A Spectrogram-Based Music Genre Classifier

Mosamat Sabiha Shaikh
SHKMOS004

Fardoza Tohab
THBFAR002

Kristen Jodie Basson
BSSKRI003

1 PROBLEM DESCRIPTION

One of the key challenges in the music industry and related fields is accurately classifying musical genres. Without a reliable genre classification system, issues can arise that hinder music customization, music discovery, organization, research, and marketing efforts. With music playing an increasingly significant role in people's lives, music tastes are diverse, and genres span a wide range. Therefore, categorizing music and recommending new music to users of music-listening platforms and applications has become a crucial and modern topic [1].

Today, a significant portion of the world's population regularly downloads and purchases music through online music stores. Users frequently categorize their tastes by genre, such as hip hop, pop, or disco. However, the majority of the songs that are now accessible are not automatically categorized into a genre. Due to the vast amount of current collections, automatic genre classification is crucial for organizing, searching, retrieving, and recommending music [2].

Because it requires careful consideration and extraction of the right auditory elements, classifying music is regarded as a highly difficult process. While unlabelled data is easily accessible, there are extremely few songs with the correct genre tags. Feature extraction and categorization are the two fundamental phases in the classification of musical genres. Various characteristics are retrieved from the waveform in the first stage. A classifier is created in the second stage utilizing the characteristics that were taken from the training data. There are several methods that have been employed to categorize music into various genres. Due to the vast volume of music on the internet, an automated classification of music genres is required [2].

For this assignment, we explore one method of music genre classification using spectrograms generated from audio signals.

Some musical genres may be clearly separated from others when taking into account the time domain representation, as seen in figure 1. Jazz here has quite unique qualities from Rock and Metal music. However, as seen once more in Figure 1, some musical genres can exhibit traits that are very similar, making music classification a difficult task. In this case, Rock and Metal are far more similar to one another than Jazz, making them harder to categorize.

Signal features in the frequency domain appear to be more distinct as seen in figure 2, which is consistent with the frequency domain representation of the signals shown in figure 1. Therefore, by taking into account both time and frequency domain representations, more useful features may be used in the classification process.

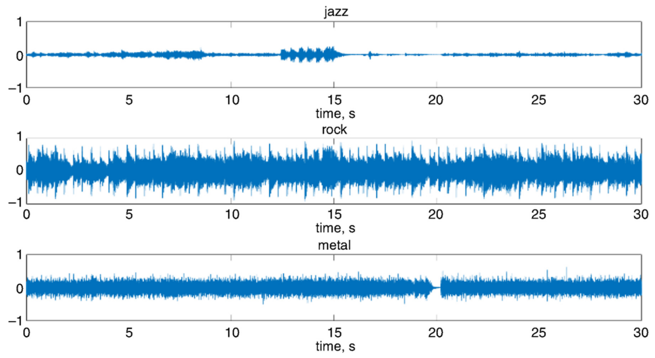


Figure 1: An illustration of the temporal domains of jazz, rock, and metal [1]

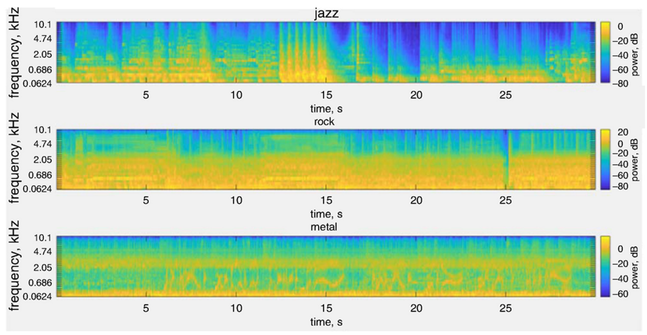
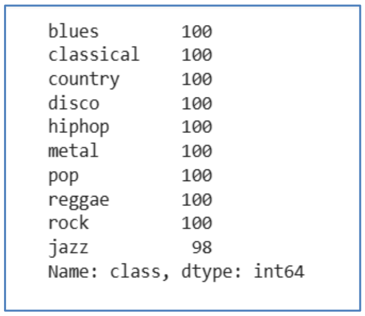


Figure 2: Spectrograms of the signals seen in Figure 1 [1]

2 DATA SET

In order to evaluate the networks in terms of classification accuracy, we made use of a popular music dataset known as the GTZAN data set obtained from Kaggle [6]. It has 1000 tracks with a sample frequency of 22,500 Hz, a 16-bit resolution, and a 30 second length. The GTZAN dataset includes the following genres: blues, classical, country, disco, jazz, metal, pop, reggae, and rock. 100 distinct samples of each genre are provided. Each audio clip has a length 30 seconds. The dataset consists of samples taken from a variety of sources such as radios, CDs, microphone recordings, etc.

