Comparison of Logistic Regression, Neural Networks, and Support Vector Machines for Speech Recognition

Lauren Funk, Suhas Rao Cheeti, Natali Thiesse, Thomas Toaz
Michigan State University, Computer Science and Engineering (CSE)
funklau1@msu.edu, cheetisu@msu.edu, thiesse4@msu.edu, toazthom@msu.edu

Abstract

This document is looking into a dataset of audio files from Pete Warden and how to improve the machine learning models used. While the previous project [1] uses a Convolutional Neural Network (CNN), we used the simpler machine learning model of Logistic Regression (LR) and Support Vector Machine (SVM) to possibly make a more accurate model and followed similar steps to make a Neural Network to compare the results of the models, especially in how accuracy and speed changed. Data preprocessing involved extracting Mel-Frequency Cepstral Coefficients (MFCCs) and standardizing inputs for consistency. Initial findings indicated that the tuned logistic regression model outperformed the untuned version across precision, recall, F1 score, and accuracy. The subsequent phase involves evaluating the Convolutional Neural Networks (CNNs), Logistic Regression (LR) and Support Vector Machines (SVMs) model's performance against these metrics to assess its effectiveness. This research contributes to optimizing speech recognition systems, with implications for improving accessibility and overcoming language barriers through efficient and accurate keyword detection. In the end, CNNs had the largest accuracy with accuracy possibly varying with more refined and explicit data in other models.

1 Introduction

In a previous project using a dataset of audio files, a convolutional neural network was trained on spectrograms generated from audio recordings, which were then used to classify and detect when specific words/commands were said. Our initial model used a logistic regression model for the binary classification of keyword detection. We then created our own convolutional neural network model, as recommended and used in previous experiments and projects using this dataset. This neural network is recommended because it excels in image and pattern recognition, and by generating stereographs from the audio files we can then easily train it for recognizing words based on their generated images. We also created a Support Vector Machine (SVM) with a nonlinear kernel. The SVM model is good with generalizations, and the nonlinear kernel is used as our dataset is not linearly separable. We initially thought that the logistic regression model would outperform the neural network and SVM due to its simplicity.

2 Project Importance

As mentioned previously, this project will be using a logistic regression model, neural network, and support vector machine (SVM). This project differs from others as we also incorporated a SVM model to further compare model results. Our model pipeline for this project is simplified as we are using a data set where the data is already collected and will be easily modified to pass to the model. Speech recognition has many real-world applications. Most notably is how speech recognition creates better accessibility. People with disabilities who may struggle to type or create written text can now create written items by talking to the machine. Speech recognition can also help mend language barriers. For example, if two people are trying to communicate in different languages, speech recognition software can be used to translate what the other is saying, providing more effective communication.

3 Related Works

There are existing projects and research associated with our specific dataset [1], as well as similar speech command datasets. The research done with the dataset we are using did an analysis with deep learning. They used TensorFlow and then calculated the canonical Top One error based on whether the model correctly identified the word sample.

A notable paper used convolutional neural networks to classify spoken words based on spectrograms generated from audio recordings and compared the results to a deep neural network model [2]. They found that using a CNN had a significantly higher classification rate. When they ran a test that limited the parameters used by the model, CNN had improvements of over 41% relatively compared to the DNN model. They also found that CNN had 29% improvement over DNN when limiting the multiples used. This shows that there can be models that are smaller in size, but just as powerful, if not more at classifying spoken words.

Another article analyzes how Alexa, Amazon's cloud base AI, uses speech recognition to function [3]. The main idea is the model takes spoken words, converts them into a spectrogram, dissects the spectrogram into different features, and utilizes a deep learning model to identify the words being spoken. Alexa uses a convolutional neural network as a feature extractor and stacks recurrent neural network layers on top to process the input sequence [4].

4 New Perspective

We are going to be utilizing a different method when diving into our research. Our project will explore a much simpler approach, using logistic regression for binary classification of keyword detection. Then creating a convolutional neural network and a support vector machine (SVM) model. By reducing how complex of a model we're using, our goal is to improve understanding and efficiency, while maintaining high accuracy in our findings.

