Automatic Speech Recognition: Classification of Phonemes

Robert Schwartzberg and Eric Mauro
New York University,
New York, US
{ rds407, eric.mauro } @nyu.edu

May 10, 2018

Abstract

Speech-to-text has been a widely studied topic for some time, however, breakthroughs are still being made and there is much more room for improvement. As a result, several different methods have been tried with varying success. In this paper, we compare the performance of four different algorithms using a phoneme classification problem. The algorithms used are k-nearest neighbor, support vector machines, random forest, and a neural network. Our findings suggest that neural networks consistently outperform the other algorithms in this task and that k-nearest neighbor performs the worst.

1 Introduction

Automatic Speech Recognition (ASR) is a huge field of interest for research because of its many applications. The capability to turn speech into text allows for the creation of voice-controlled machinery and electronics, automatic closed-captioning of videos, increased accessibility for people with disabilities, and much more [1]. Even though a lot of improvement has been made in ASR technology, the problem is far from solved. Even state-of-the-art systems have difficulty achieving human-level speech recognition consistently. Some issues ASR faces is differentiating between similar sounding words, parsing words using limited dictionaries, and filtering out background noise. As a result, there are still a lot of benefits of conducting further research in the field.

2 Background

The classification of phonemes has been studied over the last few decades in a number of ways. Some early work focused on using Hidden Markov Models and Time-Delay Neural Networks to accurately classify them [2, 3].

Lee and Hon discuss an approach using Hidden Markov Models on the TIMIT dataset. Of the 64 different phonemes in their dataset, they selected 48 to focus their research on. They showed that they were able to correctly classify phonemes 58.8 to 73.8 percent of the time. They compare their results to expert spectrogram readers, who typically score about 69% accuracy using only the spectrogram data [2].

Waibel et al. implement a 3-layer feed-forward neural network capable of classifying non-linear class distribution. However, while their approach focuses only on differentiating between the similar sounds of 'B', 'D', and 'G', they are able to achieve a 98.5% accuracy rate on three different speakers [3].

Robinson and Fallside also conducted early research on phoneme detection using neural networks [4]. Their networks used stochastic gradient descent paired with a dynamic program solution to determine the most likely phonemes to follow the previous one. Using this combination, they were able to correctly classify 68.6% of the 61 phonemes in the TIMIT dataset.

Ali et al. describe an approach used to segment and classify phonemes from continuous speech [5]. They group phonemes into four broad categories: sonorants, fricatives, stops, and silences. Utilizing the TIMIT dataset, they achieved 92% accuracy overall.

More recently, Graves et al. take advantage of neural network advances to further improve performance of phoneme classification [6]. In their paper, they describe the implementation of a deep long short-term memory recurrent neural network that classifies phonemes using the TIMIT dataset. While all 61 phonemes were used in training, they mapped their data to 39 distinct classes for testing. They showed that they were able achieve an overall accuracy score of 82.3%, surpassing the performance of the previous best method, deep feed-forward networks.

We decided to use the TIMIT dataset for our phoneme classification research both because of its widespread adaptation in academia for ASR-related problems and because of its carefully and clearly defined labels. The TIMIT dataset provides the exact start and end times of the phonemes in the recordings. This is a huge help as it removes the need for us to do novelty detection or transcribe them by hand, both of which would have introduced a lot of sources of error, hindering our ability to assess the accuracy of our phoneme classification methods.

For our project, we decided to compare the performance of four different classification methods: k-nearest neighbor (KNN), support vector machines (SVM), random forest, and a neural network. We chose to include KNN due to its simplicity. While it is unlikely to perform the best, it gives us a good baseline to compare against our more sophisticated classification algorithms. Neural networks were an obvious choice since its performance has been successfully demonstrated in a handful of ASR research papers. We also included SVM and random forest because they are popularly used in other audio classification tasks; the rationale is that if they work well with one type of audio signal, they should also perform at least decently well on speech.

