Acoustic-based Device-Free Finger Motion Tracking

Lancaster University

Yi-Xin Huang
B.Sc Computer Science

March 2020

Abstract

With recent trends suggesting that smartphone screen sizes are exponentially increasing, it is clear that there is a demand for an increased interaction surface for smartphones. Therefore this project aims to explore a potential way to increase the interaction surface of existing smartphones without the need of purchasing new devices. It does so by using the user's finger sliding across the desk as input. It is mentioned that acoustic signals when writing different letters are unique and so the system is able to extract the acoustic signal's features with the use of spectrogram images. These images were tested on different classification models of CNN and KNN and their performance was compared and analysed. It was found that KNN achieved a higher accuracy than CNN. However, the results heavily suggests that the data set used in this system is too small and lacks variance, leading both models to overfitting. However, with further works and improvement, an accurate stroke detection system can be achieved using this method.

Declaration

I certify that the material contained in this dissertation is my own work and does not contain unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation. Regarding the electronically submitted work, I consent to this being stored electronically and copied for assessment purposes, including the School's use of plagiarism detection systems in order to check the integrity of assessed work. I agree to my dissertation being placed in the public domain, with my name explicitly included as the author of the work.
Name:
Date:

Contents

Chapter 1

Introduction

Mobile devices have had a history of having a small screen size in order to achieve better portability, ease of use and compactness for the user. However, with the rise of smart devices, smartphones have now become more than a communication device but also an entertainment device. This cause users to now demand bigger screens to interact with their device, thus a new trend has risen as the screen of smartphones exponentially increases every year [4]. In response to this, companies such as Samsung are now combining compactness with bigger interactive surfaces with the newest feature being foldable phones [3]. Unfortunately, the only way for users to gain access to bigger interactive surfaces is to purchase new devices and in the case of foldable smartphones, the price ranges from 1300 and 2000 dollars making it a barrier for most users [17].

Furthermore, with the penetration rate of smartphones in the United Kingdom being around 78% as of 2018 [15], smart phones are now being exposed to a wider range of users. The percentage of users age 55-64 who uses a smartphone has increased from 9% to 71% in the past decade [16]. A bigger interactive screen would benefit these users but at the same time the controls of the smart device may be unintuitive.

Therefore, this project will explore the potential of an interaction medium using the acoustic signal generated by the finger sliding on a nearby surface as input. This interaction will consist of the user writing out individual letters on the surface next to the smartphone as if they are writing it on paper. This interaction medium has several advantages. It allows users to expand the interaction surface without the use of any other hardware or wearable devices. It also helps users in need to interact with their smartphones in a more intuitive way. An observation is made by Chen et al, that different stroke gestures on a table surface will generate unique acoustic signatures [9]. Therefore, by capturing those signals and extracting the audio features, we can classify the gestures using methods such as neural networks or instance-based learning algorithms. To further extract characteristics of the

acoustic signals, we'll convert the signals into spectrogram images and feed them into a convolutional neural network which has a good reputation at classifying images.

1.1 Project Aims

The aim of this project is to explore a system that can extends the user's interaction with their smart device beyond the screen. By writing on the surface next to the device with their fingers. In order to achieve this, the project must fulfil the following objectives:

1. Build a system that will be able to recognize user's input of the 26 capitalised letters of the English alphabet with high accuracy.
2. Explore 2 different classification methods between instance-based learning (KNN) and deep neural networks (CNN).
3. Able to differentiate the difference between a letter acoustic signal and background acoustic signals.
4. Doesn't require the user to set up additional hardware or wear external devices for the system to function.
5. The user may use the system on a variety of surfaces and still retain a high recognition accuracy .
6. (Optional) Port the system into a smartphone instead of classification on the laptop.

1.2 Project Overview

The rest of the paper will cover the following. Chapter 2 covers the background and research, providing more context into the topics such as the advantages of neural networks, how spectrograms extracts audio characteristics and other related works. Chapter 3 will explain the design and implementation of the proposed system. Chapter 4 will analyse the system through testing and evaluation. Chapter 5 are my conclusions to this project and future works and improvements that could be made to the system.

Chapter 2

Background

There have been numerous researches done on classifying acoustic signals such as speech recognition and gesture recognition [9, 10, 19]. All have a common approach of extracting acoustic signal features using Fast Fourier Transform and spectrograms and turn it into an image classification for neural networks. For this implementation the models KNN and CNN has been selected as being the two of the most popular classification methods, with CNN having a reputation of being a reliable image classification neural network.

2.1 Spectrogram

Simple raw acoustic signals do not have enough features to accurately define specific letters. As seen in the graph below, some letters will have similar features and won't be easily distinguishable through as an image.

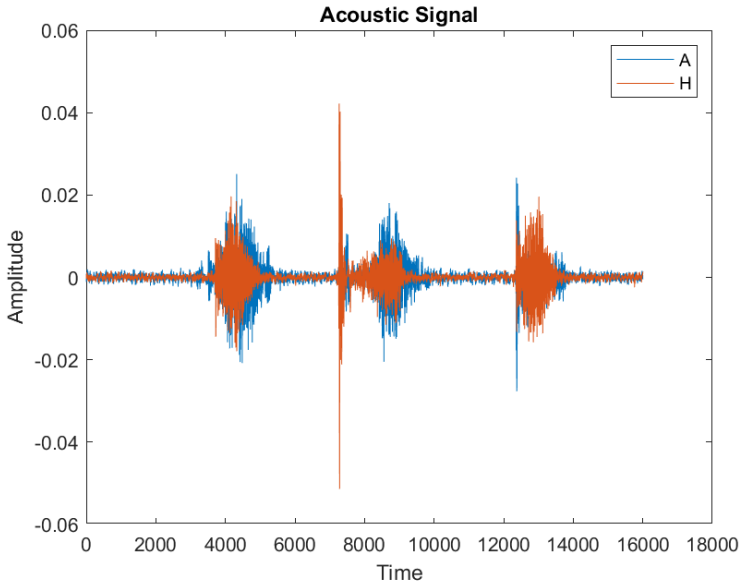


Figure 2.1: Acoustic signal of the letter A and H written on surface

To extract even more features, techniques such as Fast Fourier Transform can be used. Fourier Transform is the act of decomposing a signal into its constituent frequencies and gives us information of the magnitude of each frequency present in the signal [8]. However, even with FFT the image for each of the of the letters are still very similar. Moreover, there could be signals that are composed of similar frequencies and magnitude but the time when those same frequencies occur will be different. Therefore making this method unreliable.

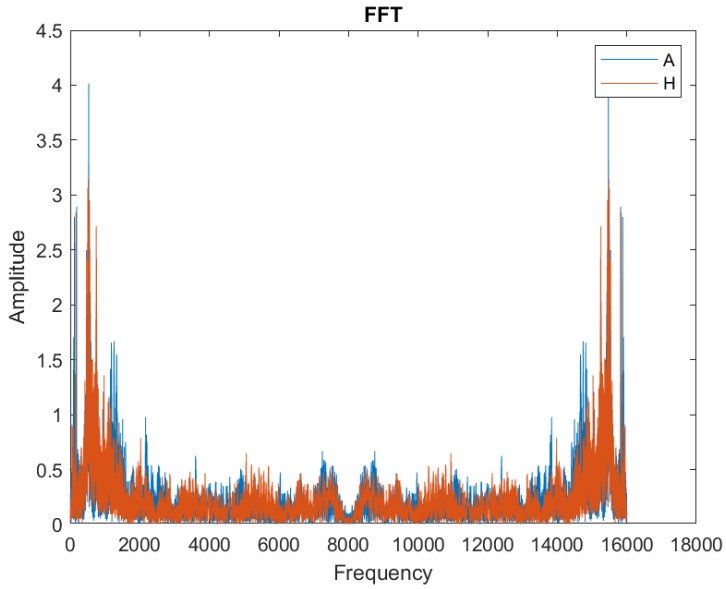


Figure 2.2: FFT of A and H acoustic signal

However, spectrogram is a great visual way to represent the signal strength of a signal over time at various frequencies. With the graph displaying 3 types of information, time on the horizontal axis, frequency on the vertical axis and amplitude represented by the brightness. With this information even with similar sounding acoustic signals, the spectrograms for them will have identifiable features as seen in the graphs below.

