Music Genre Classification Project Report

Our group addressed problem of music genre classification. Music classification is an essential tool used for informing music recommendation systems and providing swift searches of large music databases. Audio event classification in general can be applied to a number of tasks, such as identifying the migration patterns of birds, among many others. Our project utilized transfer learning to train a convolutional neural network for a multi-label music genre classification problem, using each audio sample's mel spectrogram images as features. The model was based on the VGG model trained for audio files with 'audioset', namely "VGGish". Our dataset was the magnatagatune dataset obtained from the MIREX site, and consisted of over 20,000 mono audio files each of 30 seconds long, of which we used ¾ for training and ¼ for testing. The dataset consisted of raw audio files, classified into 188 genre classes specified in a csv file. We trained the network using a GPU installed on a virtual machine instance provided by Google Cloud Platform.

The feature extraction method follows closely from the VGGish network Github repository, since the extraction methods are already tuned to the network. For each audio sample, we used librosa to load it in, preprocessed it by dividing into 30 roughly 1-second(0.96 seconds) segments. Then we extracted the mel spectrograms of each segment, designating resolution to be 64 frequency bins and 96 time frames. As a result, the features extracted for each audio clip would be a tensor of (30,96,64). Finally we randomly selected one (96,64) slice out of this tensor and designated it as feature matrix of the audio clip in interest. To improve training speed, we preprocessed all the audio files and stored the mel data along with the class label in Pickle files on our virtual machine.

In order to use batch processing, we implemented a custom data generator with pescador, which multiplexed 20 active streamers that stochastically fetch spectrograms from training and testing sets. With 12 epochs and batch size of 32, the model is trained iteratively using the adam optimizer and binary cross entropy loss function.

In preliminary experiment, we added 2 sets of batch-normalization + drop-out + fully connected layers(one using a rectified linear unit activation function and the ending dense layer using softmax activation function) on top of the VGGish model. Six convolutional layers and 4 max pooling layers are interlaid in the VGGish model, followed by a global average pooling layer. The accuracy of this model started near 98% and did not stray very far from this throughout training. We hypothesize that the model was fairly accurate at the onset due to VGGish having already been trained on a large number of audio signals. Overall loss of the model kept decreasing until around the 9th epoch. This preliminary training used training set size / batch size steps per epoch, which exceeds the actual number of pickle files we've generated, as we found out later. Thus, the training session performed more iterations of optimizations and the loss converged more noticeably. One odd result from this model is that the validation accuracy is constantly higher than accuracy. After researching, we found out that the addition of dropout layers prompted this to happen, as all features are used during test and yields more robust results.

In the following experiments, we revised the steps per epoch to actual number of pickle files divided by batch size, the accuracy is still around 0.9818 and loss became stagnant around 9th epoch. After that, we first tried adding the third set of batch-normalization + dropout + fully connected layers, and training the model only for 5 epochs. Comparing with the first 5 epochs of previous models, the result hasn't improved much. Loss even converged slightly slower than training with 2 sets of regularization layers, however test accuracy was able to be kept above 0.9819. Then we tried using the autopool layer to replace the last global averaging pooling layer in VGGish.