CSC 375 Final Project:
Rainforest Audio Classification

Eva Wu, Frank Fang

December 2020

1 Background

The problem of environmental audio classification is an essential approach in much scientific research. Till today, many different signal processing and machine learning techniques have been applied to the problem. While there is much research related to audio fields such as speech and music, work on the classification of environment sounds is comparatively weak because obtaining data and reducing background noise is much more complicated than speech audio classification.

In this project, we are tackling an audio classification problem with deep learning methods. The project aims to train a model to classify sound files using deep learning methods. The data set we are using contains 16351 wave files, while 11444 wave files are in the train data set and 4907 wave files are in the test data set. Each wave file is initially about 2 seconds long with a default sampling rate of 44KHz.

2 Methodology

2.1 Data Cleaning

The data set we are using is an audio classification data set with originally eight classes. Within each class, about a thousand 2 seconds audio file was used as training data and 500 each class as testing data. Due to the imbalance of class 'Junk', which contains(2800 inputs) and class 'tingut' (674 inputs), we decide not to include those two classes. Another way to restore the balance is to undersampling the Junk (too much data) and oversampling the tingut data (too less data) by randomly deleting some data or replicating some data to make every class evenly in terms of training data. However, this may lead to overfit by duplicate and underfit by deleting observations. We tried to do eight classes at first; however, due to the too-large class 'Junk,' the prediction quickly became all towards that class, making the training impossible and the accuracy limited to 25 percent without any improvement over models. Thus, we are doing a multi-class classification for only six classes using this data set in this project. In future studies, it is possible to do an 8-class classification with a pre-processed balanced data set.

We have first loaded all the data using the librosa library from their folders and automatically labeled every wave file with the folder into a list. The original data was 44KHz each audio wave, but we decide to downsample it to 4KHz for faster training with a cost of lower accuracy and possible information loss. We randomly separate the training data into the training validation dataset using StratifinedShuffleSplit by a ratio of 4 to 1 (20 percent of the original data becomes validation

data). After that, we transformed all the labels to 6 dimensions (for six classes) using the to-one-hot function for training purposes.

2.2 Baseline Model

In this initial model, we train a simple CNN for audio WAVE from scratch. It's not an accurate model, and the biggest problem is that audio wave is sort of a sequential data, and CNN is not good at dealing with 1D sequential data. Since we are ultimately looking at how to train audio file with CNN, We've only used a few dense layers in this model, however, for a sequential data like this, RNN might be more appropriate, but still not perfect due to the fact that audio waves also have frequencies, amplitudes, pitches, all sorts of information that cannot be presented with 1D data form.

We implement the confusion matrix as the visualization in this project to comparing models and to visualize the improvement. The diagonal represent the target training outcome that the prediction matched the true label. As we can see from this model, it barely beats the accuracy just by chance (17.5%). Also, due to the fact that the features of audio waves are mostly sequential, there's barely an improvement over epochs in terms of validation accuracy, and basically no decrements in validation loss. This indicate that the model only learned some local features of the training data, but those features are not at all capable of being generalized to similar audio waves. According to its confusion matrix, we can hardly see any patterns there, it's basically just randomly guessing with a little differentiating ability.

Although this is a awful model to do audio classification, it points out that if we are going to use CNN to train audio waves, we need to make it a 2d image that CNN can takes in and train properly so that we can retain all the characteristics of audio waves like sample rate, frequencies, and pitches.

2.3 Simple CNN with Spectrogram

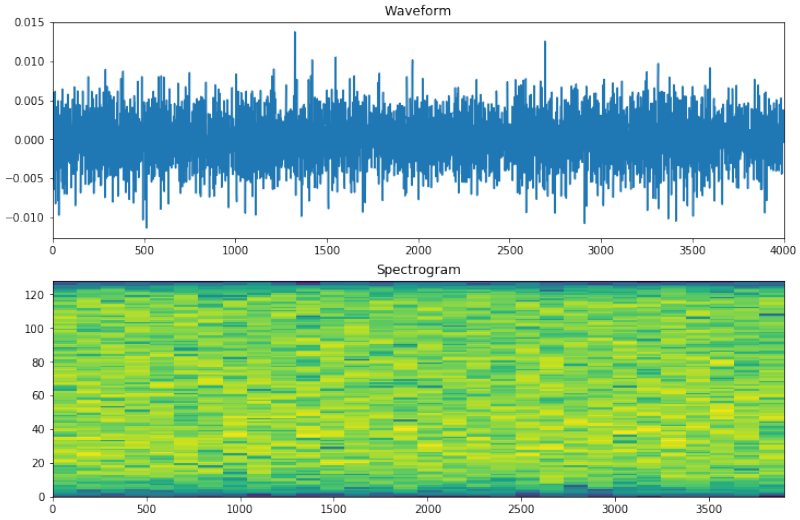

Figure 1: Waveform plot v.s. Spectrogram

In order to transform the audio waves to 2D images, we decided to use spectrograms. A spectrogram is a visual way of representing the signal strength, or "loudness", of a signal over time at various

frequencies present in a particular waveform. Not only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time. A spectrogram is usually depicted as a heat map, i.e., as an image with the intensity shown by varying the color or brightness.The vertical axis shows frequencies (from 0 to 12kHz), and the horizontal axis shows the time of the clip. Since we see that all action is taking place at the bottom of the spectrum, we can convert the frequency axis to a logarithmic one.

