CS181 Practical: Classifying the Sounds of NYC

Blake Bullwinkel, Mark Penrod

April 9, 2021

1 Part A: Feature Engineering

1.1 Approach

The SONYC audio data is provided in a high-dimensional format, with 44100 features for the amplitude data and a (128×87) dimensional feature space for the 2D Mel spectrogram data. Given that the dimensionality of our data is larger than the number of observations in our training set, we use PCA to project the data into a lower dimension, thereby minimizing the risk of overfitting.

We normalized the data by fitting Min-Max scalers to the amplitude and spectrogram data and performed PCA on the training data, keeping the first 500 principal components. For the amplitude data, these components represent the 500 directions along which the amplitude varies the most, while the principal components for the (flattened) spectrogram data represent the 500 directions along which 128 audio-related features computed from the amplitudes show the most variance.

Finally, we trained logistic regression models on the PCA transformed training data and calculated classification accuracies on both the train and test sets. Given our multi-class setting, we opted to use multinomial logistic regression, which uses a softmax function instead of the familiar sigmoid used in binary classification. We made classification predictions by finding the argmax of the normalized probability vector produced by softmax, and trained the model weights using cross-entropy loss. Table 1 summarizes the overall train and test accuracies for a baseline and three logistic regression models across the two data types, amplitude and Mel spectrogram. See Appendix 5.1 for bar plots depicting the class accuracies for each model and data type.

1.2 Results

Baseline: Naïve Model
Model 1: Logistic Regression
Model 2: Logistic + L2 Regularization
Model 3: Logistic + L2 + Class weighting

Table 1: Overall classification accuracies (%) for logistic regression models

	Baseline		Model 1		Model 2		Model 3	
	Train	Test	Train	Test	Train	Test	Train	Test
Amplitude	11.6	11.0	41.1	18.3	40.7	20.3	40.2	19.8
Mel Spectrogram	11.6	11.0	80.4	33.5	53.6	38.2	54.1	39.2

1.3 Discussion

For our baseline, we built a naïve model which randomly classified the data based on the relative frequency of each class in the data set. As reflected in Table 1, this approach consistently yielded the lowest classification accuracy. Next, we used an un-regularized logistic regression, which produced much higher accuracy scores on the training data than the test data (especially for the Mel spectrogram data). In order to mitigate overfitting, we introduced L2 regularization, which penalizes large coefficient values according to a squared magnitude. As expected, regularization produced a noticeable drop in train accuracy and an increase in test accuracy, particularly for the Mel spectrogram data. Finally, we note that the distribution of noise types represented in the training data is highly imbalanced. To address this, we introduced class weights to the model based on class frequencies. As shown in Figure 1 (Appendix 5.1), this noticeably improved classification accuracies for sounds labelled "car horn" and "gun shot," both of which are minority categories.

Overall, the models built using the Mel spectrogram data outperformed those derived from the amplitude data. We hypothesize a few reasons for this difference. First, the Mel spectrogram data is a pre-processed version of the amplitude data with information added in the form of 128 audio-related features. Second, an analysis of the variance explained by the principal components derived from the two data sets revealed that the first 500 Mel spectrogram PCs captured 92.6% of the variation, while the first 500 amplitude PCs captured only 60.1%. These aspects of the Mel spectrogram make it a richer data set that make higher classification accuracies more attainable.

2 Part B: Model Classes

2.1 Approach

In this section, we experimented with kNN, random forest, and feed-forward neural network (FFNN) models in order to see whether nonlinear models might improve classification accuracies in comparison to the logistic regressions discussed above.

For the kNN model, we used $k = 3$ for the amplitude data and $k = 5$ for the Mel spectrogram data after trying out a few values and selecting those which achieved the best test accuracies. For the random forest, we trained a model with 100 decision tree estimators, a maximum tree depth of 20, and the minimum number of samples required to split a node equal to 1. Finally, we built a feed-forward neural network with three hidden layers and an output layer with ten nodes corresponding to the ten classes. We also added two dropout layers to minimize overfitting and trained the network using an Adam optimizer with categorical cross-entropy loss. See Appendix 5.2 for the exact model architecture used and Appendix 5.3 for the per class accuracies achieved by each model.

