Models of Visual Attention in Deep Residual CNNs

Chaitanya Kaul

Doctor of Philosophy

University of York

Computer Science

November 2019

Dedication

To my parents..

Abstract

Feature reuse from earlier layers in neural network hierarchies has been shown to improve the quality of features at a later stage - a concept known as residual learning. In this thesis, we learn effective residual learning methodologies infused with attention mechanisms to observe their effect on different tasks. To this end, we propose 3 architectures across medical image segmentation and 3D point cloud analysis. In FocusNet, we propose an attention based dual branch encoder decoder structure that learns an extremely efficient attention mechanism which achieves state of the art results on the ISIC 2017 skin cancer segmentation dataset. We propose a novel loss enhancement that improves the convergence of FocusNet, performing better than state-of-the-art loss functions such as tversky and focal loss. Evaluations of the architecture proposes two drawbacks which we fix in FocusNetAlpha. Our novel residual group attention block based network forms the backbone of this architecture, learning distinct features with sparse correlations, which is the key reason for its effectiveness. At the time of writing this thesis, FocusNetAlpha outperforms all state-of-the-art convolutional autoencoders with the least parameters and FLOPs compared to them, based on our experiments on the ISIC 2018, DRIVE retinal vessel segmentation and the cell nuclei segmentation dataset. We then shift our attention to 3D point cloud processing where we propose SAWNet, which combines global and local point embeddings infused with attention, to create a spatially aware embedding that outperforms both. We propose a novel method to learn a global feature aggregation for point clouds via a fully differential block that does not need a lot of trainable parameters and gives obvious performance boosts. SAWNet beats state-of-the-art results on ModelNet40 and ShapeNet part segmentation datasets.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank for parents for their constant support throughout my education. They always pushed me to be a better version of myself and always put my needs before their own. Thank you for being my support system. Thank you for everything.

I would like to express my sincere thanks to both my supervisors - Dr Nick Pears, and Dr Suresh Manandhar for their patience and encouragement throughout the three years of my thesis. A good supervisor can make or break a PhD and I was extremely lucky to have two absolutely wonderful people to help me through my PhD journey. I cannot thank Nick enough for always making time for my questions on a daily basis and always explaining the answer to every question I had in great detail. My conversations with Suresh over the three years have really helped push me go that extra mile, which I believe has made a huge difference in my work ethic. Nick and Suresh together have shaped me into the kind of researcher I wanted to be when I started my PhD. I came to York to work under two fantastic mentors, who I now see as great friends. They made my PhD a really enjoyable experience. I would like to thank Dr Will Smith for his patience with me and for always offering a different perspective on my research. I feel my conversations with Will during our thesis advisory panel meetings pointed out aspects and directions of my research that I couldn't initially think about which pushed me to be on my toes and be better prepared with knowledge of my field. I would also like to thank Dr Bernhard Kainz for agreeing to be my external examiner.

I am also grateful to have met some absolutely wonderful people throughout the course of my PhD who have made my stay at York a real delight. I'd like to

thank Timothy Atkinson, Jason Pereira, Athena Karsa, Nils Moenning, Di Wang, Taghreed Alqaisi, Dr Savan Vacchani and Dr Hang Dai for their technical expertise, friendship, stimulating discussing on random research, and just keeping me sane over the last three years. Thanks a lot guys.

Being at this university has been a great experience for me. I feel extremely lucky to have been able to land a job during the last 3 months of my PhD and I can't wait to start the next step of my journey at the University of Glasgow. Being able to get permission to make the shift from York to Glasgow in terms of a work permit was extremely difficult for me, but thanks to Claire Fox, I was able to get through in the end. She handled things extremely promptly and professionally and is the reason I can start work immediately after my PhD, without a gap, as I always intended to. Thank you Claire for helping me out with all my admin issues and for some really fun conversations over the last two years. The research experience and exposure I have gained here has prepared me well for my job as a research associate in medical image analysis at Glasgow. I hope to extend the skills I've learnt here and further my research in medical imaging for the greater good.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Some parts of this thesis have been published in conference proceedings and journals; where items were published jointly with collaborators, the author of this thesis is responsible for the material presented here. For each published item the primary author is the first listed author.

Chapter 3 contains FocusNet and Adaptive Logarithmic Loss,

- Chaitanya Kaul, Suresh Manandhar, Nick Pears. Focusnet: An Attention-Based Fully Convolutional Network For Medical Image Segmentation. In *Proceedings of the 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 455-458, April 2019. [45]

- Chaitanya Kaul, Nick Pears, Suresh Manandhar. Penalizing small errors using an Adaptive Logarithmic Loss. [47]

Chapter 4 contains FocusNetAlpha,

- Chaitanya Kaul, Nick Pears, Suresh Manandhar. Divided We Stand: A novel Residual Group Attention Mechanism for Medical Image Segmentation. [46]

Chapter 5 contains SAWNet,

- Chaitanya Kaul, Nick Pears, Suresh Manandhar. SAWNet: A Spatially Aware Attention-Based Deep Neural Network for 3D Point Cloud Processing. [48]

Abbreviations

VOC	Visual Object Classes
mAP	mean Average Precision
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
AI	Artificial Intelligence
ISIC	International Skin Imaging Collaboration
JI	Jaccard Index
DI	Dice Index
TL	Tversky Loss
SGD	Stochastic Gradient Descent
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
ILSVRC	Imagenet Large Scale Visual Recognition Challenge

SVM Support Vector Machine

DVAN Diversified Visual Attention Mechanism

FCN Fully Convolutional Network

MLP Multi-Layer Perceptrons

RGB-D Red Green Blue - Depth

SOM Self Organizing Map

BN Batch Normalization

SE Squeeze and Excitation

ROC Receiver Operator Characteristics

CE Cross Entropy

ALL Adaptive Logarithmic Loss

HL Hybrid Loss

FLOPs Floating Point Operations per Second

AUC Area under the Curve

SAW Spatially Aware

CUDA Compute Unified Device Architecture

CUDnn CUDA Deep Neural Network Library

CAD Computer Aided Design

DGCNN Dynamic Graph Convolutional Neural Network

PCNN Point Convolutional Neural Network

mIOU mean Intersection over Union

Chapter 1

Introduction

1.1 Machine learning in computer vision

The task of image acquisition, processing and inference comes naturally to humans. In the field of computer vision, we try and replicate the human brain's ability to sense and perceive the environment around it, into machines. Over the past years, the number of sensors that capture both 2D and 3D modalities of images has increased drastically. Due to their increased production, it is now cheaper than ever to obtain such devices. This has led to an increase in the amount of available data for vision applications. But there are inherent problems with visual data, mainly the fundamental fact that it is difficult to analyse it. In-fact, the amount of visual data on the internet is increasing at such a rate that it is not possible for humans to efficiently process it. To put things into perspective, about 300 hours of video content is uploaded on YouTube every minute, making it impossible for humans to sit and tag each upload to provide better recommendation to the users. The need for understanding visual data at a rate faster than humans, calls for machine learning algorithms that can learn from some supervised or semi supervised training methodologies to solve the task faster than humans. But historically, these tasks have not been simple for machines either. For instance, finding general objects in generic snapshots of the 3D world on datasets such as the PASCAL VOC [27], had a detection score of 33.6% mAP [71]. This was an approach using deformable parts models and was generally slow. Even simple tasks such as image classification

were considered to be challenging for machine learning algorithms at one time. One of the reasons for this was the fact that the algorithms could only be as good as the features that they learnt from. As classical machine learning approaches took hand crafted features as their input, they were highly reliant on specialists with the relevant domain knowledge to compute the best possible features from the image data. As classical machine learning approaches take hand crafted features as input, researchers needed to be careful about constructing these features to be invariant to general problems such as occlusions, transformations, deformations, changes in illumination to name a few. This made the process of hand crafting features for such tasks tedious and one that required a lot of care.

1.2 Motivation

Efficient feature extraction is hard, mainly due to the expert domain knowledge needed to know the right features that should be extracted for a task. Learning the best features for a particular task, simultaneously along with the task itself eliminates this need for expert knowledge to an extent. This is the premise of deep learning. Deep learning is a subset of machine learning that performs feature extraction implicitly given a task. The emergence of deep learning has led to a shift towards computational models that can learn to optimize parameters over features and tasks without any intermediate human intervention. The effectiveness of deep learning can be associated with the hierarchical manner the features are extracted for their applications, which leads to better feature representations for the given task at hand. An artificial neural network learns the complex data representations via backpropagation, where the trainable parameters of the neural network are iterative refined based on the errors the predictions generate. The representations deep learning creates are so powerful that object detection on the PASCAL VOC datset is now considered a relatively easy task, with deep approaches giving scores of almost 90.0% mAP [27]. This is mainly possible due to the end-to-end training paradigm that deep learning employs, where the feature extraction, and the detection task, are both solved together by the same network, and the performance of

every component in the network effects every other component.

Different types of networks cater to different machine learning tasks. RNNs and their variants (LSTMs, GRUs) are extensively used in tasks involving natural language [10], such as machine translation [59], dialog systems [9] and language modelling [42]. They're also used for analysis of financial markets [76], and other sequence modelling tasks. The networks that this thesis deals with are convolutional neural networks and their variants. Convolutional neural networks extract features from image data using convolutions and solve the task at hand with the extracted features. The feature extraction and solving the task is generally done in an end to end fashion. This means that the weights of the feature extractor and the task together form one non convex optimization problem, and the computed error between the output of the network and the ground truth label, backpropagates to the feature extractor, which updates the weights of the convolution kernels and the task.

The success of deep learning is not entirely based on its power to learn better representations for the data. Advancements in technology such as the availability of better computational devices (GPUs, TPUs) as well as the availability of larger datasets, and smart data augmentation play a major role in the success of deep networks. Many open source libraries such as Tensorflow [1], Theano [85], Keras [11], PyTorch [65] etc have the capabilities of performing automatic differentiation, which allows for faster experimentation.

1.3 Context of the thesis

This thesis broadly deals with creating efficient feature representations for given data using convolutional neural networks. The applications of the methods developed in this thesis concern efficient and accurate analysis of medical images, particularly, the segmentation of regions of interest from a particular image and processing 3D point cloud data. We concentrate on creating deep learning architectures that provide a higher accuracy than the currently existing techniques.

The acceptance of deep learning by the medical community has been a relatively slow process. Even though the computer vision community has accepted deep learning and shifted towards end-to-end training of models, the healthcare sector still completely has not. A large reason for this is the lack of availability of large datasets, which is a major factor that deep learning thrives on. Medical imaging datasets usually contain just a few hundred images making deep learning a lot harder for these tasks. Another reason is the lack of resources and manpower to annotate such large datasets for medical applications, as they require specific expertise and are generally cross checked by more than one medical practitioner. Deep learning on medical images is harder, mainly because of the type of data that the task proposes. Regions of interest in such data are usually very small and fine grained feature processing is crucial to get any meaningful results. Due to this reason, most existing deep learning approaches specially pertaining to medical image segmentation have large pipelines of pre and post processing along with the neural network models as their backbones. There is hence a very important need for better algorithms that cater to the fine grained nature of the lesions present in medical images and is something that this thesis looks into in great detail.

Neural networks that process raw 3D point cloud data did not exist at the start of this PhD. In the last 3 years, numerous architectures have been proposed that are better than classical machine learning techniques at handling the challenges that 3D point clouds present. Point cloud processing finds applications in a lot of domains, the most famous being incorporation into autonomous vehicles. In order to achieve such a feat, processing data in its raw form (such as point clouds) is highly beneficial as it provides additional geometric information about the world that 2D image cannot grasp, and it also eliminates extra computation required to convert the data into a form that existing deep learning architectures can consume.

Visual attention is another topic that this thesis looks into. Human brains subconsciously throw away any excess visual stimuli and automatically concentrate

on the object that they are looking at. This task is not so effortless for machines as it requires extra computation, as well as mechanisms for their correct incorporation into computational models. This thesis enhances convolutional neural networks using attention mechanisms for 2D image data as well as 3D point cloud data. Of particular interest are attention mechanisms that learn to focus inside feature volumes, and combining them with global attention mechanisms to enhance the accuracy of deep models.

1.4 Research aims of this thesis

In this thesis, we try to encapsulate a number of contributions that aid in better feature extraction using convolutional neural networks, enabling these networks to be successfully be applied to biomedical imaging data, as well as raw 3D sensor data. The main aims of this thesis are summarized below,

- To enable a deeper understanding of residual learning and attention mechanisms in convolutional neural networks with applications to different modalities of visual data.
- To create general convolutional neural network architectures that can find applications in domains of AI beyond computer vision.
- To better analyse convolutional neural networks through a series of ablation studies to aid better understanding of these models, which are otherwise considered as black boxes.
- To create models that can provide a reasonable trade-off between model complexity and performance.

1.5 Thesis Contributions

We present three novel deep learning architectures, and one novel enhancement to loss functions for medical image segmentation, in this thesis. Two features unify these architectures, namely, residual learning and attention mechanisms. Our thesis

shows how residual learning can be combined with smart attention mechanisms to learn better feature representations of data, leading to better performance on the benchmark tasks. The contributions of this thesis are as follows:

- We present a novel and extremely efficient attention scheme, combined with residual connections to segment medical images.
- We present FocusNet, that uses this attention scheme to achieve state of the art results on the ISIC 2017 skin cancer segmentation dataset.
- We propose a novel group attention mechanism that uses residual learning. This mechanism forms the building block of FocusNetAlpha, which is our extremely efficient and accurate segmentation architecture that achieves state of the art results on 3 benchmark medical imaging datasets with the least number of parameters compared to the state of the art models. Our proposed attention block also achieves superior results compared to resnet and all it's variants.
- We introduce a novel loss enhancement to medical image segmentation tasks that speeds up convergences and increases accuracy.
- We finally present SAWNet, a novel attention based methodology to combine global and local information inside 3D point clouds, to aid the learning of a more robust per point embedding via backpropagation. SAWNet achieves state of the art results on benchmark point cloud classification and segmentation tasks.
- We are the first to propose a novel attention scheme for aggregating feature information from point cloud embeddings into a global aggregated vector.

1.6 Chapter Overview

The rest of the thesis is organized as follows:

- **Chapter 2** provides the background literature that sets the theme of this thesis. Deep learning in the context of 2D and 3D data is discussed along

with loss functions and optimization algorithms that have been used. We also provide a detailed analysis of convolutional neural networks in context to medical image segmentations.

- **Chapter 3** presents Focusnet, our attention based fully convolutional network, with skip connections, for medical image segmentation. We provide an extensive evaluation of our network via a series of ablation tests. We also compare the architecture with other state of the art architectures in the field of medical image segmentation. To improve network performance, we present the adaptive logarithmic loss which further improves performance of the network.

- **Chapter 4** presents FocusnetAlpha, our architecture that beats the state of the art results on benchmark medical image segmentation datasets with lesser parameters than them. We incorporate a hybrid loss function inside our adaptive loss strategy to train a network to give a well rounded performance across all evaluation metrics.

- **Chapter 5** presents SAWNet, our deep residual attention based convolutional neural network architecture to process 3D point clouds. Our architecture is an incremental improvement to locality based deep architectures that learn point representations based on the neighbourhood of the points. We create an attention based spatially aware architecture to solve various 3D point cloud processing tasks and compare our architecture with the various state of the art.

- **Chapter 6** gives a brief summary, the general conclusions for the thesis and future directions this research can take.

Chapter 2

Literature survey

2.1 Neural Networks and Deep Learning

Artificial neural networks are essentially function approximators, that calculate the conditional probability of an outcome, given some training data. What makes them better than generic linear classifiers, is the fact that they can approximate a more complex family of functions due to the added non linearity they induce to the tasks at hand. The simplest form of this family of functions is the perceptron. A perceptron predicts a linear function, $y = Wx + b$, for some given data x, that is then passed through some non linear activation, generally a sigmoid function, $\sigma = \frac{1}{1+e^{-x}}$. Perceptrons find a linear decision boundary between different classes of data, hence learning somewhat linear relationships. This makes them almost useless in real world scenarios, where the data exists in multiple dimensions and is generally linearly inseparable.

Stacking multiple perceptron units together in a fully connected manner results in a multi layer perceptron (MLP). The architecture of a MLP is shown in Figure 2.1. This type of network is also known as a feedforward neural network as the nodes in a particular layer connect to all nodes in the layer after it, with no connection going backwards. Each layer in a MLP contains a set of nodes that learn a set of weights that approximate linear functions. Every node also contains a non linear activation function that helps learn more complex functions, making it a lot more useful for real world applications. Stacking more hidden layers in this setting facilitates the

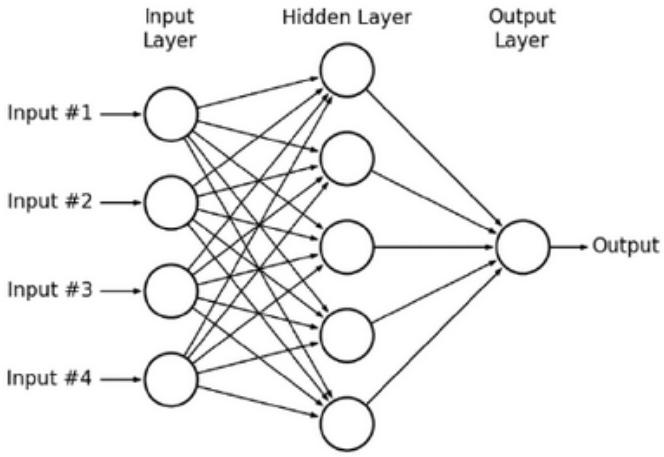


Figure 2.1: A multilayer perceptron with one hidden layer [33].

learning of hierarchical features that may be present in the data. A forward pass of input data through such a network computes the output given the current state of the trainable weights, while a backward pass adjusts the weights to account for any error in the computed output. Neural networks are generally trained in a supervised fashion, i.e., they require a corresponding ground truth given a training input. The output of the network is compared with its corresponding ground truth. MLPs are universal approximators, which means, that given an input, and atleast one hidden layer, they can approximate any continuous function arbitrarily well. Their main drawback is that they only take vectorized inputs, not taking into account spatial relations between the input.

2.1.1 Backpropagation

Neural Networks learn to adjust their weights via the backpropagation of their error. Backpropagation is a method of computing gradients using the chain rule. The computation is divided into a forward pass and a backward pass. The data is input into the network in the forward pass and flows to the output layer. An output is calculated based on the values of the weights and it is then compared to the target value for the input in the training set. The error is calculated and backpropagated through the network to adjust the weights helping the neural networks learn a representation for the data. A loss (also known as error or cost) function is used

to compute the empirical difference between the output computed by the network, and the actual output. The network's attempt to minimize this loss value is known as 'learning'.

2.1.2 Loss functions

Based on their parameters, neural networks develop complex, non linear relationships between the data and ground truth. These complex relations result in a multi dimensional landscape with dimensions equal to the number of parameters of the neural network. Loss functions compare the output of the neural networks with the ground truth images and quantify the difference between them, encapsulating them into a numerical value called the loss (or error or cost). This loss value is by how much the predicted output differs from the ground truth corresponding to the input. For instance, in terms of image segmentation, the prediction and the ground truth are both binary images with a particular height and width, that are first flattened into a 1D vector to compute the loss. A common loss function for regression tasks is the mean squared error loss. It is given by the formula,

$$L(n) = \frac{1}{N}\sum_{i=1}^{N}(p_i(n) - \hat{p_i}(n))^2$$

Where each individual p_i and $\hat{p_i}$ are the ground truth and predicted values respectively.

The most basic loss function used for classification tasks is the binary cross entropy function. In this case, the output can only take two possible values - 0 or 1. The output of a neural network is passed through a sigmoid function to predict each pixel as a 0 or a 1. The comparison with the ground truth is done as follows,

$$L(n) = -[p(n)\ \log(\hat{p}(n)) + (1 - p(n))\ \log(1 - \hat{p}(n)))]$$

Where, $p(n)$ is the probability of the ground truth label being 0 and $\hat{p}(n)$ is the probabilistic output from a logistic sigmoid function predicted as 0. Similarly, $(1 - p(n))$ denotes the ground truth label of 1 and $(1 - \hat{p}(n))$ denotes the predicted label

as 1 from the sigmoid output. When the classification task involves more than two category labels, this loss can be generalized to

$$L(n) = -\sum_{c=1}^{C} p(n)_c \log(\hat{p}(n)_c)$$

where C is the number of categories. These entropy based losses generally works well for classification and segmentation as long as the labels for all classes are balanced. If one class dominates over the other, the imbalance results in the network predicting all outputs to be the dominant class due to convergence to a non optimal local minimum. Some recently proposed loss functions such as the dice loss and the focal loss [24] tackle this problem by weighting some outputs more than others.

General evaluations of these losses is done by calculating the overall overlap between the ground truth and the prediction. These values are computed over the soft segmentation outputs (probabilities). This intersection over union metric (Jaccard Index) is given by,

$$JI = \frac{|G \cap P|}{|G| + |P| - |G \cap P|}$$

where G is the ground truth mask and P is the predicted mask. In contrast, the Dice Index assigns a higher weight to the true positives, and is given by the formula:

$$DI = \frac{2|G \cap P|}{|G| + |P|}.$$

Due to its high weight on the true positives, DI is also widely used as a loss function. The Tversky Index [75] is another function proposed that adds further weight to the false positives and false negatives to get better predictions.

$$TL = \frac{|G \cap P|}{|G \cap P| + \alpha|P \backslash G| + \beta|G \backslash P|}$$

Generally, $\alpha = 0.3,\ \beta = 0.7$ works as the most optimal setting. These similarity metrics are generally converted to loss functions by optimizing over a sum of their class-wise difference from the optimal value. Their general form is $L = \sum_c (1 - M)$ where the metric, M, can be Jaccard, Dice or Tversky Index. The subscript indicates that the summation is over the number of classes, c. Many loss functions have also

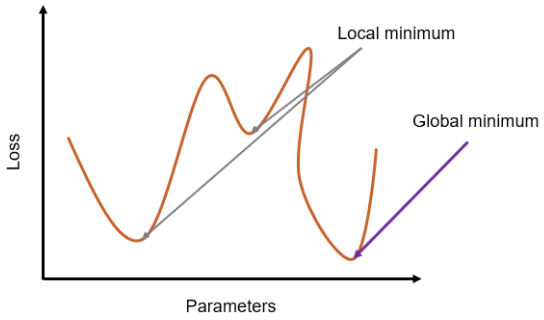


Figure 2.2: Optimization problem in neural networks.

been proposed [20] [25] [23] as weighted combinations of these losses. Some other evaluation metrics widely used are sensitivity (recall) which is the true positive rate, specificity which is the true negative rate, precision which is the positive predicted value, and accuracy which is the rate of labels (across all classes) classified correctly.

2.1.3 Optimization

How a neural network traverses a loss landscape is based on some defined optimizer. Figure 2.2 visualises this landscape. The optimization probelm is generally a non convex one with multiple sub optimal local minimum solutions. The optimizers are some variants of Stochastic Gradient Descent (SGD). SGD computes the gradient of the network parameters with respect to the loss function and then takes a step in the direction of steepest descent on the loss function, given the current location (gradient). Instead of looking at all the data at once, the algorithm looks at a smaller sample from the data instead - a mini batch. The weights of a network are updated via the following rule using SGD,

$$\theta_{i+1} = \theta_i - \beta \; L(\mathbf{X}, \mathbf{y})$$

where θ_i is the learning rate at the particular iteration, β is the learning rate that scales the loss.

2.2 Convolutional Neural Networks

Pre 2012, in order to process image data, features were hand crafted from the input and fed into machine learning systems. Since the emergence of AlexNet [53], the era of deep learning took the computer vision community by storm as end-to-end learning of features has proven to be better than hand crafting features across all domains of computer vision. Deep learning - specifically convolutional neural networks (CNNs), have revolutionized computer vision.

As impactful as CNNs have been to this field, their building blocks are quite simple. CNNs combine feature extractions via convolutions with a machine learning task, learning to do both together, via backpropagation. The main components of CNNs are -

- **Convolutions** - The convolution operation in CNNs is basically a correlation between the input image and a randomly initialized convolutional filter. For an input $\mathbf{I}$, filter $\mathbf{F}$, and output $\mathbf{C}$, the convolution can be described as, $\mathbf{F} * \mathbf{I} = \mathbf{C}$ for $\mathbf{F} \in R^{W \times H}, \mathbf{I} \in R^{k \times k}, \mathbf{C} \in R^{W' \times H'}$. $k \times k$ represents the kernel (or receptive field) of the convolutional filter.

- **Pooling** - Images are downsampled to a lower scale to be further processed by some convolution operations via some form of pooling operation. Most commonly used poolings are max pooling, where the maximum value in a kernel is used to represent the kernel, and average pooling, where the average value of the kernel represents the kernel.

- **Activations** - Convolutions are generally followed by non linearities, also known as activation functions. The most commonly used activation is the Rectified Linear Unit (ReLU), given by $ReLU(x) = \max(0, x)$, where the response of a network is killed for negative values of the features learnt.

2.2.1 Notable advances in Convolutional Neural Networks

Convolutional Neural Networks date back to before the inception of backpropagation [29]. Though, due to lack of sufficiently large datasets and compute power, they weren't considered a feasible tool in the machine learning arsenal. The pioneering piece of research that started all research in this field is the AlexNet architecture that won the ILSVRC 2012, initiating the era of deep learning in computer vision. Alexnet, inspired by the works of Fukushima [29], and LeCunn [54], showed how end to end training on GPUs can beat the performance of general then state-of-the-art machine learning systems such as SVMs that worked with hand crafted features. AlexNet treated the task of feature extraction via convolutions, and classification across the 1000 imagenet classes, as a single task, learning specialized features for this task together with learning to do the task. They introduced the ReLU non linearity that helped alleviate the vanishing gradient problem caused by the sigmoid funtion. The next notable work that pushed forward research in this field was the paper that proposed the VGGNet [81] architecture. VGGNet was an engineering masterpiece, which showed how the use of smaller kernels learnt better features and used fewer parameters in the process. They also showed that training deeper networks leads to better performance which they demonstrated with their impressive performance on the ILSVRC 2014. The following year saw some of the first work in dividing convolutions into smaller groups and concatenating the results of these groups inside a single layer. This concept was introduced in GoogLeNet [83].

2.2.2 Group convolutions

Grouped convolutions have been around since the inception of ILSVRC 2012 where the Alexnet architecture divided their computations across two GPUs and noticed that the same convolution divided into two different groups, learns two distinct representations of the data in the same layer. This effectively leads to a network learning a more robust feature representation that has a block diagonal sparsity between the different groups of the learnt filters in the same layer. Figure 2.3 shows the different filters learnt in the first convolution layer of the AlexNet architecture.

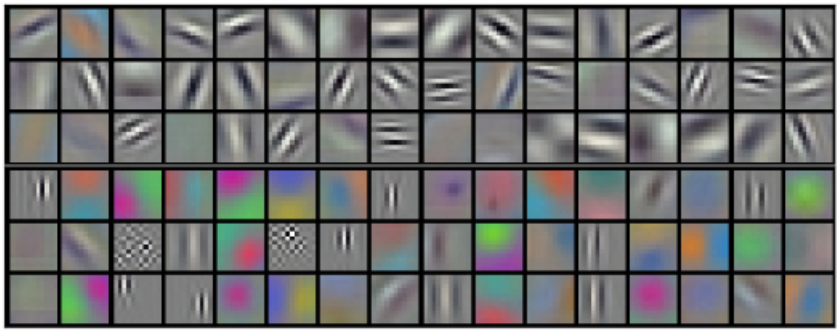


Figure 2.3: Visualization of filters learnt in the first layer of AlexNet [53].

The top 48 filters are learnt on one GPU, and the bottom 48 on another. It can be seen that grouping filters in these two groups learns distinct representations of the data.

To the best of our knowledge, one of the first works to explicitly show that grouping filter groups leads to learning better representations is deep roots [39]. They use a sparse connecting structure that resembles a tree root to reduce the number of parameters without any significant affect on the network accuracy. The impact of group convolutions was made apparent with resneXt [92] employing it at the heart of its methodology and doing impressively at the ILSVRC 2016 tasks. Depthwise separable convolutions [12] are another form of grouped convolutions that process channels in a feature volume individually before combining it together via a 1D convolution.

Due to the nature of grouped convolutions, the filter groups only look at a particular part of the input at a time. Seeing this as an apparent drawback, techniques such as ShuffleNet [98], FLGC [86] and IGCNets [97] have been proposed to aid interaction between filter groups and different parts of the feature inputs. To the best of our knowledge, all (except one) existing techniques learn fixed permutations of the features to aid interaction with filter groups. ShuffleNet looks at only one permutation of the features for interaction with the filters, IGCNets fix a permutation matrix to perform permutations following their single primary and secondary

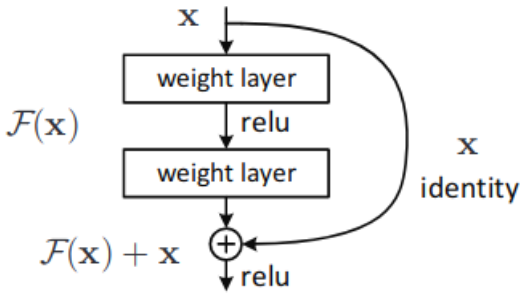


Figure 2.4: The basic residual learning building block proposed in [34].

group convolution blocks.

2.2.3 Residual Learning

ResNets introduced the concept of feature reuse in later convolution layers. The initial residual architecture [34] won the ILSVRC 2015 classification task and was also the first network to beat human level performance on it. Since their inception, resnets have been constantly optimized and refined to produce better results. The most basic form of the residual block can be visualised in Figure 2.4. This block can be formally defined as

$$y = F\{x, \mathbf{W_i}\} + x$$

where $F\{x, \mathbf{W_i}\}$ is the residual mapping that is learnt by the network and x are the features that are propagated forward by the skip connection. Residual learning allows feature reuse from previous layers that helps fight overfitting in deep networks. This technique results in the ability to train extremely deep neural networks with over 1000 layers. The residual block was optimized to produce better results in [35]. They showed how using a bn-relu-conv style residual block performed much better than the conv-bn-relu block via a series of ablation studies.

One of the first works combining group convolutions with residual learning was the resneXt [92] architecture The authors defined a new form of aggregated residual

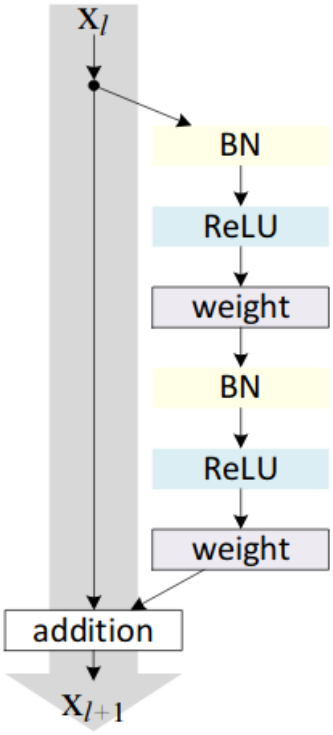


Figure 2.5: The optimized identity mapping block proposed in [35]

transformation,

$$F(x) = \sum_{i=1}^{C} T_i\{x\}$$

where $T_i\{x\}$ can be any function approximation, and is generally a set of neurons. T_i transforms the input x by projecting it into a low dimensional embedding. C is the cardinality, which is the total number of transformations that the network learns for a network layer. This aggregated transformation serves as the residual connection learnt in this architecture. The output y in this case is given by,

$$y = x + \sum_{i=1}^{C} T_i\{x\}$$

where x is the the value that the aggregated transformation works on. Another spin on the residual block shows that stacking deeper networks with more layers is equivalent to combining multiple convolution operations inside the same layer. As the convolutions in each layers have larger filter sizes, these networks are called wide residual networks [95]. Wide Resnet50, which is a 50 layer resnet architecture using

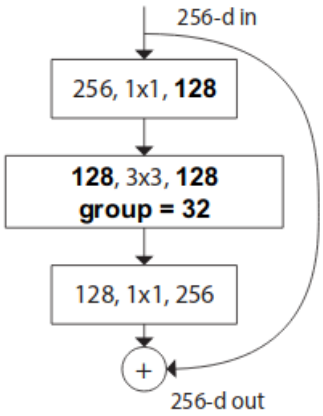


Figure 2.6: The resneXt block proposed in is the first work to combine group convolutions with residual learning.

this methodology, gets performance equivalent to a 101 layer resnet that is trained with smaller filter sizes.

The resneXt paper proposes three equivalent versions of the group residual block. The version used for the imagenet experiments is depicted in Figure 2.6. An extreme case of residual learning is connecting each output in a layer, to each output directly to every layer following it inside a block. This form of residual blocks was proposed in DenseNet [37]. A dense block can be visualized in Figure 2.7. There are 5 blocks in this structure where the first block propagates features to the 4 blocks following it, the second block propagates features to the 3 blocks following it and so on.

2.3 Attention Models

Attention based approaches enhance feature extraction in neural networks. Attention has been widely used in natural language processing, but combining it with convolutional neural networks is a recent advancement, with one of the first prominent works only dating back to 2015 [91]. In the context of this thesis, we define attention mechanisms into two broad categories - 1) pixelwise attention, and 2) global attention. Pixelwise attention techniques learn to localise the important components of an intermediate feature volume in a convolutional neural network,

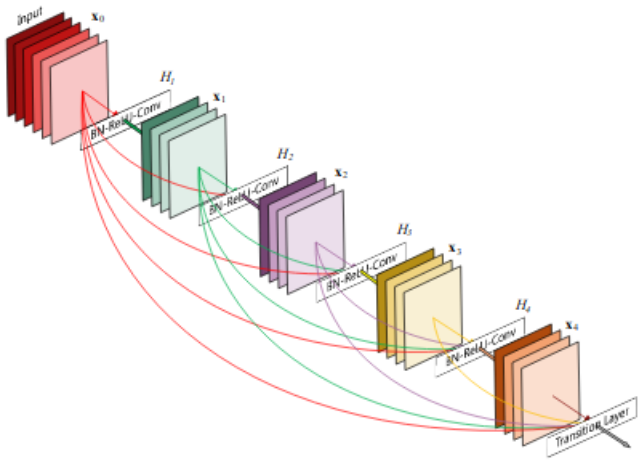


Figure 2.7: The dense block proposed in [37]

while global attention techniques process entire filter maps to learn the most important ones that contribute to the output. Attention infusion inside networks can also be categorised into two types. The output of the attention mechanism can be added to the features, and it can be multiplied to the features as well. Attention addition adds a constant value to each feature map, while the multiplication has the affect of scaling the maps. Attention weights are generally computed via first extracting attention features through convolution operations, and then processing these features by a gating mechanism. The gating is done using a sigmoid function that gives a probability between [0,1] for every input. The output of the gating operation is either added or multiplied to the volume that is input to the attention block. As these blocks contain convolutions and gating functions, they are fully differential and can be placed anywhere in a neural network.

