| **Title: Music Genre Classification Progress Report** | |
| --- | --- |
| **Project Team: 32** | |
| **Oluwasina Olowookere** | **1006648068** |
| **Tyler Wong** | **1006999134** |
| **Anna Szatan** | **1006995209** |
| **Blythe Huang** | **1007418153** |

**Abstract**

Given the ever increasing number of songs being uploaded to the internet, streaming services could benefit from automated music classification by genre to handle the large amounts of data and provide improved recommendations to their clients. The team aims to design and train a convolutional neural network (CNN) model that can classify music by its genre. CNNs are highly capable of pattern recognition and thus have been used extensively for speech recognition, lyric transcription, and image classification. Following the conception of the project plan, the team has made extensive progress in working to develop a CNN model capable of identifying the genre of a song based on its features. In order to keep on track with the project, expected deliverables were clearly outlined, and the contributions of each team member were meticulously recorded. One of the primary accomplishments of the team has been collecting music data from a public data repository and converting it to spectrogram form. Following this, the team developed two baseline models, K-Nearest Neighbours (KNN) and Artificial Neural Network (ANN), that work to achieve the goal of music classification, albeit at a lower level of precision than the CNN the team aspires to build. These models were tested and their performance verified using the processed dataset created by the team. Lastly the team has developed the structure of the CNN model, as well as the integral functions that will be used to train and evaluate its performance. Preliminary tests have been run, on this model, with promising results. Moving forward, the team aims to further refine the primary model and test its performance on novel data collected by the team.

1. **Project Description**

As music streaming services become more popular and the number of artists increase, the number of songs published to the internet has increased significantly [1]. Companies whose services revolve around music storage, and organization, such as Spotify, would benefit from the use of an automated music genre classification system to better organize and recommend their collection to consumers [2]. Although artists may pre-classify their music, the characteristics of their songs may mimic that of other genre categories, thus by having a system that can identify those features, improved recommendations can be made to music lovers. Deep learning has shown promise in the field of data organization and recommendation generation as a result of its ability to quickly handle large amounts of data and classify information [3]. As such, the team aims to create a neural network model that is capable of determining the genre of a given piece of music. The team’s interest in this topic is a result of all members having a connection to music, either through playing instruments throughout childhood or being dedicated listeners.

The team decided to build a convolutional neural network (CNN) model for this music genre classification project. This is due to the fact that CNNs are commonly used for image classification with their ability to detect patterns [14]. CNNs contain convolutional layers and each layer contains a set of filters which convolves the data such that a prediction can be made as to what the data is depicting [14]. Its pattern recognition ability shows promise for audio files as it can be used to detect rhythm, pitch and other aspects of the music [14].

To assess the performance of the CNN model, the team decided to use a K-Nearest Neighbour (KNN) model, and an Artificial Neural Network (ANN) model as baselines to compare the primary model to.