2.2 Results

Table 2: Overall classification accuracies (%) for nonlinear models

	KNN		Random Forest		FFNN	
	Train	Test	Train	Test	Train	Test
Amplitude	46.7	18.1	46.7	18.1	98.7	24.5
Mel Spectrogram	74.3	36.5	74.3	36.5	88.6	46.5

2.3 Discussion

The nonlinear models had mixed results on the amplitude and Mel spectrogram data. First, both the kNN and random forest classifiers performed about as well on the test data as the un-regularized logistic regression model from Part A. Surprisingly, these two models had the same overall train and test accuracies for both the amplitude and Mel spectrogram data using hyperparameters that led to reasonable performance.

Although kNN and random forest are both non-linear models, our results suggest that they may still not be flexible enough to learn the complexity of the amplitude and Mel spectrogram data. As shown in Table 2, the feed-forward neural network with dropout achieved significantly higher overall train and test accuracies on both data sets. This suggests that the size and flexibility of a neural network may be necessary for accurate predictions using this sound data. However, we also notice that the discrepancy between the neural network train and test accuracies is very large, indicating that it is prone to overfitting, even with dropout.

3 Part C: Hyperparameter Tuning and Validation

3.1 Approach

To improve upon the relatively poor performance of the kNN and random forest classifiers discussed in the previous section, we performed hyperparameter tuning on the number of neighbors for kNN using a linear search ($k = 1, 2, 3, 4, 5, 6$) and simultaneously on maximum tree depth and minimum number of samples required to split a node for random forest using a grid search. For each value or pair of values, we recorded the overall test accuracy achieved in order to determine the optimal hyperparameter settings. The heatmaps in Appendix 5.4 help visualize the results of grid search on both the amplitude and Mel spectrogram data for the random forest classifier.

3.2 Results

Table 3: Best test accuracies for kNN and random forest with hyperparameter tuning

	KNN	Random Forest
Amplitude	20.57	27.36
Mel Spectrogram	36.87	41.92

3.3 Discussion

As shown in Table 3, tuning the number of neighbors only marginally improved the overall test accuracies for kNN. This suggests that kNN might be an overly simplistic model given the complexity of the audio data. On the other hand, tuning the two random forest hyperparameters achieved significantly better results on both data sets than before. In fact, the test accuracy achieved by the optimized random forest on the amplitude data is much higher even than that achieved by the feed-forward neural network. Contrary to what our results from part B may have suggested, this shows that given the right hyperparameters, a random forest is, in fact, flexible enough to capture the complexity of the data and make accurate predictions. However, neural networks may still be a preferable option given their ability to achieve high accuracies without any hyperparameter tuning.

4 Part D: Long Short-Term Memory

Note: Please see Part A for our handling of class imbalance using class weighting.

4.1 Approach

As a bonus, we implemented a Long Short-Term Memory (LSTM), a type of Recurrent Neural Network (RNN) architecture. RNNs are a class of neural networks suitable for time series data because they introduce concepts of memory and context history. RNNs achieve this by carrying a set of weights known as a hidden state that is propagated throughout the network and updated at each time step. LSTMs expand and improve upon the basic RNN by introducing gates, which support skip connections (helping to mitigate the vanishing/exploding gradient problem) and allow certain inputs to be weighted more heavily (allowing the network to forget irrelevant information). Notably, we applied the LSTM bidirectionally, which allows the network to learn patterns based on both past and future information in order to make better predictions.

