CS181 Practial 2022

Henry Kuo, Zad Chin, Rudra Barua

Spring 2022

1 Part A: Feature Engineering, Baseline Models

1.1 Performing Principal Component Analysis (PCA)

Given the high-dimensional nature of the training data set, we want to reduce the dimensionality of our data because not all features will give us additional information about the input. This allows our models to generalize better (prevent overfitting) and more computationally efficient.

We arbitrarily decide to reduce our dataset to 500 principal components. The first 500 principal components represent a basis that can explain the maximal amount of variance and capture most information of the data.

To compute PCA, we utilized the library `sklearn.decomposition.PCA`. We first standardize the data to make each feature have zero mean and a standard deviation of 1. This prevents features that have different scales from affecting the results. If one feature varies less than a second feature because of differences in scale, PCA might determine that the second feature explains the data more, while the two features are equally important.

After standardizing the data, we fit the data to produce a basis of 500 principal components. Then, we project the original dataset to the basis, returning the lower-dimension dataset.

1.2 Logistic Regression

After projecting the dataset to a lower-dimension space, we used logistic regression (a probabilistic classification model) to classify the data. We perform the following steps:

- **Prediction**

  The softmax function is appended at the end of the classifier to make predictions. It takes a vector $z = [z_1, z_2, \cdots, z_K]$ of K arbitrary values and fit them to a probability distribution with each value in the range $(0, 1)$. The softmax function is defined in Definition 3.6.3 in the textbook.

- **Loss Function**

  The loss function we select for our model is L2 loss without regularization, which is in Section 2.5.2 in the textbook.

- **Gradient descent**

  To minimize the loss, we adopt gradient descent. It follows the gradient of the loss with respect to w to find the minimum of a function. The exact steps can be found in Section 3.4.1 in the textbook.

1.3 Results

We have computed the overall accuracy, per-class accuracy, and the confusion matrix for both standardized and unstandardized data in Section 4.2.

For PCA, we investigated how well our data is represented with 500 principal components for both the amplitude and the mel-spectogram data. For amplitude data, we see that the first principal component can explain about 7% variance of the data and the first 500 principal components can explain 60% of the variance. For the mel-spectogram data, we see that the first principal component also captures 7% variance of the data, but the first 500 principal components can capture 97% of the data. The results can be found in Section 4.1.

For logistic regression, we tested it on both standardized and unstandardized data before performing PCA. Results can be found in Section 4.2. In short, we can see that the accuracy on standardized amplitude data is 0.196 and standardized mel-spectogram data is 0.253.

1.4 Discussion

We can see that logistic regression performs better on Mel Spectogram Data than Amplitude data. The Mel Spectogram performs better perhaps because the basis provides a better representation of the data for a linear decision boundary. From our PCA analysis, the first 500 principal components of the Amplitude dataset only explain 60% of the variance, while that of the Mel Spectogram dataset explains 98% of the variance. Reasons for perfoming PCA before logistic regression can be found in Section 1.1.

From the confusion matrix, we hypothesize that the model primarily just classifies the test data into one group since the classes are imbalanced. We can see that for the standardized amplitude data, our model classifies majority of the data (over 800 data points) to class 2. Similarly, for standardized Mel Spectogram data, it classifies 200+ points to label 8. Both of the classes have approximately 12% (majority) representation in the original dataset.

We can also see the effect of standardizing data on logistic regression. Logistic regression performs better on standardized amplitude data, but worse on the standardized Mel Spectogram, suggesting that standardization impacts the accuracy and we should be more careful in choosing whether to standardize moving forward.

2 Part B: More Modeling

2.1 Random Forest

Random Forest is a supervised classification/regression machine learning algorithm which is an extension of bootstrap aggregation of decisions tree. Boostrap, by Definition 10.3.1 of the STAT111 textbook, is defined as a sampling technique where we draw samples from population by sampling with replacement (ie: same samples can be drawn twice). In our case, we choose to sample 80 trees(`n_components`) for amplitude data and 1000 trees(`n_components`) for mel spectogram data.

For classification problem such as ours, after the randomly sampling decisions trees, the set of decision trees will collectively collect the votes from different decision trees to decide the final prediction (ie: the final class it belongs in our case), and output the result. A simplified illustration on how the algorithm works can be found Section 4.3.