Table 1. Dataset genre distribution



```
blues        100
classical    100
country      100
disco        100
hiphop       100
metal        100
pop          100
reggae       100
rock         100
jazz          98
Name: class, dtype: int64
```

3 ETHICS

Some of the ethical considerations surrounding the use of the GTZAN music dataset include:

- Since Western popular music genres make up the majority of the GTZAN dataset, non-Western or specialist genres may be absent or underrepresented. Due to this bias, genre classification models developed using the dataset may not be as inclusive and may continue to reinforce cultural inequality.
- The 10 genres used in the GTZAN dataset to classify songs may oversimplify and generalize the significant diversity seen within each genre. Recognizing that genres might include subgenres, regional variances, and subtleties that can be missed in a general classification system is vital.
- The GTZAN dataset should only be used in accordance with copyright and fair use laws, even if it was first made available for research purposes. When employing songs that are protected by copyright, researchers should make sure that they are following all applicable copyright rules and obtain the necessary licenses.
- The audio files included in the dataset are all 30 seconds in length. This was due to copyright reasons.

4 BASELINE MODEL

4.1 Music genre classification using spectrograms

The baseline model is a simple feed-forward neural network using python's Keras and TensorFlow libraries. The input image had to be reshaped to conform to what the network would take in. The images are presumptively image_size[0] by image_size[1] pixels in size, with three channels standing for the RGB colour information. The total amount of elements in each input image is determined by the reshape_size variable, which is helpful for the next layers.

The network is built using a Sequential model which has layers built onto it sequentially. The first layer is a reshape layer which flattens the input images into a 1D vector which was required to transform 2D image data into a format compatible with the fully connected layers that follow which is a number of Dense layers. The activation function used is ReLu to introduce non-linearity to the network and aids in the networks ability to learn detailed patterns in the dataset.

The number of nodes in each Dense layer gradually increases. Starting with 32 going up 64, 128 and 512. There is a Dropout layer with a dropout rate of 0.5 to prevent overfitting and improve generalisation. Dropout reduces the network's reliance on certain features.

The final layer has 10 output nodes representing the 10 potential classes of music genres for classification. The model gives each class a probability score that represents how likely it is that the input image belongs to that class. Running the code using the GTZAN dataset, the test accuracy we got was 10%.

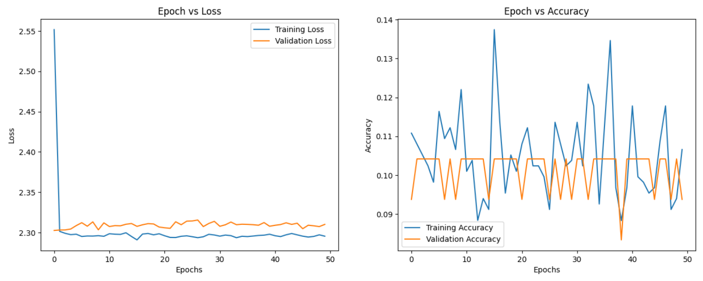


Figure 3: The graph depicts the training and validation loss and accuracy for the baseline code

4.2 Building on the Baseline model

To improve the accuracy of the model the next step was to implement a Convolutional Neural Network (CNN). The Sequential container is used again allowing a linear stack of layers to be added.

The first Layer is a Conv2D layer that has 32 filters of size 3 by 3. The convolutional layer applies the ReLu activation function to introduce non-linearity. This is followed by a MaxPool2D layer with a pool size of 2 by 2. This layer reduces the spatial dimensions of the feature maps but keeps relevant information.

