A Brief Comparison of Supervised Learning Algorithms and Feature Extraction for Speech Recognition

Mingcan Tang [1] Alexander Guthrie [1]

Abstract

In recent years, learning algorithms, particularly neural networks, have improved significantly and, consequently, become increasingly attractive methods for performing natural language processing and speech recognition. In this paper we consider the combination of convolutional neural networks (CNNs) with supervised learning algorithms for binary speech classification (BSC). CNNs offer a novel method of audio feature extraction compared to the method of Mel-frequency cepstral coefficients (MFCCs). Five different classifiers are trained on the features extracted by each of these methods and their performances are compared. Experimental results show that the SVM classifier generally performs better than the other classifiers we tested and that MFCCs provide more reliable features. We finish with a discussion about the possible causes of these observations.

1. Introduction

In recent years, natural language processing and speech recognition have become services offered with increasing popularity. Arguably, the journey of studying speech recognition in the scientific community started at least 65 years ago, when Bell Lab released Audrey which was capable of recognizing spoken digits with 90 percent accuracy, but only for the speech of its inventors (Boyd). The fact that spoken digit recognition sparked researchers' interests at this time points to the significance of this problem. Now, with more advanced machine learning algorithms we revisit this problem.

Our task of binary speech classification is to classify audio recordings as "zero" or "one". In this paper we compare the performance of 5 standard classifiers applied to this BSC task through two different feature extraction methods. Furthermore, we compare the performance of each classifier over three partitions of training and testing data: 80/20, 50/50, 20/80. These comparison are done 3 times and averaged to remove accidental error.

[1]University of California San Diego, California, USA. Correspondence to: UCSD <ucsd.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

1.1. Research Question

Although neural networks (NNs) were historically developed as a type of regressor and classifier, much like other standard classifiers such as SVM, following the development of CNN, NNs have been very extensively studied independently from other machine learning algorithms. Compared to many standard machine learning algorithms, CNNs clearly have their own distinct advantages. The idea of ensemble learning, where multiple standard classifiers are jointly applied to solve a task, motivates us to combine the advantages of CNN and standard classifiers and to study the performance of this combined classification method as compared to the application of standard classifiers along.

For the purpose of this paper, and due to the limited resources, we limit our scope to consider only 5 standard classifiers in addition to CNN: Ridge Regression Classifier, Support Vector Classifier (SVC), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), and Decision Tree. We use accuracy on the training, validation, and testing dataset as our evaluation metric to analyze the performance.

1.2. Summary of Results

In general, our data shows that MFCC features allow the best BSC task performance using traditional classifiers. Each of the 5 classifiers we tested performs better when trained on the MFCC features than the CNN features. In terms of classifiers, we see that overall the SVM classifier performs best in this BSC task. The KNN and Ridge Regression classifiers trade places for performance in second place based on the training partition. Only in the case of 80% training data can KNN outperform Ridge Regression. For the other classifiers, we see that performance varies greatly over different partitions. We would expect all classifiers to improve their accuracy with more training data, but this is not observed. This unusual result is discussed in more detail in the Experiments section. This summary of observations can be seen in the following figure:

Classifier	Average cross-dataset accuracy	CNN features accuracy	MFCC features accuracy	Average 80/20 Testing accuracy	Average 50/50 testing accuracy	Average 20/80 testing accuracy
SVM	**97.1%**	**94.7%**	**99.5%**	**97.2%**	**97.6%**	**96.5%**
Ridge Regression	95.7%	92.6%	98.9%	95.8%	96.9%	94.4%
KNN	95.3%	91.5%	99.1%	96.4%	96.7%	92.9%
LDA	90.2%	81.7%	98.6%	88.1%	93.2%	89.2%
Decision Tree	89.9%	88.6%	91.1%	90%	91.4%	88.2%

Figure 1. The average performance of classifiers over partitions and feature sets. Notice that the SVM classifier performs best overall.

