Audio Classification in Music Production

Aaron Balke
Western Governors University

C964 Computer Science Capstone

June 2023

Table of Contents

A1. Letter of Transmittal

June 8th, 2023

Mr. John Smith

Chief Executive Officer

H3 Music Corp.

42 Wallaby Way,

Sydney, Australia

Dear Mr. Smith.

I'm excited to provide the included project proposal for my Audio Classification Application. I strongly believe, and I hope you will agree, this application will remove a massive resource management burden off of our music production staff, so they can spend as much time as possible producing and providing services to our customers.

Our staff spends a great deal of time cleaning up, standardizing, and organizing resources used in music production. Through the use of Artificial Intelligence and Machine Learning, we should be able to remove this burden, allowing us to be more efficient and effective in our endeavors.

The proposed application will be built with scalability, flexibility, and resourcefulness in mind. I believe building it in-house will provide the best perspective to our needs, and the most financially viable option, only costing an estimated $2,750.00. The initial version will be used to organize our current audio resources, but the application will be flexible enough to allow for future resources to be processed upon reception, keeping things managed properly with minor oversight. I have recently completed my Bachelor's in Computer Science, and have spent almost a decade working in the music production industry. A great combination for building this solution.

Thank you for your time, and please reach out for further discussion.

Sincerely,

[illegible]

Aaron Balke

A2. Project Proposal

Problem Summary

Over the last couple of years, H3 Music Corp has continued to increase its output of music production services to scopes that were unexpected when this company was started as a side project in 2014. With this increase in services also comes an increase in digital resources used in the music production process. As the amount of these resources grow, we have trouble maintaining standardized names and metadata, especially since resources come from different entities.

With this problem, H3 Music requires a solution that looks at the actual sound information to organize the sound resources. Currently, Kick Drum, Snare Drum, Clap Drum, Closed Hat Cymbal, and Open Hat Cymbal audio samples are mixed together and need to be organized, additionally, future sound resources will need to be organized to the same standard.

Application Benefits

This application will provide the benefit of providing a classification for an inputted audio sample. When a sound resource is inputted into the application, a classification will be outputted. This allows us to automate the process of organizing current audio resources and future resources.

The model will be built on 2000+ already organized sound resources in order to provide an accurate prediction for more complex sound profiles. The model requires high accuracy, 90% and above, in order to accurately differentiate cymbal types. The difference between a Kick drum and a Snare drum is great, the difference between a Closed Hat cymbal and an Open Hat cymbal is nuanced.

Application Outline

This application will be provided as a jupyter notebook file to run the classification process against an already saved model. The files for creating the model, and the dataset used to create the model will be provided with the application for future growth of the application. Examples include the addition of more samples, such as Tom drums, and 808 Bass drums.

The online implementation will run a single input against the model, classifying one resource at a time. However, the model will be flexible enough to be implemented in a local instance that can classify larger resource amounts.

Data Description

This application will use data provided by H3 Music Corp. The data is 2000+ audio samples of drum sounds. These resources are short 0-4 second audio samples in Uncompressed WAV File formats. This data will be converted into arrays of pitch, time, and amplitude by the application to train the model.

In addition, a CSV file will be provided to give a classification to each sample. This file provides training classifications.

Objective and Hypotheses

The primary objective of this application will be to provide a drum classification for an inputted audio resource to organize current and future audio resources.

The Hypothesis is as follows:

1. From the classified audio resources, we can create a foundation to build a supervised deep-learning Convolutional Neural Network. This model will be able to classify with an accuracy of 90% or more with the dataset size of 2000+ files.
2. With this tool, H3 Music Corp personnel will be able to effectively, and efficiently organize current and future audio resources, without requiring direct manual listening, and observation.
3. This will free up people resources to spend more time creating music and managing the business.

Project Methodology

This project will be developed under a Kanban Methodology due to its lightweight and flexible approach to project management. Since a majority of the project will be completed by a single developer, it does not make sense for us to apply bureaucratic methodologies built for larger teams requiring higher levels of administration, such as waterfall (Scheiner, 2022).

The aspects of Kanban most relevant to our situation are its focus on customer expectations, work management as opposed to worker management, and constant review of provided services (Naydenov, 2019).

Kanban does not apply strict phases like waterfall, which realistically would not have been followed in a one-person team. Rather, it provides strategic goal-oriented methods to build toward deliverables. The goals are as follows: Initial Model Creation, Model Testing, Model Deployment, and Employee Training.

Initial Model Creation: The goal of this portion of the project will be to deliver a deep-learning model capable of classifying drum samples by type. This will require a clean dataset, a built model, and the ability for users to interact with the model.

Model Testing: The goal of this portion is to verify the 90%+ accuracy of the model. In order to be effective the model has to be capable of accurate classification. If the model fails to meet accuracy levels, the dataset and model will need to be adjusted.

Model Deployment: The goal of this portion is to deploy the model to all local workstations for staff usage. The installation of an application to multiple different machines may result in instability of the application, and errors related to different machines have to be detected and fixed.

Employee Training: The goal of this portion is to teach staff and faculty how to effectively use the application. The staff will be interacting with the application regularly and will require an understanding of how it works and how to use it.

Funding Requirements

H3 Music Corp has a music producer that also will work as an internal developer, so for calculating funding we will use the opportunity cost of the employee spending time working on the project as opposed to producing music. The employee's average music production income is $500 / 8-hour workday.

Since there is a large number of audio resources locally stored, it makes sense for the application to be internally hosted on a local server or an instance on each individual machine. There are no additional hosting costs.

Description	**Estimated Time**	**Cost (based on $500/day)**
Initial Model Creation	2 Work Days	$ 1000
Model Testing	1 Work Day	$ 500
Model Deployment	2 Work Days	$ 1000
Employee Training	.5 Work Days	$ 250
Total	**5.5 Work Days (44 hours)**	**$ 2750**

Stakeholder Impact

H3 Music Corp has seen an 86% growth in influence as seen through media interaction including website interaction and social media interaction. With this increase in influence, we also see a parallel increase in service demand. As this service demand increases, employees and owners of H3 Music have a vested interest in shortening or removing mundane tasks such as organizing audio resources to allow more time to be spent directly working on services.

H3 Music clients also have a vested interest in the application, since the removal of mundane tasks raises the threshold at which H3 Music Corp needs to begin raising service costs due to demand remaining and supply decreasing. If the production of a music service takes 2 hours to complete, then every 2 hours saved through the application is another service that can be performed without a decrease in supply, keeping the pricing the same.

Ethical and Legal Considerations

Since the data is in-house and owned by H3 Music Corp. the possible ethical and legal problems should be very limited. Luckily, the dataset does not involve any Personally Identifiable Information, just audio data.

The one consideration that should be taken into account is copyright and usage rights. H3 Music Corp has to maintain a dataset of legally acquired works that are available to be used in AI datasets. There has been a lot of news in regard to copyright images being used in AI datasets against owner usage rights, causing major AI companies to be sued (Growcoot, 2023). Currently, there have been similar legal actions taken in regard to whole copyrighted songs, but none in regard to short audio samples. Currently, it has not been legally tested if you can even protect a single <2 second drum sample. While you own a copyrighted sound, proving infringement, and proving lack of transformative use would be very difficult -- Perhaps an additional future use of this application. Regardless of the legal gray area that audio samples fall into, it is in H3 Music's best interest to avoid adding to the dataset samples that explicitly state copyright terms involving AI.

Developer's Expertise

H3 Music Corp. will be using an internal employee as the software developer for this application. They will be able to provide a stronger understanding of the company's processes, and music production in general. Through preliminary research, it became clear that audio Deep Learning AIs have significantly less documentation than visual. A member of the team with deep knowledge of audio production would be able to fill in this gap in knowledge. Additionally, an internal developer should be able to provide quicker support post-deployment.

B. Executive Summary

Problem Statement

As H3 Music Corp has grown, it has become difficult to maintain the audio resources required for music production. We can no longer manually organize and standardize the current resources and will continue to have problems moving forward without a solution. The goal of this project is to provide that solution by creating a drum classification application to classify our drum audio resources automatically using a supervised deep-learning Convolutional Neural Network. Enabling us to spend more time producing music, and less time managing.

Customer Summary

This application will be for internal H3 Music Corp use only. The customer is H3 Music Corp and users will include all H3 Music employees and contractors working in music production. Users will be required to have an understanding of the application, particularly in the first iteration of the product, to effectively use it.

Existing System Analysis

H3 Music Corp does not have a system in place to complete this process. Currently, resources have to be manually organized, this system would be completely new. Local instances of this application will require 531 MB, for versions capable of creating a new model, and 28.5 MB for versions just needing to run the application. Both sizes should not be a problem on current workstations, since they were originally built with music production in mind, a CPU and storage intensive process. Additionally, this application should not lead to noticeable increases in hardware maintenance or support.

Data

The dataset used to create the initial model will be a collection of H3 Music audio resources that have been properly classified. The data will be stored as the original Uncompressed WAV Files to allow easy interaction with users. The flexibility of playing an audio file to troubleshoot, and confirm situations will be very important.

The audio files will be trimmed and resampled to remove unnecessary data before processing. To feed the audio files into the model, we will require each file to be converted into arrays of information. This information will be Pitch, Time, and Amplification. The model will require equal-sized samples, we will add zero values to fill in the information that is missing.

