University of Western Australia

Data Science Research Project

Quantum Transformers: From Circuit Design to Performance Comparison

Wei Chun Nicholas Choong

Supervised by
Assoc. Prof. Wei Liu
Assoc. Prof. Du Huynh
Prof. Jingbo Wang
Prof. Fredrick Cadet

October 2024

Abstract

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Over the past few decades, the exponential growth of deep learning has led to groundbreaking advancements that have not only transformed computer science but have also permeated numerous other disciplines and industries. The profound impact of deep learning is evident in its integration into various aspects of daily life—from enhancing virtual assistants and powering autonomous vehicles to revolutionising healthcare diagnostics and financial analytics. Deep learning models have achieved superhuman performance in tasks such as computer vision, natural language processing (NLP), and generative artificial intelligence (AI), enabling the creation of human-like outputs across multiple media, including text, images, audio, and video.

This influence was further emphasised in 2024 when John J. Hopfield and Geoffrey E. Hinton were jointly awarded the Nobel Prize in Physics for their seminal contributions to artificial neural networks. Hopfield's pioneering work on associative memory models and Hinton's revolutionisation of the backpropagation algorithm laid the foundational architectures of modern deep learning. The recognition by the Nobel Committee not only highlights the interdisciplinary nature of their work but also cements the significance of deep learning in advancing scientific and technological progress.

Despite these remarkable achievements, the rapid evolution of deep learning has brought several limitations to the forefront, particularly concerning the computational and financial resources required to train increasingly complex models. As model architectures grow in depth and breadth, the demand for vast amounts of data and processing power escalates, presenting significant barriers to scalability and broader application. Classical hardware, while powerful, faces inherent limitations in handling the ever-growing demands of cutting-edge AI models. This is especially evident in emerging fields like multimodal learning, artificial general intelligence, and protein folding, where the sheer scale of computations challenges even the most advanced classical systems.

In response to these challenges, quantum computing has emerged as a promising paradigm that could potentially overcome the limitations of classical computing. By leveraging the principles of quantum mechanics, quantum computing offers the potential to process information in fundamentally new ways, providing exponential speedups for certain tasks. Quantum phenomena such as superposition and entanglement allow quantum computers to explore vast computational spaces more efficiently than classical bits, opening up new avenues for tackling problems previously considered intractable. The implications of quantum computing extend across various sectors, including cryptography, optimisation, and artificial intelligence.

At the intersection of deep learning and quantum computing lies the potential for a new class of models: quantum deep learning models, particularly quantum transformers. Inspired by classical transformer models—which have revolutionised NLP, computer vision, and generative AI—these quantum counterparts aim to enhance capabilities by exploiting the unique properties of qubits, such as superposition, entanglement, and quantum parallelism. Quantum transformers hold the promise of processing information more efficiently, enabling more scalable and resource-efficient models. This presents exciting possibilities for advancing tasks like language translation, image generation, and even more complex applications that push the boundaries of AI.

Recent developments have seen claims that quantum transformers can outperform classical models in specific scenarios, achieving comparable or superior results with fewer resources. For instance, the work of Cherrat et al. [2] suggests that quantum vision transformers can attain accuracies that meet or exceed those of classical vision transformers. However, these claims face significant challenges. Quantum hardware remains in developmental stages, making the implementation of quantum algorithms complex and resource-intensive. Additionally, training quantum neural networks, including quantum transformers, often encounters the problem of barren plateaus—regions in the optimisation landscape where gradients vanish, hindering effective training.

Moreover, the theoretical and practical aspects of integrating quantum computing with deep learning are still under active research. Issues such as error correction in quantum systems, decoherence, and the scalability of quantum circuits pose substantial hurdles. The quantum machine learning community is actively exploring algorithms and architectures that can mitigate these problems, aiming to unlock the full potential of quantum-enhanced AI.

The convergence of quantum computing and deep learning represents a frontier with immense potential but also significant uncertainty. As both fields continue to evolve, their intersection could lead to breakthroughs that redefine computational capabilities and transform industries. Understanding the current landscape, the challenges, and the opportunities is crucial for advancing this emerging area of research.

The integration of quantum mechanics into deep learning models also raises philosophical and ethical considerations. The prospect of machines that can process information in fundamentally new ways prompts questions about the future of AI, the nature of intelligence, and the potential societal impacts. There is a growing discourse on how quantum-enhanced AI could influence areas such as data privacy, security, and employment, highlighting the need for interdisciplinary collaboration in addressing these concerns.

Historically, technological advancements have often led to paradigm shifts in society. The steam engine ignited the Industrial Revolution, electricity transformed daily life and industry, and the internet revolutionised communication and information access. Similarly, the fusion of quantum computing and AI could bring forth a new era of innovation. Industries such as pharmaceuticals might see accelerated drug discovery processes, finance could benefit from more sophisticated modeling and risk assessment, and logistics could achieve unprecedented optimisation levels.

In summary, the advent of quantum computing offers a promising avenue to address the limitations faced by classical deep learning models. Quantum transformers, by harnessing the unique properties of quantum mechanics, could potentially revolutionise AI by enabling more efficient and powerful models. As we stand at the cusp of this new era, exploring the interplay between quantum computing and deep learning is not only intriguing but also essential for the future of computational science.

The journey toward realising the full potential of quantum transformers is fraught with challenges but also rich with opportunities. Continued research and innovation in this field could lead to significant breakthroughs that redefine the limits of machine learning and computation. The implications of such advancements are vast, promising transformative applications across industries and disciplines, and ushering in a new chapter in the story of technological progress.

1.2 Research Objectives

The convergence of quantum computing and deep learning presents a frontier ripe with potential for significant advancements in artificial intelligence. This research seeks to explore this intersection by focusing on quantum transformers. The primary objectives of this study are outlined as follows:

1. **Design and Optimise New Quantum Circuits**

   The goal is to explore and implement various quantum circuit designs for use in quantum transformers. This includes investigating different data encoding methods such as amplitude encoding, angle encoding, and block encoding to effectively represent input data within quantum circuits. We will employ Pennylane's basic and strong entangling layers for constructing variational quantum circuits (VQCs) to enhance the expressibility and entangling capabilities of the quantum circuits. Additionally, we will use an autoencoder to adjust the dimension of the word embedding to fit into the next layer, ensuring efficient data flow through the quantum and classical components.

