МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет	Механико-математический
Кафедра	Международный научно-образовательный математический центр
Направление подготовки	Прикладная математика и информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

Бланксон Рафаэль

Тема работы: Применение вариационных схем в архитектурах глубокого обучения для усиления дискриминативных свойств вложений в задаче идентификации дикторов

«К защите допущена»

Заведующий кафедрой

д-р физ.-мат. наук, директор ММЦ

Марчук И. В. /……

«……»……20…г.

Научный руководитель

к.ф.-м.н

Доцент ММЦ НГУ

Павловский Е. Н. /……

«……»……20…г.

Дата защиты: «……»……20…г.

Новосибирск, 2021

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS INSTITUTION OF HIGHER EDUCATION
«NOVOSIBIRSK NATIONAL RESEARCH STATE UNIVERSITY»
(NOVOSIBIRSK STATE UNIVERSITY, NSU)

Department	Department of Mechanics and Mathematics
Chair	Mathematical Center in Akademgorodok
Field of Study	Applied mathematics and computer science

MASTER THESIS

Blankson Raphael

Thesis Title: Applying Variational Circuits in Deep Learning Architectures for Improving Discriminative Power of Speaker Identification Embeddings

«Admitted to Defense»
Head of Chair
Dr.Sci. of Physics and Mathematics
Director of MCA
Marchuk I. V. /……
«……»……20…

Scientific Supervisor
Phd Mathematics, Lead Reseacher
At SDAML lab, NSU
Pavlovskiy, E.N. /……
«……»……20…

Date of Defense: «……»……20…

Novosibirsk, 2021

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

The primary focus of this thesis is to investigate how applying quantum algorithms, specifically the variational circuit to classical deep learning architectures can improve their generalization power on speaker identification problems. To achieve this, the variational circuit was attached to the end of a simple convolutional neural network and its performance examined. Also, the variational circuit was used to replace the final layer of the pre-trained ResNet model and the performance was also examined. This introductory chapter first gives a brief history of quantum computing and machine learning to establish a context for the thesis. The problem statement is then discussed to provide a foundation for defining the scope of the thesis. Finally, the purpose of the study, limitations of the study and the overview of the entire thesis is outlined.

1.1 Context of the Study

Quantum computing and machine learning are the topics that are gaining lots of attention in recent times. This is mainly because of the breakthroughs that have been achieved in machine learning and the promise of exponential speedup offered by quantum computing. Quantum computing was born in the early 1980s by Richard Feynman [1]. He described quantum computers as computers that would be able to use the quantum properties of a qubit [2]. He wanted computers that can give an exact simulation of the complexity of interactions between quantum particles and not an approximate simulation as offered by classical computers. Yuri Manin extended Feynman's idea and showed that quantum computers could solve some problems that are considered to be computationally expensive and hard for classical computers faster [3]. There have been several algorithms that have shown to solve problems that are hard for today's fastest super classical computers. One typical example is Shor's algorithm for factoring integers [4].

In the wake of all these years of research, there is no perfect quantum computer yet ie. a fault-tolerant error-free universal quantum computer that can outperform classical computers on every task. Recent advancement has been in the area of near-term devices known as noisy intermediate-scale quantum (NISQ) [5] devices that are used for performing specific tasks and for exploring the future of quantum

applications. The recent milestone by Google in 2019 for quantum supremacy shows how powerful quantum computers are compared to classical computers. The Google team solved a task that in theory will take 10000 years to solve on a classical supercomputer in 200 seconds using a quantum computer.

The first artificial neural network was introduced in 1943 by Warren McCulloch and Walter Pitts in their effort to imitate the human brain through boolean logic gates [6]. In 1958, Frank Rosenblatt [7] proposed a more advanced model of the artificial neuron called the perceptron. The perceptron was able to process numbers as inputs and had weights that allowed it to learn and be trained over time. It had the property that a neuron will activate another neuron that is close to it as described by Hebb's rule. However, the perceptron was limited and could not recognize many classes and could not learn the XOR functions [8]. This killed the progress of development and was known as the first AI winter. In 1986, the backpropagation algorithm [9] was introduced and could be used to train a multi-layer perceptron. However, because of the slow processing of computers at the time and the lack of large datasets, progress was halted and that was the second AI winter. Recently, there has been major success and advance developments in artificial intelligence. This is mainly due to the large datasets available and also the highly powerful systems we have now. This has led to breakthroughs in areas such as computer vision, natural language processing, reinforcement learning etc. Examples include translation apps like Google translate, Apple's Siri, self-driving cars etc.

Quantum machine learning is an exciting field of quantum technology that is also gaining lots of attention. It involves different approaches to fusing quantum systems into the already existing machine learning ecosystem ie. the intersection of quantum computing and machine learning. One of such approaches is the variational circuit, a classical-quantum hybrid approach where quantum circuits mirror the functionality and structure of neural networks [10].

The novelty of this research is the use of the mixup algorithm [11] to create a form of superposition applied to quantum machine learning models that can help improve performance. The research paper was accepted and presented at the just-ended International Conference on Data Science and Applications (ICDSA 2021) and would be published in SCOPUS Indexed Springer Book Series, Lecture Notes in Networks and Systems.

1.2 Problem Statement

Embeddings is a mapping that transfers object descriptions in a hilbert space where using metrics like Euclidean metrics, etc, we can distribute classes into regions.

Discriminative power of embeddings means that disimilar vectors or objects in the embedding space are very far apart and similar objects are very close.

In deep neural networks, architectures like Resnet have shown to provide very good embeddings with good discriminative power. However, the continuous-variable quantum space operates in an infinite Hilbert space whereby for finite number of points, classification problems can be solved very effectively. In [12], they introduced the fully connected CV layer that makes use of this power in infinite Hilbert space. Can we use this fully connected CV layer approach to solve some particular task in the classical domain compared with embeddings provided by classical neural networks? Can the embeddings be improved?

1.3 Aim and Scope of the Study

This research aims to investigate the performance of quantum machine learning models on audio data. Using the variational circuit architecture proposed by [12] which extends the fully connected layer structure from classical neural networks to the quantum universe, this project first combines the variational circuit with a simple convolutional network and see if there is any performance boost (generalization) compared with a classical model trained on the same speaker dataset. A second experiment is done but this time the variational circuit is combined with a pre-trained ResNet model [13]. We then attempt to explore if mixing up two audio samples [11] to create a form of superposition will be possible to improve the performance of the quantum model.

In this study, we use the continuous variable or photonic quantum model for the experiment. ResNet18 is chosen as the pre-trained classical model.

1.4 Significance of the Study

One intended outcome of the study is to extend the applicability of quantum machine learning models to the audio domain and also open the door for more interesting research in this area.

1.5 Constraints of the Study

- There is a limited number of qubits and qumodes. Currently, Google has 72 qubit computers and about 20 qubits are available in the cloud. For qumodes, there are only 8 qumodes, which are owned by Xanadu Inc at the time of writing. This makes it very difficult to encode larger classical data.
- Currently, there is no perfect quantum computer available, thus we have to make use of Near-term quantum computers with limited qumodes.
- The only photonic quantum computer available owned by Xanadu Inc. is not yet ready for machine learning problems at the time of writing, therefore we have to simulate the quantum models on classical computers.
- The overhead of simulating these models on classical computers is very high and the process is very slow.
- Creating quantum algorithms that can outperform classical computers is very difficult since the laws of physics restrict our access to information stored in quantum systems.

1.6 Preliminaries

Below are some terms used in the thesis.

