Deep Complex-Valued Neural Networks for Natural Language Processing

Nils Mönning

Doctor of Philosophy

University of York
Computer Science
March 2019

To my parents, Anke and Ludger, my brother Sascha and my love Sarah

Abstract

This thesis presents novel work on complex-valued neural networks applied to Natural Language Processing. We experimentally show the validity of complex-valued neural networks for semantic and phonetic processing of natural languages. We highlight important issues that complex networks have in comparison to their real-valued counter parts. In particular this work considers the tasks of Language Modelling, Semantic Similarity Judgement, Basic Question Answering, Phonetic Transcription and Automatic Speech Recognition.

Our contributions are the translation of neural network building blocks to the complex plane and their experimental application in a variety of natural language tasks.

We present criteria to compare real-valued and complex-valued neural networks for classification tasks. We present various complex embedding methods for words. These produce position and frequency-based word representations trainable using language models and usable in downstream tasks. We also compare a real-valued and complex-valued memory network used for Question Answering. We derive a quantum-inspired framework for languages. Additionally, we demonstrate quantum-inspired Semantic Spaces. A general framework of complex-valued attention is presented in this thesis. It is used to derive spectral self-attention with a novel activation function. We also introduce two pooling functions to reduce dimensionality of frequency-based representations. A Spectral Transformer architecture facilitates the spectral self-attention for Speech Recognition. This work also includes a novel dataset for transcription of children's utterances consisting of seven sub tasks each with fixed data splits and baselines for better comparison and reproducibility.

Throughout this thesis we find that complex-valued neural networks are suitable for natural language tasks, but require additional care in their design and training.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my examiners Prof Dr Dawei Song, Dr James Cussens and Dr Tangming Yuan, my supervisor Dr Suresh Manandhar and my colleagues Dr Sarah Spooner, Dr Alexandros Komninos, Chaitanya Kaul, Marcelo Sardelich, Taghreed Alqaisi, Di Wang and Mudita Sharma for their time, feedback and discussions. Particular thanks go to the Engineering and Physical Sciences Research Council (EPSRC) for supporting my work with a Doctoral Training Grant. Thank you to Dr Tamar Keren-Portnoy, Dr Helena Daffern, Kenneth Brown, Prof Dr Marilyn May Vihman and Prof Dr Rory A. DePaolis for providing me with valuable data and even more valuable advice. I would also like to thank the administrative staff of the Computer Science Department and the University of York.

Thank you all for your support and efforts that allowed me to learn and grow.

Nils Mönning
March 2019, York, England

Declaration

I declare that this thesis is a presentation of original work and is the result of my own investigations, except where otherwise stated. All other sources are explicitly acknowledged by references. This work has not previously been presented for an award at this, or any other, University.

Signed .....

Date .....

Some of the material contained in this thesis has appeared in the following published paper, paper awaiting publication or paper currently in peer review:

1. Nils Mönning, Suresh Manandhar (2018). Evaluation of Complex-Valued Neural Networks on Real-Valued Classification Tasks. CoRR, abs/1811.12351.

CHAPTER 1

Introduction

Machine Learning (ML) has become part of our day-to-day lives. Humans use speech recognition and generation systems to communicate with virtual personal assistants, semantic search to explore large data collections to find relevant documents, use translation engines to translate texts from one language to another or recommender systems to sift out irrelevant product reviews while shopping online. The field of Machine Learning provides methods to learn parameters of unknown functions to solve certain tasks. This research field aims to derive problem solutions directly from training data without explicitly programming this solution.

The area of Natural Language Processing (NLP) comprises tasks that require natural language understanding. NLP uses ML solutions to solve tasks that require a machine to understand, interpret or transform written or spoken language. Many attempts have been made to model natural languages. In the past the most common solutions include linguistic rule-based or statistical systems with handcrafted features.

Deep Learning (DL) has achieved significant improvements over these work-intensive solutions. DL is a sub field of ML. It is concerned with the study of neural networks and their application to various problems. Neural networks have been used to achieve state-of-the-art performance in many tasks. Image processing and language processing have particularly benefited from the development of DL. Deep neural networks are capable of learning and discovering important features in the data. This ability allows most models to achieve better results than many handcrafted and rule-based systems. To find these features they usually require large amounts of data. In DL, different building blocks (or neural layers) are used to extract these features. Recurrent networks learn representations accumulated over sequences, convolutions allow learning

of context sensitive features while, multi-layer perceptrons learn to weigh and combine features. These building blocks are combined to form complex and layered network architectures. Each building block represents a set of functions applied to the input data or the output of the previous building block. The training data is propagated forward through the layers. At the end, the model's output is used to compute a loss between a target output and the systems output or scored towards another training objective. This loss is then backpropogated through the layers updating parameters according to the loss using gradient-based methods.

In complex-valued neural networks the layers receive complex-valued data, contain complex-valued parameters or transform the data into complex-valued representations. They have mainly received attention in relation to Signal Processing. Literature suggests to use them due to their representative abilities of frequency information. However, the application of complex numbers can be motivated from many perspectives:

- Using complex numbers increases the expressibility of functions, since any polynomial can be factorised in the complex domain.
- Quantum theory allows a probabilistic and geometrical interpretation of complex-valued quantum states.
- Complex-valued frequency representations are fundamental to the Fourier analysis of signals. Recent trends in NLP encourage new types of embeddings.
- Existing literature supports the use of complex-valued convolutions as natural use cases.

1.1 Context

This work intersects the areas of Complex-Valued Data Estimation, Signal Processing and Deep Learning. We focus on natural language as a possible application. We show that it can be modelled from a semantic and phonetic perspective using complex-valued neural networks. We are particularly interested in finding natural language tasks for which complex-valued neural networks improve performance and for which they do not. While it seems to be easy to combine existing approaches to design a large complex-valued architecture which outperforms an arbitrary architecture, we focus on comparable architectures. We are able to highlight the differences between real and complex numbers in neural network. We experimentally show the validity of complex-valued architectures, but also highlight important issues. Complex numbers suffer similar problems to their real-valued counterparts, but require different solutions.

Furthermore, we point out the tasks in which the addition of complex numbers can improve performance and where they do not. The tasks we consider are Language Modelling, Semantic Similarity Judgement, Basic Question Answering, Phonetic Transcription and Automatic Speech Recognition. We approach these tasks mainly with statistical unsupervised or labelled supervised

approaches. In particular, we consider written and spoken language from various sources (books, internet crawls, audio books, curated news articles and handcrafted questions). Our approaches make use of existing publicly available data. Where we used unpublished data we are able to provide the data. We focus on supervised training to solve the considered tasks.

Our experiment code is written in Python 3.5+ and were computed using TensorFlow [8], Theano [9], Keras [10] and Tensor2Tensor [11] on an NVidia CUDA GPU. Even with this hardware many of the proposed models of this thesis trained for many hours, days and even weeks. Presented baselines have undergone extensive hyperparameter optimisation and hence should represent stable, reliable baselines. For our own models, however, little hyperparameter optimisation was possible, due to time limitations on this project.

1.2 Research Questions

Our research objective was to investigate the use of complex-valued neural networks for natural language tasks. This thesis aims to find natural language tasks which benefit from replacing real numbers with complex numbers in neural networks and explain their behaviour. We focus on modelling compositional language from a semantic and phonetic perspective. We present a diverse set of tasks and various data types. This allows us to give a greater overview of the usefulness of complex numbers for Natural Language Processing.

From various observations we derive specific research questions and experimentally find answers for them. These are outlined below:

- **Does the use of complex-valued neural networks improve performance for classification tasks by increasing the expressibility of the model?**

  Complex numbers generalise real numbers by extending them with the possibility to express roots of negative values with the *imaginary unit* $i^2 = -1$. Consequently, the real numbers $\mathbb{R}$ form a subset of the complex numbers $\mathbb{C}$ in which the imaginary part is zero. Complex numbers $\mathbb{C}$ satisfy many important properties (e.g. commutativity, associativity) that are also satisfied by the real numbers. This extension with an imaginary part and the similar properties support a larger expressibility compared to real numbers. This is often exploited in factorisation techniques. Examples are complex eigenvalues and factorisation of polynomials.

  Complex numbers can be represented by a real and an imaginary part or by a magnitude and a phase. The magnitude indicates the distance to the origin while the phase indicates its direction from the origin. Phases may help to learn structured interactions between parameters and representations which are difficult to learn with real numbers. An existing example is the XOR function. Additional expressibility may be used to encode and learn positional or structural information of the data or to learn factorisations of the input data.

However, this may also be detrimental to the learning process as finding a local or global optimum is more difficult in a complex set than in the real set.

To answer the above research question reliably we need to design comparisons with an equal number of real-valued parameters. If the models are of drastically different size, the performance difference may be a result of model size and not the increased expressibility and structure of complex numbers. Moreover, the network architectures need to be comparable by simply replacing real numbers with complex numbers.

- **Does encoding additional positional or frequency information in imaginary parts or phases improve the performance in sequence-based language processing tasks?**

  A recent trend in natural language processing is to learn dense representations of words within real-valued semantic spaces. Embeddings learn to encode the context and co-occurrence information of each word as a fixed set of features. They often implicitly facilitate statistics or information across the entirety of the data. Word embeddings also are an integral part of natural language systems tasks. For language it is fundamental to include compositionality in order to compose word representations into phrase or sentence representations. Various improvements of word embeddings towards compositionality have been made. They enrich embeddings with additional information like dependency statistics or positions. Enriched embeddings have been used to improve performance in sentence-based tasks. Complex-valued embeddings offer more possibilities to encode structured interaction.

  An alternative to complex-valued embeddings are transformations. Transformations can map real (or complex) data to the complex plane. Most frequently used is the (Discrete) Fourier Transformation which decomposes real or complex sequences into its component sine and cosine waves. The complex-valued frequency domain allows us to gain insights over the composing waves in a very compact way. Word representation in the frequency domain may give a better understanding of word statistics and can be easier to learn while containing more information.

- **What advantages and differences are there in using quantum-inspired models of language compared to non-quantum complex-valued models of language?**

  In Quantum Theory, complex vectors represent the state of quantum systems. Previous work has suggested using quantum states to represent words. Quantum-inspired word representations go one step further than complex-valued word representations. They act in the same vector spaces (concretely a Hilbert space), but require additional constraints on the vector space in the training process. These constraints give quantum states, and quantum word embeddings, a geometrical and probabilistic interpretation. Learning *normal* complex-valued embeddings relaxes these constraints. The differences these con-

straints introduce to the representation and learning process are not well studied. In both cases we can learn complex-valued word embeddings that can be used in various natural language tasks.

- **Can we move the entire neural model into the frequency domain to improve speech transcription and simplify computation?**

  If we move the entire learning process into the frequency domain, we are able to learn transformations of frequencies and frequency-based representations. Spoken language, which naturally consists of different constituent sine and cosine waves, can processed directly in the frequency domain. Most speech recognition approaches use frequency-based preprocessing (e.g. Mel-Frequency filter banks or Mel-Frequency Ceptrum Coefficients). These methods, however, discard the important phase information. Learning in the frequency domain in an end-to-end setting may improve performance in speech-related tasks. Further, convolutions can be replaced with complex-valued fully-connected layers in the frequency domain. In consequence, the layer outputs do not need to be transformed back and forwards for every layer. This potentially improves computation in the training and inference, too.

1.3 Contributions

This thesis makes the following contributions to existing literature:

- A translation of real-valued multi-layer perceptrons (MLP), recurrent and convolutional layers to the complex plane that we use in this work. These can be used as drop-in replacements.
- Criteria to compare real-valued and complex-valued MLPs for classification tasks.
- Embedding methods for words to receive position and frequency-based representations usable in down-stream NLP tasks.
- A complex-valued memory network used for Question Answering which shows us which sub tasks benefit from complex numbers.
- A general framework of quantum-inspired Semantic Spaces and instances of *npmi*- and *pmi*-based Semantic Spaces.
- A data set for transcription of children's utterances with fixed data splits and baselines for better comparison.
- A general framework of complex-valued attention that is used to derive spectral self-attention with a novel activation function. We also introduce two pooling functions to reduce dimensionality of frequency-based representations.

- Spectral Transformers architecture which facilitate spectral self-attention in Speech Recognition.

1.4 Chapter Overview

Chapter 2 introduces mathematical background and definitions. We discuss complex numbers and their representations. We introduce neural networks and their training in the complex plane.

Chapter 3 compares multi-layer perceptrons and convolutional neural networks with complex- and real-valued weights using the same number of real-valued parameters. We demonstrate the performance of different activation functions and depths on various benchmark classification tasks: MNIST [12], Reuters newswire classification [13], CIFAR-10 and CIFAR-100 [14]. The input data is not preprocessed, transformed or embedded into the complex plane, hence its imaginary parts $Im(x)$ are zero.

We find that real-valued neural networks pose an upper performance boundary for complex-valued neural networks of the same size when used on unprocessed real-valued data. We explain this behaviour with the interaction of weights and emphasise the importance of embeddings and prepossessing. We also find that initialisation and training are significant problems when training complex-valued neural networks.

Chapter 4 proposes and investigates complex-valued embeddings and transformations of words as well as semantic vectors. These approaches follow the Distributional Hypothesis [15] and Compositionality [16].

We train six language models on the Penn-Tree-Bank (PTB). We report on two baselines, two positional models and two frequency-based models. Perplexity shows that frequency information can improve n-gram language modelling. The learned word embeddings of proposed neural language models can be used in many down-stream tasks.

Inspired by Weston et al. [17] we introduce a simple complex-valued memory network which achieves better or similar performance using fewer parameters in many bAbI question answering tasks. Especially in tasks that require longer dependencies.

We present a generalised quantum-inspired framework of semantic spaces. Inspired by the findings from Levy et al. [18] we manually design Quantum-inspired semantic spaces. These semantic vectors respect the two main constraints of quantum states (amplitude norm $\sum |z_j|^2 = 1$ and orthogonality $\langle \phi | \psi \rangle = 0$). These two constraints give semantic vectors a probabilistic and geometrical interpretation. We test quantum-inspired semantic vectors on similarity judgment tasks.

Chapter 5.5 introduces a novel data set for the phonetic transcription of child utterances. Raw audio signals are transcribed with the symbols of the International Phonetic Alphabet (IPA). Hence, the transcription tasks are related to phoneme-level speech recognition.

This work is a collaboration of various departments at the University of York. The raw recordings, labels and a linguistic definition of what constitutes babble have been provided by Tamar Keren-Portnoy, Helena Daffern and Kenneth Brown. Our contribution, which is included in this thesis, is the cleaning and normalising of the labels, designing sub tasks for phonetic transcription and classification, the computation of baselines and development of a code repository.

The 7 sub tasks include 3 transcription task with a full set of available IPA symbols, 3 transcription tasks with a reduced set of IPA symbols and a binary classification task. As baselines we chose real-valued transformer architectures in 2 sizes. Further baselines are easily computed as the code repository uses the Tensor2Tensor API [11].

Chapter 5 presents a generalised attention framework, derives a novel spectral self-attention mechanism and introduces a novel activation function. The spectral self-attention can be used as scaling mechanism of frequency representations or as frequency-based extension in real-valued architectures.

We use these building blocks to define Spectral Transformer architectures that can learn in the real or complex domain. We test this new architecture on the LibriSpeech Autmomatic Speech Recognition [19] task and the Babble data set introduced in Chapter 5.5.

Chapter 6 summarises our findings and concludes our work.

In addition to the content described above, every chapter contains a review of related literature relevant to that chapter.

CHAPTER 2

Theoretical Background

This chapter introduces the mathematical background, definitions and terminology used throughout this thesis. In the following sections we will introduce complex-valued generalisations of well-established building blocks of DL. We consider these in relation to complex numbers, complex differentiability and their use in (deep) neural architectures.

In contrast to classical Machine Learning models, Neural Networks (NN) are capable of automatically discovering features. They do not need to be hand-crafted, but rely on non-linearities and a combination of input features. Modern neural architectures consists often many layers with the intermediate application of non-linear activation functions. Using complex numbers in weights and input data, a specific problem may become solvable with simpler architectures or fewer parameters. A simple example is the XOR function [20, 21].

We use the theoretical background developed in this chapter as a foundation for experiments which investigate our research questions presented in Chapter 1. The definitions for complex numbers and neural networks are used throughout this thesis. Gradients, activation and loss functions are also used and discussed in all chapters. Convolutions are used in Chapter 3 in classification experiments. Chapter 4 includes architectures that use transformations and embeddings to create frequency-based language models. We also use recurrence in language models and memory models. The developed and presented generalisation of the attention mechanism and the Convolutional Theorem is used in Chapter 5 to define a novel transformer architecture. With our approach to transformers we can implicitly implement convolutions in the frequency domain efficiently.

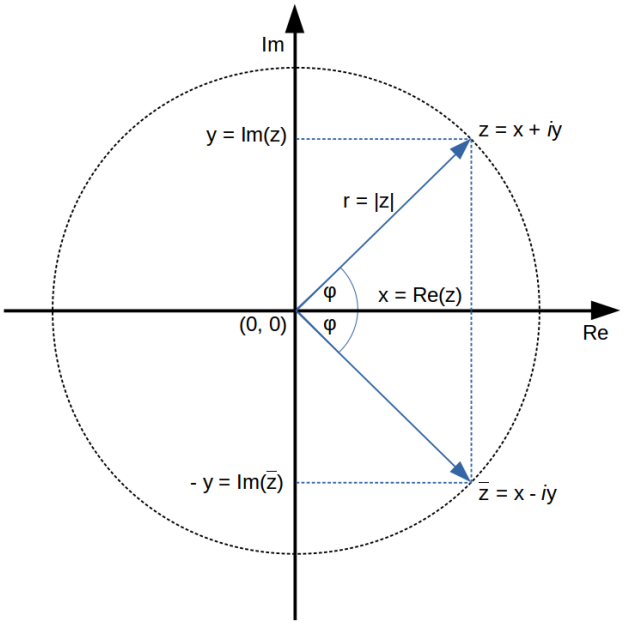

Figure 2.1: Visual representation of the complex plane.

2.1 Complex Numbers

The set of complex numbers $\mathbb{C}$ is a generalisation of real numbers $\mathbb{R}$. It defines an imaginary unit $i^2 = -1$ to express the roots of negative real numbers. Any complex number can be defined as $z = x + iy = re^{i\varphi}$. They are defined by a pair of (x, y) and (r, φ). As Cartesian coordinates they are given with a real part $Re(z) = x$ and an imaginary part $Im(z) = y$. As polar coordinates they are given with their magnitude $r = |z| = \sqrt{x^2 + y^2}$ and phase $\varphi = arctan(\frac{x}{y})$ with $\varphi \in [-\pi, \pi]$ using Euler's formula (Equation 2.1). The conjugate of a complex number $\bar{z} = x - iy = r \cdot e^{-i\varphi}$ changes the sign of the imaginary part.

$$e^{i\varphi} = \cos(\varphi) + i \sin(\varphi) \tag{2.1}$$

The set of complex numbers can be visualised as a plane (Figure 2.1). A complex-valued function $f : \mathbb{C} \to \mathbb{C}$ of a complex-variable can be expressed as a function of two real variables $f(z) = f(x, y) = f(r, \varphi)$.

The magnitude is defined as the distance to the origin:

$$\begin{aligned} |z| &= \sqrt{x^2 + y^2}, \\ |z|^2 &= z\bar{z} = x^2 + y^2 \end{aligned} \tag{2.2}$$

The trigonometric functions in the complex plane are defined by:

$$\begin{aligned} \sin(\varphi) &= \frac{y}{r}, \\ \sin(\varphi) &= \frac{x}{r}, \\ \tan(\varphi) &= \frac{y}{x} \end{aligned} \tag{2.3}$$

We can use these functions to find the Cartesian coordinates:

$$\begin{aligned} y &= r\sin(\varphi), \\ x &= rcos(\varphi) \end{aligned} \tag{2.4}$$

Despite the straightforward use and representation, complex numbers define an interaction between their two parts when used in computation. Consider the operations necessary in a regression model outlined in Equation 2.16. Each element $z_1 \in \mathbb{C}$ of the weight matrix $W \in \mathbb{C}^{n\times m}$ interacts with an element $z_2 \in \mathbb{C}$ of an input $x \in \mathbb{C}^n$:

$$\begin{aligned} z_1 z_2 &= (a + ib)(c + id) \\ &= (ac - bd) + i(ad + bc), \\ z_1 + z_2 &= (a + ib) + (c + id) \\ &= (a + c) + i(b + d) \end{aligned} \tag{2.5}$$

In an equivalent representation of a complex number in its polar form:

$$\begin{aligned} z_1 z_2 &= (r_1 e^{i\varphi_1})(r_2 e^{i\varphi_2}) \\ &= (r_1 r_2 e^{i\varphi_1+\varphi_2}), \\ z_1 + z_2 &= (r_1 e^{i\varphi_1}) + (r_2 e^{i\varphi_2}) \\ &= r_1\cos(\varphi_1) + r_2\cos(\varphi_2) \\ &+ i(r_2\sin(\varphi_2) + r_1\sin(\varphi_1)) \end{aligned} \tag{2.6}$$

Consequently, simply doubling the number of parameters in real-valued layers is not sufficient to achieve the same effect as in a complex-valued layer. This is illustrated when a complex number $z = a + ib$ is expressed in an equivalent *augmented representation*. Specifically, as 2×2 matrix M in the ring of $M_2(\mathbb{R})$:

$$M_{(a+ib)} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \text{ such that } M_{(0+i1)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, M_{(1+i0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

$$
\begin{aligned}
M_{(a+ib)}M_{(c+id)} &= \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} c & -d \\ d & c \end{bmatrix} \\
&= \begin{bmatrix} ac - bd & bc + dc \\ bc + dc & ac - bd \end{bmatrix}
\end{aligned} \tag{2.8}
$$

A deep learning architecture that performs well with real-valued parameters may not work for complex-valued parameters and vice versa. Models that do not facilitate the structure, or tasks that do not require the structure, will not show improved performance. Because of their interacting nature, complex numbers are essential to mathematically represent and handle wave properties [22].

In the following definitions we are using complex numbers $\mathbb{C}$ for inputs and weights. They can usually be switched with $\mathbb{R}$, since every real number is a complex number with zero imaginary parts. The set of real numbers are a subset of complex numbers $\mathbb{R} \subset \mathbb{C}$.

2.2 Transformations and Embeddings

In our work we use various transformations that map real-valued inputs to a complex-valued representation:

$$\mathcal{A} : \mathbb{R} \to \mathbb{C} \tag{2.9}$$

The Fourier Transformation (FT) translates a time signal to the frequency domain by decomposing it into a sequence of sine and cosine waves. The output is a sequence of complex numbers each corresponding to a frequency (bin). Each wave is characterised by its amplitude $|z_i|$ and a phase offset φ_i. The Fourier Transformation $\mathcal{F}$ of real-valued function f is defined by:

$$\mathcal{F}(k) = \int_{t=-\inf}^{\inf} f(t) e^{-i2\pi kt} dt \tag{2.10}$$

Discretisation of the Fourier Transformation results in the Discrete Fourier Transformation $\mathcal{F}$ (DFT). The DFT assumes a discrete sampling of a continuous real function. The sampled sequence given by a real-valued function $f(t)$ of time consisting of N samples.

$$\mathcal{F}(k) = \sum_{t=0}^{N-1} f(t) e^{i\frac{2\pi}{N}kt} \tag{2.11}$$

Its variation, the Short-Time Discrete Fourier Transformation (STDFT), applies a window function on the original time signal $f(t)$. It moves a window over a signal in a certain stride length. Computationally, we can use the Fast Fourier Transform (FFT) and the Discrete FFT. We

will later use the frequency representation in convolution (Section 2.5).

A further generalisation of the DFT is the Z-Transformation $\mathcal{Z}$ (ZT). The DFT is a Z-Transformation with $|a| = 1$. Formally, the Z-Transformation $\mathcal{Z}$ of a sampled signal $f(t)$ consisting of N samples is defined as:

$$\mathcal{Z}(a) = \sum_{t=0}^{N-1} f(t)a^{-t} \tag{2.12}$$

In contrast to the Z-Transformation the Discrete Cosine Transformation C (DCT) returns real-valued sequence and is defined by:

$$C(k) = \sum_{t=0}^{N-1} f(t) \cos\Big(\frac{\pi k(2t+1)}{2N}\Big) \tag{2.13}$$

Embeddings are another way of translating discrete real-valued sequences into the complex plane. They allow us to learn a set of dense vectors (or tensors) to represent discrete data (e.g. words in a vocabulary). Embeddings are implemented by projecting a sparse one-hot vector $v \in \{0,1\}^{1\times N}$ (with the j-th element being a one) onto a learnable dense embedding matrix (or tensor) $E \in \mathbb{C}^{N\times K}$ (Equation 2.14).

$$Emb(w_j) = \mathcal{E}(w_j) = vE = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} w_{11} & \dots & w_{1k} \\ w_{21} & \dots & w_{2k} \\ \vdots & \ddots & \\ w_{nj} & \dots & w_{nk} \end{bmatrix} \tag{2.14}$$

This practically 'selects' the j-th row vector of the embedding matrix E. In neural networks this operation is often referred to as the **embedding layer**. In NLP this method is often used as a process of dimensionality reduction.

2.3 Multi-Layer Perceptrons

A complex-valued neuron is defined analogously to its real-valued version. We consider its differences in structure and training. The complex neuron is defined as the output of an activation function ϕ applied to the dot product $\cdot$ of an input $x \in \mathbb{C}^n$ and the complex weight $w \in \mathbb{C}^n$ with the addition of complex bias $b \in \mathbb{C}$:

$$o = \phi(x \cdot w + b) \tag{2.15}$$

Arranging m neurons into single operation gives us a fully connected layer or dense layer or perceptron:

$$o = \phi(xW + b) \tag{2.16}$$

with an input $x \in \mathbb{C}^n, W \in \mathbb{C}^{n \times m}, b \in \mathbb{C}^m$. The output elements o_{ij} (before the non-linearity ϕ) are computed by Equation 2.17.

$$o_{ij} = b_j + \sum_{k=1}^{n} x_{ik} w_{kj} \tag{2.17}$$

Stacking multiple, fully connected dense layers results in the simplest DL model, a Multi-Layer Perceptron (MLP). The neural network combines the input features and weighs the outputs of previous layers in the next layer. This results in a repeated application of weight matrix multiplication and non-linearities (Equation 2.18).

$$\tilde{f_k}(x) = softmax(\phi_{k+2}(\phi_{k+1}(\phi_{...}(\phi_1(xW_1 + b_1)W_{...} + b_{...})W_k + b_k)W_{k+1} + b_{k+1})) \tag{2.18}$$

The MLP architecture is shown in Figure 3.1. It is mainly used for classification when combined with a probabilistic output by applying softmax resp. sigmoid to the output layer. In contrast to real-valued MLPs, complex-valued MLPs can naturally represent a periodic and/or unbounded function [23]. A gradient-based optimisation strategy can be applied with the Back-propagation algorithm to learn the weights and biases (Section 2.10).

2.4 Recurrence

We define complex-valued versions of Recurrent Neural Networks (RNN). The first definition uses a simple recurrence at time step t:

$$\begin{aligned} h_t &= (xW_{hx}) + b, \\ o_t &= \phi(h_{t-1}W_{hh} + h_t) \end{aligned} \tag{2.19}$$

with the input $x \in \mathbb{C}^n$, input (or input-to-hidden) weights $W_{hx} \in \mathbb{C}^{n \times m}$, bias $b \in \mathbb{C}^m$ and hidden-to-hidden weights $W_{hh} \in \mathbb{C}^{n \times m}$. The Long-Short Term Memory (LSTM) [24] is defined at time step t as:

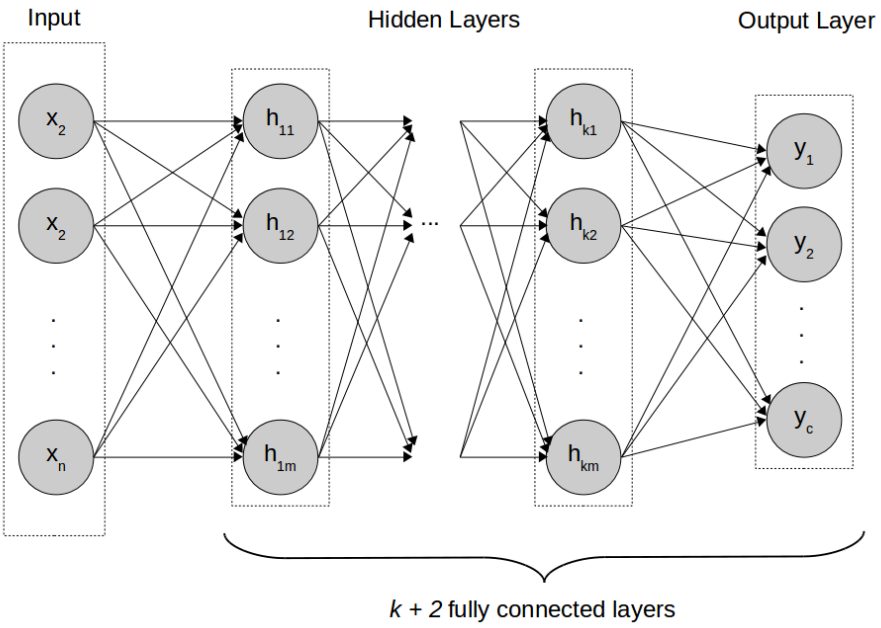

Figure 2.2: Multi-layer perceptron with k hidden layers.

$$
\begin{aligned}
f_t &= \sigma(x_t W_f + h_{t-1} U_f + b_f), \\
i_t &= \sigma(x_t W_i + h_{t-1} U_i + b_i), \\
o_t &= \sigma(x_t W_o + h_{t-1} U_o + b_o), \\
c_t &= f_t \circ c_{t-1} + i_t \circ \phi(x_t W_c + h_{t-1} U_c + b_c), \\
h_t &= o_t \circ \phi(c_t)
\end{aligned} \tag{2.20}
$$

with the input $x \in \mathbb{C}^n$, the input gate vector i, forget gate vector f, output gate vector o, kernel $W \in \mathbb{C}^{n \times m}$ and recurrent kernel $U \in \mathbb{C}^{n \times m}$, bias $b \in \mathbb{C}^m$ and hidden-to-hidden weights $W_{hh} \in \mathbb{C}^{n \times m}$. The Hadamard product (or element-wise multiplication) is denoted with $\circ$. Hochreiter and Schmidhuber [24] chooses hyperbolic tangents $\phi = \tanh$ as non-linearities.

The Gated Recurrent Unit (GRU) with complex weights is defined at time step t as:

$$
\begin{aligned}
z_t &= \phi_g(x_t W_z + h_{t-1} U_z + b_z), \\
r_t &= \phi_g(x_t W_r + h_{t-1} U_r + b_r), \\
h_t &= z_t \circ h_{t-1} + (1 + z_t) \circ \phi_h(x_t W_h + U_h(r_t \circ h_{t-1}) + b_h)
\end{aligned} \tag{2.21}
$$

with the input $x \in \mathbb{C}^n$, an update gate vector z, a reset gate vector r and weights $W \in \mathbb{C}^{n \times m}$

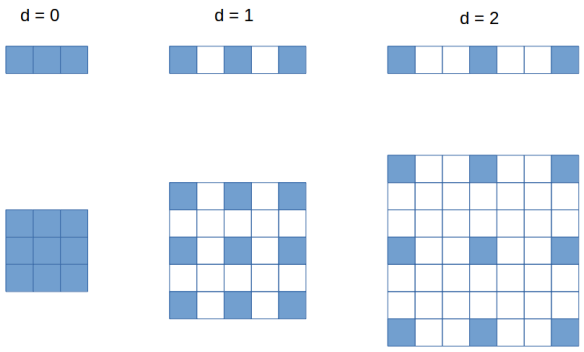

Figure 2.3: Visual representation of dilated kernels with fixed size $k = 3$ and $k = 3 \times 3$.

and $U \in \mathbb{C}^{n \times m}$, bias $b \in \mathbb{C}^m$ and an output vector h.

2.5 Convolutions

Convolutions are another fundamental building block of modern neural networks. Within Convolutional Neural Networks (CNN) this operation is mainly used to learn how to extract context-sensitive features from large inputs like images or signals. Formally, a convolution is an operation that computes the integral of two functions f and g over time t:

$$f * g = \int_{-\inf}^{\inf} f(\tau) g(t - \tau) d\tau \tag{2.22}$$

Hence, a convolution of two functions describes the area below the curve of the composite function $f(\tau)g(\tau)$ at a specific time step t. In physical systems the output describes a system's response to a signal. For sampled sequences, or other discrete inputs it is a repeated (matrix or vector) multiplication of an input vector X and a moving kernel K of size k:

$$X[n] * K[n] = \sum_{j=-\inf}^{\inf} X[j] \cdot K[n - j] = \sum_{j=-\inf}^{\inf} K[j] \cdot X[n - j] \tag{2.23}$$

In practice, a kernel (or filter) is convolved with the input by moving the kernel K along the time axis with a step size (stride) s over an input to compute a context representation with every step. Another variant uses dilated (or atrous) kernels. In these dilated convolutions each element of the kernel K is padded with d blind spots resulting in an 'exploded' kernel $K_{k,d}$ in order to extend their perceptive fields. A visualisation is shown in Figure 2.3.

Convolutional layers used in DL architectures learn m filters or kernels at the same time. They extract different sets of features from (padded) input sequences $x \in \mathbb{C}^n$ with a kernel $K \in \mathbb{C}^{k \times m}$

which is moved over the time axis with a stride s and where each kernel is of size k and a dilation of d.

$$o = conv_s = \phi(x * K_{k,d,m}) \tag{2.24}$$

Given a stride s, the kernel K and the input x the output o contains is $\frac{n-k}{s} + 1$ frames each of k dimensions. We can achieve a dimensionality reduction by pooling the output of a convolutional layer. Pooling applies a rectangular window function of size k to receive multiple windows Z and computes a single representation for each window.

$$pool_{max}(Z) = max(z_j, \ldots, z_k) \tag{2.25}$$

$$pool_{avg}(Z) = \frac{1}{t} \sum_{j=1}^{k} z_j \tag{2.26}$$

Both convolutions and pooling can be adapted to higher ranks by increasing the order of kernels and windows.

Another perspective to take for convolution in the complex-domain is to convolve a (real or complex) signal with a complex-valued kernel. Since the convolution operator is linear, the convolution of a real-valued signal $X[n]$ with a complex-valued kernel $K \in \mathbb{C}$ of size k is described by:

$$X[n] * K[n] = X[n] * Re(K[n]) + i(X[n] * Im(K[n])) \tag{2.27}$$

We also adapt pooling the complex plane by applying average and maximum pooling onto the magnitudes of the complex elements z_i:

$$pool_{avg}(|Z|) = \frac{1}{N} \sum_{i=0}^{N} |z_i| \tag{2.28}$$

$$pool_{max}(|Z|) = z_i, i = argmax(|Z|) \tag{2.29}$$

2.6 Spectral Neural Networks

Inspired by spectral analysis there has been a number of approaches to learn frequency representations. They still rely on real-valued weight matrices, outputs and most importantly real-valued input. A possible application are learned spectrograms. Spectograms are usually used to visualise frequencies of a signal, but are also used as signal fingerprint. They map the complex-valued output of Fourier Transformations directly to a real-valued representation. This discards import-

ant phase information that is necessary to reconstruct the original input signal. In consequence, a real-valued neural network is limited by the use of real-valued representations. Literature on complex-valued spectral representations created with Deep Learning is sparse [25, 26].

In our work we propose a complex-valued encoder-decoder architecture to facilitate processing and learning of complex-valued spectral representations within the network. There have been few approaches using complex-valued neural networks for speech processing. Drude et al. [25], Trabelsi et al. [27], Sarroff [28] and Choi et al. [29] motivate complex-valued neural networks for speech processing with complex-valued convolutions and the convolution theorem. Their work focuses on speech enhancement. Choi et al. [29] propose a complex-valued autoencoder architecture inspired by U-Net to enhance speech with a phase-aware masking and an adapted loss function. Parcollet et al. [30] propose hyper-complex numbers (quaternions) to allow an even higher degree of freedom when extracting features from the audio signal. Zeghidour et al. [31] propose emulating complex convolutions to learn features directly from the original input signal, but computes a real-valued representation afterwards.

Furthermore, using frequencies for representation requires a compromise between time and frequency resolution. We propose to use a window function of fixed duration and compute the DFT of the resulting frame. Hence, we use the STDFT as a single preprocessing step. The STDFT also encourages the network to learn frequency representations.

2.7 Complex-Valued Attention

We present a generalised framework of attention and derive a complex-valued attention mechanism. We use the complex-valued attention to attend to (real-valued) time or to (complex-valued) frequency representations within the network. Formally, attention is defined as a mapping of a set of key-value pairs $\{(k_1, v_1), \ldots, (k_n, v_n)\}$ according to a query q to a probability distribution α over the keys k. First, we compute the attention vector α, then a score between each key $k_i \in \mathbb{R}^m$ and the query $q \in \mathbb{R}^m$ defined as:

$$\alpha_{\mathbb{R}}(k, q) = \sigma_i(f(k_i, q)), \forall k_i \tag{2.30}$$

with softmax σ_i for every key k_i being the i-th element of the attention vector. Attention weights α_i are used to compute a weighted average representation of the n values $v_i \in \mathbb{R}^d$:

$$attention_{\mathbb{R}}(k, v, q) = \sum_{i=1}^{n} \alpha_i(k, q) v_i \tag{2.31}$$

To compute the attention vector α we require a scoring function $f : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$:

$$f(k, q) = q \cdot k \tag{2.32}$$

A further extension is Multi-Head Attention (MHA). Multiple heads attend to the same sequence which has been previously projected into different subspaces. Queries $Q \in \mathbb{R}^{r \times m}$, keys $K \in \mathbb{R}^{n \times m}$ of size m, values $V \in \mathbb{R}^{n \times d}$ of size d are stacks of individual keys k, values v and queries q. Equations 2.30, 2.31, 2.32 generalise to:

$$attention_h(K, V, Q) = concatenate(head_1, \ldots, head_h)W^O \tag{2.33}$$

MHA consists of h concatenated attention outputs. Each head attends to a different projection of the same sequence in parallel and outputs a matrix of dimension $r \times s$. It projects stacked queries, keys and values to s dimensions. In the transformer that is the model size s:

$$\begin{aligned} head_j = \sum_{i=1}^{n} \alpha_i (&split_j(K)U_j^K, \\ &split_j(Q)U_j^Q) \\ &split_j(V)U_j^V \end{aligned} \tag{2.34}$$

with $U_j^K \in \mathbb{R}^{m \times \frac{s}{h}}, U_j^Q \in \mathbb{R}^{m \times \frac{s}{h}}, U_j^V \in \mathbb{R}^{d \times \frac{s}{h}}, W^O \in \mathbb{R}^{m \times d}$. The projection step precedes a splitting of Q, K, V into h parts over the last dimension s. Each head $head_j$ outputs a matrix of dimensions $r \times \frac{s}{h}$ in which each row represents the weighted average representation to each of the r queries. They are then concatenated over the last dimension $\frac{s}{h}$. Hence, $smodh = 0$. The self-attention setting used for transformer networks sets $Q = K = V = X \in \mathbb{R}^{n \times d}$ with X being the n stacked input embeddings of dimension d, hence $r = n$ and $d = m = s$. The attention vectors are computed by the softmax. The output of $\alpha(K, Q)$ are r attention vectors each of length n:

$$\alpha(K', Q') = \sigma(f(K', Q')) \tag{2.35}$$

$$f(K', Q') = \frac{Q' K'^T}{\sqrt{m}} \tag{2.36}$$

Complex-valued queries and keys require a few adaptations. Based on the formalised attention framework shown above we present a novel a generalisation. Our approach does not exclude the possibility to attend to real-valued sequences. The main difference is the computation of attention vectors and scoring. We use the same optional masking.

