How Are You Feeling?
Inferring Mood from Audio Samples

Joel Haynie
Department of Computer Sciences
University of Wisconsin, Madison
jhaynie@wisc.edu

Ankit Vij
Department of Computer Sciences
University of Wisconsin, Madison
vij2@wisc.edu

Amanpreet Singh Saini
Department of Computer Sciences
University of Wisconsin, Madison
saini5@wisc.edu

Eric Brandt
Department of Computer Sciences
University of Wisconsin, Madison
ebrandt@wisc.edu

Abstract

Classification of audio samples is an application of deep learning that is receiving considerable attention today. Our project involves investigating this specialized area of machine learning in detail. First, we explore the background and theoretical basis for audio sample classification using deep learning methods. Second, we implement and train a practical proof-of-concept application that classifies musical samples into one of four 'moods' and report test. Finally, we comment on emerging research topics in audio classification using deep neural networks.

1 Motivation and Theoretical Basis

Our goal for this paper was to answer the following question, *How does state-of-the-art audio clip based Machine Learning (ML) categorize the emotion / feeling of a song?* Our research began by reading Hershey et al. [8] influential paper on large scale audio classification. This paper proposes using a Convolutional Neural Network (CNN) to aid in the classification. The data flow from this paper can be seen in figure 1. Our diagram was derived from the supplied code as the paper did not fully describe the data flow. We can see that the flow is broken up into two key parts the preprocessing, via spectrographs, and the feature building, via CNN.

As we continued to look for research, we found differing opinions within the ML community concerning the suitability of CNNs for audio-based training data. A key dissent was written in a series of articles by Rothmann [10], where he postulates that the present use of CNNs to do audio classification is insufficient and inaccurate. He proposes a solution pictured in figure 2. This data flow, he argues, more closely maps to the human ear.

Upon reading the works and looking closely at the two diagrams we see that the two approaches are nearly equivalent. The GammaTone Filterbank and Dilated Circular Buffer proposed in [10], are equivalent to well configured spectrographs from [8]. More detail about *well-configured* spectrographs can be read in section 1.1 Preprocessing below.

In both approaches the preprocessed frequency domain data is fed into a learned model that reduces to a feature space for later stage classification. Hershey et al.[8] uses a CNN as seen in figure 3 to do this feature reduction. Rothman [10] proposes one half of an Long Short-Term Memory (LSTM) AutoEncoder. This implies that the key functional differences are the structure of the network and how the models are trained, CNN versus LSTM AutoEncoder.

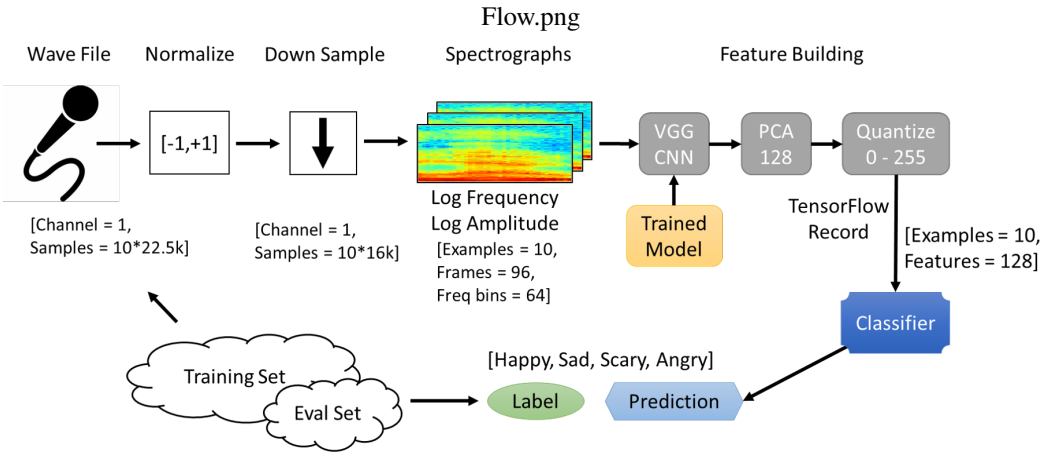

Figure 1: Hershey et al. [8] model of converting `.wav` audio to features suitable for ML