- Qubit or Qumode: The fundamental building block of quantum computing. A qubit can be defined as a mathematical object with certain quantum properties. We use operations on qubits or qumodes to construct quantum circuits. Qubits or qumodes are described as two-level quantum states, for example polarized photons, spin-1/2 particles, excited atoms, and atoms in ground state. They are usually described in the basis $|0\rangle$ and $|1\rangle$.
- Gates: Quantum circuit operations on qubits or qumodes. Table 1.1 shows some gates in the CV model.
- Superposition: Quantum particles can exist in multiple states at the same time. Mathematically, it is described as a linear combination of states given by:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \text{where } \alpha \text{ and } \beta \text{ are complex numbers} \tag{1.1}$$

- Entanglement: Extremely strong correlation that exists between quantum particles.
- Fourier Transform: converts a signal from its original domain to a representation in the frequency domain.
- Fast Fourier Transform (FFT): An algorithm that computes the discrete Fourier transform of a sequence or its inverse but has no time information.
- Short-Time Fourier Transform (STFT): Computes several FFT of a fixed frame at different time intervals and outputs a spectrogram (time + frequency + magnitude).
- Mel Frequency Cepstral Coefficients (MFCC): captures the timbral/textural aspects of sounds and approximates the human auditory system more closely.
- Spectrogram: A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time

Gate	***Unitary***
Displacement	$D_i(\alpha) = \exp(\alpha \hat{a}_i^\dagger - \alpha^* \hat{a}_i)$
Rotation	$R_i(\phi) = \exp(i\phi \hat{n}_i)$
Squeezing	$S_i(z) = \exp(\frac{1}{2}(z^* \hat{a}_i^2 - z\hat{a}_i^{\dagger 2}))$
Beamsplitter	$BS_{ij}(\theta, \phi) = \exp(\theta(e^{i\alpha} \hat{a}_i \hat{a}_j^\dagger - e^{-i\phi} \hat{a}_i^\dagger \hat{a}_j))$
Cubic phase	$V_i = \exp(i\frac{\gamma}{3\hbar}\hat{x}_i^3)$

Table 1.1: Gates in Continuous Variable (CV) model

1.7 Overview of the thesis

This thesis consists of five main chapters. In chapter 2, the relevant literature related to this current research is reviewed. Chapter 3 deals with the methodology and research design used for the experiments and the description of various tools used to perform the research. I present the results of the various experiments conducted. Chapter 5 contains the discussion and conclusions of the study.

Chapter 2

LITERATURE REVIEW

This chapter gives an overview of the relevant studies that have already been done in the different areas related to this research.

Quantum neural networks are new versions of machine learning models that work on quantum computers and use quantum properties like superposition, entanglement, etc in computations. Some quantum neural networks that have been proposed promise potential quantum advantages such as exponential speed-ups in training and also faster processing of data [14][15].

Transfer Learning is a field of machine learning where knowledge obtained in a specific context can be transferred to a different area. A representation of transfer learning can be found in [16]. One method of applying transfer learning in quantum machine learning involves the use of hybrid models where a quantum variational circuit is joined with a classical neural network to solve hard problems. This method introduces three new alternatives to the well-known classical-to-classical (CC) strategy in which already acquired knowledge is shared between classical networks. There are Classical-to-quantum (CQ), quantum-to-classical (QC) and quantum-to-quantum (QQ). Since there do not exist large error-free quantum computers and with the rise of near-term devices, the most practical method is the classical-quantum (CQ) method which combines classical neural networks with a variational quantum circuit. However, most of the CQ transfer learning has been in the qubit system and none in the qumode system [16].

There has been a breakthrough in classical machine learning using neural networks particularly deep neural networks. One of the reasons accounting for this major boost is that in deep neural networks the fundamental computational units are continuous vectors and tensors which are processed on specialized hardware (GPU and TPUs). Although the qubit systems are popular, they are not naturally suited for machine learning and rely mostly on approximations and their outputs are generally discrete. The CV system operates in a higher and infinite-dimensional space which can be leveraged for machine learning [17] and shows interesting features that can be capitalized on for machine learning tasks [18][19].

The CV model encodes information continuously and makes it a natural fit to apply

the techniques that resulted in classical breakthroughs to quantum systems. The CV network provides a native architecture for building neural network models on quantum computers similar to classical computers as shown in [7]. This approach of mimicking the structure of classical neural networks was done by creating a fully connected layer similar to the classical and applied to show a proof-of-principle but also never applied in the field of transfer learning [12].

Another reason for the major success of classical neural networks is the process of automatic differentiation. Neural networks increase the accuracy of a model gradually during training through the process of gradient-descent, ie. to minimize the loss (the inaccuracy of the model) through tweaking the weights and biases. This is done by finding the partial derivatives of the loss function. Automatic differentiation is the process that allows neural networks to be very efficient. The process is difficult to do manually and the introduction of software libraries like Pytorch, Tensorflow, MXNet, etc that comes with automatic differentiation out of the box has made it very easy. To be able to train quantum neural networks successfully for better results, automatic differentiation is key.

There has been the development of software libraries to enable automatic differentiation in quantum computing to fuel the same progress as in the classical methods: Xanadu Inc. introduced Pennylane which is an open-source software library that can perform automatic differentiation of hybrid quantum-classical computations in both qubit and qumode systems [20] and Strawberry Fields, an open-source full-stack library for designing, simulations, optimization and quantum machine learning of continuous-variable circuits also by Xanadu Inc. [10] and the recent announcement by Google introducing TensorflowQuantum (TFQ) which is also open-source but for the qubit system [21].

The recent development of quantum machine learning is quantum kernel methods which involve training a kernel method with the kernel function (2.1)

$$k(x_i, x_j) = Tr(i, j) \tag{2.1}$$

The quantum kernel methods proposed are considered to be equivalent to training a deep quantum neural network that measures observables at the end [22].

In recent times, deep learning has been used in many areas of signal processing and have outshined the traditional and previous methods used like i-vector, Gaussian mixture models(GMM), hidden Markov models, etc on a large scale where enough

data is available. Most deep learning methods like the Convolutional neural networks (CNN) have been applied to images, which are two-dimensional whiles raw audio samples are one-dimensional time-series signals that must be studied sequentially in chronological order. To use audio samples in deep learning architectures, they must be converted to two dimensions [23]. In terms of speaker recognition, research is very active and has been applied in fields like forensics, security, speaker diarization, etc. Speaker recognition in humans is a complex task that involves a combination of different features and not just the pitch content and individual speaker utterances, yet deep learning has shown significant success in the area of speaker recognition as well [24].

Chapter 3

METHODOLOGY

In this chapter, the main methods and design of the project experiment are discussed. First, a brief description of the experimental environment is given, followed by the encoding strategy used. Next, a description of the dataset and preprocessing are discussed. A description of the models with their architectures used in the experiment is presented.

Speaker identification is a part of speaker recognition and involves the process of determining which speaker made a particular utterance among a set of speakers. On a simple note, it involves comparing the given voice with a set of already stored collection of speakers in order to find out who spoke. There are two forms of speaker identification namely text-dependent and text-independent. In the text-dependent form, the words in the utterance are the same as the ones used in training of the set of speakers. Eg. Hey Siri in apple devices. The text-independent form on the other hand is not restricted to a particular word or text in the utterance of the speaker to be identified. We use the text-independent form in this work. To perform speaker identification, we extract the feature vectors (embeddings) of the given voice of the speaker and compare with the set of speakers whose embeddings we have already extracted. The embedding from the set of speakers which is similar or has the smallest vector distance to the given embedding is the speaker we are looking for. ie. similar embeddings will have smaller distance between them whiles different embeddings will have larger distance between them. An example of the vector distance is the euclidean distance.

Embeddings is the way we represent a vector of high dimensions into a low dimension using machine learning techniques. Embeddings usually captures features of the input signal and puts similar inputs close together in the embedding space. This is also known as the discriminative power of embeddings.

An embedding is a translation of a high-dimensional vector into a low-dimensional space. Ideally, an embedding captures some of the semantics of the input by placing similar inputs close together in the embedding space.

The general goal of most quantum algorithms is to perform tasks efficiently in a Hilbert space. This means embedding the data (input vectors) unto a higher

dimensional feature space to make the task easier to solve. In our case, this means we are able to separate embeddings of different speakers based on their distance easily. The CV quantum model operates in the infinite dimensional Hilbert space [19]. This makes the CV model enormously powerful such that we can use a linear model to separate input vectors (discrimination). Thus our speaker embeddings can be easily identified easily irrespective of how complex they might be.