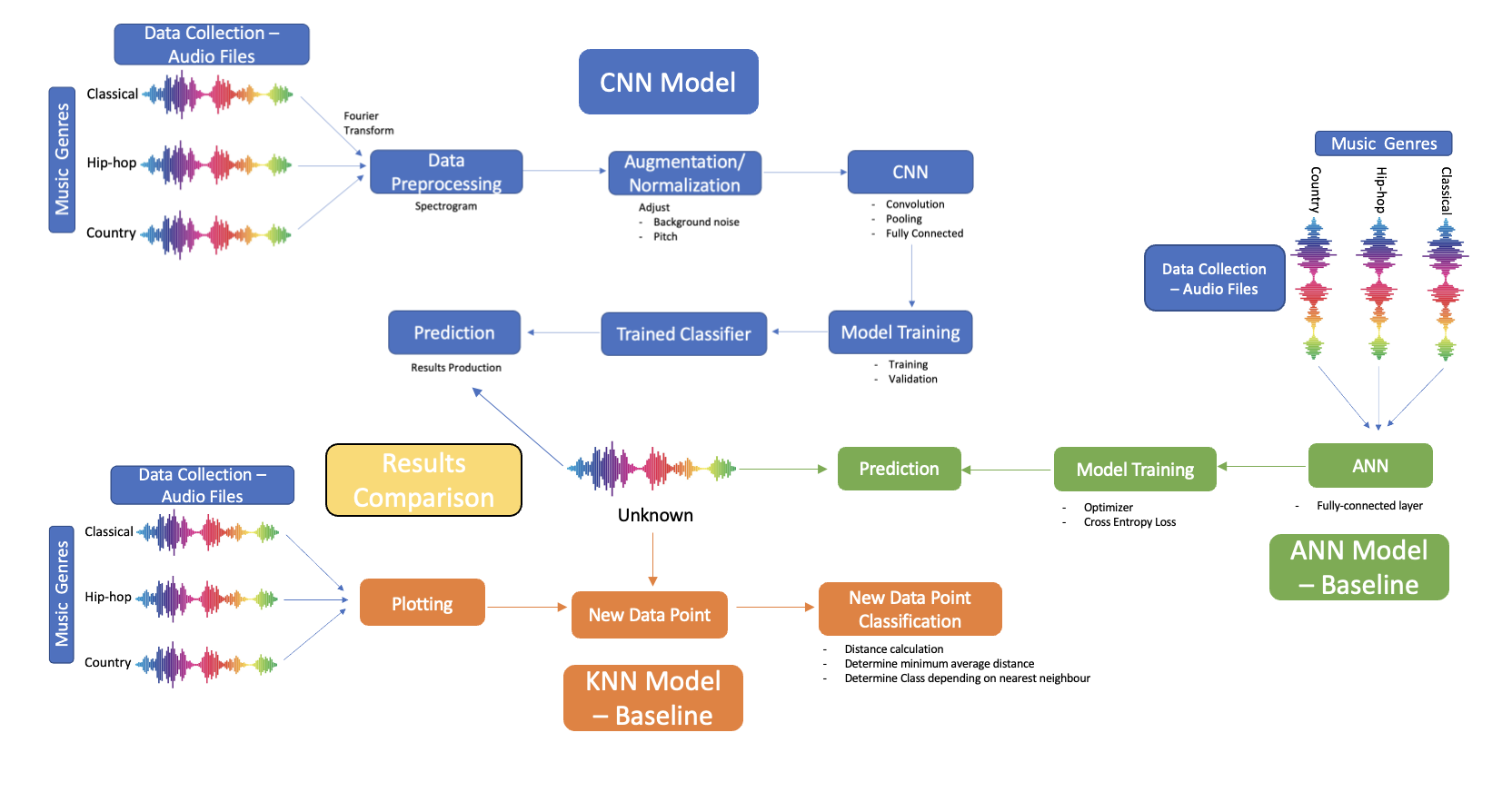


Figure X

1. **Individual Contributions and Responsibilities**

The complexity of this project requires careful consideration of the ways in which the members of the team will be expected to behave and contribute. The following sections detail the ways in which the team manages working together and meeting project deadlines.

* 1. **Collaboration and Expectations**

The team carries out meetings virtually on Mondays at 6:00pm over Zoom. Not only does this accommodate for the diverse schedules of the team members but it allows for screen sharing during discussion as a tool to aid in discussion. Outside of meetings, the team communicates over a WhatsApp group chat.

Expectations of the team are such that deliverables will be completed at least 24 hours before their deadlines, and if a team member is having difficulty with their assigned work, they are to contact other members of the team as soon as possible. Another important idea enforced within the group is to have an open mind to all new ideas and provide constructive feedback in a respectful manner.

All work for the project has been completed within a shared Google Drive folder, with the project code written in two Google Colab files. The first file contains the data processing and baseline model code, while the second contains the CNN model developed by the team.

Data Processing and Baseline:

<https://colab.research.google.com/drive/1xxQC1-cxJjhxTCY4vswfR6Sl0HYSy1Fe?usp=sharing>

CNN Model:

<https://colab.research.google.com/drive/1KZwpzHQe0cCiP3A2M5oU2SWH_dAbhUL5?usp=sharing>

The advantage of completing the project in Google Colab is that it allows for multiple team members to be working on the project at once, while also keeping track of when a team member has made changes, and the extent of their contributions to the project.

* 1. **Deadlines and Task Distribution**

In order for the project to be completed in a timely manner, the team created a list of critical deliverables and their deadlines. This list of deliverables, internal deadlines, and the time at which they were ultimately completed, can be seen in table 1.

Table 1. Project Deliverables, Deadlines, and Completion Dates

| Tasks | Internal Deadlines | Date Completed | Descriptions |
| --- | --- | --- | --- |
| Data Collection | Feb 27th | Mar 8th | Data with different categories of labeled audio are collected in the .wav format. |
| Data Processing | Feb 27th | Mar 8th | Audio files are transformed into spectrograms using fourier transform theory. |
| KNN Baseline Model | Mar 11th | Mar 12th | The KNN baseline model is complete. |
| KNN Baseline Results | Mar 12th | Mar 12th | Sample data is fed to KNN baseline model and at least 2 results are produced. |
| ANN Baseline Model | Mar 13th | Mar 14th | ANN baseline model and relevant architecture required for training is complete |
| ANN Baseline Model Results | Mar 14th | Mar 16th | Sample data is fed to ANN baseline model and at least 2 results are produced. |
| CNN - Feature Extraction | Mar 5th | Mar 12th | CNN model has a framework with functioning feature extraction tools. |
| CNN - Filter Generation | Mar 11th | Mar 13th | CNN model has the filter generation components including weights, kernel size, layers (convolutions/ pooling), epochs, batch size, and learning rate integrated. |
| CNN Model Results | Mar 12th | Mar 16th | Sample data is fed to the CNN model and at least 2 results are produced. |
| Progress Report | Mar 16th | Mar 17th | All the respective sections for progress report are completed including discussions related to data collection, data processing, KNN baseline model, CNN model, results comparison, next steps, etc. Additional sections can be referred to the progress report guideline. |
| CNN Model Training | Mar 27th | In Progress | CNN model is adjusted to optimize hyperparameters by training with testing and validation data set. |
| CNN Model Testing | Mar 31st | In Progress | The accuracy of our CNN model is evaluated against a testing data set. |
| Final Presentation - Slides | April 4 th | In Progress | All the respective sections for the final presentation are completed and put on the slides. Full instructions and required sections can be seen in the final presentation guidelines. |
| Final Presentation - Rehearsal | April 5th | In Progress | The team completes the rehearsal for the project and makes any required final edits. |
| Final Report | April 12th | In Progress | All the respective sections for the final report are completed. Full instructions and section requirements can be seen from the final report guidelines. |

In addition to the list of deliverables, the team has recorded the assigned tasks and past contributions of each team member at each stage of the project. This can be seen in table 2.

