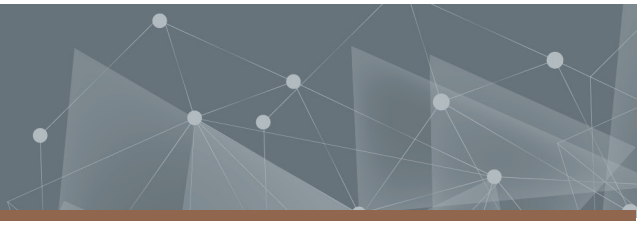
CHALMERS
UNIVERSITY OF TECHNOLOGY



Hybrid Quantum Transformer for Natural Language Processing

A hybrid transformer model using a quantum kernel as the attention mechanism

Master's thesis in Complex Adaptive Systems

WESLEY CONCEPCION

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

QNLP: A Hybrid Quantum Transformer for Sentiment Analysis
An evaluation of a quantum kernel self attention mechanism to capture higher order semantic relationships for natural language processing.
Wesley Concepcion
Department of Physics
Chalmers University of Technology

Abstract

Large language models, based on the transformer architecture, have revolutionized the field of Natural Language Processing by using deep learning techniques to capture complex linguistic patterns. This thesis explores a hybrid architecture for a transformer model for natural language processing. Specifically, a hybrid quantum transformer which incorporates a quantum self-attention mechanism for sentiment classification on the IMDB dataset. The hybrid quantum transformer leverages the strengths of both classical and quantum computing to enhance the performance of the sentiment analysis for natural language processing. The model aims to capture more complex semantic relationships and addresses the potential of quantum computing compared to classical computation. The quantum self-attention module computes the similarity measure between input tokens by first embedding the data in the Hilbert Space of quantum states, followed by an inner product between quantum states. Comparative analyses are performed against current work on quantum and classical transformer architectures for sentiment analysis. The quantum kernel archictecture is benchmarked against an attention-less transformer and previous implementations for a quantum transformer [27]. Although the quantum kernel transformer performed marginally better in terms of accuracy on the test set over five epochs, the runtime for a simplified quantum kernel transformer significantly highlighted the drawbacks for kernel computations. The long run-time ruled out the feasibility for a more complex set of hyperparameters to match the capabilities of a classic transformer. Future work would explore tuning of the hyperparameters and adding a trainable parameterized layer in the quantum circuit for the kernel to allow for a trained element to the architecture.

Acknowledgements

This thesis project would have not made it very far without the guidance from my advisor, David Fitzek and my supervisor, Mats Granath. I appreciate all of their help and insights throughout the course of the project. I would also like to thank my classmates for support and my opponent Tarek Alhaskir for his valuable questions and interrogations on my presentation and topic.
Lastly, I would like to thank my family, and my partner, Anna Rosén for all of their continued support and love they have given.
Computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) and the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE), partially funded by the Swedish Research Council through grant agreements no. 2022-06725 and no. 2018-05973.

Wesley Concepcion, Gothenburg, June 2023

Contents

List of Figures

List of Tables

1

Introduction

The development of quantum computing and quantum computers across several platforms and technologies has shown a quantum advantage using quantum simulation for simple artificial problems [4]. However, these problems simply show that quantum advantage is achievable, albeit the problems are not of practical importance.

Quantum computers today are limited to *noisy intermediate-scale quantum* (NISQ) devices, which provide limited connectivity and noisy qubits. The imperfections in the quantum hardware, such as temperature fluctuations, electromagnetic interference, and measurement errors contribute to this noise and can lead to decoherence. Today's quantum computers only have a number of qubits on the order of 100's of qubits, and this limited qubit count makes it challenging to perform large-scale problems compared to the vision for certain quantum algorithms, which makes the much anticipated algorithm like Shor's algorithm out of reach for the time being [21].

Quantum machine learning (QML) is an emerging field at the intersection between quantum computing and classical machine learning aimed at harnessing the unique properties of quantum systems to enhance traditional data analysis techniques. As quantum computers continue to evolve and demonstrate computational power, exploiting their functionality in the realm of machine learning becomes increasingly attractive.

Quantum machine learning has the potential to provide an advantage in solving certain problems that classical computers cannot solve in polynomial time.

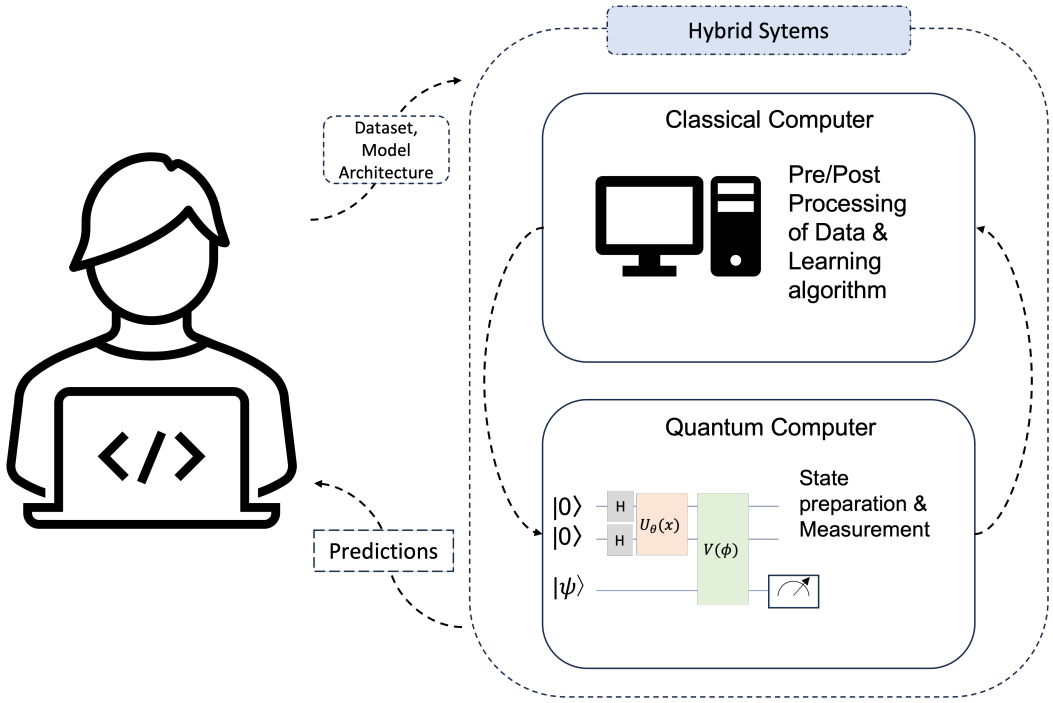


Figure 1.1: Example diagram for a hybrid machine learning model. In this figure, the classical computer does the pre and post processing of the data, in addition to adjusting the learning rates. The classical computer feeds the quantum computer with an initial state and quantum computer performs the measurements after the final state is reached and feeds that measured state back to the classical computer. The final prediction for the given prompt is then given as a result to the user.

Hybrid quantum algorithms are in the class of algorithms that combine the power of classical computing and quantum computing to solve a certain problem. The classical computer handles the computations that it can perform efficiently, such as data processing, post-processing of results, and optimization of the algorithm, while the quantum computer focuses on the tasks that can benefit from the unique physical properties of quantum mechanical systems. Figure 1.1 gives a high-level overview of a hybrid model structure. When constructing a hybrid quantum algorithm, one often looks to classical algorithms to find inspiration. As can be seen in figure 1.2, an obvious similarity comparison can be made between quantum computing and kernel methods. Quantum computing and kernel methods have many similarities. As will be explained in this thesis, both methods allow vector computations in feature spaces that are potentially inaccessible otherwise [25]. This can be accomplished without the need to compute an explicit mapping of the vectors. Kernel methods for machine learning are ubiquitous for pattern recognition.
One field where such pattern recognition is a necessary tool is in natural language processing (NLP). Large language models (LLMs) are sophisticated models that are trained on a vast amount of text data to acquire a deep understanding of language and can be used to generate human-like responses. These models are typically based on deep learning architectures such as recurrent neural networks (RNN's) [17] or transformer networks [29]. Transformers are deep learning models that specialize in sequential modeling. A block diagram of the transformer architecture can be

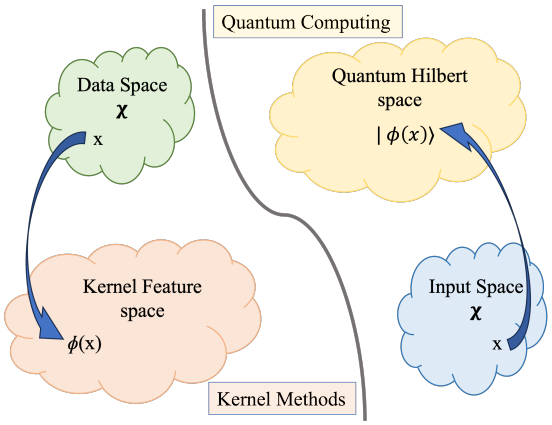


Figure 1.2: Comparing kernel methods mapping to a feature space, and quantum computing mapping to Hilbert space.

seen in Figure 1.3. The transformer has revolutionized the field of NLP due to their ability to capture long-range dependencies and parallelize computations effectively. Central to transformers is the concept of attention. Attention can be thought of as the relationship between different words in the text, or how each word attends to other words in the text. Given that human language follows patterns regardless of differences in origin, it becomes a good candidate for the hybrid model to detect patterns. We can use this natural language application as a medium to implement a hybrid quantum transformer.

1.1 Aim

As alluded to in the introduction, the transformer is a powerful architecture to handle natural language processing problems [2] [29]. It has been previously shown that quantum kernel methods have had a big impact on supervised learning tasks and in big data [8] [22] [25]. A natural progression from these two perspectives is to marry the two fields. This brings us to the hypothesis of the thesis.

Hypothesis: A hybrid quantum transformer can be implemented for Natural Language Processing using a quantum kernel as the attention. At the time of writing this thesis, there have been related work implementing a quantum enhanced transformer for sentiment analysis which uses a variational quantum circuit for attention and as a feed forward network [27], as well as a self-attention for text classification [13], and the use of multiple techniques for the implementation of a quantum vision transformer [3]. To use a quantum kernel as the attention mechanism for the transformer model to perform NLP Sentiment Analysis is a natural progression from the literature. We can ask many questions in pursuit of this goal.

Research Question 1: Can we implement a quantum kernel as the attention mechanism for Transformer models?
This research question is structured as a proof-of-concept to demonstrate the feasibility of integrating a quantum kernel in place of the attention of an encoder-only

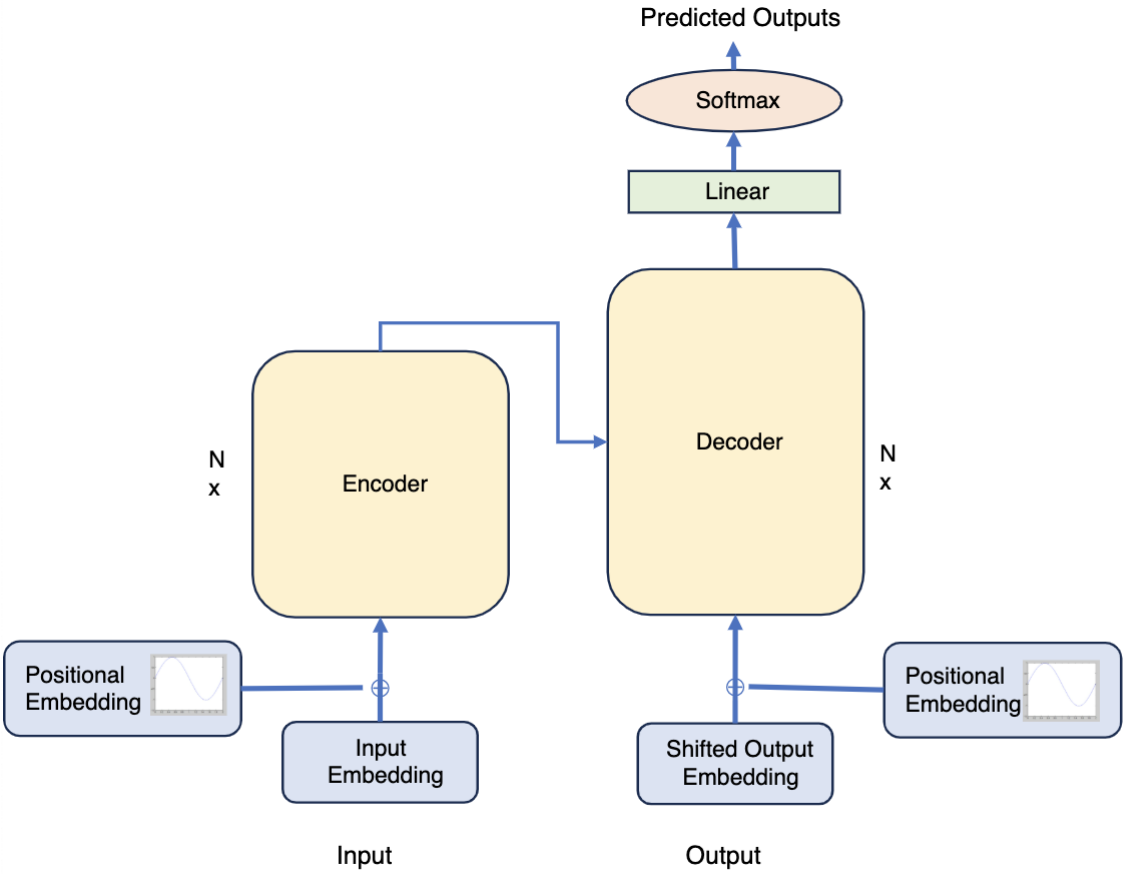


Figure 1.3: A simple block diagram of the full encoder/decoder tranformer.

transformer.
Research Question 2: Can the quantum kernel provide a computational speedup compared to the use of a classic self attention of the transformer model?
This is a fundamental question regarding any hybrid quantum machine learning algorithm. If the quantum kernel would provide a computational speedup compared to a classical attention of the transformer model, we would be implying quantum advantage which is a large milestone in the field of quantum computing.
Research Question 3: Does the training of a large language model using a quantum attention result in higher accuracy compared to a classic attention model?
As a follow up to **RQ2**, alternatively to a computational speed up, we can ask if the training of the hybrid model provides greater accuracy compared to the classical model. This is another evaluation parameter which would indicate quantum advantage, and is a fundamental question in the field if hybrid systems can out-perform classical ones.
Research Question 4: How does this implementation of a hybrid transformer with quantum kernel attention compare to previous work done using a variational quantum circuit as the attention?
At the time of writing this thesis, I am only aware of one other work on a hybrid quantum transformer, notably the work done by DiSipio and colleagues [27]. His work provides a benchmark to compare my implementation of a hybrid quantum transformer.

1.2 Structure

This thesis begins with an introduction to the field of natural language processing, machine learning, and quantum computing. Since the scope of the thesis combines these three fields, a fundamental understanding is given as relevant background information to rationalize and effectively interpret the results. The most basic principles of NLP are given along with necessary definitions as well as the specific tasks that can be solved, and different applications thereof. In order to understand the architecture of a transformer, a basic explanation of how a machine can learn from data and how we can leverage the kernel trick to assist in the learning by way of a feature map. Finally, the principles of quantum computing will be explained because we will need to leverage quantum data encoded in a quantum circuit for our attention mechanism. I will discuss the two quantum algorithms explored in this thesis, the variational quantum circuit, and the quantum kernel estimator.
After the required fundamental groundwork has been established, I will provide a framework for how to apply a hybrid quantum transformer with a quantum attention mechanism. Starting from the input dataset, I will navigate the process of how the data is prepared for the model, and then follow what is happening in the model itself, highlighting and explaining relevant features of the transformer and how they work along the way. I will first introduce the dataset, and how we break down the input strings of the dataset such that the inputs are in a more consistent and readable format for the model. In addition to the architecture of the transformer, I will explain how a quantum attention mechanism is incorporated into the architecture of the model. Lastly I will explain the different evaluation metrics and how the

model can learn from the data.
The evaluation chapter will dive into the performance of the model and how it compares to previous work on transformers for NLP. The first evaluation I will benchmark is a transformer without attention. This architecture without attention is a multi-layer perceptron (MLP) with positional encoding and token embeddings. I will then compare the classical transformer architecture against two quantum variations: the quantum transformer with a variational quantum circuit (VQC) and my implementation of the quantum attention using a quantum kernel. The primary metrics used for comparison are accuracy and loss. Because of the lack of scalability of the quantum models at this stage, we will look into hyper parameter tuning, the size of the dataset used and the impact on performance of the model for evaluating sentiment. By comparing these metrics across the classical transformer, quantum transformer with VQC, and hybrid transformer with quantum attention, we gain insights into the relative performance and potential benefits of incorporating quantum elements into the Transformer architecture.
Finally, the conclusion will summarize the results. I will also outline the limitations and challenges faced throughout this work. Throughout the thesis, the scope and scale of the project had to be revised given the complexity of the task. In the conclusion, I will highlight these challenges and future directions that can be explored that will further investigate the potential of hybrid quantum transformers and how we can continue to improve the performance.