The simplest form of attention networks are the spatial transformer networks [40] that learn the regions of interest from images with random clutter or noise. One of the first major visual attention approaches was a two level approach proposed in [91] where the images were first passed through an RCNN and selective search algorithms to generate proposals. A gating operation using softmax over the imagenet classes was used to get rid of low probability proposals. The remaining patches were then passed through a classifier which in their case was an SVM. The approach worked

well on a subset of the imagenet dataset, but requires a large amount of computation as well as hyperparameter tuning. A diversified visual attention mechanism (DVAN) was proposed in [100] where an attention mechanism extracted patches from an image. The patches could be at different scales. A VGGNet was then used to extract features from these patches which were then combined together with a DVAN LSTM. [78] proposed a combination of CNNs and RNNs to accumulate high multi-resolution glimpses of an image to make a final prediction. A large amount of techniques combine either reinforcement learning [28] or recurrent neural networks [58] along with multistage pipelines to create or process attention information making these techniques slow.

Self attention mechanisms aim to learn context beyond a networks receptive field. The first successful work in incorporating such a mechanism in CNNs was squeeze and excitation networks [36]. They proposed to global average pool feature map information into a single vector creating a global representation, that was then autoencoded and passed through sigmoid gating to generate attention weights for each feature map in an activation output. The maps were then scaled via multiplication with these attention weights. SE Blocks have been extensively used in object detection [8], image segmentation [74] and scene classification [44] to name a few applications. We extensively use variations of SE Blocks in this thesis to construct our FocusNet style attention mechanism as well as the attention mechanisms for our spatially aware layers and global aggregation units in SAWNet.

2.4 U-Net

Initial deep learning applications mainly included learning the task of feature extraction and image classification in an end-to-end setting. The first applications of deep learning for image segmentation were introduced in the paper by Long *et. al* [79] where they introduced a fully convolutional network for the task of semantic scene segmentation. The main contribution of the paper was removing the fully connected layers from a classification neural network architecture and replace it with

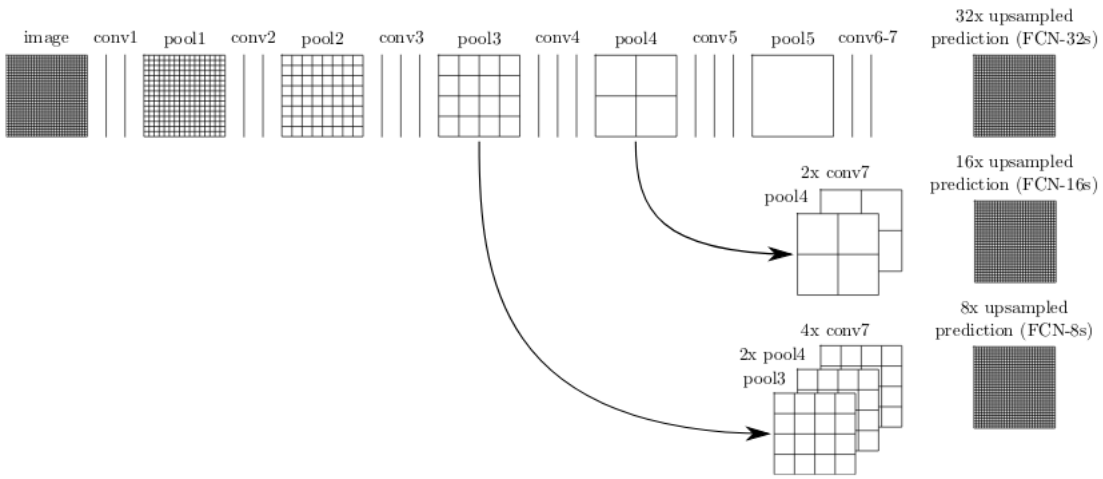


Figure 2.8: The fully convolutional network form image segmentation proposed by [79].

a convolutional layer. Due to this reason, the network came to be termed as the 'Fully Convolutional Network', as it did not contain any fully connected layers. The FCN architecture is shown in Figure 2.8.

Arguably the most influential architecture in image segmentation is the U-Net [73], originally proposed for medical image segmentation. The U-Net employs an encoder decoder structure to map an input image to it's corresponding binary segmentation mask. The architecture for the U-Net is shown in Figure 2.9.

The input image is encoded by the encoder by extracting hierarchical features. This encoder can be any generic convolutional neural network architecture (AlexNet, VGGnet, Inception, ResNet etc). How this architecture differs from its predecessors is that instead of predicting per pixel predictions as a flattened patch output (as done in [14]), or as upsampling to a mask from one intermediate layer (as done in [79]), it hierarchically upsamples to a binary mask while combining information from intermediate encoder layers. The features from the encoder are concatenated with the decoder features to aid better decoding. The features are downsampled using maxpooling and upsampled using upconv/convolution transpose. ReLU activations are used throughout. No batch normalization or dropout are used in the

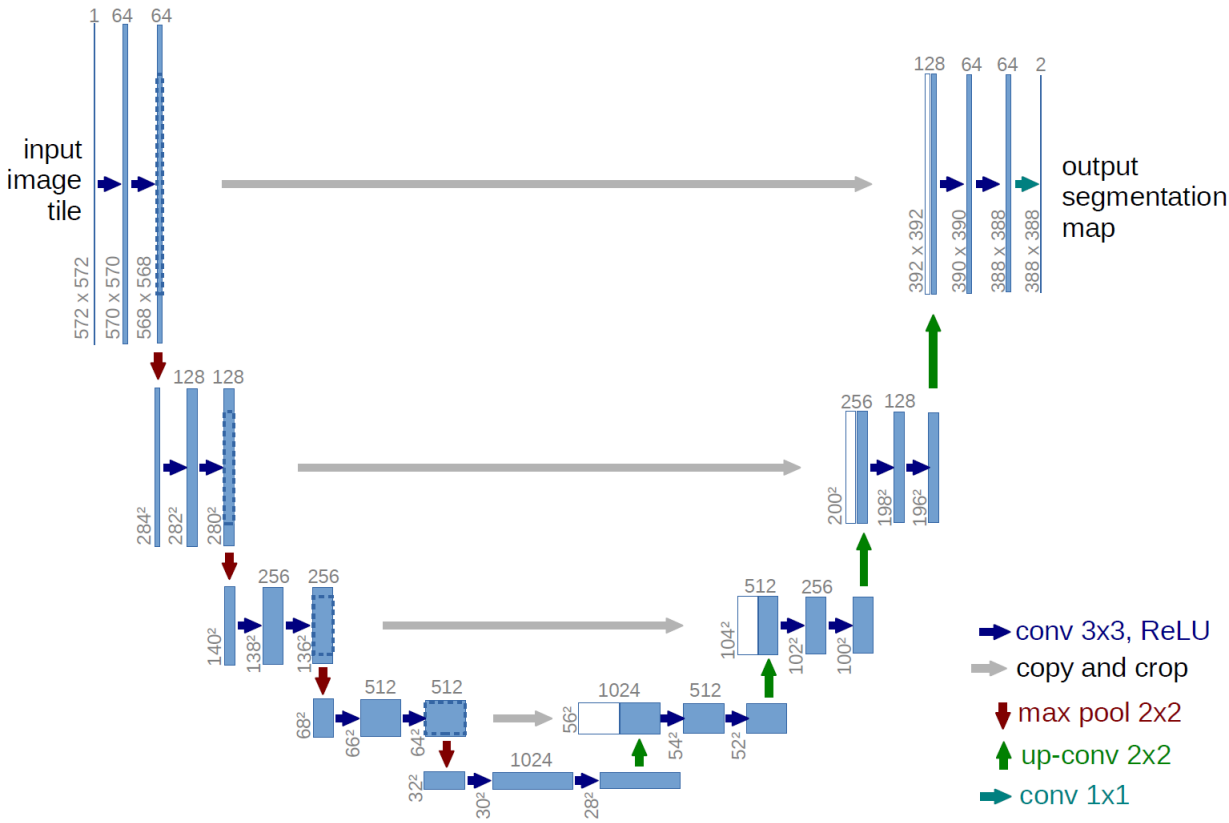


Figure 2.9: The U-Net architecture.

original architecture.

2.5 Effective modifications to U-Net

A machine learning system can only work as well as the features it extracts. To aid effective feature extraction, many different modifications have been proposed to the U-Net. In this section, we discuss the most widely applicable U-Net style architectures which we compare against in this thesis.

- **Wide U-Net** [101] is a U-Net style architecture, with the only difference being an increased number of convolution filters inside the convolutional layers. The architecture does not use residual connections but stacks wider (in terms of number of filters) blocks of sequential convolution operations in the encoder and decoder layers.

- **UNet++** [101] proposes redesigning skip connections in U-Net style architectures via adding fine grained information between the encoder and decoder, rather than a simple skip connection. They propose a dense convolution block (which is not the same as the dense block in DenseNet) that uses a pyramid style scheme to combine features from different encoder layers to pass relevant information from the encoder to the decoder.

- **R2U-net** [3] is an architecture with the same structure as the U-Net, but the convolution blocks are replaced by residual recurrent conv-relu blocks. The operations of the recurrent convolutions are performed with respect to discrete time steps that are expressed according to the fundamental layers of the RCNN.

- **Attention U-Net** [101] is a U-Net architecture with an attention mechanism that decides what information to propagate in the skip connection from the encoder to the decoder. They use a grid based gating approach to allow the attention to be more local, which increases performances compared to using a global information vector.

- **BCDU-Net** [6] is yet another spin on designing efficient skip connections that learn to propagate the most effective features from the encoder to the decoder. This effect is achieved via the use of a densely connected convolution block (from DenseNet) inside the layers learning the latent representation. The information is propagated between the encoder and the decoder via the use of a bidirectional convolutional LSTM.

In general, the trend towards most novel enhancements to U-Net currently involves redesigning better skip connections to pass relevant features from the encoder to the decoder. This works effectively in most cases, but there are two main drawbacks to this approach. First, as the network concentrates on passing information from the encoder to the decoder, if the features extracted by the encoder are not robust and descriptive enough, this hinders learning useful information in the network. Second, as the encoder just uses generic convolutions in its encoder and decoder lay-

ers, the network usually may fail to detect smaller lesions that may be present in the images which can be disastrous in the context of disease diagnosis.

2.6 Advances in medical image segmentation

In this section, we discuss the various techniques for segmenting medical images that have been proposed in the recent past. Pre deep learning, the tasks related to high level image understanding were considered to be hard. Deep learning has revolutionised the field of artificial intelligence, but no other field has been impacted as much as computer vision. The state of the art, as well as benchmarks for almost all computer vision tasks are some variants of deep (and mostly) convolutional neural networks. Due to this reason, we focus this review on the impact of deep neural networks for this task of medical image segmentation. We specifically focus on advancements with respect to the datasets that we have used to benchmark our algorithms.

Historically, the DRIVE retinal blood vessel segmentation dataset has been extensively used for benchmarking medical image segmentation tasks. It is a small dataset of 40 images, but the large image size and fine grain feature processing required facilitates patchwise training on this dataset making training and validations splits quite large. The initial work using deep learning on this dataset [64] treated segmentation as a pixel labelling problem. They constructed a classification style architecture to process 400000 patches sampled from DRIVE to classify the pixels in the patch as foreground or background. Their architecture flattened out the output of convolutions and passed it to fully connected layers for processing. They achieve impressive results mainly to their extensive amount of preprocessing. They use global contrast normalization, zero-phase whitening, augmentations using geometric transformations and gamma corrections. [62] used a similar pixel labelling technique using a deeper network but without anmy data augmentation. Techniques such as DUNet [41] propose a deformable U-Net structure which is a U-Net with the convolution blocks replaced by deformable convolution blocks. Deformable convolu-

tions and deformable ROI pooling are used for convolution kernels to predict offsets from the original kernel, hence learning deformable receptive fields. This method considerably improves performance compared to the U-Net architecture. [64] used a FCN instead of a U-Net in their pipeline for segmenting the vessels.

The ISIC skin cancer dataset [22] is one of the standard datasets when it comes to evaluating the performance of deep neural networks for segmentation, mainly due to its large size and difficulty in terms of pixel imbalance. The state of the art on ISIC 2017 melanoma segmentation [57] applies a patch based training approach on three different scales with heavy data augmentation and preprocessing. Their hybrid model at 3 different scales achieves a validation jaccard index of 0.753, while their single best model achieves a jaccard index of 0.71. [84] use a simple U-Net without any dropout or batch normalization on the ISIC 2017 dataset and achieve a jaccard index of 0.5 which highlights the importance of batch normalization in the U-Net architecture. [4] use an encoder decoder structure with recurrent layers in the decoder to aid segmentation of the cancer regions. [89] create an inception block [83] based encoder decoder architecture to segment the cancer legions.

2.7 Deep learning on 3D point clouds

3D point cloud processing is a subset of an emerging area of deep learning, known as geometric deep learning. Such an orderless arrangement of data points is obtained from sensors which sample points from a 3D surface, resulting in a 3D point cloud. Formally, a point cloud $P = \{v_1, v_2, v_3, ..., v_n\} \in \mathbb{R}^3, \forall\ v_i = \{x_i, y_i, z_i\}$. These point clouds are the closest representation to raw sensor data that is available and processing them directly in this form has greatly interested researchers, giving the created algorithms a more "end-to-end" feel, which is one of the great successes of deep neural networks.

Most preliminary work in 3D shape analysis approached the problem by first extracting hand-crafted features from 3D objects, and then selecting a machine

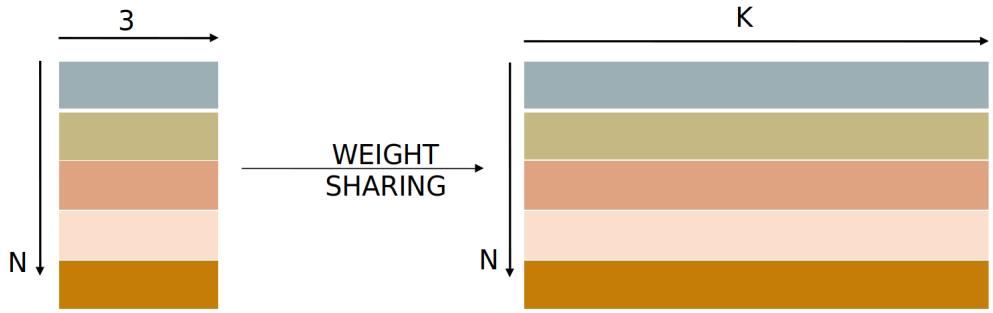


Figure 2.10: Pointnet [68] takes every point in a point cloud and embeds every x,y,z coordinate into a k - dimensional space.

learning algorithm. Usually, some local 3D shape descriptor such as 3D SIFT [77], or spin image [43], was used to extract features, and then they were classified using techniques such as Support Vector Machines (SVMs). Curvature-based methods [52] have also proven successful for the task of 3D shape analysis where principal curvatures are extracted from the shape. These values can be used to calculate shape based metrics such as shape index and curvedness which encodes local shape information, giving local geometric properties of the shape.

For 3D points, there is no grid-like structure to apply the convolution operation on. So several earlier techniques took the 3D points and projected them on a 2D grid like structure, following which, 2D CNNs were used on the projections. Networks such as MVCNN [82] sample 80 views from the point clouds and project them to 2D, which are then processed by a CNN. This is followed by a view-pooling operation, the output of which was fed into another CNN which is used for classification. Another grid-like representation is the voxel based one. VoxNet [61] sample 3D points into a $32 \times 32 \times 32$ 3D occupancy grid which is processed by a 3D CNN. The networks that require projections have some notable drawbacks, the most important of which is the computational expense. Storage and processing of multiple views or voxel based data is computationally expensive and projection of the data on to a view point or a small 3D occupancy grid leads to the loss of important information.

The first works in processing the point clouds directly were based on the observation that the network layers need to learn an order-invariant representation of the input points. Two approaches were introduced in this regard, namely permutation invariance and permutation equivariance. The permutation invariance approach was introduced in the PointNet [68] paper. Pointnet identified the problem with processing points as one that required to consider all permutations of the data presented in the data structure, as they effectively represent the same point cloud. Hence, given a $N \times 3$ point cloud, the deep network would need to take into account all N combinations of these points. They hypothesized a weight shared MLP that processes these points, creates a mapping into a k - dimensional space, where the embedding created is based on all combinations of these points (see Figure 2.10). This weight shared MLP creates a symmetric function that embeds all points. As a symmetric function creates order invariant representation of the data, the representation takes into account all possible combinations of the points. They then takes these embeddings to create a global feature using another symmetric function. They observe max pooling works best as a global symmetric aggregation function. This embedding is then passed through a MLP for classification or segmentation tasks. The major drawback of this approach was that the network only looked at the global point coordinates so this representation was not as robust. Locality information was introduced by the authors in PointNet++ [70] where they used farthest point sampling to uniformly sample points to build a local region, and grouped them based on a radius-based ball query to define a local region. The PointNet architecture was the applied to this local space. These networks have found applications in exploring spatial context for scene segmentation [19], 3D object detection from RGB-D data [67], and even normal and curvature estimation [32] from noisy point clouds. Networks such as SO-Net [55] perform hierarchical feature extraction using Self-Organising Maps (SOMs). The networks discussed so far capture local geometric information using local points. Dynamic Graph Convolutional Neural Networks (DGCNNs) [88] compute edge features based on these points to learn a relationship between the points and its features. These networks recompute the graph based on the nearest neighbours at every layer, and hence they are termed as dynamic. Networks such

as SpiderCNN [93] (SpiderConv) and PointCNN [56] (X-Conv) define convolution operations on point clouds based on local geometric information, rather than using a symmetric function to process them.

Another set of closely related CNN architectures assume point clouds to be graphs, and define convolutions on these graphs by converting the points to a different domain. Keeping with the theme of global and local point processing, graph convolutions, generally, first transform the points using a fourier transform, and then learn the parameters of the convolution filters in the transformed space. This operation can be done patch wise [7], or in a global sense [50]. The transformation bases are estimated in these networks via an eigen decomposition of the graph laplacian matrix, which is a computationally expensive process. Further, computing convolution filters in the transformed space also has added computational complexity, though techniques such as [18] localise convolution filters faster.

2.8 Summary

In this chapter, we reviewed the current literature in the field of residual learning related to medical image segmentation and 3D point cloud processing. We started with an initial discussion of deep neural networks building up to convolutional neural networks, and all the way to the application of encoder decoder structures to medical image segmentation. From our review of medical image segmentation, the following points are fairly prominent -

- The encoder decoder structures employed for these segmentation tasks optimize feature extraction over generic convolution operations, almost always, not reusing any features learnt in the previous layers to propagate down the network encoder, or up the network decoder. Even architectures that do use residual learning do not hand craft feature extraction blocks to suit the task, but use a general Resnet-50 pretrained on imagenet to fine tune. This is an obvious flaw as networks pretrained on imagenet almost always show no improved performance compared to networks trained from scratch on medical imaging.

Infact, they sometimes lead to degraded performance. Further, the amount of data available for training an architecture like Resnet-50 from scratch is definitely not sufficient in most medical imaging tasks, due to which the architectures constructed require more though that using a pre-defined/pre-trained encoder to obtain the latent space encoding.

- Attention mechanisms are not yet widely applied to medical imaging tasks in general. Given their capacity of pixel weighting and highlighting prominent regions inside images, they can surely be useful to segment small regions inside images that non attention based networks miss. It is useful to note that architectures such as Attention U-Net do have an attention mechanism that feeds the important features from the encoder to the decoder to the decoder, but these features are extracted from generic convolution blocks. Hence, if the features extracted (for say a task that requires fine grained feature information) are not good enough (which is a prominent drawback of generic convolution blocks), the network does not learn a lot of useful information.

- Class imbalance is not looked at in terms of the number of foreground and background pixels in these tasks. This problem has been identified as relevant in the medical imaging community only recently with the proposal of the dice loss [99]. Only a handful of useful techniques exist to handle pixel class imbalance and even fewer research acknowledged this as a problem at the time of starting this research.

- As an enhancement to residual learning, having multiscale image information inside the same residual block has not yet been investigated.

In terms of processing 3D point clouds, we make the following observations -

- The use of residual learning, to the best of our knowledge, has not been looked at in the context of point cloud analysis. DGCNN does have skip connections in its architecture, but the concept of feature reuse between consecutive layers has not been looked at in their research. All skip connections instead feed into a single layer.

- No research exists to see the influence of combining global and local embeddings of the point clouds. Such an idea has not yet been experimented with yet.

- Attention mechanisms had not been used at the time of starting this research for the analysis of 3D point clouds in their raw form. No investigation has been done to date, to the best of our knowledge to see the effects of having an attention mechanism over the embeddings that point cloud processing networks create.

- Pointnet identifies the global feature aggregation as a very important step in combining all information. This makes sense as the information here is what feeds into the classification or segmentation system to process the point cloud. Max pooling seems to be accepted as the norm to date to create this global aggregation and is used by all architectures to date. The use of techniques better than just using a max pooling operation has not yet been experimented with for this feature aggregation step.

This thesis is an attempt to fill the gaps in the current literature mentioned above.

Chapter 3

FocusNet

3.1 Introduction

The task of medical image segmentation is generally harder than segmentation of generic scenes. This is mainly due to the fact that there are very small lesions of interest that can go undetected, which can usually be disastrous in the context of disease diagnosis. Due to this reason, architectures such as the Fully Convolutional Network (FCN) [79] do not perform well on medical imaging tasks. The FCN uses an encoder that is generally pretrained on ImageNet to create a latent representation of the image. This pretrained encoder can be the AlexNet [53], VGGNet [81], or the ResNet [34] image classification architectures, which are fine tuned for the task of medical image segmentation. The output from one of the intermediate layers is then upsampled to produce a segmentation mask. The upsampling process is done with no intermediate feature information. Architectures that use hierarchical fine-grained information from intermediate features generally perform better at the task of medical image segmentation. One such architecture is the U-Net [73], that combines features at every intermediate layer instance to create better segmentation masks compared to the FCN. This chapter proposes a novel deep learning architecture, inspired by the encoder-decoder structure of the U-Net. Vanilla U-Nets comprise of a series of convolutions that encode an image into a latent representation, and then decode this representation using a series of upsampling layers, combining intermediate information as they go along. Surprisingly, not a lot of changes have been

proposed to this architecture setting in existing research, to improve its accuracy, as compared to other advancements in the field.

FocusNet, is an incremental improvement to U-Net that uses residual learning, as well as a series of attention mechanisms to propagate the most important information forward in the network, keeping robust feature encoding as it's objective. The architecture incorporates attention within convolutional neural networks using feature maps generated by another convolutional autoencoder that is trained end-to-end with the task at hand. The attention architecture is well-suited for incorporation with other deep convolutional networks.

This chapter is structured as follows.We discuss our attention methodology in Section 3.2. Section 3.3 introduces FocusNet - the attention-based fully convolutional network. Section 3.5 contains the series of experiments including the ablation tests done to arrive at the optimal architecture. Section 3.6 presents our results on various open source datasets. We discuss the problem of class imbalance in pixels in Section 3.4 and discuss the limitations of FocusNet in Section 3.7. Section 3.8 summarizes this chapter.

3.2 FocusNet Style Attention Mechanism

In this section, we discuss our base attention methodology (shown in Figure 3.1), which is extensively used throughout this thesis. The idea is to first learn trainable weights which predict the probability of a pixel's importance in the context of the output, and then recalibrate the global feature maps to learn the most important ones in an activation volume. Let us assume F to be the overall input to this block. Let F_1 be an intermediate feature map volume outputted from the interaction of some intermediate input volume I_1 with a set of trainable weights W_i of a convolution filter. This F_1 is passed through a series of conv-ReLU operations, followed by a final 1x1 conv operation with a sigmoid operation, that gives per pixel probabilities

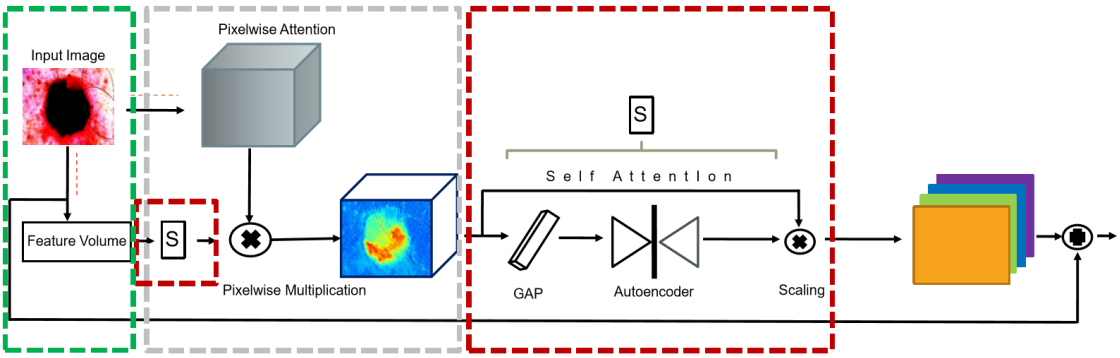


Figure 3.1: The FocusNet style of attention mechanism. Pixelwise probabilities learn to highlight malignant regions inside images containing melanoma (in this context). Feature map recalibration via squeeze and excitation then weights every map to assign higher global weights to the important maps. The dotted boxes correspond to the component blocks in the main architecture diagram shown in Figure 3.2.

for the volume. The learned output F can be denoted as,

$$F_1 = \sigma(\delta(I_1 * \mathbf{W}_1),\ \mathbf{W}_2)$$

where $*$ denotes a convolution between the filters and the learnt weights and δ denotes the (ReLU) non linearity. W_1 denotes the weights for the convolutions corresponding to feature extraction, while W_2 are the weights corresponding to the 1x1 convolution. In parallel, the input is also processed by a series of different conv-ReLU operations. The output of this block of operations has its feature maps recalibrated via the self attention mechanism. This is done by first squeezing the information into a 1D vector to get a global characteristic of each channel and then autoencoding this vector to get a per-channel probability. Let I_2 be the intermediate feature volume to this operation and F_2 be the overall final output of this operation. The squeezing of information is achieved via a global average pooling operation, given by,

$$g = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} I_2 c(i, j)$$

for a volume I_2 with dimensions $H \times W$ and channels c. g is the channel-wise statistic. This output is then autoencoded by compressing the channels c to a latent

representation given by $\frac{c}{r}$ and attempting to reconstructing the input 1D vector. This is given by,

$$f_{\mathrm{x}} = \sigma(\delta(g, \mathbf{W}_{\mathrm{k}}), \mathbf{W}_{\mathrm{r}})$$

. Here W_k are the weights learnt before compression and W_r are the weights learnt to map the input to a compressed latent space. The output from the autoencoder is passed through a sigmoid gating to get a channel-wise probability, which is used to scale each channel. This scaling is given by,

$$s = (f_{\mathrm{x}} \cdot I_2)$$

where $(\cdot)$ denotes each weight value being multiplied with the entire slice of the feature map. This output is then multiplied by the per pixel probabilities obtained earlier that re-weights every pixel inside the volume based on how much it contributes to the output, i.e,

$$P = F_1 \odot s$$

where $\odot$ is the hadamard product (where each F_i is multiplied with each s_i, like pointwise multiplication in convolutions). The output is then recalibrated via the self-attention mechanism (denoted as SA).

$$F_2 = SA(P)$$

following which we add the residual feature map volume from before these series of operations as,

$$F_{\mathrm{out}} = F + F_2$$

The attention block is end-to-end trainable and can be incorporated inside any deep learning system. All the weights described here are learnt via back propagation. If the learnt weights have a positive influence on the output, the resultant error is lower and vice versa.

3.3 Network Architecture

A conventional autoencoder first creates a low dimensional representation of the input and then upsamples from that representation to recreate the input. The low

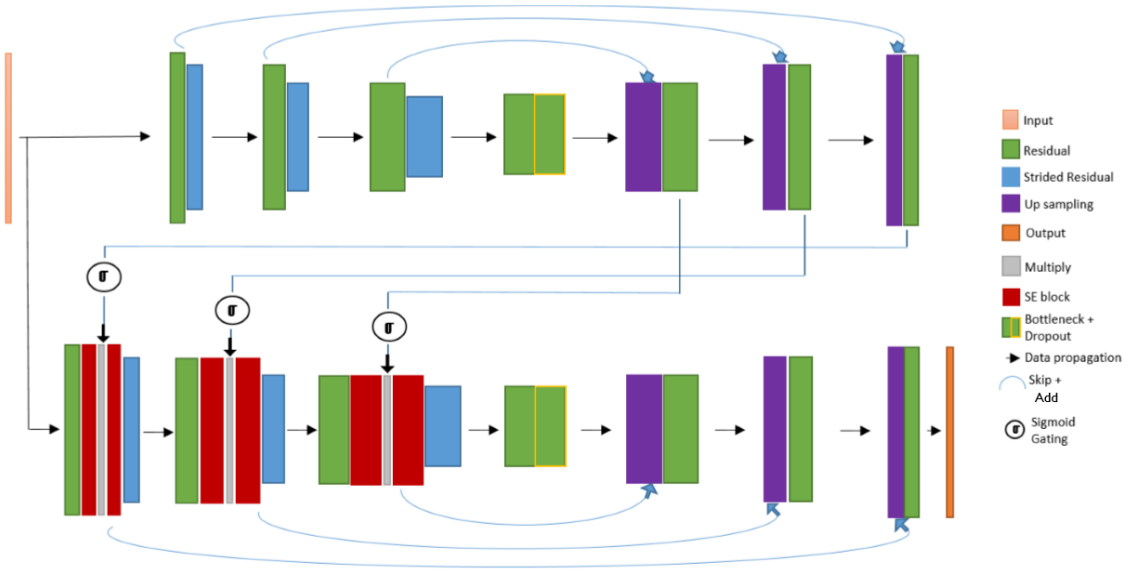


Figure 3.2: The network architecture uses attention to give better per pixel predictions, leading to better segmentation. The two branches are comprised of encoder-decoder structures where the per-layer decoded output is passed through a sigmoid gating function and multiplied with the output of the first SE block. The direction of the arrows show the direction of information flow in the network.

dimensional representation created by the encoder is known as a latent representation, or a bottleneck representation. We exploit such an encoder-decoder architecture structure to hierarchically extract latent attention maps with the aim of creating a superior encoding of the data. The FocusNet architecture (see Figure 3.2), employs two parallel branches of information flow with one branch solely devoted to calculating the attention maps. The attention branch employs an encoder-decoder structure with skip connections from the encoder to the decoder to facilitate better gradient flow. The architecture provides a strong bias for the two networks to specialise and learn different representations.

Focusnet is a fully convolutional end-to-end trained deep neural network that performs the task of learning attention maps, as well as segmentation. We employ two attention mechanisms in this architecture to propagate the most important information forward. Keeping Figure 3.2 as reference, first, we use the top autoencoder

branch to calculate pixel-wise probabilities for each input pixel's contribution to the output segmentation. This is done by taking the output of the decoder with the same filter maps as the encoder layer, and passing it through a sigmoid function. Following this gating operation, the resultant here is a probability distribution over the pixels in the bottom encoder's intermediate layer. This output is then multiplied with the activation map output in the intermediate layers to suppress the pixels that don't contribute to the final output. A self-attention mechanism is also used along with this. Once the pixel-wise probabilities are multiplied by the intermediate output, we also calculate channel-wise probabilities to learn which channels across the depth of the intermediate outputs contribute most to the output. The attention mechanism at each step first learns the most important pixels in each intermediate volume, followed by learning the most important filter maps across the depth of an intermediate activation. This leads to a better latent space which is then decoded by the decoder.

Given an image, $x_i \in X$ where X is the mini batch, each layer of an encoder learns a mapping G, given by,

$$E_l = G_l(x, \mathbf{W}_l) \tag{3.1}$$

The decoder corresponding to this layer, decodes this representation in the following form:

$$D_l = [G_l(x, \mathbf{W}_l) \ ; \ H_l(H_{l-1}(x, \mathbf{W}_{l-\mathbf{1}}))] \tag{3.2}$$

where [;] denotes addition via skip connection and H_{l-1} is the output from the previous decoder layer.
In the encoder in the second branch, the output can be represented as:

$$A_l = F_l(x, \mathbf{W}_l) \cdot \sigma(D_l) \tag{3.3}$$

where A_l is the output of the l^{th} layer of the second encoder after *gating-and-multiplication.*

The superior performance of the network is due to a number of changes from the conventional autoencoder style architectures. Instead of using generic vanilla con-

volutions to learn our features, we use a no bottleneck full pre-activation residual block [35] (see Figure 3.2) to learn our feature representations. Downsampling in the network is done using strided convolutions rather than max pooling. A constant stride of 2 is used whenever downsampling is used. Two attention mechanisms are employed to learn a better encoding of the data. Architectures generally use skip connections from the encoder to the decoder, and concatenate the representations. We observed that adding the representations rather than concatenating them leads to better performance in our network architecture. This also means that the filter maps following the addition operation are halved in every layer, compared to concatenating. The output of this network is obtained via a 1x1 convolution with a sigmoid activation that outputs per-pixel predictions of the segmentation map. The rest of the convolutions have a 3x3 receptive field. Where stride is used, a 1x1 convolution is employed to keep the dimensions of the activation maps similar. The filter bank volumes used are $32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64$.