4.2 Results

Table 4: Final classification accuracies (%) for LSTM

	Train	Validation	Test
Mel Spectrogram	84.69	73.27	47.75

See Appendix 5.5 for visualization of model training in terms of accuracy and loss

4.3 Discussion

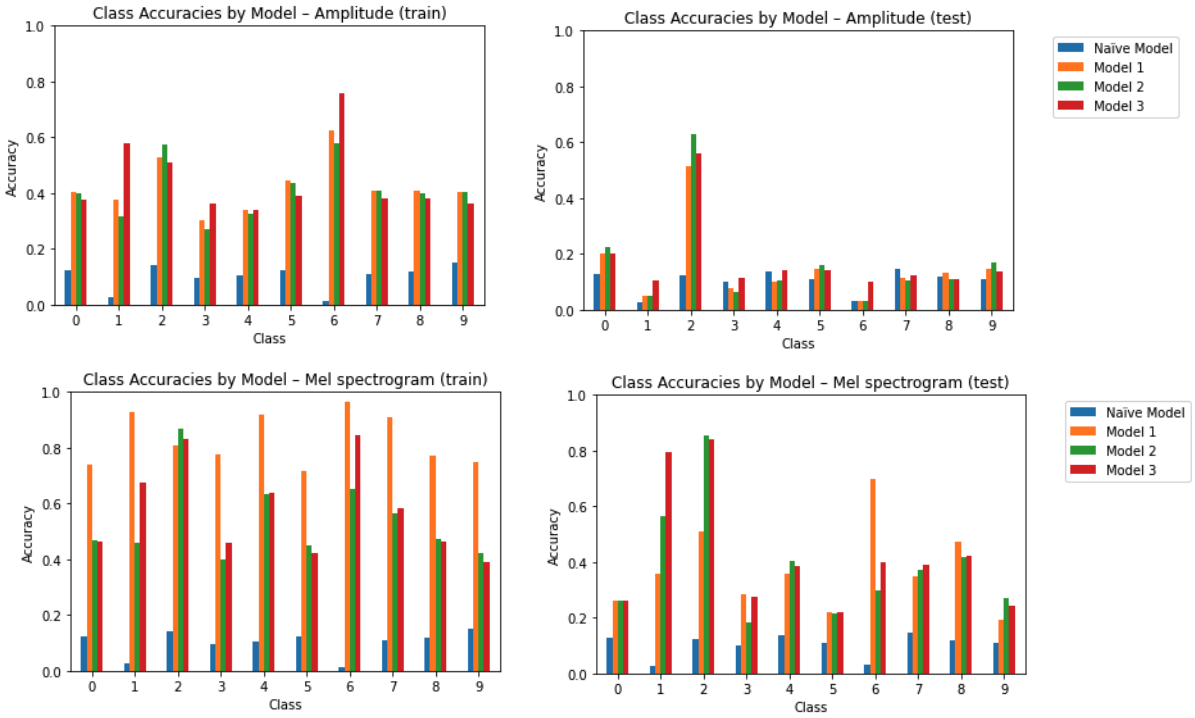
Given the time-series nature of audio data, LSTMs are a natural choice for a classification model. In general, LSTM models are designed to perform a feature analysis on the input data and pass those features through a series of dense layers, which then execute the classification task. This approach is analogous to that commonly implemented in CNNs. We tested several different architectures, including stacking multiple LSTM layers, varying the size and number of dense layers, and incorporating regularization with dropout layers, which introduced sparsity into the network by probabilistically "dropping out" nodes in the network during training. The final architecture is detailed in Appendix 5.6.

Because models trained on the the Mel spectrogram data consistently outperformed those trained with the amplitude data, we decided to train our LSTM on the former. As seen in Table 4, the LSTM achieved a noticable increase in test accuracy compared to previous models, suggesting that either the model architecture or the added complexity improved predictions. Surprisingly, the final validation accuracy and the test accuracy differed quite a bit. Given that validation sets are used during model training to estimate performance on unseen data, it was unexpected to find that validation accuracies were around 25% higher than test accuracies, regardless of the architecture and hyperparameter values. Indeed, we would only expect to see such a large discrepancy if the test data were substantially different from the training data in their class representation or distribution of features. If this were the case, we would be severely limited in our ability to achieve high test accuracies, but a more thorough analysis of the test data is required to make this determination.