There are 3 more Conv2D layers that have 46 and 128 nodes each sized 3 by 3. They are followed by MaxPool2D layers each helping to further capture more complex information from the dataset. The depth and complexity of the feature representation in the model are enhanced by these layers.

A Flatten layer is added to the data to prepare it for the fully connected layers. The output from the previous layer's multidimensional layer is flattened in this layer into a 1D vector. The model gains more non-linearity from this fully connected layer, which enables it to recognize more intricate patterns in the data.

There is a dropout of 0.5 as well in this model. The final layer has 10 outputs representing the 10 classes that will be used for classification. The Softmax activation function is used to estimate the probability of distribution over the classes and enable the model to give each class a probability score.

There was a significant increase in test accuracy noticed with an accuracy of 46%. The optimizer used was Adam. I changed the optimizer to SGD and the accuracy slightly increased to 48%.

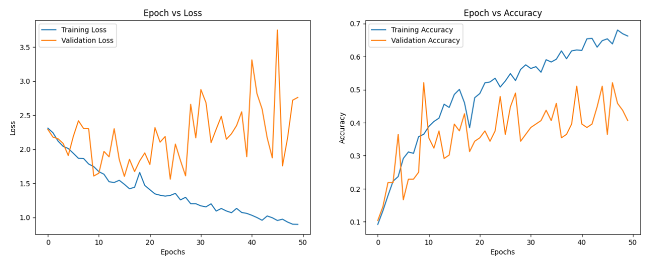


Figure 4: The graph depicts the training and validation loss and accuracy for the CNN model with Adam

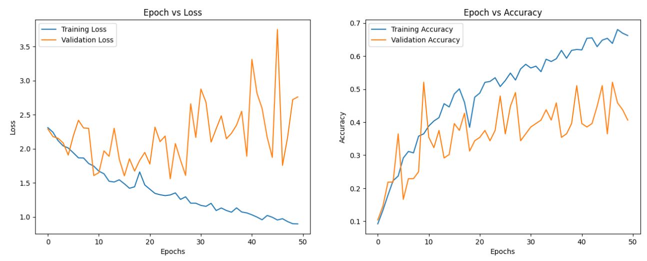


Figure 5: The graph depicts the training and validation loss and accuracy for the CNN model with SGD

We noticed a significant increase in the test accuracy using CNN than Feed Forward. CNN has a good local understanding of images that is satisfactory making it a popular neural network for image classification problems [10]. We kept this model and build on it in the next section to further improve classification.

5 EXPERIMENTAL SETUP

5.1 Dataset

We made use of the GTZAN music dataset from Kaggle [6]. It contains 10 musical genres, each having 100 audio clips in .wav format. The genres include blues, classical, country, disco, hip hop, jazz, metal, pop, reggae and rock. Each audio has a length of 30 seconds.

5.2 Data split

We split the dataset into 80% for training, 10% for testing and 10% for validation of the classifier neural network. The .wav files are first transformed into the relevant spectrograms. The CNN classifier receives these spectrograms (images) as input data.

1. Training Dataset: contains 799 images and their corresponding labels.

2. Validation Dataset: contains 100 images and their corresponding labels.

3. Test Dataset: contains 100 images and their corresponding labels.

5.3 Software tools

The deep learning model is built on the PyTorch framework in Python. PyTorch is a fully functional framework for creating deep learning models, a subset of machine learning that's frequently employed in tasks like image classification and language processing [7].

We also used Google Collaboratory as a platform to build, train and test the networks.

5.4 Performance measure

To assess the performance of the classifier, we will be using the confusion matrix and the classification accuracy. The classification accuracy is the ratio of trace of confusion matrix to the sum of all elements of the confusion matrix.

6 MODEL DESIGN AND EVALUATION

The proposed method for the musical genre classification is based on spectrograms as described. The suggested method's general block diagram is shown in Figure 3. The music signals are first transformed into the appropriate spectrograms. These spectrograms are visual representations of these musical signals. Spectrograms display the time and frequency components of musical signals. The CNN classifier is then given these spectrograms as input. The CNN classifier handles feature extraction all by itself which is the primary benefit of utilizing CNN classifier over conventional neural networks.