5 Our Dataset

For this project, we chose a speech command data set. The data in the data set was already collected and unintelligible words were filtered out. The data consists of twenty common English words shown in Table 1 and were restricted to being one second or less. There were 2,618 recorders with varying accents. Recorders were asked to record in a room without background conversations for privacy reasons and use a laptop or phone microphone. Each sample is formatted as a WAVE file. To use the data set in this project, we will do feature extraction on each sample before passing it to the model. This project will only be using the words 'stop', 'go', 'up', 'down', 'left', and 'right' for simplicity reasons.

	Label	Count
0	_background_noise_	6
1	bed	1713
2	bird	1731
3	cat	1733
4	dog	1746
5	down	2359
6	eight	2352
7	five	2357
8	four	2372
9	go	2372
10	happy	1742
11	house	1750
12	left	2353
13	marvin	1746
14	nine	2364
15	no	2375
16	off	2357
17	on	2367
18	one	2370
19	right	2367
20	seven	2377
21	sheila	1734
22	six	2369
23	stop	2380
24	three	2356
25	tree	1733
26	two	2373
27	up	2375
28	wow	1745
29	yes	2377
30	zero	2376

Table 1: Number of each recorded word

6 Methodology

6.1 Data Preprocessing and Feature Extraction

We use the deeplake library to load pre-labeled audio files from activeloop.ai. To focus only on relevant commands, we filter out classes that are not among our target labels. Using Librosa, we extract Mel-Frequency Cepstral Coefficients (MFCCs) from the audio files, which serve as key speech features. To ensure uniform input dimensions, we standardize MFCCs to a fixed length of 100, truncating longer samples and padding shorter ones with zeros. When flattened, this results in feature sets of 1300. Finally, we split the dataset using Sklearn into 80% training data and 20% testing data for model evaluation. We then create an additional training and testing data set using Sklearn's PCA function, reducing the feature dimensions as much as we can while maintaining 95% of the information contained within them, which results in feature sets of dimensions 139.

6.2 Model Setup

For our base logistic regression model, we use the Sklearn logistic regression model with maximum iterations set to 100,000. For the tuned logistic regression model, we perform additional data cleaning after feature extraction. MFCCs are scaled and normalized using Sklearn's StandarScaler. SMOTE is then used to create synthetic samples for underrepresented classes, helping to balance out the class distribution. We then use Sklearn's RandomSearchCV with hyperparameter tuning, using regularization strengths 0.01, 0.1, 1, and 10, solvers newton-cg, sag, saga, and liblinear, penalties L2 and none, max iteration set to 1000, random state 42, class weight balanced, number of iterations 6, cross validation folds 3, and number of jobs -1 to use all CPU cores. The best parameters were found to be sag for the solver, L2 as the penalty, and 0.01 as the regularization strength, which we then used to fit our model.

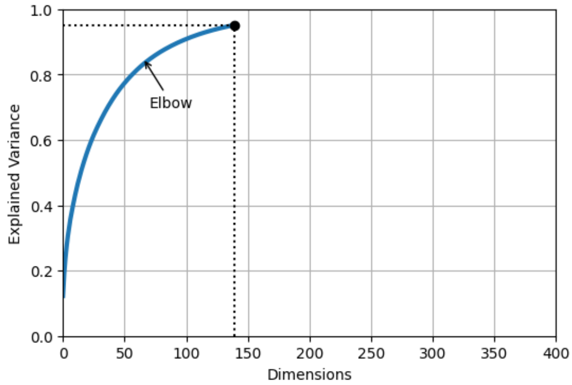

Figure 1: PCA Variance Over Dimension Reduction Size

Our convolutional neural network model used the same additional data cleaning as the tuned logistic regression model. It has four layers: an input layer with 32 neurons, a kernel size of 3, and Rectified Linear Unit (ReLU) as the activation function, two hidden layers, one for flattening, the other a dense layer with 64 neurons and ReLU as the activation function, and a dense output layer that outputs the class. The model is compiled using Adam as the optimizer, sparse categorical cross entropy for loss, and accuracy as the metric, is fit using 100 epochs.