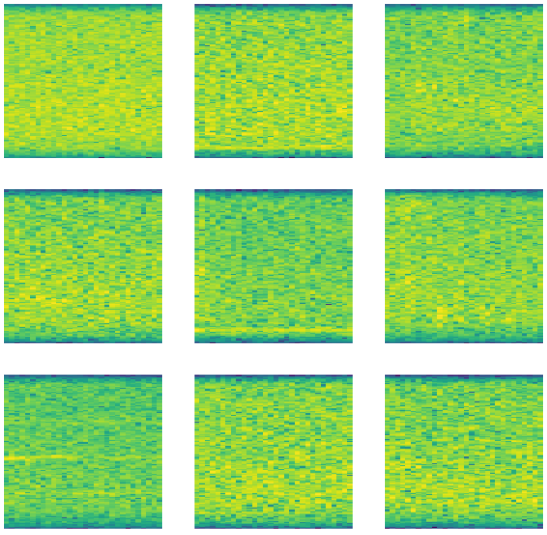

Figure 2: Spectrograms examples

In the simple CNN model, we first transformed all the audio waves in train, validation, and test data-sets into spectrograms with stft function in tensorflow library. Then we added two Conv2D layers before the dense layers in the model. Before entering the Conv2D layers, the spectrograms will be reshaped to 64 by 64 image for training purpose. The above spectrograms are the first 9 spectrogram from the training set to make sure the spectrograms have been successfully transformed.

2.4 Transfer learning with Spectrogram

For transfer learning in CNN, we tried both DenseNet and VGG16, and decide to use VGG16 in the end due to the computing power limit. The time consumption of DenseNet is expensive and it does not perform significantly better than VGG16's result. We uses the weight from imagenet for VGG16, although it's not specially designed for spectrograms so that it might not good at differentiating audio waves like it differentiate objects, we did find that the imagenet weights are still better than random weights if we not specify any. After that, we tuned the layers by de-froze 3 blocks (block 2, 3, and 4) in VGG16 to constructed a better model with hyper-tuning. The trainable weights increased to 30, with a cost of longer training time and a benefit of higher accuracy.

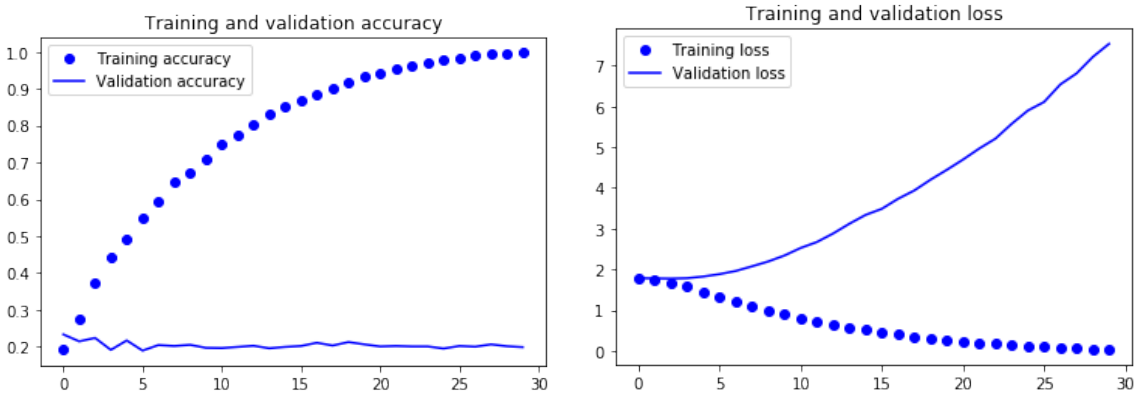

Figure 3: Accuracy and loss plot

3 Results

3.1 Results for Baseline Model

In the baseline model we implement, the training accuracy is increasing till 100 percent while the validation accuracy is stuck at 20 percent. The training loss is continue decreasing while the validation loss is increasing. Those plots show the baseline model perform poorly in this classification problem.