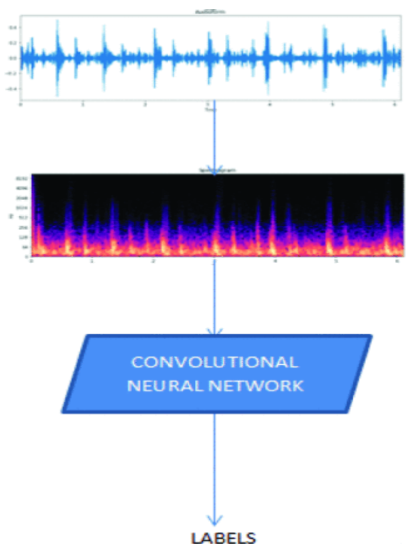


Figure 6: Block diagram of the proposed genre classification system [5]

Layers and batch normalisation

For the model design from the baseline the CNN, Adam optimiser activation function, dropout rate of 0.5 and 50 epochs remained the same. A fifth layer was added as this increases the model's ability to learn more complex features [9]. Batch normalisation was implemented for each of the layers since it normalises the activation of each layer, enables higher learning rates and is a form of regularisation since the dataset that is used is relatively small and prone to overfitting [8]. After applying all these the test accuracy increased from 0.46 from the baseline to 0.55.

Optimiser and scheduler

The optimiser was changed from Adam to SGD with a learning rate of 0.0005 and momentum of 0.9. A scheduler was added with a step size of 10 and gamma 0.1. It was seen that the network performed significantly better with the scheduler and optimiser with a test accuracy of 0.67. The training and validation accuracy in figure 7. also increased significantly. Removing the scheduler was seen to decrease the test accuracy to 0.66 thus it was kept.

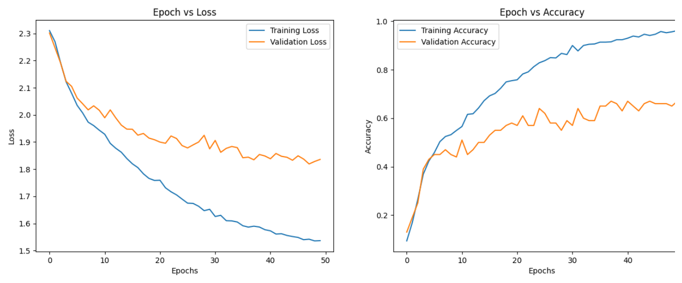


Figure 7: The graph depicts the training and validation loss and accuracy of the CNN with SGD and a learning rate of 0.0005

Momentum and gamma

It was seen that increasing or decreasing the momentum hyperparameter in the SGD optimiser yielded significantly worse accuracy therefore it was kept constant at 0.9. The gamma hyperparameter was also tested by increasing and decreasing the value but also resulted in a significantly lower test accuracy therefore was also kept at 0.1.

Learning rate

Increasing the learning rate hyperparameter in the SGD optimiser from 0.0005 to 0.0007, the test accuracy increased to 0.68. Increasing the learning rate any more than this decreased the test accuracy.

L1 and L2 regularization

L1 and L2 regularization was implemented since the dataset is relatively small and prone to overfitting. It and was seen to decrease the test accuracy as well as the training and validation accuracy. L1 was separately added without L2 and still decreased overall accuracy. L2 to was then implemented without L1 and with lambda as 0.002 which in contrast increased the test accuracy to 0.71. Lambda as 0.002 yielded the highest test accuracy.

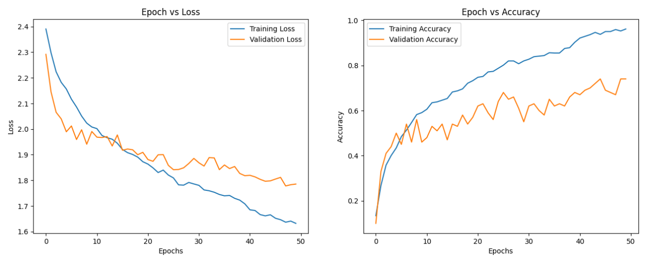


Figure 8: The graph depicts the training and validation loss and accuracy of the CNN with L2 regularization with lambda 0.002

Augmentation

Data augmentation was tested and seen to drastically worsen the results, completely separating the training and validation accuracy and loss.

