Sound Classifier Project

Subrat Mahapatra

Motivaton

This project starts off exploring the space of general purpose audio tagging, which has been generally overlooked when it comes to problems like computer vision. The motivation for this project is to understand the our environment in a way that images cannot capture. For example a clip of sounds at club with poor lighting could indicate popularity much better than analysis on low-lighting images can provide. Tagging sound clips also provides a perspective on information that is collected passively. For example, a grinder in the background of a coffee shop might indicate that the coffee is likely brewed fresh. In this example understanding these details about a coffee shop or a club might be a significant indicator of sales across a chain of coffee shops. A good model should not only be able to tag sounds with the appropriate label, but be able to differentiate between the different sounds that a microphone might pick up.

Problem Definition

The purpose of this project is to develop models that can accurately tag various sounds that can be found in the environment. More specifically, to be able to tag a dataset of sounds with the appropriate labels. The audio clips are from the Freesound General-Purpose Audio Tagging Challenge hosted by Kaggle. These clips are on average 6 seconds of audio along with their corresponding label. These clips only contain one type of sound, and one label for each file. This project is a critical component in general audio tagging because one specific sounds can be differentiated from each other, then future work could focus on denoising mixed sound signals.

Data

The dataset comprises of 9473 training examples and 9400 test examples. There are approximately 41 categories in this dataset. Of these 9473 training examples, 3710 examples are manually verified, so I only used these for training. The sound files had a resolution bit depth of 16 and were recorded with a sampling rate of 44.1kHz as a pulse code modulated format. Of these examples, I further selected the top 2 examples and visualized them (below FIgures 1 and 2). Since the test set given by Kaggle comprised of mixed audio data, I started off with a 30% test set of the training examples that contained either a saxophone or a violin (top 2) as seen in Figure 5.

Key Takeaways

My first approach consisted of including the entirety of the sound clip to train on. However these did very poorly. It turns out that sounds are not consistent. Sounds especially from instruments sound different in the beginning than the end. In fact, the beginning part of the sound clip seemed like the best indicator of the class of the label. After visualizing all of the different categories, I picked a duration of 2 seconds that balanced both the efficiency of the training time (# of epochs) and testing accuracy.

Upon visualizing the loss curve of the base model it looked like the accuracies were also not going high because the model was being overfit. So I built a model that was very similar to the base model but doubled the value of the dropout layers (Conv C in Figure 3.) There was no trace of being overfit but the train accuracies were not low.

I made a variety of model implementations that attempted to reduce the validation loss as I determined that it was a great indicator whether or not a model was overfit. I increased the number of layers (found in Conv B in Figure 3). I also implemented a flipped version of the base model (shown in Conv A in Figure 3). I also modified the activation functions and found that the linear activation function and the tanh activation function actually gave me the best results as can be seen in Figure 3. This beats the base layer's accuracy of 75.66.

Lastly I also wanted to check the variance of accuracies when I altered the learning rate. It didn't affect the accuracies much but that is mostly because of my optimizer of choice 'Adam' which varies the learning rate decay from the threshold set by the user. Toggling this threshold had effects but not drastic changes or increases in test accuracy as can be seen in Figure 3.

Future Work

This work is a fundamental step in audio tagging. The next step is to see if these patterns are replicable in multi label classification. This was limited in my exploration because of the complexity in training time that adding more labels and increasing the number of examples. Future work would move this training to a GPU enabled instance on AWS. Additionally future work would focus on mixing these sounds together and attempting to de-couple them. Assuming one could do that then general tagging could follow from de-coupling wave data and associating a tag to each type of wave. Additionally there has been some work with Convolutional 2D networks that have gotten higher accuracies. The increase in training time however prevented me from trying this out as a viable option. I would also do 10-fold cross validation to better reflect the test accuracy that I would receive on a new test dataset.

References

Thanks so much for Zafar's clear and concise explanations when it comes to dealing with sound! Sound was a new subject area for me, and I learned a lot through his explanations. Though his work was geared towards multiclass classification for mixed sounds, I was able to use his work as a foundation and shift the focus to audio tagging specific sounds.
https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-audio-data