Skip connections throughout the architecture facilitate better gradient flow, leading to easier training of a deeper network. Batch Normalization is also used inside the residual block. Dropout is used with a fixed rate of 0.5 in the penultimate encoder layer, as well as the bottleneck. This means that at every pass of the data, 50% of the learnable parameters in those layers are randomly omitted from the network. We observed this regularization technique to be quite useful in dealing with overfitting. The network was trained from a He-normal initialization of the weights throughout.

3.4 FocusNet and imbalanced segmentation

We now shift our focus to using FocusNet for segmentation tasks with a large imbalance between background and foreground pixels. Real word data hardly has any balance between the pixel classes and methods need to be introduced to deal with such segmentation tasks. We create a methodology in terms of a novel loss function enhancement that deals with imbalance and convergence close to the minimum

of the solution of the optimization problem that the network architecture weights pose. We feel such a technique is relevant as little attention has been given to enhancing loss functions to better traverse the loss landscape. Hence, we attempt to simultaneously and significantly mitigate two prominent problems in medical image segmentation namely: i) class imbalance between foreground and background pixels and ii) poor loss function convergence. To this end, we propose an adaptive logarithmic loss function which is described henceforth. We use this loss to enhance the properties of the dice loss using our methodology. We conduct an extensive hyper-parameter search for our function and empirically show that our technique leads to better convergence of the dice loss under even less optimal settings of our function. We compare with state-of-the-art for the same problems, and show performance gains over them. We primarily use the U-Net and FocusNet architectures to compare results. Our enhancement experiments with the dice loss due to its popularity in medical image segmentation tasks, but in theory, any loss function could be used here.

3.4.1 Adaptive logarithmic loss

We motivate the need for our loss based on the properties of a good loss function. Once a loss function computes the error between the label and the ground truth values, the error is backpropagated though a network in order to make it learn. This fundamental task is generally conducted well by all loss functions, though some tend to converge faster than the others. Empirically, tversky loss converges in lesser epochs compared to the earlier proposed losses such as CE or the Jaccard loss. A good loss function should not take too long to converge. It is an added bonus if it speeds up convergence. Secondly, a loss function should be able to adapt to the loss landscape closer to convergence. Keeping these points in mind, we construct a loss function that can both, converge at a faster rate, as well as adaptively refine its landscape when closer to convergence. The formula for this adaptive loss is given by,

$$ALL(x) = \begin{cases} \omega \ln(1 + \frac{|DL|}{\epsilon}) & |DL| < \gamma \\ |DL| - C & \texttt{otherwise} \end{cases} \tag{3.4}$$

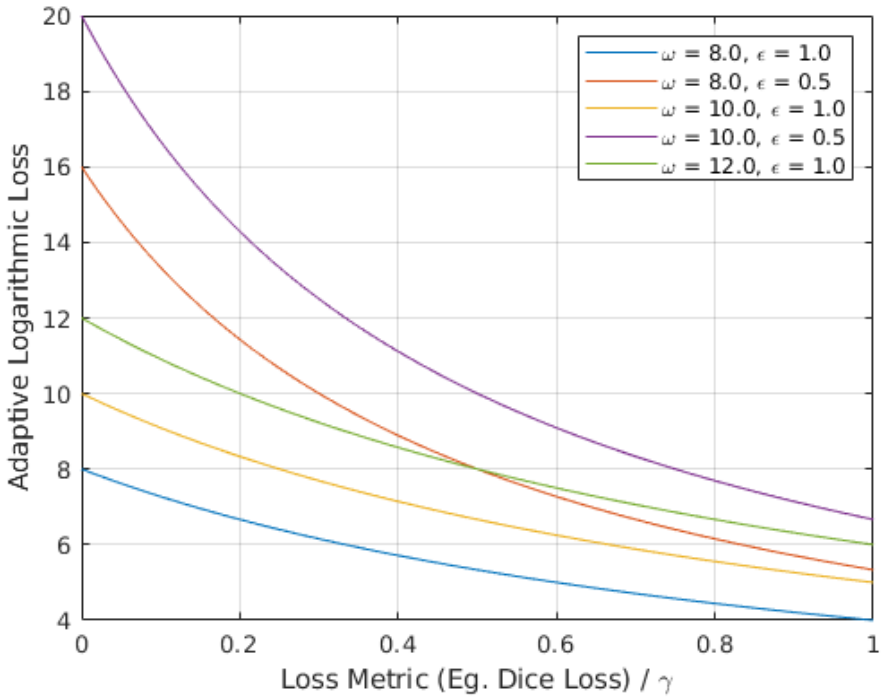


Figure 3.3: The plot shows the value of the derivative of our loss against the value of the dice loss that it optimizes. It can be seen that for smaller values of the loss metric, a larger loss is backpropagated. γ is fixed empirically based on initial experiments to any value on the x-axis.

where $C = \gamma - \omega \ln(1 + (\frac{\gamma}{\epsilon})$ is used to make the loss function differentiable and smooth at $|DL| = \gamma$ and DL is the computed dice loss. γ, ω *and* ϵ are hypermaramaeters of this loss function. Further, as the dice loss lies between [0, 1], we experiment with values of γ that are [0, 1] to find the optimal threshold to shift to a smoother log based loss for convergence close to the minima. As log is a monotonic function, it smoothens the convergence. The derivative of this loss can be computed via the chain rule. It is visually shown in Figure 3.3. Differentiating a function of a function results in the product of two derivatives. So, $[ALL(DL(\cdot))]' = ALL'(DL) \times DL'(\cdot)$, where the plot of $ALL'(DL)$ is shown in Figure 3.3. Hence, given any loss as input to our adaptive function, it's derivative will be multiplied by a smooth differentiable function that would in turn remove any discontinuities. The loss resembles the form of functions proposed in [26] [87] but to the best of our knowledge, this is the first

time it has been adapted for any image segmentation task.

After experimentation, we found the optimal values for the hyperparameters to be, $\gamma = 0.1,\ \omega = 10.0\ and\ \epsilon = 0.5$.

3.5 Experiments

In order to create the optimal network architecture, we constructed a series of ablation tests to observe the effect of the different network components on our architecture. Once the optimal setting for the architecture was found, we tested the results of the final architecture with the state of the art.

All experiments presented here were done on one Nvidia GTX 1080Ti with a batch size of 8. The Keras [11] deep learning library with a tensorflow backend was used for the experiments. The networks were trained using the dice coefficient loss. We subtracted the value of the loss from 1 to get a value between $[0, 1]$ for mathematical convenience, such that the loss could converge reducing from 1 towards zero. These hyperparameters are constant for all ablation test experiments unless stated otherwise. Additional hyperparameters if relevant, are stated in their respective subsections. For all experiments in the ablation study, we report the Sensitivity, Specificity, Accuracy and Dice Loss.

3.5.1 Ablation Tests

The ablation studies (see Figure 3.4) were done keeping in mind two goals - To observe the effects of adding and removing certain blocks from the network architecture and the best way to incorporate attention into the network architecture. We use the ISIC 2017 [15] melanoma skin cancer segmentation dataset for these experiments. The dataset contains 2000 RGB images in its training set, 150 images in its validation set and 600 test images. The images are high resolution and of varying sizes. The masks are 8-bit grayscale images with intensity value zero representing

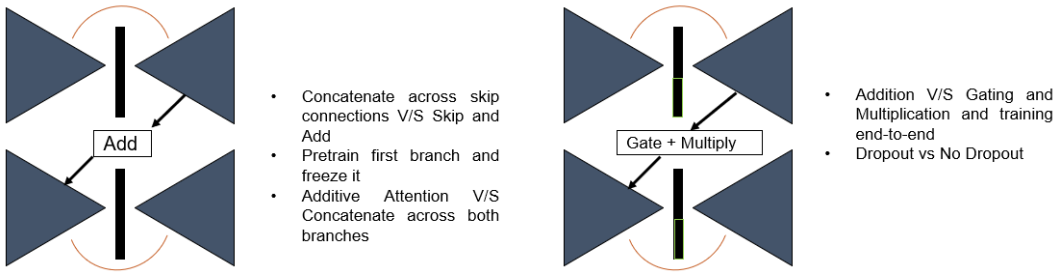


Figure 3.4: The following figure provides a brief summary of the different ablation studies conducted to empirically find the optimal structure for FocusNet.

the background, and intensity value 255 representing the cancer region. In terms of preprocessing, we first resized the images to a constant 256x256 size, after which, we scaled the value of each pixel in the input image to a value between [0, 1]. We did not subtract the mean RGB colour from the images as it seemed to give inferior performance. The masks were first thresholded to an intensity of 0 and 255 and then each pixel was divided by 255 to get a binary mask. No other preprocessing was applied for these experiments. No data augmentation was used for these experiments.

The first experiment was designed to see the effectiveness of residual learning over sequential feature extraction using generic convolutions. To this end, we trained three networks. The first network (Net-1) had the same dual branch encoder decoder structure as FocusNet, but the residual blocks were replaced with convolution operations, leaving the rest of the architecture the same, as described in Section 3.3. The second network (Net-1-BN) was same as the first network, but we added batch normalization before the activation functions to overcome the vanishing gradient problem. The third network was FocusNet. The results are summarised in Table 3.1. We observed that batch normalization considerably improves the performance of the architectures giving almost a 6% increase in the dice index.

Method	Sensitivity	Specificity	Accuracy	Dice
Net-1	0.6738	0.9076	0.8694	0.7167
Net-1-BN	0.7189	0.9432	0.8868	0.7714
FocusNet	0.7673	0.9896	0.9214	0.8315

Table 3.1: Results for Residual Learning v/s Sequential Feature Learning. The results are on the test set of the ISIC 2017 dataset.

3.5.1.1 Pixel-Wise Attention

The next experiment deals with the optimal combination of the pixel-wise probability maps with the overall network architecture. Keeping the underlying FocusNet architecture as in Section 3.3, we tried three different techniques to this end. In the first instance, the output of the feature maps, was not passed through the sigmoid gating function, to get a pixel-wise probability, but was directly concatenated with the layer output (Net-Concat). In the second instance, the feature maps were added to the intermediate layer output (Net-Add), and the final instance was the original FocusNet gating-and-multiplication operation. The results are summarized in Table 3.2. Adding or concatenating features directly across the two autoencoders was not the optimal technique to pass information through the network. SE after the residual block recalibrates the weights of the summation of the identity with the residual, which is not optimal. Optimizing over the learnt residual gives the best results because that is the trainable portion of the residual block.

3.5.1.2 Self-Attention Incorporation

An ablation study was also performed to test the optimal placement of the Self-Attention module, to note the best method for feature map recalibration. This attention module is used only in the main branch of the architecture and not in the attention branch as the computational overhead required to recalibrate the feature maps in the attention layer does not lead to any meaningful improvements in the

Method	Sensitivity	Specificity	Accuracy	Dice
Net-Concat	0.7297	0.9486	0.9012	0.8031
Net-Add	0.7381	0.9637	0.9125	0.8164
FocusNet	0.7673	0.9896	0.9214	0.8315

Table 3.2: Results for different methods of pixel-wise attention incorporation. The results are on the test set of the ISIC 2017 dataset.

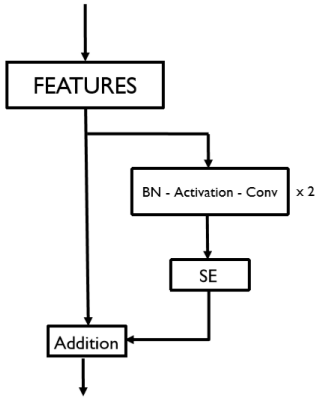
overall results. The different ablation settings are shown in Figure 3.5. Initial experiments worked with feature extraction followed by the attention module before the residual block (Attend-Before-res) which performed better than recalibrating feature maps after residual addition (Attend-After-res). The best results were obtained with feature map recalibration within the residual block, both before and after the feature extraction (FocusNet). The results for each setting are shown in Table 3.3.

Method	Sensitivity	Specificity	Accuracy	Dice
Attend-Before-res	0.7544	0.9760	0.9184	0.8217
Attend-After-res	0.7396	0.9683	0.9145	0.8096
FocusNet	0.7673	0.9896	0.9214	0.8315

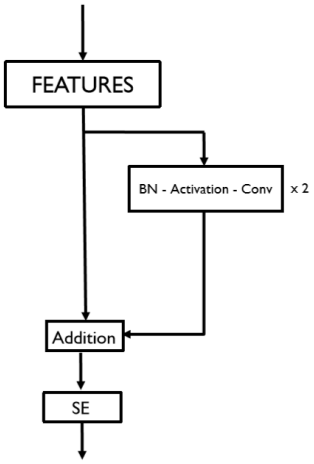
Table 3.3: Results for different self attention incorporation methods. The results are on the test set of the ISIC 2017 dataset.

3.6 Results

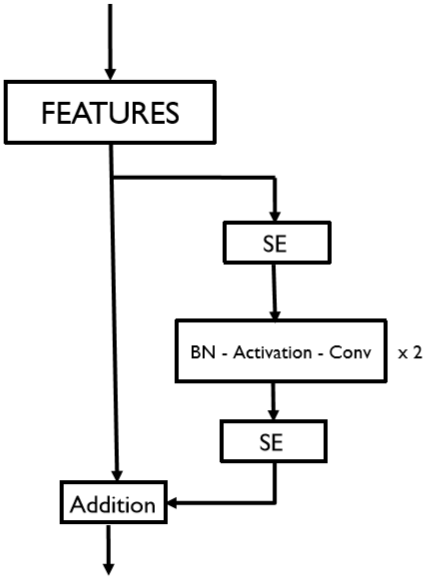
We compare the results obtained using FocusNet on benchmark datasets with the state of the art results. We show the results for our architecture here on two different datasets - the skin cancer segmentation dataset used for the ablation studies [15], and the lung segmentation dataset [60].



(a) SE after feature extraction.



(b) SE after residual block.



(c) SE before and after feature extraction. Most optimal setting.

Figure 3.5: Different ablation settings for SE block placement.

3.6.1 Melanoma Segmentation

The data used here is the same as described in Section 3.5.1. The exact same train-val-test split with the same data preparation strategy is used. We randomly zoomed into images slightly (0% - 10%) and flipped them horizontally and vertically to increase the dataset size to 6000 images.

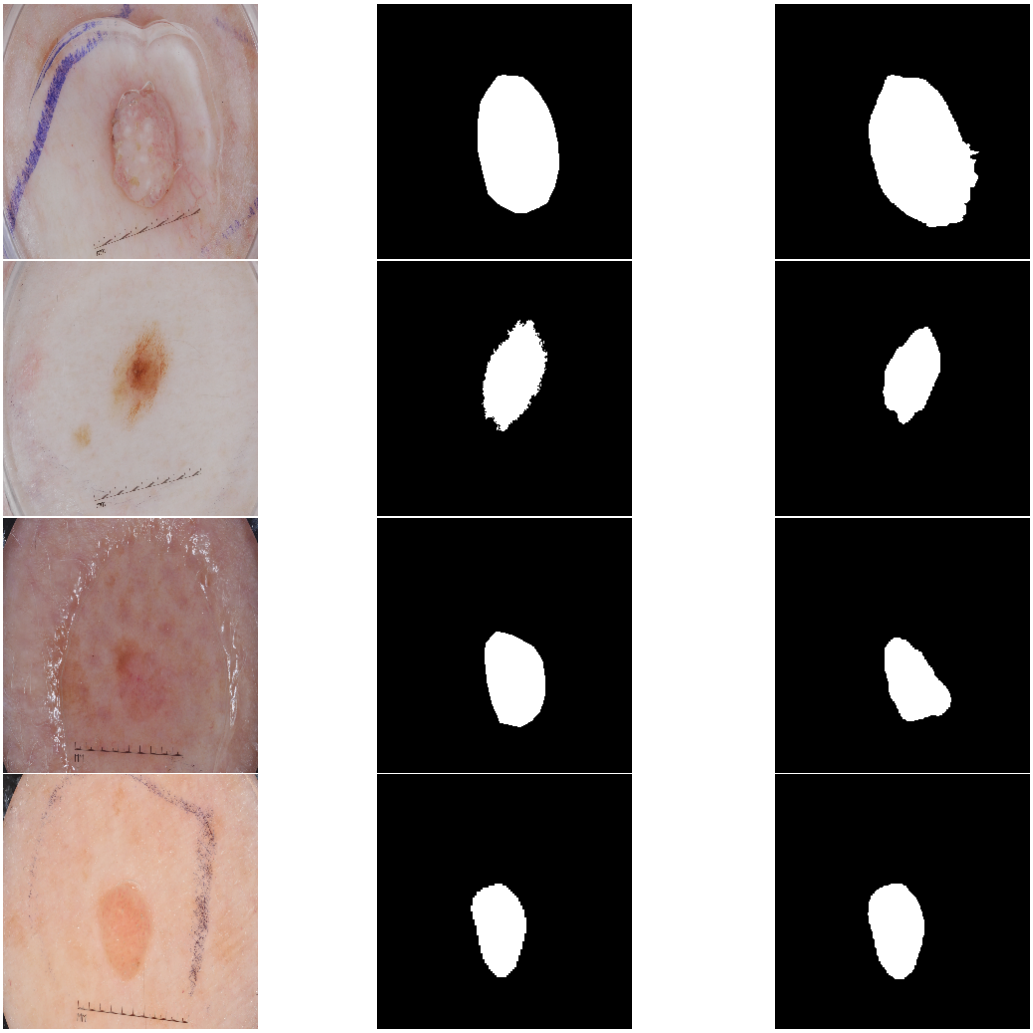


Figure 3.6: Experimental segmentation results on the melanoma dataset. Column 1 is the input image, column 2 is the ground truth, and column 3 is the segmentation.

Table 3.4 presents our results on this dataset. We mainly compare with the multiscale training approach presented by the [57]. In general, our results are comparable with the recent results on this dataset without applying any extensive pre or post processing. Our network performs at par with the multiscaled approach

Method	SE	SP	AC	JI	DI
FCN-8s [57]	0.806	0.954	0.933	0.696	0.783
U-Net [73]	0.853	0.957	0.920	0.651	0.768
II-FCN [89]	0.841	0.984	0.929	0.699	0.794
Auto-ED [4]	0.836	0.966	**0.936**	0.738	0.824
Thao *et al.* [84]	0.6513	0.9421	0.8772	0.5065	0.6317
LIN [57]	**0.855**	0.974	0.934	0.753	**0.839**
FocusNet (ours)	0.7673	**0.9896**	0.9214	**0.7562**	0.8315

Table 3.4: Segmentation results on the test set for skin cancer detection. We extend the table presented by [57] with a few more results [4], [84], including ours. The results on the FCN and U-Net are reported from [57] and have been trained on data pre-processed using their strategy but keeping the same split.

of [57]. We believe a reason for that to be the strong bias a dual branch encoding emphasises via our architecture setting. It forces the second encoder to learn a more robust and distinct representation of the data - one that is different from the one learnt by the first encoder, leading to a superior latent representation. The results obtained by our approach are visualized in Figure 3.6.

3.6.2 Whole Lung Segmentation

The lung segmentation dataset contains 2D images in .tif format with provided ground truth segmentation maps. The images are single channel with the size of 512x512 pixels. The dataset is fairly small, containing a total of 267 images. We applied the same augmentation strategy as we did in Section 3.6.1 to increase the dataset size to 1700 images.

Table 3.5 shows our results on the kaggle lung segmentation dataset. We keep these results in the evaluation to show that though this dataset is saturated by current state of the art architecture results, we still get a better performance than the benchmark U-Net architecture in every respect. The qualitative results for this

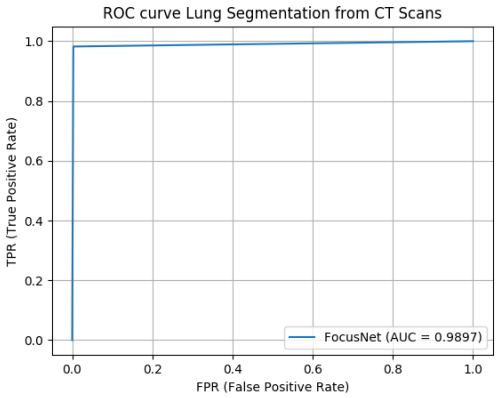


Figure 3.7: ROC AUC for the lung segmentation dataset.

Method	SE	SP	AC	JI
U-Net [3]	0.9696	0.9872	0.9828	0.9858
Res-U-Net [3]	0.9555	0.9945	0.9849	0.9850
RU-Net [3]	0.9734	0.9866	0.9836	0.9836
R2U-Net [3]	0.9826	0.9918	0.9897	0.9897
R2U-Net [3]	**0.9832**	0.9944	0.9918	0.9918
FocusNet (ours)	0.9757	**0.9981**	**0.9932**	**0.9965**

Table 3.5: Segmentation results on the validation set for lung segmentation dataset. We extend the table presented by [3] with our results on the dataset.

dataset are visualized in Figure 3.8. The ROC curve can be seen in Figure 3.7.

3.6.3 Evaluating the adaptive logarithmic loss

The experiments for our loss methodology are conducted with two architectures. We use the benchmark U-Net [73] and the attention based FocusNet. A generic U-Net is enhanced with batch normalization, dropout and strided downsampling to improve on it's performance. The FocusNet architecture used is exactly the same as in Section 3.3. We use 3 datasets that exhibit varying class imbalance for our experiments, to study the effect of our loss on them. The ISIC 2018 skin cancer segmentation dataset [22], the data science bowl 2018 cell nuclei segmentation dataset,

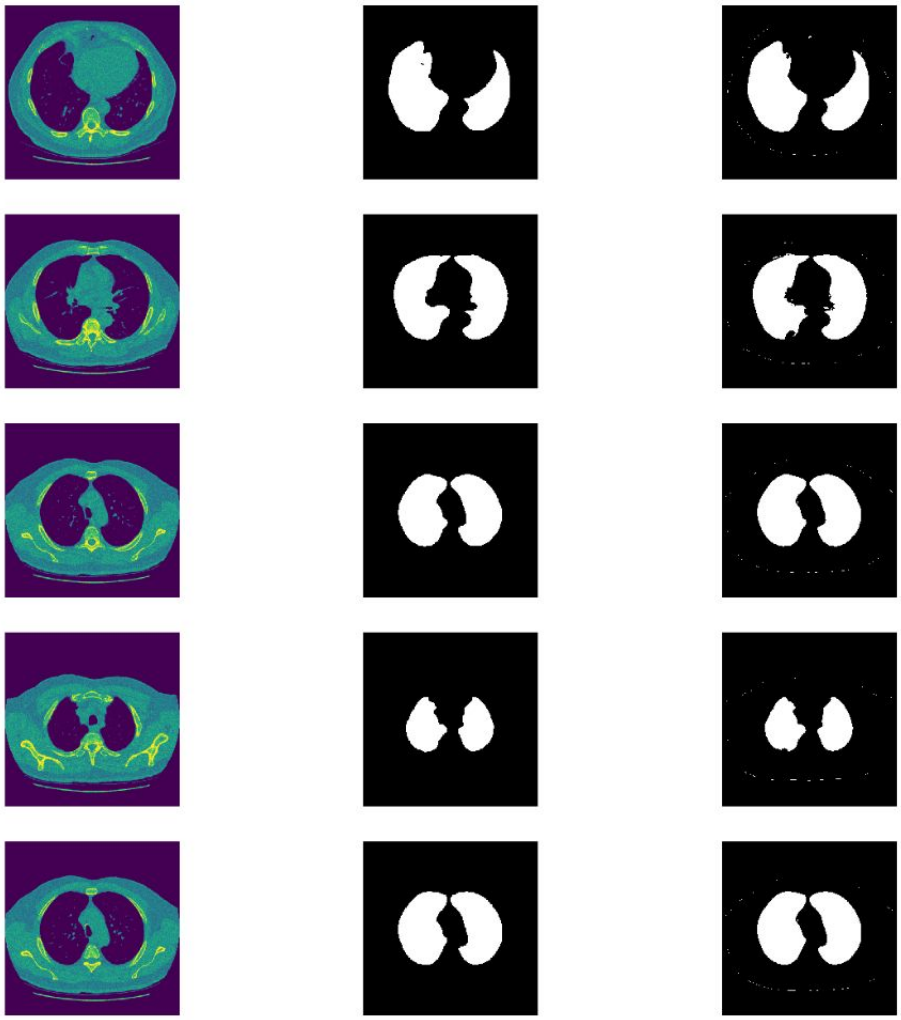


Figure 3.8: Results from lung segmentation. Column 1 is the input image, column 2 is the ground truth, and column 3 is the segmentation.

and the DRIVE retinal vessel segmentation dataset [21] are used. We don't apply any preprocessing excepting resizing the images to a constant size and scaling the pixel values between [0,1]. For the DRIVE dataset, we extract 200,000 small patches from the images (mostly within the field of view, along with some edge cases) to construct our dataset. The images for the ISIC 2018 dataset were resized to 192x256, keeping with the aspect ratio of the training set. The images for the cell nuclei segmentation task were resized to 128x128. The patches extracted from the DRIVE dataset were of the size 48x48. The data for all experiments is divided into a 80:20 split. To keep the evaluation fair, we do not use any augmentation strategies

ϵ \ ω	6	8	10	12	14	16
0.3	81.43	81.48	81.51	81.59	81.90	81.67
0.5	81.97	81.57	**82.43**	82.24	81.58	81.07
1.0	81.78	82.11	81.84	81.73	82.21	81.96
2.0	81.75	81.99	82.18	81.58	81.71	81.63

Table 3.6: Optimizing the values of ω and ϵ over the corresponding Jaccard Index (%). Values are average of 3 runs. Experiments conducted with constant $\gamma = 0.1$. JI with baseline dice loss = 71.36. Results obtained using FocusNet.

γ	0.08	0.10	0.12	0.15	0.20	0.30
JI	81.60	**82.43**	81.54	81.51	80.85	80.97

Table 3.7: Optimizing the values of γ over the Jaccard Index (%). Values are average of 3 runs. Experiments conducted with constant values of $\omega = 10, \epsilon = 0.5$. JI with baseline dice loss = 71.36. Results obtained using FocusNet.

as different augmentations can effect performances differently.

We apply a grid search style strategy to find the optimal hyperparameters of our loss, where we first run some initial tests to see the behaviour of the loss given some hyperparameters, and then tune them. Initially, we set $\gamma = 0.1$ and tuned the values of ω *and* ϵ to their optimal settings. Then we use the empirically estimated ω *and* ϵ to find the optimal value for γ. The values obtained are shown in Table 3.6 and 3.7. From the tables we can see that the loss isn't affected adversely by changes in ω *or* ϵ, but even small changes in γ can cause significant changes in the loss value. This is graphically verified via the derivative of the loss in Figure 3.3 where we can see that for small values of γ, the penalty for getting a prediction wrong is a lot larger as the derivative of the logarithmic function shows a highly non linear response for small γ values.

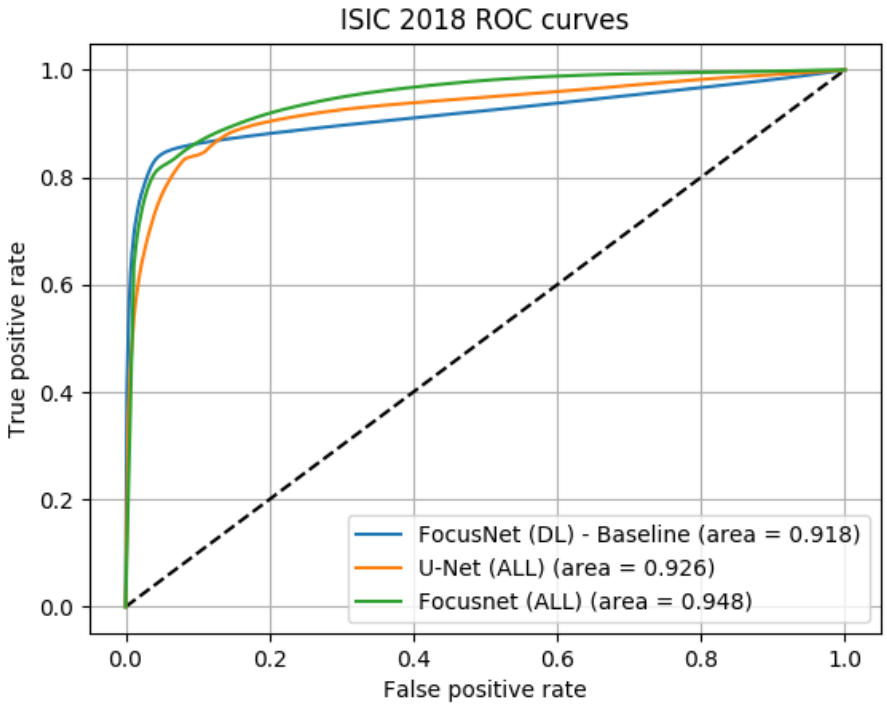


Figure 3.9: The ROC curves for the ISIC 2018 skin cancer segmentation dataset. Our loss has a better Area Under the ROC curve than the baseline. The curves are plotted for the best performing models for our experiments on this task.

All experiments were run using Keras with a tensorflow backend. Adam with a learning rate of 1e-4 was used. A constant batch size of 16 was used throughout. The experiments were run for a maximum of 50 epochs. To evaluate the performance of our loss, we compute the intersection over union (IoU) overlap, recall, specificity, F-measure and the area under the receiver operator characteristics curve (AUC-ROC) of the corresponding network predictions trained on various loss function.

3.6.4 Improving FocusNet with the adaptive logarithmic loss

We compare the performance of our loss (Table 3.8) with the jaccard loss (JL), dice loss (DL), tversky loss (TL) [75], focal loss (FL) [24] and the combo loss (CL) [23]. The ISIC 2018 dataset shows the least imbalance and hence the results are fairly

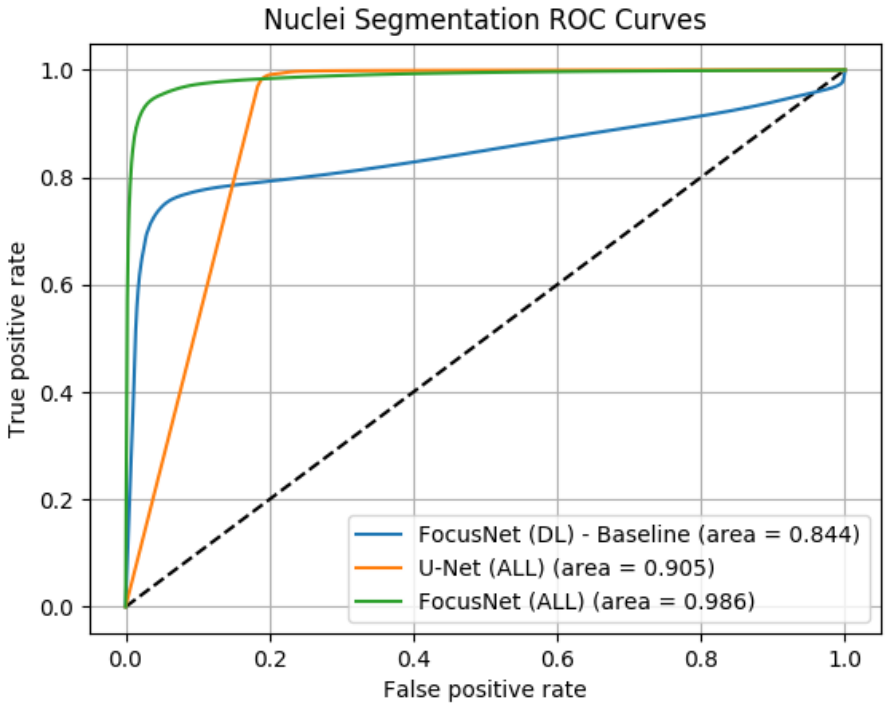


Figure 3.10: The ROC curves for the Data Science Bowl 2018 cell nuclei segmentation dataset. It can be seen that when the imbalance is high, our loss provides a much more robust and significant Area Under the ROC curve than the baseline, demonstrating a superior convergence. The curves are plotted for the best performing models from our experiments on this task.

even for this dataset, though our loss does manage to get the best in class IoU. This is also exhibited in Figure 3.9 where we plot the ROC curves. Focusnet with the ALL gets the best area under the curve. Overall, Focusnet even with the dice loss, gets a fairly competitive AUC score compared to the U-Net with the ALL loss. We also compared Focusnet trained with the ALL against the recently proposed focal tversky loss [2] for the ISIC 2018 dataset. We used the same train-test split as their implementation based on their open sourced github code and averaged our results over 3 runs to report our results. Our loss outperforms their architecture trained on their loss by 1.53 % on the dice index. We also report a better precision and recall than their methodology. The results for this experiment are shown in Table 3.9. The nuclei segmentation dataset exhibits more class imbalance, and it

is in such cases where our loss shows significantly superior performance compared to all the other losses. We get significant gains over the baseline AUC as shown in Figure 3.10 and FocusNet with the ALL loss has a 8.1% more AUC coverage than the U-Net with the same loss. FocusNet with just the dice loss suffers from poor convergence and does not get competitive results. We do not compare the DRIVE dataset by the AUC or accuracy as these metrics are fairly saturated for this dataset and don't offer any statistically significant insights. It is interesting to note that we do get an improved F-measure score and the best in class IoU, which given the large number of patches extracted, is statistically significant. Overall, our loss shows significantly better performance than the baseline dice loss for all three datasets, which means that it manages to optimize the loss to a significantly better minimum on the loss landscape leading to a more optimal solution. In all cases, the trend shown by our loss is to converge to within delta of the optimal solution and then refine the convergences using the adaptive strategy. Without the adaptive strategy, our loss often gets stuck in local minimum, which reiterates the importance of having such a piecewise continuous loss. The other loss functions (especially dice loss) exhibit slightly unstable convergence and also take longer to reach within delta of the optimal solution. We observed that our loss always converged faster than JL, DL, TL and CL. FL convergence is at par with our loss, while TL converges more smoothly. We verify this visually by plotting the behaviour of the losses against the number of epochs (see Figure 3.11).

