Experimental Comparison of Ansätze for Quantum Natural Language Processing

Nikhil Khatri

Lady Margaret Hall
University of Oxford

A thesis submitted for the degree of

Master of Science in Advanced Computer Science

Trinity 2022

Acknowledgements

Dimitri Kartsaklis, Aleks Kissinger, Mark Koch, My family, My friends back home

Abstract

The DisCoCat model of natural language is a framework which serves as the foundation for several contemporary experiments in Quantum Natural Language Processing (QNLP). In this model, sentences are represented as string diagrams which compose the meaning of the entire sentence from the meaning of each word. An essential step in performing QNLP tasks is converting DisCoCat string diagrams into quantum circuits, through a mapping known as an *ansatz*. While several ansätze have been studied in the broader quantum machine learning literature, no comparative study of ansätze exists for the specific case of QNLP models. In this work, we implement multiple ansätze and compare their performance on a variety of QNLP tasks. In doing so, the first experimental results for a QNLP model applied to the paraphrase identification task are presented. We propose a novel approach to overcome the out-of-vocabulary problem faced by contemporary QNLP models, and demonstrate that our proposed method outperforms naive approaches by a significant margin. Further, experimental results relating to the barren plateau phenomenon in quantum machine learning are provided. We present evidence suggesting that circuits derived from DisCoCat string diagrams for binary classification tasks are no more susceptible to the barren plateau problem than regular quantum machine learning circuits.

Contents

Chapter 1

Introduction

The DisCoCat model of natural language, described by Coecke et al. [1], presents a unification of two previously orthogonal approaches to modelling natural language semantics. The first is the *distributional* approach, in which word meanings are represented in finite dimensional vector spaces, where the proximity of vectors indicates semantic similarity. The second is the *compositional* approach, where meanings of expressions arise from composition of the meanings of individual components. The DisCoCat model combines these two approaches and presents a **Dis**tributional **Co**mpositional model of language, which has its foundation in constructions from **Cat**egory theory.

It has been argued that NLP tasks using the DisCoCat framework are strongly compatible with quantum computing [2, 3]. That is, NLP problems represented in the DisCoCat model are a natural fit for quantum computation, owing to the common categorical structure underlying DisCoCat and the Hilbert space formalism of quantum computing. This has motivated research into the field of *Quantum Natural Language Processing* (QNLP). Recently, experiments by Meichanetzidis et al. [4] and Lorenz et al. [5] have demonstrated that NLP tasks can be solved on contemporary quantum devices. It is the experimental realisation of QNLP that we consider in our work. We extend the state of the art in 3 ways:

1. We implement and apply new ansätze from quantum machine learning literature to the QNLP tasks considered by Lorenz et al. [5] and demonstrate that the new ansätze

outperform the standard IQP ansatz used in QNLP thus far.

2. We consider the paraphrase identification task: that of determining whether two sentences have the same meaning, and present the first experimental results for a QNLP model applied to this problem.
3. We propose a novel approach to handling out-of-vocabulary words when evaluating QNLP models, and demonstrate that models augmented with our method yield significant performance improvement.

In addition to these contributions to QNLP, we present a study of the barren plateau phenomenon. We provide evidence that the barren plateau phenomenon does not always manifest when using a non-linear loss function. We detail an argument that quantum circuits for binary classification problems generated using the DisCoCat framework are no more susceptible to the barren plateau phenomenon than general QML circuits. The rest of this report is arranged as follows:

- **Chapter 2** presents the background needed for this report.
- In **Chapter 3** we describe the DisCoCat framework for QNLP on which our work is based.
- **Chapter 4** describes the ansätze we consider, and presents results comparing the performance of these for the meaning classification and relative-pronoun classification tasks.
- In **Chapter 5** we describe the paraphrase identification task, and present the first QNLP results for the same.
- In **Chapter 6** we consider the out-of-vocabulary problem in QNLP. We propose a novel solution to mitigate its effects and present the results of our method.
- In **Chapter 7** we turn our attention to the barren plateau phenomenon in quantum machine learning. We demonstrate experimental results which suggest that some non-linear loss functions are less susceptible to the barren plateau problem. We further experimentally compare the expressibilities of DisCoCat circuits with regular layered circuits.

Chapter 2

Background

2.1 Quantum Computing

Quantum computing is the use of quantum-mechanical phenomena to perform computational tasks. While the past few years have seen quantum computing receive broad public attention, quantum computers were first proposed by Feynman and others in the 1980s [6, 7, 8]. The present generation of quantum computers are called *NISQ* (Noisy Intermediate Scale Quantum) devices [9]. These are devices of up to a few hundred qubits (quantum bits), characterised by appreciable noise when performing computation. NISQ devices are susceptible to decoherence: the loss of information in a qubit due to unwanted interaction with the environment. This decoherence limits the maximum depth of circuits which can be implemented on these devices. Despite these shortcomings of contemporary quantum computers, Google has presented some evidence of *quantum supremacy* using a 53-qubit quantum computer [10]. Further, it has been demonstrated that quantum machine learning (QML) tasks, in specific variational quantum algorithms are well-suited to the NISQ era of quantum computing [11, 12, 13]. In this section, we provide the quantum computing background necessary for our work.

2.1.1 Foundations

The fundamental unit of all quantum computation is a *qubit*, or quantum bit. The state of a qubit is represented using a unit vector in a 2-dimensional complex vector space ($\mathbb{C}^2$) [14]. There are two special qubit states, called the Z basis (or *computational basis*) states:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.1}$$

Any *pure* state of a qubit can be written as a normalised linear combination of these 2 states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad (\alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 = 1) \tag{2.2}$$

A popular method of visually representing a single qubit's state is the *Bloch sphere*, shown in Figure 2.1. Pure states of a single qubit correspond to the surface of the Bloch sphere. The two Z basis states form the north and south poles of the Bloch sphere. A system of n qubits is represented by taking the tensor product of the vectors of the individual qubits. Thus if $|\psi\rangle$ is a system of n qubits, $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$.

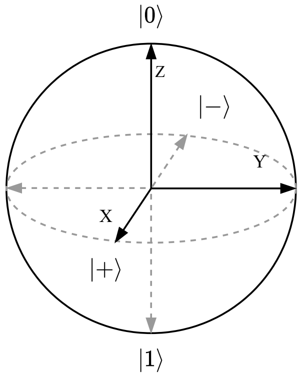


Figure 2.1: The Bloch Sphere

Quantum computation is commonly represented in circuit form. In a quantum circuit, wires represent qubits, which are acted upon using *quantum gates*. These are unitary matrices, acting on the complex vector space of qubits. Some common single-qubits gates are shown below.

$$-\boxed{X}- = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} \qquad -\boxed{Z}- = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad -\boxed{H}- = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \tag{2.3}$$

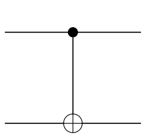
The X gate (formally the Pauli-X gate) is analogous to a classical NOT gate, and performs a simple operation: exchanging the two computational basis states.

$$|0\rangle \mapsto |1\rangle$$
$$|1\rangle \mapsto |0\rangle$$

On the Bloch sphere, the Pauli-X (X) and Pauli-Z (Z) gates represent rotations of π radians about the X and Z axes respectively. The Hadamard gate (H) sends a Z basis state into the *equal superposition* of the two Z basis states:

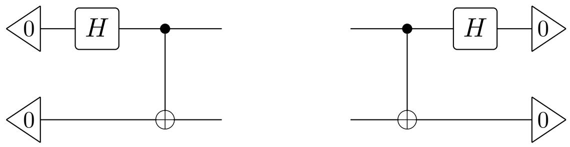
$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$
$$|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

These are the X basis states: $|+\rangle$ and $|-\rangle$. The Hadamard gate thus implements a change of basis from the Z to the X basis, and vice-versa. A fundamental 2-qubit gate is the $CNOT$ gate, which applies the NOT gate to the second qubit (*target*), if the first qubit (*control*) is in state $|1\rangle$.



$$= \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix} \tag{2.4}$$

The adjoint (conjugate transpose) of a state is called an *effect.* The adjoint of a state $|\psi\rangle$ is the effect $\langle\psi|$. A special state and effect pair which are particularly important in the DisCoCat formalism of QNLP are the *Bell state*, and *Bell effect.*



(2.5)

This state is represented by $|\Phi^+\rangle$, and is a *maximally entangled* state on 2 qubits, $|\Phi^+\rangle =$

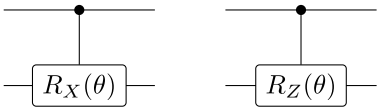
$\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, with the special property that measuring both qubits will always yield the same result. The Bell effect is the adjoint of the Bell state and is an essential construct in DisCoCat string diagrams (Section 3.1). Note that the Bell effect contains the adjoint of the $|0\rangle$ basis state, which represents a *post-selection.* If a qubit is post-selected with state $|0\rangle$, then we only record the output of an experiment when measuring this qubit yields outcome $|0\rangle$. The Bell effect has 2 post-selected qubits, the performance implications of which are discussed further in Section 2.1.5.

2.1.2 Parameterised Quantum Gates

There exist parameterised gates, for which the unitary matrix representing their action is dependent on some parameter θ. Rotation about the X and Z axis of the Bloch sphere, for example, need not be limited to π radians, and can instead be by some arbitrary angle θ.

$$-\boxed{R_X(\theta)}- = \begin{bmatrix} cos(\frac{\theta}{2}) & -i\cdot sin(\frac{\theta}{2}) \\ -i\cdot sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix} \qquad -\boxed{R_Z(\theta)}- = \begin{bmatrix} exp(-i\frac{\theta}{2}) & 0 \\ 0 & exp(i\frac{\theta}{2}) \end{bmatrix} \tag{2.6}$$

There also exist controlled versions of these gates, called CR_Z and CR_X gates respectively.

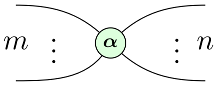


(2.7)

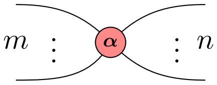
These find extensive use in the quantum circuits we describe in Chapter 4. Quantum circuits which have gates with free parameters are termed *Variation Quantum Circuits* (VQCs). These form the bedrock of quantum machine learning and provide the basis for most contemporary practical applications of NISQ quantum computing [11, 12, 13].

2.1.3 ZX Calculus

The ZX calculus, introduced by Coecke and Duncan [15] is a graphical calculus for reasoning about quantum phenomena. Quantum computations are represented as *string diagrams*, composed primarily of generators called *spiders*:

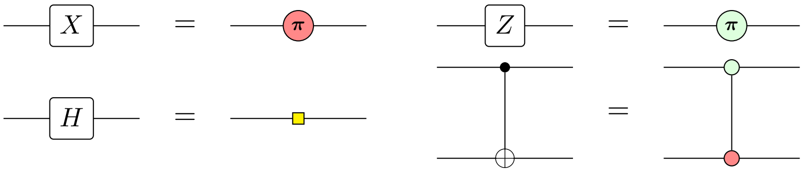


$$:= \quad |0\rangle^{\otimes n}\langle 0|^{\otimes m} + e^{i\alpha}|1\rangle^{\otimes n}\langle 1|^{\otimes m} \tag{2.8}$$



$$:= \quad |+\rangle^{\otimes n}\langle +|^{\otimes m} + e^{i\alpha}|-\rangle^{\otimes n}\langle -|^{\otimes m} \tag{2.9}$$

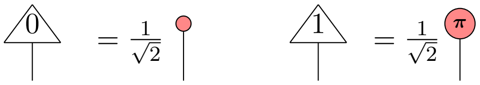
Each wire represents a single qubit ($\mathbb{C}^2$), as in quantum circuits, and the generators represent matrices. String diagrams (in the ZX calculus and more generally) can thus be viewed as enriched *tensor networks* [16, 17]. The ZX calculus has been proven to be complete for quantum mechanics by Ng and Wang [18]. Any quantum circuit can be represented as a ZX diagram. Each of the quantum circuits presented in (2.3) can be represented in the ZX calculus (up to a constant scalar factor) as follows:



(2.10)

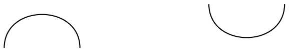
(2.11)

All ZX diagrams and DisCoCat (Section 3.1) diagrams in this work are oriented in a top-down manner. That is, states appear at the top of a diagram, and information flows downward. Thus in ZX diagrams, the Z-basis states are represented as:



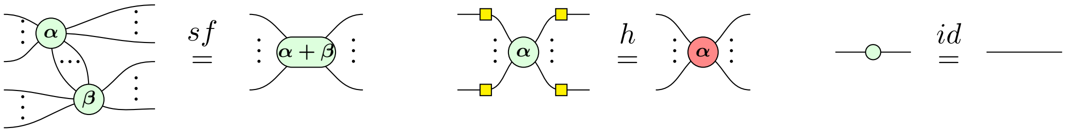
(2.12)

Often when working with ZX diagrams, non-zero scalar factors are ignored [16]. This is especially the case when it is known that the diagram represents a unitary matrix or a normalised state. The Bell state and Bell effect have particularly elegant representations in the ZX calculus, being represented by a *cap* and *cup* respectively.



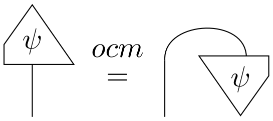
(2.13)

The cup (Bell effect) finds significant use as both a grammatical construct, and a tensor contraction in the DisCoCat language model (Section 3.1). To reason about quantum computation, the ZX calculus defines several rewrite rules. Below, we describe the rules we use in our work. For the complete set of rules, we refer the reader to [16].



(2.14)

Another important property of ZX diagrams (and DisCoCat diagrams) we make use of, is the "only connectivity matters" (*ocm*) property [19]. This states that we can move generators about the plane arbitrarily, provided that the ordering of inputs and outputs is maintained. In particular, this allows us to perform rearrangements such as:



(2.15)

This simple form of diagram rewriting is a powerful tool for optimising DisCoCat diagrams, which we discuss further in Section 3.2.2.

2.1.4 Measurements and Probabilities

For measuring quantum states, we consider the case of *projective measurement* [14]. Projective measurements are defined by an observable M: a Hermitian operator. Each such observable M has a *spectral decomposition*:

$$M = \sum_m m \cdot P_m \tag{2.16}$$

where P_m is the "projector onto the eigenspace" of M, with eigenvalue m [14]. P_m can be used to determine the probability that a measurement yields outcome m. When measuring a quantum state $|\psi\rangle$, the probability that outcome m is observed is given by:

$$p(m) = \langle\psi|P_m|\psi\rangle \tag{2.17}$$

For example, when measuring in the computational basis, the possible outcomes are $\{0, 1\}$, for which the projectors are, respectively $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$. Thus the probability of obtaining output 0 when measuring a state $|\psi\rangle$ is:

$$p(0) = \langle\psi|0\rangle\langle 0|\psi\rangle = |\langle\psi|0\rangle|^2 \tag{2.18}$$

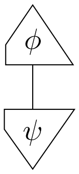
A related concept is the *expectation value* of a quantum measurement. The expectation value is the average result of a measurement using a Hermitian operator, defined as:

$$\mathbf{E}(M) = \langle M \rangle = \langle \psi | M | \psi \rangle \tag{2.19}$$

Outcome probabilities defined by projective measurements as presented above are equivalent to the measurement probabilities defined by the *Born rule.* We adapt the presentation of the Born rule from Coecke and Kissinger [19]. Given state $|\psi\rangle$, the probability of observing it in the state $|\phi\rangle$, after measuring it in the appropriate orthonormal basis, is given by:

$$P(|\psi\rangle = |\phi\rangle) = |\langle \psi | \phi \rangle|^2 \tag{2.20}$$

In diagrammatic notation, two normalised states can be compared in a very elegant manner. To compare two quantum states $|\psi\rangle$ and $|\phi\rangle$, we can evaluate the following diagram, consisting of a state $|\phi\rangle$ plugged into the effect $\langle\psi|$ (obtained by taking the adjoint of $|\psi\rangle$):



(2.21)

Taking the square of the absolute value of this diagram, we get precisely the Born rule probability $|\langle\psi|\phi\rangle|^2$. This serves as a formal measure of similarity between two normalised quantum states. To compare two quantum states in diagrammatic representation, we simply need to plug the two wires together, calculate the complex number that this diagram represents, and take the square of its absolute value. This representation is particularly useful when simulating quantum computation through tensor contraction, as discussed in the following section.

2.1.5 Implementing Quantum Circuits

Once a quantum circuit or ZX diagram representing a desired computation is generated, it is necessary to obtain the output of the computation. This can either be done by physically implementing the gates of the circuit on a quantum device, or through classical simulation. When physically implementing circuits on a NISQ device, multiple *shots* (executions) of the circuit are run, and the output is measured after each execution. This yields a probability distribution over outputs, which can be normalised to arrive at an approximation of the state produced by the circuit.

One construction that requires special consideration when implementing a circuit on a quantum device is post-selection. Typically post-selection is implemented by performing measurements on all qubits, and discarding all results where measuring a qubit did not yield the post-selected value. This process incurs a cost exponential in the number of post-selected qubits [4]. This problem is of significant consequence in the QNLP case, where the majority of qubits in a circuit are post-selected (Sections 3.2.2, 3.2.3).

In absence of a quantum device, it is possible to simulate quantum circuits on classical devices. One method in which this can be achieved is through contracting the tensor network that the quantum circuit represents. Unlike shot-based execution on quantum devices, this method yields an exact result. For an n qubit circuit, contracting the tensor network will yield precisely the state vector $|\psi\rangle \in \mathbb{C}^{2^n}$ prepared by the circuit. For tensor networks where there is no input or output wire, the contraction will yield a single complex number as output. One use-case for such a diagram is the comparison of two quantum states using the Born rule as described in Section 2.1.4. While contracting tensors is exact, it also comes at a high computational cost, since the dimension of each tensor is exponential in the number of qubits associated with it. In all experiments described in this work, we employ classical simulation through tensor contraction.

2.2 Quantum Machine Learning

Machine learning (ML) is the design of computer programs which learn to improve their performance at solving some task, through “experience” [20]. Thus a machine learning model is a program which looks at input data and learns to solve some computational task. A supervised learning task is one in which the model is provided with a training dataset (X_{train}, Y_{train}) consisting of example input-output pairs (x_i, y_i). The model f learns a mapping from the input to the expected output

$$f : X_{train} \rightarrow Y_{train}$$

and the expectation is that the model will generalise, and be able to predict output values for inputs it did not see during training:

$$f(X_{test}) = \widehat{Y}_{test}$$

Quantum Machine Learning (QML) is the application of quantum computers to machine learning tasks, where the data is often classical (non-quantum) [21]. This is typically achieved through the use of Variational Quantum Circuits (VQCs, Section 2.1.2), trained using quantum-classical hybrid optimisation schemes [21, 11]. In such a QML model, a VQC $U(\theta)$ is used to prepare a parameterised quantum state $|\psi(\theta)\rangle$ on a quantum device, or classical simulator. This quantum state is then measured, yielding an expectation value, as described in Section 2.1.4. A loss function $\mathcal{L}$ uses this expectation value to calculate how good an approximation of the expected output is generated by the QML model. A classical optimiser performs this process multiple times, with varying candidate parameter assignments θ, in order to obtain an optimal parameter assignment which minimises the value of the loss function $\mathcal{L}(\theta)$. We describe each step involved in training such a QML model for NLP tasks in detail in Section 3.2.

2.3 Natural Language Processing

Broadly, Natural Language Processing (NLP) is the use of computational techniques to understand and manipulate natural language [22]. The vast majority of contemporary approaches to NLP can be considered Machine Learning with text data. That is, they apply standard ML models like neural networks to textual datasets, without any explicit consideration of the grammatical structure of data [2]. The DisCoCat model of QNLP which we use in this work, in contrast, is motivated by linguistic structure. It is an approach to NLP that utilises the grammatical structure of sentences to shape computation. This model, and the grammatical structures which form its foundation are discussed in Section 3.1.

In our work, we apply QNLP approaches to NLP tasks which can be represented as binary classification. In Chapter 4 we consider the task of meaning classification (Task 4.2.1): determining which of two themes a given sentence belongs to. We then consider the more grammatically-involved task of determining whether a given noun phrase contains an object-relative or subject-relative clause (Task 4.2.2). Finally, we turn our attention to the paraphrase identification task, where the model attempts to predict whether two sentences have the same meaning (Task 5.0.1).

Chapter 3

QNLP with String Diagrams

3.1 DisCoCat

The model of QNLP that we use throughout this work is based on the DisCoCat model of natural language. The DisCoCat model was proposed by Coecke et al. as a **Dis**tributional, **Co**mpositional, **Cat**egorical model of natural language [1]. DisCoCat utilises the grammatical structure of a sentence to generate a semantic representation for it, through the composition of semantic representations of its words. The distributional aspect of the model is that the semantic representations of words are vectors in some finite-dimensional vector space.