2. Method

The problem of BSC requires pre-processing, feature extraction, training, and evaluation. In particular, the way CNN is used to extract speech audio features is to apply convolution on the spectrogram generated from MFSC features, which has a first dimension corresponding the number of frequency bands, second dimension corresponding to the number of frames in the time domain, and third dimension corresponding to the degree of derivates taken. This method is based on a paper by Li Deng et al. published in 2014 (et al., 2014). The different tools used in this pipeline are described and summarized alongside brief acknowledgement of related works.

2.1. Phenomes

Phenomes are used in linguistics to distinguish the envelopes that characterize sounds of human speech. The task of speech recognition is therefore best performed by training an algorithm to look for the phenomes in audio information.

2.2. Speech Processing: Mel-Frequency Spectral Coefficients or Filter Banks

It is necessary to prepossess speech by computing the Mel-Frequency Spectral Coefficients before any features indicative of phenomes can be extracted. The steps to computing MFSC of an audio recording involve adding emphasis to high frequencies, framing the sound in time for discrete sampling, computing the FFT, and computing the log power. The specific procedure can be followed by reading (X. Huang & Hon., 2001).

The resulting data is called the MFSC's Spectrogram or Filter Bank. This data is two dimensional and can be interpreted as an image. Take for example the MFSC Spectrogram from an audio ”zero” of our dataset. This image-like form motivates the use of CNN.

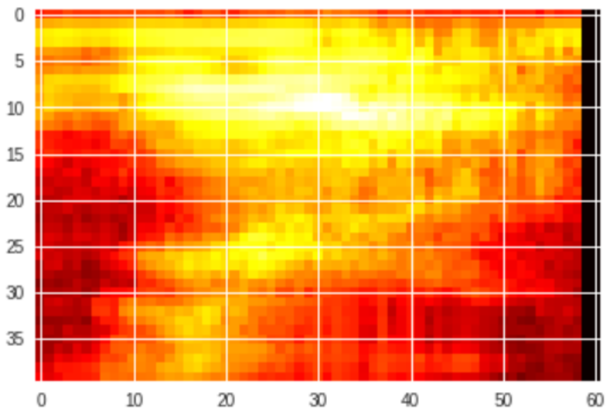

Figure 2. The MFSC's Spectrogram or Filter Bank, of an audio recording for a person saying ”Zero”. The vertical axis is the MFCS power and the horizontal axis is time frames.

2.3. Speech Processing: Mel-Frequency Cepstral Coefficients

Once the MFSC features are available, additional processing extracts the first set of features, the Mel-Frequency Cepstral Coefficients. This traditional feature extraction algorithm is computed by performing the discrete cosine transform of the MFSC and then mean-normalizing the result. The specific procedure can be followed online by (Fayek).

2.4. Convolution Neural Networks

We use a convolution neural network to extract the second set of features directly from the MFSCs instead of the MFCCs. Since CNN's are known to perform well in image tasks, they are an appropriate model to extract features from the image-like MFSCs. Our CNN is implemented using TensorFlow's tf.layers library. It is designed with one input layer, 2 convolutional + max pooling layers, two dense layers, and an output logit layer. A general CNN is modeled below.

In order to extract features from the MFSC inputs, the CNN must be trained. A CNN is trained by passing a dataset through every layer, computing the accuracy of the last layer's prediction, and accordingly feeding that information back through the model to update each layer's weights. Once a model achieves satisfactory accuracy, the weights at any layer act as learned feature components. In our case, MFSC data is passed through the model and extracted after