3.1 Experimental Environment

All the experiments were done with the classical-quantum model. The quantum model architecture was designed with Pennylane [20] with its Pytorch interface on the Strawberryfields [10] simulator while the overall training was done with Pytorch-Lightning for easier code reproducibility. The preprocessing of audio data was done with Librosa. However, in future, the experiment could be run on actual strawberryfields hardware when it becomes available.

3.2 Encoding Methodology

The best method to encode classical data into a quantum circuit is still an open research problem. In this thesis, the Displacement embedding was chosen to encode the audio data into the continuous-variable quantum circuit. Mathematically, it is given by (3.1)

$$D_i(\alpha) = exp(\alpha \hat{a} y_i^{\dagger} - \alpha^* \hat{a}_i) \tag{3.1}$$

3.3 Dataset

The dataset used in this thesis was taken from the Speaker Recognition dataset from Kaggle [25]. It contains speeches of 5 prominent leaders. Each audio is one second long with a sample rate of 16000. For this work, two speakers, Nelson Mandela and Benjamin Netanyau were selected. 25% of the data was used for test and 20% of the remaining was used for validation. This was done to prevent the model from overfitting. Experiment was run on the 2 selected speakers.

In preprocessing the data with Librosa, MFCC was extracted from the raw audio waveform at the original sample rate of 16000, with hop length of 512, frame length of 2048 of fast-Fourier transform and segmented with size 10. Since Pytorch expects the number of channels first in the shape of the vector, the shape of the vector embedding is swapped to satisfy the Pytorch requirements with the channel first. Next, the MFCCs with length equal to the expected number of MFCC vectors per segment together with their respective labels for each speaker is stored in a

JSON file that will be used in the training process. The expected number of MFCC vectors per each audio segment is calculated as expected 3.2. Figure 3.1 and figure 3.2 shows the MFCC generated from random audio files for each speaker.

$$\text{number of MFCC per segment} = \frac{\text{number of samples per segment}}{\text{hop length}} \tag{3.2}$$

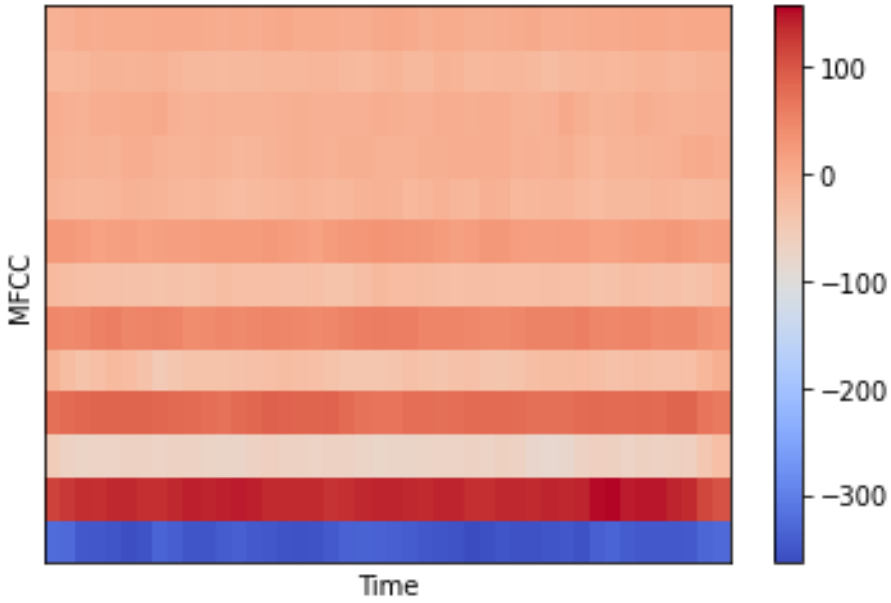

Figure 3.1: MFCC for Benjamin Netanyau

3.4 Tested Models

ResNet

The main model used across the experiments is the ResNet. It was introduced in 2015 by Microsoft to solve the problem deep neural networks face in optimizing parameters. Deeper models usually find it very difficult to converge because of vanishing and exploding gradients which could be solved by normalization, however, this poses another problem where the accuracy of the model becomes saturated and degrades rapidly. This is not caused by overfitting and stacking more layers increases the training error [26]. This was solved in the proposed ResNet model by introducing identity shortcut connections in the network architecture that skips one or more layers. In simple terms, resnet involves skip connections and stacking convolutional layers together and finally adding the originally input. A residual block is shown in figure 3.3. The identity shortcut introduces the term x in the output and therefore the output becomes $F(x)+x$. The shortcut is extremely helpful

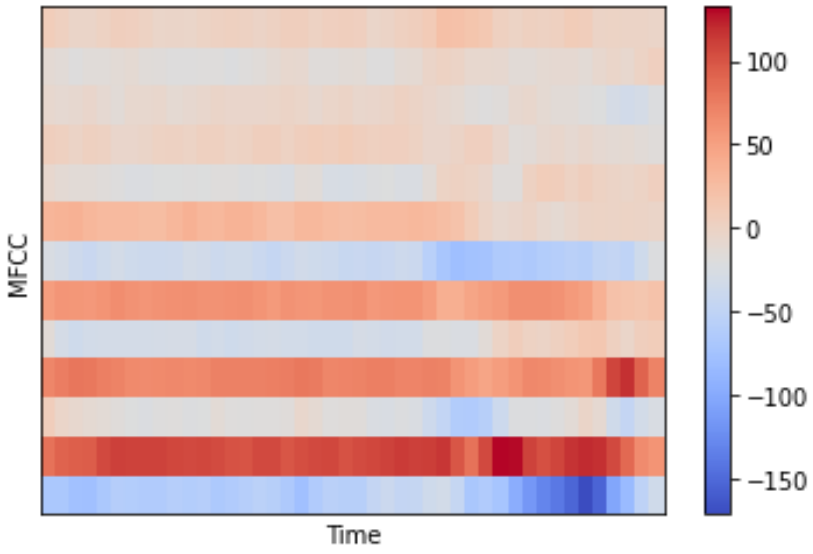

Figure 3.2: MFCC for Nelson Mandela

in the training process as it reduces the training error significantly. ResNet was selected for the experiments because it generates better embeddings from raw input features ie. embeddings with more discriminative power.

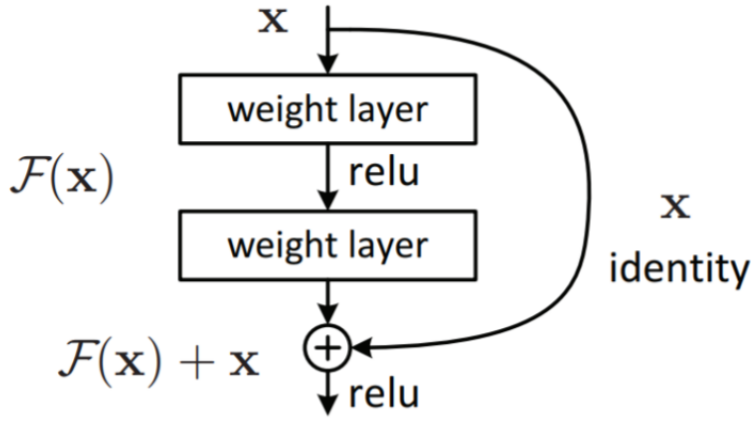

Figure 3.3: Residual Learning building block [13]

Base Model

The base model used as a benchmark for this experiment is the ResNet18 model. It is made up 18 main layers hence the name ResNet18. It was pre-trained on the ImageNet dataset with an input shape of (224×224) for classifying images into

1000 categories. Since audio data is 1-dimensional, it has to be upsampled to match the input shape required for the ResNet18 model.