2
Background

Natural Language processing can be thought of as the intersection of computer science and linguistics. Quantum computing is the intersection of quantum physics, computer science and software engineering. Moving more technical, large language models like BERT, GPT, and their successors are the intersection of machine learning and natural language processing. This section will break down these individual parts and provide the necessary information to understand the implementation of my hybrid quantum transformer.

2.1 Natural Language Processing

Natural Language Processing (NLP) is an interdisciplinary sub-field comprised from linguistics, computer science, and artificial intelligence, in which a machine can process, understand, and use the human language. As a generalization, Natural Language Processing can be broken down into two subfields: Natural Language Understanding (NLU), and Natural Language Generation (NLG). NLU focuses on the machine's ability to understand, process, and interpret the human language, while NLG focus on text generation. In both cases, the machine needs to break down the input text, understand the meaning of the input, and give an appropriate response, whether the response is a sentiment, text summary, translation, etc.

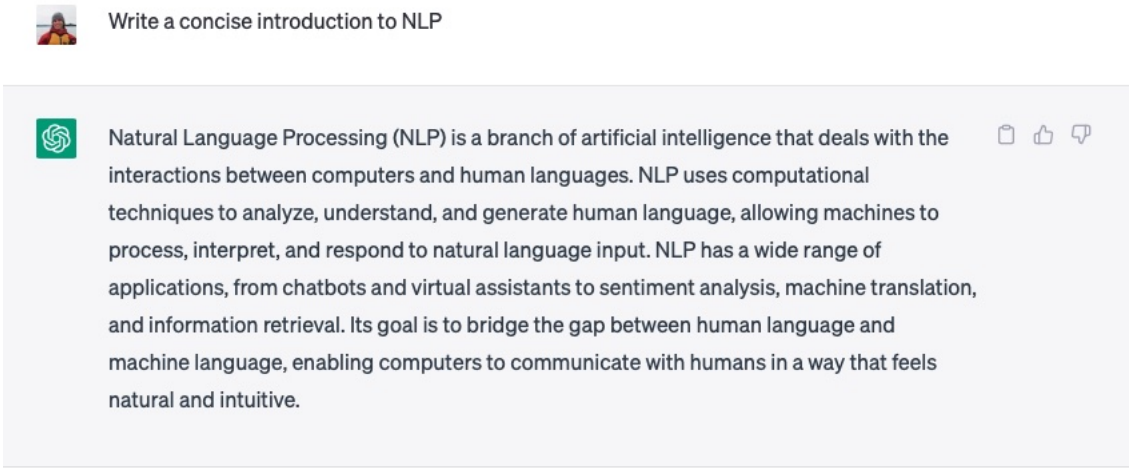


Figure 2.1: Example response from ChatGPT (GPT-3). This is an example of question/answer, a sub-category of natural language processing.

NLP has been making waves through the media recently for advancement in Large Language Models (LLMs), which use large datasets to pre-train their models, a computationally expensive task, and deploy the model while continuously fine-tuning based on feedback the model is given by users. One mainstream example of this is ChatGPT, a generative-pre-trained (GPT 4.0) chatbot which OpenAI deployed in March 2023 [19]. An example use case of the chatbot is shown in figure 2.1.

NLP can be used to accomplish a variety of tasks concerning the human language. The following are a few examples of common NLP tasks.

Table 2.1: NLP tasks

NLP Task	**Description**	**Example Text**	**Result**
Text Classification	Categorizing emails as spam or not spam	"Congratulations! You've won a free trip to Hawaii. Click here to claim your prize!"	Spam
Sentiment Analysis	Determining sentiment polarity of customer reviews	"The movie was fantastic. I loved every moment of it!"	Positive
Named Entity Recognition	Identifying names of people, organizations, and locations in a text	"John Smith works at Acme Corp in New York City."	Person: John Smith, Organization: Acme Corp, Location: New York City
Machine Translation	Translating a sentence from English to French	"Hello, how are you?" (English)	"Bonjour, comment ça va ?" (French)
Question Answering	Providing answers based on a given text and questions	"Q: What is the capital city of France?"	Paris
Text Summarization	Generating a concise summary of a long article or document	"Text: "The advancements in artificial intelligence and machine learning...transform many aspects of our lives.""	'Artificial intelligence and machine learning advancements have transformed healthcare, autonomous vehicles, finance, and entertainment industries, enabling early disease detection, self-driving capabilities, fraud detection, personalized content recommendations, and improved outcomes.'
Speech Recognition	Converting spoken language into written text	Spoken phrase: "Navigate to the nearest coffee shop."	"Navigate to the nearest coffee shop."
Language Generation	Generating human-like text, such as dialogue or stories	"Once upon a time, in a land far away..."	Once upon a time, in a land far away...

Text Classification

Text Classification is the technique in NLP which categorizes a given text into predefined categories or groups based on its context or meaning. Text classification such as spam filtering is one example in which classification of text data is used in our every day lives within email servers.

Sentiment Analysis

Sentiment Analysis is a part of NLU, in which the goal of the model is to identify and extract subjective information about the input text. These sentiments can be positive or negative, or be assigned a value, for example $1 - 5$, with 1 being most negative, and 5 most positive. Companies might use these models for social media monitoring, ranking product reviews, or threat detection.

Information Retrieval

Used to extract information from a document or text. We can use this for plagiarism scanning etc.

Question/Answering

Question and Answering is a major subsection in NLP which enables a machine to understand and respond to questions formed in natural language. As an extension to information retrieval, the machine can extract the relevant information from a given set of documents or knowledge base and provide a response to the user input. The question first needs to be analyzed to understand structure, meaning, and the information requested. The machine then summarizes the question, by identifying keywords, and determining which type of question is being posed (factual, opinion-based, list-based, etc). Then the formulated answer needs to be retrieved from the set of documents or knowledge base. This is accomplished via a document search via keyword matching, semantic similarity etc.

Summarization

Text summarization is the process of collecting a concise and coherent summary of a longer text. This can involve information retrieval to extract the most important information and the main ideas from the original text while maintaining the meaning and context. Summarization can be used when dealing with information overload, which allows the user to quickly grasp the key ideas from a publication or text.

Machine Translation

Machine Translation is the process of automatically translating from one language to another. There are two main approaches to machine translation. The first is Rule-based Machine Translation (RBMT). Rule based relies on linguistic rules and dictionaries to translate text. The machine would then follow the rules which specify how to transform the sentence structure, words or phrases from the source language to the target language. The other approach is statistical machine translation (SMT).

2.1.1 Applications

Natural language processing can be applied almost anywhere a form of communication is present. Primitive forms of natural language processing came in the forms of Google Assistant, Amazon Alexa, or customer support chatbot with question/answer type responses.

2.2 Machine Learning

Machine learning in a general sense, is a field of computer science which uses statistics to learn from input data, without being explicitly programmed. The given dataset is split into two subsets, a training set and a test set. As can be inferred from the name, a training set is the data in which the machine learning model is trained on, and the test set is the set of data that the model is evaluated. We can subdivide the training in to three major classes of training, supervised, unsupervised and reinforcement learning. In supervised learning, the algorithm learns to map the input data to output data based on specified labels provided in the dataset. The goal is for the model is to learn the correct patterns on the input data which result in an output label, such that it can make accurate predictions on new, unseen data. During the training, the algorithm tries to minimize the error between the predicted output and the true output values. Alternatively, unsupervised learning does not have the available labels to learn from, so the algorithm must learn from underlying patterns or relationships found between the data points.

Throughout training, typically only two parameters are of concern. In this thesis, the model will try to minimize the loss and maximize accuracy on a labeled dataset, with the labels being either 1 or 0. If the transformer produces an output that is consistent with the provided label, it will return a 1, and a 0 if the output does not match the label. Further explanation of the evaluation of the model will be described in the following chapter.

2.2.1 Kernel Theory

Kernel theory is an important concept in machine learning which allows for analysis of complex functions, particularly non-linear and non-paramaterizable models. To understand the fundamentals of kernel theory, we can revisit a simple problem of binary classification. If we have a set of datapoints, we need to construct a classifier to predict the class of the data points. A simple linear classifier is one such a line in two dimensions or features (x and y), or hyperplane in higher dimensions, can divide the dataset in to two groups, and assigning labels to each of the two groups on opposite sides of the line. A basic linear binary classifier can be visualized in figure 2.2. Algorithms like the support vector machines (SVM) make use of the kernel trick for the computation of the dot product in high dimension feature space, with the goal to maximize the margin of separation between classes of data [23]. The other common kernel algorithm is the principal component analysis (PCA), which can be thought of as distance algorithms for finding norm based distances between

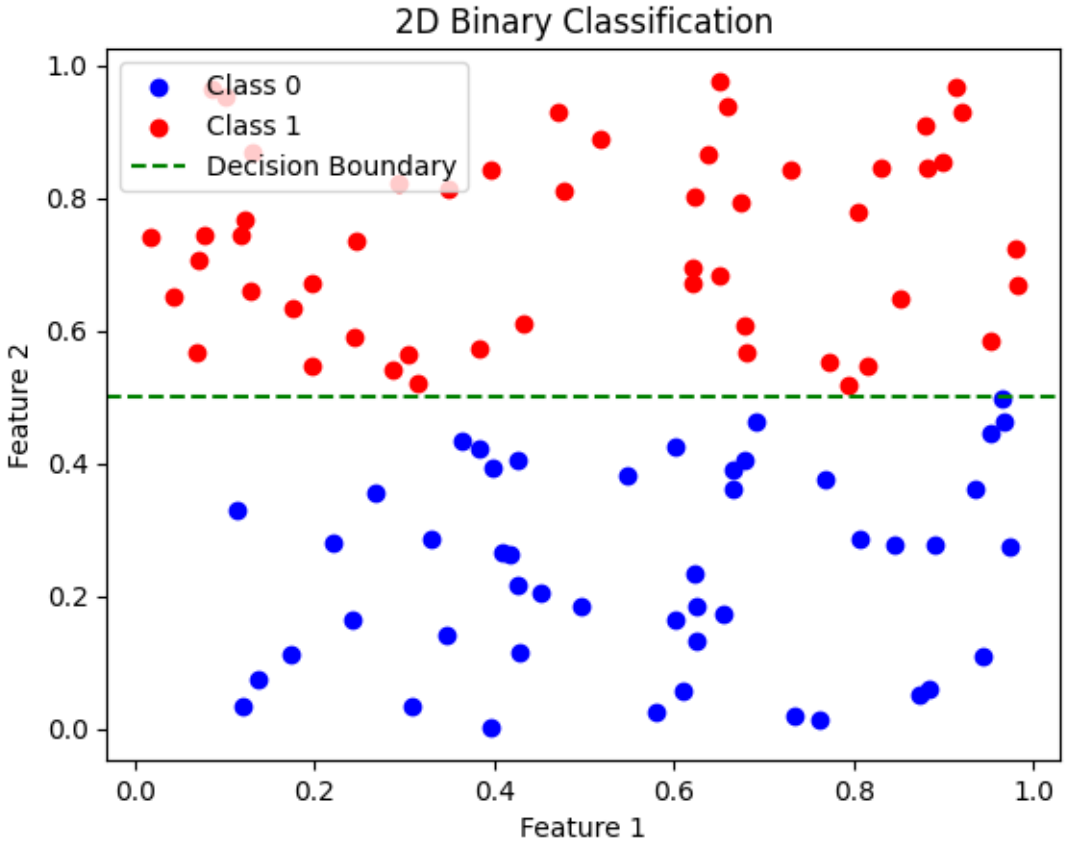


Figure 2.2: Basic binary classifier, which categorizes a two featured input data in to one of two classes. It can be seen that the green linear line separates the two classes and the classes are subsequently labeled as Class 0 or Class 1 depending on where they fall above or below the classification line.

inputs in Hilbert space.

The kernel function is a similarity measure between the data points in the input space. Classical kernel functions include the linear kernel, polynomial kernel, gaussian kernel, and sigmoid kernel, among others [9]. Each kernel function is suitable for different types of data and problems. All kernels however, result in the squared inner product of the data points in a feature space. This mapping to the feature space is performed by a feature map. The kernel trick is a crucial idea in kernel theory which allows for the computation in high-dimensional feature space be cast in terms of dot product, without explicitly computing the feature mapping. This allow for the kernel function (κ) to operate in the input space similar to if they were operating in the high dimensional feature space F. Formally, input data, x, is mapped from the data space, χ, to the feature space, F, using a feature map, ϕ: $\chi \to F$, $\mathbf{x} \to \phi(x)$ [26].

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \tag{2.1}$$

2.3 Quantum Computing

In the previous section, we explored the ideas of classical machine learning and kernel methods specifically. In this section, we are going to talk about a more nuanced and complex form of computing: quantum computing. Quantum computing is a unique form of computing in which qubits are used instead of bits, and quantum algorithms replace their classical counterparts to perform operations.

To explain the fundamentals of quantum computing, we can first begin to understand how classical computing works. It is then possible to draw comparisons from which we can base our understanding. Classical computers run on bits (short for binary digit), which can take the form of 0 or 1. Bits represent a logical state of two possible values to store information. These can take the representation of true/false, yes/no, $+/-$, or simply 0 or 1. Multiple bits can be combined to form bytes, which can represent larger pieces of information, including integers, floating point numbers, or characters as text strings. Bits can also be used to represent instructions to operate a machine. A computational *algorithm* is a sequence of these instructions to instruct the computer to perform a specific task by manipulating the states of these bits. Quantum computing can be thought of as the manipulation of quantum information to perform a specific task by manipulating the quantum states. There are many different approaches to build a quantum computer for quantum information processing, including superconducting qubits [12], trapped ion quantum computing [1], and photonic quantum computing [7], among others. All of which must satisfy DiVincenzo's criteria for the characteristics of a quantum computer [5]. These have become the main guideline and standard for building a quantum computer.

- A scalable physical system with well characterized qubits
- The ability to initialize the state of the qubits to a simple fiducial or reference state (such as $|000\rangle$)
- Long relevant coherence times, much longer than the gate operation time
- A 'universal' set of quantum gates
- A qubit-specific measurement capability

2.3.1 Qubits

In quantum computing, information is represented by qubits, or quantum bits. Instead of representing 0 and 1 like a classical bit, a quantum bit is represented as a linear combination of states, which is often described as a superposition of a ground state $|0\rangle$, and excited state $|1\rangle$, and can be mathematically described by the state ψ as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{2.2}$$

The numbers α and β are complex numbers. The states $|0\rangle$ and $|1\rangle$ are special states and form an orthonormal basis for the vector space, and are known as the *computational basis states* [18]. If we have N qubit in our system, the system can be in 2^N different states. The state of an N-qubit system can be described by the tensor product of states, for example the 2-qubit state is given as $|00\rangle \equiv |0\rangle \otimes |0\rangle$. A classical

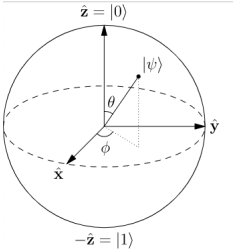


Figure 2.3: The Bloch sphere is a geometric representation of the pure state space of a two-level quantum mechanical system (qubit). Each point on the surface of the sphere corresponds to a unique quantum state. The north and south poles typically represent the basis states $|0\rangle$ and $|1\rangle$, respectively, while any other point represents a superposition of these states.

system with N bits can be in one of these 2^N states, where our quantum system can be in a superposition of all 2^N states. When we perform a measurement on the qubit, we cannot explicitly examine the quantum state, rather sample properties of the system. A measurement will collapse the state into either the $|0\rangle$ state with probability $|\alpha|^2$, or $|1\rangle$ state with probability $|\beta|^2$. Naturally, all of the probabilities must sum to one, i.e. $|\alpha|^2 + |\beta|^2 = 1$. Geometrically, we can rewrite equation 2.2 as:

$$|\psi\rangle = cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.3}$$

The angles θ and φ correspond to angles on the *Bloch Sphere* which can be visualized in figure 2.3. The operations we perform on the qubits to manipulate the state of the quantum system can be seen as, for a single qubit, rotations around the Bloch Sphere. As we will see in the following section, we can define single qubit gates as specific operations to rotate the qubit around the Bloch sphere. We can the string together sequences of these operations on one or more qubits to form our *quantum algorithm*.

