MSDS 458 Final Project

Audio Event Detection Model

Brian Tieskoetter

8/31/2019

Introduction

Neural network applications in image processing have received much attention and study lately, and their success is becoming well-known. Audio processing applications are probably a bit less intuitively accessible to most people so they have gotten less attention; but with the proliferation of microphones in voice-activated 'personal assistant' or other devices, audio applications may become as or more useful in the future. This project was a study of how neural networks interpret audio data, including how the audio signals are prepared and presented to the network and how the network interprets those signals. This will give me a foundation to design more complex and efficient applications later.

While the most immediate and intuitive interpretation of audio signals is probably a one-dimensional stream of data over time, audio signals can be interpreted more usefully in two or more dimensions, much like images or videos. The dimensions of an image are dimensions in space; with an audio signal they are in frequency and time. (Both audio signals and images can be interpreted in more dimensions, such as frequency (color) for images and with directional microphones, space for audio, but that is beyond the scope of this project.) So it is reasonable that the same network structures that are successful for visual processing alre also useful for audio processing. I used a 2-D Convolutional network structure to identify a single sound class, and then investigate the activations of the network to understand how the network is processing the signal. Based on my readings, I had originally proposed to also try a network with an LSTM layer, but that will have to go into the future work section because the data set I used is not granular enough to make it effective; but after my analysis I am very skeptical it would be effective.

I evaluated the network on several signals with varying success, but this report will focus on one: the sound of a bass drum. I am focusing on this for interpretability; it is simple and familiar enough that all readers will understand what the input is; its simplicity makes the network small and easier to picture; and it is not really as simple as it first appears because it almost always occurs as part of a very busy background signal.

Data Preparation and Representation

Processing audio signals in two dimensions requires breaking them out into those dimensions. This is done by separating the signals into frequency bands and calculating power vs. time in each band. This process is quite complex, but luckily there is a standard format available – MFCC (Mel-Frequency Cepstral Coefficients) -- used in this and most other audio models. Without getting deep into technical details, the audio signal is sampled at some sample rate, divided into frequency bands using an FFT algorithm, and a power measurement calculated. The data is a matrix of point measurements of audio power at each frequency and sample point. If the audio is sampled every second, then the data will be the power during a very short interval around that sample point, not averaged or accumulated over the entire second. When interpreting the data, it is also useful to remember that the frequency bands are evenly devided across the frequency band as humans perceive it, so the difference between two bands at low frequency sounds the same to us as the difference between two bands at high frequency. That, at least is intuitive – it may be the only thing about interpreting MFCC that is intuitive!

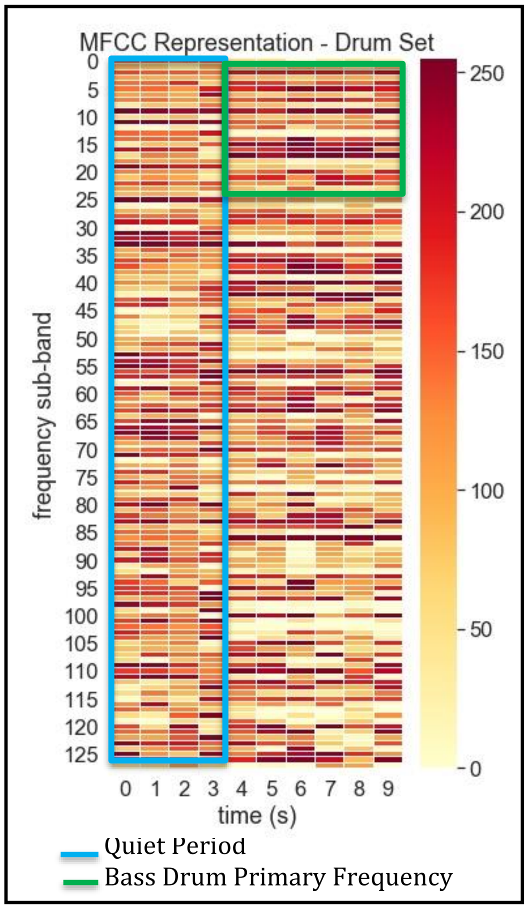
At the start of the project and imagining sound as a time-based stream of data points, I believed that controlling the sample rate and a short sampling period would be critical to my analysis. There are several software tools available to do the conversion to MFCC and raw audio sample data sets are available (both Google and Yahoo maintain sets). However, with the exception of hardware failures by far the greatest difficulty I encountered was in attempting to download and process the raw data. While I could generate an MFCC data set, I do not have a method developed to validate that it was processed correctly. I had assumed that a visual inspection for patterns I recognized audibly would be enough; but this is not the case.