Since softmax and sigmoid functions are not explicitly defined for complex inputs, we have to replace it in classifiers and the attention mechanism. We replace it with novel probability amplitude activation function ψ inspired by quantum states. A quantum state given by a complex vector $|\phi\rangle = \begin{bmatrix} z_1 & \ldots & z_c \end{bmatrix}$ satisfies the constraint $\sum_{i=1}^{c} |z_i|^2 = 1$. Additionally, each coefficient $z_i \in \mathbb{C}$ corresponds to an element of the orthogonal bases that span a Hilbert space. The probability amplitudes $|z_i|^2$ thus define a probability distribution over orthogonal bases. Our function maps a

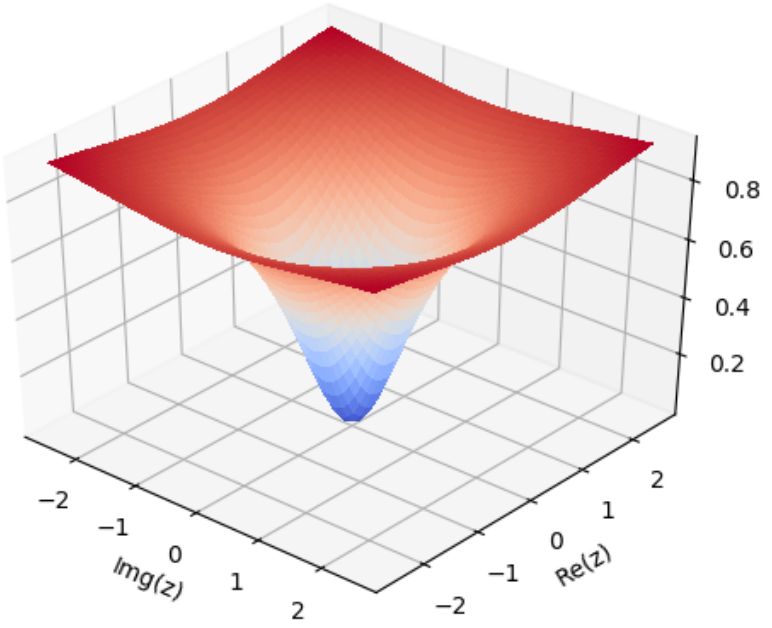

Figure 2.4: Probability amplitude activation function as an alternative to the sigmoid function.

vector of complex numbers to a probability distribution $\psi : \mathbb{C}^c \to \mathbb{R}^c$:

$$\psi_j(z) = \frac{|z_j|^2}{1 + \sum_{a=1}^{c} |z_a|^2} = \frac{Re(z_j)^2 + Im(z_j)^2}{1 + \sum_{a=1}^{c} Re(z_a)^2 + Im(z_a)^2} \tag{2.37}$$

Similarly, we define an alternative for the sigmoid function:

$$\Psi(z) = \frac{|z|^2}{1 + |z|^2} = \frac{Re(z)^2 + Im(z)^2}{1 + Re(z)^2 + Im(z)^2} \tag{2.38}$$

This **probability amplitude** activation function is non-holomorphic, but it is real differentiable at most points. Hence, we can use it for gradient-based optimisation. Unlike other activation functions it does not have any singularities and can be used as a replacement for the real-valued softmax. We also replace the scoring function of (scaled) dot-product with a (scaled) inner-product $f : \mathbb{C} \to \mathbb{C}$:

$$f(k, q) = \frac{\langle k|q\rangle}{\sqrt{m}} = \frac{k \cdot \bar{q}}{\sqrt{m}} = \frac{\sum_{j=1}^{m} k_j \bar{q}_j}{\sqrt{m}} \tag{2.39}$$

Equation 2.30 and 2.31 generalise to the new complex attention. For a query $q \in \mathbb{C}^m$, key $k_i \in \mathbb{C}^m$ and value pairs $v_i \in \mathbb{C}^d$ it is defined as:

$$attention(k, v, q) = \sum_{i=1}^{n} \alpha_i(k, q)v_i \tag{2.40}$$

$$\alpha(k, q) = \psi_i(f(k_i, q)), \forall k_i \tag{2.41}$$

Practically, the attention vector $\alpha \in \mathbb{R}^n$ scales the magnitudes of the values $v_i \in \mathbb{C}^d$, but leaves the phases intact. For complex-valued MHA $attention_h(K, V, Q)$ the projection matrices are redefined as $U_j^K \in \mathbb{C}^{m\times s}, U_j^Q \in \mathbb{C}^{m\times s}, U_j^V \in \mathbb{C}^{d\times s}, W^O \in \mathbb{C}^{m\times s}$:

$$head_j = \sum_{i=1}^{n} \alpha_i(split_j(K)U_j^K, split_j(Q)U_j^Q)split_j(V)U_j^V \tag{2.42}$$

$$\alpha(K', Q') = \psi_i(f(K', Q')) \tag{2.43}$$

The transpose needs to be replaced with the Hermitian conjugate:

$$f(K', Q') = \frac{Q'K'^{\dagger}}{\sqrt{m}} \tag{2.44}$$

This complex-valued attention mechanism can be used both in real-valued and complex-valued networks, since the resulting attention vector is real. Depending on the choice of queries Q, keys K and values V the output will be real or complex-valued.

The contributions in this chapter are the generalisations of existing neural networks building blocks to the complex plane: complex-valued framework of attention, convolutional and recurrent neural networks.

2.8 Activation Functions

In any neural network an important decision is the choice of non-linearity. It defines how fast a model may converge and which initialisation to choose. In the complex domain, however, the criteria for activation functions are different [32, 33, 34]. An important theorem is the Liouville Theorem (Equation 2.8.1).

Definition 2.8.1. The Liouville Theorem states that any bounded holomorphic function $f : \mathbb{C} \to \mathbb{C}$ (that is differentiable on the entire complex plane) must be constant. Let $f(z)$ be entire (holomorphic at any point in $\mathbb{C}$) and $|f(z)|$ bounded (there exists a number x s.t. $|f(z)| \leq x, \forall z \in \mathbb{C}$) for all values of $z \in \mathbb{C}$, then $f(z)$ is constant.

Another significant and related problem is singularities. Many complex-valued functions have points in the complex plane z_0 where they are not defined or have other singularities. We

may be able to exclude isolated singularities $\mathbb{C} \setminus \{z_0\}$. This, however, requires constrained optimisation methods and is therefore less desirable. Instead we should seek functions that do not have any singularities in the complex plane.

Due to the theorem, we need to choose unbounded and/or non-holomorphic activation functions ϕ. Activation functions for complex-valued neural networks can be categorised in complex-valued functions $\phi : \mathbb{C} \to \mathbb{C}$, split-complex functions $\phi : \mathbb{R}\times\mathbb{R} \to \mathbb{R}\times\mathbb{R}$ or real-valued functions $\phi : \mathbb{C} \to \mathbb{R}$. Singularities like poles may not be a problem, since a gradient-based optimisation process naturally avoids these singularities.

Similar to real-valued neural networks there are many activation functions proposed specifically for their use in complex-valued neural network. We chose the identity function to identify tasks that may not be linearly separable in $\mathbb{R}$, but are linearly separable in $\mathbb{C}$. The hyperbolic tangent is a well-studied function and defined at most points for complex and real numbers. In the complex numbers, however, it has singularities. The rectifier linear unit is also well understood and frequently used in a real-valued setting. However, it has not been considered in a complex-valued setting. This illustrates separate application on the two parts of a complex number. The Modulo Rectifier Linear Unit is a popular function in existing literature on complex-valued neural networks as well as the related complex hyperbolic tangent function by Hirose [23]. The magnitude and squared magnitude functions are difficult activation functions. They are chosen to map complex numbers to real numbers and to investigate their practicality. They are real differentiable except at points $z_0 = 0$, but not complex differentiable at any point. In these cases we compute the derivatives with respect to the real parts.

- Identity (or no activation function):

$$\phi(z) = z \tag{2.45}$$

- Hyperbolic tangent:

$$\phi(z) = \tanh(z) = \frac{\sinh(z)}{\cosh(z)} \tag{2.46}$$

- Split Rectifier linear unit (ReLU):

$$\begin{aligned} \phi(z) = ReLU(z) &= ReLU\Big(Re(z)\Big) + ReLU\Big(Im(z)\Big)i \\ &= max\Big(0, Re(z)\Big) + max\Big(0, Im(z)\Big)i \end{aligned} \tag{2.47}$$

- Intensity (or magnitude squared):

$$\phi(z) = |z|^2 = Re(z)^2 + Im(z)^2 \tag{2.48}$$

- Magnitude (or complex absolute):

$$\phi(z) = |z| = \sqrt{Re(z)^2 + Im(z)^2} \tag{2.49}$$

- Modulo Rectifier Linear Unit (modRelU) [27]

$$\phi(z) = RelU(|z| + b)\frac{z}{|z|} \tag{2.50}$$

- Absolute Hyperbolic Tangents [35]

$$\phi(z) = tanh(\frac{|z|}{m^2})\frac{z}{|z|} \tag{2.51}$$

To receive a real-valued loss we have to convert the complex-valued output of a neural network to a real-valued prediction. While this depends on the tasks, for classification tasks a logistic function is used to map *logits* to a probability distribution over the output classes. In complex-valued neural networks, before applying the logistic function in the last layer, we use the magnitude function $|z|$ or amplitude function $|z|^2$ to receive a real-valued loss.

$$sigmoid(|z|^2) = \sigma(|z|^2) = \frac{1}{1 + e^{-Re(z)^2 - Im(z)^2}} \tag{2.52}$$

For an probabilistic output vector $z = \begin{bmatrix} z_1 & \dots & z_c \end{bmatrix}$

$$softmax(|z_j|^2) = \frac{e^{Re(z_j)^2 + Im(z_j)^2}}{\sum_{c=1}^{C} e^{Re(z_c)^2 + Im(z_c)^2}} \tag{2.53}$$

2.9 Loss Function

We choose a real-valued loss function as objectives for our gradient-based optimisation. Complex-valued loss functions would require at least a partial ordering over the set of complex numbers (similar to a linear matrix inequality). However, there is no standard (total) ordering on the field of complex numbers, since $i^2 = -1$. We denote the parameters of a model with θ.

Mean Squared Error (MSE) on N samples is commonly used for regression problems. It has a straightforward generalisation for complex numbered outputs $\tilde{y}$ and targets y:

$$J_\theta = \frac{1}{N} \sum_{\forall (x,y)} |y - \tilde{y}|^2 \tag{2.54}$$

The intention of Maximum Likelihood Estimation (MLE) is to learn a model so that it is able to generate the observed data. Hence, we maximise the log likelihood of the model producing the data using the parameters θ. The data points x_j are assumed to be drawn from the same distribution.

$$\mathcal{L}_\theta = \frac{1}{N} \sum_{\forall x} \ln(f(x|\theta)) \tag{2.55}$$

In classification and sequence transduction tasks categorical cross entropy objectives are used to minimise the difference between an output probability distributions and the expected probability distribution. Both distributions are assumed to be defined over the same set of labels or classes L. The loss requires the model's output $\tilde{y}$ and the expected output y given an input x (Equations 2.56, 2.57). For a sequence transducer this would be a probabilistic output $p(\tilde{y}_l|x)$ of a specific label $l \in L$.

$$J_\theta = -\sum_{\forall(x,y)} \sum_{l \in L} p(y_l|x) \log(p(\tilde{y}_l|x)) \tag{2.56}$$

If $L \in \{0, 1\}$ is binary, categorical cross entropy simplifies to:

$$J_\theta = \sum_{\forall(x,y)} (-y_l \log(p(\tilde{y}_l|x)) - (1 - y_l)(\log(1 - p(\tilde{y}_l|x)))) \tag{2.57}$$

2.10 Training and Optimisation

Similar to real-valued neural networks, complex-valued neural networks can be optimised using Gradient Descent (GD) and its variants in combination with the Backpropogation algorithm. Training using gradient-based methods requires activation functions to be differentiable with respect to the model's parameters θ. The training process consists of a forward and backward pass through the neural network. After computing the loss J_θ with the result of the forward pass, we propagate the loss back through the network updating each parameter according to their gradients in order to minimise the loss.

Stochastic Gradient Descent (SGD) updates the parameters for each example in the j-th (mini) batch (X_j, Y_j) with b samples by computing the gradient with respect to the parameters θ. Momentum [36] is a method that increases steps towards repeating directions of the gradient and dampens oscillations. The update rule for normal SGD with a learning rate α is given in Equation 2.58 and with momentum γ given by Equation 2.59.

$$\theta = \theta - \alpha \nabla J_\theta(X_j, Y_j) \tag{2.58}$$

$$\begin{aligned} v_t &= \gamma v_{t-1} + \alpha \nabla J_\theta(X_j, Y_j), \\ \theta &= \theta - v_t \end{aligned} \tag{2.59}$$

In all of our experiments we use an Adaptive Momentum Optimisation (Adam) [37] with SGD. Adam is a gradient-based optimisation algorithm that stores both gradients and momenta of previous updates. The update rule changes to Equation 2.60 with a decaying average of past squared gradients v_t and a decaying average of past gradients m_t.

$$
\begin{aligned}
v_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\
m_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \\
\hat{v}_t &= \frac{m_t}{1-\beta_1^t}, \\
\hat{m}_t &= \frac{v_t}{1-\beta_2^t}, \\
\theta_{t+1} &= \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t
\end{aligned}
\tag{2.60}
$$

Backpropagation exploits the chain rule to update each weight according to their contribution to the loss. It computes partial derivates for each layer. It starts with the computed loss and propagates the error back through the network.

$$
\frac{\partial J_\theta}{\partial \theta_{ij}^k} = \frac{\partial J_\theta}{\partial o_j^k} \frac{\partial o_j^k}{\partial \theta_{ij}^k} \dots \tag{2.61}
$$

In Equation 2.61 o_j^k is the output after the activation of the j-th node of the k-th layer. The computation of these partial derivatives with respect to the parameters is repeated for each function. An example of the function composition is Equation 2.18.

For complex differentiable function we can use the same principle. However, complex differentiability is a stricter condition than real differentiability (Definition 2.10.1).

Definition 2.10.1. Similar to real differentiability, a complex function $f : \mathbb{C} \to \mathbb{C}$ at a point z_0 of an open subset $\Omega \subset \mathbb{C}$ is *complex-differentiable* if there exists a limit such that

$$
f'(z_0) = lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0} \tag{2.62}
$$

If the function f is complex-differentiable at all points of Ω it is called *holomorphic*. While in the real-valued case the existence of a limit is sufficient for a function to be differentiable, the complex definition in Equation 2.10.1 implies a stricter property. We can express this differently by representing a complex number $z \in \mathbb{C}$ with two real numbers $z = x + iy$. For $f(x + iy) = u(x, y) + iv(x, y)$ to be holomorphic, the limit not only needs to exist for the two functions $u(x, y)$ and $v(x, y)$, but the (partial) derivatives must also satisfy the Cauchy-Riemann Equations. That also means that a function can be non-holomorphic (i.e. not complex-differentiable) in z, but still be differentiable in its parts x, y. Hence, real differentiability of functions $u(x, y)$ and $v(x, y)$ are not sufficient to satisfy the conditions of the Cauchy-Riemann Equations (Definition 2.10.2).

Definition 2.10.2. A complex function $f(x+iy) = u(x, y) + iv(x, y)$ with real-differentiable functions $u(x, y)$ and $v(x, y)$ is complex-differentiable if they satisfy the Cauchy-Riemann Equations:

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{\partial v}{\partial y}, \\ -\frac{\partial u}{\partial y} &= \frac{\partial v}{\partial x} \end{aligned} \tag{2.63}$$

In order to apply the *chain rule* to non-holomorphic functions, the differentiability with respect to their real and imaginary parts can be utilised. We consider the complex function $f(z, \bar{z})$ to be a function of z and its complex conjugate $\bar{z}$. Effectively, we choose a different basis for our partial derivatives.

$$\begin{aligned} \frac{\partial}{\partial z} &= \frac{1}{2}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right), \\ \frac{\partial}{\partial \bar{z}} &= \frac{1}{2}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right) \end{aligned} \tag{2.64}$$

These derivatives are a consequence of Wirtinger calculus (or $\mathbb{C}\mathbb{R}$-calculus). They allow the application of the chain rule to many non-holomorphic functions for multiple complex variables z_i:

$$\begin{aligned} \frac{\partial}{\partial z_i}(f \circ g) &= \sum_{j=1}^{n}\left(\frac{\partial f}{\partial z_j} \circ g\right)\frac{\partial g_j}{\partial z_i} + \sum_{j=1}^{n}\left(\frac{\partial f}{\partial \bar{z}_j} \circ g\right)\frac{\partial \bar{g}_j}{\partial z_i}, \\ \frac{\partial}{\partial \bar{z}_i}(f \circ g) &= \sum_{j=1}^{n}\left(\frac{\partial f}{\partial z_j} \circ g\right)\frac{\partial g_j}{\partial \bar{z}_i} + \sum_{j=1}^{n}\left(\frac{\partial f}{\partial \bar{z}_j} \circ g\right)\frac{\partial \bar{g}_j}{\partial \bar{z}_i} \end{aligned} \tag{2.65}$$

The Wirtinger calculus provides an alternative method for computing the gradient if a function is differentiable in its real parts. Many activation functions (complex- or real-valued) are not entirely differentiable with respect to weights. However, the function does not need to be differentiable at every point in $\mathbb{C}$, but only at points at which the partial derivatives are computed. Modern gradient implementations (e.g. TensorFlow [8]) will either return the left or right derivative instead or exclude that point entirely.

This general practice allows a wide range of activation functions. The training process, however, may become unstable. An unstable training process necessitates special methods to avoid problematic regions in the function domain. Another approach is to avoid singular points when constructing the network. Fewer singularities make the training more reliable and stable. We use the above theories and equations and apply these throughout this work to train complex-valued neural networks.

CHAPTER 3

Complex-Valued Multi-layer Perceptrons for Real-Valued Classification

In this chapter we attempt to learn complex-valued features directly from the real-valued input data. We do not enforce learning in the frequency domain by feeding real-valued data into a transformation before the network. We allow the neural network to learn necessary complex features itself. We had hoped that the neural network would automatically learn complex transformations (like the DFT or ZT) in order to represent data in a complex domain. Unfortunately, this is not the case. Instead, we find that when unprocessed real-valued data is used, real-valued networks (MLPs and CNNs) pose an upper performance bound.

We present our results on benchmark classification tasks using complex and real-valued MLPs and CNNs. We designed three experiments which respect the total number of real-valued parameters. We discuss the significant problems and limitations that occur when using real-valued input data. We investigate the shown behaviour and isolate criteria to identify situations that may benefit from this increased expressibility. We also suggest alternative approaches which are further investigated in Chapter 4.

This chapter is motivated by the observation that the XOR function can be approximated by a single complex-valued neuron, as explained in Chapter 1. A single real-valued neuron, however, is not capable of approximating this logic function. This suggests that complex-valued networks can learn features that real-valued networks cannot learn. We design simple model architectures with respect to the number of real-valued parameters. In Chapter 2 we presented both used model types, MLPs and CNNs, and the fundamental aspects to design and train neural architectures used in this chapter.

3.1 Introduction

In recent years complex-valued neural networks have been successfully applied to a variety of real-world tasks. Their accomplishments are mainly in Signal Processing where the input data has a natural interpretation in the complex domain. Complex-valued neural networks should be compared to real-valued networks. However, we need to ensure that these architectures are comparable in their model size and capacity. This aspect of the comparison is rarely studied or only considered superficially. A metric for their capacity is the number of real-valued parameters. The introduction of complex numbers into a model increases the computational complexity as well as the number of real-valued parameters required, but assumes a certain structure of weights and data input.

This chapter explores the performance of complex-valued MLPs and CNNs with varying depth and width. We consider various activation functions and the number of real-valued parameters in benchmark classification tasks of real-valued data. We propose two methods to construct comparable networks: 1) by setting a fixed, alternating number of neurons/filters per layer and 2) by setting a fixed budget of real-valued parameters. The parameters are uniformly distributed over the layers. We choose the MNIST digit classification [12], CIFAR-10 image classification [14], CIFAR-100 image classification [14] and Reuters newswire topic classification [13] as benchmarks. We use classification of synthetic data for further investigation.

3.2 Related Literature

Complex-valued neural networks were first formally described by Clarke [38]. Several authors have since proposed complex versions of the backpropagation algorithm based on gradient descent [39, 40, 41]. Inspired by work on multi-valued threshold logic [42] in the 1970s, a multi-valued neuron and neural network was defined by Aizenberg et al. [43, 44] who also extends this idea to quaternions. Since then many approaches to further generalise complex-valued neural networks to quaternion neural networks have been made [45, 46]. In the 2000s and 2010s, complex neural networks were successfully applied to a variety of tasks [47, 48, 49, 50, 51, 52]. These tasks mainly involved the processing and analysis of data with an intuitive mapping to complex numbers. Images domain and time signals (in their wave form) have been transformed into the Fourier domain and used as input data to complex-valued neural networks [35, 22]. They have then be used in conjunction with convolutions [53]. While real convolutions are widely used in DL, it is possible to replace them with complex convolutions [27, 54, 55, 56]. Despite their successes, complex neural networks have been less popular than their real-valued variants. A reason may be the less intuitive training process and architecture design, stemming from stricter requirements for the differentiability of activation functions in the complex plane [57, 58].

In the past complex-valued classifiers, which are designed specifically to deal with real-

valued problems, have been shown to be superior to real-valued counterparts [59]. However, neither deep networks nor the general behaviour of complex-valued MLPs has been investigated. We systematically explore the performance of MLPs on simple classification tasks in consideration of the activation function, width and depth.

When comparing complex-valued neural networks with real-valued neural networks, many publications ignore the number of parameters altogether [44], compare only the number of parameters of the entire model [27] or do not distinguish between complex- or real-valued units [60]. Such comparisons are equivalent to comparing models of different sizes. In this chapter we explicitly design our experiments with respect to comparable capacities.

3.3 Capacity

The number of (real-valued) parameters is a metric to quantify the capacity of a network and its ability to approximate structurally complex functions. With too many parameters the model tends to overfit the data while with too few parameters it tends to underfit.

A consequence of representing a complex number $a + ib$ using pairs of real numbers (a, b) is that the number of real parameters of each layer is doubled: $p_{\mathbb{C}} = 2p_{\mathbb{R}}$. For comparative experiments the number of real-valued parameters per layer should be equal (or at least as close as possible) between the real-valued and its complex-valued variant. This ensures that models have the same capacity. Performance differences are caused by the introduction of complex numbers as parameters and not by a capacity difference.

Consider the number of parameters in a fully-connected layer in the real case and in the complex case. Let n be the input dimension and m the number of neurons, then the number of parameters of a real-valued layer $p_{\mathbb{R}}$ and of a complex layer $p_{\mathbb{C}}$ is given by:

$$\begin{aligned} p_{\mathbb{R}} &= (n \times m) + m, \\ p_{\mathbb{C}} &= 2(n \times m) + 2m \end{aligned} \tag{3.1}$$

For an MLP with k hidden layers, and c output dimensions the number of real-valued parameters (without bias) is given by:

$$\begin{aligned} p_{\mathbb{R}} &= n \times m + k(m \times m) + m \times c, \\ p_{\mathbb{C}} &= 2(n \times m) + 2k(m \times m) + 2(m \times c) \end{aligned} \tag{3.2}$$

At first glance designing comparable multi-layer architectures, i.e. with the same number of real-valued parameters in each layer, is trivial. However, halving the number of neurons in every layer will not achieve parameter comparability. The number of neurons define the output dimensions of a layer and the following layer's input dimension. We addressed this problem

by constructing networks with an even number of hidden layers k and alternating the number of neurons per layer between m and $\frac{m}{2}$. We receive the same number of real parameters in each layer of a complex-valued MLP compared to a real-valued network. Let us consider the dimensions of outputs and weights with $k = 4$ hidden layers as an example. For the real-valued case:

$$\begin{aligned}
&(1 \times n)\overbrace{(n \times m_1)}^{\text{Input layer}} \rightarrow (1 \times m_1)\overbrace{(m_1 \times m_2)}^{\text{Hidden layer}} \\
\rightarrow (1 \times m_2)&\overbrace{(m_2 \times m_3)}^{\text{Hidden layer}} \rightarrow (1 \times m_3)\overbrace{(m_3 \times m_4)}^{\text{Hidden layer}} \\
\rightarrow (1 \times m_4)&\overbrace{(m_4 \times m_5)}^{\text{Hidden layer}} \rightarrow (1 \times m_5)\overbrace{(m_5 \times c)}^{\text{Output layer}} \\
&\rightarrow \overbrace{(1 \times c)}^{\text{Model output}}
\end{aligned} \tag{3.3}$$

where m_i is the number of (complex or real) neurons of the i-th layer. The equivalent using m_i complex-valued neurons would be:

$$\begin{aligned}
(1 \times n)(n \times \frac{m_1}{2}) &\rightarrow (1 \times \frac{m_1}{2})(\frac{m_1}{2} \times m_2) \\
\rightarrow (1 \times m_2)(m_2 \times \frac{m_3}{2}) &\rightarrow (1 \times \frac{m_3}{2})(\frac{m_3}{2} \times m_4) \\
\rightarrow (1 \times m_4)(m_4 \times \frac{m_5}{2}) &\rightarrow (1 \times \frac{m_5}{2})(\frac{m_5}{2} \times c) \\
&\rightarrow (1 \times c)
\end{aligned} \tag{3.4}$$

Another approach to design comparable architectures is to work with a parameter budget. Given a fixed budget of real parameters $p_{\mathbb{R}}$ we can define real or complex MLP with an even number $k \geq 0$ of hidden layers such that the network's parameters are within that budget. The k hidden layers and the input layer have the same number of real or complex neurons $m_{\mathbb{R}} = m_{\mathbb{C}}$. The number of neurons in the last layer is defined by the number of classes c.

$$m_{\mathbb{R}} = \begin{cases} -\frac{n+c}{2k} + \sqrt{(\frac{n+c}{2k})^2 + \frac{p_{\mathbb{R}}}{k}}, & \text{if } k > 0 \\ \frac{p_{\mathbb{R}}}{n+c}, & \text{otherwise} \end{cases} \tag{3.5}$$

$$m_{\mathbb{C}} = \begin{cases} -\frac{n+c}{2k} + \sqrt{(\frac{n+c}{2k})^2 + \frac{p_{\mathbb{R}}}{2k}}, & \text{if } k > 0 \\ \frac{p_{\mathbb{R}}}{2(n+c)}, & \text{otherwise} \end{cases} \tag{3.6}$$

In the case of a convolutional neural network designing an architecture with equal number of parameters is easier. Again, let f be the size in a one-dimensional kernel $f \times 1$ or 2-dimensional kernel $f \times f$ convolving over an input $w \times h \times d$. The number of parameters in a real-valued

convolutional layer $p_{\mathbb{R}}$ and of a complex-valued convolutional layer $p_{\mathbb{C}}$ with m filters is given by:

$$\begin{aligned} p_{\mathbb{R}} &= (f \times f \times d \times m), \\ p_{\mathbb{C}} &= 2(f \times f \times d \times m) \end{aligned} \tag{3.7}$$

For an architecture with $k + 1$ convolutional layers followed by a fully-connected layer with n resp. $n \times n$ input dimensions and c output dimensions the number of real-valued parameters (without bias) is given by:

$$\begin{aligned} p_{\mathbb{R}} &= (f \times f \times d \times m) + k(f \times f \times m \times m) + (w \times h \times d) \times f \times c, \\ p_{\mathbb{C}} &= 2(f \times f \times d \times m) + 2k(f \times f \times m \times m) + \end{aligned} \tag{3.8}$$

Pooling layers may be used to decrease the dimensionality for the last layer. Using pooling, however, will change the number of parameters proportionally in a real-valued and complex-valued model. To design a convolutional neural network architecture with the same number of parameters we fix the kernel size $f \times f$ and alternate between halving the number of complex-valued kernels (filters) $m_{\mathbb{C}} = 2m_{\mathbb{R}}$ and the full number of kernels $m_{\mathbb{C}} = m_{\mathbb{R}}$.

3.4 Experiments

To compare real and complex-valued MLPs and CNNs (Figures 3.1, 3.2) we investigated them in various classification tasks. The specific version of the MNIST data [12] used in our experiments is provided by Yan LeCunn and Corinna Cortes [1]. This version is a curated subset of the original MNIST data. It is split into 50,000 images as training set and a balanced test set of 10,000 images. The mass-centred black-and-white images are 28x28 pixels in size. We use a balanced validation set of 5,000 images randomly selected from the training data.

The CIFAR dataset is a labelled subset of a large collection of 32×32 pixel images [14]. The 80,000,000 tiny images collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton were curated by Alex Krizhevsky into two datasets: CIFAR-10 and CIFAR-100[2]. CIFAR-10 consists of 60,000 images classified into 10 classes. Each class is represented with 6,000 images. The data is divided into 6 balanced batches each with 1,000 images per class. One batch is provided as a test set by the authors. The other 5 are training data. For our validation we selected a single random batch from the training data. CIFAR-100 also consists of 60,000 images, but classified into 100 classes. Each class is represented with 600 images. It is also divided into 6 balanced batches each with 100 images per class. Again, one batch is provided as test set by the authors and we choose one batch as validation set from the training data.

[1] http://yann.lecun.com/exdb/mnist/

[2] https://www.cs.toronto.edu/ kriz/cifar.html

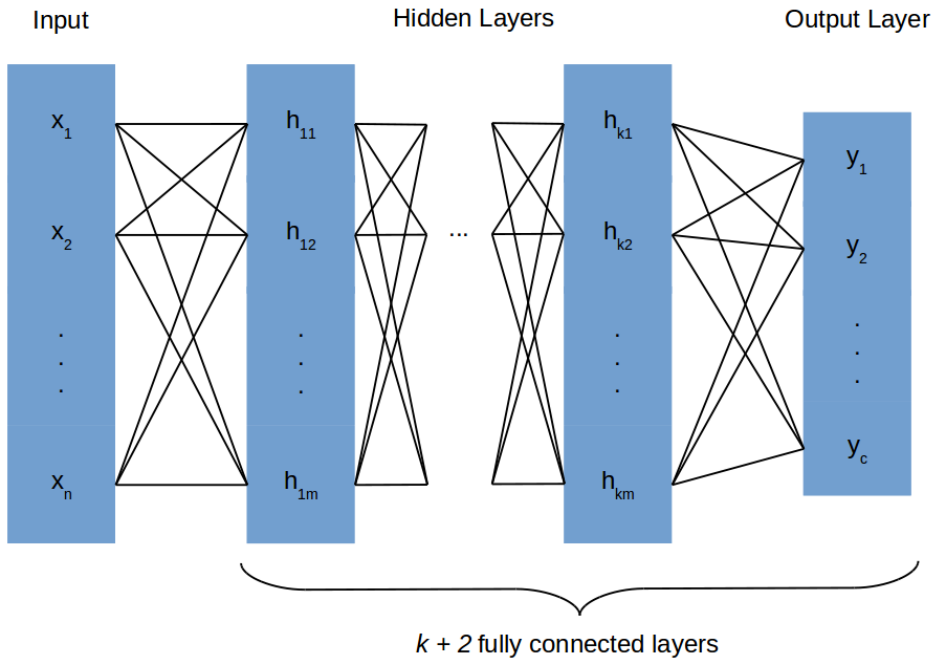

Figure 3.1: Multi-layer perceptron with k hidden layers. The number of hidden units per layer m computed with Equation 3.2 for Experiment 1 and Equations 3.5 resp. 3.6 for Experiment 2.

Reuters-21578 (Distribution 1.0) newswire topic classification is a document classification task [13]. The original dataset consists of 21,578 documents to be categorised into 90 classes, each representing a topic. We use the 'Modified Apte' split by David D. Lewis[3][4]. The training set consists of 9,603 documents and the test set of 3,299 docs documents. 8,676 documents are unused. We chose 960 random documents from the training set as a validation set. In all of the following experiments the task was to assign a single class to each real-valued data point:

- Experiment 1: We tested MLPs with $k = 0, 2, 4, 8$ hidden layers, fixed width of units in each layer in real-valued architectures and alternating 64 and 32 units in complex-valued architectures (Section 3.3). We did not apply a fixed parameter budget. We tested the models on MNIST digit classification, CIFAR-10 Image classification, CIFAR-100 image classification and Reuters topic classification. Reuters topic classification and MNIST digit classification use $m_{\mathbb{R}} = 64$ units per layer, CIFAR-10 and CIFAR-100 use $m_{\mathbb{R}} = 128$ units per layer.

- Experiment 2: We tested MLPs with fixed budget of 500,000 real-valued parameters. The MLPs have variable width according to the depth and the parameters. They are tested on

[3] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html
[4] http://www.daviddlewis.com/resources/testcollections/reuters21578/

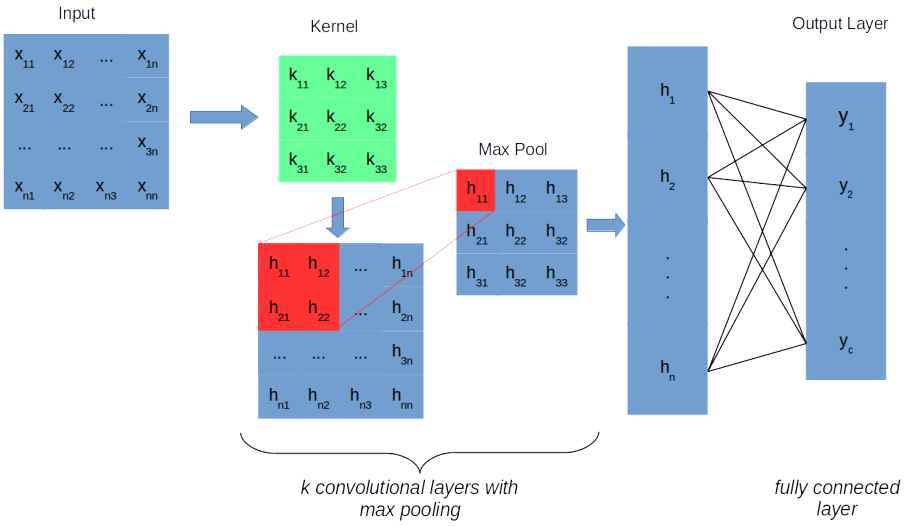

Figure 3.2: CNN architecture with $k+1$ convolutional layers each followed by a maximum pooling layer. The number of kernels per layer $m_{\mathbb{C}} = 2m_{\mathbb{R}}$ and the number of parameters respect Equation 3.8 for Experiment 3.

MNIST digit classification, CIFAR-10 Image classification, CIFAR-100 image classification and Reuters topic classification. We rounded the units in Equations 3.5 and 3.6 to the next integer.

- Experiment 3: We tested architectures with $k = 0, 2$ convolutional layers each followed by a Max Pooling layer. The convolutional blocks are followed by a single fully-connected layer with fixed number of units c. We tested the models on MNIST digit classification, CIFAR-10 Image classification, CIFAR-100 image classification and Reuters topic classification. We used fixed-size kernels 3×3 (3×1 for Reuters) and striding of 1×1. The maximum pooling relies on a pool of size 2×2 (2×1 for Reuters). Each convolutional layer trains an alternating number of $m_{\mathbb{C}} = 64, m_{\mathbb{R}} = 128$ filters/kernels.

We used the weight initialisation presented by Trabelsi et al. [27] for our experiments. To reduce the impact of the initialisation we trained each model 10 times. Each run trained the model over 100 epochs with an Adam optimisation. We used categorical or binary cross entropy loss depending on the task. We used $sigmoid(|z|^2)$ or $softmax(|z|^2)$ as the activation function for the last fully-connected layer.

3.5 Results

Tables 3.1, 3.2, 3.3, 3.4 show the results for MLPs with variable depth, fixed width and no parameter budget (Experiment 1). Tables 3.5, 3.6, 3.7, 3.8 show the results for MLPs with variable width according to the targetted depth and a fixed parameter budget of $500,000$ real-valued parameters (Experiment 2). Tables 3.9, 3.10, 3.11, 3.12 show the results for CNNs with fixed kernel sizes, but reduced number of kernels (Experiment 3). In our experiments the achieved accuracy of complex- and real-valued MLPs are close to each other. Similarly for real- and complex-valued CNNs. Nevertheless, the real networks consistently outperform complex-valued networks. The complex-valued neural networks often fail to learn any structure from the data or fail the training process.

Complex-valued MLPs and CNNs can be used to classify short dependencies as a bag-of-words (e.g MNIST digits or short texts). For the two image classification tasks CIFAR-10 and CIFAR-100 the results indicate that a complex-valued MLP does not learn any structure. These two tasks require larger weight matrices in the first layer and weight initialisation is a significant problem to be solved with an MLP. However, CNNs improve performance while reducing the number of parameters required.

The best non-linearity in complex MLP is the split-activation with rectifier linear unit *relu*, similar to the real-valued models. The hyperbolic tangents *tanh* outperforms *relu* in the real-valued case. However, the results using the rectifier linear unit *relu* are much more stable. Despite the similarity of the activation functions $|z|^2$ and $|z|$, their performance in all tasks differ significantly. The magnitude $|z|$ consistently outperforms the squared magnitude $|z|^2$. In convolutional neural networks we observe a different behaviour. Firstly, the performance difference between activation functions is smaller. The magnitude $|z|$ still outperforms the squared magnitude $|z|^2$. Moreover, it also outperforms all other activation functions for both real and complex CNNs in all considered tasks.

As expected, we observe that with a fixed number of neurons per layer (Experiment 1) and increasing depth, the accuracy increases in both cases. As we are increasing the total number of parameters, the model capacity increases. An exception here is Reuters topic classification where the performance decreases with increasing depth. When choosing the number of neurons per layer according to a given parameter budget (Experiment 2 with Equations 3.5, 3.6), the performance decreases significantly as model depth increases. Overall, the width of each layer is more important than the overall depth of the complete network. Increasing the number of convolutional layers in CNNs improves their performances for all tasks except Reuters newswire classification.

We found that the performance variance between the 10 initialisations of MLPs is very high for the complex-valued cases. We hypothesized that weight initialisation in complex MLPs becomes much more difficult with increasing depth. Hence, their performance is highly unstable.