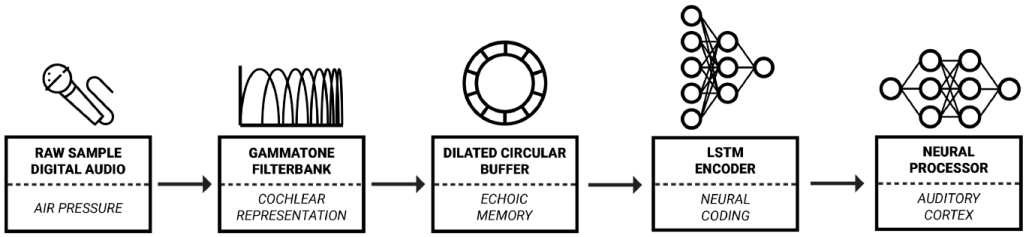

Figure 2: Rothmann's [10] ML model of human hearing

The Purpose of an LSTM AutoEncoder is to take a large input space, convert it to a smaller representative space in order to return an approximation of the original input space. Success is measured by how closely the large input space matches the large output space. Once this intermediate representation space is obtained one uses that smaller space as the feature space for feeding into the classifier. Observe that no labelled data is needed to train the network. Training a CNN needs labelled data to train the network.

Finally as an alternate view from the above two strategies there may be no need for these preprocessing stages at all. Dörfler et al. [4] prove, and Gong et al, [6] demonstrate that the spectrographs themselves can be removed by well defined layers in the feature reducing CNN.

1.1 Preprocessing

There are two key pre-processing portions referenced in [10]: GammaTone Filterbank and Dilated Circular Buffer. Let us discuss how these compare to the Spectrograph as built in Hershey et al. [8].

Let us start by comparing GammaTone Filter banks(GFCC) to the Mel-Frequency Cepstral Coefficients(MFCC) used in each spectrograph column. Burgos [2] studied this in depth and found that the The MFCC are typically the de facto standard for speaker recognition systems because of their high accuracy and low complexity; however they are not very robust at the presence of additive noise. The GFCC features in recent studies have shown very good robustness against noise and acoustic change. He further shows that there is about twice the performance improvement of GFCC over MFCC for very low, < -30dB, SNR values. He also shows there is little measurable difference for sounds where the Signal-to-Noise Ratio (SNR) goes above -5dB to 100dB. The SNR of music can vary widely, thus over the vast majority of the range these two strategies have comparable outcomes.

Now let us focus our attention on the Dilated Circular Buffer and the specific hyperparameters used to calculate the spectrographs. Rothmann [10] states that the Goal of the Dilated Circular Buffer is to emulate the echoic memory of human hearing. According to him echoic memory can be considered as a short buffer of immediate auditory impressions. He clarifies that the biological research shows

the duration of memory can vary from as short as 250 ms to 4 seconds. The Circular Buffer is a historical view of the frequency spectra, this is the definition of a spectrograph with zero window overlap. In order to bound the size of the input set to the LSTM, he chooses to make the buffer *Dilated*. This strategy averages adjacent historical values into summary GFCC spectrums. His input size is large because the temporal window's length he applies the GFCC on is 5ms. The key spectrograph hyperparameters are a frame or window length of 25ms and a hop length of 10ms. The window length defines the *resolution* of the spectrum. That is the smallest frequency that can be binned. This affects the quality of the vertical axis of the spectrograph. The hop length however reuses data already used in previous spectrum calculations which has the effect of blurring or averaging events in the spectrum. This yields a similar effect as Dilation. Finally, the spectrograph's used in [10] are 0.96 second in duration. This is right in line with a *echoic memory* of the human ear.