Table 2. Task Distribution List

| Team Members | Tasks | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Project Proposal | | Project Progress Report | | Project Presentation & Final Report | |
| Alex | * Data Processing * Ethical Considerations * Research | | * Collect the data * Process data to spectrograms * Report Discussion with data collection and spectrograms | | * Produce all data results * Adjust hyperparameters for accuracy * Test the accuracy of the CNN model * Report writing * Presentation preparation | |
| Anna | * Background & Related Work * Architecture * Baseline Model * Edit Document Coherence * Research | | * Produced KNN baseline model * Produced ANN baseline model * Produced at least 2 results from each baseline model * Report Discussion for the baseline models * Individual Contributions and Responsibilities list * Abstract * Project Description * Conclusion * References * Edit document Coherence | | * Produce all data results * Adjust hyperparameters for accuracy * Test the accuracy of the baseline model * Report writing * Presentation preparation | |
| Blythe | * Illustration   /Figure   * Project Plan * Baseline Model * Edit Document Coherence * Research | | * Producing the CNN model * Produce at least 2 results from the CNN model * Report Discussion with the CNN model * Abstract * Wrote training and accuracy code | | * Produce all data results * Adjust hyperparameters for accuracy * Test the accuracy of the CNN model * Report writing * Presentation preparation | |
| Tyler | * Introduction * Risk Register * Abstract * Research | | * Producing the CNN model * Produce at least 2 results from the CNN model * Report Discussion with the baseline model * Optimized parameters for better results | | * Produce all data results * Adjust hyperparameters for accuracy * Test the accuracy of the CNN model * Report writing * Presentation preparation | |

At this stage in the project, each team member has been assigned a significant subset of work. Alex was responsible for data collection and processing, transforming gathered .wav files into spectrograms to be read into the models. Anna was responsible for building and testing the final chosen baseline models in which the primary model will be compared to. Lastly Blythe and Tyler were responsible for developing the primary model. Specifically, Tyler constructed the CNN model (layers, activation functions, etc.), while Blythe created the framework with which the model would be trained and tested.

* 1. **Redundancies**

There are several ways in which the team has prepared to handle unanticipated situations and risks within this project. Firstly, the team has ensured that no work will be unfinished through the use of consistent check-ins on team progress, breaking down major deliverables into smaller subproblems with numerous internal deadlines, and keeping in constant contact with the teaching team through lectures, practicals, and on Piazza to ensure there are no misunderstandings regarding project concepts and expectations within the team.

In the event of a canceled meeting, a When2Meet poll was distributed to find contingency meeting times, which were determined to be Thursdays between 3:00pm to 4:00pm, and Saturdays between 11:00am to 4:00pm.

Furthermore, if a team member is experiencing difficulty, the team has encouraged all members to seek assistance right away, and the team member with the least amount of work in that time frame will begin working on the section of the project experiencing delays.

1. **Notable Contribution**

The following section details the team’s progress on the completion of this project. This includes the steps taken to collect and process data for analysis, the development of the primary CNN model, as well as the baseline models by which the primary model’s performance will be assessed against.

* 1. **Data Processing**

The data that was used to train this model is the GTZAN dataset []. We were able to retrieve this dataset from Kaggle, an online community of data scientists and machine learning practitioners []. We began by checking if the dataset met our standards, which was being large enough and having enough classes to be able to differentiate the input properly. The GTZAN data meets this because it consists of 10 classes/genres of music each represented by 100 tracks. So in total, we are working with a dataset of about 1,000 elements. The genres are: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae and rock.

Next, the data had to be retrieved from Kaggle. This involved one of the team members making an account and downloading the 1.08gb worth of data on their local machine, then making a kaggle account and uploading the dataset to their account in order for us to generate an API key.

The steps involved in the data processing are as follows:

* Step 1: Data Loading
* Loading all the required modules such as pytorch, numpy, matplotlib, etc.
* Using the ‘!pip install -q kaggle’ command line command to install Kaggle to the google colab environment
* Downloading the kaggle.json file from the dataset page of your account on Kaggle
* Using the command ‘from google.colab import files

files.upload()’

to upload the kaggle.json file into the colab environment

* Running the following commands to make directories and obtain permissions from the kaggle API to access the datasets
  + ‘! mkdir ~/.kaggle’
  + ‘! cp kaggle.json ~/.kaggle/’
  + ‘! chmod 600 ~/.kaggle/kaggle.json’
* Downloading the the datasets from the profile created by the team member
  + ‘!kaggle datasets download -d oluwasina23/gtzan-data’
* Unzipping the data and removing the zip file after unzipping
  + ‘!unzip gtzan-data.zip’
  + ‘! rm gtzan-data.zip**’**
* Step 2: Data conversion

At this point, all the data that was loaded in the previous section are all in .wav file formats, which are audio files. This means that a function is needed in order to convert the files from audio to images. This was done by using spectrograms [10]. Spectrograms are visual representations of the spectrum of frequencies of a signal as it varies with time [10]. This involved applying a Fourier transform to the raw audio data, then dividing it into small time frames and applying a windowing function to each frame[11]. The main library used in this process was the librosa library. The librosa library is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. The function was designed to take two inputs; a .wav audio file and a destination path for the resulting spectrogram. The function makes use of the ‘librosa.stft()’ method which creates a spectrogram of the .wav file which can be seen below.