The grammatical structure underlying DisCoCat is a *pregroup grammar*, due to Lambek [23]. A pregroup grammar is a partially-ordered monoid, where each element is a type p, with a left adjoint p^l and a right adjoint p^r, such that the following conditions hold:

$$p \cdot p^r \leq 1 \leq p^r \cdot p \qquad p^l \cdot p \leq 1 \leq p \cdot p^l \tag{3.1}$$

To treat this partially-ordered monoid as a grammar, the two axioms $p \cdot p^r \leq 1$ and $p^l \cdot p \leq 1$ serve as reductions in the grammar, and are represented as:

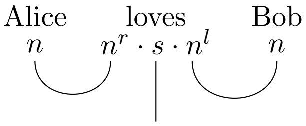
$$p \cdot p^r \to 1 \qquad p^l \cdot p \to 1 \tag{3.2}$$

Example types for a pregroup grammar are s, representing well-formed sentences, and n representing nouns or noun phrases. Using this, we can present the grammatical derivation for an example sentence: "*Alice loves Bob*". Here *Alice* and *Bob* are nouns, having type

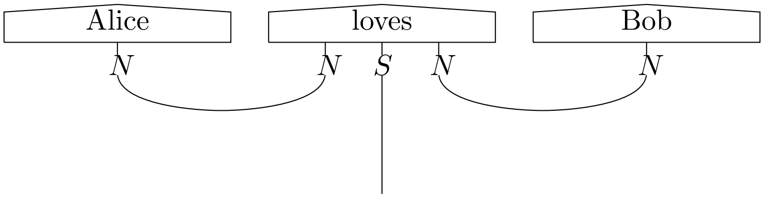
n, and *loves* is a transitive verb, with type $n^r \cdot s \cdot n^l$. The grammatical derivation for this sentence, using the reductions in (3.2) is:

$$n \cdot (n^r \cdot s \cdot n^l) \cdot n \leq \ 1 \cdot s \cdot n^l \cdot n \leq \ 1 \cdot s \cdot 1 \leq \ s \tag{3.3}$$

Such a reduction can be represented graphically as:



Here each *cup* represents the application of a reduction. This pregroup structure can be mapped to the category of finite dimension vector spaces **FDVect**, through a *functor*: a category-theoretic structure-preserving map [24]. This mapping converts types into vector spaces ($n \mapsto N,\ n^r \cdot s \cdot n^l \mapsto N \otimes S \otimes N$), and converts reductions into tensor contractions. Applying this functor to the reduction for our example sentence, we get the DisCoCat diagram:



Here each word is represented by a vector in a vector-space determined by its type. The meaning of the sentence is calculated through tensor contraction of the diagram, the structure of which is obtained from the sentence's grammatical reduction. In this manner DisCoCat provides a method to represent the meaning of a sentence through the composition of the distributional meanings of individual words.

3.2 The QNLP Pipeline

In this section we describe the sequence of steps involved in any QNLP task using the DisCoCat framework. We describe the process from receiving a training dataset to generating a trained QNLP model which can be evaluated on a test dataset. The pipeline we describe is adapted from the pipeline proposed by Kartsaklis et al. [17] for the `lambeq` library. `lambeq` is a toolkit for QNLP, implemented in Python, which provides functionality for each step of

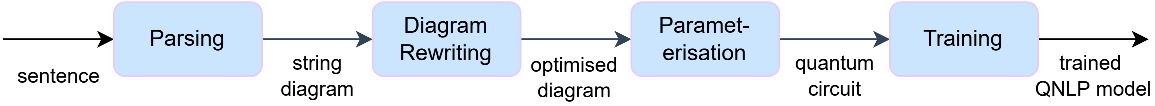


Figure 3.1: The QNLP pipeline. Adapted from Kartsaklis et al. [17]

this pipeline. We leverage `lambeq` extensively in this work, and describe it in further detail in Section 3.3. A high-level view of the pipeline can be seen in Figure 3.1.

3.2.1 Parsing and Diagram Generation

The DisCoCat model of natural language which is the foundation of our approach to QNLP, is based on representing sentences as string diagrams. The first step thus, is to generate DisCoCat diagrams for input sentences. While DisCoCat is originally defined in terms of a pregroup grammar, there is no statistical pregroup parser capable of providing derivations for arbitrary sentences [25, 17]. To circumvent this problem, Yeung and Kartsaklis [25] describe a version of DisCoCat which can generate a string diagram representing the syntactic structure of a sentence using a Combinatory Categorical Grammar (CCG) [26]. Several statistical CCG parsers are available [27, 28], which can be used to generate DisCoCat diagrams for arbitrary text. Figure 3.2 shows the DisCoCat diagram generated for the sentence "*Alice loves Bob*" by the *Bobcat* parser [27, 25] provided in `lambeq`.

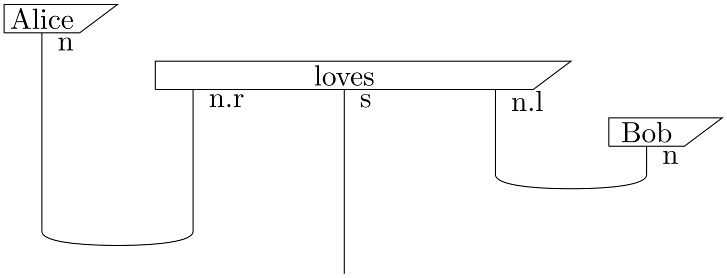


Figure 3.2: DisCoCat string diagram for the sentence *"Alice loves Bob"*

3.2.2 Diagram Rewriting

In their default state, DisCoCat string diagrams consist of a layer of word-states tensored together, followed by a layer consisting of cups, and identity wires. While this is a natural structure given the relation with pregroup grammars described above, it is not the only possible arrangement of the states. In fact, it is often possible to restructure diagrams in a manner which makes downstream tasks more efficient. Consider the diagram in Figure 3.2, generated for the sentence "*Alice loves Bob*".

Recalling that our goal is to eventually turn this diagram into a circuit which can be run on a quantum computer, we must take into account the efficiency of implementation of this quantum circuit. A constraint commonly imposed on quantum circuits is the number of qubits they are allowed to have. In classical simulation of quantum circuits, the complexity of simulation is exponential in the number of qubits, in the general case. When physically implementing a quantum circuit, it is essential that the circuit fits on the physical device. The number of qubits available on contemporary quantum computers is very limited. The largest superconducting quantum computer from IBM at the time of writing has at most 127 qubits [29]. Devices available for public use typically have far fewer qubits available, often less than 10. It is thus beneficial to minimise the number of qubits of our quantum circuits. One method of reducing the width of a DisCoCat quantum circuit, before even converting the string diagram into a circuit is to remove cups from the diagram, by converting a state into an effect, as shown in Figure 3.3.

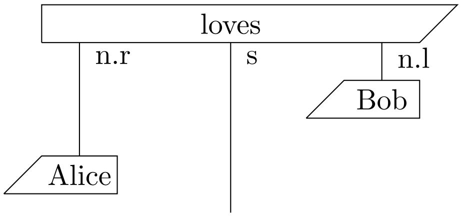


Figure 3.3: DisCoCat string diagram for the sentence *"Alice loves Bob"*, after removing cups.

Here the Alice and Bob states were turned into effects using the ocm property described in (2.15), reducing the width of the circuit. Another advantage of this is that the final circuit contains fewer post-selections, which improves efficiency when running experiments. Other diagrammatic rewrites exist which serve to improve computational efficiency (such as removing cups), or which exploit linguistic properties to improve performance on the learning task. A complete list of rewrite rules used in this work can be found in Appendix A.

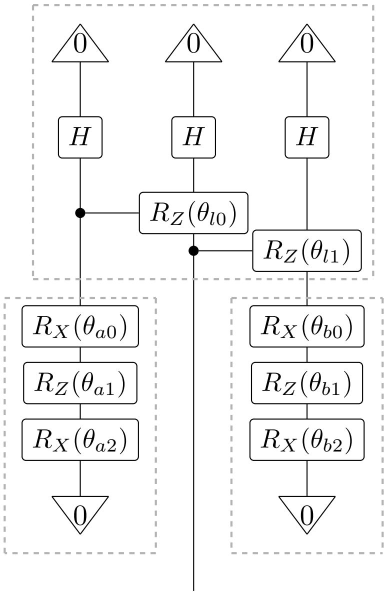


Figure 3.4: Quantum circuit generated by applying the IQP ansatz to the optimised string diagram in Figure 3.3

.

3.2.3 Parameterisation

Once a DisCoCat string diagram is generated for a sentence, for it to be run on a quantum computer requires it to be converted into a parameterised quantum circuit. For this, we employ the use of an *ansatz.* Ansätze are maps which define concrete structures for each box and wire in the string diagrams. For each box in the string diagram, the ansatz choice defines a parameterised quantum circuit (possibly a state or effect), which represents it. When using an ansatz to convert a string diagram into a quantum circuit, the number of qubits associated with each type is provided as hyperparameters. For example, q_n and q_s represent the number of qubits assigned to the noun and sentence types, respectively. These affect the number of qubits in the whole circuit, and also affect the number of parameters associated with each word. An ansatz which has found significant use in prior literature is the IQP (Instantaneous Quantum Polynomial) ansatz [30]. Figure 3.4 shows the circuit generated from the diagram in Figure 3.3, by associating a single qubit with each type ($q_s = q_n = 1$). Here $\{\theta_{a0}, \theta_{a1}, \theta_{a2}\}$, $\{\theta_{l0}, \theta_{l1}\}$ and $\{\theta_{b0}, \theta_{b1}, \theta_{b2}\}$ are the parameters associated with the words *Alice*, *loves* and *Bob* respectively.

The structure of the IQP ansatz is described in detail in Chapter 4. In QML tasks, it is common to generate a layered circuit, consisting of multiple layers of an ansatz, with no additional structure. Coecke et al. describe this as an *empirically-motivated* choice [3]. In the DisCoCat model of QNLP however, high-level circuit structure is *conceptually motivated* (by the pregroup grammar of a sentence), and only the final details are filled in by the ansatz. The choice of ansatz remains extremely important, however, as it controls the size of the final circuit, the number of parameters, and most importantly: it can significantly impact learning performance. This portion of the pipeline is what we are primarily concerned with through the rest of this work. We describe the ansätze which we use in our experiments, and their properties in Chapter 4.

3.2.4 Training

Once parameterisation is complete, it is time to begin the actual machine-learning portion of the pipeline. That is, to train the QNLP model to solve the given NLP task. In this work we exclusively consider *supervised machine learning.* This is a type of machine learning task in which the dataset consists of pairs (x_i, y_i) where x_i is the i'th input, and y_i is the corresponding label. This label can be a numerical (real) value, in which case the task is one of *regression*; or it can be one of a finite set of classes $\mathcal{C}$, in which case the task is one of *classification.* Throughout this work we exclusively consider classification tasks. Training in the QNLP case does not differ in significant respect from general QML training.

As in classical machine learning, training a QML model boils down to finding a suitable parameter assignment θ, such that the output of some *loss function* $\mathcal{L}(\theta)$ is minimised. Here θ is a vector of the model's parameters.

$$\theta = \langle \theta_1,\ \theta_2,\ ...\ \theta_i, ...\ \rangle$$

A loss function compares the model's predicted output $\widehat{y}_\theta$ with the expected output y, and returns a real value which serves as a measure of incorrectness. In classical ML, the prevalent method of finding such an optimal parameter assignment is *gradient descent.* Here an initial parameter assignment is chosen, and the gradient of the loss function with respect to each parameter is calculated. The parameter assignment is then modified in the direction of steepest descent of this gradient. The t'th step of gradient descent can be described as:

$$\theta^{t+1} = \theta^t + \gamma \frac{\partial \mathcal{L}}{\partial \theta}\bigg|_{\theta=\theta^t} \tag{3.4}$$

where the gradient is evaluated at θ^t: the parameter assignment in the t'th iteration. Here γ is the *learning rate*, a hyperparameter that controls the step-size by which θ is changed in each iteration. This process is then repeated either for a predetermined number of iterations, or until some stop-condition (a threshold on test loss or accuracy for example) is met. For QML, gradient descent would require calculating the analytical gradient of the quantum circuit. Techniques exist to calculate the gradient with respect to a parameter through circuit evaluation [31]. Each step of gradient descent would involve executing the gradient circuits for each parameter, which invokes a very high cost of computation [17, 5]. Techniques also exist to diagrammatically calculate the exact gradient [32], but these too have a high computation cost, requiring us to evaluate a diagram for each parameter in the circuit.

A more commonly employed approach to finding an optimal parameter assignment is to use a gradient-based classical optimiser which calculates an approximation of the gradient, without evaluating the analytical gradient circuit. A trivial example of such an approach is the method of finite-differences. Here for each parameter θ_i, the loss function is evaluated by changing the value of this parameter to $\theta_i + \epsilon$ and $\theta_i - \epsilon$ for some very small value ϵ. For both these parameter assignments, the optimiser runs the circuit, measures its output, and evaluates the loss function on this output. The difference between these 2 values of the loss function, divided by $2 \cdot \epsilon$ yields an approximation of the gradient with respect to θ_i:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} \approx \frac{\mathcal{L}(\langle \theta_1^t,\ \theta_2^t,\ \dots\ \theta_i^t + \epsilon, \dots\ \rangle) - \mathcal{L}(\langle \theta_1^t,\ \theta_2^t,\ \dots\ \theta_i^t - \epsilon, \dots\ \rangle)}{2 \cdot \epsilon} \tag{3.5}$$

This is done for each parameter, in order to estimate the gradient used in 3.4.

The method of finite differences is very computationally inefficient as it requires evaluating the circuit and loss function $2|\theta|$ times for each iteration. A more efficient gradient-based optimisation approach is the SPSA (Simultaneous Perturbation Stochastic Approximation) algorithm [33]. The SPSA algorithm approximates the gradient by evaluating the circuit and loss function at just two points in each iteration, irrespective of the number of parameters. The two points are $\theta^t + \Delta\theta$, $\theta^t - \Delta\theta$, which are diametrically opposite along a random direction $\Delta\theta$ in the parameter space. We use the SPSA implementation provided in `lambeq` [17] in all experiments described in this work.

3.2.5 Putting it all Together

Above we have described each of the high-level steps involved in solving a QNLP task. We now formalise how these are composed in order to train a model for a specific task, and generate predictions for unseen data, in algorithmic form. Algorithm 1 describes the process of training a QNLP model. For succinctness, we use syntax akin to Python's list comprehension in lines 2, 3 and 4. In these lines the input sentences are converted to diagrams, which are optimised and converted into quantum circuits using the methods described previously. Then for each word in the training vocabulary V_{train} a random parameter vector is initialised. The parameter vectors for each word are concatenated into a single vector θ^0, which serves as the parameter initialisation for training. Once all epochs of the optimiser have run, the final parameter assignment for each word is obtained, and stored in a dictionary called *param-map*. This *param-map* is the only output needed from the training procedure, and is used during model evaluation.

Algorithm 1 QNLP Model Training

```
 1: procedure QNLP-TRAIN($x_{train}, y_{train}$)
 2:     diagrams ← [TEXT-TO-DIAGRAM($x_i$) for each $x_i$ in $x_{train}$]
 3:     optimised-diagrams ← [REWRITE($d_i$) for each $d_i$ in diagrams]
 4:     circuits ← [ANSATZ($d_i$) for each $d_i$ in optimised-diagrams]
 5:     for each $w_i \in V_{train}$ do
 6:         $\overrightarrow{\theta}^0_i$ ← GENERATE-RANDOM-VECTOR()
 7:     end for
 8:     $\theta^0 \leftarrow \langle \overrightarrow{\theta}^0_1, \overrightarrow{\theta}^0_2, ..., \overrightarrow{\theta}^0_{|V_{train}|} \rangle$
 9:     for $t$ ← 1 to epochs do
10:         $\theta^t$ ← SPSA-ITERATION(circuits, $y_{train}$, $\mathcal{L}$, $\theta^{t-1}$)
11:     end for
12:     param-map ← {$w_i$ : $\overrightarrow{\theta}^{epochs}_i$ for each $w_i \in V_{train}$}
13:     return param-map
14: end procedure
```

Algorithm 2 describes the process to evaluate a trained QNLP model on a test dataset. The input sentences are converted into quantum circuits in the same manner as during training. For each word in the test dataset, the parameter assignment is obtained from the *parameter-map* generated during training. For simplicity it is assumed that all words in the test set were observed in the training set ($V_{test} \subseteq V_{train}$). In Chapter 6 we present a novel approach for performing inference on a dataset with words outside of the training vocabulary. Once the trained parameter assignment is obtained for each word in the vocabulary, it is possible to assign these parameters in the VQCs in *circuits*, contract the concrete circuits, and obtain the model's prediction. In Algorithm 2 this is abstracted into the *GENERATE-PREDICTIONS* function. Given the model's prediction $\widehat{y}$ and the expected output values y_{test}, the model's accuracy and loss on the test dataset can be calculated and returned.

Algorithm 2 QNLP Model Evaluation

```
procedure QNLP-EVAL($param\text{-}map$, $x_{test}, y_{test}$)
    $diagrams \leftarrow$ [TEXT-TO-DIAGRAM($x_i$) for each $x_i$ in $x_{test}$]
    $optimised\text{-}diagrams \leftarrow$ [REWRITE($d_i$) for each $d_i$ in $diagrams$]
    $circuits \leftarrow$ [ANSATZ($d_i$) for each $d_i$ in $optimised\text{-}diagrams$]
    for each $w_i \in V_{test}$ do
        $\overrightarrow{\theta_i} \leftarrow param\text{-}map[w_i]$
    end for
    $\theta_{test} \leftarrow \langle \overrightarrow{\theta_1}, \overrightarrow{\theta_2}, ..., \overrightarrow{\theta}_{|V_{test}|} \rangle$
    $\widehat{y} \leftarrow$ GENERATE-PREDICTIONS($circuits$, $\theta_{test}$)
    $accuracy \leftarrow$ MEASURE-ACCURACY($\widehat{y}, y_{test}$)
    $loss \leftarrow$ MEASURE-LOSS($\widehat{y}, y_{test}, \mathcal{L}$)
    return $accuracy, loss$
end procedure
```

3.3 lambeq

In all experiments discussed in this work, we make extensive use of lambeq: a high-level open-source Python library for QNLP [17]. lambeq provides high-level routines for each stage of the pipeline described in the following section. It provides parsers for a variety of compositional schemes such as DisCoCat, and other less grammatically rich schemes which ignore aspects of semantic structure such as word-order, or word-type. These parsers allow the user to generate string diagrams matching a chosen compositional structure for input sentences. In our experiments, we use the BobCatParser [25, 27] and the DepCCGParser [28] classes. DisCoPy [34] is used as a backend to generate and process string diagrams. `lambeq` has several pre-provided rewrite rules which allow users to optimise string diagrams. Appendix A describes the rewrite rules we use in our experiments.

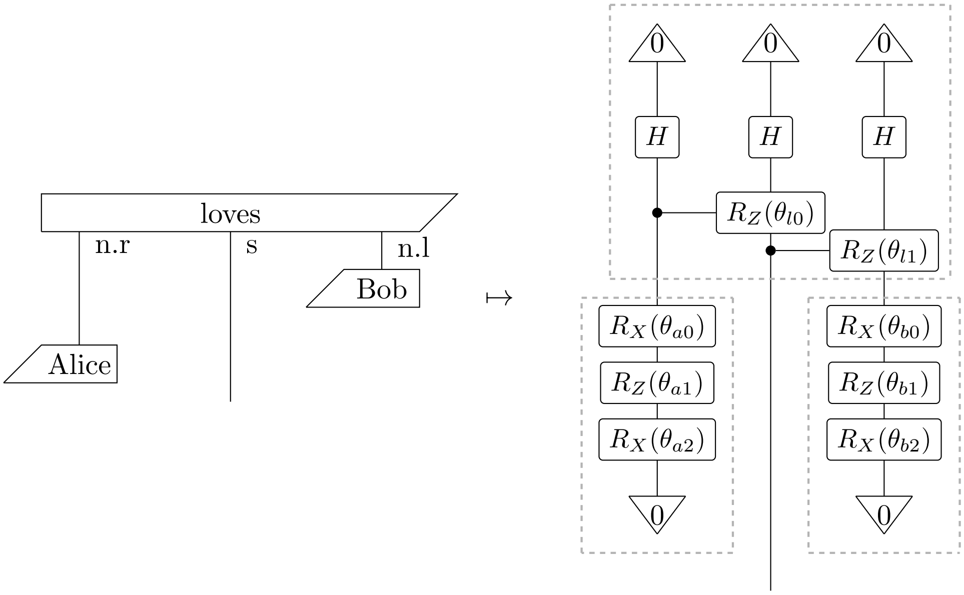
`lambeq` provides functionality relating to both classical and quantum models of learning on string diagrams. In the classical case, the string diagram is converted into a tensor network, without any quantum gates. The tensors of this network are then learnt classically, using PyTorch [35] or Jax [36]. We do not describe this in further detail, as our work is concerned exclusively with quantum models. In quantum models, the tensor network is not directly generated, and the string diagram is converted into a quantum circuit as described in Section 3.2.3. For quantum models, `lambeq` provides the IQP (Instantaneous Quantum Polynomial) ansatz [30], and provides support to add custom ansätze. In our work, we contribute two new quantum ansätze to `lambeq`, and the underlying DisCoPy library. This is detailed further in Chapter 4.

For quantum model training, `lambeq` provides an implementation of the SPSA optimiser [33], which we use in all our experiments. Execution of quantum circuits in `lambeq` is supported both through classical simulation and on quantum devices. Classical simulation is achieved through tensor-contraction using Jax [36]. Circuit execution on quantum devices is achieved through the `tket` compiler [37], which can execute quantum circuits on devices from multiple providers such as Google, IBMQ, etc. We utilise classical simulation using Jax in all experiments.

Chapter 4

Ansätze for QNLP

In domains such as quantum chemistry and quantum simulation, parameterised circuit designs are often inspired by the problem structure. However, it is also extremely common to use "hardware efficient ansätze", which are parameterised random quantum circuits [38]. In the context of QNLP, ansätze serve as maps which convert string diagrams into Variational Quantum Circuits (VQCs). Input sentences are converted into string diagrams using the parsing methodology described in Section 3.2.1. Each box in this string diagram then, can be converted into a quantum circuit using a hardware efficient ansatz. An example of this can be seen in the diagram below.



