A Comparison of Different Models for the Classification of Spoken Digits – IT1244 Group 31

Zhang Zhenjie,[1] Ng Meng Jie,[2]
Lam Jian Yi Eugene,[3] Muhammad Abu Ubaidah[4]
National University of Singapore[1,2,3,4]
e0957630@u.nus.edu,[1] e0958357@u.nus.edu,[2] e0969945@u.nus.edu,[3] e0958070@u.nus.edu[4]

Introduction

Speech serves as one of the primary means of communication between humans. However, some individuals with disabilities may encounter difficulties interacting with others and their environment. The solution: speech recognition technologies. These technologies are implemented to provide them with an alternative means of communication with devices. For instance, for people with mobility impairments, voice recognition allows them to use computers without needing a keyboard or mouse, allowing them to live more independently (Gilbert 2022). Today, such systems have become an integral part of our daily lives as the use of voice assistants like Siri and Alexa becomes more widespread partly because they are easily accessible through smartphones and laptops (Hoy 2018).

However, one of the biggest challenges to the adoption of voice technology is not its accessibility, but its accuracy. Inter-speaker variability is one of the main factors contributing to the discrepancy between what is predicted and what is said (Huang et. al. 2001). As everyone has different speech patterns, pitches, and accents, it may be difficult for voice recognition technologies to extract the important aspects from an audio clip and generalise it for a wider audience.

With this challenge in mind, our team seeks to address it by implementing a Convolutional Neural Network (CNN) on the Spoken Digit dataset containing audio clips of spoken digits by six speakers. While CNNs were originally used for Computer Vision models, it has also been successful when implemented on speech signals, showing better performance than Artificial Neural Networks (ANN) (Palaz et al. 2015). To implement our model, we used the Short-time Fourier transform (STFT) to convert the audio clips into images of spectrograms. This is preferred over the Fast Fourier Transform (FTT) as it retains some time information, making the analysis more accurate (TensorFlow 2023). These images are then fed into the neural network to extract key patterns.

Then, we evaluate the performance of other supervised learning models to solve this problem, namely, K-Nearest Neighbours, Decision Trees, and Gaussian Naïve Bayes.

Dataset

In this project, the spoken digits dataset was used. It consists of 3000 recordings from 6 speakers pronouncing a digit from 0 to 9. The recordings were captured at a sample rate of 8 kHz. Each audio clip is labelled with the correct digit spoken in the clip via their file names.

Preprocessing of Data

After reviewing the audio clips, we noticed that the recordings are not standardised and vary between 0.2s to 1.2s. To address this inconsistency, we decided to set each audio clip to 0.5s. Therefore, to ensure consistent inputs into the CNN model, we chose a value of 4000 for the output sequence length, which is derived by multiplying the sampling rate with the audio duration (8000*0.5). The image is then resized to fit a 16x16 pixels frame.

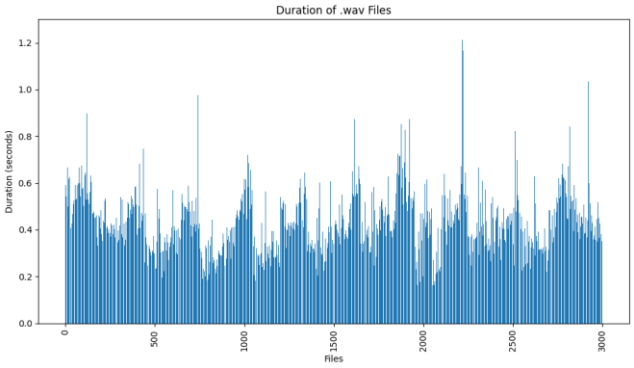

Figure 1: A graph showing the duration of all the audio files, ranging from 0.2s to 1.2s.

Methods

Firstly, to visualise the data, we plotted waveforms of a few random samples of audio files.

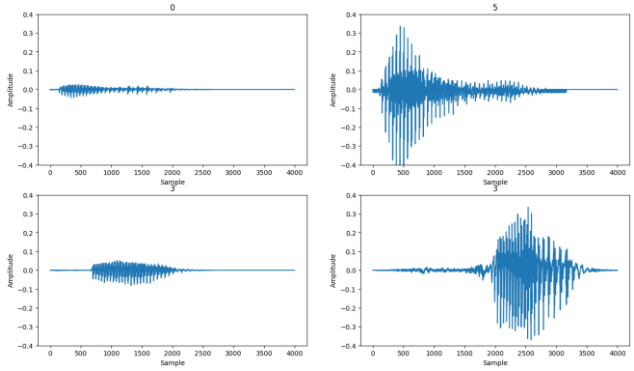

Figure 2: Waveforms of some random samples from the dataset.

Another method of visualisation we explored is converting the audio files into a spectrogram of its frequencies. Displayed below is a spectrogram of such frequencies. We chose spectrograms over waveforms as spectrograms offer more information than waveforms. For instance, spectrograms allow us to identify frequency changes over time, while waveforms only show differences in amplitude (Izotope 2020). With more information in the spectrograms, it would be easier to identify and analyse the patterns present.

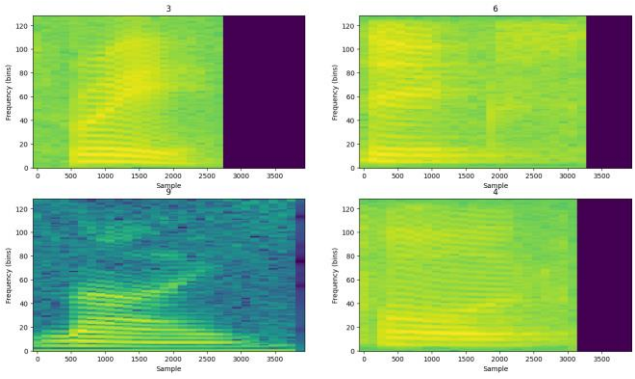

Figure 3: Spectrograms of some random samples from the dataset. The areas in purple reflect the added padding from the loading of data.

With the spectrograms, we hypothesised that the nuances in pronunciation of each digit would be reflected in the pitches present in the spectrogram. A model that we noted that excels in deriving patterns from spectrograms is Convolutional Neural Networks (CNN). CNNs can identify local patterns in data thanks to their convolutional layers. These layers apply filters to small local regions allowing them to detect features at various scales. These properties make CNNs a suitable solution for our problem.

With that, we implemented a CNN that creates a spectrogram of the audio clips and inputs them as image data into the convolution layers, with its pixels acting as numerical data for the CNN. To implement the model and evaluate its baseline performance, we took reference from a TensorFlow tutorial for our code (TensorFlow 2023). A diagram for the initial CNN architecture is as follows.

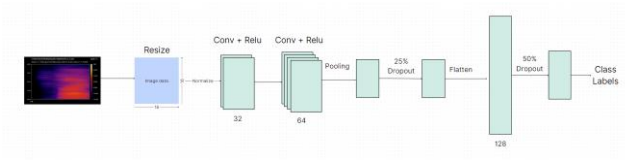

Figure 4: A flow chart of the initial CNN architecture.

As part of the pre-processing step, we also introduced a normalisation layer in the CNN to ensure that each pixel of the spectrogram would be equally weighted in the model.

For our model, we decided to set the number of epochs to 10. This is to ensure that there is sufficient time for the model to learn and improve while also reducing the risk of potentially overfitting the data.

Results and Discussion

Since the problem involves the simple classification of 10 different digits, we decided that the model accuracy and confusion matrix are the most appropriate performance metrics.

We began by training variations of the above architecture with the Adam optimiser. With the generic hyperparameters, the model returned a validation accuracy of 93%, suggesting that the CNN model may be a good solution to our Spoken Digits problem. That said, the hyperparameters used in this model can be tweaked to better fit the specific problem of classifying digits.

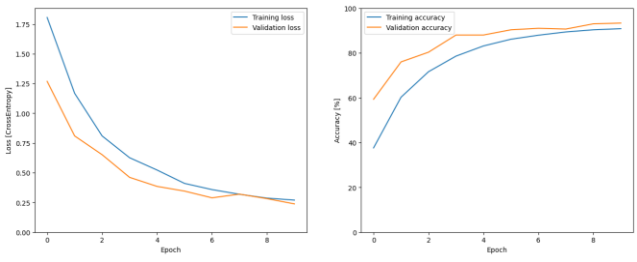

Figure 5: Plots of the training loss and accuracy of the initial model. A validation accuracy of 93% was achieved.

Finetuning of Hyperparameters

With promising results, we continued to finetune our model with the hyperparameter tuning framework, Optuna. By taking references to the code examples on Optuna's webpage, we seek to further optimise our hyperparameters instead of resorting to trial-and-error (Optuna n.d.). With these new parameters, we updated our model.

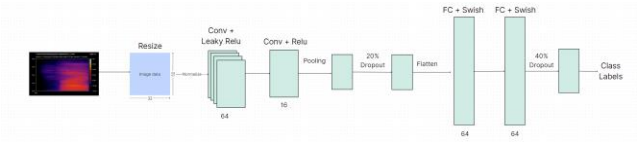

Figure 6: A flow chart of the improved CNN architecture.

Validation of Results

Furthermore, to ensure that the improvements in training accuracy are not due to overfitting, we implemented a variation of cross-validation known as the Hold-Out method. This method consists of a training-test-validation split in which a portion of the dataset, the test set, is left out of the training-validation iterative loop and is only used to provide an unbiased evaluation of the model's performance. Due to the computational costs of K-fold cross-validation, we opted for the cheaper Hold-Out method.

We decided on an 80-10-10 randomised split for the training, validation, and test datasets respectively. Firstly, an 80-20 training and validation split (also referred to as the Pareto principle) is used to ensure there is enough data to train a consistent model while also minimising variance in validation metrics. Afterwards, the validation data is split into 2 to create a testing set that is separate from the validation set.

For this model, we increased the number of epochs to 20, but also implemented early stopping to reduce the risk of overfitting the data. Additionally, the resizing of spectrogram inputs was changed from 16x16 to 32x32 to provide higher-resolution data for the model.

Final Performance of the Model

With the optimised hyperparameters and the inclusion of the validation set, we ran the model again, this time achieving a validation accuracy of ~96% while also keeping validation loss low. Good performance is also seen in the confusion matrix, with most of the results falling on the diagonal. With good results shown in the performance metrics, we believe that this model is well-suited to addressing the problem of classifying spoken digits.

Compared to previously done work, our CNN model shows similar performance. Our model outperformed a similar STFT-CNN model that had an accuracy of 90.4% for spoken English digits (Ba 2021). However, another CNN model which used Bark spectrograms as part of feature extraction achieved an average accuracy of over 99% (Safie 2022). Therefore, while the results of our current model are promising, more can be done to further optimise its performance.

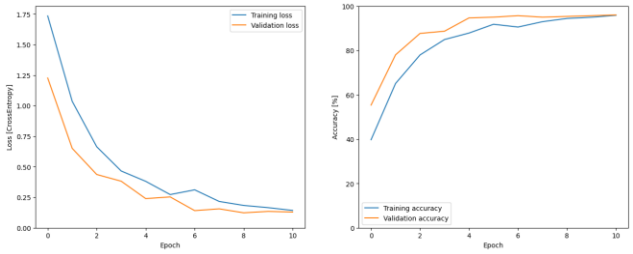

Figure 7: Plots of the training/ validation loss and accuracy of the improved model. A validation accuracy of 96% was achieved while keeping validation loss to a minimum.

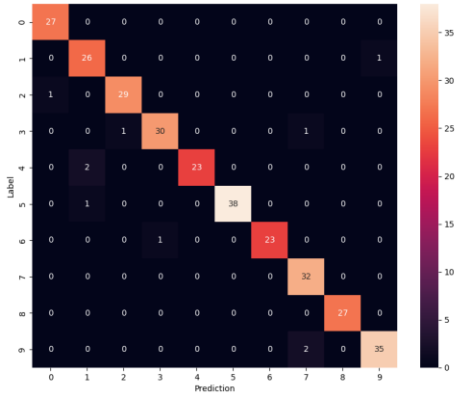

Figure 8: A confusion matrix of the results for the improved model. The results fall mostly on the diagonal, indicating a high accuracy of the model.

Alternative Approaches

We have seen how CNN was able to provide us with a great degree of accuracy and low loss. But we are also interested in how other supervised learning methods perform in the context of Audio Classification. In this section, we will explore the use of the following models and evaluate their performance.

- K-Nearest Neighbours (KNN)
- Decision Trees (DT)
- Gaussian Naïve Bayes (GNB)

To implement these models, we took reference from a GitHub page for our code (Zenkov 2020).

Feature Extraction

We have explored the use of 5 different features to be used for our supervised learning models, namely:

- Waveforms
- STFTs

- Mel-Frequency Cepstral Coefficients (MFCCs)
- Mel Spectrograms
- Chromagrams

Eventually, we decided to use MFCCs, Mel Spectrograms and Chromagrams for our next 3 models for better distinguishing power between each sample. To do so, we obtain an array from each of the 3 features and stack them horizontally to create a single feature array.

Before proceeding to the training of the models, we also used MinMaxScaler and StandardScaler from the Scikit-learn package as part of feature scaling, thereby creating 2 additional datasets on top of the unscaled dataset. This step is especially crucial for models which use distance metrics such as KNN.

Model 2: K-Nearest Neighbours (KNN)

KNN is a non-parametric supervised learning that plots our training samples' features and compares a test sample's features' distance to all those points. It then takes the k closest points to the test sample and picks the most frequent label/ class. This method assumes that data points that are close share similar features and therefore are more likely to belong to the same class.

With our KNN model, we found its accuracy across multiple runs hovering around 91-92% for all 3 datasets, making it very effective at addressing the spoken digits problem. However, we note that KNN may become computationally intensive when a large dataset is used as its algorithm involves calculating the distances between a query point and all the other data points.

Model 3: Decision Trees (DT)

A DT is a non-parametric supervised learning method that seeks to learn simple decision 'rules' from the features of the data and predict the category that a target belongs to (Scikit-learn n.d.).

For our DT model, we ran it using 3 different criteria (Gini impurity, Entropy, and Log loss) but all their accuracies remain around 75%, showing that it is underperforming for the specified problem.

Model 4: Gaussian Naïve Bayes (GNB)

Naïve Bayes is a supervised learning algorithm that leverages Bayes' theorem and assumes that the features of the data are conditionally independent (Scikit-learn n.d.). In our case, we used the Gaussian variant as we assume the features follow a Gaussian distribution.

Our GNB model returned the lowest accuracy out of all the models we have tried, having an accuracy of only 52%. However, its merit lies in the fact that its classifiers are extremely fast compared to more sophisticated methods.

Validation of Results

To have a better idea of each of the models' general performance, we implemented K-fold cross-validation. This involves first splitting the dataset into K folds, and then using K-1 folds to train the model while leaving the last fold for model validation. This step is repeated K times and the average accuracy of each run is recorded. For our models, we will be using 10-fold Cross-Validation.

	KNN	**DT**	**GNB**
Average accuracy (%)	90.08% ± 2.14%	74.17% ± 1.33%	53.92% ± 3.22%

Table 1: The results of running 10-fold cross-validation on the 3 models. The KNN model shows the best performance, followed by the DT and GNB models.

After cross validation, the KNN model remains the best-performing of the three, with an average accuracy of 90%, while the DT and GNB models have accuracies of 74% and 54% respectively. However, none of the models performed better than the CNN model with its accuracy lying at 96% for the spoken digits problem.

Conclusion and Further Work

In conclusion, the CNN model shows the best performance out of all the different approaches attempted to address the problem of classifying spoken digits. This is most likely due to the CNN's capabilities of extracting key patterns from image data, allowing it to classify audio accurately based on these learnt patterns. Furthermore, while the KNN model had a comparable accuracy from our testing, its downside of becoming computationally intensive with larger datasets makes the CNN model a preferable solution.

That said, there is still some more room for improvement. As the dataset is small (comprising only 6 speakers), the current model may not be well-suited to the voices of the general population. Female voices are unrepresented in this dataset, and the different pitches of their voices may pose some challenges for this model. The model may also find some difficulty in classifying voices of different accents due to their unique speech patterns. Therefore, more work should be done before this model is fully prepared for use by the general population.

References

Ba, H. 2021. Spoken Digit Classification: A Method Using Convolutional Neural Network and Mixed Feature. *Proceedings of 2021 the 11th International Workshop on Computer Science and Engineering (WCSE 2021)*. 7-12. doi.org/10.18178/wcse.2021.02.002.

Gilbert, C. 2022. Voice recognition assistants help people with severe speech impairments. AbilityNet. https://abilitynet.org.uk/news-blogs/voice-recognition-assistants-help-people-severe-speech-impairments. Accessed: 2023-10-31.

Hoy, M. B. 2018. Alexa, Siri, Cortana, and more: An introduction to voice assistants. *Medical Reference Services Quarterly* 37(1), 81-88. doi.org/10.1080/02763869.2018.1404391.

Huang, C.; Chen, T.; Li, S.; Chang, E.; and Zhou, J. 2001. Analysis of speaker variability. *7th European Conference on Speech Communication and Technology (Eurospeech 2001)*. doi.org/10.21437/eurospeech.2001-356.

Izotope. 2020. Understanding spectrograms. https://www.izotope.com/en/learn/understanding-spectrograms.html. Accessed: 2023-10-31.

Optuna. n.d. A hyperparameter optimization framework. https://optuna.org/#code_examples. Accessed: 2023-10-31.

Scikit-learn. n.d. Naive Bayes. https://scikit-learn.org/stable/modules/naive_bayes.html. Accessed: 2023-10-31.

Palaz, D.; Magimai.-Doss, M.; and Collobert, R. 2015. Convolutional neural networks-based continuous speech recognition using raw speech signal. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. doi.org/10.1109/icassp.2015.7178781.

Safie, S., I. 2022. Spoken Digit Recognition Using Convolutional Neural Network. *2022 Applied Informatics International Conference (AiIC)*. 24-27. doi.org/10.1109/AiIC54368.2022.9914596.

Scikit-learn. n.d. Decision trees. https://scikit-learn.org/stable/modules/tree.html. Accessed: 2023-10-31.

TensorFlow. 2023. Simple audio recognition: Recognizing keywords. https://www.tensorflow.org/tutorials/audio/simple_audio. Accessed: 2023-10-31.

Zenkov, I. 2020. Sklearn-audio-classification/sklearn_audio_classification.ipynb at master · IliaZenkov/sklearn-audio-classification. GitHub. https://github.com/IliaZenkov/sklearn-audio-classification/blob/master/sklearn_audio_classification.ipynb. Accessed: 2023-10-31.