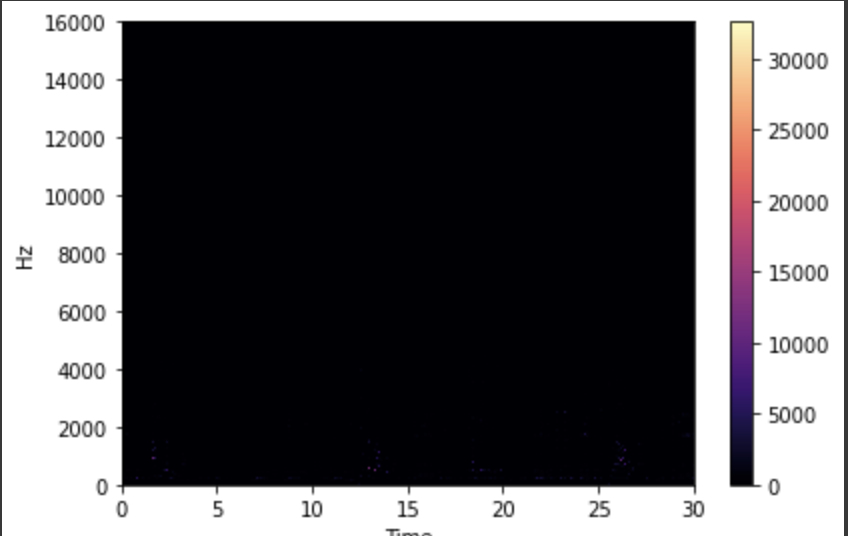


Figure 1. (Spectrogram Image)

The figure seen above does not hold much detail, it is mostly dark. This would be problematic for the model because there are no features for it to learn from. The next step was to create a Mel Spectrogram using the ‘librosa.feature.melspectrogram()’ which generated the figure below.

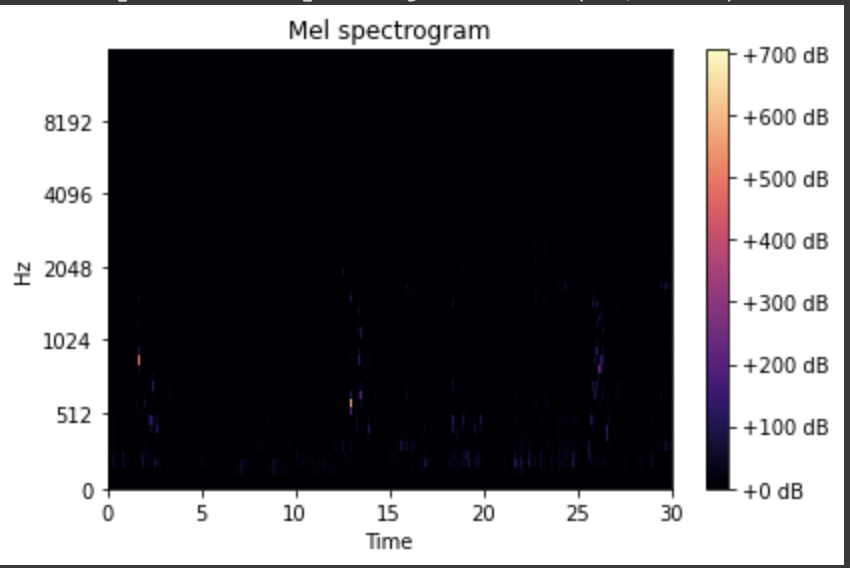


Figure 2. (Mel Spectrogram Image)

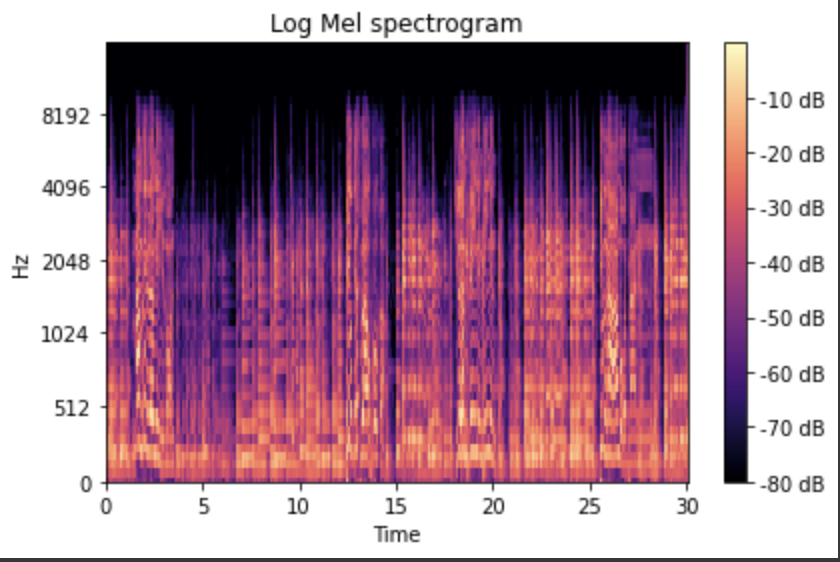
The Mel Spectrogram images are better than the regular Spectrogram images, they have more details but still are mostly black. The next step is to convert them to Log Mel Spectrograms using the ‘librosa.power\_to\_db()’ method. See result below.

Figure 3. (Log Mel Spectrogram Image)

These images have much more features and details that will make it possible for the model to learn adequately.