Table 3.1: Test accuracy of a multi-layer perceptron consisting of $k + 2$ layers each with 64 neurons (alternating 64 and 32 neurons in the complex MLP) and an output layer with $c = 10$ neurons on MNIST digit classification task (Experiment 1). Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	MNIST	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	50,816	*identity*	0.9282	0.9509
		tanh	0.9761	0.9551
		relu	0.9780	0.9710
		$\lvert z\rvert^2$	0.9789	0.9609
		$\lvert z\rvert$	0.9770	0.9746
k = 2	59,008	*identity*	0.9274	0.9482
		tanh	0.9795	0.8923
		relu	0.9804	0.9742
		$\lvert z\rvert^2$	0.9713	0.6573
		$\lvert z\rvert$	0.9804	0.9755
k = 4	67,200	*identity*	0.9509	0.9468
		tanh	0.9802	0.2112
		relu	0.9816	0.9768
		$\lvert z\rvert^2$	0.8600	0.2572
		$\lvert z\rvert$	0.9789	0.9738
k = 8	83,584	*identity*	0.9242	0.1771
		tanh	0.9796	0.1596
		relu	0.9798	0.9760
		$\lvert z\rvert^2$	0.0980	0.0980
		$\lvert z\rvert$	0.9794	0.1032

Table 3.2: Test accuracy of a multi-layer perceptron consisting of $k + 2$ layers each with 64 neurons (alternating 64 and 32 neurons in the complex MLP) and an output layer with $c = 46$ neurons on Reuters topic classification (Experiment 1). Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	Reuters	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	642,944	*identity*	0.8116	0.7939
		tanh	0.8117	0.7912
		relu	0.8081	0.7934
		$\|z\|^2$	0.8050	0.7885
		$\|z\|$	0.8068	0.7992
k = 2	651,136	*identity*	0.8005	0.7836
		tanh	0.7978	0.7320
		relu	0.7921	0.7854
		$\|z\|^2$	0.7725	0.6874
		$\|z\|$	0.7996	0.7823
k = 4	659,328	*identity*	0.7925	0.7787
		tanh	0.7814	0.4199
		relu	0.7734	0.7671
		$\|z\|^2$	0.5895	0.0650
		$\|z\|$	0.7863	0.7694
k = 8	675,712	*identity*	0.7929	0.7796
		tanh	0.7542	0.1861
		relu	0.7555	0.7676
		$\|z\|^2$	0.0053	0.0053
		$\|z\|$	0.7671	0.7524

Table 3.3: Test accuracy of a multi-layer perceptron consisting of $k + 2$ layers each with 128 neurons (alternating 128 and 64 neurons in the complex MLP) and an output layer with $c = 10$ neurons on CIFAR-10 image classification task (Experiment 1). Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	CIFAR-10	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	394,496	*identity*	0.4044	0.1063
		tanh	0.4885	0.1431
		relu	0.4902	0.4408
		$\|z\|^2$	0.5206	0.1000
		$\|z\|$	0.5256	0.1720
k = 2	427,264	*identity*	0.4039	0.1000
		tanh	0.5049	0.1672
		relu	0.5188	0.496
		$\|z\|^2$	0.1451	0.1361
		$\|z\|$	0.5294	0.1000
k = 4	460,032	*identity*	0.4049	0.1000
		tanh	0.4983	0.1549
		relu	0.8445	0.6810
		$\|z\|^2$	0.1000	0.1000
		$\|z\|$	0.5273	0.1000
k = 8	525,568	*identity*	0.4005	0.1027
		tanh	0.4943	0.1365
		relu	0.5072	0.4939
		$\|z\|^2$	0.1000	0.1000
		$\|z\|$	0.5276	0.1000

Table 3.4: Test accuracy of a multi-layer perceptron consisting of $k + 2$ layers each with 128 neurons (alternating 128 and 64 neurons in the complex MLP) and an output layer with $c = 100$ neurons on CIFAR-100 image classification task (Experiment 1). Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	CIFAR-100	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	406,016	*identity*	0.1758	0.0182
		tanh	0.2174	0.0142
		relu	0.1973	0.1793
		$\lvert z\rvert^2$	0.2314	0.0158
		$\lvert z\rvert$	0.2423	0.0235
k = 2	438,784	*identity*	0.1720	0.0100
		tanh	0.2314	0.0146
		relu	0.2400	0.2123
		$\lvert z\rvert^2$	0.0143	0.0123
		$\lvert z\rvert$	0.2411	0.0100
k = 4	471,552	*identity*	0.1685	0.0100
		tanh	0.2178	0.0157
		relu	0.2283	0.2059
		$\lvert z\rvert^2$	0.0109	0.0100
		$\lvert z\rvert$	0.2313	0.0100
k = 8	537,088	*identity*	0.1677	0.0100
		tanh	0.2000	0.0130
		relu	0.2111	0.1956
		$\lvert z\rvert^2$	0.0100	0.0100
		$\lvert z\rvert$	0.2223	0.0100

Table 3.5: Test accuracy of a multi-layer perceptron consisting of $k + 2$ dense layers with an overall budget of $500,000$ real-valued parameters on MNIST digit classification (Experiment 2). The last layer consists of $c = 10$ neurons. Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Units		Activation function φ	CIFAR-10	
	$m_{\mathbb{R}}$	$m_{\mathbb{C}}$		$\mathbb{R}$	$\mathbb{C}$
k = 0	630	315	*identity*	0.9269	0.9464
			tanh	0.9843	0.9467
			relu	0.9846	0.9828
			$\lvert z\rvert^2$	0.9843	0.9654
			$\lvert z\rvert$	0.9857	0.9780
k = 2	339	207	*identity*	0.9261	0.9427
			tanh	0.9852	0.6608
			relu	0.9878	0.9835
			$\lvert z\rvert^2$	0.9738	0.8331
			$\lvert z\rvert$	0.9852	0.9748
k = 4	268	170	*identity*	0.9254	0.2943
			tanh	0.9838	0.2002
			relu	0.9862	0.9825
			$\lvert z\rvert^2$	0.8895	0.2875
			$\lvert z\rvert$	0.9846	0.9870
k = 8	205	134	*identity*	0.9250	0.1136
			tanh	0.9810	0.1682
			relu	0.9851	0.9824
			$\lvert z\rvert^2$	0.0980	0.0980
			$\lvert z\rvert$	0.9803	0.1135

Table 3.6: Test accuracy of a multi-layer perceptron consisting of $k + 2$ dense layers with an overall budget of $500,000$ real-valued parameters on Reuters topic classification (Experiment 2). The last layer consists of $c = 46$ neurons. Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Units		Activation function φ	Reuters	
	$m_{\mathbb{R}}$	$m_{\mathbb{C}}$		$\mathbb{R}$	$\mathbb{C}$
k = 0	50	25	*identity*	0.8072	0.7970
			tanh	0.8112	0.7832
			relu	0.8054	0.7925
			$\|z\|^2$	0.8037	0.7929
			$\|z\|$	0.8059	0.7912
k = 2	49	25	*identity*	0.7992	0.7809
			tanh	0.7952	0.7289
			relu	0.7898	0.7751
			$\|z\|^2$	0.7778	0.6887
			$\|z\|$	0.7716	0.7911
k = 4	49	25	*identity*	0.7636	0.7854
			tanh	0.7796	0.4550
			relu	0.7658	0.7676
			$\|z\|^2$	0.5823	0.0289
			$\|z\|$	0.7809	0.7573
k = 8	48	24	*identity*	0.7760	0.7663
			tanh	0.7449	0.1799
			relu	0.7182	0.7484
			$\|z\|^2$	0.0053	0.0053
			$\|z\|$	0.7449	0.7302

Table 3.7: Test accuracy of a multi-layer perceptron consisting of $k + 2$ dense layers with an overall budget of 500,000 real-valued parameters on CIFAR-10 image classification (Experiment 2). The last layer consists of $c = 10$ neurons. Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Units		Activation function φ	MNIST	
	$m_{\mathbb{R}}$	$m_{\mathbb{C}}$		$\mathbb{R}$	$\mathbb{C}$
k = 0	162	81	*identity*	0.4335	0.1006
			tanh	0.5032	0.1676
			relu	0.5007	0.4554
			$\lvert z\rvert^2$	0.5179	0.1006
			$\lvert z\rvert$	0.5263	0.2381
k = 2	148	77	*identity*	0.4069	0.1000
			tanh	0.5205	0.1673
			relu	0.5269	0.4963
			$\lvert z\rvert^2$	0.1395	0.1273
			$\lvert z\rvert$	0.5315	0.1000
k = 4	138	74	*identity*	0.4052	0.1000
			tanh	0.5218	0.1475
			relu	0.5203	0.4975
			$\lvert z\rvert^2$	0.1065	0.1010
			$\lvert z\rvert$	0.5234	0.1000
k = 8	123	69	*identity*	0.4050	0.1003
			tanh	0.5162	0.1396
			relu	0.5088	0.4926
			$\lvert z\rvert^2$	0.1000	0.1000
			$\lvert z\rvert$	0.5194	0.1000

Table 3.8: Test accuracy of a multi-layer perceptron consisting of k+2 dense layers with an overall budget of 500,000 real-valued parameters on CIFAR-100 image classification (Experiment 2). The last layer consists of $c = 100$ neurons. Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Units		Activation function φ	CIFAR-100	
	$m_{\mathbb{R}}$	$m_{\mathbb{C}}$		$\mathbb{R}$	$\mathbb{C}$
k = 0	158	79	*identity*	0.2807	0.0314
			tanh	0.2308	0.0193
			relu	0.2153	0.1935
			$\|z\|^2$	0.2364	0.0124
			$\|z\|$	0.2439	0.0279
k = 2	144	75	*identity*	0.1723	0.0100
			tanh	0.2440	0.0203
			relu	0.2481	0.2224
			$\|z\|^2$	0.0155	0.0151
			$\|z\|$	0.2453	0.0100
k = 4	135	72	*identity*	0.1727	0.0100
			tanh	0.2397	0.0150
			relu	0.2381	0.2147
			$\|z\|^2$	0.0122	0.0100
			$\|z\|$	0.2390	0.0100
k = 8	121	67	*identity*	0.1706	0.0100
			tanh	0.2209	0.0164
			relu	0.2167	0.2027
			$\|z\|^2$	0.0100	0.0100
			$\|z\|$	0.2191	0.0100

Table 3.9: Test accuracy of convolutional neural network consisting of $k+1$ convolutional layers with 128 filters real-valued case or 64 filters in the complex-valued case. Each convolutional layer is followed by a Max-Pooling and eventually by an output layer with $c = 10$ neurons on MNIST digit classification task (Experiment 3). Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	MNIST	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	217,472	*identity*	0.9812	0.9804
		tanh	0.9820	0.9814
		relu	0.9855	0.9885
		$\|z\|^2$	0.9854	0.9881
		$\|z\|$	0.9862	0.9888
k = 2	297,344	*identity*	0.9900	0.9865
		tanh	0.9907	0.9786
		relu	0.9924	0.9922
		$\|z\|^2$	0.9926	0.9924
		$\|z\|$	0.9937	0.9932

Table 3.10: Test accuracy of convolutional neural network consisting of $k+1$ convolutional layers with 128 filters real-valued case or 64 filters in the complex-valued case. Each convolutional layer is followed by a Max-Pooling and eventually by an output layer with $c = 46$ neurons on Reuters topic classification task (Experiment 3). Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	Reuters	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	29,434,496	*identity*	0.7943	0.7858
		tanh	0.7956	0.7854
		relu	0.7867	0.7743
		$\|z\|^2$	0.7832	0.7716
		$\|z\|$	0.7872	0.7680
k = 2	7,446,912	*identity*	0.7792	0.7818
		tanh	0.7778	0.7796
		relu	0.7631	0.7703
		$\|z\|^2$	0.7386	0.7462
		$\|z\|$	0.7569	0.7271

Table 3.11: Test accuracy of convolutional neural network consisting of $k+1$ convolutional layers with 128 filters real-valued case or 64 filters in the complex-valued case. Each convolutional layer is followed by a Max-Pooling and eventually by an output layer with $c = 10$ neurons on CIFAR-10 image classification task (Experiment 3). Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	CIFAR-10	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	291,456	*identity*	0.6151	0.6157
		tanh	0.6169	0.6140
		relu	0.6663	0.6540
		$\|z\|^2$	0.6186	0.6195
		$\|z\|$	0.6678	0.6682
k = 2	303,488	*identity*	0.6867	0.6606
		tanh	0.7157	0.6185
		relu	0.7302	0.7156
		$\|z\|^2$	0.6683	0.6476
		$\|z\|$	0.7756	0.7471

Table 3.12: Test accuracy of convolutional neural network consisting of $k+1$ convolutional layers with 128 filters real-valued case or 64 filters in the complex-valued case. Each convolutional layer is followed by a Max-Pooling and eventually by an output layer with $c = 100$ neurons on CIFAR-100 image classification task (Experiment 3). Selected best from 10 runs. Each run was trained for 100 epochs.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	CIFAR-100	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	2,883,456	*identity*	0.2922	0.2851
		tanh	0.2973	0.2940
		relu	0.3515	0.3382
		$\|z\|^2$	0.3141	0.3179
		$\|z\|$	0.3735	0.3698
k = 2	349,568	*identity*	0.3842	0.3551
		tanh	0.4167	0.2881
		relu	0.4048	0.3991
		$\|z\|^2$	0.3241	0.2358
		$\|z\|$	0.4270	0.3765

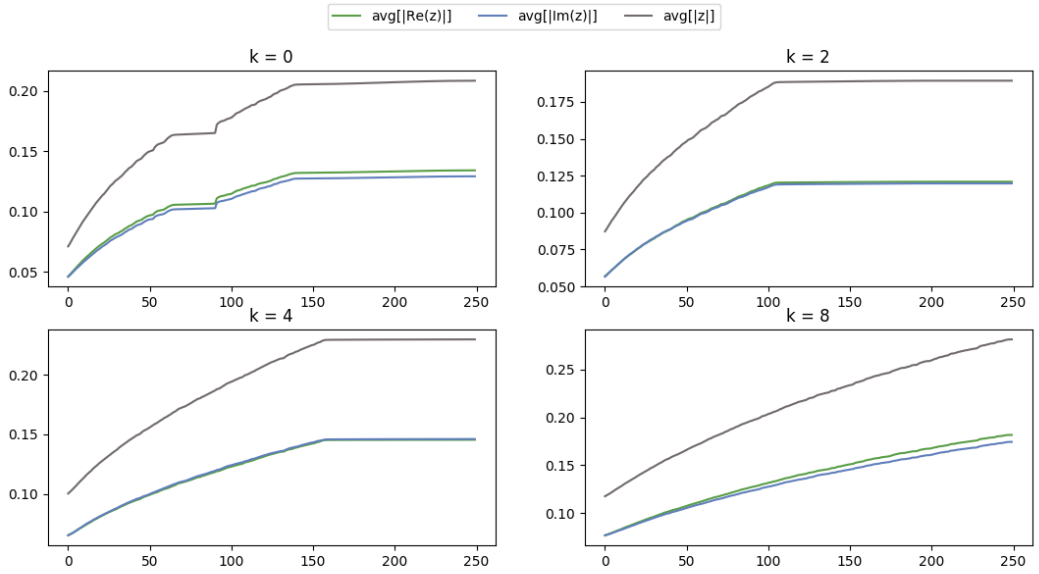

Figure 3.3: Average absolutes of real $|Re(z)|$, imaginary parts $|Im(z)|$ and average complex magnitude $|z|$ of all weights W_i over training epochs of the MNIST classification task.

We confirmed this by training a complex MLP ($k = 2$, *tanh*) with 100 runs (instead of 10 runs) on the Reuters classification task. The performance gap decreases if a setting is initialised more often. We found a test accuracy of 0.7748 in complex-valued case in comparison to 0.7978 in the real-valued case (Table 3.2). CNNs do not suffer from the same problem. Performance for CNNs are very stable.

3.6 Synthetic Tasks

In further examination of the training process we observed that the imaginary parts of the complex weights always follow the real parts of the weights in their average magnitudes. This is presented in two examples from our experiments in Figures 3.3 and 3.4.

This behaviour can be further illustrated with two synthetic classification tasks. Random complex data points $x \in \mathbb{C}^n$ are to be classified using a complex-valued MLP according to the quadrant of its sum $\sum_{i=0}^{n} x_i$ or if it is close to the origin (Figure 3.5). Real data points $x \in \mathbb{R}^n$ follow the same rule. This is equivalent to a projection of complex data points to $\mathbb{R}$ (Figure 3.6). We classify $n = 10,000$ complex resp. real input with Gaussian noise $\sigma = 0.2$ and $d = 25$ dimensions using a complex-valued MLP with $k = 2$ hidden layers and with each $m = 64$ units per layer. Again, we observe that the weight initialisation is a significant problem. However, the complex model can reliably approximate the underlying complex or real functions achieving

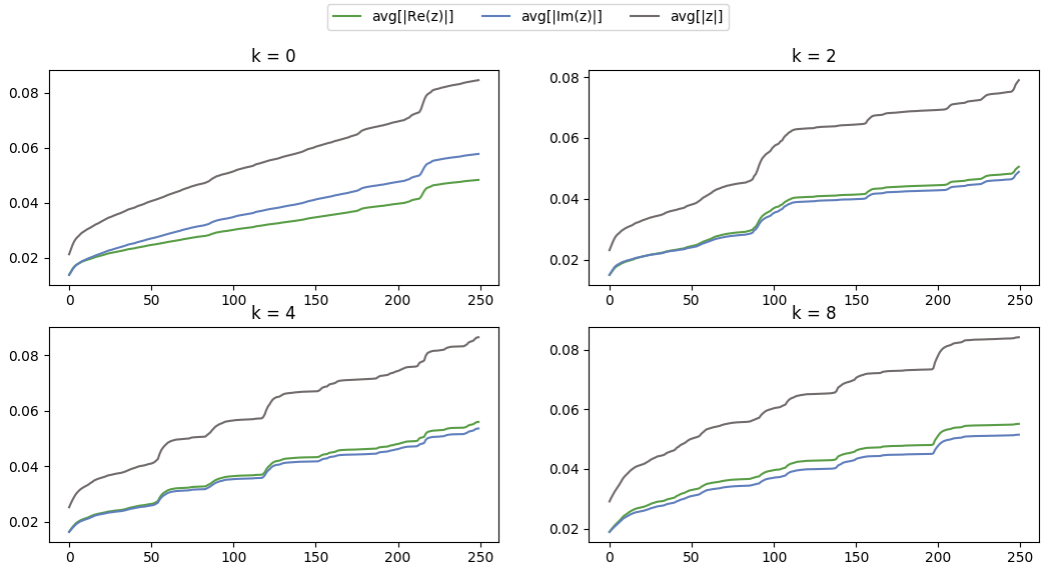

Figure 3.4: Average absolutes of real $|Re(z)|$, imaginary parts $|Im(z)|$ and average complex magnitude $|z|$ of all weights W_i over training epochs of the Reuters classification task.

training accuracy of 0.981 and a test accuracy of 0.908. We observe that our complex-valued weights behave differently depending on the input type. Using complex-valued input the real and imaginary parts develop independently and then reach convergence independently (Figure 3.7). When training with real-valued synthetic data the magnitudes of imaginary parts follow the real parts very closely (Figure 3.8). Independently of the initialisation, the real parts converge a few epochs before the imaginary part. In our synthetic experiments this is between 3 and 5 epochs before.

To explain the training behaviour of the imaginary weights, we consider the computations within a complex-valued MLP (Figure 3.9). Consider the information flow within an complex-valued MLP.

Real-valued data does not require the interaction that is provided by complex numbers. The imaginary part of the input $Im(x)$ is zero, so the multiplications in Equation 2.8 simplifies to:

$$\begin{aligned} Re(xW) &= Re(x)Re(W), \\ Im(xW) &= Re(x)Im(W) \end{aligned} \tag{3.9}$$

The real parts $Re(x)$ and $Re(W)$ dominate the overall classification of a real-valued data point. We see that the real and imaginary parts of the weights act identically on the input in order to reach a classification result. Hence, the final classification is the average of two identical or very

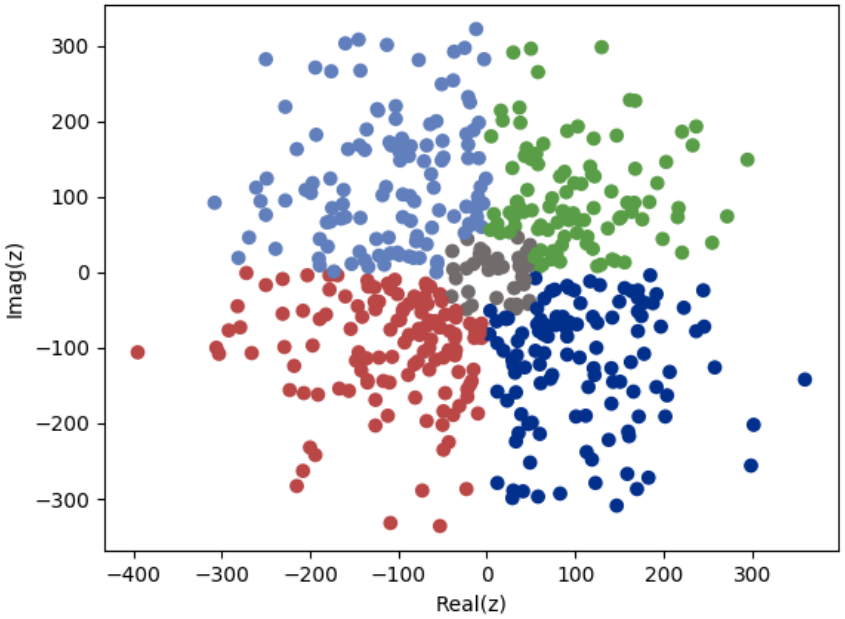

Figure 3.5: Example of synthetic data's class distribution. The class is determined by the sum's location consisting of complex elements $x_i \in \mathbb{C}$ in the input vector. Colours indicate different classes.

similar parallel classifications. The complex-valued logits of the last layer are simply averaging the output. These logits are then mapped to a probability distribution over the classes. If in the training phase the averaged absolute values of the weight's imaginary part follow those of the real parts, the imaginary part of the input is either distributed exactly the same way as its real part, or the considered task simply does not benefit from using complex-valued hypothesis. Hence this behaviour in the training process, can be used to identify tasks that may benefit from a complex-valued hypothesis. It can also be used as a criterion to select tasks for further in-depth experimenting and hyperparameter optimisation under a complex-valued hypothesis.

3.7 Discussion

For many applications that involve data which is interpretable on the complex plane complex-valued neural networks have already shown that they are superior [35]. However, the selected tasks in our work use real-valued input data.

Complex-valued MLPs consistently perform slightly worse or significantly worse than real-valued neural networks. This can be observed across activation functions and depth. This seems

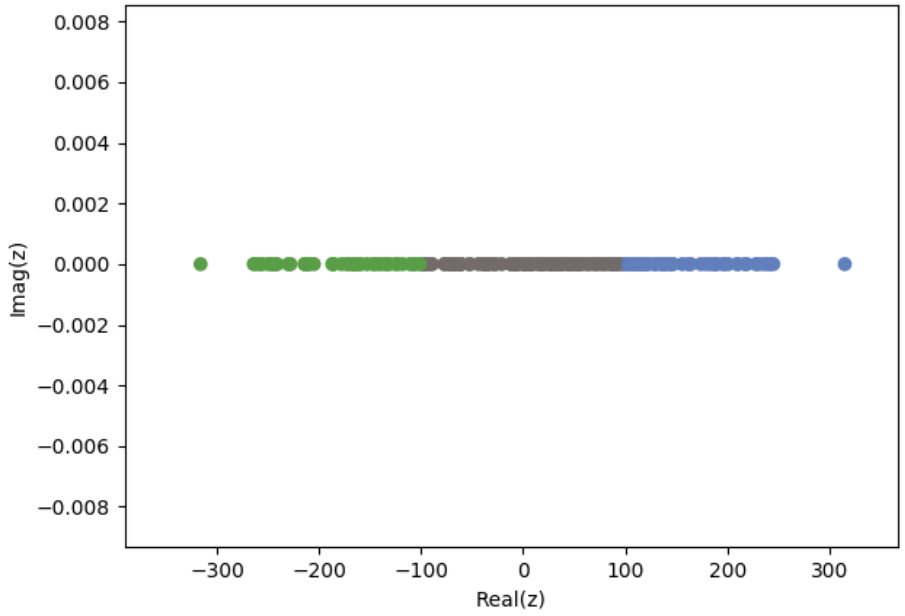

Figure 3.6: Example of synthetic data's class distribution. The class is determined by the sum's location consisting of real elements $x_i \in \mathbb{R}$ in the input vector. Colours indicate different classes.

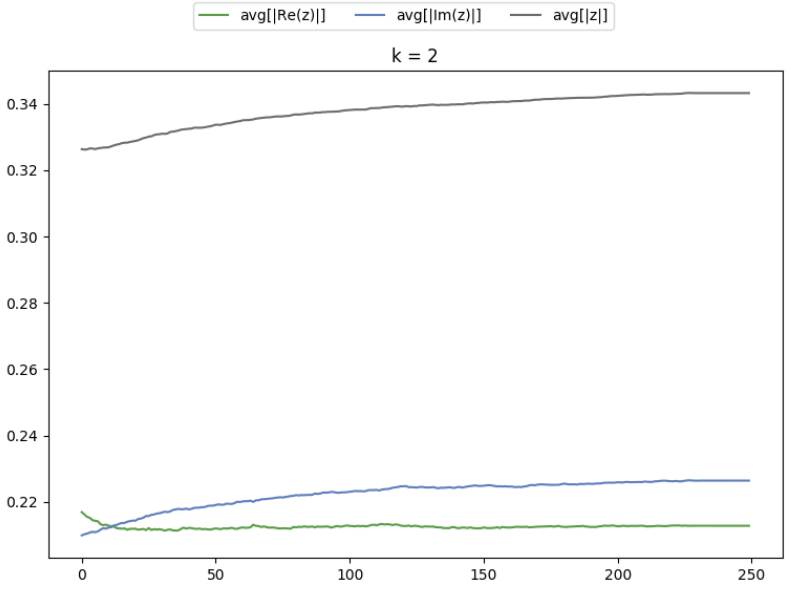

Figure 3.7: Example of independent learning behaviour of real and imaginary parts in the classification of synthetic complex-valued data. The real and imaginary parts change independently over the training. The exact trajectory of the graph depends on the weight initialisation.

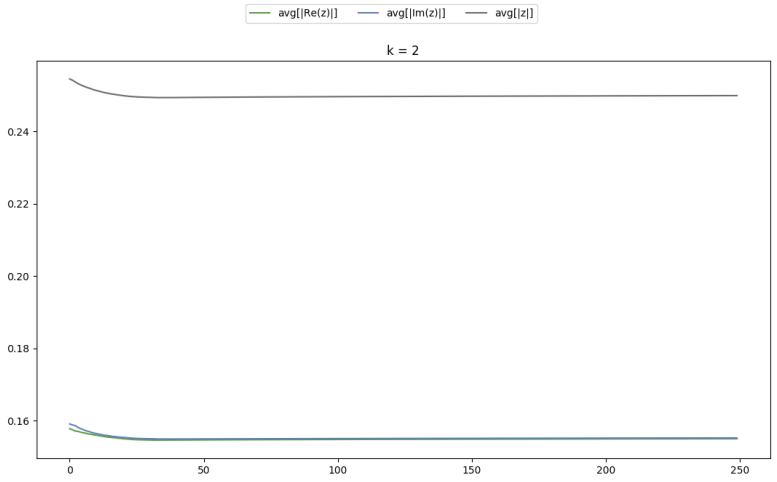

Figure 3.8: Example of dependent learning behaviour of the complex weights in the classification of synthetic real-valued data. The imaginary part follows the real part of the weight with every epoch. The exact trajectory of the graph depends on the weight initialisation.

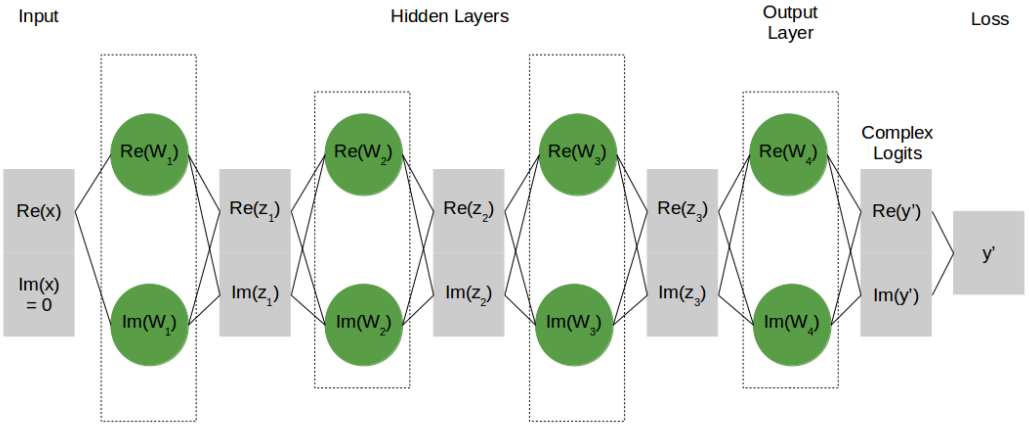

Figure 3.9: Interaction of real parts $Re(W_i)$, imaginary parts $Im(W_i)$ with the real-valued input x and complex output z_i of the i-th layer of an MLP with $k = 2$ layers.

counter-intuitive at first, since every real value is just a complex number with a zero imaginary part. Solving a real-valued problem with a complex-valued model should approximate the function at least as well as a real-valued model. If there are any features to be discovered, that a real-valued model can not capture (e.g. phases) a complex-valued hypothesis should be able to learn these features, too. Therefore, the question as to why complex-valued models are inferior arises. Through our previous analysis, we know that the neural network learns two separate and equivalent classifications. These classifications are learned in the real and in imaginary parts of the model. The imaginary weights follow the real weights in their training and come to the same classification. This results in two separate, but connected real-valued classifiers. However, each classifier contributes its own noise into their interaction. We can identify tasks that would benefit from complex-valued neural network by comparing the training behaviour (e.g. by the average absolute values) of real and imaginary weights. If the imaginary part does not follow the real parts behaviour across epochs, the task benefits from assuming a complex hypothesis. Real-valued classification tasks do not benefit from complex-valued hypothesis. It also suggests that a complex network requires explicit incentives in a real-valued loss or explicit encoding of input information to learn valuable intermediate complex representations. A possible way to encode additional input information is preprocessing the input data to a complex representation (e.g. transformations or embeddings). In this context, frequency-based representations are a natural interpretation in the complex domain. Alternatively, embeddings that capture additional positional information allow MLPs to classify based on position while at the same time treating the input as orderless. We investigate this principle in Chapters 4 and 5.

Considering CNNs across activation functions, complex-valued networks also perform similarly or slightly worse compared to their real-valued counterparts. Only under certain activation functions and depth, do they perform better. This performance difference, however, is insignificant (< 0.001). As expected, the context-aware CNNs perform better than MLPs on image classification tasks. Generally, CNNs and MLPs perform similarly well. However, convolutions reduce the number of parameters significantly. In direct comparison with MLPs, CNNs perform similarly or better. This may be due to the nature of our chosen tasks that require context-aware decisions. Specifically the image-based task benefit significantly from the use of a CNN. Furthermore, complex- and real-valued CNNs require fewer parameters by design.

The Liouville Theorem states that we need to choose an activation function that is either unbounded and/or non-holomorphic. Existing literature suggests that using unbounded activation functions may even improve the learning process in deep neural architectures by reducing the vanishing gradient problem [61]. We empirically support this claim with our observations. The split-activation variant of the Rectifier Linear Unit appears to be much more stable in deeper networks. For CNNs the best activation is the magnitude which behaves similarly to the rectifier linear unit.

We found that the training process often fails, due to singularities (poles, removable singular-

ities, essential singularities or isolated singularities) within the domain of the activation function. Singularities may break the training process. Generally, a singularity (or singular point) is a point z_0 in the function's $f(z)$ domain where it fails to be analytic. Not every singularity type is equivalently problematic for a gradient-based optimisation processes. Certain types of singularities require active methods in order to avoid these points. Others may not affect the training process at all, due to the nature of gradient descent optimisation. Poles are an example of usually unproblematic singularities. Functions act regularly in the neighbourhood of these singularities, but are infinite at these singular points. Further, a function does not need to be differentiable at all points to be used as an activation function. If a function is only non-differentiable at single specific point, it is very unlikely for the training process to break by 'hitting' these specific points. Modern frameworks assign a fixed scalar as a gradient or return the nearest derivative to a singular point to prevent a failed training process.

Since CNNs use fewer parameters by design, they are also less affected by singularities and difficult initialisations than MLPs.

When designing complex-valued neural networks, we must consider boundedness, the behaviour of a function around singular points and, real/complex differentiability of the entire function. To reduce the problem of a failing training process, we should choose unbounded activation functions which only contain removable or unproblematic singularities and which are complex and/or real differentiable at most points. Depending on the type of singularity, we need to employ other methods to avoid these points (i.e. constraint training). It is not possible to make a generalising statement about the behaviour of activations functions in the complex plane. Consequently, each function should be examined individually with respect to singular points. Other methods to stabilise the training may originate from the real domain. Normalisation methods like batch normalisation, weight normalisation and gradient clipping can be used to support the training process and its stability.

ReLU is the best performing non-linearity for both real and complex MLPs. ReLU is an unbounded, not entirely differentiable function with a singularity at $z_0 = 0$. Existing empirical evidence repeatedly confirmed the success of ReLU in the real domain. This is further supported by theoretical work on the consequences of unboundedness in deep neural networks [61]. Since it does not 'squash' the values into an interval, the gradients do not rapidly decrease smaller through layers, thus mitigating the vanishing gradient problem. In the complex domain it shows its viability as a split activation function. However, this may also be due to the setting we chose to investigate. We discovered that using the complex-valued MLP on real-valued data learns two parallel classifications with two similar weights for real and imaginary parts. By applying the same activation function within each of these, only allowing interaction of the parts in matrix multiplications, we simulate two identical parallel ReLU-activated classifications. In (complex) CNN taking both real and imaginary parts is important, due pooling function that is based on magnitudes. The magnitude function behaves very similarly to the rectifier linear unit in the real-

valued case. In the complex case it equally takes both the real and imaginary part into account. The maximum (magnitude) pooling in CNNs also contributes to the increased performance in comparison to other activation functions, since they are both based on the absolute value.

The training process was found to fail more often with increasing depth. With every layer that applies an activation function containing singular points, the training stability decreases. Each application of an activation function adds more singular points to the overall loss function. Hence, the likelihood of encountering a singular point in deeper models increases. Further, increasing depth also seem to have an impact on the performance. The classification performance varies drastically between different runs if the model depth increases. Initialisations with not entirely differentiable activation functions become more critical with increasing depth [62]. We use Trabelsi et al. [27] variance-scaled complex initialisation. This initialisation method is inspired by Glorot and Bengio [63]. This observation cannot yet be fully explained for complex-valued neural networks and lacks further empirical evidence. This area requires further research. The training process of complex-valued CNN fails less often, since CNNs have fewer parameters by design. The separation of filters provided by CNNs with pooling makes their training more stable. Even if a single filter encounters a singularity, other filters are not affected. This provides a complex-valued CNN with a certain level of fail safety.

The budget networks (Experiment 2) outperform the alternating-fixed-width networks (Experiment 1), because they use significantly more parameters per layer. In Experiment 2 we have chosen a budget of $500,000$ real-valued parameters which is purposefully oversized for most of the chosen benchmark tasks to show their use of parameters. In shallower networks these parameters are distributed over fewer layers making them wider.

We found that width is more important than depth for our classification tasks. Networks with $k = 0$ hidden layers often outperform networks with $k > 0$. Wider networks tend to memorise the training data while deeper networks learn new features with every layer. Wide networks put memorisation before generalisation and abstraction. Benchmark tasks are simple tasks and the limited data complexity rewards simply recalling the data. For example wide MLPs in the MNIST memorise possible digit variations rather than abstracting to more general features of digit images. This was found by monitoring the training and test accuracy. In both the real and complex case the training accuracy converges significantly higher than the test accuracy ($3 - 5$ percent). Shallow and wide MLPs tend to overfit. However, the over-fitting property of the complex-valued neural network tends to be less intense ($2 - 3$ percent). This indicates that complex-valued neural networks have a regularising effect on the training process.

We found that the CIFAR benchmarks are not solvable with MLPs, but that the classification is entirely random using a complex MLP. Generally, long sequential inputs are difficult or even impossible to learn for MLPs, since the image is flattened before being used as an input to the network. This flattening destroys the 2D dependencies within an image. Learning the long-distance dependencies between input features (pixels) is difficult for an MLP and exceed their

capabilities in larger images. However, complex-valued MLPs do not learn any dependencies within the input data. This finding was unexpected, because a complex MLP should have at least the same expressibility compared to real MLPs. An explanation for this observation is the unstable training process we discussed in previous findings. The complex-valued MLP simply does not converge at (local) optima. As expected, the two CIFAR benchmarks can better be approached with context-sensitive convolutions.

Certain choices made in this work limit the interpretability of our findings. We are bound by our selection of activation and pooling functions. A wider range may have led to improved results in complex-valued neural networks. Further investigation could use our current findings and include other activation functions like Hirose [35] hyperbolic tangents or *modReLU*. Further, our chosen benchmark tasks may be too easy for the MLPs. Other benchmarks task could include audio classification tasks, question classification tasks or further synthetic tasks. In our image-based tasks the images are preprocessed by flattening the 2D image into a vector. An alternative approach that respects the 2D structure may be investigated in future work. Complex convolutions have been suggested and tested in existing literature [54, 56]. We focused on MLPs, because they are very frequently used as classifiers or as part in many larger architectures. We purposefully did not apply methods to stabilise training like batch normalisation, weight normalisation or gradient clipping, since we did not want to inflate our results. Instead we aimed for a simple comparison in order to compare the impacts of real and complex numbers, not the impact of additional methods.

3.8 Conclusion

In summary, we found that real-valued models pose an upper performance bound in real-valued tasks when compared to complex-valued models of the same capacity. Real and imaginary parts act identically on the input. Hence, the MLP is simply learning two separate, but equivalent classifications. We found that singularities in activation functions may lead to failed training processes. However, not all singularities have a negative impact. When designing complex-valued neural networks, we need to consider boundedness, the behaviour of activation functions around singular points and real and complex differentiability of the entire function. We found that the split-activation function ReLU outperforms other activation functions. MLPs in our simple benchmark tasks prefer wide networks rather than deep networks, indicating a preference of memorisation over abstraction. Complex CNNs also achieve similar results to real CNNs, but prefer the magnitude as activation function and are more suitable for context-sensitive tasks.

This work contributes to the growing literature on complex-valued neural networks. We have added an in-depth investigation of complex-valued neural networks for classification. We elaborate on various problems that appear when designing and training complex neural networks and suggest possible solutions. We have discussed the impact of singularities and differentiability

of activation functions. Additionally, we generalised a criterion to identify tasks which may benefit from a complex-valued hypothesis early in the training process. We also derived various recommendations for the design of complex-valued neural networks.

In answer to our research question: 'Does the use of complex-valued neural networks improve accuracy in classification tasks by increasing the model's expressibility?' we did not observe performance improvements in classification of real-valued input data directly. When real-valued data is used as an input into a complex-valued classifier the real and imaginary parts of the weights convergence together. The complex-valued logits of the last layer are simply scaling the output. These logits are then mapped to a probability distribution over the classes. A complex-valued MLP and CNN can, therefore, at best perform on-par with its real-valued counterpart on real-valued classification tasks. However, it is more likely to perform worse, due to its initialisation and approximation error. The extended expressibility does not allow the network to learn useful information if they are not explicitly encoded or encouraged.

In existing literature we have observed that complex-valued neural networks outperform real-valued neural networks. Particularly for signal- or image-related tasks complex-valued neural networks should outperform real-valued networks. We could not find experimental evidence to further generalise the recommendation. Following our findings and the findings in existing literature, we recommend the use of complex numbers in neural networks if a) the input data has a known mapping or transformation to complex numbers, b) the noise in the input data is assumed to be distributed on the complex plane, c) complex-valued embedding or a transformation can be learned from real-valued data by explicitly encoding additional information or d) a real-valued loss function creates an explicit incentive to learn a complex-valued hypothesis. These recommendations are addressed in later chapters:

Section 4.3 and Chapter 5 present approaches that involve mapping the data into the frequency domain using Fourier and Z-transformations.

In chapter 5 the frequency representations of speech data contain noise frequencies which can be learned to be filtered out in within the representation.

Sections 4.3, 4.4 and 4.5 present methods to embed the data into the complex plane. We investigate providing positional information or implicitly learn task-specific word representations.

Instead of designing specific loss functions $J : \mathbb{C} \to \mathbb{R}$, we generalised existing loss functions (e.g. cross entropy) to the complex plane. Existing literature, however, found task-specific loss functions that emphasise complex-valued hypothesis and improve performance [23, 28, 64]. Hence, we can not ignore them in our recommendations and suggest their further investigation in future work.

We believe that new methods of learning complex-valued representation in conjunction with further work on new activation and loss functions for the complex domain will greatly improve the results of existing approaches. More investigation of the normalisation of complex-valued neural networks will stabilise the learning process further.

Our findings suggest that complex-valued models can not extract improved features from real-valued data. In the next chapter we apply alternative techniques to extract features. We achieve this by transforming and embedding real-valued data into the complex domain. We focus on language modelling and memory networks. We also found that the training process is very unstable which inspires the further investigation of constraints to the complex domain. Our findings also inspire Chapter 5 which applies the same transformation techniques on signal data. Signal data naturally has a direct translation to the complex domain. Hence, we consider the complex models to process both real and complex data.

CHAPTER 4

Complex-Valued Word Representations

In the previous chapter we worked with real-valued data as a direct input to a complex-valued neural networks and recommended to use them if there is a natural translation to a complex-valued domain.

In this chapter we focus on language representations that attempt to decide additional information in the complex plane. We show various methods that facilitate dense complex-valued representations for language processing tasks. In particular, we consider Language Models, Memory Networks and Quantum-Inspired Models. We use complex-valued neural networks to learn embeddings, or occurrence statistics to design quantum-inspired semantic representations. We facilitate recurrent neural networks and related architectures. Our approaches follow the Distributional Hypothesis [15] and attempt to capture compositionality of words [16].

Chapter 1 presents the expressibility of complex-valued representations. Its applicability to Embedding learning motivates the chapter on complex-valued word embeddings. Chapter 2 introduced recurrent networks, embeddings and transformations. We apply these building blocks to Embedding Learning and present them in this chapter. We consider general and task-specific word embeddings. This chapter is additionally motivated by our findings in Chapter 3 which indicate that embeddings and transformations are necessary to extract further features from real-valued data.

4.1 Introduction

In recent years neural language modelling has been used to train dense representations of words [65]. Embeddings are used to improve performance in more complex natural language tasks. In this work we explore complex-valued dense representations of words.

Complex-valued representations promise more interaction between the constituent words in compositions. Word embeddings are often combined (e.g. by summation) to receive sentence representations from multiple word embeddings. Complex-valued embeddings with their phase information cause an additional, structured interaction between the words. We hypothesised that complex-valued embeddings can improve performance in various natural language tasks by implicitly facilitating interference in compositions.

In this work we proposed multiple approaches to learn complex-valued representations. All methods increase their information density by using the embedding's phase or imaginary parts to encode additional information. We design quantum-inspired, positional or spectral embeddings, or use the complex weights to increase the model's degrees of freedom.

With increased degrees of freedom we can train task-specific embeddings. We demonstrate this using a question answering task (QA). For this purpose we also investigate different activation functions. We facilitate complex-valued versions of Long-Short Term Memories (LSTM), embedding layers and fully connected layers. Complex-valued neural networks, have been successfully used for a variety of tasks. However, there is a significant lack of literature reporting advantages, disadvantages and limitations of complex-valued representation learning.

4.2 Related Literature

Word embeddings are dense, low-dimensional representations of words in a specified vocabulary. They can capture semantic and syntactic relationships between words. The idea for embeddings originates in Distributional Semantics which attempt to quantify word meaning by its context words. They have been used to improve performance in down-stream natural language tasks. Pre-trained word embeddings have been used in speech recognition [65, 66], questions answering (QA) [6, 67], knowledge base completion [68] and many more applications too numerous to list. They follow the Distributional Hypothesis [15]. Historically, approaches were based on statistical features such as co-occurrence, term frequency and mutual information. Extracting these statistics from corpora resulted in sparse, high-dimensional representations which required further processing for dimensionality reduction. Earlier approaches can be categorised into generative (e.g. LDA [69]) or factorisation methods (e.g. LSA [70]). While in the past these observable features have been manually chosen, DL methods implicitly learn the features. The principle of discovering smaller sets of features for a high-dimensional input, is also applied to other data types e.g. sequences, images or signals [71, 72, 73, 74]. This research field is often

referred to as *Representation Learning*.

Representations of natural language are most commonly learned in two ways. Task-specific embeddings can be learned as a by-product of any natural language tasks, or they can be purposefully trained as general-purpose embeddings. Their reliability and quality in tasks depends on the size and type of corpus used. To overcome the frequent problem of small task-specific data sets, general semantic representations are trained on large, generic corpora in order to be retrained on the smaller task-specific corpus. Neural language modelling has often been used to create general semantic embeddings in an unsupervised setting [75]. The objective of statistical language modelling is to predict a target word given its context. Hence, n-grams are used as an input. A popular alternative is design objective functions specifically to learn embeddings. Word2Vec [76] and GloVe [77] are probably the most commonly known examples. Further investigation of earlier methods has shown that these particular embeddings are mimicking classical approaches that rely on context statistics [78, 18]. These developments have given rise to more recent approaches Context2Vec [79] and ELMo embeddings [80] which rely on bidirectional learning. Most recently, attention mechanisms are applied with recurrent networks to improve their performance on long-term dependencies [81]. By individually attending to sequences of representations, a model learns to scale the importance of elements in a weighted average. Standard recurrent cells are replaced by more complicated attention-based memory architectures to allow reliable storage and query of attended input [82, 6, 83, 81]. Attention and memory networks enhance current encoder-decoder architectures [84]. Transformer networks, a neural encoder-decoder architecture based on attention without recurrence or convolution, has been used to learn Bidirectional Encoder Representations (BERT). The learned embeddings achieve significant performance improvements in many natural language tasks [85, 86]. In this work, we use simple complex-valued language model architecture to learn general-purpose embeddings on a generic corpus and question answering embeddings on a task-specific corpus. We test the advantages and disadvantages of complex-valued embeddings in both settings.

Compositionality extends the principle of word representations to sentences and phrases. Some syntactic structure of sentences are implicitly attained in embeddings trained for sequence-oriented natural language tasks, since the statistical objective requires the model to model the word's context. As a consequence, embeddings are only meaningful in relation to each other. They are also transformed and composed within architectures to phrase representations. In sequence-based tasks, word embeddings are eventually composed into the hidden states, cell states or outputs in order to represent the entire sequence. Compositions and transformations are the main inspiration for complex-valued embeddings. The additional degrees of freedom allows an interaction between the representations based on syntactical or positional information.