2. **Compare Quantum and Classical Models and Attempt to Reproduce Existing Results**

   To conduct a thorough comparison between quantum and classical transformer models, focusing on accuracy, area under the curve (AUC), and computational resources. This includes an effort to reproduce results from recent studies to assess the performance and potential benefits of quantum circuits in these models.

3. **Reduce Quantum Model Training Time on Simulated Quantum Computers**

   To explore different quantum computing frameworks, such as Pennylane and TensorCircuit, as well as classical deep learning frameworks like PyTorch and TensorFlow. The goal is to optimise the training process, making quantum transformer models more manageable to be trained on simulated quantum computers, potentially reducing the overall training time and making them more feasible for practical applications.

By pursuing these objectives, our research aims to contribute to the foundational understanding and practical advancement of quantum transformers. This work seeks to explore and demonstrate the capabilities and limitations of integrating quantum computing with deep learning, ultimately contributing to the evolution of artificial intelligence in the quantum era.

1.3 Our Contributions

TODO: This research makes several key contributions to the field of quantum...

1.3.1 Open Source Code Contributions

As part of this research, we have contributed to the open-source community by making our code publicly available. The code for the quantum transformers, including the design and implementation of quantum circuits, is hosted on GitHub. This repository provides a comprehensive set of tools and resources for researcherss interested in quantum machine learning and transformer-based models. Our code repository can be found at `https://github.com/NicholasChoong/QuantumTransformer`.

1.4 Outline

This dissertation is structured as follows:

1. **Chapter 2: Literature Review**

   This chapter presents a comprehensive review of the existing literature on transformer models, quantum machine learning, quantum neural networks, and variational quantum circuits. It also discusses various data embedding techniques and the limitations of current models, providing a foundation for the development of quantum transformers.

2. **Chapter 3: Methodology**

   The methodology outlines the research design, data collection, and the detailed implementation of both classical and quantum transformers. The approaches for training and evaluating these models are also described.

3. **Chapter 4: Experiments and Results**

   This chapter details the experimental setup, describes the conducted experiments, and presents the results of the quantum and classical transformer models. The performance metrics and analysis of the models are also discussed in this section.

4. **Chapter 5: Conclusion and Future Work**

   The conclusion summarises the key contributions of this research, reflecting on the outcomes of the experiments and the broader implications for the field of quantum machine learning. The chapter also outlines potential directions for future research.

5. **Appendices**

   The appendices include supplementary materials such as the research proposal, pseudo code, and a detailed analysis of the training speed of the models.

Chapter 2

Literature Review

2.1 Background

2.1.1 Transformer Models

Transformer architectures have emerged as a dominant paradigm in natural language processing. Unlike traditional recurrent neural networks and convolutional neural networks, transformers avoid sequential processing in favour of self-attention mechanisms that allow for parallel processing of input sequences.

The key innovation of transformers lies in their ability to capture long-range dependencies within input sequences, making them particularly effective for tasks such as language translation, text generation, and sentiment analysis. Models such as Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT) have achieved state-of-the-art performance on various natural language processing benchmarks, demonstrating the power of transformer architectures.

2.1.2 Quantum-Inspired Machine Learning

Quantum-Inspired Machine Learning (QiML) refers to a class of machine learning algorithms that draw inspiration from quantum mechanics but are implemented on classical hardware. Unlike Quantum Machine Learning (QML), which relies on quantum computation to process data, QiML applies quantum principles to classical algorithms without the need for quantum devices. QiML has gained attention due to its potential to harness computational advantages by simulating quantum effects through classical frameworks. Techniques such as tensor networks, dequantized algorithms, and quantum-inspired optimisation algorithms are prominent examples of QiML approaches. These methods have been shown to improve efficiency and performance on certain tasks traditionally solved using classical machine learning models.

The relationship between QiML and QML can be understood as a spectrum of quantum mechanics integration, as illustrated in Figure 2.1. On the left side of the spectrum, we have classical approaches like dequantized algorithms that employ quantum-inspired techniques but operate entirely within classical systems. As we move to the right, more quantum mechanics are integrated, including tensor networks and density matrices used as features, which bridge classical and quantum methods. Finally, on the far right, we encounter true QML approaches, such as VQCs and quantum kernels, which require quantum hardware for computation.

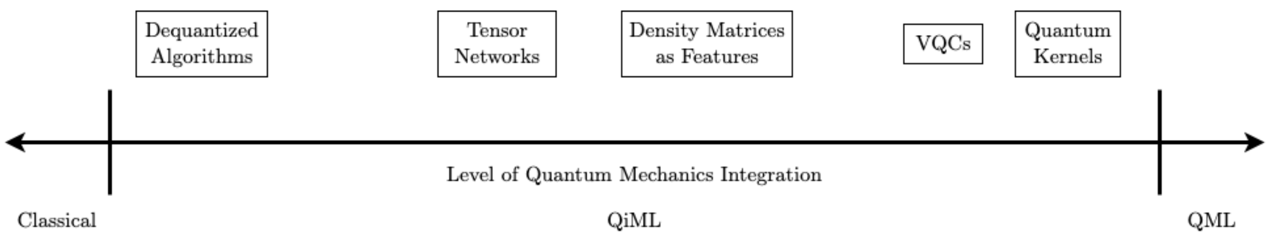


Figure 2.1: A spectrum of machine learning approaches from classical to quantum-inspired to quantum-based methods [5].

The primary difference between QML and QiML lies in their reliance on quantum hardware. While QML explores the use of quantum computers to solve machine learning problems, QiML remains entirely within the classical computational realm, applying quantum concepts without leveraging quantum hardware. This distinction makes QiML more accessible given current technological limitations in quantum computing.

It is important to note that our research focuses on QML, not QiML. In this project, we will directly employ simulated quantum hardware and algorithms to explore how quantum computing can enhance machine learning, specifically for sentiment analysis tasks using transformer models.

2.1.3 Quantum Neural Network

A Quantum Neural Network (QNN) is a machine learning architecture that employs quantum computing principles. It has a set of trainable parameters (θ) that can be realised based on an initial probability distribution. The goal of machine learning is to train these parameters and achieve a probability distribution that closely resembles the underlying problem. A model with high generalisation ability can identify existing patterns in the testing data without overfitting to the training data.