In terms of challenges that were faced during the processing of the data, converting the .wav files to spectrograms takes a lot of time, and would sometimes throw errors that we had never seen before, like a “NoBackEndError”. In order to combat this, we made modifications to the function to point out which file threw the error, so I could inspect it manually. We found that there was a problem with the 54th .wav file in the Jazz class. I removed this file from the jazz class, redownloaded it and reinserted it. This did not necessarily resolve the issue but threw a corrupted file error on teh same file. Then I decided to just create an exception for all errors in teh function where if the selected file threw an error, the function would skip it and continue onto the next file. This can be seen in the following code block:

E\_log = []

for i in classes:

count = 0

for j in os.listdir(Mpath +'/'+i):

print(j)

try:

convert\_to\_spectrogram(Mpath+'/'+i+'/'+j, '/content/Processed\_Data/Spectrograms'+'/'+i+'/'+str(count))

except Exception as e:

E\_log.append(e)

continue

count+=1

However, we did not just allow the errors to go scott free, we appended all of the errors into the list E\_log and checked it after the running the block. The reason for doing this is because if there are a large number of errors in the data, then we would need to look for a different dataset. Accounting for each and every error type would be very inefficient. Luckily, after checking the size of the error list, it only returned 1, which is something we can live with.

* Step 3: Data partitioning into training, validation and testing data

Once the data had been converted and saved into their adequate folders, the next step was to partition the data. This was done using a nested for loop, ‘os.listdir()’ and ‘shutil.copy()’ methods. The code block can be seen below.

count = 0

for i in classes:

for j in os.listdir(specPath+'/'+i):

if count <= len(os.listdir(specPath+'/'+i))\*0.6: #60% of the data goes to training

shutil.copy(specPath+'/'+i+'/'+j, train\_set+'/'+i+'/')

elif (count >= len(os.listdir(specPath+'/'+i))\*0.6

and count <= len(os.listdir(specPath+'/'+i))\*0.8): #20% goes to testing

shutil.copy(specPath+'/'+i+'/'+j, test\_set+'/'+i+'/')

else:

shutil.copy(specPath+'/'+i+'/'+j, val\_set+'/'+i+'/')#20% goes to validation

count += 1

count = 0

The code block seen splits the data into 60% for training and 20% for testing and validation. We also decided to save the split data in our team google drive to avoid having to spend time converting the data to spectrograms everytime we needed to test the model.

* Step 4: Converting into tensors

This step was done first by retrieving the resulting image size using the ‘cv2.imread.shape()’ method. This resulted in a dimension of 418x627x3. The 3 corresponds to the RGB channels. Using this, We set the transforms to a size of 418 by 627 and set the type to Tensor. The data was then extracted using the ImageFolder method and shuffled using the DataLoader method.

The result of the data processing was that we generated spectrograms for 10 genres of music. These spectrograms could then be fed into the CNN and turns our problem into an image classification problem. The spectrograms for each class can be visualized as follows:

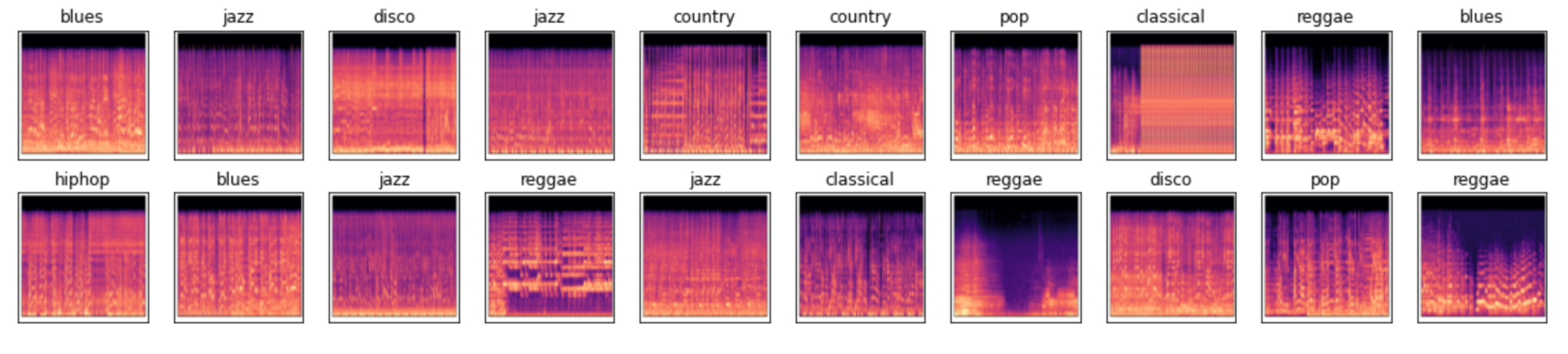


Figure 4: the spectrograms from respective classes

This split is intended to only have the 60% used for training, which is what the model is going to be frequently exposed to and used to learn the problem. The 20% validation is used as a part of training but is not exposed to the model nearly as often as the 60% training set. The 20% test set is rarely exposed to the model, we will only expose this data to the model at most three times. This is so we can keep the models real world performance accurate. Exposing test data to the model would result in the model learning the data and producing a result that is overfit.

The goal is to be able to have the model classify external input. For instance, we should be able to take a song or a playlist from spotify, convert them to .wav files, feed it into the model and get accurate predictions. Or even better, we will form a band just for the purpose of creating data to test the model with and feed the music we record into the model and see how it performs.

