

Using the statistical output from the Mixture Model to count the difference between neighbouring images could be used in conjunction with the Machine Learning (ML). Instead of using the intensities in both the neighbouring frames, the estimated change in fluorophore number could be inputted.

For the NN, the next step would be to use real data rather than simulated data or to use different simulated data for example the datasets in [104].

Also, all of these models could be tested on different simulated data, for example the datasets in [104].

ACRONYMS

ADC analog to digital converter
AF Alexa Fluor
AI artificial intelligence
sCMOS scientific Complementary Metal-Oxide Semiconductor
CNN convolutional neural network
CTS charge transfer state
DL deep learning
e-h electron-hole
EM electron microscopy
EMCCD Electronic Multiplying Charge-Coupled Devices
FISH fluorescence in situ hybridisation
FNN feedforward neural network
FP fixed pattern
FPALM fluorescence photo-activated localisation microscopy
FRET forster resonance energy transfer
FT fourier transform
GFP green fluorescent protein
GPU graphics processing unit
IFT inverse fourier transform
i.i.d. independent, identically distributed
KNN K-nearest neighbours
LM localisation microscopy
LS least-squares
LSFM Light Sheet Fluorescence Microscopy
LSM Light Sheet Microscopy
MCMC Markov Chain Monte Carlo
MH Metropolis-Hastings
MIM Modified Ising Model
ML Machine Learning
MLE maximum likelihood estimate
MNIST modified national institute of standards and technology
MRF markov random field

NA numerical aperture
NN neural network
NSOM near-field scanning optical microscopy
PALM photo-activated localisation microscopy
PDF Probability Density Function
PET photo induced electron transfer
PSF point spread function
QE quantum efficiency
QW quantum well
QY quantum yield
RL Richardson-Lucy
RMS root mean squared
RNN recurrent neural network
ROI regions of interest
SIM structured illumination microscopy
SMLM Single Molecule Localisation Microscopy
SNOM scanning near-field optical microscopy
SNR signal-to-noise ratio
SOFI Super Resolution Optical Fluctuation Microscopy
SSIM saturated structured illumination microscopy
STED stimulated emission depletion
STORM stochastic optical reconstruction microscopy
SVM support vector machines
TIRF total internal reflection fluorescence
VWCM virtual window center of mass

BIBLIOGRAPHY

[1]C. SINGER, 'The Dawn of Microscopical Discovery', Journal of the Royal Microscopical Society **35**, 317–340 (1915).

[2]S. Bradbury, *An Introduction to the Optical Microscope* (Oxford University Press, 1989), p. 5.

[3]G. Sines and Y. A. Sakellarakis, 'Lenses in Antiquity', American Journal of Archaeology **91**, 191 (1987).

[4]S. Bradbury, *The Evolution of the Microscope* (Pergamon Press, 1967).

[5]H. Gest, 'The discovery of microorganisms by Robert Hooke and Antoni van Leeuwenhoek, Fellows of The Royal Society', Notes and Records of the Royal Society **58**, 187–201 (2004).

[6]C. S. Ball, 'The early history of the compound microscope', Bios **37**, 51–60 (1966).

[7]D. Bardell, 'The First Record of Microscopic Observations', BioScience **33**, 36–38 (1983).

[8]R. Hooke, *Micrographia* (Royal Society, 1665).

[9]K. B. Wolf and G. Krötzsch, 'Geometry and dynamics in refracting systems', European Journal of Physics **16**, 14–20 (1995).

[10]M. Hausner, *Optics Inspections and Tests: A Guide for Optics Inspectors and Designers* (2017), pp. 7–32.

[11]A. Kwan, J. Dudley and E. Lantz, 'Who really discovered snell's law?', Physics World **15**, 64 (2002).

[12]R. D. Fiete, 'Optics', in *Modeling the imaging chain of digital cameras* (SPIE Press, 2010), pp. 49–72.

[13]L. Davis, C, *Thin Lenses*, http://www.physics.louisville.edu/cldavis/phys299/notes/lo_l Accessed: 2020-14-04.

[14]S. J. Ling, J. Sanny and B. Moebs, *4.1: single-slit diffraction*, https://phys.libretexts.org/?titl _Optics_and_Modern_Physics_(OpenStax)/4:_Diffraction/4.1:_Single-Slit_Diffraction%0A, Visited on 2018-05-23.

[15]G. B. Airy, 'On the Diffraction of an Object-glass with Circular Aperture', Transactions of the Cambridge Philosophical Society **5**, 283–291 (1835).

[16] *Greenfluorescentblog*, https://greenfluorescentblog.wordpress.com/2012/04/01/numerical-aperture-and-resolution/, Accessed: 2018-01-29.

[17] E. Abbe, 'Beitrage zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung', Arch. Mikroscop Anat **9**, 413–420 (1873).

[18] T. S. Tkaczyk, 'Numerical Aperture', in *Field guide to microscopy* (2010), p. 38.

[19] B. R. Masters, 'Optical Resolution and Resolving Power : What It Is , How to Measure It , and What Limits It', in *Confocal microscopy and multiphoton excitation microscopy: the genesis of live cell imaging* (SPIE Press, 2006), pp. 49–54.

[20] N. Foundation, *The nobel prize in chemistry*, 2014.

[21] S. Balaiya, R. K. Murthy, V. S. Brar and K. V. Chalam, 'Evaluation of ultraviolet light toxicity on cultured retinal pigment epithelial and retinal ganglion cells', Clinical Ophthalmology, 33–39 (2010).

[22] M. Abramowitz and M. W. Davidson, *Anatomy of a Microscope - Numerical Aperture and Resolution*, Visited on 2020-04-16.

[23] N. Foundation, *The nobel prize in physics*, 1929.

[24] W. J. Croft, *Under the microscope [electronic resource] : a brief history of microscopy* (World Scientific Publishing Co. Pte. Ltd., 2006), pp. 57–72.

[25] R. Erni, M. D. Rossell, C. Kisielowski and U. Dahmen, 'Atomic Resolution Imaging with a sub-50 pm Electron Probe', Physical Review Letters **102**, 96–101 (2009).

[26] N. Foundation, *The nobel prize in physics*, 1986.

[27] J. Ayache, L. Beaunier, J. Boumendil, G. Ehret and D. Laub, 'Artifacts in Transmission Electron Microscopy', in *Sample preparation handbook for transmission electron microscopy* (Springer, New York, NY, 2010), pp. 125–170.

[28] J. G. Mcnally, T. Karpova, J. Cooper and J. A. Conchello, 'Three-Dimensional Imaging by Deconvolution Microscopy', Methods **19**, 373–385 (1999).

[29] G. Sparacino, G. De Nicolao, G. Pillonetto and C. Cobelli, 'Deconvolution', in *Modelling methodology for physiology and medicine* (Elsevier, 2014), pp. 45–68.

[30] M. Minsky, 'Memoir on inventing the confocal scanning microscope', Scanning **10**, 128–138 (1988).

[31]M. Minsky, *Microscopy Apparatus*, 1961.

[32]M. Renz, 'Fluorescence microscopy-A historical and technical perspective', Cytometry Part A **83**, 767–779 (2013).

[33]H. Helmholtz, 'Die theoretischen Grenzen für die Leistungsfähigkeit der Mikroskope.', Annalen der Physik, 557–584 (1874).

[34]M. Muyskens and Ed Vitz, 'The Fluorescence of Lignum nephriticum: A Flash Back to the Past and a Simple Demonstration of Natural Substance Fluorescence', Journal of Chemical Education **83**, 765 (2006).

[35]G. G. Stokes, 'On the Change of Refrangibility of Light', Philosophical Transactions of the Royal Society of London **142**, 463–562 (1852).

[36]G. G. Stokes, 'On the Change of Refrangibility of Light. No. II', Philosophical Transactions of the Royal Society of London **143**, 385–396 (1853).

[37]L. J. Kricka and P. Fortina, 'Analytical ancestry: "firsts" in fluorescent labeling of nucleosides, nucleotides, and nucleic acids', Clinical Chemistry **55**, 670–683 (2009).

[38]D. J. S. Birch, Y. Chen and O. J. Rolinski, 'Fluorescence', in *Photonics, volume 4 : biomedical photonics, spectroscopy, and microscopy* (John Wiley & Sons, Incorporated, 2015), pp. 1–58.

[39]G. T. Dempsey, J. C. Vaughan, K. H. Chen, M. Bates and X. Zhuang, 'Evaluation of fluorophores for optimal performance in localization-based super-resolution imaging', Nature Methods **8**, 1027–1036 (2011).

[40]R. Smallman and A. Ngan, 'Characterization and Analysis', in *Modern physical metallurgy* (Elsevier, 2014), pp. 159–250.

[41]H. Siedentopf and R. Zsigmondy, 'Uber Sichtbarmachung und Größenbestimmung ultramikoskopischer Teilchen, mit besonderer Anwendung auf Goldrubingläser', Annalen der Physik **315**, 1–39 (1902).

[42]D. Axelrod, 'Cell-substrate contacts illuminated by total internal reflection fluorescence.', The Journal of Cell Biology **89**, 141–145 (1981).

[43]Y. Markaki and H. Harz, eds., *Light Microscopy*, Vol. 1563, Methods in Molecular Biology (Springer New York, New York, NY, 2017).

[44]B. O. Leung and K. C. Chou, 'Review of super-resolution fluorescence microscopy for biology.', Applied spectroscopy **65**, 967–80 (2011).

[45]S. W. Hell and J. Wichmann, 'Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy', Optics Letters **19**, 780 (1994).

[46]R. C. Dunn, 'Near-field scanning optical microscopy.', Chemical reviews **99**, 2891–928 (1999).

[47]M. G. L. Gustafsson, 'Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution', Proceedings of the National Academy of Sciences **102**, 13081–13086 (2005).

[48]E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz and H. F. Hess, 'Imaging intracellular fluorescent proteins at nanometer resolution.', Science (New York, N.Y.) **313**, 1642–5 (2006).

[49]M. Rust, M. Bates and X. Zhuang, 'Stochastic optical reconstruction microscopy (STORM) provides sub-diffraction-limit image resolution', Nature methods **3**, 793–795 (2006).

[50]S. T. Hess, T. P. K. Girirajan and M. D. Mason, 'Ultra-high resolution imaging by fluorescence photoactivation localization microscopy.', Biophysical journal **91**, 4258–72 (2006).

[51]T. Dertinger, R. Colyer, G. Iyer, S. Weiss and J. Enderlein, 'Fast, background-free, 3D super-resolution optical fluctuation imaging (SOFI).', Proceedings of the National Academy of Sciences of the United States of America **106**, 22287–22292 (2009).

[52]B. Huang, M. Bates and X. Zhuang, 'Super-Resolution Fluorescence Microscopy', Annual Review of Biochemistry **78**, 993–1016 (2009).

[53]D. Wildanger et al., 'Solid immersion facilitates fluorescence microscopy with nanometer resolution and sub-Ångström emitter localization', Advanced Materials **24**, 309–313 (2012).

[54]R. Heintzmann, T. M. Jovin and C. Cremer, 'Saturated patterned excitation microscopy—a concept for optical resolution improvement', Optical Society of America **19**, 1599–1609 (2002).

[55]E. H. Rego, L. Shao, J. J. Macklin, L. Winoto, G. A. Johansson, N. Kamps-Hughes, M. W. Davidson and M. G. Gustafsson, 'Nonlinear structured-illumination microscopy with a photoswitchable protein reveals cellular structures at 50-nm resolution', Proceedings of the National Academy of Sciences of the United States of America **109**, 13–15 (2012).

[56]E. Betzig, 'Proposed method for molecular optical imaging', Optics Letters **20**, 237 (1995).

[57]W. Moerner and L. Kador, 'Optical detection and spectroscopy of single molecules in a solid', Physical Review Letters **62**, 2535–2538 (1989).

[58]M. Orrit and J. Bernard, 'Single pentacene molecules detected by fluorescence excitation in a p-terphenyl crystal', Physical Review Letters **65**, 2716–2719 (1990).

[59]W. P. Ambrose and W. E. Moerner, 'Fluorescence spectroscopy and spectral diffusion of single impurity molecules in a crystal', Nature **349**, 225–227 (1991).

[60]R. M. Dickson, A. B. Cubitt, R. Y. Tsien and W. E. Moerner, 'On / off blinking and switching behaviour of single molecules of green fluorescent protein', **388**, 355–358 (1997).

[61]M. Heilemann, S. van de Linde, A. Mukherjee and M. Sauer, 'Super-resolution imaging with small organic fluorophores.', Angewandte Chemie (International ed. in English) **48**, 6903–8 (2009).

[62]M. Heilemann, S. van de Linde, M. Schüttpelz, R. Kasper, B. Seefeldt, A. Mukherjee, P. Tinnefeld and M. Sauer, 'Subdiffraction-resolution fluorescence imaging with conventional fluorescent probes.', Angewandte Chemie (International ed. in English) **47**, 6172–6 (2008).

[63]L. Stryer and R. P. Haugland, 'Energy transfer: a spectroscopic ruler.', Proceedings of the National Academy of Sciences **58**, 719–726 (1967).

[64]*Single-molecule localization microscopy: directory of smlm software*, http://bigwww.epfl.ch/sr Visited on 2020-06-19, 2017.

[65]P. D. Simonson, E. Rothenberg and P. R. Selvin, 'Single-molecule-based super-resolution images in the presence of multiple fluorophores.', Nano letters **11**, 5090–6 (2011).

[66]S. A. Jones, S.-H. Shim, J. He and X. Zhuang, 'Fast, three-dimensional super-resolution imaging of live cells', Nature Methods **8**, 499–505 (2011).

[67]A. Small and S. Stahlheber, 'Fluorophore localization algorithms for super-resolution microscopy.', Nature methods **11**, 267–79 (2014).

[68]A. V. Abraham, S. Ram, J. Chao, E. S. Ward and R. J. Ober, 'Quantitative study of single molecule location estimation techniques', Optics Express **17**, 23352 (2009).

[69]R. E. Thompson, D. R. Larson and W. W. Webb, 'Precise nanometer localization analysis for individual fluorescent probes.', Biophysical journal **82**, 2775–2783 (2002).

[70]S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice Hall, 1993).

[71]R. Henriques, M. Lelek, E. F. Fornasiero, F. Valtorta, C. Zimmer and M. M. Mhlanga, 'QuickPALM: 3D real-time photoactivation nanoscopy image processing in ImageJ.', Nature methods **7**, 339–40 (2010).

[72]R. Parthasarathy, 'Rapid, accurate particle tracking by calculation of radial symmetry centers', Nature Methods **9**, 724–726 (2012).

[73]A. J. Berglund, M. D. McMahon, J. J. McClelland and J. A. Liddle, 'Fast, bias-free algorithm for tracking single particles with variable size and shape', Optics Express **16**, 14064 (2008).