Figures

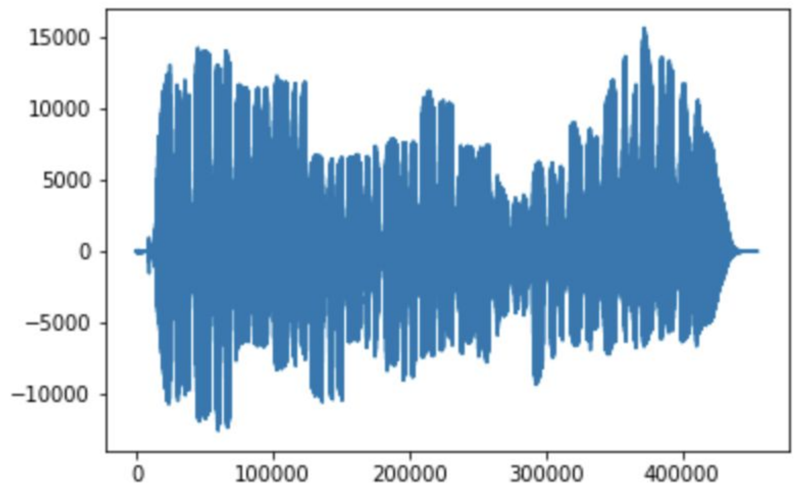


Figure 1. Wave file representing saxophone

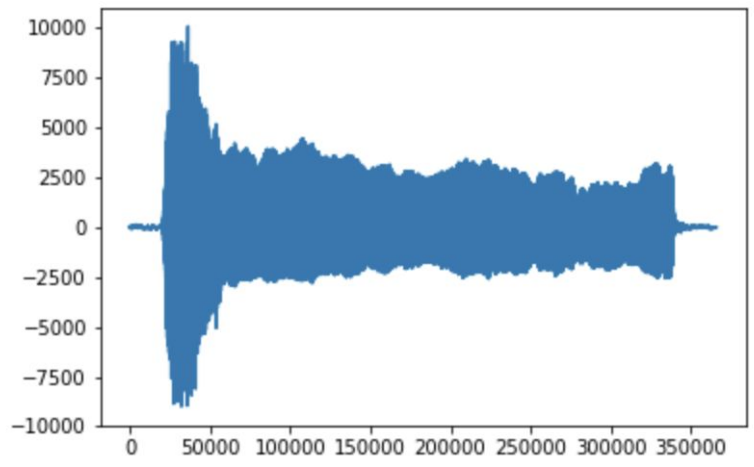
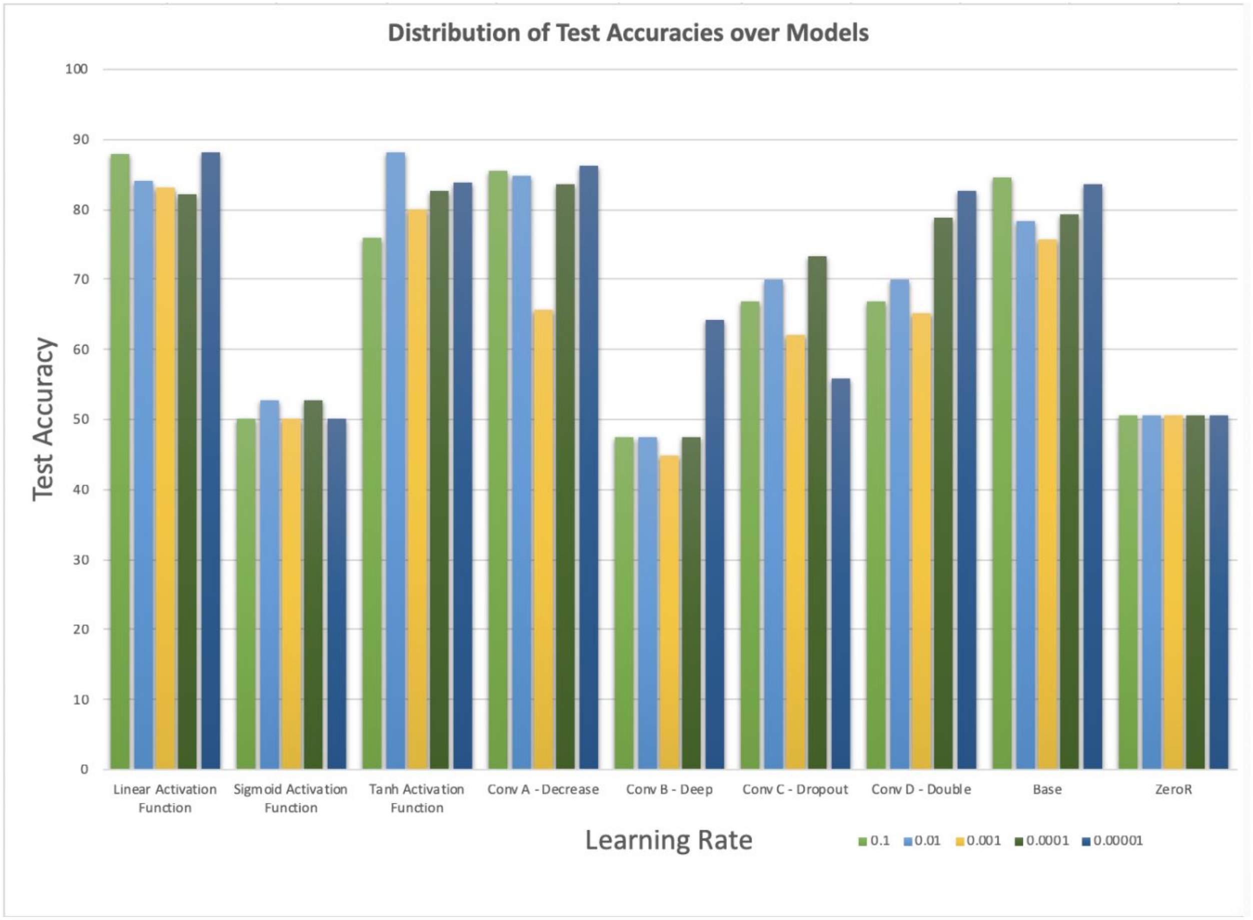
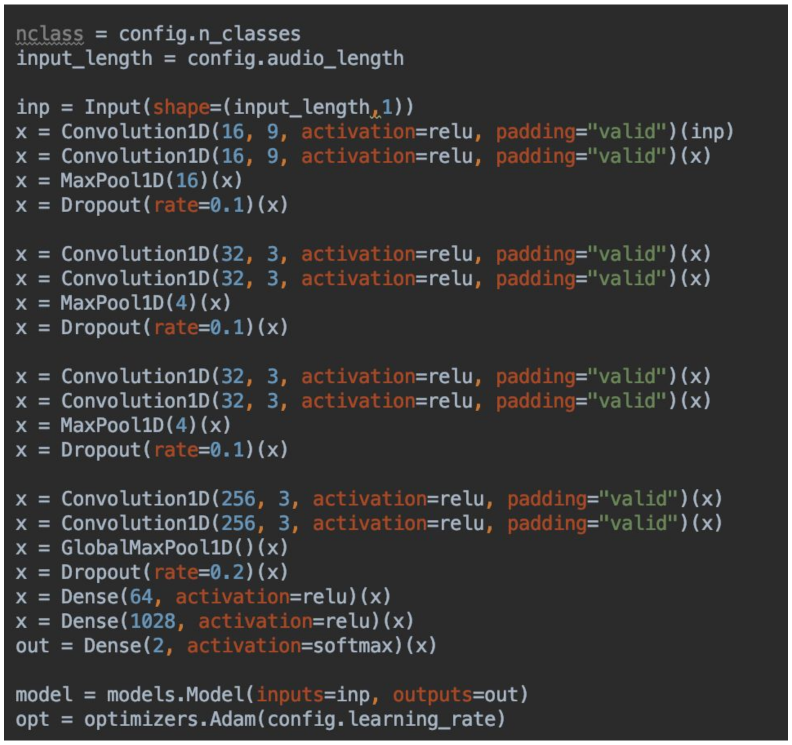