Our final model is an SVM, using the same data preprocessing as the tuned logistic regression model and the convolutional neural network. We used Sklearn's GridSearchCV with a StandardScaler and SVC pipeline, 0.1, 1, and 10 regularization strengths, an rbf kernel to detect nonlinear patterns, gamma values of scale, 0.01 and 0.001, and set scoring to accuracy. The best parameters were found to be 10 as the regularization strength and scale as the gamma value, which we then used to fit our model.

7 Results

We are evaluating our models based on four basic metrics: precision, recall, accuracy, and F1 score. Precision of a model is a metric that measures the proportion of true positive predictions out of all the occurrences predicted as positive, true or false. Recall measures how frequently our model can correctly identify true positives, out of all actual positives in our dataset. Accuracy simply measures how often our model's predictions are correct or accurate. Finally, the F1 score is the last metric we will use which combines precision and recall, outputting a single value to evaluate the model on how well it performs. We will compare these metrics between our simple, untuned logistic regression model, tuned logistic regression model, our convolutional neural network, and our SVM. We will also compare these metrics between our tuned logistic regression model, convolutional neural network, and SVM after applying feature reduction to the data.

Untuned Logist Regression Model Performance:

	precision	recall	f1-score	support
down	0.59	0.61	0.60	472
go	0.60	0.60	0.60	475
left	0.58	0.56	0.57	471
right	0.67	0.65	0.66	473
stop	0.70	0.70	0.70	476
up	0.54	0.54	0.54	475
accuracy			0.61	2842
macro avg	0.61	0.61	0.61	2842
weighted avg	0.61	0.61	0.61	2842

Table 2: Untuned Logistic Regression Classification Report

After running our untuned logistic regression model, we received the scores for each of these metrics, for each of the four words that we are including. The precision of the model for each word ranged from 54% to 70%, with an average score of 62%. The recall of the model similarly ranged from 54% to 70% as well, with an average of 62%. The F1 score ranged from 54% to 70%, with an average score of 62%. Finally, the overall accuracy of the model was 61%.

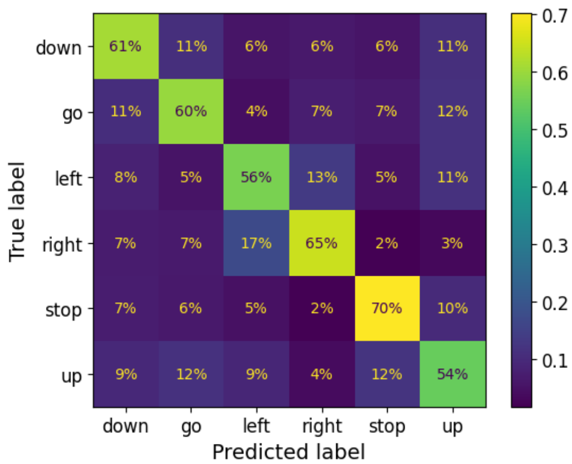

Figure 2: Untuned Logistic Regression Model Confusion Matrix

For the untuned logistic regression model, the word that the model performed best with matching the true label to the model's prediction was 'stop', accurately predicting it 70% of the time. Following 'stop' was the words 'right', down', 'go', 'left', and finally, being the least accurate with 'up'. Additionally, the model most confused the word 'right' with 'left', which occurred 17% of the time the model encountered the word 'right' in the dataset.

Tuned Logist RegressionModel Performance:

	precision	recall	f1-score	support
down	0.64	0.62	0.63	476
go	0.64	0.63	0.63	476
left	0.61	0.59	0.60	476
right	0.69	0.68	0.69	476
stop	0.72	0.73	0.73	476
up	0.58	0.62	0.59	476
accuracy			0.64	2856
macro avg	0.64	0.64	0.64	2856
weighted avg	0.64	0.64	0.64	2856

Table 3: Tuned Logistic Regression Classification Report

After running our tuned logistic regression model without feature reduction, we received these same metrics to score our model. The precision for each word ranged from 58% to 72%, with an average of 65%. The recall score ranged from 59% to 73% for all four words, with an average of 64%. The F1 score of the model ranged from 59% to 73%, with an average score of 64%. Finally, the overall accuracy of the model was 64%.

