Practical: Sample Assignment

Christy Jestin
christyjestin@college.harvard.edu

David Qian
dqian@college.harvard.edu

May 7, 2022

This is the template you should use for submitting your practical assignment. Your write-up must answer all questions under the approaches, results, and discussion subsections for each of tasks A, B, and C.

If you used existing packages or referred to papers or blogs for ideas, you should cite these in your report. You must have *at least one plot or table* for each of the results sections in parts A, B, C that details the performances of different methods tried, such as one like table 1.2. Credit will be given for quantitatively reporting (with clearly labeled and captioned figures and/or tables) on the performance of the methods you tried compared to your baselines.

Note that the limit for the report is 4 pages, please prioritize quality over quantity in your submission.

1 Part A: Feature Engineering, Baseline Models

1.1 Approach

Regular PCA LR Train Accuracy: 0.407 Test Accuracy: 0.198
Mel PCA LR Train Accuracy: 0.713 Test Accuracy: 0.308
RF Train Accuracy: 1.0 Test Accuracy: 0.282

What did you do? When relevant, provide mathematical descriptions or pseudocode. Credit will be given for:

- PCA: Describe what the top 500 principal components represent, and how you computed them.

  The goal of PCA is to reduce the number of dimensions of the data, which decreases the computational complexity and also helps the model distinguish key features.

  Each principal component expresses a feature and is linearly independent from other components. The first 500 principal components are the components that account for the greatest variance in the data.

To compute the principal components, we used the PCA function from sklearn.decomposition, as well as makepipeline from sklearn.pipeline and StandardScaler from sklearn.preprocessing to zero the data (note that we didn't use unit standard deviation for StandardScaler).

We know that minimizing the reconstruction loss is the same as maximizing the variance. So, the PCA fit function takes the eigenvectors of the empirical covariance matrix of the original data that correspond to the 500 largest eigenvalues. Then it projects each data point onto the vectorspace created by the 500 principal components to get a lower dimensional representation of our data.

- Logistic regression: Describe how the model you trained predicts output probabilities for each class.
  For logistic regression, we used the LogisticRegression model from sklearn.linear_model and we used it to fit our transformed training data. The logistic regression classifier uses a weight matrix and a bias term to generate real valued confidence scores. There are ten such values for each input, and they express the model's confidence for the input being in each of the ten classes. Then it uses the softmax function to transform the scores and get probabilities such that each probability is between 0 and 1 and the sum of all class probabilities is equal to 1. Note that the softmax function preserves ordering i.e. higher real valued confidence scores yield higher probabilities. The loss function is negative log likelihood, and it rewards the model for higher confidence scores in the correct class. The model begins with a random weight initialization and then runs gradient descent to minimize the loss by changing the weights of the matrix and adjusting the bias.

1.2 Results

This section should report on the following questions:

- What is the **overall** and **per-class** classification accuracy of the models that you implemented?

Model	Overall Acc (%)	Class 0 Acc	Class 1	Class 2	Class 3	Class 4
Baseline 1	19.8	21.0	0	54.8	7.4	9.5
Baseline 2	30.9	19.3	59.0	52.8	18.3	33.0
Model 1	10.59	17.54	...			
Model 2	13.42	16.77	...			
Model 3	7.49	9.82	...			

Model	Overall Acc (%)	Class 5 Acc	Class 6	Class 7	Class 8	Class 9
Baseline 1	19.8	15.5	6.7	11.9	12.3	15.7
Baseline 2	30.9	33.0	56.7	26.3	47.0	17.0
Model 1	10.59	17.54	...			
Model 2	13.42	16.77	...			
Model 3	7.49	9.82	...			

The above tables are the test accuracy percentages of the baseline models that use PCA and logistic regression.

The overall accuracy for the logistic regression model trained on the Mel spectrogram (Baseline 2) did significantly better than the model trained on raw amplitude. Also, for individual class accuracies, the second baseline model either did about as well (off by 2 percent) or significantly better (in some cases, the class accuracies were better by 20 or more percent).

