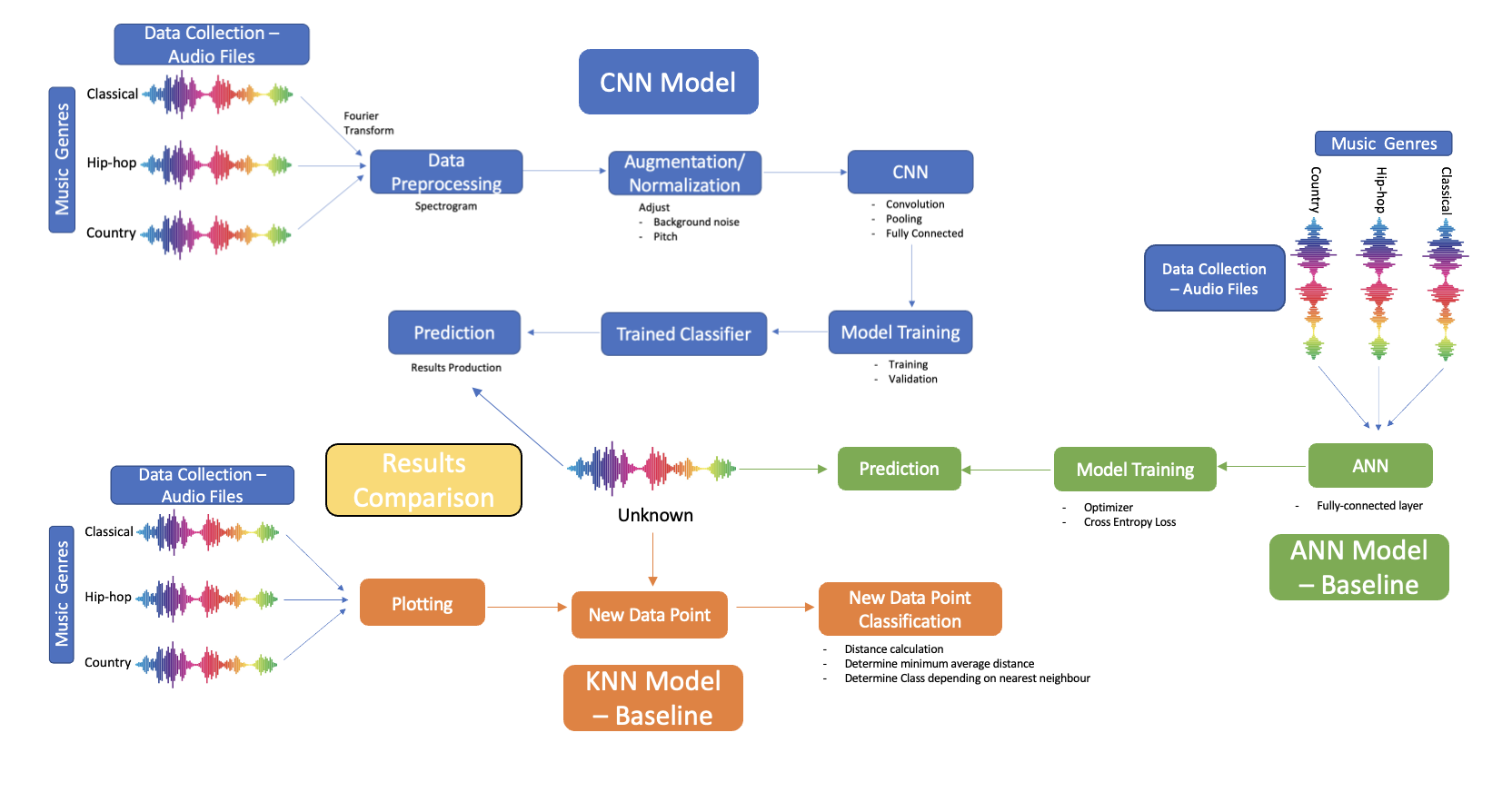
**ABSTRACT**

Given the increasing number of songs uploaded to the internet, streaming and other music organization services would benefit from automated music classification to assist in database organization and improve recommendations to users. In this project, the team developed a Residual Neural Network model to perform music genre classification. To compare to the model’s performance, a K-Nearest Neighbour and Artificial Neural Network model were also developed. These models were tested using a selection of 10,000 songs from Spotify that were transformed into log mel spectrograms for improved processing. The final model obtained a test accuracy of 69.4% and was most effective in classifying classical and jazz music. Due to overfitting and limited model learning, the model requires additional modifications to improve performance to a standard acceptable for industrial use.

**Introduction**

With the increased popularity of music streaming apps and new artists, the number of published songs has been increasing significantly [1]. Companies such as Spotify or Apple Music would benefit from automated, and accurate music genre classification to organize and recommend their libraries better for their users [2]. The goal of this project is to create a neural network that accurately classifies music genres from audio content. The milestones in this project include collecting data, building and training a residual neural network (ResNet), optimizing its parameters, and reporting on its accuracy. The team’s interest in the topic stems from all team members having some connection to music such as through learning to play different instruments or through frequent listening. Deep learning can improve user experiences through recommendations and organization of music, in addition to being faster and able to handle large amounts of data [3]. While artists may classify their own music as belonging to a specific genre, a deep learning model can provide other genres a song may apply to.

**Illustration**



**Background and Related Work**

Music genre recognition is a significant subset of the research field related to Music Information Retrieval (MIR) [4]. While most are familiar with music identification applications like *Shazam*, which can recognize songs from short samples, genre recognition is a more complex and subjective process [5]. Music genre classification is also a form of music label classification, with other forms being writer, emotion, and region classification, among others [6]. Some potential applications of music genre recognition include music organization and storage management within databases, internet search engine queries, recommendation systems within streaming services, and tools for music inspiration and creation [6].