Base Model Training

During training we replace the input and output for first convolutional layer of the ResNet18 model called `conv1` with 1, 64 for the input and ouput dimension respectively. We then added a kernel size of 7, a stride of 2 and padding of 3×3. We then freeze all the layers of the ResNet18 model except the final fully-connected layer which we replace with another fully connected layer that outputs 2, the number of classes we want to return. We train the model for 5 epochs and perform validation for every 0.25% step of each epoch. We use the cross entropy loss and the optimizer used to update the weights is the Adam optimizer.

The architecture of ResNet18 is shown in figure 3.4 and a summary of the architecture with parameters is shown in figure 3.5. The results are shown in the table 3.1. We show the loss and accuracy plot of the training in figure 3.6 and figure 3.7 respectively. Prediction results from the ResNet18 base model is shown in figure 3.8.

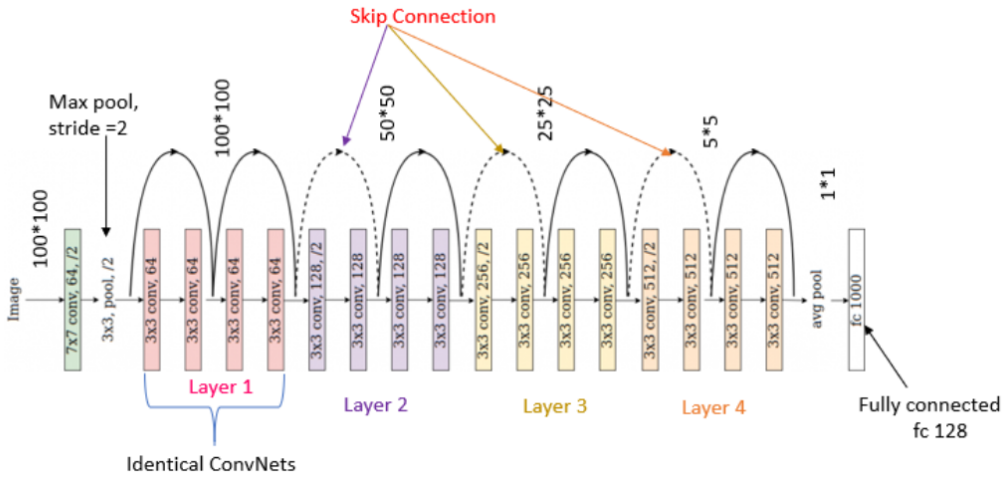

Figure 3.4: ResNet18 architecture[13]

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 3.5: ResNet18 architecture summary

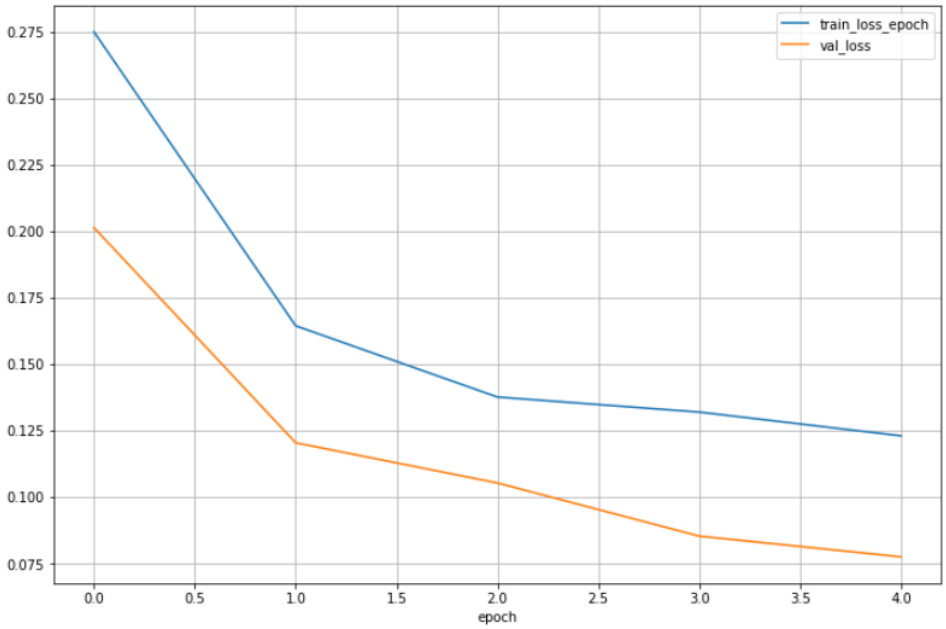

Figure 3.6: Train vs Validation loss of the base model ResNet18

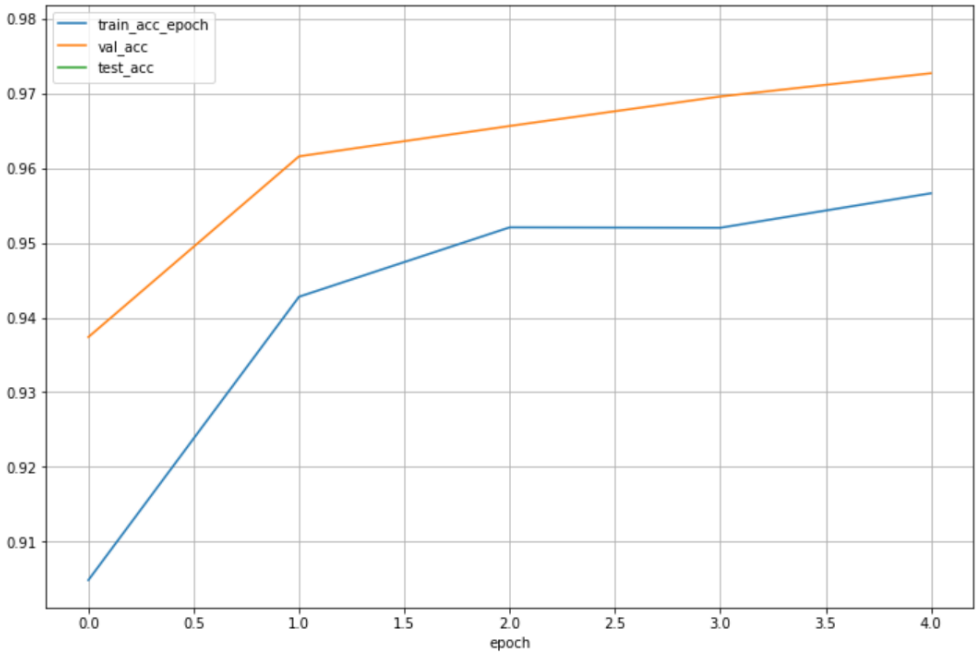

Figure 3.7: Train vs Validation accuracy of base model ResNet18

Out[64]:

	prediction	pred_label	labels
0	[-1.56, 2.52]	1	1
1	[-1.93, 2.12]	1	1
2	[2.23, -2.33]	0	0
3	[3.03, -3.12]	0	0
4	[2.9, -3.43]	0	0
5	[1.92, -2.46]	0	0
6	[-0.05, 0.09]	1	0
7	[1.67, -2.5]	0	0
8	[2.34, -2.91]	0	0
9	[-3.3, 3.4]	1	1
10	[2.01, -3.05]	0	0
11	[-3.01, 1.49]	1	1
12	[4.14, -4.8]	0	0
13	[2.9, -4.0]	0	0
14	[3.36, -3.69]	0	0
15	[-3.61, 3.72]	1	1

Figure 3.8: Prediction for base model ResNet18

Quantum Model

As described in [12], the quantum model used is the equivalence of the classical fully connected architecture in the continuous variable quantum domain. A quantum neural network can be built from a sequence of layers, with each layer taken from the set of universal quantum gates. The fully connected quantum layer used is created from the following layers stacked up: an interferometer, displacement gate and squeezing gate. These gates act as an affine transformation. The interferometer is made up of beam splitter and rotation gates. The fully connected layer uses a non-gaussian gate in our work the Kerr gate as a non-linear activation function. An image of the fully connected quantum CV layer used in the experiment is shown in figure 3.9 . The quantum model was simulated on the Xanadu "strawberryfields.fock"