[74]H. Ma, F. Long, S. Zeng and Z.-L. Huang, 'Fast and precise algorithm based on maximum radial symmetry for single molecule localization', Optics Letters **37**, 2481 (2012).

[75]T. Quan, H. Zhu, X. Liu, Y. Liu, J. Ding, S. Zeng and Z.-L. Huang, 'High-density localization of active molecules using Structured Sparse Model and Bayesian Information Criterion', Optics Express **19**, 16963 (2011).

[76]Y. Wang, T. Quan, S. Zeng and Z.-L. Huang, 'PALMER: a method capable of parallel localization of multiple emitters for high-density localization microscopy', Optics Express **20**, 16039 (2012).

[77]F. Huang, S. L. Schwartz, J. M. Byars and K. a. Lidke, 'Simultaneous multiple-emitter fitting for single molecule super-resolution imaging.', Biomedical optics express **2**, 1377–1393 (2011).

[78]F. Huang et al., 'Video-rate nanoscopy using sCMOS camera-specific single-molecule localization algorithms.', Nature methods **10**, 653–8 (2013).

[79]S. J. Holden, S. Uphoff and A. N. Kapanidis, 'DAOSTORM: an algorithm for high-density super-resolution microscopy', Nature methods **8**, 279–280 (2011).

[80]D. R. Tobergte and S. Curtis, 'Daophot: a Computer Program for Crowded-Field Stellar Photometry (Psf Photometry)', Journal of Chemical Information and Modeling **53**, 1689–1699 (2013).

[81]L. B. Lucy, 'An iterative technique for the rectification of observed distributions', Astronomical Journal **79**, 745–754 (1974).

[82]W. Richardson, 'Bayesian-based iterative method of image restoration', Optical Society of America **62**, 55–59 (1972).

[83]L. Zhu, W. Zhang, D. Elnatan and B. Huang, 'Faster STORM using compressed sensing', Nature Methods **9**, 721–723 (2012).

[84]S. Cox, E. Rosten, J. Monypenny, T. Jovanovic-Talisman, D. T. Burnette, J. Lippincott-Schwartz, G. E. Jones and R. Heintzmann, 'Bayesian localization microscopy reveals nanoscale podosome dynamics.', Nature methods **9**, 195–200 (2012).

[85]S. Wolter, U. Endesfelder, S. van de Linde, M. Heilemann and M. Sauer, 'Measuring localization performance of super-resolution algorithms on very active samples.', Optics express **19**, 7020–33 (2011).

[86]S. Wang, J. R. Moffitt, G. T. Dempsey, X. S. Xie and X. Zhuang, 'Characterization and development of photoactivatable fluorescent proteins for single-molecule-based superresolution imaging', Proceedings of the National Academy of Sciences of the United States of America **111**, 8452–8457 (2014).

[87]N. Durisic, L. Laparra-Cuervo, Á. Sandoval-Álvarez, J. S. Borbely and M. Lakadamyali, 'Single-molecule evaluation of fluorescent protein photoactivation efficiency using an in vivo nanotemplate', Nature Methods **11**, 156–162 (2014).

[88]P. J. Green, 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', Biometrika **82**, 711 (1995).

[89]H. Jeffreys, *Theory of probability*, 3rd (Press, Oxford: University, 1961).

[90]L. Savage, *The foundations of Statistics* (New York: Wiley, 1972).

[91]B. De Finetti, *Theory of probability* (London: Wiley, 1974).

[92]R. Christensen, T. E. Hanson, W. Johnson, Branscum and Adam, *Bayesian Ideas and Data Analysis : An Introduction for Scientists and Statisticians*, 1st ed. (Taylor & Francis Group, 2010).

[93]S. Chib and E. Greenberg, 'Understanding the Metropolis-Hastings Algorithm', The American Statistician **49**, 327–335 (1995).

[94]M. Jesper, ed., *Spactial Statistics and Computational Methods* (Springer, 2003), pp. 1–13.

[95]K. I. Mortensen, L. S. Churchman, J. A. Spudich and H. Flyvbjerg, 'Optimized localization analysis for single-molecule tracking and super-resolution microscopy', Nature Methods **7**, 377–381 (2010).

[96]J. R. Janesick, *Photon Transfer* (SPIE.Digital Library, 2007).

[97]I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, `http://www.deeplearningbook.org` (MIT Press, 2016).

[98]S. Mor-Yosef, A. Samueloff, B. Modan, D. Navot and J. G. Schenker, 'Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study.', Obstetrics and gynecology **75**, 944–7 (1990).

[99] V. Metsis, I. Androutsopoulos and G. Paliouras, 'Spam Filtering with Naive Bayes – Which Naive Bayes?', In CEAS (2006).

[100] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, http://www.deeplearningbook.org (MIT Press, 2016).

[101] *Neural Network Programming - Deep Learning with PyTorch*, https://deeplizard.com/learn/play Accessed: 2020-07-08.

[102] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', Nature **323**, 533–536 (1986).

[103] M. Torrisi, G. Pollastri and Q. Le, 'Deep learning methods in protein structure prediction', Computational and Structural Biotechnology Journal **18**, 1301–1310 (2020).

[104] *Single-molecule localization microscopy: collection of reference datasets*, http://bigwww.epfl.ch/smlm/datasets/index.html, Visited on 2020-06-19, 2017.

[105] H. Li and J. C. Vaughan, 'Switchable Fluorophores for Single-Molecule Localization Microscopy', Chemical Reviews **118**, 9412–9454 (2018).

[106] B. A. Griffin, S. R. Adams, R. Y. Tsien, B. A. Griffin, S. R. Adams and R. Y. Tsient, 'Specific Covalent Labeling of Recombinant Protein Molecules Inside Live Cells Published by : American Association for the Advancement of Science Stable URL : http://www.jstor.org/stable/2896025 REFERENCES Linked references are available on JSTOR for this', Science **281**, 269–272 (1998).

[107] L. W. Miller, Y. Cai, M. P. Sheetz and V. W. Cornish, 'In vivo protein labeling with trimethoprim conjugates: A flexible chemical tag', Nature Methods **2**, 255–257 (2005).

[108] A. Keppler, S. Gendreizig, T. Gronemeyer, H. Pick, H. Vogel and K. Johnsson, 'A general method for the covalent labeling of fusion proteins with small molecules in vivo', Nature Biotechnology **21**, 86–89 (2003).

[109] A. Gautier, A. Juillerat, C. Heinis, I. R. Corrêa, M. Kindermann, F. Beaufils and K. Johnsson, 'An Engineered Protein Tag for Multiprotein Labeling in Living Cells', Chemistry and Biology **15**, 128–136 (2008).

[110] G. V. Los et al., 'HaloTag: A novel protein labeling technology for cell imaging and protein analysis', ACS Chemical Biology **3**, 373–382 (2008).

[111] I. Y. Iourov, *Fluorescence In Situ Hybridization (FISH)*, edited by T. Liehr, Springer Protocols Handbooks (Springer Berlin Heidelberg, Berlin, Heidelberg, 2017), pp. 17–25.

[112]M. E. Brezinski, *Noise and system performance with td-oct and sd-oct* (Academic Press, Amsterdam ; 2006), pp. 175–195.

[113]T. Bushnell, *What is autofluorescence*, https://expert.cheekyscientist.com/what-is-autofluorescence/, Visited on 2020-06-09.

[114]B. Huang, W. Wang, M. Bates and X. Zhuang, 'Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy.', Science (New York, N.Y.) **319**, 810–3 (2008).

[115]M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych and G. M. Hagen, 'ThunderSTORM: A comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging', Bioinformatics **30**, 2389–2390 (2014).

[116]B. A. Cipra, 'An Introduction to the Ising Model', The American Mathematical Monthly **94**, 937–959 (1987).

[117]I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, 'Chapter 9 - Probabilistic methods', in *Data mining practical machine learning tools and techniques*, edited by I. H. Witten, E. Frank, M. A. Hall and C. J. B. T. D. M. (E. Pal (Morgan Kaufmann, 2017), pp. 335–416.

[118]B. A. Landman, I. Lyu, Y. Huo and A. J. Asman, 'Chapter 6 - Multiatlas segmentation', in *Handbook of medical image computing and computer assisted intervention*, edited by S. K. Zhou, D. Rueckert, G. B. T. H. o. M. I. C. Fichtinger and C. A. Intervention (Academic Press, 2020), pp. 137–164.

[119]Orchard, Peter, *Markov random field optimisation*, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0809/ORCHARD/, Accessed 2014-08-28.

[120]Q. Jackson and D. Landgrebe, 'Adaptive Bayesian contextual classification based on Markov random fields', Geoscience and Remote Sensing, ... **40**, 2454–2463 (2002).

[121]A. Sonnleitner, G. Schütz and T. Schmidt, 'Free Brownian Motion of Individual Lipid Molecules in Biomembranes', Biophysical Journal **77**, 2638–2642 (1999).

[122]M. K. Cheezum, W. F. Walker and W. H. Guilford, 'Quantitative comparison of algorithms for tracking single fluorescent particles', Biophysical Journal **81**, 2378–2388 (2001).

[123]U. Kubitscheck, O. Kückmann, T. Kues and R. Peters, 'Imaging and Tracking of Single GFP Molecules in Solution', Biophyiscal Journal **78**, 2170–2179 (2000).

[124]*Markov random fields*, https://www2.isye.gatech.edu/isyebayes/bank/handout16.pdf, Accessed August 11, 2022.

[125] *Classification Learner*, https://uk.mathworks.com/help/stats/classificationlearner-app.html, Accessed: 2020-08/-8.

[126] M. J. Rust, M. Bates and X. Zhuang, 'Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM).', Nature methods **3**, 793–5 (2006).

[127] R. P. Nieuwenhuizen, M. Bates, A. Szymborska, K. A. Lidke, B. Rieger and S. Stallinga, 'Quantitative localization microscopy: Effects of photophysics and labeling stoichiometry', PLoS ONE **10**, 1–18 (2015).

[128] D. Gross and W. Webb, 'Molecular counting of low-density lipoprotein particles as individuals and small clusters on cell surfaces', Biophysical Journal **49**, 901–911 (1986).

[129] B. M. Burton, K. A. Marquis, N. L. Sullivan, T. A. Rapoport and D. Z. Rudner, 'The ATPase SpoIIIE Transports DNA across Fused Septal Membranes during Sporulation in Bacillus subtilis', Cell **131**, 1301–1312 (2007).

[130] M. C. Leake, J. H. Chandler, G. H. Wadhams, F. Bai, R. M. Berry and J. P. Armitage, 'Stoichiometry and turnover in single, functioning membrane protein complexes', Nature **443**, 355–358 (2006).

[131] S. K. Das, M. Darshi, S. Cheley, M. I. Wallace and H. Bayley, 'Membrane protein stoichiometry determined from the step-wise photobleaching of dye-labelled subunits', ChemBioChem **8**, 994–999 (2007).

[132] K. Tsekouras, T. C. Custer, H. Jashnsaz, N. G. Walter and S. Pressé, 'A novel method to accurately locate and count large numbers of steps by photobleaching', Molecular Biology of the Cell **27**, edited by D. Lidke, 3601–3615 (2016).

[133] A. Lee, K. Tsekouras, C. Calderon, C. Bustamante and S. Pressé, 'Unraveling the Thousand Word Picture: An Introduction to Super-Resolution Data Analysis', Chemical Reviews **117**, 7276–7330 (2017).

[134] E. Alpaydin, 'Introduction', in *Introduction to machine learning* (2010), pp. 1–19.

[135] C. Campbell and Y. Ying, 'Learning with Support Vector Machines', Synthesis Lectures on Artificial Intelligence and Machine Learning **5**, 1–95 (2011).

[136] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (Cambridge University Press, Mar. 2000).

[137]V. Hodge and J. Austin, 'A Survey of Outlier Detection Methodologies', Artificial Intelligence Review **22**, 85–126 (2004).

[138]M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks', Neural Networks **106**, 249–259 (2018).

[139]L. Deng, 'The mnist database of handwritten digit images for machine learning research [best of the web]', IEEE Signal Processing Magazine **29**, 141–142 (2012).

[140]S. L. Smith, P.-j. Kindermans, C. Ying, Q. V. Le and G. Brain, 'D on ' T D Ecay the L Earning R Ate , I Ncrease the B Atch S Ize', 1–11 (2018).

[141]*Cross entropy loss*, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, Accessed April 02, 2021.

[142]Z. Zhang and M. R. Sabuncu, 'Generalized cross entropy loss for training deep neural networks with noisy labels', Advances in Neural Information Processing Systems **2018-December**, 8778–8788 (2018).

[143]*Cross-entropy loss function*, https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e, Accessed April 02, 2021.