Method	ISIC 2018			Data Science Bowl 2018			DRIVE		
	Recall	Specificity	Jaccard	Recall	Specificity	Jaccard	F-Measure	Recall	Jaccard
U-Net (JI)	78.62	85.21	72.96	76.27	81.29	73.64	78.46	73.28	65.37
U-Net (DL)	76.12	83.74	69.34	74.92	82.85	64.57	78.94	74.10	67.79
U-Net (TL)	80.82	86.98	74.18	79.21	85.81	77.72	79.89	74.47	66.18
U-Net (FL)	**83.76**	**89.85**	**79.17**	78.27	86.88	78.82	80.07	75.65	68.96
U-Net (CL)	82.19	87.96	75.87	77.34	85.63	78.24	79.26	74.33	67.42
U-Net (**ALL**)	83.56	88.47	77.69	**79.88**	**87.27**	**79.71**	**81.41**	**75.83**	**69.23**
FocusNet (JI)	80.13	86.17	72.28	77.12	84.19	74.97	77.87	73.28	64.67
FocusNet (DL)	80.78	85.81	71.92	78.37	84.92	77.82	77.86	73.98	64.71
FocusNet (TL)	84.86	90.62	77.63	79.64	88.27	77.28	81.31	74.19	68.66
FocusNet (FL)	86.19	**93.95**	82.78	79.26	89.17	78.73	81.28	**76.89**	69.57
FocusNet (CL)	84.82	86,19	78.63	80.65	87.34	79.35	78.64	74.18	68.34
FocusNet (**ALL**)	**86.62**	92.78	**82.84**	**82.51**	**90.86**	**81.37**	**82.17**	76.13	**70.96**

Table 3.8: Segmentation results for the three datasets. All values in the ISIC 2018 experiments, Data Science Bowl and the DRIVE retinal blood vessel segmentation datasets are averaged over 5, 3 and 2 runs respectively to average out the effects of random weight initialization as much as possible. The values reported are all in %.

Method	Dice	Precision	Recall
U-Net (FTL) [2]	82.92	79.74	92.61
Att-U-Net+M+D (FTL) [2]	85.61	85.82	89.71
FocusNet (**ALL**)	**87.14**	**88.11**	**90.47**

Table 3.9: Experiments run for the ISIC 2018 dataset training-validation-test split in [2]. Our reported values (in %) are averaged over 3 runs. 'M' denotes Multi Scale Input. 'D' denotes deep supervision.

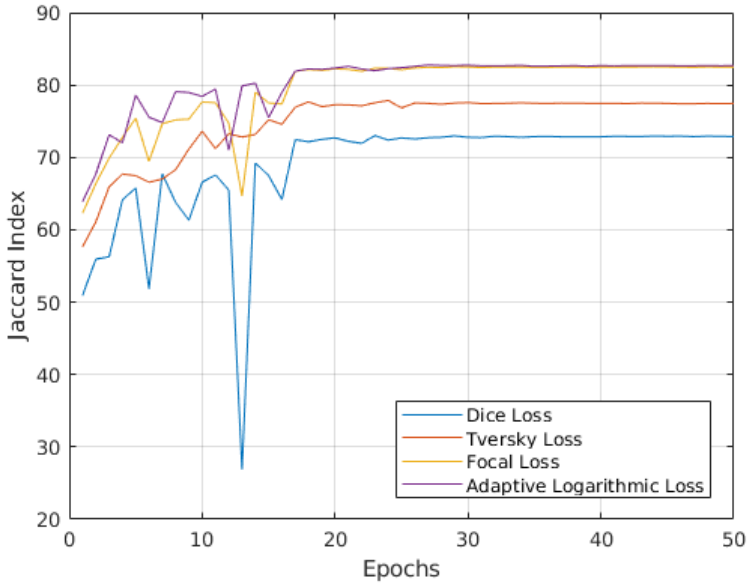


Figure 3.11: Comparing the influence of different loss functions on FocusNet. The plot shows the validation Jaccard Index on the ISIC 2018 dataset vs the number of epochs.

3.7 Limitations of FocusNet

FocusNet's strength mainly lies in it's robust attention mechanism and superior data encoding. And though it performs well and gets accurate results across various datasets, it does have some inherent drawbacks. Evaluating FocusNet on the ISIC 2017 dataset, we noted two main drawbacks about the architecture. First, in order to obtain attention maps, however accurate they are, FocusNet employs an entire convolutional autoencoder to this task. This does lead to performance gains, but

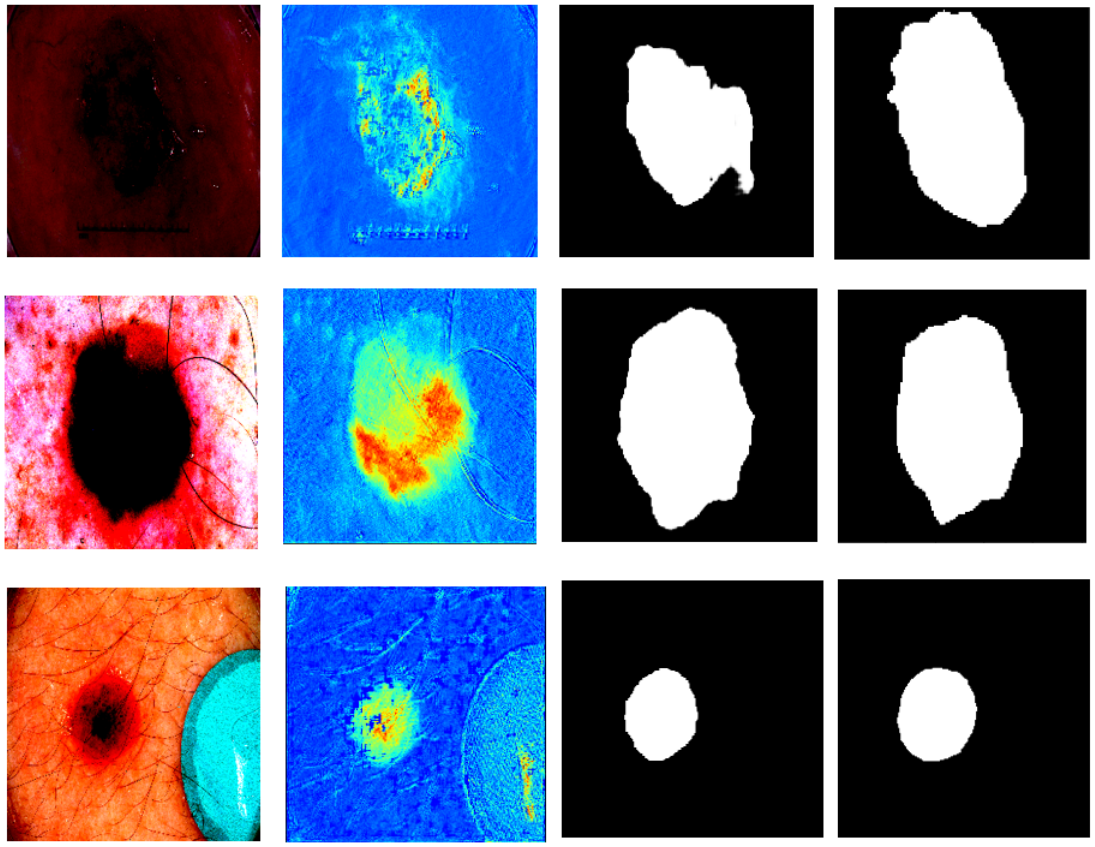


Figure 3.12: The following figure shows the output of the activation maps after the first FocusNet style attention block from Figure 3.2 and the corresponding decoded output.

it also leads to an increased number of parameters in the network. The architecture is robust enough as a little over half the parameters are actually contributing to the segmentation task, and yet it achieves competitive results, but we need to think of a new method to use our attention mechanism with fewer parameters. The visualization of the activation maps following the self attention mechanism after the first residual block in the architecture are shown along with some decoded outputs in Figure 3.12. It can be seen that the attention mechanism works well even with extreme changes in illumination and localizes the cancerous region well. The decoding on the other hand lets the architecture performance down. Hence, we believe a future step for this research is, keeping the same attention mechanism, we need to construct a better decoding scheme to handle such edge cases. Further, we would

like to embody the same attention scheme with fewer parameters.

3.8 Summary

In this chapter, we proposed an attention based fully convolutional network, FocusNet that uses a dual branch encoder-decoder structure to create effective encodings of images into latent representations using a fairly robust attention mechanism. We showed a series of ablation studies demonstrating that the architecture setting and attention mechanism are in-fact placed in a way to exploit maximum performance from this architecture. FocusNet outperforms other methods on the tasks of skin cancer regimentation and whole lung segmentation, but like any other architecture calls for the need of a better loss to handle class imbalance between segmented pixels. To tackle this problem, we propose an adaptive logarithmic loss. We show that the derivative of this loss shows a highly non linear response for small values of the loss, which means it penalizes the loss more for getting a pixel wrong from an undersampled class. FocusNet with the adaptive logarithmic loss show state of the art results across the ISIC 2018, nuclei segmentation and the DRIVE retinal blood vessel segmentation datasets, as well as show better results than the Attention U-Net trained with the focal tversky loss. We then address the shortcomings of our architecture and highlight that though our attention mechanism works well under varying illumination conditions, our constructed decoder needs improvement. Along with this, we pose the question, whether a smarter methodology for attention incorporation is possible keeping the same focusNet style attention mechanism, which we know works well, preferably one that does not use too many parameters.

Chapter 4

FocusNetAlpha

4.1 Introduction

Given that convolutional neural networks extract features via learning convolution kernels, it makes sense to design better kernels which can in turn lead to better feature extraction. Learning better feature extractors is the most important task a network can do, especially for attention based architectures, as the attention mechanisms are learnt over features extracted via convolutions. Hence, the better the feature, the better the output. Recent research has placed a lot of emphasis on optimizing convolutions, and in-turn learning better feature extractors, but until very recently, a very important technique went unnoticed. The Alexnet [53] architecture, due to memory constraints, divided their convolution kernels in each layers on two different GPUs. This resulted in an interesting effect where each kernel on a particular GPU learnt an explicit feature type. For the first convolutional layer, the kernels on one GPU learnt grayscale edge features, while the kernels on the second GPU learnt only colour specific information. This led to the initial empirical results showing the affect of grouping convolutions in convolutional neural networks. Filter groups learn a sparsely correlated set of features in each group, where each group learns a specific distinct feature of the input volume [96]. Even though filter grouping has shown promising improvements in the accuracy obtained by CNNs for various tasks, their use in state-of-the-art approaches is at a very early stage. There are a lot of unanswered questions in research involving filter grouping. We

focus on the two which we consider are the most important - 1) No research to date incorporates attention mechanisms inside filter groups, and 2) Existing research has no grounding work that shows the best way to combine information learnt in each group. This is an important step as filter groups in their most basic form do not interact with each other. In architectures that propose to have certain cross talk between these groups, they define a fixed permutation of the groups rather than learning the best way to combine the information together.

To this end, we propose FocusNetAlpha, a deep learning architecture for medical image segmentation, that harnesses the power of grouped convolutions and combines it with a FocusNet style attention mechanism to get a better performance than FocusNet, with less than half the number of parameters. We enhance the decoding using fine grained information from each decoder scale which helps improve the network's decoding ability. We show that instead of using a single permutation to aid filter group interaction, such as proposed in techniques like ShuffleNet [98] and IGC-Net [97], creating an embedding using a shared weighted function learns to embed the features into a space invariant to all possible permutations of the feature maps. We compare with state of the art architectures, namely, Wide UNet, UNet++, R2U-Net, Attention U-Net, BCDU-Net and FocusNet architectures and outperform them all.

The rest of the chapter is organized as follows. Section 4.2 proposes the overview of our architecture. Section 4.2.1 describes our novel residual group attention block and its working. We the define a hybrid loss for our problem in Section 4.3. Section 4.4 summarizes our experiments which contain the ablation studies (Section 4.4.2) conducted for our architecture. We present our results on three benchmark datasets in Section 4.5. We provide a critique of our architecture in terms of model performance and complexity in Section 4.6. We finally conclude the chapter in Section 4.7.

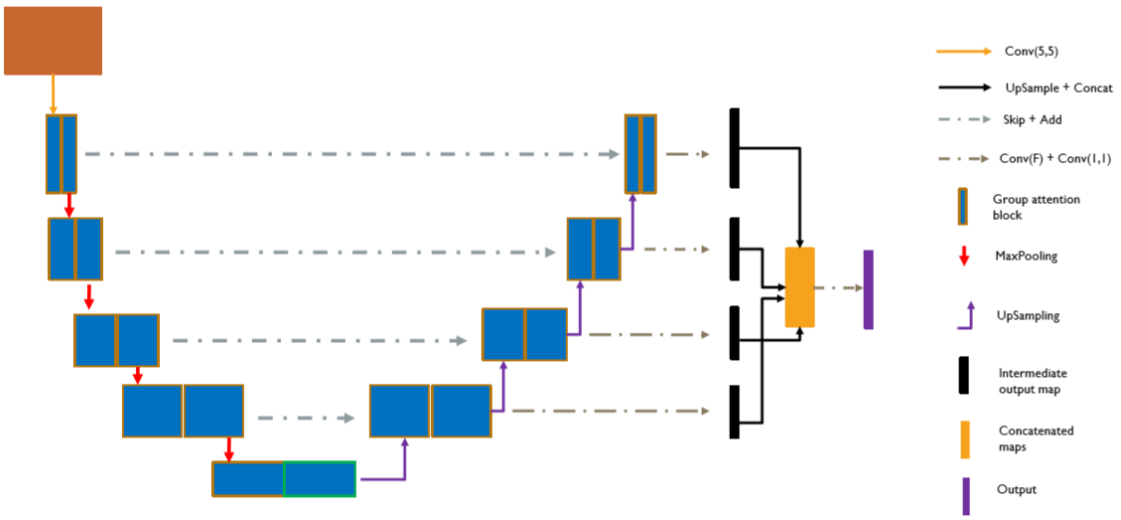


Figure 4.1: The figure shows the architecture diagram for FocusNetAlpha. The input image is processed by a series of residual group attention-max pooling blocks into a bottleneck and then decoded into a segmentation masks.

4.2 Network Architecture

The encoder decoder structure of FocusNetAlpha works like any convolutional autoencoder should. The input is processed by an encoder into a bottleneck, which is then decoded into an output segmentation mask by the decoder. The main building block of our architecture is the residual group attention block (see Section 4.2.1) that employs our novel attention methodology inside group convolutions for effective feature extraction. We address the problem of the relatively inferior decoding ability of FocusNet in FocusNetAlpha. We do this by creating a scheme that combines the output from each decoder scale to the final output which leads to superior performance. The output from each scale passes through a conv-bn-LeakyReLU-conv-sigmoid operation to give intermediate outputs which are upsampled if needed to the output size, and then concatenated together. The concatenated volume is then passed through a conv-bn-LeakyReLU-conv-sigmoid block to get the final output segmentation map. Downsampling in our architecture is done using the max pooling operation. We add skip connections from the encoder to the decoder rather than concatenating them. We upsample via repeating values in a kernel from a lower

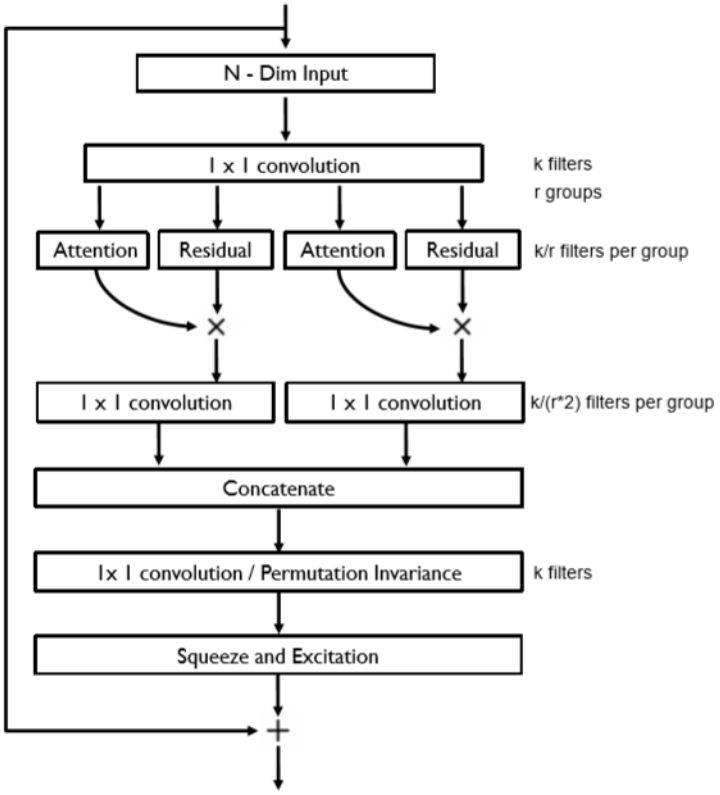


Figure 4.2: Our novel residual block that first employs pixelwise attention inside filter groups, followed by combining the groups via a permutation invariant embedding. The squeeze and excitation block then recalibrates the feature maps which is followed by the residual mapping.

scale into a upsampled scale and letting convolutions learn their correct values. We use dropout in the bottleneck layer with a rate of 0.5. The receptive field of the first convolution kernel is 5x5. Following that, all convolutions kernels have a receptive field of 3x3 when they're used for feature extraction, and 1x1 when they are used to learn attention weights (preceding the sigmoid gating). The architecture can be visualized in Figure 4.1.

4.2.1 Residual group attention block

The residual group attention block is shown in Figure 4.2. The input to the block is a feature volume that is processed by a 1×1 convolution operation. The general

form of the identity mapping is

$$M = \mathbin{\mathsf{C}}_{i=1}^{r} P_i(x)$$

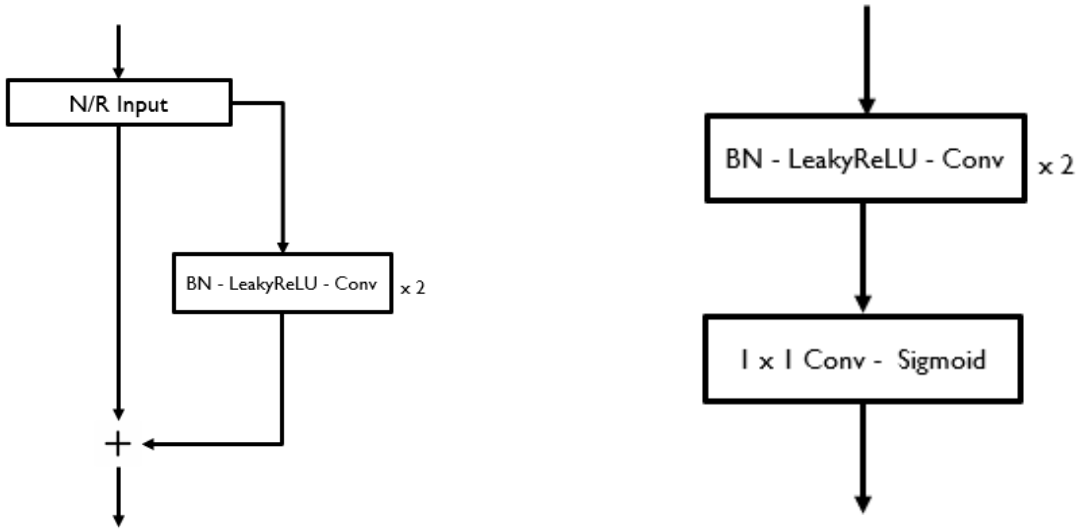
where M is the output of the residual block, C denotes concatenation, and $P_i(x)$ is some transformation learnt by r separate stackings of trainable neurons transforming some input x. Here, $r = 4$, as we divide this input features into groups of 4, to be processed by 4 separate convolution groups. Each group, alternatively, is responsible for learning the attention weights for the group to it's right, and the next group learns the features that need to be extracted. The attention weights for each attention group are obtained via two bn-LeakyReLU-conv operations followed by a conv-sigmoid operation to get the per pixel probabilities. Each attention group transforms its input in the form

$$A_{\mathrm{r}} = \sigma(\mathbf{W}_{\mathrm{a}}, \delta(x_{\mathrm{r}}, \mathbf{W}_{\mathrm{k}}))$$

Here W_k *and* W_a are the convolution weights and the attention weights respectively. x_r is the r^{th} group that is input into this block, and δ denotes the leakyReLU activation. The structure of this block is shown in Figure 4.3b. The residual block contains two bn-LeakyReLU-conv operations followed by a skip connection which adds the features from the previous step to the residual block features. If the residual mapping is given by

$$O_{\mathrm{r}} = x_{\mathrm{r}} + F(x_{\mathrm{r}})$$

then the network learns this $F(x_r)$ using some weights W_k as $F(x_{\mathrm{r}}) = \delta(x_{\mathrm{r}}, \mathbf{W}_{\mathrm{k}})$. The block is shown in Figure 4.3a. The output from the residual block is multiplied pointwise with the output from the attention block as $A = A_{\mathrm{r}} \bigodot O_{\mathrm{r}}$, weighting the pixels with a higher importance more prominently. the $\bigodot$ here denotes the hadamard product. The attention infused output for each group propagates further in the block and is convolved with a convolution block with a 1×1 receptive field and twice the number of filters. These intermediate filter maps are then concatenated together and passed through a final 1×1 convolution. The feature maps are then recalibrated using a squeeze and excitation operation which is followed by a residual connection.



(a) Residual block (b) Attention block

Figure 4.3: Different components of our group attention block.

4.3 Hybrid adaptive logarithmic loss

Accuracy alone is not a good measure of a network's performance. Hence, we try to create a loss in such a way, that we can optimise various metrics at the same time. Importantly, we want a network that can predict true positive as efficiently as possible, and reduce their mis-classifications. This effectively translates to having a high recall. In order to optimize our architecture to have better recall, we adapt the balanced cross entropy loss with the tversky loss in a novel way to create our hybrid loss function. The loss is defined as, $HL = (k)BaCE + (1-k)TL$, where,

$$BaCE = \Omega p \log(\hat{p}) + (1-\Omega)(1-p)\log(1-\hat{p})$$

$TL = \sum_c (1 - TI)$ where the subscript indicates a summation over the number of classes c and,

$$TI = \frac{|G \cap P|}{|G \cap P| + \alpha|P \backslash G| + \beta|G \backslash P|}$$

To create a higher emphasis on the true positives, we select $\Omega = 0.7$. Generally, $\alpha = 0.3,\ \beta = 0.7$ works as the most optimal setting in TL, adding higher weights to optimize over false positives and false negatives, so we stick with those hyperparameter values. We weight the influence of both losses equally by setting k=0.5. In

order to optimize over a loss, we use a function whose derivative gives a non linear response closer to the global minimum. Hence, to mitigate the problem of pixel class imbalance and poor convergence to the minimum, we use the adaptive logarithmic loss for our problem. The loss is defined as,

$$ALL - HL(x) = \begin{cases} \omega \ln(1 + \frac{|HL|}{\epsilon}) & |HL| < \gamma \\ |HL| - C & \texttt{otherwise} \end{cases} \tag{4.1}$$

where $C = \gamma - \omega \ln(1 + (\frac{\gamma}{\epsilon})$. We observe that the default hyperparameters of this loss work as the optimal ones for our experiments. Hence, we set $\gamma = 0.1$, $\omega = 10.0$ *and* $\epsilon = 0.5$.

4.4 Experiments

In this section, we conduct a series of ablation tests to test the effect of each component on our architecture. We also compare against the various resnet variants to show that our residual block results in superior feature extraction compared to the different residual blocks that have been previously proposed.

4.4.1 Hybrid adaptive logarithmic loss vs hybrid loss

We compare the Adaptive logarithmic loss with our hybrid function formulation (ALL-HL), with the vanilla hybrid loss (HL). We run two experiments to this end the results for which are shown in Table 4.1. The ISIC 2017 dataset is used for this experiment which is described in Section 4.4.2. In the result, particular interest to us is the value of the recall that we are effectively optimizing over by weighting our true positives higher. Our loss improves the recall by 1.85% which in turn gives a higher dice index. We use the ALL-HL for all our experiments henceforth.

4.4.2 Ablation study

Our ablation studies test the influence of the different components of the blocks in our architecture. We remove certain components without replacement and check how it affects network performance. The component is then added back, and another

Method	Precision	Recall	Accuracy	DI	JI	F1
FocusNetAlpha (HL)	0.7921	0.8037	0.0.9276	0.8214	0.7631	0.8262
FocusNetAlpha (ALL-HL)	0.8002	0.8222	0.9349	0.8404	0.7817	0.8336

Table 4.1: ALL-HL gives better results compared to HL. Dataset used is ISIC 2017.

is removed to see the contribution of the other block on the network performance. All experiments in our ablation experiments are conducted on the ISIC 2017 skin cancer segmentation dataset. We use the same preprocessing, train-val-test split and augmentation strategy as FocusNet in order to have a valid comparison with the architecture.

We start with comparing the performance of our decoder with a standard decoder, that just upsamples the bottleneck without any multiscale refinement (FocusNetAlpha - multiscale decoder), in the style of the U-Net decoder, but with our group attention block. The rest of the blocks in the architecture remain the same. We then dive deeper into our group attention block where we replace all filter groups with a residual attention block, computing features, and attention weights inside the same block (FocusNetAlpha + ResAttention). The structure of this block is the same as Figure 4.3a. The difference is that the feature extraction is followed by a conv-sigmoid block and multiplied with the skip connection. The same skip connection is also added to the attention scaled value in parallel. Our final study deals with concatenating the features of a group depthwise with the features of the group to its left (FocusNetAlpha + ConcatHorizontal). This means using only residual blocks inside groups which are concatenated with the groups to the right, instead of multiplying as in our proposed block.

Our experiments show that the multiscale decoding helps improve performance of the architecture. Combining feature extraction with attention in a single block has a degrading effect on the performance which we believe happens due to the lack of features extracted in the blocks. Horizontal concatenation gives comparable

performance with our methodology, though it does not improve the recall sufficiently to be looked into further.

Method	Recall	DI	JI
FocusNetAlpha - multiscale decoder	0.8111	0.8357	0.7712
FocusNetAlpha + ResAttention	0.7936	0.8167	0.7591
FocusNetAlpha + ConcatHorizontal	0.8118	0.8352	0.7741
FocusnetAlpha	0.8222	0.8404	0.7817

Table 4.2: Ablation studies on FocusNetAlpha using the ISIC 2017 dataset.

4.4.3 Permutation invariance vs channel shuffle

Combining information from the different filter groups into one feature is an active area of research and many techniques have been proposed in the past that fix permutation matrices to aid channel interaction. The most prominent of these techniques is ShuffleNet that proposes a fixed permutation across the different channels so that a different combinations of channels learnt in different filter groups is propagated forward. We compare our methodology with ShuffleNet as it is the current state of the art in combining filter group information. A 1D convolution operation shares weights across it's kernels, hence mapping the different filter groups on to a symmetric function. This embedding created is invariant to all permutation of the filter groups. Hence, we hypothesise that a permutation invariant embedding such as that created by a 1D convolution, will always outperform hand crafting permutations via a channel shuffle. This concept has found great success in creating permutation invariant representations of 3D data in the recent past [68] and is the technique incorporated in resneXt combine filter groups. We run two different networks to this end, where one is FocusNetAlpha, and the other is the same architecture, but with the 1×1 convolution before the SE block changed to the shuffle block from ShuffleNet (FocusNetAlpha + channel shuffle (CS)). The results are shown in Table 4.3. It can be seen that defining fixed permutations considerably decreases the

performance of the network versus learning a permutation invariant mapping. The jaccard and dice index degrade considerably, and the recall is also affected.

Method	Precision	Recall	Accuracy	DI	JI	F1
FocusNetAlpha + CS	0.7844	0.7991	0.9214	0.8169	0.7612	0.8184
FocusNetAlpha	0.8002	0.8222	0.9349	0.8404	0.7817	0.8336

Table 4.3: Permutation invariance vs channel shhuffle in FocusNetAlpha on the ISIC 2017 dataset.

4.4.4 Comparison with resnet variants

Our next series of experiments observe the influence of residual blocks on our network architecture. We train 5 different architectures to this end. The first is the generic identity mapping [34] proposed in the initial resnet paper. This is followed by the no bottleneck full preactivation mapping [35] . We then train two architectures using the ResNext group convolution blocks and ResneXt + Squeeze and excitation group convolution blocks. All residual blocks listed are placed in the FocusNetAlpha architecture and the results are observed. The results are summarized in Table 4.4. We observe that for such shallow encoding decoding schemes with just a few layers, there is only a marginal increase in performance from the basic residual block to the identity mapping block. ResneXt group convolution block has a marginally better recall than when combined with squeeze and excitation. We believe it to be due to the global attention mechanism squeeze and excitation incorporates, suppressing entire channels of information that. Pointwise attention combined with squeeze and excitation gives a superior recall and fixes this problem, which is what FocusNetAlpha does. FocusNetAlpha achieves a significantly better dice score as well as F1 score compared to other blocks which leads us to believe that it handles the true positives and false negatives better than the other techniques.

Method	Recall	Accuracy	DI	JI	F1
Basic residual block	0.7519	0.9176	0.8023	0.7261	0.8009
Identiy mapping block	0.7576	0.9186	0.8034	0.7281	0.8012
ResneXt group convolution block	0.8145	0.9276	0.8214	0.7689	0.8196
ResneXt + SE	0.8123	0.9311	0.8206	0.7761	0.8229
FocusnetAlpha	0.8222	0.9349	0.8404	0.7817	0.8336
Focusnet	0.7673	0.9214	0.8315	0.7562	0.8191

Table 4.4: Comparing with different residual blocks on ISIC 2017 dataset.

4.5 Results

To show the superior performance of FocusNetAlpha, we compare how it fairs on 3 different benchmark datasets against state of the art network architectures. For all the listed experiments, the train-val-test split is constant and no data augmentation is used. As a preprocessing step, we scale all pixel values to fall between [0,1]. Our initial experiments suggest that mean pixel subtraction worsens the performance of the networks, so we choose not to use it as a preprocessing step. We convert the segmentation mask to binary by setting every pixel above the threshold of 0.5 to 1.

Training

All our experiments are trained with the adaptive logarithmic loss using our hybrid loss (ALL-HL) strategy. The experiments are conducted in keras using a tensorflow backend. The batch size for all experiments is kept constant at 8. 1 Nvidia GTX 1080Ti is used to train the FCN, U-Net, Wide U-Net, Attention U-Net, UNet++, R2U-Net and FocusNetAlpha architectures. BCDU-Net and FocusNet are trained on 2 Nvidia GTX 1080Tis.

We have optimized every architecture trained for these experiments by first running them with different learning rates to see their behaviour, and then creating a customized learning rate schedule for each experiment, for each architecture. All

architectures were trained for a maximum of 50 epochs and the best model weights were saved by monitoring the validation loss. No early stopping was used.

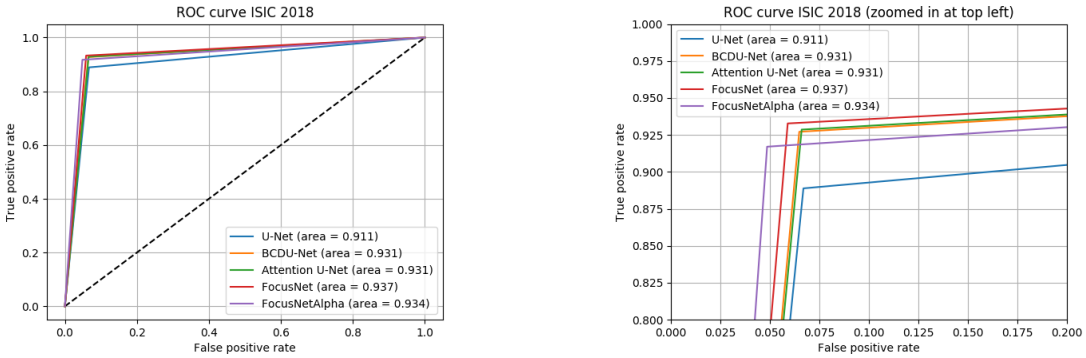
4.5.1 ISIC 2018 segmentation

The ISIC 2018 skin cancer segmentation has become a major benchmark dataset for the evaluation of medical imaging algorithms. We use the 2594 images with corresponding ground truths localizing lesions on skin images containing melanoma. We divide these images into a training set of 1815 images, a validation set of 259 images, and the rest of the 520 images. The images are preprocessed as described earlier. The images in the dataset are 700×900 in dimension. We resize every images to a smaller 256×256 size, via an antialising downsampling technique, to be processed by the networks.

Table 4.5 summarizes our results for the experiments. FocusNetAlpha outperforms every architecture across all metrics significantly for the dataset with considerably fewer parameters and FLOPs (see Section 4.6). The ROC curves for the architectures are shown in Figure 4.4. We defined an accuracy threshold and sampled images higher and lower than the threshold to visualize our predictions and qualitatively evaluate the performance of our network. Figure 4.6 shows the results which FocusNetAlpha found easy to segment. Figure 4.7 shows the more challenging results. Even on the more challenging set of images, it can be seen that FocusNetAlpha (column 5) performs significantly better than Attention U-Net (column 3) and FocusNet (column 4). It is useful to note that one of the problems with FocusNet was its decoding ability which FocusNetAlpha aims to fix. It can be seen that, our improvement performs better than FocusNet to this end. The validation dice index, validation jaccard index and validation loss plots are shown in Figure 4.5.

4.5.2 Cell nuclei segmentation

We now shift our focus to segmentation of smaller regions inside images. For this, we use the cell nuclei segmentation dataset which was a part of the Data Science



(a) The ROC curves for different architectures on ISIC 2018 dataset.

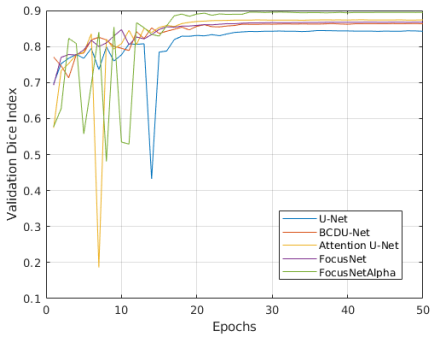
(b) The same ROC curves zoomed in top left for better viewing.