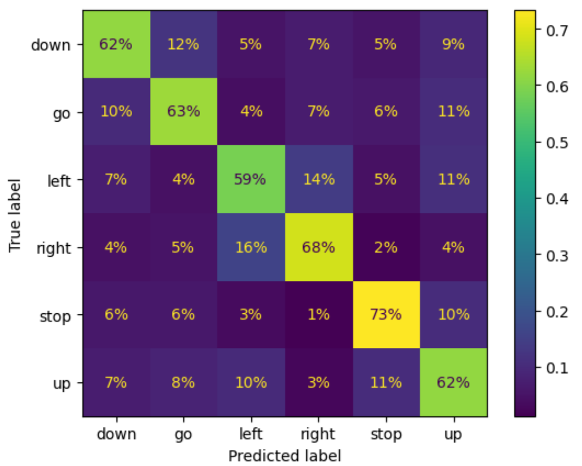

Figure 3: Tuned Logistic Regression Model Confusion Matrix

Before applying feature reduction to the data, the word that the tuned logistic regression model most accurately predicted was 'stop', similarly to the untuned logistic regression model. It accurately matched the predicted label to the true label 73% of the time. The words in order of the model's best performance are 'stop', 'right', 'go', 'up, 'down', and finally, ‘left’. In a similar manner as the untuned logistic regression model, both models most frequently confused the word ‘left’ with ‘right’ as well.

Tuned PCA Logistic RegressionModel Performance:

	precision	recall	f1-score	support
down	0.65	0.61	0.63	476
go	0.65	0.63	0.64	476
left	0.61	0.60	0.60	476
right	0.68	0.68	0.68	476
stop	0.73	0.73	0.73	476
up	0.57	0.62	0.60	476
accuracy			0.65	2856
macro avg	0.65	0.65	0.65	2856
weighted avg	0.65	0.65	0.65	2856

Table 4: PCA Dimension Reduction Tuned Logistic Regression Model Classification Report

Then, applying dimension reduction to the data, our tuned logistic regression model scored differently. The precision of the model ranged from 57% to 73%, with an average score of 65%. The recall score had ranged from 60% to 73%, with an average recall score of 65%. The F1 score of the model ranged from 60% to 73%, with an average score of 65% as well. The overall accuracy of the model after applying feature reduction to the data was 65%.

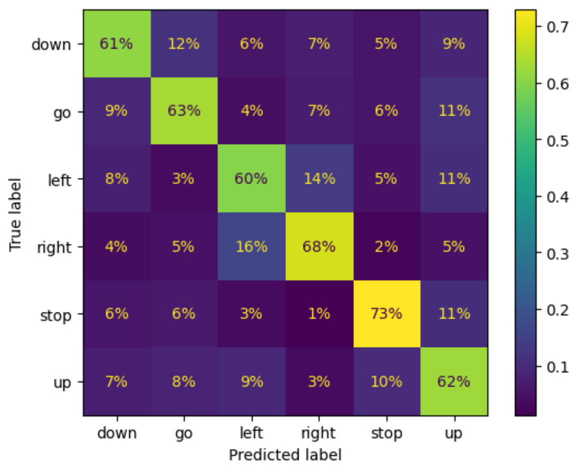

Figure 4: PCA Dimension Reduction Tuned Logistic Regression Model Confusion Matrix

After applying dimension reduction to the data, we are not able to observe many key differences in terms of which words the model was able to predict most accurately. The word with the highest percentage of predicted labels to true labels remained ‘stop’. The following words in order of how frequently the model correctly predicted them is also the same. The only major differences fall within the words ‘left and ‘down’. After the further tuning, the percentage of predicted label matched to the true label for ‘left’ went up from 59% to 60%, and for ‘down’ went up from 61% to 62%.

Convolutional Neural Network Model Performance:

	precision	recall	f1-score	support
0	0.82	0.83	0.82	476
1	0.85	0.74	0.79	476
2	0.77	0.82	0.79	476
3	0.90	0.86	0.88	476
4	0.92	0.86	0.89	476
5	0.73	0.85	0.79	476
accuracy			0.83	2856
macro avg	0.83	0.83	0.83	2856
weighted avg	0.83	0.83	0.83	2856

Table 5: Convolutional Neural Network Classification Report

After running our convolutional neural network model, we received these same metrics to score our model. The precision of the model for each word ranged from 77% to 92%, with an average score of 84%. The recall of the model similarly ranged from 74% to 86%, with an average of 80%. The F1 score ranged from 79% to 89%, with an average score of 84%. Finally, the overall accuracy of the model was 83%.