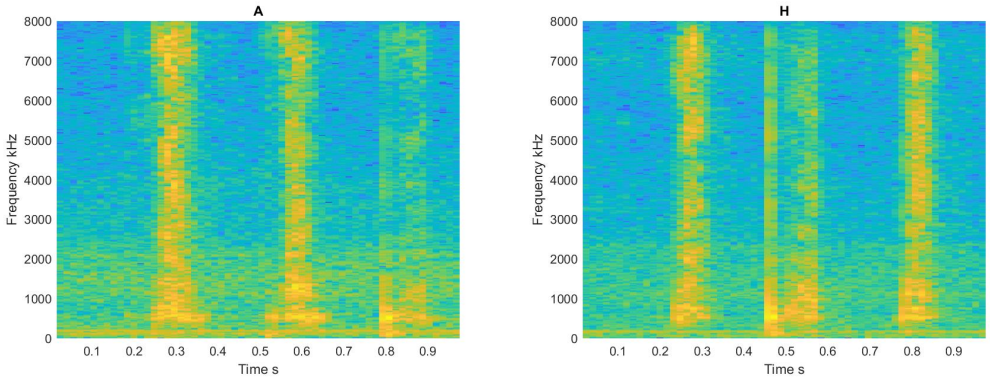


Figure 2.3: Spectrogram of A(left) and H(right) signal

2.2 Convolutional Neural Network

Convolutional neural network (CNN) has become a prominent neural network ever since Alex Krizhevsky won ImageNet competition in 2012 by reducing the classification error from 26% to 15% [12]. It is a deep neural network that uses multiple layer perceptrons to classify data, consisting of an input layer, an output layer and hidden layers in between. There are 3 basic ideas to CNN: local receptive fields, shared weights and pooling [14].

2.2.1 Local Receptive Fields

In CNN, the image input will have a width and height, and each pixel in that image will be considered an input neuron in the network. Those neurons will have values corresponding to the pixel intensity of the image. But instead of connecting each individual neuron to every neuron to the hidden layer, each hidden neuron will be connected to a small region of input neurons. The value of the connect neuron would be the element wise multiplication of the neurons in the smaller region.

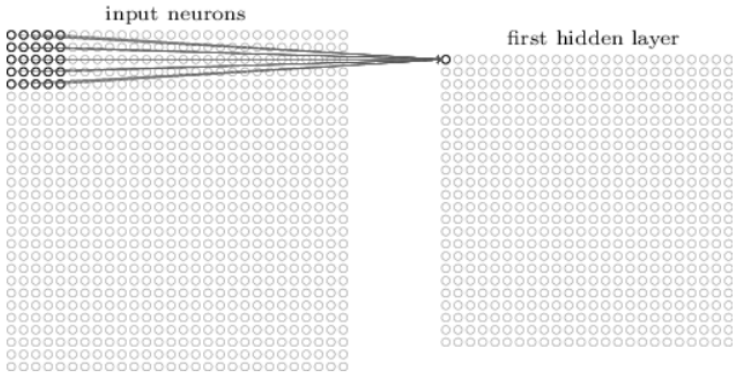


Figure 2.4: visualisation of 5x5 local receptive field on input image as first neuron in first hidden layer [14]

These smaller region on the input neurons are called the local receptive fields and thus the connected neuron in the hidden layer is to analyse this local field. To make up the next neuron in the hidden layer, the local receptive field gets slide to the right by one or more neurons, and those neurons in the local receptive field is connected to the next hidden neuron, this process is also known as convolving.

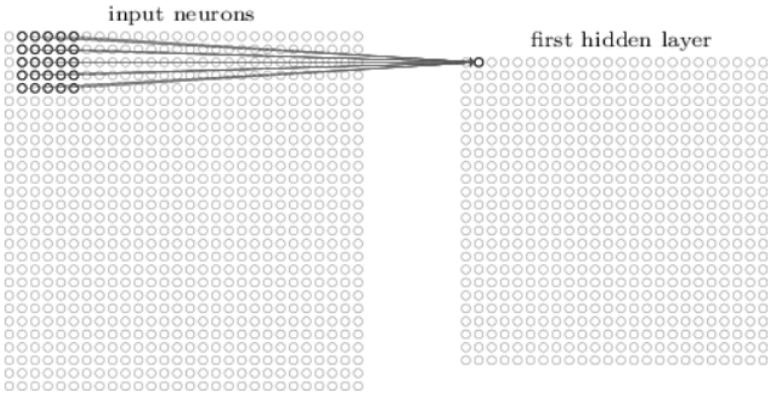


Figure 2.5: sliding the receptive field by one neuron to connect to next neuron in first hidden layer [14]

Repeat this process until each neuron in the input gets connected to the next hidden layer.

2.2.2 Shared weights and biases

Each neuron in the hidden layers will also learn a weight and overall bias and these weights and biases will be applied to all the other neurons in the hidden layer. Effectively becoming a feature map. The weight that defines this feature map is also known as the shared weight

and the bias is also known as the shared bias. This means that all the neurons in that hidden layer will detect the same feature at different locations of the input image, making CNN robust against features that don't appear at the same place constantly. Therefore, a full convolutional layer must consist of more than one feature map in order to detect multiple features on the input image.

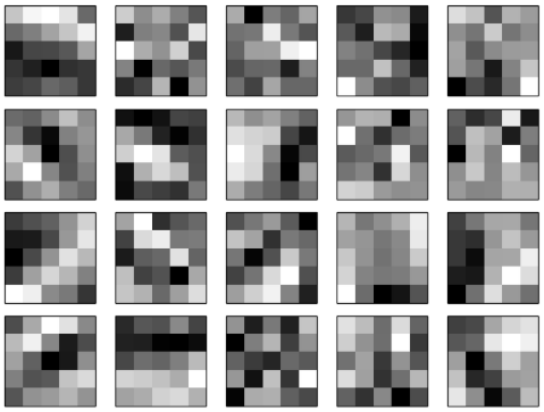


Figure 2.6: 20 images corresponding to 20 feature maps [14]

On each feature map, the weights of each pixel correspond to the brightness of the block, light colour means less weight whilst darker colour means larger weights. If any part of the image has similar feature to one of the feature maps above, that feature map will activate which tells the network that the image has this feature.

The advantages of sharing weights and biases is to reduce the number of parameters involved. For example, if an input of 28 x 28 neuron is to fully connect to 30 hidden neurons, there will be 28 x 28 = 784 total input neurons. Which will then connect to 30 weights and biases meaning 784 x 30 = 23,550 total parameters. Compared to the feature maps illustrated above, each map will have 25 shared weights and 1 bias making it 26 parameters for each map, with 20 maps it means there's a total of 520 parameters. Resulting in faster computational time.

The way the network trains and adjust the weights of each feature is through the process of back propagation. Back propagation allows the neural network to learn through identifying which weights contributed to the most when it got a correct classification and updates the weights correspondingly to reduce the percentage loss.

2.2.3 Pooling Layer

Pooling layers simplify the information output from the convolutional layers. Each feature mapped in the convolutional layer will give an output of the activations. The pooling layer will summarise a small region of neurons in the previous layer, usually 2x2. The most common pooling is Max Pooling, which takes the maximum activation of the region and that represents the activation for that small region and thus gives us a condensed map of features activated across the feature map. This is applied to all the feature maps separately, giving us a condensed output of the convolutional layer.

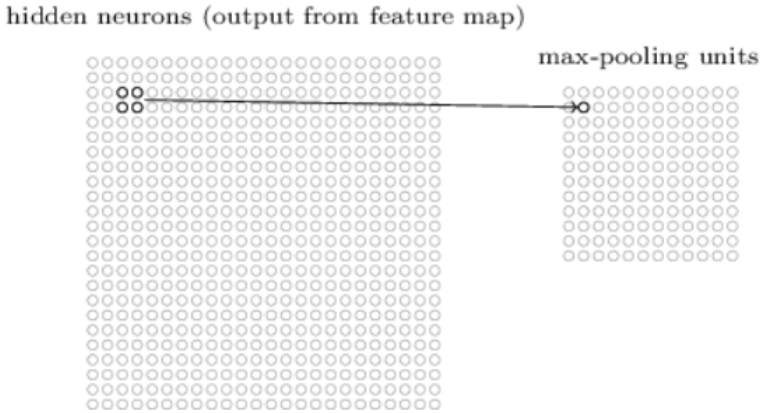


Figure 2.7: 2x2 max pooling layer on a small region of hidden layer [14]

The final layer of the network is a fully connected layer which connects to every neuron from the max-pooled layer to one or more output neurons. The output neurons are the ones which represents the probability of the image input to be a certain class, and the input is therefore classified with the output that has the highest probability.

