

**Progress Report**

**Fall 2018-2019**

**Team Number ECE-30**

**Instrument Recognition System**

**Team Members**

|  |  |  |
| --- | --- | --- |
| **Name** | **Department** | **Email** |
| Gabriel Collins | ECE | gjc52@drexel.edu |
| Anna Mookerjee | ECE | anm66@drexel.edu |
| John Dwyer | ECE | jvd37@drexel.edu |
|  |  |  |

**Team Advisor(s)**

|  |  |  |
| --- | --- | --- |
| **Name** | **Department/Company** | **Email** |
| Dr. Youngmoo Kim | ECE/ExCITe Center | [ykim@drexel.edu](mailto:ykim@drexel.edu) |
| Andrew Wiggins | ECE/ExCITe Center | awiggins@drexel.edu |
|  |  |  |

**Group Leader signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Advisor Signature (or attach e-approval) : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Format**

* **Update 1). Team number, 2). Project title, 3). Team members, 4). Team advisor(s) on the title page.**
* **Max of 15 pages (references and appendices do not count towards this limit)**
* **Single spaced, Times New Roman (Body - 12, Headings and sub-headings - 14), Justified.**
* **Start each section on a new page**
* **Include page numbers at bottom right of the page.**
* **Delete all instruction areas in red boxes.**
* **Submit as upload to BBLearn Class site**
* **Do not change the section numberings.**

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

1. **Abstract**

The music streaming business has been booming since the dawn of the 2010s. Whether it be companies like Spotify, iHeartRadio, Pandora, Apple Music, or Google Play Music; everyone has their own service that they use on a daily basis. But, there’s one big problem with all of these different services: their recommendation algorithms. We’ve all been there at one point or another, you are listening to a “curated” playlist, but it ends up just being music you’ve already heard. The reason for this is that a lot of these companies create their recommendation algorithms based off of top selling artists, genres you’ve previously listened to, or playlists you’ve created. As a result, you’re constantly running into a feedback loop/echo chamber of music that you’ve already heard.

The ideal recommendation system wouldn’t focus on the music you’ve already heard, it would analyze the music you’re currently enjoying and find songs with similar components. Through Machine Listening and Audio Event Detection, we plan to introduce another layer into the algorithm. This will allow listeners to discover new music through the instruments within the music they currently enjoy.

**Table of Contents**

1. Problem Discussion
2. Proposed Work and Deliverables 5

3.1 Methods and Solutions5

3.1.1 Machine Learning Models5

3.1.2 Feature Extraction5

3.1.3 Data Set

3.2 Audio Permutation

3.3 Prior Work

3.4 State of the Art

3.5 Progress Workflow:

1. Completed Work
2. Work Schedule / Proposed Timeline
3. Industrial Budget
4. Out-of-Pocket Budget
5. Societal, Environmental or Ethical Impacts
6. Summary/Conclusions

**List of Figures**

**2. Problem Description**

The music streaming business has been booming since the dawn of the 2010s. Whether it be companies like Spotify, iHeartRadio, Pandora, Apple Music, or Google Play Music; everyone has their own service that they use on a daily basis. But, there’s one big problem with all of these different services: their recommendation algorithms. We’ve all been there at one point or another, you are listening to a “curated” playlist, but it ends up just being music you’ve already heard. The reason for this is that a lot of these companies create their recommendation algorithms based off of top selling artists, genres you’ve previously listened to, or playlists you’ve created. As a result, you’re constantly running into a feedback loop/echo chamber of music that you’ve already heard.

For many casual listeners, there’s an interest and a need in a more feasible recommendation system that doesn’t just give them the same songs over and over again. The need for this solution, we believe, is to add another set of features to the recommendation algorithms created by these companies. The ideal recommendation system wouldn’t just focus on the music you’ve already heard, it would also analyze the music you’re currently enjoying and find songs with similar components. Through Audio Event Detection made possible by Machine Learning, we plan to create these missing features in these algorithms. Our project’s end goal is to find a means in which we can analyze musical recordings for specific instruments, and create a new piece of metadata for that. Implementing this will allow listeners to hear new music with similar instrument structures to the music they currently enjoy, and also give upcoming musicians a better chance to shine and show their talents. This need for an additional piece of the algorithm gives us the perfect opportunity to pounce and create the needed solution.