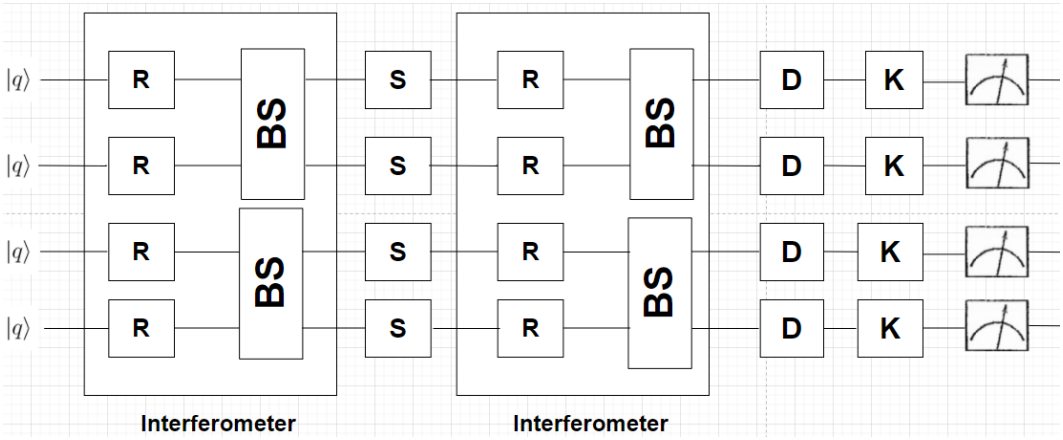

Figure 3.9: Circuit structure of a single CV fully connected layer

*CV fully connected layer made up of interferometer made up of **R** - Rotation gate ans **BS** - Beam Splitters, **S** - a squeezing gate, another interferometer, **D** - a displacement gate and the non-gaussian gate **K**- Kerr gate*

simulator interface and programmed using Xanadu's Pennylane software with a cut-off dimension equal to 7.

Quantum Experiments

The experiments performed in this work uses an hybrid approach which involves combining a classical neural network model and a CV quantum model. As mentioned in section 1.5, it takes over 4 hours to simulate and run such a small dataset on our classical computer for just one epoch. Another hinderance is that, our computational resources are not enough to run for several hours. We performed two experiments with the quantum model.

Simple Classical-Quantum Model

The first experiment was done with a simple classical convolutional network combined with the fully connected quantum model. The details of the architecture is shown in 3.11. The classical model was made up of a Convolutional layer, max pool layer and a fully connected layer. The convolutional layer used RELU activation with a filter size of 3-by-3. The fully connected layer has 768 hidden neurons, a RELU activation and a dropout layer of 0.3 and returns 64 vectors after the dropout. We use a fully connected layer as pre-processing block for the quantum layer which takes the 64 neurons and returns 4 neurons which is equal to the number of qumodes needed as the input for the quantum model. The classical full connected layer was built with the pytorch function
`torch.nn.Linear(in_features, out_features, bias=True)`
which applies a linear transformation to the incoming data of the form of equation 3.3 and the in_features, out_features are the size of the input and output samples respectively.

$$y = xA^T + b \tag{3.3}$$

The RELU (rectified linear unit) function is defined mathematically as equation 3.4 and the convolutional layer is built in pytorch with the function

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,groups=1, bias=True, padding_mode='zeros')`

$$\varphi(x_i) = \max(0, x_i) \tag{3.4}$$

After the pre-processing block, we feed the 4 output neurons to a tanh function to put the values between (-1, 1) and multiply by the constant $\frac{\pi}{2}$ for scaling. The neurons are then embedded in the quantum space using displacement embedding to produce 4 qumodes. The 4 qumodes are then fed to an inteferometer, a squeezing gate, another interferometer, a displacement gate in that order. The output are then fed to the non-linear kerr gate. The system is then measured on all four qumodes or wires to obtain a list of classical expectation values. The measurement outputs are then post-processed using a classical linear layer. The classical layer takes the 4 measurement outputs as inputs and returns 2 neurons which are then passed to sigmoid layer to be classified based on highest probability. The architecture design for this experiment is shown in figure 3.10.

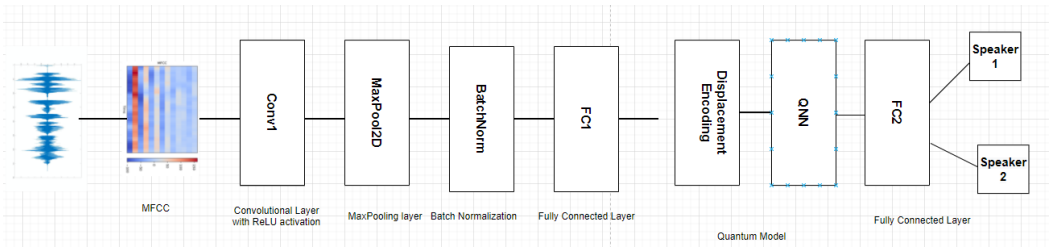

Figure 3.10: Architecture of simple CNN with the fully connected CV quantum layer