(4.1)

Here the (optimised) string diagram for the sentence "*Alice loves Bob*" was converted into a VQC using the IQP ansatz, which we describe in the following section. The grey boundaries indicate the portion of the VQC corresponding to each string diagram box. One can view the string diagram as specifying the high-level structure and connectivity of the circuit, while the ansatz fills in the details. Thus, using different ansätze changes the final VQC for which parameters are learned. For hardware efficient ansätze, there is usually no apriori method to indicate the relative performance of two ansätze for a specific task. Thus, in this section, we select 3 ansätze from the literature, and experimentally investigate their performance on two binary-classification NLP tasks.

4.1 Considered Ansätze

In this study, we consider 3 ansätze. The first ansatz that we consider is the IQP (Instantaneous Quantum Polynomial) ansatz [30]. In the `lambeq` implementation, one layer of the IQP ansatz for n qubits consists of a Hadamard gate on each qubit, followed by $n-1$ CR_Z gates, arranged in a ladder. A single layer of the IQP ansatz for 3 qubits can be seen in Figure 4.1. Each of the CR_Z gates is parameterised by some angle θ_j, which is modified as part of the training process. This is omitted from the diagram for brevity. The Hadamard gate is unparameterised.

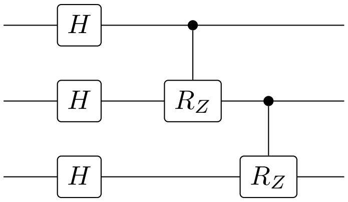
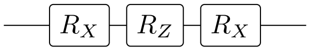


Figure 4.1: IQP ansatz for 3 qubits

The IQP ansatz has been overwhelmingly favoured by previous experiments in QNLP. It is the ansatz used in experiments by Meichanetzidis et al. [4] and Lorenz et al. [5]. An important factor motivating the use of this ansatz is that it is composed of gates which can be natively implemented on IBMQ devices [5]. Further, the successful application of IQP circuits to quantum ML tasks was previously demonstrated by Havlíček et al. [30]. The "instantaneous" nature of the IQP circuit comes from the fact that all gates after the Hadamard layer are diagonal unitaries, which can commute past each other and be applied in any order. Observe that if a box in a DisCoCat diagram has a single qubit as its output type, the default IQP specification will degenerate to a single Hadamard gate (since there is no pair of qubits to apply a CR_Z between). Thus, for single qubit boxes we employ the *Euler decomposition*:



where each gate is parameterised. This construction is used for the *Alice* and *Bob* boxes in the circuit generated in (4.1). This same template is used in the single-qubit case for all ansätze we consider.

In addition, we consider 2 new ansätze which have not previously been applied to QNLP tasks. We consider an adapted form of circuits 14 and 15 from the work of Sim et al. [39]. We refer to these as *Sim14* and *Sim15* henceforth. Sim14 is chosen as it has very high *expressibility* compared to the other circuits considered by Sim et al. [39]. Quantum circuits used in QML must have sufficient expressibility to encode a solution to the considered ML task [12]. The notion of expressibility is discussed in greater detail in Section 7.5. Sim15 was chosen as despite having a very similar structure to Sim14, it is significantly less expressible. Sim15 has the same layout as Sim14, but with different gates and only half as many parameters per layer. Our experiments explore whether this difference in expressibility impacts performance on QNLP tasks.

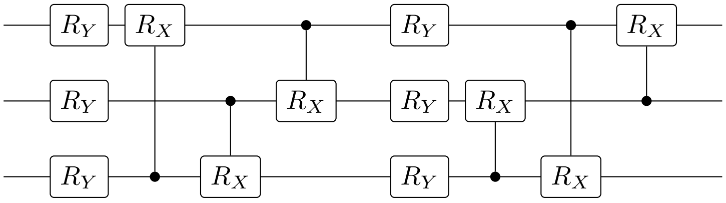


Figure 4.2: Sim14 ansatz for 3 qubits

For an n qubit circuit, each layer of Sim14, consists of two "sublayers". Each sublayer consists of n R_Y gates, followed by n CR_X gates, arranged in a ring topology[1]. In the second sublayer, the ring of CR_X gates iterates in reverse. A single layer, 3-qubit Sim14 ansatz can

[1] The ring topology is defined as: for each $i \in [1...n]$. $CR_X(i, (i-1) \bmod n)$

be seen in Figure 4.2. All R_Y and CR_X gates are parameterised. In the original Circuit 14 from Sim et al., the two sublayers are constructed in the form of *circuit-blocks*, as described by Schuld et al. [40]. We replace this construction with the simpler ring topology. In addition to providing a simpler structure, the ring topology ensures that circuit depth and the number of parameters increase linearly in the number of qubits in the circuit. This is not the case for the circuit-block design [39, 40].

Sim15 has the same layout as Sim14, but replaces all CR_X gates with $CNOT$ gates. A 3 qubit single layer Sim15 ansatz can be seen in Figure 4.3. Since $CNOT$ gates are unparameterised, each layer of Sim15 has only half the number of parameters as a layer of Sim14.

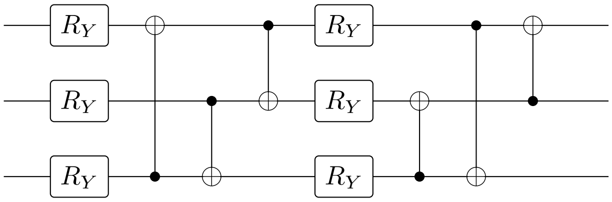


Figure 4.3: Sim15 ansatz for 3 qubits

All 3 considered ansätze are unitaries on n qubits. To convert a box in a string diagram into a VQC, some hyperparameters need to be provided for the ansatz. The first is the number of qubits representing each type associated with the box. A box could have multiple inputs and outputs of different types. The width of each type determines the number of qubits for the ansatz circuit which will replace this box. Further, the number of layers l of the ansatz must be specified. This affects the number of parameters, and properties such as the expressibility and entangling ability of the ansatz [39]. As shall be apparent in the results presented in Section 4.4, varying the number of layers can substantially affect the QNLP model's performance.

Table 4.1 compares some important statistics for a single layer of n-qubit circuits for each of the 3 considered ansätzes. Observe that the Sim ansätze are at least twice the size of the IQP ansatz along each statistic, owing to their double-layer design. Thus, for a fair comparison, we compare l layers of the Sim ansätze with an IQP ansatz of $2l$ layers. The statistics for this are shown in the table as 2×IQP. Henceforth we use the term IQP to refer to this double-layer ansatz.

	IQP	**2×IQP**	**Sim14**	**Sim15**
gates	$2n-1$	$4n-2$	$4n$	$4n$
2-qubit gates	$n-1$	$2n-2$	$2n$	$2n$
parameters	$n-1$	$2n-2$	$4n$	$2n$
depth	n	$2n$	$2n+2$	$2n+2$

Table 4.1: Per-layer statistics for n-qubit ansätzes.

4.2 Tasks

For a first comparison of the chosen ansätze, we consider the two tasks described in the work of Lorenz et al. [5]. These are the Meaning Classification (MC) task, and the Relative Pronoun (RP) task. We describe these tasks in detail below.

4.2.1 Meaning Classification Task

The meaning classification task involves assigning to each given sentence a domain belonging to a finite set $\mathcal{C}$. The task can be formally defined as:

Task 4.2.1 (Meaning Classification)**.** *Given a sentence S, return* 1 *if it relates to the domain 'food', and* 0 *if it relates to the domain 'IT'.*

The dataset for the MC task consists of 100 sentences, each belonging to one of the classes $\mathcal{C} = \{\textit{food, IT}\}$. Example sentences from the dataset can be seen below.

"skillful person prepares meal",	*food*
"skillful woman runs program",	*IT*

At present, it is not feasible to perform QNLP tasks with substantially larger datasets than this. Exclusive access to the majority of contemporary quantum devices is rarely available. Lorenz et al. report that running QNLP experiments on a shared quantum device can take in the order of 3 days [5]. When training using classical simulation, the tensor contraction operation is computationally expensive due to the dimension of the tensors being exponential in the number of qubits [4].

`noun_phrase`	→	`noun`
`noun_phrase`	→	`adjective noun`
`verb_phrase`	→	`verb noun_phrase`
`sentence`	→	`noun_phrase verb_phrase`

Table 4.2: Context Free Grammar for the MC task. Reproduced from [5].

Of the 100 sentences, 70 are used in training, and 30 are used as a test set. The 70 training sentences have a 39 / 31 distribution between the two classes. Each of the sentences is generated using a context-free grammar, presented in Table 4.2. The dataset is curated such that all words in the test set occur in the training set. The task for the QNLP model is one of supervised binary classification, where it learns a mapping from the sentence to the binary class labels. This task does not require significant grammatical understanding, since the primary difference between the two classes is in the vocabulary (*meal* vs. *program* for example). Thus it is reasonable to expect that even a simple "bag-of-words" style model which ignores grammar entirely would perform well on this dataset.

4.2.2 Relative Pronoun Task

The Relative Pronoun (RelPron), or RP task is also a binary classification task, but one which involves more grammatical structure than the MC task. Each entry of the dataset consists of a noun-phrase, and a label indicating whether the noun phrase contains a subject-relative clause, or an object-relative clause. Thus, the classes are $\mathcal{C} = \{$*subject-relative, object-relative*$\}$. Example noun-phrases for both these classes are given below:

"scientist that visit island",	*subject-relative*
"document that company publish",	*object-relative*

Here, the first noun-phrase is subject-relative, since the *head-noun*, "scientist", is the subject of the phrase. In the second noun-phrase, the head-noun "document" is the object of the phrase. This dataset is a modified version of the RelPron dataset, by Rimell et al. [41]. The original RelPron task described by the authors involved matching phrases to the terms they define. This was modified by Lorenz et al. [5] to the version we consider here.

Hyperparameter	Description	MC values	RP values
q_s	Number of qubits associated with sentence type	1	0
q_n	Number of qubits associated with noun type	$[1, 2, 3]$	1
l	Number of ansatz layers	$[1, 2, 3, 4]$	$[1, 2, 3, 4]$

Table 4.3: Ansatz hyperparameters for the MC and RP binary classification tasks.

Task 4.2.2 (Relative Pronoun)**.** *Given a noun phrase S, return* 1 *if it contains a subject-relative clause, and* 0 *if it contains an object-relative clause.*

The subset of the original RelPron dataset we use consists of 74 training phrases, and 31 test phrases. The training set has a 46 / 28 split between the two classes. The test dataset vocabulary consists of 74 words, of which 30 words (40.5%) do not occur in the training set. This limits the performance of the model, since there is no way for the model to learn parameterisations for words which do not occur in the training set. This problem is considered in greater detail in Chapter 6.

4.3 Experimental Setup

As described in the previous sections, both tasks that we consider involve binary classification, where each input consists of a single phrase or sentence. As a first step for both tasks, we parse the sentences using the *BobCatParser* class included in `lambeq` [17, 25, 27]. This generates a string diagram for each of the sentences in the train and test datasets. Then all cups from the resulting string diagrams are removed using the `remove_cups` method provided in `lambeq` (described in Section 3.2.2). The resulting string diagrams are then ready for parameterisation through an ansatz. There are three hyperparameters for the ansatz which must be specified. A description of these, and the values used in this experiment are provided in Table 4.3.

Since both tasks are based on binary classification, it is natural for the output type of each sentence's circuit to be a single qubit. Then, each class corresponds to a 0 or 1 measurement outcome in the computational basis. In the MC task, each sentence's output is of type s, thus we set the number of qubits associated with the sentence type (q_s) to 1, ($q_s := 1$) in all cases.

Analogously for the RP task, each phrase is a noun-phrase with output type n, thus $q_n := 1$.

Using a selected combination of ansatz hyperparameters, we convert the string diagrams into VQCs. These circuits are then used in an optimisation routine, which finds the optimum parameter assignment which minimises the loss function. We utilise the Binary Cross Entropy (BCE) loss, a loss function commonly used in classical machine learning for binary classification tasks. The BCE loss function is defined as:

$$\mathcal{L}_{BCE} = \frac{-1}{N} \sum_{i=1}^{N} y_i \cdot log(P_i(1)) + (1 - y_i) \cdot log(1 - P_i(1))$$

where the i'th sample has label $y_i \in \{0, 1\}$, and $P_i(1) \in [0, 1]$ indicates the model's predicted probability that sample i belongs to class 1. To train the classifier, the Simultaneous Perturbation Stochastic Approximation (SPSA) [33] optimiser was used. We use the SPSA optimiser hyperparameters described in the `lambeq` documentation[2]. The initial learning rate a is set to 0.05, the initial parameter-shift scaling c is set to 0.06, and the stability constant A is set to 1. 1000 epochs of SPSA were run, and the loss and accuracy on the testing set were evaluated on every 10th epoch. Each configuration of the ansatz was trained 5 times with different random seeds, and the results presented in the following section are the average across all runs for each circuit. Multiple runs are used since the gradient calculated by the SPSA procedure is approximately calculated, and quantum machine learning performance is known to be very sensitive to the initial parameter assignment [12, 42, 38]. Both these factors are impacted by the random seed. The landscape of the loss function for quantum machine learning, and the associated barren plateau problem are discussed in detail in Chapter 7. All experiments were run on a computer with two Intel Xeon Gold 6326 CPUs with 64 threads in total, running at a clock speed of 2.90 GHz, equipped with 256 GiB of RAM. In the following section, we present the results for the MC and RP tasks, when using the three considered ansätze.

[2] `https://cqcl.github.io/lambeq/tutorials/trainer_quantum.html`

4.4 Results

Table 4.4 shows the train and test accuracies achieved for the MC task, after 1000 epochs of training. We observe that when the noun type was represented using 1 or 2 qubits, the Sim ansätze achieved higher training accuracy in nearly all cases. Further, these ansätze also generalised well, outperforming the IQP ansatz across all layers for these noun widths. When representing the noun type with 3 qubits, we observe that the IQP ansatz achieves higher training accuracy than the Sim ansätze in all but one case. However, we observe that despite the higher training accuracy, the IQP ansatz models fail to generalise to the test set well. In nearly all cases, the Sim14 ansatz has the best performance on the test set. Note that in no case does the IQP ansätze achieve the 100% test accuracy achieved by the Sim ansätze in just 1 layer, with $q_n = 1$. Between the Sim14 and Sim15 ansätze, there is little difference for the noun widths of 1 and 2. Both ansätze achieve simultaneous train and test accuracies of 100% in multiple cases. In the $q_n = 3$ case however, Sim14 outperforms Sim15 on the test set for every single layer.

		Train			**Test**		
$\mathbf{q_n}$	**Layers**	**IQP**	**SIM14**	**SIM15**	**IQP**	**SIM14**	**SIM15**
1	1	0.904	**1.000**	**1.000**	0.821	**1.000**	**1.000**
	2	0.973	**1.000**	0.997	0.943	**1.000**	0.986
	3	**1.000**	0.994	**1.000**	0.993	**1.000**	**1.000**
	4	0.997	**1.000**	**1.000**	0.971	**1.000**	**1.000**
2	1	0.934	**1.000**	0.982	0.857	**0.986**	0.871
	2	0.988	**0.997**	**0.997**	0.907	0.971	**0.986**
	3	0.997	**1.000**	**1.000**	0.943	**1.000**	**1.000**
	4	**1.000**	0.991	0.997	0.900	**1.000**	0.993
3	1	0.976	0.976	**1.000**	0.886	**0.971**	0.950
	2	**1.000**	0.997	0.979	0.886	**0.971**	0.900
	3	**1.000**	0.991	0.916	0.943	**0.964**	0.857
	4	**0.997**	0.934	0.797	**0.929**	0.907	0.764

Table 4.4: Accuracy results for MC task. The highest train and test accuracy in each row is highlighted.

Figure 4.4 plots the loss and accuracy values while training the 4 layer model, with $q_n = 1$ for both the train and test sets. It is immediately apparent that all ansätze achieve very high accuracy on both train and test sets within a few 100 epochs. The loss values for both train

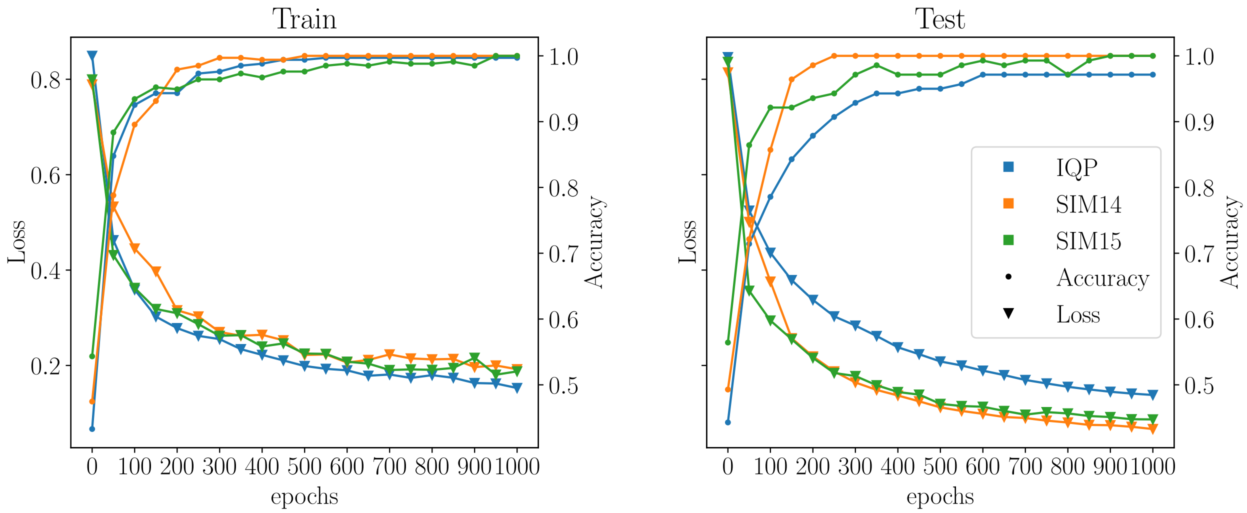


Figure 4.4: Training graphs for the MC task, for the 4-layer model with $q_n = 1$, for all considered ansätze, averaged across all 5 random initialisation.

and test sets continue to decrease throughout the experiment (1000 epochs), which establishes that the models are not overfitting to the train dataset. While there is a negligible difference between the accuracy on the training set for the three ansätze, the IQP ansatz achieves the lowest train loss. However, it is apparent that the IQP ansatz did not generalise well, since for the test dataset, its loss curve is higher than the curves for the Sim ansätze. The IQP ansatz's test accuracy also increase slower than the other two ansätze, and does not reach as high a maximum value. Both Sim ansätze have similar trends of loss and accuracy on both the train and test sets, with the Sim14 ansatz achieving 100% accuracy in fewer epochs than Sim15. Considering the timing information in Figure 4.5, we observe that both Sim ansätze take significantly longer to train than the IQP ansatz. In the worst case, it takes 2.9 times as long to train a 4-layer Sim14 circuit with $q_n = 3$, as it takes to train an IQP circuit of the same specification.

In summary, both Sim ansätze noticeably outperform the IQP ansatz for the MC task, at the cost of severely increased training time. The Sim ansätze both achieve 100% train and test accuracy. Of the 2 Sim ansätze, Sim15 takes only about half the time to train as Sim14

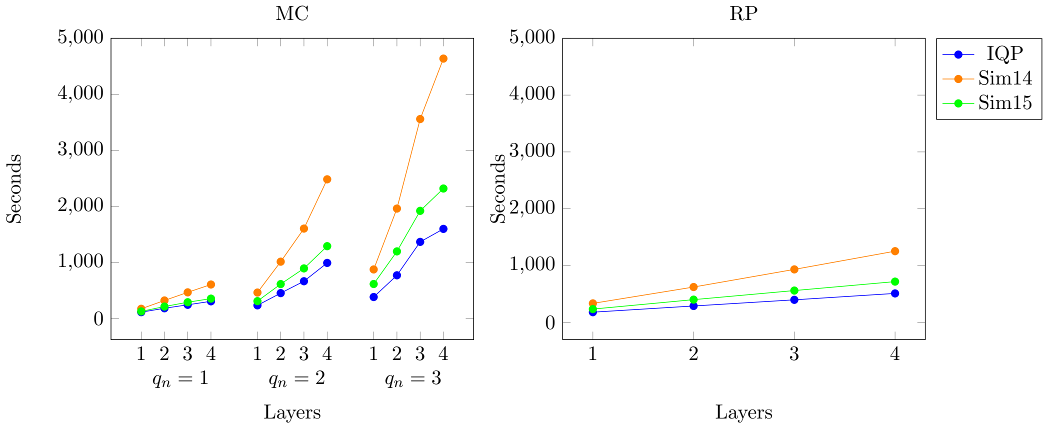


Figure 4.5: Training time for MC and RP tasks. Sim14 takes significantly more time to train than the other 2 ansätzes for both tasks, for all hyperparameter configurations.

	Train			**Test**		
Layers	**IQP**	**SIM14**	**SIM15**	**IQP**	**SIM14**	**SIM15**
1	0.930	0.986	**0.989**	0.845	0.910	**0.929**
2	0.983	0.986	**0.992**	0.890	0.897	**0.910**
3	0.980	0.986	**0.994**	0.903	**0.910**	**0.910**
4	0.989	**0.994**	0.980	0.923	**0.942**	0.890

Table 4.5: Accuracy results for RP task. The highest train and test accuracy in each row is highlighted.

and has only half as many parameters.

The train and test accuracy for the RP task after 1000 epochs can be seen in Table 4.5. Here, in all but the final case, the Sim15 ansatz achieves the highest accuracy on the training set. In each of these cases it generalises well and also achieves the highest test accuracy. For the 4-layer model, the Sim14 ansatz performs the best on both the training and the test sets. Note also, that this model achieves a 94.2% accuracy on the testing set, which is the highest of any of the models tested.