A

APPENDIX A: INTRODUCTION

A.1 EXAMPLE MARKOV CHAIN MONTE CARLO (MCMC) CODE

```
function result = GIBBS(matrixData, M)

%constants from data
n = length(matrixData);
xbar = mean(matrixData);
s2 = var(matrixData);

% to be saved
%sal.mu = NaN(1,M);
%sal.lamda = NaN(1,M);

%constants for lam prior
a=0.1;
b=0.1;
%constants for m prior
nu = 0;
tau = 0.2;
%constants for p prior
alpha = 0.1;
beta = 0.1;

%constants in posterior
aStar = a+n/2;
alphaStar = alpha+ 1/2;

%initialising
mu = NaN(1,M);
mu(1) = 1;
lambda = NaN(1,M);
```

```
lambda(1) = 1;
m = NaN(1,M);
m(1) = 1;
p = NaN(1,M);
p(1) = .1;



for i = 2:M
   % for mu
%   keyboard
    pstar = lambda(i-1)*n+p(i-1);
    mstar = (n*lambda(i-1)*xbar + p(i-1)*m(i-1))/pstar;
   % draw a mu
    mu(i) = normrnd(mstar, 1/sqrt(pstar) );
    %result.mu(i) = mu;

   % for lambda
    bstar = b + (n*(mu(i)-xbar)^2)/2 + n*s2/2;
   % draw a lambda
    lambda(i) = gamrnd(aStar,1/bstar);
    %result.lambda(i) = lambda;

   %for p
    betaStar = beta + (mu(i)-m(i-1))^2;
   % draw a p
    p(i) = gamrnd(alphaStar, 1/betaStar);

   % for m
   taustar = (p(i)+tau);
   nustar = (2*p(i) + tau*nu)/taustar;
   % draw an m
    m(i) = normrnd(nustar, 1/sqrt(taustar));


end

result.mu = mu;
result.lambda = lambda;
```

A.2 CONTINUOUS DISTRIBUTIONS

The different continuous distributions that were used in this thesis are included in table A.1

Name	Notation	p.d.f.	Mean $\mathbb{E}[X \mid \boldsymbol{\theta}]$	Variance $\mathbb{V}[X \mid \boldsymbol{\theta}]$	Notes
Beta	$\mathrm{Be}(x \mid \alpha, \beta)$	$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$ $X = (0,1)$ $\Theta = \{(\alpha,\beta) \in \mathbb{R}^2 : \alpha > 0, \beta > 0\}$	$\frac{\alpha}{\alpha+\beta}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$	
Gamma	$\mathrm{Ga}(x \mid \alpha, \beta)$	$f(x) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$ $\Gamma(n) = (n-1)!$ $X = \mathbb{R}_+$ $\Theta = \{(\alpha,\beta) \in \mathbb{R}^2 : \alpha > 0, \beta > 0\}$	$\frac{\alpha}{\beta}$	$\frac{\alpha}{\beta^2}$	
Gaussian (Normal)	$\mathcal{N}(x \mid \mu, \lambda)$	$f(x) = \frac{\lambda^{\frac{1}{2}}}{\sqrt{2\pi}} \exp[-\frac{\lambda}{2}(x-\mu)^2]$ $X = \mathbb{R}$ $\Theta = \{(\mu,\lambda) \in \mathbb{R}^2 : \lambda > 0\}$	μ	$\frac{1}{\lambda}$	Can also be parameterised in terms of the standard deviation $\sigma = \frac{1}{\lambda^{1/2}}$

Table A.1: Continuous distributions

B

APPENDIX B: SIMULATIONS

B.1 SIMULATIONS CODE

```
foldername1 = 'SimTest_11_6_2020/images-';
mkdir 'SimTest_11_6_2020/images-'

foldername2 = 'nic-';


tic
Num_mols = 5000;
frame_num = 500;
frame_time = 10;

% simulate on off
M = MolsMatrix_test(Num_mols, frame_num, frame_time);

% create empty matrix
M_binned = NaN(Num_mols, frame_num);
% sum into frames
for i = 1:frame_num
     first = (i-1)*10 +1;
     last = i*10;
     temp = M(:,first:last);
     a = find(temp == -1);
     temp(a) = 0;
     b = sum(temp,2);
     M_binned(:,i) = b;
 end % for

 %define paramaters needed
```

```
FOV = 10000;
PSF = .004;
pixelsize = 100;

offset = 100;
efficiency = 0.7;

noisevalue = 50;

% create positions for each molecule
moleculelist = NaN(Num_mols,3);

% Center = round(FOV * .5);
% for i = 1:5
%     for j = 1:5
%         mol = j + 5*(i-1);
%
%         % 10nm apart
%         x_pos = Center + (i * 10 - 10);
%         y_pos = Center + (j * 10 - 10);
%
%         moleculelist(mol,1) = x_pos;
%         moleculelist(mol,2) = y_pos;
%     end % j
% end % i
areaMols= FOV* 0.8;
%moleculelist(:,1)=randi(FOV, Num_mols,1);
%moleculelist(:,2)=randi(FOV, Num_mols,1);
moleculelist(:,1)=randi(areaMols, Num_mols,1);
moleculelist(:,2)=randi(areaMols, Num_mols,1);

moleculelist = moleculelist+(FOV*0.1);

% create image stack

%foldername = 'Data_2July/image-';
partc='.tif';

% mean of the exponential
%mu = 5202;
%mu = 7500;
mu = 6019
```

```
for i = 1:frame_num
    map=zeros(FOV);
%     moleculelist(:,3) = exprnd(mu, Num_mols, 1).*(M_binned(:,i)/10);
    moleculelist(:,3) = poissrnd(mu, Num_mols, 1).*(M_binned(:,i)/10);

    map = addmolecules(map,moleculelist);


    %fourier transform that image
    B = fft2(map);
    %make optical transfer function
    R=makemask(PSF, FOV);
    B = B.*R;

    %fourier transform back into real space
    Y = ifft2(B);
    Y = real(Y.*Y);

    % capture on camera (bin in space)
    [captureimage, loopnumber] = capturecamera(FOV, pixelsize,
Y,efficiency);

    %add noise level
    captureimage = round(captureimage) + offset;

    %change so size is automated?
%     noisemap = gauspdf(100,300,0.24);
%     noisemap = normpdf(100,300,0.24);
 %   noisemap = normrnd(300,20,[100]);
    noisemap = poissrnd(20,100);

    captureimage = captureimage + noisemap;

    imagenumber = num2str(i,'%05d');
    s = strcat(foldername1,foldername2, imagenumber, partc);
    imwrite(uint16(captureimage),s);

end %for


s = strcat(foldername1,'matlab.mat');
```

```
%keyboard
save(s)

toc1
```

B.1.1 *Adding molecules to the image*

```
function [ map ] = addmolecules( map,moleculelist )
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
for i =1:size(moleculelist,1)
%     keyboard
    map(moleculelist(i,1),moleculelist(i,2))=moleculelist(i,3);
end
end
```

B.1.2 *Creating a mask*

```
function [ R ] = makemask( r,SX )
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here


x = linspace(-1,1,SX);
y = linspace(-1,1,SX);
[X,Y] = meshgrid(x,y);
R = sqrt(X.^2 + Y.^2);
%keyboard
R(R>r) = 0;
R(find(R)) = 1;

end
```

B.1.3 *Binning the data in space*

```
function[captureimage,loopnumber] = capturecamera(FOV,pixelsize,
Y,efficiency)
%UNTITLED5 Summary of this function goes here
```

```
%   Detailed explanation goes here
loopnumber = round(FOV/pixelsize);
captureimage=zeros(loopnumber+1);
for i = 1:loopnumber-1
    for j = 1:loopnumber-1
        captureimage(i,j) = efficiency*sum(sum(Y(((i-1) * pixelsize)
+ 1 :(i*pixelsize) + 1, (( j-1 ) * pixelsize) + 1:((j*pixelsize)+1))));
    end
end

captureimage=captureimage(1:loopnumber,1:loopnumber);

end
```

C

APPENDIX C: GIBBS ISING

C.1 GIBBS ISING CODE

```
function result = Gibbs_Ising_Jul2018(matrixData)
M = 1e4;

validateattributes( matrixData, {'double'} , ...
{'2d', 'real', 'nonempty', 'finite', 'nonnan'} )

[m,n] = size(matrixData);
Gibbs_threshold = 0.8;

%%  Run through Gibbs
datavec = reshape(matrixData, [],1);
result.Gibbsdata = twoPopLoopJul2018( datavec, M );

Gibbs_noBurnin = result.Gibbsdata.Z(:, M/4+1:end);

Gibbs_a = mean(Gibbs_noBurnin,2);
Gibbs_b = reshape(Gibbs_a, [m,n]);
result.Gibbs_probs = Gibbs_b;
result.Gibbs_map = Gibbs_b > Gibbs_threshold;
result.Gibbs_image = result.Gibbs_map .* matrixData;

%% Run through Ising
result.Ising = Ising_Jul2018(Gibbs_b);

end % function
```

C.1.1 *Gibbs Sampler for two populations*

```
function [ gibbs , ising ] = twoPopLoopJul2018( datavec, M )
%TWOPOPLOOPJUL2018 Summary of this function goes here
%   Detailed explanation goes here

validateattributes( datavec, {'double'} , ...
{'2d', 'real', 'nonempty', 'finite', 'nonnan'} )
validateattributes( M , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'integer', 'finite', 'nonnan', 'positive'} )

%% Prior values - precision beta (a, b) mean gaussian (m,p)
%signal
a_sig = 0.01;
b_sig = 0.01;
%m_sig = 200;
m_sig = quantile(datavec,0.975);
p_sig = 0.05;
%p_sig = 1;

%noise
a_noise = 0.01;
b_noise = 0.01;
%m_noise = 140;
m_noise = quantile(datavec,0.025);

p_noise = 0.01;

%% find values needed and preallocate space for results
N = length(datavec);
gibbs.muNoise = NaN(1,M);
gibbs.muSignal = NaN(1,M);
gibbs.lamNoise = NaN(1,M);
gibbs.muSignal = NaN(1,M);
gibbs.Pn = NaN(M,1);
gibbs.Z = NaN(N,M);

index = SplitByAverage_Jul2018( datavec );


for i = 1:M
    % Seperate into 2 populations using index
```

```
    [data_S, data_N] = seperate_Jul2018(index,datavec);

    % calculate some values
    a = sum(index); %number of signal data points
    b = N - a; % number of noise data points
    % calculate the probability of noise pixels from beta
distribution
    % (prior = beta(1/2, 1/2) )
    Pn = betarnd(a + 1/2, b + 1/2);
    gibbs.Pn(i) = Pn;

    %Gibbs Sampler for both signal and noise
    Signal = GibbsGauss_Jul2018(data_S,1,a_sig,b_sig,m_sig,
p_sig);
    Noise = GibbsGauss_Jul2018(data_N,1,a_noise,b_noise,
m_noise,p_noise);

    % ensure they haven't swopped
    if Signal.mu > Noise.mu
        gibbs.muNoise(i) = Noise.mu;
        gibbs.lamNoise(i) = Noise.lam;
        gibbs.muSignal(i) = Signal.mu;
        gibbs.lamSignal(i) = Signal.lam;
    else
        gibbs.muNoise(i) = Signal.mu;
        gibbs.lamNoise(i) = Signal.lam;
        gibbs.muSignal(i) = Noise.mu;
        gibbs.lamSignal(i) = Noise.lam;
    end

    probabilities = prob1_Jul2018(datavec,gibbs.muNoise(i), ...
gibbs.muSignal(i),gibbs.lamNoise(i),gibbs.lamSignal(i), Pn);
    index = binornd(1,(1-probabilities));

    % check for no empty sets (must have data in both
signal and noise data
    % sets
    dummy = 1;
    while dummy>0
        n = sum(index);
        if n<1 || n == N
            idx = newZ(probabilities);
```

```
        else
            break;
        end %if
    end %while

    gibbs.Z(:,i) = index;
    % how far through the code has it gone?
    if ~mod(i,M/10)
        disp([num2str(i/M*100),'% done'])
    end % if

end %for


end %function
```

C.1.2 *Separating the signal and noise*

```
function [noise, signal] = seperate_Jul2018( index, datavec )
%SEPERATE Seperates datavec into two populations using index
%   Detailed explanation goes here

validateattributes( datavec , {'double'} , ...
  {'2d', 'real', 'nonempty', 'finite', 'nonnan'} )
validateattributes( index , {'double'} , ...
  {'2d', 'real', 'nonempty', 'finite', 'nonnan'} )

noise = datavec(~logical(index));
signal = datavec(logical(index));

end %function
```

C.1.3 *Gibbs Sampler for one population*

```
function result = GibbsGauss_Jul2018(datavec,M,a,b,m,p)
%GIBBSGAUSS_JUL2018 Gibbs Sampler
%   Detailed explanation goes here
validateattributes( datavec, {'double'} , {'2d', 'real', 'nonempty',
'finite', 'nonnan'} )
validateattributes( M , {'double', 'single'} , ...
```

```
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)
validateattributes( a , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)
validateattributes( b , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)
validateattributes( m , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan'} )
validateattributes( p , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)

%% Statistics
n = length(datavec);
xbar = mean(datavec);
s2 = var(datavec);

%% update a (stays constant throughout)
astar = n/2 +a;

%% allocate data space for mu and lam
result.mu = NaN(M,1);
result.lam = NaN(M,1);

%% starting point
lam_cur = 1;

for i = 1:M
    pstar = n*lam_cur +p;
    mstar = (n*lam_cur*xbar + m*p)/pstar;
    mu_cur = normrnd(mstar, 1/sqrt(pstar));
    bstar = b+ n/2*( s2 + (xbar - mu_cur)^2);
    lam_cur = gamrnd(astar, 1/bstar);
    result.mu(i) = mu_cur;
    result.lam(i) = lam_cur;

end %for
end %function


function result = prob1_Jul2018(data,muNoise,muSignal,
```

```
lamNoise,lamSignal,Pn)
%PROB1_JUL2018 Summary of this function goes here
%   Detailed explanation goes here

validateattributes( data, {'double'} , ...
{'2d', 'real', 'nonempty', 'finite', 'nonnan'} )
validateattributes( muNoise , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)
validateattributes( muSignal , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)
validateattributes( lamNoise , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)
validateattributes( lamSignal , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)
validateattributes( Pn , {'double', 'single'} , ...
{'scalar', 'real', 'nonempty', 'finite', 'nonnan', 'positive'}
)

%%
B = (lamSignal*(data - muSignal).^2 - lamNoise*(data - muNoise).^2);
A = sqrt(lamSignal/lamNoise)*exp(-1/2*B);

P = (1-Pn)./Pn;

S = (A).*P;

result = 1./(1+S.*P);

end %function


function sal = newZ(probabilities)

% N = length(probabilities);
%sal = NaN(N,1);

sal = binornd(1,probabilities);

%for i = 1:N
```

```
 %   sal(i,1) = binornd(1,probabilities(i,1));
%end


function result = Ising_Jul2018( probs_matrix )
%ISING_JUL2018 Summary of this function goes here
%   Detailed explanation goes here

%check inputs
validateattributes( probs_matrix, {'double'} , ...
  {'2d', 'real', 'nonempty', 'finite', 'nonnan'} )

%%  make so initial conditions are always the same (repeatable)
randn('state',3) %set the seeds (state) to have
rand ('state',3) %the constancy of results

%% set some variables (initial)

z = 1e4;  %number of iterations
sigma = .5;      %must be less than 3(2)
% J(1) = 1;       %Reciprocal Temperature

%priors
a = 2;
b= 0.6;

% make proabilities go from -1 to 1 (not 0 to 1)
y = 2*probs_matrix - 1;

s = size(probs_matrix);
theta = ones(s(1),s(2)); %start with ones

%%
k = 1;
randvec = randperm( s(1)*s(2) );
J = NaN(z,1);  % preallocate space for reciprocal temperature
J(k) = gamrnd(a,b);  % first reciprocal temp from a gamma dist
(priors above)


IsingMatrix = zeros(s(1),s(2));  %start with zeros

for j = 1:z
```

```
    % disp how much is done - needs changing
    if ~mod(j,z/10)
        disp([num2str(j/z*100),'% done'])
    end

    for pos = randvec   % go through pixels in a random order
determined above
        thetaP = -theta(pos);  % propose that the pizel changes
state
        LikeRatio = exp(y(pos)*(thetaP - theta(pos))/sigma.^2);
% how likely is it to change

        neighborhood = pos + [-1,1,-s(1),s(1)];

        [iy,ix]=ind2sub([s(1),s(2)],pos); %change from ind to
x,y

        % using cliques
        neighborhood([iy==1,iy==s(1),ix==1,ix==s(2)]) = [];
        disagree = sum(theta(neighborhood)~=theta(pos)); % how
many disagree
        disagreeP = sum(theta(neighborhood)~=thetaP); % how many
disagre with the proposal
        DelLogPr = 2 * J(k) * (disagree - disagreeP);
        alpha = exp(DelLogPr) * LikeRatio;
            if rand < alpha  % if a random number is smaller than
alpa keep the propossal
                theta(pos) = thetaP;
                disagree = disagreeP;
            end %if

        % update J (only b changes, a is constant)
        bstar = b + 2*disagree;
        J(k + 1) = gamrnd(a,bstar);
        J_rat = J(k + 1)/J(k);
        J_dif = J(k+1)-J(k);
        alpha2 = ((J_rat)^(a-1))*exp(-2*bstar*J_dif);
            if rand > alpha2  % keep new J with prob rand
                J(k + 1) = J(k);
            end %if
```

```
        % update cliques model (add or take one from the relative
position)
       IsingMatrix(pos) = IsingMatrix(pos) + theta(pos);
       k = k+1;

    end %for

end %for

IsingMatrix = IsingMatrix/z; %get back to numbers between -1 and
1
IsingMatrix = (IsingMatrix + 1)/ 2; % change to between 0 and
1

result = IsingMatrix;

end % function
```

C.2 SENSITIVITY TEST RESULTS

Table C.1: Table shows the different prior values chosen for the sensitivity test performed on figure 4.2a. The results are shown in table C.2.