2.3.2 Qubit Gates

The composition of quantum algorithms begins with the formalism of quantum gates. Operations on a single qubit can be described by a 2x2 unitary matrix. The most common are the Pauli matrices:

$$X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \quad Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; \quad Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.4}$$

By combining the X and Z gates, we obtain the Hadamard gate, which transforms the qubit state from the $|0\rangle, |1\rangle$ basis to the $|+\rangle, |-\rangle$ basis, and vice versa.

$$H = \frac{X+Z}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.5}$$

If we were to make a measurement on the $|+\rangle$ state, we would observe the $|0\rangle$ state with a probability of $1/2$, and we would observe the $|1\rangle$ state with a probability of

1/2. By adding the Hadamard gate to the circuit, we have created a superposition state. In order to do useful things with quantum algorithms which a classical algorithm could not, we need to create entanglement or interactions between two qubits through multi-qubit gates. Entanglement can be described as a highly correlated interaction between two particles. Qubits are entangled if the state of the system cannot be described as a product of two individual qubits, but must be considered as a whole [6]. We can apply a controlled operation of a second qubit depending on the state of the first. Typical application is a Controlled-NOT (CNOT or C-X) or controlled-Z (CZ) gate, where we can do nothing to the second qubit if the first in state $|0\rangle$, and apply the unitary to the second if the first is in state $|1\rangle$.

$$CU = \begin{pmatrix} I_2 & 0_2 \\ 0_2 & U \end{pmatrix} \tag{2.6}$$

Where U is the unitary matrix for a single qubit. For example, we can look at a CNOT gate, if the first qubit is in the state $|0\rangle$, ($|00\rangle, |01\rangle$ remains unchanged to $|00\rangle, |01\rangle$), and would apply a NOT (X) gate if the first qubit is in the state $|1\rangle$, ($|10\rangle, |11\rangle$ changes to $|11\rangle, |10\rangle$). In matrix form, this CNOT gate is represented by:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.7}$$

The other two qubit gate that we will pay attention to at this stage is the SWAP gate, which is not a controlled gate, but also creates entanglement. This gate will swap the states of the two qubits:

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.8}$$

Physical limitations of many quantum computational systems allows for limited nearest neighbor connectivity, which prohibits direct implementations of multi qubit gates of order three or more. However qubit gates of three or more can be decomposed into single and two-qubit gates [6]. Furthermore, universal quantum computation is achievable if the gate set can approximate any unitary transformation on an arbitrary number of qubits to any desired level of precision [6]. By defining a universal gate set, we can now perform universal quantum computation with which we now can construct quantum algorithms to accomplish a given task. With quantum computing still in its infancy, we are left with noisy-intermediate-scale quantum (NISQ) devices. Given current state of quantum devices, quantum algorithms are being developed via simulation in which programs can efficiently simulate quantum systems on classical computers. The Gottesman-Knill theorem states that only a quantum circuit composed of the following elements can be simulated on a classical computer:

- Preparation of the qubits in the computational basis

- Gates from the Clifford group (H, CNOT, S) and
- Measurements in the computational basis

Restricting our quantum circuits to the requirements above, we can make the circuit classically simulatable even though it contains superposition, entanglement, and complex amplitudes [6]. However, in order to work with these systems, we need to work with quantum data.

2.3.3 Quantum Data

In order to work with classical data on a quantum computer, it needs to be prepared in a form which can be read by a quantum circuit. There are different approaches to data encoding and each has their respective applications, advantages and disadvantages.

Basis Encoding

Basis encoding encodes a classical N-bit string into an N-qubit state. The binary representation of the input x is given as $x := b_{n-1}b_{n-2}...b_1b_0$ is encoded into the N-qubit state given by $x := |b_{n-1}b_{n-2}...b_1b_0\rangle$ [31]. Formally the binary transformation of state x is given by:

$$X \approx \sum_{i=-k}^{m} b_i 2^i \mapsto |b_m \dots b_{-k}\rangle \tag{2.9}$$

The state is encoded in a specific basis, typically the computational basis, or the eigenbasis of a specific observable. To encode the data in this basis, one can apply a Pauli-X, or Pauli-Z gates to the qubits. These gates will manipulate the individual qubits and transform the state according to the desired basis encoding defined by the problem at hand. For example, if we want to encode the classical string "10", we need to assign the appropriate probability amplitudes to the basis states corresponding to the classical bits. The basis states of the 2-qubit system are: $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. The classic bit string tells us we want to have the first qubit in the $|1\rangle$ state, and the second qubit in the $|0\rangle$ state. Therefore all other amplitudes (i.e. $|00\rangle$, $|01\rangle$, $|11\rangle$) will be zero. This however identifies a drawback of using basis encoding. We will need a large amount of qubits to represent the data, given the binary representation of numbers. As the values for our vectors grow, the circuit will become very sparse [6].

Amplitude Encoding

Amplitude embedding is another way to encode classical data into a quantum state. Every quantum system can be described by a wave function ψ which also defines the measurement probabilities. We use the amplitude of the wave function to encode the data. Therefore the amplitudes of the quantum system are used to represent the data values. The advantage of amplitude encoding is that it only requires $log_2(n)$ qubits to encode. For example if we have a vector $\mathbf{x} = (x_0, x_1, ..., x_n)^T$, we can

normalize to length 1 and encode using:

$$|\psi\rangle = \sum_{i=0}^{n-1} x_i |i\rangle \tag{2.10}$$

If we have data vectors of different lengths, we need to ensure the lengths of the data to be encoded is a power of 2, so padding to length is necessary. Amplitude encoding seems like a promising and simple way to represent data, however the storage of such data requires QRAM or quantum random access memory, and to date, we are unsure how to build such a memory device [6].

Angle Encoding

To map the data to a given angle encoding, we need to normalize the data over the interval $[0, 2\pi)$. Each value of the vector $x_i = (x_0, x_1, ..., x_n)^T$ can be encoded using a Rotation gate ($R(\vartheta)_{x,y,orz}$), where the rotation angle allows for unique encoding by a specified angle [30]. We leverage this to encode the data by rotations around the Bloch sphere by a relative phase. The angle encoding is efficient in terms of the number of qubits needed to encode the data. In theory, one could encode a rotational gate to infinite precision, however limitations on the accuracy of the hardware prevent practical implementation of such precision.

2.4 Quantum Machine Learning

Having outlined the components of a quantum circuit model to encode data into a quantum circuit, the question now becomes what can we do with classical data in a quantum circuit. Quantum computers have the potential to provide a computational speed up compared to their classical counterpart, but are still to early in the development for practical use. Quantum computers are still in the noisy intermediate scale quantum computer (NISQ) level, which errors and limited qubit connectivity limit application toward complex problems [21]. Quantum computing on the other hand, deals with quantum algorithms, which can be simulated on a classical machine. Quantum algorithms have given rise to quantum machine learning (QML), a sub-field in which a quantum model can learn from data. Machine learning and quantum computing have the potential to solve previously untenable problems. Of these algorithms, this thesis will explore the application of Variational Quantum Algorithms (VQA), and Quantum Kernel methods.

2.4.1 Variational Quantum Circuit

The general concept of a variational quantum circuit (VQC) is to vary the parameters of the circuit and iteratively vary the parameters of the circuit until an objective function has been minimized such as the mean square error (MSE) between the output and the labels. VQA's are a class of hybrid quantum models which the quantum computer or simulator will encode the data into a variational circuit with a certain set of parameters and efficiently estimate the cost function (or it's gradients). The weights are trained as tunable rotations (θ) for each qubit embedding. Schuld et

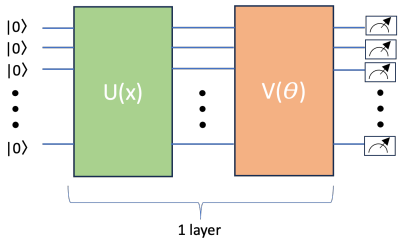


Figure 2.4: Generic block diagram for a variational circuit. The weights are trained as tunable rotations for each qubit embedding, and we can stack these circuits to increase the capability of the model.

al. proposed these circuits with multiple layers to increase the model's capabilities [25]. This information is then fed into a classical optimizer, and the optimizer will navigate the cost landscape and solve the optimization problem. Once a termination condition is set, the VQA will output the estimate of the solution.

2.4.2 Quantum Kernel Methods

As mentioned in section 2.1.1 on kernel methods, the kernel trick allows for high dimensional data to be computed in the lower dimensional space. We can use these fundamentals in the quantum space as well. The states of the quantum systems are vectors in high dimensional Hilbert Space. The Hilbert space of an n-qubit quantum computer is the vector space C^N where $N = 2^n$. The quantum embedding kernel is simply the overlap of the two quantum states in the Hilbert space. One way to evaluate relationships between data is to evaluate the similarity between the two datapoints. Conveniently, the inner product of two quantum states is just that.

$$\kappa\left(x, x'\right) = \left\langle 0 \ldots 0 \left| V_\phi(\mathbf{x}) V_\phi^\dagger\left(\mathbf{x}'\right) \right| 0 \ldots 0 \right\rangle \tag{2.11}$$

The benefit of kernel methods is to provide a feature map from an high dimensional Hilbert Space to a reduced dimensional feature space for which we can linearly classify data.
As discussed in the introduction, a quantum kernel can be used to provide a computational speedup compared to a classical kernel. However, we need to implement a feature map that is hard to simulate classically. Havlicek et al. describe a feature map (figure 2.5) generated by the unitary, $\mathcal{U}_{\phi(\mathbf{x})}$, of such a mapping on n-qubits[8].

$$\begin{aligned} \mathcal{U}_{\phi(\mathbf{x})} &= U_{\phi(\mathbf{x})} H^{\otimes n} U_{\phi(\mathbf{x})} H^{\otimes n} \\ U_{\phi(\mathbf{x})} &= \exp\left(i \sum_{S \subseteq [n]} \phi_S(\mathbf{x}) \prod_{i \in S} Z_i \right) \end{aligned} \tag{2.12}$$

This feature map is known as the ZZ-feature map. This feature map is named after the Pauli-Z tensor product of quantum states which embeds classical data into the quantum feature space. The qubits are encoded with a Hadamard and the rotational z-gate, and are entangled in the section with the ZZ gates. The circuit diagram for the ZZ feature map can be seen in figure 2.6.

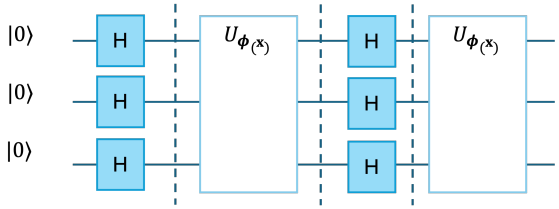


Figure 2.5: The unitary, $\mathcal{U}_{\phi(\mathbf{x})}$, is composed of the conventional Hadamard, and $U_{\phi(\mathbf{x})}$, which is diagonal in the Pauli Z-basis. The circuit will act on $|0\rangle^n$ as the initial state.

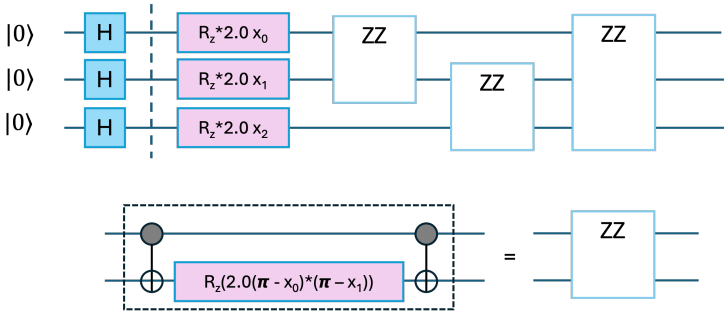


Figure 2.6: An example 3 qubit circuit diagram for the ZZ Feature map proposed by Havlicek et al. The circuit diagram is a combination of angle encoding and the ZZ-feature map entangling. In this case, only $|S| \leq 2$ interactions are considered.

Once the classical data is mapped to the quantum space, we can then perform the inner product of these states to get a similarity measure between data points.

$$K(x, y) = |\langle \phi(x) | \phi(y) \rangle|^2 \tag{2.13}$$

By embedding the data into a quantum feature space, we can use a parameterized quantum circuit to map the data point into the quantum state [10]. The kernel value is the overlap of the parameterized quantum states. The mutual kernel values from all data-points in the set form the kernel matrix. In our case, the constructed hyper-plane to classify the data is purely classical. Only the kernel function is evaluated on the quantum computer.

3

Framework

With the background information necessary for understanding the context of this thesis, I will now elaborate how the research questions will be addressed. I will begin by explaining the dataset, and how it's used for sentiment analysis, and then dissect the transformer both classically and how it's migrated to a hybrid architecture. The overall architecture can be found in figure 1.3, which was first described by Vaswani *et al* in 2017. Due to the fact that most machine learning libraries are written in Python, along with previous work [27] [24], the natural choice of language to navigate this thesis was Python. To manage the loading and manipulation of the dataset, Pytorch, an open source Deep Learning framework for building and training neural network architectures, was used for tensor computations and the handling of gradient based training.

To accomplish the Natural Language Processing task of Sentiment Analysis, we are using the IMDB movie review dataset for this task. The dataset is composed of $50,000$ highly polar movie reviews, compiled from the Internet Movie Database (IMDB) along with their associated sentiment, either positive or negative. This dataset was compiled by Andrew Mass *et al* at Stanford [15]. Because the actual movie reviews are considerably long ($>$ 500 words), five different simplified text examples can be seen in the table 3.1. The written review is formatted as an input string in English text form. The labels of the dataset are binary 0, for a negative review, and 1 for a positive review, and are listed under the heading "*Sentiment*" in the table.

Table 3.1: A sample of 5 movie reviews from the IMDB dataset with corresponding sentiment. A "1" sentiment indicates a positive sentiment about the given text, and a "0" indicates a negative sentiment.

Review	**Sentiment**
"This movie was absolutely fantastic! The acting was superb, and the plot kept me on the edge of my seat throughout."	1
"I found the movie to be quite disappointing. The story was confusing, and the acting was subpar."	0
"A heartwarming film that left me with a smile on my face. The characters were well-developed, and the cinematography was stunning."	1
"I couldn't stand this movie. The dialogue was poorly written, and the performances were uninspiring."	0
"This is a must-see movie! The plot twists were mind-blowing, and the acting was top-notch."	1

We will be training our dataset on the movie reviews with known labels and testing on new reviews without labels to predict sentiment of the text. The output of the model can be deployed in a variety of areas as talked about in the background section.

3.1 Data Pre-Processing

As with all machine learning tasks, we need to prepare the input data such that is in a clean and readable format for the model. In addition to checking the datatset for null values, incomplete data, entry errors, etc, we use a variety of pre-processing steps which are specific for sentiment analysis string format data. Before we can pre-process the text, we first must convert the input text to tokens. **Tokenization** helps to split the text into smaller units, which allows for easier analysis and processing. Common tokenization methods include splitting the text on whitespace, punctuation marks, or a more advanced method with more complex language models involves splitting on word segments. Continuing with the sample data we can visualize how we split the review text into tokens.

