Audio-Capturing Music Notation System

Digital Signal Processing & Machine Learning Combination for Audio Detection

By Joseph Meng, ECE '19

Introduction

Our previous senior design project was creating a car security system that used sound detection, digital signal processing, and machine learning to detect theft. In response to some sudden changes to our team, we have since dramatically changed our project to something similar: using signal processing to extract musical notes. However, in my tech notes, I still want to explore not just the application of digital signal processing but the combination of digital signal processing and machine learning for audio recognition.

DSP Module for Audio Transformations

The first block within audio recognition is the digital signal processing (DSP) block. In this block, microphones would feed in audio data in real time. This continuous stream of data would then be sampled every so and so millisecond. What is sampling and why is it used? In real life, audio is in the form of waves in the air. However, when we store the wave's information into the computer, there is no way to store all of its information because real time is continuous. Computers are digital and what that means is we can only store information discretely (one by one). To solve this problem, we use sampling, which means we take the information of the wave every x seconds. Of course, the faster we sample, the more information we'll take in. So now that we have the information of the audio wave, we can send the data directly to the machine learning (ML) block and have it finish the job for us, right? That could be a solution, BUT, sound file features do not characterize the source of audio very well. For example, if we look at an audio file of someone screaming and another audio file of a dog barking, we are not going to see a big difference.

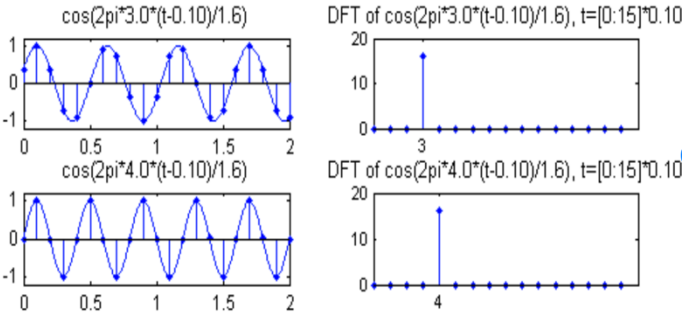

Figure 1. These two wave signals look almost indistinguishable until we look at their respective frequency components on the right: 3 and 4.

What is easier to do is if instead of looking at these audio signals as a whole, we break them down into their frequency components. What

does that mean? There is a principle in signal processing theory that every signal can be represented as a sum of waves with different frequencies. By using what's called the Fourier Transform, we can transform a signal into its sum of waves, look at how impactful each wave is, and pick out what frequencies these waves hold. These frequencies would be the components that define the signal. In short, analyzing the signal after the transformation can give us a quicker impression of what its characteristics are. If we perform the transformation on our data before passing them to the ML block, the ML block will thank us greatly.

Audio Transformation Design Parameters (DSP Theories)

So, we have a plan now. After every "x" sample of data, we will treat those samples as one signal, transform it, and then store it. After doing this process repeatedly, we will have a full picture of what we're hearing from the real world, specifically, what frequency components exist within which interval of time. But that "x" really sets a lot of things. For example, if we increase "x", that means we are transforming more samples. This means we are transforming with more information, and therefore, our analysis of the frequency components will be more accurate.

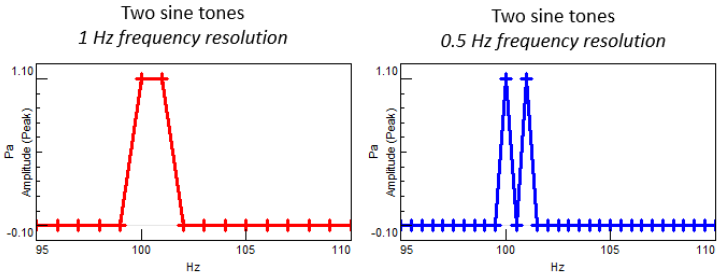

Figure 2. Higher frequency resolution leads to better distinguishing of different frequency components.

But the downside? If we want more information before each transformation, that means we're transforming less frequently. So, we may be outputting very accurate frequencies, but we are not as accurately representing the timing of when those frequencies exist. Therefore, when adjusting "x", there is a trade-off between how accurately we are analyzing frequencies and how accurately we are representing time.