	Noise Distribution				Signal Distribution			
test #	m_n	p_n	a_n	b_n	m_s	p_s	a_s	b_s
1	2.5th quantile	0.01	2	0.6	97.5th quantile	0.01	2	0.6
2	2.5th quantile	0.01	0.01	0.01	97.5th quantile	0.01	2	0.6
3	2.5th quantile	0.01	2	0.6	97.5th quantile	0.01	0.01	0.01
4	2.5th quantile	0.01	0.01	0.01	97.5th quantile	0.01	0.01	0.01
5	2.5th quantile	0.5	2	0.6	97.5th quantile	0.5	2	0.6
6	2.5th quantile	0.5	0.01	0.01	97.5th quantile	0.5	2	0.6
7	2.5th quantile	0.5	2	0.6	97.5th quantile	0.5	0.01	0.01
8	2.5th quantile	0.5	0.01	0.01	97.5th quantile	0.5	0.01	0.01
9	2.5th quantile	0.01	2	0.6	97.5th quantile	0.5	2	0.6
10	2.5th quantile	0.01	0.01	0.01	97.5th quantile	0.5	2	0.6
11	2.5th quantile	0.01	2	0.6	97.5th quantile	0.5	0.01	0.01
12	2.5th quantile	0.01	0.01	0.01	97.5th quantile	0.5	0.01	0.01
13	2.5th quantile	0.5	2	0.6	97.5th quantile	0.01	2	0.6
14	2.5th quantile	0.5	0.01	0.01	97.5th quantile	0.01	2	0.6
15	2.5th quantile	0.5	2	0.6	97.5th quantile	0.01	0.01	0.01
16	2.5th quantile	0.5	0.01	0.01	97.5th quantile	0.01	0.01	0.01

Table C.2: Table shows the mean values for μ_n, λ_n, μ_s and λ_s for each of the sensitivity tests performed on figure 4.2a. The priors used in each of these test are shown in table C.1

sim test #	μ_n	$\lambda_n/10^{-3}$	μ_s	$\lambda_s/10^{-4}$
1	187.182	2.168	269.8683	6.2783
2	187.1987	2.176	269.6278	6.2774
3	187.1798	2.151	270.6738	6.2684
4	187.2036	2.150	270.8551	6.2653
5	187.2348	2.171	269.8886	6.2771
6	187.2651	2.171	269.9749	6.2741
7	187.2302	2.156	270.6084	6.2698
8	*187.2659*	**2.147**	*271.1548*	**6.2598**
9	187.2341	2.169	269.9961	6.2751
10	187.2611	*2.178*	269.6512	6.2773
11	187.2367	2.154	270.6619	6.2710
12	187.2612	2.155	270.7730	6.2659
13	**187.1817**	2.175	**269.5224**	*6.2810*
14	189.1944	2.177	269.5814	6.2792
15	187.1915	2.153	270.6397	6.2695
16	187.194	2.153	270.6731	6.2657
% difference	0.045%	1.44%	0.6057%	0.33867%

D

APPENDIX D: COUNTING

D.1 THE TWO STAGE GIBBS ALGOIRHTM

```
function sal = doublegibbs(matrixdata, M)
% input data, run Gibbs Sampler to seperate signal from noise.
 Then run signal
% through gibbs sampler againg to seperate signal into S1 and
S2 (S1 is
% higher than S2)

%subplot(2,2,1)
%imagesc(matrixdata)
%keyboard
%threshGibbsNew
tic
datavec = reshape(matrixdata,[],1);
% Run through Gibbs Sampler to classify into signal and noise
sal.gibbs1 = twoPopLoopNew(datavec,M);

% find the average of the Z matrix along the second dimension
(finds the
% probability of each pixel being noise
mz = mean(1-sal.gibbs1.Z,2);
s = size(matrixdata);
% threshold the data (> .7) to classify pixels
mask = threshold(mz,s(1),s(2));
sal.mask = mask;
% set all the pixels classified as noise to zero
sal.gibbsimage = mask.*matrixdata;
%subplot(2,2,2)
%imagesc(sal.gibbsimage)
```

```
toc

%indexSignal
ind = find(mask);
signal = matrixdata(logical(mask));

%sal.signal = [signal, ind];

%gibbsvector (gibbs2)
sal.gibbs2 = twoPopLoopNew2(signal ,M);

sal.S12 = mean(1-sal.gibbs2.Z,2);

sal.signal = [signal, ind, sal.S12];

sal.idx = sal.S12 > 0.7;

%noise
noise = matrixdata(logical(~mask));
Npos = find(~mask);
sal.N = [noise, Npos];

%s1
S1 = sal.signal(logical(sal.idx));
S1pos = ind(logical(sal.idx));
sal.S1 = [S1, S1pos];

%s2
S2 = sal.signal(~logical(sal.idx));
S2pos = ind(~logical(sal.idx));
sal.S2 = [S2, S2pos];

Nsize = size(sal.N);
N0vector = zeros(Nsize(1),1);
N = [N0vector, sal.N];

S1size = size(sal.S1);
S11vector = ones(S1size(1),1);
S12vector = 2*S11vector;
S1 = [S12vector, sal.S1];

S2size = size(sal.S2);
```

```
S21vector = ones(S2size(1),1);
%S22vector = 2*S21vector;
S2 = [S21vector, sal.S2];

all =[N; S1; S2];
order = sortrows(all, 3);

s = size(matrixdata);
sorted = reshape(order(:,1),s(1),s(2));
sal.sorted = sorted;

%subplot(2,2,3)
%imagesc(sorted)

orig = reshape(order(:,2),s(1),s(2));
%subplot(2,2,4)
%imagesc(orig)


subplot(2,3,1)
imagesc(matrixdata)
subplot(2,3,2)
imagesc(sal.gibbsimage)
subplot(2,3,3)
plot(sal.gibbs1.muNoise)
hold on
plot(sal.gibbs1.muSignal)
hold off
subplot(2,3,4)
imagesc(sorted)
subplot(2,3,5)
plot(sal.gibbs2.muSignal1)
hold on
plot(sal.gibbs2.muSignal2)
hold off
subplot(2,3,6)
imagesc(orig)
```