Figure 4.4: Receiver operator characteristics (ROC) curves for the ISIC 2018 dataset.

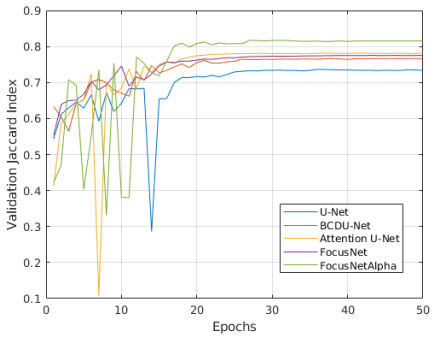
Bowl 2018. It contains 670 images which we divided into a training set of 540 and a validation set of 130. We resize all images to 256×256 using the same anti aliasing technique as we used for the ISIC 2018 dataset. For this task, we test the performance of our architecture by reducing the number of parameters (by reducing the number of filters per layer) for it in a way that it has less than 1 million FLOPs. We call this lightweight architecture FocusNet-α-Lite. We also reduced the number of parameters for the other architectures to account for the smaller size of this dataset so that we don't overfit on it. Our results are summarized in Table 4.6. We get results competitive with BCDU-Net even though we use lesser parameters and FLOPs than the architecture. It should be noted that without the constraint we set on our model to be under 1 million FLOPs, we outperform all existing architectures with a slightly 'bigger' version of FocusNet-α-Lite which contains 2.32 million parameters and 1.2 million FLOPs. Figures 4.11 and 4.10 show the qualitative results, validating FocusNetAlpha's competitive performance compared to BCDU-Net and FocusNet, and superior performance compared to Attention U-Net. The validation dice index, validation jaccard index and validation loss plots are shown in Figure 4.9.

4.5.3 Retinal blood vessel segmentation

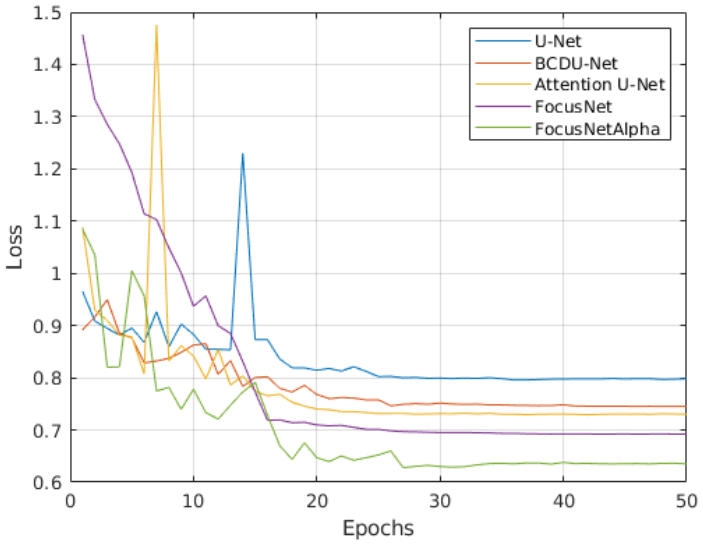
For the third task, we see the effect of our architecture on fine grained pixel segmentation. We use the DRIVE retinal vessel segmentation dataset for this task.



(a) Validation Dice Index vs Epochs on ISIC 2018.



(b) Validation Jaccard Index vs Epochs on ISIC 2018.



(c) Validation Loss (ALL-HL) vs Epochs on ISIC 2018.

Figure 4.5: Visualizing the different validation curves on ISIC 2018.

The dataset is small and contains 40 high resolution images. But given the fine grained analysis needed for it, we can sample smaller patches from the images to be processed by the networks. We sample 200000 patches around the field of view of the fundus images, from which we use 170000 for training and 30000 for validation. The extracted patches are of size 64×64. Mean colour subtraction is applied to the images as initial experiments showed that it boosts performance for this task. Table 4.7 summarizes our results. The DRIVE dataset is fairly saturated in terms of accuracy and ROC-AUC as it's metrics. Hence, we compare the results based on the recall, F1 score and the jaccard index. FocusNetAlpha outperforms every

Method	Precision	Recall	Accuracy	DI	JI	F1
FCN [79]	0.7176	0.8966	0.9011	0.7861	0.7013	0.7832
U-Net [73]	0.7398	0.9043	0.9187	0.8167	0.7268	0.8004
Wide UNet [101]	0.7439	0.9167	0.9234	0.8224	0.7334	0.8039
R2U-Net [3]	0.7381	0.9122	0.9172	0.8271	0.7511	0.8097
BCU-Net [6]	0.7576	0.9272	0.9337	0.8637	0.7665	0.8138
UNet++ [101]	0.7516	0.8889	0.9249	0.8437	0.7435	0.8145
Attention U-Net [63]	0.7526	0.9286	0.9330	0.8741	0.7813	0.8214
FocusNet	0.7805	0.9328	0.9395	0.8676	0.7751	0.8499
FocusNetAlpha	**0.8322**	**0.9471**	**0.9447**	**0.9014**	**0.8271**	**0.8717**

Table 4.5: Segmentation results on ISIC 2018 dataset. The results in bold are the best results obtained on the dataset. FocusNetALpha outperforms every architecture with fewer parameters and FLOPs.

architecture across all three metrics significantly. The results are on a dataset of size 30000, which shows that our architecture is significantly better across segmenting true positives and true negatives correctly than the rest of the architectures.

4.6 Model efficiency

We compare the trade off between performance and efficiency for our model in this section. We focus on the ISIC 2018 dataset for FocusNetAlpha (see Table 4.8), and on the nuclei segmentation dataset for FocusNet-α-Lite (see Table 4.9). It can be seen that FocusNetAlpha outperforms every architecture in terms of having the lowest number of parameters and floating point operations (multiplications, additions, subtractions etc). In terms of benchmark evaluation metrics, we observe a considerable increase in performance compared to the state of the art. FocusNet-α-Lite gets extremely competitive results with an architecture that has almost 2.5 times more parameters than it's nearest competitor (BCDU-Net), which it achieves with over 10 times less FLOPs. The only real concern is the amount of time our architecture

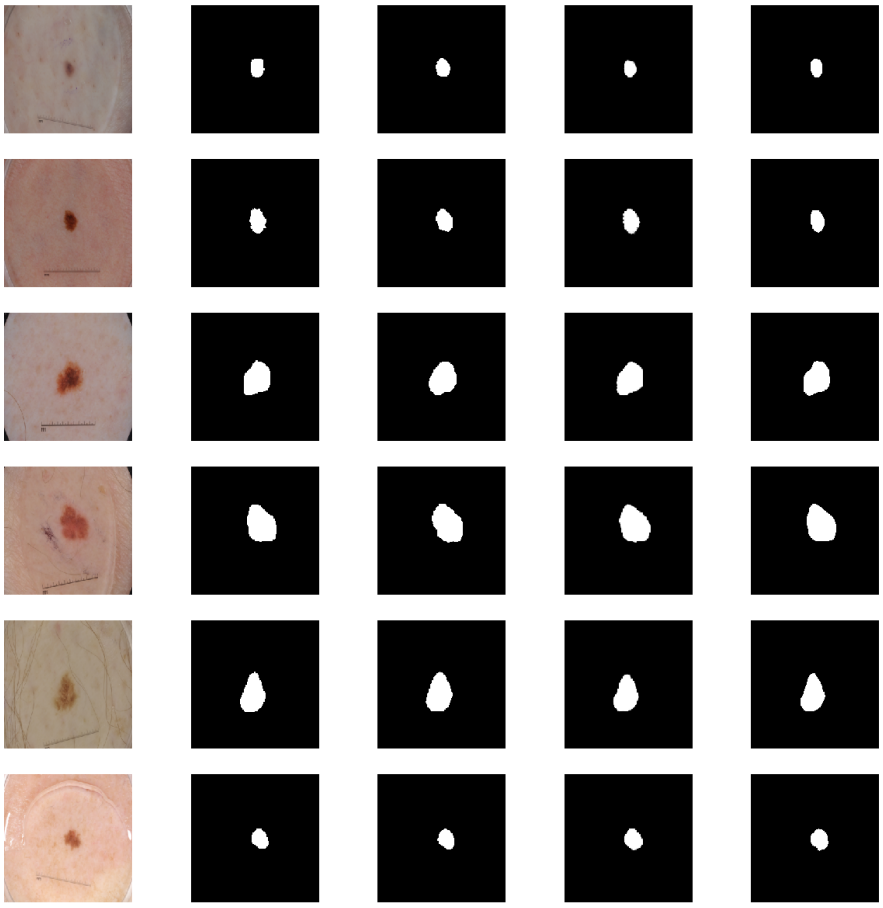


Figure 4.6: ISIC 2018 good segmentation results. From left, column 1 is the input image, column 2 is the ground truth and columns 3, 4 and 5 are the outputs of Attention U-Net, FocusNet and FocusNetAlpha respectively.

takes for a forward pass during inference. It is the slowest compared to the other architectures. But this is effectively just a technology bottleneck. Convolution operations on GPUs run on a CUDNN backbone which work on a principle of model and data parallelism. Current technology implements data parallelism efficiently into GPUs due to which deep learning is now a reality. But to date, open source CUDA C/C++ kernels do not exist that can speed up group convolutions across any existing deep learning framework as group convolutions are till a very young and active area of research. Frameworks such as PyTorch have tried speeding up the inference time of group convolutions, so we can expect faster inference times in that

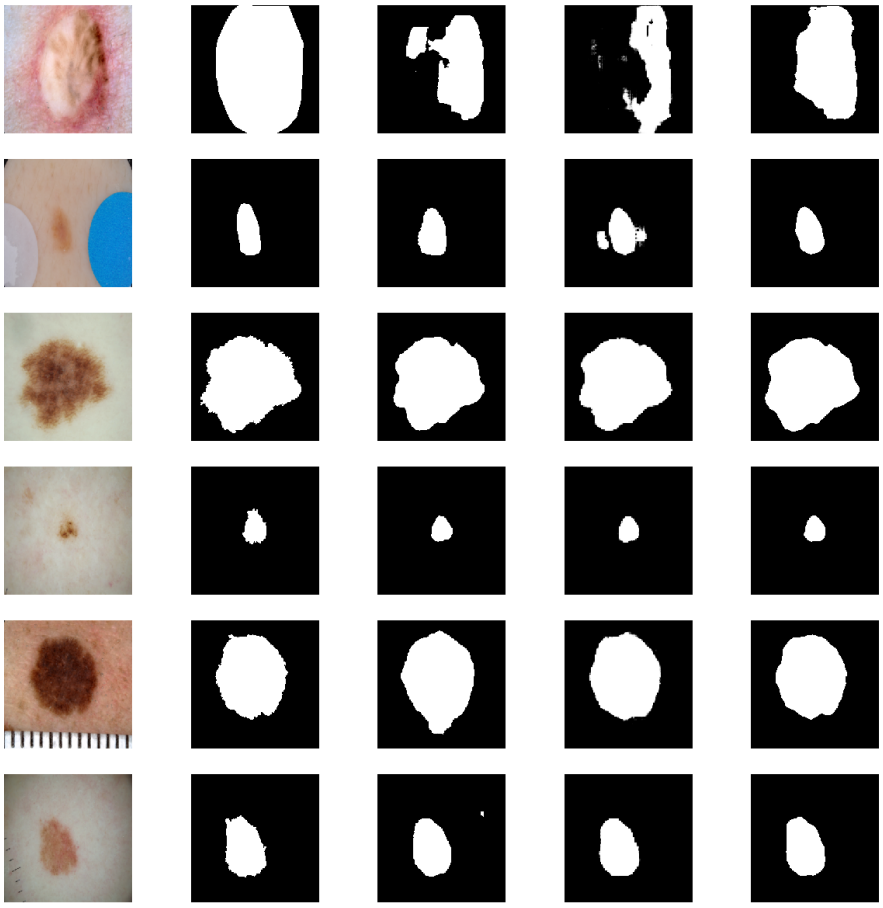
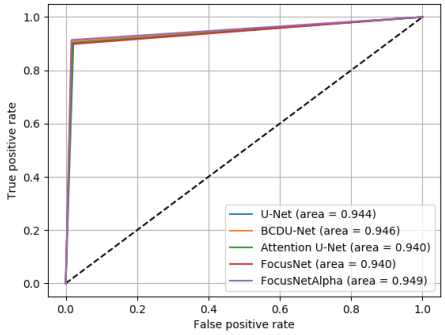
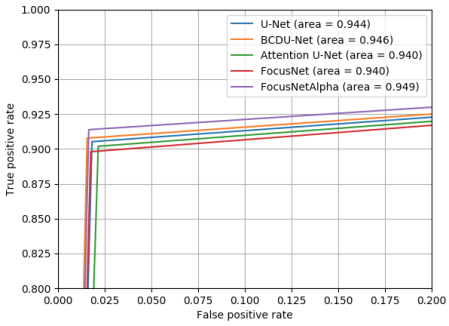


Figure 4.7: ISIC 2018 bad segmentation results. From left, column 1 is the input image, column 2 is the ground truth and columns 3, 4 and 5 are the outputs of Attention U-Net, FocusNet and FocusNetAlpha respectively. FocusNetAlpha provides better segmentation results even under challenging scenarios.

framework, but it still remains slow compared to generic convolutions which have highly efficient CUDNN backends. Due to this reason, our forward pass inference is slow. But as the technology catches up with the research, we expect the inference time for our model to be lower too.

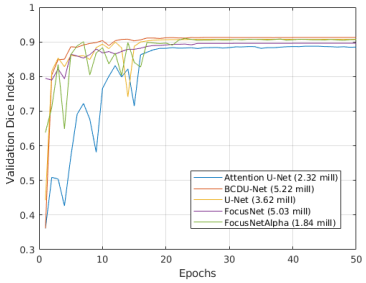


(a) Cell nuclei segmentation ROC curves.

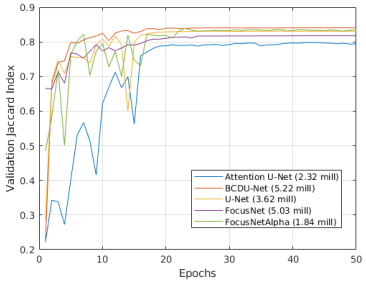


(b) Same ROC curves zoomed in top left.

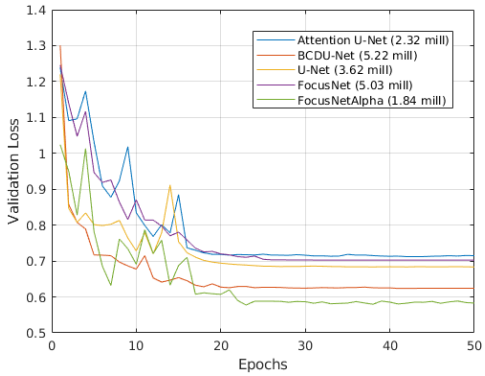
Figure 4.8: Receiver operator characteristics (ROC) curves for the cell nuclei segmentation dataset.



(a) Validation Dice Index vs Epochs on the cell nuclei segmentation dataset.



(b) Validation Jaccard Index vs Epochs on the cell nuclei segmentation dataset.



(c) Validation Loss (ALL-HL) vs Epochs on the cell nuclei segmentation dataset.

Figure 4.9: Visualizing the different validation curves on the cell nuclei segmentation dataset.

Method	Precision	Recall	Accuracy	F1	DI	JI
U-Net [73]	0.8976	0.9052	0.9604	0.8994	0.9075	0.8310
BCU-Net [6]	0.9024	0.9078	0.9728	0.9101	**0.9129**	**0.8410**
Attention U-Net [63]	0.8782	0.9019	0.9672	0.8899	0.8876	0.7984
FocusNet	0.9016	0.8981	0.9697	0.8998	0.8961	0.8176
FocusNet-α-Lite	**0.9173**	**0.9139**	**0.9768**	**0.9106**	0.9077	0.8386

Table 4.6: Segmentation results on the data science bowl 2018 dataset. The results in bold are the best results obtained on the dataset.

Method	Recall	F1	JI
U-Net [73]	0.7614	0.7898	0.7219
BCDU-Net [6]	0.7991	0.8165	0.7503
Attention U-Net [63]	0.7861	0.8033	0.7517
FocusNet	0.7892	0.8097	0.7498
FocusNetAlpha	**0.8288**	**0.8306**	**0.7854**

Table 4.7: Segmentation results on the DRIVE retinal blood vessel segmentation dataset.

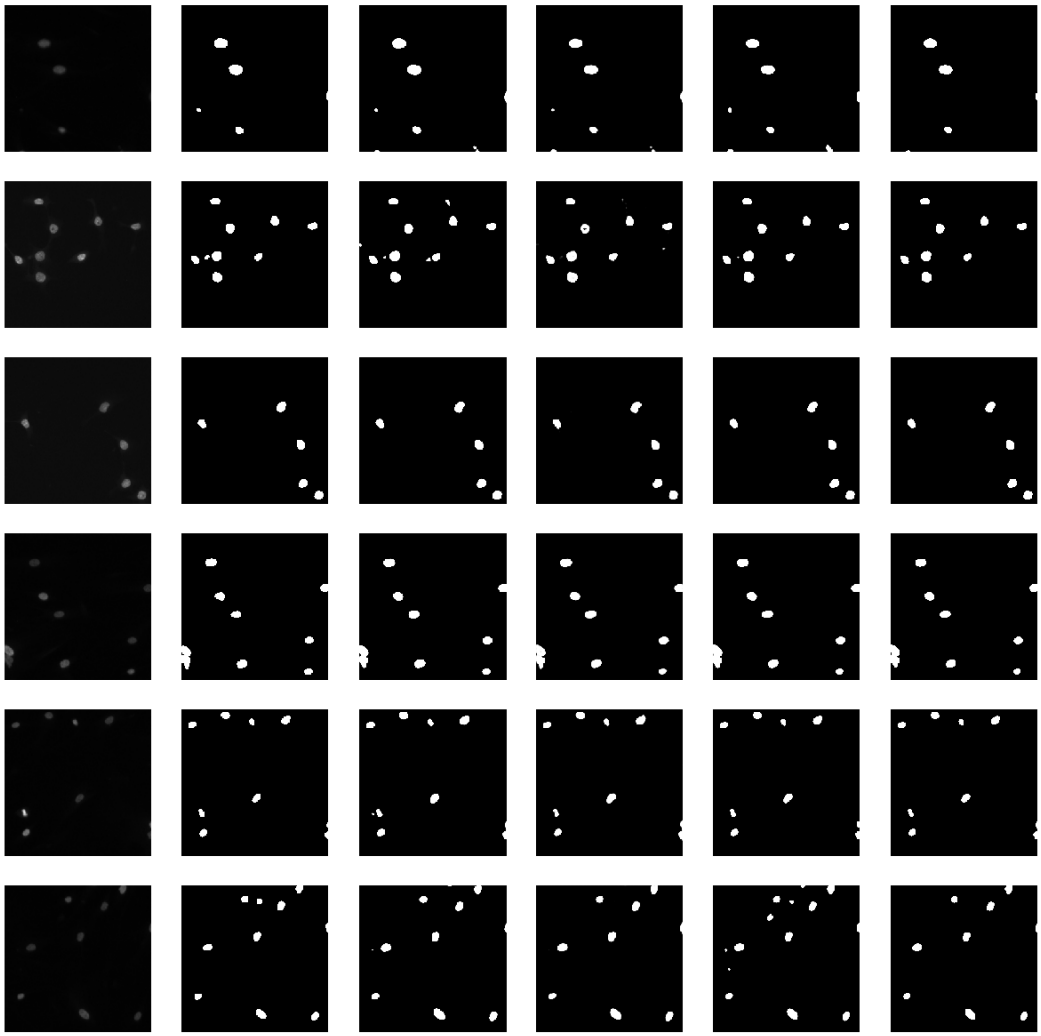


Figure 4.10: Cell nuclei segmentation good segmentation results. From left, column 1 is the input image, column 2 is the ground truth and columns 3, 4, 5 and 6 are the outputs of BCDU-Net, Attention U-Net, FocusNet and FocusNetAlpha respectively.

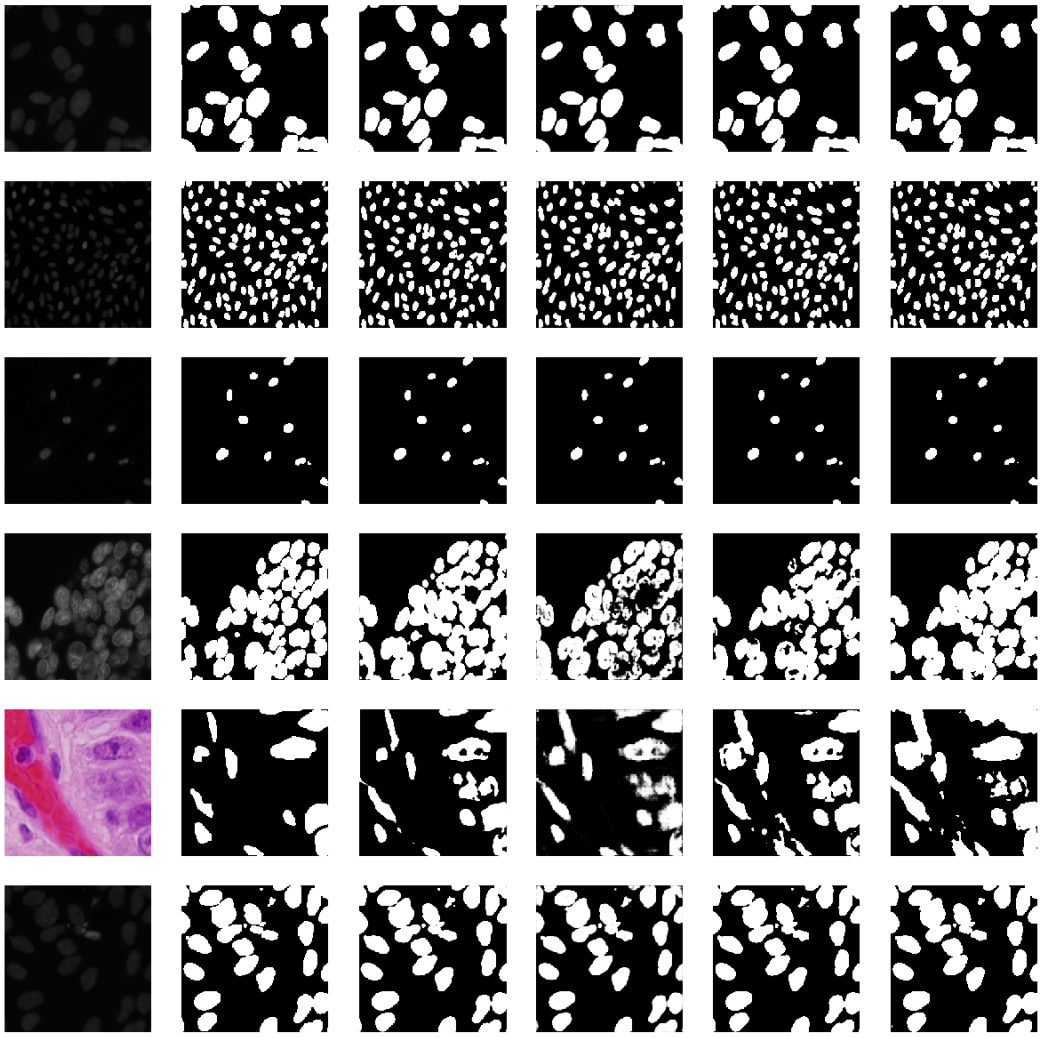


Figure 4.11: Cell nuclei segmentation bad segmentation results. From left, column 1 is the input image, column 2 is the ground truth and columns 3, 4, 5 and 6 are the outputs of BCDU-Net, Attention U-Net, FocusNet and FocusNetAlpha respectively.

Architecture	Params ($\times 10^8$)	FLOPs ($\times 10^8$)	Time (ms/step)	F1	JI
UNet [73]	7.94	16.12	8	0.8004	0.7268
UNet++ [101]	9.04	42.44	21	0.8145	0.7435
BCDU-Net [6]	20.66	39.76	30	0.8138	0.7665
Attention U-Net [63]	8.91	17.82	9	0.8214	0.7813
FocusNet [45]	19.07	91.36	29	0.8499	0.7751
FocusNetAlpha	7.80	15.64	41	0.8717	0.8271
FocusNet-α-Lite	2.16	1.34	32	0.8167	0.7844

Table 4.8: Comparing the model complexity and performance (on ISIC 2018) for FocusNetAlpha against state of the art segmentation architectures. FocusNet-α-Lite results also added for reference.

Architecture	Params ($\times 10^8$)	FLOPs ($\times 10^8$)	Time (ms/step)	Precision	F1
UNet [73]	3.62	1.89	3	0.8976	0.8994
BCDU-Net [6]	5.22	9.98	18	0.9024	0.9101
Attention U-Net [63]	2.32	1.84	5	0.8782	0.8899
FocusNet [45]	5.03	22.38	16	0.9016	0.8998
FocusNet-α-Lite	1.84	0.98	29	0.9173	0.9106

Table 4.9: Comparing the model complexity and performance (on cell nuclei segmentation) for FocusNetAlpha against state of the art segmentation architectures.

4.7 Summary

In this chapter we presented FocusNetAlpha which builds on the pros of FocusNet and successfully eliminates it's cons. We highlighted the main drawbacks of FocusNet to be it's large number of parameters, and inadequate decoding. To eliminate these faults, we combined the FocusNet style attention mechanism with group convolutions which gets rid of the entire autoencoder branch in FocusNet that was required to extract pixelwise attention maps. We proposed a novel residual group attention block that forms the main building block of FocusNetAlpha and showed that 1x1 convolutions in these blocks are actually functions learning a permutation invariant mapping of the groups in a feature space that is of dimensions equal to the number of filters of the 1x1 convolution kernel. We compared the results of our methodology with state of the art deep learning architectures proposed for medical image segmentation and comprehensively beat the state of the art across every benchmark metric. We then shifted our attention towards lightweight segmentation, for which we proposed FocusNet-α-Lite, which gets competitive performance with the state of the art with 2.5 times less parameters and 10 times less FLOPs. We show that our residual group attention block performs better than the benchmark as well as the latest major enhancements to residual learning that have been proposed to date and can be easily used in existing deep learning architectures to boost their performance.

Chapter 5

SAWNet

5.1 Introduction

Even though deep neural networks have established themselves as the state of the art methodology in almost all computer vision tasks to date, their application to processing data lying on non-euclidean domains is still a very active area of research. One such area is the analysis of 3D point cloud data. This task poses a special challenge for neural networks due to the lack of any specific order in which point clouds can be arranged in. Many recent techniques have been proposed, spearheaded by the PointNet [68] architecture as a potential solution to the problem of point cloud analysis. PointNet maps the 3D points on to a symmetric function, thus creating an order invariant representation of the points in the feature space. To date, proposed techniques use either global or local information from the point clouds to extract a latent representation for the points, which is then used for the task at hand (classification/segmentation). In this chapter, we propose a series of improvements to the task of point cloud analysis using deep learning. We start by introducing a novel neural network layer that combines both global and local features for every point to in turn, produce better embeddings of these points. We observe that residual feature reuse in this setting propagates information more effectively between the layers, and also makes the network easier to train. We employ an attention mechanism to propagate the most important features in every embedding forward which in turn learns the best features from both representations. We then shift our

attention to encoding global features for this task. To date, a simple max pooling operation has been used to achieve this global feature vector. We propose a novel fully differentiable attention block to create a superior global feature aggregation that outperforms max pooling. We combine our contributions into one architecture - SAWNet, our spatially aware deep neural network, that achieves state-of-the-art results on the tasks of point cloud classification (ModelNet40 dataset), as well as point cloud segmentation (ShapeNet part dataset).

The rest of this chapter is organized as follows. In Section 5.2, we establish some background that motivates or methodology. Section 5.3 briefly describes the SAWNet architecture, the components of which are then explained in detail in Sections 5.3.1, 5.3.2 and 5.3.3. We follow that by an extensive evaluation of SAWNet and its components in Section 5.4 where we discuss the architectures performance on point cloud classification (Section 5.4.1), and segmentation tasks (Section 5.4.5), while also performing further experiments as well as ablation studies (Section 5.4.4). We finally present our conclusions in Section 5.5.

5.2 Motivation

Given a set of points in 3D space, each point has a value, which is an x, y, z coordinate value encoded as a distance from some set origin. A PointNet style approach, encodes the x, y, z coordinates of these points into a high dimensional feature space using a 1D convolution operation. A shared weight approach of this kind is the equivalent of creating a symmetric function that learns all possible permutations of the data structure, creating a permutation invariant embedding of the data. This approach is robust to random point arrangement, and outperforms dynamic embedding approaches inspired by variants of LSTM based architectures. It also works better than sorting the points before inputting them to a fully connected layer. Sorting is not an efficient way to create the data representation, as sorting higher order encodings increases the computation time exponentially. PointNet introduced a field of architecture learning where unordered data could be parsed efficiently, but

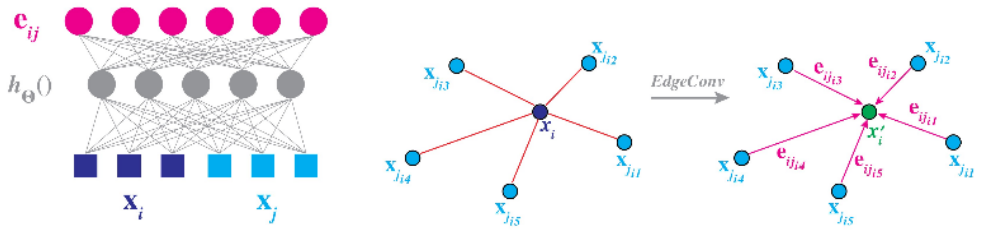


Figure 5.1: Point encoding via edgeconv in Dynamic Graph CNNs [88]. Given a set number of nearest neighbours, the dependency of each point on its neighbour is encoded by the edgeconv operation.

their approach to this technique has an inherent drawback. PointNet encodes the global location of points based on an absolute coordinate system. This means that every point encoding is created without any local shape information, but only using the location of the point with respect to a set origin. Such an approach means that PointNet fails to parse scenes and objects which are even slightly occluded as the global encodings are extremely susceptible to such conditions.

In order to overcome the drawbacks of PointNet, many architectures have been proposed using the principle of learning a symmetric point set encoding [93] [70], but with a well defined neighbourhood. Such an approach encodes a point in terms of its local coordinate system, making it a more robust encoding of order invariant data. One such technique computes nearest neighbours of a particular and learns how the neighbours influence the point's position with respect to them. This approach is presented in dynamic graph CNNs [88]. Dynamic graph CNNs use edgeconv operations (Figure 5.1) to learn how a point's neighbours influence it. The approach computes 20 nearest neighbours of the points based on distance, and uses a 1D convolution operation to learn an encoding of the point of interest with respect to these points. The effectiveness of this technique is based on backpropagation learning this encoding to be robust enough, encapsulating local geometric information regarding to the data.

Thus, given a global point encoding, and a local point encoding, our central

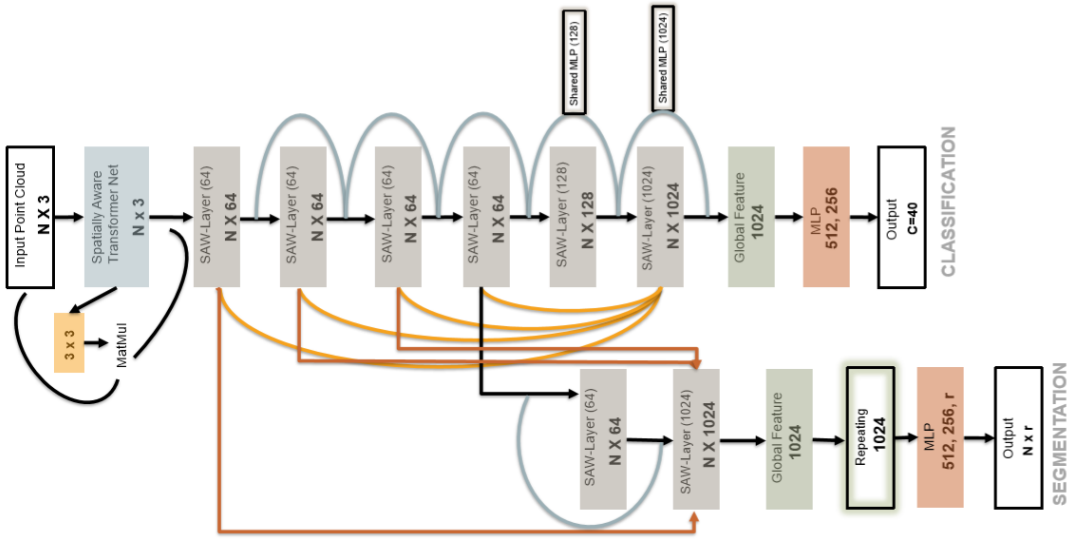


Figure 5.2: Architecture for the classification and segmentation tasks. Our networks contain a spatially-aware input transformer that uses SAW-layers without the feature attention module to regress a 3x3 transformation matrix to align the point clouds. Residual connections are used throughout to learn the point embeddings. A final global information aggregation is calculated via our novel Global Feature Aggregation Unit. The feature is then fed into a 3 layer MLP for classification. For segmentation, a $N \times r$ matrix is predicted, which is a point-wise prediction. The flow of information is from left-to-right. The navy blue arrows denote residual connections. The maroon and orange arrows denote concatenation. Black arrows denote sequential information flow. The weight shared MLPs are used with the residual connections to match the embedding dimensions for adding the identity mapping.

principle is to investigate whether backpropagation can learn an encoding that is superior to the two individual encodings. As the encodings at each step, have a global and local information, we refer to it as being spatially aware of its global information (via the global embedding), as well as local information (via the local embedding). Once we verify the effectiveness of such an embedding, we enhance it using an attention mechanism.