1.3 Discussion

This section should report on the following questions:

- Why do you hypothesize one feature representation performed better than the other?
- Why might have asked you to perform PCA first, and what is the impact of that choice?

The Mel Spectrogram performed better because the model was directly provided with information about the frequencies rather than having to infer them from the time and signal amplitude data. In addition, transforming the frequencies under the Mel scale is important because the audible differences in frequency parallel their graphical visualization on the spectrogram i.e. 200 Hz and 400 Hz are much more different than 9800 Hz and 10000 Hz. Formatting the data in a way that is more comprehensible to humans also works to assist the model in detecting patterns and learn better (https://towardsdatascience.com/audio-deep-learning-made-simple-part-2-why-mel-spectrograms-perform-better-aad889a93505).

It is harder to classify by learning with raw amplitude data because the graph plots amplitude vs time and thus does not distinguish the frequencies or pitches. On the other hand, the spectrogram shows the amplitudes of different frequencies over time, and not having this separation of frequencies is a key factor in the model's poor ability to perform sound classification.

Doing PCA first decreased the dimension of each data point from 44100 or 11136 (87×128) to 500. This significantly reduces runtime while not losing too information in the case of the logistic regression since it cuts down on the number of weights that need to be trained. However, running PCA means that we no longer have a time relation within the data. Before the 44100 data points represented consecutive, evenly-spaced discrete time steps, but the transformed 500 data points no longer have this interpretation. Similarly, the 87 sets of audio features represented overlapping time windows in the Mel data, but we lose this meaning when we flatten and run PCA. The loss of time information is especially relevant when we use models like RNNs and CNNs where relative positioning is the crux of the model. However other nonlinear models may also benefit from having direct access to the time information, and it turns out that this was the case for our Random Forest classifiers.

2 Part B: More Modeling

2.1 First Step

2.1.1 Approach

What did you do? Credit will be given for:

- Provide mathematical descriptions or pseudocode to help us understand how the models you tried make predictions and are trained.

For our first step, we decided to train a random forest classifier on the Mel spectrogram data. Random Forest takes data and uses multiple decision trees to classify datapoints. Every branch of each decision tree splits the data by a certain feature. The class decision of the Random Forest is the most common classification from all of the decision trees (https://towardsdatascience.com/understanding-random-forest-58381e0602d2). The model uses the Gini impurity as the loss function to measure the quality of splits at branches, which is calculated by quantifying the probability of an incorrect classification after a given split. We find better splits by increasing the Gini gain, which is the amount of impurity removed by a split (original impurity - weighted impurity of split) (https://victorzhou.com/blog/gini-impurity/).

For hyperparameters, we set the number of estimators, or number of trees, to 500 to ensure that we had enough trees for the random forest to benefit from multiple classifiers voting and set the max depth to 10 because this corresponds to $2^{10} = 1024$ leaves. We thought this would be enough leaves to perform well without the model just memorizing classes i.e. there will be several samples per leaf but few enough that each leaf will represent something meaningful. We checked our hypothesis that max depth of 10 leads to balanced accuracy and runtime in part 2, where we tuned the hyperparameters.

We also trained a convolutional neural network (CNN) on the Mel spectrogram data. CNN's are meant to check for patterns in the data while being agnostic about where the pattern appears in the data. For example, a CNN could help recognize a dog whether it appears on the left or right side of an image even though this would completely change the underlying pixel values. It does so by applying a kernel or filter to various parts of the data and then checking if a desired pattern appeared anywhere by pooling the filter outputs across the data. In our case, the CNN would be looking for a pattern at different times within the same clip. Additionally, we had the CNN treat the 128 audio features as different channels i.e. like RGB color channels in a photograph. Note that we only have 1 dimension (time) in this case, so our CNN is 1 dimensional rather than the usual two for images.

We followed the same general structure as the CNN in the practical section notes. I used a kernel size of 3 since there's only 87 time windows and a pooling size of 2 for the same reason.

Like the logistic regressor, the CNN model uses cross entropy loss to reward confident correct predictions and uses gradient descent to train (we used Adam as the optimizer and added a dropout layer to make the gradient descent more robust).