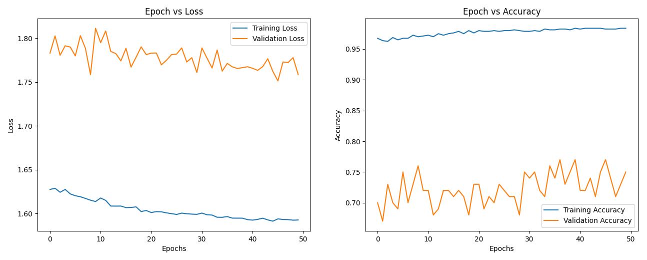


Figure 9: The graph depicts the training and validation loss and accuracy of the CNN with augmentation

Early stopping

Early stopping was implemented which finds the point on the validation set at which the model's performance is optimal. It was tested with a default patience of 5 and then increased and decreased. The accuracy did not improve reaching at the highest 0.63 test accuracy.

Normalisation

Normalisation was implemented in the training and validation dataset which can remove any bias from the dataset, unfortunately when implementing this the test accuracy drops to 0.1 thus not improving the network.

Final model

The final adapted model is a CNN with 5 convolutional layers and batch normalisation, using the ReLu activation function and the softmax function in the fully connected layer. The SGD optimiser was used with a learning rate of 0.0007 and momentum of 0.9 as well as a scheduler with a step size of 10 and gamma value of 0.01. A dropout rate of 0.5 is used as well as L2 regularization with lambda as 0.002. The final test accuracy for this model is 0.72 and below is the validation and training accuracy and loss.

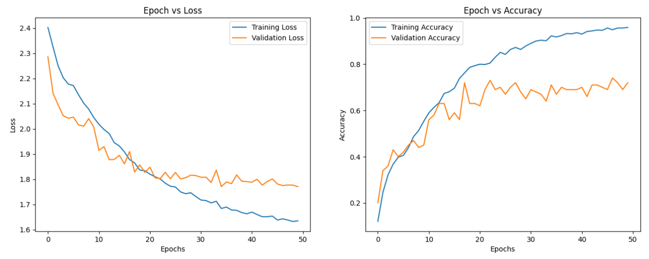


Figure 10: The graph depicts the final CNN training and validation accuracy and loss

7 RESULTS AND ANALYSIS OF MODEL PERFORMANCE

A ten-class classification of music files was done. The following files were chosen for classification:

- Blues (100 .wav files)
- Metal (100 .wav files)
- Country (100 .wav files)
- Classical (100 .wav files)
- Rock (100 .wav files)
- Pop (100 .wav files)
- Jazz (100 .wav files)
- Reggae (100 .wav files)
- Disco (100 .wav files)
- Hip-hop (100 .wav files)

The appropriate spectrograms for the .wav files were created. The CNN classifier was provided with these spectrograms as input. The following observations were made:

The number of epochs was set as 50. The training and validation reach a maximum training accuracy of 96%. A dropout of 0.5 is provided to our sequential CNN model this improved the overfitting. Regularization, in general, penalizes the coefficients that cause the overfitting of the model. L1 regularization is called Lasso Regression and L2 is called Ridge Regression. Lasso regression solution is sparse while Ridge regression solution is non-sparse. When implementing L2 regularization it was seen to improve overfitting quite significantly as can be seen in figure 8.

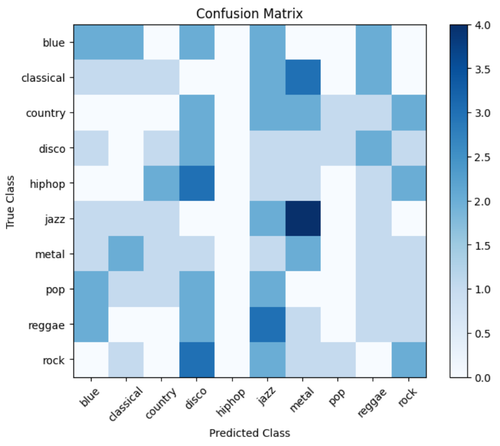
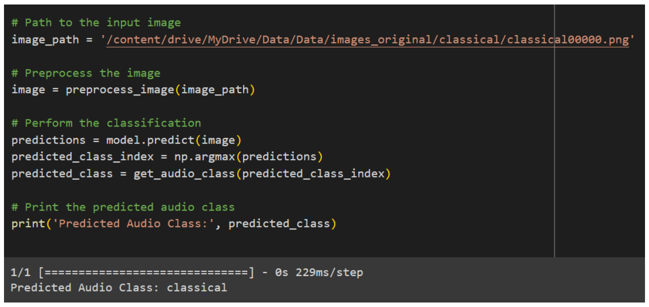