Figure 4.6 shows the loss and accuracy curves for the 4-layer models. As with the MC graph, all models achieve high accuracy within a few 100 iterations. The train and test loss continues to decrease throughout the experiment, indicating no overfitting. Once more, there is negligible difference in the test accuracy curves for all three ansätze. The IQP ansatz has the lowest (best) train loss curve, while the Sim14 ansatz has the highest (worst). Despite

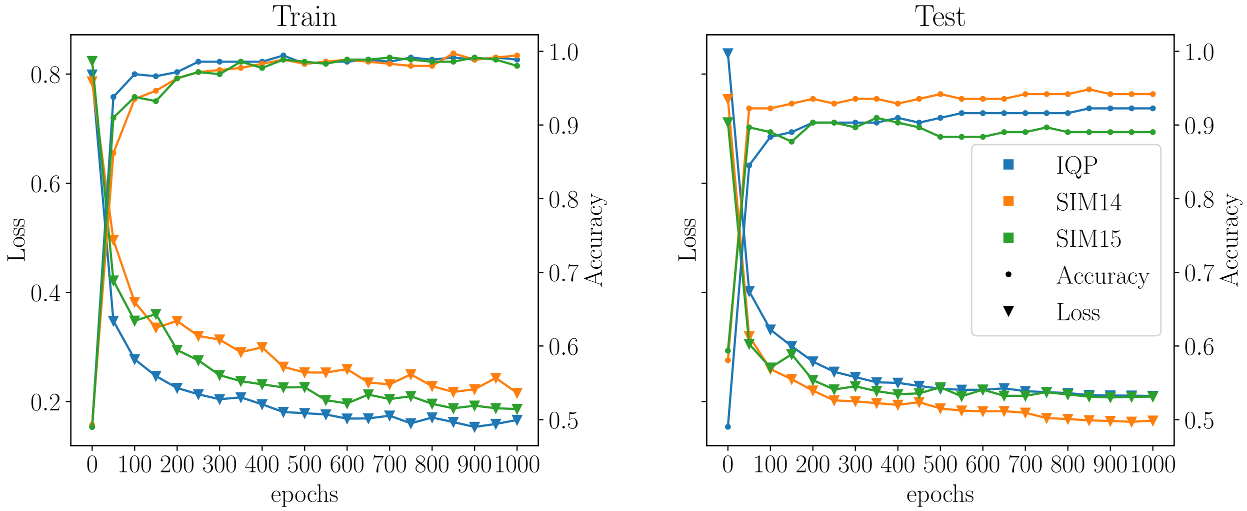


Figure 4.6: Training graphs for the RP task, for the 4-layer model for all considered ansätze, averaged across all 5 random initialisations.

this, Sim14 generalises to the test dataset the best, achieving the lowest test loss, and the highest test accuracy. As with the MC task, the Sim ansätze take longer to train than the IQP ansatz, with Sim14 taking up to 2.5 times the duration of IQP, and Sim15 taking up to 1.4 times the duration.

Thus in summary, we observe that Sim14 and Sim15 outperform the IQP ansatz for the MC and RP tasks. This performance improvement comes at the cost of an increase in training time. Of the 2 Sim ansätze, Sim15 takes far less time to train than Sim14, while achieving competitive test accuracies for the MC and RP tasks. Across both tasks, we observed that the Sim15 ansatz was able to match or beat the performance of the Sim14 ansatz on the test set in a majority of cases. This is despite the Sim15 circuit being significantly less expressible [39]. This suggests that for the MC and RP tasks, the difference in expressibility did not significantly impact classification performance.

4.4.1 Comparison with Prior Work

Both MC and RP tasks, as appear here were first considered by Lorenz et al. [5], using the IQP ansatz. In their work, the highest test accuracies achieved were 79.8% for the MC task using a single-layer IQP ansatz, and 72.5% for the RP task, using a 2-layer IQP ansatz. Our models outperform both these by a significant margin. There are some key differences in the experimental setup between our work and the experiments of Lorenz et al. The first difference is that while we use automatic parsing through *BobCatParser* class provided in `lambeq`, Lorenz et al. use knowledge of the datasets to semi-automatically generate string diagrams. For the MC task they use the production rules defined in Table 4.2. For the RP task they use pre-defined diagrams for the subject-relative and object-relative clauses respectively. This difference in diagram generation implies that the diagrams we generated are not necessarily identical to the diagrams generated in Sim et al. Further, the specification of the IQP ansatz for both tasks is subtly different. For the MC task, Lorenz et al. use a single-layer IQP model. Recall from Section 4.1 that we use a double-layer IQP block as the ansatz. Thus our 1-layer IQP model is only comparable with a 2-layer IQP ansatz. For all tasks, in the single qubit case we use the Euler decomposition ($R_X R_Z R_X$), described in Section 4.1. For the RP task, Lorenz et al. use a single R_X gate in the single-qubit case. Further to these differences, the hyperparameter settings we use for the SPSA optimiser are different. We attribute the difference in results between our work and Lorenz et al. to the differences in diagram generation, circuit structure, and subtle differences in optimisation hyperparameters. We note that the accuracies our models achieve for the MC task are very similar to those reported by Kartsaklis et al., using the IQP ansatz, and leveraging the "classical simulation of quantum pipeline" in `lambeq` [24].

Chapter 5

Paraphrase Identification Using QNLP

In the previous chapter, the tasks which we considered were relatively trivial, given the complexity of tasks that modern classical NLP can solve. In this chapter, we apply QNLP methods to a more challenging NLP task; namely the paraphrase identification task. This can be set as a binary classification problem as follows:

Task 5.0.1 (Praphrase identification)**.** *Given 2 sentences S_1 and S_2, return* 1 *if they have the same meaning, and* 0 *if they do not.*

While MC and RP tasks could be solved efficiently using either the vocabulary or the grammatical parse of the sentence, the paraphrase identification task cannot be reliably solved using either of these methods. Paraphrase identification requires a model that considers both the vocabulary and grammar of its input. In this chapter we describe a circuit to compare two sentence states (Section 5.2), and apply this construction to the paraphrase identification task. In Section 5.4, we present the first results for a QNLP model applied to the paraphrase identification task on real-world datasets.

5.1 Dataset

We evaluate QNLP performance for the paraphrase identification task on two different datasets. The first, which we will refer to as *PPDB*, is a subset of the Paraphrase DataBase [43]. We utilise the English phrasal version of the PPDB[1], from which the first 100 unique phrase pairs were extracted. These provide the positive samples for the paraphrase identification task. The original dataset does not contain any predefined negative samples. To generate a negative sample, we randomly selected 2 positive samples, and take one phrase from each to make a new pair, which is a negative sample with high probability. We generated as many negative pairs as positive pairs, to ensure a balanced dataset. Example pairs are shown below.

("applying the principle of", "the implementation of the principle of", $+$)
("achievement of the goals of the", "adopted in accordance with this", $-$)

The processed PPDB dataset has 93 training samples, 46 test samples, and an overall vocabulary of 295 words. As is apparent from the example shown above, negative samples are likely to have low overlap of vocabulary between the two sentences in the pair. Thus it is possible that a simple bag-of-words style model which ignores grammar may also achieve reasonable performance on this dataset.

To present a more semantically-rich challenge to the QNLP model, we consider the *MRPC* dataset, a subset of the Microsoft Research Paraphrase Corpus, by Dolan et al. [44]. Pairs in this dataset were obtained from multiple news sources, which were then annotated by human reviewers. We select all pairs from the original dataset where both sentences have at most 10 words. This is done to ensure that training the QNLP model can be achieved in a tractable amount of time, given the hardware at our disposal. Unlike PPDB, the original MRPC dataset also includes negative samples. The following are example pairs from the MRPC dataset.

("Only Intel Corp. has a lower dividend yield",
"Only Intel's 0.3 percent yield is lower", $+$)
("ISC and NASCAR officials declined to comment",
"NASCAR officials could not be reached for comment Tuesday", $-$)

[1] Available online at `http://paraphrase.org/#/download`

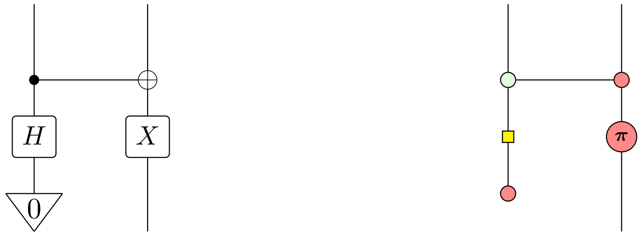
Compared with examples from the PPDB dataset, it is clear that negative samples are less obviously distinct. Specifically, a significant part of the vocabularies of the 2 sentences is shared. Thus, it is reasonable to expect that performance for the MRPC dataset will likely be lower than for the PPDB dataset. The processed MRPC dataset has 56 training samples, 32 test samples, and an overall vocabulary of 938 words. This is a large vocabulary given very few training samples, which is likely to adversely impact the performance of the model. Problems related to vocabulary are discussed further in Section 5.4, and in greater detail in Chapter 6.

5.2 State Comparison Circuit

In the tasks consider in Chapter 4, the circuit design choices were limited to the ansatz hyperparameters. We simply ensured that the output dimension of the sentence state was 1 qubit, allowing us to easily extract a predicted probability distribution over the 2 classes. The paraphrase identification task, however, does not involve a class assigned to a single sentence state, and requires the comparison of 2 states.

A natural method for comparing two quantum states is to use the Born rule: $|\langle\psi|\phi\rangle|^2$ (Section 2.1.4). A value of 1 indicates that the 2 states are equal, and a 0 indicates the states are orthogonal (or maximally different). This is the method for comparison suggested by Coecke et al. [1, 3]. To measure the Born rule probability, we can simply prepare states $|S_1\rangle$ and $|S_2\rangle$ for the 2 sentences to be compared, and evaluate the diagram $\langle S_1|S_2\rangle$ through tensor contraction. The square norm of the resulting complex number is then calculated to ensure we get a real value between 0 and 1.

A caveat to this is that quantum states generated from DisCoCat diagrams are in general sub-normalised states [5]. That is, they are not of unit magnitude. This is because in addition to unitary gates, they consist of post-selection and caps, which are non-unitary operations. A crucial corollary of this fact is that a state, when compared with itself will not yield 1 ($|\langle\psi|\psi\rangle|^2 \neq 1$). Thus, before measuring the Born rule probability, it is essential to normalise the states for the two sentences independently. However, it is often more convenient to prepare a single tensor network, contract it entirely, and only normalise the result as a final post-processing step. If we prepare the diagram $\langle S_1|S_2\rangle$ for some input parameterised states S_1 and S_2 generated using DisCoCat, the normalisation constant is not known apriori, as it depends on the parameter assignment (which changes in each step of the optimisation process.) To overcome this challenge, we propose a circuit C_{eq}, shown in Figure 5.1, to compare two potentially sub-normalised states S_1 and S_2.

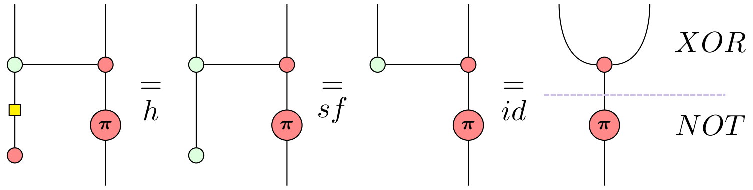


(a) Circuit representation

(b) ZX diagram representation

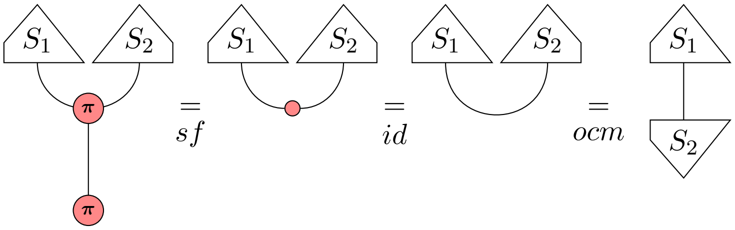
Figure 5.1: Circuit C_{eq} for comparison of two possibly sub-normalised single-qubit states, in circuit and ZX diagram representations.

Using simple rules of the ZX calculus, it can be shown that C_{eq} is equivalent to an *XNOR* gate, as generated by composing *NOT* ∘ *XOR*, using the logical circuit definitions provided on page 212 of Coecke and Kissinger [19].



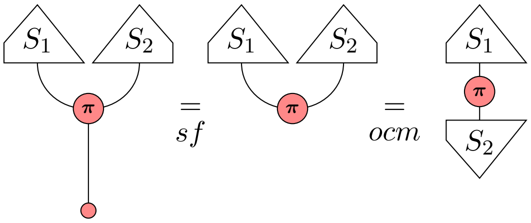
(5.1)

When applying C_{eq} to a pair of states, we conjugate one of the states, for reasons which are apparent in the diagrammatic representation in equations (5.2) and (5.3). Thus, the state resulting from applying C_{eq} to a pair of states $(|\psi\rangle, |\phi\rangle)$ is $C_{eq}(|\psi\rangle \otimes \overline{|\phi\rangle})$. When measuring (in the computational basis) the output of C_{eq} applied to two states yields output $|1\rangle$, the diagram reduces to the Born rule probability $\langle S_1|S_2\rangle$, as demonstrated in equation (5.2).



(5.2)

This implies that for normalised states, our circuit yields output 1 with the same probability as that yielded by the Born rule. Our proposed approach is therefore equivalent to the Born rule method, for normalised states. In the case where this measurement yields a $|0\rangle$ outcome, the diagram corresponds to a Born rule probability with a Pauli-X gate in the middle: $\langle S_1|X|S_2\rangle$.



(5.3)

Recalling that the Pauli-X gate is the quantum analogue of logical negation, we can represent this measure as $\langle \neg S_1|S_2\rangle$. This can be interpreted as the Born rule of the negation of a sentence S_1, compared with S_2. This is a reasonable conceptual negation of the Born rule probability, which can be used as the other element of the vector which we normalise to arrive at a probability distribution. Thus, the output of the comparison circuit is $[\langle \neg S_1|S_2\rangle, \langle S_1|S_2\rangle]$, which is the vector that we normalise to arrive at the prediction of the QNLP model. In this manner, we have presented a construction which allows us to generate a 2-dimensional unit vector representing the degree of similarity of 2 possibly sub-normalised states.

5.3 Experimental Setup

Once the datasets are filtered in the manner described in Section 5.1, Each sentence is parsed using the *DepCCGParser* class [28] provided in `lambeq`. This parser was chosen since the default *BobCatParser* class used for the MC and RP tasks failed to parse some input sentences from the MRPC dataset. The resulting string diagrams are then optimised using rewrite rules provided in `lambeq`. These are described in Appendix A.

Then, any pair in which both sentences do not yield the same final type is discarded. At this stage, the string diagrams are converted into quantum circuits using one of the ansätze. Once more, we use each of the 3 ansätze IQP, Sim14 and Sim15, described in Chapter 4. In all cases we assign a single qubit as the width of each type ($q_n := 1, q_s := 1$). The output type of each sentence circuit is thus a single qubit. This is both to ensure that the circuit for each sentence is compatible with the comparison circuit C_{eq}, and to bound the size of the generated circuits. At this stage, for each pair of sentence states $(|S_1\rangle, |S_2\rangle)$, we generate the complete diagram using the comparison circuit as $C_{eq}(|S_1\rangle \otimes \overline{|S_2\rangle})$. For each ansatz, we experiment with layers in $\{1, 2, 3\}$. Training with each hyperparameter combination is repeated 3 times with different random seed values. As before, the SPSA optimiser is run for 1000 epochs, with the optimiser hyperparameter configuration described in Section 4.3 (a: 0.05, c: 0.06, A: 1). In the following section we present the results of this experiment.

5.4 Results

Table 5.1 shows the train and test accuracies for the paraphrase identification task on the PPDB dataset. For 1 and 2 layers, we observe that Sim14 achieves the highest accuracy on both the training and test sets. For the 3-layer case IQP achieves the highest accuracy for the training set. However, this fails to generalise well to the test set, where Sim14 once more outperforms both other ansätze. Sim14 thus generalises the best for all circuit configurations. The highest accuracy on the test set is 70.3%, achieved by the Sim14 ansatz with 2 layers.

	Train			**Test**		
Layers	**IQP**	**SIM14**	**SIM15**	**IQP**	**SIM14**	**SIM15**
1	0.928	**0.935**	0.914	0.638	**0.688**	0.623
2	0.910	**0.943**	0.889	0.652	**0.703**	0.609
3	**0.953**	0.932	0.892	0.630	**0.645**	0.638

Table 5.1: Accuracy for paraphrase identification task on PPDB dataset. Highest train and test accuracy in each row are highlighted in bold.

The IQP ansatz outperforms Sim15 on the test set for the 1-layer and 2-layer cases, but achieves slightly lower accuracy for the 3-layer case. Across all layers and ansätze, the test

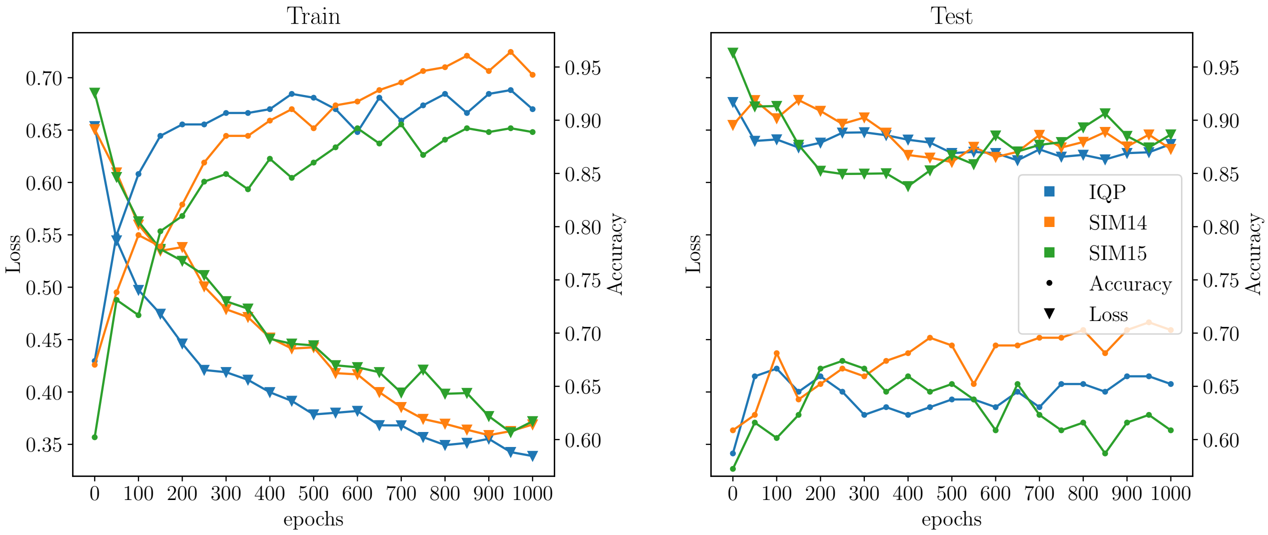


Figure 5.2: Training graphs for the PPDB task, for the 2-layer model for all considered ansätze, averaged across all 3 random initialisations.

accuracy is appreciably less than the train accuracy. Figure 5.2 shows the loss and accuracy curves for the 2-layer model, for all three ansätze. It is immediately apparent that the models do not converge to a solution as quickly as the models did for the MC and RP tasks (Figures 4.4, 4.6). The test accuracy increases in an erratic manner, compared to the near-monotonic increase demonstrated by models for the MC and RP tasks. The test loss is significantly higher than the training loss for all three ansätze, even at the end of training. The Sim15 ansatz appears to overfit the training dataset after a few 100 epochs, after which its test accuracy begins to decrease. This is not the case with the other two ansätze. This overfitting is despite Sim15 achieving the lowest train accuracy of the three. Since Sim15 has fewer parameters than Sim14 (Table 4.1), excess parameterisation can be ruled out as a cause. A more likely source of the poor generalisation is suboptimal training-time parameter initialisation, which is known to strongly affect QML model performance. Figure 5.3 plots the training time for each of the ansatz, as a function of the number of layers. As we observed for the MC and RP tasks, Sim14 takes significantly longer to train than the IQP ansatz. The Sim15 ansatz takes only marginally longer to train.

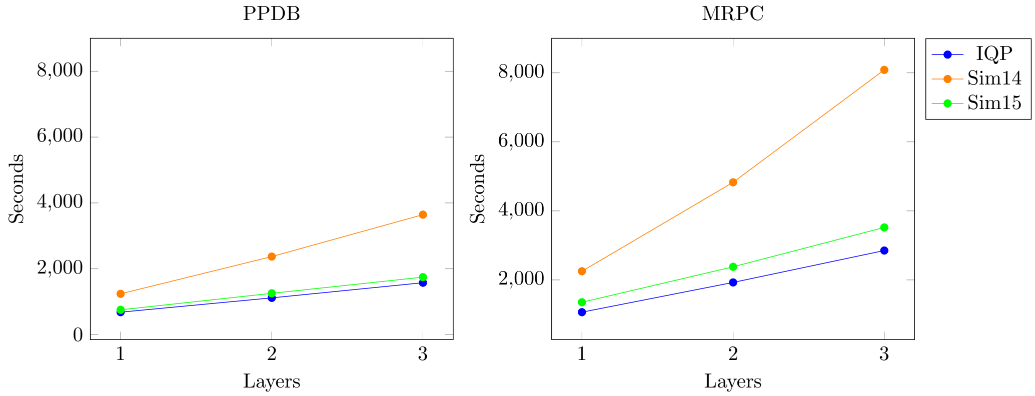


Figure 5.3: Training time for PPDB and MRPC paraphrase identification tasks. Sim14 takes significantly more time to train than the other 2 ansätze, for all hyperparameter configurations in both tasks.