Table 3.2: Tokenized input text

Review	Sentiment
"["This", "movie", "was", "absolutely", "fantastic", "!", "The", "acting", "was", "superb", ",", "and", "the", "plot", "kept", "me", "on", "the", "edge", "of", "my", "seat", "throughout", "."]."	1
["I", "found", "the", "movie", "to", "be", "quite", "disappointing", ".", "The", "story", "was", "confusing", ",", "and", "the", "acting", "was", "subpar", "."]	0
["A", "heartwarming", "film", "that", "left", "me", "with", "a", "smile", "on", "my", "face", ".", "The", "characters", "were", "well-developed", ",", "and", "the", "cinematography", "was", "stunning", "."]	1
"I couldn't stand this movie. The dialogue was poorly written, and the performances were uninspiring."["I", "couldn't", "stand", "this", "movie", ".", "The", "dialogue", "was", "poorly", "written", ",", "and", "the", "performances", "were", "uninspiring", "."]	0
["This", "is", "a", "must-see", "movie", "!", "The", "plot", "twists", "were", "mind-blowing", ",", "and", "the", "acting", "was", "top-notch", "."]	1

The first text pre-processing step is to remove numbers. As we do not expect numbers to pay a significant role in the overall sentiment of the text, we can simply remove them. As there are no numbers in the previous examples, a sample text which the function to remove number is applied can be seen as: "There are 3 balls in the bag, and 12 in the other one" which becomes: "There are balls in the bag, and in the other one". This simply removes the inputs in the string which are not part of English alphabet. After processing this for removing numbers, we need to rejoin the string such that there are no extra spaces.

Stopwords are commonly used words in a language that do not carry significant meaning or sentiment. Examples of stopwords in English include "the," "is," "and," "but," and "in." These words can occur frequently in a text document, but they may not contribute much to the sentiment expressed in the text. By removing stopwords, we can reduce noise and focus on the more meaningful words that carry sentiment information.

Table 3.3: Tokenized input text with stopwords removed

Review	Sentiment
["movie", "absolutely", "fantastic", "!", "acting", "superb", ",", "plot", "kept", "edge", "seat", "throughout", "."]	1
["found", "movie", "quite", "disappointing", ".", "story", "confusing", ",", "acting", "subpar", "."]	0
["heartwarming", "film", "left", "smile", "face", ".", "characters", "well-developed", ",", "cinematography", "stunning", "."]	1
["couldn't", "stand", "movie", ".", "dialogue", "poorly", "written", ",", "performances", "uninspiring", "."]	0
["must-see", "movie", "!", "plot", "twists", "mind-blowing", ",", "acting", "top-notch", "."]	1

As we look through the examples, it can be seen that the meaning and overall sentiment of the review remains.
Punctuation marks such as periods, question marks, and exclamation marks help define the structure of a sentence. They indicate the beginning and end of a sentence, and they help differentiate between declarative statements, interrogative sentences, and exclamatory expressions. Without proper punctuation, sentences can become ambiguous or difficult to understand. Although punctuation provides structure to our sentences and can help clarify relation between text, the primary reason to remove punctuation is to focus on the words themselves and concentrate on the lexical content and sentiment of the words. By removing punctuation, the text becomes a sequence of words without the interruptions or breaks introduced by punctuation marks. This streamlined text allows our model to focus on the sentiment of the words, their relationships, and their overall impact on the classification.

Table 3.4: Tokenized input text with punctuation removed

Review	Sentiment
["movie", "absolutely", "fantastic", "acting", "superb", "plot", "kept", "edge", "seat", "throughout"]	1
["found", "movie", "quite", "disappointing", "story", "confusing", "acting", "subpar"]	0
["heartwarming", "film", "left", "smile", "face", "characters", "well-developed", "cinematography", "stunning"]	1
["couldn't", "stand", "movie", "dialogue", "poorly", "written", "performances", "uninspiring"]	0
["must-see", "movie", "plot", "twists", "mind-blowing", "acting", "top-notch"]	1

Lemmatization is the process of reducing words to their dictionary form, or lemma. The Natural Language Tool Kit (NLTK) is a python library for Natural language processing tasks, including lemmatization. Lemmatization first employs another NLP tool, Part-of-Speech (POS) tagging, to determine the grammatical category of

each word in the text (noun, verb, adjective, etc). The lemmatization tool then can use this POS tag and returns the lemma of the word. The lemmatization is provided by WordNet [28], a lexical database in English which groups each grammatical category into sets of cognitive synonyms (synsets).

Table 3.5: Tokenized input text with lemmatization applied

Review	**Sentiment**
['movie', 'absolutely', 'fantastic', 'act', 'superb', 'plot', 'keep', 'edge', 'seat', 'throughout']	1
['find', 'movie', 'quite', 'disappoint', 'story', 'confuse', 'act', 'subpar']	0
['heartwarming', 'film', 'leave', 'smile', 'face', 'character', 'well-developed', 'cinematography', 'stun']	1
['could', "n't", 'stand', 'movie', 'dialogue', 'poorly', 'write', 'performances', 'uninspiring']	0
['must-see', 'movie', 'plot', 'twist', 'mind-blowing', 'act', 'top-notch']	1

For example, we see that the words "written" and "acting" were changed to "write" and "act", thus simplifying each word. Lemmatization ensures that different inflections of a word are mapped to a single form.
Another method to reduce words into a simpler form is called stemming. Stemming is a faster, but less accurate way to truncate the word into a simpler form without considering meaning or context. The truncated words might not always be a valid form of the word, but the processing time of finding the stemwords is significantly faster than performing lemmatization. To begin, the model used for this project used stemwords for efficiency, as the processing time for providing the lemmas for all 30k input reviews in the training set was $1.7s$, whereas the processing time for stemming was $0.5s$.

Table 3.6: Tokenized text with stemword processing.

Review	**Sentiment**
['movi', 'absolut', 'fantast', 'act', 'superb', 'plot', 'kept', 'edg', 'seat', 'throughout']	1
['found', 'movi', 'quit', 'disappoint', 'stori', 'confus', 'act', 'subpar']	0
['heartwarm', 'film', 'left', 'smile', 'face', 'charact', 'well-develop', 'cinematographi', 'stun']	1
['could', "n't", 'stand', 'movi', 'dialogu', 'poorli', 'written', 'perform', 'uninspir']	0
['must-se', 'movi', 'plot', 'twist', 'mind-blow', 'act', 'top-notch']	1

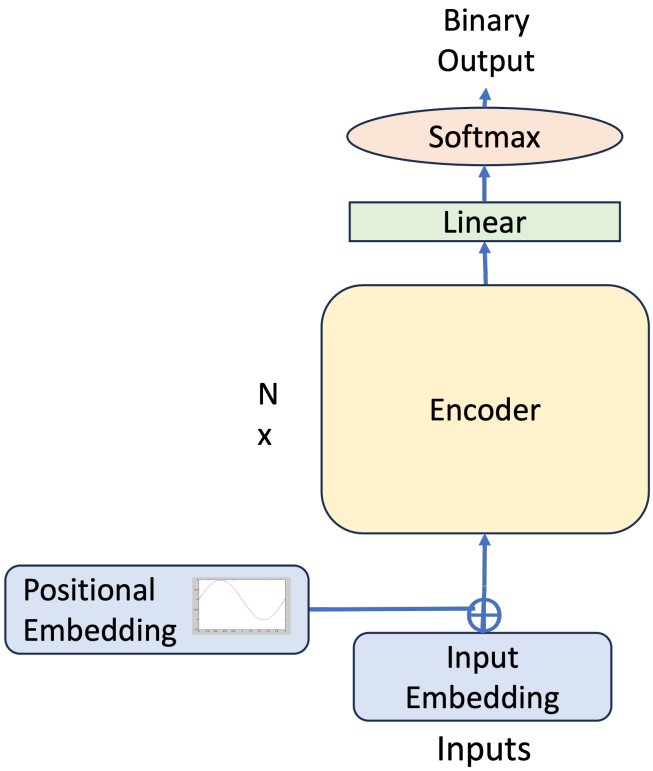


Figure 3.1: Architecture for the encoder only transformer

We can see a number of inaccuracies in the words produced after stemming. As we will see, these words will be mapped to a dictionary where the model will learn the meaning of these words and their relations to each other. So the relationship between stemwords should not be affected greatly by the stemming. However, if we provide further processing of the text by lemmatization, we can expect to see improvement in the model performance.
After the input tokens have been processed, they need to be stored in a dictionary, which we call the open vocabulary dictionary. This vocabulary contains all of the unique tokens that appear in the training set. The size of this dictionary is a hyperparameter to be explored, but for the purposes of this thesis, we will fix the size to be $20,000$ unique tokens.

3.2 Classic Transformer

Transformers have revolutionized the field of natural language processing and were introduced in 2017 in the paper "Attention is all you need" [29]. Most modern NLP models are based on a transformer architecture, and have been applied to a wide range of NLP tasks. A survey of such use cases was written by Yang *et al.* [33]. The architecture for the encoder only transformer, as first introduced in [29] is shown in figure 3.1, contrary to the encoder/decoder transformer shown in the introduction. Prior to the transformer, recurrent models and convolutional neural networks processed input sequences in sequential manner, or word by word. As sequence lengths

get longer, sequential models might lose some long-range dependency. In other words, if there is a word at the end of the sequence which relies on the input at the beginning of the sequence, the model might forget which word was at the beginning by the time it gets to the later word. Transformers on the other hand, have the possibility to process the entire sentence in parallel, making them much more efficient. The transformer is a sequence agnostic model, in which relations between tokens is handled by the Multi-Head Attention as will be described in section 3.2.3. This Multi Headed attention is packaged in what is called the Encoder and Decoder stacks. This architecture has been used in recurrent models, however the Attention is where the main differences lie. The encoder will process the input text, while the decoder will produce the output text. The following sections will break down the components of the transformer in more detail.

3.2.1 Positional Encoding and Word Embeddings

To capture the meaning of the words, the individual tokens are vectorized. The word vectors are then assigned to a random vector with the size of the embedding dimension. The embedding dimension is a hyper-parameter which can be optimized. During the training process, the embeddings are updated based on the objective of the model and patterns in the data. How one initially embeds the data can be random, by pretrained word embeddings such as Word2Vec [16], GloVe [20], or fastText [11], or by contextual word embeddings which not only capture the meaning of the individual words, but also their context within the sentence or document. For simplicity, our model will use randomly assigned embedding vectors. In sequential data, such as input text, the order of the words conveys important information. Transformers are order agnostic, as they process the entire sequence in parallel, so the concept of word order is lost. To combat this, Vaswani et al proposed a clever solution to incorporate sequential position information into the transformer. They proposed a set of learned sinusoidal functions to encode the position information. The positional encoding is then added to the word embeddings.

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{model}}\right) \tag{3.1}$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{model}}\right) \tag{3.2}$$

where pos is the position in the sequence, and i is the dimension. By using a large enough dimension (d_{model}), the positional encoding can distinguish between closely spaced positions effectively [29].

3.2.2 Encoder

The embedded text information is then passed on to the encoder stack. This stack is composed of two sub-layers which have a residual connection followed by a layer normalization. The two main sublayers are a Multi-Headed Attention layer, and a simple fully connected feed forward neural network. For NLG tasks, transformer architecture employs the use of both an encoder and decoder stack, however for sentiment analysis, we do not need to utilize the future position dependent output, so we will only use an encoder layer for the attention. To increase the complexity of

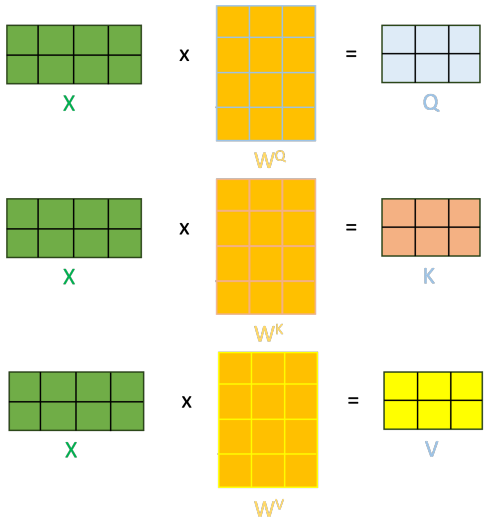


Figure 3.2: Matrix representation of the QKV calculation from the input embeddings.

the transformer, and to pick up new information after each pass, the encoder (and decoder layer if applicable), can be repeated N number of times.

3.2.3 Attention

At a high level, the attention can be understood as a similarity measure between elements of the input sequence, where the weights or similarities are determined by weighted combinations of the elements. The attention can be broken down even further to the query, key, and value matrices. The keys (K) is computed by a linear transformation of the word embeddings. The key captures the knowledge or information of each word and it's relevance to the other words in the sequence. The query (Q) vector is another linear transformation of the word embeddings. The query vector represents the word for which the attention scores are being calculated. The value matrix carries the actual information of each word and is obtained by another linear transformation of the word embeddings. These vectors are obtained by multiplying the input vector by a weight matrix W, with the corresponding subscript W_Q, W_K, W_V.
Vaswani et al described this approach as the scaled dot product attention, during which the query vector of a word is compared with the key vectors of all other words in the sequence to determine their similarity or relevance. The dot product of the two vectors captures the similarity between the two. For high dimensionality, the dot products can grow very large in magnitude, so the dot product is thus scaled by $\frac{1}{\sqrt{d_k}}$.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.3}$$

Multi-Headed Attention therefore is simply the self attention mechanism performed multiple times in parallel. This parallel attention allows the model to capture different kinds of relationships. Each attention head *attends* to different parts of the

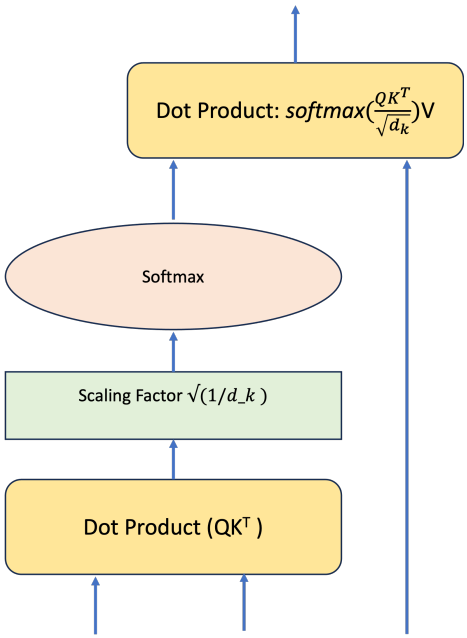


Figure 3.3: Attention block diagram

input sequence which allows for various dependencies to be captured at different levels of granularity. Each of the projected heads is then concatenated and a final output with dimension of d_v is projected on to the feed forward network.

3.2.4 Feed Forward Network

The classical feed forward network consists of two linear transformations with non-linear activation functions in between, typically Rectified Linear Unit (ReLU) activation. The inner layer has a hidden dimension of

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{3.4}$$

Where W_1 and W_2 are learned weights. The output of this layer has the same dimensionality of the model, which is then given to a final Linear transformation and softmax function to give the final output classification.

3.3 Hybrid Transformer

The hybrid transformer is a member of the class of algorithms which are deemed hybrid quantum algorithms. These algortihms leverage the classical optimizers ability to pre and post-process the data, perform the optimization over the model parameters, and defines the quantum circuit.

3.3.1 Simple Quantum Attention

Quantum attention refers to the quantum kernel embedding in the Hilbert Space. We take the tokens for each of the sequences, and perform quantum attention on

the relation to all other words in the sequence. This attention, provides the inner product to provide a similarity measure between tokens. This similarity measure, is then passed on to the feed forward network to provide the trainable weights for learning the representation. We can think of attention as how well each word 'attends' to every other word in the sequence.
For this thesis, the implementation of the attention is a scaled down version, in which we use the kernel function to measure the pair-wise similarity between tokens in an input text. As mentioned in the background, I will begin by using the kernel that is known as the ZZ-Feature map. The choice of kernel is another parameter that can be tuned throughout the project.
Because the operation is computationally expensive, ($O(L^2)$, where L is sequence length), and the current pennylane framework is not compatabile with batching over inputs, the initial architecture for computing the kernel attention is limited to a single evaluation of the kernel matrix for each input sequence.

4
Evaluation

With the algorithmic framework set, we now have the ability to answer the original research question. The evaluation of the hybrid transformer compared to the classical transformer will be compared using the IMDB dataset for training. To further understand the power of the transformer framework and the power of attention, the transformer will also be analyzed with the attention layer deactivated.

4.1 IMDB dataset

A full evaluation of the dataset can be seen in Appendix B. To not introduce a bias in the dataset for length of review and how that influences the sentiment, all of the reviews were truncated or padded to a specified length. In addition, the maximum number of tokens that our model has a limit of 512 tokens. Simple exploratory analysis of the dataset was done to visualize any predefined patterns in the dataset when it comes to sentiment. The average length review across the dataset was around 200 words as can be seen in figure 4.1.