2.3 K-Nearest Neighbour

K-Nearest Neighbour is a non-parametric classification algorithm that classifies an unknown instance by putting its vector features as a point in the feature space[11]. It then calculates the distance between that point and other points in the training data to find its K closest neighbour. It uses feature similarity to classify any new data and labelling it with the class most common amongst its K neighbours [7]. Non-parametric implies that KNN classifies its data only with the given data set and because it does not do any training, KNN will not generalise the data set unlike a neural network.

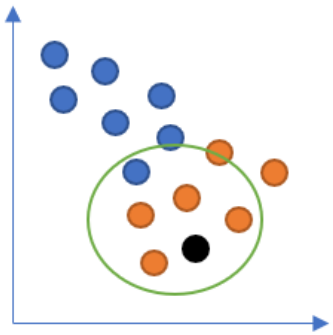


Figure 2.8: KNN latent space with K = 5

For example, in the latent space above, if we are the classify which class the black sample is to either a blue or orange with K = 5. Within the five nearest data to the black sample, four are classed as orange and only one is classed as blue which means that black has features similar to the orange class. Thus it can be deduced that the unknown sample is also an orange. The most widely used distance function to calculate the distance between two points is the Euclidean distance. The equation to calculate the distance is as follows

$$distance(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2}$$

Figure 2.9: Euclidean distance equation

However, the disadvantages of KNN is high computational power required as it stores the entirety of the training data set and the computational time also scales with the amount of training data it has.

2.4 Other Related Works

2.4.1 LLAP

LLAP is a device-free gesture tracker using acoustic phase to track the movements of a hand or finger [8]. LLAP emits a high constant ultrasound signal at a fixed frequency and measure the phase change of the acoustic signal reflected by the user's hand to track their movement. The system can recognise cursive writing using this method which can be something to investigate further in the future.

However, the disadvantages of LLAP is it can't differentiate the user's finger from their hands when detecting the reflected signal. Hence two fingers gestures and commands such as pinch or zoom are not supported, limiting the number of gestures and commands to one

finger. Furthermore, constant emission of ultrasound will introduce risks of noise pollution to the user's pets who can hear the sound, and long-term exposure can even make users uncomfortable [10].

2.4.2 WiSee

Wisee proposed a gesture tracking system using wireless signals (Wi-Fi) and the Doppler effect [11]. It utilizes the Doppler shifts generated from the user's gestures. For example, waving a hand closer to the receiver will create and positive shift and moving away may create negative shifts. It also takes advantage of wireless signals able to traverse through solid object such as walls, WiSee can provide users with a whole-home gesture recognition system, able to gesture to the device without being in the line of sight.

The limitation of WiSee is the requirement of setting up and purchasing additional hardware, such as the antenna receiver and transmitter. Also, the repetitive gesture from the user to prevent false positives may induce fatigue to the user.

2.4.3 IPanel

While other systems mentioned above emits a fixed frequency in order to track user's gestures and movements. IPanel is a system which only relies on the acoustic signals generated by a finger touching the surface [6]. Allowing users to input gestures and alphabets by having the user write on the surface next. Due to the inconsistent nature of the acoustic signal generated from the movement, IPanel extracts unique features of the signal and turns them into images and employ CNN to recognize the gesture input.

IPanel detects user input with what refers to as the 'touch peak', which is the burst of energy when the fingers first touch the surface. That indicates to the system the start of the gesture and ends with the 'end peak' which is when the finger leaves the surface. Therefore, this approach struggles with alphabets that requires 2 strokes such as 'x', 'i' and 'j' with the accuracy decreasing to 85%.

Chapter 3

Methodology and Implementation

3.1 Software

3.1.1 Matlab R2019b

Both CNN and KNN will be implemented on Matlab programming environment using Matlab's deep learning toolbox. This toolbox provides the necessary frameworks to implement CNN and KNN with relative ease. As well as other helpful built-in functions necessary to analyse and complete the implementation.

3.1.2 Wo Mic

Wo mic is a software that allows the user to link a smartphone's microphone to a computer. This is used so that when we are collecting audio signals for the dataset and when testing the implementation in real time, the audio signals are the signal that is authentically next to the smartphone.

3.2 Hardware

All the Matlab code implementations are developed and tested on an ASUS FX553V with an Intel core i7 processor and a NVIDIA GeForce GTX 1050 GPU. The GPU is a great asset which helps train neural network models significantly faster. For acoustic signal collection, an Honor view20 smartphone is connected to the laptop via Wo mic software to utilise the embedded microphone to obtain authentic data for training.

3.3 Data Collection

The dataset for this implementation consists of roughly 1000 one second audio clips of each capitalised letter of the English alphabet and clips of background noise taken from Google's speech command dataset that's publicly available [2]. The capitalised clips are recorded using Matlab's *audiorecorder()* function and setting the microphone to be wo mic (smartphone's microphone). It will then collect 60 seconds of continuous audio signals generated from the finger writing the same letter next to the smartphone. It will then go through the audio buffer and extract one second clips of the stroke audio signal and save it. In total there are around 26,000 audio clips, and a split of 8:1:1 ratio of train, test and validation respectively. The split is done using the Matlab's *splitData()* function [1]. As a 10-20% split proved to have to best accuracy with the least amount of diminishing returns [5]. By splitting the data and test sets, it allows for cross validation during the training of the network which gives a representation if the network.

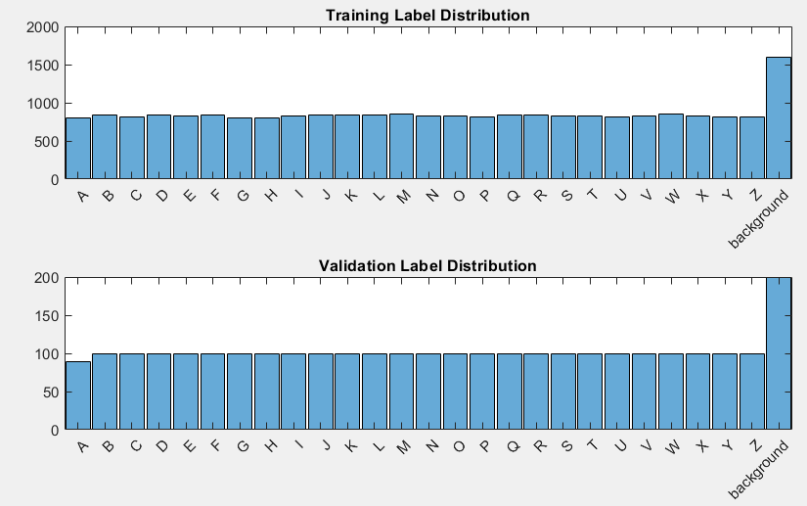


Figure 3.1: histogram of each letter and background noise

3.4 Image Augmentation

Neural networks that are trained on small data sets populated with similar data are prone to overfitting. Overfitting is the action of training the neural network to correspond too precisely to the training data and therefore will fail to fail to predict new data if it's not very similar as the training data. By randomly augmenting the training images in small data sets, the data set will now have slight differences and train the neural network with more variance. Effectively increasing the size of the data set. Common augmentations seen are rotating the images, scaling images along the x or y axis or translate the parts

of the image. For this implementation, each training spectrogram image is augmented by Matlab's **augementedDataStore()** function to randomly translate parts of the spectrogram by 100 milliseconds and scale the spectrogram along the x axis by 30%.

3.5 Audio to Spectrogram

The audio data collected and the background noise is converted to spectrogram images through the *speechSpectrogram()* and *backgroundSpectrograms()* function provided by Matlab [1]. The function takes in the following parameters: 1) datastore of audio clips 2) duration of the clips 3) duration of each spectrogram 4) time shift between each spectrogram frame 5) frequency bands. Within the function, it goes through each audio clip and applies the *spectrogram()* function to convert the audio signals into spectrograms with the specified parameters above.

3.6 Convolution Neural Network

3.6.1 Rectified Linear Unit (ReLU) Layers

ReLU is the activation layer that follows each convolutional layer to introduce nonlinearity to linear computation (element wise multiplication and summations) of the layer's output. ReLU is simple and helps backpropagation correct the weights and maintain fewer parameters which helps the model train and run faster. ReLU activation function states that any positive values will retain its value whilst all values lower than one will become 0.