Compared to the MC and RP tasks of the previous section, much lower test accuracy is achieved by even the best model for the PPDB paraphrase identification task. This reduced performance is particularly apparent in the training curves in Figure 5.2. We attribute this regression to two factors. The first is the very high percentage (50%) of words in the test set which were not present in the training set. This means that a large number of parameters at inference time were randomly initialised. This problem is considered in detail in Chapter 6. The second factor is the inherently difficult nature of the task. While reasonable performance can be achieved for the MC and RP tasks using rule-based classifiers, the paraphrase identification task requires a more complete semantic analysis of the input sentences.

Table 5.2 shows the accuracy for the MRPC dataset. It is immediately apparent that while the train accuracies are very high, all models generalise to the test set very poorly. For the training set, the IQP ansatz achieved the highest accuracy for the 2-layer and 3-layer cases, while the Sim14 ansatz has the highest accuracy for the single-layer case. Each of the ansätze achieves a train accuracy of more than 95% for at least one layer configuration. In contrast, the highest test accuracy is only 55.2% achieved by the 3-layer Sim15 model. On a test dataset with a 53-47% class split, this accuracy is only marginally better than random chance. Figure 5.4 plots the training curves for the MRPC dataset, for the 3-layer model for each ansatz. As in the case of the PPDB dataset, the test accuracy increases erratically, at

	Train			**Test**		
Layers	**IQP**	**SIM14**	**SIM15**	**IQP**	**SIM14**	**SIM15**
1	0.964	**0.982**	0.976	0.490	**0.521**	0.490
2	**0.988**	0.911	0.899	**0.479**	0.458	**0.479**
3	**0.982**	0.756	0.798	0.521	0.542	**0.552**

Table 5.2: Accuracy for paraphrase identification task on MRPC dataset. Highest train and test accuracy in each row are highlighted in bold.

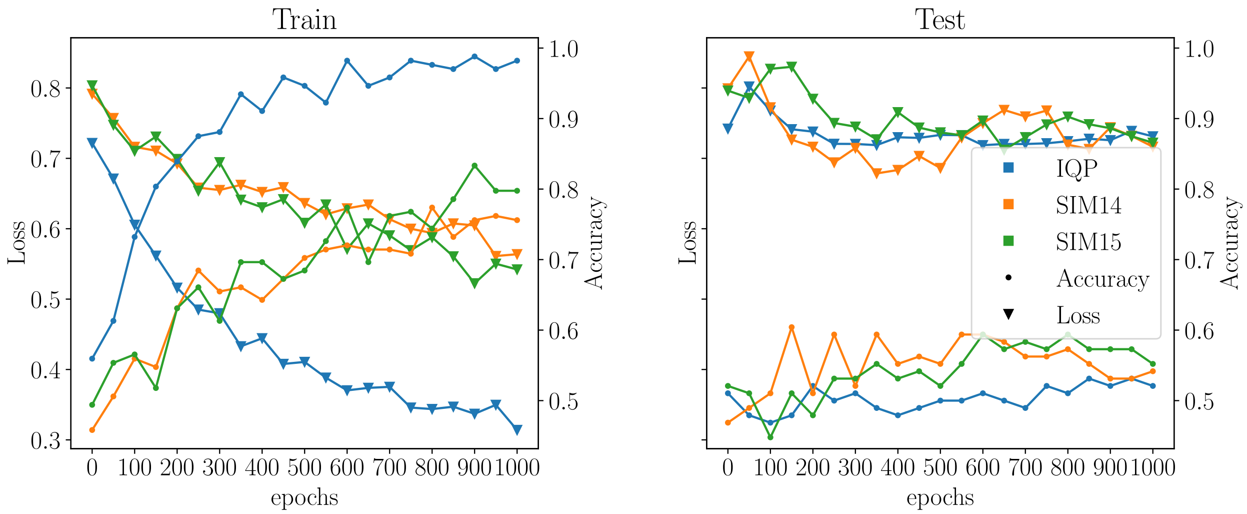


Figure 5.4: Training graphs for the paraphrase identification task on the MRPC dataset, for the 3-layer model for all considered ansätze, averaged across all 3 random initialisations.

a very low rate compared to the training accuracy. The IQP ansatz's train accuracy curve is much higher than the curve for the other two ansätze, but this fails to generalise to the test set where its accuracy increases the slowest.

Figure 5.3 shows the time taken to train models for the MRPC task. As for all examples thus far, Sim14 takes significantly longer to train than the other two ansätze. The Sim15 ansatz takes marginally longer than IQP to train. Of all QNLP models discussed in this work, the 3-layer Sim14 model for the MRPC task takes the longest to train: nearly 2 hours and 15 minutes.

The best test accuracy achieved for the MRPC dataset is appreciably worse than the test accuracy achieved for the same task on the PPDB dataset. There are 2 likely causes for the deterioration in performance. First, the PPDB dataset's negative samples are phrase pairs which are more obviously mismatched. That is, the two sentences in a pair differ in meaning significantly, often having nearly disjoint vocabularies. The negative samples in the MRPC dataset, meanwhile, are carefully curated pairs which can have similar, but not equivalent meanings. Negative pairs from the MRPC dataset often share significant proportions of their vocabulary, defeating trivial bag-of-words style solutions. The second potential cause is the increased percentage of out-of-vocabulary (OOV) words: words in the test set which were not seen during training. While the PPDB dataset had an OOV percentage of 50%, the value for the MRPC dataset is 81.5%. This is a very high proportion, indicating that only a small minority of words in the test dataset have trained parameter assignments available. This problem, and a novel solution are discussed in detail in Chapter 6.

5.5 Summary of Ansatz Comparison

In Table 5.3, we order each ansatz by the maximum accuracy achieved on the test set for each task and dataset, from Chapters 4 and 5. We consider the highest accuracy achieved for any hyperparameter configuration (q_n, q_s, layers) of the model. This allows us to compare all three ansätze in a summary manner, by considering the best test accuracy each ansatz can achieve, with the optimum hyperparameter configuration. For 3 of the 4 tasks and datasets, the IQP ansatz achieved the lowest test accuracy, and did not achieve the highest accuracy on any dataset. Consequently in each of the tasks considered in our work, test accuracy can be increased by using one of the Sim ansätze instead of the IQP ansatz. Between the two Sim ansätze, the choice is less obvious. The Sim14 ansatz achieves the highest test accuracy on 3

Task / Dataset	**Test**	**Accuracy**	**Ordering**		
MC	IQP	$<$	Sim14	$=$	Sim15
RP	IQP	$<$	Sim15	$<$	Sim14
PPDB	Sim15	$<$	IQP	$<$	Sim14
MRPC	IQP	$<$	Sim14	$<$	Sim15

Table 5.3: Ordering of ansätze by the highest test accuracy achieved for each task and dataset, with any hyperparameter configuration.

of the 4 datasets, while Sim15 achieves the highest accuracy on 2 datasets. The Sim15 ansatz achieves the lowest performance for the paraphrase identification task on the PPDB dataset, while the Sim14 ansatz never performs the worst of the three considered ansätze. For optimal test accuracy, the Sim14 ansatz appears to be the most appropriate choice for a majority of cases. Recall however that it takes significantly longer to train the Sim14 ansatz than Sim15, for each of the considered tasks. Thus, Sim15 may provide a balance between performance and training time.

Chapter 6

Handling Unknown Words in QNLP

A problem faced by QNLP models which requires special attention is the handling of unknown, or Out-Of-Vocabulary (OOV) words. As discussed in previous sections, training a QNLP model involves learning an optimal assignment for multiple parameters associated with each word in the dataset, such that the loss value is minimized on the training set. Then, when a test sample's output must be predicted, we generate a VQC for the input sentence, and for each word in the new sentence, assign to its associated parameters values learned during training. An important consideration that this description omits is the handling of words in the test set which did not appear in the training set. For these words, no parameter assignment is readily available. This problem is called the out-of-vocabulary, or OOV problem. In this chapter we consider possible mitigation strategies for the OOV problem, and present experimental results comparing these.

6.1 Background

The OOV problem is not endemic to the quantum methodology of NLP, and several solutions to the problem can be found in the domain of classical NLP. Often, the OOV problem is mitigated by either removing all words below a certain frequency, or replacing them with a dedicated '*UNK*' token to indicate an unknown word. Another approach in NLP which mitigates the OOV problem is the use of pretrained *word embeddings*. These are typically task-agnostic vector word representations, which are generated for a corpus with a large vocabulary in an unsupervised manner. These embeddings can then be used as input to downstream task-specific NLP models [45, 46].

6.1.1 Word Embeddings

A word embedding is a mapping which associates with each word w in a vocabulary V, a vector in $\mathbb{R}^d$. Modern word embeddings are dense word representations ($d < |V|$), learned from a corpus, the vocabulary of which is expected to cover the train and test vocabularies with high likelihood.

One common method to generate word embeddings is the *Continuous Bag Of Words* (CBOW) model [47]. This model attempts to predict a masked word, given the words surrounding it in an input sentence (its context). In attempting to solve this task, the model generates an embedding for each context word. It is this embedding which is used as a vector representation. Another method to generate word embeddings is the *skip-gram* model [47]. This training approach is roughly the inverse of the approach used in CBOW. Here, rather than attempting to predict a word from its context, the model aims to predict the context of a word. That is, given an input word, it attempts to predict the surrounding words in the corpus. Often, the skip-gram is weighted, assigning higher importance to context words nearer the current word. Word2Vec [47], and FastText [48] are two popular word embeddings from the literature that use the CBOW and skip-gram models. Alternative methods for generating word embeddings, such as global co-occurrence statistics (GloVe [49]), and context-aware methods using transformers (BERT [50]) are also widely used.

A common theme across multiple word embeddings is that they succinctly capture relationships between words in vector operations [51, 52]. Typically, synonymous words and words with similar themes have low distance in the embedding space. Word embeddings can also capture relations between words more complex than synonymy. Mikolov et al. show that relations such as (France - Paris $\approx$ Italy - Rome), (Einstein - scientist $\approx$ Picasso - painter) hold for Word2Vec skip-gram embeddings, indicating that the embedding can capture complex relations such as professions, and capitals of countries [47]. This semantic richness of neural word embeddings makes them suitable as task-agnostic embeddings which can be used as input to a task-specific model [46].

6.1.2 FastText Embeddings

FastText, by Bojanowski et al. is a method to generate vector representations for words from their subwords [48]. FastText treats each word as a set of its character n-grams, which are substrings of n characters of the word. For example, $\{$<*h, he, el, ...*$\}$ are 2-grams of the word "*hello*". It then learns a vector representation for each of the character n-grams through a skip-gram model. The vector representation of the word then, is obtained by taking the sum of its n-gram vectors. Because of the n-gram method of generating word-vectors, FastText is also able to generate vector representations for words that were not in its training corpus. This is achieved by decomposing the novel word into its n-grams, which were present in the training corpus with high likelihood. This is the main feature which motivated us to use the FastText embedding method in our approach. Through the use of FastText, we ensure that the classical embedding method does not suffer from the OOV problem, and thus embeddings are available for all words. Further, FastText embeddings pretrained on large corpora are available online [53].

6.2 OOV Mitigation Strategies

We propose a method that extends the word embeddings found in classical NLP to the QNLP case. The pretrained FastText embedding provides a largely task-agnostic dense embedding of a large vocabulary $V_{FT} \supseteq V_{train} \cup V_{test}$. The final parameter assignment, θ_{train} generated by an optimised QNLP model provides a task-specific embedding for a limited vocabulary V_{train}. Our proposed method suggests training a *Multi-Layer Perceptron* (MLP) [54], to learn a map from the FastText embeddings of the training vocabulary $\mathrm{FT}(V_{train})$, to the QNLP model's trained parameter assignment θ_{train}. The expectation then is that this MLP will generalise to unseen vocabulary, such that it can generate a predicted parameter assignment for the test set

$\widehat{\theta}_{test}$. Broadly, the method generates task-specific embeddings for new words from their known task-agnostic embeddings. Symbolically, the MLP learns mapping FT$(V_{train}) \xrightarrow{MLP} \theta_{train}$, and generalises to FT$(V_{test}) \xrightarrow{MLP} \widehat{\theta}_{test}$. Our method is more formally described in algorithmic representation below.

Algorithm 3 QNLP Model Training (OOV Embedding-Enhanced)

```
procedure QNLP-TRAIN-OOV($x_{train}, y_{train}$)
    $diagrams \leftarrow$ [TEXT-TO-DIAGRAM($x_i$) for each $x_i$ in $x_{train}$]
    $optimised\text{-}diagrams \leftarrow$ [REWRITE($d_i$) for each $d_i$ in $diagrams$]
    $circuits \leftarrow$ [ANSATZ($d_i$) for each $d_i$ in $optimised\text{-}diagrams$]
    for each $w_i \in V_{train}$ do
        $\overrightarrow{\theta^0_i} \leftarrow$ FT($w_i$)
    end for
    $\theta^0 \leftarrow \langle \overrightarrow{\theta}^0_1, \overrightarrow{\theta^0_2}, ..., \overrightarrow{\theta}^0_{|V_{train}|} \rangle$
    for $t \leftarrow 1$ to $epochs$ do
        $\theta^t \leftarrow$ SPSA-ITERATION($circuits$, $y_{train}$, $\mathcal{L}$, $\theta^{t-1}$)
    end for
    $param\text{-}map \leftarrow \{w_i : \overrightarrow{\theta}^{epochs}_i$ for each $w_i \in V_{train}\}$
    $MLP_{embed} \leftarrow$ TRAIN-EMBEDDING-MLP($\theta^0, \theta^{epochs}$)
    return $MLP_{embed}$, $param\text{-}map$
end procedure
```

Algorithm 3 describes our proposed method for training a QNLP model capable of overcoming the OOV problem. This algorithm is a modified version of the "naive" QNLP model training method presented in Algorithm 1. The first steps, up to preparing the parameterised circuits remain unchanged. The first difference from the naive algorithm is in parameter initialisation. For each $w_i \in V_{train}$, we generate an initial parameter assignment by querying its FastText embedding[1] $\overrightarrow{\theta^0_i} \leftarrow FT(w_i)$. Using this initialisation, we train the QNLP model using an optimizer to minimize the value of an objective function, as before. This yields a trained parameter assignment $\overrightarrow{\theta}^{epochs}_i$ for each word w_i.

Then, we train a multilayer perceptron MLP_{embed} to learn the mapping from the FastText embedding to the learned parameter assignment: $\forall w_i \in V_{train} \ . \ FT(w_i) \xrightarrow{MLP_{embed}} \overrightarrow{\theta}^{epochs}_i$. A trained model thus consists of the trained parameter assignment for V_{train}, stored in *param-map*, and the trained multilayer perceptron MLP_{embed}.

[1] This embedding may need to be truncated to $|\overrightarrow{\theta_i}|$. This is omitted from the algorithm for simplicity.

Algorithm 4 QNLP Model Evaluation (OOV Embedding-Enhanced)

```
1: procedure QNLP-EVAL-OOV($MLP_{embed}, \textit{param-map}, x_{test}, y_{test}$)
2:     for each $w_i \in V_{test} \setminus V_{train}$ do
3:         $\widehat{\theta}_i \leftarrow MLP_{embed}(\mathrm{FT}(w_i))$
4:         $\textit{param-map}[w_i] \leftarrow \widehat{\theta}_i$
5:     end for
6:     return QNLP-EVAL($\textit{param-map}, x_{test}, y_{test}$)
7: end procedure
```

Algorithm 4 describes the process for evaluating the OOV-aware QNLP model on a test set. For each word which was in V_{train}, we use the parameter assignment learned by the QNLP model directly, from *param-map*. For each OOV word $w_i \in V_{OOV} = V_{test} \setminus V_{train}$, we generate a predicted parameter assignment by querying its FastText embedding, and providing it as input to the MLP. This is done in line 3 of Algorithm 4. In this manner, a parameter assignment can be generated for each word in the test set, whether or not the word appeared in the training set. *param-map* is updated with the parameters generated by the MLP, and this can be used to invoke the naive model evaluation method *QNLP-EVAL* described in Algorithm 2. We compare 2 variations of this approach and 2 more naive methods which we treat as baselines.

- **Embed-NN**: This is the method described above, using the MLP to generate parameter assignments from FastText embeddings.

- **Embed-raw**: This is a simplified version of *Embed-NN*, where we replace the MLP with an identity function. That is, we directly use the truncated FastText embedding as the parameter assignment for the OOV words, without any intermediate processing.

- **Random**: This strategy involves generating a uniformly random parameter assignment for OOV words. This is the default behaviour in `lambeq`. We treat this strategy as a baseline, as it is a reasonable choice in absence of further information.

- **Zero**: This strategy trivially assigns 0 to all parameters for OOV words. In the previous 3 strategies, each OOV word is assigned a unique embedding (always for *Embed-raw*, and with high probability for *Embed-NN* and *Random*.) In contrast, the *Zero* strategy assigns all OOV words equal embeddings. Some words can still be distinguished by their pregroup type, which in turn affects their dimension. Two OOV words with equal dimensions are indistinguishable under the *Zero* OOV strategy. This provides some intuition that this strategy may not be as performant as the previous three.

Note that in each of these cases, the training-time initialisation is still the FastText embedding. Below, we discuss some implementation details of our approach, and the experimental setup used to evaluate the proposed strategies.

6.3 Experimental Setup

Of the 4 tasks considered previously, we know that the MC task does not suffer from the OOV problem. This is because $V_{test} \subseteq V_{train}$, by design. Thus, we consider the RP task and the paraphrase identification task on the PPDB and MRPC datasets. For each of these tasks, we make no changes to the dataset and circuit design from the naive models of Chapters 4 and 5. We exclusively change parameter assignment at two stages for each:

1. **Training time**: Previously, parameters were randomly assigned at training time. In this experiment, we initialise parameters using the FastText embedding as described in Algorithm 3. This implies that the *Random* initialisation strategy will likely not yield the same results as those shown in previous sections. This is because QML performance is known to be dependent on parameter initialisation [13, 42, 38].

2. **Inference time**: At inference time, for words in V_{train}, we use the QNLP model's learned parameter assignment. For OOV words, we employ one of the 4 strategies described above.

We apply all 4 strategies for each of the 3 ansätze from Chapter 4, for circuits of up to 3 layers. An important practical consideration when implementing our OOV model is the dimension of the FastText embedding employed. For our experiments, we begin with the 300-dimension embedding for English available online[2]. This model is pretrained on data

[2] `https://fasttext.cc/docs/en/crawl-vectors.html`

from CommonCrawl and Wikipedia. The number of parameters for each word depends on the ansatz chosen, and the number of layers thereof. For each ansatz, we determine the maximum number of parameters d_{max} needed for any one word, for a specific number of layers. We then reduce the original 300 dimension embedding to the d_{max} value for the 3-layer circuit, using Principal Component Analysis (PCA) [55]. While this means that the reduced FastText embedding matches the dimension required for the word with most parameters in the 3-layer case, multiple words with fewer parameters likely exist in the dataset. In these cases, we simply truncate the reduced embedding to the required dimension. Further, since d_{max} for the 3-layer circuit is greater than the d_{max} values for the 1 and 2-layer circuits, the embedding will need to be truncated for these circuits. This is a tradeoff we make for simplicity, which is discussed further in the conclusion of the chapter.

For the MLP used in the *Embed-NN* model, we implement a dense neural network in Keras [56]. For each ansatz configuration, this model has input dimension equal to the dimension of the FastText embedding used in the experiment (3-layer d_{max}), and has output dimension equal to d_{max} for the specific ansatz configuration. There is one hidden layer, the dimension of which is the average of the input and output dimensions of the MLP. The *tanh* activation function is used for both layers. *tanh* is a suitable choice for the output layer since it has a range of $(-1, 1)$, implying that the generated parameters will yield rotations in the complete range $(-\pi, \pi)$. The mean absolute error is used as the loss function, and the model is trained using the Adam optimiser, with a learning rate of 0.001, for 120 epochs.

6.4 Results

Table 6.1 shows the test accuracies for each of the OOV strategies discussed in the previous section. In 5 out of 9 model configurations, the highest test accuracy is achieved by the *Embed-NN* model. Interestingly, the *Embed-raw* model also performs very well, achieving the highest test accuracy in 5 cases. This is an interesting result since the FastText embeddings are derived from a classical CBOW model, and there is no apriori reason to expect them to perform well as VQC parameter assignments without further processing.

RP: Accuracy

Ansatz	Layers	Train	Embed-NN	Embed-raw	Random	Zero
IQP	1	0.860	**0.720**	**0.720**	0.693	**0.720**
	2	0.874	0.731	**0.753**	0.717	0.731
	3	0.928	**0.785**	0.774	0.729	**0.785**
SIM14	1	0.968	0.763	**0.806**	0.754	0.731
	2	0.950	**0.796**	0.785	0.763	0.742
	3	0.995	0.796	**0.806**	0.769	0.785
SIM15	1	0.991	0.785	**0.828**	0.755	0.785
	2	0.995	**0.839**	0.785	0.794	**0.839**
	3	1.000	**0.828**	0.806	0.797	0.763

Table 6.1: Accuracies for multiple OOV strategies, for the RP task. The highest test accuracy in each row is highlighted in bold.