Since the 1990s, genre recognition has been studied by many data and computer scientists [6]. This work is predominantly conducted using MP3 and WAV audio files [6].

In 2006, researchers in Singapore developed the first application of a “deep confidence network” to music genre classification [6]. This was done by using a multilayer classification system with Support Vector Machine (SVM) learning [6, 7]. One of the main methods of feature selection used was through using beat spectrum analysis, which automatically classifies rhythm and tempo [7]. To develop predictions, the research team utilized the theory that a subsection of a piece of music can be approximated by a linear combination of other related music samples [7].

In 2011, another group of researchers developed feature vectors for their genre recognition machine learning model through the use of a two-dimensional feature map which accounted for pitch and rhythm [6]. The feature vector coefficients generated were then used as approximations for melody [6]. This method of identifying feature vectors led to an 81% accuracy rate with the model [6].

Related work #3

In 2019, a team of engineers from India developed a Residual Neural Network (RNN or ResNet) to classify the genre of a song [A]. The 18-layer model developed by the team was given 3-second audio clips from songs within the GZTAN dataset to train and test with and had an accuracy rate of 82 – 94.5% [A].

Related work #4

In 2018, a student at the University of Waterloo trained a VGG-16 convolutional neural network (CNN) transfer learning model to identify the genres of over 40,000 10-second audio clips of music from the *Audio Set* database [B]. In order to improve the Signal-to-Noise ratio of the song clips, they applied a pre-emphasis filter during data-processing [B]. The model obtained 89.1% accuracy [B].

Related work #5

In 2002, a group of researchers explored how music genres could be defined as a combination of common characteristics, such as instrument type, rhythm, and harmonies [C]. In their work, the team used a statistical pattern recognition software to classify songs based on timbral texture, rhythm, and pitch [C]. The final model had an accuracy of 61% across 10 genres [C].

**Data Processing**

Initially, this model was trained using the GTZAN dataset [8]. The data was retrieved from Kaggle, an online community of data scientists and machine learning practitioners. The dataset consisted of 10 classes/genres of music each represented by 100 tracks. In total, this is a dataset of about 1,000 elements. The genres are: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae and rock.

However, as the project progressed the model was overfitting the data and was not giving accurate predictions. This was attributed to the dataset not being large enough. This was mitigated by writing a script to compose a larger dataset. The script communicated with the Spotify API to extract information such as the genre of the song and its URL, which was entered into a function from the ‘SPotDL’ library in python that downloaded the music to a local environment [9]. A description of the script is seen below:

1. **Data Collection Script**

The steps used in data collection and data loading are as follows:

* + Loading all the required modules such as pytorch, numpy, matplotlib, etc.
  + Going to ‘<https://developer.spotify.com/>’ and making an account to generate the API credentials
* Using the ‘!pip install spotipy’, ‘!pip install spotdl’, ‘!pip install pydub’ and ‘!pip install ffmpeg’ command line command to install the spotify library that helps manage Spotify API credentials and download the files
* Manually going through spotify to find playlists of a specific genre and getting their URL’s
* Writing a function called ‘call\_playlist’ that goes through the songs in the playlist and retrieves their information from the Spotify API
* Implementing the ‘call\_playlist’ function with a for loop that once verified, changed the directory that of the song using the ‘os.chdir()’ method
* Once the directory of the song had been determined, the url of the song would be saved as a variable and the command ‘!spotdl download $url’ would be executed in order to download the song to the genre directory

In total, about 10,000 songs were collected with at least 1000 songs per genre/class (the same 10 classes as GTZAN). Once the data was collated, the downloaded songs were in .mp3 format which was not suitable input for our model. The data needed to be preprocessed.

1. **Data Preprocessing**

The preprocessing steps are outlined as follows:

* Step 1: Data Loading

The songs had already been moved into their respective genre directories as full length .mp3 files. The following was required:

* Looping through each directory using a for loop and ‘os.listdir()’ method
* Assigning each song in the directory as a temporary variable and loading the song using the ‘x=pydub.AudioSegment(song)’ method[10]
* Splitting the audio like an array into 30 sec segments i.e. ‘x[30000, 60000]’ pydub splits audio in order of milliseconds, so 30 sec would be 30000 millisec.
* Using the ‘x=pydub.export(destination, .wav format)’ method to export the split audio file into its destination directory in a waveform file format.
* Step 2: Data conversion

All the data that was loaded in the previous section are in .wav file formats. A function is needed in order to convert the files from audio to images. This was done by using spectrograms. Spectrograms are visual representations of the spectrum of frequencies of a signal as it varies with time [11]. This was done using a Fourier transform to the raw audio data, then dividing it into small time frames and applying a windowing function to each frame. 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.

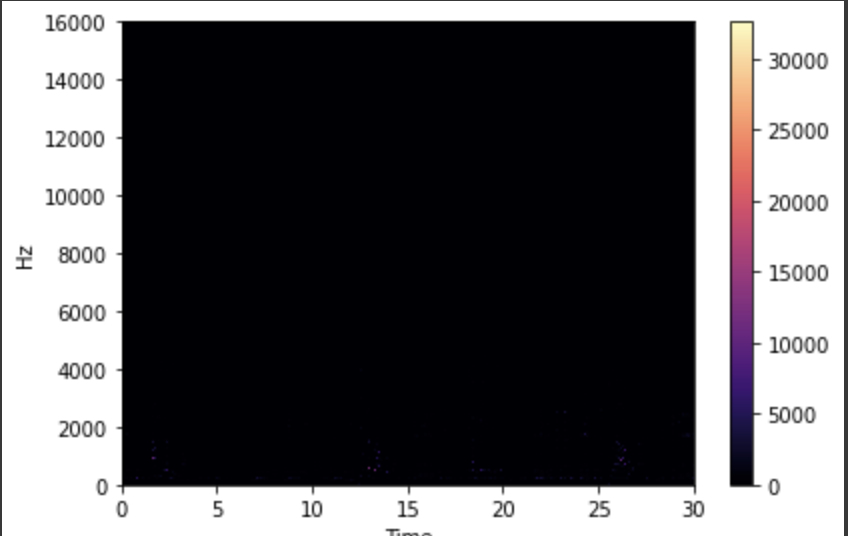


Figure #. (Spectrogram Image)

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.