5 Appendix

5.1 Logistic Regression Class Accuracies

Figure 1: Class accuracies for each model and data type in Part A

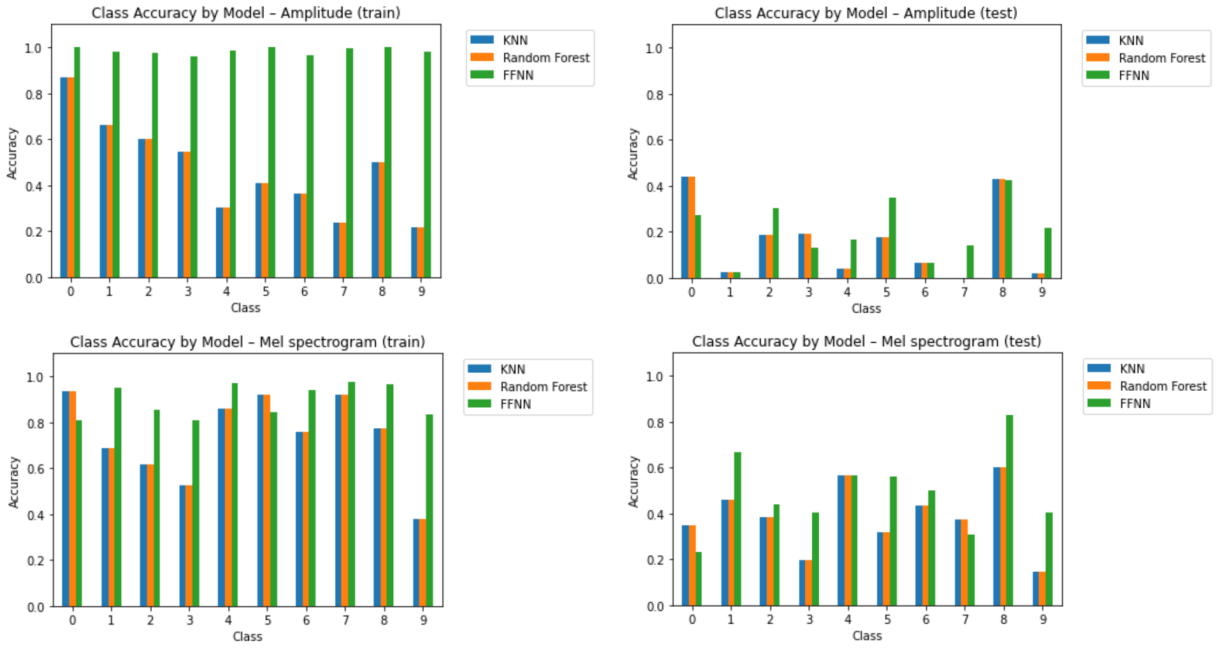


5.2 FFNN Architecture and Hyperparameters

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 500)               250500
_________________________________________________________________
dropout (Dropout)            (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 300)               150300
_________________________________________________________________
dropout_1 (Dropout)          (None, 300)               0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               30100
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1010
=================================================================
Total params: 431,910
Trainable params: 431,910
Non-trainable params: 0
_________________________________________________________________
```