Google and Yahoo also provide pre-processed audio files consisting of segments of audio tracks from Youtube or Instagram videos already sampled and converted to MFCC format. These are already thoroughly vetted, although the sampling rate cannot be changed. The data from the Google Audiset in particular is sampled at 1 Hz which I expected to be much too wide to be useful.

I downloaded pre-processed features from the Google Audioset to use as a reference point and did my initial modelling on them, and found to my surprise that I was able to model sounds much better than I expected. Given that it was working and vetted, and the low sample rate creates smaller data points that are more practical to visualize, I completed my analysis with these preformatted, 1Hz-sampled feature sets.

The feature sets consist of audio tracks from YouTube videos, sampled at 1-Hz and divided into 128 frequency sub-bands. Each band corresponds to about 15 Hz at low frequencies and about 150 Hz at high frequencies. Most segments are 10 seconds long, though some are shorter. The data, then is an 128 x 10 array of values. (Short segments are padded to 10s.) The data is already normalized, which is helpful for modelling but makes visualizing it more difficult.

To help visualize the raw data, the MFCC format is pictured to the right. Frequency bands/samples with more audio power are darker; though the data is normalized. The video this comes from can be viewed at: https://www.youtube.com/watch?v=Fwdl3YZeLRk The segment is from 1:20 to 1:30 in the video. The first four seconds of the segment are almost silent, as a drummer sits down at a drum set; the remainder is primarily bass drum and cowbell instruments. The quiet segment of the video is marked; it is not at all obvious that it is silent. The primary frequency range for a bass drum is also marked; there does appear to be a signal in this segment but not all such segments appear to have signal there. It is so clear here because there is little other backround noise – though you could never conclude that just from the MFCC representation.

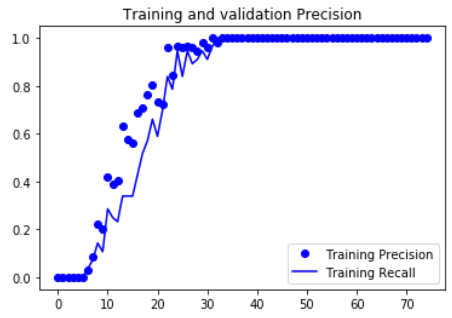


The feature sets are provided with separate training and evaluation datasets, so I made use of that convenience. Training was done with the 'balanced train' dataset and testing was done with the 'eval' dataset.

Model Structure

The model is based loosely on one of the two most successful model proposed in (Benito-Gorron, et al, 2019), which consists of seven 3x3 CNN layers separated by 2x2 pooling layers and followed by one 512-node densely connected layer. I was classifying one class instead of many, and I could control the task, so my objective was much simpler than theirs. I reported mid-project that I was achieving good performance with three 3x3 CNN layers of 128 channels, 64 channels, and 32 channels each separated by max pooling layers, and a fully connected layer of 64 nodes. Because the sampling rate was so low and there were only 10 samples on the time axis, I treated the samples on the time axis as independent of each other and I only pooled data on the long axis (each layer had 10 samples on the time axis). I always pooled by a factor of two on the frequency axis.

On examination of the activations of the hidden layers, many still appeared not to be activating so I reasoned that the network could be reduced further; I was able to reduce it by more than half. The results below are presented for a network with three 3x3 convolutional layers of 56, 28, and 14 nodes each followed by a max pooling layer, and one densely connected layer of 28 nodes. This network does occasionally fail to train at all, with 0% precision and 0% recall, but when it does train it achieves high precision and recall. (There seems to be no middle ground, as long as I train it long enough.) A graph of training precision and recall vs. training epoch is shown to the right. The network also achieved perfect precision and recall on the evaluation set, which is surprising but which shows that this is a quite simple classification task despite noise in the background of almost all samples.