**3. Proposed Work and Deliverables:**

**3.1 Methods and Solutions**

When determining how to solve an issue like this, we first have to consider what type of problem this is. When we forget about the end goal of introducing our system into the algorithms of companies of Spotify, Pandora, and streaming systems as a whole, our goal is to accurately identify if an instrument is present within a recording. We must also consider that in music multiple instruments will be playing at the same time and that they can overlap each other in time. This problem then becomes a subset of a known problem with known solutions, with that problem being known as Audio Event Detection (AED). These solutions require three main components that will be explored throughout this section; audio feature extraction, machine learning models, and an extremely large dataset. A diagram of the training and testing of this model can be found at the end of this section.

**3.1.1 Machine Learning Models**

When first approaching this problem one of the main things to consider is what type of model should be used to detect instruments. To do so you must first know the inputs and outputs of a model. All machine learning models considered in this project consist of two input sets: features and predictors. These features and predictors have a relationship where all of the features correspond to one predictor of a much smaller set, therefore, each predictor corresponds to a much larger number of features. The predictors are what you want your output of a finished model to be. The model will initially be trained on a large data set that is split into train and test set with some more advanced models adding a third partition called the generalization set. The validation of the models training is done by comparing the output of the model on the test features with their predictors. If all of the outputs match the predictors then the model has a validation rating of 1. This doesn’t mean that the model is perfect however, it just indicates that within the testing set it was given, it was able to correctly predict everything that it read. This is why extremely large data sets are important to train accurate models.

The models that were explored in this project consisted of models provided by Scikit Learn, an open source python library that is used to teach machine learning fundamentals. We have also used Tensorflow/Karas models as well, which are much more accurate machine learning models that focus on the customization of Neural Networks (NNs) which will be discussed in a later section

**3.1.2 Feature Extraction**

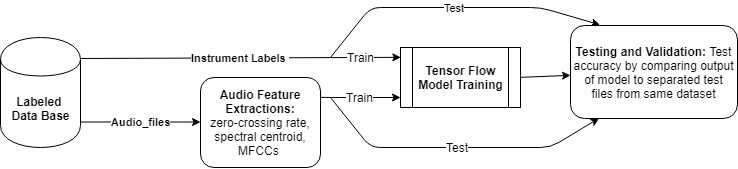
In machine learning models, the process of training a model can vary, but in most cases, the general idea is to find statistical information that can be used to predict the correct value. In the case of NNs, the goal is to recreate how neurons in the human brain function, which simply put is varying electrical pulses from neuron to neuron. Since the goal is to recreate how the brain processes data, that means that the information it receives should be something similar to what we perceive. This is where Feature extraction comes into play. We perceive audio in pitch, time, and timbre which are not well represented in a 1D time array of power. Instead, we must find some other way to represent these important features in a data structure of some kind. There are many different types of features that can be used ranging from zero crossing rates to spectrograms, to center frequencies, and to MFCCs (Mel Frequency Cepstral Coefficients)  and each of them have their advantages and disadvantages. We decided to conduct our own experiments to determine what would work best for us.

**3.1.3 Data Set**

A model is only as good as the data that it is trained on and only correctly identify information that is similar to the data within its data set. This means that the first obstacle put in the way of development is to determine a useful data set that reflects what we want. This means thinking about the project on a level of input and output. Our goal is to identify music within an audio recording. In these recordings, multiple instruments will usually be playing at once, and some of these instruments will be electronic, acoustic, or a mixture of the two.

In recent years several audio data sets have been released with the goal of trying to solve Audio Event Recognition as a whole with the main data sets being used in the field being Google Audio Set and YouTube 8M. These datasets contain thousands of different predictors, ranging from a baby crying to a car horn, to an orchestra playing. While these data sets are usually the first used in trying to solve the problem of audio event detection as a whole their cases for musical instruments are too broad and are sometimes inaccurate. Their predictors come from tags on youtube videos and are not validated by a human listener making them problematic to use in a real musical environment.