2.1.2 Results

This section should report on the following questions:

- What is the overall and per-class classification accuracy of the models that you implemented?

The Mel Random Forest test accuracy with max_depth set to 10 was about 47% and the training accuracy was about 95%. After 2000 epochs of training, the test accuracy of the CNN was about 47%, and training accuracy was about 77%. We suspect the training accuracy would get higher given more epochs, but it didn't seem worthwhile since the test accuracy had been more or less stagnant since epoch 250.

Train Accuracy for Class 0: 97 Test Accuracy for Class 0: 35.3 Train Accuracy for Class 1: 90.4 Test Accuracy for Class 1: 20.5 Train Accuracy for Class 2: 94.7 Test Accuracy for Class 2: 57.2 Train Accuracy for Class 3: 90.2 Test Accuracy for Class 3: 41.0 Train Accuracy for Class 4: 95.7 Test Accuracy for Class 4: 55.3 Train Accuracy for Class 5: 98.1 Test Accuracy for Class 5: 43.6 Train Accuracy for Class 6: 100 Test Accuracy for Class 6: 23.3 Train Accuracy for Class 7: 97.3 Test Accuracy for Class 7: 46.2 Train Accuracy for Class 8: 95.8 Test Accuracy for Class 8: 47.5 Train Accuracy for Class 9: 94 Test Accuracy for Class 9: 53

Model	Overall Acc (%)	Class 0 Acc	Class 1	Class 2	Class 3	Class 4
Baseline 2	30.9	19.3	59.0	52.8	18.3	33.0
Model 1 (RF)	46.7	35.3	20.5	57.2	41	55.3
Model 2 (CNN)	47	30	33	77	37	56

Model	Overall Acc (%)	Class 5 Acc	Class 6	Class 7	Class 8	Class 9
Baseline 2	30.9	33.0	56.7	26.3	47.0	17.0
Model 1 (RF)	46.7	43.6	23.3	46.2	47.5	53
Model 2 (CNN)	47	43	30	39	45	46

2.1.3 Discussion

Compare your results to the logistic regression models in Part A and discuss what your results imply about the task.

Both the random forest and the CNN do significantly better than the logistic regression models. We think that this means that linear models aren't as well suited for the task as non-linear models. Class 1 and 6 (car horn and gun shot respectively) were the only classes that had much better accuracies for the linear model. We hypothesize that factors that caused this might include that they are very distinct sounds and were more infrequent in the data.

2.2 Hyperparameter Tuning and Validation

2.2.1 Approach

What did you do? Credit will be given for:

- Making tuning and configuration decisions using thoughtful experimentation. How did you perform your hyperparameter search, and what hyperparameters did you search over?

  For the Random Forest, we focused on tuning the max depth. We did this by outputting training and test accuracies max depth values from 2 to 22 for every other value.

  For the CNN, we checked different kernel sizes in the list $[3, 6, 10, 12, 15, 20, 25, 30]$ and trained each model for 250 epochs. We didn't train any of models past this number of epochs since the test accuracy had already been stagnant for some time.