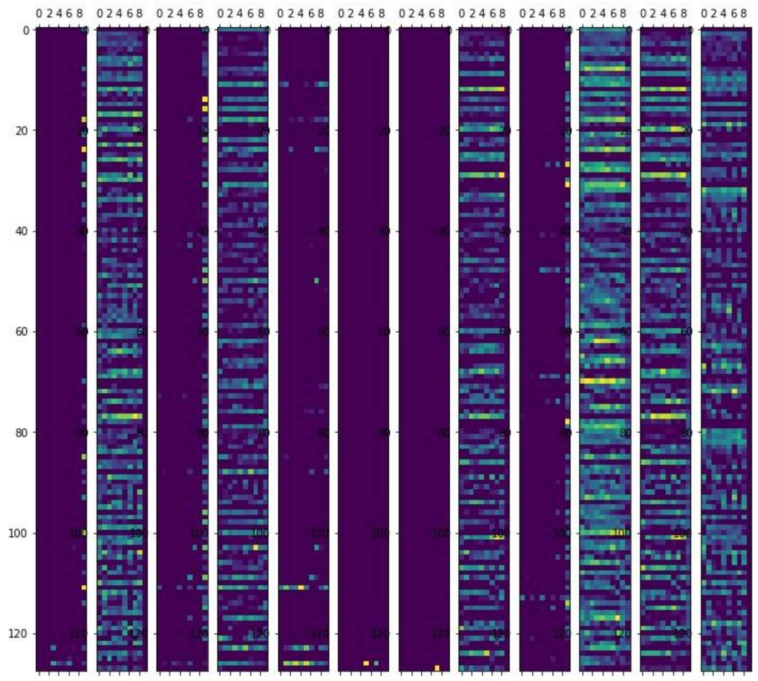


The data set I am using is very unbalanced, with less than 90 of ~20,000 audio files containing the true feature (for bass drum) in both the balanced set (used for training) and evaluation set (used for validation). Because of the sparseness, I am using precision and recall as metrics of model performance.

Note that the smallest network I describe (56-node first layer) is optimized for the bass drum feature only. I was able to get good training results with several feature classes with the larger network (128-node first layer) but only very simple features can be trained with the smallest. Reducing the smallest network in any dimension resulted in no training success at all (0% precision and 0% recall). Three convolutional layers appears to be both the minimum to classify any audio feature and sufficient for any feature that I was able to classify at all. (At least one, 'baby crying', was not classifyable in this data set even with a much larger network. I believe there is not enough resolution in this set to distinguish that feature; it requires measuring increases in sound level / frequency over time, so 1 Hz sampling rate is too low.)

Visualizing Hidden Layer Activations

Finally we get to the fun part – analyzing what features the model identified. Based on my knowledge of how a bass drum sounds, I would expect the model to have strong activations in the region of the drum's primary frequency range, from ~frequency sub-bands 5-30, encompassing roughly the frequency range 50-400 Hz. The model activations do not seem to be strong in this range at all, but it is be picking up on features that I would not associate with bass drum at all.

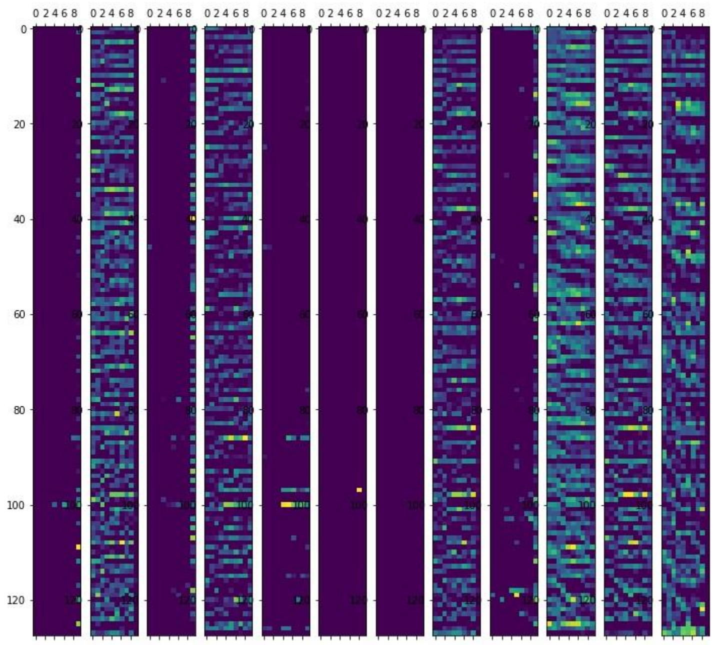


Successive layers in convolutional neural networks activate on more and more abstract features, as the pooling layers sum features into denser and sparser sets. The first and largest layers activate on patterns that are more recognizable to us, but the model is making its classification based on the activations in the last, smallest pooling layer. In an attempt to be brief, I will focus my discussion on selected channels in the first row, for two different audio clips; and compare the activations in the final row for the same clips to understand what the model is using to identify the presence of a bass drum. I will include the full activations for all three convolutional and three pooling layers for one audio clip in the appendix if the reader wishes to view it.