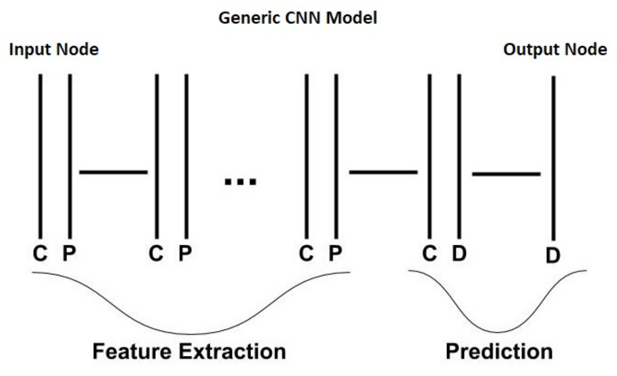

Figure 3. A CNN Has 3 Basic Components: (C) Convolution Layer (P) Pool Layer (D) Dense Layer: Classification

the the 2nd convolutional layer's weights are applied. These weighted MFSC data are the features we use to train our classifiers.

2.5. Learning Algorithms

Once we have both data sets of extracted features we explore the performance of the following five classifiers:

Linear SVC
Linear SVC uses support vectors to compute the maximum margin between hyper planes that separate the different classes of data. The algorithm places the decision boundary in the middle of these margins. We explore the margin normalization hyper parameter C = [10,1,1e-1,1e-2,1e-3]. We implement SVC using the SciKit-Learn library sklearn.svm.SVC.

Ridge Regression
Ridge Regression Classifier also utilizes the idea of support vectors and margins to separate the data. Ridge regression also penalizes data based on its distance from the corresponding hyper plane. It's penalty is continuous unlike SVC's constant step penalty. This allows the derivation of a closed form solution for the optimal decision boundary in ridge regression. We explore hyper parameter alpha = [10,1,1e-1,1e-2,1e-3]. We implement ridge regression using the SciKit-Learn library sklearn.linear_model import RidgeClassifier.

KNN
KNNs make simple predictions based on the locality of the data - that is, it compares the labels of its K nearest neighbors in the training set and predict that dominate label. The distance metric is usually euclidean, although other distance metrics are also available. For our application of KNN we apply a euclidian metric and explore hyper parameter for the number of compared neighbors K = [5,10,15,20,25]. We implement KNN using the SciKit-Learn library sklearn.neighbors.KNeighborsClassifier.

LDA
LDA finds an axis that grants the smallest variance of both classes and the greatest separation distance. Then a hyperplane is placed along this axis to maximally separate the two classes of data. There are no parameters to be trained. We implement LDA using the SciKit-Learn library sklearn.discriminant_analysis.LinearDiscriminantAnalysis.

Decision Tree
Decision Trees separate data through sequential rules. Each rule is applied at a node of the tree while the edges represent the flow of data between nodes. Entropy or gini impurity are target function used to optimize for increasing or decreasing nodes of a Decision Tree. Here, we explore criterion = ['entropy', 'gini'] as well as max depth = [5,10,15,20,25]. We implement LDA using the SciKit-Learn library sklearn.tree.DecisionTreeClassifier.

2.6. Performance Metrics

The performance of each classifier is measured using accuracy. This is the formula for accuracy:

$$Accuracy = \frac{\text{Correctly Predicted}}{\text{Total Sample Size}}$$

3. Experiment

The experiments of this project are conducted over two feature datasets extracted from a single parent audio dataset. The parent dataset is the Free Spoken Digit Dataset consisting of multi-authored community recordings of english spoken digits 0-9. The data are 1,500 short audio wav files, trimmed for minimal silence, recorded at 8kHz. Each audio file is named by its corresponding label. We downsample this multiclass dataset to a binary dataset. We chose a binary dataset of 300 samples: 150 samples of the audio "zero" and 150 samples of the audio "one". We decided to downsample in order to manage the computational time for preprocessing and feature extraction in this project.

3.1. Preprocess and Feature Extraction

From the 300 sample audio dataset we first preprocess the MFSCs. Each audio file is loaded by a python program. The MFSC is processed from the audio as described in the methods and stored as a (61 x 40 x 3) numpy array. This numpy array is then flattened to (7,320 x 1) and saved to disk under the file name corresponding to the original audio. Once all audio files are preprocesed into MFSC's they are loaded again into another python program that reloads the MFSCs and vertically concatenates them into a (300 x 7,320) numpy ”Pre-Feature Array”. At the same time, this python program constructs a (300 x 1) ”Label” array corresponding to each MFSC’s labeled file name. Both the Pre-Feature and Label arrays are exported to disk as numpy files.