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}.$$

Figure 3.2: ReLU equation

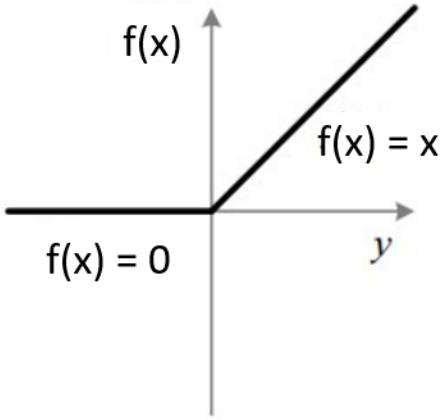


Figure 3.3: ReLU graph

However, one disadvantage of ReLU is if a neuron outputs a zero, there could be an update that causes that neuron to never activate again, resulting in dead neurons. Leaky ReLU attempts to solve this by multiplying negative values with a scale, scaling down the negative values and ensuring that there is at least some output.

The validation accuracy for both ReLU and leaky ReLU in the implementation varies slightly, with both activation function giving a final validation accuracy of around 95-98%. Hence the implementation the leaky ReLU layer is chosen in case of potential dead neurons.

$$f(x) = \begin{cases} x, & x \geq 0 \\ scale * x, & x < 0 \end{cases}$$

Figure 3.4: Leaky ReLU equation

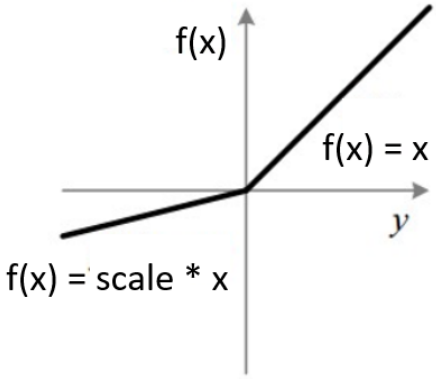


Figure 3.5: Leaky ReLU graph

3.6.2 Hyper Parameters Tuning

Learning Rate

The learning rate is a hyper parameter which controls the weight adjustments of the network with respect to the loss gradient calculated using the stochastic gradient descent algorithm. If the learning rate is too small, then it takes the model longer to reach the best accuracy by arriving at the local minima. However, if the learning rate is too high, it may overshoot the minima and may even fail to converge towards the local minima.

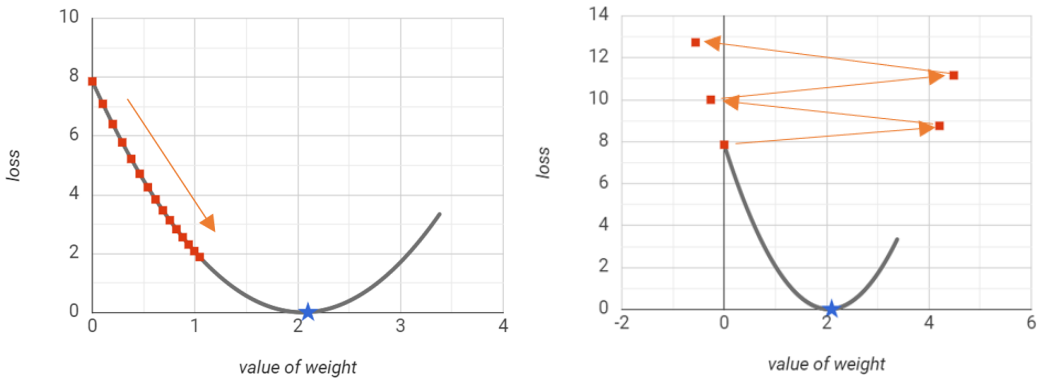


Figure 3.6: Left: Small learning rate **Right:** High learning rate

It has been shown that a learning rate within the range of less than 1 and greater than 1 x 10-6 and the value 0.01 to be the default value [6].

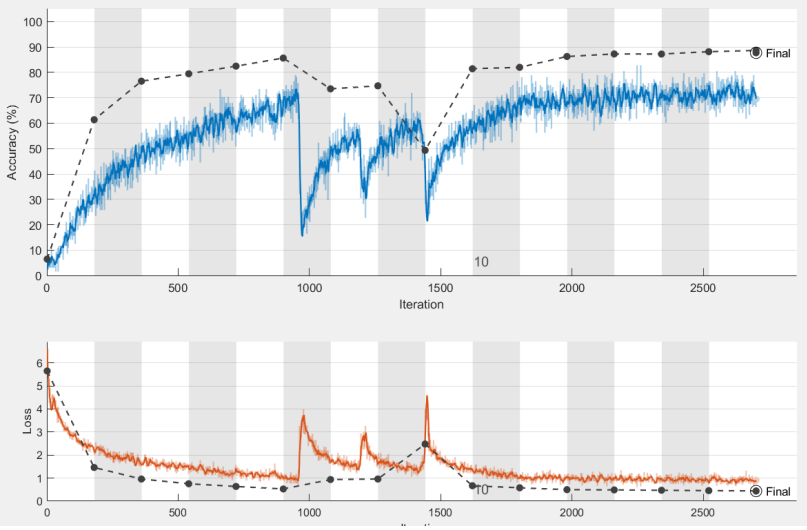


Figure 3.7: Loss rate with 0.01 learning rate

For this implementation, the values 0.01 and 0.001 have been tested. The learning rate value of 0.01 have certain areas where the loss spikes up. This suggests that it has overshoot the local minima and that the learning rate is too high.

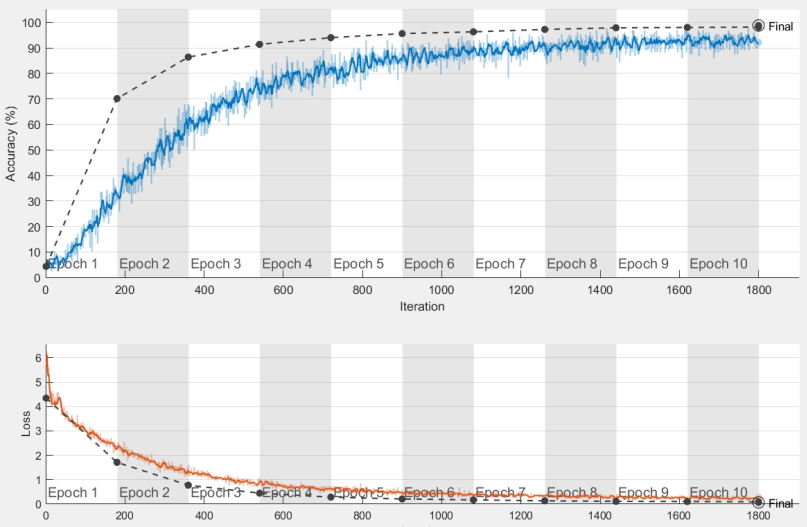


Figure 3.8: Loss rate with 0.001 learning rate

Compared to the learning rate of 0.001 which has a smooth downward slope, converging towards the local minima. Therefore, the learning rate of 0.001 has been chosen.

Number of Hidden Layers

Number of Hidden Layers	Validation Accuracy	Single-image prediction time
3	98.89%	4.1155ms
4	99.28%	4.4401ms
5	99.50%	4.8651ms

Table 3.1: Validation accuracy with number of hidden layers

The number of hidden layers in between the input and output layer determines the amount of feature abstractions it can extract from the input as it propagate through the network. An experiment has been conducted to see if an increase in layers will increase the validation accuracy with 3, 4, and 5 layers. As seen in table 3.1 the increase in convolutional layers only increases the accuracy slightly and with a payoff of a increase of prediction time. Therefore to save computational load, the implemented CNN will only have 3 hidden layers.

Dropout Probability

Dropout is a technique which temporarily remove random sets of activations in that layer by setting them to zero. This is to ensure that complex co-adaptation doesn't happen. As neurons are updating to reduce the overall loss value, some neurons may compensate for other neurons mistake, making those neurons rely on others and thus leads to complex co-adaptation. Which may then lead to overfitting as co-adaptation does not generalise to unseen data. By dropping some activations, a neruon can't rely on others to correct its mistake and will learn to perform well in different contexts [18].

It was recommended and observed in the implementation that 0.5 is the best drop out probability for most neural networks, as going too high has caused a massive drop in accuracy and a lower value runs the risk of overfitting [18].