Selected first-layer activations from a segment with two drums (first 5 seconds is a rhythm on a wood block and the second 5 seconds is the same rhythm on a bass drum). You an clearly see some frequencies activated in a pattern, presumably some resonant frequencies, in some channels; others are activating on single frequencies in single time segments. The first, third, and ninth channels in the picture appear to be activating mainly on the last time segment, while the second and last channels pictured appear to be almost entirely ignoring that segment entirely. This, and the segments activating on a few points, suggest the network is overfitting the training data. I didn't see evidence of that in the evaluation data set; but possibly some dropout would help this network generalize more effectively (and be more parsimonious).

First Layer Activations, Two Drums Playing

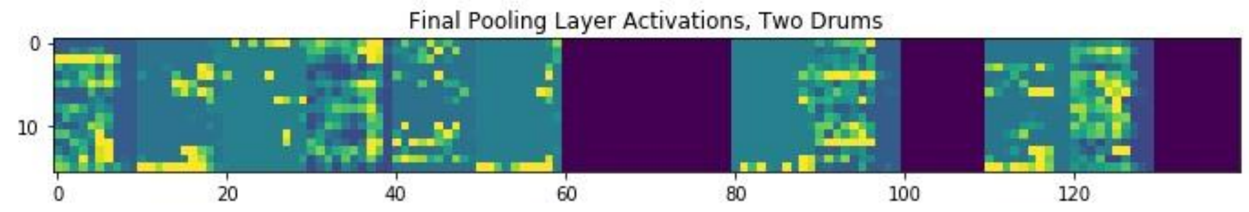
For comparison, the activations generated on a second audio segment are also pictured. This is from the same audio segment as pictured in the data preparation section; the segment is almost silent for four seconds, when a drum set starts playing. The resonances apparent in the previous file are much less clear here but otherwise the activations are similar.

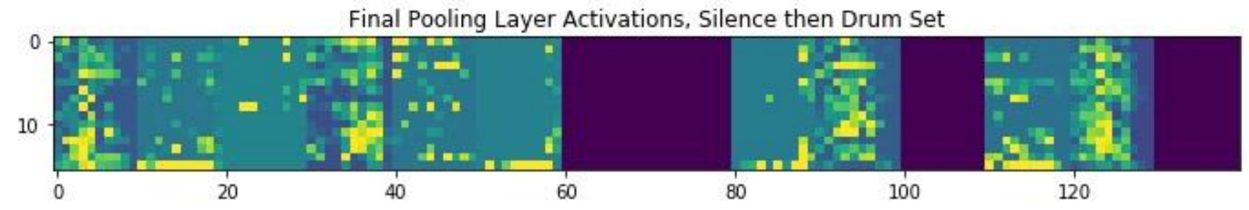


1First Layer Activations, Silence then Drum Set

The model will combine the maximum values from different channels to reduce the data to identify more abstract patterns. Those more abstract, pooled patterns are what the final classification is based on, so they should show most directly what characteristics the model is identifying.

The final max pooling layer activations generated for both files are shown together below. There are now 14 channels of 16 frequency bands, but still 10 time segments.





Looking for characteristics that are common between these two sequences, I do not see many in the frequency range of a bass drum. The first and 10th segment (90 to 100) do have strong activations in that range, but the fourth also has a null there. Instead, I see many activations at the highest frequency segment and several other blocks at high frequencies. Four of the 14 channels are not activated at all; but presumably they hold information that would be activated for other drumming combinations. I have to conclude that the model is looking or some signal in the drum frequency range, combined with other higher-frequency components that we cannot directly identify but which distinguish a bass drum from other signals that have the same low-frequency component.