With this in mind, one unique data set remains known as NSynth, another open source data set provided by Google to create combinations of sounds from original audio recordings. What is unique about this data set is that contains 305,979 musical notes, with 1,006 different instruments, with each recording containing a unique pitch, timbre, and envelope. Each sample of audio has 3 seconds of attack followed by one second of fall off. Each of the instruments falls into 10 different Instrument Families, giving a general hierarchy. This dataset, with all of its meta data on each sample, will give us perfect control over how we train these models as far as I\O is concerned.



**3.2** **Audio Permutation**

In order to correctly identify instruments within a song, we must be able to correctly replicate what traditional recordings might sound like. This means that the recordings must have one, multiple, and no instruments in it, as to reflect the universal set of audio. This means that the machine learning problem is a Multiple Instance Learning Problem, requiring constraints to make it possible. It is impossible and impractical to analyze a whole song at once, but instead, it would be easier to pick a certain amount of time and determine what instruments are present within that amount of time. The NSynth data set as it does not give this type of data, however, we can synthetically create data by combining audio files and modifying predictors to reflect the mixture. Doing this with all 300 thousand plus files would be computationally impractical, but we can leverage our knowledge of music theory and music productions to put common instrument types together and common musical note together, avoiding tones and chords that are often never heard in music. This will significantly limit the data that needs to be processed and trained. The numbers for this have yet to be computed on our end, but this is the current solution for this problem. If this processing is still computationally impractical the data sets produced will be limited by some factor until processing becomes feasible.

**3.3 Prior Work**

As the problem of AER has only been recently furthered in previous years, the main goals of the community as a whole have been to classify broader information, with the niche of instrument identification falling to the wayside. As such there have been no major reports of successful instrument identification within the scope of this project, however, several other attempts at AER have been successfully made with promising results.

Dr. Qiuqiang Kong of the University of Surrey, Uk showed in his paper called *Audio Set Classification with Attention Model: A Probabilistic Perspective (2018)* that it is more than possible to train MIL models of different types of sounds within a certain amount of time, using an attention model combined with a deep CNN (Convolutional Neural Network). Kong trained his data set on Google Audio Set and outperformed Google’s baseline for the AER that was published when Google made Audio Set public.