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 5, 13]             640
              ReLU-2           [-1, 64, 5, 13]               0
         MaxPool2d-3            [-1, 64, 2, 6]               0
           Flatten-4                 [-1, 768]               0
            Linear-5                  [-1, 64]          49,216
              ReLU-6                  [-1, 64]               0
           Dropout-7                  [-1, 64]               0
            Linear-8                   [-1, 5]             325
================================================================
Total params: 50,181
Trainable params: 50,181
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.08
Params size (MB): 0.19
Estimated Total Size (MB): 0.27
----------------------------------------------------------------
```

Figure 3.11: Details of Simple CNN used

ResNet18 classical model with Quantum Model

In the main experiment, a pre-trained ResNet18 model was combined with the fully connected quantum model described above. The shape of the MFCC generated from the raw audio files has the shape $(1, 4, 13)$, where 1 is the input channel but since the ResNet18 was pre-trained on imageNet with input shape $(3, 224, 224)$ ie. 3 channels and width, height = 224, the MFCC generated from the embeddings was upsampled to the shape $(1, 224, 224)$. The final fully connected layer of ResNet18 was removed and the remaining layers used as a feature-extractor that returns 512 output features. Another fully connected layer used as a pre-processing layer is attached. It takes the 512 feature vectors and returns 4 neurons that will be used as

input for the fully-connected quantum model. We feed the these output neurons to a tanh non-linear activation function to put the values between (-1, 1) and multiply by the constant $\frac{\pi}{2}$ for scaling. The output of this operation are then embedded in the quantum space using displacement embedding to produce 4 qumodes just as described earlier. The output qumodes are then fed to an inteferometer made up of rotation gates and beam splitters, a squeezing gate, another interferometer, a displacement gate in that order. The output are then fed to the non-linear kerr gate. The system is then measured on all four qumodes or wires to obtain a list of classical expectation values. The measurement outputs are then post-processed using a classical linear layer. The classical layer takes the 4 measurement outputs as inputs and returns 4 neurons. We use an additional fully connected layer to map the 4 output neurons to two outputs corresponding to each speaker which are then passed to sigmoid layer to be classified based on highest probability. The binary cross-entropy loss is also used in this experiment. The architecture for ResNet18 combined with the quantum model is shown in figure 3.12.

Quantum model Training

The quantum models are trained on the same dataset as the base model. We train the models for 5 epochs. During training we accumulated the gradients over every 8 steps of a batch. The justification is because our resources are not too powerful to run the quantum models for longer hours without crushing. We use the binary cross-entropy loss and the Adam optimizer to adjust the model weights during training. Mathematically, the binary cross-entropy loss is given by equation (3.5):

$$\begin{aligned} L_{BCE} &= -\sum_{i=1}^{2} y_i \log(p_i) \\ &= -[y \log(p) + (1-y)\log(1-p)] \\ &\text{where } y_i \text{ is the true labels and } p_i \text{ is the softmax probability for class i} \end{aligned} \tag{3.5}$$

The results from the two experiments was worse as compared to the base model. Table 3.1 shows the results together with some hyper-parameters used. The quantum models performed worse compared to the base model. Although the quantum models maybe powerful, the smaller size of the dataset and the duration of 1 second are the possible cause of the poor performance. We observed that the loss was just hovering around the region of 0.65 without any major changes for the quantum models. The model checkpoint was saved to a csv file. We read the metrics from the checkpoint using pandas into a dataframe and plotted the loss and accuracy with matplotlib

library. The plots are shown in figure 3.13 and 3.14.

Parameters	ResNet18	CNN + QNN	ResNet18 + QNN
Sample rate	16000	16000	16000
Num of epochs	5	1	3
Batch size	16	16	16
Learning rate	0.01	0.01	0.017
Time per epoch (min)	1.25	258.636	219.37
Test Accuracy	98%	92%	74%

Table 3.1: Initial Results

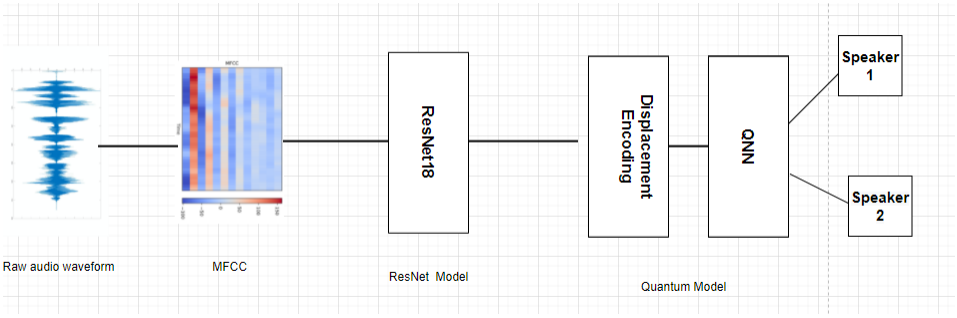

Figure 3.12: ResNet combined with the fully connected CV quantum layer

Improving the Models Using Mixup

From table 3.1, we can see that the hybrid models performed badly as compared to the base model. To improve the performance of the model, a data-agnostic data augmentation technique called mixup was used [11]. Mixup extends the distribution of the training data by incorporating linear combinations of the hidden features of the training data. This enables the model to explore and examine the important areas of the embedding space effectively. We perform the same linear combination also for the labels which are one-hot encoded to mix the labels as well. Mixup solves the problem of neural networks memorizing training data even when strong regularization techniques are applied. This problem is evident when neural networks are applied to a dataset from a slightly different distribution than the training data. Neural network models in this instance give the wrong prediction with a very high confidence.

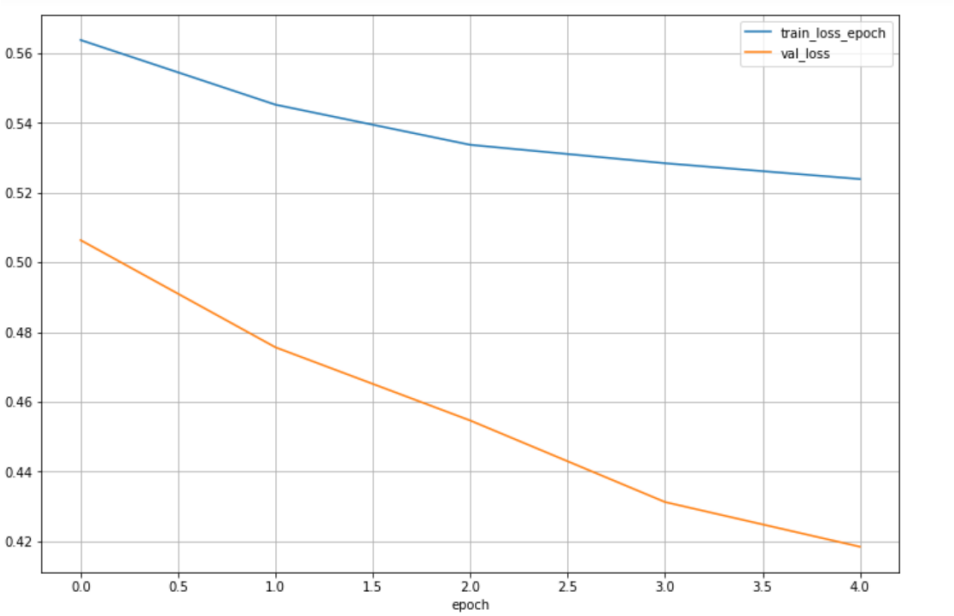

Figure 3.13: loss for quantum model

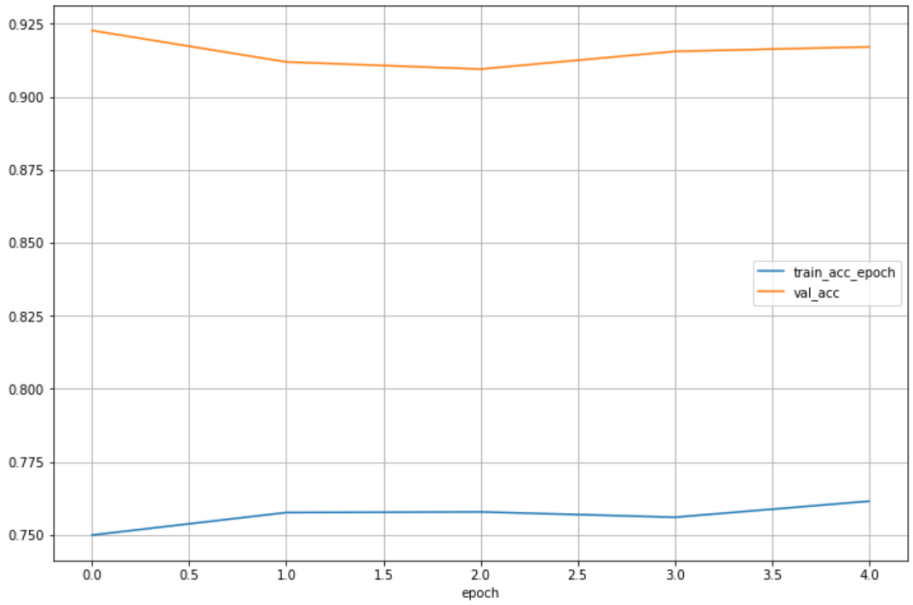

Figure 3.14: accuracy for quantum model

We have chosen to apply mixup because it has been proven to increase the generalization power of neural networks and also increase their prediction on novel data with different distribution. It is also very easy to implement using very few lines of code as can be seen in figure 3.15. Mixup creates virtual training examples using

equation (3.6)

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j \tag{3.6}$$

(x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data where x_i, x_j are raw inputs and y_i, y_j are one-hot encoded labels and $\lambda \in [0, 1]$ drawn from the beta distribution. In this experiment, we took $(x_i, y_i), (x_j, y_j)$ from speaker A and speaker B respectively. It is possible to run mixup on the raw inputs before feeding them to the model. We did not use this approach because we run out of memory when generating the examples from mixup on the raw input. However, we applied mixup on each batch of training data to generate new batches that contains the two audio input from the speakers mixed, a form that mimics superposition of quantum systems.

In [23]:
```python
# mixup trial
def mixup_data(x, y, alpha=1.0):
    """Returns mixed input, pairs of targets and lambda"""
    x1, x2 = x
    y1, y2 = y

    if alpha > 0:
        lam = np.random.beta(alpha, alpha)
    else:
        lam = 1
    mixed_x = lam * x1 + (1 - lam) * x2
    return mixed_x, y1, y2, lam
```

Figure 3.15: mixup code in python

We use the same models as described earlier in the earlier experiments.ie. the ResNet18 base model and the two hybrid models. However during training, we unpack two features from the batch belonging to each speaker with their labels before feeding into the neural network models as shown in figure 3.16. We also use a custom loss function which we call the mixup-criterion that incorporates the mixup technique in a loss function in our case the binary cross-entropy loss shown in figure 3.17. We choose an $\alpha = 0.7$ that is used to select λ randomly from the beta distribution.