Furthermore, we are interested in comparing the performance of these algorithms on different size classification problems. While there are 61 distinct phonemes in the TIMIT dataset, many of them have similar characteristics, making it difficult to distinguish between them. We also want to look at how our algorithms perform when subjected to more inclusive phoneme classes. Using phoneme groups instead, we can assess how much error from the original phoneme classification problem was due to mismatching similar phonemes. For example, if an algorithm often mistakes "S" sounds with "Z' sounds, it is a much more forgivable error than mistaking a vowel sound for a silence.

Lopes and Perdigao list several different ways phonemes have been grouped together in previous research projects [1]. We decided to use three different popular groupings, two described by Halberstadt and one as described in Scanlon et al., in addition to the 61 ungrouped phonemes [7] [8]. One of the two phoneme groupings taken from Halberstadt's thesis contains 6 different groups of phonemes, while the other broader grouping only contains 3. The grouping taken from Scanlon et al. contains 5 phoneme groups.

3 Methods

The TIMIT dataset contains phonetically-rich speech data from 630 speakers from 8 different American English dialect regions. It also includes transcriptions with words and phonemes for each sample. Using the phoneme transcriptions, we first separated each phoneme in the audio data. Overall, there were 177080 phonemes in the training set and 64145 phonemes in the test set.

Next, each phoneme audio sample was partitioned and windowed into frames of 256 samples (approximately 16 ms) with 50% overlap. We settled on these parameters after calculating the minimum, maximum, standard deviation, and average phoneme lengths from our dataset. We wanted to ensure that our window length was small enough that even the shortest phonemes would be split into multiple windows so that we can derive ΔMFCC and $\Delta\Delta$MFCC's from them. However, the shortest phonemes in the dataset were prohibitively short. In order to accommodate those outliers, we would have had to reduce our window size to 8 samples, which would have made processing our dataset take far too long as well as raising other questions about whether the window is too small to extract temporal features of the audio signal. We settled on 256 samples since it was a fraction of the length of most phonemes and still allowed us to process the data in a reasonable amount of time.

The rest of the parameters used to calculate the MFCCs were derived using a $\mu+\lambda$ evolutionary algorithm. Due to processing time constraints, we only ran 10 generations on a population of 21 randomized parameter vectors. At each stage of the algorithm, the fitness of a set of parameters was calculated by creating the MFCCs for a subset of our training data and testing their overall accuracy using KNN. The least fit parameter sets were replaced by new parameter sets either through random mutation or crossover. Random mutation altered the parameters by +/- 10% of its current value, while crossover took its parameters directly from two randomly selected parents.

After 10 generations, the most fit parameters were selected to be used as the parameters to create our MFCCs for the entire training and test sets. The parameters the algorithm settled on were as follows:

- Minimum frequency: 50 Hz
- Maximum frequency: 7000 Hz
- Number of Mel filters: 40
- Number of MFCC's: 12
- FFT length: 8192 samples

MFCC, ΔMFCC, and $\Delta\Delta$MFCC parameters were calculated over the frames, and the means and standard deviations of these parameters were calculated for each phoneme. Overall, there were 72 features for each phoneme.

For each of our algorithms, we first trained them using the extracted features from a random sample of training set. The random sample needed to be small enough to minimize the program execution time in MATLAB while also providing enough samples. It was chosen to be 20000 samples, just above 10% of the total dataset.

KNN was set up to find the nearest neighbor, with k set to 1. While increasing the value of k might have led to better performance from the algorithm, we decided not to experiment with other k values because running KNN on our dataset was very slow. Instead of running it multiple times with different k values to see which worked best, we decided to spend that time exploring the other more promising algorithms.

The SVM classifier used default MATLAB parameters with a one-vs-one strategy, linear kernel function, and Iterative Single Data Algorithm (ISDA) for the optimization routine. While the kernel function or optimization routine could be tuned to better suit our data, we kept the default settings for simplicity.

The random forest algorithm was trained using the training dataset with 150 trees. For testing purposes, the predicted label was chosen as the label that had the highest confidence level, irrespective of how high or low it was or how close the second most confident label was to it.