3.2 Results for Simple CNN with Spectrogram

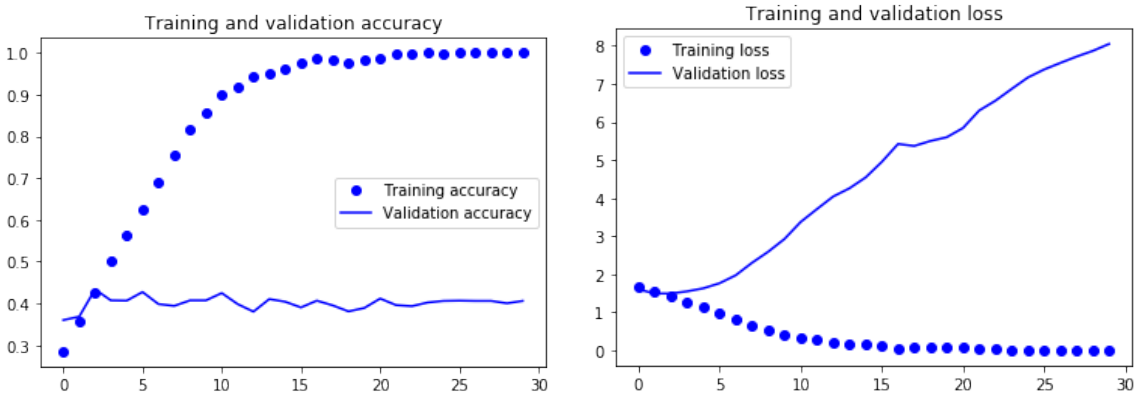

Figure 4: Accuracy and loss plot

As we can see from following accuracy and loss charts, this model is significantly improved compare to the previous 1D wave training model. The accuracy gets to around 40 to 45 percent. And the test accuracy is about 40 percent for this model. From the spectrogram we can see that the diagonal is more obvious in terms of color (lighter color suggests more matches). This shows that the model is onto something in differentiating the audio waves, however, there are still some heavily miss-matched classes, like cryvar and crystr. The model overfits at about 4th epoch. This model's improvement indicates the necessity of transforming audio waves to spectrograms. However, the problem of this

model is it's too simple in terms of layers. That might causes the problem of why it's overfitting too fast. Thus, in the next section, we utilized transfer learning to better construct the prediction model.

3.3 Results for Transfer learning

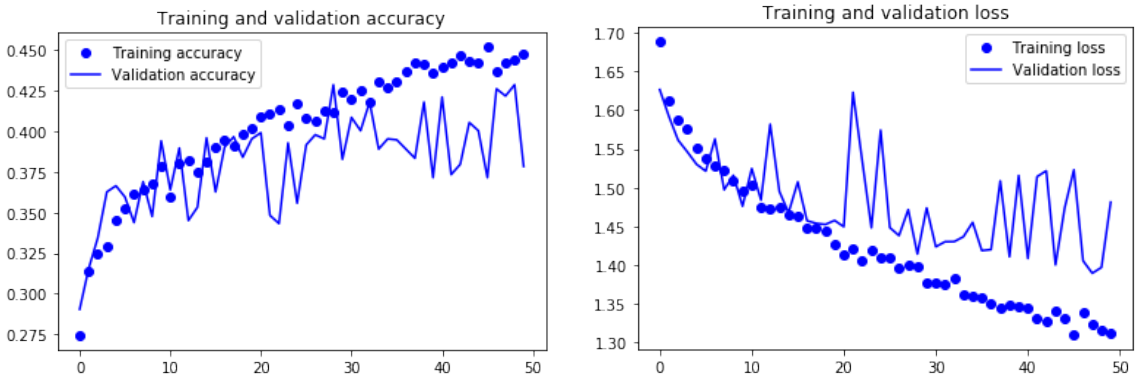

Figure 5: Accuracy and loss plot

For this model, we firstly converted all data to 3 channels instead of 1 channel. The input size is (30,129,3) for each spectrogram instead of (30, 129, 1). This is due to the fact that VGG16 requires a 3 channel RGB input image, even if spectrogram is essentially just a garyscale image, we have to replicate it 3 times to match the size requirement. We froze all VGG16 layers to begin with, and will tune the hyper parameters by defreezing some layers in the next section. When all layers frozen, only 8 weights can be trained in this model, which is faster of course, but less accurate for the specific data-set. The model did not seem to overfit in 50 epochs, but the increment in accuracy and decrements in loss does not seem to significantly improve after 50, so we decide to retain the 50 epoch number for the next hyper-tuned model.