5.3 Network Architecture

In this section, we discuss SAWNet (Figure 5.2), our architecture that introduces the concept of spatial awareness in deep neural network based point cloud processing. The architecture takes a $N \times 3$ point cloud as input and outputs a class label or a per point semantic segmentation label for it. The input is fed into a transformer net (Section 5.3.1) that uses SAW-Layers (without attention) to regress a 3×3 transformation matrix for point cloud alignment. The transformation matrix is multiplied by the points to align them. This step is crucial as real world data is hardly ever in a normalized pose and usually requires some initial alignment. Due to this reason, we use the unaligned version of ModelNet40 to test the classification ability of our model rather than the aligned version. The aligned points are then fed into a series of SAW-Layers (5.3.2) to create a permutation invariant embedding of the points. We use residual connections to transfer information between layers. This allows us to train deeper and more stable network architectures. For layers where the embedding size increases, the residual mappings account for that with a weight shared MLP (analogous to a 1D convolution) before the addition of the previous embedding with the next. We concatenate the output of every previous SAW-Layer before passing it into the final 1024-D SAW-Layer to get a better global contextual symmetry aggregation. The output of the final 1024-D embedding is then aggregated using our novel SAWNet global aggregation unit (5.7), to get a single 1024 dimensional vector, which contains the global contextual information for the point clouds. This information is used to classify or segment the points using a generic multi layer perceptron. For the segmentation tasks, the output of the MLP is reshaped to get a per point semantic segmentation.

5.3.1 Spatially Aware Transformer Net

The spatially aware transformer network contains 3 SAW-Layers (without attention) which embed the point cloud into a 64, 128 and a 1024 dimensional space respectively. The output embedding from the final layer is aggregated into a global context (pooled) feature vector that is fed into a multi-layer perceptron with two hidden lay-

ers of size 512 and 256. The output is a final dense layer of size 9 that is reshaped into a 3×3 transformation matrix, the elements of which are the learnt cropping, rotation, scaling and skew transformation values for the point cloud. The matrix is initialized as a 3×3 identity matrix so as to reproduce the input points at the output when not trained. The parameters are then refined via end to end training to learn the relevant transformations. For a given point cloud, the transformation is denoted as

$$I^t = M_\theta \times I = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \\ \theta_7 & \theta_8 & \theta_9 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Here, M_θ is the transformation matrix with parameters θ_i. A simple matrix multiplication of this matrix with the input point cloud, aligns the points in 3D space. The architecture can be seen as a mini version of the SAW-Net architecture. We tried different experiments to see the effectiveness of our transformer on our network output (see Section 5.4.4). SAWNet without any alignment of input point clouds on the ModelNet40 dataset performs inferior to even the most basic transformation matrix that 1D convolutions can learn.

5.3.2 Spatially AWare-Layers

The building blocks of a single SAW-Layer can be seen in Figure 5.3. The figure also shows how the information in the layer propagates to the next layer via skip connections. Each SAW-Layer has separate global and local embeddings calculated which are refined using our feature attention mechanism. We concatenate the features together and pass them to the next layer. Residual connections are also used to facilitate information flow between consecutive layers. Consider the input to the layer to be the feature embedding from an intermediate layer. This input is then examined globally via a point embedding mechanism, and also by computing a dynamic graph, based on 20 nearest neighbors, and then learning the dependency of the neighbours on the point via the edge embedding. The point embedding operation corresponds to a PointNet style feature extraction using a weight shared MLP (conv1D), while the edge embedding corresponds to learning the edge weights via a

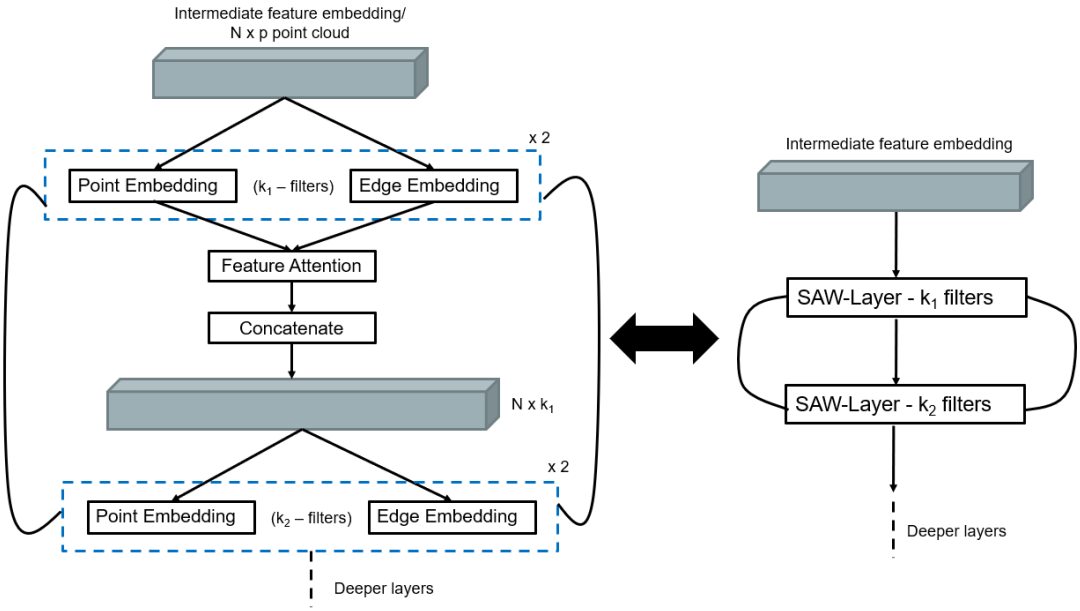


Figure 5.3: Spatially AWare layer structure that is used in SAWNet.

conv1D. We use a feature attention block which is depicted in Figure 5.4 to calculate attention weights for each individual feature embedding which are multiplied by the embeddings to weight them. In the feature embedding block, we learn the attention weights for each global and local feature using a weight shared autoencoder with two dense layers. The input embeddings are first maxpooled to get a single vector following which they are fed into the autoencoder. The first dense layer takes an input the size of the maxpooled output. The second dense layer is the bottleneck layer that compresses this input via some compression ratio (the ratio here is 8). The bottleneck representation is then upsampled back to the original dimension and passed through a sigmoid gating to get attention weights which are multipled by the individual global and local features to weight their effect on the output. The weighted output from the edge embedding layer is then maxpooled across the neighbors, concatenated with the output from the point embedding MLP, and fed into the next layer. Each point embedding computes an embedding using conv1D, batch normalization (BN) and LeakyReLU as a part of its output estimation. The output from the second point embedding in the SAW-Layer propagates via a residual connection to the output of the first point embedding of the next layer. The edge embedding has a similar structure of conv1D - BN - LeakyReLU, which are used to

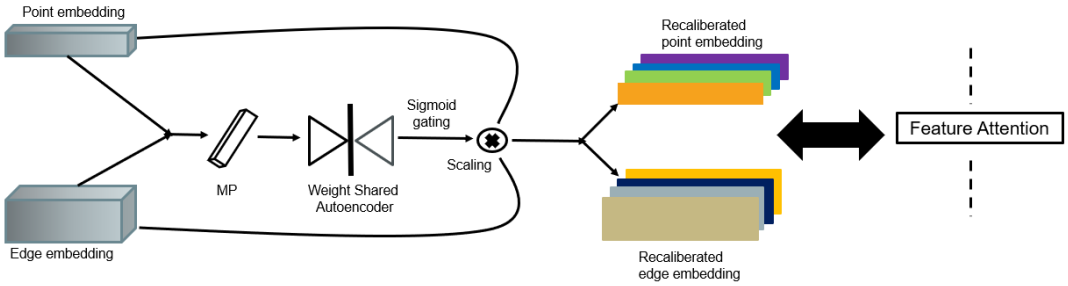


Figure 5.4: Visualizing the inner workings of the feature attention block present in every SAW-Layer.

learn edge weights.

Consider a set of points $X = x_1, ...x_n$, where $x_n \in \mathbb{R}^D$. Here each point is a D dimensional embedding of some point cloud. These points are processed by the SAW-Layer in the following way. The input is passed into two blocks in parallel to get the point embedding and the edge embedding. The point embedding first applies a transformation on the points, $f(t) = f(x_1, ...x_n) = \{h_1(x_1), ..., h_1(x_n)\}$, $h : \mathbb{R}^D \rightarrow \mathbb{R}^M$ where M is the length of the embedding. h_1 here is the shared weighted MLP. The output of the first shared MLP layer is $S_1 = \delta(B_1(f(t)))$. δ is the activation function LeakyReLU and B_i is the batch normalization for the i^{th} instance. This output is then fed into another shared MLP - BN combination. The output can be represented as, $S_2 = B_2(h_2(S_1))$. S_2, is the global feature computation which is obtained after processing the input using the two point embeddings. Similarly, if e denotes the edge embedding, which computes the dependence of a point on its k - nearest neighbours, and $f(k)$ denotes the input in terms of its k nearest neighbours, called the edge feature, then one pass of the input X through conv1D-BN-LeakyReLU would give $E_1 = \delta(B_r(e_1(f(k))))$. Applying another conv1D-BN would give $E_2 = B_s(e_2(E_1))$. For a weight shared autoencoder with weights $\mathbf{W_1}$ and $\mathbf{W_2}$, the the attention weights are computed in the following way, $E_w = \text{sigmoid}(\mathbf{W_2}, \delta(\mathbf{W_1}, E_2))$, *and* $S_w = \text{sigmoid}(\mathbf{W_2}, \delta(\mathbf{W_1}, S_2))$. These are then multiplied by the embeddings as $E_2 = E_2 \cdot E_w$, $S_2 = S_2 \cdot S_w$. The weighted

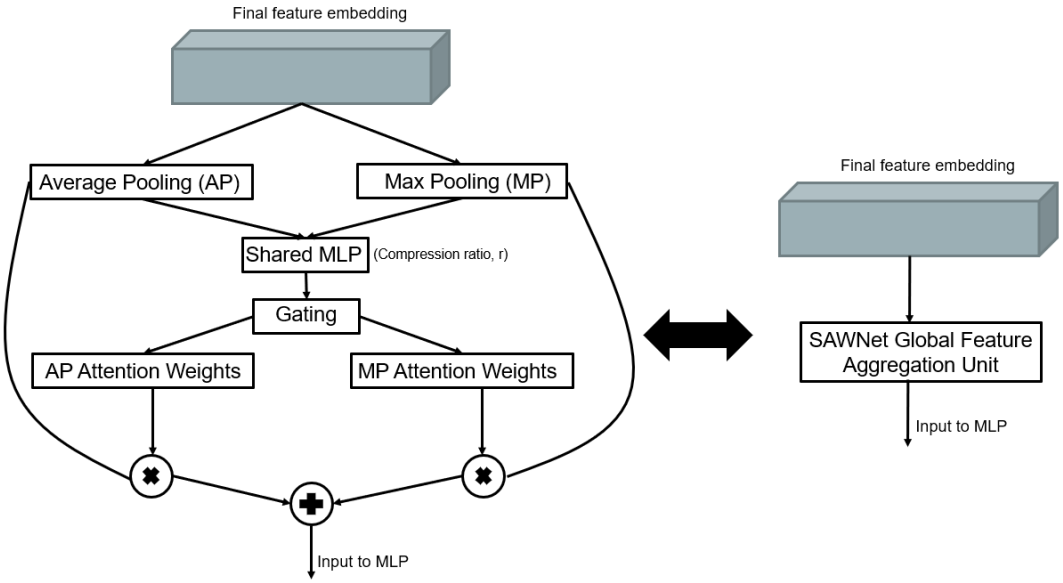


Figure 5.5: Average pooling and max pooling operations are passed through a weight shared autoencoder (compression ratio = 16) followed by a sigmoid gating to give attention weights for both poolings. The scaled outputs are added together to give the SAWNet Global Feature Aggregation Unit.

output of the edge embedding is then maxpooled over its nearest neighbours to match its dimensions with the point embedded output. These values are now the same in dimension, and can be concatenated together, to form the output of the SAW-Layer, $S = [E_2 : S_2]$, where $[\,:\,]$ denotes concatenation. This point embedding is fed into the subsequent SAW-Layers.

5.3.3 Global Feature Aggregation

Existing point cloud processing architecture embed the points into a higher dimensional feature space and follow up with a max pooling based global feature aggregation. Given that all the information present in the embeddings is combined into this global aggregation, it certainly demands further study to see if better encodings can be created of the information extracted via the conv1D operations. To the best of our knowledge, no prior work has been done to study, or modify this global feature aggregation since the inception of PointNet itself. One of the contributions of PointNet was showing that max pooling aggregation works superior to the average

pooling global feature aggregation. But that by no means suggests that none of the average pooled global aggregated features are inferior and cannot contribute to the overall network output. To this end, we propose to take the average pooled and max pooled features and learn the attention weights to scale their individual influences on the output. We take the final feature embedding from the 1024-D SAW-Layer and simultaneously max pool and average pool them to obtain two vectored representations. These representations are both fed into a shared weighted autoencoder similar to the one in Figure 5.4 with a compression ratio 16, the output of which is passed through a sigmoid gating to get probabilities for both aggregations. These probabilities are then multiplied with their respective pooling operations and the scaled outputs are added together. Over the epochs, the network learns to suppress the features in the embeddings that do not contribute to the output leaving the most influential features from both global feature aggregation techniques combining to give better results. The block is illustrated in Figure 5.5.

5.4 Evaluation

We test the performance of our network on the benchmark point cloud classification and point cloud segmentation tasks to show its effectiveness. For each task, we compare our results with the state of the art architectures.

5.4.1 Classification

We use the ModelNet40 dataset [90] which contains 12,311 CAD models for 40 objects in 3D. The models are man made for the 40 object categories and the dataset is divided into a training and test split of 9,843 and 2,468 respectively. We use the same data split for our experiments and report the results on the test set.

To demonstrate that our method improves performance over PointNet and DGCNN, we use the same pre-processing as in their experiments. Hence, for each of the 3D models, we sample 1024 points uniformly from the mesh faces and normalise these points to a unit sphere. The dataset contains points as well as surface normals. We

only consider the point cloud coordinates and discard the remaining information for our experiments. During training, we use data augmentation to add variations in our dataset. Random rotations and scaling are added along with per point jitter to perturb the location of the points. This is the same strategy as was used for data augmentation in PointNet++ [70] training.

Training

We use Tensorflow [1] for all our implementations, unless explicitly stated otherwise. For the classification task, we use the same training setting as in PointNet [68]. Our classification network was trained using a batch size of 8, on one Nvidia GTX 1080 Ti and categorical cross entropy loss. We use Adam [49] with a learning rate of 0.001, which is reduced using a decay parameter of 0.7. The decay rate for batch normalization is 0.9. We train our model for 250 epochs.

Results

Our results are summarised in Table 5.1. The table shows the results for all architectures with their classification results using the (x,y,z) coordinates. All networks shown in the table are trained on 1024 points except SO-Net, which takes twice the number of points as its input. Our network achieves state of the art results in terms of both class as well as instance accuracy. As our architecture is effectively built over PointNet and DGCNN as backbone architectures, it is useful to see if our methodology provides any incremental improvement over the two. To this end we implement SAWNet vanilla, which does not use any attention mechanisms in the SAW-Layers, and also aggregates global features using max pooling just like the other two. It is an extension of PointNet and DGCNN that combines their embeddings to produce spatially aware embeddings with no other extensions. SAWNet vanilla outperforms PointNet's class and instance accuracy by 3.0% and 2.6% respectively, which can be seen as a considerable performance gain. Re-running DGCNN from the author's official github repository gave a class and instance accuracy of 89.2% and 91.6% respectively. As we used that implementation as our backbone, we compare with

Method	Class Accuracy (%)	Instance Accuracy (%)
3D ShapeNets [90]	77.3	84.7
VoxNet [61]	83.0	85.9
Subvolume [69]	86.0	89.2
ECC [80]	83.2	87.4
PointNet [68]	86.0	89.2
PointNet++ [70]	-	90.7
KD-Net (Depth 10) [51]	86.3	90.6
KD-Net (Depth 15) [51]	88.5	91.8
DGCNN [88]	90.2	92.2
SO-Net [55] (2048×3)	87.3	90.9
SpiderCNN [93]	-	90.5
PCNN [5]	-	92.3
PointCNN [56]	88.1	92.2
SAWNet Vanilla (Ours)	90.0	91.8
SAWNet (Ours)	**90.6**	**93.2**

Table 5.1: Classification results on the ModelNet40 dataset.

the results we obtained during our runs. We observe a 0.8% and 0.2% performance gain over the two accuracy metrics by just combining the two embeddings which we believe empirically justifies the need for such spatially aware point embeddings. SAWNet itself on the other hand, outperforms every architecture that processes points in their raw form considerably, on ModelNet40. We outperform DGCNN, the previous state of the art, by 0.4% in terms of class accuracy and by 1% in terms of instance accuracy. The best in class instance accuracy was previously obtained by PCNN [5]. Our architecture outperforms their results by 0.9% with approximately 63% less number of trainable parameters.

5.4.2 Model Performance

Method	Parameters (million)	Model Size (MB)	Inference time (ms)	Accuracy (%)
PointNet [68]	3.5	40	16.6	89.2
PointNet++ [70]	1.5	12	163.2	90.7
DGCNN [88]	1.9	21	27.2	92.2
PCNN [5]	8.2	94	117.0	92.3
SAWNet	2.7	41	52.3	93.2

Table 5.2: Model complexity vs performance for different architectures. Our model provides a good trade-off between model complexity and accuracy (as reported on the ModelNet40 dataset). Our forward pass for inference is considerably faster than PointNet++ and PCNN approaches along with a manageable model size.

Table 5.2 shows the overall model performance of SAWNet with respect to its model complexity. We only compare our model with officially reported values of the architectures. SAWNet contains 2.7 million parameters which means its fairly lightweight. It has less parameters than PointNet even though it combines embeddings from PointNet and DGCNN. This is due to the fact that PointNet contains an intermediate feature transformer that computes a transformation matrix for deeper feature embeddings in the network to account for its lack of geometric invariance

in feature space which is a drawback of global feature embeddings. As our architecture looks at local geometry, it is invariant to these transformations, and doesn't require an intermediate feature transformer. Our inference time is understandably slightly slower than that of pointnet and DGCNN. But it is over 3 times faster than pointnet++ and over 2 times faster PCNN while also being more accurate than the two. Overall, we conclude that our model achieves a fairly good trade-off between model complexity and performance. It is also useful to know that the inference times depend on the type of GPUs used. We used Nivida GTX 1080Tis for all our experiments which have lesser number of cores, compared to the Nvidia Titan X which was used by the respective authors to run DGCNN, PointNet, PointNet++ and PCNN.

5.4.3 Further Experiments on ModelNet40

We used the ModelNet40 dataset for a series of tests to finalise our network architecture. We now discusses these experiments in detail.

Experiments with PointNet

Method	Accuracy (%)
PointNet	88.8
Grouped Embeddings	89.3
Depthwise Embeddings	89.4
Residual Embeddings	90.0

Table 5.3: Experimentation with learning different embeddings for the PointNet architecture. Results are based on our re-implementation of the architecture with the Keras API on the ModelNet40 dataset.

We re-implemented the PointNet architecture with the aim of testing the effectiveness of the shared MLP against other methods for embedding learning. All networks for these experiments were implemented in Keras [11].

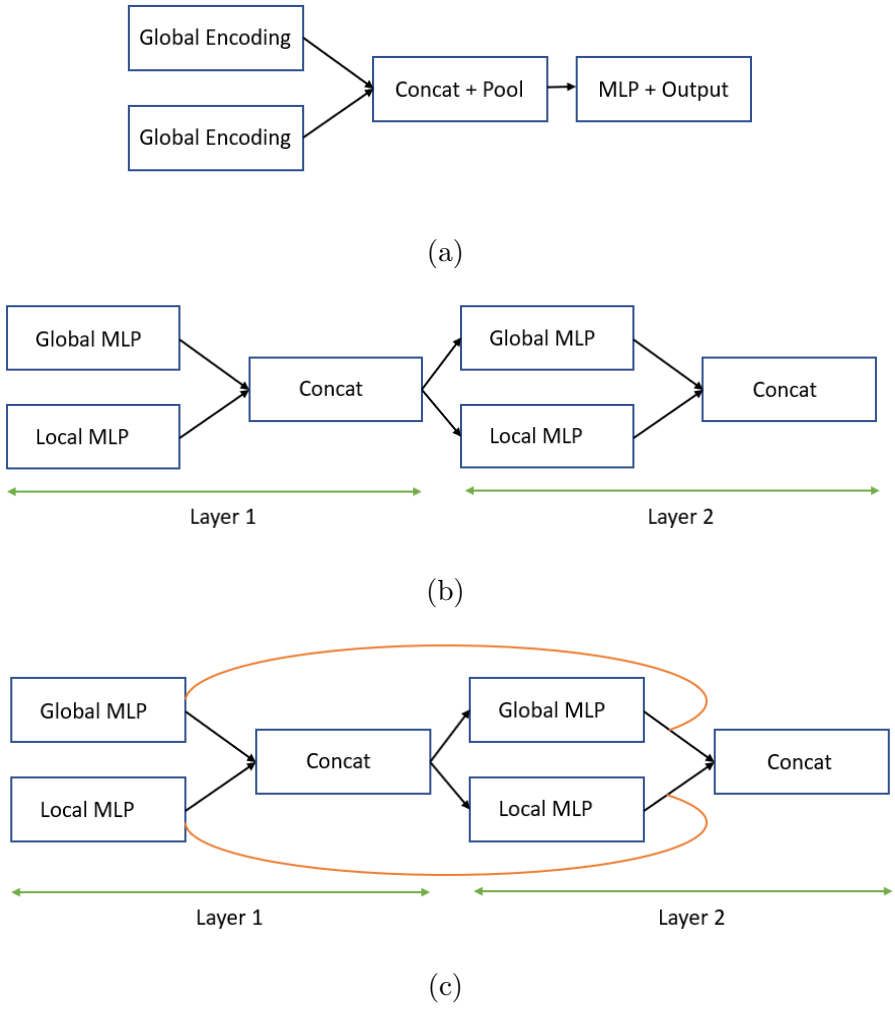


Figure 5.6: Different ablation setups experimented with for spatial awareness.

For the first experiment, we replaced the shared MLP layer with a grouped [38] MLP layer, where we split the MLP into different groups within the same layer, which forces each group to learn to give outputs independent of the other groups. This can be seen as learning multiple shared MLPs within a single layer. Weights are shared within the groups. In the next experiment, the shared MLP layer was replaced by a depthwise [13] shared MLP layer. Here, we learn individual functions for each point in the point cloud, and then use a weight-shared MLP to learn a symmetric function over those functions. Our final experiment consisted of using a residual mapping with the conventional shared MLP. This method gave the best performance increase with PointNet and so we embedded this methodology into our

final architecture. It is useful to note that the grouped MLP and the depthwise shared MLP did offer performance increases over the current shared MLP layer as well, but this increase in accuracy came with its own addition to computational complexity. The residual mapping is chosen mainly due to its greater performance increase, as well as its ease of integration into existing architectures. Table 5.3 summarises our results. The reported value is the instance accuracy (Original PointNet instance accuracy - 89.2%). The training and loss curves for these experiments can be visualised in Figure 5.7.

Effect of nearest neighbours

Nearest Neighbours	Instance Accuracy DGCNN (%)	Instance Accuracy SAWNet (%)
5	90.5	91.6
10	91.4	92.4
20	92.2	93.2
40	91.7	93.3

Table 5.4: Increasing the number of nearest neighbours used to compute edge features.

We experimented with the effect of varying the number of nearest neighbours to compute edge embeddings for our SAW-Layers with DGCNN. The accuracy of DGCNN saturates and starts degrading after 20 nearest neighbours, but the accuracy of our architecture increases with the increases in the number of neighbours. Even though there is a slight performance increase, we still restrict the number of neighbours to 20 to keep model complexity under check. Our results are summarized in Table 5.4.

Experimenting with spatial awareness

Certain combinations of local and global geometric properties work better than others in terms of performance. We combined the global and local vectors at different

Method	Accuracy (%)
Combine at end	88.9
Combine per layer	89.6
Residual + Combine per layer	90.0
Residual + Combine per layer - 1 FC	63.6

Table 5.5: Search for the best method to incorporate spatial awareness in SAWNet. Results are the values for the class accuracy on the ModelNet40 dataset.

stages in the network to test our hypotheses. As we are seeing the effect of spatial awareness in these experiments, we do not use any attention mechanisms here. Our first experiment comprised of running two parallel blocks of embeddings, each looking at global (PointNet), and local (DGCNN) geometric properties respectively. Global feature aggregation over the outputs of these embeddings was concatenated to give a vector that comprised of information from both networks. This was then fed into a 3 layer MLP for classification. We also computed, and then combined, the global and local feature vectors per layer, and computed further embeddings based on this combination.

The combination per layer results in a better accuracy than combining the features at the end. This adds spatial awareness to the network, which it further uses to compute a better embedding of the points.

We then added residual connections to this setting which further improved the performance of the networks by 0.4% and we used this as our final architecture. Further experiments were conducted, such as removing the fully connected layers, but they all resulted in a drastic drop in the accuracy of the network. This leads us to believe that the fully connected layers are an important part of the architecture. We did try to increase the number of SAW-Layers to create a much deeper network which does not require fully connected layers and observed that the model's ability to learn these point cloud processing tasks does improve with deeper layers. Unfortunately,

this also considerably increases the training time, due to which reason we did not proceed further with these experiments. Our results are summarized in Table 5.5. The block diagrams to visualize these ablations are shown in Figure 5.6.

Robustness to missing points

To test how our network would do with sparse input points, we observed the degradation of our network's accuracy with respect to a small number of input points. To drop the points we adopted the same strategy as pointnet and it's variants. All points with indices greater than a threshold index were dropped from the point cloud. We ran the network for 75 epochs with the same hyperparameters as discussed in Section 5.4.1, and recorded the best test set accuracy. Surprisingly, the accuracy of SAWNet does not degrade to a large extent, even when the number of points input to the network drops down to 128, showing how robust our network is to random point dropout. Figure 5.8 shows the comparison of our observation with DGCNN. The graph on the left is the performance of our architecture, and the graph on the right has been reproduced from the DGCNN paper [88]. The performance of DGCNN reduces considerably when the number of points drop below 384, while our network manages to maintain a 86.1% accuracy, even with 128 points.

5.4.4 Ablation Studies

We now shift our attention to show that the final architecture created, is in-fact the most optimal setting of the building blocks that it comprises of. To this end, we take each component which we believe is a contributing factor to the architecture's performance and remove it from the architecture to see its affect on the model's instance accuracy. If removing these components degrades the performance of the architecture, it signifies that it's presence is a contributing to its superior performance.

We single out 4 reasons due to which we believe our architecture outperforms state of the art. These factors are the feature attention block in the SAW-Layer, skip connections between the layers and our global feature aggregation unit. We have al-

Transformation	Accuracy (%)
SAWNet - Feature Attention in SAW-Layer	93.0
SAWNet - Global Feature Aggregation Unit	92.8

Table 5.6: Summarising the two ablation settings experimented with.

ready shown the superiority of having spatially aware layers in Table 5.1 so we omit it for this experiment. Further, in Section 5.4.3, we have already shown how skip connections learn better feature representations. So we experiment with removing the feature attention block while keeping the rest of the network as proposed, and then changing the global feature aggregation unit with max pooling based global aggregation, keeping the rest of the network as proposed, and report our results. In both cases, we observe a decrease in the performance of the architecture, showing that they are important components of our proposed architecture. The proposed global feature aggregation unit seems play a larger effect in our networks performance as it is responsible for almost a 0.4% increase in performance of our final architecture. Table 5.6 summarises these results.

We also experimented with different types of aggregation functions in our Global feature aggregation block. Our options were concatenating, max pooling, and adding the attention scaled pooling values. Concatenating these values after attention scaling performed the worst. Max pooling provided good results, but addition gave the best results due to which we used it in our architecture. Table 5.7 summarises our results. The accuracy and loss curves can be visualised in Figure 5.9.

We finally compared the results of our Spatially Aware transformer network to the transformer network presented in PointNet. We observe a 0.6% performance gain over processing point clouds unaligned, and a 0.3% performance gain over using the PointNet transformation network using our approach. Our results are summarized in Table 5.8.

Method	Instance Accuracy (%)
PointNet + Max Pool Aggregation	88.8
PointNet + Concatenated Aggregation	89.1
PointNet + Max Pool Attention Aggregation	89.2
PointNet + SAWNet Global Feature Aggregation Block	89.4
SAWNet + SAWNet Global Feature Aggregation Block	93.2

Table 5.7: Global feature aggregation ablations. PointNet results are based on our Keras re-implementation that achieves 88.8% instance accuracy (Original instance accuracy - 89.2%)

Transformation	Accuracy (%)
SAWNet + Unaligned ModelNet40 (No Transformer)	92.6
SAWNet + T-Net (PointNet Input Transformer)	92.9
SAWNet + Spatially Aware T-Net (Our Contribution)	93.2

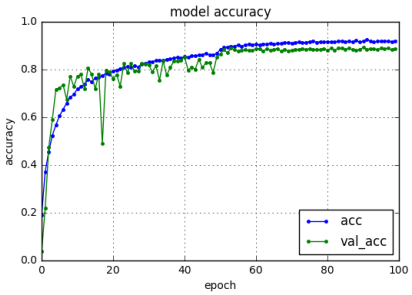
Table 5.8: Effect of input transformer on the SAWNet output.

5.4.5 Segmentation

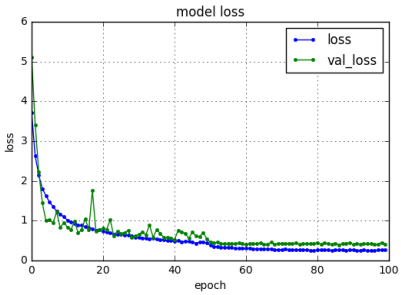
We now compare the performance of our network with respect to the state of the art in 3D point cloud segmentation. For this section, the architecture is the same as the segmentation architecture shown in the bottom branch of Figure 5.2. We embed the transformer aligned 3D points into a higher dimensional space using SAW-Layers, and finally use our SAWNet global aggregation unit to get a point cloud statistic. Residual connections feed information to a final layer before the global aggregation unit were it is concatenated and processed by a single SAW-Layer. We use our global aggregation statistic as the input into a 3 layer MLP which is then reshaped to give the point wise segmentation result. We ran this network with a batch size of 8 on 2 Nvidia GTX1080 Tis. We used 2048 points as input to the architecture.

3D Part Segmentation

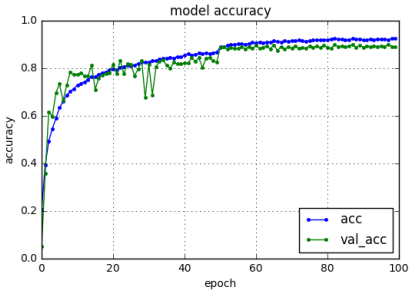
For this section, we use the shapenet part dataset introduced in [94]. Given a 3D point cloud, the task at hand is to segment semantic parts of the point cloud. (Eg, for an aeroplane, the wings, the body etc.). The dataset contains 16,881 3D models of 16 object categories with 50 part segmentation ground truths. Table 5.9 shows our results on the dataset. The evaluation metric for this task is the mean intersection over union (mIoU) for all the shapes in a particular category. It is computed by averaging out the IoUs of all the different shapes belonging to each object category. We use the official train/val/test split for consistency with other results.