Once the MFSC are computed, we performed the MFCC and CNN feature extraction. The MFCC feature extraction is performed first. The Pre-Feature Array and Label dataset are loaded into a python program that computes the MFCC from the MFCC as described in the methods. The MFCC dataset is (300 x 12 x 61) before flattening the last two dimensions into the final (300 x 732) ”MFCC Feature” array. This array is then exported to disk as a numpy file. The order of these features is retained and is easily matched with the Labels.

The CNN feature extraction is performed next. The Pre-Feature Array and Label dataset are once again loaded into a python program that shuffles the data and trains the CNN model as described in the methods. The 300 samples of data are batched into subsets of 50 samples. The entire dataset is feed through the CNN from the input node through every layer to the output layer, with feedback enabled, roughly 1,660 times in order to ensure appropriate fitting. Once the model is fitted with appropriate weights, we pass the un-shuffled data through one last time. This time, feedback is disabled and the data is extracted before it passes through the deep layers. The resulting data extracts the features learned by the CNN. These features are shaped (300 x 10 x 16 x 64) before the last dimensions are flattened into the (300 x 1240) ”CNN Features” array. This array is then exported to disk as a numpy file. The order of these features is retained and is easily matched with the Labels.

3.2. Classifier Training

Once the MFCC features and CNN-extracted features for the 300 audio files are computed, classification using the 5 standard classifiers on these feature vectors can be done. For the CNN-extracted feature data, we shuffle the data to perform the first round of analysis: we split the data into a parition subset of 80 percent and 20 percent for training and testing set; we then perform a three-fold cross validation on the training set to obtain the optimal hyper-parameters for 4 of the 5 standard classifiers: Ridge Regression, SVC, KNN, and Decision Tree. Notice how the training accuracy improves as with more optimal hyper-parameter choice, as seen for the SVM in figure 4.

During each partition the classifiers’ training, validation accuracy, and hyper-parameters are recorded and the hyper-parameters are subsequently used for testing the classifier accuracy. The one classifier without a hyper-parameter to train is LDA. It is directly trained to obtain the training and testing accuracy due to the fact that LDA does not require any hyper-parameter. We then record the training, validation, and testing accuracy for later analysis. This cross-validation and testing process is then repeated on a 50-50 train-test partition and a 20-80 train-test partition. This process yields 45 accuracy metrics (3 partitions * 5 classifiers * 3 error metrics) as well as 15 hyper-parameters (3 partitions * 5 classifiers).

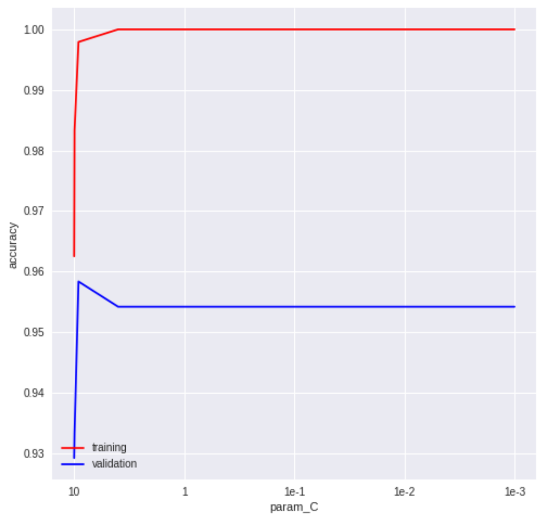

Figure 4. Within each cross-validation, a classifier’s hyperparameter was tuned much like this SVM’s hyperparameter C.