The specific approach involves providing the data to the quantum model through a process called a data embedding strategy, which can significantly impact the functional representation of the model. The objective function is the expectation value of a variational quantum circuit. These circuits make use of continuous variable group rotations, allowing for the manipulation of quantum states.

Since there is no established framework for designing variational quantum circuits, it is crucial to fine-tune the circuit architecture, as it can directly affect the performance of the quantum neural network model. Fortunately, many quantum computing libraries, such as Pennylane and Qiskit, provide templates for VQCs.

2.1.4 Data Embedding

There are many encoding strategies for loading classical data into a quantum computer but for this literature, we will only be discussing about three encoding strategies. They are Angle embedding, Amplitude embedding, Quantum Approximate Optimization Algorithm (QAOA) embedding and Block Encoding.

- Angle embedding [9] is a straightforward strategy where classical data features are encoded into the angles of quantum gates. Each feature is mapped to the rotation angle of a qubit, typically using single-qubit rotation gates such as RX, RY, or RZ. To encode n features, Angle embedding requires n qubits, as each feature is directly assigned to a qubit's rotation. While this encoding method is straightforward, it can be resource-intensive for large datasets, as the number of qubits required grows rapidly with the number of features.

- Amplitude embedding [9] is a more complex strategy that encodes classical data into the amplitudes of a quantum state. This method involves preparing a quantum state where the amplitudes of the state vector represent the classical data values. Given a classical data vector $x = [x_1, x_2, \ldots, x_n]$, the first step is to normalise the data. The normalisation is done by dividing each element by the Euclidean norm $\|x\|$, where:

  $$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{2.1}$$

  This ensures the total probability of the quantum state is 1. The normalised vector is:

  $$\tilde{x} = \frac{1}{\|x\|}[x_1, x_2, \ldots, x_n] \tag{2.2}$$

  Next, the normalised vector $\tilde{x}$ is encoded into the amplitudes of a quantum state $|\psi\rangle$, which can be represented as:

  $$|\psi\rangle = \sum_{i=1}^{n} \tilde{x}_i |i\rangle \tag{2.3}$$

  where $|i\rangle$ are the computational basis states. To encode n features, $\log_2(n)$ qubits are required, assuming n is a power of two, because the quantum state must accommodate all n amplitudes, and $\log_2(n)$ qubits are needed to create a state with n possible amplitudes. For example, if $n = 4$, the resulting quantum state would be:

$$|\psi\rangle = \tilde{x}_1|00\rangle + \tilde{x}_2|01\rangle + \tilde{x}_3|10\rangle + \tilde{x}_4|11\rangle \tag{2.4}$$

This encoding strategy is particularly useful for applications that involve processing high-dimensional data, as it can significantly reduce the number of qubits required compared to other encoding methods.

- The Quantum Approximate Optimization Algorithm (QAOA) embedding [7] is a hybrid approach that encodes classical data into the parameters of a QAOA circuit. QAOA is originally designed for solving combinatorial optimisation problems but can be adapted for data encoding by associating the problem parameters with data features. The number of qubits required for QAOA embedding depends on the specific problem and the depth of the QAOA circuit but generally requires at least as many qubits as there are features to encode the data adequately. For n features, a minimum of n qubits is typically needed, with additional qubits possibly required depending on the circuit's complexity and the problem structure. In general, QAOA embedding requires several qubits that scale with the number of variables and constraints in the optimisation problem. This encoding strategy is particularly useful for solving combinatorial optimisation problems on quantum computers.

- Block encoding embeds a non-unitary operator as a sub-block of a larger unitary matrix, allowing non-unitary matrices to be processed within quantum circuits. In general, a matrix $A \in \mathbb{C}^{N\times N}$, where $N = 2^n$, can be block-encoded into a unitary matrix by extending the dimension of the matrix and adding ancilla qubits.

  The block-encoded matrix U can be represented as:

$$U = \begin{pmatrix} A & * \\ * & * \end{pmatrix} \tag{2.5}$$

  The block-encoded form allows us to work with non-unitary operators in a quantum framework while maintaining the overall unitary evolution required by quantum mechanics. To perform block encoding, we use a combination of quantum oracles U_A and U_B. The oracle U_A encodes the matrix elements $A_{i,j}$ into the amplitude of an ancillary qubit, while U_B ensures proper indexing over the matrix entries. The combination of these operations allows the matrix A to be embedded as a block within a larger unitary matrix. This technique is particularly useful for quantum algorithms that involve manipulating large, structured, or sparse matrices, as it efficiently encodes the matrix into a quantum state without directly applying non-unitary operations. Block encoding is crucial for several quantum machine learning and linear algebra algorithms, enabling the practical implementation of otherwise challenging quantum computations.

2.1.5 Variational Quantum Circuit

In classical neural networks, increasing the number of parameters enhances the expressivity of the models. However, in quantum circuits, having too many parameters can lead to redundancy because over-parameterised circuits may not improve performance and can make optimisation more challenging. Excess parameters can result in entangled states that do not contribute meaningfully to the solution, potentially causing the circuit to explore a larger, less relevant portion of the parameter space. As mentioned earlier, there are no standard frameworks for designing these architectures, resulting in varying designs from one author to another. Nonetheless, quantum computing libraries offer templates for VQCs to facilitate their use.

$$U(\theta) = \prod_{i=1}^{L} U_i(\theta_i) \tag{2.6}$$

where L is the number of VQC iterations, U is the VQC unitary, $\theta = (\theta_1, ..., \theta_L)$ is a set of trainable parameters.

The variational quantum circuit [6] consists of repeated rotation gates applied to each qubit, along with CNOT gates that entangle the qubits to form a highly entangled system. These rotation gates, such as RX, RY, or RZ, adjust the qubit states based on the trainable parameters. The CNOT gates, on the other hand, create entanglement between pairs of qubits, enabling complex interactions across the entire quantum system.

This combination of rotation and entangling gates allows the circuit to explore a wide range of quantum states, enhancing its ability to model complex functions and relationships within the data. By fine-tuning the parameters of these gates, the quantum circuit can be optimised to solve specific problems or recognise patterns in data. However, designing these circuits requires careful consideration, as the arrangement and number of gates can significantly impact the circuit's performance and computational efficiency.