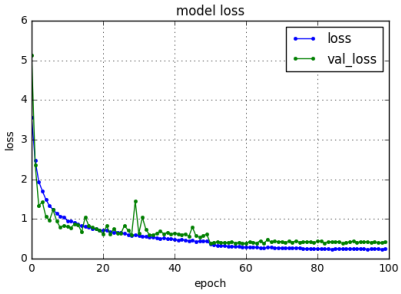
(a) PointNet + grouped point embedding (accuracy vs epochs)



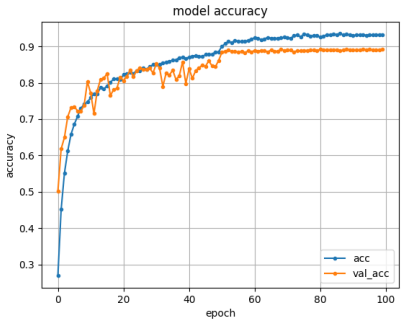
(b) PointNet + grouped point embedding (loss vs epochs)



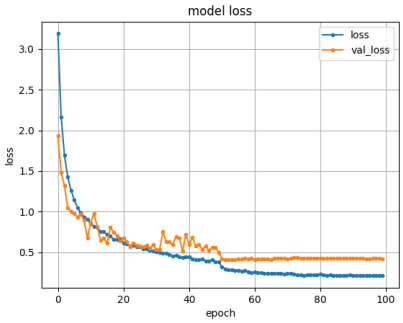
(c) PointNet + residual point embedding (accuracy vs epochs)



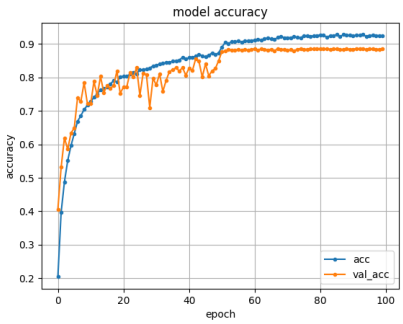
(d) PointNet + residual point embedding (loss vs epochs)



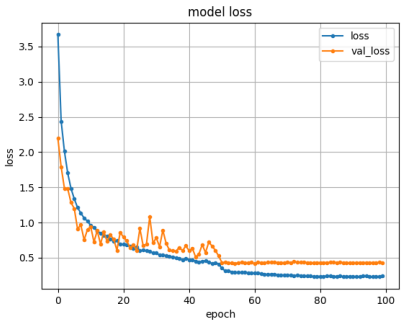
(e) PointNet (accuracy vs epochs)



(f) PointNet (loss vs epochs)



(g) PointNet + grouped residual point embedding (accuracy vs epochs)



(h) PointNet + grouped residual point embedding (loss vs epochs)

Figure 5.7: Ablation studies with different point embeddings

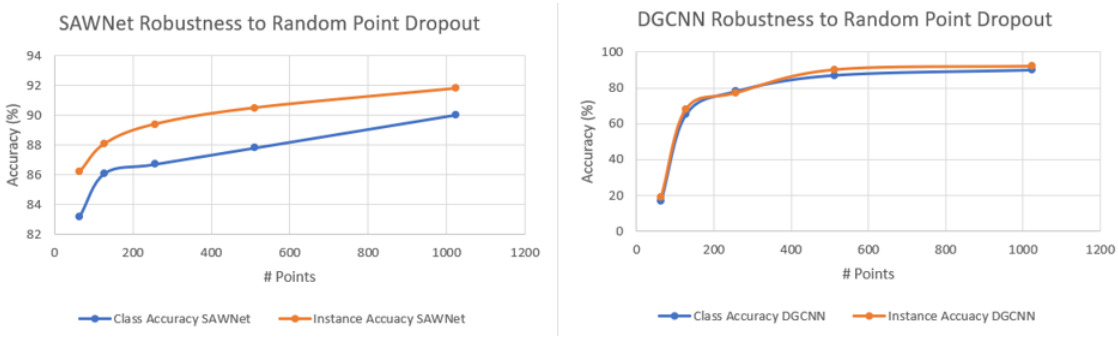
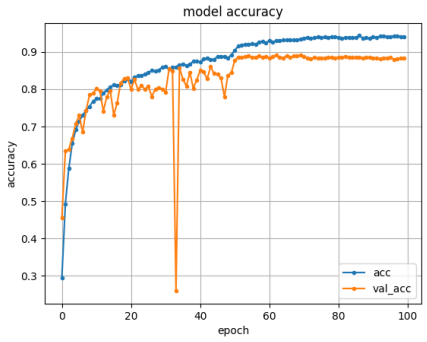
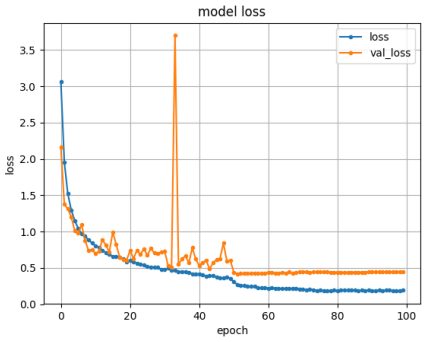


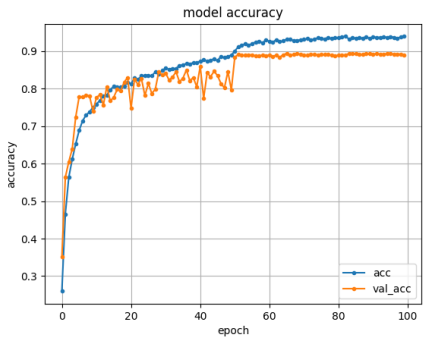
Figure 5.8: Comparing the robustness to sparse point inputs of our network with DGCNN.



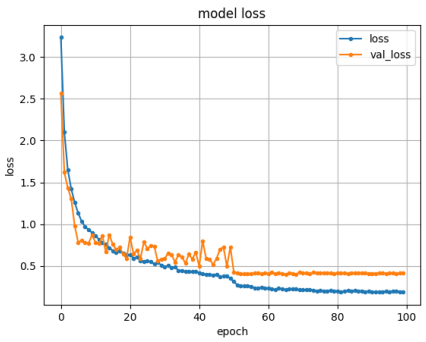
(a) PointNet + global aggregation unit with max pooling (accuracy vs epoch)



(b) PointNet + global aggregation unit with max pooling (loss vs epoch)



(c) PointNet + proposed global aggregation unit with addition (accuracy vs epoch)



(d) PointNet + proposed global aggregation unit with addition (loss vs epoch)

Figure 5.9: Ablation studies with different global aggregations.

Method	IoU	Aero	Bag	Cap	Car	Chair	EarP	Guit	Knif	Lamp	Lapt	MotB	Mug	Pstl	Rckt	Skte	Tbl
PointNet [68]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	**91.5**	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++ [70]	85.1	82.4	79.0	**87.7**	77.3	90.8	71.8	91.0	85.9	**83.7**	95.3	71.6	94.1	81.3	58.7	**76.4**	**82.6**
Kd-Net [51]	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
SO-Net [55]	84.6	81.9	83.5	84.8	78.1	90.8	72.2	90.1	83.6	82.3	95.2	69.3	**94.2**	80.0	51.6	72.1	82.6
DGCNN [88]	85.1	**84.2**	83.7	84.4	77.1	**90.9**	78.5	**91.5**	87.3	82.9	**96.0**	67.8	93.3	**82.6**	**59.7**	75.5	82.0
SAWNet (w/o GAU)	85.3	83.1	83.7	84.2	78.3	90.6	78.3	91.4	87.2	81.4	94.8	71.7	93.8	82.2	58.9	75.5	82.2
SAWNet	**85.5**	83.2	**84.1**	84.5	**78.6**	90.2	**78.7**	91.2	**87.4**	81.6	94.8	**71.9**	94.1	**82.6**	58.3	76.0	82.1

Table 5.9: Part segmentation results on the ShapeNet part dataset. The evaluation metric is mean intersection over union (mIoU).

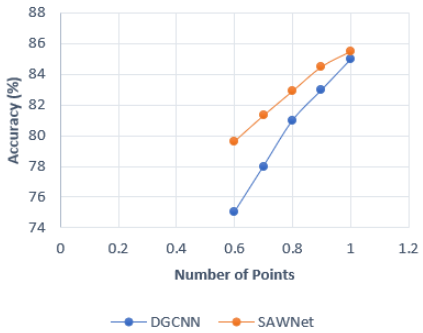


Figure 5.10: Comparing the robustness of DGCNN with SAWNet for part segmentation.

We obtain two results for this task. The first result is obtained using SAWNet without the global aggregation unit (GAU) and the other, with the full architecture as shown in Figure 5.2. Point cloud segmentation has been shown in PointNet to be sensitive to combining the right type of global and local features to obtain the best results. Hence, it is no surprise that our architecture outperforms all existing architectures in terms of the overall benchmark metric on the test set. We get an overall mIOU of 85.5%, which is 0.4% higher than that obtained by PointNet++ and DGCNN, and 1.8% higher than PointNet. Out of the 16 categories in the shapenet dataset, SAWNet gets the best results in 6 categories which is the joint highest with DGCNN. Even on the objects that the architecture does not get the best in class results, it mostly gets a IoU value close to the best for that object. Some visualizations of the part segmentation output are shown in Figure 5.11.

Robustness to random point dropout

We observe the test time performance of our segmentation architecture by dropping out points randomly and seeing it's effect on the network performance. We directly compare with the results presented in DGCNN. Our architecture is robust and accurate, providing a mIOU of almost 80% even when almost half the points are dropped from the point cloud. The performance of our architecture is far beter in general, compared to that of DGCNN in terms of robustness to lesser point inputs. The mIoU of DGCNN drops down steeply with a decrease in the number of points,

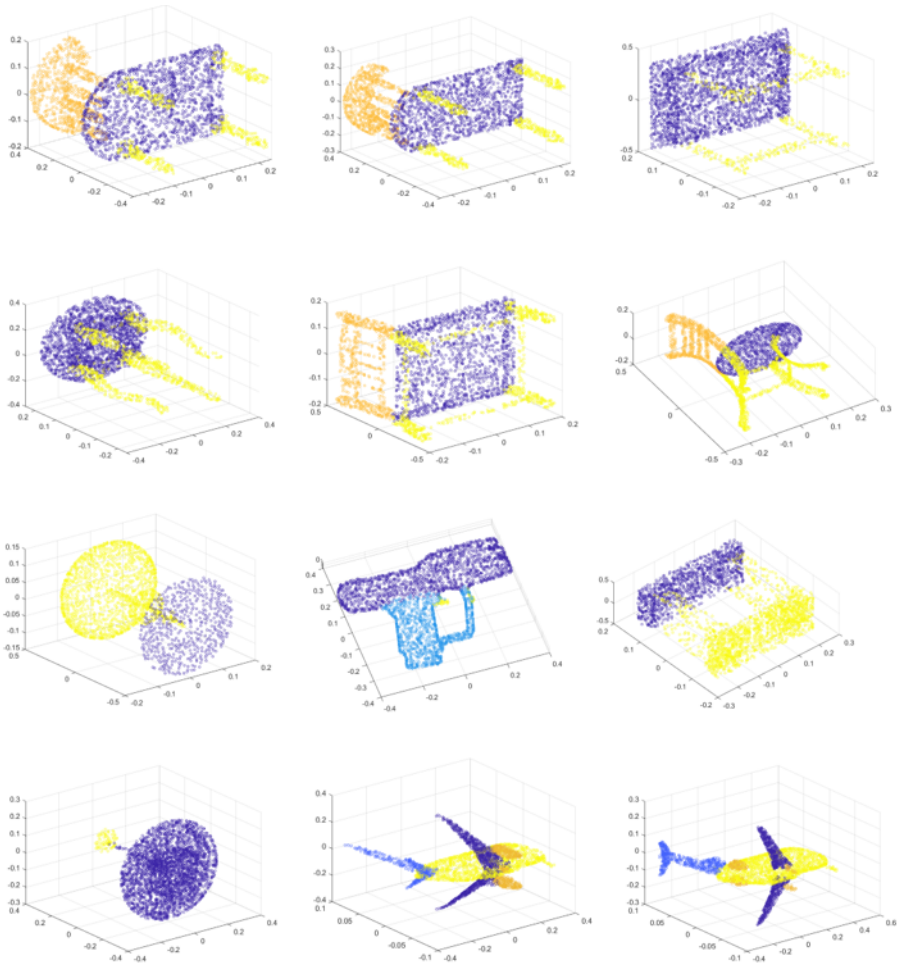


Figure 5.11: Segmentation results on the Stanford semantic scene parsing dataset. The left column shows the predicted segmentation, while the right column shows the corresponding ground truths.

while our architecture still gives a robust performance. We believe it is due to the spatial awareness that SAW-Layers incorporate into the network. The graph showing the quantitative differences between the performance of DGCNN and SAWNet for this task can be visualized in Figure 5.10.

5.5 Summary

In this chapter, we proposed SAWNet, our architecture for 3D point cloud processing that employs residual learning, along with attention mechanisms to achieve

state of the art results on benchmark point cloud analysis tasks. Perhaps the most impressive quality of our architecture is its robustness to random point dropout as it manages to achieve a high accuracy (for classification) and mIoU (for segmentation) for the given tasks where its counterpart architectures perform quite poorly. We hypothesised at the start of the chapter that point cloud processing networks generally learn efficient representations for the points via backpropagation and proposed that spatially aware layers can further increase performance by incorporating additional context in the architectures. Our idea was validated through a series of ablation tests, and by performance of SAWNet on benchmark tasks, keeping the SAW-Layer as it's backbone. The architecture we proposed provides a decent trade-off between model complexity and performance. Our spatially aware layer is a simple novel enhancement that can be easily incorporated into any existing 3D point cloud processing pipeline to get a performance boost.

Chapter 6

Conclusion

We conclude the thesis in this chapter by providing the summary of what we achieved in Section 6.1. We provide the general conclusions for our research in Section 6.2 and present future directions of research that this research can take in Section 6.3.

6.1 Thesis summary

In the introduction of this thesis, we set the scene for the need for residual learning and attention mechanisms in the field of medical image segmentation and 3D point cloud processing, starting from the performance of conventional machine learning algorithms for generic computer vision tasks. We provide the context in which this research is useful, which is largely, superior feature extraction to get better results as well as understanding the role of attention mechanisms and residual learning in the discussed domains.

The literature review started with an initial background of neural networks building up to deep learning and convolutional neural networks. We discussed the literature on residual learning, U-Net and it's variants and point cloud in detail. The literature review focused on the architectures that this thesis compares against. We highlighted a few drawbacks in current literature - 1) Specialized residual architectures hardly exist in current literature for medical image segmentation. Most architectures use a backbone pretrained on imagenet and refine the weights using

the training set. 2) Visual attention mechanisms are not widely incorporated into medical image segmentation pipelines. 3) Residual learning has not been looked at in the context of 3D point cloud analysis. 4) The influence of attention mechanisms on point cloud embeddings had not been investigated. Neither were ways to aggregate point features better into aggregated global embeddings. Our thesis contributions address these gaps in the current literature.

6.1.1 FocusNet

FocusNet proposed a novel attention mechanism that combined spatial attention with self attention in terms of feature map recalibration, in an attempt to propagate the most relevant information forward in a network. We investigated the influence of this dual branch encoder decoder scheme on ISIC 2017 and lung segmentation datasets which showed results superior to existing techniques. We proposed a loss function enhancement whose derivative has a non linear response close to the minimum on the loss landscape, to converge to better a solution. We showed such a loss outperforms existing loss functions and boosts performance in U-Net and FocusNet. We highlighted the drawbacks in FocusNet, hence setting the scene for a more improved architecture.

6.1.2 FocusNetAlpha

We propose an extremely efficient and accurate medical image segmentation based on our novel residual group attention block which outperforms state of the art architectures. We also propose an extremely lightweight variant of this architecture that performs at par with architectures almost 2.5 times its size. We adapt the tversky loss and balanced cross entropy loss in the adaptive logarithmic loss setting to boost performance over true positives and true negatives to get more well rounded segmentations. This can be validated by adequately high dice index, recall and F1 scores that the architectures get compared to just using the hybrid loss. We show the trade-off between model complexity and performance for various architecture, where our architecture consumes the least parameters and FLOPs while giving the

best results.

6.1.3 SAWNet

We discussed how learning point cloud representations is effectively based on the ability of backpropagation to converge to a better solution, having set some initial state (point embeddings, edge embeddings). We propose to combine local and global information from different embeddings to produce a superior spatially aware embedding. This embedding infused with attention, a spatial transformer network and a novel global aggregation unit, form the building blocks of SAWNet which beats state of the art results on point cloud classification and segmentation tasks. The model also achieves a reasonable tradeoff between complexity and accuracy.

6.2 Conclusions

We provide a general critique based on our investigations in this thesis.

- The networks presented in this thesis are different from the general trend that exists in deep learning. Architectures generally tend to go deeper to get better results. Our proposed architectures on the other hand have a more horizontally branched out structure with multiple branches processing the inputs concurrently. We have noticed this trend to work better than stacking deeper layers in convolutional neural networks.

- As these architectures get 'wider' and more branched, we observed dropout to stop being an effective regularizer and actually in certain cases, degrade the performance of the networks. This was especially true in FocusNetAlpha where we only implement dropout in the final bottleneck layer which has a filter size of 512. As the usefulness of dropout decreases, we observe batch normalization to play a really important role in training networks with a branched out structure. In FocusNetAlpha, adding a bn-LeakyReLU block after the permutation invariance 1×1 convolution itself increases performance by almost 0.5%. We noticed a similar trend in SAWNet, where batch normalization gives SAWNet

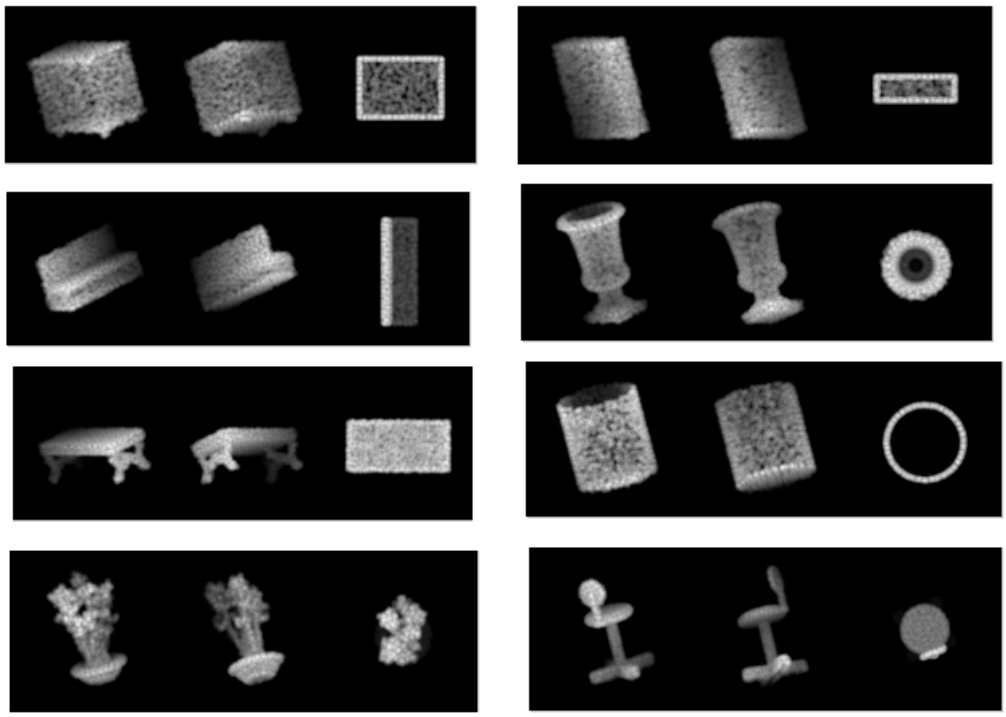


Figure 6.1: Three views of a misclassificed point cloud projected on a 2D grid for visualization. The predicted and ground truth labels are listed in the text.

a bigger boost in performance, that it does for PointNet or DGCNN.

- We tried to visualize the samples SAWNet misclassified on the ModelNet40 dataset. We noticed that the samples that SAWNet misclassifies do potentially resemble the features of the object corresponding to the misclassified label. We visualize some examples to better understand this. Figure 6.1 shows the point clouds projected on to a 2D surface. From left, they are, a dresser predicted as a night stand, a wardrobe predicted as a box, a bench predicted as a sofa, a flower pot predicted as a vase, a table predicted as a bench, a vase predicted as a cup, flower a pot predicted as a plant and a stool predicted as a chair.

- Our architectures and loss functions are already open sourced on github. They are also constantly updated over time with additional experiments and results.

6.3 Future work

There are quite a few directions this research can be extended in. Here, we list the most interesting ones.

- Even though we managed to considerably improve the decoding scheme in FocusNet style architectures, we do believe there are ways to further boot performance. Investigating better ways to combine features at different scales is of particular interest, but is something we did not have time to look into in this thesis.

- Present experiments with the adaptive loss relies on grid search to look for the best hyperparameters of the function. An interesting future approach would be to make these hyperparameters trainable, and use techniques such as genetic algorithms or end to end training along with the network weights to find their optimal setting.

- We ran some initial experiments that suggest that whenever there exists a 1D convolution in a skip connection to match filter map sizes for residual addition, the performance of the networks decreases slightly. We implemented a residual concatenation rather than a residual addition to eliminate the need for 1d convolutions and noticed a 1.5% increase in performance on the validation set of CIFAR-10. This is due to the fact that these 1D convolutions inhibit propagation of the gradient back to the initial layers showing that there is still room to improve gradient backpropagation in residual networks. We believe this work could be taken forward and better residual networks can be designed that that this feature into account.

- Open source implementations of group convolutions with CUDA C/C++ would accelerate research into networks that use them as a backbone. The increases in performance based on unleashing the full power of grouped convolutions is a direction, we expect future research to take in the coming years.

Appendix A

Optic disc segmentation using a deep fully convolutional neural network

We describe a set of preliminary experiments on optic disc segmentation that led to the initial works of this thesis towards FocusNet. The chapter is written in the form of a research papers and is, to an extent, self contained. Our described methodology placed 3^{rd} in the IDRID optic disc segmentation challenge at ISBI 2018 (https://idrid.grand-challenge.org/Leaderboard/).

A.1 Introduction

Optic disc (OD) segmentation is a fundamental task when it comes to processing eye images. Many diseases can cause the optic nerve irreversible damage and can lead to blindness. Therefore, segmenting the optic disc is an important step in creating a frame of reference for diagnosing optic nerve head pathologies. This, in turn, calls for a reliable OD segmentation technique for automatic screening of optic nerve head abnormalities.

Recent years have witnessed the emergence of deep learning. Methods based on convolution neural networks are quickly becoming the baseline for most computer

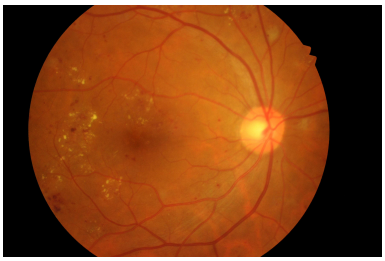



Figure A.1: The following are images obtained from a fundus camera. The bright round patches in the images are called the optic discs.

vision tasks. The work of [34] has suggested that a deeper network, doesn't necessarily mean a better network. They introduced identity mappings to facilitate better learning and to allow the networks to go deeper and also use fewer parameters. [79] Introduced the fully convolutional networks for image segmentation, which used skip connections to combing predictions from the previous layers. The stride was adjusted as a hyperparameter to get finer predictions. [73] Introduced an encoder decoder structure called the U-Net which has found great success in biomedical imaging tasks. We blend the power of residual connections into the structure of the U-Net to facilitate segmentation of the optic disc from fundus images.

The structure of this chapter is as follows. In the next section, we introduce the residual U-Net architecture, which is followed by the the experiments conducted. We end with our results on the dataset, and conclusions.

A.2 Methodology

A.2.1 Architecture

A.2.1.1 Unet

The U-Net [73] gets its name from its 'U' shape. To the left is an encoder style structure, and to the right is a decoder style structure. The architecture employs low level information from images to do pixel level classification. The encoder learns a hierarchical representation of the image and the decoder upsamples the image back to its original size giving pixel level predictions. The output of a particular block in the encoder is concatenated with the output of the corresponding decoder layer

to facilitate better information flow through the network (see Figure A.3). This is termed as a contracting path by the authors and helps capture context and precise segment localization. The task at hand (OD segmentation) has very few training images to work with. As the U-Net architecture was built with this very problem in mind, it is hence the architecture of choice for this problem.

A.2.1.2 Residual Mappings

A simple residual connection [34] is the concatenation of the input to the output of a weight layer. It is given by the equation,

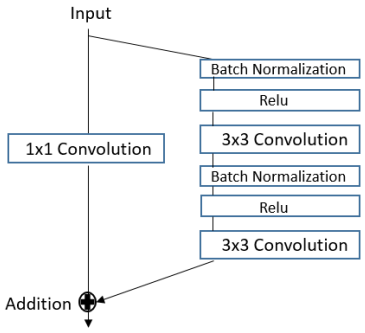
$$y_i = h(x_i) + F(x_i, w_i),$$

$$x_{i+1} = f(y_i)$$

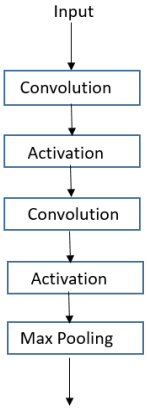
Such residual blocks can be stacked one after another between pooling layers to get a resnet style architecture. However, there are more complex identity mappings which help build deeper and better networks. The advantage of using residual connections is that they help in better gradient flow between layers due to their identity mappings which helps to alleviate the vanishing gradient problem to a huge extent. As the gradient can be propagated better through the layers using these mappings, networks can now be constructed which are deeper and have improved performance. [35] Presents a detailed analysis of different residual mappings. Our residual block can be seen in Figure A.2 (a). It contains batch normalization after the input layer. ReLU activations are used and there are two convolution operations, with a 3×3 receptive field. Instead of an identity mapping, we concatenate the input to the output following a 1×1 convolution which helps in reducing the number of parameters in the network.

A.2.1.3 Residual U-Net

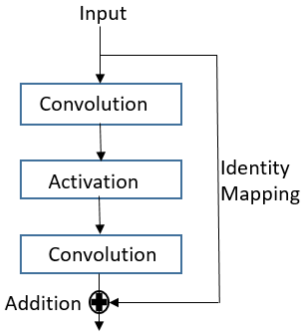
The residual U-Net is the U-Net architecture where the convolutions are replaced by the residual blocks. The combination of residual connections with the U-Net helps propagate information even better through the network than a vanilla U-Net



(a) Residual block in the proposed architecture



(b) Unet Conventional data flow



(c) Simple identity mapping

Figure A.2: Different Mappings from information propagation in the unet architecture

(which can be seen through the results). The architecture can be seen in Figure A.3.

The input is a $256 \times 256 \times 3$ image, passed into a convolution layer. This is followed by a residual block and a max pooling layer. There are 9 residual blocks in the encoder side and 9 in the decoder side. Downsampling is done using the pooling operation in the encoder. The decoder uses the upsampling operation as a means of 'unpooling' to increase the size of the feature maps spatially. ReLU activations are used throughout. After the last residual block, a 1x1 convolution is followed by a sigmoid activation to project the feature map into the desired segmentation. Our network is 'deeper' than the vanilla U-Net with 33 convolution layers, compared to

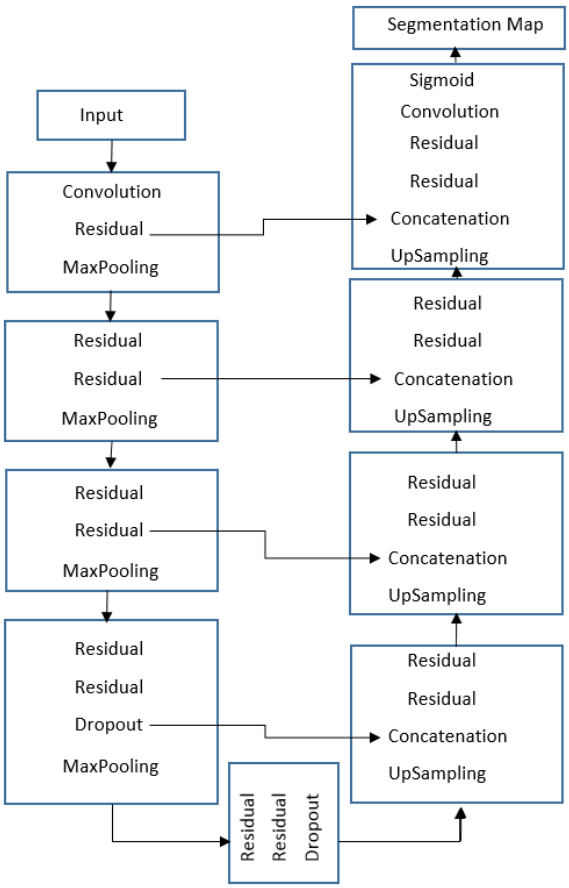


Figure A.3: Proposed Residual U-Net Architecture. The convolution layers use ReLU activation except the last one which uses sigmoid.The output of the residual layers and the first dropout layer is concatenated with the upsampled output in each decoder block.

the 23 of the vanilla U-Net.

A.2.2 Loss Function

We train the network on a loss function which is the negative of the global dice index. This is given by,

$$Negative\ Dice\ Loss\ =\ -\frac{2 * |X \cap Y|}{|X| + |Y|}$$

It is global in the sense that it is trained over the values in a batch rather than each image separately. It is also easier to optimize as a cost function, as it gets rid of

any discontinuities that arise in the cost function due to trying to map a continuous probability to a discrete 0 or 1 value.

A.2.3 Evaluation Metrics

The evaluation metrics used are the dice index, the jaccard index, sensitivity and specificity. The dice index is discussed in Section 2.2. The jaccard index is an intersection over union metric, like the dice index. Sensitivity is defined as the true positives divided by the total positive samples and specificity is true negatives divided by the total negative samples.

A.3 Experiments

A.3.1 Dataset and Augmentation

The dataset contains 54 very high resolution (4288x2848) fundus images with the optic disc segmented in the ground truths. This amount of data is obviously not enough for a deep learning task, so data augmentation was applied. We applied random zooms, rotations and channel shifts to create a dataset of roughly 1400 images and a validation set of 270 images. The exact same random transformations applied to the images, were also applied to the ground truth masks. As a few optic discs were close to the edge of the images, care was taken to not crop them out during the augmentation process. We initially tried running the architecture with 128x128x1 grayscale images but settled on 256x256x3 preprocessed RGB images as it gave better results.

A.3.2 Implementation Details

Before the augmentation, data preprocessing was applied. The fundus images are affected by camera illumination problems, and they also have a thick black border which affects the quality of the segmentation. So the image was first scaled to an arbitrary size which reduced the black border, and then mean local colour was subtracted from the image in an attempt to get rid of illumination variations. The

images were then re-sized to 256x256x3 which formed the input to the network. Inputting this data into the network saw us run into the exploding gradient problem. This was rectified by standardizing the data. The model was implemented using the keras [11] library with a tensorflow backend. It was run on a Nvidia GTX 1080 with a batch size of 16. The Adam optimizer was used with a learning rate schedule which was determined experimentally. Dropout was used to improve the generalization capacity of the network, with a rate of 0.2, which means that 20% of the nodes were randomly dropped at one iteration. The network converges in about 50 epochs.

A.3.3 Results and comparisons with the vanilla U-Net

The results are summarized in Tables A.1 and A.2. The values of all the evaluation metrics are calculated by comparing the original sized ground truth masks and the predicted masks for the 54 fundus images. It can be seen that the residual U-Net performs considerably better than the vanilla U-Net. Some results of images after the optic disc segmented are also shown in Figure A.4.

Table A.1: Unet Results [73]

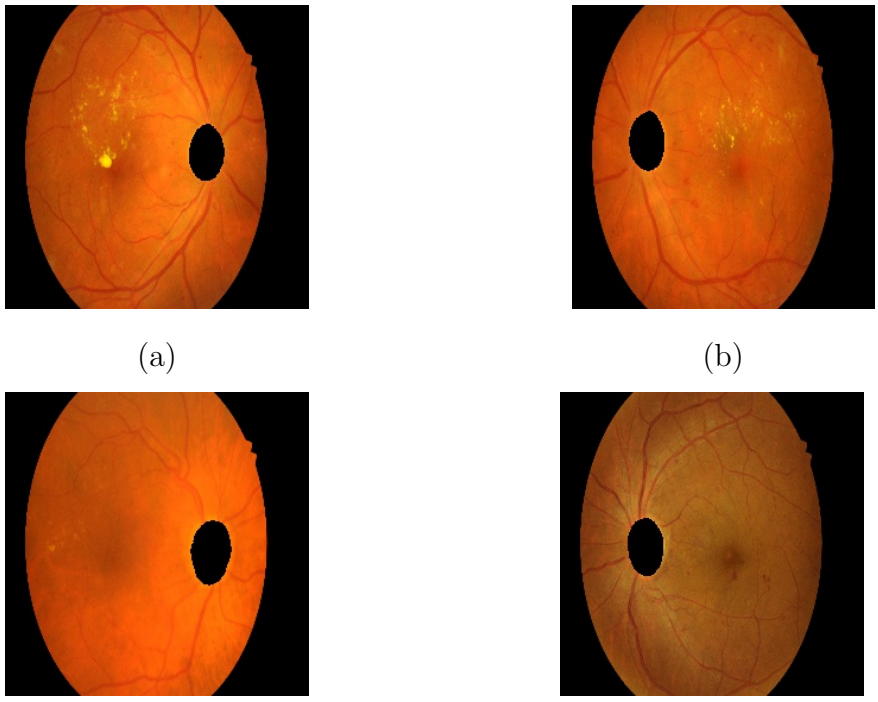
Dice Index	Jaccard Index	Sensitivity	Specificity
0.9693	0.9404	0.9987	**0.9999**

Table A.2: Residual Unet Results

Dice Index	Jaccard Index	Sensitivity	Specificity
0.9805	**0.9618**	**0.9999**	**0.9999**

A.4 Conclusion

In this chapter, we presented out residual U-Net architecture for the segmentation of optic discs from fundus images. The combination of residual connections with the architecture of the U-Net provided promising results on the dataset. The residual



(a) (b)

(c) (d)

Figure A.4: Fundus images with the optic disc segmented out

connections facilitate better gradient flow due to which the accuracy of the residual U-Net is better compared to its conventional counterpart.

Appendix B

3D face landmarking using attention-based deep convolutional neural networks

We describe a set of preliminary experiments on 3D face landmarking that led to the initial works of this thesis towards SAWNet. The chapter is written in the form of a research papers and is, to an extent, self contained.

B.1 Introduction

Landmarking images is a key first step in many applications in computer vision and graphics. In terms of face processing, these sets of anatomical points can play a key role in various tasks such as estimating pose, alignment, model construction, recognition and classifying emotions. In terms of a 3D face dataset, landmarks provide an initial set of sparse correspondences between all the scans, after which dense correspondences can be obtained. An important application that requires landmark estimation is the building of 3D Morphable Models (3DMMs) of the face. The quality and ability of a model to be useful is mainly governed by the quality and diversity of images provided in its training set, and the quality of the dense correspondences between the 3D scans. Accurate landmarking is a key first step. Detecting these landmarks often refers to finding well-defined regions of high curvature on the 3D

face, such as the eye corners and lip corners. It also refers to landmarks that are less well-defined such a the nose tip and chin. It is much harder to establish a clear manually-labelled ground-truth location for such landmarks.

Most of the work in learning landmark locations on 3D faces has been dominated by statistical machine learning techniques. Since the surge of deep learning, 2D face landmarking has received a lot of interest from the research community, but significantly less effort has been made to extend this work to 3D face images. This may be due to lack of the amount of data that is required to create accurate models using deep neural networks.

In this chapter, we present an approach to learn landmark positions on faces using attention-based convolutional neural networks. Our architecture is trained on a combination of synthetic and real-world 3D face images. The synthetic faces are generated using the Basel Face Model (BFM) [30], a 3D morphable model of face shape variation. We generate 40,000 images using the model in different poses and expressions to generate a diverse dataset which alleviates the problem of scarcity of training data. We follow the theme of convolutional neural networks (CNNs) using residual mappings to train deeper networks, as they have been shown to provide better results than stacking convolution layers in a VGG-style network. We train a U-Net style architecture to learn to segment candidate landmark regions from depth maps. Following this, we combine the intermediate outputs from the upsampling layers of this network, with the intermediate layer outputs from our regressor network to give better landmark estimation. For the combined output volume in every layer, we learn a weighting of the activation maps, such that ones contributing more to the output have a higher weight.