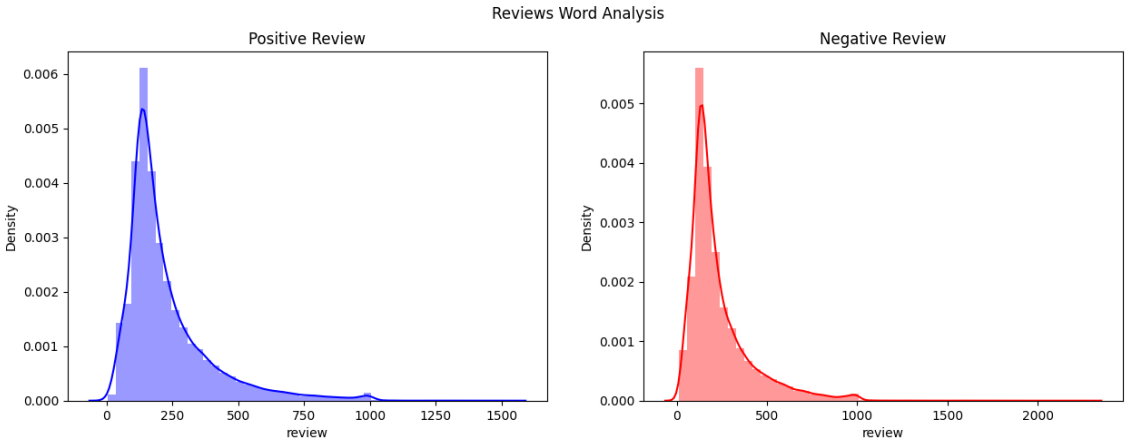


Figure 4.1: Review length corresponding to sentiment. No significant indication exists whether positive or negative reviews affect the sentiment.

4.2 Evaluation Metrics

When evaluating a machine learning model, the common evaluation metrics are accuracy and loss. In short, the accuracy measure for a classification problem is a

measure of correct predictions over the total number of predictions. This can be formalized by the equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

where TP and TN are true positives and true negatives respectively, and FP and FN are false positives and false negatives respectively. For sentiment analysis, the true positives would be cases where the model correctly predicted the positive sentiment of the text as indicated by the label, and true negative would be the case where the model correctly predicted the negative sentiment of the text. False positive and false negatives would then be incorrect predictions of the text. Before the accuracy can be measured, the model must compute the loss function as seen in equation 4.2. Combined with an activation function to transform a raw probabilistic prediction to a binary classification, the Binary Cross Entropy (BCE) with logits (raw, un-normalized predictions) loss is a PyTorch built in loss function for deep learning models. Where the binary cross entropy calculates the loss by measuring the difference between the predicted probabilities and the true labels.

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log \sigma\left(x_n\right) + \left(1 - y_n\right) \cdot \log\left(1 - \sigma\left(x_n\right)\right)\right] \tag{4.2}$$

The classical optimizer Adam is used to to update the weights of the transformer. After each iteration, the optimizer computes the gradients with respect to the loss function. The parameters are updated according to the learning rate using a combination of the first and second moment of the gradients. This can be thought of as an adaptive learning rate, where parameters with a large gradients will not change the learning rate as much and those with small gradients will increase the learning rate. Due to ease of use, no other optimizer or loss function will be explored for this thesis. During the training, to avoid overfitting the data, we use a dropout after each sublayer of the model. The idea behind dropout is to introduce some noise or randomness into the deep network by temporariliy removing a subset of neurons during the training. This prevents the network from relying too heavily on specific neurons. [14]

4.3 Benchmarking

The power of transformers arises from this concept of attention. However, a traditional multi-layer perceptron is still a powerful tool for classification. Assessing the performance of the attention mechanism in a transformer model is crucial for understanding its impact on the model's overall performance. This evaluation can be conducted by comparing the model's performance with the attention mechanism deactivated and then activating it. When the attention mechanism is deactivated, the transformer model loses its ability to capture relationships between different tokens or elements in the input sequence. In this scenario, the model relies solely on the feed-forward layers, which process each token independently. The hyperparameters used for this assessment can be seen in table 4.1. The model was trained over 20 epochs, with a learning rate of 0.05. The results of the training and testing evaluation can be seen in following figure 4.2.

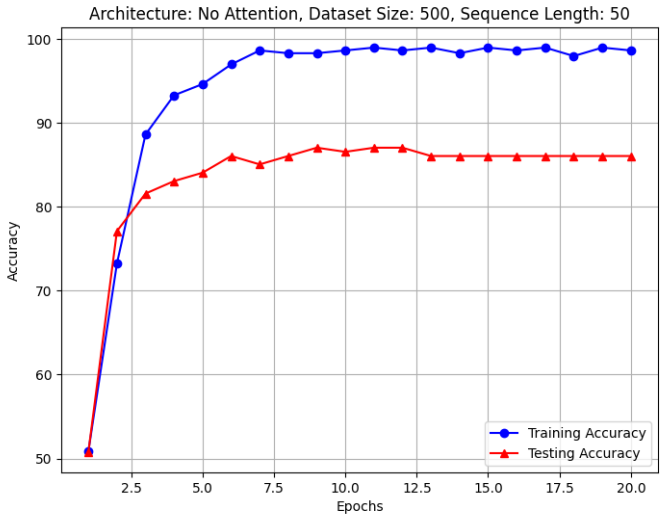


Figure 4.2: Training of the transformer model with the attention deactivated

Table 4.1: Hyperparameters for the attention-less transformer

Parameter	**Value**
Size of dataset	500
Batch size	16
Sequence Length	50
Embedding Dimension	8
Feed Forward Dimension	64
Number Epochs	20
Number of Transformer blocks	0
Number of heads	1
Vocab size	20,000
Dropout	0.1

At the beginning of the training, the model's performance shows a significant improvement as it learns from the data. This is evident by observing an upward trend in the accuracy metric in the graph. As the training continues, the performance gains gradually diminish, and the graph starts to level out. This indicates that the model is reaching a point of saturation where additional training epochs may not yield significant improvements. The leveling out of the graph suggests that the model has learned most of the patterns and information from the training data. It implies that the model is now adequately trained and can generalize well to unseen data. By observing this graph, one can conclude that training the model for more than 20 epochs may not provide substantial benefits in terms of performance improvement. However, it can be concluded that a simple multi-layer perceptron does provide a good framework for learning based on the performance.

4.3.1 Previous work

We will now look at the performance of a classical transformer. The classical transformer has been used for sentiment analysis and there are many implementations of this [32]. Because I am not competing to be the best classical transformer, a simple implementation would suffice for the purposes of this thesis as a benchmark. Also because the classical transformer is not the main focus of the paper, an exhaustive hyperparameter tuning was not performed. The hyperparameters used as comparison to the Di Sipio paper [27] can be seen in table 4.2.

Table 4.2: Hyperparameters for the previous work on a quantum transformer

Parameter	**Value**
Size of dataset	50000
Batch size	32
Sequence Length	64
Embedding Dimension	8
Feed Forward Dimension	8
Number Epochs	1
Number of Transformer blocks	1
Number of heads	2
Vocab size	20,000
Dropout	0.1

When running the hybrid transformer using the hyperparameters outlined in the paper [27], the runtime for such a model was too long to train on the current hardware and my implementation. Therefore, it was deemed necessary to modify the hyperparameters such that a feasible runtime could be achieved to obtain results. From this point, I started to modify the classical transformer and build upon this architecture.

4.3.2 Classical transformer

The classic transformer with an encoder-only architecture was run using the full 50k review dataset. Where 30k reviews were used for the training set and 20k reviews were used for the test set. The results for the best performing classic transformer can be seen in the figure 4.3 with the hyperparameters in 4.3.

Table 4.3: Hyperparameters for the classical transformer

Parameter	**Value**
Size of dataset	50000
Batch size	32
Sequence Length	512
Embedding Dimension	64
Feed Forward Dimension	256
Number Epochs	10
Number of Transformer blocks	4
Number of heads	8
Vocab size	20,000
Dropout	0.1

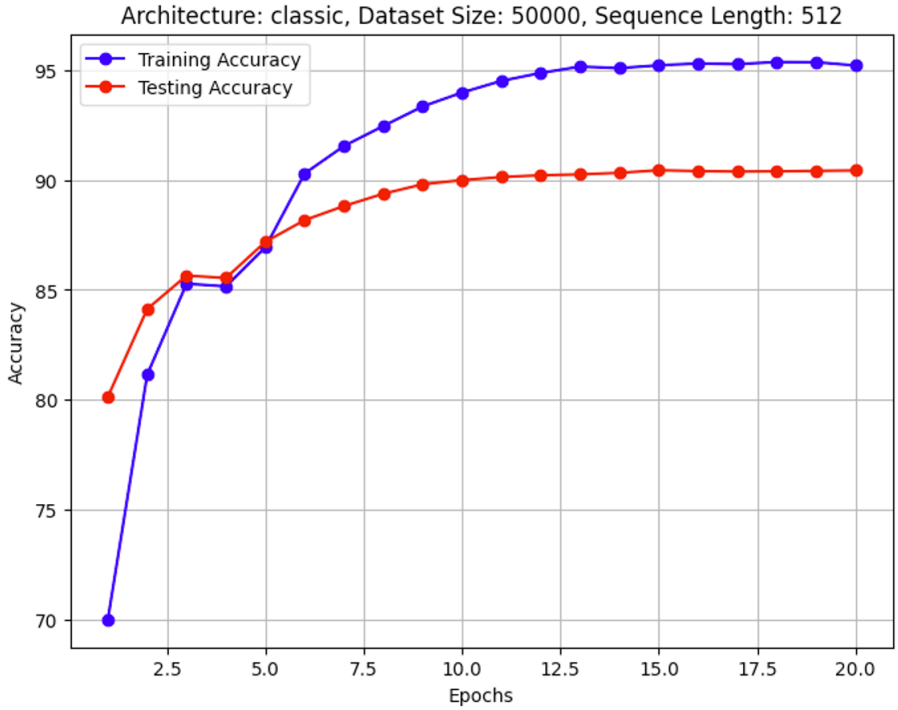


Figure 4.3: Full dataset used, sequence length of 512 tokens, and ran for 20 epochs. The runtime was $\approx$ 43 minutes.

Table 4.4: Accuracy and Loss results for the Classical Transformer

Epoch	**Training Loss**	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Epoch Runtime (s)**
0	0.560	70.02	80.14	132.4
1	0.414	81.15	84.13	117.7
2	0.345	85.28	85.65	118.5
3	0.347	85.16	85.54	117.1
4	0.311	86.93	87.18	118.4
5	0.245	90.29	88.17	119.2
6	0.222	91.54	88.81	116.2
7	0.203	92.45	89.37	116.5
8	0.184	93.34	89.80	117.9
9	0.171	93.97	89.99	117.1
10	0.157	94.50	90.13	117.1
18	0.142	95.35	90.41	118.1
19	0.144	95.20	90.43	118.5
Total Runtime: 43m, 3s				

It is clear from the test and training data that this model performs well. However, comparing to published classical transformers analyzed on the same dataset, it falls short by about 2% from making the leaderboard [32]. After about 10 epochs, the training seems to slow down as it reaches a plateau at around 90% on testing accuracy. We can see a dip in the accuracy after the 2nd and 3rd epoch, but the model quickly improves after.

4.3.3 Variational Quantum Circuit

The variational circuit was the best performing architecture for the quantum attention due to the availble tuning of the hyperparameters. This increase in performance is likely linked to the bigger size of the dataset, i.e. being trained on more data. With the first 5000 reviews used, and a sequence length of 50 tokens, the runtime for 5 epochs with this model was 3841.8 minutes. Compared to the classical transformer, which ran on a dataset that was $10x$ the size, and 20 epochs were completed in 43 minutes. The resulting data can be seen in table 4.6 and figure 4.6.

Table 4.5: Hyperparameters for the variational quantum attention transformer

Parameter	**Value**
Size of dataset	5000
Batch size	32
Sequence Length	50
Embedding Dimension	8
Feed Forward Dimension	8
Number Epochs	5
Number of Transformer blocks	1
Number of heads	1
Vocab size	20,000
Dropout	0.1

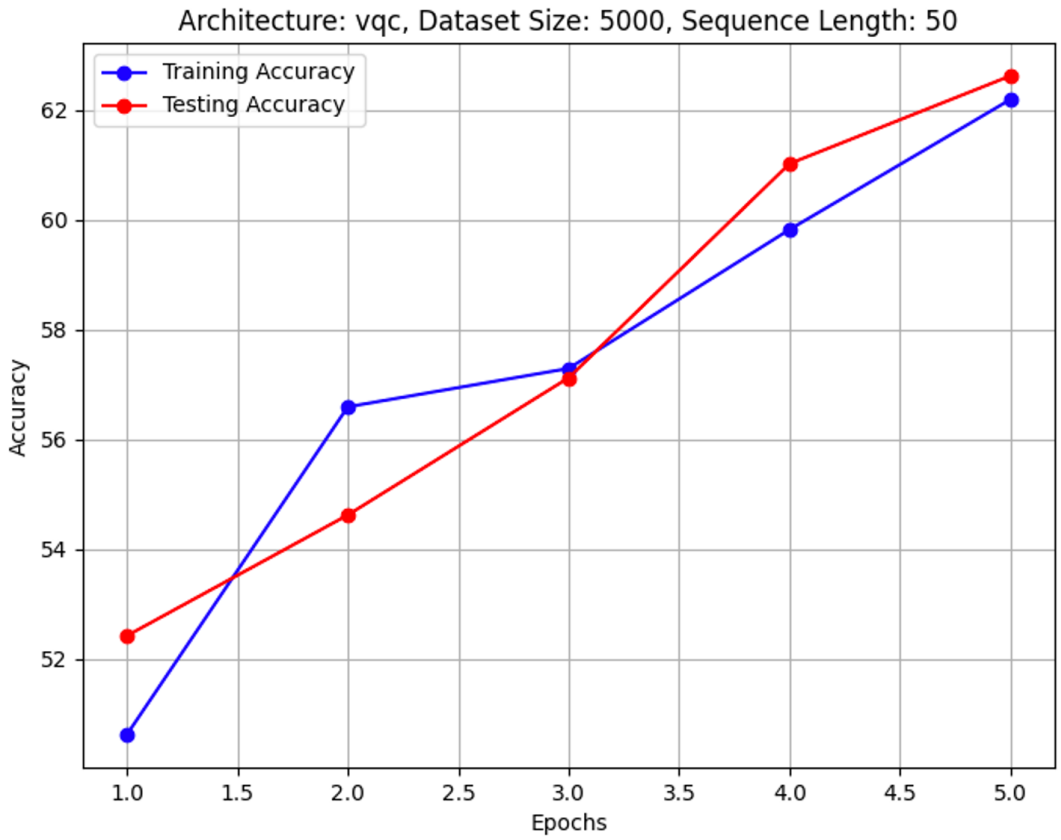


Figure 4.4: First 5000 reviews used, sequence length of 50 tokens, and ran for 5 epochs. The run-time was 3841.8 minutes

Table 4.6: Accuracy and Loss results for the Variational Quantum Circuit Attention Transformer

Epoch	**Training Loss**	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Epoch Runtime (s)**
1	0.693	50.62	52.42	46506.4
2	0.685	56.59	54.62	46300.4
3	0.681	57.29	57.12	45958.0
4	0.673	59.82	61.02	45852.4
5	0.661	62.19	62.62	45869.0
Total Runtime: 3841m, 47s				

The increase in runtime was not feasible to scale this model any higer to directly compare the best running classical transformer or attention-less transformer. The runtime comparison between the three models will be discussed in section 4.5.

4.4 Quantum Kernel Attention

The kernel attention mechanism is currently running without training weights, but just as a single pairwise similarity measure. Even with this scaled down version of the attention mechanism, after 5 epochs, the training had taken 107.8 hours. This scale of runtime is not feasible for even a small model, and parallelization will need to be done to optimize runtime. However, it is interesting to note that while it took a long time, the model was able to produce accuracy results that were better than the variational circuit for this scale. The comparison can be seen in table 4.10.