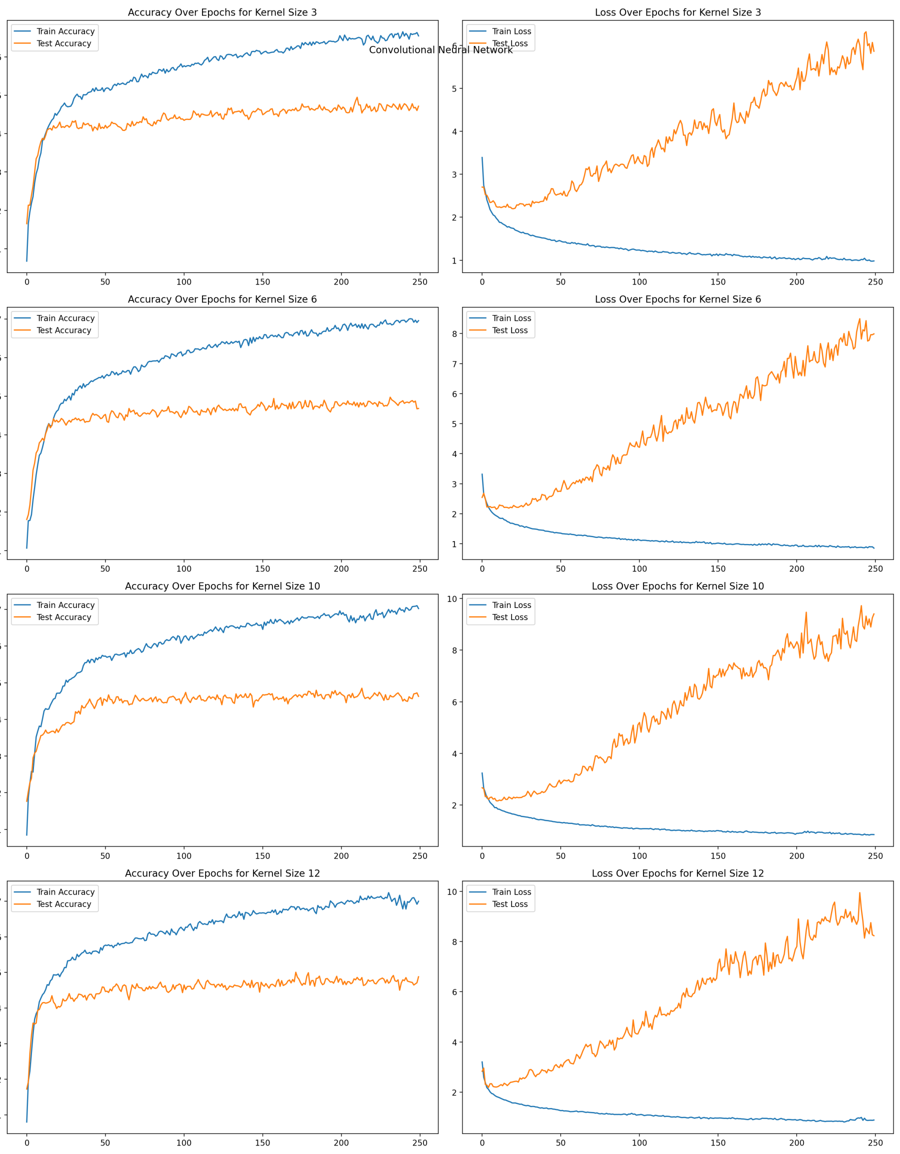
  The training and test accuracies began to plateau for max depth values greater than 10.