The neural network used for this work was simply made up of an input layer with 72 units for our features, a hidden layer with 512 units and sigmoid activation, and an output layer with number of units equal to the number of classes (ie. 61 when using all phonemes) and softmax activation. The number of hidden units was originally 256 but had better results when tested with 512. The number could be tested over several more values to find better results, but we were satisfied with its performance. The Adam optimizer with a learning rate of 0.001 was used since it is very widely used. Finally, we trained the neural network over 25 epochs with a batch size of 100. These values allowed the accuracy to converge.

We then ran the testing set on each of the algorithms and recorded the predicted labels. Then, we matched those predicted labels to the actual labels and returned a confusion matrix as well as the overall accuracy score. The confusion matrix was normalized by the sum of actual phonemes (rows in the matrix) to see how accurate the classification was for each phoneme.

We repeated the process again for each of the different phoneme groupings. Before providing the training data to the algorithm, both the training data and the test data were relabeled by mapping the phoneme into the correct group. The algorithms were then trained and tested on the relabeled data to obtain accuracy scores for those phoneme groups.

4 Results

The overall accuracy of KNN for classification on phonemes was only about 30%. Figure 1 shows the confusion matrix for KNN, and it shows that phonemes were often misclassified as silence ($h\#$). There were also a few similar-sounding phonemes that were misclassified such as z misclassified as s. Figures 2a and 2b show the confusion matrices from KNN classification using the groupings suggested by [7]. These groupings have overall accuracies of 73.5% and 81.7% respectively. In the first grouping, weak fricatives are often misclassified as closures, and nasals and stops have low accuracy as well. Figure 3 shows the confusion matrix from KNN classification using the groupings suggested by [8]. This grouping has an overall accuracy of 74.6%, and the vowels and silences were the most accurately estimated sounds.

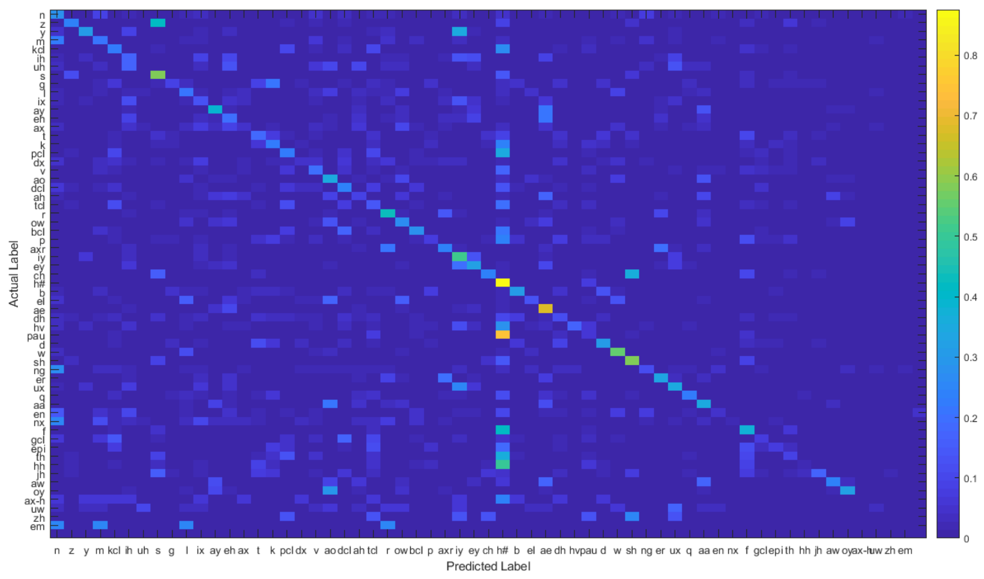
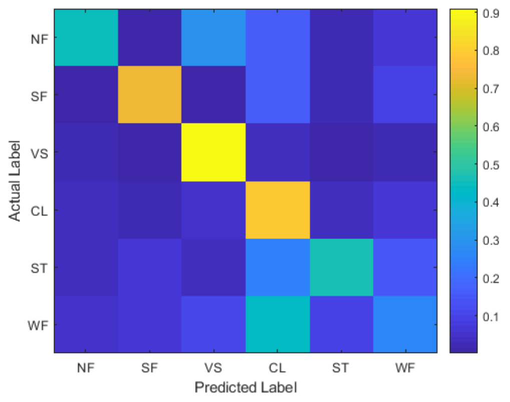
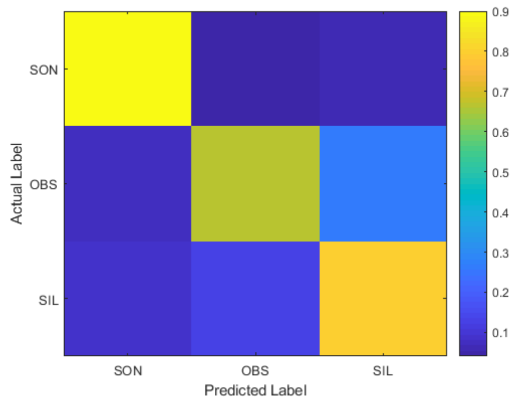