D.2 USING THE GIBBS STATISTICS TO CALCULATE THE CHANGE IN 'ON' FLUOROPHORES

```
filelist_images = LM_filelist('/Users/Nicola/Google Drive/code_data
/matlab/SimCode_Nic/20Bleaching_1/Bleaching20_12P_many/images');
filelist_GibbsProbs = LM_filelist('/Users/Nicola/Google Drive/code_data/
matlab/SimCode_Nic/20Bleaching_1/Bleaching20_12P_many/images');
filelist_IsingProbs = LM_filelist('/Users/Nicola/Google Drive/code_data
/matlab/SimCode_Nic/20Bleaching_1/Bleaching20_12P_many/isingProbs2');



All = length(filelist_GibbsProbs);

images_gibbs = NaN(100 ,100,All);
%images = images(:,:,1:All);

images_summed = NaN(50,1);
for i = 1:All
    temp = sum(sum(images(:,:,i)));
    images_summed(i,1) = temp;
end %for
figure('Name', 'Images Summed'), plot(images_summed);
%numb = 45;
%images_small = NaN(13,13,numb);

%% put all ising images in matrix

for i = 1:47
%for i = 1:numb
    filepath_ising=filelist_IsingProbs{i};
    filepath_GibbsProbs = filelist_GibbsProbs{i};
    Z_ising = double(importdata(char(filepath_ising)));
 %   Z_gibbs = double(importdata(char(filepath_GibbsProbs)));
    %Z_image = Z_image(34:66,34:66,:);
 %   Z_image = Z_image(44:56,44:56,1);

  %  images(:,:,i) = Z_image;
    images_ising(:,:,i) = Z_ising;
end %for
```

```
%keyboard

% for threshold = .7:.1:1
%     for i = 1: numb
%
%         threshold
%
%         mask = images(
%
%     end %for images
% end %threshold
%keyboard
%images_gibbs = images_gibbs/255;
%%  plot the sum of just the signal

mask_7 = images_gibbs > .7;
mask_8 = images_gibbs > .8;
mask_9 = images_gibbs > .9;

%keyboard

%image_thresh_7 = mask_7 .* images_small;
%image_thresh_8 = mask_8 .* images_small;
%image_thresh_9 = mask_9 .* images_small;
image_thresh_7 = mask_7 .* images;
image_thresh_8 = mask_8 .* images;
image_thresh_9 = mask_9 .* images;

figure('name', 'images summed just signal');
hold on
plot(squeeze(sum(sum(image_thresh_7))));
plot(squeeze(sum(sum(image_thresh_8))));
plot(squeeze(sum(sum(image_thresh_9))));
legend('below 0.7','below 0.8','below 0.9')

%% sum of just the signal for each image
%sum_image_7 = sum(image_thresh_7(:));
sum_image_7 = squeeze(sum(sum(image_thresh_7)));
%sum_image_8 = sum(image_thresh_8(:));
sum_image_8 = squeeze(sum(sum(image_thresh_8)));
%sum_image_9 = sum(image_thresh_9(:));
sum_image_9 = squeeze(sum(sum(image_thresh_9)));
```

```
%%
%sum_image_7t = [ sum_image_7 ; 0];
%sum_image_8t = [ sum_image_8 ; 0];
%sum_image_9t = [ sum_image_9 ; 0];


%sum_image_7t_p1 = [0; sum_image_7];
%sum_image_8t_p1 = [0; sum_image_8];
%sum_image_9t_p1 = [0; sum_image_9];

%difference
%E_t7 = sum_image_7t_p1 - sum_image_7t ;  %need to remove top
and bottom
%E_t8 = sum_image_7t_p1 - sum_image_8t ;
%E_t9 = sum_image_7t_p1 - sum_image_9t ;


%a = length(E_t7);
%E_t7 = E_t7(2:a-1);
%E_t8 = E_t8(2:a-1);
%E_t9 = E_t9(2:a-1);
%could have done
E_t7 = -diff(sum_image_7);
E_t8 = -diff(sum_image_8);
E_t9 = -diff(sum_image_9);


%GibbsData = GibbsData(1:All, :);
%%
Mus = NaN(size(GibbsData,1)-1, 1);
Prec = NaN(size(GibbsData,1)-1, 1);
%keyboard

for i = 1: size(GibbsData,1)-1
    Mus(i) = ( GibbsData(i,3) + GibbsData(i+1,3) )/2 ;
    Prec(i) = ( GibbsData(i,4) + GibbsData(i+1,4) )/2 ;
end %for

stDevs = (sqrt(Prec).^(-1));

%Mus = Mus(1:numb-1);
```

```
E_t7_mid = E_t7./Mus;
E_t8_mid = E_t8./Mus;
E_t9_mid = E_t9./Mus;

E_t7_low = E_t7./(Mus + stDevs) ;
E_t8_low = E_t8./(Mus + stDevs) ;
E_t9_low = E_t9./(Mus + stDevs) ;

E_t7_high = E_t7./(Mus - stDevs) ;
E_t8_high = E_t8./(Mus - stDevs) ;
E_t9_high = E_t9./(Mus - stDevs) ;



%figure; plot(E_t7)
%figure; plot(E_t8)
%figure; plot(E_t9)

%figure('Name','E_t7'); plot(E_t7)
%figure('Name','E_t8'); plot(E_t8)
figure('Name','E_t9'); plot(E_t9_mid)

number_on_3 = sum(M_binned>3);
number_on_5 = sum(M_binned>5);
number_on_7 = sum(M_binned>7);

number_on_3 = number_on_3(1:All);
number_on_7 = number_on_7(1:All);

%figure('Name', 'number_on' ); plot(number_on_3, 'r'); hold on;
...
%    plot(number_on_5, 'g'), plot(number_on_7, 'b');
%legend('on > 3', 'on >5', 'on >7')

%figure('Name', '-diff_num_on'); plot(-diff(number_on_3)); ...
%    hold on; plot(-diff(number_on_7));
%legend('difference in number on >3','difference in number on
>7');


figure('Name', 'both'); plot(-diff(number_on_3), '--');
```

```
hold on
plot(-diff(number_on_5),'--');
plot(-diff(number_on_7), '--');
%hold on;
plot(E_t9_mid,'r');
plot(E_t9_low,'g'); plot(E_t9_high, 'b')
legend('difference number on >3','difference number on >5', 'difference
number on >7', 'expected number on (mid)','expected number on
(low;)','expected number on (high)')


figure('Name', 'num_on and E_t9');
plot(E_t9_mid,'r'); hold on; plot(number_on_3)
legend('expected number using Gibbs','actual number on (>3)')

%% for updated mus threshold 7


Mus7 = NaN(size(GibbsData,1)-1, 1);
Prec7 = NaN(size(GibbsData,1)-1, 1);
%keyboard

for i = 1: size(GibbsData7,1)-1
    Mus7(i) = ( GibbsData7(i,3) + GibbsData7(i+1,3) )/2 ;
    Prec7(i) = ( GibbsData7(i,4) + GibbsData7(i+1,4) )/2 ;
end %for

stDevs7 = (sqrt(Prec7).^(-1));

%Mus = Mus(1:numb-1);

E7_t7_mid = E_t7./Mus7;
%E_t8_mid = E_t8./Mus;
%E_t9_mid = E_t9./Mus;

E7_t7_low = E_t7./(Mus7 + stDevs7) ;
%E_t8_low = E_t8./(Mus + stDevs) ;
%E_t9_low = E_t9./(Mus + stDevs) ;

E7_t7_high = E_t7./(Mus7 - stDevs7) ;
%E_t8_high = E_t8./(Mus - stDevs) ;
%E_t9_high = E_t9./(Mus - stDevs) ;
```

```
%figure; plot(E_t7)
%figure; plot(E_t8)
%figure; plot(E_t9)

%figure('Name','E_t7'); plot(E_t7)
%figure('Name','E_t8'); plot(E_t8)
figure('Name','E7_t7'); plot(E7_t7_mid)

number_on_3 = sum(M_binned>3);
number_on_5 = sum(M_binned>5);
number_on_7 = sum(M_binned>7);

%number_on_3 = number_on_3(1:All);
%number_on_7 = number_on_7(1:All);

figure('Name', 'number_on' ); plot(number_on_3, 'r'); hold on;
...
    plot(number_on_5, 'g'), plot(number_on_7, 'b');
legend('number on >3', 'number on >4', 'number on >5');

figure('Name', '-diff_num_on'); plot(-diff(number_on_3)); ...
    hold on; plot(-diff(number_on_7));
legend('difference in number on >3','difference in number on >7');

figure('Name', 'both'); plot(-diff(number_on_3), '--');
hold on
plot(-diff(number_on_7), '-');
%hold on;
plot(E7_t7_mid,'r');
plot(E7_t7_low,'g'); plot(E7_t7_high, 'b')
legend('difference number on >3', 'difference number on >7', 'expected
number on (mid)','expected number on (low;)','expected number
on (high)')

figure('Name', 'num_on and E7_t7');
plot(E7_t7_mid,'r'); hold on; plot(number_on_3)
legend('expected number using Gibbs','actual number on (>3)')


%% for updated mus threshold 8
```

```
%difference
%E_t7 = sum_image_7t_p1 - sum_image_7t ;  %need to remove top
and bottom
%E_t8 = sum_image_7t_p1 - sum_image_8t ;
%E_t9 = sum_image_7t_p1 - sum_image_9t ;


%a = length(E_t7);
%E_t7 = E_t7(2:a-1);
%E_t8 = E_t8(2:a-1);
%E_t9 = E_t9(2:a-1);
%could have done
%E_t7 = -diff(sum_image_7);
E_t8 = -diff(sum_image_8);
%E_t9 = -diff(sum_image_9);


%GibbsData = GibbsData(1:All, :);

Mus8 = NaN(size(GibbsData8,1)-1, 1);
Prec8 = NaN(size(GibbsData8,1)-1, 1);
%keyboard

for i = 1: size(GibbsData8,1)-1
    Mus8(i) = ( GibbsData8(i,3) + GibbsData8(i+1,3) )/2 ;
    Prec8(i) = ( GibbsData8(i,4) + GibbsData8(i+1,4) )/2 ;
end %for

stDevs8 = (sqrt(Prec8).^(-1));

%Mus = Mus(1:numb-1);

%E8_t7_mid = E_t7(1:46)./Mus8;
E8_t8_mid = E_t8(1:46)./Mus8;
%E8_t9_mid = E_t9(1:46)./Mus8;


%E8_t7_low = E_t7(1:46)./(Mus8 + stDevs8) ;
E8_t8_low = E_t8(1:46)./(Mus8 + stDevs8) ;
%E8_t9_low = E_t9(1:46)./(Mus8 + stDevs8) ;

%E8_t7_high = E_t7(1:46)./(Mus8 - stDevs8) ;
```

```
E8_t8_high = E_t8(1:46)./(Mus8 - stDevs8) ;
%E8_t9_high = E_t9(1:46)./(Mus8 - stDevs8) ;



%figure; plot(E_t7)
%figure; plot(E_t8)
%figure; plot(E_t9)

%figure('Name','E_t7'); plot(E_t7)
%figure('Name','E_t8'); plot(E_t8)
figure('Name','E8_t9'); plot(E8_t9_mid);
legend('expected dif thresh 9');
%number_on_3 = sum(M_binned>3);
%number_on_5 = sum(M_binned>5);
%number_on_7 = sum(M_binned>7);

%number_on_3 = number_on_3(1:All);
%number_on_7 = number_on_7(1:All);

figure('Name', 'number_on' ); plot(number_on_3, 'r'); hold on;
...
    plot(number_on_5, 'g'), plot(number_on_7, 'b');
legend('number on >3', 'number on >4', 'number on >5');

figure('Name', '-diff_num_on'); plot(-diff(number_on_3)); ...
    hold on; plot(-diff(number_on_7));
legend('difference in number on >3','difference in number on >7');

figure('Name', 'both'); plot(-diff(number_on_3), '--');
hold on
plot(-diff(number_on_7), '-');
%hold on;
plot(-diff(E8_t8_mid),'r');
plot(-diff(E8_t8_low),'g'); plot(-diff(E8_t8_high), 'b')
legend('difference number on >3', 'difference number on >7', 'expected
number on (mid_8)','expected number on (low_8)','expected number
on (high_8)')



figure('Name', 'num_on and E_t9');
plot(E8_t8_mid,'r'); hold on; plot(number_on_3)
```

```
legend('expected difference number using Gibbs','actual difference
number on (>3)')

%%
%% for updated mus threshold 9

%difference
%E_t7 = sum_image_7t_p1 - sum_image_7t ;  %need to remove top
and bottom
%E_t8 = sum_image_7t_p1 - sum_image_8t ;
%E_t9 = sum_image_7t_p1 - sum_image_9t ;


%a = length(E_t7);
%E_t7 = E_t7(2:a-1);
%E_t8 = E_t8(2:a-1);
%E_t9 = E_t9(2:a-1);
%could have done
%E_t9 = -diff(sum_image_7);
%E_t9 = -diff(sum_image_8);
E_t9 = -diff(sum_image_9);


%GibbsData = GibbsData(1:All, :);

Mus9 = NaN(size(GibbsData9,1)-1, 1);
Prec9 = NaN(size(GibbsData9,1)-1, 1);
%keyboard

for i = 1: size(GibbsData9,1)-1
    Mus9(i) = ( GibbsData9(i,3) + GibbsData9(i+1,3) )/2 ;
    Prec9(i) = ( GibbsData9(i,4) + GibbsData9(i+1,4) )/2 ;
end %for

stDevs9 = (sqrt(Prec9).^(-1));

%Mus = Mus(1:numb-1);

%E8_t7_mid = E_t7(1:46)./Mus8;
%E8_t8_mid = E_t8(1:46)./Mus8;
E9_t9_mid = E_t9(1:46)./Mus9;
```

```
%E8_t7_low = E_t7(1:46)./(Mus8 + stDevs8) ;
%E8_t8_low = E_t8(1:46)./(Mus8 + stDevs8) ;
E9_t9_low = E_t9(1:46)./(Mus9 + stDevs9) ;

%E8_t7_high = E_t7(1:46)./(Mus8 - stDevs8) ;
%E8_t8_high = E_t8(1:46)./(Mus8 - stDevs8) ;
E9_t9_high = E_t9(1:46)./(Mus9 - stDevs9) ;



%figure; plot(E_t7)
%figure; plot(E_t8)
%figure; plot(E_t9)

%figure('Name','E_t7'); plot(E_t7)
%figure('Name','E_t8'); plot(E_t8)
figure('Name','E_t9'); plot(E9_t9_mid)

%number_on_3 = sum(M_binned>3);
%number_on_5 = sum(M_binned>5);
%number_on_7 = sum(M_binned>7);

%number_on_3 = number_on_3(1:All);
%number_on_7 = number_on_7(1:All);

figure('Name', 'number_on' ); plot(number_on_3, 'r'); hold on;
...
    plot(number_on_5, 'g'), plot(number_on_7, 'b');
legend('number on >3', 'number on >4', 'number on >5');

figure('Name', '-diff_num_on'); plot(-diff(number_on_3)); ...
    hold on; plot(-diff(number_on_7));
legend('difference in number on >3','difference in number on >7');

figure('Name', 'both'); plot(-diff(number_on_3), '--');
hold on
plot(-diff(number_on_7), '-');
%hold on;
plot(-diff(E9_t9_mid),'r');
plot(-diff(E9_t9_low),'g'); plot(-diff(E9_t9_high), 'b')
```

```
legend('difference number on >3', 'difference number on >7', 'expected
number on (mid_8)','expected number on (low_8)','expected number
on (high_8)')


figure('Name', 'num_on and E_t9');
plot(E9_t9_mid,'r'); hold on; plot(diff(number_on_3))
legend('expected difference number using Gibbs','actual difference
number on (>3)')



%% for updated mus threshold 7 (second try)

%E_t7 = -diff(sum_image_7);
E_t7_2 = E_t7 / 7;

Mus7 = NaN(size(GibbsData,1)-1, 1);
Prec7 = NaN(size(GibbsData,1)-1, 1);
%keyboard

for i = 1: size(GibbsData7,1)-1
    Mus7(i) = ( GibbsData7(i,3) + GibbsData7(i+1,3) )/2 ;
    Prec7(i) = ( GibbsData7(i,4) + GibbsData7(i+1,4) )/2 ;
end %for

stDevs7 = (sqrt(Prec7).^(-1));

%Mus = Mus(1:numb-1);

E7_t7_mid_2 = E_t7_2./Mus7;
%E_t8_mid = E_t8./Mus;
%E_t9_mid = E_t9./Mus;

E7_t7_low_2 = E_t7_2./(Mus7 + stDevs7) ;
%E_t8_low = E_t8./(Mus + stDevs) ;
%E_t9_low = E_t9./(Mus + stDevs) ;

E7_t7_high_2 = E_t7_2./(Mus7 - stDevs7) ;
%E_t8_high = E_t8./(Mus - stDevs) ;
%E_t9_high = E_t9./(Mus - stDevs) ;
```

```
%figure; plot(E_t7)
%figure; plot(E_t8)
%figure; plot(E_t9)

%figure('Name','E_t7'); plot(E_t7)
%figure('Name','E_t8'); plot(E_t8)
figure('Name','E7_t7_2'); plot(E7_t7_mid_2)

number_on_3 = sum(M_binned>3);
number_on_5 = sum(M_binned>5);
number_on_7 = sum(M_binned>7);

%number_on_3 = number_on_3(1:All);
%number_on_7 = number_on_7(1:All);

figure('Name', 'number_on' ); plot(number_on_3, 'r'); hold on;
...
    plot(number_on_5, 'g'), plot(number_on_7, 'b');
legend('number on >3', 'number on >4', 'number on >5');

figure('Name', '-diff_num_on'); plot(-diff(number_on_3)); ...
    hold on; plot(-diff(number_on_7));
legend('difference in number on >3','difference in number on >7');

figure('Name', 'both'); plot(-diff(number_on_3), '--');
hold on
plot(-diff(number_on_7), '-');
%hold on;
plot(E7_t7_mid_2,'r');
plot(E7_t7_low_2,'g'); plot(E7_t7_high_2, 'b')
legend('difference number on >3', 'difference number on >7', 'expected
number on (mid)','expected number on (low;)','expected number
on (high)')

figure('Name', 'num_on_diff and E7_t7');
plot(E7_t7_mid_2,'r'); hold on; plot(-diff(number_on_3))
legend('expected number using Gibbs','actual diff number on (>3)')

E7_t7_mid_2 = reshape(E7_t7_mid_2, 1,[]);
figure('Name', 'diff between actual and expected')
plot(-diff(number_on_3)-E7_t7_mid_2)
hold on
```

```
plot(-diff(number_on_5)-E7_t7_mid_2);
plot(-diff(number_on_7)-E7_t7_mid_2)
legend('diff >3','diff >5','diff >7');

%%
Diff_act3_exp7 = -diff(number_on_3)-E7_t7_mid_2;
Diff_act5_exp7 = -diff(number_on_5)-E7_t7_mid_2;
Diff_act7_exp7 = -diff(number_on_7)-E7_t7_mid_2;
```

D.3 THE SUPPORT VECTOR MACHINES (SVM) CLASSIFICATION LEARNER

```
function [trainedClassifier, validationAccuracy] =
trainClassifierSVM( trainingData )
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% returns a trained classifier and its accuracy. This code recreates
the
% classification model trained in Classification Learner app. Use
the
% generated code to automate training the same model with new
data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: a matrix with the same number of columns
and data type
%       as imported into the app.
%
%  Output:
%      trainedClassifier: a struct containing the trained classifier.
The
%       struct contains various fields with information about
the trained
%       classifier.
%
%      trainedClassifier.predictFcn: a function to make predictions
on new
%       data.
%
```

```
%      validationAccuracy: a double containing the accuracy in
percent. In
%       the app, the History list displays this overall accuracy
score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your
original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original
data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on
new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a matrix containing only the predictor columns used
for
% training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 20-Aug-2019 16:05:48


% Extract predictors and response
% This code processes the data into the right shape for training
the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1',
'column_2', 'column_3', 'column_4'});

predictorNames = {'column_1', 'column_2', 'column_3'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_4;
isCategoricalPredictor = [false, false, false];
```

```
% Train a classifier
% This code specifies all the classifier options and trains the
classifier.
template = templateSVM(...
    'KernelFunction', 'gaussian', ...
    'PolynomialOrder', [], ...
    'KernelScale', 1.7, ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    predictors, ...
    response, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', [0; 1; 2; 3; 4]);

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
predictorNames);
svmPredictFcn = @(x) predict(classificationSVM, x);
trainedClassifier.predictFcn =
@(x) svmPredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.ClassificationSVM = classificationSVM;
trainedClassifier.About = 'This struct is a trained model exported
from Classification Learner R2019a.';
trainedClassifier.HowToPredict = sprintf('To make predictions
on a new predictor column matrix, X, use: \n  yfit = c.predictFcn(X)
\nreplacing ''c'' with the name of the variable that is this struct,
e.g. ''trainedModel''. \n \nX must contain exactly 3 columns because
this model was trained using 3 predictors. \nX must contain only
predictor columns in exactly the same order and format as your
training \ndata. Do not include the response column or any columns
you did not import into the app. \n \nFor more information, see
<a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
''appclassification_exportmodeltoworkspace'')">How to predict using
an exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training
the
```

```
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1',
'column_2', 'column_3', 'column_4'});

predictorNames = {'column_1', 'column_2', 'column_3'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_4;
isCategoricalPredictor = [false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationSVM,
'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] =
kfoldPredict(partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError');
```