2.1.6 Gradient Calculation

The most common gradient calculation method in classical machine learning is the back-propagation algorithm. This technique computes the gradient of each function that the trainable parameters pass through, using the chain rule to create an automatically differentiable machine learning routine. In the context of quantum machine learning, back-propagation can also be employed. However, there's a key difference: it requires access to the quantum state vector. As a result, its application is limited to quantum simulators rather than real quantum processing units (QPUs).

As a result, it is crucial to find alternative methods that can operate on actual QPUs. One such alternative is the finite difference method [10], a form of numerical differentiation used to approximate derivatives. The finite difference method estimates the derivative of a function by evaluating the function at slightly shifted parameter values and calculating the ratio of the change in the function value to the change in the parameter.

However, while finite difference provides a useful approximation, it can be quite unstable when used iteratively in processes such as gradient descent. This instability arises because the method is sensitive to numerical errors, which can accumulate over multiple iterations, leading to inaccurate gradient estimates. Additionally, near-term quantum hardware is inherently noisy, which further exacerbates the inaccuracy of finite differences, making it unreliable for precise optimisation tasks.

Another approach is the parameter shift rule [10], which provides an exact derivative by evaluating the circuit twice for each trainable parameter. Unlike finite difference methods, it does not require a small perturbation but instead can use a macroscopic shift. By optimising the shift to maximise the distance in parameter space between the two circuit evaluations, the parameter shift rule offers a robust and precise gradient estimation.

The parameter shift rule involves shifting the parameter by a fixed amount in two directions and using the difference in the circuit's outputs to calculate the gradient. Specifically, for a parameter θ, the gradient $\frac{\partial f(\theta)}{\partial \theta}$ is obtained by evaluating the function $f(\theta + s)$ and $f(\theta - s)$ where s is the shift value.

The exact derivative is then computed as:

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{f(\theta + s) - f(\theta - s))}{2sin(s)} \tag{2.7}$$

This method is advantageous because it provides an unbiased estimator of the gradient, ensuring convergence even when the gradient is estimated with a single shot. In contrast, finite difference methods approximate the derivative by evaluating the function at slightly perturbed values of the parameter, which can be unstable and sensitive to noise, especially in iterative processes like gradient descent. Finite differences rely on a small perturbation to estimate the gradient, which can accumulate numerical errors over multiple iterations, leading to less accurate results.

However, the parameter shift rule does pose challenges. The number of circuit evaluations required increases linearly with the number of trainable parameters, which can limit the complexity of quantum models. As the number of parameters grows, the computational resources needed for gradient estimation also increase, potentially making the process resource-intensive.

2.1.7 Hybrid Classical-Quantum Autoencoder

The variational quantum circuit uses angle embedding, where the input dimension is restricted to a certain size due to the limitations of the available qubits. This restriction limits the quantum circuit's ability to process larger input dimensions directly.

To address this limitation, a classical autoencoder is integrated into the model, comprising an encoder and a decoder:

- The encoder acts as a compression mechanism, performing the squeezing of high-dimensional input data into a smaller latent space that fits the fixed input dimension of the quantum circuit. This is crucial for ensuring that the data can be processed within the qubit limitations of the VQC.
- The decoder performs the reverse operation, acting as an unsqueezing mechanism. After the quantum circuit processes the compressed data, the decoder expands the quantum-processed data back to its original size or another appropriate dimension for subsequent layers in the network.

By using this autoencoder structure, the model effectively bridges the dimensional gap between high-dimensional classical data and the qubit-limited quantum circuit. The encoder enables dimensionality reduction while maintaining key features, and the decoder restores the data to a useful size for further processing. This combination allows the quantum circuit to be utilised efficiently within the overall model without losing the ability to work with larger datasets.

2.2 Quantum Transformers

2.2.1 Basic Quantum Transformer

In 2021, Sipio et al. [11] proposed a Quantum Transformer that replaces 5 linear layers with 5 Quantum layers which are ansatzes, 4 of which are in the multi-head attention, and 1 of which is in the feed-forward network. The quantum layers use the basic entangler layer template provided by PennyLane.

A variational quantum circuit cannot change the dimensionality of the input but only rotate the state of the qubits, so the VQC is sandwiched between 2 linear layers. A linear layer is used to compress the dimension of the input to match the number of qubits and another linear layer is used to uncompress the dimension of the output to match the original dimension of the input so it can be passed to the next layer.

This paper presents a conflicting scenario where the embedding dimension and the number of qubits are mismatched. This incongruity poses a significant challenge because the published code mandates that the embedding dimension and the number of qubits align for the multi-head attention mechanism to function correctly.

The parameters for training are as follows: 1 epoch, a batch size of 32, a vocabulary size of 50,000, a maximum sequence length of 64, an embedding dimension of 8, 1 transformer block, 2 transformer heads (the number of attention mechanisms in the multi-head attention component), 1 quantum layer, and 2 qubits. The dataset used for training and testing is the IMDB dataset, which consists of movie reviews labelled as either positive or negative. Both the training and testing sets contain 25,000 reviews each.

The author did not publish the results of the quantum transformer, only mentioning that it took 100 hours to train the classifier for a single epoch. I replicated the transformer and tested it on a subset of the dataset, consisting of just 3,200 reviews. My results showed that after 40 epochs, the training accuracy was 73% and the testing accuracy was 63%. However, the model appeared to be overfitted, as indicated by the train and test loss. More fine-tuning is needed to increase the accuracy and prevent overfitting.

As this is one of the first papers on Quantum Transformers, I assume that the author employed fundamental quantum machine learning techniques to replace one or two components from classical to quantum. By focusing on basic modifications, this entry-level code provides a clear and accessible starting point for understanding the process of converting Transformer components. This incremental approach allows for a step-by-step conversion, making it easier to grasp how quantum machine learning can be applied iteratively to transition from classical to quantum models. This understanding is crucial for developing more advanced quantum models in the future, as it lays the groundwork for integrating quantum computing into existing machine learning frameworks.

2.2.2 Basic Quantum Vision Transformer

In 2023, Comajoan Cara et al. [3] adapted Sipio et al. [11]'s basic quantum transformer into a basic quantum vision transformer. Comajoan Cara et al. [3] used 3 datasets, MNIST dataset, Quark-Gluon dataset, and Electron-Photon dataset.

- MNIST dataset contains 70,000 1-channel 28x28 images of handwritten digits, which are labelled with the corresponding digit. The dataset is split into 60,000 images for training and 10,000 images for testing.