Figure 1: Confusion matrix using KNN, all phonemes



(a) 6 phoneme groups



(b) 3 broader groups (Sonorant, Obstruent, Silence)

Figure 2: Confusion matrix using KNN, groups from [7]

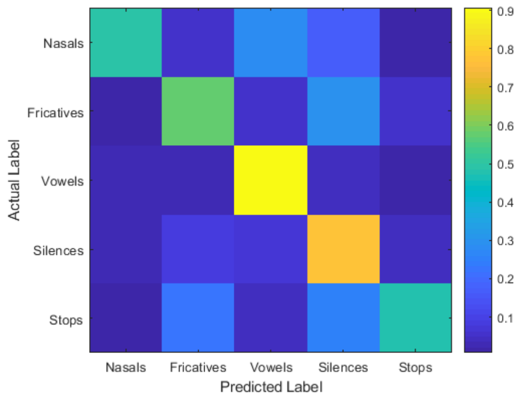


Figure 3: Confusion matrix using KNN, 5 groups [8]

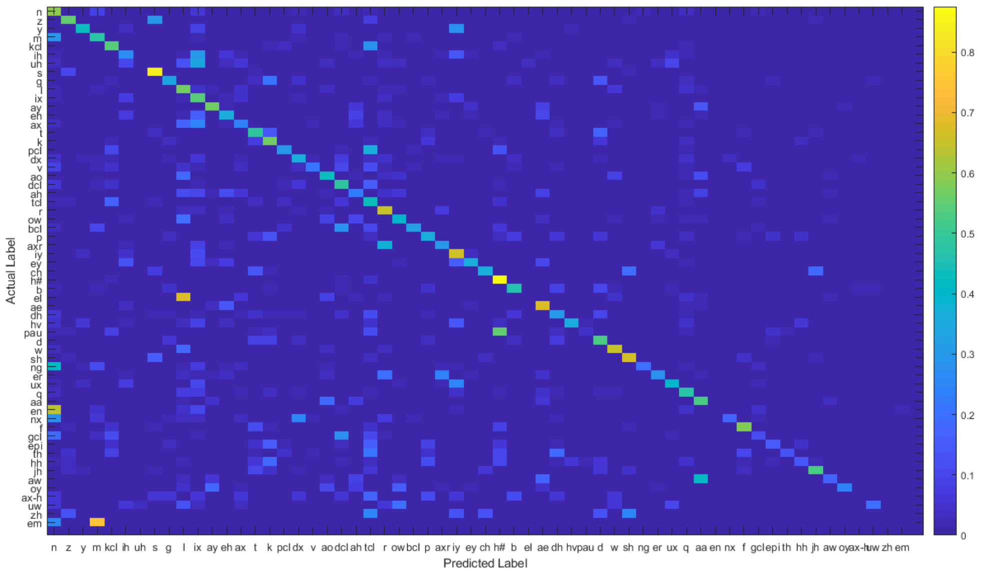
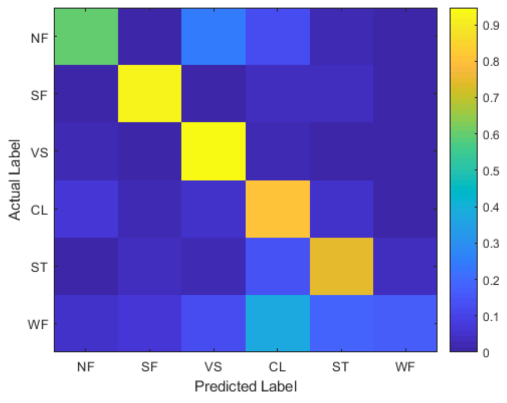