Figure 2. Wave file representing violin



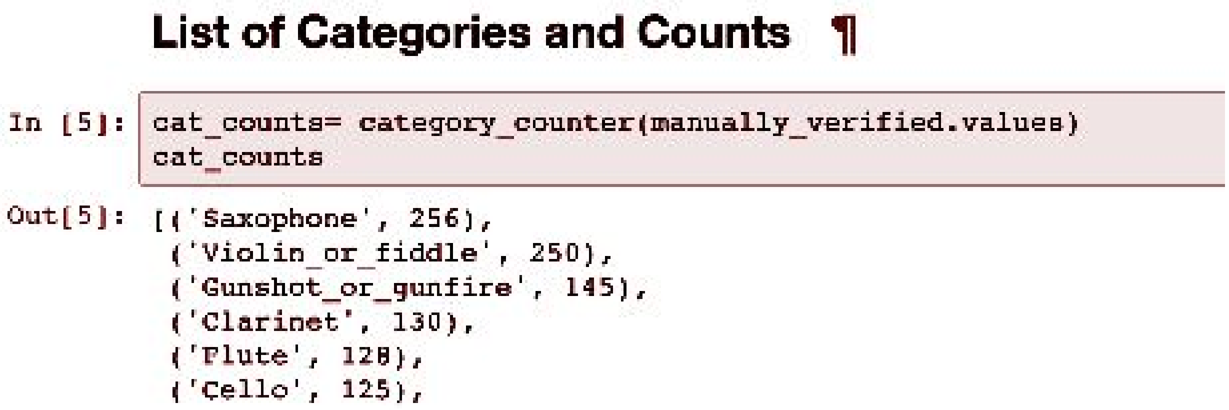
Networks	lr=0.1	lr=0.01	lr=0.001	lr=0.0001	lr=0.00001
Linear Activation Function	87.83	84.21	83.22	82.24	88.16
Sigmoid Activation Function	50.00	52.63	50.00	52.63	50.00
Tanh Activation Function	75.99	88.16	79.93	82.57	83.88
Conv A - Decrease	85.53	84.87	65.79	83.55	86.18
Conv B - Deep	47.37	47.37	44.74	47.37	64.14
Conv C - Dropout	66.78	70.07	62.17	73.36	55.92
Conv D - Double	66.78	70.07	65.13	78.95	82.57
Base	84.54	78.29	75.66	79.28	83.55
ZeroR	50.59	50.59	50.59	50.59	50.59