- Quark-Gluon consists of 933,206 3-channel 125x125 images, with half representing quarks and the other half gluons. Each of the three channels in the images corresponds to a specific component of the Compact Muon Solenoid (CMS) detector of the LHC: the inner tracking system (Tracks) that identifies charged particle tracks, the electromagnetic calorimeter (ECAL) that captures energy deposits from electromagnetic particles, and the hadronic calorimeter (HCAL) which detects energy deposits from hadrons.

- Electron-Photon contains 498,000 2-channel 32x32 images, with half representing electrons and the other half representing photons. Here, only information from the CMS electromagnetic calorimeter (ECAL) is used. In particular, the first channel contains energy information (as in the Quark-Gluon dataset), and the second one contains timing information.

To convert the quantum transformer for text classification into a quantum vision transformer for image classification, several modifications are necessary. One major change involves splitting the image into n patches and passing these patches through a linear layer to flatten them before feeding them into the transformer. Another significant change is appending the class tag to the very front of the patches so that the multi-layer perceptron can use it for classification.

The parameters for training the quantum vision transformer are as follows: 50 epochs, a patch size of 32, an embedding size of 8, 8 transformer blocks, 2 transformer heads, and 8 qubits. The training results are as follows:

- For the MNIST dataset, the classical transformer achieved 99.71% accuracy, while the quantum transformer achieved 98.94%.
- For the Quark-Gluon dataset, the classical transformer achieved 79.76% accuracy, while the quantum transformer achieved 77.62%.
- For the Electron-Photon dataset, the classical transformer achieved 76.50% accuracy, while the quantum transformer achieved 77.93%.

From these results, we can see that the classical transformer outperformed the quantum transformer on two of the datasets. For the third dataset, the difference in performance between the classical and quantum transformers is negligible.

Comajoan Cara et al. [3] was aware that Sipio et al. [11]'s Quantum Transformer requires a long training time, so they experimented with different libraries to compare training durations. After testing various options, they found that the library utilising tensor networks was the fastest. This observation highlights that the choice of quantum machine learning library can significantly influence not only the training time but also the performance of the models. Different libraries may offer optimisations, algorithms, and implementations that impact efficiency and accuracy. Therefore, selecting the appropriate library is crucial for achieving optimal results in quantum machine learning experiments and applications. This insight suggests that the performance of the Quantum Transformer could be further improved by using tensor networks.

2.2.3 Quantum Self Attention Neural Network

In 2023, Li et al. [6] proposed a Quantum Self Attention Neural Network for classification. This implementation has 2 parts. The first part is in the quantum realm and the second part is in the classical realm. The input is x, but this paper did not use any positional encoding. The Quantum Self-Attention mechanisms are connected sequentially and they are called the Quantum Self-Attention Layer. Each layer has a set of four ansatzes. The outputs of the final layer are averaged and fed into the second part which is a classical fully connected layer for classification.

In the Quantum Self-Attention Layer, the embedding dimension of each of the inputs has the same size as the number of qubits as each element in the embedding is mapped to its corresponding qubits in the ansatzes to encode the embeddings into their corresponding quantum states. Then, a set of three ansatzes representing query, key, and value is applied to each state. Note that it is the same set of ansatzes applied to all the input states.

The measurement outputs of the query z-rotation part and the key z-rotation part are computed through a Gaussian function to obtain the quantum self-attention coefficients. We then calculate classically weighted sums of the measurement outputs of the value and add the inputs to get the outputs of the current layer where the weights are the normalised coefficient.

All the ansatz parameters and weight are initialised from a Gaussian distribution with zero mean and 0.01 standard deviation, and the bias is initialised to zero. Here, the ansatz parameters are not initialised uniformly from $[0, 2\pi)$ is mainly due to the residual scheme applied before the output of the current layer. They are using the parameter shift rule for the gradient calculation.

The QSANN is not a transformer as it lacks two key components: multi-head attention and position encoding. In their study, two simple synthetic datasets were used, named MC and RP. The MC dataset contains 17 words and 130 sentences (70 for training, 30 for development, and 30 for testing), with each sentence consisting of 3 or 4 words. The RP dataset contains 115 words and 105 sentences (74 for training and 31 for testing), with each sentence containing 4 words.

The results are somewhat unusual. The model achieved 100% accuracy on both training and testing for the MC dataset. For the RP dataset, the training accuracy was 95.35% and the testing accuracy was 67.74%. The second dataset's results seem more reasonable than the first. The anomalous result for the MC dataset might be due to the extremely small size of the dataset, which contains only 17 unique words.

Unlike previous authors, these authors provide a comprehensive and detailed explanation of their implementation, starting from the basics of quantum computing and progressing through to the classical gradient chain rule and the quantum parameter shift rule. They also include clear and informative diagrams to aid in their explanations. This thorough approach ensures that readers can follow the concepts and methodologies step by step. The depth of detail and clarity in this paper have been incredibly helpful in enhancing my understanding of how Quantum Transformers are implemented on quantum computers. Their meticulous explanation and visual aids make complex concepts more accessible and comprehensible, bridging the gap between classical and quantum machine learning techniques. This paper stands out as a valuable resource for anyone looking to grasp the intricacies of Quantum Transformers.

2.2.4 Quantum Vision Transformer

In 2024, Cherrat et al. [2] proposed two Quantum Vision Transformers. They designed a custom data loader with two registers. The top register loads the norms of each row of the input patch vector, while the lower register sequentially loads each row by applying the vector loader and its adjoint for each row, using CNOTs controlled by the corresponding qubit of the top register. In essence, they use amplitude encoding to transfer classical data into the quantum realm.

Cherrat et al. [2] also designed three circuits which they called the orthogonal layer: Pyramid, X, and Butterfly. Instead of using CNOTs to entangle the qubits, they employed a different entangling gate known as the RBS gate. The process begins with a Hadamard gate applied to each of the two qubits, followed by a two-qubit CZ gate. This is then followed by a Ry rotation gate on each qubit, where the top qubit is rotated by $\frac{\theta}{2}$ and the bottom qubit by $-\frac{\theta}{2}$. Another two-qubit CZ gate is applied, and finally, a Hadamard gate is applied to each qubit.