Complex-valued semantic representations were first mentioned by van Rijsbergen [87]. He argued for a quantum theoretical approach towards Information Retrieval (IR) due to the geometric and probabilistic interpretation of quantum states. This basic idea was further discussed

in context in a larger body of literature Bruza et al. [88], Song et al. [89]. In the following years notable approaches have been made to quantum-inspired representations in IR. González and Caicedo [90] developed a quantum variant of existing latent topic models. Widdows and Cohen [91] have further developed the field of quantum-inspired semantic spaces. Eventually Sordoni et al. [92] applies quantum-like neural language models. While Widdows and Cohen [91] mainly argue from a geometrical interpretation, Sordoni et al. [93] and González and Caicedo [90] use the statistical perspective to argue in favour of a quantum-inspired language model. The body of literature has grown in recent years. Quantum-like language models and representation learning learn quantum states and combine them under the principle of quantum theory [94, 95, 92, 96, 97, 98, 99, 100]. Some of these approaches exploit the quantum representation and notation, but ignore the complex-valued nature of quantum states entirely. Statistical and geometrical constraints are the main benefits of a quantum theoretical models. We believe, however, this approach will only reach its full potential if more progress for complex-valued embeddings has been made. The use of complex numbers over real numbers is a fundamental difference of our work compared to previous work on NLP and Representation Learning.

In addition to quantum-inspired approaches, there have been other attempts to use complex numbers in generic representation learning. Trouillon et al. [101] and Trouillon et al. [102] developed complex embeddings for link identification in knowledge base completion. [25, 103, 26] apply the Fourier Transform to learn representations of real-valued audio signals. Scattering networks have been found useful in learning representations of images [104, 105] to achieve a compromise between location and frequency resolution. Scattering networks are based on the application of wavelet transformations of differing resolution scales. Whereas, spectral representations fix the resolution in convolutional neural networks [106].

Recurrent neural networks reduce the sequence to a single representation or a sequence of hidden states. Thus, they are popular for encoding word sequences. There have been a number of existing approaches to complex-valued recurrence in neural networks [107, 108, 109]. Existing work with complex-value recurrent neural networks mainly involves the processing and analysis of complex-valued data. Transformed input signals are processed with complex weights [110]. Additionally, the properties of complex numbers and matrices can be used to define constraints on DL models. Introduced by Arjovsky et al. [111] and further developed by Wisdom et al. [112], recurrent weights can be constrained to be unitary. Unitary computation steps ease vanishing or exploding gradient problems in the training process of deep networks and reduce the number of parameters required.

4.3 Language Modelling

The goal of language models is to predict the next word based on the previous word sequence. Formally, we maximise the average log probability of a target word w_j given its previous context

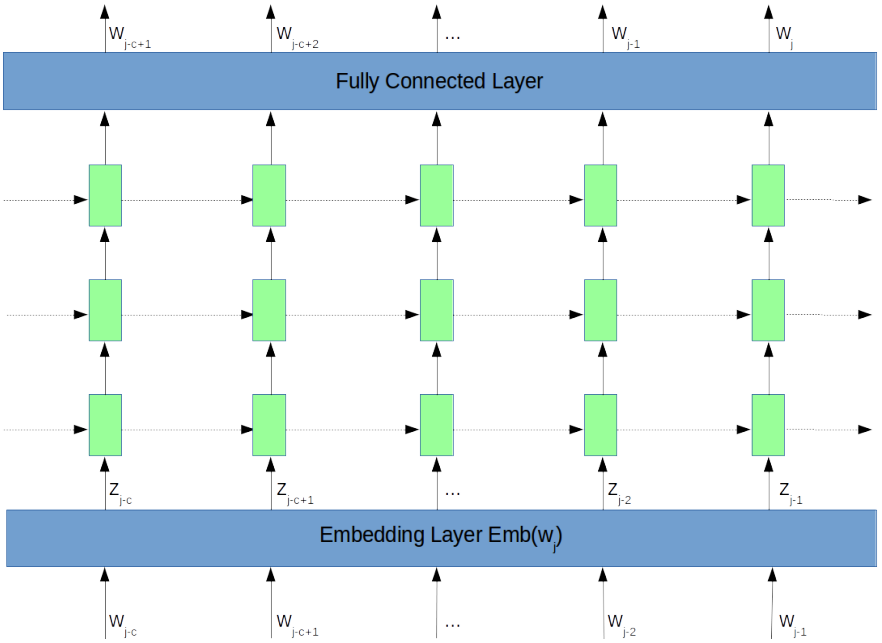

Figure 4.1: Architecture for neural language model as real and complex baselines by Zaremba et al. [5]. Green boxes represent single regularised LSTM layers.

sequence $C = w_{j-c}, \ldots, w_{j-1}$ in a corpus of length M. The context sequence C may be limited by a sentence of length m or be of constant size c across sentences:

$$\min J = -\frac{1}{M} \sum_{j=0}^{M-1} \ln p(w_j | w_{j-c}, \ldots, w_{j-1}) \tag{4.1}$$

$$p(w_j | w_{j-c}, \ldots, w_{j-1}) = \frac{p(w_{j-c}, \ldots, w_j)}{p(w_{j-c}, \ldots, w_{j-1})} \tag{4.2}$$

We compare our language models to a real-valued (LM-$\mathbb{R}$) and a complex-valued (LM-$\mathbb{C}$) baseline inspired by Zaremba et al. [5]. The authors present a number of experimental results with a simple baseline architecture. The architecture itself, including the application of dropout regularisation, have been optimised for LSTMs and this specific task. Zaremba et al. [5] demonstrated the architecture using various hyperparameter settings. Hence, the chosen setting is a very strong baseline for this task. These baseline architecture consist of an embedding layer, multiple LSTM layers and a densely connected layer for prediction of the next word (Figure 4.1).

As a by-product we learn a dense representation for each word in the vocabulary V. We learn real embeddings $z_i \in \mathbb{R}^{1\times K}$ or complex embeddings $z_i \in \mathbb{C}^{1\times K}$. Each embedding z_i is the i-th row vector from the weight matrix $E \in \mathbb{R}^{N\times K}$ or $E \in \mathbb{C}^{N\times K}$ of the embedding layer or composed from two separate vectors. They are extracted by projecting a one-hot vector of word w_i (with the i-th

element being a one) onto the embedding matrix E. We denote embedding selection of a word w_i from an embedding matrix as a function $Emb(w_i)$ (Equation 2.14).

4.3.1 Positional Slot Models

We propose two models which explicitly utilise the complex embeddings by feeding positional information into the imaginary part instead of implicitly determining the imaginary part. The complex-valued embedding is therefore a combination of two real-valued embeddings. The word is embedded separately from its position in the sentence and composed to a complex-valued embedding to be fed into the complex-valued recurrence (Figure 4.2).

Our positional language model (LM-ℂ-slots) uses an indexing function (Equation 4.3) to embed the positional information as an imaginary part of the embedding (Equation 4.4). The number of positional slots S determines into how many parts every sentence is divided. The semantic and positional embeddings are learned separately. The semantic word embeddings $Emb(w_i)$ are trained following the approach described for the real-valued case. The positional embeddings $Emb(p_s)$ are similarly trained, but the index is selected by a positional slot function $s(j, m) = 0, \ldots, S-1$ of the word's position j in a sentence of length m:

$$p_s = s(j, m) = floor(j\frac{S}{m}), \forall j \in \{0, \ldots, m\} \tag{4.3}$$

The semantic embedding of a word $Emb(w_i)$ and the positional embedding $Emb(p_s)$ are composed to the complex word embedding z_i:

$$z_i = Re(z_i) + iIm(z_i) = Emb(w_i) + iEmb(p_s) \tag{4.4}$$

4.3.2 Phase Language Model

Our phase-based model applies a phase shift as a positional embedding p_ϕ to the semantic embedding $Emb(w_i)$ (Figure 4.3). The semantic embeddings $Emb(w_i)$ and the phase ϕ are composed to a complex word embedding z_i:

$$z_i = Emb(w_i) \cdot e^{i\phi} = |Emb(w_i)| \cdot e^{i\varphi+\phi} = |Emb(w_i)| \cdot (\cos(\varphi + \phi) + i \sin(\varphi + \phi)) \tag{4.5}$$

Every word receives a phase offset $\phi(j, m)$ based on its position j in a sentence of length m:

$$\phi(j, m) = (j\frac{2\pi}{m}), \forall j \in \{0, \ldots, m\} \tag{4.6}$$

While we test a single instance of a phase-shift model, this defines a family of models. Equation 4.6 divides the unit circle into m equally sized parts, thus the phases are distributed over

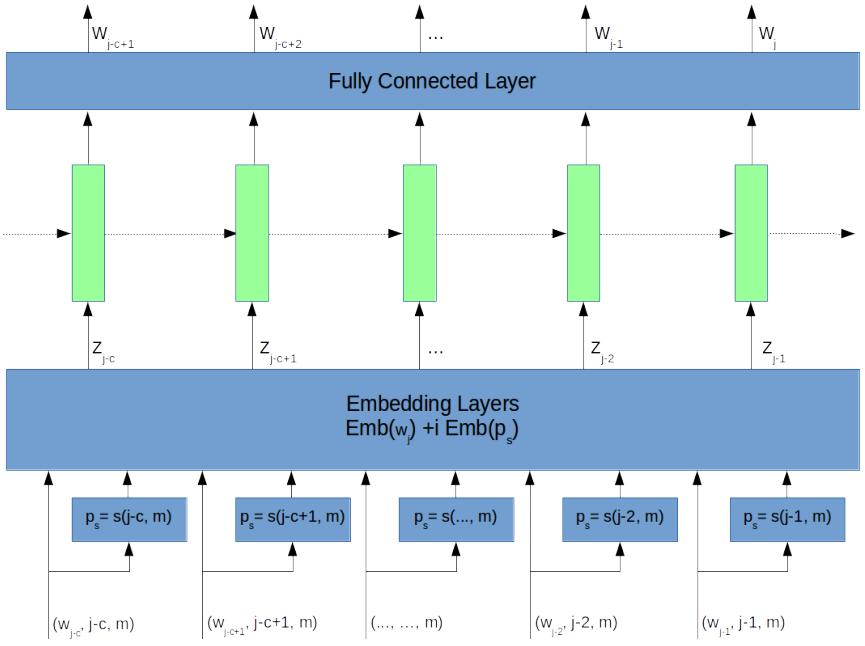

Figure 4.2: Architecture for a neural language model using complex-valued slot positional embeddings. Green boxes summarise multiple regularised LSTM layers.

the unit circle. Each word embedding in a sentence is then assigned a phase according to its position in the sentence. It starts with a start-of-sentence token at 0, $\frac{2\pi}{m}, 2\frac{2\pi}{m}, \ldots$. Alternatively, a phase shift vector ϕ_i can be learned independently for each word w_i. Phase learning can also be applied to existing embeddings (e.g. word2vec) by initialising the weights with an existing embedding matrix. Phase learning requires further and more in-depth consideration, which may be considered in future research.

4.3.3 Frequency Language Models

We propose another type of language model: the Frequency Language Model. Frequency Language Models apply transformations to the frequency domain onto a sequence of real-valued embeddings. In our first frequency-based model we apply a Discrete Fourier transformation (DFT) $\mathcal{F}(x[K])$ (LM-ℂ-fourier, Figure 4.4). The DFT decomposes each embedding into a sequence of cosine and sine waves of the same length in the frequency domain (Equation 4.7). In our case the DFT is applied on the sequence of embeddings $Emb(w_0), \ldots, Emb(w_m)$. This results in a complex-valued frequency representation of the same size as input to the recurrent layers.

$$\begin{bmatrix} z_0 & \ldots & z_m \end{bmatrix} = \mathcal{F}(Emb(w_j)), \forall w_j \in C \tag{4.7}$$

Our last model is a generalisation of our complex models that we have previously presented

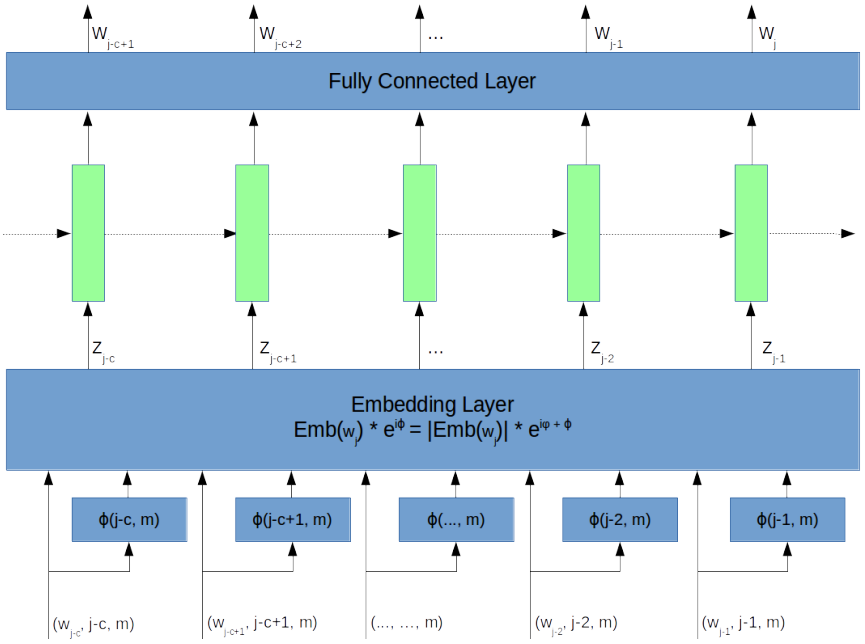

Figure 4.3: Architecture for neural language model using phase shifts on real-valued embeddings. Green boxes summarise multiple regularised LSTM layers.

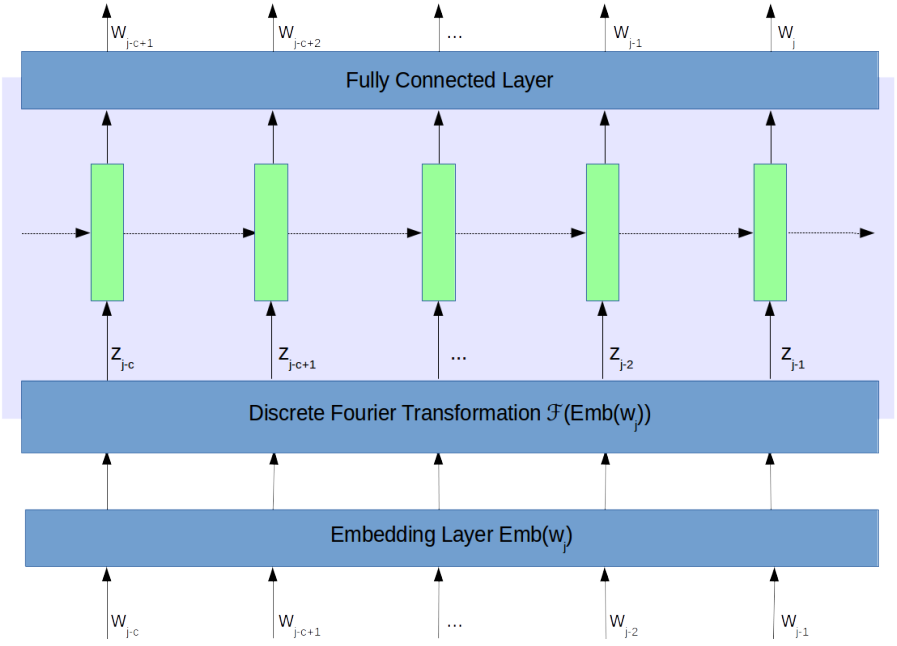

Figure 4.4: Architecture for neural language model in the Fourier Domain. Green boxes summarise multiple regularised LSTM layers and light blue area indicates states, inputs and outputs in the Fourier domain.

(LM-ℂ-transform). Instead of applying the DFT that decomposes a signal into cosine and sine functions in the frequency domain, we apply the unilateral Z-Transform $\mathcal{Z}_a(x[K])$ (Equation 4.8). The ZT is a generalisation of the DFT where $|a| = 1$. In this model we learn vector complex variables a to transform embeddings into the Z-domain (Figure 4.5). Overall, we learn a set of possible transformations and the recurrence works within the frequency domain.

$$
\begin{aligned}
&\begin{bmatrix} z_0 & \dots & z_m \end{bmatrix} \\
=&\begin{bmatrix} \mathcal{Z}_a(Emb(w_0)) & \dots & \mathcal{Z}_a(Emb(w_m)) \end{bmatrix} \\
=&\begin{bmatrix}
Emb(w_0)[0]a^{-0} & Emb(w_0)[1]a^{-1} & \dots & Emb(w_0)[K]a^{-K} \\
Emb(w_1)[0]a^{-0} & Emb(w_1)[1]a^{-1} & \dots & Emb(w_1)[K]a^{-K} \\
\vdots & & \ddots & \\
Emb(w_m)[0]a^{-0} & Emb(w_m)[1]a^{-1} & \dots & Emb(w_m)[K]a^{-K}
\end{bmatrix}
\end{aligned}
\tag{4.8}
$$

We extract the ZT as series in a complex vector with $a = Re(a) + iIm(a) = |a|(\cos(\phi) + i\sin(\phi))$:

$$
\mathcal{Z}(x[K]) = \begin{bmatrix} x[0]a^{-0} & \dots & x[K]a^{-K} \end{bmatrix} \tag{4.9}
$$

4.3.4 Experiments

Table 4.1: Test log perplexity of language models. Trained for 100 epochs with random uniform, but variance-scaled initialisations (Glorot uniform distribution). Each model has $l = 2$ recurrent layers each with $m_{\mathbb{C}} = 325$ resp. $m_{\mathbb{R}} = 650$ units after embeddings of the same size. We apply $\phi_s = relu$ activation function and $\phi_g = tanh_h$ in each recurrent layer and add a $softmax$ classifier.

Model	Parameters	PTB
LM-ℝ	19,765,200	6.5095
LM-ℂ	16,385,200	6.7012
LM-ℂ-slot	25,790,100	7.0166
LM-ℂ-phase	22,081,100	7.1389
LM-ℂ-fourier	33,030,400	6.2781
LM-ℂ-transform	33,030,410	6.2290

We experiment with our language models on the Penn-Tree-Bank Corpus (PTB) [113]. The PTB corpus contains 887,521 tokens in the training set, 70,390 tokens in the validation set and 78,669 token in the test set. We choose our baseline hyperparameters similar to Zaremba et al. [5] medium-sized regularised experiments. We attempted to use the standard hyperbolic tangents

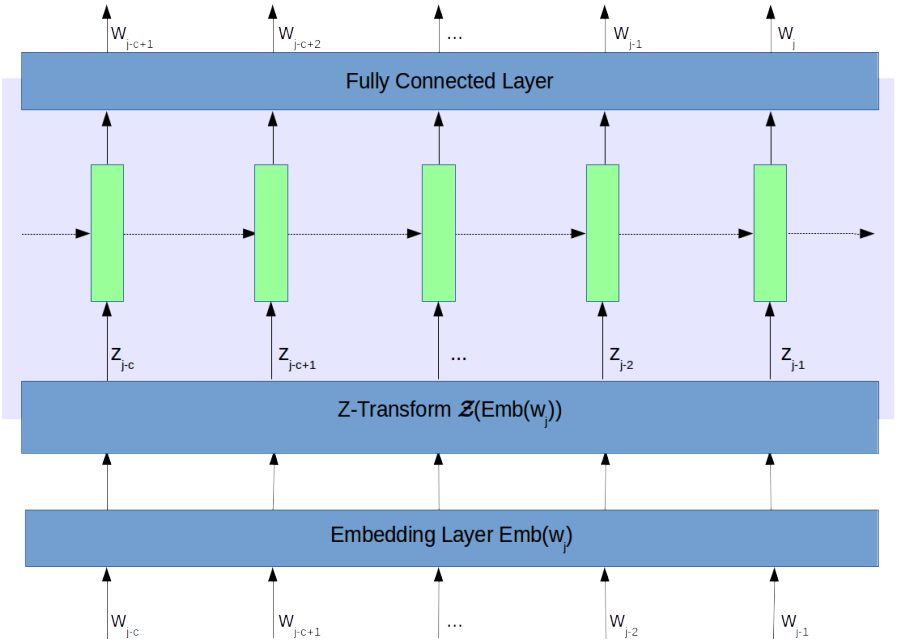

Figure 4.5: Architecture for neural language model in the Frequency Domain using a trained Z-transformation. Green boxes summarise multiple regularised LSTM layers and light blue areas indicate states, inputs and outputs in the frequency domain.

$\phi_g = tanh$ in our complex-valued models, but we had to replace it with Hirose [35]'s hyperbolic tangent tan_h to improve the training stability. For our complex versions we used $\sigma(|z|^2)$ as recurrent activations and $\sigma(|z|^2)$ as the activation function for the output layer.

Our results are shown in Table 4.1. We observe that the complex-valued baseline does not outperform the real-valued baseline. The performance is significantly worse than the baseline. This supports our findings in Chapter 3. The tested methods to explicitly encode further information into the complex representation show slightly different results. Explicit encoding of positional information perform even worse than the real baselines, but frequency-based representation outperform our real-valued baseline.

We also observe that the numerical stability is problematic in larger language models. The positional slot model and the phase model often fail in training and need to be restarted from a checkpoint. Gradient norm clipping improves this behaviour, but does not solve it. This may still be due to the choice of activation functions. These observations have also been made in Chapter 3.

We found that our results are far worse than more recently reported results, since we do not use any regularisation, stateful training [5]. While their baselines also underwent extensive regularisation, hyperparameter tuning and other methods to improve the results, we intended to compare real- and complex-valued language modelling without these methods. Regularisation

(e.g. dropout) and hyperparameter tuning (e.g. initialisation) in the complex domain require new methods and other considerations. Our models are also not as deep nor wide as state-of-the-art approaches.

4.4 Memory Networks

Memory networks were introduced by Weston et al. [17] and further developed by Sukhbaatar et al. [6] and Kumar et al. [114]. They have been widely successful for Question Answering (QA) and Question Classification tasks [115]. Existing approaches use complex-valued recurrent networks as image memories [116] or density matrices as joint sentence representations of questions and answers in an end-to-end setting [117]. In this section we consider complex-valued memory networks in comparison to real-valued memory networks. We modify the end-to-end memory network architecture [6] to achieve a better comparison of the real-valued and complex-valued model. Instead of forcing certain information into the complex-valued embeddings, the chosen complex architecture allows embeddings to be learned freely.

The model structure is shown in Figure 4.6. Suppose we want to store an input sequence $x = x_1, x_2, \dots x_m$ and retrieve an answer $\tilde{y}$ to a query $q = q_1, q_2, \dots, q_o$. The input and query are embedded to $Emb_A(x_i) \in \mathbb{R}^K$ or $Emb_A(x_i) \in \mathbb{C}^K$ and $Emb_Q(q_j) \in \mathbb{R}^K$ or $Emb_Q(q_j) \in \mathbb{C}^K$. They are dense representations, but originated in different semantic spaces of the same dimensionality. We compute a probability distribution p over a matching score between the embedded input sequence and question using dot-product or inner product:

$$p = \sigma(Emb_A(x) \cdot Emb_Q(q)) \tag{4.10}$$

In the complex case we compute the distribution as:

$$p = \sigma(|\langle Emb_A(x)|Emb_Q(q)\rangle|^2) \tag{4.11}$$

We compute another *memory output* embedding $Emb_B(x_i) \in \mathbb{R}^Q$ or $Emb_B(x_i) \in \mathbb{C}^Q$ with Q being the maximum query length. For the memory response vector o the memory output embedding $Emb_B(x_i)$ is weighted by the probability distribution p and summed with the memory output embedding $Emb_B(x_i)$:

$$o = \sum_{i=1}^{m} p_i Emb_B(x_i) \tag{4.12}$$

The memory output vector o is concatenated with the query embedding $Emb_Q(q)$ and passed to an LSTM. Eventually, the last state h_c is classified with a softmax classifier:

$$\tilde{y} = \sigma(h_c W + b) \tag{4.13}$$

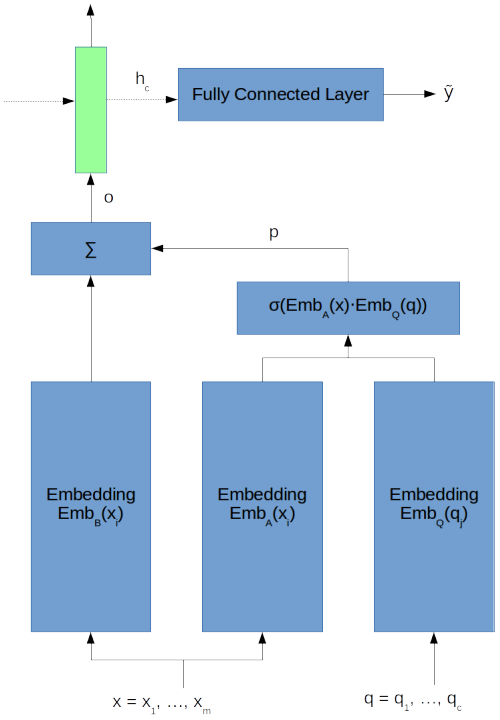

Figure 4.6: Simplified end-to-end memory network architecture for comparison inspired by Sukhbaatar et al. [6].

The complex-valued version again relies on the squared amplitude:

$$\tilde{y} = \sigma(|h_c W + b|^2) \tag{4.14}$$

4.4.1 Experiments

We investigated this architecture in all 20 bAbI question answering tasks [118]. The data set consists of a large number of tasks for each subtask. Thus, we are able to train task-specific embedding using the bAbI data set. We compared the architecture as complex-valued network with a real-valued version. We replaced real-valued weight matrices with complex-valued matrices. For the embeddings, however, the complex-valued embeddings have half the length $\frac{1}{2}K$ compared to real-valued embeddings K. For our complex version we used $\sigma(|z|^2)$ as recurrent activations and $\sigma(|z|^2)$ as the activation function for the classifier. Our real-valued version uses *relu* and *sigmoid* activation functions.

In a best effort comparison of complex and real-valued Memory Networks we applied a random search over the hyperparameter space. Specifically, we considered the activation functions,

initialisations and initialisation scales, learning rate and rate decay, dropout probability for embeddings and recurrence. We chose the largest possible model with $m_{\mathbb{R}} = 512$ and $m_{\mathbb{C}} = 256$ recurrent units and embedding size of 256 in the real case or respectively 128 in the complex case. We used 150 iterations of random search over the hyperparameter search space. Each time we trained the models until convergence, but stopped after a maximum of 200 training epochs. Table 4.2 displays the best effort results.

Table 4.2: Test accuracy of best effort memory network in complex and real version on the bAbI tasks. Selected best of 150 iterations of random search over the hyperparameter search space. Each iteration was trained for 200 epochs. The model sizes are chosen independently for real and complex models and only are limited by the hardware available.

bAbI task	$\mathbb{R}$-valued network	$\mathbb{C}$-valued network
Q1: Single Supporting Fact	**0.9960**	0.9910
Q2: Two Supporting Facts	**0.4190**	0.4130
Q3: Three Supporting Facts	0.3560	0.3560
Q4: Two Arg. Relations	1.0000	1.0000
Q5: Three Arg. Relations	**0.9050**	0.8860
Q6: Yes/No Question	**0.9870**	0.9700
Q7: Counting	**0.9060**	0.8920
Q8: Lists/Sets	**0.8600**	0.8510
Q9: Simple Negation	**0.9880**	0.9610
Q10: Indefinite Knowledge	**0.9870**	0.9530
Q11: Basic Co-Reference	0.9960	0.9970
Q12: Conjunction	1.0000	0.9990
Q13: Compound Co-Reference	1.0000	0.9990
Q14: Time Reasoning	0.5100	**0.5160**

Continued on next page

Table 4.2 - Continued from previous page

bAbI task	$\mathbb{R}$-valued network	$\mathbb{C}$-valued network
Q15: Basic Deduction	0.6080	**0.6220**
Q16: Basic Induction	0.5280	0.5280
Q17: Positional Reasoning	**0.9970**	0.9780
Q18: Size Reasoning	**0.9400**	**0.9410**
Q19: Path Finding	**0.4080**	0.3970
Q20: Agent's Motivations	0.9890	0.9890

We found that the majority of tasks are solvable with our memory network. This finding supports the choice of our memory network architecture. We further investigated different activation functions ϕ and smaller sized models. We used the weight initialisation discussed by Trabelsi et al. [27]. We initialised each setting 10 times and trained it for 200 epochs with an Adam optimisation. Tables 4.4 show our results. Due to the nature of this architecture it was difficult to create a complex- and real-valued version with exactly the same number of parameters. We compare the number of real-valued parameters in Table 4.3. Again, the embeddings were not pre-trained on a larger corpus, but trained only using the bAbI data.

Table 4.3: Number of real-valued parameters of the memory network versions in respect to the bAbI tasks 1 to 20 (Experiment 2). Each model has either $m_{\mathbb{R}}$ real or complex $m_{\mathbb{C}}$ neurons in the LSTM layer. In general, the capacity of the different sized models can be ordered: $m_{\mathbb{C}} = 8 \leq m_{\mathbb{R}} = 16 \leq m_{\mathbb{C}} = 16 \leq m_{\mathbb{R}} = 16 \leq m_{\mathbb{C}} = 21 \leq m_{\mathbb{C}} = 24 \leq m_{\mathbb{C}} = 64$. Exceptions to this rule are marked with *. The largest real-valued model is always significantly larger than the largest complex-valued model.

	$\mathbb{R}$-valued units			$\mathbb{C}$-valued units			
bAbI task	$m_{\mathbb{R}} = 16$	$m_{\mathbb{R}} = 32$	$m_{\mathbb{R}} = 64$	$m_{\mathbb{C}} = 8$	$m_{\mathbb{C}} = 16$	$m_{\mathbb{C}} = 21$	$m_{\mathbb{C}} = 24$
Q1	12,814	24,750	54,766	10,364	18,716	24,456*	28,092
Q2	45,912	89,048	181,464	43,568	83,120	108,360	123,696
Q3	130,845	258,669	520,461	128,618	252,858	331,028	378,122
Q4	8,944	17,488	40,720	6,528	11,488	15,108*	17,752
Q5	61,658	119,610	241,658	59,476	113,844	148,344	169,236
Q6	20,906	38,730	80,522	18,612	32,852	42,272	48,116
Q7	32,720	61,232	124,400	30,528	55,456	71,556	81,408
Q8	35,596	66,540	134,572	33,336	60,696	78,316	89,080
Q9	13,997	26,397	57,341	11,626	20,442	26,472	30,282
Q10	15,108	28,484	61,380	12,744	22,536	29,176	33,352
Q11	14,369	26,929	58,193	11,954	20,930	27,060	30,930
Q12	14,179	27,347	59,827	11,734	21,318	27,828	31,926
Q13	15,009	28,209	60,753	12,594	22,210	28,740	32,850
Q14	17,168	32,304	68,720	14,848	26,400	34,140	38,976
Q15	12,228	22,724	49,860	9,864	16,776	21,616*	24,712
Q16	10,808	20,984	47,480	8,368	14,960	19,600*	22,576
Q17	10,331	19,467	43,883	8,070	13,622	17,612*	20,198
Q18	19,565	37,981	80,957	17,258	32,090	41,880	47,946
Q19	13,204	23,556	50,404	11,064	17,832	22,582*	25,624
Q20	16,026	28,346	59,130	13,844	22,580	28,560	32,340

Table 4.4: Test accuracy of the different versions of the memory network in complex and real version on the bAbI tasks (Experiment 2). Each version has $m_{\mathbb{R}}$ resp. $m_{\mathbb{C}}$ neurons in the LSTM layer. Selected best of 10 runs. Each run was trained for 200 epochs.

		$\mathbb{R}$			$\mathbb{C}$			
bAbI task	Activation function ϕ	$m_{\mathbb{R}} = 16$	$m_{\mathbb{R}} = 32$	$m_{\mathbb{R}} = 64$	$m_{\mathbb{C}} = 8$	$m_{\mathbb{C}} = 16$	$m_{\mathbb{C}} = 21$	$m_{\mathbb{C}} = 24$
Q1: Single Supporting Fact	*identity*	0.5210	0.5860	0.8020	0.5350	**0.5440**	0.5410	0.5420
	tanh	0.5110	0.5410	0.5830	0.5280	0.5380	0.5360	0.5390
	relu	0.5290	0.7650	0.8910	0.5360	0.5350	0.5380	0.5360
	$\|z\|^2$	0.5920	0.7630	0.8620	**0.5670**	0.5280	0.5390	**0.5930**
	$\|z\|$	**0.7120**	**0.8480**	**0.9090**	0.5160	0.5300	**0.5900**	0.5310
Q2: Two Supporting Facts	*identity*	0.3940	0.3970	0.4080	0.4060	0.4030	0.4120	0.4110
	tanh	0.3030	0.3700	0.4030	0.3250	0.3800	0.3870	0.3780
	relu	0.3910	**0.4110**	0.4090	**0.3800**	**0.3870**	**0.4060**	**0.4090**
	$\|z\|^2$	0.4020	0.4070	**0.4170**	0.3360	0.3800	0.3660	0.4000
	$\|z\|$	**0.4170**	0.4090	0.4130	0.3640	0.3760	0.3900	0.3830
Q3: Three Supporting Facts	*identity*	0.3190	0.3050	0.2760	**0.3480**	**0.3560**	**0.3460**	**0.3450**
	tanh	0.2870	0.2950	0.2690	0.2590	0.2990	0.2970	0.3060
	relu	**0.3330**	0.2960	0.2880	0.3060	0.3220	0.3420	0.3320
	$\|z\|^2$	0.3490	**0.3380**	**0.3540**	0.3020	0.2540	0.2610	0.2510
	$\|z\|$	0.2960	0.2900	0.2840	0.3020	0.3090	0.3330	0.3080
Q4: Two Arg. Relations	*identity*	**0.7940**	0.7970	0.7950	**0.8030**	**0.7990**	0.8000	0.7980
	tanh	0.7670	0.7950	**0.8020**	0.7910	0.7970	0.7980	0.8020
	relu	0.7830	**0.8010**	0.7980	0.7870	**0.7990**	**0.8090**	**0.8040**
	$\|z\|^2$	0.7730	0.7970	0.7930	0.7290	0.7910	0.7940	0.7910
	$\|z\|$	0.7820	0.7980	0.7990	0.7650	0.7950	0.7920	**0.8040**
Q5: Three Arg. Relations	*identity*	**0.8760**	**0.9050**	0.9030	**0.8700**	0.8670	0.8690	0.8640
	tanh	0.7870	0.8280	0.8800	0.5130	0.6800	0.5890	0.6450
	relu	0.8590	0.8710	0.8990	0.8050	**0.8820**	**0.8780**	**0.8730**
	$\|z\|^2$	0.8540	0.8520	0.8970	0.6050	0.6600	0.6250	0.6590
	$\|z\|$	0.8690	0.8990	**0.9060**	0.7850	0.8170	0.8490	0.8540
Q6: Yes/No Questions	*identity*	**0.8500**	0.8490	0.8550	0.8480	0.8520	0.8480	0.8510
	tanh	0.8080	0.8400	**0.8820**	0.8420	0.8470	0.8520	0.8500
	relu	0.8470	**0.8530**	0.8510	**0.8490**	**0.8540**	0.8500	0.8530
	$\|z\|^2$	0.8480	0.8520	0.8530	0.8480	0.8470	0.8510	0.8530
	$\|z\|$	0.8470	0.8490	0.8540	0.8470	0.8520	**0.8580**	**0.8590**
Q7:Counting	*identity*	**0.8480**	0.8510	0.8720	0.8270	0.8290	**0.8590**	**0.8870**
	tanh	0.8250	0.8420	0.8650	0.8040	0.8050	0.8050	0.8110
	relu	0.8220	0.8660	0.8400	**0.8410**	0.8130	0.8570	0.8370
	$\|z\|^2$	0.8140	0.8200	0.8200	0.8000	0.8030	0.8040	0.8110
	$\|z\|$	0.8390	**0.8720**	**0.8730**	0.8270	**0.8500**	0.8470	0.8330
Q8: Lists/Sets	*identity*	0.7740	0.7810	0.7740	**0.7700**	0.7700	0.7710	0.7730
	tanh	0.7520	0.7650	0.7780	0.7360	0.7630	0.7620	0.7550
	relu	0.7700	0.7770	0.7760	0.7660	0.7700	**0.7750**	**0.7770**
	$\|z\|^2$	0.7760	0.7740	0.7820	0.7660	0.7540	0.7650	0.7660
	$\|z\|$	**0.7770**	**0.7880**	**0.7840**	0.7690	**0.7740**	**0.7750**	0.7740

Continued on next page

Table 4.4 - Continued from previous page

		$\mathbb{R}$			$\mathbb{C}$			
bAbI task	Activation function ϕ	$m_{\mathbb{R}} = 16$	$m_{\mathbb{R}} = 32$	$m_{\mathbb{R}} = 64$	$m_{\mathbb{C}} = 8$	$m_{\mathbb{C}} = 16$	$m_{\mathbb{C}} = 21$	$m_{\mathbb{C}} = 24$
Q9: Simple Negation	*identity*	0.8410	0.8760	0.9050	**0.8580**	**0.8750**	**0.9130**	0.8890
	tanh	0.6420	0.8210	0.8460	0.7880	0.7280	0.7590	0.8070
	relu	**0.8850**	**0.9000**	**0.9430**	0.8050	0.8410	0.8960	**0.9150**
	$\|z\|^2$	0.8740	0.8520	0.9130	0.8300	0.8260	0.8360	0.8390
	$\|z\|$	0.8300	0.8950	0.9420	0.8110	0.8600	0.9050	0.8650
Q10: Indefinite Knowledge	*identity*	**0.8830**	0.9050	0.9240	**0.8560**	**0.8930**	**0.9040**	**0.9230**
	tanh	0.4930	0.7100	0.9010	0.7360	0.5120	0.7430	0.7490
	relu	0.8130	0.8640	0.7930	0.8430	0.8770	0.9030	0.8780
	$\|z\|^2$	0.8220	0.9060	0.8700	0.7490	0.7210	0.7420	0.7530
	$\|z\|$	0.8190	**0.9310**	**0.9300**	0.7490	0.8720	0.8800	0.8940
Q11: Basic Co-Reference	*identity*	0.6980	0.6990	0.7060	0.6920	0.6920	0.7010	0.7040
	tanh	0.6600	0.7000	0.7110	**0.6990**	**0.7090**	0.7030	0.7060
	relu	**0.7310**	**0.9390**	**0.9610**	0.6960	0.6970	**0.7090**	0.7070
	$\|z\|^2$	0.7040	0.7020	0.7470	0.6890	0.6830	0.6970	**0.9030**
	$\|z\|$	0.6960	0.7070	0.7100	0.6970	0.6860	0.6910	0.7070
Q12: Conjunction	*identity*	**0.7160**	0.7120	0.7140	**0.7140**	0.7130	0.7130	0.7130
	tanh	0.7120	0.7120	0.7120	0.7130	0.7130	0.7130	0.7140
	relu	0.7120	**0.7140**	**0.7150**	0.7120	**0.7130**	**0.7150**	**0.7170**
	$\|z\|^2$	0.7110	0.7120	0.7140	0.7020	0.7090	0.7140	0.7140
	$\|z\|$	0.7130	0.7130	0.7140	0.7110	0.7120	0.7130	0.7160
Q13: Compound Co-Reference	*identity*	0.9320	0.9320	0.9330	0.9330	0.9330	0.9320	0.9330
	tanh	0.9330	0.9320	0.9330	0.9330	0.9320	0.9320	0.9320
	relu	0.9330	**0.9370**	**0.9340**	0.9330	0.9330	0.9330	**0.9340**
	$\|z\|^2$	0.9320	0.9330	0.9330	0.9320	0.9320	0.9320	0.9330
	$\|z\|$	0.9330	0.9330	0.9340	0.9320	0.9320	0.9320	0.9330
Q14: Time Reasoning	*identity*	0.4730	0.4710	**0.5090**	**0.4950**	0.4950	**0.4980**	**0.4990**
	tanh	0.4140	0.4620	0.4820	0.4150	0.4730	0.4600	0.4810
	relu	**0.4830**	0.4600	0.4680	0.4820	**0.4960**	0.4970	0.4860
	$\|z\|^2$	0.4640	0.4820	0.4870	0.4080	0.4510	0.4570	0.4650
	$\|z\|$	0.4780	**0.4900**	0.4800	0.3980	0.4670	0.4560	0.4550
Q15: Basic Deduction	*identity*	0.5960	0.5930	0.5950	0.5980	0.5990	0.6000	0.5990
	tanh	0.5780	0.5810	0.5930	**0.6000**	0.6030	0.5970	**0.6020**
	relu	0.5930	**0.5940**	0.5980	0.5960	**0.6080**	**0.6060**	0.5970
	$\|z\|^2$	**0.6000**	**0.5940**	**0.6010**	0.5730	0.5990	0.5950	0.5920
	$\|z\|$	0.5980	0.5910	0.6000	0.5980	0.6010	0.5960	0.5950
Q16: Basic Induction	*identity*	**0.5220**	0.5210	0.5200	0.5160	**0.5210**	0.5180	0.5150
	tanh	0.5070	0.5110	0.5190	**0.5200**	0.5190	0.5180	0.5150
	relu	0.5150	0.5160	0.5200	0.5130	0.5160	0.5160	**0.5250**
	$\|z\|^2$	0.5140	0.5140	0.5150	0.4810	0.5100	0.5150	0.5120
	$\|z\|$	0.5140	**0.5200**	**0.5250**	0.5060	0.5170	**0.5220**	0.5240
Q17: Positional Reasoning	*identity*	0.6330	**0.6390**	**0.6380**	**0.6490**	0.6410	0.6310	0.6390
	tanh	0.6270	0.6340	0.6290	0.6450	0.6360	0.6390	0.6390
	relu	0.6360	0.6360	0.6340	0.6350	**0.6530**	**0.6560**	**0.6530**
	$\|z\|^2$	0.6360	0.6340	0.6320	0.6360	0.6320	0.6430	0.6350

Continued on next page

Table 4.4 - Continued from previous page

		$\mathbb{R}$			$\mathbb{C}$			
bAbI task	Activation function ϕ	$m_{\mathbb{R}} = 16$	$m_{\mathbb{R}} = 32$	$m_{\mathbb{R}} = 64$	$m_{\mathbb{C}} = 8$	$m_{\mathbb{C}} = 16$	$m_{\mathbb{C}} = 21$	$m_{\mathbb{C}} = 24$
	$\|z\|$	**0.6380**	0.6350	**0.6380**	0.6310	0.6340	0.6320	0.6350
Q18: Size Reasoning	*identity*	0.9290	0.9290	**0.9390**	0.9360	**0.9310**	**0.9290**	**0.9330**
	tanh	0.9270	0.9300	0.9280	0.9110	0.9250	0.9200	0.9230
	relu	**0.9360**	0.9300	0.9350	0.9340	0.9290	0.9280	0.9300
	$\|z\|^2$	0.9300	0.9250	0.9240	**0.9370**	0.9280	0.9260	0.9260
	$\|z\|$	0.9290	**0.9310**	0.9290	0.9270	0.9280	**0.9290**	0.9300
Q19: Path Finding	*identity*	0.1260	0.1260	0.1250	**0.1370**	0.1370	0.1350	**0.1390**
	tanh	0.1160	0.1170	0.1260	0.1310	0.1290	0.1280	0.1360
	relu	0.1210	0.1180	**0.1830**	0.1360	0.1300	**0.1380**	0.1310
	$\|z\|^2$	**0.1370**	**0.1270**	0.1170	0.1320	**0.1390**	0.1280	0.1310
	$\|z\|$	0.1250	0.1230	0.1290	0.1310	0.1250	0.1320	0.1290
Q20: Agent's Motivations	*identity*	0.9870	**0.9890**	0.9880	**0.9870**	**0.9890**	**0.9890**	**0.9880**
	tanh	0.9870	0.9880	0.9880	0.9580	0.9630	0.9650	0.9700
	relu	0.9880	0.9870	0.9880	0.9860	0.9870	0.9860	**0.9880**
	$\|z\|^2$	**0.9890**	0.9880	0.9880	0.9580	0.9880	0.9870	0.9830
	$\|z\|$	0.9880	**0.9890**	**0.9890**	0.9860	0.9880	0.9860	0.9860

A memory network type is considered to solve a sub task if its test accuracy is ≥ 0.95 or higher in the best effort experiment. A network type (real-valued or complex-valued) is considered to outperform the other if a) at least one network of its type shows the same or better performance than a larger network of the other type; or b) if every network of a type shows better performance than the similarly sized network(s) of the other type. With these conditions preference was given to smaller networks over larger networks with equal performance. We compared the maximum performance achieved amongst all activation functions. We considered each network size separately. Our findings are:

Q1 - Single Supporting Fact: Both model types solve the sub task very well. Real networks significantly outperform complex networks, because they require significantly fewer parameters to achieve significantly higher accuracy.