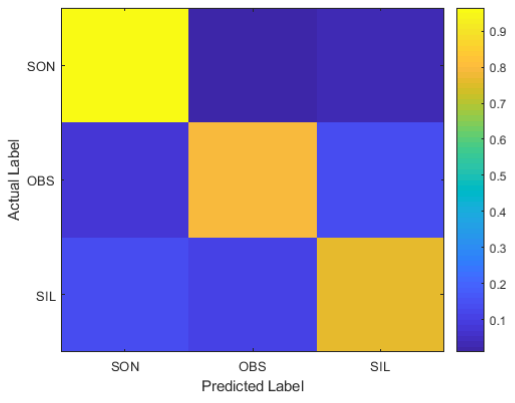
Figure 4: Confusion matrix using SVM, all phonemes

Figure 4 shows the confusion matrix from SVM classification on all of the phonemes. There is an overall accuracy of 48.8%, and the diagonal of the confusion matrix is slightly more defined than in the KNN in Figure 1. However, there are worse misclassifications in phonemes such as l for el and m for em. Figures 5a and 5b show the confusion matrices from SVM classification using the groupings from [7]. Both are similar to the KNN case in 2a and 2b but with better overall accuracies of 81.5% and 87.7% respectively. Figure 6 shows the confusion matrix of SVM classification using the grouping from [8]. Again, it is very similar to the results using KNN but has an overall accuracy of 81.5%.

Using the random forest method gives similar results compared to SVM. Figure 7 shows the confusion matrix for classification on all phonemes using random forests. The overall accuracy is 47.3%, and there are several common misclassifications like iy for y and l for el. Figures 8a and 8b show the confusion matrices for random forest classification using the groupings from [7]. There are again errors in identifying nasal sounds and weak fricatives, and these have overall accuracies of 82.4% and 89.6% respectively. Figure 9 shows the confusion matrix for random forest classification using the grouping suggested in [8]. Most errors occur when



(a) 6 phoneme groups



(b) 3 broader groups (Sonorant, Obstruent, Silence)

Figure 5: Confusion matrix using SVM, groups from [7]

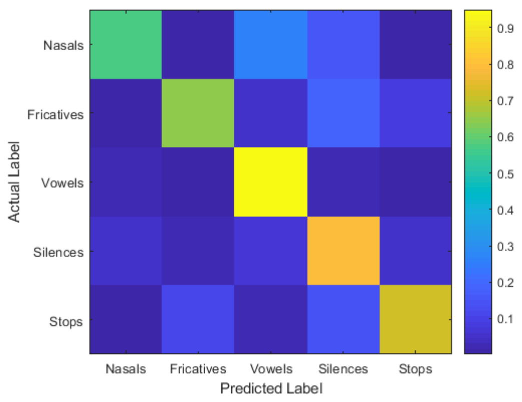


Figure 6: Confusion matrix using SVM, 5 groups [8]

classifying the nasal sounds as vowels, but the overall accuracy is 83.5%.

Finally, the artificial neural network gives the most improved performance. Figure 10 shows the confusion matrix from neural network classification using all phonemes. With an overall accuracy of about 59%, most phonemes are classified correctly, and there is a strong diagonal in the confusion matrix. However, there are still many misclassified phonemes, especially with "ng" mistaken for "eng". Figures 8a and 8b show the confusion matrices using a neural network with groupings from [7], and these have overall accuracies of about 88% and 92% respectively. Figure 12 shows the confusion matrix using a neural network with groupings from [8], and it has an overall accuracy of about 88% as well.