1.2 VGGish's CNN

In Hershey et al.[8] the CNN is used to reduce the large input space, 10 x 6.1k x 4 byte values down to a more manageable and concise 10 x 128 byte feature vector. The shape and meaning of the data that leaves the preprocessing portion is 10 examples of 2-D log-mel-spectrograms of 96 overlapping frames covering 64 mel-log frequency bands. The CNN's structure can be seen in figure 3. Specifically, the CNN contains all convolutions of 3x3 filters with strides of 1 and max pooling operations with 2x2 filter with strides of 2. With this configuration, the CNN contains in sequence, a convolution layer of 64 filters, a max-pooling layer, another convolution layer with 128 filters, another max-pooling layer, two convolution layers with 256 filters each, another max-pooling layer, another two convolution layers with 512 filters each, a max-pooling layer, from which the output is flattened and fed into two fully connected layers with 4096 outputs each, sequentially, and finally the output goes into a fully connected layer (the embedding layer) with 128 output units. The CNN produces an output of 10 examples of the activations of the 128-D embedding layer. This embedding is the 10x128 element feature vectors that can now be fed into the final stage, the classifier.

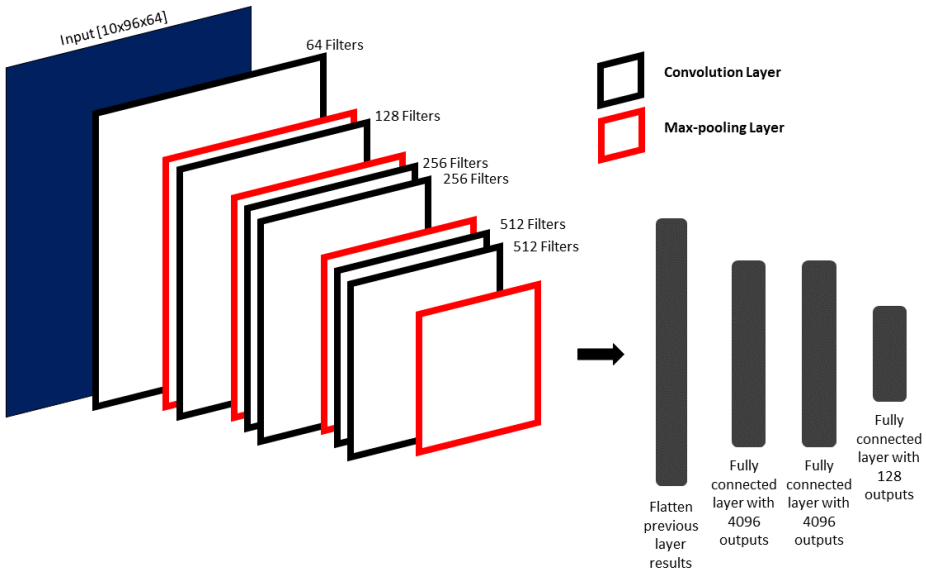

Figure 3: Deep network layers of the VGGish Convolutional Neural Network

2 Practical Implementation

2.1 Data Acquisition and Extraction

We collected our data from Google's AudioSet [7] which consists of an expanding ontology of 632 audio event classes and a collection of 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a

wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds. From the dataset, we focused on music mood samples and extracted data instances having labels of four mood classes- *Happy*, *Sad*, *Angry*, and *Scary*. The dataset selected was divided into two groups: training and evaluation.

To form our data sets, we filtered two disjoint sets from the ontology: First, a set of 400 instances with our corresponding labels for training. Of these 400, we ensured we chose 100 entries of each mood classification. The second set contained 223 data points with 56 instances per label which we used for evaluating our models. To avoid contamination, these 223 data instances were held aside during training and not introduced to the model until the evaluation stage. We created `.csv` files containing, for each instance, the YouTube video ID, start and stop times within the video, and the label associated with the instance. These `.csv`s were then used to run a batch processing script to download the audio samples in `.wav` format directly from their source (YouTube).

2.2 Transformation to Features

The next step in the application pipeline is to transform the raw `.wav` files into features that can be input into a classifier network. This is done via the process described above in sections 1.1 and 1.2, using an open source TensorFlow model called VGGish [5]. We made initial attempts to implement this layer ourselves, but the training required for the CNN to achieve suitable results is beyond the computational capacity of our local workstations. In this instance, 'to see further, we stood on the shoulders of giants [9]' (in this case, Google) and used the pre-trained CNN model to feature-ize the `.wav` files. We converted each of our 400+223 `.wav` file instances using VGGish to 10x128 element feature vectors.