RP: BCE Loss

Ansatz	Layers	Train	Embed-NN	Embed-raw	Random	Zero
IQP	1	0.287	**0.440**	0.442	0.479	0.456
	2	0.275	0.425	**0.420**	0.464	0.443
	3	0.197	0.394	**0.388**	0.453	0.418
SIM14	1	0.145	0.400	**0.367**	0.468	0.416
	2	0.164	**0.396**	0.452	0.466	0.465
	3	0.092	0.397	**0.382**	0.435	0.420
SIM15	1	0.078	0.508	0.492	0.452	**0.388**
	2	0.075	0.424	**0.413**	0.417	0.496
	3	0.056	**0.312**	0.439	0.425	0.411

Table 6.2: Binary Cross Entropy loss values for multiple OOV strategies, for the RP task. The lowest test loss in each row is highlighted in bold.

For 3 model configurations, *Zero* initialisation equals the test accuracy of the highest performing strategy. For each of the 9 model configurations, the highest accuracy is achieved by either the *Embed-raw* or the *Embed-NN* initialisation strategies. Table 6.2 shows the values of the BCE loss for each of the initialisation strategies on the test set. For 8 out of 9 model configurations, the *Embed-raw* and *Embed-NN* strategies achieve the lowest loss values. This is largely consistent with the results for accuracy.

Of the initialisation strategies considered for the RP task, the highest test accuracy is 83.9%, achieved by the Sim15 ansatz with 2 layers, using the *Embed-NN* and *Zero* strategies. In the results of Chapter 4, the highest accuracy achieved for the RP task, using a naive QNLP

PPDB: Accuracy

Ansatz	Layers	Train	Embed-NN	Embed-raw	Random	Zero
IQP	1	0.817	**0.725**	**0.725**	0.658	0.703
	2	0.882	**0.717**	**0.717**	0.646	0.681
	3	0.921	0.674	**0.688**	0.656	0.681
SIM14	1	0.932	0.688	0.710	0.687	**0.717**
	2	0.961	0.739	0.761	0.693	**0.783**
	3	0.953	**0.732**	0.725	0.703	0.725
SIM15	1	0.932	**0.804**	0.739	0.658	0.783
	2	0.939	0.732	0.645	0.666	**0.754**
	3	0.946	**0.710**	0.703	0.665	**0.710**

Table 6.3: Accuracies for multiple OOV strategies, for the paraphrase identification task on the PPDB dataset. The highest test accuracy in each row is highlighted in bold.

model with random training-time parameter initialisation was 94.2%, which is significantly higher than any of the OOV strategies discussed here. It is known that QML model performance is very sensitive to parameter initialisation (at training time) [38, 42, 13]. Thus, using FastText embedding as a training-time initialisation is likely suboptimal for this dataset and task, yielding a trained parameter assignment that generalises poorly to the test set. Further, the 94.2% test accuracy achieved by the naive model is already very high, leaving minimal room for improvement.

Table 6.3 lists the test accuracies for various parameter initialisation strategies for the paraphrase identification task on the PPDB dataset. For 5 out of the 9 model configurations, the *Embed-NN* strategy achieves the highest accuracy on the test set. For the IQP models, *Embed-NN* and *Embed-raw* achieve very similar accuracies for all 3 model sizes. Surprisingly, the *Zero* initialisation strategy performs particularly well on this dataset, achieving the highest test accuracy in 4 model configurations. *Zero* also outperforms *Random* in each case. In total, for 6 out of 9 configurations, the highest accuracy is achieved by one of the *Embed* strategies, while the *Zero* strategy achieves the highest accuracy for the remaining 3.

Table 6.4 lists the BCE loss for all the parameter initialisation strategies. In contrast to the accuracies in Table 6.3, here we observe that the *Embed-NN* and *Embed-raw* strategies achieve the lowest values of the loss function in all cases. Despite having a test loss of more than 3 times that of the Embed strategies, the *Zero* strategy achieves very high test accuracy for

PPDB: BCE Loss

Ansatz	Layers	Train	Embed-NN	Embed-raw	Random	Zero
IQP	1	0.365	0.610	**0.598**	0.612	1.877
	2	0.281	**0.527**	**0.527**	0.625	1.929
	3	0.274	0.614	**0.606**	0.638	1.897
SIM14	1	0.239	0.598	**0.595**	0.596	1.841
	2	0.187	**0.553**	0.564	0.593	1.786
	3	0.203	**0.570**	0.574	0.580	1.829
SIM15	1	0.225	**0.508**	0.551	0.628	1.783
	2	0.196	**0.581**	0.661	0.639	1.872
	3	0.197	**0.609**	0.624	0.671	1.903

Table 6.4: Binary Cross Entropy loss values for multiple OOV strategies, for the paraphrase identification task on the PPDB dataset. The lowest test loss in each row is highlighted in bold.

multiple model configurations. This was not the case for the RP task, where the embedding-based strategies outperformed other strategies on both loss and accuracy. While loss functions are chosen such that accuracy and loss demonstrate an inverse relation on average, they do not share a strict monotonic relation.

This can be demonstrated with a trivial example. Consider the ground truth values of a classification task: $[(0, 1), (0, 1)]$. Then, consider two models which each produce the predicted class distribution $[(0.51, 0.49), (0.51, 0.49)]$ and $[(0.49, 0.51), (0.99, 0.01)]$ respectively. The BCE loss for the models will then be 0.71 and 2.64 respectively, while their accuracies are 0% and 50%. The second model achieved higher accuracy, but by being very confident in an incorrect prediction, had a very high loss value. It is thus entirely feasible for a model with higher loss on the test set to outperform one with lower loss.

The highest test accuracy for the PPDB paraphrase identification task is 80.4% achieved by the *Embed-NN* strategy for the single layer Sim15 ansatz. The highest accuracy achieved by the naive model in Chapter 5 was 70.3%. Thus, our embedding strategy has yielded a 10+ percentage point improvement over the best model that did not specifically handle the OOV problem.

Table 6.5 shows the test accuracy for multiple OOV strategies applied to the paraphrase identification task on the MRPC dataset. For the IQP ansatz, in 2 out of 3 cases the *Embed-*

MRPC: Accuracy

Ansatz	Layers	Train	Embed-NN	Embed-raw	Random	Zero
IQP	1	0.976	**0.604**	0.563	0.501	0.573
	2	0.988	**0.677**	0.604	0.509	0.656
	3	0.911	0.510	0.458	0.513	**0.573**
SIM14	1	0.940	0.594	**0.615**	0.538	**0.615**
	2	0.988	0.625	0.635	0.523	**0.667**
	3	0.976	**0.625**	0.573	0.541	0.594
SIM15	1	0.798	0.625	**0.656**	0.560	0.625
	2	0.863	0.542	0.500	**0.544**	0.531
	3	0.952	0.552	**0.563**	0.526	0.521

Table 6.5: Accuracies for multiple OOV strategies, for the paraphrase identification task on the MRPC dataset. The highest test accuracy in each row is highlighted in bold.

NN strategy achieved the highest test accuracy, while the *Zero* strategy achieved the best performance for the 3-layer model. For Sim14, the *Zero* strategy achieved the best test performance for 1 and 2 layers, while the *Embed-NN* strategy outperformed it for 3 layers. The *Embed-raw* strategy performed the best for Sim15 models of 1 and 3 layers. The *Random* strategy only marginally outperforms *Embed-NN* for the 2-layer case. In total, the *Embed-raw* and *Embed-NN* strategies, between them achieve the highest test accuracy for 6 out of 9 models. The highest test accuracy overall is 67.7%, achieved by the *Embed-NN* strategy, for a 2-layer IQP model. This represents a 12.5 percentage point improvement over the 55.2% achieved by the best naive model (Section 5.4). This performance is appreciably better than random chance and brings QNLP performance for the MRPC task in line with the naive QNLP model's performance on the PPDB task.

6.4.1 Comparison with Naive Models

Table 6.7 presents a comparison of the models presented in this chapter with models from the previous 2 chapters, which did not consider the OOV problem explicitly (naive models). For each dataset, the test accuracy for the OOV models is the maximum across all OOV strategies and ansatz configurations. For the RP task, the model described in Chapter 4 outperforms even the best OOV-aware model by more than 10pp. Observe that the naive model performed particularly well on this task, despite the 40.5% OOV fraction. Clearly,

MRPC: BCE Loss

Ansatz	Layers	Train	Embed-NN	Embed-raw	Random	Zero
IQP	1	0.176	**0.648**	0.801	0.779	3.633
	2	0.235	**0.730**	0.804	0.764	3.510
	3	0.363	0.814	0.876	**0.768**	3.628
SIM14	1	0.271	0.784	**0.698**	0.738	3.577
	2	0.190	0.712	**0.703**	0.754	3.460
	3	0.248	**0.685**	0.705	0.734	3.529
SIM15	1	0.370	0.823	0.733	**0.715**	3.503
	2	0.306	0.953	0.975	**0.731**	3.661
	3	0.240	0.909	0.893	**0.746**	3.649

Table 6.6: Binary Cross Entropy loss values for multiple OOV strategies, for the paraphrase identification task on the MRPC dataset. The lowest test loss in each row is highlighted in bold.

Task	OOV%	Max Naive Accuracy	Max OOV Accuracy	Improvement (pp)	Max OOV Strategy
RP	40.50%	94.20%	83.90%	−10.30%	Embed-NN, Zero
PPDB	50%	70.30%	80.40%	+10.10%	Embed-NN
MRPC	81.50%	55.20%	67.70%	+12.50%	Embed-NN

Table 6.7: Improvement in accuracy obtained through use of OOV strategies described in this section. Improvement is in percentage points (pp).

the OOV problem did not appreciably impact the performance of this model. Any potential gains in performance from OOV strategies were neutralised by reduction in performance likely caused by the unsuitability of the FastText embedding as a training-time parameter initialisation for this task.

For the paraphrase identification task, on the other hand, we observe significant improvement for both datasets. The naive model's 70.3% accuracy on the PPDB dataset is improved to 80.4%, which is a very high accuracy given the inherently difficult nature of the task. For the MRPC dataset, the severity of the OOV problem (81.5%) contributed significantly to the poor performance of the naive model, which was not appreciably better than random chance. With the *Embed-NN* strategy, accuracy on this task was improved to 67.7% which is quite substantial, given the small minority of words in the test set that the QNLP model had seen during training.

For all three tasks, the *Embed-NN* strategy achieved the highest accuracy among the considered OOV strategies. Our results demonstrate that the *Embed-NN* strategy can make QNLP models practical even for datasets with extremely high OOV percentages. We believe that the results here do not represent the optimal performance of the *Embed-NN* method. Improved training-time initialisation strategies would serve to further extract performance from models which already achieve high accuracy, such as in the case of the RP task. It is likely that further performance can be extracted through the use of a lower-dimension initial FastText embedding. Rather than reducing a 300-dimension embedding through PCA, a directly learned lower-dimension FastText embedding will likely preserve information that is lost in the PCA reduction process. Further, a variety of alternative embeddings are available (See Section 6.1.1), whose utility in the *Embed-NN* model must be experimentally investigated.

Chapter 7

Investigating Barren Plateaus

Thus far we have discussed a variety of tasks in the NLP domain, and have presented QNLP models to solve these. One topic which we have not delved into previously is the gradient-based optimisation which plays a central role in QNLP and QML in general. In this chapter, we discuss the *barren plateau* problem: a well-known characteristic of the loss landscape in QML which impedes learning. In Section 7.1 we describe the barren plateau phenomenon and cover the background necessary for the rest of the chapter. In Section 7.2 we calculate an expression for the derivative of the binary cross entropy loss function, for both normalised and sub-normalised quantum states. In Section 7.3 we prove that a loss function defined as the mean of multiple loss functions with barren plateaus will also have a barren plateau. The expression derived in Section 7.2 is used in Section 7.4 to evaluate whether the binary cross entropy loss function has a barren plateau, for regular layered quantum circuits. Finally, in Section 7.5, we consider whether circuits generated from DisCoCat diagrams are more susceptible to the barren plateau problem than regular layered ansatz circuits, and provide empirical evidence that this is not the case.

7.1 Barren Plateaus

As in the case of classical machine learning, QML training relies on the minimisation of a loss function $\mathcal{L}_\theta(\widehat{y})$. This function compares the QML model's predicted label $\widehat{y}$ (parameterised by θ: the model's parameters) with the true label y, and returns a value which serves as a measure of incorrectness. QML (and thus QNLP) models learn by minimising the value of this loss function $\mathcal{L}_\theta(\widehat{y})$ by varying θ through a gradient-based optimisation procedure. As described in Section 3.2.4, this is achieved through a classical optimiser such as SPSA [33]. In each iteration, the optimiser attempts to find a new assignment $\theta' := \theta + \Delta\theta$, such that $\mathcal{L}_{\theta'} < \mathcal{L}_\theta$. In each step, the parameter assignment is changed by a small amount $\Delta\theta$. Such methods iteratively explore the loss landscape, following the gradient towards regions of lower loss.

The *barren plateau* phenomenon, first described by McClean et al. [38] is a characteristic of the landscape of a loss function, which can pose serious challenges to the performance of a QML model. A barren plateau occurs when the loss landscape is overwhelmingly flat. Informally, this means that the gradient evaluated at the majority of parameter assignments is near-zero. Thus when parameter initialisation is done by sampling a random distribution, the gradient will be 0 with extremely high probability. This poses a significant obstacle to learning, since if the gradient is 0, gradient-based optimisation procedures will degenerate to a random walk (or not update the parameters at all), and it is very unlikely that the optimiser will be able to find an optimal parameter assignment.

McClean et al. define a barren plateau as occurring when the gradient of the loss function along any direction is non-zero with exponentially decreasing probability in the number of qubits [38]. For this to be the case, they provide two conditions:

- **Condition 1**: The mean of the gradient of the loss function should be zero.

$$\mathrm{Mean}\big[\partial_j \mathcal{L}\big] = 0$$

- **Condition 2**: The variance of the gradient of the loss function should decrease exponentially in the number of qubits.

$$\mathrm{Var}\big[\partial_j \mathcal{L}\big] \in \mathcal{O}\bigg(\frac{1}{\mathrm{EXP}(n)}\bigg)$$

In their work, they consider a loss function defined as the expectation value of some

Hermitian operator H, with respect to a parameterised quantum circuit $U(\theta)$:

$$E(\theta) = \langle 0|U(\theta)^{\dagger} H U(\theta)|0\rangle \tag{7.1}$$

They prove that this simple, but ubiquitous loss function has a barren plateau when $U(\theta)$ approximates a *2-design*. A 2-design is a parameterised quantum circuit which approximates a *Haar-random unitary* up to the second moment. Haar random unitaries are unitary matrices sampled from the Haar measure on the group of n-qubit unitaries [57]. Less formally, the Haar measure allows us to sample n-qubit unitary operations uniformly at random.

McClean et al. prove the existence of barren plateaus for 2-designs with circuit depth $\mathcal{O}(\text{POLY}(n))$ [38]. Cerezo et al. extend this result and prove that barren plateaus depend on locality of the Hermitian operator [58]. They prove that when the loss function uses a *global observable* for a specific class of circuits (alternating layer ansätze) that approximate 2-designs, the barren plateau exists regardless of the circuit depth. They show that the barren plateau can be avoided through the use of shallow circuits, and cost functions with *local observables.*

7.2 Gradient of the Binary Cross Entropy Loss Function

As a prerequisite to evaluating the barren plateau phenomenon for the Binary Cross Entropy (BCE) loss function, it is necessary for us to define its gradient. This derivation finds use in the experiments we describe in Section 7.4. The BCE loss function is defined as:

$$\mathcal{L}_{BCE} = \frac{-1}{N}\sum_{i=1}^{N} y_i \cdot log(P(y_i)) + (1 - y_i) \cdot log(1 - P(y_i)) \tag{7.2}$$

where y_i is the label of the i'th data sample, and $P(y_i)$ is the model's predicted probability that $y_i = 1$. We define a sample-specific loss:

$$\mathcal{L}_i = y_i \cdot log(P(y_i)) + (1 - y_i) \cdot log(1 - P(y_i)) \tag{7.3}$$

such that the BCE loss function is represented as the (negated) average of the individual loss on N samples:

$$\mathcal{L}_{BCE} = \frac{-1}{N}\sum_{i=1}^{N} \mathcal{L}_i \tag{7.4}$$

For our analysis, the negation of the average is not important, since it does not impact the variance. Thus, the results we prove concerning the variance of the mean (Theorem 7.3.1 and Corollary 7.3.2) apply equally to the BCE loss function. The gradient of the loss function, w.r.t an arbitrary parameter θ_j can be written as:

$$\partial_j \mathcal{L}_{BCE} \equiv \frac{\partial \mathcal{L}_{BCE}}{\partial \theta_j} = \frac{-1}{N} \sum_{i=1}^{N} \frac{\partial \mathcal{L}_i}{\partial \theta_j} \tag{7.5}$$

Following McClean et al. we use ∂_j as shorthand for $\frac{\partial}{\partial \theta_j}$. It is convenient for us to consider the probability of measuring outcome 0 from the prepared state, measured in the Z-basis. For this purpose, we employ the following substitution:

$$\widehat{y_i} := P(y_i) = 1 - P_{\theta,i}(0) \tag{7.6}$$

Where $P_{\theta,i}(k)$ gives the probability of measuring an outcome k, from a circuit prepared for input sample i, parameterised by θ. $\widehat{y_i}$ is introduced for notational convenience, and is the same as $P(y_i)$. $\mathcal{L}_i$ can then be rewritten as:

$$\mathcal{L}_i = y_i \cdot log(1 - P_{\theta,i}(0)) + (1 - y_i) \cdot log(P_{\theta,i}(0)) \tag{7.7}$$

Calculating the partial derivative of $\mathcal{L}_i$:

$$\begin{aligned} \partial_j \mathcal{L}_i &= \frac{-y_i}{1 - P_{\theta,i}(0)} \cdot \partial_j P_{\theta,i}(0) + \frac{1 - y_i}{P_{\theta,i}(0)} \cdot \partial_j P_{\theta,i}(0) & (7.8) \\ &= \left(\frac{1 - y_i}{P_{\theta,i}(0)} - \frac{y_i}{1 - P_{\theta,i}(0)} \right) \cdot \partial_j P_{\theta,i}(0) & (7.9) \end{aligned}$$

Substituting in $\widehat{y_i}$, we get

$$\partial_j \mathcal{L}_i = \left(\frac{1 - y_i}{1 - \widehat{y_i}} - \frac{y_i}{\widehat{y_i}} \right) \cdot \partial_j P_{\theta,i}(0) \tag{7.10}$$

To calculate $\partial_j P_{\theta,i}(0)$, we must consider 2 cases:

Case 1 (*Normalised State*): In the case where each input is associated with a normalised state $|\psi_i(\theta)\rangle$, the probabilities can be calculated as follows:

$$\begin{aligned} P_{\theta,i}(0) &= \langle \psi_i(\theta) | H | \psi_i(\theta) \rangle & (7.11) \\ &= \langle \psi_i(\theta) | 0 \rangle \langle 0 | \psi_i(\theta) \rangle & (7.12) \\ &= |\langle \psi_i(\theta) | 0 \rangle|^2 & (7.13) \\ \therefore \partial_j P_{\theta,i}(0) &= \partial_j |\langle \psi_i(\theta) | 0 \rangle|^2 & (7.14) \end{aligned}$$

Here $H = |0\rangle\langle 0|$ is a projective measurement associated with the $|0\rangle$ state. H is also a Hermitian operator (since it is self-adjoint), and the probability $P_{\theta,i}(0)$ is the associated expectation value of the observable. Section 2.1.4 provides the necessary background on measurement and expectation values. Using this form, it is possible to obtain the gradient of $P_{\theta,i}(0)$ using diagrammatic differentiation [59] of the expression in (7.14), as implemented in DisCoPy [34].

Case 2 (*Sub-normalised State*): In the case of DisCoCat diagrams, states are in-general sub-normalised [5]. Lorenz et al. define the probability as:

$$P_{\theta,i}(0) := \frac{l^0_{\theta,i}}{l^0_{\theta,i} + l^1_{\theta,i}} \tag{7.15}$$

$$l^k_{\theta,i} := ||\langle k|\psi_i(\theta)\rangle|^2 - \epsilon| \tag{7.16}$$

Here ϵ is a small positive constant (10^{-9} in [5]), used to ensure that $(l^0_{\theta,i}, l^1_{\theta,i})$ is a well-defined probability distribution. Considering the derivative of $l^k_{\theta,i}$, with respect to parameter θ_j:

$$\begin{aligned} \partial_j l^k_{\theta,i} &= \partial_j ||\langle k|\psi_i(\theta)\rangle|^2 - \epsilon| && (7.17) \\ &= \operatorname{sgn}(|\langle k|\psi_i(\theta)\rangle|^2 - \epsilon) \cdot \partial_j |\langle k|\psi_i(\theta)\rangle|^2 && (7.18) \end{aligned}$$

Here, the first component only takes values in $\{-1, 1\}$, and the second component can be evaluated diagrammatically using DisCoPy's diagrammatic differentiation. This can be used to calculate the gradient of the probability:

$$\begin{aligned} \partial_j P_{\theta,i}(0) &= \partial_j \frac{l^0_{\theta,i}}{l^0_{\theta,i} + l^1_{\theta,i}} && (7.19) \\ &= \frac{(l^0_{\theta,i} + l^1_{\theta,i}) \cdot \partial_j l^0_{\theta,i} - l^0_{\theta,i} \cdot \partial_j (l^0_{\theta,i} + l^1_{\theta,i})}{(l^0_{\theta,i} + l^1_{\theta,i})^2} \qquad \text{(Quotient rule)} && (7.20) \\ &= \frac{l^1_{\theta,i} \cdot \partial_j l^0_{\theta,i} - l^0_{\theta,i} \cdot \partial_j l^1_{\theta,i}}{(l^0_{\theta,i} + l^1_{\theta,i})^2} && (7.21) \end{aligned}$$

Each of $\partial_j l^0_{\theta,i}$ and $\partial_j l^1_{\theta,i}$ can be calculated using diagrammatic differentiation in the form described in (7.18), and the value of each $l^k_{\theta,i}$ can be obtained by calculating the expectation value $|\langle k|\psi_i(\theta)\rangle|^2$, and processing its output as described in (7.16). Thus we have provided expressions for calculating the gradient of the binary cross entropy loss function for both normalised and sub-normalised states. We use the derived formula for **Case 1** (7.14) in

Section 7.4 to evaluate whether this loss function has a barren plateau.