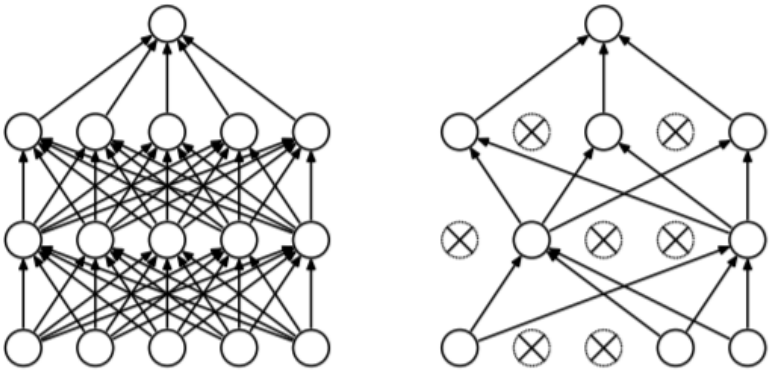


Figure 3.9: Left: Normal neural net. **Right:** Neural net with dropout [18]

Epochs and Mini-batch Size

The batch size refers to the number of samples propagated through the network before the model is updated, this could range to the entirety of the data set. However, with large data sets used in neural networks, it is inefficient and time consuming to sample the entire data set, hence the mini-batch size which only takes random subsets of the data set during one iteration and average their accuracy. Epoch refers to the number times the entire data set has propagate through the network so the higher the number of epochs, the more the neural network will be trained with entire data set.

The effects of these hyper parameters seem to only affect computational and training time [6]. Therefore, to find the optimal number of epochs and mini-batch size we monitor the training accuracy progression.

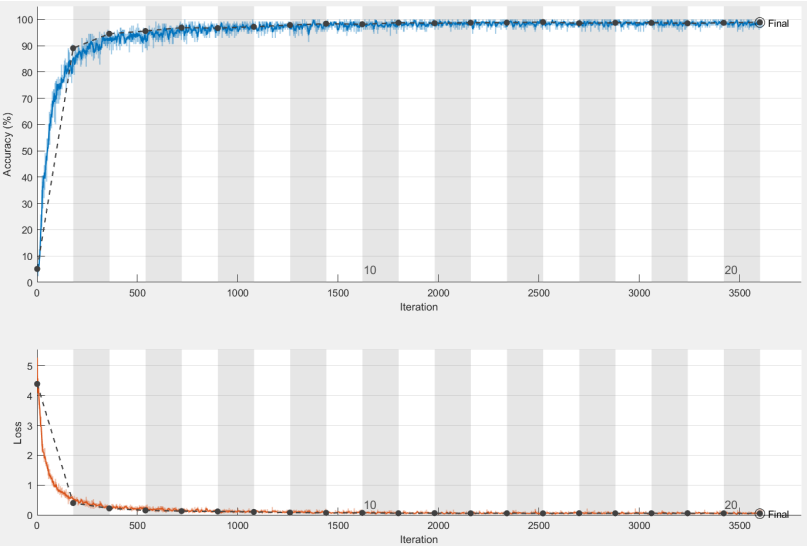


Figure 3.10: T
raining progression with 20 epochs and 128 mini-batch size

With 20 epochs, we can see that the neural network has peaked its accuracy after around 10 epochs, anymore training is redundant and have the risk of over fitting. Therefore, an epoch 10 and mini-batch size of 64 is implemented.

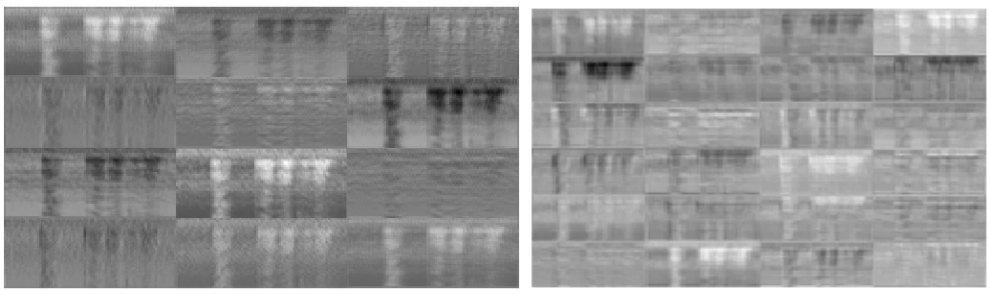
CNN layer architecture

layer	name	configuration
	Image Input	40x98x1 images with 'zerocenter' normalization
	Convolution	12 3x3x1 convolutions with stride [1 1] and padding 'same'
1	Batch Normalization	Batch normalization with 12 channels
	Leaky ReLu	Leaky ReLU with scale 0.01
	Max Pooling	3x3 max pooling with stride [2 2] and padding 'same'
	Convolution	24 3x3x12 convolutions with stride [1 1] and padding 'same
2	Batch Normalization	Batch normalization with 24 channels
	Leaky ReLU	Leaky ReLU with scale 0.01
	Max Pooling	3x3 max pooling with stride [2 2] and padding 'same'
	Convolution	48 3x3x24 convolutions with stride [1 1] and padding 'same'
3	Batch Normalization	Batch normalization with 48 channels
	Leaky ReLU	Leaky ReLU with scale 0.01
	Max Pooling	1x13 max pooling with stride [1 1] and padding [0 0 0 0]
	Dropout	50% dropout
4	Fully Connected	27 fully connected layer
	Softmax	softmax
	Classification Output	Weighted cross entropy

Table 3.2: CNN Layer Structre

The implemented CNN has has a total of 4 layers with the structure as seen in table 3.2 with three hidden layers as mentioned in section 3.6.2. The parameters of these layers are based to that in reference [1]. The first three layers are used to extract features and characteristics of the input image whilst the last layer is used for classification. The sizes for the convolutional layers all have the same size of 3x3. Research has found out that having the same size for each layer could yield better or the same result than layers that increases or decreases through the network [13].

An example of feature maps for each of the three convolutional layers can be seen below to see what features activates the neurons for each convolutional layer.



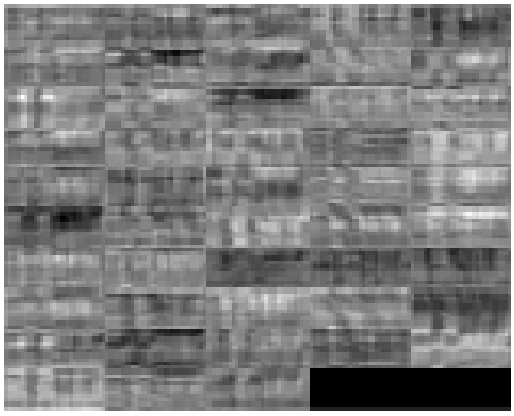


Figure 3.11: Feature map for conv_1 (**Top left**), conv_2 (**Top right**) and conv_3 (**Middle**)

CNN system architecture

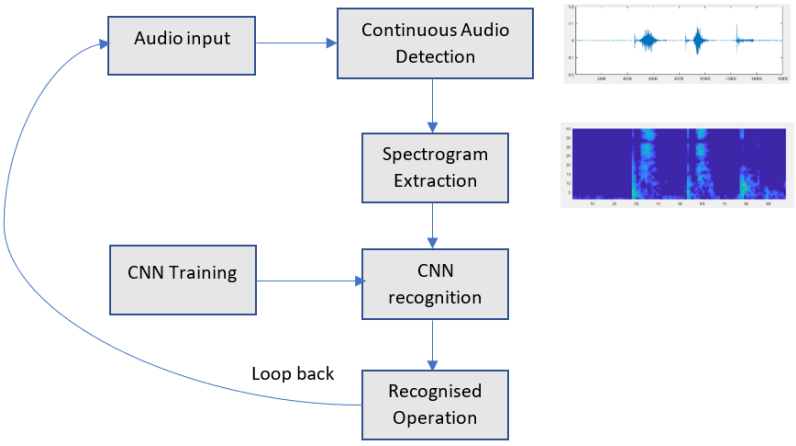


Figure 3.12: Implemented CNN system architecture

The real time implemented CNN system architecture flow is highlighted in the figure 3.12. The CNN will first be trained using the hyper parameters mentioned in section 3.6.1 and 3.6.2. When the model is ready, the system listens for a continuous acoustic signal through the smartphone's microphone. The acoustic signal then gets placed into a buffer and is turned in spectrogram using the same methods as mentioned in section 3.5. The spectrogram is then placed into the CNN classify method which predicts a label and the label is placed into a label buffer. The label buffer is then tested with the following threshold operations: 1) The mode of the label buffer is not classified as 'background' 2) If the mode label appears frequent enough pass a threshold 3) If the probability of the of the classification is high enough. If one of these thresholds is met, then output the the classification label.