This process is then repeated for two more rounds under different ordering of the same MFCC data set and the corresponding accuracies are averaged (to reduce accidental error) across these three data partitions to yield 45 accuracy metrics and 45 sets of hyper-parameters (3 rounds * 15 hyper-parameters/round). The collected metrics and hyper-parameters are presented and analyzed in subsequent sections.

3.3. Performances and Results

In general, our data shows that standard classifiers obtain better results when applied directly to the manually computed MFCC features instead of the CNN-extracted features, and that SVC outperforms the other 4 standard classifiers on both data sets. This can be observed in the figure table 1.

That MFCC features allow better classifier performance is surprisingly contradictory to our hypothesis. To further understand this result, we analyzes the features extracted from CNN and the MFCC. We begin our analysis by visualizing the feature space, but because of the large dimensionality of both features, choose only the two most representative features in the feature space.

For the 300 speech data, we assume the greatest variance is most representative of the feature space. We compute the variance on each features across all data, and pick out the two features with the greatest variance. We plot the 300 speech data with these two chosen features (See Fig. 5, Fig. 6). Upon looking at the plot, an embedded collinearity between the data is immediately obvious in the CNN-extracted feature space, whereas this issue is not nearly as prominent in the MFCC case. It is understood that many standard classifiers do not handle collinear data well(Rich Caruana, 2006), especially LDA, we hypothesize that the collinearity could be a contributing factor to the bad performance of standard classifiers combined with CNN-extracted features. With the visualization, we have also realized the complexity of the problem, as clearly the data are not separable in the chosen dimensions.

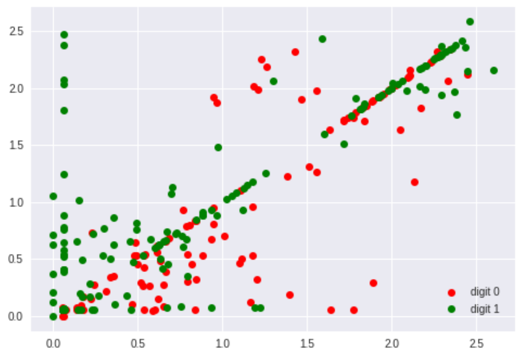

Figure 6. The two largest features are computed across samples. For each sample, we plot those features.

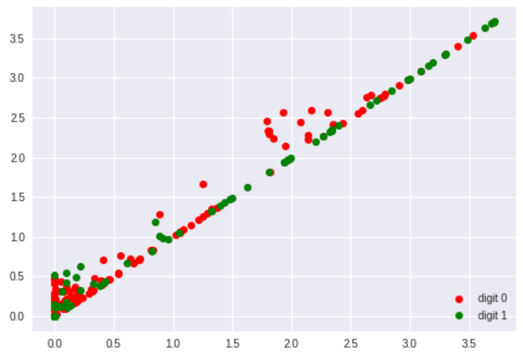

Figure 5. The two CNN-extracted features with greatest variance are plotted.

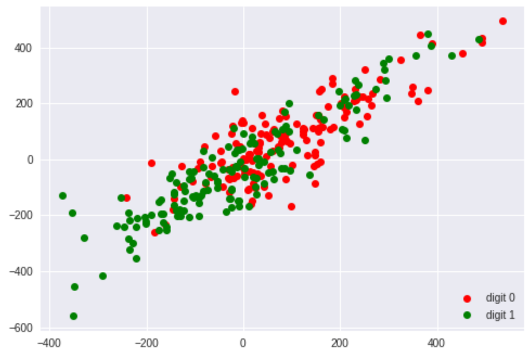

Figure 7. The two MFCC features with greatest cross-data variances are plotted.

To further analyze the result, with our knowledge that in NN, in the output of each layer, the higher value tends to contribute more to the final output layer, we hypothesize the feature with maximal sum can also potentially be representative of the data. With this, we similarly plot the 300 speech data with two features that sums the maximal weights in the space of CNN-extracted features as well as in MFCC feature space. For brevity, we have only included the plot for CNN-extracted features (Fig. 6), as the one for MFCC features in this case visually resembles the last case. Here, we in face do see that the two classes are somewhat separated by a diagonal boundary. This suggests one of the advantages of CNN-extracted features, which are organized in such a way that facilitates the classification problem.