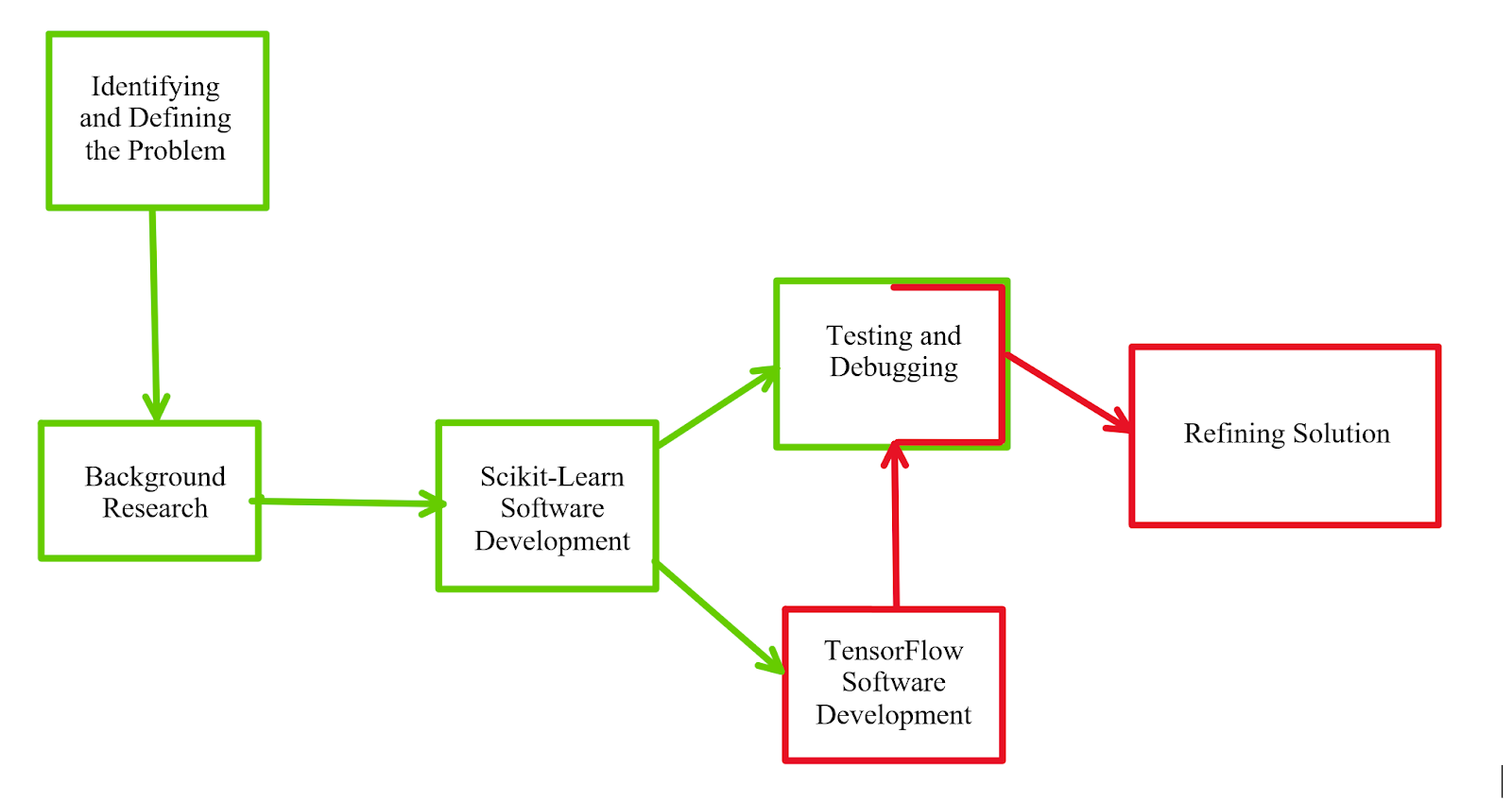
Another example of success in this field is Shawn Hershey of Google, who in his publication of *Cnn Architectures for Large-Scale Audio Classification (2017)* tested multiple types of CNNs and successfully identified instruments. The main problem with the Hersy paper is that it uses video as well as audio to help identify audio events, which does not carry over into the scope of the project. A majority of the input features are indeed audio, so his findings are not useless. Another issue is that the data set used was YouTube-100M, which would later be refined into Google Audio Set, but at the time consisted of more inaccurate labels.

**3.4 State of the Art**

As discussed previously the field of Instrument recognition within AED is limited meaning that there is no current state of the art for this technology. While this means that there isn’t a current solution for the problem this also allows us to look at this problem with a unique perspective, allowing us to be flexible to find what does and doesn’t work. It also gives us little to no benchmark to compare to, however, the training of the models themselves will allow for some sort of benchmarking.

The lack of any state of the art identification systems as well as any formal challenge issued by streaming services or Google to identify musical instruments shows that this problem has either yet to be formally recognized by the industry or another data set is being put together but has yet to be released to the public. Given the current advancements in this field we can expect a challenge of some sort to be released within the next two to three years.

**3.5 Progress Workflow:**

****

**4. Completed Work**

Before the submission of our Interim Progress Report, the majority of our work was focused around experimenting with Python packages to make basic music extraction programs so that we could identify any specifically useful functions that could be beneficial in our final algorithm, come the time we have to create our own. This work was mostly conducted through SciKit Learn, which works off of mostly premade functions and allowed for us to focus more on developing a working dataset, along with building our knowledge base for these functions instead of working from scratch on functions that may not work.