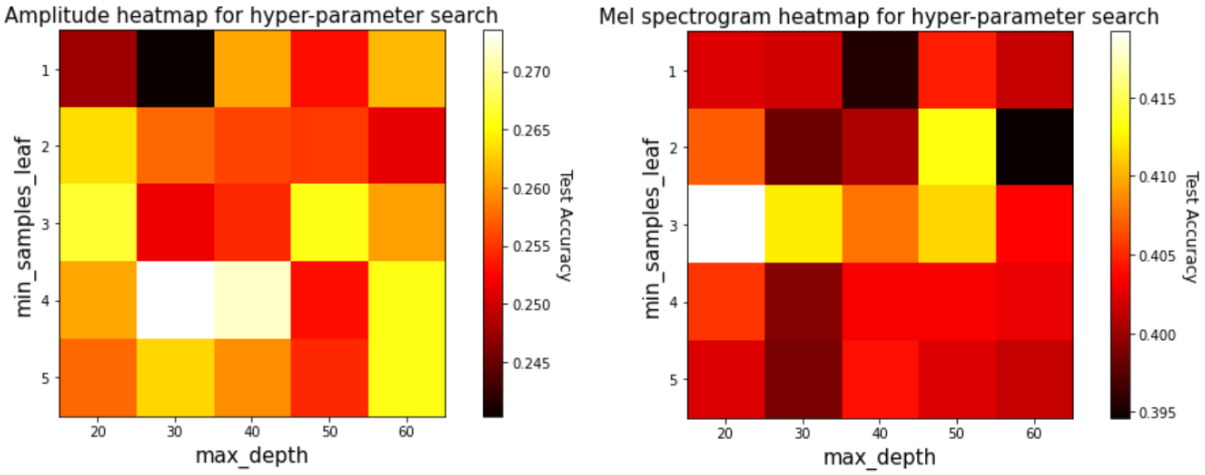
5.3 Nonlinear Model Class Accuracies

Figure 2: Class accuracies for each model and data type in Part B



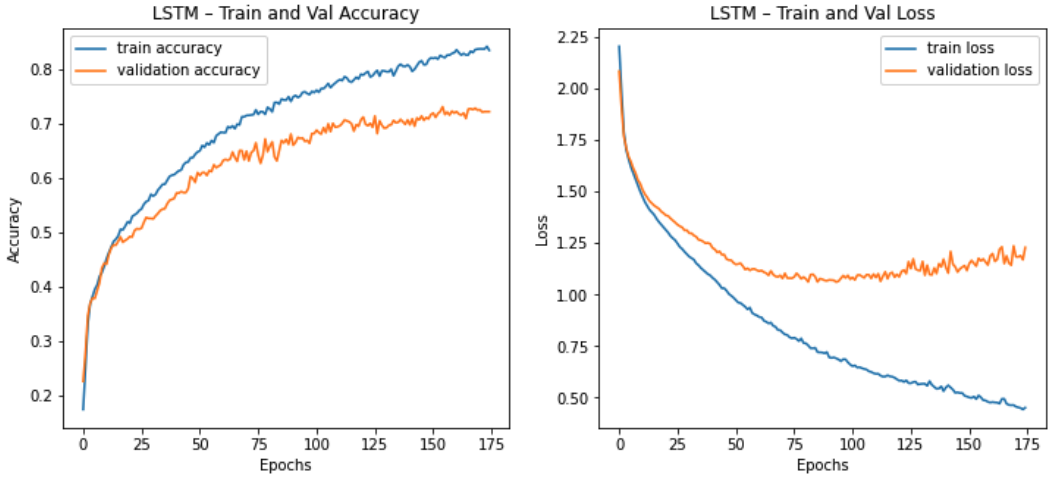
5.4 Hyperparameter grid search heatmaps

Figure 3: Heatmaps showing the test accuracies achieved using grid search on random forest hyperparameters



5.5 LSTM Training Results

Figure 4: Plots showing loss and accuracy during LSTM training



5.6 LSTM Architecture and Hyperparameters

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectional (None, 128)               77824
_________________________________________________________________
dense (Dense)                (None, 256)               33024
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 145,034
Trainable params: 145,034
Non-trainable params: 0
_________________________________________________________________
```

- Hidden state size: 128
- Dropout rate: 0.05