Q2 - Two Supporting Facts: Real and complex do not solve the sub task. Real networks outperform complex networks, because they require fewer parameter to achieve better accuracy.

Q3 - Three Supporting Facts: Real and complex networks do not solve the sub task. Complex networks outperform real networks, because they require fewer parameters to achieve a similar accuracy as real networks.

Q4 - Two Arg. Relations: Real and complex networks solve the task very well. Complex networks outperform real networks, because they require fewer parameters and achieve slightly higher accuracy.

Q5 - Three Arg. Relations: Real and complex networks do not solve the sub task. Real networks outperform complex networks, because they require fewer parameters for better accuracy.

Q6 - Yes/No Question: Real and complex networks solve the sub task. Complex networks outperform real networks, because they require fewer parameters. Only significantly larger real networks outperform complex networks.

Q7 - Counting: Real and complex networks do not solve the sub task. Complex networks outperform real networks, because they require fewer parameters to achieve a improved accuracy.

Q8 - Lists/Sets: Real and complex networks do not solve the sub task. Real networks outperform complex networks, because they require fewer parameters to achieve a improved accuracy.

Q9 - Simple Negation: Real and complex networks solve the sub task. Complex networks outperform real networks, because similarly sized complex networks achieve higher accuracy ($m_{\mathbb{R}} = 32, m_{\mathbb{C}} = 21$). Only significantly larger real networks outperform complex networks.

Q10 - Indefinite Knowledge: Real and complex networks solve the sub task. Real networks outperform complex networks, because they require fewer parameter to achieve higher accuracy.

Q11 - Basic Co-Reference: Real and complex networks solve the sub task very well. Real networks significantly outperform complex networks, because they require fewer parameters to achieve significantly higher accuracy.

Q12 - Conjunction: Real and complex networks solve the sub task very well. Complex networks outperform real networks, because they require fewer parameters to achieve same accuracy and higher accuracy with more parameters.

Q13 - Compound Co-Reference: Real and complex networks solve the sub task very well. Real networks outperform complex networks, because they require fewer parameters to achieve higher accuracy.

Q14 - Time Reasoning: Real and complex networks do not solve the sub task. Complex networks outperform real networks, because they require fewer parameters to achieve higher accuracy.

Q15 - Basic Deduction: Real and complex networks do not solve the sub task. Complex networks outperform real networks, because they require fewer parameters to achieve higher accuracy.

Q16 - Basic Induction: Real and complex networks do not solve the sub task. Complex networks outperform real networks, because they require fewer parameters to achieve the same accuracy.

Q17 - Positional Reasoning: Complex and real solve the sub task very well. Complex networks outperform real networks, because they require fewer parameter to achieve higher accuracy.

Q18 - Size Reasoning: Complex and real networks do not solve the sub task. Complex networks outperform real networks, because they require fewer parameters to achieve slightly higher accuracy.

Q19 - Path Finding: Complex and real networks do not solve the sub task. Complex networks outperform real networks, because they require fewer parameters for slightly improved accuracy. Significantly larger real networks outperform complex networks.

Q20 - Agent's Motivations: Complex and real networks solve the sub task. Complex and real networks perform similarly with a similar number of parameters.

Of the 20 considered bAbI tasks the architecture in both in real and complex version solves the same 10 sub tasks. We found that in 7 sub tasks real-valued networks outperform the complex-valued networks (Q1, Q2, Q5, Q8, Q10, Q11, Q13). In one task both types perform similarly well (Q20). In 12 tasks the complex-valued network outperform the real-valued networks (Q3, Q4, Q6, Q7, Q9, Q12, Q14, Q15, Q16, Q17, Q18, Q19). In most cases complex memory networks use significantly fewer parameters. We also acknowledge that the performance improvements from the real to complex-valued memory networks are often small and inconsistent. This is the case in the sub tasks Q3, Q4, Q12, Q13, Q16. We found that our complex-valued memory network outperforms the real-valued memory network mainly in logic and reasoning tasks.

4.5 Quantum-Inspired Models of Language

Quantum theory is defined as the mathematical framework to describe physical phenomena. Quantum theory offers a geometrical and probabilistic interpretation of complex-valued vectors. These notions have always been important in Computational Semantics. Semantic spaces have been spanned by facilitating the output of probabilistic generative models or factorisation methods. With this historical development of Computational Semantics, Quantum Theory motivates new types of semantic spaces. There have been a number of different approaches in the past to establish semantic quantum states in Computational Semantics. In recent literature we observe encouraging results in quantum-inspired language modelling [92, 98].

Quantum-inspired semantic spaces commonly satisfy a number of properties. They are usually complex Hilbert spaces in which vectors satisfy a probability amplitude constraint $\sum |z_j|^2 = \sum_{j=0}^{k} p(z_j|\psi) = 1$ and in which the bases $\langle \beta_i | \beta_j \rangle = 0, \forall \beta_j, \beta_i \in \mathcal{B}$ are orthogonal to each other. Classical semantic spaces often satisfy similar properties. However, instead of the probability amplitude, they apply another probabilistic constraint $\sum_{j=0}^{k} z_j = 1$. This is mostly the case if they have been generated by probabilistic models. Their bases also satisfy the orthogonal constraint depending. The fundamental difference between quantum-inspired spaces and classical spaces is the use of complex representations, the type of probabilistic constraint and the operations used.

In this section we discuss the postulates, notation and properties of Quantum Theory and their relation to Computational Semantics. These postulates are the fundamental axioms that serve as premises within this area. We derive a quantum-inspired framework of semantic spaces. To transfer the concept of interference we chose the basis for our quantum states to be learnable or fixed statistical features. Interference of semantic quantum states is equivalent to composition and an implicit disambiguation. The process of measurement is equivalent to interpreting meaning within a context. Evolution is used to prepare word states for word compositions. We test two quantum-inspired instances on similarity judgment tasks of individual words and compositions.

Quantum-inspired Machine Learning should not be confused with the area of Quantum Machine Learning. Quantum-inspired models respect some or all of the constraints that are used to model natural phenomena when designing models. In contrast Quantum Machine Learning aims to use these natural phenomena in computational methods to accelerate the learning process and pattern recognition using quantum computers [119].

4.5.1 Quantum States

The central object in quantum theory is a quantum state. The state of a closed quantum system is described by a complex vector $\mathbb{C}^n$. Commonly, quantum states are presented as linear combination of their basis states $|\beta_i\rangle \in \mathbb{B}$ with complex coefficients $z_j \in \mathbb{C}$. The coefficients satisfy $\sum_j^n |z_j|^2 = 1$. A state can be interpreted as a probability distribution over the elements of $\mathcal{B}$. In Dirac's notation a column vector representation of a quantum state, is called *ket* (Equation 4.15). Its counterpart, a row vector representation, is called a *bra* (Equation 4.16). Hence, Dirac's notation is sometimes called the *Bra-Ket* notation. The bra is the complex conjugate of a ket. The conjugate transpose of a matrix $A^\dagger = (\bar{A})^T$ is annotated with the dagger symbol. If a matrix A is Hermitian, or self-adjoint, it is its own conjugate transpose $A = A^\dagger$.

$$|\phi\rangle = \sum_{j=1}^{n} z_j |\beta_j\rangle = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} \tag{4.15}$$

$$\langle\phi| = |\phi\rangle^{\dagger} = \sum_{j=1}^{n} \bar{z_j} |\beta_j\rangle = \begin{bmatrix} \bar{z_1} & \bar{z_2} & \dots & \bar{z_n} \end{bmatrix} \tag{4.16}$$

The linear combination captures superposition of quantum systems. Superposition describes a system that is simultaneously in different, distinguishable basis states. On measurement the state can be found in one of its basis states. The squared magnitude of the coefficient $|z_i|^2$ of an associated basis $|\beta_i\rangle$ returns the probability to find the quantum state in that basis state at the moment of measurement. A state is defined by its complex coefficients $z_j = |z_j|e^{i\theta_j} = r_j(\cos(\theta_j) + i\sin(\theta_j))$. While the magnitudes $|z_j| = r_j$ determine the probability distribution, the phases θ_j participate in compositions.

Quantum states can be classified as pure or mixed states. A pure state is exactly one possible state of a quantum system, and it can be represented by a single ket. A pure state is an exactly known distribution over the basis. A mixed state is a statistical ensemble of several pure states from the same state space. A mixed state defines a distribution over a set of distributions with the same basis.

Any state of quantum system, pure or mixed, can be displayed with a *density matrix* ρ. The density matrix for a pure state is formed by the outer product of its vector representation (Equation 4.17). Generally, a density matrix is a weighted mixture of pure states (Equation 4.18). The probabilities p_i of a pure state are used as weights. The weightings for the pure states satisfy $\sum_i p_i = 1$. One can determine whether a state is pure by computing the trace of the density matrix. If the trace $tr(\rho^2) = 1$, the state ρ is a pure state and if $tr(\rho^2) < 1$ the state ρ is a mixed state.

$$\rho = |\phi\rangle \langle\phi| \tag{4.17}$$

$$\rho = \sum_j p_j |\phi\rangle_j \langle\phi|_j \tag{4.18}$$

The density matrix is hermitian $A = A^{\dagger}$, positive semi-definite $\bar{x}\rho x \geq 0$, of trace one $\sum_i^n |z_i|^2 = \sum_i^n \rho_{ii} = 1$ and of infinite dimensional n.

We propose semantic quantum states represented by $|w\rangle$ for individual words without a context. We also propose to use density matrices ρ_w as a generalised semantic representation for words within contexts, phrases and sentences. Consequently, words are represented by pure states and phrases are represented by mixed states. We will discuss this proposal further in Section 4.5.7.

The orthogonal bases $|\beta_i\rangle \in \mathbb{B}$ used to span the semantic space represent semantic features. They may originate from observable variables, factorisation or generative models. Hence, polysemy is captured as superposition of the semantic basics. Consequently, we interpret the eigen-

states of the density matrix as possible unambiguous meanings or concepts. Let the basis states be the topic vectors $topic_j$ of a latent topic model method. The words w and v can be described by the states $|w\rangle$ and $|v\rangle$:

$$|w\rangle = \sum_j^n a_j e^{i\theta_j} |topic_j\rangle = \sum_j^n |topic_j\rangle \langle topic_j|w\rangle \tag{4.19}$$

with $a_j e^{i\theta_j} = \langle topic_j|w\rangle$.

$$|v\rangle = \sum_j^n b_j e^{i\phi_j} |topic_j\rangle = \sum_j^n |topic_j\rangle \langle topic_j|v\rangle \tag{4.20}$$

with $b_j e^{i\phi_j} = \langle topic_j|v\rangle$.

The probability of words being generated by the j-th topic follows the squared amplitude (complex absolute, magnitude or length).

$$\begin{aligned} |\langle topic_j|w\rangle|^2 = |a_j e^{i\theta_j}|^2 = p(w|topic_j) \\ \text{and} \\ \langle topic_j|v\rangle = |b_j e^{i\phi_j}|^2 = p(v|topic_j) \end{aligned} \tag{4.21}$$

4.5.2 Interference

In this section we will define interference from the perspective of classical mechanics and quantum mechanics. We explain the quantum interference based on the double slit experiment. Eventually, we will discuss its relation to semantic quantum states.

Interference is a physical phenomenon which occurs if two or more waves overlap. It describes the displacement or difference of overlapping wave amplitudes. The total wave amplitude resulting from two or more constituent waves at any point in time is the sum of the displacement of its constituent's amplitudes at that point in time. Interference is called constructive, if the constituent's phases are the same or similar. It is destructive interference, if the phases are opposing. We require the phase and the amplitude of a wave to define their interference. We can consider waves as a linear combination of non-interfering waves. The wave is a superposition of simpler individual waves. In quantum mechanics the wave amplitudes have a probabilistic interpretation. Interference of constituent waves affect the probability distribution over its constituent waves by changing the amplitude's height.

Quantum interference is easily demonstrated by the double slit experiment. This experiment also illustrates the wave-particle duality of photons or electrons. In the double slit experiment a source emits photons or electron waves (e.g. a light source or a electron gun) onto a non-penetrable screen or wall with two parallel slits. Behind the slit screen an absorbing backstop captures the photons or electrons (Figure 4.5.2).

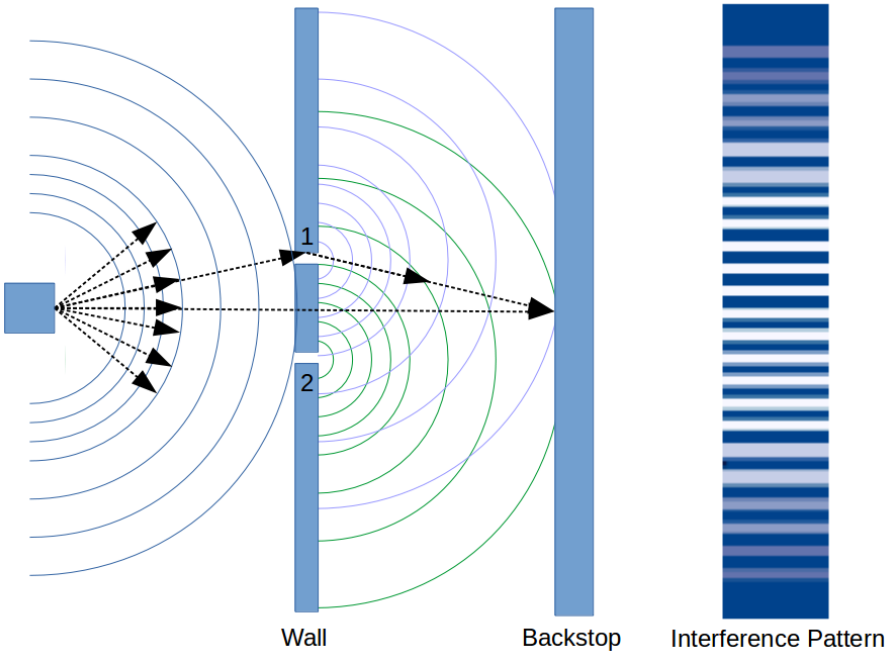

Figure 4.7: Double slit experiment [7].

The electrons pass as particles either through slit 1 or slit 2. Let E_1 be the event that the particle passes through slit 1 and E_2 be the event the particle passes through the 2. The probability $p(E_1)$ and probability $p(E_2)$ must add up to one:

$$p(E_1) + p(E_2) = 1 \tag{4.22}$$

Classical Mechanics suggests that the absorbed electrons create a pattern consisting of two parallel slits (with slightly defused edges) on the backstop. However, in a real-world experiment the pattern which can be observed on the backstop is an interference pattern (Figure 4.5.2).This behaviour is usually attributed to waves, not to particles. The interference pattern suggests that electrons possess both particle and wave properties.

We can describe the pattern on the backstop in terms of their paths. We interpret the emitted electrons as waves with given probability amplitudes. The emitted electron wave passes through the two slits simultaneously. The wave on the other side of the screen can be describe in terms of two sub waves. Let the amplitude $\alpha = ae^{i\theta}$ describe the wave which passes through the slit 1 and the amplitude $\beta = be^{i\phi}$ describe that which passes through the slit 2. The waves corresponds to events E_1 and E_2. Their probability for the electron to pass through the slits $p(E_1)$ and $p(E_2)$ is given in terms of the amplitudes (Equations 4.23 and 4.24). This is summarised by the **Born rule**. It states that the probability of events is given as the squared magnitude of the probability

amplitude of the complex number corresponding to the (sub) wave.

$$p(E_1) = |\alpha|^2 = |ae^{i\theta}|^2 \tag{4.23}$$

$$p(E_2) = |\beta|^2 = |be^{i\phi}|^2 \tag{4.24}$$

Together the probabilities have to satisfy the law of total probability (Equation 4.22), so that $|ae^{i\theta}|^2 + |be^{i\phi}|^2 = 1$. So let the event E_{12} be the event of an electron hitting the backstop at any point passing through any of the two slits. From a classical perspective the two probabilities $p(E_1) + p(E_2)$ should add up to the probability $p(E_{12})$. This would be equivalent of the electrons forming a pattern of two parallel slits and would violate the wave properties. From a quantum perspective the probabilities do not simply add up: $p(E_{12}) \neq p(E_1) + p(E_2) = |\alpha|^2 + |\beta|^2$

The two sub waves α and β interfere with each other on the other side of the slit screen. The new resulting wave amplitude γ describes the probability. Mathematically, α and β are summed up first and then the probability is derived. The probability $p(E_{12})$ is given by the squared magnitude of the total probability amplitude γ:

$$p(E_{12}) = |\gamma|^2 = |\alpha + \beta|^2 = |ae^{i\theta} + be^{i\phi}|^2 = 1 \tag{4.25}$$

The interfering amplitudes $\alpha = ae^{i\theta}$ and $\beta = be^{i\phi}$ describe the total probability amplitude γ. It is in superposition of the two partial waves or the two slits. The total probability amplitude γ is given with:

$$\gamma = ae^{i\theta} + be^{i\phi} \tag{4.26}$$

The probability for the event E_{12} which may occur by two distinct events E_1 and E_2 the interference term is as follows:

$$\begin{aligned} p(E_{12}) &= |\gamma|^2 = |\alpha + \beta|^2 \\ &= (ae^{i\theta} + be^{i\phi})(ae^{i\theta} + be^{i\phi})^* \\ &= (ae^{i\theta} + be^{i\phi})(ae^{-i\theta} + be^{-i\phi}) \\ &= a^2 e^{i(\theta-\theta)} + abe^{i(\theta-\phi)} + abe^{i(\phi-\theta)} + b^2 e^{i(\phi-\phi)} \\ &= a^2 + b^2 + ab(e^{i(\theta-\phi)} + e^{i(\phi-\theta)}) \\ &= a^2 + b^2 + ab(e^{i(\theta-\phi)} + e^{-i(\theta-\phi)}) \end{aligned} \tag{4.27}$$

Euler's formula $e^{i\theta} + e^{i\phi} = \cos(\theta) + i\sin(\theta) + \cos(\phi) + i\sin(\phi)$ and $|e^{i\theta} + e^{i\phi}|^2 = (e^{i\theta} + e^{i\phi})(e^{-i\theta} + e^{-i\phi}) = 2 + 2\cos(\theta - \phi)$ makes this mathematically clearer:

$$p(E_{12}) = |\gamma|^2 = a^2 + b^2 + 2ab\cos(\theta - \phi) \tag{4.28}$$

The first term $a^2 + b^2$ expresses the classical probability, but the second interference term $2ab\cos(\theta - \phi)$ expresses the interaction.

The only plausible interpretation of our observation is that the emitted electron travels through both slits simultaneously and interferes with itself behind the slit screen. The interaction with itself creates the interference pattern on the backstop. Hence, the superposition is a necessary to create the pattern. If we observe a single electron, it indeed only passes through one of slits and hits the backstop. By observing the particle the superposition collapses and the interference with itself does not occur.

Orthogonality is an important property of the superposition. Two basis states given as vectors are orthogonal if their inner product $\langle \beta_i | \beta_j \rangle = 0$. To have distinguishable outcomes the basis vectors need to be orthogonal. Moreover, the Born rule implies:

$$p(\beta_j|\psi) = |\langle \beta_j|\psi\rangle|^2 \tag{4.29}$$

Often it is the case that we cannot say that a quantum state is in a specific state $|\psi\rangle$ with absolute certainty, but that the system is in one of many states $\{|\psi_i\rangle\}$ each with the probability p_i. If we know that a state is a specific superposition the state is a **pure state**. If we have to express the state of a quantum system or particle statistically, the state is a **mixed state**. We can express this in a density matrix ρ with a probability distribution $\{p_i = p(|\psi_i\rangle)\}$ over a set of states $\{|\psi_i\rangle\}$:

$$\rho = \sum_{i=1}^{n} p_i |\psi_i\rangle \langle \psi_i| \tag{4.30}$$

Interference does also appear in the representation of density matrices ρ. For a statistical ensemble ρ with a set of pure states $\{|\psi_i\rangle\}$ weighted with a probability distribution $\{p_i = p(|\psi_i\rangle)\}$ and $\sum_{i=0}^{n} p_i = 1$ (Equation 4.31) the interference between pure states is expressed by the off-diagonal terms. These elements are often referred to as coherences.

$$\begin{aligned}
\rho &= \sum_{i=1}^{n} p_i |\psi_i\rangle \langle \psi_i| \\
&= \sum_{i=1}^{n} p_i \begin{bmatrix} |z_1^i|^2 & z_1^i \bar{z_2}^i & \dots & z_1^i \bar{z_k}^i \\ z_2 \bar{z_1}^i & |z_2^i|^2 & & \\ \vdots & & \ddots & \\ z_k^i \bar{z_1}^i & & & |z_k^i|^2 \end{bmatrix} = \begin{bmatrix} \sum_i p_i |z_1^i|^2 & \sum_i p_i z_1^i \bar{z_2}^i & \dots & \sum_i p_i z_1^i \bar{z_k}^i \\ \sum_i p_i z_2 \bar{z_1}^i & \sum_i p_i |z_2^i|^2 & & \\ \vdots & & \ddots & \\ \sum_i p_i z_k^i \bar{z_1}^i & & & \sum_{w_i} p_i |z_k^i|^2 \end{bmatrix}
\end{aligned} \tag{4.31}$$

Omitting the weighting p_i the entries ρ_{xy} contribute the interference of x-th and y-th basis of

all pure states contributing to this ensemble.

$$\rho_{xy} = \sum_i z_x^i \bar{z_y}^i \tag{4.32}$$

Let us consider Lagrange's identity for real numbers [120]:

$$\begin{aligned}(\sum_{j=1}^{n} a_j b_j)^2 &= (\sum_{j=1}^{n}(a_j)^2)(\sum_{j=1}^{n}(b_j)^2) - \sum_{i=1}^{n-1}\sum_{k=i+1}^{n}(a_i b_k - a_k b_i)^2 \\ &= \sum_{j=1}^{n}(a_j)^2(b_j)^2 + 2\sum_{i=1}^{n-1}\sum_{k=i+1}^{n} a_i b_i a_k b_k\end{aligned} \tag{4.33}$$

and a rewritten Lagrange's identity for complex numbers:

$$\Big|\sum_{j=1}^{n} a_j b_j\Big|^2 = (\sum_{j=1}^{n}|a_j|^2)(\sum_{j=1}^{n}|b_j|^2) - \sum_{i<k}^{n}|a_i b_k^* - a_k b_i^*|^2 \tag{4.34}$$

Lagrange's identity for complex numbers show the interference that appears on the off-diagonal terms of the density matrix ρ. We use Lagrange's identity for the entries ρ_{xy} and set $a_j = z_x^i$ and $b_j = \bar{z_y}^i$ for any quantum state. Again we can identify the term for classical probability and interference.

We propose interference to model implicit word disambiguation in word compositions such as phrases and sentences. Similar to photons or electrons which have properties of waves and particles, compositions and words have semantic and syntactical properties. These properties define a word's behaviour in interactions and the results of these interactions.

Meaning can be ambiguous as compositions and words can have multiple meanings. Individual word meaning can be expressed probabilistically as a superposition of concepts, topics or context words. If these superpositional representations are composed and interact in word compositions, their meanings interfere with each other. Certain concepts or topics are enforced (constructive interference), others cancel each other out (destructive interference). The participating words are disambiguated by and within their context. A semantic interference pattern can be observed on the backstop. It is defined by the syntactic properties of the word composition. This can be achieved by selecting relative phases of participating words appropriately.

Consider the word *bank* as an example. It has multiple meanings represented by concepts or topics: a) financial institute or b) land alongside a river or lake. As long as we do not observe *bank* within word composition it is in a superposition. It can be described as a distribution over these concepts. Observing the word *bank* without a context does not allow us the infer its meaning. We might be able to say that is more often used as the financial institute than to describe the land alongside a river, but we can not be completely sure. Its meaning is uncertain. If we compose it with the word *clerk* to form the compound noun *bank clerk*, the concept of the financial institute

is reinforced. The concept related to land alongside a river is cancelled out. If we interpret *bank* within this composition, we will observe the concept of the financial institute with much higher probability than any alternative meaning. We find that the syntactic structure as well as the constituent words are important to decide which concept is reinforced and which cancelled out. Word sense disambiguation follows a similar principle as interference.

Interference is also capable of modelling commutativity and non-commutativity. Phase shifts can be used to prepare the word representation according to the compositions structure. It is present where the meaning is the same, but syntax differs (e.g. *Bob works in a factory* and *Bob is a factory worker*). Generally, language lacks commutativity. Compositions where the structure stays the same (*Bob likes Mary* and *Mary likes Bob*). The relative phases can model these cases by altering the direction depending on the structure.

4.5.3 State Spaces

Quantum states are elements of a complex Hilbert space $\mathcal{H}^n$. A Hilbert space is a complete vector space with a linearly independent basis and an inner product structure. A vector space is called complete if it does not have any 'missing' vectors. This is the case if any sequence of vectors (Cauchy sequence) that become arbitrarily close to each other converges to another element in the vector space. The basis vectors of a quantum space represent the possible observable states of a quantum system. The basis vectors are required to be orthonormal in order to represent distinguishable outcomes. Non-orthogonal basis states can not reliably be distinguished [121]. Orthonormality is satisfied if their norm $\| |\beta\rangle \| = 1$ and their pairwise inner product $\langle \beta_i | \beta_j = 0 \rangle$. Note that $\langle \phi | \phi \rangle = 1$, due to the law of total probability.

$$\langle \phi | \psi \rangle = \langle \phi | \, | \psi \rangle = \langle \phi | \psi \rangle = \begin{bmatrix} \bar{z_1} & \bar{z_2} & \dots & \bar{z_n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{4.35}$$

Any quantum-inspired semantic space is a complex Hilbert space which is spanned by orthonormal concepts or topics. It contains semantic quantum states and provides a inner product structure. The inner product is a generalisation of the dot product and allows us to define distances within a vector spaces. Both are frequently used as a measure of semantic similarity in semantic vector spaces. We propose to use the inner product structure analogous to the dot product as a semantic measure between complex-valued representations within a Hilbert space.

4.5.4 Evolution

When a closed quantum system changes, a unitary linear operator U is applied onto the original state [121]. The evolution of a quantum system is described by a unitary transformation (Equation 4.36). The unitary operator preserves the structure and probabilistic properties of the state vectors. A unitary transformation is reversible. A unitary matrix is a complex-valued squared matrix if its conjugate transpose is its inverse: $U^\dagger = U^{-1} \rightarrow U^\dagger U = I$.

$$U\,|\phi\rangle = |\phi\rangle' \tag{4.36}$$

This postulate is often used to prepare quantum states for composition or computation. States are often prepared by shifting the basis states into a superposition using a Hadamard operator H (Equation 4.37).

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.37}$$

The phase shift operator R_θ is an essential operator. It allows shifting phases without affecting amplitudes. The entries $e^{i\theta}$ implement the shifts of corresponding bases (Equation 4.38). The example shifts the phase of first basis state $|\beta_1\rangle$ to $e^{i\theta}\,|1\rangle$ and does not affect the basis $|\beta_2\rangle$.

$$R_\theta = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \tag{4.38}$$

The operators R_θ and H are defined for tensors of rank two. We propose the unitary evolution as a generalised preparation step for complex-valued semantic representations. Further, its definition can be used as a constraint in neural architectures.

4.5.5 Composition

In order to compose n quantum systems their composite state space is the tensor product of the components' Hilbert spaces (Equation 4.39). The size of the composite state grows exponentially with the number of systems.

$$\mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3 \otimes \ldots \mathcal{H}_n = ((\mathcal{H}_1 \otimes \mathcal{H}_2) \otimes \mathcal{H}_3) \otimes \ldots \mathcal{H}_n \tag{4.39}$$

The simplest composite states are the tensor products of pure quantum states $|\psi\rangle$ and $|\phi\rangle$ (Equations 4.40 and 4.41). The tensor product of two vectors is equal to the left Kronecker product.

$$
\begin{aligned}
|\psi\rangle_A \otimes |\phi\rangle_B = (\sum_{i=1}^{n} a_i |\alpha_i\rangle) \otimes (\sum_{j=1}^{m} b_i |\beta_j\rangle) = \\
\sum_{i=1}^{n}\sum_{j=1}^{m} a_i b_i |\alpha_i\rangle |\beta_j\rangle = \sum_{i=1}^{n}\sum_{j=1}^{m} a_i b_i |\alpha_i \beta_j\rangle
\end{aligned}
\tag{4.40}
$$

$$
|\psi\rangle_A \otimes |\phi\rangle_B = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ a_2 b_1 \\ a_2 b_2 \end{bmatrix} \tag{4.41}
$$

Composition of two mixed quantum states ρ_A and ρ_B are more complex and computationally expensive (Equation 4.42).

$$
\rho_A \otimes \rho_B = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mm} \end{bmatrix} = \begin{bmatrix} a_{11}\rho_B & a_{12}\rho_B & \dots & a_{1n}\rho_B \\ a_{21}\rho_B & a_{22}\rho_B & \dots & a_{2n}\rho_B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}\rho_B & a_{n2}\rho_B & \dots & a_{nn}\rho_B \end{bmatrix} \tag{4.42}
$$

with

$$
a_{ij}\rho_B = \begin{bmatrix} a_{ij}b_{11} & a_{ij}b_{12} & \dots & a_{ij}b_{1m} \\ a_{ij}b_{21} & a_{ij}b_{22} & \dots & a_{ij}b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{ij}b_{m1} & a_{ij}b_{m2} & \dots & a_{ij}b_{mm} \end{bmatrix} \tag{4.43}
$$

We suggest to compose quantum-inspired semantic spaces in order to analyse the relationship between two sets of embeddings. These can originate from different methods, modalities, corpora or languages. The combinatorial nature of the composition postulate allows us theoretically to consider all possible meanings over semantic spaces. However, computationally this structure is expensive to implement.

4.5.6 Measurements

Measurements in a quantum mechanics are not deterministic, but statistical. We present the measurement postulate formally by introducing two kinds of measurements: projective measurements (or von Neumann measurements) and positive operator-valued measurements (POVM). The measurement postulate is closely related to evolution, since any measurement acts on the

state space. Generally, it maps a state $|\psi\rangle$ to new state $|\psi'\rangle$ by applying one or more measurement operators:

$$M|\psi\rangle = |\psi'\rangle \text{ with } |\psi\rangle, |\psi'\rangle \in \mathcal{H} \tag{4.44}$$

4.5.6.1 Projective Measurement

In Quantum Mechanics a projective measurement is characterised by a measurable physical property of a quantum state (e.g. position, translational momentum, orbital angular momentum, spin, total angular momentum, energy). It is represented by a Hermitian operator, also called an observable A. Mathematically, a projective measurement is an orthogonal projection into an eigenspace V_j of the observable A. Therefore, the observable has a decomposition which is a linear combination of projectors P_j :

$$A = \sum_j \lambda_j P_j = \sum_j \lambda_j |a_j\rangle \langle a_j| \tag{4.45}$$

The coefficients λ_j are eigenvalues of an observable A and correspond to possible outcomes which may or may not occur after the measurement. The eigenvectors (or eigenstates) $|a_j\rangle$ span the subspace V_j in which we are projecting the measured quantum state. A single specific outcome λ_j occurs with the probability $p(\lambda_j)$:

$$p(\lambda_j) = \langle\psi| P_j |\psi\rangle = \langle\psi|a_j\rangle \langle a_j|\psi\rangle = |\langle\psi|a_j\rangle|^2 \tag{4.46}$$

using the projector $P_j = |a_j\rangle \langle a_j|$, an idempotent (self-adjoint) $P_j = P_j^2$ operator. This corresponds to the Born rule (Section 4.5.2).

The expectation value $\langle A\rangle$ of an observable A of a state $|\psi\rangle$ is defined as:

$$\begin{aligned} \langle A\rangle &= \sum_j \lambda_j p(\lambda_j) = \sum_j \lambda_j \langle\psi| P_j |\psi\rangle \\ &= \langle\psi| (\sum_j \lambda_j P_j) |\psi\rangle = \langle\psi| A |\psi\rangle \end{aligned} \tag{4.47}$$

After the measurement the state $|\psi\rangle$ is left in a new state $|\psi'\rangle$:

$$|\psi'\rangle = \frac{P_j |\psi\rangle}{\sqrt{\langle\psi| P_j |\psi\rangle}} = \frac{P_j |\psi\rangle}{\sqrt{p(a_j)}} \tag{4.48}$$

Intuitively, the probabilities of possible outcomes have to sum to 1: $\sum_j p(\lambda_j) = \langle\psi| P_j |\psi\rangle = 1$. The set of projectors $\{P_j\}$ has to satisfy the completeness relation:

$$\sum_j P_j = I \tag{4.49}$$

Previous equations have all considered an arbitrary pure state $|\psi\rangle$. However, projective measurements behave similarly on mixed states ρ. Given a mixed state $\rho = \sum_i p_i |\phi_i\rangle \langle\phi_i|$ the probability of an observable A to be a specific eigenvalue λ_j of A is given with:

$$\begin{aligned} p(\lambda_j) &= \sum_i p_i \langle\phi_i| |a_j\rangle \langle a_j| |\phi_i\rangle \\ &= \sum_i p_i Tr(|\phi_i\rangle \langle\phi_i| P_j) \\ &= Tr(\sum_i p_i |\phi_i\rangle \langle\phi_i| P_j) \\ &= Tr(\rho P_j) \end{aligned} \tag{4.50}$$

The expectation value of the observable $\langle A\rangle$ for a mixed state $\rho = \sum_i p_i |\phi_i\rangle \langle\phi_i|$ is defined as:

$$\begin{aligned} \langle A\rangle &= \sum_i p_i \langle\phi_i| A |\phi_i\rangle \\ &= \sum_i p_i Tr(|\phi_i\rangle \langle\phi_i| A) \\ &= Tr(\sum_i p_i |\phi_i\rangle \langle\phi_i| A) \\ &= Tr(\rho A) \end{aligned} \tag{4.51}$$

After measurement the system is left in the state ρ'_j :

$$\rho'_j = \frac{P_j \rho P_j^\dagger}{tr(P_j^\dagger P_j \rho)} = \frac{P_j \rho P_j}{Tr(P_j \rho)} \tag{4.52}$$

Equation 4.52 describes the state of a sub-ensemble. The complete ensemble (or statistical state) is described as follows:

$$\begin{aligned} \rho' &= \sum_j p(\lambda_j) \rho'_j \\ &= \sum_j Tr(P_j \rho) \frac{P_j \rho P_j}{tr(P_j \rho)} \\ &= \sum_j P_j \rho P_j \\ &= \sum_j P_j \rho P_j^\dagger \end{aligned} \tag{4.53}$$

The operators P_j of a projective measurement satisfy **completeness relation, orthogonality, idempotence and they are Hermitian**. Making a measure in a basis $|a_i\rangle$ is equal to making a projective measurement. It projects a quantum state $|\psi\rangle$ with a projector $P_j = |a_j\rangle \langle a_j|$ into a

subspace of the state space $\mathcal{H}$. For this the eigenstates $|a_j\rangle$ of the observable A the basis states $|e_j\rangle = |x_j\rangle$ of the Hilbert space $\mathcal{H}$ are implicitly chosen. If we now consider the state after the measurement, pure $|\psi'\rangle$ or mixed ρ' , we see that physical measurements result in an information loss. From a mathematical perspective, this is due to the projection into an eigenspace of the observable A. Physically, the state collapses. Superposition information is irreversibly lost, since the measurement apparatus itself interacts with the measured quantum system as a new system. The system we want to measure has lost its coherence. This process of losing information is called decoherence.

4.5.6.2 Positive Operator-Valued Measurements

In general, any measurement is characterised by a set of Hermitian measurement operators $\{M_j\}$. Since they are Hermitian, the operators are there own complex conjugate transpose $M_j = M_j^\dagger$. The set of measurement operators satisfies the completeness relation (Equation 4.54), so that the probabilities $p(j)$ of all possible outcomes sum to 1.

$$\sum_j M_j^\dagger M_j = I \tag{4.54}$$

The index j corresponds to the possible outcomes. The probability $p(j)$ that j occurs upon measurement is given with:

$$p(j) = \langle\psi| M_j^\dagger M_j |\psi\rangle \tag{4.55}$$

A system's state $|\psi'\rangle$, after the measurement with a specific M_j, is given with:

$$|\psi'\rangle = \frac{M_j |\psi\rangle}{\sqrt{\langle\psi| M_j^\dagger M_j |\psi\rangle}} \tag{4.56}$$

The probability $p(j)$ in terms of a mixed state $\rho = \sum_i p_i |\phi_i\rangle \langle\phi_i|$ is given with:

$$\begin{aligned} p(j) &= \sum_i p_i \langle\phi_i| M_j^\dagger M_j |\phi_i\rangle \\ &= \sum_i p_i Tr(|\phi_i\rangle \langle\phi_i| M_j^\dagger M_j) \\ &= Tr(\sum_i p_i |\phi_i\rangle \langle\phi_i| M_j^\dagger M_j) \\ &= Tr(M_j^\dagger M_j \rho) \end{aligned} \tag{4.57}$$

Applying a measurement operator M_j on a mixed state ρ leaves the system in a new state ρ'_j:

$$\rho'_j = \frac{M_j \rho M_j^\dagger}{Tr(M_j^\dagger M_j \rho)} \tag{4.58}$$

The complete ensemble after the measurement is given with a state ρ' :

$$\begin{aligned}\rho' &= \sum_j p(j)\rho'_j \\ &= \sum_j Tr(M_j^\dagger M_j \rho)\frac{M_j \rho M_j^\dagger}{Tr(M_j^\dagger M_j \rho)} \\ &= \sum_j M_j \rho M_j^\dagger\end{aligned} \tag{4.59}$$

POVMs allow more freedom of choice for the basis of the measurement operators M_j. If the linear operators P_j are idempotent $P_j^2 = P_j$. This property makes the operator a projector. Projectors can be applied multiple times without the measurement outcome changing and they are their own complex conjugate transpose. Moreover, this projection is orthogonal. A projection is orthogonal if and only if the projectors are hermitian $A = A^\dagger$. Unlike the operators P_j for a projective measurement, the operators M_j for a POVM do not have to satisfy orthogonality and idempotence. For a POVM measurement it is sufficient that the operators are positive and satisfy the completeness relation. Projective measurements are POVMs, due to their orthogonality and idempotence $P_j = M_j$:

$$M_j^\dagger M_j = M_j M_j = (M_j)^2 = M_j = P_j \tag{4.60}$$

Quantum mechanical observations are not necessarily intuitive, since the act of observing causes the state to collapse. We propose the measurement postulate as the process of interpreting a word's meaning within a context. An isolated word representation is polysemous and is therefore ambiguous. In context, however, a word loses this property or at least the uncertainty is reduced. In an optimal case exactly one meaning is reinforced and all other possible meanings reduced by the context. Using the eigenvectors of the polysemous representation, we can construct a set of measurement operators. This allows us to measure word states towards a certain meaning. The state after measurement is the disambiguated word within a context. The act of understanding a word within its context is the act of measurement. However, to understand a meaning and reconstruct all possible meanings, we will have to measure it multiple times.

We propose projective measurements as tools to investigate and compare the meaning spaces of words, phrases, sentences or even documents. The projective measurements project the state into semantic sub spaces derived from another phrase. We assume the other phrase to be an observable. This may provide us with possibilities to detect paraphrases or semantically similar documents.