The model itself will be a Convolutional Neural Network (CNN). The power of the CNN in our application will be its high efficiency and accuracy in media processing (Datagen, 2023), and its ability to recognize patterns in samples that are spatially different (Nandi, 2021). For example, If a kick drum sample has a beat 1-2 seconds into the sample instead of immediately like all other samples, it will still be able to process it the same way. Another example would be pitch, if the kick drum is higher in pitch than the other ones, it will still be processed the same way. This flexibility is required in our application.

One complication involved in our model choice will be CNNs are built for image classification, not audio. To remedy this we will have to apply our amplifications equally to all three image channels (RGB) to confuse the model into thinking the data is image data. (Nandi, 2021)

Project Methodology

This project will be developed under a Kanban Methodology due to its lightweight and flexible approach to project management. Since a majority of the project will be completed by a single developer, it does not make sense for us to apply bureaucratic methodologies built for larger teams requiring higher levels of administration, such as waterfall (Scheiner, 2022).

The aspects of Kanban most relevant to our situation are its focus on customer expectations, work management as opposed to worker management, and constant review of provided services (Naydenov, 2019).

Kanban does not apply strict phases like waterfall, which realistically would not have been followed in a one-person team. Rather, it provides strategic goal-oriented methods to build toward deliverables. The goals are as follows: Initial Model Creation, Model Testing, Model Deployment, and Employee Training.

Initial Model Creation: The goal of this portion of the project will be to deliver a deep-learning model capable of classifying drum samples by type. This will require a clean dataset, a built model, and the ability for users to interact with the model.

Model Testing: The goal of this portion is to verify the 90%+ accuracy of the model. In order to be effective the model has to be capable of accurate classification. If the model fails to meet accuracy levels, the dataset and model will need to be adjusted.

Model Deployment: The goal of this portion is to deploy the model to all local workstations for staff usage. The installation of an application to multiple different machines may result in instability of the application, and errors related to different machines have to be detected and fixed.

Employee Training: The goal of this portion is to teach staff and faculty how to effectively use the application. The staff will be interacting with the application regularly and will require an understanding of how it works and how to use it.

Project Outcome

The project's outcome will be the result of all deliverables being provided.

Deliverables:

1. Classified Dataset: The data provided by H3 Music will have to be properly organized, and standardized to create the dataset for the model. Outlier samples should be removed, including noisy, irregular, and distorted samples. This will be required to train the initial model and subsequent updated models.
2. Usable Model: The saved usable model will be required to run the application. A usable model will have to be delivered to provide functionality to multiple instances of the application.
3. Application: A runnable version of the application will be required, one that meets the requirements by receiving an inputted audio file and properly classifies the file.
4. Program to Update Model: An additional program will be required to provide the functionality of updating the model to an updated version with a new dataset.
5. User Guide: A User Guide to simply and practically explain functionality to Users will be required to support staff and faculty.

Implementation Plan

This project will be implemented using a Kanban Methodology due to its single-developer workflow. The benefit of Kanban is its goal-oriented workflow. The goals are as follows: Initial Model Creation, Model Testing, Model Deployment, and Employee Training.

1. Initial Model Creation: The goal of this portion of the project will be to deliver a deep-learning model capable of classifying drum samples by type.
    a. The dataset has to be cleaned up. This will require an assisted manual process of cleaning audio samples. One example is removing sub-bass humming noises from samples, this can cause distortion and artifices that can interfere with deep

learning. Another example would be the manual correction of reverbed instruments. Some samples will have audio "tails" from reverb lasting upwards of 10-30 seconds, without providing any additional data for the model. These should be trimmed. Both examples can be remedied through sample length analysis, each sample over 2 seconds long should be manually reviewed before adding it to the dataset.

   b. The data has to be converted from audio files into arrays of data. This process requires pitch, time, and amplitude data to be converted into arrays. The Convolutional Neural Network will require "image data" as opposed to audio data, to remedy this audio data has to be duplicated across all 3 image channels (RGB) (Nandi, 2021).
   c. The model has to be built. A Convolutional Neural Network has to be built to effectively classify audio data. This model will take the data of an audio file, and output a predicted classification of the drum type.
   d. User Interaction has to be built. A method for accessing the model to predict audio classifications will be required. This will most likely be a separate file from the one to build the model.

2. Model Testing: The goal of this portion is to verify the 90%+ accuracy of the model. In order to be effective the model has to be capable of accurate classification. If the model fails to meet accuracy levels, the dataset and model will need to be adjusted.
   a. The model will require a testing dataset. A portion of the dataset will have to be used to test the model. Standard test data sizes range from 20-25% of the entire dataset. This will be used to verify accuracy.
   b. The model will require a validation dataset. A portion of the dataset will have to be used to validate the model. Standard test data sizes range from 15-20% of the entire dataset. This will be used to optimize parameters.
   c. The goal is to have training accuracy above 90%.
3. Model Deployment: The goal of this portion is to deploy the model to all local workstations for staff usage.
   a. The model will have to be deployed to multiple workstations in the organization. The software has to be verified to run on all machines without problems related to hardware and software configuration. The software has to be stable.

4. Employee Training: The goal of this portion is to teach staff and faculty how to effectively use the application.
    a. Staff and faculty usually have very different experiences and perspectives, which will require thorough explanations and guidance.
    b. Additionally, the additional experiences and perspectives may lead to software revisions to better suit the users.

Evaluation Plan

For each goal verification and validation will be required. At goal one, the Initial Model Creation, verification of proper conversion from audio files to arrays will be required. This step will be completed by proper visualization of amplification and pitch before and after conversion. At goal two, Model Testing, the model itself will be tested with built-in testing and validation datasets, to examine the accuracy and losses of the model. This step will require 90%+ training model accuracy. For goal three, Model Deployment, executive involvement in verification will be required, the application will have to be black-box tested using Boundary Value Analysis. It will be important to compare close values, such as Closed Hat Cymbals against Open Hat Cymbals to very the model works on all platforms effectively, especially at close limits. For goal four, Employee Training, users will have to provide acceptance of the software or request further testing, quality assurance or training.

Resources and Costs

Hardware and Software:

There should be no costs associated with hardware and software resources. The application will run on already-in-use producer workstations, and all software resources specific to the application are open-source. The producer workstations and development environment all run 64-bit Windows 10.

The required software and recommended versions:

- Python 3.10 (Language)
- Jupyter 6.5.4 (Interactive Computing)
- Pip 23.1.2 (Module Installer)
- matplotlib 3.7.0 (Graph Library)
- pandas 1.5.3 (Data Analysis Library)
- librosa 0.10.0 (Audio Library)

- sklearn 0.0 (Machine Learning Library)
- numpy 1.23.5 (Data Structures Library)
- tensorflow 2.10.0 (Machine Learning Library)
- IPython 8.10.0 (Interactive Library)

Labor:

H3 Music Corp has a music producer that also will work as an internal developer, so for calculating funding we will use the opportunity cost of the employee spending time working on the project as opposed to producing music. The employee's average music production income is $500 / 8-hour workday.

Since there is a large number of audio resources locally stored, it makes sense for the application to be internally hosted on a local server or an instance on each individual machine. There are no additional hosting costs.

Description	**Estimated Time**	**Cost (based on $500/day)**
Initial Model Creation	2 Work Days	$ 1000
Model Testing	1 Work Day	$ 500
Model Deployment	2 Work Days	$ 1000
Employee Training	.5 Work Days	$ 250
Total	**5.5 Work Days (44 hours)**	**$ 2750**

Environment:

Beyond the human resource costs mentioned previously, there will be no additional environmental costs. The application will be hosted on local machines already owned by the company, and the infrastructure to use the application is already maintained by the company. Maintenance of the application will require human resources based on the developer's time. Major maintenance will be required when dependencies break, or H3 Music Corp decides they require a model and dataset update. Model and dataset updates will require similar time and costs to the initial deployment.

Timeline and Milestones

Milestone	**Start Date**	**End Date**

Initial Model Creation	June 5th	June 6th
Model Testing	June 7th	June 7th
Model Deployment	June 8th	June 9th
Employee Training	June 10th	June 10th

C. Application

Visualizations

These visualizations are available in the application “drum_prediction_model.ipynb” file. If they do not load properly these are the visualizations.