Whilst this was going on, our group was also working to decide on a dataset that fit our needs as something that was usable in these early stages, along with being large enough that it can be used in future terms so that our accuracy calculations will not be slanted due to differing in size and scope data. The three that were looked at most intently were that of Google AudioSet, Youtube-8M, and NSynth.

Before starting the project, most of us had experience with Google AudioSet as it worked mostly off of manually annotated audio events from Youtube. While this would work in some projects, for us it wasn’t ideal as the dataset just wasn’t that large, and overall it wasn’t very descriptive in comparison to some of the others we were looking into. As a result, we decided to move to Youtube-8M.

While 8M is also made by Google, it was created on a larger scale and stretched a much larger margin. Coming directly from Youtube’s database, this seemed like the perfect dataset for our purposes. But again, we ran into a major issue. The problem with Youtube-8M is that it's not very specific in its differentiation. Being a project focused on instrument identification, we needed to be able to have a dataset that focused more heavily on specific instruments within different families and sources. This is where we came to the NSynth dataset.

With over 300,000 audio notes from just over 1000 different instruments, the NSynth dataset made the most sense for our project. With it also being a highly annotated set, we were able to specifically pick out notes that we wanted to focus on in our first true testing. Along with having a large base of notes, these notes also stretch different pitches and velocities, allowing us to have a variety of different test cases available to us in the event we need them.

The first step in actually designing the system was to create a feature extraction function. This was achieved using functions from the music and audio analysis python package “LibROSA” which is useful for a lot of music information retrieval programs and was utilized greatly in our first designs. The LibROSA package contains many common audio feature separation algorithms like zero-crossing rate, spectral centroid, and MFCCs which were all helpful for our project. A feature extraction function was created using these algorithms. It was written to accept a loaded audio file and its sample rate as inputs, run the audio data through five different types of extraction algorithms and returned lists of the separated features. To test the efficiency of the extraction program and start comparing the usefulness of certain types of features, two-second audio samples of snare and kick drums were run through the feature extractor program. The returned features were then analyzed to make sure they indicated a separation between the classes of instruments.Figure ## below is a graph of the average zero crossing rate features extracted from the first samples and the separation between kick and snare features is very prominent. It was important to check this since these features were going to be used to train the basic models, if there was no separation then there was no hope for the classifier to see a difference between the instruments.

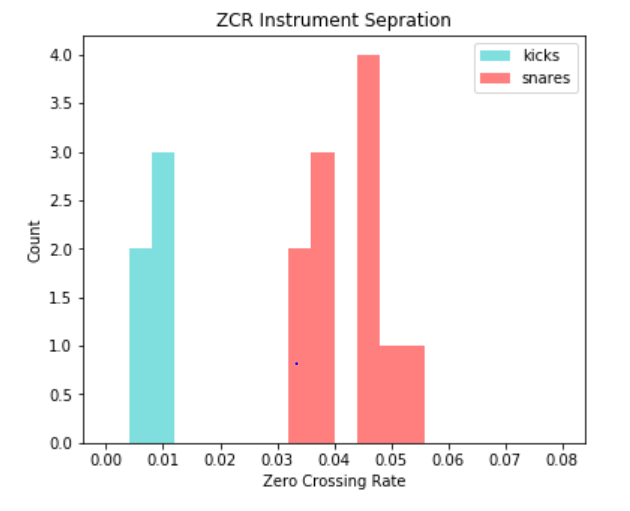
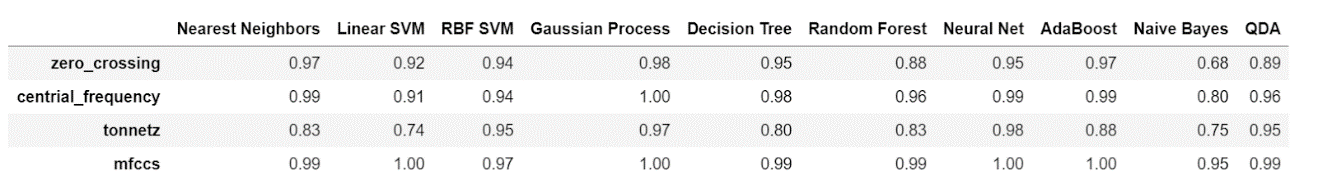