4.5.7 Quantum-Inspired Semantic Framework

There have been many approaches to formalise quantum-inspired Semantic Spaces. Some take a theoretical perspective and study semantic spaces themselves [122, 123]. Others take more practical approach. Since we have formalised our framework, strongly related work has been published [95, 124, 125]. Our approach differs to previous attempts in the semantic interpretation of quantum theoretic principles and their consequences. Most recently, there have been very similar approaches published. They also rely on density matrices as semantic representations to capture word polysemy in pure states and compositions as mixed states [126, 117].

We generalise Semantic Spaces and operations acting on semantic representations and spaces. Our framework allows us to transfer properties and metrics from Quantum Mechanics to language. Entropy can be used as a metric for syntactic complexity of phrases, Hilbert spaces allow us to define semantic similarity by the inner product and eigenvectors represent distinguishable meanings or concepts. Measurement operators created using eigenvectors interpret semantic density matrices. We use $topic_j$ in our definitions to denote learned semantic features. They may originate from classical latent topic models, but may also be learned entirely in constrained embedding layers.

Our generalised framework represents words as pure and phrases as mixed states in the same semantic space. Both are defined by a statistical ensemble as density matrix. This allows us to interpret mathematical properties of density matrices and its use in Quantum Mechanics from a semantic perspective.

We represent words w by a pure state $|w\rangle$ (Equation 4.61) or with their corresponding density matrix ρ_w (Equation 4.62). Syntactical information are captured in relative phases θ_j and the semantic information by the magnitude $|z_j|$ of complex coefficients $z_j = |z_j|e^{i\theta_j}$.

$$|w\rangle = \sum_j^k z_j \, |topic_j\rangle = \sum_j^k \sqrt{p(w|topic_j)} e^{i\theta_j} \, |topic_j\rangle \tag{4.61}$$

For an arbitrary word representation as pure state $|w\rangle$ the corresponding density matrix ρ_w is defined as:

$$\rho_w = |w\rangle \langle w| = \begin{bmatrix} z_1 \\ \dots \\ z_k \end{bmatrix} \begin{bmatrix} \bar{z_1} & \dots & \bar{z_k} \end{bmatrix} = \begin{bmatrix} |z_1|^2 & z_1\bar{z_2} & \dots & z_1\bar{z_k} \\ z_2\bar{z_1} & |z_2|^2 & & \\ \vdots & & \ddots & \\ z_k\bar{z_1} & & & |z_k|^2 \end{bmatrix} \tag{4.62}$$

The density matrix of size $k \times k$ where k is the number of features. A phrase or sentence of length s given with a sequence S of words w_i can be expressed as mixed state ρ_p of the

contributing word states (Equation 4.63). The meaning of the sentence is a weighted sum (a statistical ensemble) of the possible meanings of the contributing words. Each element of a phrase is prepared by a unitary evolution and weighted into an ensemble represented by a mixed state.

$$\begin{aligned}
\rho_S &= \sum_{w_i \in S} p_i \left|w_i\right\rangle \left\langle w_i\right| \\
&= \sum_{w_i \in S} p_i \rho_{w_i} \\
&= \sum_{w_i \in S} p_i \begin{bmatrix} |z_1^i|^2 & z_1^i \bar{z_2}^i & \dots & z_1^i \bar{z_k}^i \\ z_2 \bar{z_1}^i & |z_2^i|^2 & & \\ \vdots & & \ddots & \\ z_k^i \bar{z_1}^i & & & |z_k^i|^2 \end{bmatrix} \\
&= \begin{bmatrix} \sum_i p_i |z_1^i|^2 & \sum_i p_i z_1^i \bar{z_2}^i & \dots & \sum_i p_i z_1^i \bar{z_k}^i \\ \sum_i p_i z_2 \bar{z_1}^i & \sum_i p_i |z_2^i|^2 & & \\ \vdots & & \ddots & \\ \sum_i p_i z_k^i \bar{z_1}^i & & & \sum_{w_i} p_i |z_k^i|^2 \end{bmatrix}
\end{aligned} \tag{4.63}$$

with $\sum_{i=0}^{s} p_i = 1$. The resulting compositional density matrix ρ_S captures information of classical probabilities on its diagonal. The off-diagonal elements capture non-classical information as interference terms.

This framework gives us the freedom to represent words and compositions of arbitrary length s with the same dimensioned representation. The framework uses weighted addition with preceding state evolution in order to compose words of arbitrary length. The word representations are prepared or weighted according to the syntactical structure in S or a specific task. Further, we are able to use various tools to analyse and compare their meaning. The relative phases can be derived for the composition structure.

With this framework, many properties and tools developed to analyse quantum mechanical systems can be used for the semantic and syntactic analysis of words and sentences:

The inner product structure (Equation 4.35) of the Hilbert space describes a semantic distance between the pure states of two words w and v given as $|w\rangle$ and $|v\rangle$. As the generalisation of the dot product, we can use this to identify semantic clusters within the semantic state space.

The Von Neumann entropy is widely used in Quantum Information Theory. It is a measure for the *mixedness* or randomness of a statistical ensemble (Equation 4.64).

$$S(\rho) = -tr(\rho \ln(\rho)) \tag{4.64}$$

If the state ρ is written with its eigenvectors $|\psi_i\rangle$ and the corresponding eigenvalues λ_i $\rho = \sum_i \lambda_i |\psi_i\rangle \langle\psi_i|$ the computation simplifies to Equation 4.65.

$$S(\rho) = -\sum_i \lambda_i \ln(\lambda_i) \tag{4.65}$$

We note that the entropy $S(\rho) = 0$ if ρ is a pure state. The maximal entropy of a mixed state ρ is $S(\rho) = \ln(k)$. In our framework entropy is a measure for syntactic complexity and semantic interaction. Entropy not only allows us to compare sentences semantically, but also to quantify semantic interaction within a sentence caused by its structure.

Eigenvectors of density matrices allow us to represent the density matrix as a mixture of its eigenvectors and eigenvalues. Within our framework the eigenvectors can be interpreted as the fundamental meanings of sentences, phrases or words. We are able to semantically compare sentences or phrases of different lengths. Eigenvalues and eigenvectors have the following properties:

- While the k eigenvalues of any density matrix are real, the corresponding eigenvectors may be complex.
- There exist an complex orthonormal basis such that all off-diagonal elements (coherences) are 0, since ρ is Hermitian.
- Eigenvectors are orthogonal for distinct eigenvalues, since ρ is Hermitian.
- Any finite density operator can be diagonalised by a unitary matrix, since ρ is Hermitian (finite-dimensional spectral-theorem).
- Any finite density operator has k linearly independent eigenvectors, since ρ is Hermitian and its eigenvalues are distinct.

Due to the finite-dimensional spectral theorem, we can find eigenvalues and eigenvectors by decomposing:

$$\rho = U\Lambda\bar{U} \tag{4.66}$$

with U being a unitary matrix containing orthogonal eigenvectors as columns and Λ being a diagonal matrix containing only real eigenvalues. Furthermore, we can use the Gram-Schmidt procedure to find an orthonormal basis for the complete state space using the eigenvectors. This way we are able to compare the meanings of sentences or phrases of different lengths. This comparison could be achieved by pairwise calculation of the cosine similarity.

The measurement postulate using projective measurement or POVM (Section 4.5.6) allows us to use the eigenstates for semantic analysis of sentences. Since we interpret a density matrix's eigenvectors as fundamental meanings, we are able to measure these eigenvectors and calculate

their probabilities. We construct a projective measurement from the eigenvectors of a density matrix and can conduct measurements of meaning for phrases or words. The information of eigenvectors also helps us to discover word disambiguation. Formulating projective measurements based on the word's eigenvectors and conducting measurements on the phrase state we can discover the natural disambiguation in the composition. The act of measurement is the linguistic equivalent of understanding within a context. The eigenvectors of the density matrix will be linearly independent, so that we can interpret them as truly distinct meanings.

An alternative is to construct observables A, B from two phrases ρ_A, ρ_B and measure the observable A on the quantum state ρ_B and the observable B on the quantum state ρ_A. With these procedures we can discover mutual meanings or decide over the semantic similarity of phrases of different length.

If we construct a POVM measurement or a projective measurement using the word representations used in a composition, then measure the compositional state and analyse the state after the measurement we are able to make statements of word disambiguation of a specific word. We can facilitate this model to achieve paraphrase detection and other evaluation tasks. We construct a measurement operator for a POVM measurement using the isolated word states of a paraphrase and measure the mixed state of the sentence which we want to test for this specific paraphrase. While we can use the inner product of two semantic states $|w\rangle$ and $|v\rangle$ of two words w and v, the semantic measurement of an observable A interprets the disambiguated meaning of the composition vw with respect to an eigenvector of the entire semantic space, word or phrase.

Overall, our approach allows us to intuitively interpret eigenvectors, entropy and inner products directly to properties of words and word compositions. We consider them as tools and methods for semantic and syntactic analysis.

4.5.8 Semantic Spaces

In this subsection we present different models of language inspired by previous work in Computational Semantics. These models are simple instances of our framework. We translate the real-valued Simple Semantic Space and the Simple Topic Space into our quantum-inspired semantic framework using pure states.

4.5.8.1 Simple Semantic Space

Simple Semantic Spaces follow the distributional hypothesis. Words are represented by vectors containing measures derived from co-occurrence statistics. Co-occurrence is defined using a context window of a size c. Each dimension of a target word vector $w \in V$ corresponds to a possible context word $v \in V$, such that $n = |V|$. The vectors span the vector space $\mathbb{R}^n$. The necessary counts are directly extracted from the corpus. Previous research has also shown that modern neural models used to learn general-purpose embeddings achieve this by applying factorisation

methods on these types of measures [18]. Hence, we use Pointwise Mutual Information (PMI) of various scaling. The Simple Semantic Space and the different variants serve as baselines. We also show the performance compositional similarity judgments.

PMI is a measure of association widely used in Computational Semantics. It is based on co-occurrence and word appearance statistics. For two words w and v PMI is defined as shown in Equation 4.67.

$$pmi(w, v) = \log\left(\frac{p(w, v)}{p(w)p(v)}\right) \tag{4.67}$$

We refer to the variants of PMI as pmi_a, pmi_b and pmi_c. All of them are based directly on the co-occurrence frequency $f(w, v)$ of a context word v with a target word w. Our variants of the PMI pmi_a, pmi_b, pmi_c (Equations 4.68, 4.69, 4.70) differ in scaling or collection procedure (i.e. how sums are extracted from the statistics).

$$\begin{aligned} pmi_a(w, v) &= \log(1 + \frac{p(w, v)}{p(w)p(v)}) \\ &= \log(1 + \frac{\frac{f(w,v)}{N}}{\frac{f(w)}{N}\frac{f(v)}{N}}) \\ &= \log(1 + \frac{\frac{f(w,v)}{N}}{\frac{f(w)f(v)}{N^2}}) \\ &= \log(1 + \frac{f(w, v)N}{f(w)f(v)}) \end{aligned} \tag{4.68}$$

$$\begin{aligned} pmi_b(w, v) &= \log(\frac{p(w, v)}{p(w)p(v)}) \\ &= \log(\frac{\frac{f(w,v)}{N}}{\frac{f(w)}{N}\frac{f(v)}{N}}) \\ &= \log(\frac{\frac{f(w,v)}{N}}{\frac{f(w)f(v)}{N^2}}) \\ &= \log(\frac{f(w, v)N}{f(w)f(v)}) \end{aligned} \tag{4.69}$$

$$
\begin{aligned}
pmi_c(w, v) &= \log(\frac{p(w, v)}{p(w)p(v)}) \\
&= \log(\frac{\frac{f(w,v)}{M}}{\frac{f'(w)}{M}\frac{f'(v)}{M}}) \\
&= \log(\frac{\frac{f(w,v)}{M}}{\frac{f'(w)f'(v)}{M^2}}) \\
&= \log(\frac{f(w, v)M}{f'(w)f'(v)})
\end{aligned}
\tag{4.70}
$$

with $f(w) = \sum_{v \in V} f(w, v)$ and $N = \sum_{w \in V} f(w)$ so that that $p(w, v) = \frac{f(w,v)}{N}$ and $p(x) = \frac{f(x)}{N}$. Also $f'(w)$ given as the frequencies of word w in the corpus and $M = \sum_{w \in V} f'(w)$, so that $p(w, v) = \frac{f'(w,v)}{M}$ and $p(x) = \frac{f'(x)}{M}$.

Our first baseline uses the raw frequency counts, so that a word w is represented by a vector containing counts of context words v_j (Equation 4.71). The entries v_j correspond to frequencies $f(w, v_j)$ being the word v_j within the context of w.

$$
w = \begin{bmatrix} f(w, v_1) & \dots & f(w, v_n) \end{bmatrix} \tag{4.71}
$$

Another strong baseline is achieved by normalising the counts to probabilities, thus the vector is of unit length $|w| = 1$. Entries v_j correspond to the probability $p(w, v_j) = \frac{f(w,v_j)}{f(w)}$ that the word v_j appears within the context of a target word w.

$$
w = \begin{bmatrix} p(w, v_1) & \dots & p(w, v_n) \end{bmatrix} \tag{4.72}
$$

Other baseline embeddings use the PMI scoring functions presented above:

$$
w = \begin{bmatrix} pmi_a(w, v_1) & \dots & pmi_a(w, v_n) \end{bmatrix} \tag{4.73}
$$

with entries $w_j = pmi_a(w, v_j) = \log\left(1 + \frac{p(w,v_j)}{p(w)p(v_j)}\right) = \log\left(1 + \frac{f(w,v_j)N}{f(w_j)f(v_j)}\right)$.

$$
w = \begin{bmatrix} pmi_b(w, v_1) & \dots & pmi_b(w, v_n) \end{bmatrix} \tag{4.74}
$$

with entries $w_j = pmi_b(w, v_j) = \log\left(\frac{p(w,v_j)}{p(w)p(v_j)}\right) = \log\left(\frac{f(w,v_j)N}{f(w)f(v_j)}\right)$.

$$
w = \begin{bmatrix} pmi_c(w, v_1) & \dots & pmi_c(w, v_n) \end{bmatrix} \tag{4.75}
$$

with entries $w_j = pmi_c(w, v_j) = \log\left(\frac{p(w,v_j)}{p(w)p(v_j)}\right) = \log\left(\frac{f(w,v_j)M}{f'(w)f'(v_j)}\right)$.

4.5.8.2 Simple Semantic Quantum Space

Generally, we define the Simple Semantic Quantum Space as Hilbert Space consisting of word quantum states $\{|w\rangle\}$. They are defined as a linear combination of orthonormal basis states $|v_j\rangle$ which represent PMI score with the context words v_j.

Each complex coefficient $z_j = x + iy = r_j e^{i\theta_j}$ satisfies $\sum_{v_j \in V} |z_j|^2 = 1$. The relative phases θ_j and the amplitude r_j are learned by a model or set to be a corpus metric or statistic. In this instance of our semantic quantum framework the coefficients indicate the probabilities or scores to find the word $|w\rangle$ in context with $|v_j\rangle$ upon measurement. A word w is represented with a quantum state $|w\rangle$ (Equation 4.76).

$$w = \sum_{v_j \in V} z_j |v_j\rangle = \sum_{v_j \in V} r_j e^{i\theta_j} |v_j\rangle \tag{4.76}$$

Phases can be imposed by applying a phase shift operator R_θ onto a word representation $|w\rangle$ (Equation 4.77). The phase shift is parametrised by a vector of relative phases θ.

$$w' = R_\theta w = \begin{bmatrix} e^{i\theta_1} & 0 & \dots & 0 \\ 0 & e^{i\theta_2} & & \\ \vdots & & \ddots & \\ 0 & & & e^{i\theta_n} \end{bmatrix} w \tag{4.77}$$

We create a quantum state from co-occurrence statistics by normalising the counts. To satisfy the a quantum state, we compute the square root of the coefficients. A word representation $|w\rangle$ is given with:

$$|w\rangle = \sum_{v_j \in V} \sqrt{\frac{f(w, v_j)}{\mathcal{N}}} |v\rangle = \sum_{v_j \in V} \sqrt{p(w, v_j)} |v_j\rangle \tag{4.78}$$

We use the L1-norm defined with a norm $\mathcal{N} = \sum_{v_j \in V} f(w, v_j)$.

Other variants of the PMI $pmi(w, v)$ normalise the score to $npmi(w, v)$. The computed scores are within the interval $[-1, +1]$ (Equation 4.79). The $npmi$ score is -1 if there is total anti-correlation, 0 if there is no correlation and $+1$ if there is total positive correlation [127].

$$npmi(w, v) = \frac{pmi(w, v)}{-\log(p(w, v))} \tag{4.79}$$

Our other variants make use of the normalised variants of PMI. For the case that $npmi(w, v_j)$ results in negative numbers, the score is simply considered as a complex number with a phase of π, otherwise 0.

$$|w\rangle = \frac{1}{\mathcal{N}} \sum_{v_j \in V} npmi_x(w, v_j) |v_j\rangle \tag{4.80}$$

The factor $\frac{1}{\mathcal{N}}$ is a normalisation factor to impose the constraint $\sum_{v_j \in V} |z_j|^2 = 1$ on a quantum state. So far we have used our Simple Semantic Quantum state only with phases of 0 or π and, therefore, the entries are of the same or opposing phases. We extend our approach using the normalised pointwise mutual information to define the phase $\theta_j = cos^{-1}(npmi(w, v_j))$ for each complex coefficient.

$$|w\rangle = \frac{1}{\mathcal{N}} \sum_{v_j \in V} |npmi_x(w, v_j)| e^{i\left(cos^{-1}(npmi_x(w, v_j))\right)} |v_j\rangle = \frac{1}{\mathcal{N}} \sum_{v_j \in V} c\, |v_j\rangle \tag{4.81}$$

The occurrence and co-occurrence statistics derived from the corpus contain limited information about the possible composition without explicitly taking syntactical structure into account.

4.5.8.3 Simple Topic Quantum Space

We define a Simple Topic Quantum Space as a quantum embedding of the Simple Topic Space. In the Simple Topic Quantum Space we use topics from a latent topic model as basis states $|topic_j\rangle$. Each basis has corresponding complex coefficients $t_j = x + iy = r_j e^{i\theta_j}$ which satisfy $\sum_{topic_j \in V} |t_j|^2 = 1$. The probability $p(topic_j||w\rangle) = p(w|topic_j) = |t_j|^2$ expresses the probability of the word w to be generated by the j-th topic. A word w is defined with a quantum state $|w\rangle$ (Equation 4.82).

$$|w\rangle = \sum_{t_j \in \mathcal{T}} t_j |v_j\rangle = \sum_{v_j \in V} \sqrt{p(w|topic_j)} e^{i\theta_j} |v_j\rangle \tag{4.82}$$

4.5.9 Experiments

Our experiments focus on two important properties in semantic spaces. Semantic similarity and compositionality. In the past there have been a number of considerations of compositionality in vector-based semantic spaces [128]. They focus on combining vector representation of words into various composition types like compound noun, adjective-noun, verb-object or subject-verb-object [3, 129, 130]. We continued this work for quantum-inspired models as their natural extension.

Table 4.5 shows the performance of the different models in the semantic similarity and relatedness task for words. For a more detailed comparison between classical and quantum-inspired models, we allow the composition of pointwise addition $\oplus$ and pointwise multiplication $\odot$. Both have been found to be strong baselines [3]. We compare complex-valued $npmi_x$ models and complex-valued $npmi_x$ phase models to their real-valued pmi_x counterparts. Note that while the $npmi_x$ models do not explicitly define a phase, they still have direction. They can be negative or positive, hence their phase is either 0 or π. We report the performance of the unprocessed count vectors (*raw*) or the normalised probabilities as representations.

We found that complex-valued Simple Semantic Quantum Spaces outperform their real-valued Simple Semantic Spaces in word similarity and word relatedness judgments. All quantum models show improved correlation of similarity (SimLex-999) and relatedness judgement (WordSim353) compared to their real-valued baselines. Exceptions are phase models using $npmi_b$ and $npmi_c$. Another comparison between the quantum-like models using equal phases ($npmi_a, npmi_b, npmi_c$) and models using NPMI information for their phases ($npmi_a$ phases, $npmi_b$ phases, $npmi_c$ phases) shows that NPMI phase information decrease the performance. Especially for $npmi_b$ phases, $npmi_c$ phases the performance drops significantly.

Table 4.5: Semantic similarity computed by the inner product of word representations evaluated using WordSim353 [1] and SimLex999 [2]. Results given in Spearman correlation ρ calculated by averaging over participant correlation. Arrows $\uparrow, \downarrow, \bullet$ indicate the performance compared to model's direct baseline.

	WordSim353	SimLex999
Simple Semantic Space		
- raw	0.03421	-0.0473
- probability	0.31530	0.13979
- pmi_a	0.42813	0.18239
- pmi_b	0.35153	0.21168
- pmi_c	0.38594	0.16174
Simple Semantic Quantum Space		
- normalised probability amplitude	0.44581 $\uparrow$	0.22998 $\uparrow$
- $npmi_a$	**0.47046** $\uparrow$	0.21569 $\uparrow$
- $npmi_b$	0.39408 $\uparrow$	**0.23467** $\uparrow$
- $npmi_c$	0.39702 $\uparrow$	0.17132 $\uparrow$
- $npmi_a$ phases	**0.46913** $\uparrow$	0.21521 $\uparrow$
- $npmi_b$ phases	0.17549 $\downarrow$	0.08304 $\downarrow$
- $npmi_c$ phases	0.30796 $\downarrow$	0.12289 $\downarrow$
Simple Topic Space		
- 50 topics	0.18855	0.19004
- 200 topics	0.18164	0.21271
Simple Topic Quantum Space		
- 50 topics	0.19819 $\uparrow$	0.20179 $\uparrow$
- 200 topics	0.18637 $\uparrow$	0.18499 $\downarrow$

Table 4.6 shows the results of simple composition using pointwise addition $\oplus$. Using this composition function is of particular interest, since interference occurs in the complex addition.

Generally, we found that the quantum-inspired embeddings outperform the classical baselines in all composition types under addition. Quantum embeddings using $npmi_a$ and $npmi_a$ phases perform best for compound nouns and adjective-nouns. For verb-object and subject-verb-object

Table 4.6: Semantic similarity computed by the magnitude of the inner product between the representations of compound nouns, adjective-noun, verb-object and subject-verb-object composition under addition using data sets from Mitchell and Lapata [3] and Grefenstette and Sadrzadeh [4]. Results given in Spearman correlation ρ calculated by averaging over participant correlation. $\uparrow, \downarrow, \bullet$ indicate the performance compared to model's baseline.

Addition $\oplus$	compound nouns	adjective-noun	verb-object	subject-verb-object
		Simple Semantic Space		
- raw	0.36077	0.17974	0.16809	0.07917
- probability	0.21503	0.18284	0.28605	0.03632
- pmi_a	0.40721	0.32530	0.27142	0.09379
- pmi_b	0.36488	0.23312	0.35151	0.20141
- pmi_c	0.37561	0.31964	0.24611	0.15315
		Simple Semantic Quantum Space		
- probability amplitude	0.39218 $\uparrow$	0.33889 $\uparrow$	0.37040 $\uparrow$	0.15391 $\uparrow$
- $npmi_a$	**0.41134** $\uparrow$	**0.35581** $\uparrow$	0.29824 $\uparrow$	0.19314 $\uparrow$
- $npmi_b$	0.30764 $\downarrow$	0.27678 $\uparrow$	**0.37964** $\uparrow$	0.19701 $\downarrow$
- $npmi_c$	0.39089 $\uparrow$	0.33133 $\uparrow$	0.25356 $\uparrow$	**0.20179** $\uparrow$
- $npmi_a$ phases	0.41076 $\uparrow$	0.35290 $\uparrow$	0.29784 $\uparrow$	0.19299 $\uparrow$
- $npmi_b$ phases	0.13381 $\downarrow$	0.09962 $\downarrow$	0.18731 $\downarrow$	0.09017 $\downarrow$
- $npmi_c$ phases	0.34003 $\downarrow$	0.27044 $\downarrow$	0.25741 $\uparrow$	0.16839 $\uparrow$
		Simple Topic Space		
- 50 topics	0.52060	0.42080	0.36657	0.15481
- 200 topics	0.57235	0.27311	0.34206	**0.27679**
		Simple Topic Quantum Space		
- 50 topics	0.57860 $\uparrow$	**0.42313** $\uparrow$	**0.41405** $\uparrow$	0.15701 $\uparrow$
- 200 topics	**0.60321** $\uparrow$	0.31530 $\uparrow$	0.38048 $\uparrow$	0.27058 $\downarrow$

the quantum-inspired $npmi_b$ and $npmi_c$ perform best. They outperform the classical baselines significantly. The quantum topic models derived from topic representations by Komninos and Manandhar [131] show improved correlation in similarity judgement tasks for compound nouns, adjective-nouns and verb-object composition over their classical representations. Other variants of the quantum semantic spaces $npmi_b$, $npmi_b$ phases and $npmi_c$ phases show more mixed results. They improve the correlation for some composition types, but decrease the performance for others. We note that judgements for verb-object and subject-verb-object compositions benefit from $npmi_c$ phases, but decrease with $npmi_b$ phases. In comparison to their real-valued baselines as well as their non-phase variants, only the $npmi_c$ phase model improves in verb-subject composition. Other variants do not change the performance or have a detrimental effect.

Table 4.7 shows the results of simple composition using pointwise multiplication $\odot$. Under multiplication the phases do not have an impact on the magnitudes within the composition, since

Table 4.7: Semantic similarity computed by the the magnitude of the inner product between the representations of compound nouns, adjective-noun, verb-object and subject-verb-object composition under multiplication using data sets from Mitchell and Lapata [3] and Grefenstette and Sadrzadeh [4]. Results given in Spearman correlation ρ calculated by averaging over participant correlation. $\uparrow, \downarrow, \bullet$ indicate the performance compared to model's direct baseline.

multiplication $\odot$	compound nouns	adjective-noun	verb-object	subject-verb-object
		Simple Semantic Space		
- raw	0.19044	0.21046	0.24326	-0.09030
- probability	0.12331	-0.00601	0.16559	-0.00890
- pmi_a	0.42101	0.39442	0.33030	0.28821
- pmi_b	0.11449	0.26109	0.31260	0.21638
- pmi_c	0.31189	0.31192	0.3382	0.10721
		Simple Semantic Quantum Space		
- probability amplitude	0.17910 $\uparrow$	0.04940 $\uparrow$	0.13845 $\downarrow$	-0.03053 $\uparrow$
- $npmi_a$	**0.44925** $\uparrow$	**0.36386** $\downarrow$	0.31745 $\downarrow$	0.30894 $\uparrow$
- $npmi_b$	0.12294 $\uparrow$	0.26838 $\uparrow$	0.31402 $\downarrow$	0.25201 $\uparrow$
- $npmi_c$	0.36540 $\uparrow$	0.32566 $\uparrow$	**0.35660** $\uparrow$	0.12088 $\uparrow$
- $npmi_a$ phases	0.44254 $\uparrow$	0.36115 $\downarrow$	0.31173 $\downarrow$	**0.30737** $\uparrow$
- $npmi_b$ phases	0.15310 $\uparrow$	0.15356 $\downarrow$	0.25943 $\downarrow$	0.10894 $\downarrow$
- $npmi_c$ phases	0.33706 $\uparrow$	0.30690 $\downarrow$	0.30242 $\downarrow$	0.09078 $\downarrow$
		Simple Topic Space		
- 50 topics	0.51571	0.23729	0.16304	0.02042
- 200 topics	0.36820	-0.01494	0.25912	0.03226
		Simple Topic Quantum Space		
- 50 topics	**0.52973** $\uparrow$	**0.24142** $\uparrow$	0.17898 $\uparrow$	**0.03236** $\uparrow$
- 200 topics	0.40309 $\uparrow$	-0.01800 $\downarrow$	**0.26000** $\downarrow$	0.01534 $\downarrow$

the relative phases are added and the amplitudes are multiplied. However, the phase of the composition representation is still important for the similarity judgments, because it computes the inner product.

Generally, we note that the quantum embeddings also outperform the classical baselines in all composition types under multiplication. For compound nouns, adjective-noun and subject-verb-object composition, the quantum variants $npmi_a$ and $npmi_a$ phases perform best. They outperform their baselines significantly. For verb-object and subject-verb-object compositions $npmi_b$ and $npmi_c$ perform best. They outperform their baselines significantly as well. Under multiplication the Quantum Topic Spaces outperform the Simple Semantic Spaces consistently. Comparing the phase variants to their equal-phase models, we observe that manually set phases have either no effect or a detrimental effect on the performance.

Overall, the best results in compositional similarity of both functions are achieved by the

quantum embedding with $k = 200$ topics under addition. These are followed by multiplication in models $npmi_a$ and $npmi_a$ phase spaces for compound noun, adjective-noun and for subject-verb-object. For verb-object composition the second best is $npmi_b$ under addition. Comparing the two composition functions the best results under multiplication composition outperform the best results under addition. This is also true for the baselines. The Simple Topic Space and the Simple Topic Quantum Space with 50 and 200 topics perform in all compositions significantly better under addition as composition. For the lexical similarity evaluation the $npmi_a$ and $npmi_a$ phases achieve the best results.

We found that quantum-inspired representations improve the performance in similarity judgments of words, compound nouns, adjective-noun, verb-object and subject-verb-object compositions. We found that the probability amplitude baselines outperforms some *pmi*-based real-valued baselines in compositions.

We also found that the performance improvements between the real-valued topic model and the quantum-inspired topic model are not as significant as those observed in the Simple Semantic (Quantum) Spaces.

We observe that manually chosen phases have no effect or have a detrimental effect on the performance if compared with both the real-valued baseline and their non-phase quantum representation. The effects of interference depend on the composition type.

4.6 Discussion

4.6.1 Neural Language Modelling

In section 4.3 we present complex-valued LSTM and embeddings for language modelling.

We found that exchanging the recurrence and embedding matrix with complex-valued version does not improve perplexity scores. In fact, the performance is significantly worse, because optimisation towards a local minimum on the complex domain is more difficult. The explicit encoding of positional information as phase or slot embeddings also performs worse than the baseline. This may be explainable by the fact that the real-valued baseline already extracts positional information, because LSTMs are order-aware. Frequency-based models which apply a transformation to the real-valued embeddings outperform the baseline. This indicates that frequency information which are computed over the entire input sequence is beneficial to prime the recurrence towards correct output sequences.

We found that larger language models become unstable in their training process. Changing the activation functions and adding gradient norm clipping improved stability, but did not solve the problem.

We are limited by the small size of the chosen corpus and the limited training steps we used in for our experiment. The Penn-Tree-Bank corpus is comparatively small. More recent literature

found that language models tend to improve performance on vast and diverse corpora. They also require significantly more training steps than we assumed. Furthermore, the significantly worse results compared to results reported in existing literature indicate that hyperparameter tuning is exceptionally important for language modelling. Our intention was to avoid additional measures (e.g. batch normalisation) or excessive hyperparameter tuning to receive a fair comparison.

Future work should include the change to larger corpora and more training steps. Examples are to wiki-text corpora [132]. The activation function should be changed and stabilising methods should be introduced. Weight or batch normalisation are necessary to train larger networks in the complex setting.

4.6.2 Memory Networks

In Section 4.4 we present a simplified memory network architecture for a comparison between real-valued and complex-valued architectures. We tested it in different sizes on the bAbI QA data set using various activation functions.

We found that over the entire data set the complex-valued neural networks outperform the real networks more often. We attribute this to the increased expressibility of complex-valued neural networks and embeddings. We acknowledge that the performance improvements are small, but complex-valued neural networks tend to require fewer parameters for similar or slightly better performance.

We observe that complex-valued neural networks perform well in logic and reasoning tasks. They perform less well in short dependencies and co-referencing. We conclude that the increased expressibility allows the network to approximate logic operations more efficiently. This is also supported by the fact that complex-valued neurons are known to be superior in the approximation of logic functions (e.g the XOR function) [22, 120, 110]. The required information are available in the data itself which the memory network needs to extract from natural language. Even without explicit incentives, the model learns to encode logic into the word embeddings. A decision for or against using complex numbers should be based on the task.

The memory network embeds real-valued data into the complex plane. However, it is unclear what type of information is encoded to improve performance in these tasks. Previous work uses similar principles to exploit learning links for knowledge graph completion [101, 26, 133, 102]. We assume that complex-valued embeddings contain features to provide the network with the ability to learn logic functions in a probabilistic way.

Our findings are limited by the data set choice and architecture. The bAbI data set consists of simple toy questions. More complex questions will most likely require more elaborate architectures like periodic memory networks. However, we chose this data set as a benchmark task to show the validity of the complex-valued approach to question answering.

Future work should include experiments on more complex questions.

4.6.3 Quantum-Inspired Models of Language

In Section 4.5 we have described a quantum-inspired framework for languages. We found three underlying (mathematical) differences between existing Simple Semantic Spaces and Quantum-inspired Semantic Spaces:

1. Complex-valued $\mathbb{C}$ representations instead of real-valued representations $\mathbb{R}$
2. Probability: The squared magnitude (probability amplitudes) have to sum to one $\sum_{j=0}^{n} |z_j|^2$
3. Geometrical: The bases e need to be pairwise orthogonal $\langle e_j | e_i \rangle = 0$

Together they define a semantic space as a complex-valued Hilbert space. We conclude that Quantum-inspired semantic spaces are a useful extension. The introduced constraints allow straightforward interpretation of axioms and postulates.

In our related experiments we found that quantum-inspired representations improve the performance in similarity judgments of words, compound nouns, adjective-noun, verb-object and subject-verb-object compositions. The reason for this are scaling effects as a result of applying the probability amplitude constraint. This is supported by the observation that the probability amplitude baselines already outperform *pmi*-based real-valued baselines in compositions. Another supporting finding is that the performance improvements between the real-valued topic model and the quantum-inspired topic model are not as significant as those observed in the Simple Semantic (Quantum) Spaces. The scaling that leads to the improved correlations for Simple Semantic Quantum Spaces has already been implemented in the training of topic models.

In our framework and experiments we show that interference is a powerful principle to model interaction between two representations. We found that manually chosen phases have no effect or are more likely have a detrimental effect on the performance. This becomes clearer if we compare the models that include phases with both the real-valued baseline and their non-phase quantum representation. Modelling interference manually is difficult. If the phases are chosen incorrectly, the interference will decrease the overall performance. It then becomes another source of noise. Hence, particular care is required when designing interference. Since it is difficult to identify tasks that naturally benefit from an interference pattern, we suggest adapting loss functions to explicitly take phases into account.

We also observe that the effects of interference depend on the composition type and the level of comparison. We observe that *npmi* phase models have positive effects on word comparisons and verb-object phrases. This suggests that it is necessary to prepare semantic representations to be used in compositions. Our framework suggests to use unitary matrices to evolve semantic quantum states before they are used in word compositions.

The limitations of our experiments are similar to classical distributional models. Firstly, we assume that the distributional hypothesis is true and the meaning of a word is indeed given by its context. Our composition results depend on the quality of statistics and the topic embeddings

computed. Both heavily depend on the size of the corpus. However, we have used a large concatenated corpus of Gigaword and Wikipedia. Moreover, we have manually selected features to represent word context (counts, probability, *pmi* and *npmi*). While these have shown to be good predictors, they are not fine tuned for the tasks. Levy et al. [18] suggests, however, that existing neural approaches learn similar features. They learn to apply factorisation methods on matrices containing co-occurrence counts or similar metrics.

This work shows that quantum-inspired representations of language are an interesting topic, but require more care to create productive interference. Future work on these models should include learned phases which are explicitly promoted in the loss. We found manually defined phases and completely free phases to be detrimental to the overall performance. In this thesis we focus mainly on the first constraint which replaces real numbers with complex numbers. This increases the model's expressibility. Existing literature considers Neural Language models that learn quantum-inspired semantic representations directly while satisfying one or more of the presented constraints. We suggest building on our framework and derive representations in the complex plane using neural networks. Layers should have unitary weight matrices, apply weight normalisation or batch normalisation between the layers in order to enforce the probability amplitude constraints and improve the stability of the learning process. Measurement and evolution operators should be learned in tandem for different compositions in the same semantic space.

We showed that we can use complex-valued neural networks to learn complex-valued representations of words. Quantum-inspired models of language form a subclass of complex-valued representations. They are further constrained due to their probabilistic and geometrical interpretation. However, a problem of learning word embeddings with complex-valued neural networks is that they are unstable. We found generalisations of existing methods (i.e. gradient clipping and norming) to stabilise the training to be beneficial for complex-valued neural networks. We also found that the constraints of quantum theory may improve the performance of word embeddings. We may be able to combine these findings to improve the training stability and the performance of word embeddings. We can achieve this by moving towards a neural network design that respects the quantum-inspired framework presented in Section 4.5:

- Apply an adapted Batch Normalisation to force each embedding to be quantum state.
- Apply unitary constraints onto weight matrices of layers. A fully-connected layer or perceptron consequentially implements quantum evolution with a non-linearity. Without the non-linearity we do not need to another Batch Normalisation to normalise the output to be a quantum state, since unitary multiplication conserves the norm.
- Apply gradient normalisation and clipping to further stabilise the training process.

In addition, quantum theoretic postulates within neural networks will stabilise the network in consideration of the vanishing gradient problem.

4.7 Conclusion

In this chapter we investigate the use of complex-valued neural networks for representation learning of written natural language.

We found that frequency-based language learning outperforms real-valued baselines. Positional or freely learnable embeddings do not improve perplexity scores. Complex-valued embeddings support question answering tasks that require logic and reasoning, since complex-valued neurons are superior in their expressibility of logic functions. We also found quantum-inspired Semantic Spaces based on statistic outperform classical Semantic Spaces in similarity tasks. Phase models can improve performance in similarity tasks, but interference should be explicitly modelled and learned by the model.

Our contributions are embedding methods for words to receive position and frequency-based word representations trainable using language models and usable in down-stream NLP tasks. We showed complex-valued memory networks used in QA which perform better or similarly in the majority of bAbI tasks. We generalised a complex and quantum-inspired model of language. We experimentally showed the use of quantum-inspired Semantic Spaces based on *npmi*.

In answer to our second research questions: 'Does encoding additional positional or frequency information in imaginary parts or phases improve the performance in sequence-based language processing tasks?' Similar to complex-valued classification tasks, learning complex-valued embeddings without explicitly adding further information does not improve the performance in language modelling. Positional information also do not improve the performance of order-aware models. Adding frequency information, however, improves the perplexity scores. We also achieved a similar or slightly improved performance with fewer parameters for Question answering even without integrating frequencies. This is mainly the case for tasks which require logic and reasoning due to the increased expressibility of complex-valued embeddings.

To answer our third research question: 'What advantages and differences are there in using quantum-inspired models of language compared to non-quantum complex-valued models of language?' Quantum-inspired models of language enforce additional constraints on the representations. Literature suggests that the use of related approaches (such as batch norms, weight norms or gradient norms) improve the training stability of complex-valued neural networks. We can confirm this for complex-valued neural networks and representations across our experiments. We can also report improved performance in similarity judgements tasks, due to enforcement of the constraints. However, interference is difficult to model manually and should be automatically learned. It can be promoted in the loss function and model design. Our Simple Quantum Semantic Spaces achieve slightly improved similarity judgments for individual words and word compositions. We conclude that quantum-inspired models of language are a valuable approach model language.

In Chapter 3 we advised applying complex-valued neural networks on transformed or embed-

ded data. Alternatively, the model can be encouraged to use the increased expressibility with an adapted training objective. In this chapter we have attempted transformations and embeddings, but ignored the possibility of an altered loss function. We found that once real-valued data is embedded or transformed into the complex plane, some tasks (i.e. logic-based bAbI tasks or language modelling) do not require the changed objective functions. Instead they require stabilisation of the learning process.

Our findings in this chapter motivate the next chapters in different ways. The presented Quantum-inspired framework of language (Section 4.5.7) facilitates both geometrical and probabilistic interpretations. The constraints improve the stability of the training process in complex-valued neural networks. Hence, we use gradient norming and gradient clipping in this and future chapters to stabilise the training. Our findings also suggest that frequency-based representations should further be investigated (presented in Section 4.3 and 4.4), since they support similar performance by models with fewer parameters or improved performance by similar-sized models. In the next chapter we use transformations and frequency representations inside neural networks to process signal data rather than real-valued textual data.

CHAPTER 5

Speech Recognition with Complex-Valued Neural Networks

In this chapter we use frequency information to weigh representation outside of the frequency domain by designing a spectral approach to Automatic Speech Recognition (ASR). We also develop a novel dataset for phonetic transcription in this chapter called Babble.

We use the generalised attention to derive two novel self-attention mechanisms. Further, we use these attention mechanisms to create the Spectral Transformer architecture. Inspired by previous work it requires neither recurrence nor convolutions, simplifies preprocessing for speech recognition and compromises between time and frequency resolution.

Due to their ability to represent frequencies, which is explained in Chapter 1, complex numbers lend themselves very well to sound and speech processing. Chapter 2 presented the Fourier transformation, Convolution Theorem and a generalised attention framework. We apply the Fourier transformation and the Convolution Theorem to create a novel architecture that learns in the frequency domain. In this chapter we instantiate spectral attention and spectral scaled attention from our attention framework. Chapter 3 showed us that real-valued models show an upper performance bound for real-valued classification. However, the findings from Chapter 4 showed that transforming real-valued data into frequency-based representations may improve performance and may allow smaller model size.