Figures Set 1: Sample Waveform of each Drum Type (Amplification over Time)

Figure 1: clap_0001.wav

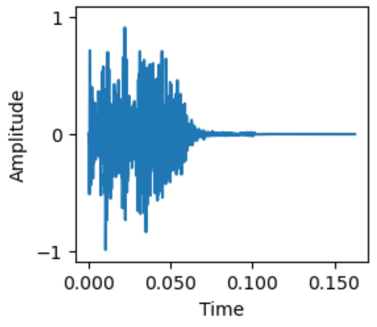


Figure 2: closedhat_0001.wav

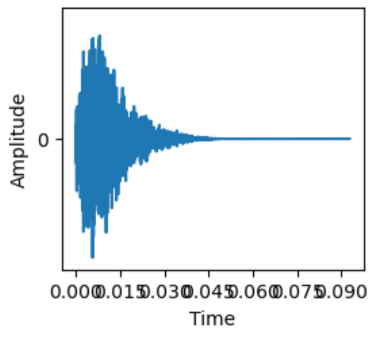


Figure 3: kick_0001.wav

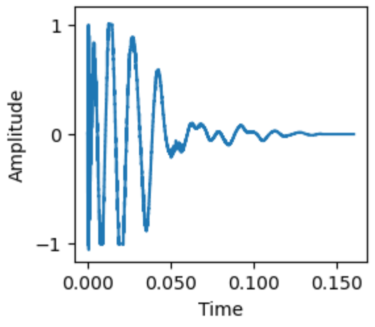


Figure 4: openhat_0001.wav

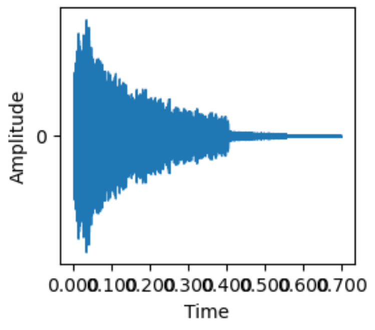
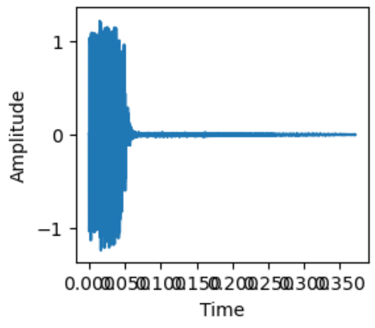


Figure 5: snare_0001.wav



Figures Set 2: All Spectrograms, organized by Drum Type (Pitch over Time)

Figure 1: Clap Samples

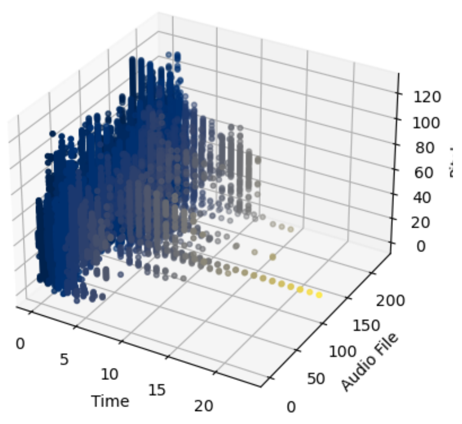


Figure 2: Closed Hat Samples

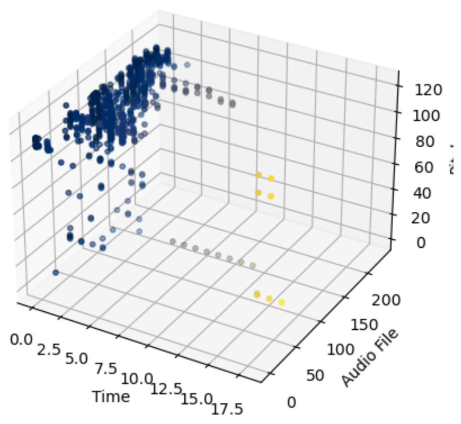


Figure 3: Kick Samples

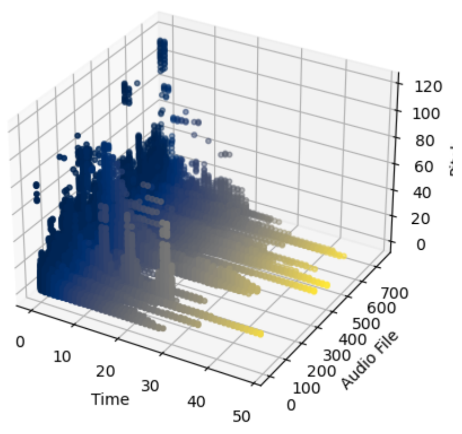


Figure 4: Open Hat Samples

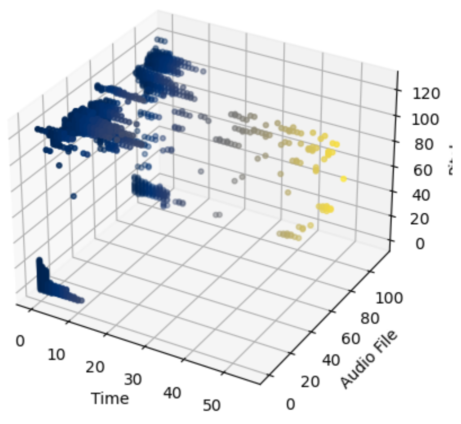
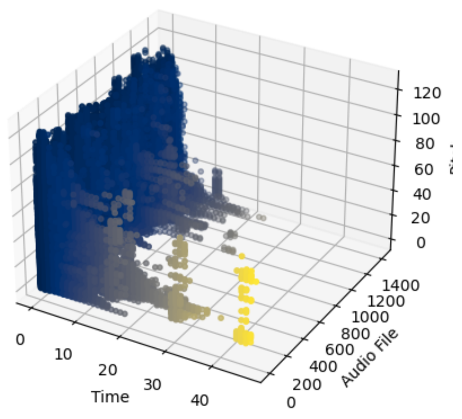
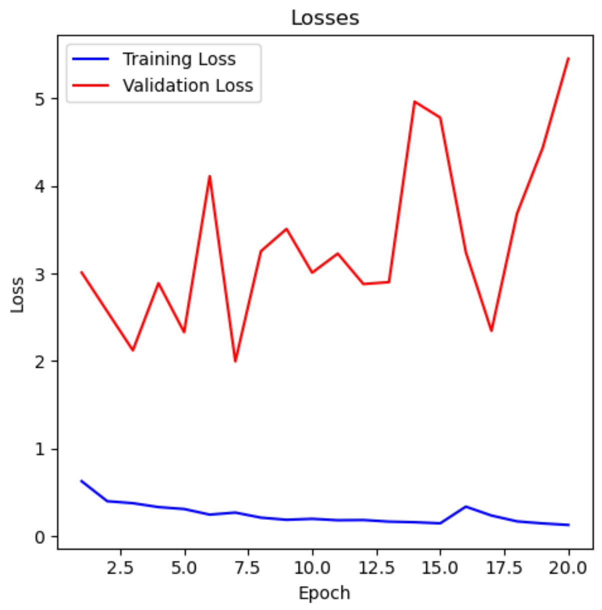
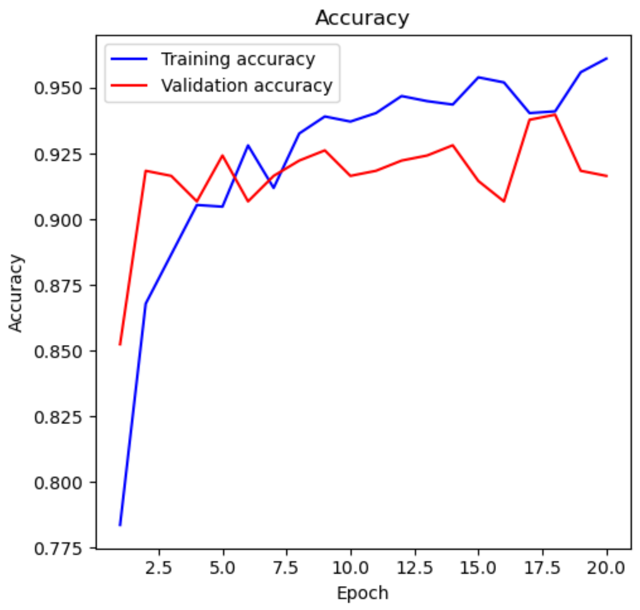


Figure 5: Snare Samples



Figures Set 3: Accuracy and Losses of Saved Model





D. Post-Implementation Report

Business Vision

The original purpose of this project was to provide a drum classification application to assist H3 Music Corp with organizing music production resources. With the increasing scale of H3 Music's services and commitments, time spent on mundane tasks such as organizing files would be better suited to directly working on services for clients.

The built application removes a massive management task for the staff and faculty at H3 Music Corp.

The most time-consuming part of the organization process was manually listening to each file in order to properly classify the file. Now, the application processes the file to determine the classification of the sample without requiring manual auditorial verification. This allows the current audio resources to be easily organized. Additionally, future resources will easily be organized to the same standard as they are added to H3 Music Corp. This process is completed in an objective manner compared to the previous method since it removes the individual's manual organization.

Datasets

The raw dataset is directly from H3 Music Corp, consisting of 2,746 drum audio samples with a CSV File associating the file with a drum type classification. These files were provided and maintained in Uncompressed WAV file formats. The reasoning for this is for development and post-development it is easier for users and developers to verify and interact with audio files rather than arrays. If a developer needs to verify the type of drum sample, they can just listen to it, removing a lot of hassle.

The raw Uncompressed WAV Files are available in the dataset/samples folder. The CSV File to associate types is available in the dataset folder, titled “samples_metadata.csv”.

Data Product Code

The product code is broken into 2 files. “Drum_prediction_model.ipynb” to create the model and “drum_prediction.ipynb” to run the saved model against an input file.