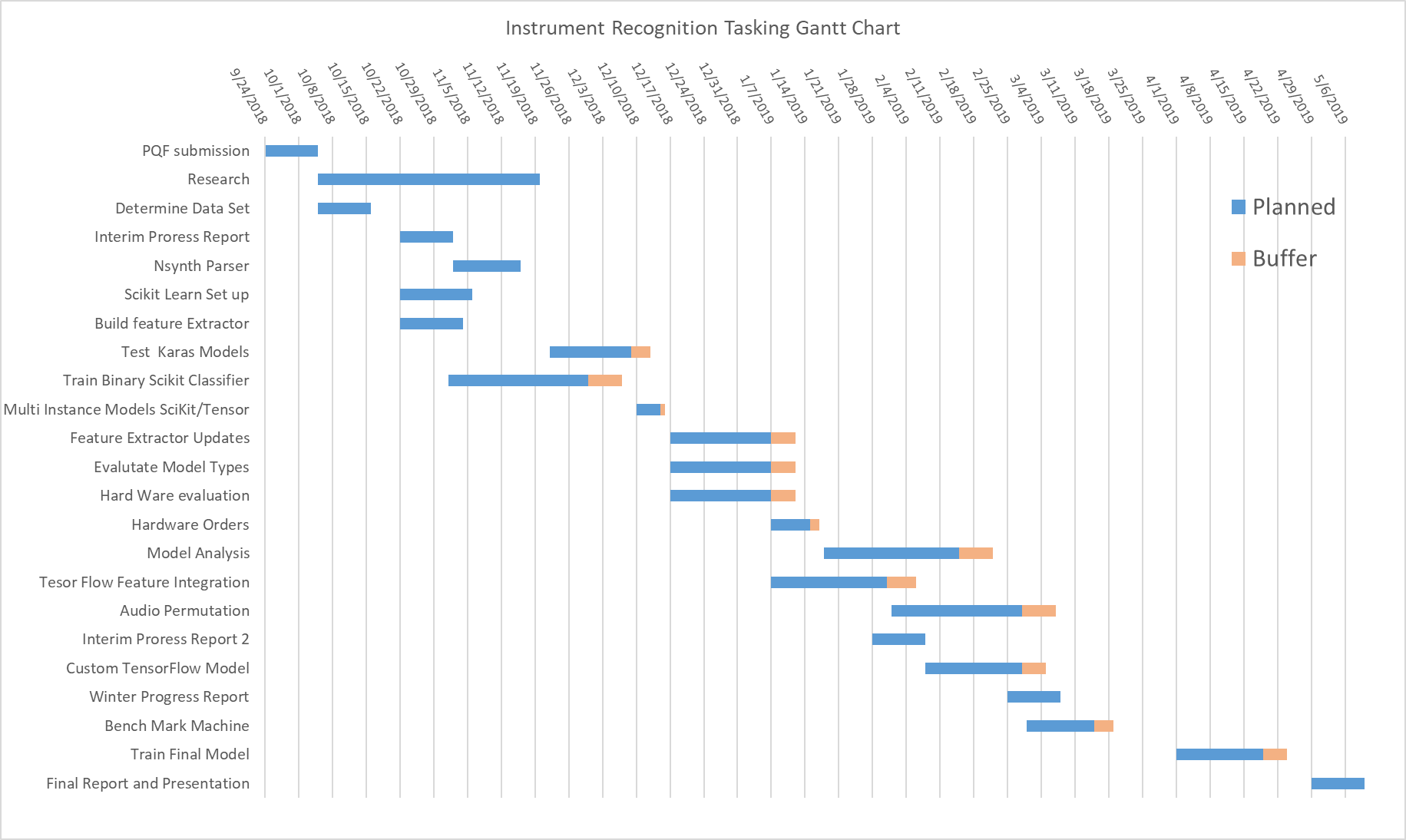
Figure ##: Zero Crossing Rate Feature comparison