3.7 K-Nearest Neighbour

3.7.1 Feature Vector

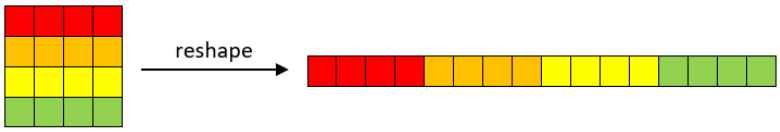


Figure 3.13: Reshaping spectrogram values

As mentioned in section 2.3, KNN classifies an unknown instance by placing its vector features as a point. Vector features in KNN are in the shape of a 1xN matrix where N is the number of vector features. In order to fit the spectrogram features in this suitable format, the implemenation uses Matlab's *reshape()* function to reshape the 40x98 matrix to a 1x3920 matrix where each pixel value of the spectrogram will be a feature vector. The order of the pixels will as seen in figure 3.13 where each row of the matrix is appended one after the other.

3.7.2 Optimum K value

Different K values have been tested on the implementation to find the K value which gives the least resubstitution loss and cross-validation loss. Using the Matlab function *resubLoss()* to find the misclassifications from the training data. However, these results may not be accurate as it evaluates the accuracy using the same data as the training data. To overcome this problem we also used Matlab's *kfoldloss()* function to cross validate and predict the accuracy of classifying unknown data by spliting the trainig sets into different groups. In the testing the split is 9:1 for testing to validation sets.

K Value	Resubstitution Loss	Cross-validation loss
1	0	1.58%
2	0.64%	1.86%
3	0.86%	1.82%
4	1.22%	1.92%
5	1.30%	1.56%
6	1.48%	2.11%
7	1.53%	2.11%
8	1.70%	2.28%
9	1.77%	2.41%
10	1.85%	2.39%

Table 3.3: Resubstutution and cross-validation loss on different K values

As seen in table 3.3, it seems that the higher the K value, the more inaccurate the model becomes. This is a strong indication that the model is overfitting to the data. However with the cross-validation, value there is a the K value with least accuracy loss is 5 which will be the K value use in the implementation.

Chapter 4

Testing and Evaluation

The testing phase of this project comprises of testing the models in real time and comparing the result to the validation accuracy. Each letter is written 100 times next to smartphone and the each successful classification is recorded to find the real time accuracy.

4.0.1 CNN

Validation Accuracy

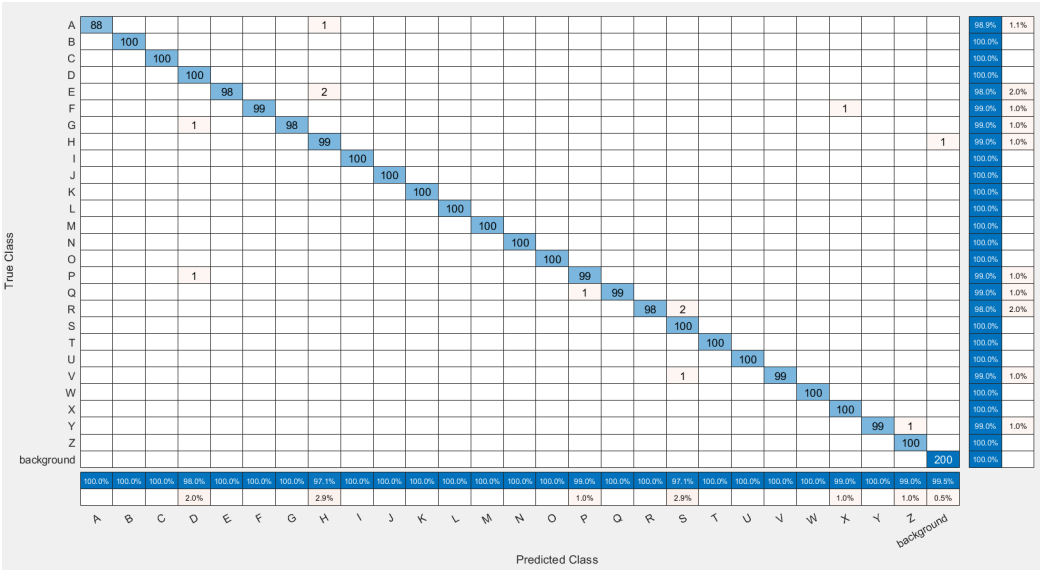


Figure 4.1: Confusion Matrix of CNN

The confusion matrix in figure 4.1 shows that the neural network achieved a high number of accuracy and be able to predict most classes. Reaching a final accuracy of 99.57% and training accuracy of 99.77%.

Real Time Accuracy

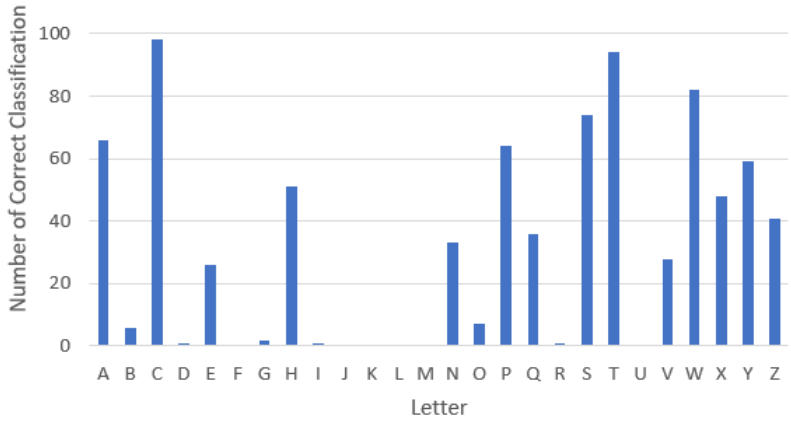


Figure 4.2: Number of correct classification per letter

The real time accuracy was significantly different from the validation accuracy. As seen in figure 4.2, out of 2,600 letters, only 818 total strokes are correctly classified. Resulting in an accuracy of only 31.46%, a drastic decrease from 99.57% validation accuracy. The model can consistently classify some letters such as 'C' or 'T' however the majority of the letters are classified incorrectly.

4.0.2 KNN

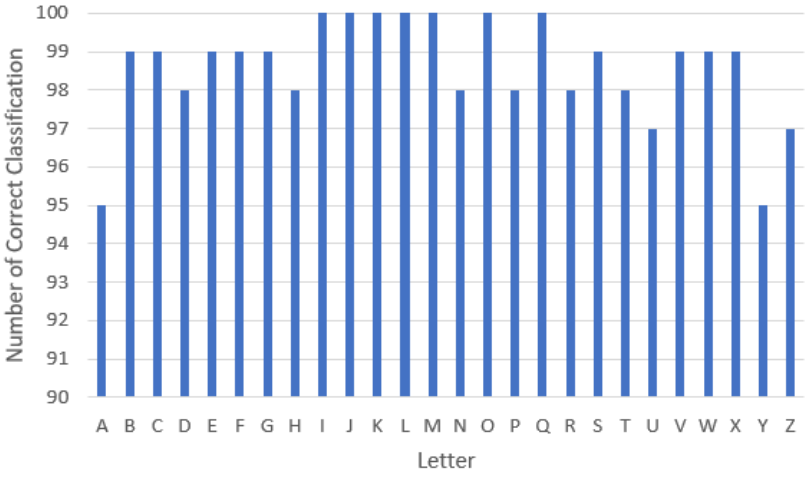


Figure 4.3: Number of correct classification per letter

KNN is too slow to be classified at real-time therefore a new set of acoustic signal of 100 clips per letter is used to test its accuracy. As seen in figure 4.3 the model is able

to correctly classify most of the letters with an accuracy of 98.57%. This is accurately predicted in section 3.7.2 where the cross-validation loss of when K = 5 is 1.56%.