Conclusions

My goal entering this project was to understand how a convolutional neural network identifies patterns in audio signals to better design neural networks for audio applications. The outcome of this project was simpler than I originally envisioned, but I did draw some useful conclusions that will guide future investigations and designs:

- My biggest surprise and greatest take-away is the relatively low importance the time element was in identifying different signals. I have typically thought of audio signals as a time-based signal in one dimension. I had assumed that without a high sample rate to detect changes in power and frequency over time, identifying sounds would be ineffective. However, although I focused my attention and this report on one signal class, I was able to classify a variety of signals using a simple network and data sampled in 1Hz segments. This is not to imply that the time element is not important; for example, I could not identify crying at all. But a major component of identification is in the frequency content alone.
- The complexity of the smallest effective network I could devise was orders of magnitude smaller than I started with and that the literature had led me to believe. In a world where smaller and more efficient networks will be more and more important, it is useful to have an idea what the lower limit of the network can be.
- The number of channels in my analysis that focused on one or two points in the data, or that appeared to focus on characteristics that appear to be tuning to the training set specifically (or example, channels that focused on one time segment) suggest that even this small model was overfitting. The precision and recall measured over the evaluation set were perfect, but possibly the model could be more robust and even smaller if I used dropout or trained on a larger training sample.

- Based on my understanding of the processing through the convolutional layers, I am skeptical that an LSTM layer can improve the performance of a convolutional network model much. The LSTM layer would be inserted after the last max pooling layer and would emphasize features in the layer that were consistent between time samples. But that time information is already included in the features detected by the convolutional layers; and most audio signals that have a time component are varying over time, not consistent.

Future Directions

With the threat of missing a deadline removed, I plan to revisit the audio processing routines and generate my own MFCC – coded files using sample rates I can vary. Many audio signals do have a time component that cannot be ignored (i.e. clock ticking) and I need to deepen my understanding of these networks with that component included.

Benito-Gorron, D. D., Lozano-Diez, A., Toledano, D. T., & Gonzalez-Rodriguez, J. (2019). Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset. *EURASIP Journal on Audio, Speech, and Music Processing, 2019*(1). doi:10.1186/s13636-019-0152-1 Retrieved from https://asmp-eurasipjournals.springeropen.com/articles/10.1186/s13636-019-0152-1

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., & Ritter, M. (1970, January 1). Audio Set: An ontology and human-labeled dataset for audio events. Retrieved from https://ai.google/research/pubs/pub45857

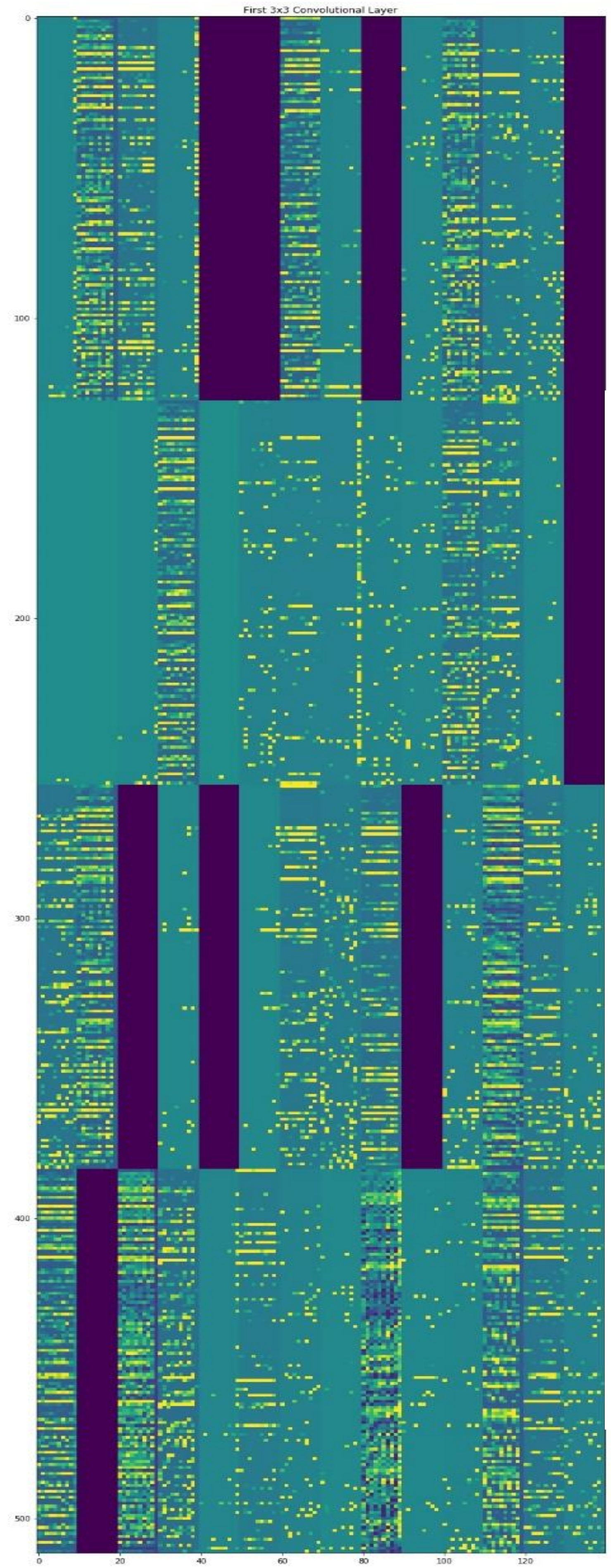
Appendix -- Visualization of All Layer Activations