5 Discussion

Overall, the different classification methods worked about as well as expected. Table 1 shows the overall accuracies for each classification method and grouping. KNN has a rather poor performance. While it is

	Overall Accuracy			
Classification method	All phonemes	6 groups [7]	3 groups [7]	5 groups [8]
KNN	29.9%	73.5%	81.7%	74.6%
SVM	48.8%	81.5%	87.7%	81.5%
RF	47.3%	82.4%	89.6%	83.5%
NN	∼59%	∼88%	∼92%	∼88%

Table 1: Comparison of overall accuracy for different classifiers and groupings

the simplest to implement, it only yields an overall accuracy of about 30% using all the phonemes. Many samples are also misclassified as silence ($h\#$). This could be due to over-fitting since silence appears at the beginning and end of every audio sample. Using a random sample with a more equal distribution would most likely help with this issue. For the different groupings, KNN is decent but not great. For the 6 groups of [7], weak fricatives are very poorly classified and more often misclassified as closures.

SVM and random forests improve the model to just under 50% overall accuracy using all phonemes. Both of these methods perform similarly to each other and have several different specific misclassified phonemes that have similar sounds. While the overall accuracy is better than the KNN model, these models have trouble distinguishing these specific sounds. As for the groupings, both make similar errors as KNN but improve the overall accuracy. Weak fricatives in [7] are poorly classified for both, but SVM has better accuracy when classifying nasal sounds. For these methods, different parameters could be changed, such as the kernel function in SVM or the number of trees in the random forest, to better tune our system for the data. Compared to KNN, these methods are much more computationally intense and take longer to execute in MATLAB. Depending on desired classification (ie. all phonemes or groupings) SVM and random forests

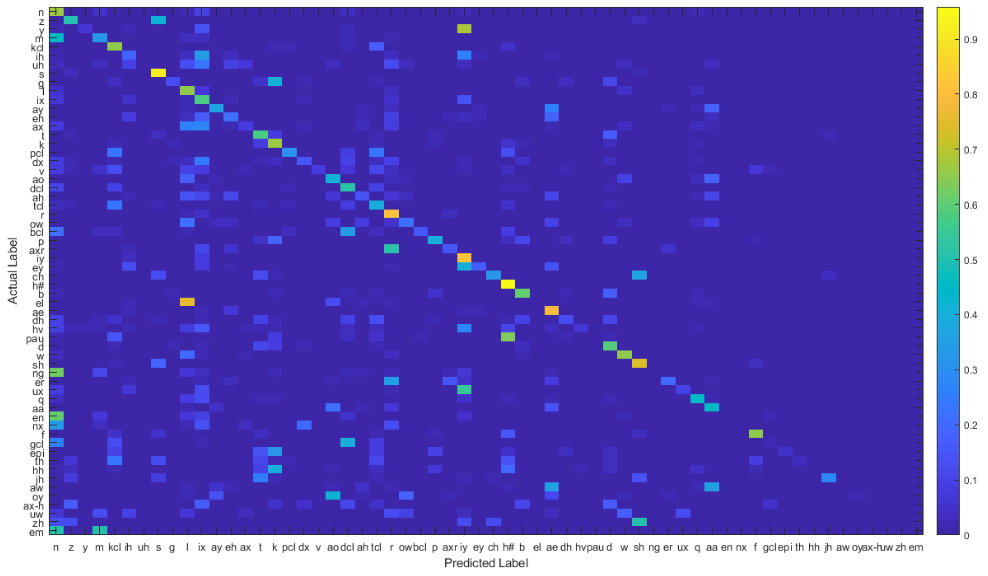
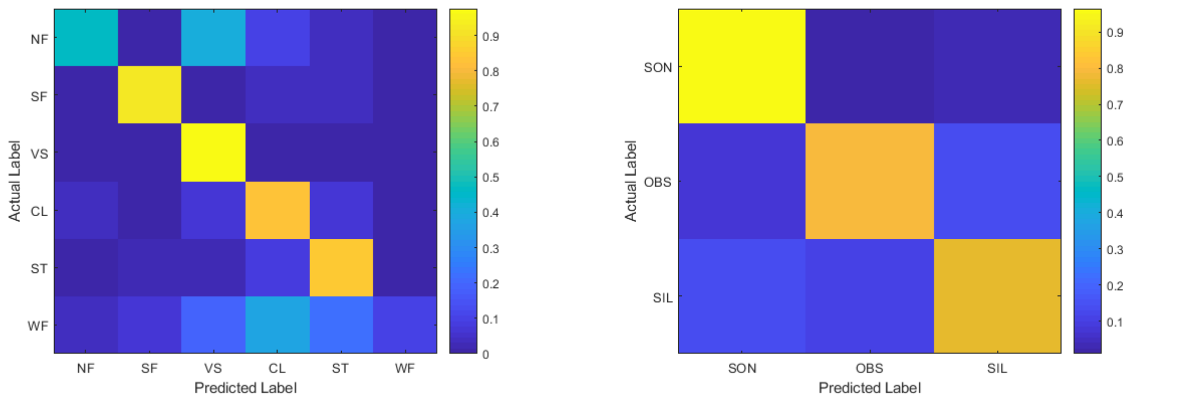