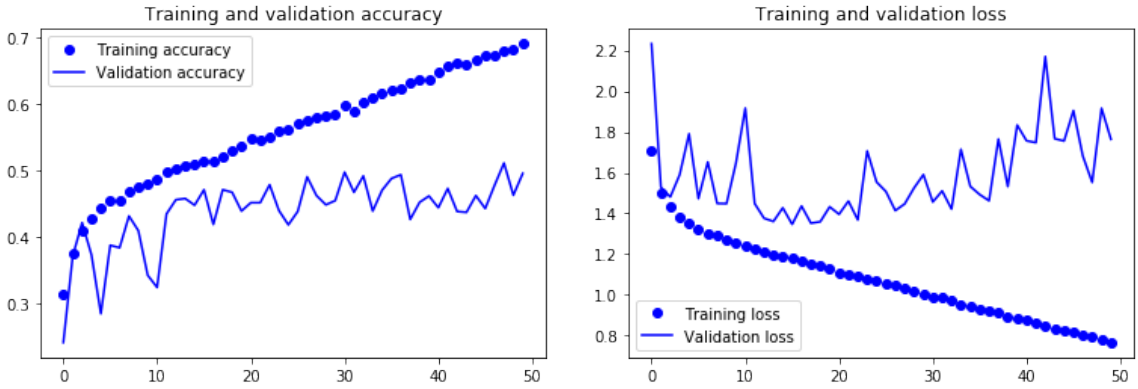

Figure 6: Accuracy and loss plot

We can see that the validation accuracy reaches the highest among all models to about 50 percent, which is not a large increase compare to previous model, but still a significant one that cannot

be ignored. Also, as previously illustrated, due to the downsampling from 40KHz to 4KHz, we've discarded basically 9/10 of the contained information in the actual audio files, this increase should be more obvious if we train with more data, also higher accuracy. Although to be said, this has a 50 percent of accuracy of predicting the testing data set, which is 10 percent higher than the simple CNN for spectrogram model.

4 Analysis and Discussion

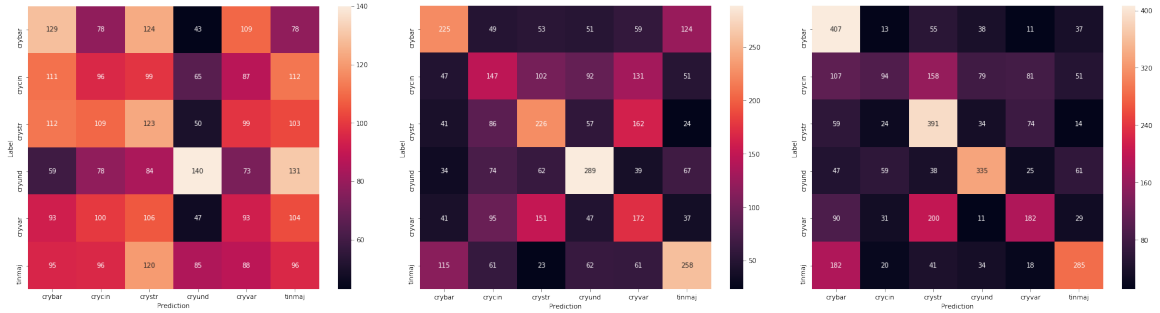

Figure 7: Confusion Matrix for baseline, Simple CNN, VGG16

The final model with tuned transfer learning gets to a accuracy about 50% for validation, and also a 50% accuracy of predicting the testing set. Compare to previous models, this is best in terms of both accuracy and loss.Due to our downsampling to about 1/10 of size, it's an acceptable level of accuracy. If using the whole data for training, the performance of the model can definitely gets way better than this.

As the confusion matrix suggest, among all six classes we trained and predicted, crystr and cryvar, crybar and tinmaj, and crystr and crycin, are more likely being wrongly classified in the data. However, comparing between the confusion matrix of simple CNN for Spectrogram and transfer learning, we can see that less crycin are being wrongly identified as cryvar, crybar and crystr have higher accuracy in terms of detection, but still a lot of tinmaj were wrongly predicted as crybar across models. Another big problem of this dataset is that the audios are all with a lot of background noises. We assume a noise reduction filter before training might also improve this model. For future works, the model can be trained with the complete dataset and the complete eight classes, we assume it can be more accuracy to at least 80 percent. Also, preprocess the audio waves with noise reduction filter would be a good choice for a better predictions. We did not manage to find an existing one that we can use, but with further research, it's possible to construct one on our own.

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[2] Albert Einstein. *Zur Elektrodynamik bewegter Körper*. (German) [*On the electrodynamics of moving bodies*]. Annalen der Physik, 322(10):891–921, 1905.

[3] Knuth: Computers and Typesetting,
`http://www-cs-faculty.stanford.edu/~uno/abcde.html`

[4] Palanisamy: Rethinking CNN Models for Audio Classification,
`https://arxiv.org/abs/2007.11154`