It was then time to start working with SciKit-Learn. To familiarize ourselves with this tool, example code from a classifier project found on SciKit-Learn’s website was used as a starting point. This example used randomly generated datasets to train a collection of SciKit-Learn’s built-in classification models and then test the model with other similar data displaying the accuracy results in graphs. This was a great example to help us learn how the classifiers worked and what parameters they needed and the format our training data needed to be in in order for the SciKit-Learn functions to work. Once we broke down the example and understood how it all worked, we started writing a simple script for a binary classification system.

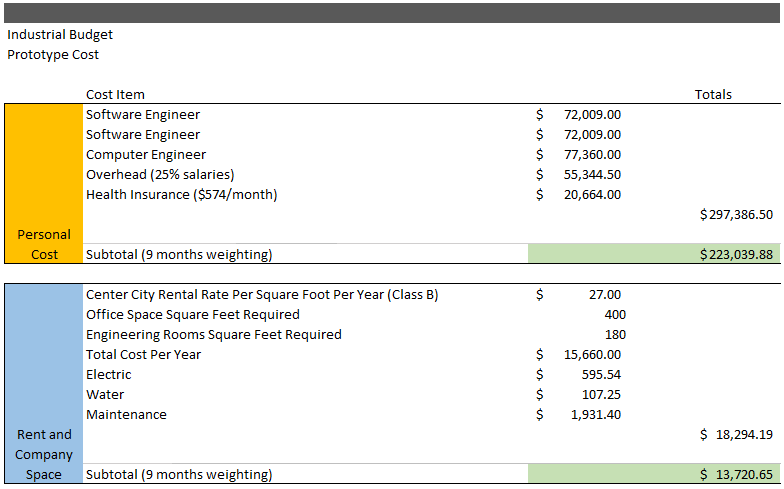
The training data for the classification models was the features from the feature extraction program. Each data point needed to have a binary label indicating the type of instrument added in order for the models to learn which instrument each feature corresponded to, since we were only looking at doing binary classification at this point, one of the instruments was assigned the binary tag  “1” and the other “0”. The classification program took the data and trained multiple different classifier models on about 60% of the data provided and then shuffled the remaining 40% and used that for testing accuracy validation. To begin with, a small set of about 16 snare drum samples and 16 kick drum samples were run through the feature extractor then classifier and then the sample sizes were increased to about 150 samples each and most recently 1000 samples.

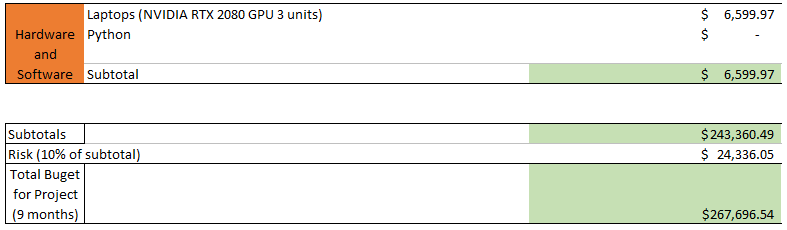
The 1000 samples were taken from the NSynth dataset and we decided to use vocals and bass as the instrument types. Training this size of samples started to emphasize the negative side of using SciKit-Learn which is primarily computation speed. Loading and training this amount of data took about an hour and a half, but 2000 total samples is nowhere near the amount of data that we eventually will need to be running. Figure ## below is a table generated at the end of the classification program where the models tested are listed in columns while the types of features extracted are shown in the rows.   
  
  
Figure ##. Accuracy Comparison of Classification type vs Extraction Features from the binary classification of 1000 Samples

The values represent the accuracy, a “1.00” would mean that the model trained using that feature and classification model guessed the correct instrument for every test sample. Any value less than “1.00” had some errors in identification. This was a helpful way of visually comparing the different types of features and classifiers and overall determined that MFCC features were the most accurate across the classification models while Gaussian Processing and basic a Neural Network proved to be the overall most accurate models. Using these results the next steps were to try and expand on these working combinations of features and models, but in a more customizable and powerful machine learning environment like TensorFlow.