2.1.1 Results

For Random Forest trained on standardized amplitude data, the accuracy is 0.2366.
For Random Forest trained on standardized mel spectogram data, the accuracy is 0.3855.

The Per-class accuracy for Random Forest, as well as confusion matrix for random forest trained on amplityde or Mel Spectogram data can be found in Section 4.3.

2.1.2 Discussion

Similar to logistic regression, we see that random forest has higher accuracy when trained with mel spectogram data than amplitude data. As hypothesized in Section 1.3, it might be because the first 500 components capture 97% mel spectogram data but only 60% of amplitude data.

Comparing the result against logistic regression, we see that overall, random forest have higher accuracy for both the amplitude data and mel spectogram data. We hypothesize that this is because random forests have more parameters than logistic regression, which can improve accuracy.

Furthermore, comparing the per class accuracy for both Random Forest and Logistic Regression in Section 4, we see that overall, Random Forest performs better than Logistic Regression. However, for classes that is underrepresented (such as class 1 and class 8), logistic regression have higher accuracy for those classes, suggesting that logistic regression performs better when there is unbalanced class distributions or noisy variables.

2.2 Hyperparameter Tuning and Validation

2.2.1 Approach

For this part, we adopted sklearn `GridSearchCV` to help find the best parameters to minimize the cross-validation loss for Random Forest. Grid search tests all possible permutations of the hyperparameters and return the model variant that leads to the best results.

The hyperparameters we varied are:

- `n_parameters` : The number of trees in the forest (`[320, 360, 400, 440, 480, 520]`).
- `max_depth` : The maximum depth of the tree (`[None, 10, 50, 90]`).
- `class_weight` : The weights associated with each class (`[None, 'balanced']`). `None` will make each class have the same weight. `'balanced'` adjusts weights inversely proportional to class frequencies in the input data.

2.2.2 Results

For both types of data, the best performing hyperparameters is `n_parameters = 520`, `class_weight = 'balanced'`, `max_depth = None`.

These hyperparamters give a mean test score of 0.243820 and 0.429672 for Amplitude and Mel Spectogram Data, respectively.

2.2.3 Discussion

We observe that increasing the number of estimators produces better accuracy scores for both Amplitude and Mel Spectogram data. This is because having more trees means that the predictions are made from a larger number of "votes" from a diverse group of trees. However, having more trees increases computational expenses. For `n_parameters = 520`, it takes about 10 min. to run.

For `max_depth`, we see that if we constrained the number of leaves, it performs worse than allowing the tree to keep diverging until all leaf nodes are pure (i.e., all data on the leaf is from the same class). This is because having a larger number of splits allow each tree to better explain the variation in the data.

For `class_weight`, we see that `class_weight = 'balanced'` performed better for both types of data. This is because the original dataset was slightly imbalanced, hence balancing the class weight can prevent the classifier to only predict the majority class.

3 Feed-Forward Deep Neural Network

We implemented a feed-forward deep neural network (FFNN) with two hidden layers and an output layer with ten nodes for each sound class. However, as we were testing early implementations of the FFNN, there was a high degree of overfitting on the training data. To address this issue, we added three dropout layers to randomly set input values coming into layers to zero at a frequency of rate 0.1 at each step of training. By randomly setting inputs to zeroes, we reduce the overfitting of our model to the data. Furthermore, we fit and train the model using a categorical cross entropy loss using an Adam optimizer. We used this loss function because it allows us to differentiate between multiple probability distributions, which means it can be used in the case of classifying different sound categories. We choose the Adam optimizer because our research showed us that Adam was well-suited for training neural networks faster and more efficiently without much fine-tuning up front. This was perfect for our needs of trying out various network architectures and parameters.

3.1 Results

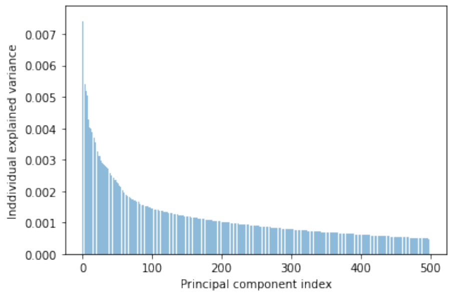
The results can be found in Section 4.4. We were able to acheive a test accuracy of about 50%.