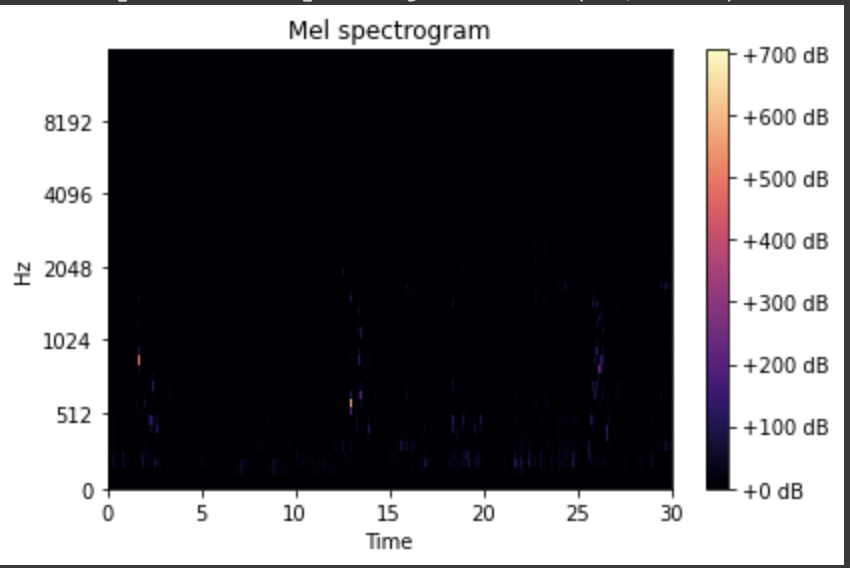


Figure #. (Mel Spectrogram Image)

The Mel Spectrogram images contain 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.

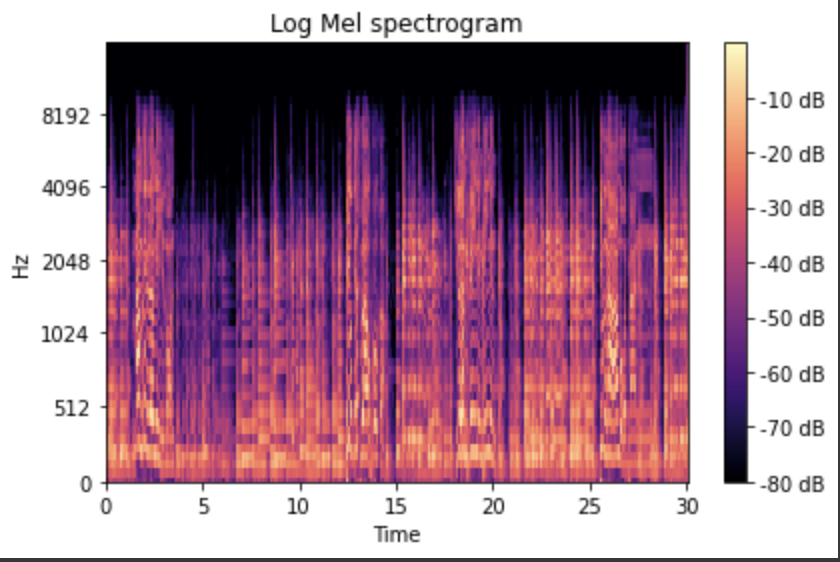


Figure #. (Log Mel Spectrogram Image)

These images contain more features that will make it possible for the model to learn adequately.

* 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.This splits the data into 60% for training and 20% for testing and validation. The split data was saved on our team google drive to avoid having to spend time converting the data to spectrograms everytime the model needed to be trained.

* 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, the transforms were set 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 spectrograms were generated 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:

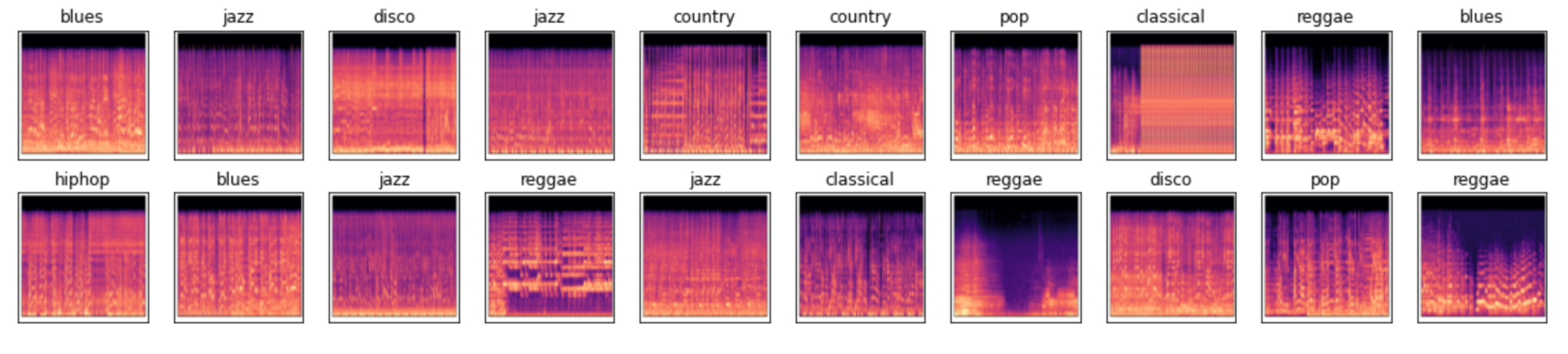


Figure #: the spectrograms from respective classes

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, this split would be exposed to the model at most three times. This is so the model produces an accurate prediction. Exposing test data to the model would result in the model learning the data and producing a result that is overfit.