7.3 Barren Plateaus in Multi-Circuit Models

Before we proceed to experimental results relating to the barren plateau, we provide one final theoretical result relating to ensembles of loss functions. One unique aspect of QNLP models is that they do not consist of a single circuit. Typically, a circuit is generated for each sample in a dataset, and parameters are shared across these. This is because words typically occur in multiple samples. The loss function then, is some linear combination of the loss functions on individual circuits. It is reasonable then, to ask whether this sharing of parameters between circuits can alleviate the barren plateau problem. Specifically, we evaluate whether multiple circuits which each individually demonstrate a barren plateau, demonstrate a more favourable training landscape in ensemble. It turns out, however, that this is not the case.

Theorem 7.3.1. *Given a loss function $\mathcal{L}$ defined as the mean of N individual loss functions $\{\mathcal{L}_1, \mathcal{L}_2, ...\mathcal{L}_N\}$:*

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_i$$

then the variance of the gradient (w.r.t some parameter θ_k) of the combined loss function is no greater than the largest individual variance of a loss function.

$$\sigma^2 \leq MAX\{\sigma_1^2, \sigma_2^2, ...\sigma_N^2\}$$

Proof. We utilise the following fact about the variance of the linear combination of random variables:

$$\sigma^2 = \frac{1}{N^2}\sum_{i=1}^{N}\sigma_i^2 + \frac{2}{N^2}\sum_{i=1}^{N}\sum_{j<i}\mathrm{Cov}(\partial_k\mathcal{L}_i, \partial_k\mathcal{L}_j) \tag{7.22}$$

Without loss of generality, we assume the following ordering of variances:

$$\sigma_1 \geq \sigma_2 \geq .. \geq \sigma_N \tag{7.23}$$

$$\implies \sigma_1^2 \geq \sigma_2^2 \geq .. \geq \sigma_N^2 \tag{7.24}$$

It then follows that:

$$\sigma^2 \leq \frac{1}{N^2} N.\sigma_1^2 + \frac{2}{N^2} \sum_{i=1}^{N} \sum_{j<i} \mathrm{Cov}(\partial_k \mathcal{L}_i, \partial_k \mathcal{L}_j) \tag{7.25}$$

$$\sigma^2 \leq \frac{\sigma_1^2}{N} + \frac{2}{N^2} \sum_{i=1}^{N} \sum_{j<i} \mathrm{Cov}(\partial_k \mathcal{L}_i, \partial_k \mathcal{L}_j) \tag{7.26}$$

The covariance of two random variables X_1 and X_2 is known to observe the following inequality:

$$|\mathrm{Cov}(X_1, X_2)| \leq \sigma_{X_1} \cdot \sigma_{X_2} \tag{7.27}$$

Using this equation, for any 2 samples i and j where $i \neq j$, we have:

$$|\mathrm{Cov}(\partial_k \mathcal{L}_i, \partial_k \mathcal{L}_j)| \leq \sigma_i \cdot \sigma_j \tag{7.28}$$

$$\implies \mathrm{Cov}(\partial_k \mathcal{L}_i, \partial_k \mathcal{L}_j) \leq \sigma_i \cdot \sigma_j \tag{7.29}$$

From the ordering imposed on the variances, we know that:

$$\forall\ i, j \in \{1, 2, ...N\}\ .\ (\sigma_1 \geq \sigma_i) \wedge (\sigma_1 \geq \sigma_j)$$

It then follows that $\forall\ i, j \in \{1, 2, ...N\}$ where $i \neq j$

$$\mathrm{Cov}(\partial_k \mathcal{L}_i, \partial_k \mathcal{L}_j) \leq \sigma_1^2$$

Using this result, we can simplify equation 7.26 to:

$$\sigma^2 \leq \frac{\sigma_1^2}{N} + \frac{2}{N^2} \cdot \frac{N(N-1)}{2}.\sigma_1^2 \tag{7.30}$$

$$\implies \sigma^2 \leq \sigma_1^2 \tag{7.31}$$

Recalling that σ_1^2 is used only due to the ordering we assigned, we can drop this ordering to arrive at the desired statement :

$$\sigma^2 \leq MAX\{\sigma_1^2, \sigma_2^2, ...\sigma_N^2\} \tag{7.32}$$

□

This theorem implies that if $\forall i$. $\mathrm{Var}[\partial_k \mathcal{L}_i]$ decreases exponentially in the number of qubits, then $\mathrm{Var}[\partial_k \mathcal{L}]$ will also decrease exponentially in the number of qubits. Further, if the expectation value of each individual $\partial_k \mathcal{L}_i$ is 0, then by the linearity of expectation:

$$\forall i.\ \text{Mean}[\partial_k \mathcal{L}_i] = 0 \implies \text{Mean}[\partial_k \mathcal{L}] = 0 \tag{7.33}$$

These 2 observations directly yield a result about barren plateaus:

Corollary 7.3.2. *If a set of loss functions $\{\mathcal{L}_1, \mathcal{L}_2, ...\mathcal{L}_N\}$ each have a barren plateau, then the loss function defined as the mean of the individual losses:*

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i$$

will also have a barren plateau.

This result extends to the BCE loss function, as defined in (7.4), since the negative sign affects neither the mean (since $\text{Mean}[-\partial_k \mathcal{L}_i] = \text{Mean}[\partial_k \mathcal{L}_i] = 0$) nor the variance. In the case of QNLP, this implies that if the loss landscapes for individual circuits each have a barren plateau, then the loss function across the whole dataset will also have a barren plateau.

7.4 Experimental Evaluation of the Barren Plateau for the BCE Loss Function

In the previous section, we established that having multiple circuits sharing parameters does not shield us from the barren plateau problem. This implies that if individual DisCoCat circuits are susceptible to the barren plateau problem, the average of the loss on individual circuits is not immune from having a barren plateau. We now turn our attention to the loss function we have been using in all experiments in Chapters 4, 5 and 6. The proofs of the existence of barren plateaus for 2-designs due to McClean et al. and Cerezo et al. discuss exclusively the use of an expectation value as the loss function [38, 58]. They do not, however, discuss the variance of a non-linear function $\mathcal{L}$, which uses the expectation value as input.

An example of such a function is the Binary Cross Entropy (BCE) loss function which we have used in all experiments in previous chapters. Here, we aim to experimentally evaluate whether the variance of the gradient of the BCE loss function also decreases exponentially in the number of qubits, as in the case of the raw expectation value.

Recall our definition of the BCE loss function for a single sample (7.3):

$$\mathcal{L}_{BCE} = y \cdot log(1 - P_\theta(0)) + (1 - y) \cdot log(P_\theta(0)) \tag{7.34}$$

Note that the i subscript has been omitted for brevity. For the case of a normalised state $|\psi(\theta)\rangle$, we defined the models predicted probability in 7.11 as

$$P_\theta(0) = \langle\psi(\theta)|H|\psi(\theta)\rangle \tag{7.35}$$

where $H = |0\rangle\langle 0|$ is a Hermitian operator, indicating the probability of measuring the $|0\rangle$ state. $P_\theta(0)$ is then the expectation value of the Hermitian operator $|0\rangle\langle 0|$. In previous work, this expectation value is treated as the loss function directly [38, 58]. That is:

$$\mathcal{L}_E = \langle\psi(\theta)|H|\psi(\theta)\rangle \tag{7.36}$$

It has been proved that the variance of the gradient of the expectation value decreases exponentially in the number of qubits:

$$Var[\partial_k \mathcal{L}_E] \in \mathcal{O}\left(\frac{1}{EXP(n)}\right)$$

and that the mean of the gradient is 0, for 2 designs [11]. However, it is not known whether $\mathcal{L}_{BCE}$ suffers from the same problem. In this section, we evaluate whether the variance of the gradient of $\mathcal{L}_{BCE}$ decreases exponentially in the number of qubits for layered circuits of our chosen ansätze, and demonstrate empirical evidence that in some cases, the BCE loss function does not suffer from the barren plateau phenomenon.

7.4.1 Experimental Setup

For $\mathcal{L}_{BCE}$, the loss value is dependent on the true label y of the sample, not just the parameter assignment. Thus for each circuit, we calculate both the gradient when $y = 0$ and when $y = 1$. We label these loss functions $\mathcal{L}^0_{BCE}$ and $\mathcal{L}^1_{BCE}$ respectively. Thus, the 3 loss functions of which we analyse the gradient are:

$$\mathcal{L}_E = \langle\psi(\theta)|H|\psi(\theta)\rangle \tag{7.37}$$

$$\mathcal{L}^0_{BCE} = \log(\langle\psi(\theta)|H|\psi(\theta)\rangle) \tag{7.38}$$

$$\mathcal{L}^1_{BCE} = \log(1 - \langle\psi(\theta)|H|\psi(\theta)\rangle) \tag{7.39}$$

To evaluate the variance of the gradient of the considered loss functions, we generate circuits using the 3 ansätze we introduced in Chapter 4: IQP, Sim14 and Sim15. We generate circuits of 2 to 10 qubits, with 1 to 3 layers. For each circuit, we generate the diagram representing the expectation value and use DisCoPy to generate the diagram representing its gradient with respect to the first circuit parameter. This can be used to calculate the gradient of each considered loss function using the process described in Section 7.2. We evaluate the gradients at 50 random parameter initialisations. The variance and mean across these samples are then calculated and reported. While 50 samples may seem small, we note that for each ansatz we consider, we observe results matching prior theoretical results from the literature. Namely that the variance of the gradient of $\mathcal{L}_E$ decreases exponentially in the number of qubits.

7.4.2 Results

Figure 7.1 plots the variance of the gradient for the considered loss functions, for IQP, Sim14 and Sim15 circuits of each considered circuit size. For the case where the loss function is the expectation value $\mathcal{L}_E$, we observe that the variance of the gradient decreases exponentially in the number of qubits. This matches previous results from the literature [38, 58, 42]. For the BCE loss function, we observe different trends for the two values of y. In the case where $y = 1$, $\text{Var}[\partial\mathcal{L}^1_{BCE}]$ decreases exponentially in the number of qubits, at approximately the same rate as $\text{Var}[\partial\mathcal{L}_E]$ for all 3 ansätze. For the BCE loss function when $y = 0$ however, we observe a very different trend. $\text{Var}[\partial\mathcal{L}^0_{BCE}]$ does not decrease as the number of qubits is increased. For all 3 ansätze, $\text{Var}[\partial\mathcal{L}^0_{BCE}]$ is consistently higher than the variances of the other two loss functions.

Figure 7.2 plots the mean of the gradients for each of the loss functions for all 3 ansätze. For $\partial\mathcal{L}_E$ and $\partial\mathcal{L}^1_{BCE}$, we observe that the mean of the gradients is approximately equal to 0 for all ansätze and for all qubits. Thus, since $\text{Var}[\partial\mathcal{L}_E]$ and $\text{Var}[\partial\mathcal{L}^1_{BCE}]$ decrease exponentially in the number qubits, and $\text{Mean}[\partial\mathcal{L}_E]$ and $\text{Mean}[\partial\mathcal{L}^1_{BCE}]$ are approximately equal to 0 for all considered circuits, we conclude that the gradients concentrate exponentially around 0. Thus,

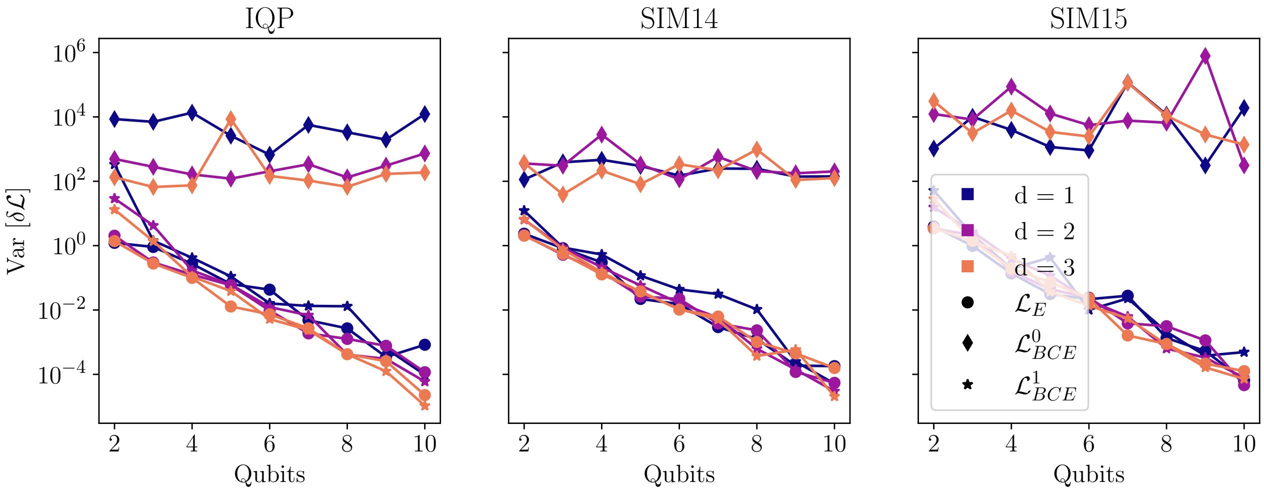


Figure 7.1: Variance of the gradient of considered loss functions for each ansatz. For each ansätze the variance of $\partial\mathcal{L}_E$ and $\partial\mathcal{L}^1_{BCE}$ decrease exponentially in the number of qubits, while the variance of $\partial\mathcal{L}^0_{BCE}$ remains largely unchanged.

we have provided experimental evidence that all 3 circuits: IQP, Sim14 and Sim15 suffer from the barren plateau problem, for the $\mathcal{L}_E$ and $\mathcal{L}^1_{BCE}$ loss functions.

An interesting result from the experiment described in this section is that the Binary Cross Entropy loss function appears to not have the barren plateau problem when the ground-truth label is 0 ($y = 0$). This is evinced by the fact that the variance of $\partial\mathcal{L}^0_{BCE}$ does not decrease exponentially in the number of qubits, and the mean of this gradient does not approach zero. This implies that the gradient does not exponentially concentrate around zero, indicating the absence of a barren plateau. In practice when applying machine learning to a binary classification problem, it is common to ensure a balanced dataset. That is, to ensure a roughly equal number of samples with $y = 0$ and $y = 1$. Thus for a well-balanced dataset, the total loss function $\mathcal{L}$, will be the arithmetic mean of an approximately equal number of $\mathcal{L}^0_{BCE}$ and $\mathcal{L}^1_{BCE}$ per-sample losses. This implies that for a binary classification task, approximately half of the samples will not be individually affected by the barren plateau problem. While our experimental results suggest this, a theoretical proof of this fact is not yet available. Below we pose a concrete question, the answer to which will shed further light on the impact of non-linear cost functions on the barren plateau phenomenon.

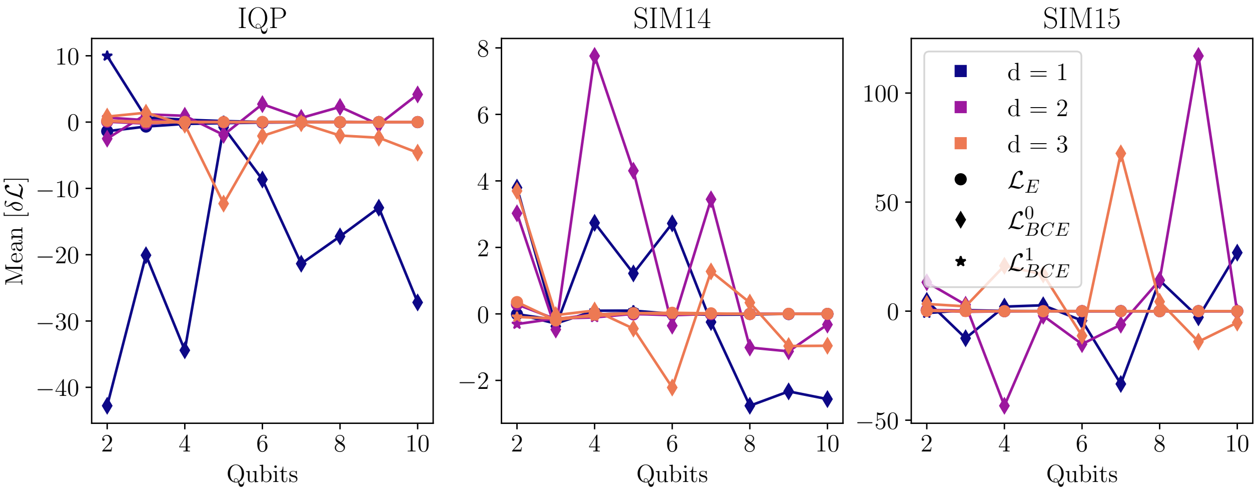


Figure 7.2: Mean of the gradient for all considered loss functions for each ansatz. For all ansätze the mean of $\partial\mathcal{L}_E$ and $\partial\mathcal{L}^1_{BCE}$ is near-0. For $\partial\mathcal{L}^0_{BCE}$, the mean is erratic and does not approach a single value.

> *Given an expectation value defined as*
>
> $$\mathcal{L}_E = \langle\psi(\theta)|H|\psi(\theta)\rangle$$
>
> *for some Hermitian observable H and parameterised quantum state $|\psi(\theta)\rangle$, if it is known that Var $[\partial_k\mathcal{L}_E]$ decreases exponentially in the number of qubits n, of state $|\psi(\theta)\rangle$, then for a function defined as*
>
> $$\mathcal{L}^0_{BCE} = \log(1-\mathcal{L}_E)$$
>
> *what is the trend in Var $[\partial_k\mathcal{L}^0_{BCE}]$ as the number of qubits n is varied?*

While this result is interesting, it does not imply that the BCE loss function entirely circumvents the barren plateau problem. This class-specific nature of the barren plateau exhibited by the BCE loss functions suggests that the model may only be able to learn on samples with $y = 0$, since the samples with $y = 1$ have a near-zero sample-specific gradient. Further study is necessary to determine whether this is the case in practice.

In this section, we presented an experiment yielding interesting results about the effect of using the BCE loss function on the barren plateau problem. These are initial results which we believe motivate further investigation into the relation between non-linear cost functions and

QML loss landscapes. In addition to the open problem posed above, we list some directions for extending our work below.

Due to limitations on the computational resources and time at our disposal, it was not feasible to evaluate circuits deeper than 3 layers. Deeper circuits may be closer approximations to a 2-design, yielding different results. In our experiments, we use a small number of samples to estimate the mean and gradient. Using more samples will yield higher fidelity for these values. This likely did not significantly impact our results, since we observe the exponential decrease in gradient variance for 2 loss functions exactly as described in prior literature. Techniques exist to calculate the exact variance of the gradient directly using the ZX calculus [32]. The use of such techniques will eliminate the need for approximating the variance using multiple samples, yielding precise results.

7.5 Expressibility of DisCoCat Circuits

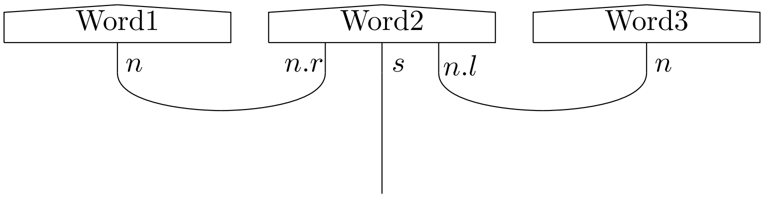
One of the reasons we cited in Chapter 4 for selecting the Sim14 ansatz was its high expressibility. Highly expressive ansätze are favoured as they are more likely to span the solution space [39]. A recent result from Holmes et al. however, shows that there is a direct correlation between the expressibility of a circuit and the severity of the barren plateau phenomenon [12]. The expressibility of a parameterised circuit is described by Sim et al. as being a measure of how closely the parameterised circuit distribution approximates the Haar measure [39]. Haar random unitaries, previously described in Section 7.1 are unitaries sampled from the Haar measure on n-qubit unitaries. This definition of expressibility provides some intuition for the result of Holmes et al. The original work on barren plateaus proved their existence for 2-designs [38, 58]. That is, circuits which approximate the Haar distribution up to the second moment. A circuit with higher expressibility thus is a closer approximation of the Haar distribution and is therefore an approximate 2-design. Consequently, as the expressibility of a circuit increases, it is a closer approximation of a 2-design, making it more susceptible to the barren plateau phenomenon.

Thus far in this chapter, we have focused our attention on regular layered ansatz circuits.

In this section, we turn our attention to DisCoCat circuits, formed from the pregroup grammar of sentences. Typically in DisCoCat diagrams, the output dimension of a sentence state (n_o qubits) is smaller than the dimension used to prepare the state (n_p qubits). This is because there are multiple qubits associated with internal types of the sentence, which are not part of the output. Due to this relation ($n_p > n_o$), a DisCoCat state of n_o qubits will often have far more parameters than a layered state of n_o qubits, even when using the same ansatz. A reasonable worry then, is that DisCoCat states are significantly more expressible than regular layered ansatz states, which would make them more susceptible to the barren plateau phenomenon. It is computationally difficult to analytically calculate the gradient for DisCoCat circuits in the manner we did for the layered circuits in Section 7.4, owing to their larger size even for a small number of output qubits n_o. Thus we use the notion of expressibility described above as a proxy for the barren plateau phenomenon. In this section, we evaluate the expressibility of DisCoCat states, compared with regular layered ansatz states, and argue that DisCoCat states are likely no more susceptible to the barren plateau phenomenon than layered circuits.