Figure 7: Confusion matrix using random forest, all phonemes



(a) 6 phoneme groups

(b) 3 broader groups (Sonorant, Obstruent, Silence)

Figure 8: Confusion matrix using random forest, groups from [7]

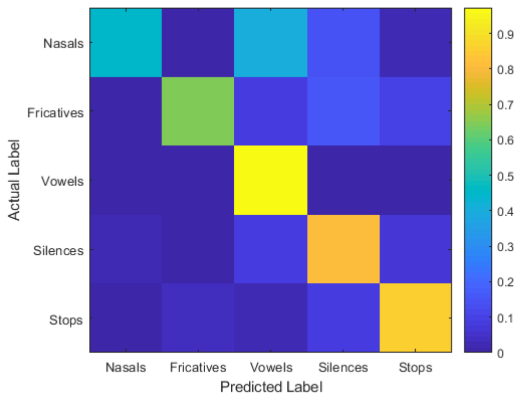


Figure 9: Confusion matrix using random forest, 5 groups [8]

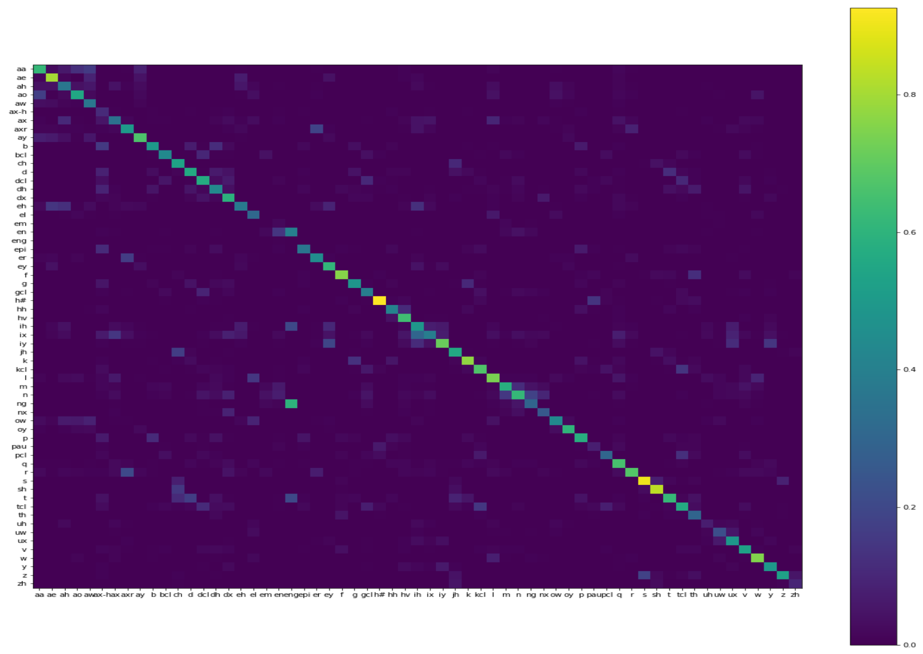
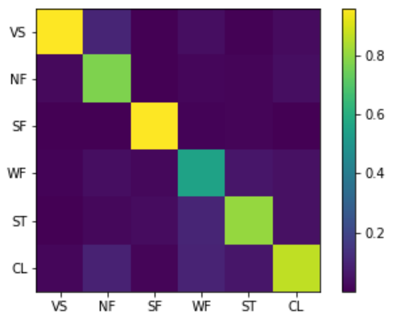
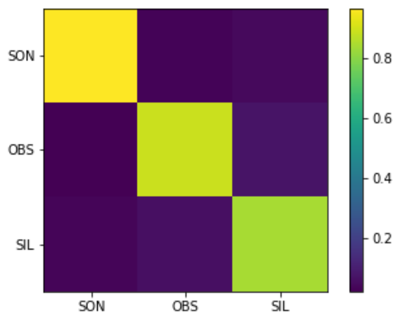