D.4 THE K-NEAREST NEIGHBOURS (KNN) CLASSIFICATION LEARNER

```
function [trainedClassifier, validationAccuracy] =
trainClassifierKNN10(trainingData)
% [trainedClassifier, validationAccuracy] =
trainClassifier(trainingData)
% returns a trained classifier and its accuracy. This code recreates
the
% classification model trained in Classification Learner app. Use
the
% generated code to automate training the same model with new
data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: a matrix with the same number of columns
and data type
%       as imported into the app.
```

```
%
%  Output:
%      trainedClassifier: a struct containing the trained classifier.
The
%       struct contains various fields with information about
the trained
%       classifier.
%
%      trainedClassifier.predictFcn: a function to make predictions
on new
%       data.
%
%      validationAccuracy: a double containing the accuracy in
percent. In
%       the app, the History list displays this overall accuracy
score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your
original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original
data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on
new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a matrix containing only the predictor columns used
for
% training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 20-Aug-2019 16:04:41


% Extract predictors and response
```

```
% This code processes the data into the right shape for training
the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1',
'column_2', 'column_3', 'column_4'});

predictorNames = {'column_1', 'column_2', 'column_3'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_4;
isCategoricalPredictor = [false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the
classifier.
classificationKNN = fitcknn(...
    predictors, ...
    response, ...
    'Distance', 'Minkowski', ...
    'Exponent', 3, ...
    'NumNeighbors', 10, ...
    'DistanceWeight', 'Equal', ...
    'Standardize', true, ...
    'ClassNames', [0; 1; 2; 3; 4]);

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
predictorNames);
knnPredictFcn = @(x) predict(classificationKNN, x);
trainedClassifier.predictFcn = @(x) knnPredictFcn
(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.ClassificationKNN = classificationKNN;
trainedClassifier.About = 'This struct is a trained model exported
from Classification Learner R2019a.';
trainedClassifier.HowToPredict = sprintf('To make predictions
on a new predictor column matrix, X, use: \n  yfit = c.predictFcn(X)
\nreplacing ''c'' with the name of the variable that is this struct,
e.g. ''trainedModel''. \n \nX must contain exactly 3 columns because
this model was trained using 3 predictors. \nX must contain only
predictor columns in exactly the same order and format as your
```

```
training \ndata. Do not include the response column or any columns
you did not import into the app. \n \nFor more information, see
<a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
''appclassification_exportmodeltoworkspace'')">How to predict using
an exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training
the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1',
'column_2', 'column_3', 'column_4'});

predictorNames = {'column_1', 'column_2', 'column_3'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_4;
isCategoricalPredictor = [false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationKNN,
'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError');
```

D.5 THE NEURAL NETWORK (NN)

```
import torch
import matplotlib.pyplot as plt
import numpy as np
import torch.nn as nn
import pandas as pd
import torch.nn.functional as F
import torch.utils.data as data_utils
```

```
from torch.utils.data import DataLoader, TensorDataset, random_split
from torch.utils.tensorboard import SummaryWriter
import time
from PIL import Image
from collections import OrderedDict
from collections import namedtuple
from itertools import product

class mynetwork(nn.Module):
    def __init__(self, molecules, dropout):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=6,
kernel_size=5)
        self.conv2 = nn.Conv2d(in_channels=6, out_channels=12,
kernel_size=5)
        self.fc1 = nn.Linear(in_features=12 * 5 * 5, out_features=120)
        self.fc2 = nn.Linear(in_features=120, out_features=60)
        self.out = nn.Linear(in_features=60, out_features=molecules)
      #  self.ash = nn.Dropout(dropout)

    def forward(self, t):
        t = t

        # (2) hidden conv layer
        t = self.conv1(t)
        t = F.relu(t)
        t = F.max_pool2d(t, kernel_size=2, stride=2)

        # (3) hidden conv layer
        t = self.conv2(t)
        t = F.relu(t)
        t = F.max_pool2d(t, kernel_size=2, stride=2)

        # (4) hidden linear layer
        t = t.reshape(-1, 12 * 5 * 5)
        t = self.fc1(t)
        t = F.relu(t)

        # (5) hidden linear layer
        t = self.fc2(t)
        t = F.relu(t)
        t = self.ash(t)
```

```
        # (6) output layer
        t = self.out(t)
        # t = F.softmax(t, dim=1)

        return t

def data_get(file_name, csv_data, test_number):
    train_target = np.genfromtxt(csv_data, delimiter=',')
    image = Image.open(file_name)
    empty = []
    for I in range(image.n_frames):
        image.seek(I)
        empty.append(np.asarray(image))
    data = np.asarray(empty)
    data = data.astype(float)
    length_data = np.shape(data)[0]
    train_target = train_target[:, 0:length_data]
#    print(np.shape(train_target)[1])

    train = torch.tensor(data).type(torch.float)
    train = train.unsqueeze(dim=1)
    train_target = torch.tensor(np.sum((train_target > 5),
axis=0)).type(torch.long)
    test = train[-test_number:, :, :, :]
    train = train[:-test_number:, :, :, :]
    test_target = train_target[-test_number:]
    train_target = train_target[:-test_number]
    return (train, train_target, test, test_target)

def calculate_correct(out, labels):
    return torch.eq(out.argmax(dim=1), labels).sum().numpy()
    #return torch.eq(out.sum(dim=1), labels).sum().numpy()
    # Is position stupid

class RunBuilder():
    @staticmethod
    def get_runs(params):

        Run = namedtuple('Run', params.keys())

        runs = []
```

```
        for v in product(*params.values()):
            runs.append(Run(*v))
        return runs

class RunManager():
    def __init__(self):

        self.epoch_count = 0
        self.epoch_loss = 0
        self.epoch_num_correct = 0
        self.epoch_start_time = None

        self.run_params = None
        self.run_count = 0
        self.run_data = []
        self.run_start_time = None

        self.network = None
        self.loader = None
        self.tb = None

    def begin_run(self, run, network, loader):
        self.run_start_time = time.time()
        self.run_params = run
        self.run_count += 1
        self.network = network
        self.loader = loader
        self.tb = SummaryWriter(comment=f'-{run}')
        characteristics, labels = next(iter(self.loader))

    def end_run(self):
        self.tb.close()
        self.epoch_count = 0
        print(time.time() - self.run_start_time)
        final = network(test.cuda())
#        print(len(test_target))
        print(run.file_name)
        print(np.int( 100 * calculate_correct(final.cpu(), test_target)
/ len(test_target)))

    def begin_epoch(self):
        self.epoch_start_time = time.time()
```

```
        self.epoch_count += 1
        self.epoch_loss = 0
        self.epoch_num_correct = 0

    def end_epoch(self):
        epoch_duration = time.time() - self.epoch_start_time
        run_duration = time.time() - self.run_start_time

        loss = self.epoch_loss / len(self.loader.dataset)
        accuracy = self.epoch_num_correct / len(self.loader.dataset)

        self.tb.add_scalar('Loss', loss, self.epoch_count)
        self.tb.add_scalar('Accuracy', accuracy, self.epoch_count)

    def _get_num_correct(self, preds, labels):
        return preds.argmax(dim=1).eq(labels).sum().item()

    def track_loss(self, loss, batch):
        self.epoch_loss += loss.item() * batch[0].shape[0]

    def track_num_correct(self, preds, labels):
        self.epoch_num_correct += self._get_num_correct(preds,
labels)

    def inform(self, discrete_n):
        if self.epoch_count % discrete_n == 0:
            print(self.epoch_count, ' ', self.run_count)

params = OrderedDict(lr=[0.001], batch_size=[100], molecules=[6],
dropout=[0.2],file_name=['Images/SIM32_6.tif', 'Images/Gibbs.tif'])
m = RunManager()


for run in RunBuilder.get_runs(params):
    train_tensor, train_target, test, test_target = data_get(run.file_name,
'M_binned.csv', 60)
    train_data = data_utils.TensorDataset(train_tensor, train_target)
    network = mynetwork(run.molecules + 1, run.dropout)
    if (torch.cuda.device_count() == 1):
        print('CUDA')
        network = network.cuda()
```

```
    loader = DataLoader(train_data, batch_size=run.batch_size,
shuffle=True)
    optimizer = torch.optim.Adam(network.parameters(), lr=run.lr)
    m.begin_run(run, network, loader)

    for epoch in range(100):
        m.begin_epoch()
        for batch in loader:
            characteristics, labels = batch
            if (torch.cuda.device_count() == 1):
                characteristics = characteristics.cuda()
                labels = labels.cuda()
            preds = network(characteristics)  # Pass Batch
            loss = F.cross_entropy(preds, labels)  # Calculate
Loss
            optimizer.zero_grad()  # Zero Gradients
            loss.backward()  # Calculate Gradients
            optimizer.step()  # Update Weights
            m.track_loss(loss, batch)
            m.track_num_correct(preds, labels)
        m.inform(100)
        m.end_epoch()
    m.end_run()
```

D.6 THE 3D NEURAL NETWORK (NN)

D.6.1 *Main Code*

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "collapsed": true,
    "pycharm": {
     "name": "#%%\n"
```

```
   }
  },
  "outputs": [],
  "source": [
   "import numpy as np\n",
   "import torch\n",
   "import torch.nn as nn\n",
   "import torch.nn.functional as F\n",
   "import torch.optim as optim\n",
   "\n",
   "\n",
   "torch.set_grad_enabled(True)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 1,
  "outputs": [
   {
    "ename": "NameError",
    "evalue": "name 'nn' is not defined",
    "output_type": "error",
    "traceback": [
     "\u001B[0;31m-----------------------------------------
---------------------------------\u001B[0m",
     "\u001B[0;31mNameError\u001B[0m
       Traceback (most recent call last)",
     "\u001B[0;32m/var/folders/sm/33wx21p95j9479zfzh0bt80h0000gn
/T/ipykernel_72020/765017263.py\u001B[0m in \u001B[0;36m<module>
\u001B[0;34m\u001B[0m\n\u001B[0;32m----> 1\u001B[0;31m \u001B
[0;32mclass\u001B[0m \u001B[0mNetwork\u001B[0m\u001B[0;34m(\u001B
[0m\u001B[0mnn\u001B[0m\u001B[0;34m.\u001B[0m\u001B[0mModule\
u001B[0m\u001B[0;34m)\u001B[0m\u001B[0;34m:\u001B[0m\u001B
[0;34m\u001B[0m\u001B[0;34m\u001B[0m\u001B[0m\n\u001B[0m\u001B
[1;32m      2\u001B[0m     \u001B[0;32mdef\u001B[0m \u001B
[0m__init__\u001B[0m\u001B[0;34m(\u001B[0m\u001B[0mself
\u001B[0m\u001B[0;34m)\u001B[0m\u001B[0;34m:\u001B[0m\u001B
[0;34m\u001B[0m\u001B[0;34m\u001B[0m\u001B[0m\n\u001B[1;32m
3\u001B[0m         \u001B[0msuper\u001B[0m\u001B[0;34m
(\u001B[0m\u001B[0;34m)\u001B[0m\u001B[0;34m.\u001B
[0m\u001B[0m__init__\u001B[0m\u001B[0;34m(\u001B[0m
\u001B[0;34m)\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;34m\u001B
```

```
  [0m\u001B[0m\n\u001B[1;32m      4\u001B[0m         \u001B
  [0mself\u001B[0m\u001B[0;34m.\u001B[0m\u001B[0mconv1\u001B[0m
  \u001B[0;34m=\u001B[0m \u001B[0mnn\u001B[0m\u001B[0;34m.\u001B
  [0m\u001B[0mConv2d\u001B[0m\u001B[0;34m(\u001B[0m\u001B
  [0min_channels\u001B[0m\u001B[0;34m=\u001B[0m\u001B[0;36m1
  \u001B[0m\u001B[0;34m,\u001B[0m \u001B[0mout_channels\u001B
  [0m\u001B[0;34m=\u001B[0m\u001B[0;36m6\u001B[0m\u001B[0;34m,
  \u001B[0m \u001B[0mkernel_size\u001B[0m\u001B[0;34m=\u001B
  [0m\u001B[0;36m5\u001B[0m\u001B[0;34m)\u001B[0m\u001B[0;34m
  \u001B[0m\u001B[0;34m\u001B[0m\u001B[0m\n\u001B[1;32m
  5\u001B[0m         \u001B[0mself\u001B[0m\u001B[0;34m.\u001B
  [0m\u001B[0mconv2\u001B[0m \u001B[0;34m=\u001B[0m \u001B[0mnn
  \u001B[0m\u001B[0;34m.\u001B[0m\u001B[0mConv2d\u001B[0m
  \u001B[0;34m(\u001B[0m\u001B[0min_channels\u001B[0m\u001B
  [0;34m=\u001B[0m\u001B[0;36m6\u001B[0m\u001B[0;34m,\u001B[0m
  \u001B[0mout_channels\u001B[0m\u001B[0;34m=\u001B[0m\u001B
  [0;36m12\u001B[0m\u001B[0;34m,\u001B[0m \u001B[0mkernel_size
  \u001B[0m\u001B[0;34m=\u001B[0m\u001B[0;36m5\u001B[0m
  \u001B[0;34m)\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;34m\u001B
  [0m\u001B[0m\n",
      "\u001B[0;31mNameError\u001B[0m: name 'nn' is not defined"
     ]
    }
   ],
   "source": [
    "class Network(nn.Module):\n",
    "    def __init__(self):\n",
    "        super().__init__()\n",
    "        self.conv1 = nn.Conv2d(in_channels=1, out_channels=6,
kernel_size=5)\n",
    "        self.conv2 = nn.Conv2d(in_channels=6, out_channels=12,
kernel_size=5)\n",
    "\n",
    "        self.fc1 = nn.Linear(in_features=12 * 4 * 4,
out_features=120)\n",
    "        self.fc2 = nn.Linear(in_features=120, out_features=60)\n",
    "        self.out = nn.Linear(in_features=60, out_features=2)\n",
    "\n",
    "    def forward(self, t):\n",
    "        t = t\n",
    "\n",
    "        # (2) hidden conv layer\n",
```

```
    "        t = self.conv1(t)\n",
    "        t = F.relu(t)\n",
    "        t = F.max_pool2d(t, kernel_size=2, stride=2)\n",
    "        #    print(t.size())\n",
    "\n",
    "        # (3) hidden conv layer\n",
    "        t = self.conv2(t)\n",
    "        t = F.relu(t)\n",
    "        t = F.max_pool2d(t, kernel_size=2, stride=2)\n",
    "        #   print(t.size())\n",
    "\n",
    "        # (4) hidden linear layer\n",
    "        t = t.reshape(-1, 12 * 4 * 4)\n",
    "        t = self.fc1(t)\n",
    "        t = F.relu(t)\n",
    "        #  print(t.size())\n",
    "\n",
    "        # (5) hidden linear layer\n",
    "        t = self.fc2(t)\n",
    "        t = F.relu(t)\n",
    "        # print(t.size())\n",
    "\n",
    "        # (6) output layer\n",
    "        t = self.out(t)\n",
    "        t = F.softmax(t, dim=1)\n",
    "        #print(t.size())\n",
    "\n",
    "        return t\n",
    "\n",
    "    def __repr__(self):\n",
    "        return \"Bunny Kitten\"\n",
    "\n"
   ],
   "metadata": {
    "collapsed": false,
    "pycharm": {
     "name": "#%%\n"
    }
   }
  }
 ],
 "metadata": {
```