2.3 Classification

After initial processing of the wav files into feature vectors (matrices) of dimension 10x128 we investigated different models to perform the multi-class classification task of predicting the *mood* of the music from the feature vector.

To evaluate different classification models, Python was used, enlisting the libraries TensorFlow [1] and Keras [3].

The 400 training samples were evenly divided by class for stratified cross validation and used to train 3 different neural networks:

1. Simple multi-class logistic regression classifier
2. 1-Layer LSTM (Long-short term memory) recurrent neural network
3. 3-Layer LSTM (Long-short term memory) recurrent neural network

In each case, the model was trained using batches of 40 samples, randomized at each presentation, for a sufficient number of epochs to infer steady-state accuracy.

We evaluated the performance of each of the 3 models by two methods:

1. Validation set accuracy over an increasing number of epochs to watch for over-training and the any generalization gap.
2. Evaluation on our Test Set of 223 never-seen-before data instances.

The performance of the training sessions is shown in figure 4.

After training, the evaluation on the 223-instance test set, we observed the accuracies shown in table 1.

Finally, to make sure that our four chosen classes do not have an abnormal correlation between any combinations of classes, we also computed confusion matrices for the models. The confusion matrix for 1-LSTM (arguably the best performing classifier) is shown in table 2.

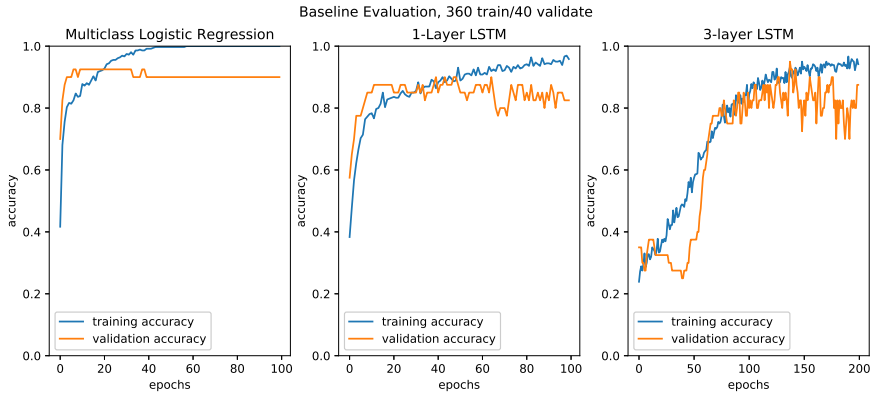

Figure 4: Training performance for 3 models.

Table 1: Accuracy on held-aside test set of 223 instances for 3 models.

Model	Accuracy
Logistic Regression	0.798
1-Layer LSTM	0.839
3-Layer LSTM	0.776

Table 2: Confusion matrix for 1-LSTM classifier of 223 test instances.

	Happy	Sad	Angry	Scary
Happy	45	9	2	1
Sad	11	39	2	4
Angry	0	1	53	4
Scary	0	7	3	42

3 Conclusion

3.1 Discussion

After evaluating Logistic Regression, 1-Layer LSTM, and 3-Layer LSTM learning models on the evaluation set, we can draw some important conclusions. First, the input feature sets (output of VGGish) must already be nearly linearly separable, as evidenced by the strong performance of the simple multiclass logistic regression classifier. VGGish's preprocessing of the raw audio waveforms into 10x128 element feature vectors clearly has produced data that is well separated without significant further processing. Evidence of the linearly separable feature data is supported by the fact that much more complicated non-linear classifiers (1-Layer LSTM and 3-Layer LSTM) do not yield better performance. Second, it is also clear that complicated models with more parameters, particularly the 3-Layer LSTM take significantly more time to train. There is evidence that the LSTM models are subject to overtraining at higher numbers of epochs, as validation set accuracy decreases somewhat with increasing number of epochs. This leads us to believe that for the practitioner who wishes to write an audio-based ML application and use AudioSet and VGGish as a 'black box', he or she is likely to do very well with only a simple linear classifier at the end. Finally, specific to our data set, the confusion matrix suggests, surprisingly, that 'Happy' and 'Sad' are the most often confused classifications, and that 'Scary' and 'Angry' are comparatively easy to predict.