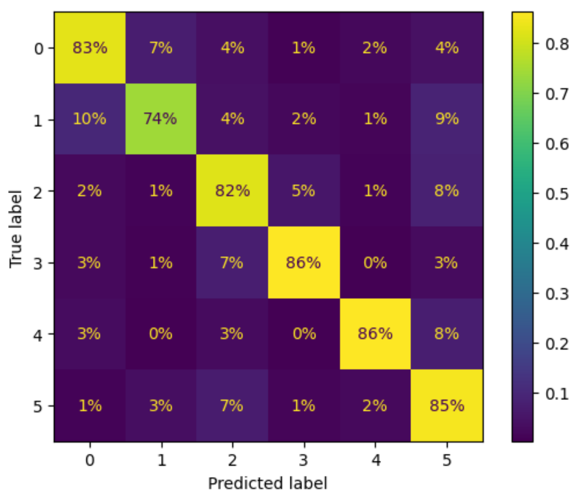

Figure 5: Convolutional Neural Network Confusion Matrix

Since the convolutional neural network can only have integer input, the labels were encoded as integers. The corresponding labels for Figure 5 in the range 0-5 are as follows: ‘down’, ‘go’, ‘left’, ‘right’, ‘stop’, and ‘up’. The word that the model was able to correctly match the predicted label with the true label most frequently was ‘stop’, similar to the tuned logistic regression model, with an accuracy of 86%, tied with ‘right’. The next word would be ‘left’, with an accuracy of 82%, and then ‘up’, ‘down’ and ‘go’. The two words that the model most frequently confused with one another was ‘down’ and ‘go’.

Convolutional Neural Network Model Performance With PCA:

	precision	recall	f1-score	support
0	0.87	0.84	0.85	476
1	0.87	0.88	0.87	476
2	0.88	0.87	0.87	476
3	0.91	0.90	0.91	476
4	0.90	0.91	0.90	476
5	0.84	0.85	0.85	476
accuracy			0.88	2856
macro avg	0.88	0.88	0.88	2856
weighted avg	0.88	0.88	0.88	2856

Table 6: PCA Dimension Reduction Convolutional Neural Network Classification Report

After running our convolutional neural network while using dimension reduction, the metrics differed. The precision for each word ranged from 84% to 91%, with an average precision score of 88%. The recall score ranged from 84% to 91%, with an average score of 88%. The F1 score for the model ranged from 85% to 91%, averaging 88%. The overall accuracy of the model after using the PCA dimension reduction technique was 88%.

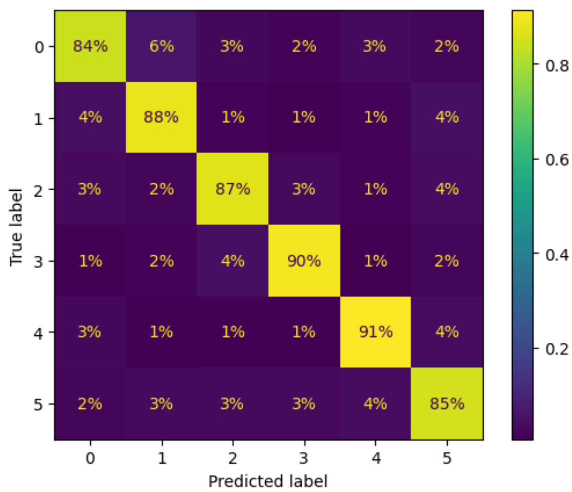

Figure 6: PCA Dimension Reduction Convolutional Neural Network Confusion Matrix

After running the convolutional neural network while applying dimension reduction to the data, the model performed much better in terms of matching the predicted labels to the true labels. Again, the corresponding labels for Figure 6 in the range 0-5 are as follows: ‘down’, ‘go’, ‘left’, ‘right’, ‘stop’, and ‘up’. The model once again performed best with the word ‘stop’, increasing the accuracy percentage from 86% up to 91%. The model also performed well with the rest of the words, increasing each accuracy, with the greatest increase being with the word ‘go’, which increased from 74% up to 88%.