Table 4.7: Hyperparameters for the Quantum Kernel Attention Transformer

Epoch	**Value**
Size of dataset	1000
Batch size	16
Sequence Length	50
Embedding Dimension	8
Feed Forward Dimension	256
Number Epochs	5
Number of Transformer blocks	1
Number of heads	1
Vocab size	20,000
Dropout	0.1

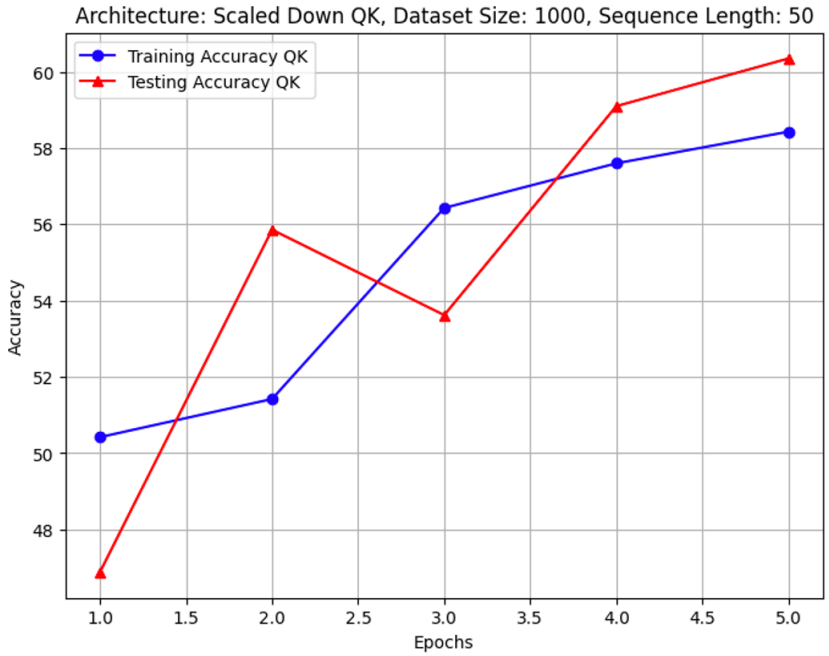


Figure 4.5: The first 1000 reviews were used. A sequence length of 50 tokens, which was ran for 5 epochs. The runtime of this was 6469 minutes.

Table 4.8: Accuracy and Loss results for the Quantum Kernel Attention Transformer

Epoch	**Training Loss**	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Epoch Runtime (s)**
1	0.699	50.42	46.88	79449.3
2	0.692	51.42	55.86	77806.0
3	0.687	56.43	53.62	77725.3
4	0.686	57.60	59.10	75905.4
5	0.685	58.43	60.35	77305.2
Total Runtime: 6469m, 50s				

4.5 Comparison

This section will explore how the three models compare against each other. While the optimal performances had used a variety of different parameters, this section will explore how each of the models perform across an equally complex architecture using the same hyperparameters, numbeer of reviews, sequence length, number of epochs ran, etc. Table 4.9 shows the parameters used for comparison.

Table 4.9: Parameters used to compare the three model architectures

Epoch	Value
Size of dataset	1000
Batch size	32
Sequence Length	50
Embedding Dimension	8
Feed Forward Dimension	8
Number Epochs	5
Number of Transformer blocks	1
Number of heads	1
Vocab size	20,000
Dropout	0.1

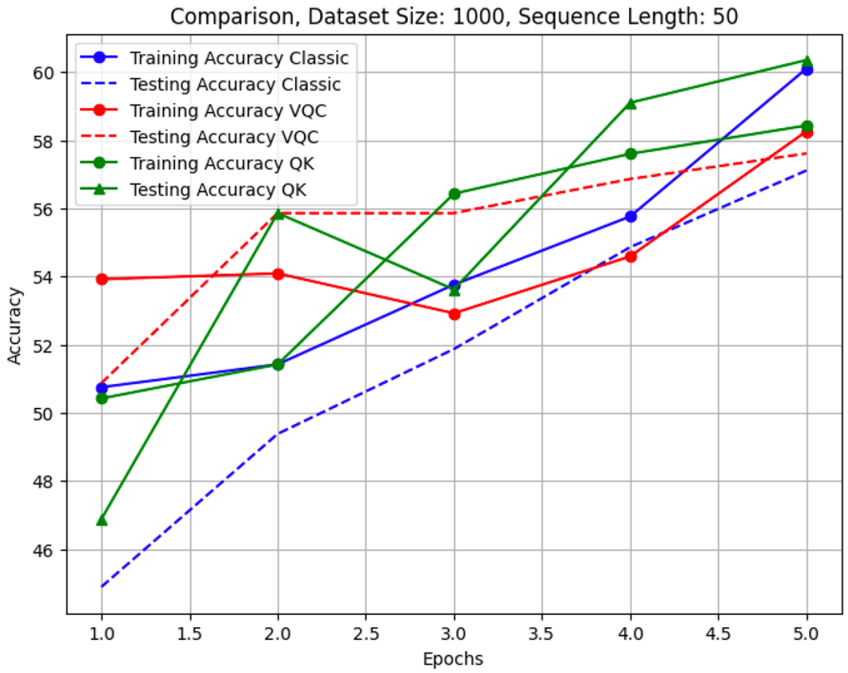


Figure 4.6: First 1000 reviews used, sequence length of 50 tokens, and ran for 5 epochs. A comparison across all three models tested. The accuracies and runtimes of each model can be seen in the table 4.10
. It is important to note that the models were scaled to use the same hyperparameters during this comparison test.

Looking at the comparison testing, the quantum kernel performed the best among the three models. However, it is worthwhile to note that we saw a similar dip in accuracy at around 5 epochs for the classical transformer when trained on the full size dataset and a review length $5x$ longer. It can only be assumed that if the classical transformer were to run for a longer time, that the accuracy would increase considering the performance seen in table 4.4. It is important to look at the runtimes for each of the models as well. Even across this small subset of data, the runtime

Table 4.10: Comparison of different architectures based on accuracy and runtime after 5 epochs.

Architecture	**Train Accuracy (%)**	**Test Accuracy (%)**	**Runtime (m, s)**
Classic	60.10	57.11	0m 29s
VQC	58.26	57.61	778m 47s
Quantum Kernel	58.43	60.35	6469m 50s

for the classical transformer was 29 seconds including the preprocessing of the data which adds roughly 18s in runtime while the runtime for the two quantum models was 778 minutes and 6469 minutes for the variational quantum circuit and the quantum kernel attention, respectively. While the quantum kernel performed the best according to the accuracy score, the computational cost was extremely high. The most efficient model is evidently the classical model, which also has the highest training accuracy.
Additional figures for the other configurations tested can be found in appendix B. In these runs, the size of the dataset used was varied from 500 reviews, 1000 reviews, 5000 reviews and 50000 reviews for the classic transformer as well as varying the sequence length from 30, 50 to 512. These were ran for 5 and 20 epochs. While not much difference is seen from 500 to 1000 reviews, when using a minimum sequence length of 30 and running for 5 epochs, the 50000 review model significantly out performed the others, suggesting that the size of the dataset has a considerable impact on the performance of the model. We can also see this trend when looking at the two hybrid models as well with performance increases from 500 to 1000 reviews and the same sequence length and number of epochs ran.
A less significant increase in performance is seen when looking at the effect of the sequence length on performance. But the impact on the runtime naturally increases with increased complexity of the model.
When investigating the results even further, we can see how the performance on the training set and testing set differs based on the size of the data used. Compared to the optimal classical model and optimal variational quantum circuit transformers, the quantum kernel performs poorly, comparing 60% accuracy compared to around 90% for the classical transformer. Even with the attention component deactivated, that model significantly out-performed the quantum kernel model and the vqc relative to runtime, and training and accuracy scores. However, since the run-time for the hybrid transformer is much longer than that of a classical transformer, we needed to truncate the dataset and the length of the review. When we look at the sample texts from the testing set, it can be seen that the words that appear in the testing set, have not appeared in the training set, such that the model will have not been trained on the sentiment of that word so probability of getting that particular word correct, and how it relates to other words in the review would be simply a guess.

5
Conclusion

This chapter will summarize the results and what was learned throughout the course of the thesis. I will also address the limitations and future directions of this work. As with many projects, the scope and scale of the project had to be adapted to accommodate complications that arose during programming.

5.1 Summary

In summary, I focused on comparing three different transformer architectures and analyzed their performance using the IMDB sentiment analysis dataset. I began this thesis with a background on Natural Language Processing, and the different tasks that it can accomplish. Then to understand how the implementation of these NLP tasks is accomplished, I inserted a brief background on machine learning and quantum computing, including the qubit and circuit model, and how quantum gates can be used to construct quantum algorithms. We walked through the framework of the model and how the evaluation is to take place. After looking at the results, we can now address the research questions.

Research Question 1 Can we implement a quantum kernel as the attention mechanism for Transformer models?
Yes, the attention is just a similarity measure between tokens in a high dimensional feature space. Because the kernel is not weighted at this point, the attention is just the similarity measure and that similarity value is then passed on to the feed forward network to then learn based on those similarities. I would say that this is also a future work, to add weights to the kernel values such that the circuit is parameterized and those parameters can be attended to throughout training.
Research Question 2 Can the quantum kernel provide a computational speedup compared to using a classic self attention of the transformer model?
Definitely not at this point, at least not with this implementation.
Research Question 3 Does the training of a large language model using a quantum attention result in higher accuracy compared to a classic attention model? At this point, the classical circuit is the best performing model
Research Question 4 How does this implementation of a hybrid transformer with quantum kernel attention compare to previous work done using a variational quantum circuit as the attention? Compared to previous work, the performance is similar with regards to accuracy. However, the runtime of the two differ by about a factor of 5 for even the simplest of models.

5.2 Limitations

The question regarding why kernels are not used more frequently in today's architectures can be answered for the same reason why the performance of the architecture did not perform as well as the other transformer architectures. The pairwise computation of the kernel limits how scalable the model is. With the current support from the Quantum Machine Learning backend Pennylane, there is not a good way to parallelize the computation of the pairwise attention. As such, the computation is very slow and limits the usability of the framework due to time constraints. As I was writing this thesis, TorchQuantum library was introduced out of an MIT lab, which enables batching support for quantum circuits. However, the implementation would have required to learn a new library and the time did not permit this. Another limitation of this work, which I think was also a valuable learning opportunity for me was how to efficiently write scalable code for deep tensor networks. Due to my inexperience with using deep learning frameworks, I think that I had a steep learning curve to become comfortable with writing efficient code. I think ultimately, there might have been a better way to implement the kernel for the quantum attention which would have scaled better with the model, but for the purposes of the thesis, I will leave that for future work.

5.3 Future work

Because there was not an exhaustive hyper parameter search, I also think that this is a future direction. I began exploring the effects of the learning rate and how that effects the training. Also would like to explore to add a trainable parameterized layer in the quantum circuit for the kernel. To be able to train a quantum kernel, would create a frame work more similar to the classical attention that works so well. Without a trainable parameter, the attention in the quantum feaure space is just a scalar value that does not change throughout the course of the training.

Bibliography

[1] Francesco Bernardini, Abhijit Chakraborty, and Carlos Ordóñez. Quantum computing with trapped ions: a beginner's guide. *arXiv preprint arXiv:2303.16358*, 2023.

[2] Avi Chawla, Nidhi Mulay, Vikas Bishnoi, Gaurav Dhama, and Anil Kumar Singh. A comparative study of transformers on word sense disambiguation. *CoRR*, abs/2111.15417, 2021.

[3] El Amine Cherrat, Iordanis Kerenidis, Natansh Mathur, Jonas Landman, Martin Strahm, and Yun Yvonna Li. Quantum vision transformers. *arXiv preprint arXiv:2209.08167*, 2022.

[4] Andrew J. Daley, Immanuel Bloch, Christian Kokail, et al. Practical quantum advantage in quantum simulation. *Nature*, 607(7919):667–676, 2022.

[5] David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9–11):771–783, September 2000.

[6] Giulia Ferrini, Anton Frisk-Kockum, Pontus Vikstål, and Ariadna Soro. Quantum computing lecture notes, 2023.

[7] Fulvio Flamini, Nicolò Spagnolo, and Fabio Sciarrino. Photonic quantum information processing: a review. *Reports on Progress in Physics*, 82(1):016001, November 2018.

[8] Vojtvech Havlicek, Antonio D. Corcoles, 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, March 2019.

[9] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3), June 2008.

[10] Thomas Hubregtsen, David Wierichs, Elies Gil-Fuster, Peter-Jan H. S. Derks, Paul K. Faehrmann, and Johannes Jakob Meyer. Training quantum embedding kernels on near-term quantum computers. *Physical Review A*, 106(4):042431, October 2022.

[11] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[12] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2):021318, 2019.

[13] Guangxi Li, Xuanqiang Zhao, and Xin Wang. Quantum self-attention neural networks for text classification. *Science China Information Sciences*, 67(4):142501, 2024.

[14] Zhuang Liu, Zhiqiu Xu, Joseph Jin, Zhiqiang Shen, and Trevor Darrell. Dropout reduces underfitting. In *International Conference on Machine Learning*, pages 22233–22248. PMLR, 2023.

[15] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

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

[17] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

[18] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[19] OpenAI. Chatgpt release notes. `https://help.openai.com/en/articles/6825453-chatgpt-release-notes#h_e18c6a1b3a`, 2023.

[20] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[21] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, Aug 2018.

[22] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, September 2014.

[23] Bernhard Schölkopf and Alexander J. Smola. A short introduction to learning with kernels. In Shahar Mendelson and Alexander J. Smola, editors, *Advanced Lectures on Machine Learning*, volume 2600 of *Lecture Notes in Computer Science*, pages 41–64. Springer, Berlin, Heidelberg, 2003.

[24] Maria Schuld. Kernel-based training of quantum models with scikit-learn. `https://pennylane.ai/qml/demos/tutorial_kernel_based_training`, 02 2021.

[25] Maria Schuld. Supervised quantum machine learning models are kernel methods. *arXiv preprint arXiv:2101.11020*, 2021.

[26] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Quantum Science and Technology. Springer, 2nd edition, 2021.

[27] Riccardo Di Sipio, Jia-Hong Huang, Samuel Yen-Chi Chen, Stefano Mangini, and Marcel Worring. The dawn of quantum natural language processing. *CoRR*, abs/2110.06510, 2021.

[28] Princeton University. About wordnet. `https://wordnet.princeton.edu/`, 2010.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[30] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. Expanding data encoding patterns for quantum algorithms. In *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pages 95–101, 2021.

[31] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. Data encoding patterns for quantum computing. *ACM Computing Surveys*, 55(9):1–37, 2022.

[32] Papers with Code. State-of-the-art sentiment analysis on imdb. `https://paperswithcode.com/sota/sentiment-analysis-on-imdb`, 2023.

[33] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–32, 2024.

A

Appendix 1

A copy of the source code for this project can be found at:

https://gitlab.com/WConcepcion/masterthesis

B

Appendix 2

B.1 Classic Transformer

Upon inspection of the accuracy and loss results from the different hyperparameters chosen for the classic transformer, we can see that an increased dataset size from 500 to 1000 did not have a significant impact on the performance. However, when using the full 50000 reviews, the runtime increased by around $4x$, but the accuracy measurements also were much higher even at a lower sequence lengths and running only for 5 epochs. When we use the full sequence length and ran for 20 epochs, the model had a larger vocabulary to train on and was able to find even more patterns in the data, which we can see in the $> 90\%$ accuracy scores on the training set.