```
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 2
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython2",
   "version": "2.7.6"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 0
}
```


D.6.2 *3D neural network (NN) - the main code*


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [
    {
     "ename": "ModuleNotFoundError",
     "evalue": "No module named 'torch'",
     "output_type": "error",
```

```
     "traceback": [
      "\u001b[0;31m---------------------------------------------
  --------------------------\u001b[0m",
      "\u001b[0;31mModuleNotFoundError\u001b[0m
        Traceback (most recent call last)",
      "\u001b[0;32m<ipython-input-1-5d2c842df08f>\u001b[0m in
\u001b
  [0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[0;32m----> 1\u001b
  [0;31m \u001b[0;32mimport\u001b[0m \u001b[0mtorch\u001b[0m\u001b
  [0;34m   \u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b
  [0m\u001b[1;32m      2\u001b[0m \u001b[0;32mimport\u001b[0m
\u001b
  [0mnumpy\u001b[0m \u001b[0;32mas\u001b[0m \u001b[0mnp\u001b[0m
  \u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
  3\u001b[0m \u001b[0;32mfrom\u001b[0m \u001b[0mtorch\u001b[0m
\u001b
  [0;32mimport\u001b[0m \u001b[0moptim\u001b[0m\u001b[0;34m\u001b[0m
  \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m      4\u001b[0m
\u001b
  [0;32mimport\u001b[0m \u001b[0mtorch\u001b[0m\u001b[0;34m.\u001b[0m
  \u001b[0mnn\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mfunctional\u
  001b[0m \u001b[0;32mas\u001b[0m \u001b[0mF\u001b[0m\u001b
  [0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
  5\u001b[0m \u001b[0;32mimport\u001b[0m \u001b[0mtorch\u001b
  [0m\u001b[0;34m.\u001b[0m\u001b[0mnn\u001b[0m
  \u001b[0;32mas\u001b[0m \u001b[0mnn
  \u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b
  [0m\u001b[0m\n",
      "\u001b[0;31mModuleNotFoundError\u001b[0m: No module named
’torch’"
     ]
    }
   ],
   "source": [
    "import torch\n",
    "import numpy as np\n",
    "from torch import optim\n",
    "import torch.nn.functional as F\n",
    "import torch.nn as nn\n",
    "from PIL import Image\n",
    "import os\n",
    "torch.set_grad_enabled(True)\n",
```

```
    "from torch.utils.tensorboard import SummaryWriter"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 56,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "def loadtiffs(file_name):\n",
    "    img = Image.open(file_name)\n",
    "    #print('The Image is', img.size, 'Pixels.')\n",
    "    #print('With', img.n_frames, 'frames.')\n",
    "\n",
    "    imgArray = np.zeros((img.size[1], img.size[0], img.n_frames),
np.uint16)\n",
    "    for I in range(img.n_frames):\n",
    "        img.seek(I)\n",
    "        imgArray[:, :, I] = np.asarray(img)\n",
    "    img.close()\n",
    "    return(imgArray)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 57,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "def get_file_list(dir):\n",
    "    file_list = []\n",
    "    for file in os.listdir(dir):\n",
    "        if file.endswith(\".tif\"):\n",
    "            file_name = dir + '/' +file\n",
```

```
    "            file_list.append(file_name)\n",
    "    return file_list"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 58,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "fileListName = \"images\"\n",
    "fileList = get_file_list(fileListName)\n",
    "fileList = sorted(fileList)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 61,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "data = loadtiffs(fileList[0])\n",
    "data = np.append(data, loadtiffs(fileList[1]), axis=2)\n",
    "data = np.append(data, loadtiffs(fileList[2]), axis=2)\n",
    "#data = np.append(data, loadtiffs(fileList[3]), axis=2)\n",
    "#data = np.append(data, loadtiffs(fileList[4]), axis=2)\n",
    "data = data.astype(float)\n",
    "data = np.swapaxes(data,0,2)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 62,
   "metadata": {
```

```
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "x_data = torch.tensor(data)\n",
    "x_data = torch.unsqueeze(x_data, 0)\n",
    "x_data = torch.unsqueeze(x_data, 0)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 63,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "#N   For mini batch (or how many sequences do we want to
feed at one go)\n",
    "# Cin For the number of channels in our input (if our image
is rgb, this is 3)\n",
    "#D   For depth or in other words the number of images/frames
in one input sequence (if we are dealing videos, this is the number
of frames)\n",
    "#H   For the height of the image/frame\n",
    "#W   For the width of the image/frame\n",
    "\n",
    "input = torch.randn(20, 3, 10, 50, 100)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 81,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
```

```
   "outputs": [],
   "source": [
    "class Network(nn.Module):\n",
    "    def __init__(self):\n",
    "        super().__init__()\n",
    "        #self.conv1 = nn.Conv2d(in_channels=3, out_channels=12,
kernel_size=5)\n",
    "        self.conv1 = nn.Conv3d(in_channels = 1, out_channels
= 8, kernel_size=(3,3,3))\n",
    "        #    nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5)\n",
    "\n",
    "        self.fc1 = nn.Linear(in_features=8 * 62 * 62,
out_features=120)\n",
    "        self.fc2 = nn.Linear(in_features=120, out_features=60)\n",
    "        self.out = nn.Linear(in_features=60, out_features=10)\n",
    "\n",
    "    def forward(self, t):\n",
    "        t = t\n",
    "\n",
    "        # (2) hidden conv layer\n",
    "        #print(t.size())\n",
    "        t = self.conv1(t)\n",
    "        t = F.relu(t)\n",
    "        #t = F.max_pool2d(t, kernel_size=2, stride=2)\n",
    "#        print(t.size())\n",
    "\n",
    "        # (3) hidden conv layer\n",
    "#        print(t.size())\n",
    "#        t = self.conv2(t)\n",
    "#        t = F.relu(t)\n",
    "#        t = F.max_pool2d(t, kernel_size=2, stride=2)\n",
    "#        print(t.size())\n",
    "\n",
    "        # (4) hidden linear layer\n",
    "        t = t.reshape(-1, 30752)#61504)\n",
    "        t = self.fc1(t)\n",
    "        t = F.relu(t)\n",
    "        #  print(t.size())\n",
    "\n",
    "        # (5) hidden linear layer\n",
    "        t = self.fc2(t)\n",
```

```
  "        t = F.relu(t)\n",
  "        # print(t.size())\n",
  "\n",
  "        # (6) output layer\n",
  "        t = self.out(t)\n",
  "        #t = F.softmax(t, dim=1)\n",
  "        #print(t.size())\n",
  "\n",
  "        return t\n",
  "\n",
  "    def __repr__(self):\n",
  "        return \"Bunny Kitten\"\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": 82,
 "metadata": {
  "pycharm": {
   "name": "#%%\n"
  }
 },
 "outputs": [],
 "source": [
  "m = Network().double()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 83,
 "metadata": {
  "pycharm": {
   "name": "#%%\n"
  }
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "torch.Size([1, 1, 3, 64, 64])\n",
    "torch.Size([1, 8, 1, 62, 62])\n"
```

```
     ]
    }
   ],
   "source": [
    "out = m(x_data)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 84,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "tensor([[-5.5895, -3.4079,  1.1544, -1.2945,  3.3398, -3.4924,
 3.1246,  2.5768,\n",
      "          0.6130,  6.9213]], dtype=torch.float64,
 grad_fn=<AddmmBackward0>)\n"
     ]
    }
   ],
   "source": [
    "print(out)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 85,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "values = np.genfromtxt('M_on3_num.csv',delimiter=',')\n",
```

```
  "labels = torch.tensor(values)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 86,
 "metadata": {
  "pycharm": {
   "name": "#%%\n"
  }
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "torch.Size([1, 1, 3, 64, 64])\n",
    "torch.Size([1, 8, 1, 62, 62])\n"
   ]
  }
 ],
 "source": [
  "#test\n",
  "current = 1\n",
  "data = loadtiffs(fileList[current-1])\n",
  "data = np.append(data, loadtiffs(fileList[current]), axis=2)\n",
  "data = np.append(data, loadtiffs(fileList[current+1]), axis=2)\n",
  "data = data.astype(float)\n",
  "data = np.swapaxes(data,0,2)\n",
  "x_data = torch.tensor(data)\n",
  "x_data = torch.unsqueeze(x_data, 0)\n",
  "x_data = torch.unsqueeze(x_data, 0)\n",
  "preds = m(x_data)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 87,
 "metadata": {
  "pycharm": {
   "name": "#%%\n"
  }
```

```
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "False\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
      "torch.Size([1, 8, 1, 62, 62])\n",
      "torch.Size([1, 1, 3, 64, 64])\n",
```

```
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
"torch.Size([1, 8, 1, 62, 62])\n",
"torch.Size([1, 1, 3, 64, 64])\n",
```

```
     "torch.Size([1, 8, 1, 62, 62])\n",
     "torch.Size([1, 1, 3, 64, 64])\n",
     "torch.Size([1, 8, 1, 62, 62])\n",
     "torch.Size([1, 1, 3, 64, 64])\n",
     "torch.Size([1, 8, 1, 62, 62])\n",
     "torch.Size([1, 1, 3, 64, 64])\n",
     "torch.Size([1, 8, 1, 62, 62])\n"
    ]
   },
   {
    "ename": "KeyboardInterrupt",
    "evalue": "",
    "output_type": "error",
    "traceback": [
     "\u001b[0;31m-----------------------------------------
--------------------------------\u001b[0m",
     "\u001b[0;31mKeyboardInterrupt\u001b[0m
Traceback (most recent call last)",
     "\u001b[0;32m/var/folders/sm/33wx21p95j9479zfzh0bt80h0000gn
/T/ipykernel_88707/4219333407.py\u001b[0m in \u001b[0;36m
<module>\u001b[0;34m\u001b[0m\n\u001b[1;32m     23\u001b[0m
          \u001b[0mx_data\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mtorch
 \u001b[0m\u001b[0;34m.\u001b[0m\u001b
 [0munsqueeze\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mx_data
 \u001b[0m\u001b[0;34m,\u001b[0m \u001b[0;36m0\u001b
 [0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b
 [0;34m\u001b[0m\u001b[0m\n\u001b[1;32m     24\u001b[0m
 \u001b[0mx_data\u001b[0m \u001b[0;34m=\u001b[0m \u001b
 [0mtorch\u001b[0m\u001b[0;34m.\u001b[0m\u001b
 [0munsqueeze\u001b[0m\u001b[0;34m(\u001b[0m\u001b
 [0mx_data\u001b[0m\u001b[0;34m,\u001b[0m \u001b[0;36m0
 \u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m
 \u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m--->
 25\u001b[0;31m         \u001b[0mpreds\u001b[0m
 \u001b[0;34m=\u001b[0m \u001b[0mnetwork\u001b[0m\u001b[0;34m
 (\u001b[0m\u001b[0mx_data\u001b[0m\u001b[0;34m)\u001b
 [0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m
 \n\u001b[0m\u001b[1;32m     26\u001b[0m         \u001b
 [0mlabel\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0;34m
 (\u001b[0m\u001b[0mlabels\u001b[0m\u001b[0;34m[\u001b
 [0m\u001b[0mcurrent\u001b[0m\u001b[0;34m]\u001b
 [0m\u001b[0;34m.\u001b[0m\u001b[0munsqueeze\u001b[0m\u001b
```

```
   [0;34m(\u001b[0m\u001b[0;36m0\u001b[0m\u001b[0;34m)
   \u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m.\u001b[0m
   \u001b[0mtype\u001b[0m\u001b[0;34m(\u001b[0m\u001b
   [0mtorch\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mLongTensor
   \u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m
   \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m     27\u001b[0m
        \u001b
   [0mloss\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mF\u001b
   [0m\u001b[0;34m.
   \u001b[0m\u001b[0mcross_entropy\u001b[0m\u001b[0;34m
   (\u001b[0m\u001b[0mpreds\u001b[0m\u001b[0;34m,\u001b[0m
    \u001b[0mlabel\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m
   \u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m~/miniforge3/lib/python3.9/site-packages
  /torch/nn/modules/module.py\u001b[0m in \u001b[0;36m_call_impl
  \u001b[0;34m(self, *input, **kwargs)\u001b[0m\n\u001b
  [1;32m   1100\u001b[0m         if not (self._backward_hooks
  or self._forward_hooks or self._forward_pre_hooks or
  _global_backward_hooks\n\u001b[1;32m   1101\u001b[0m
  or _global_forward_hooks or _global_forward_pre_hooks):
  \n\u001b[0;32m-> 1102\u001b[0;31m
  \u001b[0;32mreturn\u001b[0m \u001b[0mforward_call
  \u001b[0m\u001b[0;34m(\u001b[0m\u001b
  [0;34m*\u001b[0m\u001b[0minput\u001b[0m\u001b[0;34m,\u001b[0m
\u001b[0;34m**\u001b[0m\u001b[0mkwargs\u001b[0m\u001b[0;34m)\u001b[
  0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[
  0m\u001b[1;32m   1103\u001b[0m         \u001b[0;31m# Do not
call
  functions when jit is used\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m
  \u001b[0m   \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
  1104\u001b[0m         \u001b[0mfull_backward_hooks\u001b[0m
  \u001b[0;34m,\u001b[0m \u001b[0mnon_full_backward_hooks\u001b[0m
  \u001b[0;34m=\u001b[0m \u001b[0;34m[\u001b[0m\u001b[0;34m]\u001b
  [0m\u001b[0;34m,\u001b[0m \u001b[0;34m[\u001b[0m\u001b[0;34m]
  \u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m/var/folders/sm/33wx21p95j9479zfzh0bt80h0000gn/
  T/ipykernel_88707/1846076806.py\u001b[0m in \u001b[0;36mforward
  \u001b[0;34m(self, t)\u001b[0m\n\u001b[1;32m     30\u001b[0m
  \u001b[0;31m# (4) hidden linear layer\u001b[0m\u001b[0;34m
  \u001b[0m\u001b
  [0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
  31\u001b[0m         \u001b[0mt\u001b[0m \u001b[0;34m=\u001b[0m
```

```
  \u001b[0mt\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mreshape
  \u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m-\u001b[0m\u001b
  [0;36m1\u001b[0m\u001b[0;34m,\u001b[0m \u001b[0;36m30752
  \u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;31m#61504)
  \u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b
  [0m\n\u001b[0;32m---> 32\u001b[0;31m         \u001b[0mt\
  u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mself\u001b
  [0m\u001b[0;34m.\u001b[0m\u001b[0mfc1\u001b[0m\u001b[0;34m
  (\u001b[0m\u001b[0mt\u001b[0m\u001b[0;34m)\u001b[0m\u001b
  [0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b
  [0m\u001b[1;32m     33\u001b[0m         \u001b[0mt\u001b
  [0m \u001b[0;34m=\u001b[0m \u001b[0mF\u001b[0m\u001b
  [0;34m.\u001b[0m\u001b[0mrelu\u001b[0m\u001b[0;34m
  (\u001b[0m\u001b[0mt\u001b[0m\u001b[0;34m)\u001b[0m\u001b
  [0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
       34\u001b[0m         \u001b[0;31m#  print(t.size())\u001b
   [0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b
[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m~/miniforge3/lib/python3.9/
  site-packages/torch/nn/modules/module.py\u001b[0m in
  \u001b[0;36m_call_impl\u001b[0;34m(self, *input, **kwargs)
  \u001b[0m\n\u001b[1;32m   1100\u001b[0m
  if not (self._backward_hooks or self._forward_hooks or
  self._forward_pre_hooks or _global_backward_hooks\n\u001b[1;32m
  1101\u001b[0m                 or _global_forward_hooks
  or _global_forward_pre_hooks):\n\u001b[0;32m-> 1102\u001b[
  0;31m             \u001b[0;32mreturn\u001b[0m
  \u001b[0mforward_call\u001b[0m\u001b[0;34m(\u001b[0m\u001b
  [0;34m*\u001b[0m\u001b[0minput\u001b[0m\u001b[0;34m,\u001b[0m
   \u001b[0;34m**\u001b[0m\u001b[0mkwargs\u001b[0m\u001b[0;34m)
   \u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b
   [0m\n\u001b[0m\u001b[1;32m   1103\u001b[0m
   \u001b[0;31m# Do not call functions when jit is
    used\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[
0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
1104\u001b[0m
\u001b[0mfull_backward_hooks\u001b[0m\u001b[0;34m,\u001b
[0m \u001b[0mnon_full_backward_hooks\u001b[0m \u001b[0;34m=
\u001b[0m \u001b[0;34m[\u001b[0m\u001b[0;34m]\u001b
[0m\u001b[0;34m,\u001b[0m \u001b[0;34m[\u001b[0m\u001b[0;34m]
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m~/miniforge3/lib/python3.9/site-packages
```

```
  /torch/nn/modules/linear.py\u001b[0m in \u001b
  [0;36mforward\u001b[0;34m(self, input)\u001b[0m\n\u001b
  [1;32m    101\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n
  \u001b[1;32m    102\u001b[0m     \u001b[0;32mdef\u001b
  [0m \u001b[0mforward\u001b[0m\u001b[0;34m(\u001b[
  0m\u001b[0mself\u001b[0m\u001b[0;34m,\u001b[0m \u001b
  [0minput\u001b[0m\u001b[0;34m:\u001b[0m \u001b[0mTensor
  \u001b[0m\u001b[0;34m)\u001b[0m \u001b[0;34m->\u001b[0m
  \u001b[0mTensor\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b
  [0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m--> 103\u001b[0;31m
       \u001b[0;32mreturn\u001b[0m \u001b[0mF\u001b
   [0m\u001b[0;34m.\u001b[0m\u001b[0mlinear\u001b
   [0m\u001b[0;34m(\u001b[0m\u001b[0minput\u001b[0m\u001b[0;34m,
   \u001b[0m \u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b

   [0mweight\u001b[0m\u001b[0;34m,\u001b[0m
\u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mbias
\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b
[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b
[1;32m    104\u001b[0m \u001b[0;34m\u001b[0m\u001b
[0m\n\u001b[1;32m    105\u001b[0m
\u001b[0;32mdef\u001b[0m \u001b[0mextra_repr\u001b[0m
\u001b[0;34m(\u001b[0m\u001b[0mself\u001b[0m\u001b
[0;34m)\u001b[0m \u001b[0;34m->\u001b[0m \u001b
[0mstr\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m~/miniforge3/lib/python3.9/site-packages
  /torch/nn/functional.py\u001b[0m in \u001b[0;36mlinear
  \u001b[0;34m(input, weight, bias)\u001b[0m\n\u001b[1;32m
  1846\u001b[0m     \u001b[0;32mif\u001b[0m
  \u001b[0mhas_torch_function_variadic\u001b[0m\u001b[0;34m
  (\u001b[0m\u001b[0minput\u001b[0m\u001b[0;34m,\u001b[0m
  \u001b[0mweight\u001b[0m\u001b[0;34m,\u001b[0m
  \u001b[0mbias\u001b[0m\u001b[0;34m)\u001b[0m\u001b
  [0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m
  \u001b[0m\u001b[0m\n\u001b[1;32m   1847\u001b[0m
           \u001b[0;32mreturn\u001b[0m \u001b[0mhandle_torch_function
    \u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mlinear
    \u001b[0m\u001b[0;34m,\u001b[0m \u001b[0;34m(\u001b[0m\
    u001b[0minput\u001b[0m\u001b[0;34m,\u001b[0m
    \u001b[0mweight\u001b[0m\u001b[0;34m,\u001b[0m
    \u001b[0mbias\u001b[0m\u001b[0;34m)\u001b[0m
```

```
   \u001b[0;34m,\u001b  [0m \u001b[0minput\u001b
   [0m\u001b[0;34m,\u001b[0m \u001b[0mweight\u001b
   [0m\u001b[0;34m,\u001b[0m \u001b[0mbias
   \u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mbias
   \u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\
   u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b
   [0;32m-> 1848\u001b[0;31m     \u001b[0;32mreturn\u001b[0m
    \u001b[0mtorch\u001b[0m\u001b[0;34m.\u001b[0m\
u001b[0m_C\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m_nn
\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mlinear\u001b
[0m\u001b[0;34m(\u001b[0m\u001b[0minput\u001b[0m
\u001b[0;34m,\u001b[0m \u001b[0mweight\u001b[0m
\u001b[0;34m,\u001b[0m \u001b[0mbias\u001b[0m\u001b[0;34m)
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m
\u001b[0m\n\u001b[0m\u001b[1;32m
 1849\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
 1850\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;31mKeyboardInterrupt\u001b[0m: "
     ]
    }
   ],
   "source": [
    "print(torch.cuda.is_available())\n",
    "network = Network().double()\n",
    "tb = SummaryWriter()\n",
    "\n",
    "number_run = 200\n",
    "numberImages = 99\n",
    "holdall = np.zeros(number_run)\n",
    "closs = np.zeros(number_run)\n",
    "#optimizer = optim.Adam(network.parameters(), lr=0.01)\n",
    "optimizer = torch.optim.SGD(network.parameters(), lr=0.1,
momentum=0.9)\n",
    "for I in range(number_run):\n",
    "    total_correct = 0\n",
    "    optimizer.zero_grad()\n",
    "    for current in range(1,numberImages):\n",
    "\n",
    "\n",
    "        data = loadtiffs(fileList[current-1])\n",
    "        data = np.append(data, loadtiffs(fileList[current]),
axis=2)\n",
```

```
    "        data = np.append(data, loadtiffs(fileList[current+1]),
axis=2)\n",
    "        data = data.astype(float)\n",
    "        data = np.swapaxes(data,0,2)\n",
    "        x_data = torch.tensor(data)\n",
    "        x_data = torch.unsqueeze(x_data, 0)\n",
    "        x_data = torch.unsqueeze(x_data, 0)\n",
    "        preds = network(x_data)\n",
    "        label = (labels[current].unsqueeze(0)).
type(torch.LongTensor)\n",
    "        loss = F.cross_entropy(preds, label)\n",
    "        #print(loss)\n",
    "        #optimizer.zero_grad()\n",
    "        loss.backward() # Calculating the gradients\n",
    "        optimizer.step()\n",
    "        if (torch.argmax(preds)) == label:\n",
    "            total_correct = total_correct + 1\n",
    "    tb.add_scalar('Loss', loss, I)\n",
    "    tb.add_scalar('Number Correct', total_correct, I)\n",
    "    print(total_correct)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
```

```
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.8.5"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 1
}
```