SVM Performance:

	precision	recall	f1-score	support
down	0.87	0.86	0.86	476
go	0.89	0.87	0.88	476
left	0.87	0.88	0.88	476
right	0.90	0.91	0.90	476
stop	0.90	0.90	0.90	476
up	0.83	0.84	0.83	476
accuracy			0.88	2856
macro avg	0.88	0.88	0.88	2856
weighted avg	0.88	0.88	0.88	2856

Table 7: SVM Classification Report

Finally, we ran our SVM, or Support Vector Machine, and received the same metrics to score that model, first without applying feature reduction to the data. The precision of the SVM ranged from 83% to 90% for each of the words, with an average precision score of 88%. The recall score for the model ranged from 84% to 91%, with an average of 88%. The F1 score for the model ranged from 83% to 90%, with an average score of 88%. Finally, the overall accuracy of the model was 88%.

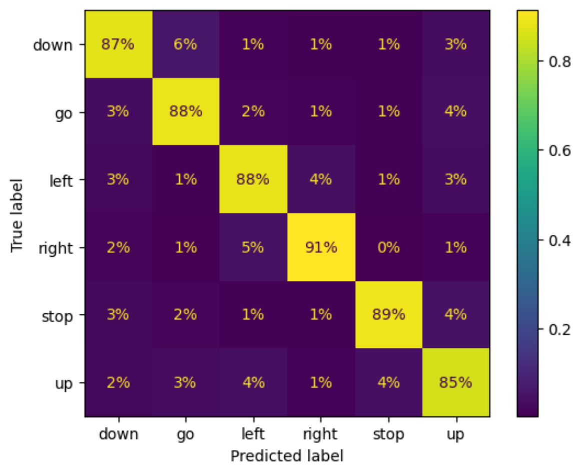

Figure 7: SVM Model Confusion Matrix

After running the simple SVM model, we can observe that the word the model was most accurately able to predict was ‘right’, differing from the two logistic regression models. It correctly predicted ‘right’ 91% of the time. The following words it was able to correctly predict in order of accuracy are ‘stop’, ‘left’, ‘go’, ‘down’ and finally the least accurate being ‘up’ with 85%

accuracy. The word that it most frequently confused the predicted with the true label was predicting ‘go’ when the true word was ‘down’.

PCA SVM Performance:

	precision	recall	f1-score	support
down	0.87	0.86	0.86	476
go	0.89	0.87	0.88	476
left	0.87	0.88	0.88	476
right	0.90	0.91	0.90	476
stop	0.90	0.90	0.90	476
up	0.83	0.84	0.83	476
accuracy			0.88	2856
macro avg	0.88	0.88	0.88	2856
weighted avg	0.88	0.88	0.88	2856

Table 8: PCA Dimension Reduction SVM Classification Report

After using dimension reduction, the SVM metrics scored about the same. The precision of the model ranged from 83% to 90%, with an average precision score of 88%. The recall score ranged from 84% to 91%, with an average recall of 88%. The F1 score for the model ranged from 83% to 90%, with an average of 88%. Finally, the accuracy of the SVM after applying dimension reduction to the data was 88%.