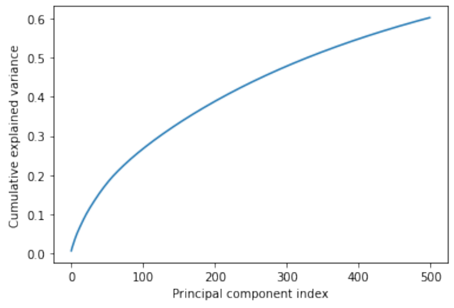
3.2 Discussion

As shown in the results, the FFNN models are at best achieving a test accuracy of 50% regardless of what hyperparameters or architecture is used. Furthermore, not only are denser and deeper networks achieving similar peak accuracy as the less complex networks, they are taking substantially longer to train. Hence, we saw no point in creating more complex networks when we were achieving similar results but with diminishing computational efficiency. This may imply that this feed-forward neural network lacks the flexibility to capture the nuances in data. Furthermore, we saw that the neural network was prone to overfitting even though we added dropout layers and increases the frequency of dropout. Although the FFNN models with dropout layers significant improved the overfitting of the model early on, it was still achieving above 90% accuracy on the training data regardless of how much dropout there was. Lastly, regardless of if there is dropout or not, our FFNN models converge early on in the training process, at around 10 to 15 epochs. This means that its peak test accuracy of 50% is achieved very early on its training process and the rest of its training is spent overfitting to the training data with small fluctuations in the test accuracy. This seems to make the FFNN one of the best models we have worked with. Not only has it achieved the highest test accuracy thus far, it also achieves that test accuracy with minimal hyperparameter tuning and converges to its peak performance very early on.

4 Appendix: Graphs and Diagrams

4.1 PCA





4.2 Logistic Regression

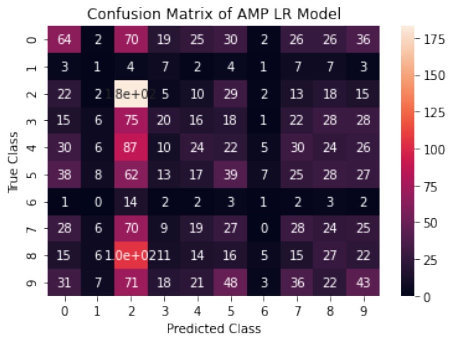
Logistic Regression Accuracy

Data	Accuracy	
	Standardized	Unstandardized
Amplitude	0.196	0.191
Mel Spectogram	0.253	0.315

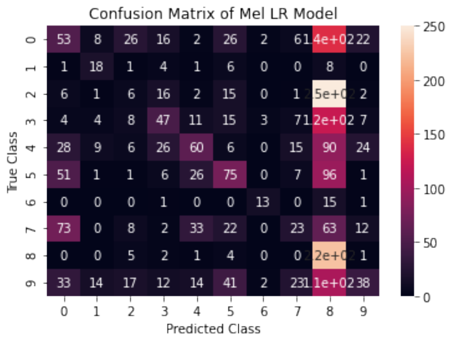
Logistic Regression Per-Class Accuracy For Standardized Data

Data	Class									
	0	1	2	3	4	5	6	7	8	9
Amplitude	0.213	0.026	0.612	0.087	0.091	0.148	0.033	0.119	0.114	0.143
Mel Spectogram	0.177	0.462	0.02	0.205	0.227	0.284	0.433	0.097	0.945	0.127

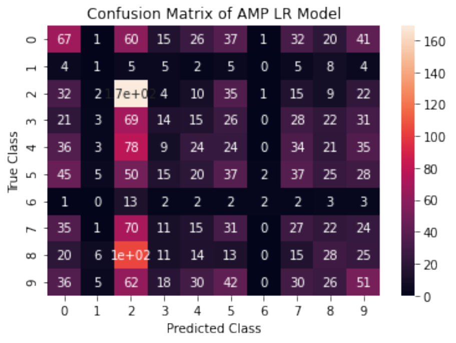
Confusion Matrix for Logistic Regression Models