5.1 Introduction

Sequence Transduction or sequence-to-sequence tasks involve converting input sequences x of length n to output sequences y of length m. Like many other tasks, Sequence Transduction has

benefited from developments in DL. Modern accomplishments in Machine Translation, Signal Transcription or Image Generation tasks were made possible with Neural Sequence Transducers. However, Automatic Speech Recognition (ASR) still posses a number of challenges:

- Varying types of target labels used by models make them incomparable. Target labels may be acoustic like phonemes, symbolic like characters or graphemes, or semantic like words or word pieces.
- Multiple correct transcriptions of the same input (audio signal)
- Differing source and target modality
- Non-standard speakers
- Multiple speakers
- Noisy environments or unclear speech
- Long dependencies within very long inputs
- Inputs and outputs with drastically differing lengths

In the past these challenges have been approached with varying success by combining models trained towards new objectives or by extensive pre- and post-processing. Our approach is to adapt the attention mechanism and the transformer architecture to the complex plane. We overcome the alignment problem with a complex-valued self-attention mechanisms using frequency-based representations. Signals and other long sequences can be taken into account with an approach that implicitly learns a reduced frequency representation.

5.2 Related Literature

In this review we focus on Automatic Speech Recognition. We present current and frequent solutions to problems in speech-related processing tasks. Each challenge was approached with a number of changes to the neural architecture, the pre- and post-processing and the objective. Architectures used can be categorised into two main groups: the encoder-classifier and the encoder-decoder architecture. One of the most important alterations of Sequence Transduction architectures is the extension with an attention mechanism.

5.2.1 Speech Processing

Dissimilar source and target modalities require implicitly or explicitly learned acoustic and language features together. While the acoustic features are directly extracted from the audio input signal, features over the written language need to be extracted from the targets transcription.

Acoustic ASR transducers have been extended with explicit language models [134, 135, 136]. For this purpose a prediction network embeds the input elements into a dense representation and attempts to model the acoustic input elements based on the previous textual outputs. Training the language and acoustic features in tandem allows the model to benefit from word or character predictions [135, 137, 138]. Other models only implicitly include a language model (or can be extended with one) [139, 140]. Another solutions is to learn acoustic word embeddings which are not embedded in a semantic space, but in an acoustic space [66]. Acoustic embeddings of similar sounding words are close together.

Speech transcription of non-standard speakers requires learning closer to the acoustic signal. This can be achieved with phonetic or grapheme-based approaches rather than word or word piece approaches. Non-standard speakers include children or infants, non-native speakers, speakers with strong accents or speakers with dialects [141, 99, 100, 142]. Training examples to learn necessary features are often significantly under-represented in existing data sets. A solution to this problem is to pre-train models on large standard-speaker data set and fine tune on specifically curated data sets. Most existing approaches also use characters, graphemes or phonemes as target labels.

Multiple speakers require additional, mostly frequency-based, speaker separation. DL models for speaker separation need to overcome arbitrary permutations of an unknown number of speakers in a long input signal. The output for this related task is a mask over the original input signal, similar to a segmentation mask. The mask extracts various sources. Recently, successful DL approaches like Deep Clustering or Attractor Networks have been proposed [143]. An attractor network creates masks by estimating attractor points using an embedding layer from a bidirectional multi-cell LSTM. The input signal is represented by the log spectral amplitudes of the Short-Time Discrete Fourier Transformation (STDFT). In the training phase ideal masks are used as an additional input to the network. In the inference stage these are replaced with a K-means standard mask.

Noisy environments or unclear speech are enhanced to sharpen speech and denoise the signal [144, 25, 145, 146]. Speech enhancement is a separate task to speech recognition with the goal to increase the speech intelligibility. It may include the algorithmic removal of noise, restoration or normalisation of the input signal. Most neural networks are trained to optimise MSE (Equation 2.54) between a clean signal and corrupted or directly use Short-Time Objective Intelligibility (STOI) [147] as a target evaluation metric. Deep learning architectures proposed for this task include MLPs with heavy preprocessing [144, 145] or fully convolutional networks (FCN) [146]. Complex-valued MLPs may be a useful addition to this problem.

For a comparison of speech recognition models it is necessary to choose a combination of metrics. Accuracy (*acc*) and label error rates (*LER*) express the quality of transcriptions [148]. Label error rates are defined by the edit distance (number of deletions, inserts and substitutions) in relation the sequence length N (Equation 5.1) while accuracy is defined as the ratio between

correctly predicted (True Positives and True Negatives) and predicted labels (True Positives, True Negatives, False Positives, False Negatives) (Equation 5.2). Another metric often used is the top-K accuracy which considers a label to be correctly predicted if it is one of the k most likely labels returned by the model (Equation 5.3).

$$LER = \frac{\text{\# edits}}{\text{\# length of target sequence}} = \frac{EDIT(y, \tilde{y})}{\text{N}} \tag{5.1}$$

$$acc = \frac{\text{\# correct predictions}}{\text{\# total predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

$$acc_{top-k} = \frac{\text{\# correct predictions within k most likely classes}}{\text{\# total predictions}} \tag{5.3}$$

To be able to process the very long inputs, a preprocessing step with significant dimensionality reduction is necessary, alternatively, the model requires learning directly from the time signal. Popular frequency-based solutions use Short-Time Discrete Fourier Transformation (STDFT). STDFT applies a window function (e.g. Hamming function) and computes the Discrete Fourier Transformation (DFT) on the windowed snippet. Often frequencies with low amplitude are simply discarded. Alternatives may rely on the Hartley Transform [149] or Discrete Cosine Transformation (DCT) [150]. They return real-valued representation instead of complex transformations. In both cases this achieves a dimensionality reduction while compromising between frequency and time resolution. This principle also underlies Mel-Frequency Cepstrum Coefficient (MFCC) representations which have been frequently used for Speech Processing. MFCC representations are computed with the following steps [151]:

1. Windowing of time signals into a sequence of (overlapping) frames
2. Apply Discrete Fourier Transformation to receive complex-valued frequency representation of the window (STDFT)
3. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
4. Compute logs of the powers at each of the mel frequencies.
5. Compute discrete cosine transform of the list of mel log powers, as if it were a signal.
6. MFCC fingerprint consisting of the amplitudes of the resulting spectrum.

MFCC and other preprocessing use fixed window functions that are equivalent to manually-defined feature extraction. In end-to-end approaches, however, convolutions can implicitly learn to extract features from the original time signal or a preprocessed representation [152, 31]. To increase the receptive field, convolutions are replaced with dilated convolutions [152].

Inputs and outputs with drastically differing lengths make it necessary to learn alignments between the input and output sequence. To learn these alignments the model requires time positional information. We lose these time relevant signals when moving a signal into the frequency domain. Hence, a compromise between time and frequency resolution needs to be found. This can be achieved with the principle of STDFT and MFCC described above. Wavelets naturally offer this compromise between time and frequency resolution [153, 49]. Another approach uses positional embeddings [154, 155]. They usually extend model architectures that by design do not take order into account, due to order invariant operations. Combining positional embeddings with frequency representations can give us the best possible frequency-time resolution.

Switching from a cross entropy objective to a Connectionist Temporal Classification (CTC) criterion [148, 156] encourages alignment learning. In contrast to CTC type training (Equation 5.4), cross entropy considers each element of the output sequence independently and assumes that the input is already segmented according to the output. CTC attempts to implicitly learn the alignment between an input $x \in L^M$ of length M and a shorter output sequence $y \in L^T$ of a (symbol) vocabulary L of length T by taking unsegmented inputs into account. The output is a conditional probability distribution over the set of all possible transcription sequences. It extends the vocabulary with a $\langle blank \rangle$ symbol $L \cup \{\langle blank \rangle\}$. It also adds a reduction step which removes multiple symbols and blank symbols in the model's output. This reduction step is formally a mapping $\mathcal{B} : (L \cup \{\langle blank \rangle\})^{\geq T} \rightarrow L^T$ for a model output sequence $\tilde{y}$.

$$\begin{aligned} p(\tilde{y}|x) &= \sum_{\pi \in \mathcal{B}^{-1}(\tilde{y})} p(\pi|x), \\ p(\pi|x) &= \Pi_{t=1}^{T} \tilde{y}_t^k \end{aligned} \tag{5.4}$$

with an individual label $\pi \in (L \cup \{\langle blank \rangle\})$ from the extended vocabulary. The probability of observing label k at time step t is given by $\tilde{y}_t^k$. The CTC objective is a maximum log likelihood estimation which maximises the log probabilities of the target labels. Equation 5.5 is to be minimised.

$$J_\theta = -\sum_{\forall (x,y)} \ln(p(y|x)) \tag{5.5}$$

A further developed variant of the CTC criterion is the Auto Segmentation (ASG) criterion. ASG refrains from adding a blank symbol and does not use normalised scores at nodes and transitions. It does not use a global normalisation factor, but it relies on a duration model as transition scores [157].

Multiple correct transcriptions of speech necessitate scoring potential outputs. Usually the decision for an output label π is made by selecting the highest probability for each step in an output sequence of length T:

Table 5.1: Types of attention.

Query q		Keys k		Values v
input sequence or states or outputs $(n \times k)$	=	input sequence or states or outputs $(n \times k)$	=	input sequence or states or outputs $(n \times k)$
current input step $(1 \times k)$	$\neq$	input sequence or states or outputs $(n \times k)$	=	input sequence or states or outputs $(n \times k)$
hidden state, single fixed vector task or encoder $(1 \times k)$	$\neq$	input sequence or states or outputs $(n \times k)$	=	input sequence or states or outputs $(n \times k)$
current input step $(1 \times k)$	$\neq$	encoder output $(m \times k)$	$\neq$	encoder $(m \times k)$

$$\pi_t = argmax(p(\tilde{y}_t|x)) \tag{5.6}$$

The last layer may output a distribution $p(\tilde{y}|x)$ over the labels $l \in L$, but does not consider the predicted sequence. Beam search is applied as part of decoders to find the best possible transcription sequence [158]. A beam search with k beams keeps track of the k most likely outputs. With every output step, the next label in each of the beams is generated. The most likely outputs become the output at time step t in each beam. After all beams have been terminated, the most likely sequence is selected.

The most important development in Sequence Transduction has been attention. Initially used in image processing, attention was introduced as a memory-access and selection mechanism in sequence tasks. Attention and self-attention have frequently been used to improve performance in sequence-to-sequence problems, not only in conjunction with memory networks [82, 84, 83, 139, 81, 67]. Attention weighs the elements of sequences or interim representations (values) by scoring the relationship between a query and keys representing the values. Keys and values can be hidden outputs, states, extracted features or other learned representations. Throughout existing literature there are many different types of attention and their use depends on the performed task. These mainly differ in their choice of query, keys and values (Table 5.1).

Attention allows a neural network to assign elements of a sequence varying task-specific importance. Often these weights are used to compute a single weighted average representation of a sequence. Some approaches, however, use attention to weigh each element of a sequence without averaging or as additional features in the next layer. In ASR models various types of attention have been successfully used [159, 140, 138, 160]. Further improvement of the attention mechanism has resulted in the development a new architecture type: the transformer network

[154]. We present further details of this architecture in later sections.

Modern ASR sequence transducers either use an encoder-classifier (Figure 5.1) or an encoder-decoder architectures (Figure 5.2). Encoder-classifiers encode input sequences into a sequence of dense representations of reduced dimensionality and classify each state according to its target labels. Instead of classifying the state directly, an encoder-decoder architecture decodes the encoded state (sequence) into a new target modality. ASR models are mostly trained using cross entropy, a CTC or ASG criterion targetting characters, phonemes or words [161].

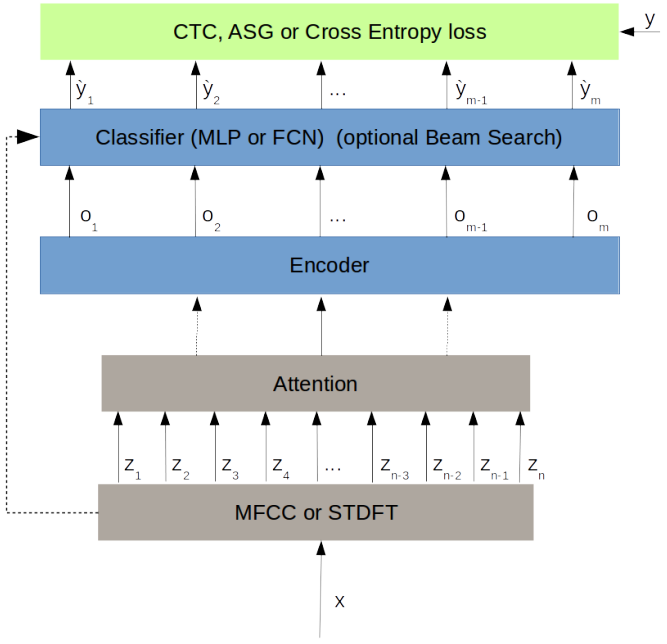

Figure 5.1: Encoder-classifier architecture for Automatic Speech Recognition. Attention and preprocessing blocks are optional.

Encoders are often multi-layer recurrent neural networks that use preceding convolutional and pooling layers for feature extraction [139, 136, 31]. Few architectures are able to learn directly from the waveform [152, 31]. Most architectures, however, preprocess the input signal to a frequency-based representation like MFCC or Mel-frequency filter banks [135, 162, 163]. Decoders usually consist of an optional attention mechanism and recurrent neural networks with a classifier [139, 162, 140, 164, 138]. Many models add optional convolutional networks and beam search. Decoders often rely on attention mechanisms to attend to the encoder's output sequence or state sequence [139, 140, 136, 137, 138, 164, 163]. They can also be extended with language models [140, 136, 138]. Existing literature found that bidirectional RNNs outperform unidirectional approaches in both encoders and decoders [135, 139, 140, 136]. There is a growing number of less computationally intensive, fully convolutional variants. These convolutional

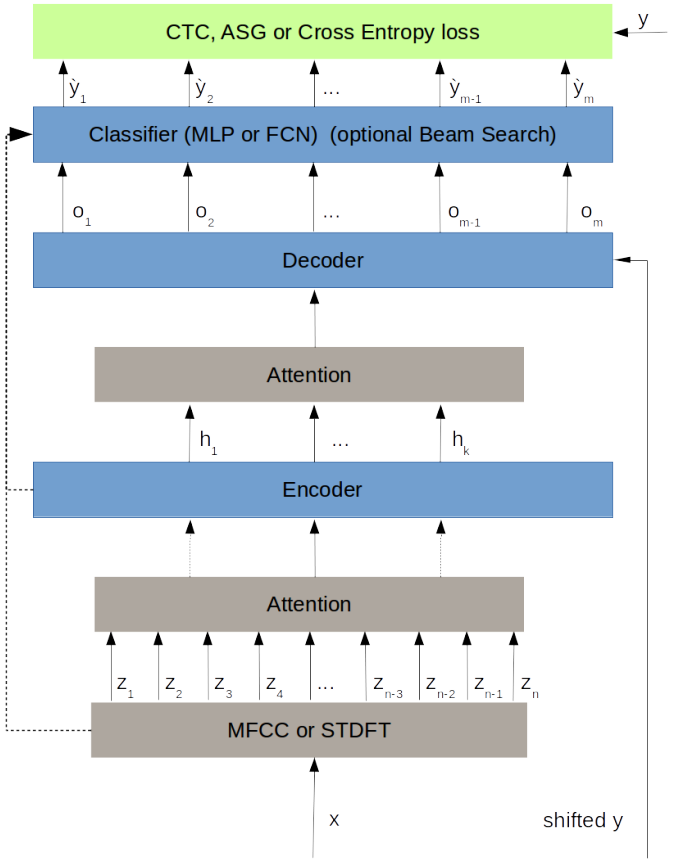

Figure 5.2: Encoder-decoder architecture for Automatic Speech Recognition. Attention and preprocessing blocks are optional.

approaches are mostly trained towards the ASG criterion and are able to learn directly from the time signal [165, 152, 155, 161, 166, 31].

5.3 Spectral Self-Attention

We use the definitions of complex-valued attention to derive two novel multi-head self-attentions specifically for speech processing tasks. They differ in the choice of queries Q, keys K and values V and in consequence what they return. Let $X \in \mathbb{R}^{n \times d}$ be the frequency representation consisting of n windows each of length d samples in the time domain. The original time signal is X and PPE describes a positional embedding.

Fully Spectral Self-Attention:

$$Q = K = V = \text{pooling}(\mathcal{F}(X) + iPPE)) \tag{5.7}$$

This type of attention returns complex representations, since the values V are themselves complex frequency bins.

Spectral-Scaled Self-Attention

$$Q = K = \mathcal{F}(X + PPE) \tag{5.8}$$

This type of attention returns real representations, since the values V are windowed time signals.

5.4 Spectral Transformers

Transformer networks completely replace recurrent neural networks and convolutions with multi-head attention. They are capable of attending to the entire sequence at once and thus have no need to store information in hidden states or other memories. Transformers do not provide a memory in the sense of other attention-based models, since they do not store any information. Information does not persist over multiple examples or batches. Instead the transformer focuses directly to the input or computed representations. The attention mechanism reduces the distance between input and output to a constant allowing the network to learn long-term dependencies which eases learning alignments. Figure 5.3 shows the original architecture presented by Vaswani et al. [154] which can be used for sequence-to-sequence tasks.

We propose to adapt this architecture to use the spectral self-attention by replacing the MHA block with a spectral MHA. This necessitates a few changes to the transformer. We also propose to use of the (Short-Time) Discrete Fourier Transformation (Equation 2.11) on the input to allow learning in the frequency domain. The **Convolution Theorem** simplifies the computation of a convolution to an element-wise multiplication $\odot$:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \odot \mathcal{F}(g) \tag{5.9}$$

and

$$\mathcal{F}(X * K) = \mathcal{F}(X) \odot \mathcal{F}(K) \tag{5.10}$$

Although, it requires padding of the kernel K to the same size of the input sequence X, this trick is being used to allow faster computation. Practically, this is used for large kernels. After padding, the DFT is applied to move an input into the frequency domain, before computing the element-wise multiplication, followed by the Inverse Discrete Fourier Transformation $\mathcal{F}^{-1}$ (IDFT) to transform it back into the time domain. If the entire learning process is moved into the complex plane, we do not need to apply padding or the IDFT for every example. After applying the DFT on the input, the model learns frequency kernels implicitly. This further reduces

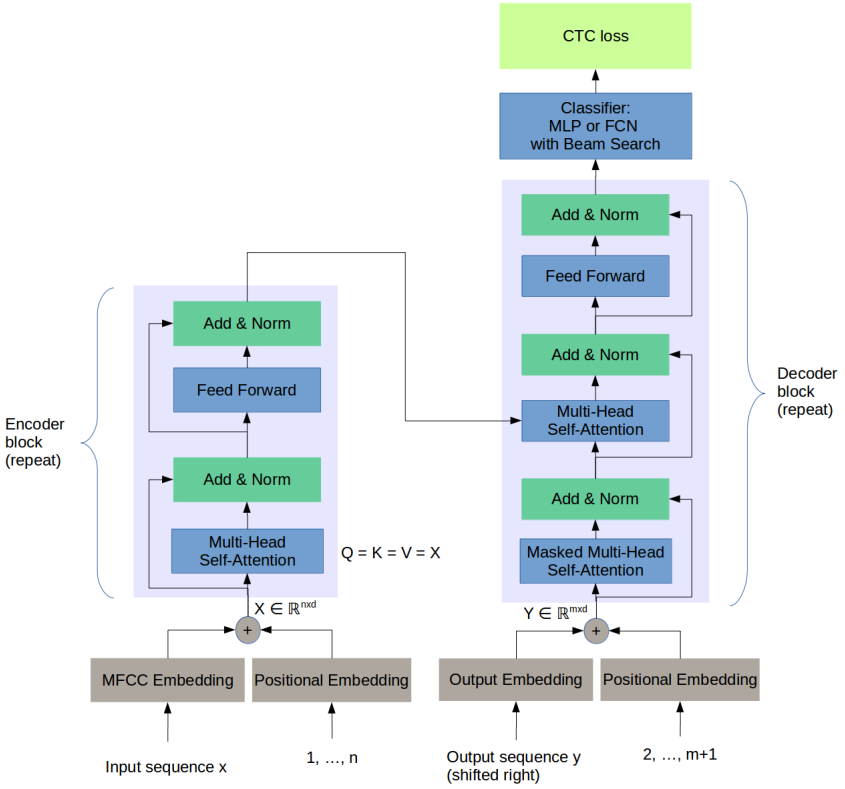

Figure 5.3: Real-valued transformer architecture for speech recognition.

the computational complexity. Complex-valued frequency kernels practically scale the magnitude and shift the phase. We are able to interpret a fully-connected layer as an unconstrained transformation. Pooling applied in the frequency domain extends the idea of frequency binning resulting from the transformation itself. The disadvantage of this approach is that we loose any temporal or positional information, due to the Fourier Transformation. A compromise can be found with the SDFT. Applying a window function first to create a sequence or frame and then computing the DFT on the windows allows the network to also have a relative order in the time domain.

For our complex-valued transformers we use the Convolution Theorem to allow the transformer to learn kernels within the frequency domain. We replace the original attention with our multi-head spectral self-attention defined above. Using spectral attention in the frequency domain enables a network to learn filters which select frequencies from a representation by rescaling amplitudes of the frequency bins. This procedure is often used to denoise various signals. Our preprocessing block consists of STDFT outputting complex-valued frames in the frequency

domain. We also use element-wise multiplication and pooling. This is used instead of input projection before the MHA. We propose a magnitude-based pooling over the frequency bins (dimension d) after the DFTs reduce the dimensionality of the individual frames. We test two pooling functions for this purpose: average magnitude pooling (Equation 2.28) and max magnitude pooling (Equation 2.29). Striding and pool size are hyper parameters.

Another adapted building block in transformers are positional embeddings. To improve performance we perform positional phase shifts. They are equivalent to shifting the spectral representation of a window based on its position within the signal. The **Positional Phase Embedding** (PPE) is given by:

$$\begin{aligned} PPE_{pos} &= \begin{bmatrix} p_1 & \dots & p_d \end{bmatrix}, \\ p_j &= e^{i \frac{pos}{10000^{2j/d}}} = \cos \frac{pos}{10000^{2j/d}} + i \sin \frac{pos}{10000^{2j/d}} \end{aligned} \tag{5.11}$$

We can now construct transformers in the frequency domain. We refer to a complex-valued transformer that uses STDFT, positional phase shift, pooling and element-wise multiplication as a Fully Spectral Transformer (FST). The FST also exchanges real-valued weights with complex-valued weights. The real-valued transformer that applies FT onto keys and queries only to computed spectral attention weights is referred to as Spectral-Scaled Transformer (SST). The weights in the SST stay real-valued. The FST is shown in Figure 5.4 and the SST is shown in Figure 5.5. It is important to note that the FST also requires classifiers and norms to be able to handle complex values.

To fully show the validity of this approach we suggest Speech Recognition (e.g. [19]) and Phonetic Transcription experiments.

5.5 Babble

We present a novel data set for phonetic transcription of infant vocalisations called **Babble**. Babble[1] was developed as a collaboration between the departments of Linguistics, Electronic Engineering, Music and Computer Science at the University of York. The database of utterances and babble was created to develop Machine Learning tools for Linguists and to investigate speech development in infants and toddlers. We hope to inspire new tools and approaches for both Linguists and Machine Learning experts working in Automatic Speech Recognition (ASR). Recent developments have resulted in methods that are capable of recognising subtle phonetic differences in speech of non-standard speakers [138, 166, 161, 149]. Further research in this particular aspect of ASR promises to improve results with accents and dialects.

Babble is a data set of 11 hours of audio recordings and over 160,000 phonetic annotations. The candidate's contribution is the statistical analysis of the collected and annotated data, creation

[1] Version 0.9.0

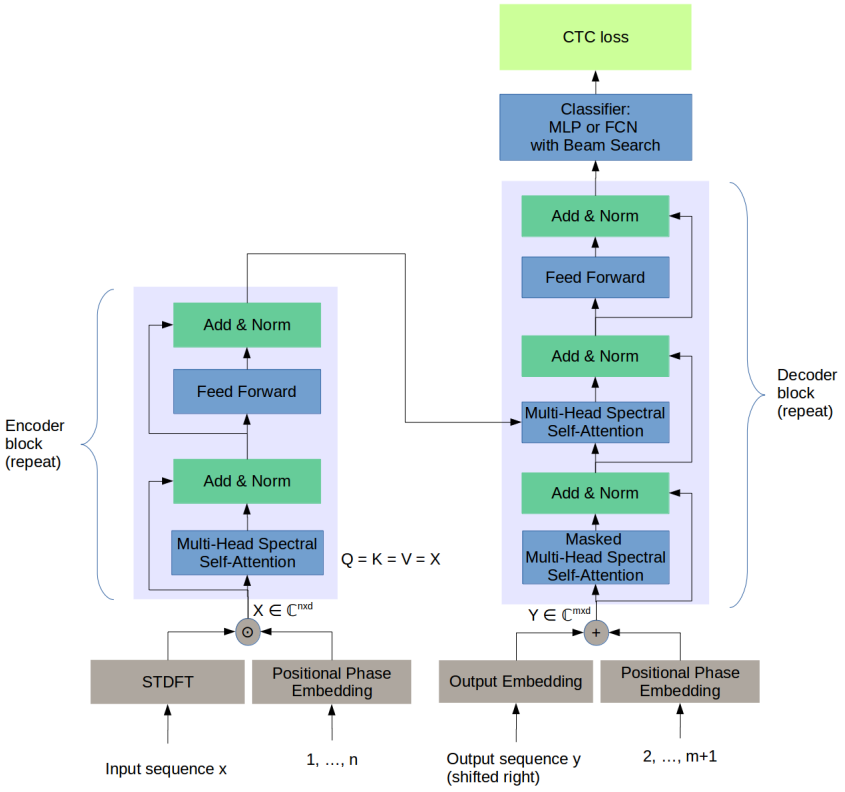

Figure 5.4: Fully Spectral Transformer architecture. Values V are frequency domain representations.

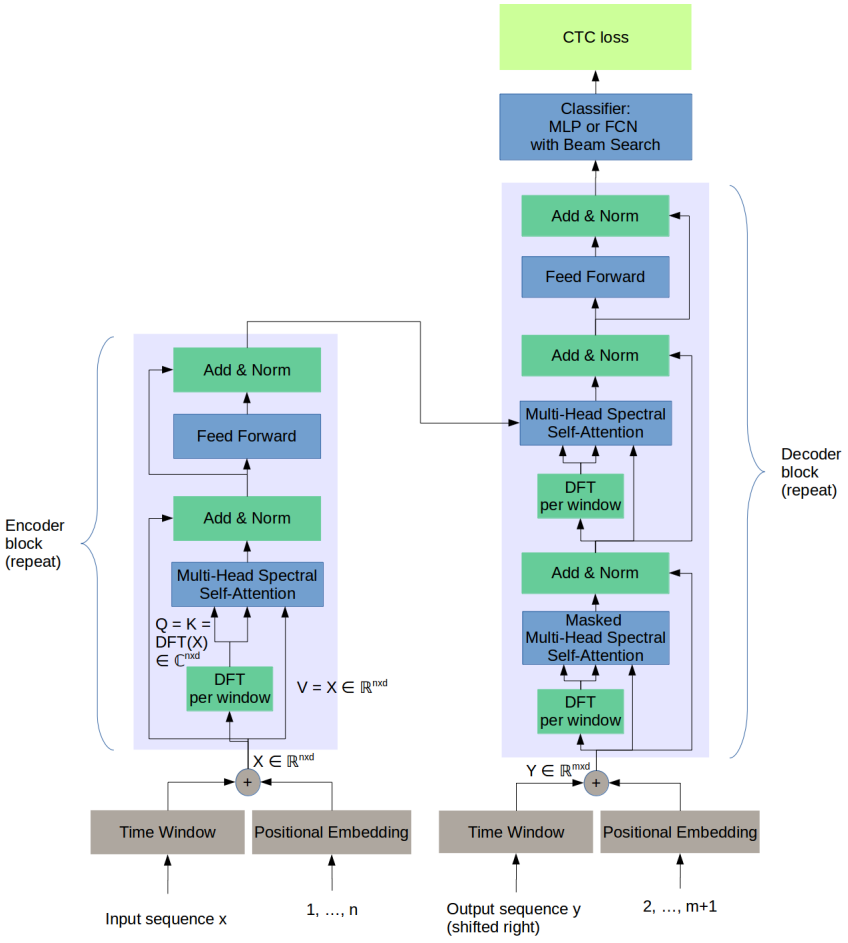

Figure 5.5: Spectral-Scaled Transformer architecture. Values V are time domain representations.

of data splits and baseline models.

Additional (technical) information about the dataset are provided in the Appendix A.

5.5.1 Data Collection

The data was recorded in 27 session with 59 child participants as part of a project lead by M. M. Vihman (Department of Language and Linguistic Science, University of York), R. A. DePaolis (Department of Communications Sciences and Disorders, James Madison University) and T. Keren-Portnoy (Department of Language and Linguistic Science, University of York). From this raw data relevant sequences of utterances were manually cut out and trimmed. Very distorted or noisy sequences were discarded for further processing. After cleaning we received approximately 30,572 audio sequences of varying length. Figure 5.6 shows the duration of all audio recordings in a histogram.

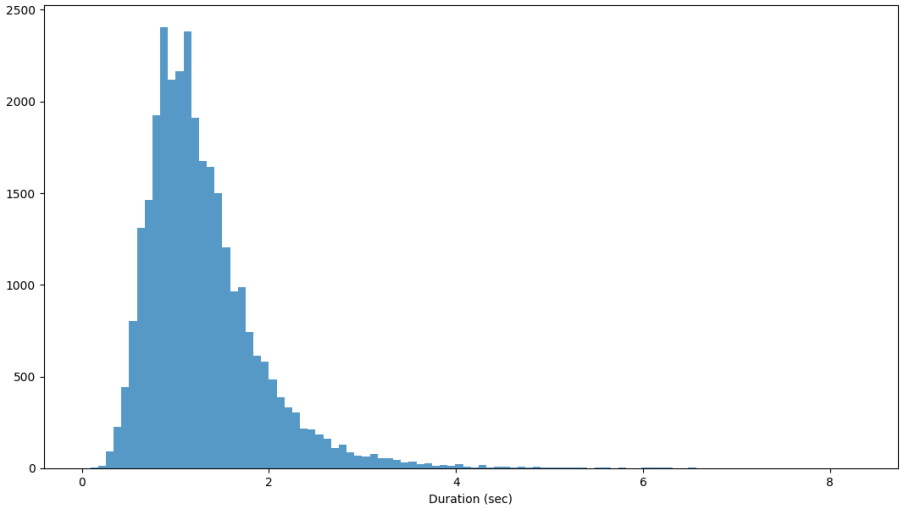

Figure 5.6: Durations of trimmed audio recordings.

All audio files contain monaural (single channel) sound stored as .wav files with a sampling rate of 48kHz. Overall there are 658 minutes of trimmed audio recordings. The shortest audio sub sequence is 0.008 seconds long (382 samples), the longest is 8.309 seconds long (398,854 samples).

5.5.2 International Phonetic Alphabet

The International Phonetic Alphabet (IPA) is a standardised system used to represent spoken language [167]. Each of the trimmed sequences were manually transcribed by 5 trained research assistants. Due to the nature of the collected data, not every IPA symbol is used or may be used

very infrequently. Table 1 presents all symbols and their counts of appearance. Figure 5.5.2 shows the symbol histogram across the data set.

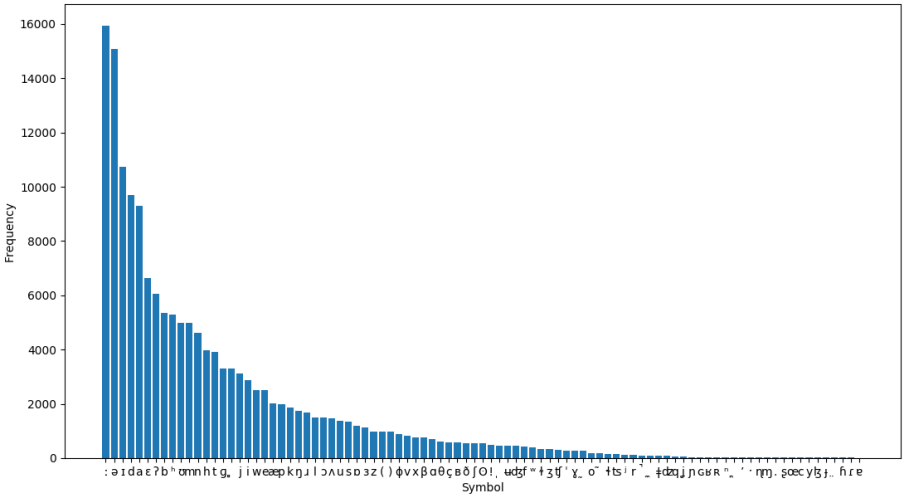

Figure 5.7: Symbol histogram of all used IPA symbols.

We extended the IPA with the following symbols to encode further information in the transcription:

- Spaces or underscores are used to annotate a break in speaking/babbling to improve human readability of the transcription. Systems do not have to predict them . They are filtered out when the data set is created.

- Round brackets or parenthesis are used for indistinguishable utterances (e.g. whispering). Our vocabulary has been extended with this symbol. Depending on the sub task these should be predicted by a model.

We provide a machine- and human-readable overview of the entire IPA (including unused symbols) with *ipa.csv*.

5.5.3 Tasks

We define seven sub tasks of varying difficulty. The first three tasks are transcription tasks (b1 to b3) which use all available IPA (*full_ipa*). The following three tasks (b4 to b6) are transcription tasks that exclude certain symbols from the transcription (*simplified_ipa*). The last sub task (b7) is a binary classification task which requires the system to decide if an utterance constitutes babble. A recording is considered as babble if it has at least a vowel in combination with a consonant. If an invalidating symbol or no such combination is in the recording it is marked as non-babble

1. Task b1: Transcription of child utterances with all available IPA symbols including unclear utterances (e.g. mumbled or not clearly identifiable) and noisy environments. Annotations within parentheses transcribe unclear utterances. Parentheses and all symbols are to be transcribed by the system. Column *full_ipa* contains the target annotations.

2. Task b2: Transcription of child utterances with all available IPA symbols including unclear utterances (e.g. mumbled or not clearly identifiable) and noisy environments. Parentheses marking unclear utterances are removed, however, the symbols within the parentheses are to be transcribed by the system. Column *full_ipa_no_brackets* contains the target annotations.

3. Task b3: Transcription of child utterances with all available IPA symbols without unclear annotations. Parentheses and their contents are removed. This sub task still includes noisy environments. Column *full_ipa_no_unclear* contains the target annotations.

4. Task b4: Transcription of child utterances with a simplified set of IPA symbols including unclear utterances (e.g. mumbled or not clearly identifiable) and noisy environments. Parentheses and symbols are to be transcribed by the system. The simplified set of IPA symbols excludes diacritics, suprasegmentals, tones, word accents and IPA symbols categorised as Other Symbols. Column *simplified_ipa* contains the target annotations.

5. Task b5: Transcription of child utterances with a simplified set of IPA symbols including unclear utterances (e.g. mumbled or not clearly identifiable) and noisy environments. Parentheses marking unclear utterances are removed, however, the symbols within the parentheses are to be transcribed by the system. The simplified set of IPA symbols excludes diacritics, suprasegmentals, tones, word accents and IPA symbols categorised as Other Symbols. The column *simplified_ipa_no_brackets* contains the target annotations.

6. Task b6: Transcription of child utterances with all available IPA symbols without unclear annotations. Parentheses and their contents are removed. This sub task still contains noisy environments that are to be ignored. The simplified set of IPA symbols excludes diacritics, suprasegmentals, tones, word accents and IPA symbols categorised as Other Symbols. The column *simplified_ipa_no_unclear* contains the target annotations.

7. Task b7: Binary classification of child utterances as babble or non-babble. The system needs to decide if a recording contains babble or not. A recording is marked as babble if it has at least a vowel (a) that is not invalidated by being unvoiced and (b) in combination with a consonant. Either in the order of consonant-vowel or in the opposite order. If an invalidating symbol or no such combination is in the recording it is marked as non-babble. The column *babble* contains the binary label.

5.5.4 Training, Validation and Test

For each sub task the data set was partitioned into train (80%), validation (10%) and test (10%) sets. The tasks uses fixed splits to allow a reliable comparison of models. Mixtures of data splits should not be used. We do not partition based on the participants. Data points of children that appear in the training set can also appear in the test and validation set. Individual samples in the training set do not appear in the test or validation set. Similarly the test and validation sets are completely distinct.

While many data sets partition based on audio files, Babble is partitioned based on the target symbols. Each distinct partition approximates the symbol distribution. Figure 5.5.2 shows Babble's symbol frequency. However, this overall distribution changes depending on the task. For example in task b3 parentheses and their contents are removed. Therefore, the training, validation and test sets are created to approximate the new symbol distribution according to the selected column. For the binary classification task the validation and test subsets have same number of positive and negative samples.

Another important aspect is the frequency of each individual target symbol. To be able provide sufficient training and testing examples for a each symbol, we have excluded all data points with target sequences containing an infrequent symbol. A symbol is infrequent, if it appears fewer than 10 times in the selected task column. As a consequence each task defines a subset view and data points used may be different.

5.5.5 Baselines

We report baselines of transformer networks in Table 5.2 in two different model sizes: small, and large. Transformer networks introduced by Vaswani et al. [154] are DL architectures based on multi-head-attention. They do not require convolutions or recurrent layers. Due to the attention mechanism they can learn patterns and dependencies in a very long input. In many sequence-to-sequence tasks (e.g. translation) they show state-of-art results [154, 85]. We normalised them to adapt for volume differences between audio signals. We then computed a Mel-Frequency Ceptrum Coefficient (MFCC) fingerprint of the audio input. While we provide the original (trimmed) audio recordings, we also provide program code for the preprocessing. For further information on the offline preprocessing and its parameters please refer the T2T code repository. We trained each model for $250,000$ training steps each with a batch size $b = 2,100,000$. We followed the same training routine for all models as described in Vaswani et al. [154] including warm-up phases. Table 5.3 shows the hyper parameters and the number of trainable parameters for each baseline model.

Table 5.2: Baseline accuracy on the seven sub tasks (b1 to b7).

Sub task	Transformer-Small	Transformer-Large
b1	0.2178	0.4224
b2	0.4201	0.4437
b3	0.4654	0.4622
b4	0.4696	0.4204
b5	0.3445	0.3739
b6	0.3861	0.3902
b7	0.6285	0.6285

Table 5.3: Overview of the baseline's hyper parameters.

	Transformer-Small	Transformer-Large
Attention-Heads	2	2
Hidden Size	256	384
Filter Size	1024	1536
Encoder Layers	4	6
Decoder Layers	2	4
Feed-Forward	CNN (9×9)	CNN (9×9)

5.6 Conclusion

In this chapter we introduced Babble, a new data set for the phonetic transcription of infant vocalisations to the Machine Learning community. We provide 30,500 audio files containing close to 11 hours of audio recordings of child speech and babble. We hope that this new task can improve tools for researchers and practitioners within Linguistics and also improve existing speech recognition approaches with added phonetic transcription.

The findings and contributions of this chapter are theoretical. Empirical proof for the validity of the approach is needed and is not contained in this chapter. We used the general framework of attention and Convolution Theorem (Chapter 2) to develop spectral attention, spectral transformers and methods to learn convolution kernels directly in frequency domain.

We found learning in the frequency domain provides theoretical advantages for speech recognition tasks. The computational complexity of convolutions is reduced, since the operation simplifies to an element-wise multiplication which does not require padding and inverse DFTs. Moreover, using spectral attention to scale frequency bins enables the network to select frequencies. This should increase the reliability in speech-related tasks. We have also found that transformers require a large number of training steps to achieve appropriate results. This is the

case in their real-valued and complex-valued settings.

The contributions in this chapter are the spectral attention mechanism, the spectral transformer architectures and methods to learn in the frequency domain like pooling and convolutions. The attention mechanism can be used in both complex-valued and real-valued architectures. It is a drop-in alternative to existing attention mechanisms. Speech Recognition and Phonetic Transcription experiments are needed to empirically show the value of spectral transformers and attention.

In response to our research question: 'Can we move the entire neural model into the frequency domain to improve speech transcription and simplify computation?' The answer is to this question is yes. The Convolution Theorem allows us to compute convolutions in the frequency domain as element-wise multiplications with a large (potentially padded) kernel. However, this is a trade-off between computational complexity and the number of parameters in the model. To address this problem we suggest frequency-based pooling. Spectral attention mechanisms provide a method of selecting frequencies and filter unused frequencies. The proposed methods of learning frequency representations or learning within the frequency domain can be considered to be part of Spectral Deep Learning.

CHAPTER 6

Conclusion

This thesis focuses on Natural Language Processing solved with complex-valued Deep Learning. It focuses on modelling compositional language from a semantic and phonetic perspective. Complex-valued neural networks and their training methods have long been proposed. Despite their predisposition for Speech Processing very few attempts have been made to apply them to speech and phonetic transcription. Even fewer attempts to process written language. This thesis attempts to close this gap in the literature and give a wide contextual overview of its application in the field of NLP.

6.1 Findings

We find that complex neural networks are a valuable addition to DL and NLP. Even though they are outperformed by real-valued networks in real-valued classification task, they are useful for when the data is transformed to the complex domain. This can be achieved with frequency transformations. They can also be used to learn entirely within the frequency domain which leads to improved computational complexity. Moreover, they can be used for semantic models of language. Frequency-based embeddings and quantum-inspired models with a probabilistic and geometrical interpretation allow their application to language modelling. Complex data representations can facilitate the introduced structure in order to solve a problem using fewer parameters. These smaller neural networks come at the cost of more computationally intensive training. This can be observed in language tasks involving long dependencies. Complex-valued transformations and embedding also allow us to encode additional information compared to real-

valued embeddings using extended dimensionality. However, we acknowledge the difficulty of initialisation and training. Training of complex-valued neural networks is unstable and requires various methods to improve stability. In our experiments batch normalisation and gradient clipping have demonstrated improvements to the training stability. However, enforcing constraints from quantum theory we can also stabilize the training process.