3.2 Future Work

In the future, we plan to extend the capabilities and accuracy of our model and by including recent developments taking place in the field of sound recognition. Tokozume et al. [11] talk about Between-Class learning (BC learning) which is a novel learning method for deep sound recognition. In a similar

direction to overcome undesirable behaviors such as memorization and sensitivity to adversarial examples by neural networks, Zhang et al. [13] propose a simple learning principle called mixup. Both approaches train neural networks on convex *combinations* of pairs of examples and their label. This creates a new feature space by combining sounds from different classes. This would not only provide variations in feature space and remove sensitivity to adversarial examples, but will also regularize the positional relationship between the class feature distributions. The results from the two papers also show that the model provides better performance, so exploring this direction could provide better accuracy in mood detection from audio.

Another direction that is also very appealing is the use of speaker adaptation and regularization techniques for neural networks for sound recognition. Tomashenko et al. [12] propose a novel way to improve speaker adaptive training for neural network acoustic models using GMM derived features for automatic speech recognition. Taking inspiration from this work, we can also extend our system to detect mood from user's speech using the techniques as suggested by Tomashenko et al. which will improve its capabilities of mood detection beyond music.

As a future application, the detection of mood from audio signals could be put to use for various recommendation systems. One possible application is the mood based song recommendation service and another could be targeted product recommendation based on inferred mood of the listener. There could be other possible applications like depression detection systems based on the user's choice of audio playlists etc.

References

[1] Google Brain. Tensorflow: An open source machine learning framework for everyone, 2018. URL `https://www.tensorflow.org`.

[2] William Burgos. Gammatone and mfcc features in speaker recognition, 2014. URL `https://pdfs.semanticscholar.org/5ac9/5dcd2dc86cd435824e8fd3fc66fb3480a7fd.pdf`.

[3] François Chollet. Keras: The python deep learning library, 2015. URL `https://keras.io`.

[4] M. Dörfler, R. Bammer, and T. Grill. Inside the spectrogram: Convolutional neural networks in audio processing. In *2017 International Conference on Sampling Theory and Applications (SampTA)*, pages 152–155, July 2017. doi: 10.1109/SAMPTA.2017.8024472.

[5] Dan Ellis, Shawn Hershey, Aren Jansen, and Manoj Plakal. Vggish tensorflow model, 2018. URL `https://github.com/tensorflow/models/tree/master/research/audioset`.

[6] Yuan Gong and Christian Poellabauer. How do deep convolutional neural networks learn from raw audio waveforms?, 2018. URL `https://openreview.net/forum?id=S1Ow_e-Rb`.

[7] Google. Audioset: A large-scale dataset of manually annotated audio events, 2018. URL `https://research.google.com/audioset/index.html`.

[8] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. URL `https://arxiv.org/abs/1609.09430`.

[9] Isaac Newton. Quotes attributed to isaac newton, 1727. URL `https://en.wikiquote.org/wiki/Isaac_Newton`.

[10] Daniel Rothmann. What's wrong with cnns and spectrograms for audio processing?, 2018. URL `https://towardsdatascience.com/whats-wrong-with-spectrograms-and-cnns-for-audio-processing-311377d7ccd`.

[11] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition. *CoRR*, abs/1711.10282, 2017. URL `http://arxiv.org/abs/1711.10282`.

[12] Natalia A. Tomashenko, Yuri Y. Khokhlov, and Yannick Estève. Speaker adaptive training and mixup regularization for neural network acoustic models in automatic speech recognition. In *Interspeech*, 2018.

[13] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. URL `http://arxiv.org/abs/1710.09412`.