```
    def training_step(self, batch, batch_idx):
#         print(batch)
        d1 = batch['data1']
        d2 = batch['data2']
#         print(type(x1))
        audio1, label1 = d1[0], d1[1]
        audio2, label2 = d2[0], d2[1]

        audio, label_a, label_b, lam = mixup_data((audio1, audio2), (label1, label2), 0.7)
        outputs = self(audio)
        labels = lam * label_a + (1 - lam) * label_b
        loss = self.bce(outputs.float(), labels.float())
#         loss = mixup_criterion(self.bce, outputs.float(), label_a.float(), label_b.float(), lam)
        acc = metrics.functional.accuracy(outputs, labels.long())

        self.log_dict({'train_loss': loss, 'train_acc': acc}, prog_bar = True, on_step = True, on_epoch=True, logger=True)
        return loss #{'val_loss': loss, 'val_acc': acc}
```

Figure 3.16: mixup during training step code in python

```
In [24]: def mixup_criterion(criterion, pred, y1, y2, lam):
             return lam * criterion(pred, y1) + (1 - lam) * criterion(pred, y2)
```

Figure 3.17: mixup loss function code in python

The training process is the same as the earlier experiments without mixup. Results for mixup is shown in the next chapter.

Chapter 4

RESULTS

The results from the mixup experiments performed in this project are shown shown in this chapter. Table 4.1 shows the results of the experiment with mixup incorporated. The plot of the loss and accuracy for the base model with mixup is shown in figure 4.1 and figure 4.2 respectively. Figure 4.3 and figure 4.4 shows the plot for loss and accuracy of the quantum model with mixup.

Experiment	ResNet18	CNN + QNN	ResNet18 + QNN
Validation Accuracy	100%	98%	99%
Test Accuracy	100%	100%	100%

Table 4.1: Results With Mixup

Mixup gave all the models a boost in performance to reach 100% accuracy. One reason is because of the smaller size of the dataset. Because of the intensive computation power needed for the quantum models, the number of epochs differed between the models to avoid out of memory errors. Although, quantum models achieved accuracy on par with the classical models, they have a higher computation time because it takes a lot of time and resources to simulate quantum models on classcal systems. This compute time can be minimized in future when actual photonic quantum processing units (QPU) becomes available. We also observed that using mixup did not affect the training time, which proves that mixup does not add any overhead to computation. The validation accuracy of the quantum models did not change from the first epoch which means more training would not affect the results.

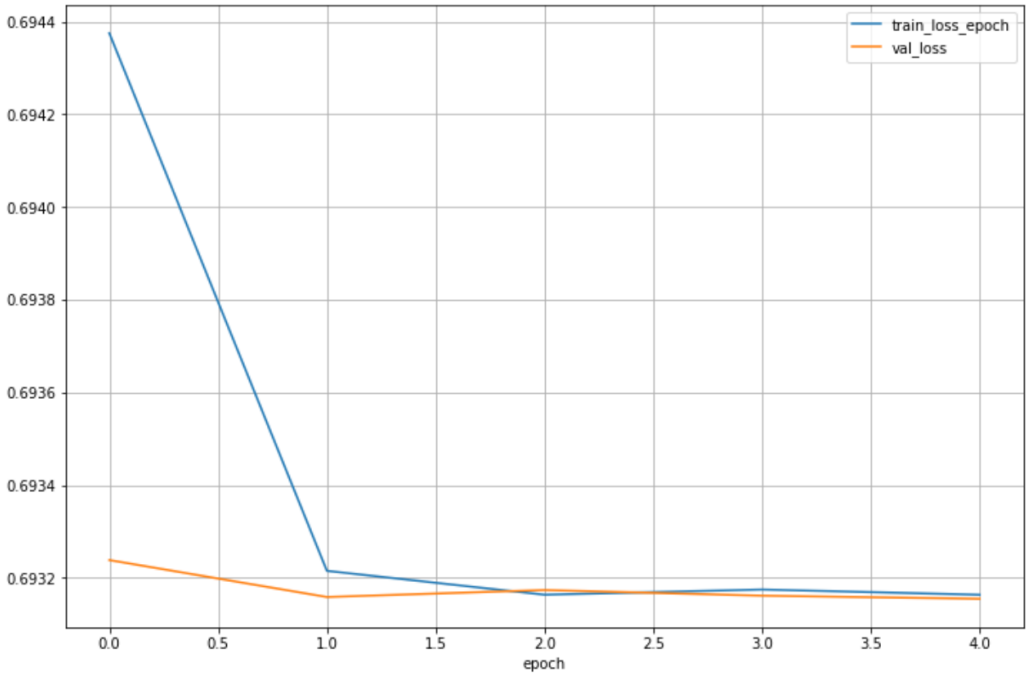

Figure 4.1: plot of loss for the base model ResNet18 with mixup

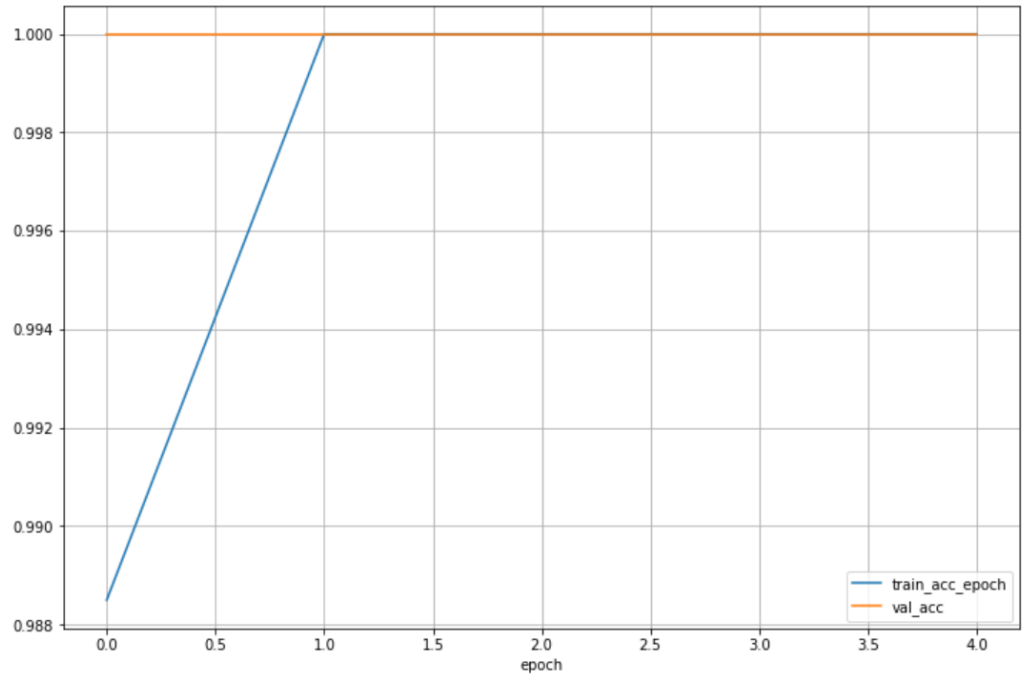

Figure 4.2: plot of accuracy for base model ResNet18 with mixup

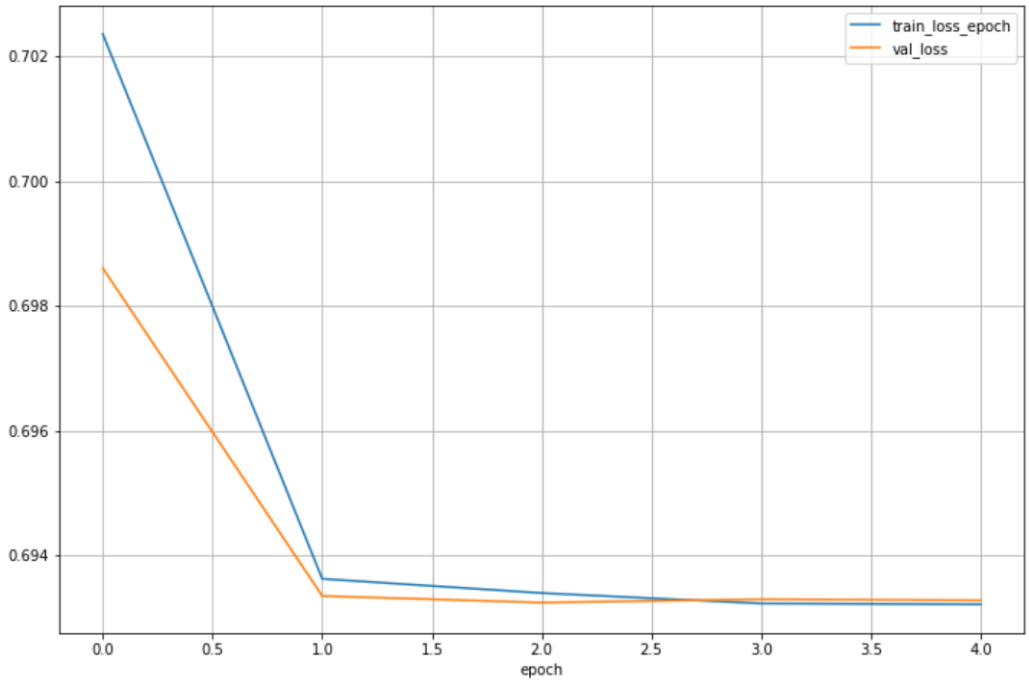

Figure 4.3: plot of loss for the quantum model with mixup

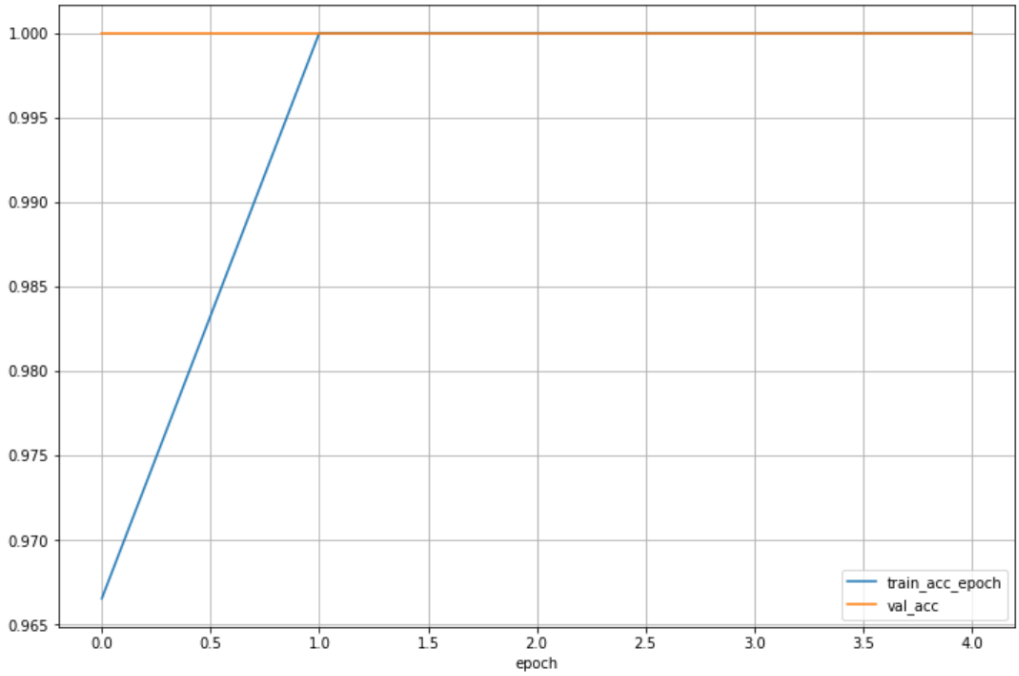

Figure 4.4: plot of accuracy for quantum model with mixup

Chapter 5

CONCLUSION

The main goal of the thesis was to investigate if speaker embeddings can benefit from the power of quantum models with a focus on continuous-variable (photonic) quantum models. A simple classical convolutional model and ResNet18 were used together with a quantum version of the fully connected feed-forward neural network.

The experiments showed that we can use the CV model to improve speaker embeddings in the classical domain. Eventhough the initial experiments produced worst results compared to the classcal model, by using mixup, our results was on par with the classical model. The results of the experiments improved when the mix-up technique was applied.