The first transformer they proposed is called the Quantum Orthogonal Transformer. This circuit begins by loading the input patch vector, followed by a trainable quantum orthogonal layer. Next, an inverse data loader for the input patch vector creates a state where the probability of measuring 1 on the first qubit is exactly the square of the attention coefficient. Another circuit, the Orthogonal Patch-wise Neural Network, is similar but lacks the inverse data loader. Its output, the feature vector, combined with the attention coefficient, can then be classically processed to obtain the attention vectors.

Additionally, they designed a circuit called Direct Quantum Attention to calculate the attention output within the quantum computer. This circuit consists of two registers: the top register loads the attention coefficients, while the bottom register contains a data loader for the input patch vector. The loader is controlled by the top register using a CNOT gate, so the input patch vector is loaded with all rows rescaled according to the attention coefficients. The data is then passed through a quantum orthogonal layer, which performs matrix multiplication between the feature vector and the newly encoded vector. The output of the second register is the attention vector. With this tool, the Quantum Orthogonal Transformer with Direct Quantum Attention can perform attention calculations entirely within the quantum realm.

They also proposed another Quantum Vision Transformer implementation, claiming it utilises quantum-native linear algebraic operations that cannot be efficiently performed classically. However, they did not publish their code, and Cara's attempts to replicate their work were unsuccessful, noting that some explanations in their work were unclear. Therefore, this implementation will only be briefly mentioned.

They trained and tested their models on the MedMNIST dataset, a collection of 12 pre-processed, two-dimensional, open-source medical image datasets. Their results demonstrated that the orthogonal transformer and the compound transformer, both of which implement non-trivial quantum attention mechanisms, delivered very competitive performances compared to the classical vision transformer. Notably, these quantum transformers outperformed the classical vision transformer on 7 out of the 12 MedMNIST datasets.

The authors tested their models on actual quantum computers, providing a practical demonstration of their implementation. Furthermore, they developed a Quantum Transformer capable of calculating the attention output directly on the quantum computer. This approach contrasts with previous quantum transformers, which rely on classical computation for attention calculation. By integrating the attention mechanism fully within the quantum framework, their Quantum Transformer potentially offers improved performance and efficiency, showcasing a significant advancement in the application of quantum computing to machine learning tasks. This novel approach demonstrates the feasibility and benefits of performing more complex computations directly on quantum hardware.

2.3 Synthesis

All the mentioned models use variational quantum circuits to replace classical neural networks with quantum neural networks. Most of these models calculate the attention output classically, maintaining some dependence on classical computation. The only model that calculates the attention output directly on the quantum computer is the direct quantum attention add-on for Cherrat et al. [2]'s model, setting it apart as a more integrated quantum solution.

Despite these differences, a common feature among all models is the use of a classical fully connected layer for classification. This reliance on classical components means that all these models are hybrid quantum transformers rather than purely quantum-native transformers. The hybrid approach leverages the strengths of both quantum and classical computing but also indicates that fully quantum-native transformers are still under development.

Moreover, while these models share the goal of enhancing computational efficiency and performance through quantum computing, their varying approaches highlight the experimental nature of the field. For instance, the choice of entangling gates, data encoding strategies, and specific quantum operations differ across models, reflecting ongoing exploration of optimal techniques.

Interestingly, none of the papers discuss the issue of barren plateaus, a problem in quantum neural networks where gradients vanish during training. This phenomenon can severely affect training efficiency or even bring it to a halt, as the gradient of the model tends to zero in all directions. According to McClean et al. [8], quantum neural networks must be shallow and use a limited number of qubits to remain trainable, which contradicts the vision of high-dimensional, million-qubit quantum machine learning models. This issue underscores a significant challenge in the field that future research must address to realise the full potential of quantum neural networks.

Chapter 3

Methodology

3.1 Research Design

3.2 Data Collection

3.3 Implementation Details

3.4 Approach

Chapter 4

Experiments and Results

4.1 Setup

The experiments were conducted within a high-performance computing environment to facilitate both classical and quantum transformer model training and evaluation. The following describes the computational setup:

1. **Computational Environment**

    - **Hardware**: All experiments were performed on UWA's *Kaya High-Performance Computing (HPC)* cluster. The HPC environment was equipped with NVIDIA GPUs, utilised to accelerate the training process for classical models and simulate quantum circuits.
    - **Quantum Simulators**: For quantum-related experiments, we employed quantum simulators available in *Pennylane* and *TensorCircuit*. These frameworks were used to simulate quantum circuits in the absence of physical quantum hardware, allowing for experiments involving quantum gate-based computations.
    - **Deep Learning Frameworks**: The classical transformer models were implemented using *PyTorch* and *TensorFlow*, leveraging GPU acceleration where applicable.

4.2 Experiments

4.3 Results

4.4 Analysis

Chapter 5

Conclusion and Future Work

5.1 Summary of Contributions

5.2 Future Work

Appendix A

Research Proposal

A.1 Background

The advent of deep learning has been transformative, marking a paradigm shift in our approach to artificial intelligence. Prior to 2011-2012, deep learning was largely deemed impractical. This perception changed dramatically when a team participating in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 leveraged deep learning to secure a dominating lead, outperforming the second-place team by 9.8 percentage points—a margin that highlighted the difference between second and third place of 0.8.

The roots of deep learning extend back to 1943, yet it remained largely theoretical until recent advancements in hardware made its practical application feasible. Today, we stand on the shoulders of giants, benefiting from the perseverance of researchers who continued to explore this field despite its challenges.

Quantum computing holds similar transformative potential. The transformer model represents the next evolutionary step in deep learning, and by laying the groundwork for a quantum transformer, we can pave the way for future advancements.

Sipio et al. [11] proposed a quantum-enhanced transformer for sentiment analysis by adapting the multi-head self-attention and feed-forward layers to the quantum realm while Li et al. [6] introduced Gaussian Projected Quantum Self-Attention, which they used to create a Quantum Self-Attention Neural Network for text classification. They argued that this method is more suitable for handling quantum data than the straightforward inner-product self-attention used in the work by Sipio et al. [11]. Notably, in both models, the computation of attention coefficients and outputs remains within the classical domain.

In terms of performance, Cherrat et al. [2] provided results supporting the notion that quantum transformers may match the performance of their classical counterparts while requiring fewer resources in terms of runtime and parameter count. However, their approach has faced criticism, particularly regarding the exponential cost associated with encoding matrices into quantum states.