Drum_prediction_model.ipynb:

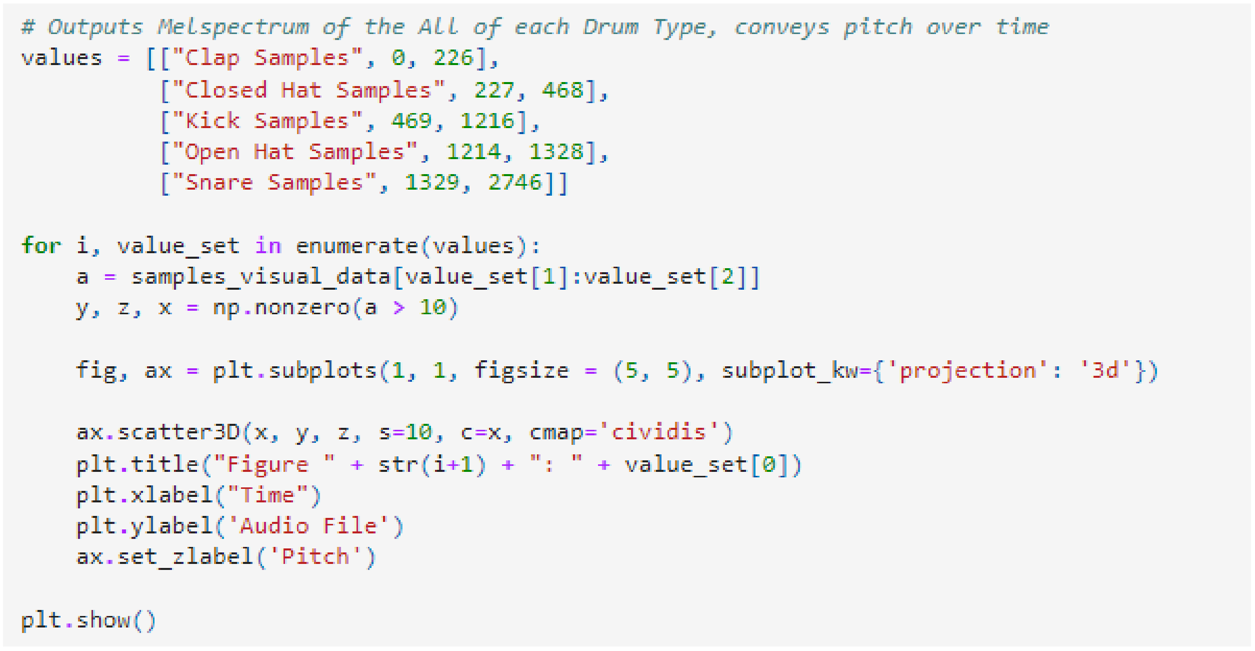
The first major code block is used to convert the audio files into data arrays to be fed into the model. For each inputted sample, a multidimensional array of zeros is created to keep all arrays fed into the model the same size. The audio is loaded, trimmed to remove silence, and resampled to keep all files at the same audio sample rate. Using the melspectrogram feature of librosa, the edited audio is converted into a multidimensional array, of pitch (frequency: 0-128), time (samples), and the amplification of that pitch. These values then overwrite the zeros in the multidimensional zero array. This multidimensional array is added to a list of sample data. Additionally, the corresponding drum type is added to a list of sample drum types. Each corresponding sample and drum type will have the same index.

```python
# Converts Audio Samples in single Numpy Array to train model

# Gets relative path to audio samples for dataset
samples = abs_path + "\\dataset\\samples\\"

# Create blank lists to fill
samples_data = []
samples_visual_data = []
samples_data_type = []

for sample in os.listdir(samples):
    # Blank arrays of zeros to be overwritten by sample data, required to make all sample arrays the same size
    # Array Dimensions are as follows: D1: Pitch Slices (128), D2: Time samples (LENGTH), and D3: value of power/amplitud
    # D3 is duplicated across 3 values in the model version since CNNs were made with RGB (3) image values in mind. It's
    # simpler to make the CNN Model think it is a RGB image than try and make it work for audio specifically
    sample_array = np.zeros((128, LENGTH, 3))
    visual_array = np.zeros((128, LENGTH))

    # Load, Trim, and resample the audio file
    audio, sr = librosa.load(samples+sample,sr=22050)
    audio, _ = librosa.effects.trim(audio, top_db=50)
    audio = librosa.resample(y=audio, orig_sr=sr, target_sr=22050)

    # Converts the audio file into a numpy array based on melspectrogram values (Pitch, Time and Power)
    melspect = librosa.feature.melspectrogram(y=audio)

    # Overwrites zeros in sample_array and visual_array with actual audio sample data
    for i, _ in enumerate(melspect): #128
        for j, _ in enumerate(melspect[i]): #LENGTH
            sample_array[i][j] = melspect[i][j]
            visual_array[i][j] = melspect[i][j]


    # Adds audio samples data to the list of samples_data, and samples_visual_data
    samples_data.append(sample_array)
    samples_visual_data.append(visual_array)

    # Adds audio samples type of drum to samples_data_type
    samples_data_type.append([val for key,val in drum_types.items() if key in sample])

# Convert Lists to Numpy Arrays
samples_data = np.array(samples_data)
samples_visual_data = np.array(samples_visual_data)
samples_data_type = np.array(samples_data_type)
```

The next code block prints a waveform graph of each first drum type. This provides visualizations for verification.

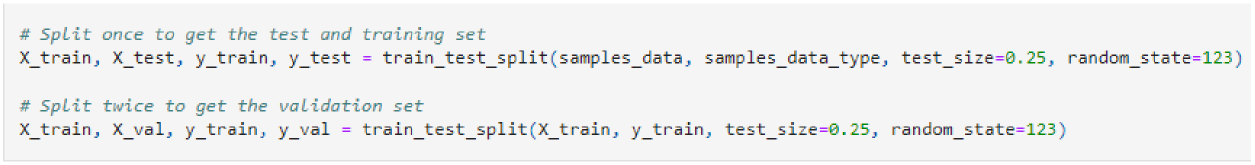
```python
# Outputs Waveform of the first of each Drum Type, conveys amplitude over time
for i, drum in enumerate(drum_types):
    y, sr = librosa.load(samples + drum + '_0001.wav')

    y,z = librosa.effects.trim(y, top_db=50)
    y = librosa.resample(y=y, orig_sr=sr, target_sr=22050)

    plt.figure(figsize=(3, 3))
    plt.suptitle("Figure " + str(i+1) + ": " + drum + '_0001.wav',x=0.5, y=0.915,fontsize=12)
    plt.xlabel("Time")
    plt.ylabel('Amplitude')

    plt.xticks(np.arange(0, 1, 1))
    plt.yticks(np.arange(-2, 2, 1))
    plt.tight_layout(pad=1.0)

    librosa.display.waveshow(y, sr=sr)
```

The next code block provides Spectrogram Graphs for all samples, separated by drum type. This helps visual the Pitch over Time differences between drum types.

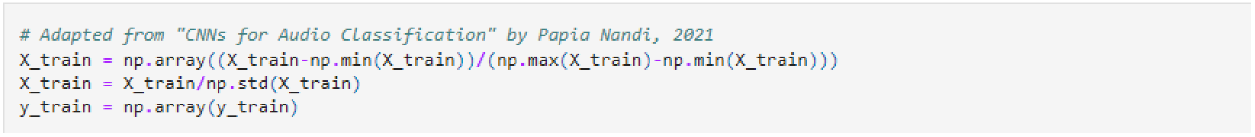


```
# Outputs Melspectrum of the All of each Drum Type, conveys pitch over time
values = [["Clap Samples", 0, 226],
          ["Closed Hat Samples", 227, 468],
          ["Kick Samples", 469, 1216],
          ["Open Hat Samples", 1214, 1328],
          ["Snare Samples", 1329, 2746]]

for i, value_set in enumerate(values):
    a = samples_visual_data[value_set[1]:value_set[2]]
    y, z, x = np.nonzero(a > 10)

    fig, ax = plt.subplots(1, 1, figsize = (5, 5), subplot_kw={'projection': '3d'})

    ax.scatter3D(x, y, z, s=10, c=x, cmap='cividis')
    plt.title("Figure " + str(i+1) + ": " + value_set[0])
    plt.xlabel("Time")
    plt.ylabel('Audio File')
    ax.set_zlabel('Pitch')

plt.show()
```

The following code blocks split the formatted sample data into three portions, the training dataset, the test dataset, and the validation dataset. These are used for Testing and validation of the model.