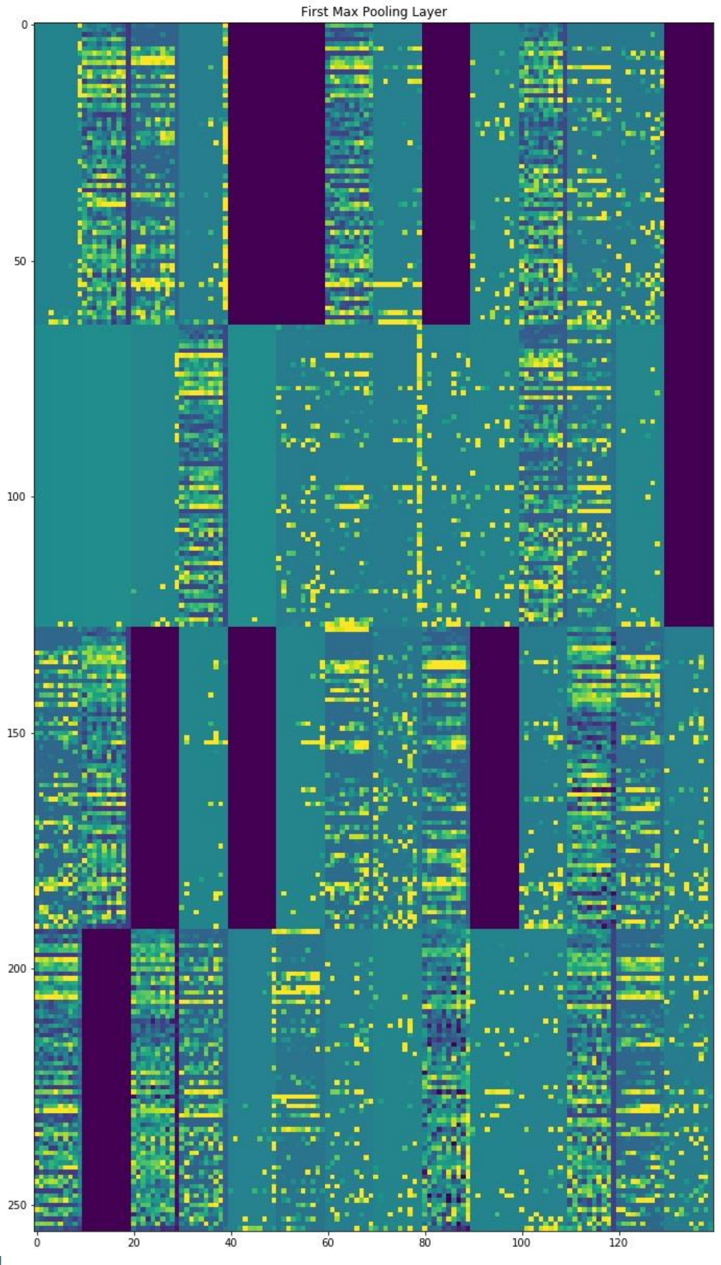
The activations of all convolutional layers generated by the model when a sigle audio segment is used as input is shown below. The audio segment used to generate this activation is taken from the audio track of a YouTube video, and the track consists of two types of drums repeating the same pattern. The first half of the track is one drum (sounds like some sort of wood block) and the second half is a bass drum. This is provided to show all steps of the model, as only the most interesting were presented above for brevity.