1. **New Data**

In order to obtain a more accurate picture of the model’s performance, new data not found on Spotify was collected. The goal of this test was to ascertain the model’s ability to identify music genre regardless of the song production and composition. To achieve this, a selection of songs were obtained from YouTube. These songs were covers (e.g., Post Malone’s cover of Brad Paisley’s *I’m Gonna Miss Her*), songs performed in different styles (e.g., 1920’s Jazz style cover of *Levitating* by Dua Lipa), and other songs not released to streaming services. A total of 20 songs were collected with 2 songs in each genre (same genres as GZTAN). The songs were downloaded using a YouTube to .wav converter, then processed using the same means as identified earlier.

**Architecture**

ResNets are convolutional neural network (CNN) models that are designed to allow very deep and complicated networks without being impacted by vanishing gradients [M]. This design allows the model to perform better on image classification than CNN models with fewer than 18 layers [M].

Tyler: The team’s primary model was inspired by the 18-layer ResNet Model developed by Microsoft Research [N]. To begin, the model has a convolutional layer, taking in an rgb image tensor of size 224 x 294 , and applying 32, 7 x 7 sized kernels with strides of 2. This is followed by batch normalization, ReLU, and max pooling with kernel size 7 x 7 and a stride of 1. The ResNet models mostly consist of ‘blocks’, consisting of 2 convolutional layers, batch normalization, followed by a ReLU activation function. After each block, the model takes a downsample from the previous block, and adds it to the output. The detailed parameters for the blocks can be seen in Also known as a skip connection, this downsample prevents gradient loss by allowing gradients to propagate to deeper layers of the model without being reduced to near 0 values. The ResNet-18 model has 2 blocks per network layer, with a total of 4 layers made of blocks. These blocks have feature outputs 32, 64, 128, 256, respectively. Further details on the blocks can be seen in Table # below. Lower feature outputs were chosen to reduce training times, since the team previously tested pretrained 101-layer models, which performed similarly to the 18 layer model. The additional fully connected layer also helps prevent some of the more detailed features from being lost when reducing the 1024 output size of the last convolutional layer, to 10 classes. Pooling layers

|  | Convolutional layer details | | | |
| --- | --- | --- | --- | --- |
| Layer | Kernel size | Stride | Padding | Feature Output |
| 1 | 3 x 3 | Conv Layer 1: 1  Conv Layer 2: 1 | 1 | 32 |
| 2 | 3 x 3 | Conv Layer 1: 2  Conv Layer 2: 1 | 1 | 64 |
| 3 | 3 x 3 | Conv Layer 1: 2  Conv Layer 2: 1 | 1 | 128 |
| 4 | 3 x 3 | Conv Layer 1: 2  Conv Layer 2: 1 | 1 | 256 |

*Table #. Parameter details for each block in the modified ResNet model*

In total the model has 17 convolutional layers, as well as 2 fully connected layers, a modification of the original ResNet 18 architecture. Additional changes include reducing the number of features outputted by the model for layers 2-15, to be half of that in the original, per block.

The team’s primary model was inspired by the 18-layer ResNet Model developed by Microsoft Research [N]. The primary model contains 4 convolutional layers and 2 fully connected layers. The first convolutional layers contain 7 x 7 kernels with 64 out channels with a stride of 2. Then, 3 x 3 max pool with a stride 2 will be conducted to downsample the input image and maintain the most important features. The second convolution layers will repeat two times with two blocks of 3 x 3 kernels with 64 out channels, and between these two blocks, the skip connections were applied. Applying skip connections allows the model to continue learning new features while still recalling what has been learned before. Most importantly, skip connections address the issue with vanishing gradients during the training stage. The third and fourth convolution layers contain duplications of the 2 blocks 3 x3 kernels with 128 out channels and 2 blocks 3 x 3 kernels with 256 out channels respectively. Average pooling yields 1024 output features, which are then fed into 2 fully connected layers, then further be characterized into 10 music genre categories. Between the convolutional layers, a skip connection, downsample, and ReLU activation functions are applied every two blocks. Figure X provides a visual representation of our ResNet architecture model.