B.1.1 Accuracy

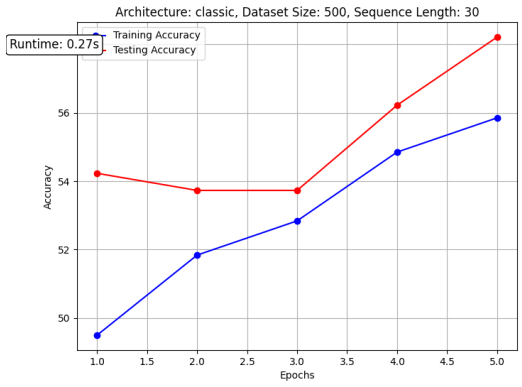


Figure B.1: Dataset size: 500, Sequence Length: 30, Num Epochs: 5

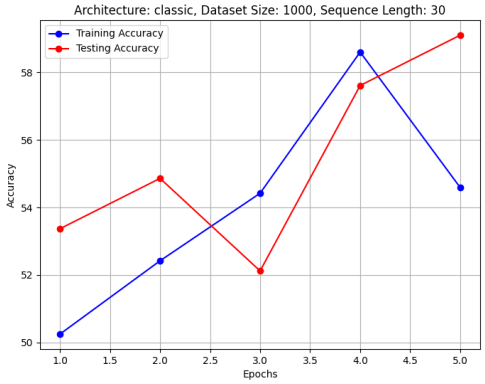


Figure B.2: Dataset size: 1000, Sequence Length: 30, Num Epochs: 5

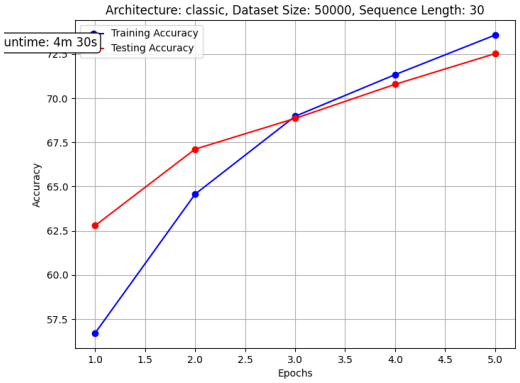


Figure B.3: Dataset size: 50000, Sequence Length: 30, Num Epochs: 5

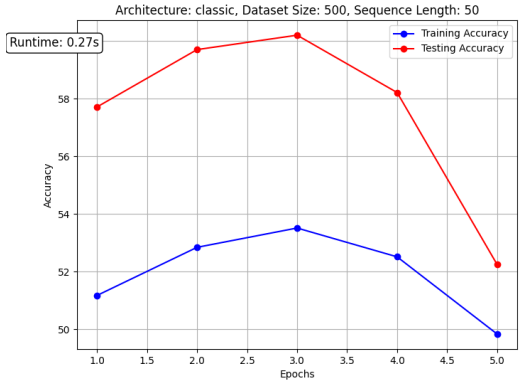


Figure B.4: Dataset size: 500, Sequence Length: 50, Num Epochs: 5

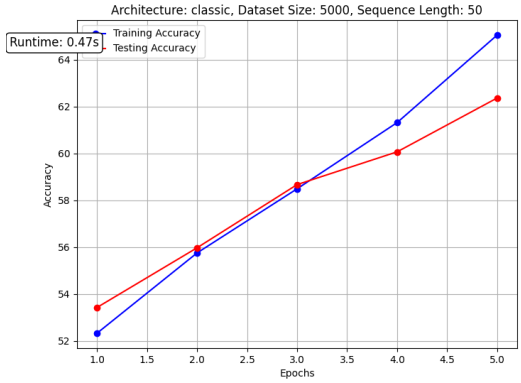


Figure B.5: Dataset size: 5000, Sequence Length: 50, Num Epochs: 5

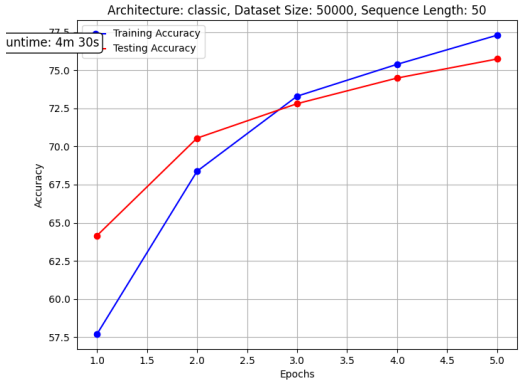


Figure B.6: Dataset size: 50000, Sequence Length: 50, Num Epochs: 5

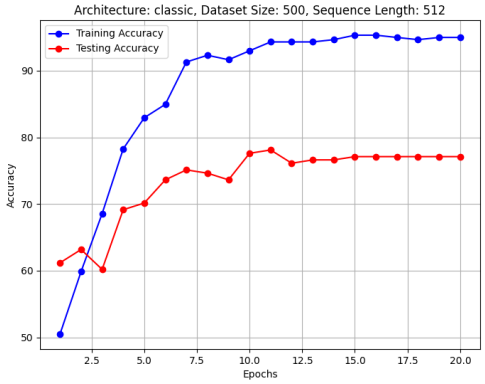


Figure B.7: Dataset size: 500, Sequence Length: 512, Num Epochs: 20

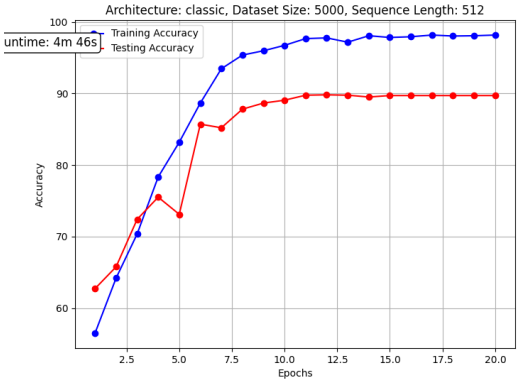


Figure B.8: Dataset size: 5000, Sequence Length: 512, Num Epochs: 20

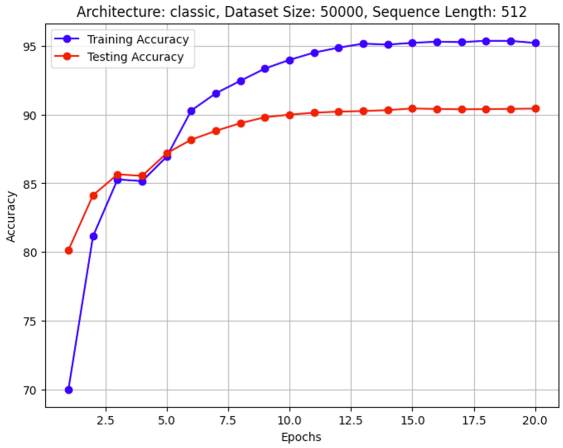


Figure B.9: Dataset size: 50000, Sequence Length: 512, Num Epochs: 20

B.1.2 Loss

[H]

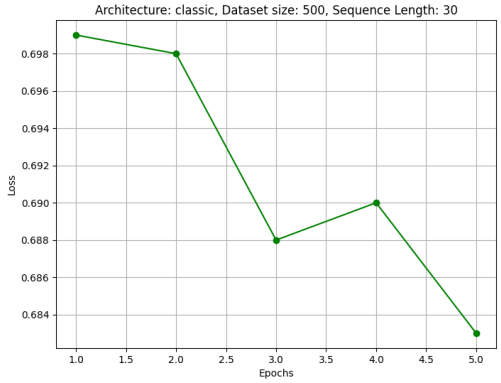


Figure B.10: Dataset size: 500, Sequence Length: 30, Num Epochs: 5

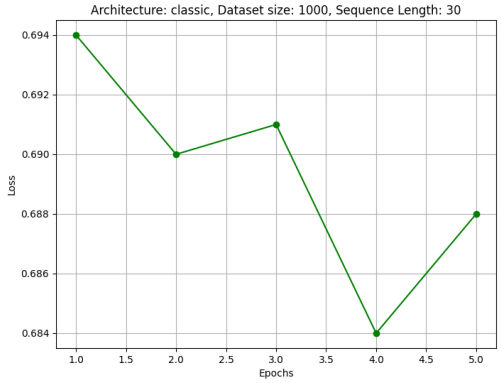


Figure B.11: Dataset size: 1000, Sequence Length: 30, Num Epochs: 5

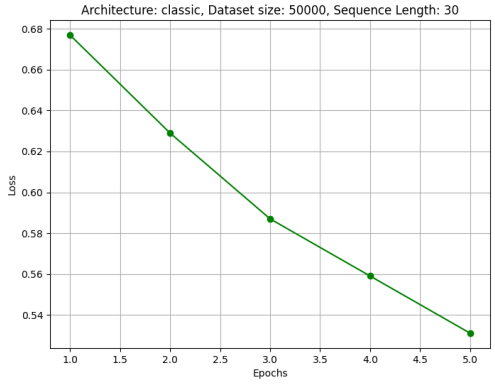


Figure B.12: Dataset size: 50000, Sequence Length: 30, Num Epochs: 5

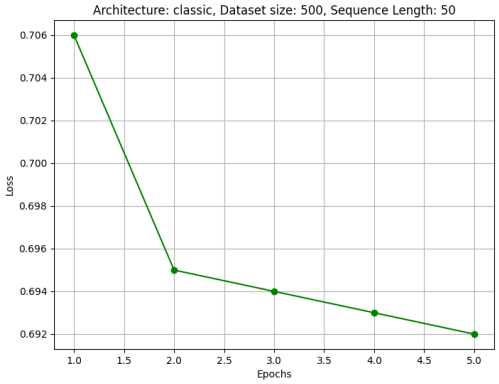


Figure B.13: Dataset size: 500, Sequence Length: 50, Num Epochs: 5

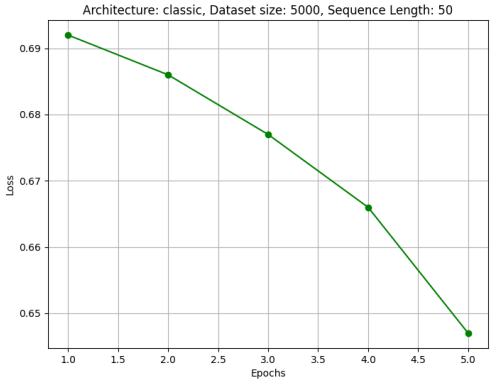


Figure B.14: Dataset size: 5000, Sequence Length: 50, Num Epochs: 5

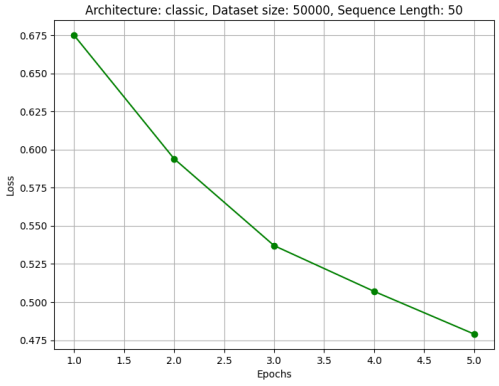


Figure B.15: Dataset size: 50000, Sequence Length: 50, Num Epochs: 5

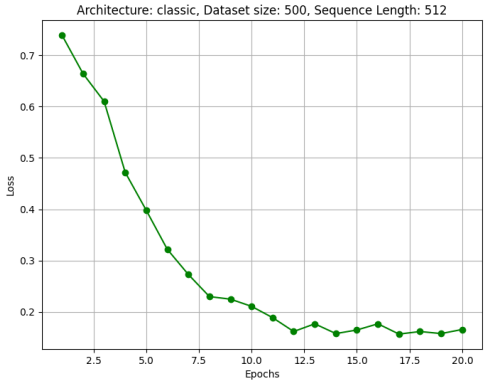


Figure B.16: Dataset size: 500, Sequence Length: 512, Num Epochs: 20

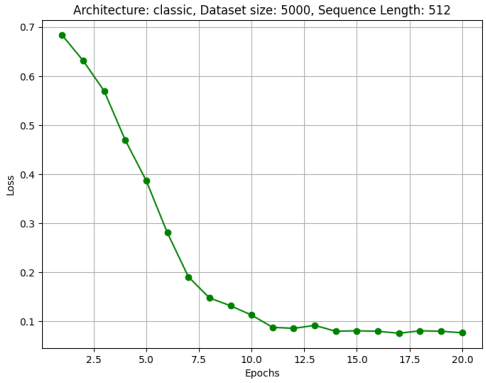


Figure B.17: Dataset size: 5000, Sequence Length: 512, Num Epochs: 20

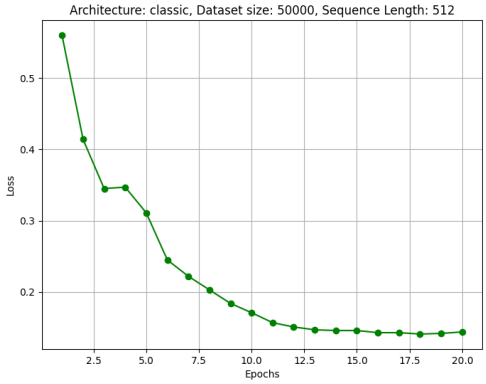


Figure B.18: Dataset size: 50000, Sequence Length: 512, Num Epochs: 20

B.2 Variational Quantum Circuit

Similarly to the classical transformer, when increasing from 500 to 1000 reviews in the dataset, at a low sequence length of 30, the model performs quite poorly, with minimal increase in the test set for the 1000 reviews run. However, we do see an increase in accuracy when increasing the sequence length to 50 compared to 30. When we increase the dataset size to 5000, that is when we saw the biggest increase in performance for the VQC, however the runtime for this was prohibitively long at 3841.8 minutes.