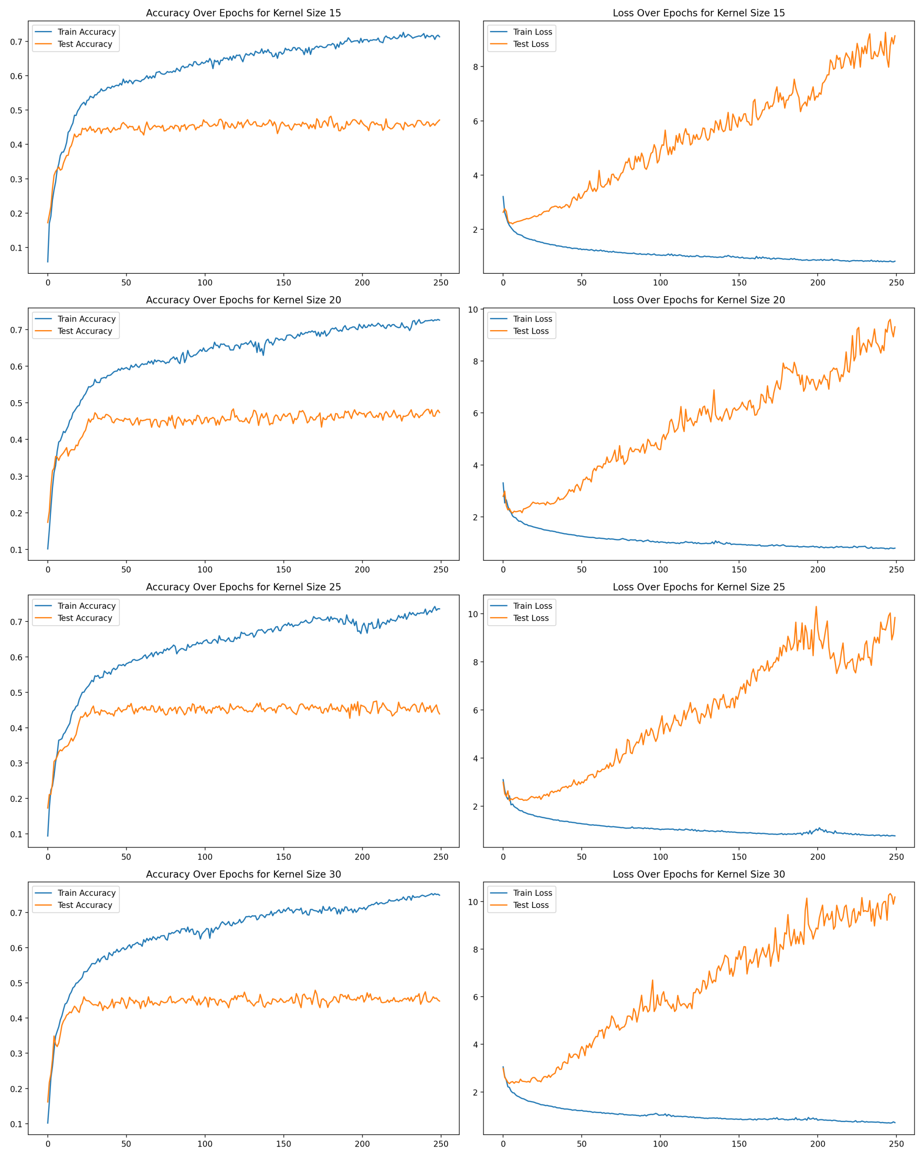
2.2.2 Results

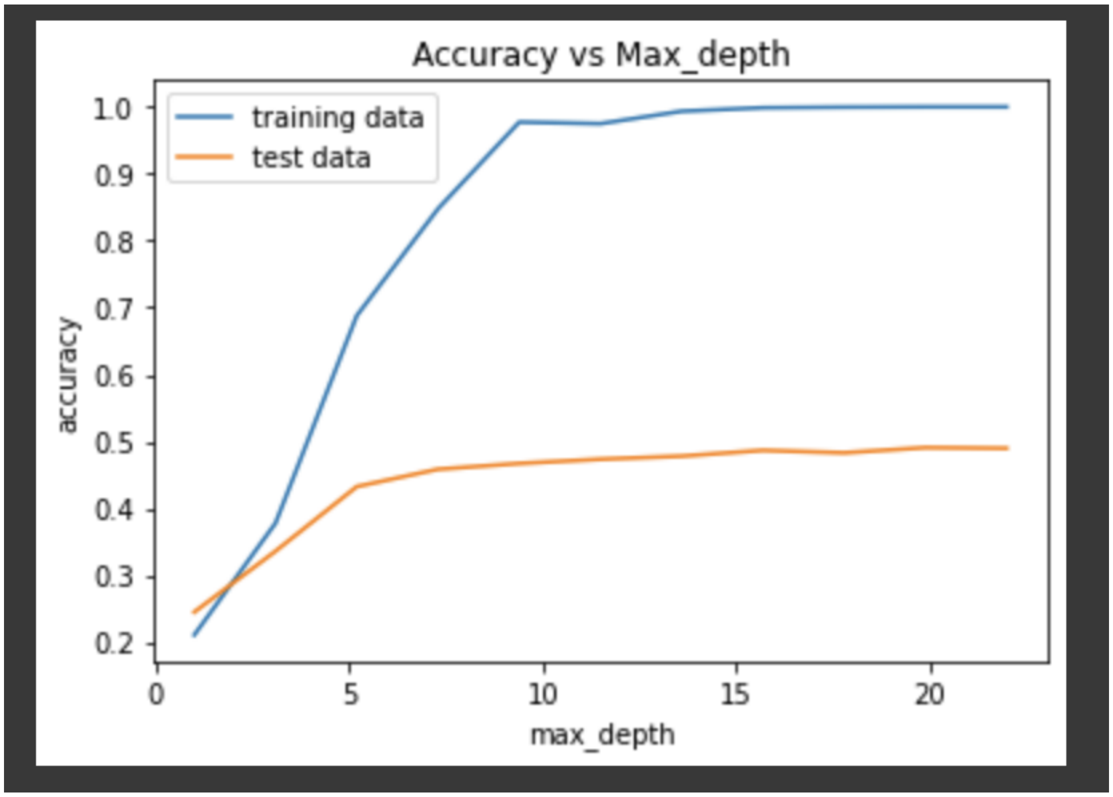
Present your results of your hyperparameter search in a way that best reflects how to communicate your conclusions.