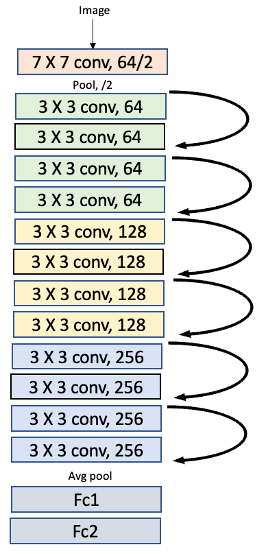
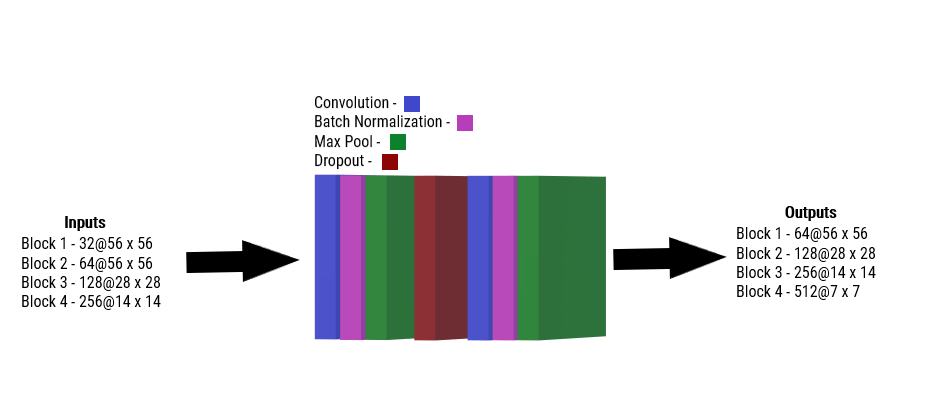
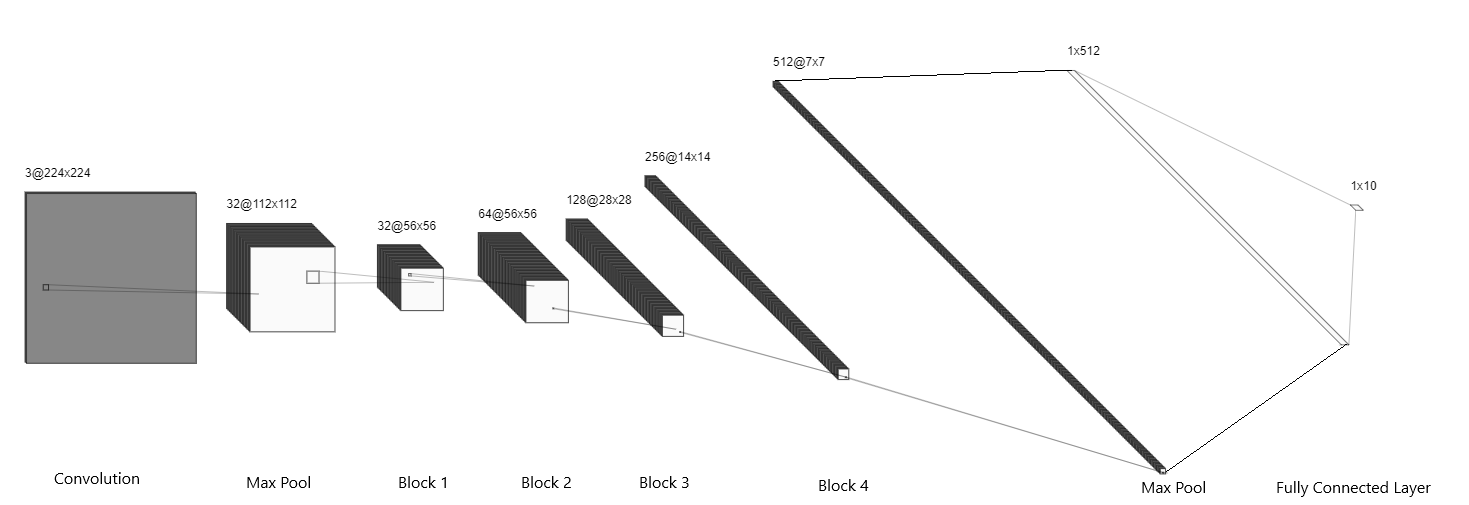


Figure X: the visual representation of the ResNet model

To customize the model, first, the team implemented dropout2d of value 0.5 every two blocks during the convolutional layer and another dropout of value 0.1 between the two fully connected layers. This technique prevents overfitting of the model and makes the ResNet model more robust by improving the generalization of the model. Second, the team implemented 2D batch normalization at the end of each block to improve the training speed, stability, convergence of the optimization process, and generalization performance of the model. Third, the team adjusted hyperparameters such as the number of epochs, learning rate, and batch size to fine tune and optimize the performance of the model. After trial and error, the team decided to use 50 epochs (optimal performance at epoch 17), a batch size of 32, and a maximum learning rate of 0.01, which is obtained through the use of an applied learning rate scheduler. The learning rate scheduler was used to improve the convergence and performance of the model. The following images summarize the model architecture.



*Figure #. A diagram of the convolutional blocks used in the ResNet18 architecture*



*Figure #. A diagram of the ResNet18 Model*

* Discuss final params:
  + Epoch: 50 (optimal at 17)​
  + Learning rate: Max 0.01, applied learning rate scheduler​
  + Batch Size: 32​
  + Dropout: 0.1 ​
  + Dropout2d: 0.1​
  + Feature output (into fc): 1024​
  + # of layers: 18​

**Baseline Model (4pts)**

In order to appropriately assess the performance of the ResNet model, the team developed baseline models by which the primary model can be compared. The primary model performance should be better than the baseline.

The first baseline model 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 [D]. If the input has characteristics that align closely with a certain class, the input will be labeled as a member of that class [E]. 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 [E].

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 [F][G]. 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 3020 images, the K value will be set to 55.

The secondary baseline model developed was an Artificial Neural Network (ANN). This model was built using inspiration from the University of Toronto course APS360's lab on ANNs [XXX].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 was set to have a batch size of 32, learning rate of 0.0005, and run with 100 epochs.

**Quantitative Results (4pts)**

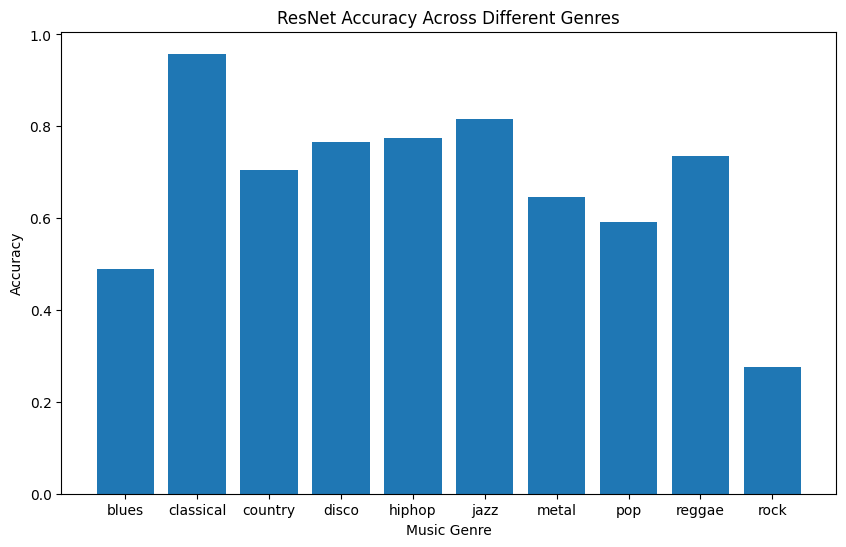
This section details the quantitative results of the assessment of the models developed.

In order to assess the performance of the models developed by the team, the model accuracy was observed. This accuracy was based on whether the top prediction matched the ground truth label. The following table details the results obtained from each model when using the training, validation, and testing datasets. Obtaining the accuracy from various datasets (training, validation, and testing) allows for the assessment of the models’ learning capabilities and whether overfitting occurs.

| Results | KNN (Baseline) | ANN (Baseline) | ResNet |
| --- | --- | --- | --- |
| Training Accuracy | N/A | 61.8% | 77.8% |
| Validation Accuracy | N/A | 45.8% | 70.5% |
| Testing Accuracy | 31.8% | 44% | 69.2% |

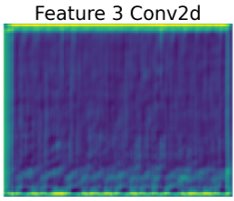
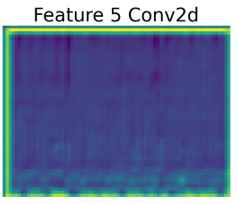
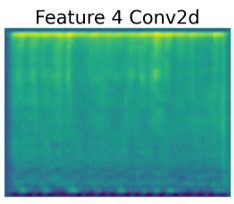
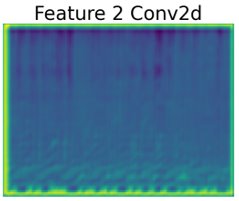
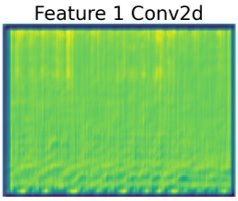
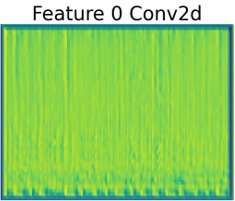
**Qualitative Results**

While testing the model performance, the team obtained the performance of the ResNet model by genre, which can be seen in the figure below.



It is evident that the model is more likely to classify classical, jazz and reggae songs correctly, while rock, blues, and pop are less frequently classified correctly. This can be seen in the prediction confidence level for different sample songs obtained by the team. For example, for classical song *Für Elise* by Ludwig Van Beethoven, the softmax prediction value for the Classical label was 0.8588. Comparatively, for *Enter Sandman* by Metallica, a metal song, the model predicted the genre to be Rock, with a softmax prediction value of 0.4. The prediction value for Metal was 0.0974. This indicates that songs in the classical, jazz, and reggae genres are more likely to have easily distinguishable features, thus increasing the confidence of the model’s prediction of songs in those categories.

Another song tested by the ResNet model was Creature by Pop Smoke. This song is hiphop/rap, however the model predicted the top three genres to be pop, rock, and blues. The feature maps generated for the song show how there are minimal to no distinct characteristics for the model to learn from, and as the network becomes deeper, the maps become even more blurred. The indiscriminate features could be limiting the model accuracy. The reason the model is not made obsolete despite the indistinct characteristics in the feature maps is that the intrinsic architecture of ResNet models ensures that basic characteristics and gradients are not lost as the network goes deeper.

* Model overfits. 30% gap between training and validation accuracy in later epochs, and 15% gap between these accuracies in optimal model
* Despite overfitting, testing results indicate the model learns up to 70% of the required features to classify spectrographs
* Features maps do not show any distinct characteristics but are more blurred as a consequence of going deeper in the network. ResNet ensures the basic characteristics and gradients are not lost.
* Include description of accuracy per genre
* 

**Evaluate model on new data**

* Take the top 3 predictions
* Covers (i.e. Metal versions of songs, regular covers)
* New compositions
* Acoustic/Instrumental solos
* Made efforts to ensure the collection of songs that would never have been seen by the model before
  + Covers, songs performed in different styles (i.e. 1920s Jazz version of Levitating by Dua Lipa), songs not released to streaming or Spotify in particular

Changes to Include:

* We got data from Spotify and checked that they were not in the original dataset list
* From YouTube we got covers, live performances, and genre swapped music (i.e. Dua Lipa’s *Levitating* as a Jazz song)
* The difference in performance could indicate that the quality of the audio (i.e., recorded vs. live) impacts the spectrogram readability; genre swapped songs pose a new challenge to the model

To evaluate the performance of the model in practical situations, it was tested against 2 additional songs from each genre, never before seen by the model. The uniqueness of the data was ensured by cross-referencing the original 10k Spotify dataset. The model classified these new songs at an accuracy of 54%; a 15% decrease in accuracy compared to the testing set. However, it outperforms the baseline models by 20-25%. Similarly to other music genre classification papers, the team conducted additional accuracies should the model have predicted the genre correctly within its top 3, and 2 predictions. The values for the top 3 and 2 predictions are 64% and 75%. This indicates with additional data such as instruments, longer time segments, and frequencies, the model has the potential to greatly improve its accuracy. Additional new data was collected from YouTube. The songs collected were covers, live performances, and genre swapped music (e.g., Dua Lipa’s *Levitating*  as a Jazz song). The accuracies of the primary model, as well as the baseline, KNN, and ANN models are compare in Tables # and # below for Spotify and YouTube data respectively. The Difference in performance could indicate that the quality of the audio (i.e., recorded vs. live) impacts the spectrogram readability. Furthermore, genre swapped songs may pose an additional challenge to the model.

| **New Spotify Data** | | | | |
| --- | --- | --- | --- | --- |
| **Within top \_\_ predictions** | **Primary Model Accuracy** | **ANN Accuracy** | **KNN Accuracy** | **Baseline Accuracy** |
| **1** | 54.25% | 30% | 35% |  |
| **2** | 64.75% | 45% | 45% |  |
| **3** | 75.5% | 70% | 60% |  |

| **Youtube Data** | | | | |
| --- | --- | --- | --- | --- |
| **Within top \_\_ predictions** | **Primary Model Accuracy** | **ANN Accuracy** | **KNN Accuracy** | **Baseline Accuracy** |
| **1** | 34% | 10% | 15% |  |
| **2** | 55% | 30% | 30% |  |
| **3** | 68.25% | 45% | 45% |  |

**Discussion (8pts)**

The performance of the model, while meeting the benchmark requirement of being better than the baseline, could use improvement.

To begin, the ResNet model developed is overfitting to the training data. This can be seen in the accuracy curve obtained during model training, showing a 30% accuracy difference between training and validation in later epochs, and about 5-10% difference at the optimal epoch (epoch 17). The overfitting can also be observed through the loss curve, where the loss of the training data decreases with the increasing epoch numbers. However, the validation loss does not follow the same trend, instead it starts to increase after epoch 17.

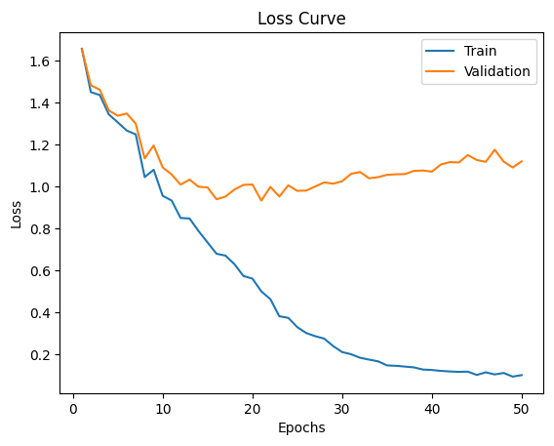
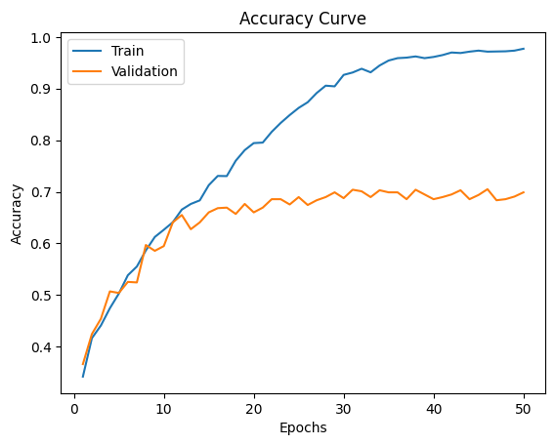


Figure X: a) the accuracy curve b) the loss curve for both the train and validation data with epoch

Significant efforts were made to optimize the model performance to its current state. First, when overfitting was observed, dropouts on both the convolutional and fully connected layers were included. Second, to increase the model’s learning speed, the batch size was increased. In order to enable the model to continue learning, the learning rate was increased and a learning rate scheduler was introduced. Last, data augmentation was also introduced with normalization to make the spectrograms more vibrant and can be seen in Figure Z.

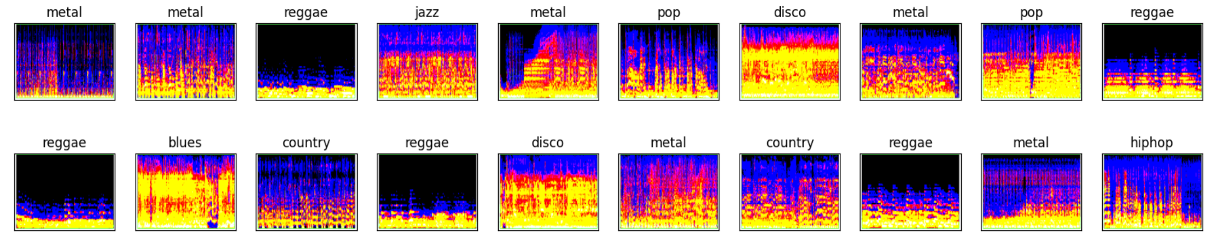


Figure Z: the spectrograms with normalization

Several of these attempted modifications yielded interesting results. For example, the increased dropout rate slowed the learning curve, however did not improve validation accuracy, only minimally improving the training accuracy. Also, changing the number of output features (doubling and halving), had no noticeable effect on the accuracy of the performance. However, applying normalization for the colour data augmentation increases the validation accuracy by 7%. Additionally, when testing different ResNet sizes, the deeper networks (34, 51, and 101 layers) had minimal to no changes to the model accuracy, instead only increasing training time and GPU usage. The 101 layer pretrained ResNet had the highest validation accuracy of 71%, however, it required significant time to train and training from scratch was infeasible.

The performance of the model was also affected by the way in which different genres appear in spectrogram format. Some genres lack distinguishing features that make them easily identifiable. These can be seen from the feature maps output at each convolutional layer referring to Figure XXXX. This suggests how the ResNet model and KNN model were unable to obtain consistent accuracy across all genres. Additionally, the characteristics of songs within a single genre can vary significantly. This adds to the difficulty of classifying songs accurately, as there are fewer common characteristics for the models to identify. However, the team is able to learn that colour data augmentation can help improving the accuracy of the model. Therefore, it is worth investigating different colour data augmentation strategies such as adjusting brightness, contrast, and saturation to further optmize the model.

Compared to other related works, the overall performance of the team’s ResNet model mimicked that of earlier models developed in the field of music genre recognition, such as the 2002 statistical pattern software model developed by Princeton computer scientist George Tzanetakism [G]. When compared to the ResNet model developed by Indian engineers, which is of similar architecture, the team’s primary model performs 12-25% worse [A].

**Ethical Considerations**

The ethical considerations in data collection and impact of using the model are as follows:

1. Informed Consent:

Informed consent is a key factor because the dataset contains audio data that was created by an individual. As such, it is important to ensure that the creators are informed about their music being used to compile a dataset, how their music will be used, and obtain permission to use their work. This is more to prevent intellectual theft and copyright infringements. This also implies the team as collectors being transparent with the data collection means and making the collected data replicable.

1. Bias and Fairness:

It is imperative that the dataset be representative of a variety of genres and styles to avoid bias. Selection of music should be based on a clear and objective criteria, not personal biases or preferences. A consequence that should be avoided is publishing a false classification as this could harm the sales of the artists whose music is processed by the model.

1. Privacy:

The music creators or rights holders should be assured that their personal information, such as their name and contact information, will not be shared or disclosed without their consent. They should also be informed about how their music will be stored, secured, and used to ensure their privacy is protected.

The size of the music collection of the user of the model limits the performance accuracy and usefulness. Companies with music distribution rights such as Spotify are more likely to be able to utilize a music genre classification model as they already have permission from the original song creators to utilize the music. Moreover, use of a model such as the one developed by the team is limited based on its area of use. Despite the advantage of a larger dataset collection, the usage of the model by companies like Spotify may be limited as they are responsible for presenting the music licensed to them in the appropriate genre category. If a company uses the model and it places a well known pop song in the country category, there may be fewer streams of the song by the target audience, thus impeding the amount of money earned by the artist. As such, models such as this would be better suited to research areas as there is less impact on artist earnings.

**Project Difficulty/Quality**

The project difficulty for music genre image classification is medium to high, as evidenced by several factors such as the complexity of the ResNet model and the data used in the project.

When attempting to develop a primary model, the initial choice was a CNN. When the validation accuracy for the model was found to be low, and no modifications resulted in any performance improvement, the team was faced with choosing a new model. Adding more layers to the CNN would lead to vanishing gradients, which leads to most of the original features being lost. As such, the team decided to switch to a ResNet model, which, through the implementation of skip connections, allows for more layers without sacrificing features.

The use of the ResNet ultimately did improve the overall accuracy, however, there were still barriers to high accuracy. This is in part due to the nature of the spectrograms. It is evident that the images produced do not have very clear identifiable patterns in appearance, which is reflected in the lack of features in the feature maps. In addition, as seen in section 7.3, the way in which the songs are collected can impact the model’s prediction ability. Songs from streaming services like Spotify are of better quality and will contain less noise, unlike live recordings or music videos from places like YouTube. As such, the accuracy of the model changed when songs were collected from different sources. Another challenge was attempting to identify songs that mimicked another genre (i.e., Levitating by Dua Lipa as a Jazz song). These types of covers proved difficult for any of the developed models to identify with high accuracy.

The team attempted to use various data augmentation techniques such as flips, rotation, translation, scaling, and adding noise to the images. However, these techniques did not help the project's success. Instead, the team normalized the images to create a clear color distinction between the features and increase the model's performance.

Lastly, the team attempted to look at the model performance as a measure of the top X predictions to see if a collection of predicted genres would allow the model to more accurately capture the true genre of a song. This yielded interesting results, as for the new Spotify data, while the model’s first guess was accurate 54.25% of the time, 75.5% of the time, the top 3 guesses included the true genre. This reflects the fact that songs could belong to a certain genre but have characteristics of another.

To further improve the model's quality and complexity, the team suggests exploring additional color data augmentation techniques such as adjusting brightness, contrast, and saturation to evaluate the model's performance. Additionally, the team could implement Vision Transformer (ViT), a transformer architecture specifically designed for image processing, to increase the image classification accuracy and overcome the model's difficulties.

[].

Conclusion:

In conclusion, the team generates a Resnet model for the use of music genre image classification. The Resnet model obtains a testing accuracy of 69.2% in comparison to the KNN and ANN baseline models, which obtain testing accuracies of 31.8% and 44% respectively. The project provides difficulty due to the vague and lack of distinct features across different genre’s of spectrograms. The team recommends to explore more colour data augmentation techniques to further improve the model’s performance.

Piazza: “it is subjective. And you can discuss your judgment and justification of your project's difficulty and what you have done to make your project different from those taught in tutorials and labs, including technical complexity, experiment design, practical impact, etc.”

* Data augmentation
* Different models (CNN, AlexNet, ResNet)