B.2.1 Accuracy

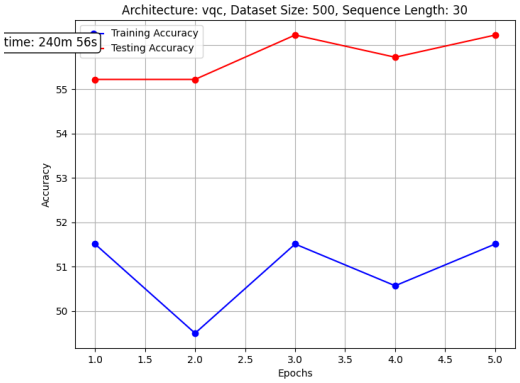


Figure B.19: Dataset size: 500, Sequence Length: 30, Num Epochs: 5

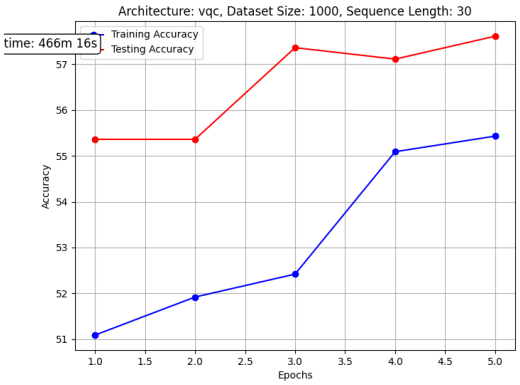


Figure B.20: Dataset size: 1000, Sequence Length: 30, Num Epochs: 5

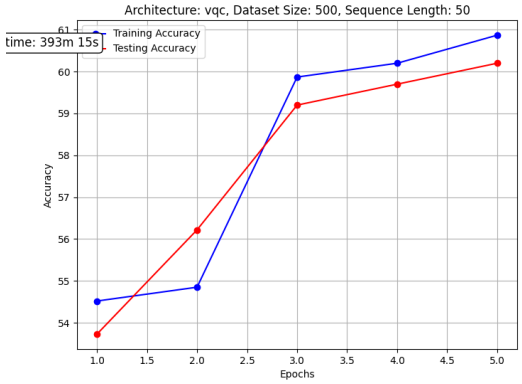


Figure B.21: Dataset size: 500, Sequence Length: 50, Num Epochs: 5

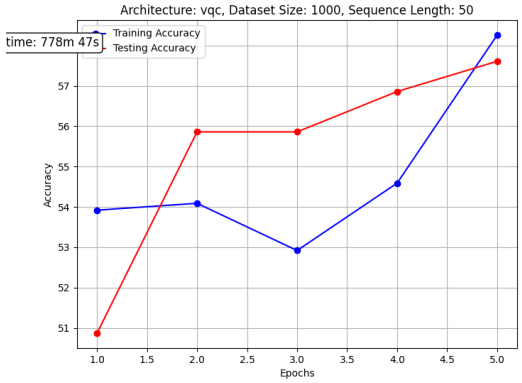


Figure B.22: Dataset size: 1000, Sequence Length: 50, Num Epochs: 5

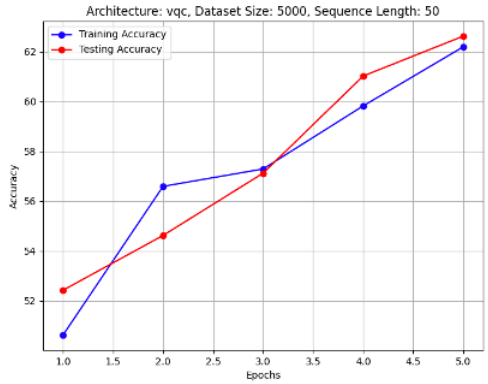


Figure B.23: Dataset size: 5000, Sequence Length: 50, Num Epochs: 5

B.2.2 Loss

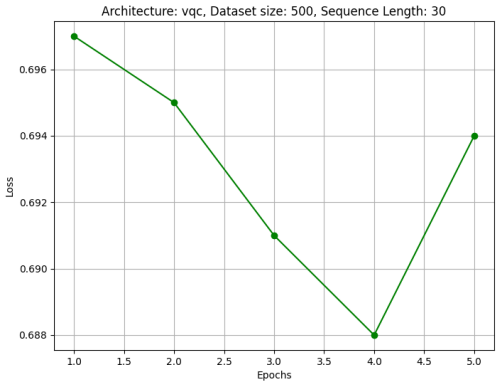


Figure B.24: Dataset size: 500, Sequence Length: 30, Num Epochs: 5

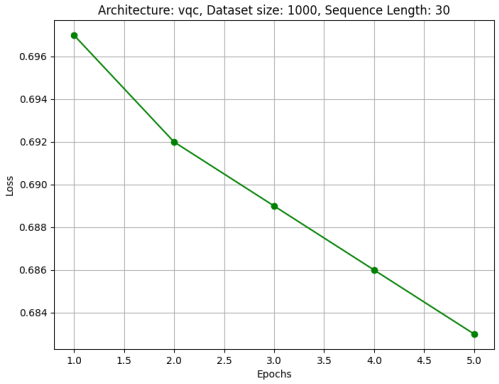


Figure B.25: Dataset size: 1000, Sequence Length: 30, Num Epochs: 5

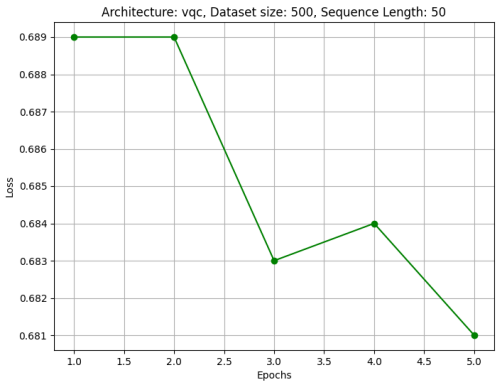


Figure B.26: Dataset size: 500, Sequence Length: 50, Num Epochs: 5

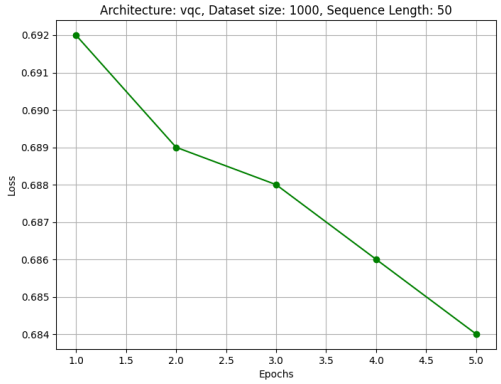


Figure B.27: Dataset size: 1000, Sequence Length: 50, Num Epochs: 5

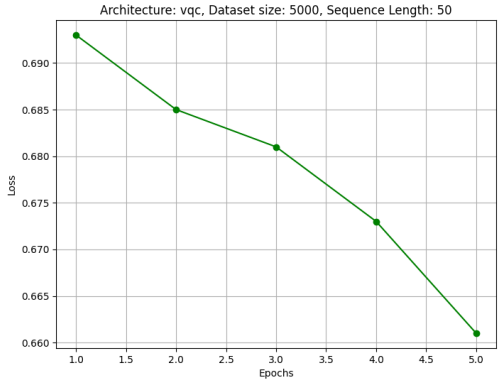


Figure B.28: Dataset size: 5000, Sequence Length: 50, Num Epochs: 5

B.3 Scaled down Quantum kernel

The pairwise inner product calculations for the quantum kernel render this architecture far too inefficient. The runtime for the simplest hyperparameters was already 1177 minutes. The positive conclusion to draw from this architecture is that already at a dataset size of 1000 reviews, and a sequence length of 30, that we are getting accuracy measurements over 60% on the training set. This was not seen for these hyperparameters for the other two models tested. We also saw an increase in the accuracy when looking at the larger sequence length of 50 tokens. We can see a trend that the more tokens/larget dataset, then the larger the runtime. The dataset size of 1000 and a sequence length of 50 were the last hyperparameters that I was able to run for this architecture due to the runtime being so long.

B.3.1 Accuracy

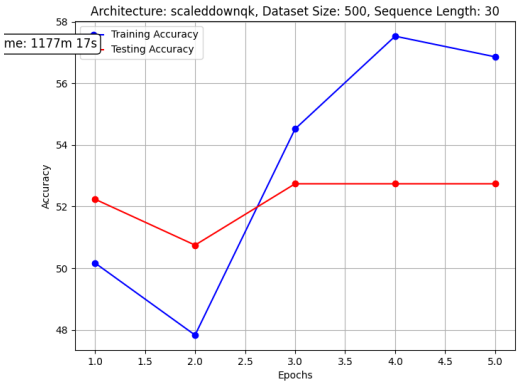


Figure B.29: Dataset size: 500, Sequence Length: 30, Num Epochs: 5

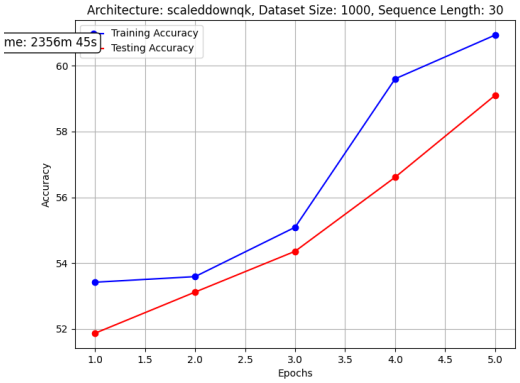


Figure B.30: Dataset size: 1000, Sequence Length: 30, Num Epochs: 5

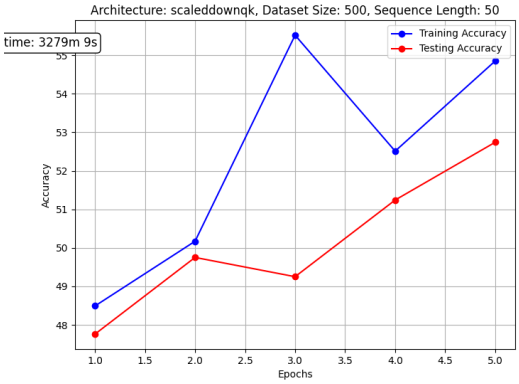


Figure B.31: Dataset size: 500, Sequence Length: 50, Num Epochs: 5

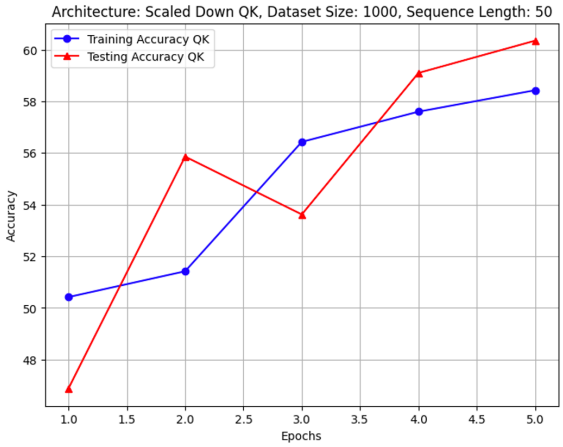


Figure B.32: Dataset size: 1000, Sequence Length: 50, Num Epochs: 5

B.3.2 Loss

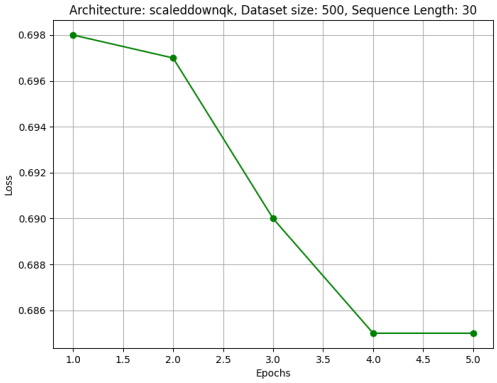


Figure B.33: Dataset size: 500, Sequence Length: 30, Num Epochs: 5

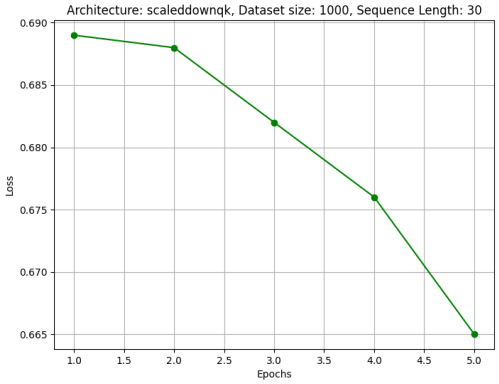


Figure B.34: Dataset size: 1000, Sequence Length: 30, Num Epochs: 5

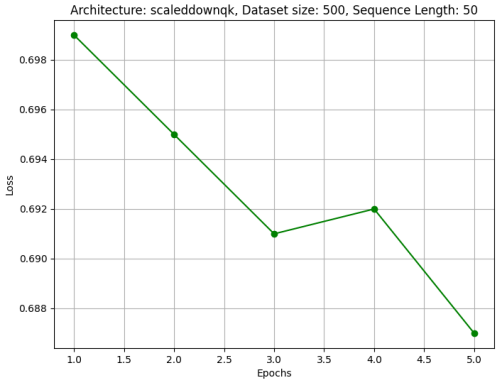


Figure B.35: Dataset size: 500, Sequence Length: 50, Num Epochs: 5

B.4 Attention-less Transformer

The attentionless transformer reached a testing accuracy of about 94% on the training set and about 84% on the testing set after 5 epochs, and leveled off slightly higher than that over the course of 20 epochs. With a dataset size of 500 and sequence length of 50, the attention is not necessary to arrive at a conclusion about the overall sentiment of the review.

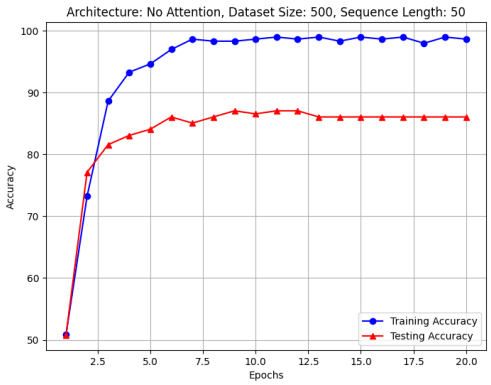


Figure B.36: Dataset size: 500, Sequence Length: 20, Num Epochs: 20

C

Appendix 3

C.1 IMDb Movie Review Data Analysis

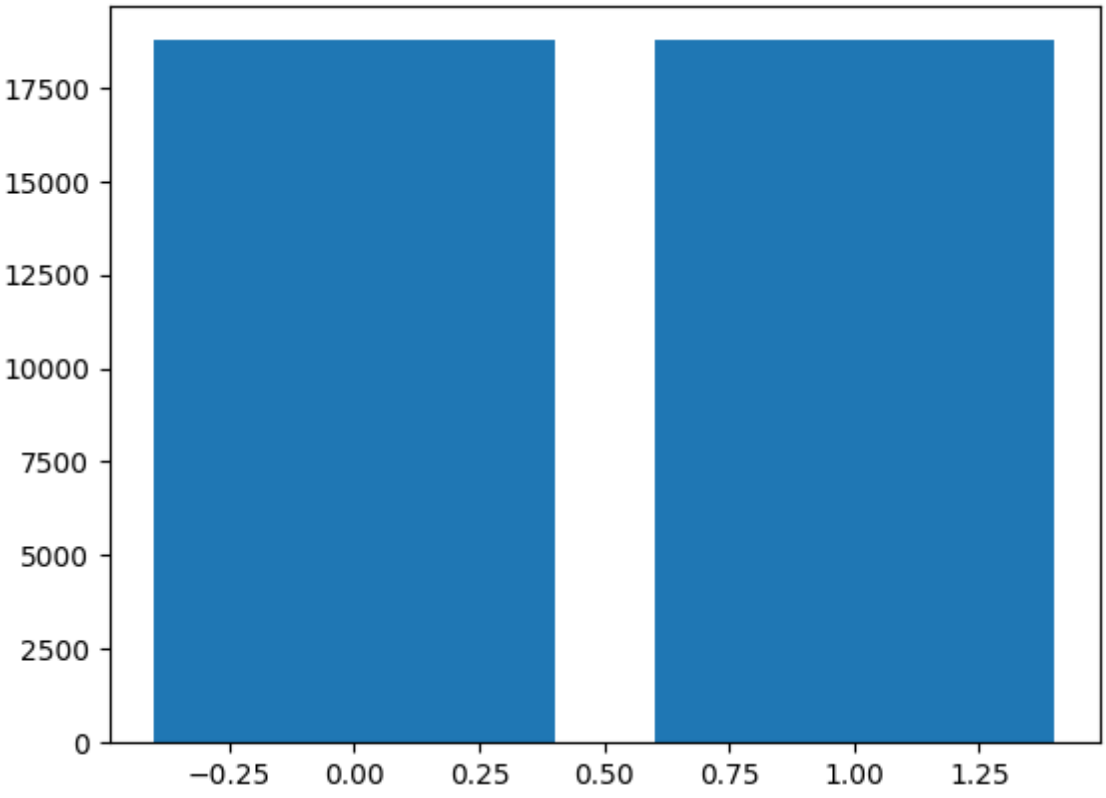


Figure C.1: Number of reviews for each sentiment

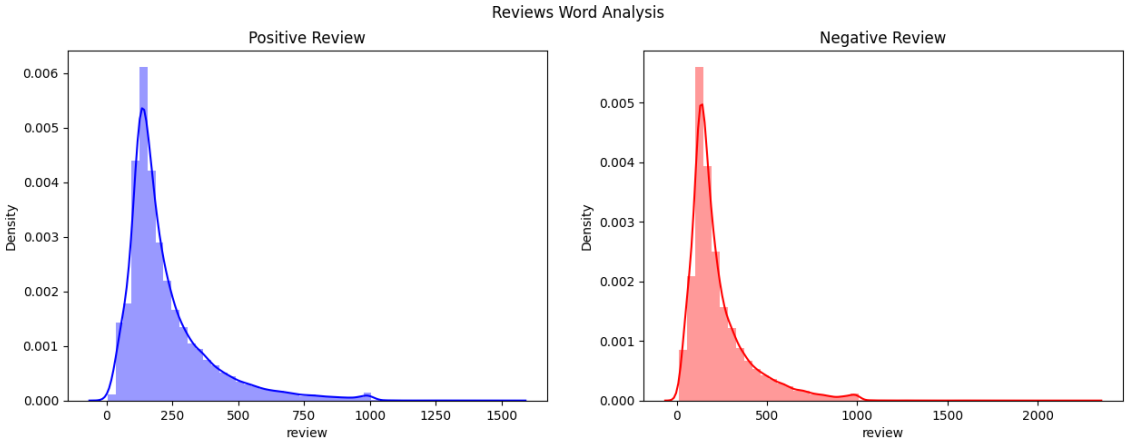


Figure C.2: Positve and negative reviews by length

```
First 10 samples of good reviews
 43032    This is an amazing film to watch or show young...
19124    This Movie was amazing, it is the kind of movi...
7620     This is one excellent Sammo Hung movie. Actual...
43776    I've received this movie from a cousin in Norw...
38271    I first see this film almost 21 years ago when...
46449    There are few films that have had me waiting a...
14904    I cannot understand why so many people did not...
28767    One of my favorite villains, the Evil Princess...
48281    This is a very realistic movie. It's the most ...
7913     I still can't describe what to feel when I rec...
Name: review, dtype: object
First 10 samples of bad reviews
 19569    This film was so amateurish I could hardly bel...
49316    I must say that during my childhood I'm quite ...
2750     Slither is a horror comedy that doesn't really...
22914    Someone else called this film a "fable-horror"...
37281    I am wanting to make a "Holmes with Doors" pun...
48695    What a terrible misfire. Not only the title bu...
5251     1983 was a bumper year for Stephen King books ...
41064    Dodgy plot, dodgy script, dodgy almost everyth...
32495    This is a rip-off of already crappy hollywood ...
9867     Its incredible to me that the best rendition o...
Name: review, dtype: object
```

Figure C.3: Samples of reviews

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY