Identifying accents with machine learning

Team (60)

Adam Le, n10998683
Brendon Dam, n11072059
Lin Shane Chen, n11083603
Allem Karaahmetovic, n11084219

Table of Contents

1 Problem Definition and Background

1.1 Introduction

Accents are a unique trait that are associated with different cultures, regions and social groups which play an extremely important part when it comes to communication. The ability to identify accents can enhance many applications, in particular having the potential to improve speech recognition systems. As such the motivation for this project stems from the increasing importance of effective communication in such a globalised world. Throughout worldwide communications, accents can influence the accuracy of speech recognition systems, which are integral to virtual assistants and AI customer services which have raised in popularity. Misinterpretation due to differences in accents leads to frustration and inefficiency, which is indicative of the need for systems that are adaptable to multiple speech patterns. In turn, this sparked the investigation into how to efficiently classify different accents in the audio recordings of speakers.

1.2 Related Work

Methods for Audio Classification:

M1. **Naive Bayes** - Chen, Lee, and Neidert [2]

M2. **Logistic Regression** - Chen, Lee, and Neidert [2]

M3. **Multiple-Layer Perceptron (MLP)** - Sheng and Edmund [3]

M4. **Convolutional Neural Networks (CNN)** - Sheng and Edmund [3]

M5. **Triplet State Loss Function** - Li [4]

M6. **Deep RNN (DRNN)** + **Recurrent LSTM** - Scarpiniti et al [5]

M7. **Bidirectional RNN** - Yu et al [6]

M8. **Recurrent Autoencoder** - Amiriparian et al [7]

M9. **Masked Autoencoder** - Huang et al [8]

M10. **CNN-LSTM Network** - Sang [9]

M11. **KNN** - Patil & Nemade [9]

{M 1-11} all did some form of audio classification. {M 1,2,3,4} was accent classification {M 5,9 } was speaker recognition {M6} was construction site audio classification {M7, 11} was music genre classification, {M8} was acoustic scene classification, {M9} speech command classification and {M10} urban sound classification.

{M1,M2} are simple non-deep learning approaches to accent classification,they are both not optimal due to the data such as the assumption of independence for {M1} or the sequential nature of audio data for {M2}. As expected the performance is poor, obtaining testing accuracies of 58.65% and 59.90% respectively.

Similarly, {M11} performs 5% better than {M1,M2}. This advantage is likely due to KNN's capacity to navigate the high-dimensional feature spaces common in audio analysis, where the relationships between samples can be intricate and non-linear.

Contrastingly, {M3,M4} are deep learning approaches that are capable of learning complex relationships. Not surprisingly, their test accuracies were 80% {M3} and 88% {M4}.

{M5} builds upon the techniques in {M4} → how is the performance compared to them, what is the benefit, what problem does it solve or why is it weaker

{M6,M7} are similar approaches. {M6} leverages the ability of LSTMs to handle long-term dependencies in sequences, which is crucial for understanding complex patterns in audio signals over time, on top of DCNN's. This approach demonstrated high accuracy in classifying various sounds, reaching up to 97% overall accuracy. Demonstrating its prowess on sequential data, due to the ability to capture temporal dependencies within the data.

{M7} Primarily is used in natural language processing. A Bidirectional RNN adds another layer to the standard LSTM, reversing the direction of information flow. Allowing models to utilise information from both past and future contexts simultaneously.

{M10} builds upon {M4 and M6}The CNN layers preprocess the audio data to extract meaningful features, which are then fed into the LSTM layers. The LSTMs analyse these features over time, allowing the model to understand the sequence of audio events and their durations. This approach is interesting because it seeks to leverage the strengths of both architectures to enhance classification performance. The results achieving 79% accuracy in the best out of 4 models, were moderately good. However, {M4} which is purely CNN achieved 88%. However the accuracies are not directly comparable due to the dataset differences. The trade off of this approach is its increased model complexity which can lead to longer training times,{M10} will most likely take longer to train than all of {M1,M2,M3,M5}.

{M8} Focuses on leveraging the sequential nature of data through RNNs to learn efficient representations. It focuses on encoding and decoding sequences to capture temporal dependencies.

{M9} Took a novel approach where the autoencoder is trained to predict masked portions of the input data. This method involves randomly masking certain segments of the input sequence during training and forcing the model to predict these masked values based on the unmasked parts. The goal is to encourage the model to learn robust and meaningful representations that can generalise well to unseen data.

Models that fall into the category of high complexity, {M4,5,6,7,8,9,10} consequently are considered to have long training times. In contrast {M1,2,11} are considered to be low complexity and have short training times. Lastly, {M3}is moderate complexity and training time.

2 Data

Validated data from the Mozilla Common Voice Corpus 1 dataset was used which contained 490,483 mp3 audio samples of 26,744 individuals speaking a random sentence in English. This dataset was selected for its range of 17 accents, which provides a more realistic representation for training and classification. The data was first pre-processed by dropping all samples that did not have an accent labelled, reducing the size to 215,116. From here, There were two major class imbalances discovered in the data.

	Male	Female	Other
Before setting class limit	161607	46719	4563
After setting class limit	10875	12127	802

Figure 1: Number of samples for Male, Female and Other genders in the sample data before and after applying 2000 sample limit to class sizes.

Males make up 76% of the total sample sizes with *females* and *others* making up only 22% and 2% accordingly. Alongside the accent imbalances shown in figure 2, the data is heavily biased towards Male US speakers. To address both the accent and gender imbalance, accents were limited to a maximum of 2,000 samples and those removed samples would be only male speakers. This resulted in the final data size of 24,391 samples with a gender split of 46%, 51% and 3% for *males*, *females*, and *others* accordingly.

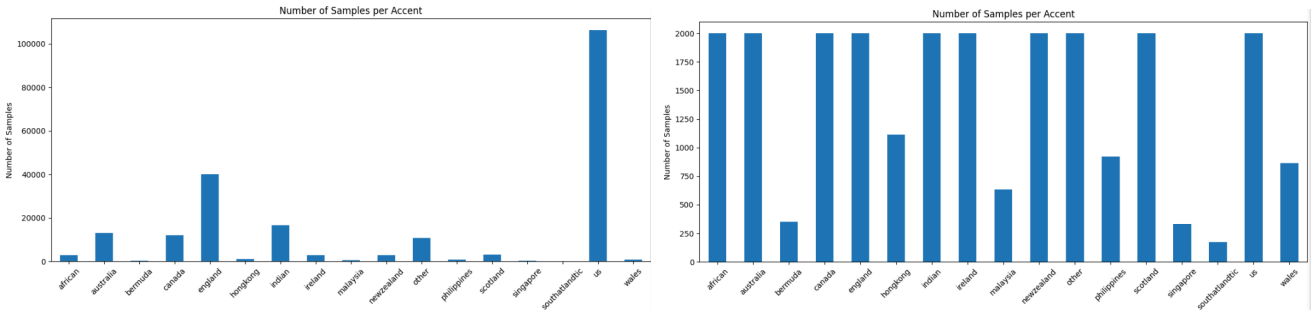


Figure 2: Class distributions of accents before setting class limits (left) and after setting class limits (right)

Imbalances for accents were significantly improved however they still contain minority classes such as *South Atlantic, Bermuda and Singapore* which could be improved by further limiting class sizes. This was left as is to not lose too many samples. Feature extraction was conducted by calculating Mel Frequency Cepstral Coefficients (*MFCCs*) for each audio sample. These coefficients extract different aspects of the audio signal's spectral properties in which samples of the same accent will exhibit similar coefficients, whereas different classes of accents will show distinct spectral properties. 50 Coefficients were calculated per sample to balance between capturing fine details and computational complexity. Dimension reduction was then performed by converting MFCCs to their mean value. This was mainly done to simplify working with the data at the cost of information richness.

Data was then randomly split (using the same seed amongst all models) into a train, validation and test subset containing 50%, 15% and 15% of the data accordingly.

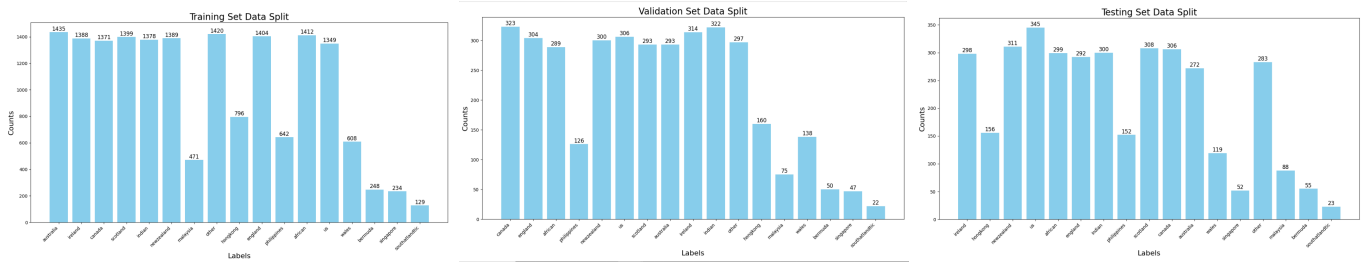


Figure 3: Training splits for training (left) validation (middle) and testing (right) sets.

3 Discussion of Proposed Approaches

3.1 K-Nearest Neighbours Classifiers (KNN)

KNN was chosen as one of the four methods to be explored due to its simplicity, computational efficiency, and ease of interpretation. Unlike deep learning models, KNN does not require a training phase, making it less intensive computationally. Despite not expecting the best results, KNN is still a suitable method for several reasons:

1. **Nonparametric Nature**: KNN makes minimal assumptions about data structure, unlike SVM's, which assume linear separability. This flexibility allows KNN to adapt to the complexities in accent classification and MFCC data, which are often high-dimensional and non-linearly separable.
2. **Multi-class Capability**: KNN can handle multi-class classification, which is essential for identifying 17 different accents. Although the method is sensitive to missing data, preprocessing has addressed the NaN values common in the crowdsourced dataset.

A critical aspect of using KNN is selecting the appropriate value for (k). Smaller (k) values may lead to overfitting, while larger values reduce sensitivity to noise but may miss small details and rare classes. A grid search with values [1, 3, 5, 9, 19, 49, 99, 199] will be used to find the optimal (k), considering class imbalances in the dataset. Additionally, different distance metrics (Euclidean, Manhattan, Chebyshev, Minkowski) will be compared as they emphasise different aspects of the data.

The approach combines Linear Discriminant Analysis (LDA) for dimensionality reduction and KNN for classification, optimised through grid search. LDA reduces the dimensionality of MFCC features while maximising class separability, enhancing computational efficiency and classifier performance. The Best parameters for LDA + KNN was euclidean metric and n neighbours of 1.

3.2 Long Short-Term Memory Network

Another method that was proposed for this task is Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN) that is well-suited for sequence prediction problems. As mentioned in the related work section, traditional RNNs have a limitation where it struggles with the vanishing gradient

problem. This can cause severe disadvantage to this task because accents involve precise variations in pronunciation, tone, and stress patterns over time. Capturing these characteristics requires the network to remember information across long sequences of audio data. This means that the vanishing gradient problem can significantly hinder the results of the neural network. LSTMs on the other hand, is a type of RNN designed to capture long-term dependencies in sequential data. It has a more sophisticated structure which allows it to effectively maintain and update information over long periods. The LSTM also requires less computational power compared to the Deep RNN or Bidirectional RNN models mentioned before making it a more suitable model for this task.

3.3 Siamese Network with Triplet Loss

Siamese networks with triplet loss are neural network models designed to learn a similarity function amongst a set of triplets of input representations. The objective is to find a function that increases the similarity between the anchor and the positive example while decreasing the similarity between the anchor and the negative example. In this case, Anchor, positive and negative examples represent the MFCCs of a chosen sample, a sample belonging to the same accent, and a sample of a different accent, respectively. This separation improves class distinguishment for classification performed later on.

A network from the authors of [11] was extended upon. Two changes were introduced. Triplets instead of pairs were used for the loss function for better discriminatory power. Dropout layers in the base network were removed as empirical testing showed removing it significantly improved accuracy scores. This may be due to having enough regularisation power from using L2 regularisation.

Hence the model uses a base network with three dense layers of 128, 64 and 32 nodes respectively with an L2 regularisation function on the final layer. The base network was not frozen during classification training to also be able to learn from classification training. Empirical testing confirmed this improved performance. A triplet loss network is attached using a margin of 0.5 (best value from empirical testing). A final classification head consisting of one dense layer of 17 nodes is attached to the overall network. A summary of the network can be found in appendix 2.

3.4 Feed-Forward Network

Multi-layer Perceptron (MLP) networks are well suited to handle the feature vectors of our data which take the form of 50 MFCCs. They are known for their effectiveness in learning patterns and relationships between the features of these vectors. Typically, MLPs consist of an input layer, where basic features are extracted; multiple hidden layers, which combine features learned by each prior layer to form more complex representations; and an output layer that aggregates all the learned features to form a comprehensive representation of the input data for classification. A Feed-Forward Multi-layer Perceptron (MLP) network from the authors of [3] was extended upon for this topic. Their model achieved an accuracy of 80% which showed great potential in extending it to our task.

Hence, our model follows a similar structure to [3], consisting of four fully connected dense layers with the first layer modified to take an input size of [, 50] in order to fit the structure of our data. An output size of 17 in the final dense layer with a softmax activation was used to identify the most likely class amongst the 17 accents. All dense layers are followed by batch normalisation and a 50% dropout overfitting and to stabilise / accelerate the training process accordingly. They are also followed by a ReLu activation to

introduce non-linearity to the model hence, enabling complex patterns to be learnt in training. Summary of the model can be found in appendix 1. From empirical testing, a batch size of 128 gave the best performance thus was used.

4 Evaluation and Discussion

All models trained on the same set of hardware to facilitate direct comparison of training times, allowing for a standardised evaluation of model efficiency and performance optimization opportunities.

	Accuracy	Precision	Recall	F1-Score	Training Time
KNN	0.66	0.88	0.66	0.72	1 Second
PCA + KNN	0.31	0.64	0.29	0.34	1.5 Seconds
LDA + KNN	0.73	0.72	0.73	0.72	3 Seconds
LSTM	0.67	0.69	0.67	0.66	14 Minutes
LSTM + PCA	0.50	0.51	0.50	0.49	14 Minutes 5 Seconds
Siamese Network with Triplet Loss	0.76	0.76	0.76	0.76	1 Minutes 26 Seconds
Feed Forward Network	0.83	0.84	0.83	0.84	7 Minutes 30 Seconds

Figure 4: summary of model performances run on the testing data set

4.1 KNN

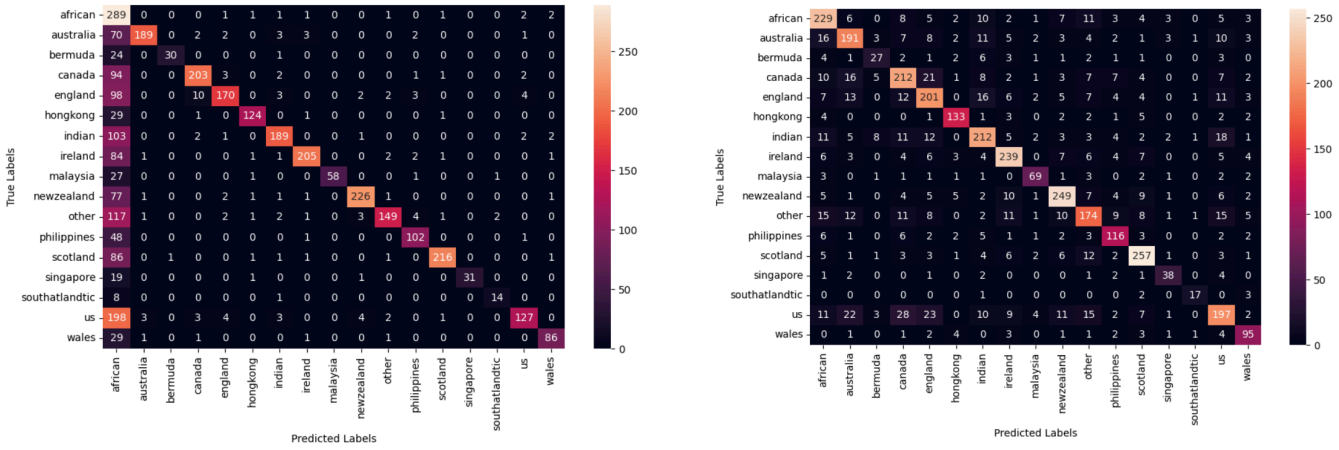


Figure 5: KNN Confusion Matrix (left) and LDA + KNN Confusion Matrix (right)

Figure 5 shows the KNN model without LDA has a significant outlier with the African accent class, achieving a low precision score of 0.21 but high recall of 0.97, indicating many false positives. This highlights a limitation in handling uncertainty among closely related classes. Conversely, integrating LDA offers more stable results. The KNN confusion matrix reveals low off-diagonal values, indicating fewer

misclassifications except for the African class, with a higher overall precision score of 0.88 compared to using LDA which yields 0.72.

Despite KNN's faster model loading times (<1 sec), its accuracy is the second lowest (0.66). Whereas LDA integration achieves an accuracy of 0.73 which is only 3% worse than Siamese Network with Triplet Loss but only with a 3 second model loading time. Both models share an average F1-Score of 0.72, but KNN's min to max range (0.34 to 0.86) contrasts with LDA+KNN's narrower range (0.52 to 0.85), showing KNN's variable performance across classes which makes the outputs less trustworthy.

Compared to a similar KNN-based audio classification study which achieved a mean accuracy of 0.64, precision of 0.70 and recall of 0.40 [10] , these findings suggest that the complexity of the feature space affects KNN's effectiveness as that model used much more complex music genre data. LDA integrated KNN performed better due to LDA's ability to reduce dimensionality and focus on the most discriminative features, thereby improving the efficiency and effectiveness of KNN in classification tasks.

When compared to LSTM, LDA integrated KNN performs almost 10% better which is interesting as it has a much shorter model loading time vs LSTM’s training time of 14 minutes. KNN with LDA frequently misclassifies the US accent as Australian, Canadian, or English, highlighting KNN's limitations in discerning subtle differences between closely related accents stemming from its reliance on distance metrics rather than learning from the data, making it challenging to capture nuanced differences. This underscores the necessity for more sophisticated methods capable of better capturing the nuances between accents. The feed-forward network's 14% improvement over LDA integrated KNN demonstrates its enhanced classification capabilities, underscoring the benefits of advanced techniques in accent recognition tasks.

4.2 LSTM

The LSTM model showed a significantly longer training time compared to all the other models being trained with the same dataset, epochs, and batch size. Specifically, it takes almost twice as long to train the LSTM compared to the second-longest model, the Feed-Forward Neural Network (FNN). This indicates that the LSTM is a more complex model compared to the others and that it could be an indication that the accuracy results would be better.

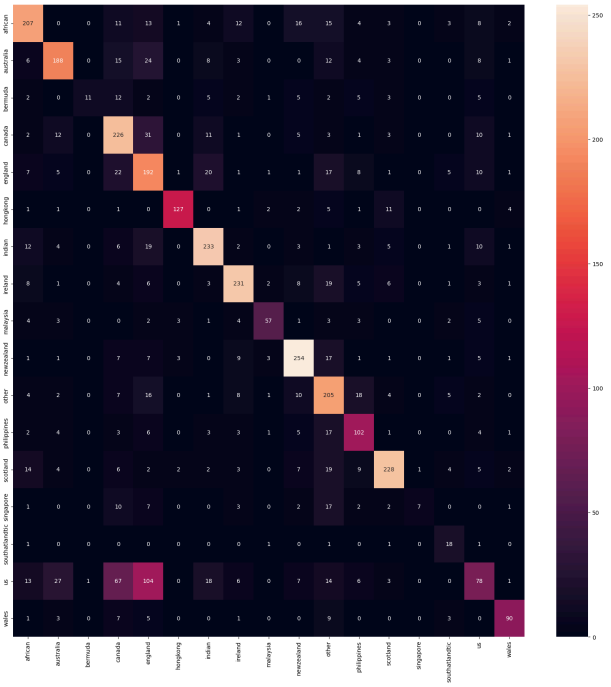


Figure 6: Confusion chart of LSTM on the test set.

Figure 6 shows the confusion matrix of the LSTM model. It is evident that most of the classes are predicted correctly, with the notable exception of the US accent. The US accent is frequently misclassified as either a Canadian or English accent. In fact, the model predicts the US accent as an English accent more often than it correctly identifies it as a US accent. This misclassification significantly impacts the overall performance metrics of the model. It is likely to be the reason why the precision score, recall score, and f1 score are only 0.69, 0.67, and 0.66 respectively. These scores are slightly lower compared to other deep-learning methods implemented while having a much longer training duration, which may indicate several issues. One possibility is that the LSTM model might not be suitable for accent classification. However, this is unlikely because related work shows that RNN-like models, including LSTMs, are often successfully used for audio classification tasks.

Another more logical reason is that the preprocessing of the data used for this task is not suitable for the LSTM. As mentioned in the previous section, feature extraction was conducted by calculating the audio's MFCC then dimension reduction was then performed by converting MFCCs to their mean value; however, one of the key strengths of LSTM models is its ability to capture temporal dependencies in sequential data. By taking the mean of MFCCs, you lose the sequential information that could be critical for distinguishing accents, which are often characterised by subtle temporal patterns in speech.

For this task, the LSMT model performed poorly compared to the other deep learning methods implemented. From previous studies the results are also unstable with a testing accuracy of 59.7% on the music genre classification [12] which is approximately 10% less than this task. The author of this paper performed similar preprocessing techniques where they extracted the MFCCs from the raw audio data. This may be because music genres can involve much more complex and longer-term temporal patterns. Music includes various instruments, rhythms, harmonies, and structural elements that can span longer time periods makes it more complex for LSTM to capture these patterns as sequential information is lost from preprocessing. However, it may also be caused by the model only being trained for 30 epochs whereas for our model, it was trained for 400. Another study of urban sound classification [13] resulted in a testing accuracy of 84.25%. This is a much better performance compared to our model. However, this may be because for the urban sound classification, the author pre-processed the model with a combination of mel-spectrum, MFCCs, and Cross Recurrence Plot (CRP) images. Ultimately, the loss in performance of the LSTM is most likely a result of the loss of spatial information, particularly the sequential information, that occurs as a result of the reduction in dimensions.

4.3 Siamese Network with Triplet Loss

The model performed relatively well with a score of 0.76 for Precision, Recall and F1 as previously seen in figure 4. The best individual siamese model used by [11] beats this with a 80.4% accuracy score. Their loss function used pairs rather than triplets, but consisted of 100,000 positive pairs and 900,000 negative pairs. Our model used only 100 triplet pairs which gives our model less samples to learn features from. This was not a likely bottleneck for our performance however, as training always converges at around 100 epochs regardless of the number of triplets generated as seen in figure 8. As the models are very similar in structure, performance is likely attributed to the difference in data. [11] particularly excluded Western accents that our model struggled with, and primarily used accents that are more distinguishable e.g. Russian, Brazilian. In combination with having less accents to identify (10), these could have contributed to the difference in

performance. Performance of [11] is not perfectly comparable however as their accuracy uses top-3 identification ranks for accents instead.

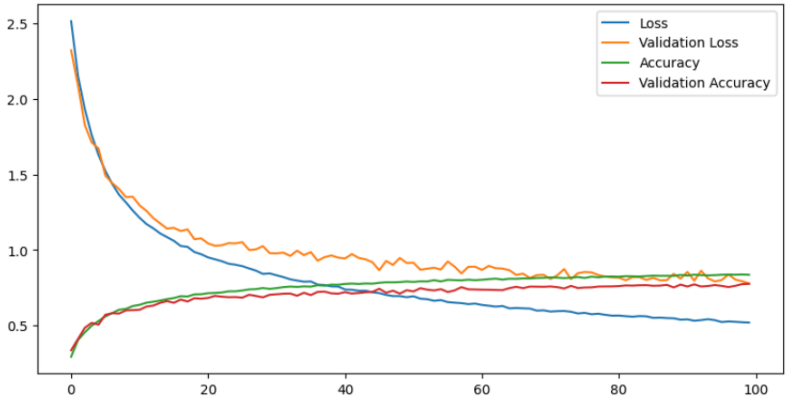


Figure 7: Graph of training and validation accuracies and losses during training for Siamese Network.

Training time was 1 minute 26 seconds to converge at 100 epochs. This is much quicker than both the *LSMT* and *MLP* models, both of which converged at 400 epochs showing the siamese network overfitting quicker.

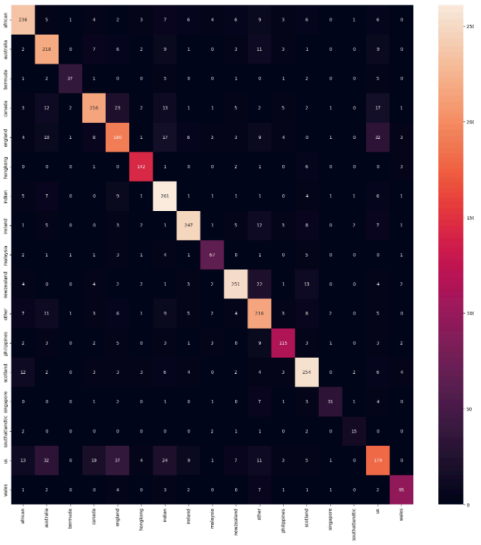


Figure 8: Confusion chart of Siamese network with triplet-loss on the test set.

Figure 8 shows similar errors found in the other models amongst Western accents such as US, Canada, Australia and England. This is backed up when looking at the embedding space of figure 9. Well differentiated accents such as Hong Kong and Ireland are clustered in the same outer embedding space. Scattering of US samples however are distributed amongst the left half of the space where other Western classes are primarily located which highlights how the accents are also similar in embeddings.

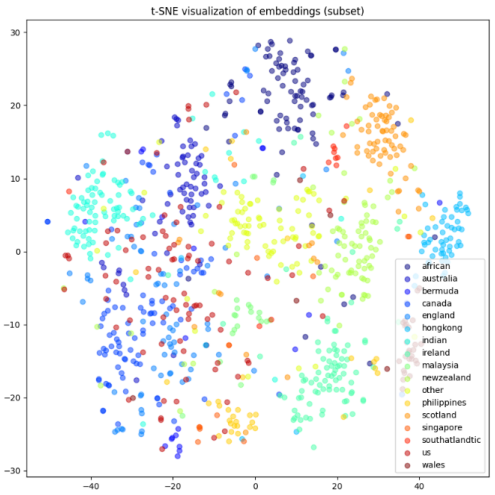


Figure 9: Embedding space of 1000 random samples created from the base network using t-SNE (n_components = 2)

Overall, the Siamese network was still mostly successful in pulling MFCCs with similar coefficients (that represent similar accents) close together, and pushing different ones apart in the embedding space. As previously mentioned, the loss in temporal information for LSMTs was enough for the Siamese model to outperform it in accuracy. However, the Siamese model was outperformed by the Feed-Forward network as seen in figure 2.

Difference in performance could be attributed to the MFCCs themselves. Triplet loss is designed to learn a distance metric, where the objective is to minimise the distance between similar samples and maximise the distance between dissimilar samples. Samples of same-class accents are not of the exact same speaker and carry differences in tone, punctuation and pitch amongst the speakers. Hence, these dissimilarities reflect in the MFCCs and the corresponding embeddings created, hindering models ability to separate classes but still having enough correlated information to classify most accents correctly.

4.4 Feed-Forward Network

Confusion chart from figure 11 shows the model's performance on the testing set with an F1, Precision and Recall score of 0.84, 0.84, 0.83 respectively. Scores outperform testing results from author [3] but only by 3%. The small difference can be likely attributed to the difference of databases used and quality of audio. The scores also outperform all the other models in this project. As mentioned in previous sections, the data was processed in a format not typical to specialised network LSTM and had flaws for the Siamese network. Furthermore the complexity of the data was suitable for a simple network like MLP to perform best on. MFCCs represent independent feature vectors that *MLPs* are able to learn deep features and patterns upon. Naturally, it also outperformed KNN, being able to distinguish samples using deep patterns not learnt by non-deep models.

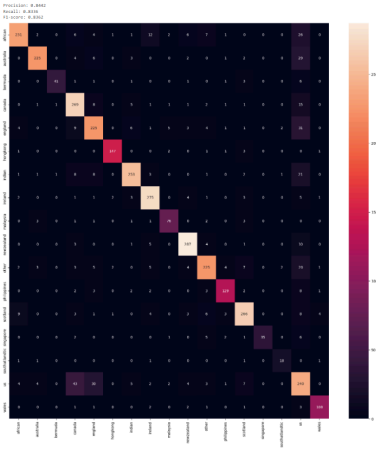


Figure 10: Confusion chart of Feed-Forward *MLP* performance on the test set.

Common errors from the other models are still present as seen in Figure 10, although are not as frequent. US accents are still most commonly mistaken for other Western accents such as Canada, England, Australia and vice-versa. Notably, the model has significantly less Australian samples falsely classified as US than the other models. The model was likely able to better capture richer nuances of the Australian pronunciation. In general the model was able to better capture features to distinguish accents compared to the other models, having less overall errors and outliers.

Training time was 7 minutes 34 seconds. *MLP*'s time ranks in the middle amongst all models, being quicker than LSMT but worse than the KNN and Siamese models. Time could be cut significantly by reducing the number of epochs. This would be a very feasible choice as performance per epoch diminishes heavily towards convergence as seen in figure 11. This also suggests that the model was able to learn deeper hierarchical representations given the total epochs to convergence. Given its performance, *MLP's* training time is relatively good for the cost of computation.

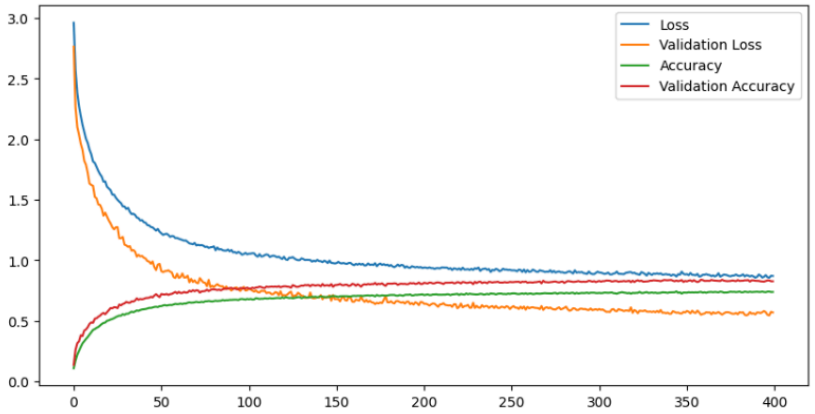


Figure 11: Graph of training and validation accuracies and losses during training.

5 Conclusion

The results from our project show the capabilities of relatively low-demanding models to classify accents of speakers from diverse backgrounds, although improvements can be made in various areas.

The most notable finding from this report is the performance of the KNN and MLP network when compared to the other methods explored. KNN excelled in its speed whilst providing solid performance. Whereas, MLP displayed the best performance at a moderate training time. Therefore, if the objective is to find a satisfactory performing model but speed is the key determining factor, LDA integrated KNN is a suitable choice. Similarly, if performance is the main concern while maintaining a short training time, MLP is a suitable choice.

There were downsides to the way we processed the data for its suitability to the models. Using the means of MFCCs, the data format was most suitable for simple networks like *MLP* to perform best in but lost important sequential information that is useful for *LSMT* models. Hence, using the entire sequence of MFCC frames would be a better choice to improve LSMTs performance to ensure the model can retain deep temporal information from the data.

The MFCCs used were calculated from audio of different sentences. Different sentences can introduce various phonetic elements as an additional characteristic that models can learn from to better differentiate accents. This also however causes models to learn variability not directly related to the accent such as word choice. Hence, experimenting with aligning similar phrases and specific keywords in samples may improve performance in this regard. This is especially true for Siamese Networks which results in more similar embeddings for same-class accents by removing differences in speech (by creating more similar MFCCs and their corresponding means).

Data augmentation could also be included in pre-processing to increase performance, through adding background noise, and varying pitch and speed of samples. This would help to better generalise the data and assist the model in becoming more robust in recognizing accents under various circumstances, emulating real world conditions.

Furthermore, setting maximum sample size at 2000 per class improved the class imbalances (by bringing the difference ratios much closer) but still contained imbalances. This was due to the lack of samples in rare accents like South Atlantic and Bermuda. Hence, using a larger dataset like *Common Voice Corpus 17* could further mitigate class imbalances by providing more samples to small sized classes.

6 References

Patil, N. M., & Nemade, M. U. (2017). Music Genre Classification Using MFCC, K-NN and SVM Classifier. *International Journal of Computer Engineering In Research Trends*, *4*(2), 43-47. https://ijcert.org/ems/ijcert_papers/V4I206.pdf

Thiruvengatanadhan, R. (2017). Speech/Music Classification using MFCC and KNN. *International Journal of Computational Intelligence Research*, *Volume 13*(Number 10), 2449-2452. Retrieved may 31, 2024, from https://www.ripublication.com/

[2] Neidert, J., Chen, P., Lee, J., Foreign accent classification, http://cs229.stanford.edu/proj2011/ChenLeeNeidert-ForeignAccentClassification.pdf

[3] Sheng, L., Edmund, M., Deep Learning Approach to Accent Classification, https://cs229.stanford.edu/proj2017/final-reports/5244230.pdf

[4] Li, T., Speaker Recognition System Based on Triplet State Loss Function, https://www.researchgate.net/publication/373315967_Speaker_Recognition_System_based_on_Triplet_State_Loss_Function

[5] M. Scarpiniti, D. Comminiello, A. Uncini and Y.-C. Lee, "Deep recurrent neural networks for audio classification in construction sites", *Proc. 28th Eur. Signal Process. Conf. (EUSIPCO)*, pp. 810-814, Jan. 2021. https://ieeexplore.ieee.org/document/9287802 https://www.eurasip.org/Proceedings/Eusipco/Eusipco2020/pdfs/0000810.pdf

[6] Y. Yu, S. Luo, S. Liu, H. Qiao, Y. Liu and L. Feng, "Deep attention based music genre classification", *Neurocomputing*, vol. 372, pp. 84-91, Jan. 2020. https://www.sciencedirect.com/science/article/pii/S0925231219313220#fig0002

[7] S. Amiriparian, M. Freitag, N. Cummins and B. Schuller, "Sequence to sequence autoencoders for unsupervised representation learning from audio", *Proc. DCASE*, pp. 17-21, 2017. https://dcase.community/documents/challenge2017/technical_reports/DCASE2017_Amiriparian_173.pdf

[8] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, et al., "Masked autoencoders that listen", *arXiv:2207.06405*, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/file/b89d5e209990b19e33b418e14f323998-Paper-Conference.pdf

[9] J. Sang, S. Park and J. Lee, "Convolutional recurrent neural networks for urban sound classification using raw waveforms", *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, pp. 2444-2448, Sep. 2018.https://ieeexplore.ieee.org/document/8553247

[10] Patil, N. M., & Nemade, M. U. (2017). Music Genre Classification Using MFCC, K-NN and SVM Classifier. *International Journal of Computer Engineering In Research Trends*, *4*(2), 43-47. https://ijcert.org/ems/ijcert_papers/V4I206.pdf

[11] Siddhant, Jyothi and Ganapathy. (2017). Leveraging Native Language Speech For Accent Identification Using Deep Siamese Networks. https://www.cse.iitb.ac.in/~pjyothi/files/ASRU2017

[12] Premtibadiya., "Music Genre Classification using RNN-LSTM". (2020). https://medium.com/@premtibadiya/music-genre-classification-using-rnn-lstm-1c212ba21e06

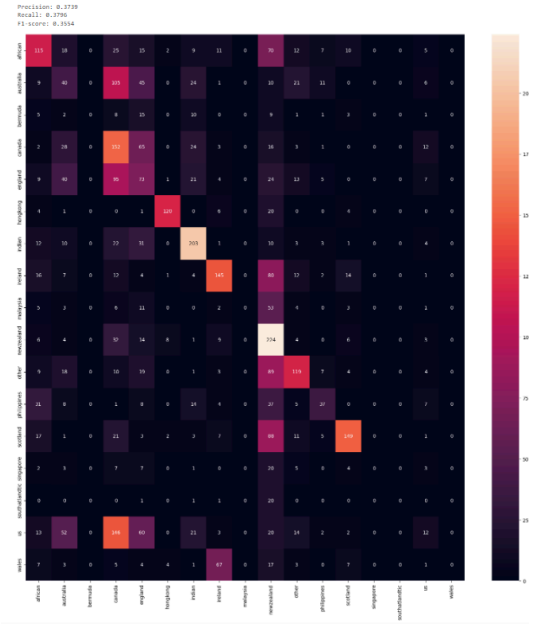
[13] Lezhenin. I., Bogach. N., Pyshkin. E. "Urban Sound Classification using Long Short-Term Memory Neural Network", https://annals-csis.org/Volume_18/drp/pdf/185.pdf

7 Appendix

Team Contribution

NAME	Percentage Split	Contributions	Signature to **show agreement** with table
Brendon Dam, n11072059	Full Contribution	• KNN • KNN Section • Helped in other parts of the report	BD
Lin Shane Chen, n11083603	Full Contribution	• LSTM- • LSTM Section- • Helped in other parts of the report	LSC
Allem Karaahmetovic, n11084219	Full Contribution	• Introduction • Feed Forward Network • Helped in other parts of the report	AK
Adam Le, n10998683	Full Contribution	• Data • Conclusion • Siamese Network with Triplet Loss • Helped in other parts of the report	AL

1. Siamese Network performance using 50% dropout layers.



2. Siamese-Network Summary (base network top, triplet network middle, classification head bottom)

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 50)	0
dense (Dense)	(None, 128)	6,528
batch_normalization (BatchNormalization)	(None, 128)	512
dense_1 (Dense)	(None, 64)	8,256
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dense_2 (Dense)	(None, 32)	2,080
batch_normalization_2 (BatchNormalization)	(None, 32)	128
lambda (Lambda)	(None, 32)	0

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 50)	0	-
input_layer_2 (InputLayer)	(None, 50)	0	-
input_layer_3 (InputLayer)	(None, 50)	0	-
functional_1 (Functional)	(None, 32)	17,760	input_layer_1[0][0], input_layer_2[0][0], input_layer_3[0][0]
concatenate (Concatenate)	(None, 96)	0	functional_1[0][0], functional_1[1][0], functional_1[2][0]

_

Layer (type)	Output Shape	Param #
input_layer_4 (InputLayer)	(None, 50)	0
functional_1 (Functional)	(None, 32)	17,760
dense_3 (Dense)	(None, 17)	561

3. Feed-Forward Network Summary

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 256)	13,056
batch_normalization (BatchNormalization)	(None, 256)	1,024
activation (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 256)	65,792
batch_normalization_1 (BatchNormalization)	(None, 256)	1,024
activation_1 (Activation)	(None, 256)	0
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65,792
batch_normalization_2 (BatchNormalization)	(None, 256)	1,024
activation_2 (Activation)	(None, 256)	0
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32,896
batch_normalization_3 (BatchNormalization)	(None, 128)	512
activation_3 (Activation)	(None, 128)	0
dropout_3 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 17)	2,193
activation_4 (Activation)	(None, 17)	0