4.0.3 Evaluation

Even though the accuracy of KNN is really high, the computational time it requires is a huge disadvantage and making it not suitable for smartphone. Furthermore with the current small variance of the data set, and the test results in section 3.7.2 suggests that the vector features may have overfitting. Future work can be tested through using acoustic data that is recorded on other surfaces to confirm this suspicion.

For CNN, the massive decrease of 68% accuracy from the validation accuracy heavily suggests that there is overfitting happening in this CNN model. Steps has been taken as an attempt to prevent this situation in section 3.6 by the adjusting the hyper parameters and the augmentation of the data set to increase variance. The inconsistency of accurately identifying some letters heavily suggests that the model is not trained with enough data per stroke. For example the letters ‘M’ and ‘W’ have very similar acoustic signals therefore when writing the letter ‘M’ during testing, the model almost always classifies it as a ‘W’. This means to more data sets of each letter will be required into order increase the models real time accuracy and as a result the readjustments of the hyper parameters as well.

Chapter 5

Conclusion

To conclude the project report, we will evaluate the aims that the system is suppose to achieve, the potential impacts that this project may bring as well as discussing future related to device-free finger motion tracking.

5.0.1 Review Project Aims

1. **Build a system that will be able to recognize user's input of the 26 letters of the English alphabet with high accuracy**
   This aim has been partially achieved with KNN able to classify most of the letters with a high accuracy as described in chapter 4. However the CNN model in real-time is not able to classify the acoustic signals accurately and only achieved an accuracy of 31%.

2. **Explore 2 different classification methods between instance-based learning (KNN) and deep neural networks (CNN)**
   This aim has been achieved with chapter 3 and 4 highlighting the different approaches these models use to classify data as well as the limitations of both models.

3. **Able to differentiate the difference between a letter acoustic signal and background acoustic signals**
   This aim has been met as the models are trained to identify background noise and CNN has threshold checks that only displays the classification result when the majority of the label buffer is a letter or is confidant in its letter classification.

4. **Doesn't require the user to set up additional hardware or wear external devices for the system to function**
   This aim has been met as the only actions required by the users to use the system is

to have a smartphone with a microphone and write the letters using their fingers next to the device.

5. **The user may use the system on a variety of surfaces and still retain a high recognition accuracy**
   This aim has not been met as the models could not accurately identify the letters on a variety of surfaces. This is due to the current data set not having a high enough variance for CNN to generalise enough to classify acoustic signals on different surfaces.

6. **(Optional) Port the system into a smartphone instead of classifying on the laptop**
   This aim has not been achieved due to the time constraints on the project due to data collection and development of the neural network model took longer than expected. Therefore this paper is to explore a proof of concept if acoustic signals are a viable way to user track finger motions and future development of porting the system to a smartphone can be achieved.

5.0.2 Potential Impacts

Allowing users to have more ways to interact with their smartphones beyond the screen has many potentials. As mentioned, older users may have trouble typing on the keypads, therefore an intuitive writing detection free from the restriction of the screen size can help get over this barrier and allow them communicate effectively. Furthermore, I believe that when users have more interactive area with their smartphone, it can expand the control possibilities of applications. For example in gaming where the controls are either oversimplified or clustered over the screen, this system will be able to move some controls beyond the screen to deliver a smoother user control experience.

5.0.3 Future Works

This project can be further extended by first completing the failed aims mentioned above. To potentially solve the current overfitting problem and achieve a high accuracy model, gathering more variance and quantity of acoustic signals for each letter would be needed. An increase in the variance of the data could be having different signals that is generated from a variety of surfaces. This in turn will train the model to recognise letters in a variety of surfaces. The amount of gestures can also be built upon to include simple directional gestures such as pinch and zoom or numbers (0-9). Further development could be explored by fully implementing the system onto a smartphone using Matlab's Simulink blocks or Android studio.

5.0.4 Closing Remarks

The initial implementation of the project was considerably more ambitious than what was achieved here. However, due to time constraints and oversight on the amount of data required to trained the models. Down sizing the number of initial stroke detection of simple gestures, 10 numbers and 26 alphabets to only classifying the 26 alphabets. Overall, despite the result being different from expectation because of an overfitting problem, the project is ultimately fulfilling to work on and I have gained valuable insight and understanding of CNN and KNN. I have learnt that despite the great reputation CNN has in image recognition and the ways of adjusting its hyper parameters to optimize the model. The accuracy of the results depends heavily the quantity and quality of the data set which is harder to collect than expected. However, I believe there a highly accurate device-free finger motion tracking system is achievable with future development.

Bibliography

[1] Speech command recognition using deep learning. https://uk.mathworks.com/help/deeplearning/examples/deep-learning-speech-recognition.html;jsessionid=89b4b5450f6438d5a98e3a910512.

[2] Launching the speech commands dataset. https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html, Aug 2017.

[3] Samsung galaxy fold 5g: World's first foldable phone. https://www.samsung.com/uk/smartphones/galaxy-fold/, Jan 2020.

[4] BARREDO, A comprehensive look at smartphone screen size statistics and trends. https://medium.com/@somospostpc/a-comprehensive-look-at-smartphone-screen-size-statistics-and-trends-e61d77001ebe, May 2014.

[5] BARRY-STRAUME, J., TSCHANNEN, A., ENGELS, D. W., AND FINE, E. An evaluation of training size impact on validation accuracy for optimized convolutional neural networks. *SMU Data Science Review 1*, 4 (2018), 12.

[6] BENGIO, Y. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[7] BRONSHTEIN, A. A quick introduction to k-nearest neighbors algorithm. https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7, May 2019.

[8] CHAUDHARY, K. Understanding audio data, fourier transform, fft, spectrogram and speech recognition. https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520, Mar 2020.

[9] CHEN, M., YANG, P., XIONG, J., ZHANG, M., LEE, Y., XIANG, C., AND TIAN, C. Your table can be an input panel: Acoustic-based device-free interaction recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 3*, 1 (2019), 1–21.

[10] DU, H., LI, P., ZHOU, H., GONG, W., LUO, G., AND YANG, P. Wordrecorder: Accurate acoustic-based handwriting recognition using deep learning. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (2018), IEEE, pp. 1448–1456.

[11] HU, L.-Y., HUANG, M.-W., KE, S.-W., AND TSAI, C.-F. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus 5*, 1 (2016), 1–9.

[12] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[13] LAROCHELLE, H., BENGIO, Y., LOURADOUR, J., AND LAMBLIN, P. Exploring strategies for training deep neural networks. *Journal of machine learning research 10*, Jan (2009), 1–40.

[14] NIELSEN, AND A., M. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/chap6.html, Jan 1970.

[15] O'DEA, S. Smartphone adoption in the united kingdom 2011-2018. https://www.statista.com/statistics/271460/smartphone-adoption-in-the-united-kingdom-uk/, Oct 2019.

[16] O'DEA, S. Smartphone usage by age uk 2012-2018. https://www.statista.com/statistics/300402/smartphone-usage-in-the-uk-by-age/, Oct 2019.

[17] O'DEA, S. Us: folding phone retail prices 2020. https://www.statista.com/statistics/1103866/folding-phone-retail-prices/, Mar 2020.

[18] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*, 1 (2014), 1929–1958.

[19] WANG, W., LIU, A. X., AND SUN, K. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking* (2016), pp. 82–94.

Appendix A

Project Proposal

Can I interact with my smartphone on any surface?

Yi-Xin Huang

Abstract:

This project is to explore the potential of extending the interaction of a smartphone beyond its screen. This project will enable users to write letters and convey simple gestures to the devices without touching the surfaces of the device. This project has the potential to benefit people with disabilities, improve the quality of the user's life or enable users' access to bigger interactive surface without purchasing bigger devices.

The beginning of the project is to create a system that uses smartphone's microphone to record the acoustic signal generated by the user's input, turn the signal into a spectrogram and identify the spectrogram to the corresponding gesture or letter using deep neural network. Further stages include improving the accuracy of the system on different surfaces and to try improving the system to recognize beyond single letter to recognize complete words. Finally, we can see the impact of this system on selected users mentioned above.

Introduction:

Mobile devices had a history of having a small screen to gain portability and compactness for the user's ease of carrying the device. However, with the rise of smart devices especially smartphones, a new trend has risen as the screen of smartphones exponentially increases every model [1]. Smartphones can now do so much more than making phone calls such as gaming, surfing the web, and installing various applications. This cause consumers to now demand bigger screens to interact with their devices as shown by this trend.