7.5.1 Experimental Setup

The space of DisCoCat states is very large, and it would be infeasible to evaluate the expressibility of any significant subset explicitly. We limit our experiments to DisCoCat states which have the following structure:



Such a diagram is generated when parsing phrases of the form “subject - transitive-verb - object”[1]. Here, the output dimension will be $n_o = q_s$ qubits, since the output type is s, while the dimension of the state used to generate this final state is $n_p = q_s + 4 \cdot q_n$, due to the presence of the two cups of type n. To evaluate the expressibility of a quantum state generated using

[1] “*Alice loves Bob*”, “*He pets dogs*” are phrases of this type.

such a DisCoCat diagram, we generate circuits using the IQP, Sim14 and Sim15 ansätze, of 1 to 5 layers each, for output dimensions q_s of 1 and 2 qubits. Small values of q_s are chosen since the tasks to which QNLP methods have been applied thus far are classification tasks on datasets with a small number of classes. We vary q_n in the range $\{0, 1, 2\}$. When $q_n = 0$, the circuit degenerates to a regular layered ansatz on the base state (Word2), and loses its DisCoCat structure. Thus we treat $q_n = 0$ as the baseline, against which to compare the DisCoCat diagrams. To evaluate the expressibility of a circuit, we employ the method described by Sim et al. [39]. For each configuration of each ansatz, we generate 500 pairs of states $(|\psi_\theta\rangle, |\psi_\phi\rangle)$ with random parameter assignments θ, ϕ. Then, we measure the *fidelity* $F = |\langle\psi_\theta|\psi_\phi\rangle|^2$ of each pair. We use these fidelities to generate a probability distribution P_{Circ}, by dividing the range of fidelities $[0, 1]$ into 75 buckets and normalising. A similar distribution for the ideal Haar distribution can be calculated analytically [39, 60]. For a histogram bucket defined by the limits (l, u), the Fidelity distribution for the Haar distribution is defined as:

$$\mathrm{P}_{\mathrm{Haar}}(l, u) = \int_l^u \mathrm{P}_{\mathrm{Haar}}(F)\ dF = (1-l)^{q_s} - (1-u)^{q_s} \tag{7.40}$$

where $\mathrm{P}_{\mathrm{Haar}}(F)$ is the probability that two states sampled from the Haar distribution have fidelity F. As before, q_s is the number of qubits in the state. The expressibility is defined as the Kullback-Leibler (KL) divergence between $\mathrm{P}_{\mathrm{Circ}}$ and $\mathrm{P}_{\mathrm{Haar}}$:

$$\mathrm{Expr} = \mathbf{D_{KL}}(\mathrm{P}_{\mathrm{Circ}}||\mathrm{P}_{\mathrm{Haar}}) \tag{7.41}$$

Here we have borrowed the definition of Expr from Sim et al. Note that circuits with low Expr (low KL-divergence), are highly expressible circuits, and vice versa. In the following section, we present the results of this expressibility study of DisCoCat circuits.

7.5.2 Results

1-qubit States

The KL-divergences for 1-qubit states ($q_s = 1$) generated by up to 5 layers of each ansatz are shown in Figure 7.3. The case where $q_n = 0$, represents a non-DisCoCat circuit. For all ansätze, the single-qubit circuit is a special case, generated by the Euler decomposition

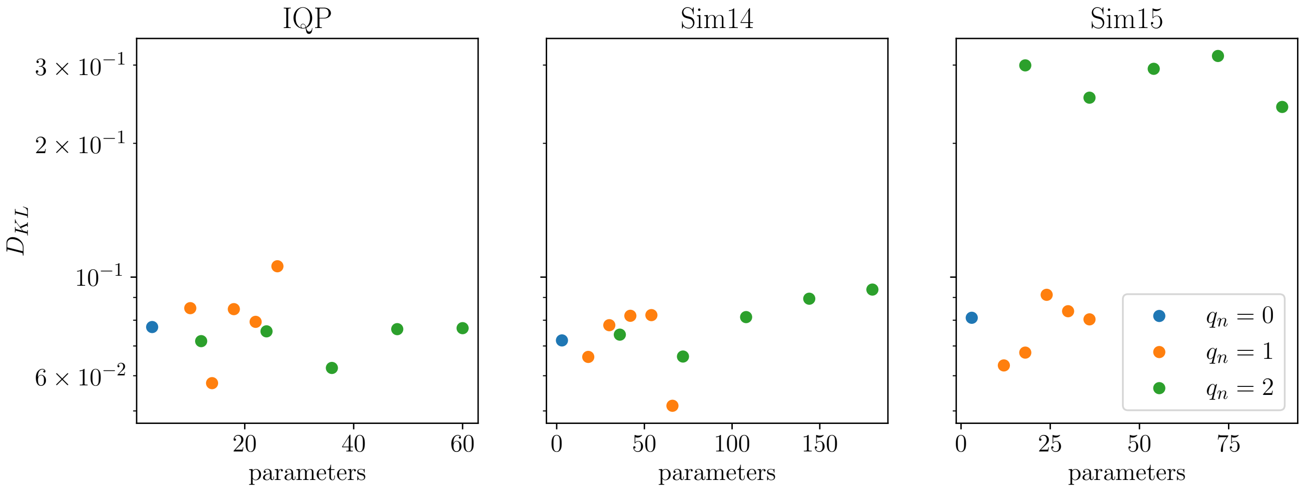


Figure 7.3: KL-divergence for 1-qubit states ($q_s = 1$), for varying q_n for all ansatz. Lower D_{KL} implies higher expressibility. Note that y-axis is log-scaled.

as described in Chapter 4, regardless of the number of layers. Thus for $q_n = 0$, $q_s = 1$, there is a single KL-divergence plotted for each ansatz. These represent identical circuits, but vary marginally due to different random seeds, and the finite number of samples we use to calculate the probability distribution. Recall from the previous section that expressibility and KL-divergence have an inverse relation: a circuit with low KL-divergence is highly expressive.

For the IQP ansatz, we observe that for the majority of circuit depths, the $q_n = 1$ DisCoCat circuit is slightly less expressive than the non-DisCoCat circuit, despite higher parameterisation. The $q_n = 2$ DisCoCat circuit has higher expressibility, but only by a small margin. For the Sim14 ansatz, the $q_n = 1$ DisCoCat circuit has very similar expressibility to the non-DisCoCat circuit for the first 4 layers, but has higher expressibility for the 5-layer circuit. The $q_n = 2$ DisCoCat circuit shows a marginal decrease in expressibility through the layers, and has very-similar or lower expressibility than the non-DisCoCat circuit for all layers. This is a useful result since it demonstrates clearly that increased circuit depth and parameterisation do not directly imply higher expressibility.

We observe the most drastic change for the Sim15 ansatz. The $q_n = 1$ DisCoCat circuit has slightly higher expressibility than the non-DisCoCat circuit for shallow circuits, but has roughly the same expressibility for circuits of depth 3 and greater. The $q_n = 2$ DisCoCat

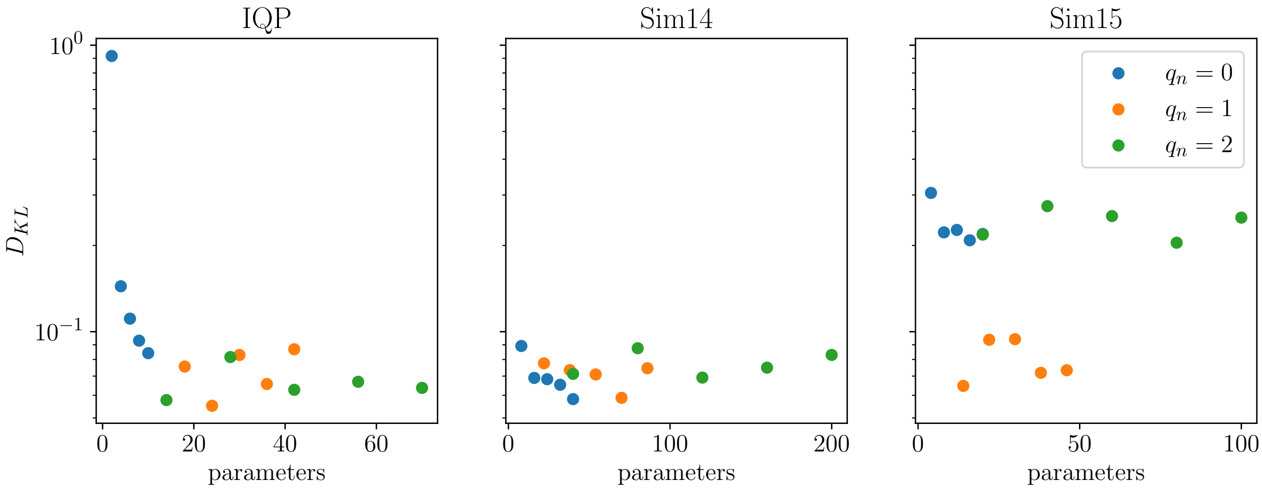


Figure 7.4: KL-divergence for 2-qubit states ($q_s = 2$), for varying q_n for all ansatz. Lower D_{KL} implies higher expressibility. Note that y-axis is log-scaled.

circuit however, has significantly lower expressivity than both other cases, for all circuit depths.

For 1-qubit states, we observe that for no ansatz did the DisCoCat circuits demonstrate distinctly higher expressibility than the non-DisCoCat model. This is an encouraging result, since it suggests that for binary-classification tasks where the output is a single qubit, DisCoCat circuits are not in-general more susceptible to the barren plateau problem than regular parameterised circuits, despite having up to 2 orders of magnitude more parameters.

2-qubit States

The KL-divergences for 2-qubit states ($q_s = 2$) generated by each considered circuit configuration are shown in Figure 7.4. For the IQP ansatz with $q_n = 0$ (non-DisCoCat circuit), we observe that expressibility increases drastically with increase in layers. For the IQP DisCoCat circuits ($q_n = 1, q_n = 2$), we observe that expressibility is higher than in the non-DisCoCat case for almost all circuit depths. For the Sim14 ansatz, we observe a much tighter grouping of expressibilities than for the IQP ansatz. For this ansatz, we observe that increasing q_n did not have a significant impact on expressibility. The circuits have very similar expressibilities for all values of q_n. For the Sim15 ansatz, we observe the most stark difference in expressibility. For the $q_n = 1$ circuits, the expressibility increases dramatically. This trend does not continue for the $q_n = 2$ case however, where the expressibilities are of the same order as the non-DisCoCat circuits. This trend can likely be explained by compositional characteristics of expressibility, which we consider in the following subsection.

Unlike in the single-qubit case, our results vary by ansatz for the 2-qubit case. The IQP DisCoCat circuits demonstrate higher expressibility than the non-DisCoCat circuit. Even the most expressive IQP DisCoCat circuit however, is not appreciably more expressible than the Sim14 non-DisCoCat circuits. For the Sim14 ansatz, DisCoCat circuits are no more expressive than the standard circuits. The Sim15 ansatz has more interesting compositional properties, which can in some cases yield a more expressive circuit than the non-DisCoCat circuit. However, even this circuit is not more expressive than the non-DisCoCat Sim14 circuit with 5 layers.

Compositional Characteristics of Expressibility

For the $q_s = 2$ Sim15 circuits, we observed that the $q_n = 1$ case had far higher expressibility than the $q_n = 2$ case. One possible reason for this is that when $q_n = 1$, the noun state is generated using the single-qubit Euler decomposition. We know that this single-qubit circuit (Sim15 $q_s = 1, q_n = 0$ in Figure 7.3) is more expressive than the layered Sim15 2-qubit ansatz (Sim15 $q_s = 2, q_n = 0$ in Figure 7.4). Thus in the ($q_n = 1$) case, Word1 and Word3 states are the highly expressive Euler decompositions, while for ($q_n = 2$) these words are the less-expressive Sim15 ansätze for 2 qubits. This implies that the circuit for $q_n = 1$, plugging in two highly expressive states (Word1, Word3), into the base state yielded a resultant highly-expressive state. Whereas for the $q_n = 2$ state, the states for Word1 and Word3 are less expressive, yielding a final state which is similarly less expressive. This is a particularly interesting result, since it clearly demonstrates that a base state can have vastly different expressibilities, depending on the expressibilities of other states which are plugged into a selection of its qubits. Further, this result motivates the need for a compositional study of expressibility. Specifically, what compositions of quantum states reduce or increase the expressibility of the resultant state. This would allow evaluating a DisCoCat diagram to determine the likely expressibility of the final state, and would motivate ansatz choice so as to generate circuits of desired expressivity.

Chapter 8

Conclusion

8.1 Summary of Results

In this work, we have presented experimental results comparing ansätze for various QNLP tasks. We presented the first results for QNLP applied to the task of paraphrase identification. We have presented a novel approach, *Embed-NN*, to overcome the out-of-vocabulary problem faced by QNLP models. Through our approach, it is possible to apply QNLP models to datasets where a large percentage of the test set's vocabulary was not observed during training. Further, we have provided some experimental and theoretical results relating to the barren plateau phenomenon in QML.

In Chapter 4 we compared the Sim ansätze with the IQP ansatz and demonstrated that they achieve superior performance on the MC and RP tasks. In Chapter 5 we presented the first QNLP results for the paraphrase identification task. This task is more difficult than the previous tasks to which the DisCoCat form of QNLP has been applied, requiring significant semantic understanding. We demonstrated that the Sim14 ansatz is a suitable choice for the majority of QNLP tasks, across datasets. We argued that a significant factor limiting QNLP performance at the paraphrase identification task was the out-of-vocabulary (OOV) problem. In Chapter 6 we considered this problem in detail and presented a novel approach to overcoming it, using classical word embeddings. We demonstrated that our method yields significant performance improvement for the paraphrase identification task.

In Chapter 7 we considered the barren plateau phenomenon, a problem affecting the loss landscape in QML. We experimentally observed that the binary cross entropy loss function does not appear to demonstrate a barren plateau when $y = 0$. Further, we demonstrated empirically that circuits derived from DisCoCat diagrams for binary classification tasks are not in-general more expressive than regular layered circuits, suggesting that they are not more susceptible to the barren plateau phenomenon, despite much higher parameterisation.

8.2 Future Work

One limitation of our results is that all experiments were executed exclusively through exact simulation using tensor contraction. Running our experiments on a NISQ device, or on a noisy quantum simulator will likely impact each of the ansätze we consider in different ways. The performance of each ansatz will be affected by the topology of the specific device, and the noise associated with individual qubits and gates. Further investigation is necessary to determine which ansätze perform well on specific quantum devices.

In all our experiments, we use a single ansatz for each box in a diagram. While this is a natural choice, it is not formally required. An alternative approach worth investigating is to assign a specific ansatz to each word type. This could potentially be used to ensure that nouns and words important to the specific task have highly expressive representations, while words deemed less relevant to the task have limited parameterisation, thereby bounding model complexity. In the OOV experiments in Chapter 6, we observed that the FastText embedding may not be an optimal training-time parameter initialisation in all cases. Further investigation is necessary to evaluate alternate training-time initialisations, in conjunction with the *Embed-NN* method we proposed. The choice of FastText embeddings in the *Embed-NN* method is not canonical, and research into the relative performance of Word2Vec, BERT or other classical embeddings would be of value.

In Chapter 7, we presented experimental evidence that the BCE loss function does not experience the barren plateau phenomenon for $y = 0$. This is a surprising result, which requires further mathematical analysis. We formally pose this question in Section 7.4.2.

We used expressibility as a proxy for the barren plateau phenomenon and experimentally showed that circuits derived from DisCoCat diagrams are in-general not more expressive than regular layered circuits. This proxy was employed due to the computational complexity of statistically measuring the variance of the gradient. Our results can be refined through the use of diagrammatic techniques to analytically calculate the variance of the gradient [32], which would allow us to directly reason about the barren plateau phenomenon.

Bibliography

[1] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*, 2010.

[2] William Zeng and Bob Coecke. Quantum algorithms for compositional natural language processing. *arXiv preprint arXiv:1608.01406*, 2016.

[3] Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis, and Alexis Toumi. Foundations for near-term quantum natural language processing. *arXiv preprint arXiv:2012.03755*, 2020.

[4] Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice, and Bob Coecke. Grammar-aware question-answering on quantum computers. *arXiv preprint arXiv:2012.03756*, 2020.

[5] Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis, and Bob Coecke. QNLP in practice: Running compositional models of meaning on a quantum computer. *arXiv preprint arXiv:2102.12846*, 2021.

[6] Richard P Feynman. Simulating physics with computers. In *Feynman and computation*, pages 133–153. CRC Press, 2018.

[7] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of statistical physics*, 22(5):563–591, 1980.

[8] Y I Manin. *Vychislimoe i nevychislimoe.* Sov. radio,, 1980.

[9] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[10] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[11] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[12] Zoë Holmes, Kunal Sharma, Marco Cerezo, and Patrick J Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3(1):010313, 2022.

[13] Nishant Jain, Brian Coyle, Elham Kashefi, and Niraj Kumar. Graph neural network initialisation of quantum approximate optimisation. *arXiv preprint arXiv:2111.03016*, 2021.

[14] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[15] Bob Coecke and Ross Duncan. Interacting quantum observables. In *International Colloquium on Automata, Languages, and Programming*, pages 298–310. Springer, 2008.

[16] John van de Wetering. ZX-calculus for the working quantum computer scientist. *arXiv preprint arXiv:2012.13966*, 2020.

[17] Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark, and Bob Coecke. lambeq: An Efficient High-Level Python Library for Quantum NLP. *arXiv preprint arXiv:2110.04236*, 2021.

[18] Kang Feng Ng and Quanlong Wang. A universal completion of the ZX-calculus. *arXiv preprint arXiv:1706.09877*, 2017.

[19] Bob Coecke and Aleks Kissinger. Picturing quantum processes. In *International Conference on Theory and Application of Diagrams*, pages 28–31. Springer, 2018.

[20] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[21] Maria Schuld and Francesco Petruccione. *Supervised learning with quantum computers*, volume 17. Springer, 2018.

[22] KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

[23] Joachim Lambek. *From Word to Sentence: a computational algebraic approach to grammar*. Polimetrica sas, 2008.

[24] Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, Stephen Pulman, and Bob Coecke. Reasoning about meaning in natural language with compact closed categories and Frobenius algebras. *Logic and algebraic structures in quantum computing*, page 199, 2013.

[25] Richie Yeung and Dimitri Kartsaklis. A CCG-based version of the DisCoCat framework. *arXiv preprint arXiv:2105.07720*, 2021.

[26] Mark Steedman. *The syntactic process*. MIT press, 2001.

[27] Stephen Clark. Something old, something new: Grammar-based CCG parsing with transformer models. *arXiv preprint arXiv:2109.10044*, 2021.

[28] Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. A* CCG parsing with a supertag and dependency factored model. In *ACL (1)*, 2017.

[29] Hugh Collins and Kortney Easterly. Ibm unveils breakthrough 127-qubit quantum processor, 2021. `https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor`, Accessed on 30/12/2021.

[30] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[31] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[32] Quanlong Wang and Richie Yeung. Differentiating and integrating ZX diagrams. *arXiv preprint arXiv:2201.13250*, 2022.

[33] James C Spall. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on aerospace and electronic systems*, 34(3):817–823, 1998.

[34] Giovanni de Felice, Alexis Toumi, and Bob Coecke. Discopy: monoidal categories in python. *arXiv preprint arXiv:2005.02975*, 2020.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[36] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[37] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t|ket>: a retargetable compiler for NISQ devices. *Quantum Science and Technology*, 6(1):014003, 2020.

[38] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.

[39] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[40] Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

[41] Laura Rimell, Jean Maillard, Tamara Polajnar, and Stephen Clark. Relpron: A relative clause evaluation data set for compositional distributional semantics. *Computational Linguistics*, 42(4):661–701, 2016.

[42] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[43] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China, July 2015. Association for Computational Linguistics.

[44] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[45] Mihir Kale, Aditya Siddhant, Sreyashi Nag, Radhika Parik, Matthias Grabmair, and Anthony Tomasic. Supervised contextual embeddings for transfer learning in natural language processing tasks. *arXiv preprint arXiv:1906.12039*, 2019.

[46] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[48] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[49] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[50] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[51] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069*, 2019.

[52] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[53] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

[54] Simon Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994.

[55] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

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

[57] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Physical Review A*, 80(1):012304, 2009.

[58] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, 12(1):1–12, 2021.

[59] Alexis Toumi, Richie Yeung, and Giovanni de Felice. Diagrammatic differentiation for quantum machine learning. *arXiv preprint arXiv:2103.07960*, 2021.

[60] Saesun Kim. Enhance qiskit papers database & replication study. https://github.com/bagmk/Quantum_Machine_Learning_Express, 2022.

Appendix A

Rewrite Rules for Paraphrase Identification Task

The following lambeq diagram rewrite rules were used in processing the input for the paraphrase identification tasks on PPDB and MRPC datasets.

1. **prepositional_phrase**: Simplifies prepositions by passing through the noun wire using a cap.
2. **determiner**: Removes determiners (such as "the") by replacing them with caps.
3. **coordination**: Simplifies "and" by replacing it with a layer of interleaving spiders.
4. **connector**: Removes sentence connectors (such as "that") by replacing them with caps.

Descriptions are reproduced from the lambeq [17] documentation online[1].

[1] https://cqcl.github.io/lambeq/tutorials/rewrite.html