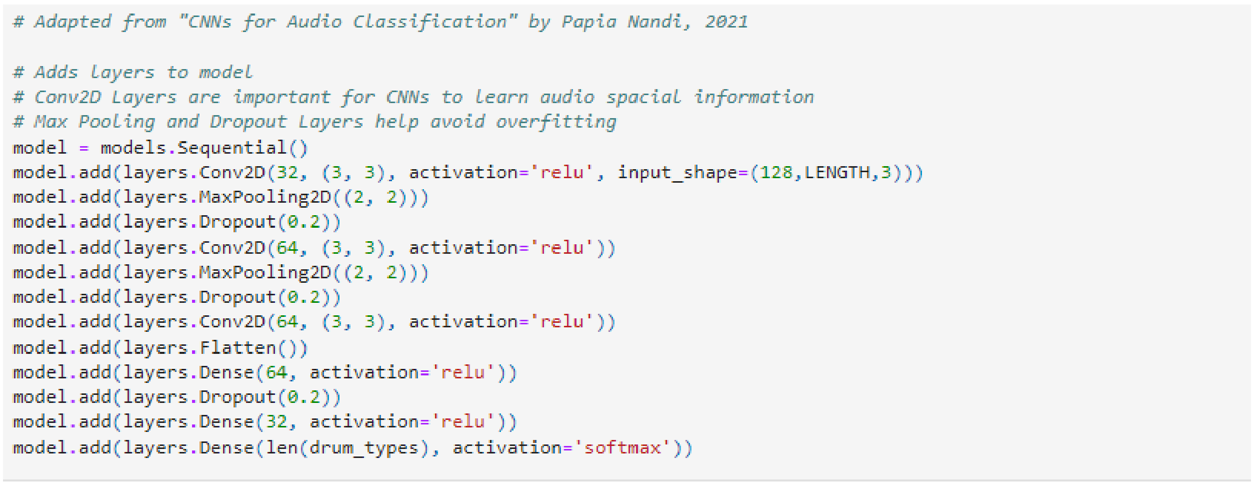


```
# Split once to get the test and training set
X_train, X_test, y_train, y_test = train_test_split(samples_data, samples_data_type, test_size=0.25, random_state=123)

# Split twice to get the validation set
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=123)
```

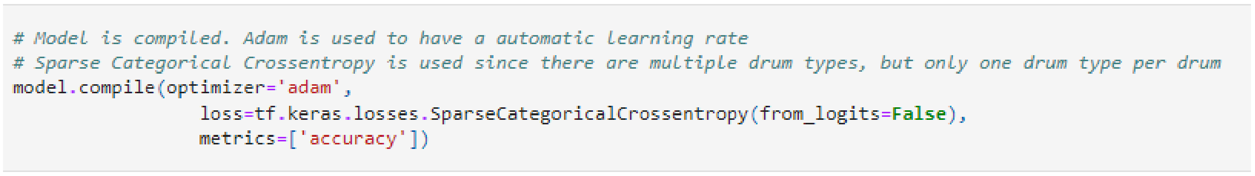


```
# Adapted from "CNNs for Audio Classification" by Papia Nandi, 2021
X_train = np.array((X_train-np.min(X_train))/(np.max(X_train)-np.min(X_train)))
X_train = X_train/np.std(X_train)
y_train = np.array(y_train)
```

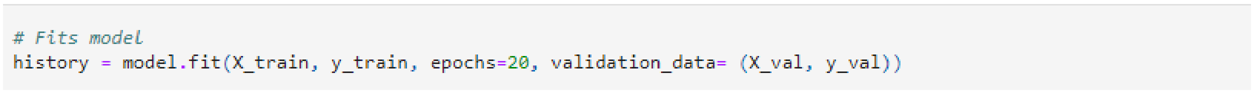
The next code blocks create the model and train it. A CNN model is characterized by its ability to recognize translated features through the use of feature maps. These maps are created through Convolutional Layers, Conv2D in our program. The Convolutional Layer abstracts inputted data into feature maps to be outputted to the next layer. The Pooling and Dropout layers are used to flatten and simplify the Convolutional Layers to avoid overfitting and an over-saturation of data. The final Dense layers fit the model into the original parameters of drum types, in our case 5 options. The model is compiled using Adam to allow an automatic learning rate, and Sparse Categorical Crossentropy is used since each sample has one corresponding type, not multiple. A kick drum sample can only be a kick drum sample, not a kick and a snare. (featuredpeow, 2019)



```
# Adapted from "CNNs for Audio Classification" by Papia Nandi, 2021

# Adds layers to model
# Conv2D Layers are important for CNNs to learn audio spacial information
# Max Pooling and Dropout Layers help avoid overfitting
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128,LENGTH,3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.2))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.2))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(len(drum_types), activation='softmax'))
```



```
# Model is compiled. Adam is used to have a automatic learning rate
# Sparse Categorical Crossentropy is used since there are multiple drum types, but only one drum type per drum
model.compile(optimizer='adam',
                 loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                 metrics=['accuracy'])
```



```
# Fits model
history = model.fit(X_train, y_train, epochs=20, validation_data= (X_val, y_val))
```

In the next block, the model is exported as loss and accuracy graphs. These graphs allow us to visualize optimization problems, such as overfitting and underfitting, and overall accuracy over each epoch.

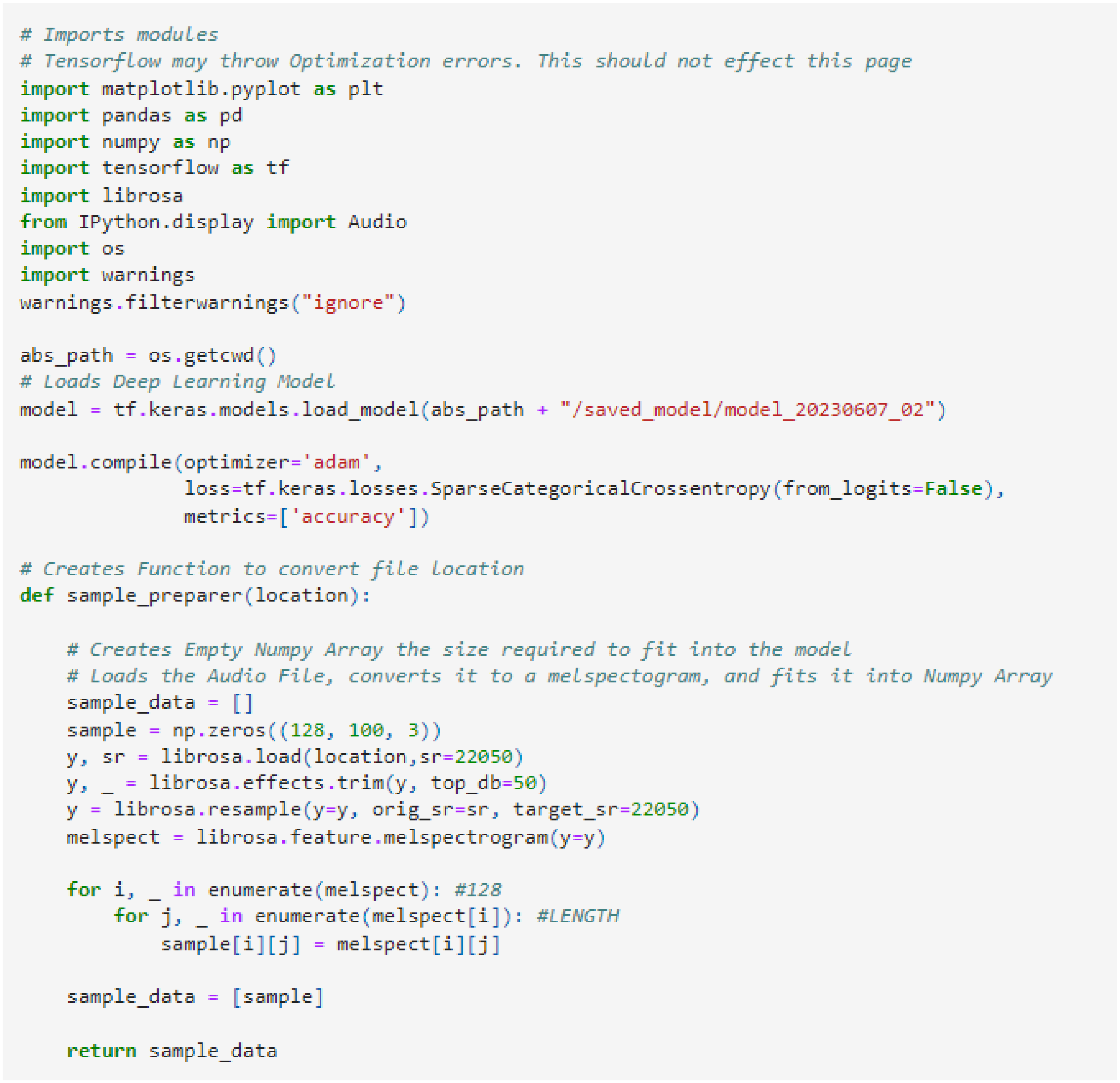
```
# Adapted from "CNNs for Audio Classification" by Papia Nandi, 2021
# Nandi originally adapted from Deep Learning with Python by Francois Chollet, 2018

# Outputs graphs of Losses and Accuracy of model.
# By epoch 20 we are sitting at Training losses of .1278 and Accuracy of .9611
history_dict=history.history
loss_values=history_dict['loss']
accuracy_values=history_dict['accuracy']
val_loss_values = history_dict['val_loss']
val_accuracy_values=history_dict['val_accuracy']
epochs=range(1,21)
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,5))

ax1.plot(epochs,loss_values,'blue',label='Training Loss')
ax1.plot(epochs,val_loss_values,'red', label='Validation Loss')
ax1.set_title('Losses')
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Loss')
ax1.legend()

ax2.plot(epochs,accuracy_values,'blue', label='Training accuracy')
ax2.plot(epochs,val_accuracy_values,'red',label='Validation accuracy')
ax2.set_title('Accuracy')
ax2.set_xlabel('Epoch')
ax2.set_ylabel('Accuracy')
ax2.legend()

plt.tight_layout()
plt.show()
```

The final block of code in the model creation process saves the model to be used in the classification portion.

```
# Saves model to be used by the prediction Jupyter Notebook
model.save('saved_model\model_20230607_02')
```

Drum_prediction.ipynb:

The drum_predicition file is used to predict the inputted audio sample's drum classification. The first code block will prepare the classification by importing and loading the saved model, and defining the function used to convert the inputted audio file into the arrays used by the model. This is identical to the conversion used to train the model initially.



```python
# Imports modules
# Tensorflow may throw Optimization errors. This should not effect this page
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import tensorflow as tf
import librosa
from IPython.display import Audio
import os
import warnings
warnings.filterwarnings("ignore")

abs_path = os.getcwd()
# Loads Deep Learning Model
model = tf.keras.models.load_model(abs_path + "/saved_model/model_20230607_02")

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])

# Creates Function to convert file location
def sample_preparer(location):

    # Creates Empty Numpy Array the size required to fit into the model
    # Loads the Audio File, converts it to a melspectogram, and fits it into Numpy Array
    sample_data = []
    sample = np.zeros((128, 100, 3))
    y, sr = librosa.load(location,sr=22050)
    y, _ = librosa.effects.trim(y, top_db=50)
    y = librosa.resample(y=y, orig_sr=sr, target_sr=22050)
    melspect = librosa.feature.melspectrogram(y=y)

    for i, _ in enumerate(melspect): #128
        for j, _ in enumerate(melspect[i]): #LENGTH
            sample[i][j] = melspect[i][j]

    sample_data = [sample]

    return sample_data
```

The next code block will request a file path to an audio sample to be classified. The input will be stored, and an audio player will load to allow verification of the file.

```python
# Asks for User to Input File Path to audio file needing classification

# Available file paths are avialable below, copy them into the input upon request
# Clap: dataset/samples_reserved/input_clap.wav
# Closed Hat: dataset/samples_reserved/input_closedhat.wav
# Kick: dataset/samples_reserved/input_kick.wav
# Open Hat: dataset/samples_reserved/input_openhat.wav
# Snare: dataset/samples_reserved/input_snare.wav

location = input("Input Audio File Path: ")

# Removes quotes around filepath if they exist
location = location.strip('\"')

y, sr = librosa.load(location)

Audio(data=y, rate=sr)
```

The final code block will make a prediction based on the inputted audio sample. Of the prediction, the max value, or most likely drum type will be pulled and printed to the console.

```python
# Audio file is fed into the model, and a prediction for classification is returned
prediction = model.predict(np.array(sample_preparer(location)))

# Unneeded data is removed, only the highest predicted result is required
type_num = np.argmax(prediction,axis=1)

drum_types = ['Clap', 'Closed Hat', 'Kick', 'Open Hat', 'Snare']

# Numeric Drum Classification is converted to String name of drum and outputted.
print(f"\nDrum Sample is: {drum_types[int(type_num)]}")
```

Objective Verification

The Hypothesis is as follows:

1. From the classified audio resources, we can create a foundation to build a supervised deep-learning Convolutional Neural Network. This model will be able to classify with an accuracy of 90% or more with the dataset size of 2000+ files.
2. With this tool, H3 Music Corp personnel will be able to effectively, and efficiently organize current and future audio resources, without requiring direct manual listening, and observation.
3. This will free up people resources to spend more time creating music and managing the business.

Hypothesis 1 was met. The CNN was able to classify with an accuracy of 94.88% what drum type an audio sample was. It used 2,746 files to train the model.

Hypothesis 2 and 3 are related to business evaluation and would require a qualitative analysis of the system directly, or a quantitative analysis of business productivity on a larger scale. Examples include surveying subjective opinions of the staff or comparing business service output pre-application to output post-application.

Effective Visualization and Reporting

Through visualizations we are able to easily see the aspects of the audio being analyzed from the model. Firugre Set 1 & 2 allow easy data exploration, we can visualize the amplification and pitch shapes over time for individual audio files or entire drum types. This helps us understand what data CNN is learning from. For example, we can assume the CNN is differentiating the Kick from other drums based on pitch information since the Kick drum is predominately a lower-pitched instrument.

Figures Set 1: Sample Waveform of each Drum Type (Amplification over Time)

Figure 1: clap_0001.wav

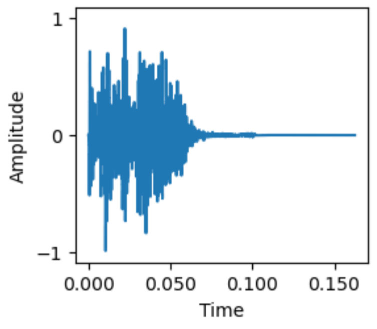


Figure 2: closedhat_0001.wav

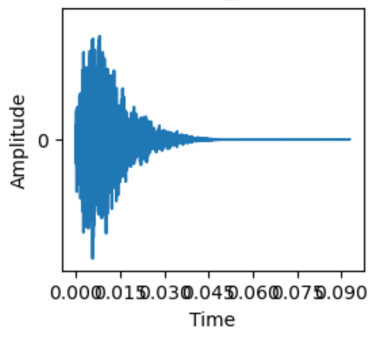


Figure 3: kick_0001.wav

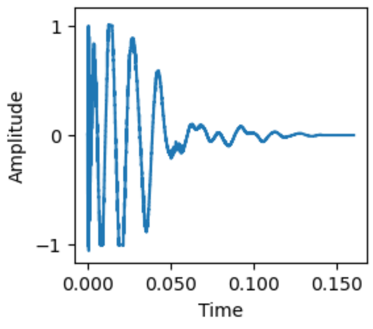


Figure 4: openhat_0001.wav

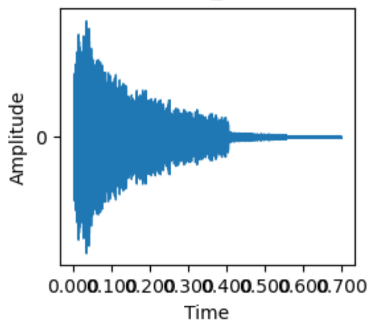
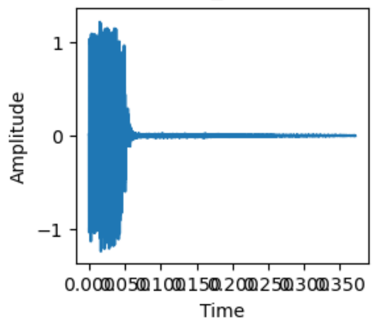


Figure 5: snare_0001.wav



Figures Set 2: All Spectrograms, organized by Drum Type (Pitch over Time)