Figure 3. This shows the various test accuracies of the various models I built compared to those of ZeroR and the Base model.



```
nclass = config.n_classes
input_length = config.audio_length

inp = Input(shape=(input_length,1))
x = Convolution1D(16, 9, activation=relu, padding="valid")(inp)
x = Convolution1D(16, 9, activation=relu, padding="valid")(x)
x = MaxPool1D(16)(x)
x = Dropout(rate=0.1)(x)

x = Convolution1D(32, 3, activation=relu, padding="valid")(x)
x = Convolution1D(32, 3, activation=relu, padding="valid")(x)
x = MaxPool1D(4)(x)
x = Dropout(rate=0.1)(x)

x = Convolution1D(32, 3, activation=relu, padding="valid")(x)
x = Convolution1D(32, 3, activation=relu, padding="valid")(x)
x = MaxPool1D(4)(x)
x = Dropout(rate=0.1)(x)

x = Convolution1D(256, 3, activation=relu, padding="valid")(x)
x = Convolution1D(256, 3, activation=relu, padding="valid")(x)
x = GlobalMaxPool1D()(x)
x = Dropout(rate=0.2)(x)
x = Dense(64, activation=relu)(x)
x = Dense(1028, activation=relu)(x)
out = Dense(2, activation=softmax)(x)

model = models.Model(inputs=inp, outputs=out)
opt = optimizers.Adam(config.learning_rate)
```

Figure 4. Base model taken from Zafar's Kaggle Blog (https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-audio-data)



List of Categories and Counts ¶

```
In [5]: cat_counts= category_counter(manually_verified.values)
        cat_counts

Out[5]: [('Saxophone', 256),
         ('Violin_or_fiddle', 250),
         ('Gunshot_or_gunfire', 145),
         ('Clarinet', 130),
         ('Flute', 128),
         ('Cello', 125),
```

Figure 5. This shows the count per categories for a few of the categories.

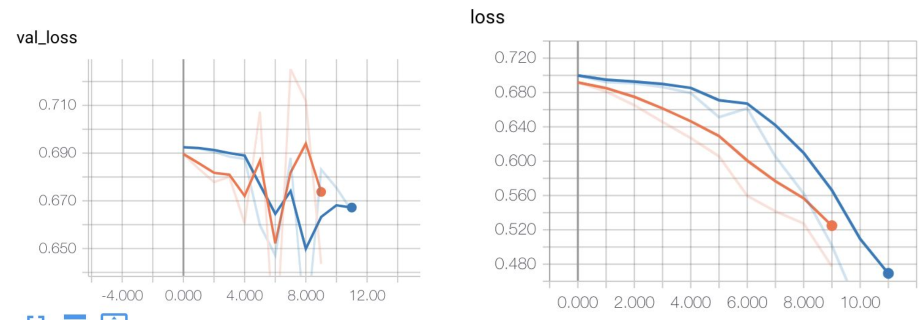


Figure 6. This shows the validation loss and the training loss of Conv C (Figure 3). Increasing the dropout layer did in fact reduce overfitting but the training accuracies were low. The different colors here symbolize the different folds of the cross validation.

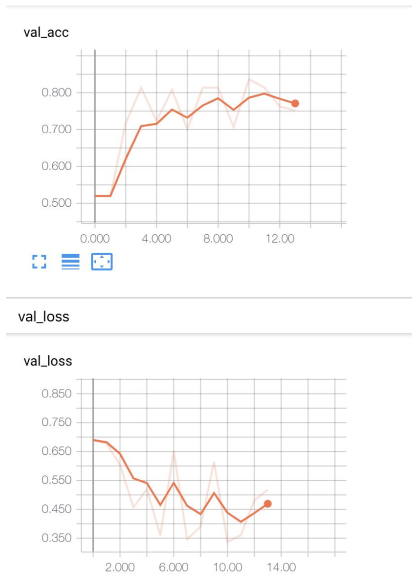


Figure 7. Model with the Tanh activation function. Outperformed the base model.