D.6.3 *3D neural network (NN) - the NN*

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "collapsed": true,
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import torch\n",
    "import torch.nn as nn\n",
    "import torch.nn.functional as F\n",
    "import torch.optim as optim\n",
    "\n",
    "\n",
    "torch.set_grad_enabled(True)"
   ]
```

```
 },
 {
  "cell_type": "code",
  "execution_count": 1,
  "outputs": [
   {
    "ename": "NameError",
    "evalue": "name 'nn' is not defined",
    "output_type": "error",
    "traceback": [
     "\u001B[0;31m-----------------------------------------
---------------------------------\u001B[0m",
     "\u001B[0;31mNameError\u001B[0m
 Traceback (most recent call last)",
     "\u001B[0;32m/var/folders/sm/33wx21p95j9479zfzh0bt80h0000gn
 /T/ipykernel_72020/765017263.py\u001B[0m in \u001B
 [0;36m<module>\u001B[0;34m\u001B[0m\n\u001B
 [0;32m----> 1\u001B[0;31m \u001B[0;32mclass\u001B[0m
  \u001B[0mNetwork\u001B[0m\u001B[0;34m(\u001B[0m\
  u001B[0mnn\u001B[0m\u001B[0;34m.\u001B[0m\u001B
  [0mModule\u001B[0m\u001B[0;34m)\u001B[0m\u001B
  [0;34m:\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;34m
  \u001B[0m\u001B[0m\n\u001B[0m\u001B[1;32m
  2\u001B[0m     \u001B[0;32mdef\u001B[0m \u001B[0m__init__
  \u001B[0m\u001B[0;34m(\u001B[0m\u001B[0mself\u001B
  [0m\u001B[0;34m)\u001B[0m\u001B[0;34m:\u001B[0m\u001B
  [0;34m\u001B[0m\u001B[0;34m\u001B[0m\u001B[0m\n\u001B
  [1;32m      3\u001B[0m         \u001B[0msuper\u001B[0m
  \u001B[0;34m(\u001B
  [0m\u001B[0;34m)\u001B [0m\u001B[0;34m.\u001B[0m\u001B
  [0m__init__\u001B[0m\u001B[0;34m(\u001B[0m\u001B
  [0;34m)\u001B[0m\u001B[0;34m\u001B[0m\u001B[0;34m
  \u001B[0m\u001B[0m\n\u001B[1;32m      4\u001B[0m
          \u001B[0mself\u001B[0m\u001B[0;34m.\u001B[0m
  \u001B[0mconv1\u001B[0m \u001B[0;34m=\u001B[0m
   \u001B[0mnn\u001B[0m\u001B[0;34m.\u001B[0m\u001B
[0mConv2d\u001B[0m\u001B[0;34m(\u001B[0m\u001B
[0min_channels\u001B[0m\u001B[0;34m=\u001B[0m\u001B
[0;36m1\u001B[0m\u001B[0;34m,\u001B[0m \u001B
[0mout_channels\u001B[0m\u001B[0;34m=\u001B
[0m\u001B[0;36m6\u001B[0m\u001B[0;34m,\u001B[0m
 \u001B[0mkernel_size\u001B[0m\u001B[0;34m=\u001B
```

```
 [0m\u001B[0;36m5\u001B[0m\u001B[0;34m)\u001B[0m
 \u001B[0;34m\u001B[0m\u001B[0;34m\u001B[0m\u001B
 [0m\n\u001B[1;32m      5\u001B[0m         \u001B
 [0mself\u001B[0m\u001B[0;34m.\u001B[0m\u001B
 [0mconv2\u001B[0m \u001B[0;34m=\u001B[0m
  \u001B[0mnn\u001B[0m\u001B[0;34m.\u001B[0m\u001B
  [0mConv2d\u001B[0m\u001B[0;34m(\u001B[0m\u001B
  [0min_channels\u001B[0m\u001B[0;34m=\u001B[0m
  \u001B[0;36m6\u001B[0m\u001B[0;34m,\u001B[0m
  \u001B[0mout_channels\u001B[0m\u001B[0;34m=\u001B
  [0m\u001B[0;36m12\u001B[0m\u001B[0;34m,\u001B[0m
  \u001B[0mkernel_size\u001B[0m\u001B[0;34m=\u001B
  [0m\u001B[0;36m5\u001B[0m\u001B[0;34m)\u001B[0m\
  u001B[0;34m\u001B[0m\u001B[0;34m\u001B[0m\u001B[0m\n",
      "\u001B[0;31mNameError\u001B[0m: name 'nn' is not defined"
     ]
    }
   ],
   "source": [
    "class Network(nn.Module):\n",
    "    def __init__(self):\n",
    "        super().__init__()\n",
    "        self.conv1 = nn.Conv2d(in_channels=1, out_channels=6,
kernel_size=5)\n",
    "        self.conv2 = nn.Conv2d(in_channels=6, out_channels=12,
kernel_size=5)\n",
    "\n",
    "        self.fc1 = nn.Linear(in_features=12 * 4 * 4,
out_features=120)\n",
    "        self.fc2 = nn.Linear(in_features=120, out_features=60)\n",
    "        self.out = nn.Linear(in_features=60, out_features=2)\n",
    "\n",
    "    def forward(self, t):\n",
    "        t = t\n",
    "\n",
    "        # (2) hidden conv layer\n",
    "        t = self.conv1(t)\n",
    "        t = F.relu(t)\n",
    "        t = F.max_pool2d(t, kernel_size=2, stride=2)\n",
    "        #    print(t.size())\n",
    "\n",
    "        # (3) hidden conv layer\n",
```

```
   "        t = self.conv2(t)\n",
   "        t = F.relu(t)\n",
   "        t = F.max_pool2d(t, kernel_size=2, stride=2)\n",
   "        #   print(t.size())\n",
   "\n",
   "        # (4) hidden linear layer\n",
   "        t = t.reshape(-1, 12 * 4 * 4)\n",
   "        t = self.fc1(t)\n",
   "        t = F.relu(t)\n",
   "        #  print(t.size())\n",
   "\n",
   "        # (5) hidden linear layer\n",
   "        t = self.fc2(t)\n",
   "        t = F.relu(t)\n",
   "        # print(t.size())\n",
   "\n",
   "        # (6) output layer\n",
   "        t = self.out(t)\n",
   "        t = F.softmax(t, dim=1)\n",
   "        #print(t.size())\n",
   "\n",
   "        return t\n",
   "\n",
   "    def __repr__(self):\n",
   "        return \"Bunny Kitten\"\n",
   "\n"
  ],
  "metadata": {
   "collapsed": false,
   "pycharm": {
    "name": "#%%\n"
   }
  }
 }
],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
```

```
   "codemirror_mode": {
    "name": "ipython",
    "version": 2
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython2",
   "version": "2.7.6"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 0
}
```