Figure 1: Clap Samples

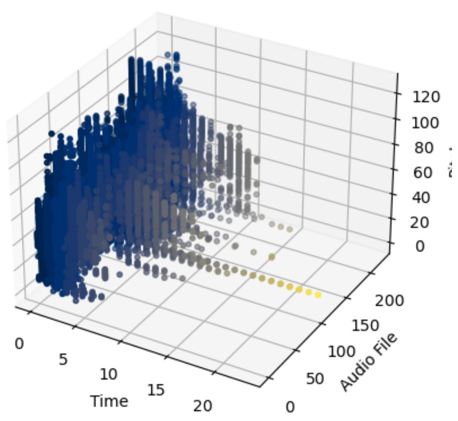


Figure 2: Closed Hat Samples

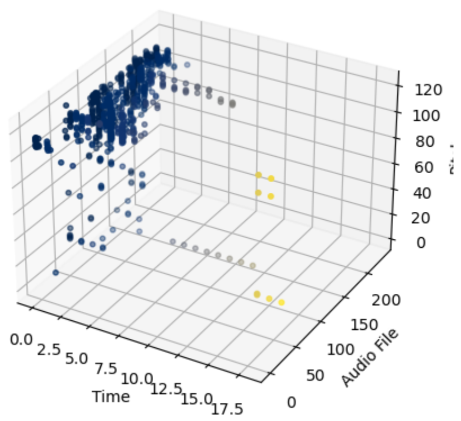


Figure 3: Kick Samples

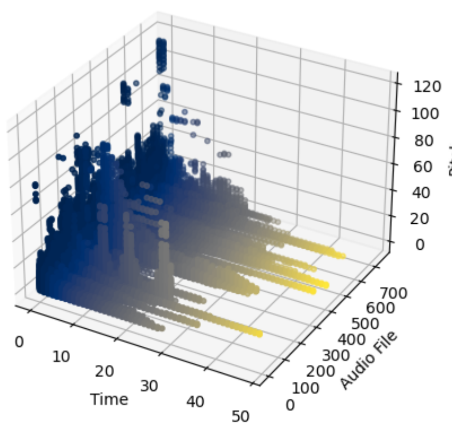


Figure 4: Open Hat Samples

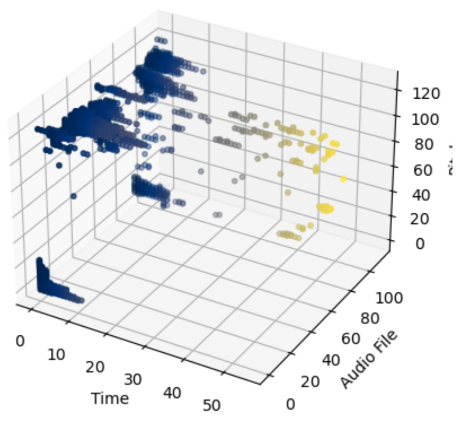


Figure 5: Snare Samples

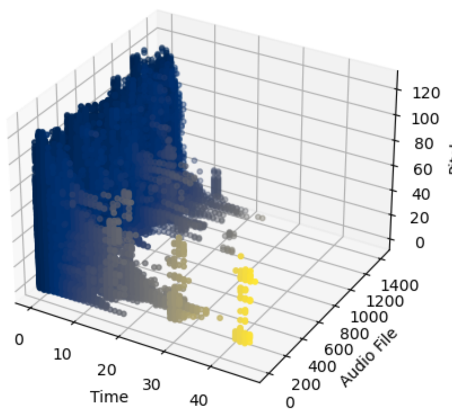
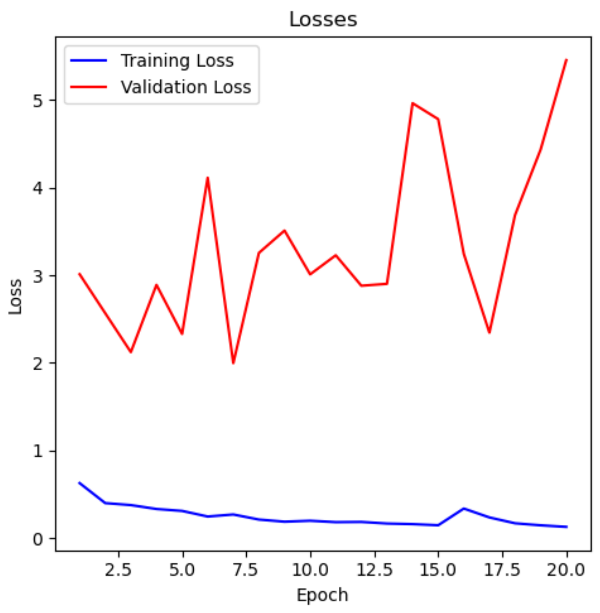
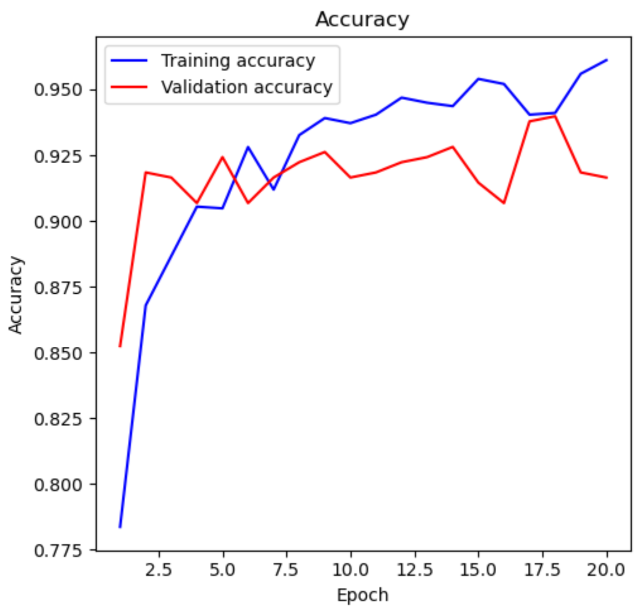


Figure Set 3 provides us the ability to summarize and analyze the model's effectiveness by providing us with loss and accuracy data. We can see the increase in accuracy with each epoch, and make optimization decisions in regard to the model over or underfitting.

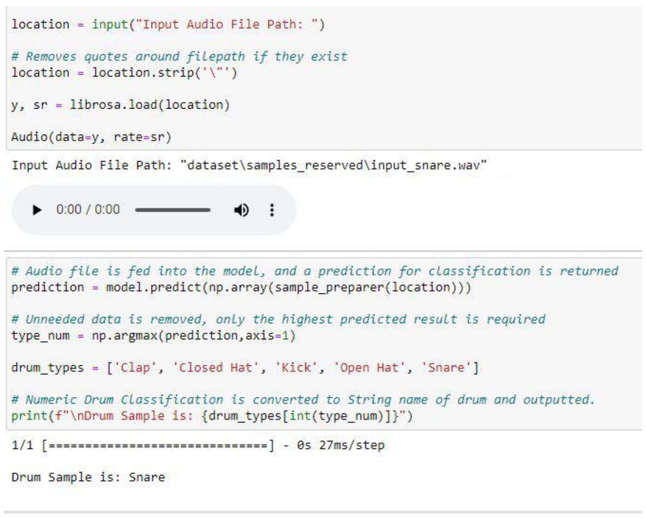
Figures Set 3: Accuracy and Losses of Saved Model





Accuracy Analysis

On a simple level, the classification model accuracy can be analyzed by how effectively it can classify samples. It is accurate enough if it does not have trouble distinguishing different drum types. A black box test for this simple analysis would be Boundary Value Analysis. We can take the two closest-sounding drum types and see if the model can differentiate them. In our case this would be closed and open hat cymbals, which the saved model can differentiate. A quantitative alternative to this method would be to assume 90% training accuracy is required for these results. As we get harder to differentiate sounds, we will require higher accuracy. 90% as a metric provides us with the same outcomes as the black box analysis. On the right is an example of the black box testing of a snare sample.



```
location = input("Input Audio File Path: ")

# Removes quotes around filepath if they exist
location = location.strip('\"')

y, sr = librosa.load(location)

Audio(data=y, rate=sr)
```

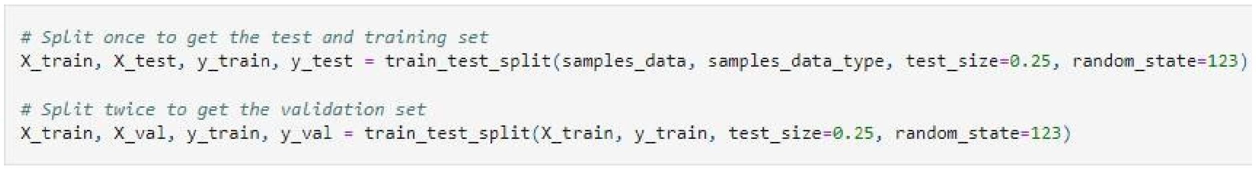
```
Input Audio File Path: "dataset\samples_reserved\input_snare.wav"
```

```
# Audio file is fed into the model, and a prediction for classification is returned
prediction = model.predict(np.array(sample_preparer(location)))

# Unneeded data is removed, only the highest predicted result is required
type_num = np.argmax(prediction,axis=1)

drum_types = ['Clap', 'Closed Hat', 'Kick', 'Open Hat', 'Snare']

# Numeric Drum Classification is converted to String name of drum and outputted.
print(f"\nDrum Sample is: {drum_types[int(type_num)]}")
```

```
1/1 [==============================] - 0s 27ms/step

Drum Sample is: Snare
```

Application Testing

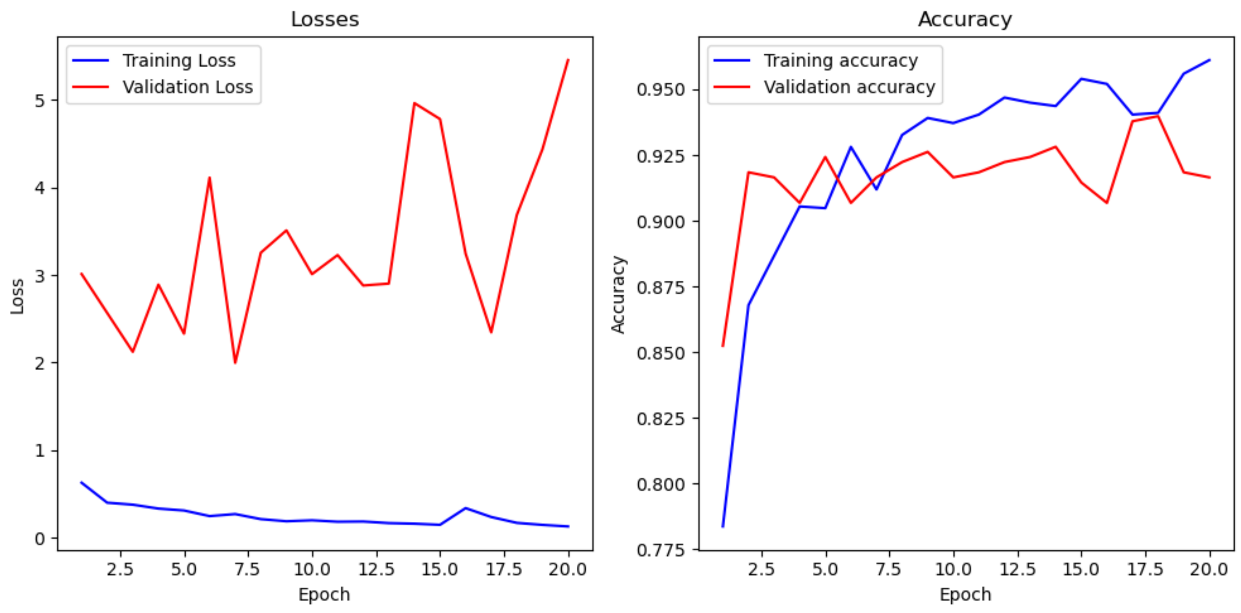
Testing and validation datasets were created to obtain model accuracy and losses.



```
# Split once to get the test and training set
X_train, X_test, y_train, y_test = train_test_split(samples_data, samples_data_type, test_size=0.25, random_state=123)

# Split twice to get the validation set
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=123)
```

These datasets helped the optimization process and verified the model met high enough accuracy to be effective at classifying drum samples. As seen in Figure Set 3, our training model had losses nearing 0.0 and an accuracy of nearly 95%.