The field of quantum computing is poised to revolutionise deep learning, yet the integration of quantum mechanisms within neural network architectures remains under-explored. This study seeks to address the problem of how quantum components, specifically variational quantum circuits, can be effectively incorporated into transformer models to enhance their learning efficiency and reduce generalisation errors. Specifically, we seek to enhance the quantum-enhanced transformer initially proposed by Sipio et al. [11]. We hypothesise that a quantum transformer will require less training time and exhibit improved performance metrics compared to its classical counterparts.

A.2 Aim

To develop and validate a quantum transformer model that integrates variational quantum circuits into its architecture, thereby enhancing learning efficiency and reducing generalisation errors. This will involve:

1. Analysing the current limitations of classical transformer models in terms of learning efficiency and generalisation.
2. Designing a quantum transformer architecture that incorporates variational quantum circuits as a core component.
3. Comparing the performance of the proposed quantum transformer with classical models, focusing on training time and performance metrics.
4. Assessing the feasibility of the quantum transformer in practical applications, considering both its computational efficiency and the quality of its outputs.

A.3 Methodology

A.3.1 Data Collection

Before diving into our models, it is essential to outline the datasets employed in this study. We start with a lightweight dataset, which lays the groundwork for our initial analysis then transition to a more sophisticated dataset, presenting a richer set of challenges and opportunities for our models to tackle.

In this project, we will use three well-known text classification datasets: the IMDb, Amazon Polarity, and Yelp Reviews Polarity datasets, to explore the performance of our models.

The IMDb dataset consists of 50,000 movie reviews, making it ideal for natural language processing and text analytics. This dataset is designed for binary sentiment classification, where each review is classified as either positive or negative. With 25,000 movie reviews allocated for training and another 25,000 for testing, this dataset offers a robust amount of data for sentiment analysis tasks. Our model will predict the polarity of reviews, aiming to differentiate between positive and negative sentiments using deep learning or classification algorithms.

The Amazon Polarity dataset is a collection of product reviews sourced from Amazon over an 18-year period. The dataset includes approximately 35 million reviews up to March 2013. Reviews with ratings of 1 and 2 are labelled as negative (class 1), while reviews rated 4 and 5 are labelled as positive (class 2), with reviews rated 3 being excluded. The dataset provides 1,800,000 training samples and 200,000 testing samples for each class, making it a large-scale dataset well-suited for binary classification tasks. The inclusion of product and user information, ratings, and plaintext reviews enables us to further explore consumer sentiment through our text transformer.

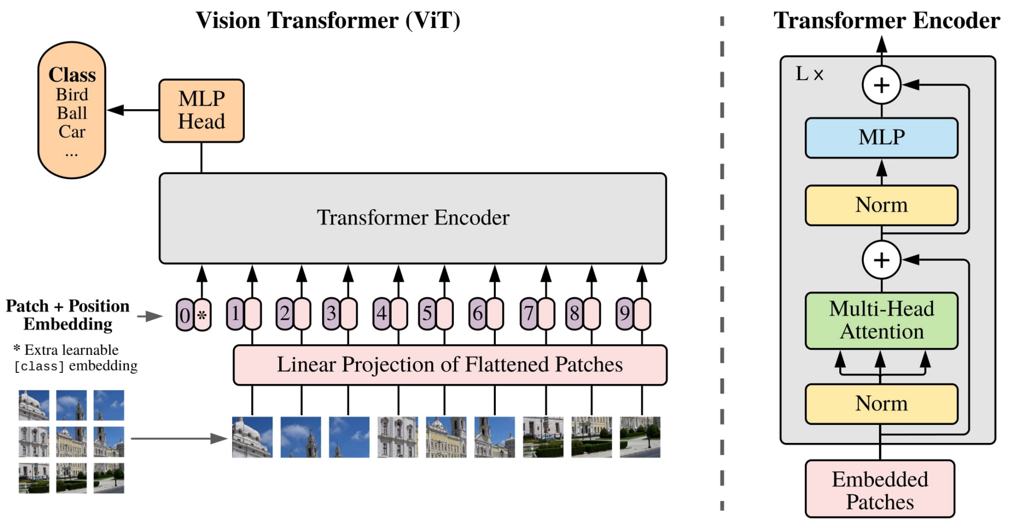
The Yelp Reviews Polarity dataset is derived from the 2015 Yelp Dataset Challenge and consists of 1,569,264 samples of review text. For this polarity classification task, reviews with star ratings of 1 and 2 are labelled as negative, while reviews rated 3 and 4 are labelled as positive. The dataset provides 280,000 training samples and 19,000 test samples per polarity class. This dataset provides another large and diverse set of review data that can be utilised to refine our text transformer model, further strengthening its ability to classify sentiment in various domains.

A.3.2 Models

Classical Transformer

We will implement a classical transformer based on the standard architecture as shown in Figure A.1. While the diagram represents the structure of a vision transformer, the overall structure of the text transformer will be similar, with key modifications to accommodate text-based input. We chose the standard transformer encoder model to serve as the baseline for our benchmark. The general steps of our text transformer are as follows:

1. The input text is tokenised into sequences of words or subwords using a pre-trained tokenizer.
2. Each token is then transformed into embeddings using a pre-trained word embedding model such as BERT.
3. Since the transformer processes the tokens in parallel, it does not inherently capture the order of the sequence. To address this, positional embeddings are added to the token embeddings to encode the sequential information.
4. The new embeddings, along with an additional special token embedding, are passed through the transformer encoder.
5. The output of the encoder is then processed by a multi-layer perceptron (MLP) head, which converts the token representations into a class prediction for sentiment analysis.