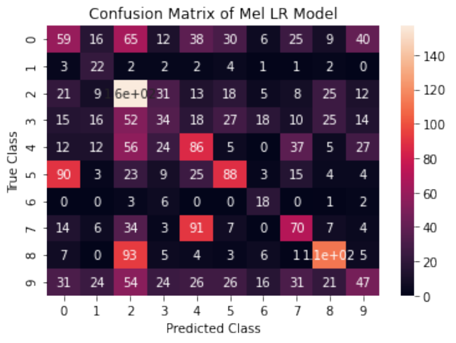
(a) Heat map:standardized amplitude data



(b) Heat map:standardized Mel Spectogram data



(c) Heat map: unstandardized amplitude data



(d) Heat map: unstandardized Mel Spectogram data

4.3 Random Forest Result

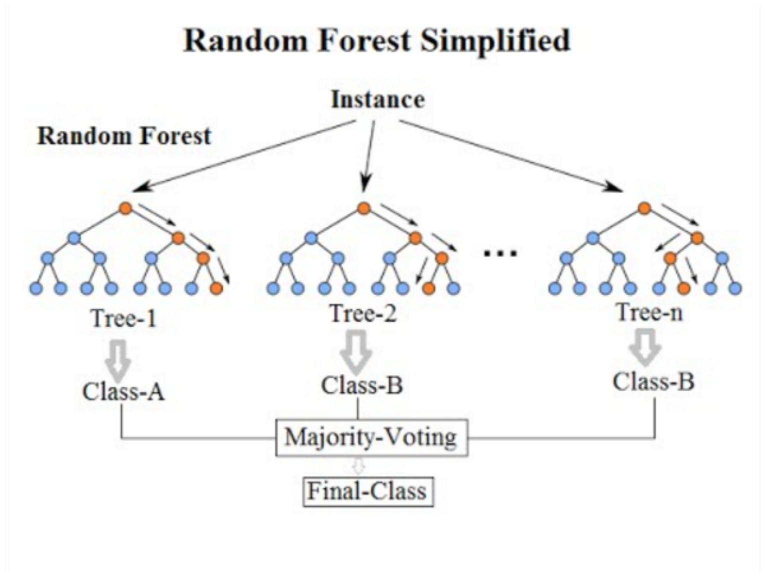
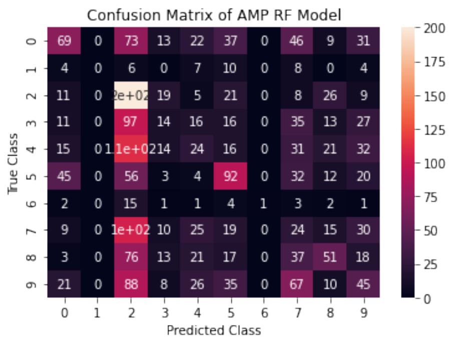
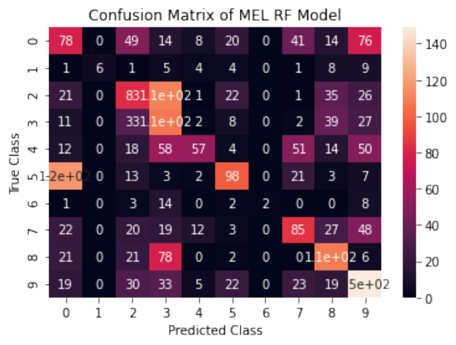


Figure 1: A simple illustration on how Random Forest works
Source Wikipedia



(a) Heat map: Random Forest Amplitude data



(b) Heat map: Random Forest Mel Spectogram data

Per Class Accuracy For Random Forest

Data	Class									
	0	1	2	3	4	5	6	7	8	9
Amplitude	0.23	0	0.669	0.061	0.091	0.348	0.033	0.102	0.216	0.15
Mel Spectogram	0.297	0.128	0.542	0.397	0.22	0.379	0.067	0.335	0.479	0.493