Companies are now looking for ways to combine compactness with bigger interactive surfaces with the newest feature being foldable phones. Unfortunately, the only way for users to get bigger interactive surfaces is for them to purchase new phones every year, and with phone prices exponentially increasing with features such as foldable and 5G [2], not all users will be able to gain access to a bigger interactive surface.

Furthermore, users with disabilities such as Parkinson will require a bigger interactive surface to combat their issue and provide them with a better experience with their smartphone without the purchase of additional or special hardware. In addition, this system can be utilized to improve the experience for all users in certain situations such as their hands are too dirty to interact with the device or too wet to be recognized by the touch screen.

This project will explore the potential of a new interaction medium using finger input on a nearby surface based heavily on the system called IPanel[3]. The system should be able to extend the input of any existing commercially off the shelf (COTS) mobile device to nearby flat surfaces such as woodened tables, floor, paper, etc. to give users more freedom to interact with the device unrestricted by its screen. The system will support a wide range of gestures like flick right/left/up/down, scroll up/down as well as recognizing 10 numbers (0-9) and 26 letters of the English alphabet. The users will be able to input these gestures and letters through writing it individually (similarly to writing on paper) on the surface besides the device, extending the interaction in a natural and intuitive way.

As every input of the letters and gesture vary, therefore we must implement a deep neural network to accurately recognize the gestures and letters the user is writing down. Using the built-in microphone of the COTS mobile device, the system will observe the user's input through acoustic signals generated by the user's finger sliding across the surface and identified the signal with the corresponding gesture or letter using deep neural network.

Background:

Projects that are similar to this project includes a device-free gesture tracking using acoustic phase to track the movements of a hand or finger called LLAP [4]. LLAP emits a constant ultrasound signals at fixed frequency and measures the phase change of the acoustic signal reflected by the user's hand to track their movement. The system can recognize cursive writing using this method which can be something to investigate further when improving my proposed project.

However, the limitation of LLAP is it can't differentiate between the hands and the fingers when relying on the reflected signals. Hence two finger gestures such as pinch and zoom are not supported, and the environment must be quiet with little background to retain its accuracy. Furthermore, the constant emission of ultrasound signals will introduce a risk to noise pollution to pets who can hear it or uncomfortableness to the user due to long term exposure [5].

Other projects such as WiSee[6] proposed a gesture tracking method using wireless signals (Wi-Fi) and the doppler effect. Due to the nature of wireless signal able to traverse through walls, WiSee can provide users with a whole-home gesture recognition system, able to gesture to the device without the device being in their line of sight. WiSee utilizes the doppler shifts generated from user's different gestures e.g. waving a hand closer to the receiver will create a positive doppler shift.

The limitations of WiSee is the requirement of additional hardware, in this case, the antenna receiver and transmitter set up in the living room and the repetitive gesture from the user to prevent false positives which could be tiring for the user.

The Proposed Project

Aims and Objectives

The aim of this project is to create a system that can extend the user's interaction surface with the smartphone screen by recognizing simple gestures and letters that the user input through writing on a nearby surface. In order to achieve this the system must fulfil the following objectives;

- An Android application that provide feedback to the user's gesture and letter inputs.
- The system must be able to recognize the user's input of simple gestures, the English alphabet and numbers with high accuracy.
- Data collection of each gestures, letters and numbers for the neural network.
- Able to differentiate the difference between a gesture/letter acoustic signal and background acoustic signals.
- No additional hardware set up or requiring the user to wear any addition devices
- Able to use on any Android COTS device.
- The user may use the system on a variety of surfaces.

Methodology

To create this system, an android application will be created to provide feedback to the user inputs. The application will be coded using Java on Android Studio however the deep neural network will be coded using python with external libraries such as PyTorch.

To collect acoustic signals from the phone using its embedded microphone, I will be using the following API; AudioTrack() to capture audio and AudioRecord() to record the audio at the default rate of 44kHz.

After having the system continuously listen for acoustic signals, the system needs to identify which signals are user inputs and segment those out. An acoustic signal produced by a user will have a burst of audio signal when their finger hits the surface[3] which signals a start of user input as well as end peak when user finished their input. A code needs to be implemented where after a certain threshold of energy (start peak), it records the acoustic signal until the end peak.

When the system has the input acoustic signal, it will convert that signal into a spectrogram as every gesture will produce a unique spectrogram acoustic signal[3]. Using the spectrogram, the system can then use convolutional neural network(CNN) to identify the spectrogram to a corresponding gesture, letter or number by extracting its characteristics. We use CNN because its regularized multilayer perceptron is suited for image identification.

To create a ground-truth of Dataset I need to generate a dataset of gestures, letters and numbers possibly with the aid from volunteers. To make the system more robust, the data will be collected in different environments such as library, lecture theatre, café, labs, etc. In

addition, data on different surfaces can also be collected. The dataset is then used to train the CNN and a trained CNN is implemented into the application.

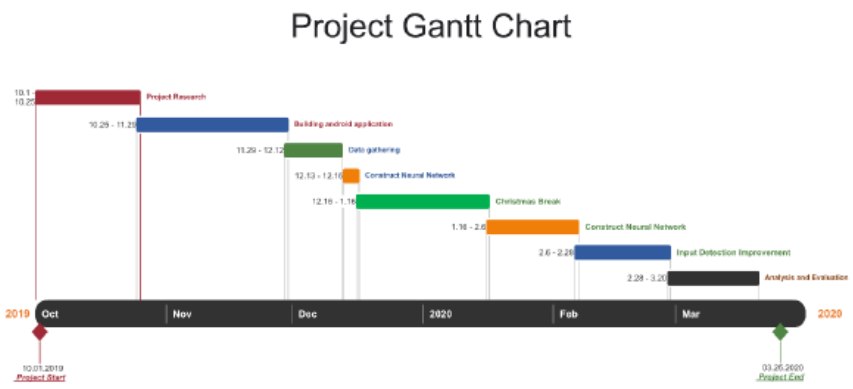
For testing and evaluation, a user will draw different letters/numbers and gestures and the application will output what it thinks the user has input. When the application has achieved a high enough level of accuracy it can be tested with real users and collect their feedback.

Programme of Work

The project will begin on October 2019 and end on March 2020 and will be broken into the following stages;

- Project research – this is to research and understand the basic concepts need to fulfil this project.
- Building the android application – this is to build the GUI and gain permission to record and store acoustic signals.
    - Making sure that the application can is continuously recording the acoustic signals.
    - Code the segmentation of the acoustic signal for gestures
    - Transform the acoustic signal segment into a spectrogram and saves it
- Data gathering – gather volunteers to help record the dataset for CNN training in different environment and surfaces. Each gesture and letters will be repeated 50 times in each environment
- Construct Neural Network - build a CNN for spectrogram identification. Train the CNN with the dataset that is gathered and find the optimal configuration for the model.
- Input Detection Improvement – try to improve the detection of the system by including cursive words instead of inputting one letter at a time.
- Analysis and Evaluation – Implement the optimized neural network into the android application. Undergo user testing to see its potential application of the system and viability.

The Gantt chart below shows the schedule of the project from 2019-2020:



Resource Required

Resources required are a working android mobile device to test the application during development, to ensure that the application is working when distributed to volunteers when collecting data for acoustic signals. It also tests if the application work on different models of android devices. Using Android studio to code the application. Libraries and APIs includes PyTourch, AudioTrack() and AudioRecord().

References

[1] https://medium.com/@somospostpc/a-comprehensive-look-at-smartphone-screen-size-statistics-and-trends-e61d77001ebe

[2] https://www.cnet.com/news/even-more-proof-that-your-iphone-and-android-are-more-expensive-in-2019/

[3] Mingshi Chen, Panlong Yang, Jie Xiong, Maotian Zhang, Youngki Lee, Chaocan Xiang, and Chang Tian. 2019. Your Table Can Be an Input Panel: Acoustic-based Device-Free Interaction Recognition. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 3, 1, Article 3 (March 2019), 21 pages.

[4] Wei Wang, Alex X Liu, and Ke Sun 2016. Device-free gesture tracking using acoustic signals. In proceedings of 22nd Annual international Conference on Mobile Computing and Networking. ACM, 82-94

[5] https://www.nhs.uk/news/medical-practice/calls-for-research-into-health-effects-of-ultrasound-exposure/

[6] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home gesture recognition using wireless signals. In Proceedings of the 19th annual international conference on Mobile computing & networking (MobiCom). ACM, 27-28