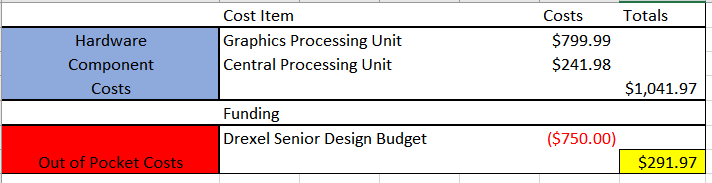
**5. Work Schedule / Proposed Timeline**

**6. Industrial Budget**





**7. Out-of-Pocket Budget**

****

**8. Societal, Environmental or Ethical Impacts**

Being a project most closely focused on helping the development of a new piece of technology, the only true impacts we will face are those of society. From our work, the most likely impact that we will face is that this new method of analyzing audio will become more common in many music services, and it will help to give people better recommendations on their music, or even find completely new music that they can listen.

In another vein of the same area, the possibility for this new area in the analysis of what people may enjoy could potentially impact future research and development in the audio and music fields. The doors that could be opened up by analyzing the qualities and instruments within the music could lead to the potential for entirely new ways to bring music to a listener, perhaps as far as never requiring a user to suggest their own interests.

Finally, the last and perhaps most important area of societal impact that our work will have is on that of undiscovered and young artists who have not had a break in their careers. Our product could help bring their music to more audiences as the way their music sounds and is created could be similar to that of a big market band. This, in turn, could lead to more success for not only larger bands who want to reach bigger audiences, but the smallest of creators just looking to find their place in the pond.

**9. Summary/Conclusions**

When looking back on the work we’ve done this term, we have made great strides toward our ultimate goal. Right out of the gate, we spent alot of our time researching just how we wanted to go about doing our project. The idea behind the combined research was a shortened amount of time trying to gather all of the possibilities for how we wanted to go about the development of our system. Finally, after making determinations on the Python Package we wanted to use (3.6), and where we wanted to focus the majority of our early work (SciKit Learn); we started our earliest stages of creation.

Following the decisions of how we wanted to develop, immediately our group went to work on developing both a feature extractor and audio classifier for development. For the data we wanted to use for our earliest development, the decision was made to use NSynth as it had the most versatile information and the capability for us to use it moving forward without having to unpack a new set of information and potentially change our work. The first version’s of our classifier and feature extractor came through SciKit Learn with the assistance of LibROSA. Working up from the original dataset given by SciKit, the models were trained on new information. Finally, after 5 weeks of development, we’ve completed our first set of feature extractors and classified over 1000 samples of information.

Moving forward, we plan to open up our testing to more samples and different types of information so we can truly put our classification methods to the test. Also, we want to create a more complex set of information to be inputted into all of these systems, allowing us to really see just how accurate our classifier is. With the Winter Term fast approaching, we expect for the majority of our work in SciKit to conclude by the midpoint of that term, and we will finally be ready to move into the final form of our model within the Tensorflow toolset.

**10. References**

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Please adhere to the IEEE citation style. This does not count against the 15 page limit.**

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Appendix A: Design Constraints Summary**

Team Number: ECE-##

Project Title:

Summary of the Design Aspects:

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**One to two paragraphs summarizing all design aspects of the project. This includes hardware, software, testing protocols, lesson plans, etc.**

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Design Constraints:

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Discuss how each design constraint was addressed in your project.**

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Economic:

Manufacturability:

Sustainability:

Environmental:

Ethical, health, and safety:

Social:

Political:

Standards and Regulations

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Cite list of Standards/Regulations that were used or evaluated for the project (use**

**IEEE Reference-style). Make sure you understand why a certain standard is to be met by your project, you will be questioned regarding this during your presentation. Don't simply mention random standards and regulations without studying their uses and requirements.**

**\*\*\*\*\*\*\*\*\*DELETE THIS SECTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Appendix B: Resumes**