Figures Set 3: Accuracy and Losses of Saved Model



In addition to the testing done while the model is built, additional user-level testing is available. 5 Files for testing are available in the "samples_reserved" folder to provide testing of the overall system. Users can input these files into the drum_prediction.ipynb file, and very the correct classification is outputted.

Application Files

All application files are available at: https://github.com/aabalke33/drum-prediction

Hierarchy:

Drum-prediction/

---.ipynb-checkpoints/

---dataset/

------samples/

------samples_reserved/

------samples_metadata.csv

------samples_types.csv

---documentaiton/

---saved_model/

---drum_prediction.ipynb

---drum_prediction_model.ipynb

User's Guide

The GitHub repository includes the user file "drum_prediction.ipynb", the file used to build the model "drum_prediction_model.ipynb", the dataset, the saved model and all documentation related to the assessment.

The application has been built and tested with mybinder.org hosting in mind.

To run the application.

1. Go to https://github.com/aabalke33/drum-prediction.
2. Click the launch binder button. This will create a virtual environment to load the application. This process usually takes 2-5 minutes. If an error occurs please retry, you shouldn't have to retry more than once.

Once this page is loaded, all files related to this submission will be accessible, including the ones used to create the model. As a user, the only required file to access is "drum_prediction.ipynb".

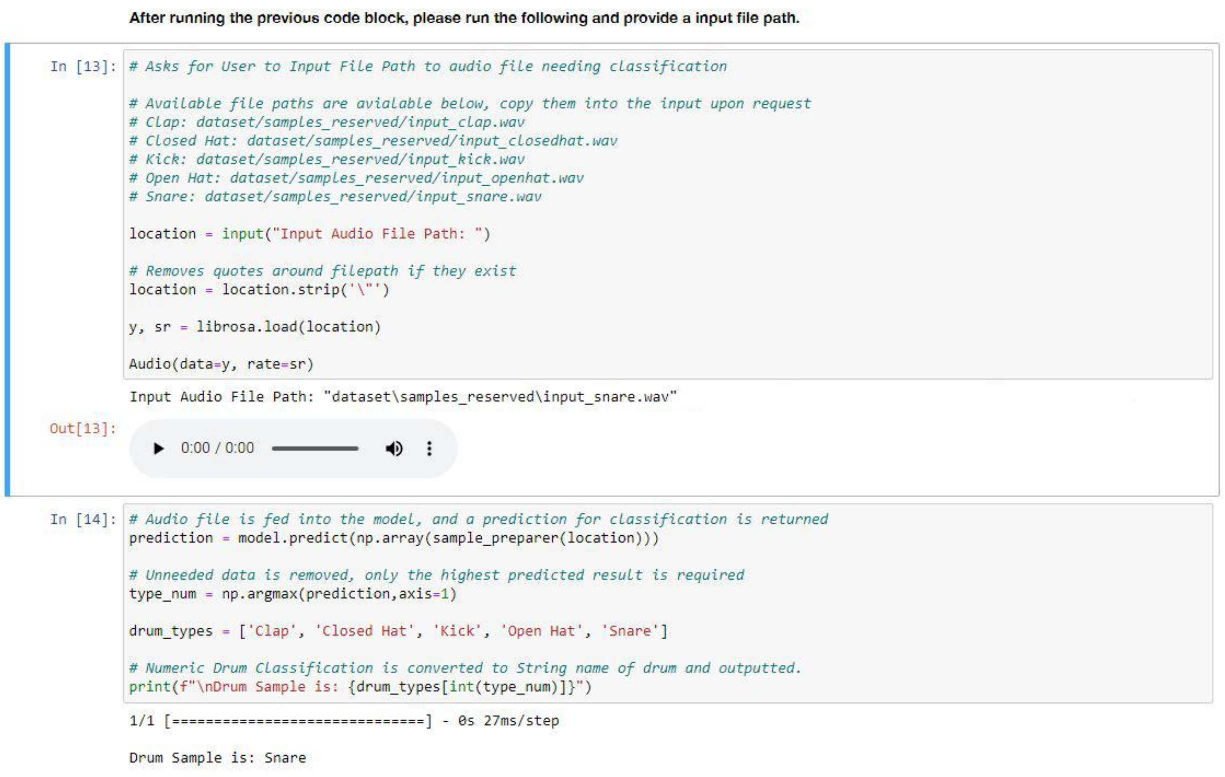
1. Access drum_prediction.ipynb
2. Run the first Code Block (Ctrl + Enter). This will load all required modules and the saved model. You will get warnings regarding Tensorflow dependencies and build suggestions. These do not affect this program.
3. Run the second Code Block (Ctrl + Enter). This will prompt a file path to classify the file. For demonstration purposes, 5 files have been uploaded to classify. The following paths can be copied in as inputs.

Name	File Path
Clap Sample	dataset/samples_reserved/input_clap.wav
Closed Hat Sample	dataset/samples_reserved/input_closedhat.wav
Kick Sample	dataset/samples_reserved/input_kick.wav
Open Hat Sample	dataset/samples_reserved/input_openhat.wav
Snare Sample	dataset/samples_reserved/input_snare.wav

After the file has been loaded, an audio player will load to verify the file chosen.

4. Run the third Code Block (Ctrl + Enter). This final block of code will run a prediction against the saved model to classify the audio sample inputted. The console will print “Drum Sample is: (drum type)” upon completion.

See Figure A for a correctly executed Code Block 2 and Code Block 3.


After running the previous code block, please run the following and provide a input file path.

```
In [13]: # Asks for User to Input File Path to audio file needing classification

         # Available file paths are avialable below, copy them into the input upon request
         # Clap: dataset/samples_reserved/input_clap.wav
         # Closed Hat: dataset/samples_reserved/input_closedhat.wav
         # Kick: dataset/samples_reserved/input_kick.wav
         # Open Hat: dataset/samples_reserved/input_openhat.wav
         # Snare: dataset/samples_reserved/input_snare.wav

         location = input("Input Audio File Path: ")

         # Removes quotes around filepath if they exist
         location = location.strip('\"')

         y, sr = librosa.load(location)

         Audio(data=y, rate=sr)

         Input Audio File Path: "dataset\samples_reserved\input_snare.wav"
Out[13]:
```

```
In [14]: # Audio file is fed into the model, and a prediction for classification is returned
         prediction = model.predict(np.array(sample_preparer(location)))

         # Unneeded data is removed, only the highest predicted result is required
         type_num = np.argmax(prediction,axis=1)

         drum_types = ['Clap', 'Closed Hat', 'Kick', 'Open Hat', 'Snare']

         # Numeric Drum Classification is converted to String name of drum and outputted.
         print(f"\nDrum Sample is: {drum_types[int(type_num)]}")

         1/1 [==============================] - 0s 27ms/step

         Drum Sample is: Snare
```

Figure A: a correctly executed Code Block 2 and Code Block 3.

All other files are provided for evaluation and are not expected to be executed. “Drum_prediction_model.ipynb” would create a new model, if every Code Block is executed in order; however, I do not believe mybinder.org has the processing power to run it.

Summation of Learning Experience

While I have only been truly programming for less than a year, I have been using small scripts and programs to provide solutions to problems I have dealt with since I was fourteen years old. The greatest skill I brought to the table with this project was not a deep understanding of programming and machine learning, but rather Problem-Solving. My ability to problem solve opens the door to programming and machine learning, making these skills easier to learn. In addition to problem-solving, I also have a massive amount of experience in Music Production. I began learning Classical Piano in 2007, and started a Hip-Hop Music Production company in 2014, H3 Music Corp. In history classes, my papers were on Music History, in Physics, my papers were on Acoustics.

Machine Learning was a completely new concept to me. I had no experience in the process or the science behind it. The Artificial Intelligence course I took at Study.com to transfer to WGU was very generalized and related more to the applications and uses of Machine Learning, rather than the actual building of an application. Through actually building a Machine Learning application I have a significantly higher respect for Data Science, which frankly 12 months ago I thought sounded like a ridiculous field. In completing this project, I also realize:

1. I have only scratched the surface of AI and Machine Learning
2. There are tons of applications for AI and Machine Learning in my own work

I will genuinely be using this application to organize H3 Music Corp samples. I also have plans to expand it to convert my tracks from audio files into midi files, or even generate music based on my tracks.

Sources

Datagen. (2023, April 3). Convolutional Neural Network: Benefits, Types, and Applications. Datagen. https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/

featuredpeow. (2019, September 14). Sparse_categorical_crossentropy vs categorical_crossentropy (keras, accuracy). Stack Exchange. https://datascience.stackexchange.com/questions/41921/sparse-categorical-crossentropy-vs-categorical-crossentropy-keras-accuracy

Growcoot, M. (2023, February 7). Getty Images is Suing Stable Diffusion for a Staggering $1.8 Trillion. PetaPixel. https://petapixel.com/2023/02/07/getty-images-are-suing-stable-diffusion-for-a-staggering-1-8-trillion/

Nandi, P. (2021, July 30). CNNs for Audio Classification. Medium. https://towardsdatascience.com/cnns-for-audio-classification-6244954665ab

Naydenov, P. (2019, August 24). Kanban Project Management: How to Use and Implement It. Kanbanize Blog. https://kanbanize.com/blog/how-to-use-kanban-for-project-management/

Scheiner, M. (2022, May 26). Project Management Methodologies Comparison (11 PM Methods). CRM.org. https://crm.org/news/project-management-methodologies