This shows that the generalization performance of the network can be improved by quantum techniques and shows techniques that can be used in future research to advance quantum machine learning. However, further research is needed to achieve a quantum advantage over classical models.

REFERENCES

[1] Richard P Feynman. "Quantum mechanical computers". In: *Optics news* 11.2 (1985), pp. 11–20.

[2] Richard P Feynman. "Simulating physics with computers". In: *Feynman and computation*. Vol. 296. 1982, pp. 467–488. DOI: https://doi.org/10.1007/BF01886518.

[3] Y Manin. "Classical Computing, Quantum Computing, and Shor's Algorithm". In: *Talk at the Bourbaki Seminar*. 1999.

[4] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332.

[5] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79.

[6] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[7] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[8] Marvin Minsky and Seymour Papert. "An introduction to computational geometry". In: *Cambridge tiass., HIT* (1969).

[9] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating". In: *errors. Cognitive modeling* (1988).

[10] Nathan Killoran et al. "Strawberry fields: A software platform for photonic quantum computing". In: *Quantum* 3 (2019), p. 129.

[11] Hongyi Zhang et al. "mixup: Beyond empirical risk minimization". In: *arXiv preprint arXiv:1710.09412* (2017).

[12] Nathan Killoran et al. "Continuous-variable quantum neural networks". In: *Physical Review Research* 1.3 (2019), p. 033063.

[13] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[14] Maria Schuld. *Supervised learning with quantum computers*. Springer, 2018.

[15] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "The quest for a quantum neural network". In: *Quantum Information Processing* 13.11 (2014), pp. 2567–2586.

[16] Andrea Mari et al. “Transfer learning in hybrid classical-quantum neural networks”. In: *Quantum* 4 (2020), p. 340.

[17] Maria Schuld and Nathan Killoran. “Quantum machine learning in feature Hilbert spaces”. In: *Physical review letters* 122.4 (2019), p. 040504.

[18] Siddhartha Das, George Siopsis, and Christian Weedbrook. “Continuous-variable quantum gaussian process regression and quantum singular value decomposition of nonsparse low-rank matrices”. In: *Physical Review A* 97.2 (2018), p. 022315.

[19] Hoi-Kwan Lau et al. “Quantum machine learning over infinite dimensions”. In: *Physical review letters* 118.8 (2017), p. 080501.

[20] Ville Bergholm et al. “Pennylane: Automatic differentiation of hybrid quantum-classical computations”. In: *arXiv preprint arXiv:1811.04968* (2018).

[21] Michael Broughton et al. “Tensorflow quantum: A software framework for quantum machine learning”. In: *arXiv preprint arXiv:2003.02989* (2020).

[22] Hsin-Yuan Huang et al. “Power of data in quantum machine learning”. In: *Nature communications* 12.1 (2021), pp. 1–9.

[23] Hendrik Purwins et al. “Deep learning for audio signal processing”. In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (2019), pp. 206–219.

[24] Mitchell McLaren, Yun Lei, and Luciana Ferrer. “Advances in deep neural network approaches to speaker recognition”. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 4814–4818.

[25] kongaevans. *Kaggle speaker dataset*. URL: `www.kaggle.com/kongaevans/speaker-recognition`.

[26] Kaiming He and Jian Sun. “Convolutional neural networks at constrained time cost”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5353–5360.