4.4 Feed-forward deep neural network

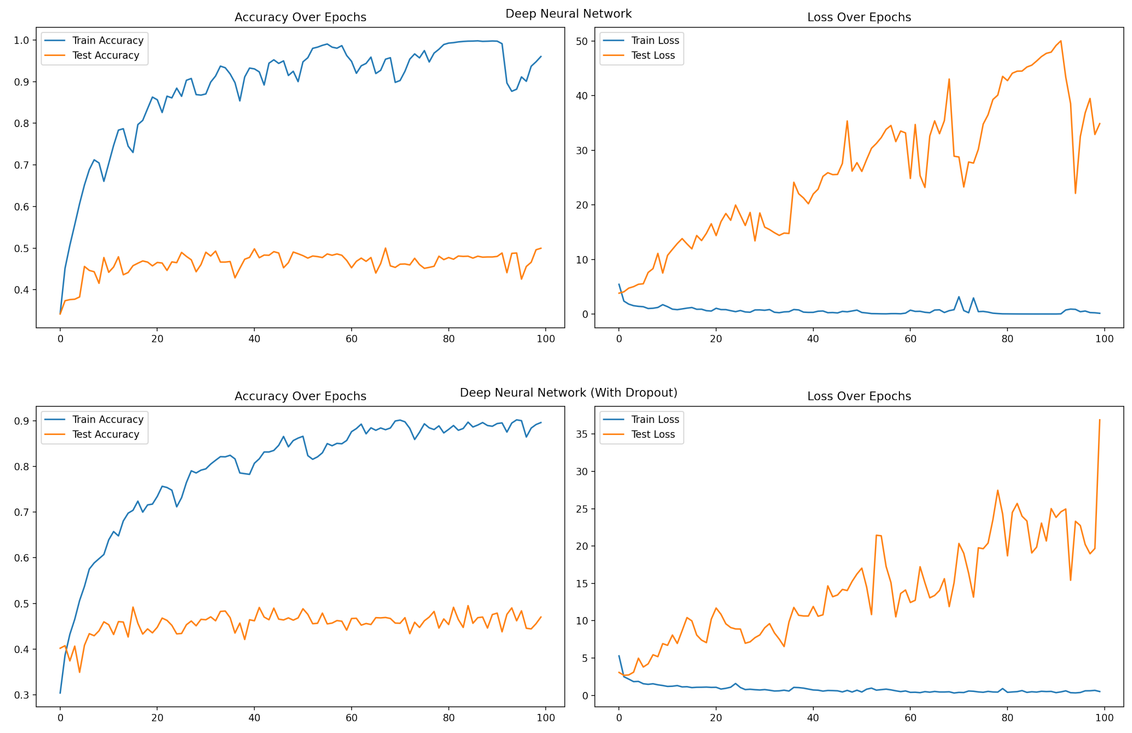


Figure 2: Results for feed-forward deep neural network.

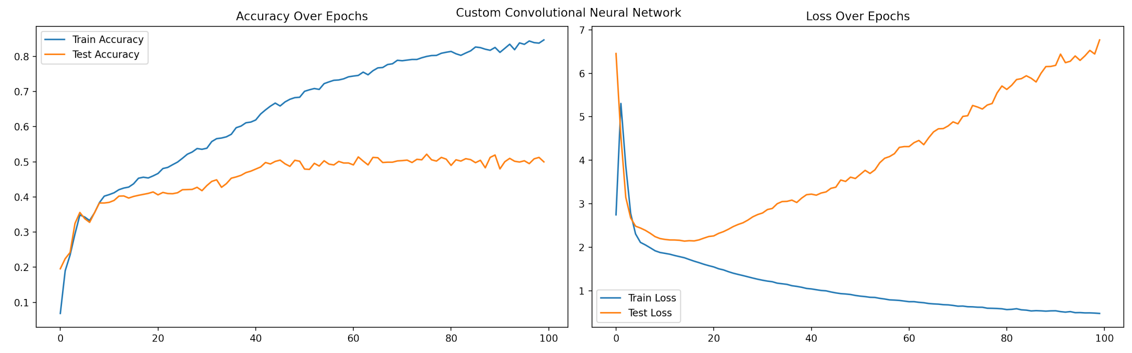


Figure 3: Results for Skyler's Convolutional Neural Network.