* 1. **Baseline Model**

In order to appropriately assess the performance of our CNN model, the team will be developing a baseline by which the CNN accuracy will be compared to. The primary model performance should be better than the baseline.

The chosen baseline model for this project is a K-Nearest Neighbour (KNN) model. KNN is a regression and classification model that takes a new data input and compares its characteristics to points within a labeled training set [A]. If the input has characteristics that align closely with a certain class, the input will be labeled as a member of that class [B]. The ‘K’ factor represents the number of ‘neighboring’ data points from the training set to take the labels from when deciding what class the input data belongs to [B].

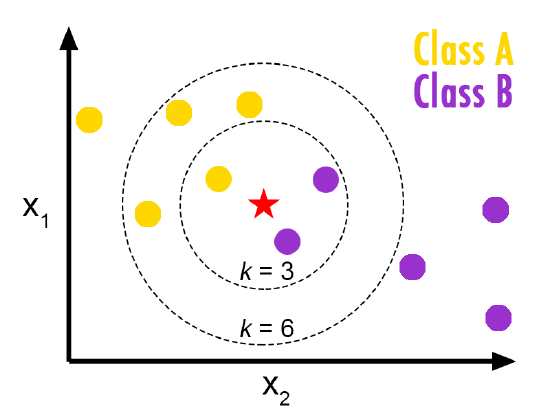


Figure 5: Simplified diagram depiction of the mechanism of KNN [S]

For this project, in order to calculate the distances appropriately, the spectrogram images were converted into flattened vectors. Each test vector was passed to a function that would calculate its distance from all of the data points from the training set. From there the distances of the three closest training points and their labels were saved and the most frequently appearing class was chosen as the predicted class. The distance between vectors was Euclidean distance, and was calculated through the use of the function *np.linalg.norm(Xt[i] - vec)*, where *Xt[i]* denotes a vector in the training set, and *vec* denotes the given test input vector being analyzed by the model. If all of the found K neighbors were different (no class with multiple points), the class of the point with the nearest distance from the input vector was chosen as the predicted class.

The design of the model was inspired by a model created for the purpose of music genre classification using MFCC files, and an image classification KNN model using the CIFAR-10 image dataset [C][D]. The data used to assess the model’s performance was the entirety of the training and testing spectrogram dataset created as mentioned in the previous section. While choosing the K value for a model is not a simple task, a rule of thumb is to set it as the square root of the number of samples in the training set, rounded to an odd number [T]. As the training set has a total of 609 images, the K value will be set to 25.

After running the model, the baseline had an accuracy of 37.5%. This is extremely low. However, the performance was not consistent within all genres. As seen in the figure below, ‘metal’ had the highest accuracy rate (about 100%), while ‘reggae’ and ‘rock’ each had around 0% accuracy.

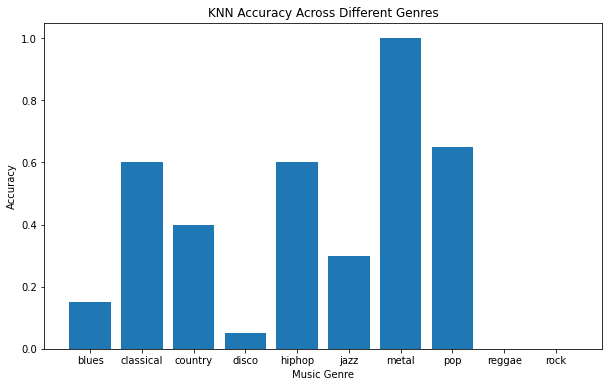


Figure 6: Graph of accuracy rate by music genre

The difference in accuracy implies that there are more clearly identifiable features shared by songs in the ‘metal’ genre, which make its area within the KNN set space more distinctive. As such, new ‘metal’ song inputs are more likely to fall within the same set space area and be categorized correctly. In contrast, reggae and rock have fewer features that distinguish them from other music genres.

While conceptually a KNN model is simple, determining the K value was difficult. While a small K value can lead to low accuracy due to a greater proportion of noise preventing clear boundaries between classes, a K value that is large is computationally expensive, leading to a longer time to process the data [T]. As such, using the rule of thumb as mentioned above enabled the team to choose a reasonable value for the baseline model to run with.

While the overall accuracy rate is better than if an individual were to guess the genre randomly (random guess accuracy is 10% as there are 10 categories), the team decided to create another baseline model in hopes of using it to better assess the performance of the primary model.

The secondary baseline model developed is an Artificial Neural Network (ANN). This model was built using inspiration from the University of Toronto’s APS360 labs and tutorials [].The ANN contains two layers with ReLU activation functions. The training loss calculation is Cross Entropy.

Following trial and error with several different hyperparameter values, the final ANN model specifications were chosen as:

* Batch Size: 32
* Learning Rate: 0.0005
* Epochs: 100

The figure below shows the loss and accuracy curves for both the training and validation test sets.

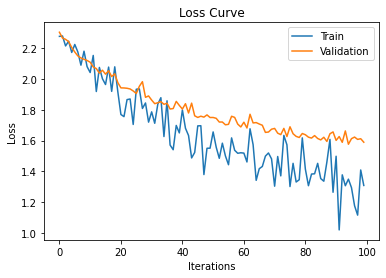


Figure 7: The Loss Curve for Train and Validation Data

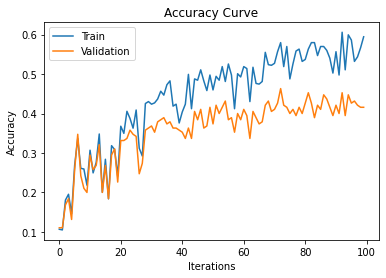


Figure 8: The Accuracy Curve for Train and Validation Data

The final accuracy for the ANN baseline model with the test set was 45%. A model trained after 100 epochs was chosen because the accuracy rate could be seen to plateau for the validation set and the loss curve was bottoming out, thus implying limited improvements could be made to the model performance. Furthermore, given the divergence of the training and validation accuracy, this is a sign that the model was beginning to overfit to the training set.