Mel (No PCA) Class Accuracies Train Accuracy for Max Depth 2: 0.31820637493246895 Test Accuracy for Max Depth 2: 0.2872098315885298 Train Accuracy for Max Depth 4: 0.49432739059967584 Test Accuracy for Max Depth 4: 0.38188438780154754 Train Accuracy for Max Depth 6: 0.6828741220961643 Test Accuracy for Max Depth 6: 0.4283113336367774 Train Accuracy for Max Depth 8: 0.8467495047721952 Test Accuracy for Max Depth 8: 0.45607646791078743 Train Accuracy for Max Depth 10: 0.9524581307401405 Test Accuracy for Max Depth 10: 0.46700045516613564 Train Accuracy for Max Depth 12: 0.9882946155231407 Test Accuracy for Max Depth 12: 0.48475193445607645 Train Accuracy for Max Depth 14: 0.9972987574284171 Test Accuracy for Max Depth 14: 0.47792444242148385

These graphs show both loss and accuracy for the different CNNs:


Convolutional Neural Network
Accuracy Over Epochs for Kernel Size 3
Loss Over Epochs for Kernel Size 3
Accuracy Over Epochs for Kernel Size 6
Loss Over Epochs for Kernel Size 6
Accuracy Over Epochs for Kernel Size 10
Loss Over Epochs for Kernel Size 10
Accuracy Over Epochs for Kernel Size 12
Loss Over Epochs for Kernel Size 12
Train Accuracy
Test Accuracy
Train Loss
Test Loss


Accuracy Over Epochs for Kernel Size 15
Loss Over Epochs for Kernel Size 15
Accuracy Over Epochs for Kernel Size 20
Loss Over Epochs for Kernel Size 20
Accuracy Over Epochs for Kernel Size 25
Loss Over Epochs for Kernel Size 25
Accuracy Over Epochs for Kernel Size 30
Loss Over Epochs for Kernel Size 30
Train Accuracy
Test Accuracy
Train Loss
Test Loss



2.2.3 Discussion

Why do you expect the tuned models to perform better than the baseline models and the model used in First Step? Discuss your validation strategy and your conclusions.

The Accuracy vs Max_Depth graph shows that the accuracies have minimal improvement after max depth values of 10 and larger. However, the more levels of depth we allow, the longer the runtime is to train the model. Thus, having a max depth of 10 is a perfect balance of accuracy and runtime.

3 Optional Exploration, Part C: Explore some more!

3.1 Approach

What did you do? Credit will be given for:

- Diving deeply into all of the model classes and/or pre-processing algorithms that you tried (rather than just trying off-the-shelf tools with default settings). When relevant, provide mathematical descriptions or pseudocode to help us understand how the models you tried make predictions and are trained.

3.2 Results

Describe your results in a way that is appropriate for the experiments that you ran.

3.3 Discussion

Credit will be given for:

- Explaining the your reasoning for why you sequentially chose to try the approaches you did (i.e. what was it about your initial approach that made you try the next change?).
- Explaining the results. Did the adaptations you tried improve the results? **Why or why not?** Did you do additional tests to determine if your reasoning was correct?