The main contributions and highlights are as follows:

- We present a novel pipeline to combine feature maps of one network, with feature maps of another, as an attention mechanism.
- To the best of our knowledge, this is the first work to analyze the use of such

deep networks for the task of 3D face landmarking.

- We provide an extensive analysis of our mechanism and empirically show that our technique of *concatenate-and-weight* the activation maps is the reason for the improved performance of the landmark regressor.

The rest of the chapter is organized as follows. We discuss related literature in Section B.2. B.3 describes the overview of our approach. Section B.4 briefly describes the construction of our dataset. Section B.5 focuses on our attention-based pipeline, where we explain our mechanism in detail. We give our analysis in Section B.6, results in B.7, and finally present conclusions.

B.2 Related Work

We first present related literature on 3D face landmarking and then on CNNs.

B.2.1 3D Face Landmarking

Many earlier approaches do not use CNNs but instead use hand-crafted features. Usually some local 3D descriptor, such as 3D SIFT [77], or the spin image [43] have been used to localize and predict landmark locations. Conde *et al.* [16] used both curvature and spin images to extract geometric features from images and used SVMs for their classification. Perakis *et al.* [66] proposed a facial landmark model based on spin images and the shape index to generate templates for each landmark on the 3D face, creating a method that worked well on faces with large pose variations. Creusot *et al.* [17] first computed feature vectors containing different local descriptors that were then normalized with respect to the distribution of the landmarks. This is followed by extracting an optimal function of this vector that separates the landmarks from the rest of the facial region. Romero *et al.* [72] use a point-pair descriptor approach by encoding 3D shape between a pair of 3D points. The descriptors used in this approach were spin images and cylindrical sampled Radial Basis Functions (RBFs). Gilani *et al.* [31] used a deep learning approach to landmark 3D faces. They first rendered an RGB image using depth, azimuth and elevation images and cre-

ated dilated ground truth maps by projecting the landmarks to a 2D grid and then convolving it with a 10x10 disk structuring element. They trained their Deep Landmark Identification Network (DLIN) using these images and ground truth, treating the problem as an image segmentation task. The output maps for the FRGC and Bhosphorus datasets were then used to find the points back in 3D, which obtained impressive localization results on these datasets.

B.2.2 Convolutional Neural Networks

Since Alexnet [53], CNNs have achieved state-of-the-art performance in many areas of computer vision. Architectures have evolved through recent years, with the research community providing insights into how the accuracy of these networks can be increased even further, and in some cases, even surpass human accuracy [34]. VGGnet [81] was the first 'deep' network that increased depth in an architecture, which used very small receptive fields (3x3) throughout the network. More recently, architectures based on learning the residual have shown impressive results on various 2D image datasets. ResNets [34] hypothesized that it is easier for a CNN to learn the residual mapping than a sequential mapping of convolution layers in a neural network, such as the VGGnet [81]. They facilitate the building of deeper networks whose accuracy doesn't degrade with the addition of more layers. Since the initial ResNet paper, many studies have been conducted to improve these residual blocks further. The introduction of batch normalization and a non-linear activation inside the vanilla residual block improved the accuracy even further by propagating information better and deeper through the network. CNNs have also been applied extensively to tasks of image segmentation and object detection. In terms of segmentation, fully convolutional networks for semantic segmentation [79] fine-tuned conventional architectures via transfer learning and by using skip connections to combine semantic information from the deeper layers of the network for upsampling. The U-Net [73] is another segmentation architecture in the form of a convolutional autoencoder. It hierarchically upsamples the output of the encoder and combines intermediate encoder outputs with the decoder's upsampled outputs via skip connections. Attention is a concept that involves focusing the network's observation

towards the parts of the image that contribute the most to the neural network's decision. It is used extensively in sequence modelling [59] [10], and localizing parts in an image [91] [10]. Xiao *et al.* [91] proposed a two-stage visual attention mechanism for fine-grained image classification where an object-level attention mechanism selects relevant object candidates and then localizes its discriminative parts. They use their attention mechanism to train domain-specific networks. Spatial Transformer Networks [40] (STNs) are a self-contained, fully-differentiable attention module for neural networks, which learn invariance to translation, scale and rotation. They have been shown to work well with classification of images containing noise. The 'Squeeze-and-Excitation' (SE) block [36] is another lightweight module that can be seen as a self-attention mechanism as it learns a recalibration of weights corresponding to the output feature maps of an intermediate convolution layer. The SE block has been shown to learn these feature-wise weights across the depth of an output feature map volume, independent of the classes at the lower layers, and in a highly class-specific manner in the deeper layers. We use the SE block as our lightweight gating mechanism to learn which parts of the concatenated volume of the attention and feature maps contribute more to the final output.

B.3 Overview

Given a set of n facial depth maps, $F = \{f_1, f_2 \ldots, f_n\}$, where $f_i \in \mathbb{R}^{M \times N}$, our goal is to train an end-to-end face landmarking system that predicts the landmarks on the representation of a 3D face image in a robust and accurate manner. To this end, we train two convolutional neural networks (CNNs) - one for the task of coarse landmark localization, and the other to refine this localization to predict the fine position of the landmark with minimal possible error. We use depth maps projected along the z-axis of the 3D faces, as our representation. Given these depth maps as inputs, a CNN finds a non-linear mapping from the inputs to the outputs of the form $D(F) \rightarrow L$, where $L = \{l_1, l_2 \ldots l_n\}$ is the set of predicted landmark locations for the n faces. For each face there are p landmarks, so that $l_i = \{x_1, y_1, x_2, y_2 \ldots x_p, y_p\} \in \mathbb{R}^{2p}$. are the coordinate landmark locations on the depth map. Given these points, we can

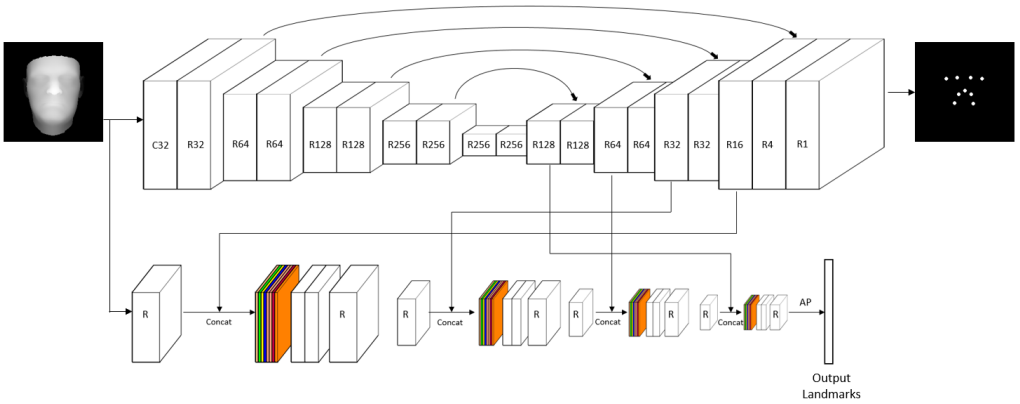


Figure B.1: The full pipeline of our face landmarker. R denotes a residual block and C denotes a convolution block which is followed by a batch norm and ReLU activation. The numbers are the filters used in the layers. The arrows from the encoder to the decoder of the convolutional autoencoder denote the skip connections. The arrows from the autoencoder to the regressor denote attention map concatenation.

use 2D-3D coordinate mapping of the datasets to project the points to 3D.

We have employed residual networks throughout for building our architectures, due to their desirable quality of providing an increase in accuracy with increased network depth. This gives us the ability to build deeper networks that don't saturate. The landmark localization is achieved by creating a dilated mask for every depth map, which contains the coarse location of the landmarks. We train a (U-Net style) convolutional autoencoder to learn the location of these regions, in turn, learning the coarse locations of the landmarks on the face. We then use the output from the upsampled layers of our decoder to guide the landmark refinement task of our regressor. This provides us with the locations of the landmarks with a higher accuracy. Since we use the intermediate outputs from the pre-trained autoencoder to guide our regressor to achieve a coarse-to-fine landmark prediction, we call this our attention mechanism.

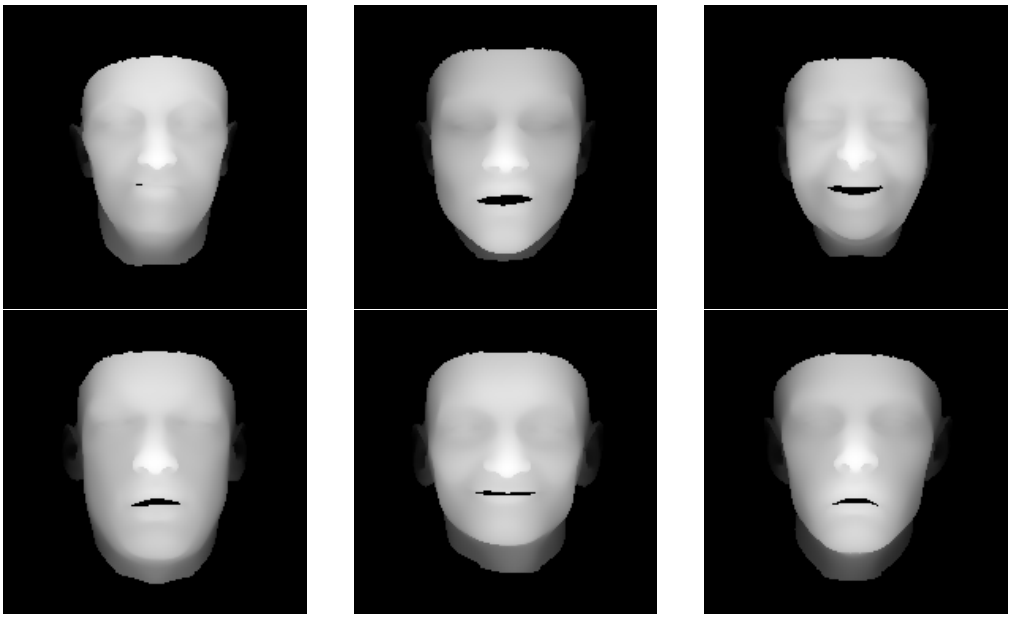


Figure B.2: Sample depth maps generated using the 2017 Basel Face Model. The images rendered are rich in expression variation.

B.4 Dataset

A large amount of data is required to train a CNN to produce meaningful results. Unfortunately such a large corpus of 3D images doesn't exist, as 3D image acquisition is a relatively hard task compared to its 2D counterpart. Hence, we use a 3D morphable model of face shape, texture and expression variation to generate data for our training and validation sets. We use the 2017 Basel Face Model [30] learnt from 200 3D faces with neutral expressions and 160 with expression deformations. It is a robust model of 3D faces built with improved diversity in terms of age and expression in its training data. The model obtains correspondences using a Gaussian Process based registration algorithm which incorporates face-specific domain knowledge into the model. We sample across the model to generate faces using the 199 principal components that characterize the shape and texture of the model, and the 100 principal components that characterize the expressions. We generate depth maps for each face in five different poses - neutral, $\pm 10^{\circ}$ pitch, and $\pm 10^{\circ}$ roll. The depth maps generated are of size 192x192. Example images are shown in Figure B.2.

We further use images from the FRGCv2 database with the motive of adding some real world image data to our training set. We use the spring2004 images in

the dataset to render depth maps in the same poses as the depth maps generated for the morphable model.

We generate ground truth masks for training our attention network in the manner where we project the 3D landmarks on to a 2D grid, the same size as the corresponding depth map, and dilate it using a 4x4 structuring element.

B.5 Incorporating Attention in CNNs

Our attention-based CNN architecture is based on learning the weighting of the concatenated volume of attention and regressor feature maps, based on which maps in the volume contribute most to the output. We treat the task of landmark localization as a regression task and use a deep residual network as our backbone architecture to regress to their locations. The diagram of the proposed architecture can be seen in Figure B.1.

B.5.1 Attention Network

Our attention architecture is derived from Ronneberger *et al.* [73]. We created a convolutional autoencoder by first downsampling the input image through a series of convolutions, and then upsample from the bottleneck layer while adding skip connections from the encoder to the decoder for every intermediate upsampling. We use the full-preactivation, no bottleneck residual block instead of standard convolutions. The input to the residual block follows two paths to the output, where the intermediate output values of the two paths are added to produce the output of the block. The first path is an identity mapping, while the data in the second path flows through a batch normalization layer, followed by an activation layer, followed by a convolution layer, twice in the same manner. Considering the fact that 3D faces have smooth transitions across their surface, a smaller number of convolution operations are required to capture the nuances across it. Keeping this in mind, the encoder is composed of 4 layers. The first layer contains an initial convolution block with `conv-batchnorm-activation`, followed by a residual block and a max pooling operation. Each of the remaining three layers contain two residual blocks followed

by max pooling to downsample the input volume to the layer. The bottleneck layer contains two residual blocks encoding the representation of the data. This representation is upsampled hierarchically and corresponding feature maps of the same shape as the upsampled volume are concatenated with it. The concatenation is done using skip connections from the encoder to the decoder. These connections facilitate better gradient and information flow through the network and also help alleviate the problem of vanishing gradients in a network as deep as this. Refer to Figure B.1 for more details. Each upsampling operation is followed by two residual blocks. The final output of the network is a 1x1 convolution with sigmoid activation. This is because we treat the landmark localization problem as a two class segmentation problem and are interested in localizing the location of the landmarks at this step, and not predicting it exactly. Dropout is used in the fourth layer of the encoder and in the bottleneck layer as a means to avoid overfitting on the data. The pool size for the max pooling operation is 2x2. A LeakyReLU activation function is used throughout.

B.5.2 Attention Implementation

We implement the attention network using the Keras deep learning library with a tensorflow backend. The network is trained on 2 Nvidia GTX 1080Tis with a batch size of 32. We use Stochastic Gradient Descent with nesterov momentum as the optimizer, with an initial learning rate of 0.03, and the default momentum value. The learning rate is reduced on a plateau to half its original value if the change in the validation loss is less than 0.001 over 5 epochs. Early stopping is used to stop the network if there is no change in the validation loss over 10 epochs. We train the network using both binary cross entropy and the negative of the dice coefficient, given by,

$$Negative\ Dice\ Loss\ =\ -\frac{2 * |X \cap Y|}{|X| + |Y|} \tag{B.5.1}$$

as the loss function and monitor the change in the dice coefficient and jaccard index values as our evaluation metrics. We observe that using 1 - dice coefficient leads to a slightly more stable convergence towards the optimal value, so we choose to employ this in the final network. The network converges in approximately 90 epochs.

B.5.3 Landmark Regressor

We rely on our attention mechanism to guide the training of our regressor, through a combination of our guided attention mechanism, coupled with the self attention of the SE block. Our final end-to-end architecture contains a frozen attention network providing the attention maps to the trainable regressor, which then uses the information to learn the coordinates of the landmarks. The regressor is based on the ResNet-18 architecture with feature map weighting. The input is passed through a `conv-batchnorm-activation` block. Following this, information is propagated further by first passing it through a full preactivation, no bottleneck residual block, followed by which, attention maps are concatenated with the feature maps. This resultant volume undergoes a feature map weighting via the SE block. This is then passed through two layers of `conv-batchnorm-activation`. Finally, a last residual block is used with strided convolutions to process and downsample the input. Four such blocks are stacked one after the other to process the input following which, an average pooling layer is used. We do not use any fully connected layers. The output is taken from a dense layer with a `linear` activation. No max pooling is used as the feature maps are downsampled using strided convolutions. ReLU activations are used throughout. Further information about convolution kernel size, filters used, and a detailed look at the architecture of the regressor can be obtained from Table I.

B.5.4 Feature Map Weighting

Instead of incorporating attention by simply concatenating the attention maps into our network, we train our network to identify which maps are important and should be propagated further through the network, by learning a weighting for each individual feature map in the concatenated volume. Given $H \times W \times C_1$ and $H \times W \times C_2$, as the two volumes,

$$c = H \times W \times C,\ where,\ C = C_1 + C_2 \tag{B.5.2}$$

denotes the channel wise concatenation of the feature maps of the two architectures. This weighting can be seen as a self-attention mechanism used by the network. We

learn this weighting using the *'squeeze-and-excite'* block [36]. The *squeeze* block squeezes the global spatial information into a channel descriptor. To achieve this, global average pooling is is used.

$$s_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} c_c(i,j) \tag{B.5.3}$$

Where s_c is the channel-wise statistic for the concatenated feature volume. The squeeze block tackles the issue of convolution filters not being able to exploit information beyond their receptive fields. The *excite* block captures channel wise dependencies.

$$e = \sigma(g(\mathbf{s}, \mathbf{W})) \tag{B.5.4}$$

Where e is the channel-wise weights and σ is the gating function. W is the combination of the weights of the dense layers of the block. This is followed by channel-wise re calibration of the volume.

$$x_c = e_c \cdot c_c \tag{B.5.5}$$

B.5.5 Wing Loss

We experimented with the Huber loss and the wing loss [26] as the loss functions to train our network. The wing loss converged to a smaller error in the evaluation metric and was hence used as the loss function of choice. Wing loss is given by the formula,

$$wing(x) = \begin{cases} w \ln(1 + |x|/\epsilon) & |x| < w \\ |x| - C & \texttt{otherwise} \end{cases} \tag{B.5.6}$$

where $C = w - w \ln(1 + w/\epsilon)$. The function behaves as the Huber loss function for large errors and penalizes the smaller losses as a log loss. By penalizing the smaller losses via the log loss, the influence of small errors is increased and it empirically leads to convergence to a smaller evaluation metric value. w *and* ϵ are the hyper parameters in the loss function. We use the default values presented in [] and set $w = 10$ *and* $\epsilon = 2$. We believe that the wing loss outperforms the Huber loss in our problem due to its ability to handle smaller errors better.

B.5.6 Attention Landmarker Implementation

To train our system end-to-end, we first initialize the weights of the attention architecture with weights from pre-training the network. We then freeze all the layers of this network (including the batch normalization layers throughout the attention network) so that we can not alter the output of this network in any way. We train the network on 4 Nvidia GTX 1080Tis with a batch size of 32. We use wing loss as the loss function and monitor the mean absolute error between the landmark coordinates as our evaluation metric. We train this network with Stochastic Gradient Descent with Nesterov momentum as the optimizer with an initial learning rate of 0.03, and the default momentum value. Just like the attention network, we reduce the learning rate on a plateau, monitoring the change in validation loss over 5 epochs. Early stopping is also used. The network converges in about 50 epochs.

B.6 Experiments

We conducted a series of experiments to modify the architecture of the ResNet-18 to find the best way to incorporate attention into the architecture. The following section summarizes the experiments. We empirically show that our method is the best attention mechanism over those tested. The following experiments are conducted on a subset of the training data. Here, we used 12000 images for the training set and 3000 for the validation set. We judge the performance of the network based on the the smallest value of validation loss, corresponding to which we report the value of the evaluation metric (mean absolute error/validation dice coefficient).

B.6.1 Convolution Attention vs Residual Attention

We compare the performance of a simple convolution based autoencoder, with our residual convolutional autoencoder architecture for generating attention maps. Both architectures are built and trained in the same style as the attention architecture described in Section IV.A. The only difference is that the residual blocks are replaced by `conv-ReLU` blocks throughout the architecture. The residual attention

architecture outperforms its counterpart considerably as seen in Table B.1.

Just Conv Attention	Residual Attention
0.8004	**0.8946**

Table B.1: Dice loss for the two attention networks. Residual blocks considerably improve the accuracy of the network

B.6.2 Attention vs No attention

We train the ResNet-18 [34] architecture on our data for regressing the landmark coordinates. The entire network is the same as the original ResNet-18, except the output of the solitary dense layer uses a linear activation. The attention architecture we compare it with is our final architecture with the SE block followed by a bottleneck layer. The results can be seen in Table B.2.

B.6.3 Different ways to attend

We tried different styles of blocks to create our regressor network. The blocks used can be seen in Figure B.3.

B.6.3.1 Concat vs Concat with SE

Experiments were conducted to see whether just concatenating the attention maps gave good results compared to concatenating the maps and following them with a SE block. The concatenate-with-SE approach proved superior, suggesting that the SE block plays an influential role when it comes to improving the accuracy of the network. This experiment showed that weighting of the feature volume leads to improved results.

B.6.3.2 Different attention blocks

We experimented with bottleneck blocks and sequential information flow through the network. We added these blocks to improve the generalization ability of the

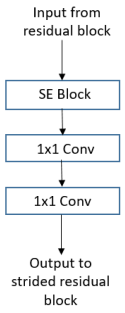
network of the network. A bottleneck block contains a 3x3 convolution sandwiched between two 1x1 convolutions and have a computation complexity similar to two 3x3 convolutions. We started our search for the ideal attention block by using 1x1 convolutions after the SE block, all the way to having a bottleneck separable convolution block. The different blocks tested are shown in Figure B.3. The results of the experiments are shown in Table B.2. We observed that the bottleneck block along with the sequential 3x3 convolution block provide comparable results. We use the bottleneck block in our final architecture as it provides comparable results with lesser parameters.

Method	Error
Resnet18(No Attention)	0.8673
Just Concat	0.8112
Concat with SE	0.7712
SE with 2 1x1 Conv	0.6094
SE with 2 1x1 Seprable Conv	0.5793
SE with 2 3x3 Conv	**0.5445**
SE with 2 3x3 Seprable Conv	0.5689
SE with bottleneck	0.6154
SE with Separable bottleneck	0.5897
Resnet50	0.5904

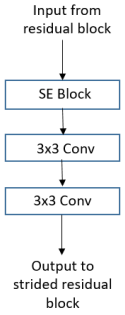
Table B.2: Mean Absolute Error loss for the different regressor networks

B.6.3.3 Our method vs Gating and multiply

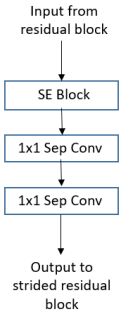
Our final experiment included comparing our final architecture with a gate-and-multiply mechanism. For this, we pass the intermediate outputs of the autoencoder layers through a gating function (sigmoid) to obtain a mask. We pass the output of the first residual block through a SE block, following which, we multiply it with the output from the gating function. the rest of the architecture (SE followed by bottleneck) remains the same. This method gives better results as our method but



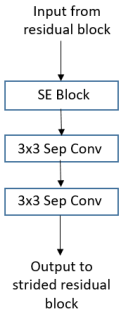
(a) Two 1x1 conv



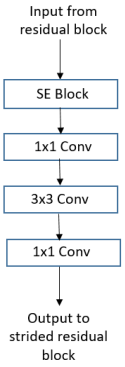
(b) Two 3x3 conv



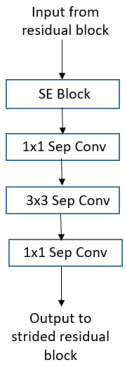
(c) Two 1x1 sep conv



(d) Two 3x3 sep conv



(e) Conv bottleneck



(f) Sep conv bottleneck

Figure B.3: Different blocks tested as possible candidates for the final architecture.

takes longer to converge due to the stronger inductive bias applied via the gating function.

B.6.4 Our Attention Mechanism vs ResNet-50

To see how our attention mechanism fairs against deeper residual nets, we trained a 50 layer resnet with full pre-activation no bottleneck residual blocks, passing the

output of the dense layer through a linear activation function. Our attention mechanism outperforms the resnet50 by 7%.

B.7 Results

We compare the performance of our network with the landmark localization accuracy of the DLIN and the FCN and report results on the FRGCv2 dataset. It can be seen that our network outperforms the DLIN in terms of overall mean error by about 13% and the FCN by about 16%. It is also useful to note that the DLIN was run for 200 epochs over 3 days, while our network gives optimal performance in a shorter time. Table B.3 shows the landmark localization error in millimetres for the 9 landmarks along with the overall mean error.

B.8 Conclusions

In this chapter, we demonstrated the use of attention for landmarking 3D faces. We integrated feature maps from one network into another using a concatenation operation and then used a *'Squeeze-and-Excite'* block to learn a weighting of the volume. We provided an analysis of our technique showing how we finalized our architecture choice and compared the performance of our network with the DLIN, in terms of performance on the FRGCv2 dataset. Our network outperforms DLIN and ResNet-50, a network over twice the depth. The residual connections facilitate better gradient flow through the network, allowing us to train deeper networks giving better accuracy on the dataset.

Mean localization error (mm)										
Method	LEOC	LEIC	REOC	REIC	NT	NCL	NCR	MLC	MRC	Mean
Perakis *et al.* [66] [SIEM-NP]	7.18	6.85	6.92	6.18	6.32	-	-	7.84	7.42	6.95
Perakis *et al.* [66] [SISI-NP]	5.81	5.59	5.87	5.19	5.28	-	-	6.47	5.71	5.70
Perakis *et al.* [66] [SISI-NPSS]	5.70	5.52	5.82	5.10	4.88	-	-	6.42	5.64	5.61
FCN [79]	2.9	-	-	-	2.4	3.5	-	3.0	-	3.1
DLIN [31]	2.8	-	-	-	2.4	3.4	-	2.7	-	3.0
Ours	2.6	2.4	2.5	2.4	2.2	2.6	2.7	2.2	2.3	**2.5**

Table B.3: Results of landmark localization on scans of the FRGCv2 dataset.The FCN and DLIN results are an average of the landmarks occurring in pairs.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Nabila Abraham and Naimul Mefraz Khan. A novel focal tversky loss function with improved attention u-net for lesion segmentation. *CoRR*, abs/1810.07842, 2018.

[3] Md. Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M. Taha, and Vijayan K. Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *CoRR*, abs/1802.06955, 2018.

[4] Mohamed Attia, Mustafa Hossny, Saeid Nahavandi, and Anousha Yazdabadi. Spatially aware melanoma segmentation using hybrid deep learning techniques. *CoRR*, abs/1702.07963, 2017.

[5] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Trans. Graph.*, 37(4):71:1–71:12, July 2018.

[6] Reza Azad, Maryam Asadi-Aghbolaghi, Mahmood Fathy, and Sergio Escalera. Bi-directional convlstm u-net with densley connected convolutions, 2019.

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[8] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. *CoRR*, abs/1904.11492, 2019.

[9] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *CoRR*, abs/1711.01731, 2017.

[10] Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. ABC-CNN: an attention based convolutional neural network for visual question answering. *CoRR*, abs/1511.05960, 2015.

[11] François Chollet et al. Keras. https://keras.io, 2015.

[12] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[13] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[14] Dan Ciresan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc., 2012.

[15] Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin K. Mishra, Harald Kittler, and Allan Halpern. Skin lesion analysis toward

melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (ISIC). *CoRR*, abs/1710.05006, 2017.

[16] Cristina Conde, Roberto Cipolla, Licesio J. Rodríguez-Aragón, Ángel Serrano, and Enrique Cabello. 3d facial feature location with spin images. In *MVA*, 2005.

[17] Clement Creusot, Nick Pears, and Jim Austin. A machine-learning approach to keypoint detection and landmarking on 3d meshes. *International Journal of Computer Vision*, 102(1):146–179, Mar 2013.

[18] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[19] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. *CoRR*, abs/1802.01500, 2018.

[20] Fabian Isensee et. al. nnu-net: Self-adapting framework for u-net-based medical image segmentation. *CoRR*, abs/1809.10486, 2018.

[21] J.J. Staal et. al. Ridge based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, 2004.

[22] Noel C. F. Codella et. al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (ISIC). *CoRR*, abs/1902.03368, 2019.

[23] Saeid Asgari Taghanaki et. al. Combo loss: Handling input and output imbalance in multi-organ segmentation. *CoRR*, abs/1805.02798, 2018.

[24] Tsung-Yi Lin et. al. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[25] Wentao Zhu et. al. Anatomynet: Deep 3d squeeze-and-excitation u-nets for fast and fully automated whole-volume anatomical segmentation. *CoRR*, abs/1808.05238, 2018.

[26] Zhen-Hua Feng et. al. Wing loss for robust facial landmark localisation with convolutional neural networks. *CoRR*, abs/1711.06753, 2017.

[27] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[28] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4476–4484, 2017.

[29] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980.

[30] Thomas Gerig, Andreas Forster, Clemens Blumer, Bernhard Egger, Marcel Lüthi, Sandro Schönborn, and Thomas Vetter. Morphable face models - an open framework. *CoRR*, abs/1709.08398, 2017.

[31] Syed Zulqarnain Gilani, Ajmal Mian, and Peter Eastwood. Deep, dense and accurate 3d face correspondence for generating population specific deformable models. *Pattern Recognition*, 69:238 – 250, 2017.

[32] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNET: learning local shape properties from raw point clouds. *CoRR*, abs/1710.04954, 2017.

[33] Hassan Hassan, Abdelazim Negm, Mohamed Zahran, and Oliver Saavedra. Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: Case study el burullus lake. *International Water Technology Journal*, 5, 12 2015.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[36] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 7, 2017.

[37] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[38] Yani Ioannou, Duncan Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving cnn efficiency with hierarchical filter groups. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[39] Yani Ioannou, Duncan P. Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving CNN efficiency with hierarchical filter groups. *CoRR*, abs/1605.06489, 2016.

[40] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.

[41] Qiangguo Jin, Zhaopeng Meng, Tuan D. Pham, Qi Chen, Leyi Wei, and Ran Su. Dunet: A deformable network for retinal vessel segmentation. *Knowledge-Based Systems*, 178:149–162, Aug 2019.

[42] Kun Jing and Jungang Xu. A survey on neural network language models. *CoRR*, abs/1906.03591, 2019.

[43] Andrew Edie Johnson. Spin-images: A representation for 3-d surface matching. Technical report, 1997.

[44] M. Juneja, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 923–930, June 2013.

[45] C. Kaul, S. Manandhar, and N. Pears. Focusnet: An attention-based fully convolutional network for medical image segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 455–458, April 2019.

[46] Chaitanya Kaul, Nick Pears, and Suresh Manandhar. Divided we stand: A novel residual group attention mechanism for medical image segmentation, 2019.

[47] Chaitanya Kaul, Nick Pears, and Suresh Manandhar. Penalizing small errors using an adaptive logarithmic loss, 2019.

[48] Chaitanya Kaul, Nick Pears, and Suresh Manandhar. Sawnet: A spatially aware deep neural network for 3d point cloud processing, 2019.

[49] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[50] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[51] Roman Klokov and Victor S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. *CoRR*, abs/1704.01222, 2017.

[52] Jan J Koenderink and Andrea J van Doorn. Surface shape and curvature scales. *Image and Vision Computing*, 10(8):557 – 564, 1992.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[55] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. *CoRR*, abs/1803.04249, 2018.

[56] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *CoRR*, abs/1801.07791, 2018.

[57] Yuexiang Li and Linlin Shen. Skin lesion analysis towards melanoma detection using deep learning network. *CoRR*, abs/1703.00577, 2017.

[58] Xiao Liu, Tian Xia, Jiang Wang, Yi Yang, Feng Zhou, and Yuanqing Lin. Fully convolutional attention networks for fine-grained recognition, 2016.

[59] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[60] Kevin Mader. Finding and measuring lungs in ct data. https://www.kaggle.com/kmader/finding-lungs-in-ct-data/home.

[61] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*, 2015.

[62] Martina Melinscak, Pavle Prentasic, and Sven Loncaric. Retinal vessel segmentation using deep neural networks. In *VISAPP*, 2015.

[63] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018.

[64] Américo Oliveira, Sérgio Pereira, and Carlos A. Silva. Retinal vessel segmentation based on fully convolutional neural networks. *Expert Systems with Applications*, 112:229 – 242, 2018.

[65] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[66] P. Perakis, G. Passalis, T. Theoharis, and I. A. Kakadiaris. 3d facial landmark detection under large yaw and expression variations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1552–1564, July 2013.

[67] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, abs/1711.08488, 2017.

[68] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[69] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. *CoRR*, abs/1604.03265, 2016.

[70] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[71] Joseph Redmond. Ancient secrets of computer vision. https://pjreddie.com/courses/computer-vision/.

[72] M. Romero-Huertas and N. Pears. 3d facial landmark localisation by matching simple descriptors. In *2008 IEEE Second International Conference on Biometrics: Theory, Applications and Systems*, pages 1–6, Sep. 2008.

[73] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[74] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. Concurrent spatial and channel squeeze excitation in fully convolutional networks, 2018.

[75] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. Tversky loss function for image segmentation using 3d fully convolutional deep networks. In *Machine Learning in Medical Imaging*, pages 379–387, Cham, 2017. Springer International Publishing.

[76] Marcelo Sardelich and Suresh Manandhar. Multimodal deep learning for short-term stock volatility prediction, 2018.

[77] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 357–360. ACM, 2007.

[78] Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization, 2014.

[79] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, April 2017.

[80] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017.

[81] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[82] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[83] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[84] L. T. Thao and N. H. Quang. Automatic skin lesion analysis towards melanoma detection. In *2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES)*, pages 106–111, Nov 2017.

[85] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[86] Xijun Wang, Meina Kan, Shiguang Shan, and Xilin Chen. Fully learnable group convolution for acceleration of deep neural networks. *CoRR*, abs/1904.00346, 2019.

[87] Xinyao Wang, Liefeng Bo, and Fuxin Li. Adaptive wing loss for robust face alignment via heatmap regression. *CoRR*, abs/1904.07399, 2019.

[88] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[89] Hongdiao Wen. II-FCN for skin lesion analysis towards melanoma detection. *CoRR*, abs/1702.08699, 2017.

[90] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014.

[91] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. *CoRR*, abs/1411.6447, 2014.

[92] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.

[93] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. *CoRR*, abs/1803.11527, 2018.

[94] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6):210:1–210:12, November 2016.

[95] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[96] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc., 2017.

[97] Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions for deep neural networks. *CoRR*, abs/1707.02725, 2017.

[98] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017.

[99] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *CoRR*, abs/1805.07836, 2018.

[100] Bo Zhao, Xiao Wu, Jiashi Feng, Qiang Peng, and Shuicheng Yan. Diversified visual attention networks for fine-grained object classification. *CoRR*, abs/1606.08572, 2016.

[101] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. *CoRR*, abs/1807.10165, 2018.