The challenge with this model is determining the appropriate hyperparameters such that the greatest model accuracy could be reached.

* 1. **Primary Model**

The CNN (convolutional neural networks) model was chosen as the primary model because CNN is the leading architecture used for image recognition and detection tasks. [] First, the team started to visualize the spectrogram from each class. Referring to figure X, different classes from the training samples were loaded randomly to display.

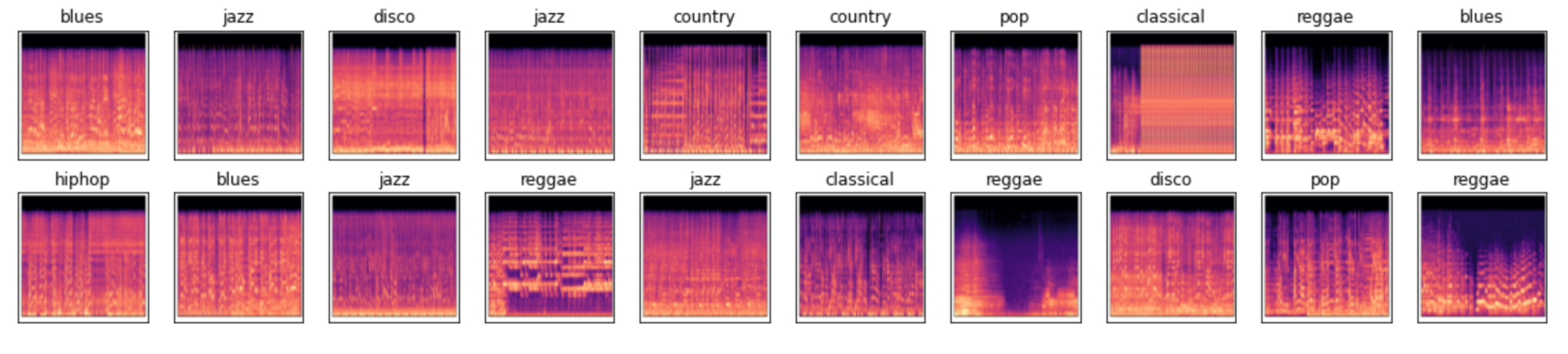


Figure 9: the spectrograms from respective classes

**Description of the model**

The CNN model is made of 4 convolutional layers, and 2 fully connected layers. Each convolutional layer applies a 2D convolution over an input of images. The convolutions apply a 3 by 3 filter to the images with a stride of 1, reducing their size and deriving information from the image. The weights of these filters are what the model learns from, and the out channels describe how many filters are applied to the image. The output image from this process can be calculated using the formulas [<https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html>]:

After each convolution, the layer continues by applying a ReLU activation function, and max pooling to the input. After each layer, a dropout is applied, where channels are zeroed out randomly with a probability of 0.1 to prevent overfitting and improve independence between features. The number of input channels in the first layer is 3 for the RGB values in the input spectrographs. The out channels increase from layers 1 to 3 where it reaches 128 features, and reduces again to 64 out channels.

The output of the last convolutional layer is put through a fully connected network made up of 2 linear layers, and a ReLU activation function after each one. The first linear layer maps the features from the last convolutional layer to a vector of size 512. The second linear layer then takes this output and maps it to a vector of size 10 since there are 10 classes (music genres). The highest value in the vector is the predicted class and its index corresponds to the predicted label.

A diagram of the neural network can be seen in Figure # below.

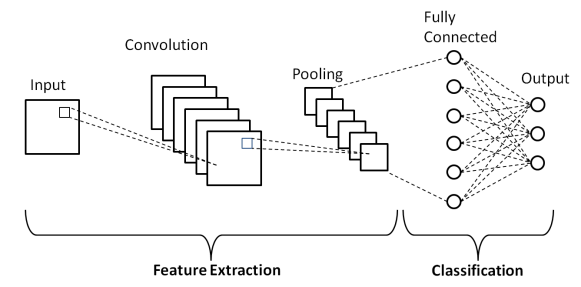


Figure 10: A diagram of a typical CNN model [<https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909> ].

The training accuracy and validation accuracy were evaluated with the training and validation datasets. The function first inputs the data through the model to obtain the predicted output. Then for each prediction, the function selects the index with the highest prediction score; in our case it is the most likely class. Then the function compares the predicted labels with the ground truth labels and counts the number of correct predictions. Last, the function increments the “total” variable by the number of images in the batch. Therefore, after evaluating all the batches, the function will return the accuracy of the entire model, which is calculated based on the number of correct image classification over the total number of images.

**Results**

The primary model was trained with GPU to speed up the training with parameters of 75 epochs, learning rate of 0.001, and a batch size of 128. The results for these parameters can be seen in the recorded training and validation accuracy in Figures # and # below. The best values in this set were at epoch 60, where the training accuracy reached 0.7521 and a validation accuracy of 0.4421. The training accuracy being relatively higher than the validation is an indication that the model is overfitting to the training set.