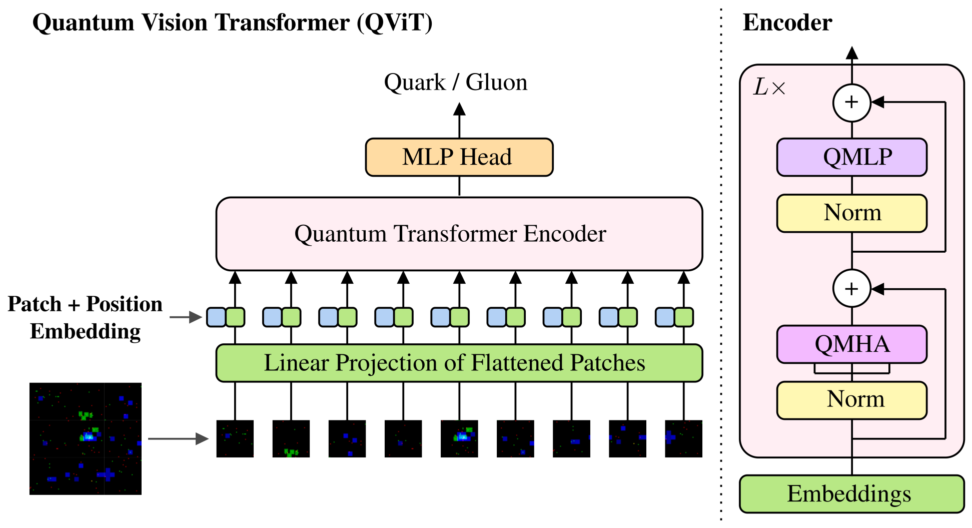
(a) Vision Transformer Architecture

Figure A.1: An overview of the classical model [4].

In the standard transformer encoder, we use multi-head attention, which employs several self-attention mechanisms to capture different types of relationships between tokens. The multi-layer perceptrons (MLPs) contain two layers with Gaussian Error Linear Unit (GELU) non-linearity to model complex interactions between tokens. Layer normalisation stabilises and accelerates training, while residual connections prevent the vanishing gradient problem.

This architecture will be adapted to text-based sentiment analysis tasks using datasets such as IMDb, Amazon Polarity, and Yelp Reviews Polarity.

Quantum Transformer


Quantum Vision Transformer (QViT)
Quark / Gluon
MLP Head
Quantum Transformer Encoder
Patch + Position Embedding
Linear Projection of Flattened Patches
Encoder
L×
QMLP
Norm
QMHA
Norm
Embeddings

(a) Quantum Vision Transformer Architecture

Figure A.2: An overview of the quantum model [1]. The illustration of the Transformer Encoder was inspired by Sipio et al. [11].

Cara [1] introduced a model depicted in Figure A.2, which incorporates variational quantum circuits into the multi-head attention and multi-layer perceptron components of the original architecture. These circuits serve as the equivalent of fully-connected layers within both components, potentially offering improved performance in specific tasks. As our comprehension of the quantum transformer deepens, the model may be further augmented with additional quantum components.

A.3.3 Training and Evaluation

To maintain consistency with the previous work by Cara [1] and Cherrat et al. [2], we will adopt similar hyper-parameters: cross-entropy loss function, Adam optimizer, 100 training epochs, a batch size of 32 and a learning rate of 10^{-3}.

For model performance evaluation, we will employ two metrics: the area under the receiver operating characteristic (ROC) curve (AUC) and accuracy (ACC).

A.4 Software and Hardware Requirements

A.4.1 Software

- Python: An easy-to-learn and really popular programming language to write quantum code.
- PyTorch: A Python library for deep learning.
- TensorFlow: Another Python library for deep learning.
- PennyLane: A Python library for quantum machine learning.
- TensorCircuit: Another Python library for quantum machine learning.
- Overleaf: An online latex editor for documentation.
- GitHub: An online hub to version control the code for this project.
- Visual Studio Code: A code editor to write Python code and SSH into Kaya.
- Hugging Face: A machine learning platform for sharing machine learning models and datasets.

A.4.2 Hardware

- Personal Laptop: A device to write and edit code.
- Kaya: The UWA High-Performance Computational Research Platform.

A.5 Timeline

The following comprises a rough timeline for the outlined project.

- Research proposal: 18th of March
- Quantum Computing Study: February - 14 October
  - Textbook Readings:
    - Explorations in Quantum Computing (Recommended by Jingbo)
    - Quantum Computing for Computer Scientists (Recommended by Microsoft)
    - Quantum Computation and Quantum Information (Recommended in the Quantum Computing Subreddit)
  - Interactive Learning:
    - IBM Quantum
    - Microsoft Azure Quantum
    - PennyLane Xanadu Quantum Codebook
- Implement a classical transformer from scratch: 18 March - 31 March
- Train and evaluate the transformer: April - May
- Literature Review: March - 13 May
- Implement a quantum transformer: May - August
- Seminar Abstract: August - 16 September
- Seminar: August - (30 September - 4 October)
- Thesis: August - 14 October

Appendix B

Pseudo Code

Appendix C

Training Speed Analysis

Appendix D

User Manual

Bibliography

[1] M. C. Cara. Quantum transformers in google summer of code 2023 at ml4sci, 2023. URL https://salcc.github.io/blog/gsoc23/.

[2] E. A. Cherrat, I. Kerenidis, N. Mathur, J. Landman, M. Strahm, and Y. Y. Li. Quantum vision transformers. *Quantum*, 8:1265, Feb. 2024. ISSN 2521-327X. doi: 10.22331/q-2024-02-22-1265. URL http://dx.doi.org/10.22331/q-2024-02-22-1265.

[3] M. Comajoan Cara, G. R. Dahale, Z. Dong, R. T. Forestano, S. Gleyzer, D. Justice, K. Kong, T. Magorsch, K. T. Matchev, K. Matcheva, and E. B. Unlu. Quantum vision transformers for quark–gluon classification. *Axioms*, 13(5), 2024. ISSN 2075-1680. doi: 10.3390/axioms13050323. URL https://www.mdpi.com/2075-1680/13/5/323.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[5] L. Huynh, J. Hong, A. Mian, H. Suzuki, Y. Wu, and S. Camtepe. Quantum-inspired machine learning: a survey, 2023. URL https://arxiv.org/abs/2308.11269.

[6] G. Li, X. Zhao, and X. Wang. Quantum self-attention neural networks for text classification, 2023.

[7] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran. Quantum embeddings for machine learning, 2020.

[8] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), Nov. 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07090-4. URL http://dx.doi.org/10.1038/s41467-018-07090-4.

[9] M. Schuld. Supervised quantum machine learning models are kernel methods, 2021.

[10] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3), Mar. 2019. ISSN 2469-9934. doi: 10.1103/physreva.99.032331. URL http://dx.doi.org/10.1103/PhysRevA.99.032331.

[11] R. D. Sipio, J.-H. Huang, S. Y.-C. Chen, S. Mangini, and M. Worring. The dawn of

quantum natural language processing, 2021.