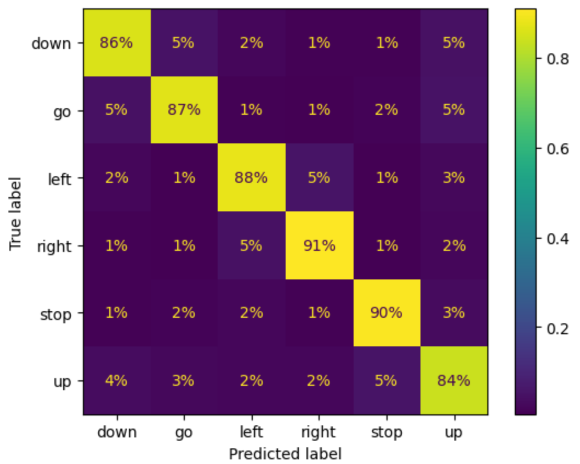

Figure 8: PCA Dimension Reduction for SVM Confusion Matrix

Using PCA dimension reduction in our data, there are some differences within the SVM confusion matrices. The word that the model most accurately predicted remains ‘right’. The order for the words that the model was most accurate at predicting also remained the same, before and after dimension reduction was applied. The key differences are with the words ‘stop’, ‘up’, ‘go’, and ‘down’. The accuracy for the word ‘stop’ increased from 89% to 90%. For the word ‘up’, the accuracy decreased from 85% to 84%. For ‘go’ the accuracy also decreased from 88% to 87%, and similarly with ‘down’ the accuracy decreased from 87% to 86%.

Comparing all four of these metrics we received after running all four of our models without applying PCA dimension reduction to our data set, our tuned logistic regression model outperformed the simple model in all four categories, and our convolutional neural network outperformed both. However, the SVM performed the best out of all four models scoring 88% overall to the convolutional neural network’s 83%. It seems that audio features are more complex than we first expected, with our more advanced models being able to pick up the nuances and patterns between different command phrases much better than the more simplistic logistic regression models, which is not what we were expecting in our initial hypothesis.

Our next steps were to further tune our models to see how close we can get the performance of a tuned logistic regression model to the convolutional neural network and the SVM. We did this by applying PCA dimension reduction to our data in hopes of receiving higher accuracy scores for the models. We tested our models again after using the dimension reduction technique, but only with our tuned logistic regression model, convolutional neural network, and SVM.

Comparing the metrics between our tuned logistic regression model before and after simplifying our data, the model only slightly improved, from 64% to 65%. Our convolutional neural network model outperformed that with a score of 88%, increasing from an original 83%. Our SVM’s accuracy, on the other hand, stayed around the same at 88% as well. From these findings, we can conclude that applying PCA dimension reduction to our data did not assist with increasing the performance of the logistic regression model, nor the SVM. It did, however, greatly increase the accuracy score of our convolutional neural network. The more advanced models once again were able to better identify the patterns in the different commands much better after cleaning our data. Our initial hypothesis, being that the simplicity of the logistic regression model would allow it to outperform more advanced models, is concluded to be incorrect, with the simplicity seeming to be the hindering factor.

8 Further Research

The next steps in this research is to add more preprocessing and cleaning of the data. This would allow the data to be moved into a feature space that generates better classifications. One way this would be done is by putting the data through a non-linear transform. By using a non-linear transform, the accuracy should increase in all the models, especially the logistics regression model as the data is in a space that can be separated into two classes more easily. We could also apply this research to a real-world application, such as making our own speech-to-text software using the model with the best performance. In that scenario, we would record our own dataset incorporating more background noise and variety of recorded voices to create a more well-rounded set of samples.

9 Author Bios

Suhas Rao Cheeti, cheetisu, Computer Science and Engineering (CSE), 3rd Year
Natali Thiesse, thiesse4, Computer Science and Engineering (CSE), 3rd Year
Lauren Funk, funklau1, Computer Science and Engineering (CSE), 4th year
Thomas Toaz, toazthom, Computer Science and Engineering (CSE), 3rd year

10 References

[1] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," Apr. 2018. https://arxiv.org/pdf/1804.03209

[2] T. Sainath and C. Parada, "Convolutional Neural Networks for Small-footprint Keyword Spotting," *Google Inc.*, 2015, Available: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43969.pdf

[3] A. Trivedi, "How Amazon Alexa Works Using NLP," *Analytics Vidhya*, Aug. 25, 2024. https://www.analyticsvidhya.com/blog/2024/08/how-amazon-alexa-works-using-nlp/

[4] M. Sun, "Two new papers discuss how Alexa recognizes sounds," *Amazon Science*, Apr. 18, 2019. https://www.amazon.science/blog/two-new-papers-discuss-how-alexa-recognizes-sounds (accessed Mar. 15, 2025).

[5] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Sebastopol, California: O'Reilly Media, Inc, 2023