Figure 10: Confusion matrix using a neural network, all phonemes



(a) 6 phoneme groups



(b) 3 broader groups (Sonorant, Obstruent, Silence)

Figure 11: Confusion matrix using a neural network, groups from [7]

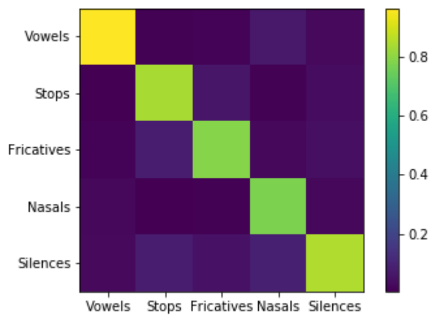


Figure 12: Confusion matrix using a neural network, 5 groups [8]

might not be the best options because of the computational cost.

The neural network classifier performs the best out of all of the tested methods. For all 61 phonemes, there is an overall accuracy of about 59%. While it is not fantastic, it is still correct a majority of the time. Early work with neural networks in [4] resulted in about 69% accuracy. With some better tuning of parameters, it is possible that we could improve our neural network performance. There are still plenty of misclassified phonemes using this method, and there is one phoneme that is more often misclassified (n as en). Using the different groupings, the neural network reached values close to about 90% accuracy. While our classifiers are not perfect, there are parameters that could be changed to improve on these results.

6 Conclusions and Future Work

Our neural network consistently outscored all other classification methods. KNN was simple to implement but yielded poor performance results. SVM and random forests performed similarly in between the other two methods.

All of the tested methods make misclassification errors between similar sounding phonemes. KNN often mistakes phonemes for silence ($h\#$) while the other methods often have trouble distinguishing sounds like en and n. Intuitively, it was also expected that using fewer phoneme groups would result in an increase in accuracy. When put into broader categories, the issues with distinguishing certain sounds becomes irrelevant.

While our results are not as accurate as some other similar methods [4, 5, 6], there are many different ways to improve our classifiers. The MFCC parameters could be altered by using a wider frequency bandwidth, more Mel filters, or changing the window size. Our parameters were chosen to minimize the processing time since TIMIT is a large dataset. We could also improve the ΔMFCC and ΔΔMFCC functions or increase the total number of parameters to get more information from the phonemes. Our random sample of data could be more equal in the distribution of phonemes to get better results, especially when using KNN. There are even different groupings that we could test. Some groupings might be better suited for different applications. Finally, there more advanced methods, such as convolutional neural networks, that are likely to improve performance.

References

[1] C. Lopes and F. Perdigao, "Phoneme recognition on the timit database," 2011.

[2] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," 1989.

[3] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," 1987.

[4] A. Robinson and F. Fallside, *Phoneme recognition from the TIMIT database using recurrent error propagation networks.* University of Cambridge, Department of Engineering, 1990.

[5] A. A. Ali, J. van der Spiegel, and P. Mueller, "An acoustic-phonetic feature-based system for automatic phoneme recognition in continuous speech," 1999.

[6] A. Graves, A. rahman Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," 2013.

[7] A. K. Halberstadt, "Heterogeneous acoustic measurements and multiple classifiers for speech recognition," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[8] P. Scanlon, D. P. Ellis, and R. B. Reilly, "Using broad phonetic group experts for improved speech recognition," *IEEE transactions on audio, speech, and language processing*, vol. 15, no. 3, pp. 803–812, 2007.