We return to our research questions from Chapter 1.2 and answer them with the findings from this thesis:

- **Does the use of complex-valued neural networks improve accuracy in classification tasks by increasing the model's expressibility?**

  We did not observe performance improvements in the classification of real-valued input data directly. When real-valued data is used as an input into a complex-valued MLP, the real and imaginary parts of the weights convergence together. The complex-valued logits of the last layer are simply scaling the output. These logits are then mapped to a probability distribution over the classes. A complex-valued MLP or CNN can, therefore, at best perform on-par with its real-valued counterpart on real-valued classification tasks. However, it is more likely to perform worse, due to its initialisation and approximation error. The extended expressibility does not allow the network to learn useful information if they are not explicitly encoded or encouraged. We suggest that embeddings or transformations of real-valued data into the complex domain are required. Complex-valued input data requires complex-valued neural networks, since the noise is distributed in the complex plane.

- **Does encoding additional positional or frequency information in imaginary parts or phases improve the performance in sequence-based language processing tasks?**

  Similar to complex-valued classification tasks, learning complex-valued embeddings without explicitly adding further information does not improve the performance in language modelling. Positional information do also not improve the performance of order-aware models. Integrating frequency information into embeddings, however, improves the perplexity scores. We also achieved a similar or slightly improved performance with fewer parameters for Question answering even without integrating frequencies. This is mainly the case for tasks which require logic and reasoning, due to their increased expressibility.

- **What advantages and differences are there in using quantum-inspired models of language compared to non-quantum complex-valued models of language?**

  Quantum-inspired models of language enforce additional constraints on the representations. Literature suggests that the use of related approaches (such as batch norms, weight norms or gradient norms) improve the training stability of complex-valued neural networks. We can confirm this for complex-valued neural networks and representations across our experiments. We can also report improved performance in similarity judgements tasks,

due to enforcement of the constraints. However, interference is difficult to model manually and should be automatically learned. It can be promoted in the loss function and model design. Our Simple Quantum Semantic Spaces achieve slightly improved similarity judgments for individual words and word compositions. We conclude that quantum-inspired models of language are a valuable approach model language.

- **Can we move the entire neural model into the frequency domain to improve speech transcription and simplify computation?**

  Yes, the Convolution Theorem allows us to compute convolutions in the frequency domain as element-wise multiplications with a large (potentially padded) kernel. However, this is a trade-off between computational complexity and the number of parameters in the model. To address this problem we suggest frequency-based pooling. Spectral attention mechanisms provide a way of selecting frequencies and filter unused frequencies. The proposed methods of learning frequency representations or learning within the frequency domain can be considered to be part of Spectral Deep Learning. The experimental results are pending.

In conclusion, complex numbers have advantages over real numbers in that they can reduce the number of parameters needed, allow frequency based representation and allow additional information to be encoded. However, they require particular care when designing the architecture e.g. choosing activation and loss functions to guarantee training stability. If it is known that there is no need to express additional information, real numbers may be preferentially used. Complex numbers are incredibly powerful tools, however, issues of stability mean that it can be difficult to design training which consistently performs in a reliable way. As improvements in normalisation and training are made, it is possible that the use of complex numbers in neural networks may become more mainstream.

6.2 Future Work

Our research has shown the validity of applying complex-valued neural networks to Natural Language tasks. Learning word embeddings and frequency representations of spoken or written language improves performance while reducing the parameters required. However, we found the training process to be unstable, due to various types of singularities. Future work should aim to improve complex-valued neural networks in general or find novel applications outlined below.

6.2.1 Improving Complex-valued Neural Networks

Future research should focus on stabilising the training process of complex-valued neural networks. This may be achieved by using novel optimisers, weights constraints, training methods or activation functions.

Optimisers developed specifically for complex-valued neural networks should be able to recover from breaking points, bad initialisations and singularities. Another approach would be to avoid singularities in the training process and the initialisations. This requires identifying different kinds of singularities, keeping a history of and the ability to discover valid optimisation paths. Novel optimisers would improve future and existing work on complex-valued neural networks. Furthermore, they would be directly applicable to real-valued neural networks. These analytic insights may inspire new ways of training real-valued neural networks.

In this work we used straight-forward generalisations of gradient clipping and weight initialisations. Gradient clipping improves the behaviour around singularities. Other useful gradient-based improvements may include gradient scaling and norming. Initialisations allow optimisers to find local minima easier. Normalisation-based methods may also be used to improve training. These include weight normalisation and batch normalisation. For example, normalising weight matrices to be unitary reduces the effects of vanishing gradients [111]. Additionally, they stabilise the training by preserving matrix norms across layers. Particularly, quantum-inspired constraints are interesting, since they support a geometrical and probabilistic interpretation of representations and matrices while implementing normalisation. However, generalisation of existing methods require more investigation to understand their consequences on the complex plane.

Another approach to improve stability is to change the architecture design. As we know holomorphic activation functions are not an option, due to Liouville's theorem. Hence, functions which include 'simpler' singularities need to be developed and investigated in-depth. This requires an experimental and analytic approach.

6.2.2 Applications

There are many applications that could benefit from complex-valued representations and hypotheses.

In this work we have investigated the fundamentals of Frequency Domain Learning. New applications for this use of complex-valued neural networks may be Image Processing or electronic Signal Processing. Further experiments are needed to determine the validity of the approach in these contexts.

Future research should also examine Embedding Learning. Complex-valued embeddings allow further information and complex interaction to be stored. Words are considered to be interacting semantic signals that are in a superposition of multiple meanings. In a quantum-inspired setting these are represented as quantum states. By learning phases we learn an interference pattern between two or more embeddings. Our frequency-based and quantum-inspired word embeddings show promising results. Future work should particularly focus on learning either frequency representations or learn within a quantum-inspired setting. Our Quantum-inspired framework of languages naturally supports the use of weight constraints, weight and batch norm-

alisation. Future research should focus on larger corpora and more complex question answering tasks.

For new applications, however, we may need novel loss functions $J : \mathbb{C} \to \mathbb{R}$ that promote learning of specific phases or imaginary parts.

Appendices

A Babble

This part of the appendix contains (technical) details to the Babble dataset presented in Section 5.5.

A.1 Symbol Overview

Table 1: Overview of IPA symbols and their usage in the Babble dataset.

ID	Hex-Code	Name	Category	Simple IPA?	Count
101	0x0070	voiceless bilabial plosive	consonants (pulm.)	Yes	1983
102	0x0062	voiced bilabial plosive	consonants (pulm.)	Yes	5343
103	0x0074	voiceless dental/alveolar plosive	consonants (pulm.)	Yes	3912
104	0x0064	voiced dental/alveolar plosive	consonants (pulm.)	Yes	9691
107	0x0063	voiceless palatal plosive	consonants (pulm.)	Yes	21
108	0x025F	voiced palatal plosive	consonants (pulm.)	Yes	18
109	0x006B	voiceless velar plosive	consonants (pulm.)	Yes	1873
110	0x0261	voiced velar plosive	consonants (pulm.)	Yes	3308
111	0x0071	voiceless uvular plosive	consonants (pulm.)	Yes	69
112	0x0262	voiced uvular plosive	consonants (pulm.)	Yes	37

ID	Hex-Code	Name	Category	Simple IPA?	Count
113	0x0294	glottal plosive	consonants (pulm.)	Yes	6059
114	0x006D	voiced bilabial nasal	consonants (pulm.)	Yes	4970
115	0x0271	voiced labiodental nasal	consonants (pulm.)	Yes	26
116	0x006E	voiced dental/alveolar nasal	consonants (pulm.)	Yes	4629
117	0x0273	voiced retroflex nasal	consonants (pulm.)	Yes	28
118	0x0272	voiced palatal nasal	consonants (pulm.)	Yes	38
119	0x014B	voiced velar nasal	consonants (pulm.)	Yes	1755
121	0x0299	voiced bilabial trill	consonants (pulm.)	Yes	567
122	0x0072	voiced dental/alveolar trill	consonants (pulm.)	Yes	112
123	0x0280	voiced uvular trill	consonants (pulm.)	Yes	36
124	0x027E	voiced dental/alveolar tap	consonants (pulm.)	Yes	14
126	0x0278	voiceless bilabial fricative	consonants (pulm.)	Yes	890
127	0x03B2	voiced bilabial fricative	consonants (pulm.)	Yes	749
128	0x0066	voiceless labiodental fricative	consonants (pulm.)	Yes	434
129	0x0076	voiced labiodental fricative	consonants (pulm.)	Yes	813
130	0x03B8	voiceless dental fricative	consonants (pulm.)	Yes	600
131	0x00F0	voiced dental fricative	consonants (pulm.)	Yes	554
132	0x0073	voiceless alveolar fricative	consonants (pulm.)	Yes	1333
133	0x007A	voiced alveolar fricative	consonants (pulm.)	Yes	970
134	0x0283	voiceless postalveolar fricative	consonants (pulm.)	Yes	548
135	0x0292	voiced postalveolar fricative	consonants (pulm.)	Yes	330
136	0x0282	voiceless retroflex fricative	consonants (pulm.)	Yes	23
138	0x00E7	voiceless palatal fricative	consonants (pulm.)	Yes	592
139	0x029D	voiced palatal fricative	consonants (pulm.)	Yes	44
140	0x0078	voiceless velar fricative	consonants (pulm.)	Yes	777
141	0x0263	voiced velar fricative	consonants (pulm.)	Yes	267
143	0x0281	voiced uvular fricative	consonants (pulm.)	Yes	37
146	0x0068	voiceless glottal fricative	consonants (pulm.)	Yes	3960
147	0x0266	voiced glottal fricative	consonants (pulm.)	Yes	14

ID	Hex-Code	Name	Category	Simple IPA?	Count
148	0x026C	voiceless dental/alveolar lateral fricative	consonants (pulm.)	Yes	161
149	0x026E	voiced dental/alveolar lateral fricative	consonants (pulm.)	Yes	19
151	0x0279	voiced dental/alveolar approximant	consonants (pulm.)	Yes	1682
153	0x006A	voiced palatal approximant	consonants (pulm.)	Yes	3119
155	0x006C	voiced dental/alveolar lateral approximant	consonants (pulm.)	Yes	1484
170	0x0077	voiced labial-velar approximant	other symbols	No	2500
176	0x0298	bilabial click	consonants (non-pulm.)	Yes	547
178	0x01C3	(post) alveolar click	consonants (non-pulm.)	Yes	472
179	0x01C2	palatoalveolar click	consonants (non-pulm.)	Yes	80
209	0x026B	velarized voiced dental/alveolar lateral approximant	consonants (pulm.)	Yes	335
211	0x02A6	voiceless dental/alveolar affricate	additional symbols	Yes	145
212	0x02A3	voiced alveolar affricate	additional symbols	Yes	80
213	0x02A7	voiceless postalveolar affricate	additional symbols	Yes	314
214	0x02A4	voiced postalveolar affricate	additional symbols	Yes	443
301	0x0069	close front unrounded vowel	vowels	Yes	2861
302	0x0065	close-mid front unrounded vowel	vowels	Yes	2491
303	0x025B	open-mid front unrounded vowel	vowels	Yes	6625
304	0x0061	open front unrounded vowel	vowels	Yes	9291
305	0x0251	open back unrounded vowel	vowels	Yes	714
306	0x0254	open-mid back rounded vowel	vowels	Yes	1481
307	0x006F	close-mid back rounded vowel	vowels	Yes	189
308	0x0075	close back rounded vowel	vowels	Yes	1386
309	0x0079	close front rounded vowel	vowels	Yes	20
311	0x0153	open-mid front rounded vowel	vowels	Yes	22
313	0x0252	open back rounded vowel	vowels	Yes	1180

ID	Hex-Code	Name	Category	Simple IPA?	Count
314	0x028C	open-mid back unrounded vowel	vowels	Yes	1477
318	0x0289	close central rounded vowel	vowels	Yes	455
319	0x026A	near-close near-front unrounded vowel	vowels	Yes	10726
321	0x028A	near-close near-back rounded vowel	vowels	Yes	4991
322	0x0259	mid central vowel	vowels	Yes	15078
324	0x0250	near-open central vowel	vowels	Yes	11
325	0x00E6	near-open front unrounded vowel	vowels	Yes	2002
326	0x025C	open-mid central unrounded vowel	vowels	Yes	1114
401	0x02BC	ejective	consonants (non-pulm.)	Yes	34
404	0x02B0	aspirated	diacritics	No	5291
405	0x0324	breathy voiced	diacritics	No	15
406	0x0330	creaky voiced	diacritics	No	263
407	0x033C	linguolabial	diacritics	No	81
408	0x032A	dental	diacritics	No	34
410	0x033B	laminal	diacritics	No	3293
420	0x02B7	labialized	diacritics	No	383
421	0x02B2	palatalized	diacritics	No	126
424	0x0303	nasalized	diacritics	No	175
425	0x207F	nasal release	diacritics	No	36
427	0x031A	no audible release	diacritics	No	97
431	0x0329	syllabic	diacritics	No	461
501	0x02C8	primary stress	suprasegmentals	No	279
503	0x02D0	long	suprasegmentals	No	15931
504	0x02D1	half-long	suprasegmentals	No	30
506	0x002E	syllable break	suprasegmentals	No	24

A.2 Structure

This data set is delivered in two zip files: *babble_speech.zip* contains all files and directory structure needed. This includes a problem definition for T2T which also automatically downloads and unpacks the second file containing audio data, *babble_speech_data.zip*.

This data archive contains the audio data as .wav-files. We provide the audio as original recording. The expert transcriptions for each audio file are saved in the main *babble.csv* file. The file is the main entry point to create further sub tasks and contains information for each of the audio files:

- identifier: Identifier of the data point and audio file is composed of meta information:

$$(child\_id)(session\_id)[OK|PO] - (seq\_id) \tag{1}$$

- *file_path*: relative path to the audio data (.wav-file)
- *child_id*: child participant identifier
- *session_id*: session identifier
- *seq_id*: Sequence identifier within a recording session
- *duration_sec*: length of the audio signal in seconds
- *length_samples*: length of the audio signal in number of samples (i.e.
- the length of the array when loaded)
- *full_ipa, full_ipa_no_brackets, full_ipa_no_unclear, simplified_ipa, simplified_ipa_no_brackets, simplified_ipa_no_unclear* are transcription annotations Tasks b1, b2, b3, b4, b5, b6. See Section 5.5.3 for further information.
- *babble*: Binary label that indicates if a sequence is considered babble (True, 1) or non-babble (False, 0). Used for binary classification in Task b7. See section 5.5.3 for further information.

Our packaged files unzip into the following directory structure:

```
babble
├── data
│   ├── C01
│   ├── C02
│   └── ...
├── experiments
├── tasks
│   ├── babble.task1.index.txt
│   ├── babble.task1.test.csv
│   ├── babble.task1.train.csv
│   ├── babble.task1.valid.csv
│   └── ...
├── t2t_data
├── scripts
│   ├── speech_transcription_babble.py
│   └── babble_check.py
├── babble.csv
├── README
├── t2t_babble_task1_transformer_small
├── t2t_babble_task1_transformer_medium
├── t2t_babble_task1_transformer_large
├── ...
└── t2t_babble_test
```

The main file contains the following sub directories:

- *data* contains all necessary audio files sorted by participants. Their file names are the identifiers described above.

- *experiments* is the default destination path for all outputs of training using the provided scripts. Each script stores training checkpoints, graph meta data, decoding examples and other training- and evaluation-related data in an automatically created sub directory.

- *t2t_data* is the default destination for t2t-datagen's binary Tensorflow DataRecords. Babble's raw data will be preprocessed, normalised, encoded and then stored into sub directories of this folder. t2t-trainer will train the model with the binary data in that folder.

- *scripts* contains all necessary custom scripts to run a T2T model on the Babble data set. This directory needs to be defined as usr_dir when running t2t binaries.

- *tasks* contains the pre-defined training, validation and test partitions (data splits), indices and vocabularies for each sub task of Babble. These files are used by the problem definition to generate the binary data for training, validation and test. Each line in a partition .csv file contains the relative path to an audio file and its target transcription or label.

The *tasks* directory contains the three partitions with the following naming convention:

$$babble.task(1 \text{ to } 6).(\text{train} \mid \text{test} \mid \text{valid}).csv \tag{2}$$

The partition files can be used with other frameworks for preprocessing or model training. They are comma-separated files with two columns:

$$(\text{relative path to audio file}), (\text{binary label} \mid \text{target sequence of symbols}) \tag{3}$$

Since not every software supports handling IPA symbols by default, we also provided pre-coded version using indices. Indices (*.index.txt*) and vocabulary files (*.voc.txt*) are provided for the transcription tasks. Indices can be used to encode a target sequences from symbols to IDs or decode model outputs from IDs to symbols. In fact, by default the given index files are used by other scripts. **Note that the index 0 is reserved for padding $\langle pad \rangle$ and the index 1 is reserved for unknown symbols $\langle unk \rangle$ in a sequence.** The vocabulary file shows counts of each symbol within the task.

A.3 Scripts and Usage

We provide our baselines scripts as a first entry point to run other models and recreate our baselines:

- t2t_babble_test - example usage script using T2T (similar to baseline scripts but smaller and simpler)
- t2t_babble_task*_transformer_small - baseline script to train a small transformer using T2T
- t2t_babble_task*_transformer_medium - baseline script to train a medium transformer using T2T
- t2t_babble_task*_transformer_large - baseline script to train a large transformer using T2T

These baseline scripts require a Python 3.5+ installation and the following packages:

- Numpy
- SciPy
- TensorFlow 1.12
- Tensor2Tensor 1.12

Assuming that you have a working Python 3.5+ environment, use the commands *pip3 install numpy scipy tensorflow tensor2tensor* (or with CUDA GPUs *pip3 install numpy scipy tensorflow-gpu tensor2tensor*) to install the required Python packages. Once the software requirements are

installed move the Babble directory, change rights of the scripts to executable and test your installation using the *t2t_babble_test* script. This will train a very small transformer with few training steps. The required directory structure will be generated automatically.

The directory *scripts* contains the T2T problem definition and further utility functions. The definition implements two separate T2T problems *babble_transcription* and *babble_classification*. *babble_transcription* is the definition used for the first six transcription sub tasks. It uses original symbols as targets and encodes them in the data generation phase using the index file. When using this definition it requires the user to specify the task id between 1 and 6. *babble_classification* implements an additional 7-th classification sub task. Practically, this definition uses 4 classes: $\langle pad \rangle$, $\langle unk \rangle$, True (T), False (F)). In both cases the encoding can be changed by swapping out the corresponding index file with a custom version. *babble_precoded* are already encoded and does not require the encoding step, but decoding needs to be done manually.

Since we have followed the *T2T* interfaces for our implementations, further baselines can easily be added by changing baseline scripts in order to train different model types. To add new models, implement the T2T model interface[1][2]. We recommend copying the python module containing the T2T model implementation (model sub class with *@registry.register_model* decorator) into the *script* directory. Add an import statement into the *__init__.py* and adapt a baseline script according to your model. Make sure that the hyper parameters and the flag *–data_dir* are correct. The *–usr_dir* flag needs to point to the *script* directory containing the data set definition *speech_transcription_babble.py* and your own model definition. T2T also provides a large number of pre-defined models. Use *t2t-trainer –registry_help* to output an overview of ready-to-use models.

[1] `https://github.com/tensorflow/tensor2tensor/tree/master/tensor2tensor/models`
[2] `https://github.com/tensorflow/tensor2tensor`

Bibliography

[1] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," *ACM Transactions on Information Systems*, vol. 20, no. 1, pp. 116–131, January 2002.

[2] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *CoRR*, vol. abs/1408.3456, 2014.

[3] J. Mitchell and M. Lapata, "Composition in distributional models of semantics," *Cognitive Science*, vol. 34, no. 8, pp. 1388–1439, 2010.

[4] E. Grefenstette and M. Sadrzadeh, "Experimental support for a categorical compositional distributional model of meaning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 1394–1404.

[5] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

[6] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," *CoRR*, vol. abs/1503.08895, 2015.

[7] R. Feynman, R. Leighton, and M. Sands, "The Feynman Lectures on Physics," Boston, 1963.

[8] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[Online]. Available: https://www.tensorflow.org/

[9] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[10] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[11] A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, L. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, and J. Uszkoreit, "Tensor2tensor for neural machine translation," *CoRR*, vol. abs/1803.07416, 2018.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] D. D. Lewis, "Reuters-21578 text categorization test collection."

[14] A. Krizhevsky, V. Nair, and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[15] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 23, pp. 146–162, 1954.

[16] G. Frege, *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Halle, Germany: Halle a/S: Verlag von Louis Nebert, 1879.

[17] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *CoRR*, vol. abs/1410.3916, 2014.

[18] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP*. IEEE, 2015, pp. 5206–5210.

[20] T. Nitta, "The computational power of complex-valued neuron," in *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003*. Berlin, Heidelberg: Springer Verlag, 2003, pp. 993–1000.

[21] I. Aizenberg, *Complex-Valued Neural Networks with Multi-Valued Neurons*. Springer Publishing Company, Incorporated, 2016.

[22] A. Hirose, "Nature of complex number and complex-valued neural networks," *Frontiers of Electrical and Electronic Engineering in China*, vol. 6, no. 1, pp. 171–180, March 2011.

[23] ——, *Complex-Valued Neural Networks: Advances and Applications*, May 2013.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[25] L. Drude, B. Raj, and R. Häb-Umbach, "On the appropriateness of complex-valued neural networks for speech enhancement," in *Interspeech*, 2016.

[26] A. M. Sarroff, V. Shepardson, and M. A. Casey, "Learning representations using complex-valued nets," *CoRR*, vol. abs/1511.06351, 2015.

[27] C. Trabelsi, S. Subramanian, N. Rostamzadeh, S. Mehri, D. Serdyuk, J. F. Santos, Y. Ben-

gio, and C. Pal, "Deep complex networks," 2017.

[28] A. M. Sarroff, "Complex neural networks for audio," May 2018.

[29] H.-S. Choi, J. Kim, J. Huh, A. Kim, J.-W. Ha, and K. Lee, "Phase-aware speech enhancement with deep complex u-net," in *International Conference on Learning Representations*, 2019.

[30] T. Parcollet, Y. Zhang, M. Morchid, C. Trabelsi, G. Linars, R. De Mori, and Y. Bengio, "Quaternion convolutional neural networks for end-to-end automatic speech recognition," 06 2018.

[31] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end speech recognition from the raw waveform," in *Interspeech 2018*, ser. Proceedings of Interspeech 2018, Hyderabad, India, Sep 2018.

[32] Y. Kuroe, M. Yoshida, and T. Mori, "On activation functions for complex-valued neural networks: Existence of energy functions," in *Proceedings of the 2003 Joint International Conference on Artificial Neural Networks and Neural Information Processing*, ser. ICANN/ICONIP'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 985–992.

[33] S. Scardapane, S. V. Vaerenbergh, A. Hussain, and A. Uncini, "Complex-valued neural networks with non-parametric activation functions," *CoRR*, vol. abs/1802.08026, 2018.

[34] D. Xu, H. Zhang, and L. Liu, "Convergence analysis of three classes of split-complex gradient algorithms for complex-valued recurrent neural networks," *Neural Computation*, vol. 22, no. 10, pp. 2655–2677, 2010, pMID: 20608871.

[35] A. Hirose, "Complex-valued neural networks: The merits and their origins," in *2009 International Joint Conference on Neural Networks*, June 2009, pp. 1237–1244.

[36] N. Qian", "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145 – 151, 1999.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[38] T. L. Clarke, "Generalization of neural networks to the complex plane," in *1990 IJCNN International Joint Conference on Neural Networks*, June 1990, pp. 435–440 vol.2.

[39] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 967–969, Apr 1992.

[40] G. M. Georgiou and C. Koutsougeras, "Complex domain backpropagation," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, no. 5, pp. 330–334, May 1992.

[41] T. Nitta, "A back-propagation algorithm for complex numbered neural networks," in *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, vol. 2, Oct 1993, pp. 1649–1652 vol.2.

[42] I. Aizenberg, "'multiple-valued threshold logic' translated by claudio moraga," 1977.

[43] N. N. Aizenberg and I. N. Aizenberg, "Cnn based on multi-valued neuron as a model of as-

sociative memory for grey scale images," in *CNNA '92 Proceedings Second International Workshop on Cellular Neural Networks and Their Applications*, Oct 1992, pp. 36–41.

[44] I. Aizenberg and C. Moraga, "Multilayer feedforward neural network based on multi-valued neurons (mlmvn) and a backpropagation learning algorithm," *Soft Computing*, vol. 11, no. 2, pp. 169–183, Jan 2007.

[45] T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. D. Mori, and Y. Bengio, "Quaternion recurrent neural networks," *CoRR*, vol. abs/1806.04418, 2018.

[46] C. J. Gaudet and A. S. Maida, "Deep quaternion networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–8.

[47] D.-C. Park and T.-K. J. Jeong, "Complex-bilinear recurrent neural network for equalization of a digital satellite channel," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 711–725, May 2002.

[48] S. Goh, M. Chen, D. Popovi, K. Aihara, D. Obradovic, and D. Mandic, "Complex-valued forecasting of wind profile," *Renewable Energy*, vol. 31, no. 11, pp. 1733–1750, 2006.

[49] Y. Özbay, "A new approach to detection of ecg arrhythmias: Complex discrete wavelet transform based complex valued artificial neural network," *Journal of Medical Systems*, vol. 33, no. 6, p. 435, Sep 2008.

[50] A. B. Suksmono and A. Hirose, "Adaptive noise reduction of insar images based on a complex-valued mrf model and its application t o phase unwrapping problem," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 3, pp. 699–709, March 2002.

[51] M. Ceylan, Y. Ozbay, O. Nuri, U. An, and E. Yildirim, "A novel method for lung segmentation on chest ct images: Complex-valued artificial neural network with complex wavelet transform," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 18, 07 2010.

[52] A. Rahman Abdul Ghani, A. Y. H. Al Nuaimi, F. Amin, and K. Murase, "Classification of skeletal wireframe representation of hand gesture using complex-valued neural network," *Neural Processing Letters*, vol. 42, 08 2014.

[53] J. Bruna, S. Chintala, Y. LeCun, S. Piantino, A. Szlam, and M. Tygert, "A theoretical argument for complex-valued convolutional networks," *CoRR*, vol. abs/1503.03438, 2015.

[54] N. Guberman, "On complex valued convolutional neural networks," *CoRR*, vol. abs/1602.09046, 2016.

[55] C. A. Popa, "Complex-valued convolutional neural networks for real-valued image classification," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 816–822.

[56] R. Haensch and O. Hellwich, "Complex-valued convolutional neural networks for object detection in polsar data," in *8th European Conference on Synthetic Aperture Radar*, June 2010, pp. 1–4.

[57] H.-G. Zimmermann, A. Minin, and V. Kusherbaeva, "Comparison of the complex valued and real valued neural networks trained with gradient descent and random search algorithms," in *ESANN*, 2011.

[58] T. Nitta, "Learning dynamics of the complex-valued neural network in the neighborhood of singular points," *Journal of Computer and Communications*, vol. 2, no. 1, pp. 27–32, 2014.

[59] S. Ramasamy, S. Suresh, N. Sundararajan, and H. Kim, "A fully complex-valued radial basis function classifier for real-valued classification problems," *Neurocomputing*, vol. 78, no. 1, pp. 104 – 110, 2012, selected papers from the 8th International Symposium on Neural Networks (ISNN 2011).

[60] Z. Zhang, H. Wang, F. Xu, and Y. Q. Jin, "Complex-valued convolutional neural network and its application in polarimetric sar image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 12, pp. 7177–7188, Dec 2017.

[61] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Applied and Computational Harmonic Analysis*, 5 2015.

[62] S. K. Kumar, "On weight initialization in deep neural networks," *CoRR*, vol. abs/1704.08863, 2017.

[63] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[64] H. Narasimhan, "Learning with complex loss functions and constraints," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Storkey and F. Perez-Cruz, Eds., vol. 84. Playa Blanca, Lanzarote, Canary Islands: PMLR, 09–11 Apr 2018, pp. 1646–1654.

[65] T. Mikolov, M. Karafit, L. Burget, J. Cernock, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech*. ISCA, 2010, pp. 1045–1048.

[66] S. Bengio and G. Heigold, "Word embeddings for speech recognition," in *Interspeech*, 2014.

[67] X. Liu, Y. Shen, K. Duh, and J. Gao, "Stochastic answer networks for machine reading comprehension," *CoRR*, vol. abs/1712.03556, 2017.

[68] D. Q. Nguyen, "An overview of embedding models of entities and relationships for knowledge base completion," *CoRR*, vol. abs/1703.08098, 2017.

[69] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, March 2003.

[70] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*,

vol. 41, no. 6, pp. 391–407, 1990.

[71] H. Xu, G. Qi, J. Li, M. Wang, K. Xu, and H. Gao, "Fine-grained image classification by visual-semantic embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 1043–1049.

[72] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug 2013.

[73] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov, "Devise: A deep visual-semantic embedding model," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2121–2129.

[74] Y. Zhang, D. Song, P. Zhang, P. Wang, J. Li, X. Li, and B. Wang, "A quantum-inspired multimodal sentiment analysis framework," *Theoretical Computer Science*, vol. 752, pp. 21–40, December 2018.

[75] T. Mikolov, "Statistical language models based on neural networks," 2012.

[76] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[77] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1532–1543.

[78] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2177–2185.

[79] O. Melamud, J. Goldberger, and I. Dagan, "context2vec: Learning generic context embedding with bidirectional lstm," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2016, pp. 51–61.

[80] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[81] G. Salton, R. Ross, and J. Kelleher, "Attentive language models," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, 2017, pp. 441–450. [Online]. Available: http://aclweb.org/anthology/I17-1045

[82] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to

align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[83] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *CoRR*, vol. abs/1601.06733, 2016.

[84] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015.

[85] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[86] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[87] C. J. van Rijsbergen, *The Geometry of Information Retrieval*. Cambridge, England, UK: Cambridge University Press, 2004.

[88] P. D. Bruza, K. Kitto, D. Nelson, and C. McEvoy, "Is there something quantum-like about the human mental lexicon?" *Journal of Mathematical Psychology*, vol. 53, no. 5, pp. 362–377, October 2009.

[89] D. Song, M. Lalmas, K. van Rijsbergen, I. Frommholz, B. Piwowarski, J. Wang, P. Zhang, G. Zuccon, P. D. Bruza, S. Arafat, L. Azzopardi, E. D. Buccio, A. Huertas-Rosero, Y. Hou, M. Melucci, and S. Ruger, "How quantum theory is developing the field of information retrieval," in *AAAI Fall Symposium on Quantum Informatics for Cognitive, Social and Semantic Processes 2010*. Arlington, Va: AAAI Press, 2010, pp. 105–108.

[90] F. A. González and J. C. Caicedo, "Quantum latent semantic analysis," in *Proceedings of the Third International Conference on Advances in Information Retrieval Theory*, ser. ICTIR'11. Berlin, Heidelberg: Springer Science+Business Media, 2011, pp. 52–63.

[91] D. Widdows and T. Cohen, "Real, complex, and binary semantic vectors," in *Quantum Interaction - 6th International Symposium, QI 2012, Paris, France, June 27-29, 2012, Revised Selected Papers*, 2012, pp. 24–35.

[92] A. Sordoni, J.-Y. Nie, and Y. Bengio, "Modeling term dependencies with quantum language models for ir," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '13. New York, NY, USA: ACM, 2013, pp. 653–662.

[93] A. Sordoni, J. He, and J.-Y. Nie, "Modeling latent topic interactions using quantum interference for information retrieval," in *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, ser. CIKM '13. New York, NY, USA: ACM, 2013, pp. 1197–1200.

[94] T. Van de Cruys, T. Poibeau, and A. Korhonen, "A tensor-based factorization model semantic compositionality," in *Proceedings of the NAACL-HLT 2013*, ser. NAACL-HLT 2013, 2013.

[95] W. Blacoe, E. Kashefi, and M. Lapata, "A quantum-theoretic approach to distributional semantics," in *Proceedings of the 2013 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2013, pp. 847–857.

[96] A. Sordoni and J.-Y. Nie, "Looking at vector space and language models for ir using density matrices," in *Selected Papers of the 7th International Conference on Quantum Interaction - Volume 8369*, ser. QI 2013. Berlin, Heidelberg: Springer-Verlag, 2014, pp. 147–159. [Online]. Available: https://doi.org/10.1007/978-3-642-54943-4_13

[97] M. Xie, Y. Hou, P. Zhang, J. Li, W. Li, and D. Song, "Modeling quantum entanglements in quantum language models," in *24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*. AAAI Press, July 2015, pp. 1362–1368.

[98] J. Li, P. Zhang, D. Song, and Y. Hou, "An adaptive contextual quantum language model," *Physica A: Statistical Mechanics and its Applications*, vol. 456, pp. 51–67, August 2016.

[99] H. Zhang and J. Ma, "Hartley spectral pooling for deep learning," *CoRR*, vol. abs/1810.04028, 2018.

[100] Q. Li, S. Uprety, B. Wang, and D. Song, "Quantum-inspired complex word embedding," in *Proceedings of the 3rd Workshop on Representation Learning for NLP*. Association for Computational Linguistics, July 2018, pp. 50–57, hosted by the 56th Annual Meeting of the Association for Computational Linguistics.

[101] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International Conference on Machine Learning (ICML)*, vol. 48, 2016, pp. 2071–2080.

[102] T. Trouillon, C. R. Dance, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Knowledge graph completion via complex tensor factorization," *CoRR*, vol. abs/1702.06879, 2017.

[103] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 1955–1961.

[104] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, August 2013.

[105] E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. B. Blaschko, and E. Belilovsky, "Scattering networks for hybrid representation learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[106] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 2449–2457.

[107] M. Wolter and A. Yao, "Complex gated recurrent neural networks," in *NeurIPS*, 2018.

[108] H.-G. Zimmermann, A. Minin, and V. Kusherbaeva, "Historical consistent complex valued recurrent neural network," in *Artificial Neural Networks and Machine Learning – ICANN 2011*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 185–192.

[109] A. Minin, A. Knoll, and H.-G. Zimmermann, "Complex valued recurrent neural network:

From architecture to training," vol. 3, 2012.

[110] H. Michel and A. Awwal, "Artificial neural networks using complex numbers and phase encoded weights," *Applied Optics*, vol. 49, no. 10, pp. B71–82, 04 2010.

[111] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," *CoRR*, vol. abs/1511.06464, 2015.

[112] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas, "Full-capacity unitary recurrent neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4880–4888.

[113] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, June 1993.

[114] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," *CoRR*, vol. abs/1506.07285, 2015.

[115] X. Li and D. Roth, "Learning question classifiers," in *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, ser. COLING '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–7.

[116] H. Aoki and Y. Kosugi, "An image storage system using complex-valued associative memories," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 2, Sept 2000, pp. 626–629 vol.2.

[117] P. Zhang, J. Niu, Z. Su, B. Wang, L. Ma, and D. Song, "End-to-end quantum-like language models with application to question answering," in *32nd AAAI Conference on Artificial Intelligence (AAAI-18)*. Association for the Advancement of Artificial Intelligence, February 2018.

[118] J. Weston, A. Bordes, S. Chopra, and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," *CoRR*, vol. abs/1502.05698, 2015.

[119] P. Wittek, *Quantum Machine Learning*, 1st ed. Academic Press, 8 2014, vol. 1.

[120] R. E. Greene and S. G. Krantz, *Function Theory of One Complex Variable*. American Mathematical Society, 2002.

[121] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[122] S. Clark, B. Coecke, and M. Sadrzadeh, "Mathematical foundations for a compositional distributed model of meaning," *Linguistic Analysis*, vol. 36, no. 1-4, pp. 345–384, 2011, categoryb theory mathematics for compositional distributional semantics.

[123] R. Piedeleu, D. Kartsaklis, B. Coecke, and M. Sadrzadeh, "Open system categorical quantum semantics in natural language processing." in *CALCO*, ser. LIPIcs, vol. 35. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 270–289.

[124] W. Blacoe, "Semantic composition inspired by quantum measurement," in *Proceedings of the 8th International Conference on Quantum Interaction (QI 2014)*, 2014, pp. 41–53.

[125] J. R. Busemeyer and P. D. Bruza, *Quantum Models of Cognition and Decision*, 1st ed. Cambridge, England, UK: Cambridge University Press, 2014.

[126] B. Wang, Q. Li, M. Melucci, and D. Song, "Semantic hilbert space for text representation learning," *CoRR*, vol. abs/1902.09802, 2019.

[127] G. Bouma, "Normalized (pointwise) mutual information in collocation extraction."

[128] S. Reddy, D. McCarthy, and S. Manandhar, "An empirical study on compositionality in compound nouns," in *Proceedings of 5th International Joint Conference on Natural Language Processing (IJCNLP-2011)*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, November 2011, pp. 210–218.

[129] E. Guevara, "A regression model of adjective-noun compositionality in distributional semantics," in *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, ser. GEMS '10. Association for Computational Linguistics, 2010, pp. 33–37.

[130] E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni, "Multi-step regression learning for compositional distributional semantics," *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*, 2013.

[131] A. Komninos and S. Manandhar, "Dependency based embeddings for sentence classification tasks," in *HLT-NAACL*, 2016.

[132] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *CoRR*, vol. abs/1609.07843, 2016. [Online]. Available: http://arxiv.org/abs/1609.07843

[133] M. Kobayashi, "Dual-numbered hopfield neural networks," *IEEJ Transactions on Electrical and Electronic Engineering*, 2017.

[134] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012.

[135] A. Graves, A. rahman Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649.

[136] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm," in *INTERSPEECH*, 2017.

[137] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur, Y. Li, H. Liu, S. Satheesh, D. Seetapun, A. Sriram, and Z. Zhu, "Exploring neural transducers for end-to-end speech recognition," *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 2017.

[138] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," *CoRR*, vol. abs/1805.03294, 2018.

[139] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for

large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4960–4964.

[140] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4945–4949.

[141] C. Huang, T. Chen, and E. Chang, "Accent issues in large vocabulary continuous speech recognition," *International Journal of Speech Technology*, vol. 7, no. 2, pp. 141–153, Apr 2004.

[142] A. Rosenberg, S. Thomas, B. Ramabhadran, and M. Hasegawa-Johnson, "Joint modeling of accents and acoustics for multi-accent speech recognition," 2018.

[143] Z. Chen, Y. Luo, and N. Mesgarani, "Speaker-independent speech separation with deep attractor network," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. PP, 07 2017.

[144] A. Kumar and D. A. F. Florêncio, "Speech enhancement in multiple-noise conditions using deep neural networks," in *INTERSPEECH*, 2016.

[145] M. Kolbæk, Z.-H. Tan, and J. Jensen, "Monaural speech enhancement using deep neural networks by maximizing a short-time objective intelligibility measure," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5059–5063, 2018.

[146] S.-F. Fu, T.-W. Wang, Y. Tsao, X. Lu, and H. Kawai, "End-to-end waveform utterance enhancement for direct evaluation metrics optimization by fully convolutional neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1570–1584, Sep. 2018.

[147] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of timefrequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, September 2011.

[148] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 369–376.

[149] Z. Zhang, A. Cristia, A. Warlaumont, and B. Schuller, "Automated classification of childrens linguistic versus non-linguistic vocalisations," in *Proceedings of Interspeech 2018*, 2018, pp. 2588–2592.

[150] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu, "Cnnpack: Packing convolutional neural networks in the frequency domain," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. USA: Curran Associates Inc., 2016, pp. 253–261.

[151] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based

transformation in mfcc computation for speaker recognition," *Speech Communication*, vol. 54, no. 4, pp. 543 – 565, 2012.

[152] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[153] C. Li, X. Liao, and J. Yu, "Complex-valued wavelet network," *Journal of Computer and System Sciences*, vol. 67, no. 3, pp. 623 – 632, 2003.

[154] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[155] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 1243–1252.

[156] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–1764–II–1772.

[157] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *CoRR*, vol. abs/1609.03193, 2016.

[158] S. Wiseman and A. M. Rush, "Sequence-to-sequence learning as beam-search optimization," in *EMNLP*, 2016.

[159] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[160] J. Salazar, K. Kirchhoff, and Z. Huang, "Self-attention networks for connectionist temporal classification in speech recognition," 01 2019.

[161] V. Liptchinsky, G. Synnaeve, and R. Collobert, "Letter-based speech recognition with gated convnets," *CoRR*, vol. abs/1712.09444, 2017.

[162] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, J. Chen, J. Chen, Z. Chen, M. Chrzanowski, A. Coates, G. Diamos, K. Ding, N. Du, E. Elsen, J. Engel, W. Fang, L. Fan, C. Fougner, L. Gao, C. Gong, A. Hannun, T. Han, L. Johannes, B. Jiang, C. Ju, B. Jun, P. LeGresley, L. Lin, J. Liu, Y. Liu, W. Li, X. Li, D. Ma, S. Narang, A. Ng, S. Ozair, Y. Peng, R. Prenger, S. Qian, Z. Quan, J. Raiman, V. Rao, S. Satheesh, D. Seetapun, S. Sengupta, K. Srinet, A. Sriram, H. Tang, L. Tang, C. Wang, J. Wang, K. Wang, Y. Wang, Z. Wang, Z. Wang, S. Wu, L. Wei, B. Xiao, W. Xie, Y. Xie, D. Yogatama, B. Yuan, J. Zhan, and Z. Zhu, "Deep speech 2 : End-to-end speech recognition in english and mandarin," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 173–182.

[163] C.-C. Chiu, T. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," 2018. [Online]. Available: https://arxiv.org/pdf/1712.01769.pdf

[164] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition," in *INTERSPEECH 2017*, 2017.

[165] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent, Y. Bengio, and A. Courville, "Towards end-to-end speech recognition with deep convolutional neural networks," 09 2016, pp. 410–414.

[166] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, "Fully convolutional speech recognition," *CoRR*, vol. abs/1812.06864, 2018.

[167] I. P. Association, "International phonetic alphabet," 2018. [Online]. Available: http://www.internationalphoneticassociation.org/content/ipa-chart