Acknowledging this colinearity in our feature data, we turn back to compare performances across classifiers, we observe some striking results. First, we should note that the SVC classifier performs best, with an accuracy of 97.1 over the average of all partitions and both feature datasets. This can be clearly seen in figure 8. Looking more carefully at figure 8 we see the next best performing classifiers are Ridge Regression, KNN, LDA, and Decision Tree. For the 80/20 partition the second best performing classifier is KNN. For the 50/50 partition the second best performing classifier is Ridge Regression. For the 20/80 partition the second best performing classifier is Ridge Regression. LDA and Decision tree are consistently the worse performers for this BSC task.

Looking more carefully at figure 8 we see that

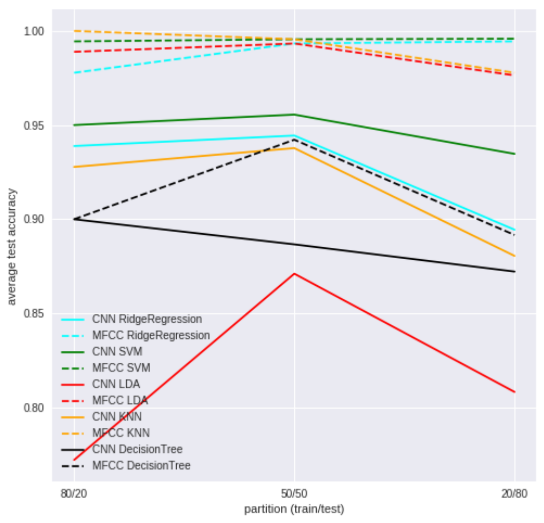

Figure 8. Each classifier, regardless of the feature set, performs differently across partitions.

4. Conclusion

Based on the data and accompanying analysis, we have demonstrated that applying standard classifiers along on the MFCC feature data in fact performs better than applying them on the CNN-extracted data, which turns out to be quite surprising. The reason could be attributed to the collinearity of the CNN-extracted data set, as CNN-extracted data set contains higher dimensions (10240 dimensions) than the supplied MFSC features (40 * 61 * 3 = 7320 dimensions), and some standard classifiers do not handle well the collinearity present in the data (as demonstrated in Fig.5 and Fig.7). Since this is simply an educated hypothesis, further researches into this somewhat paradoxical result may be of interests in the machine learning academia.

Aside from this, we have also demonstrated that SVC obtains the optimal classification results, and that classifiers in general perform better with increased training sample size, although in many cases their accuracies plateaus when supplied with 50 percent of all data as training data.

5. Bonus Points

We believe that this project deserves bonus points because of the novel combination of CNN's with the traditional supervised classifiers as well as the large effort in our own data preparation in addition to the quality analysis of five classifiers. Our efforts expended on data preparation include manually transforming the original speech audio file into MFCC and MFSC feature vectors, and the MFSC features are further fed into a CNN to extract useful features for training.

References

Boyd, C. The past, present, and future of speech recognition technology. https://cdn-images-1.medium.com/max/2000/1*LtuOUuCSwAKh8Um7_NCHmA.png. Accessed: 2018-12-16.

et al., L. D. Convolutional neural networks for speech recognition. In *IEEE/ACM Transaction on Audio, Speech, and Language Processing, Vol. 22, NO. 10*, pp. 1533–1545. IEEE, 2014.

Fayek, H. Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between. https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html. Accessed: 2018-12-12.

Rich Caruana, A. N.-M. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. ICML.

X. Huang, A. A. and Hon., H. (eds.). *Machine Learning: An Artificial Intelligence Approach, Vol. I.* Prentice Hall, 2001.