Training Data

Now that the DSP block is done with its job, we will pass the big picture it has generated to the machine learning (ML) block for performing recognition. A machine learning classifier takes in inputs (in this case, the transformed audio) and outputs a classification (in the case of our previous project, a classification of "threating" or "non-threatening"). For this classifier to accurately map input to output, it has to be trained. During training, a set of data (usually very large) will teach the classifier what it is in the inputs causes a specific output.

Specifications of the data used to train the machine learning model are very important. The first specification is the data size. If we use a larger data to train the model, our classifier is going to be way more accurate in differentiating sound later on. However, a larger data also means slower training. The second specification is the data organization. One of the biggest audio databases on the internet right now is the AudioSet provided by google. Google created AudioSet by compiling different segments of youtube videos that have been tagged by users. For example, there may be a video on youtube that features dogs. A segment of the video is a dog barking and users tag this segment of the video as "dog barking". Google therefore takes all this information and creates a data entry by inputting the video's link, the time frame of the video where the dog barks, and the label of that segment: "dog barking". On top of that, AudioSet generates an excel file with about 2 million entries, each with a unique segment from a youtube video and a corresponding label (or even multiple labels). By training a classifier with that series of entry, the classifier would be a master of audio recognition by the end of the training.

Machine Learning Training Model

The final consideration in this ML block is exactly how the classifier will "learn" from a given data. The training model is the final step to completing our detection system. With the data in hand, how do we train the machine learning model? Training can be classified as supervised training versus unsupervised training. Supervised training means that our data has both inputs and outputs and we want to train a model to find a connection between the two. Unsupervised training means that our data only feeds in inputs and we are not outputting anything, and instead, trying to find groupings between the data. In simple terms, unsupervised training just means we are looking at patterns within the different inputs and how they relate to each other. For example, if we define the inputs as products that customers buy, the unsupervised learning would show purchasing trends (etc. customers that buy X would likely buy Y). In audio detection, we definitely want supervised learning. The reason is that our training data has both inputs (features of the audio signal) and outputs (classifications they belong in). Eventually, we are creating a model that can classify audio with classes.

In conclusion, both Digital Signal Processing and Machine Learning are both very crucial in creating a system that recognizes sound. The DSP block transforms input audio data into features (in terms of frequencies) by creating Fourier Transforms at each time segment of the sampled audio. The features then go through the machine learning classifier to be mapped to an output. In terms of training the classifier, the data is very important and with databases like AudioSet, we can organize the labels we want into larger classes. Finally, we can pick different training methods to create a trained model.

References

[1] Blade, Lightning. "Artificial Neural Networks Explained." Good Audience. July 24, 2018. Accessed December 14, 2018. https://blog.goodaudience.com/artificial-neural-networks-explained-436fcf36e75.

[2] Brownlee, Jason. "Supervised and Unsupervised Machine Learning Algorithms." Machine Learning Mastery. March 16, 2016. Accessed December 14, 2018. https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.

[3] DataArt. "Sound Classification with TensorFlow." Iot for All. November 10, 2017. Accessed December 14, 2018. https://www.iotforall.com/tensorflow-sound-classification-machine-learning-applications/.

[4] DSP Related. "FINITE IMPULSE RESPONSE DIGITAL FILTERS." DSP Related. Accessed December 14, 2018. https://www.dsprelated.com/freebooks/filters/Finite_Impulse_Response_Digital.html.

[5] National Instruments. "IIR Filters and FIR Filters." National Instruments. Accessed December 14, 2018. http://zone.ni.com/reference/en-XX/help/370858N-01/genmaths/genmaths/calc_filterfir_iir/.

[6] PJS. "Introduction to Filters: FIR versus IIR." Siemens Community. August 29, 2018. Accessed December 14, 2018. https://community.plm.automation.siemens.com/t5/Testing-Knowledge-Base/Introduction-to-Filters-FIR-versus-IIR/ta-p/520959.

[7] Virtanen, Tuomas, Mark D. Plumbley, and Dan Ellis. *Computational Analysis of Sound Scenes and Events*. Springer International Publishing. Chapter "file:///Users/jm/Downloads/9783319634494-c2.pdf"