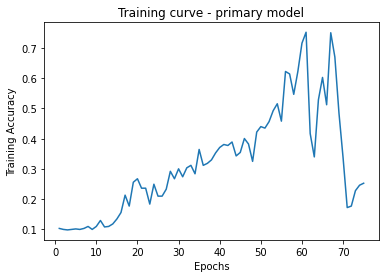
****

Figure #. The training accuracy curve for the primary CNN model

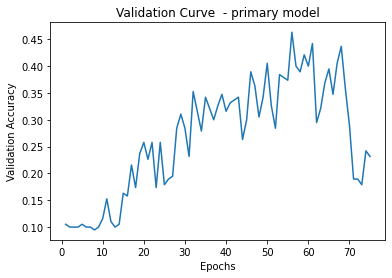
****

Figure #. The validation accuracy curve for the primary CNN model

**Improvements**

Improvements in accuracy and reduced overfitting can be solved with parameter optimization. The number of parameters in the CNN model is 26; these can be adjusted for better performance. For example, increasing the dropout rate could help with overfitting, optimizing the number of layers could be increased along with the total features (filter sizes, etc) to add complexity, and normalize batch to speed up the training and improve the generalization by normalizing the inputs of each layer. In addition, data augmentation such as random cropping, rotations, or scaling could be applied to the spectrographs. The team also plans to implement transfer learning to either a less complex CNN or RNN model. These adjustments will be made for the final project to add further complexity and make a stronger model than the baseline model. At the end, the team will evaluate the test dataset once as it will be a complete unseen data for our model and will be fair to assess the performance of the model.

1. **Conclusion**

To conclude, the need for music genre classification stems from the potential it has to aid music service providers in organizing their collection and providing their users with customized recommendations. The capabilities of deep learning models to handle large amounts of data and perform automated classification make it an excellent candidate for the facilitation of music genre classification. Due to its strength in image classification, the team chose to create a CNN model to classify music. Since the inception of the project, the team has demonstrated significant progress in developing a deep learning model to identify the genre of a given piece of music. This includes gathering audio data and transforming it into spectrogram format, developing two baseline models, and a preliminary CNN model. The next steps of this project include refining the primary model in order to obtain the highest possible classification accuracy, and test its performance on never-before-seen data.

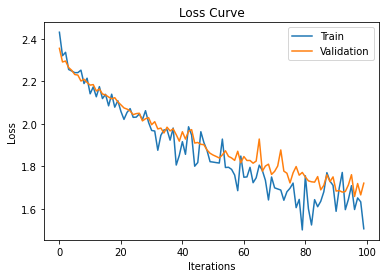
1. **References**

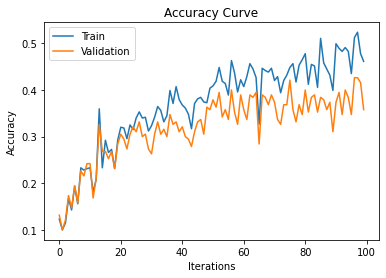
# REPORT INSTRUCTIONS

1. **Brief Project Description**: In your own words provide a brief description of the motivations behind your project, the goal of your project, why it is interesting or important, and why deep learning is a reasonable approach.
2. **Individual Contributions and Responsibilities**: Provide a summary of how your team is working together. Describe the project management software that you are using to communicate with each other, track progress and results, track updates to shared code, etc. You should provide details on what each person is responsible for and has accomplished so far, along with an updated list of tasks and deadlines for each member.
3. **Notable Contribution**: Provide a detailed summary with results (figures and diagrams) of the following:
   1. **Data Processing**: You can describe the data that you have collected and cleaned. Be clear and specific when describing what you have done, to the point that someone could reproduce your work. If possible, show some statistics about your cleaned data (e.g., number of examples in each class), and new data for final testing of the model. We prefer you collect some **new data** yourself or if that is not possible, use an entirely different dataset which was not used anywhere before.
   2. **Baseline Model**: You can provide a diagram to describe the baseline model that you have tested and how it was/will be compared with our primary neural network model. The baseline model can be a simple machine learning model (e.g., SVM, Random Forests, Models from labs and tutorials, etc.), a hand-coded heuristic model (that does not use machine learning), or something else. **A model that performs randomly is not acceptable**. The expectations for the baseline model will vary from project to project. You should also provide at least one quantitative and qualitative result. These results could be learning curves, or results showing the performance of the model in selected samples of data. The focus here is on assessing the feasibility of your model to achieve the project objectives.

The difference in accuracy implies that there are more clear identifiable features shared by songs in the ‘metal’ genre, which make it’s area within the set space more distinctive. While the overall accuracy rate is better than if an individual were to guess the genre randomly (random guess accuracy is 10% as there are 10 categories), the team decided to create another baseline model in hopes of using it to better assess the performance of the primary model.

The secondary baseline model developed is an Artificial Neural Network (ANN). The ANN contains two layers with ReLU activation functions. The training loss calculation is Cross Entropy. The model was trained using a learning rate of 0.001 and 100 epochs. The figure below shows the loss and accuracy curves for both the training and validation test sets.





The final accuracy for the ANN baseline model with the test set was 39%. A model trained after 100 epochs was chosen because the accuracy rate could be seen to plateau for the validation set, thus implying no further improvements could be made to the accuracy.