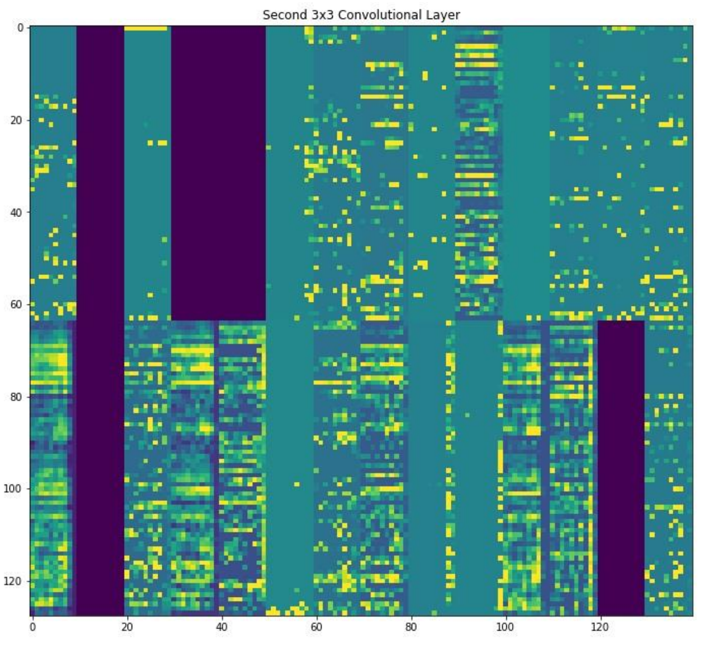
The video track can be found at https://www.youtube.com/watch?v=9tBOegUHrL0.

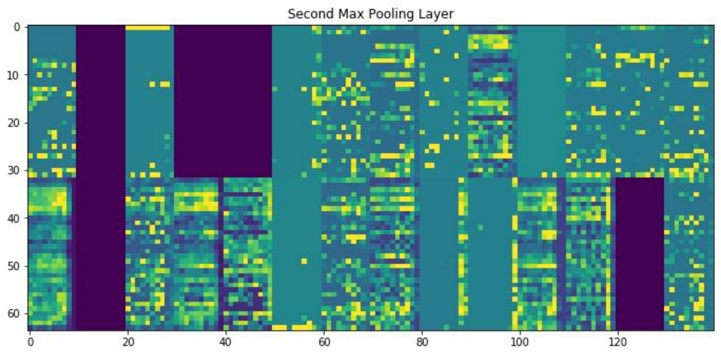
The audio track covers seconds 1:20 to 1:30 but the track is similar through the video.

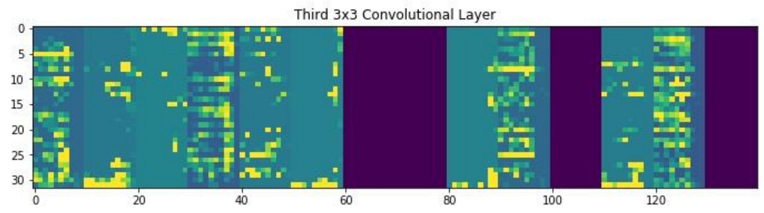
The layer activations are shown in order down the page, as I found it impossible to show them in any sort of meaningful way without taking up a lot of space.

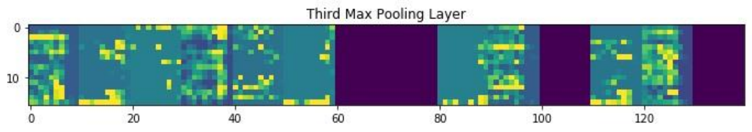

First 3x3 Convolutional Layer
0
100
200
300
400
500
0
20
40
60
80
100
120


First Max Pooling Layer
0
50
100
150
200
250
0
20
40
60
80
100
120


Second 3x3 Convolutional Layer
0
20
40
60
80
100
120


Second Max Pooling Layer
0
10
20
30
40
50
60
80
100
120


Third 3x3 Convolutional Layer
0
5
10
15
20
25
30
40
60
80
100
120


Third Max Pooling Layer
0
10
20
40
60
80
100
120