Figure 11: Confusion Matrix for the Base CNN model

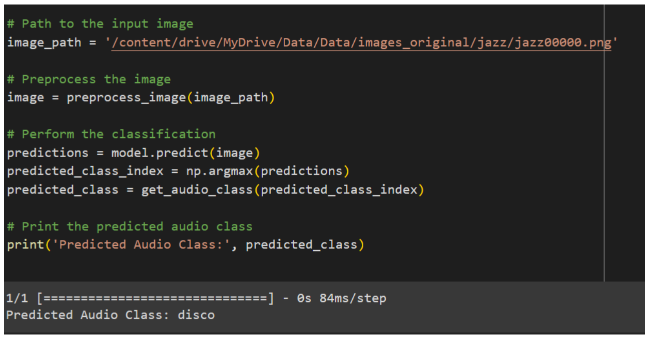
Figure 10 shows the confusion matrix that was plotted for the third iteration of the Base model which is a CNN model that uses SDG and attained a test accuracy of 48%. The matrix provides a summary of the prediction results for the classification.

As it can be seen, the model has high values in off diagonal cells indicating a high number of misclassifications. The model is mostly predicting incorrectly the class of the input. There is an imbalance in the distribution and a clear lack of pattern. We can deduce that the model has poor generalization and lacks meaningful learning. The model is failing to correctly capture the underlying patterns in the data and is not performing well in classifying.



```
# Path to the input image
image_path = '/content/drive/MyDrive/Data/Data/images_original/classical/classical00000.png'

# Preprocess the image
image = preprocess_image(image_path)

# Perform the classification
predictions = model.predict(image)
predicted_class_index = np.argmax(predictions)
predicted_class = get_audio_class(predicted_class_index)

# Print the predicted audio class
print('Predicted Audio Class:', predicted_class)

1/1 [==============================] - 0s 229ms/step
Predicted Audio Class: classical
```

Figure 12: Classifying a classical music spectrogram image



```
# Path to the input image
image_path = '/content/drive/MyDrive/Data/Data/images_original/jazz/jazz00000.png'

# Preprocess the image
image = preprocess_image(image_path)

# Perform the classification
predictions = model.predict(image)
predicted_class_index = np.argmax(predictions)
predicted_class = get_audio_class(predicted_class_index)

# Print the predicted audio class
print('Predicted Audio Class:', predicted_class)

1/1 [==============================] - 0s 84ms/step
Predicted Audio Class: disco
```

Figure 13: Classifying a jazz music spectrogram image

As can be seen from Figure 12 and Figure 13, the model is able to classify an image spectrogram correctly but also fails for other images.

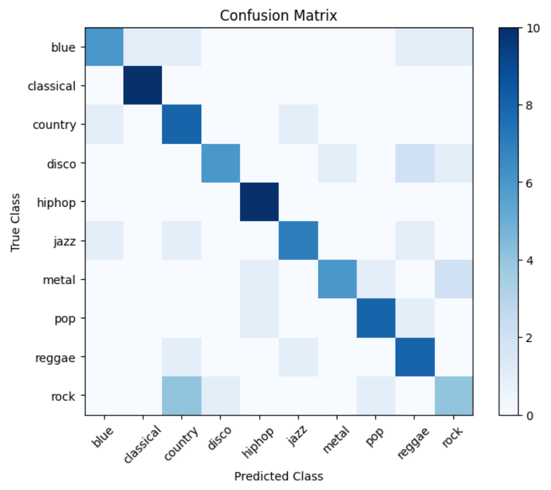
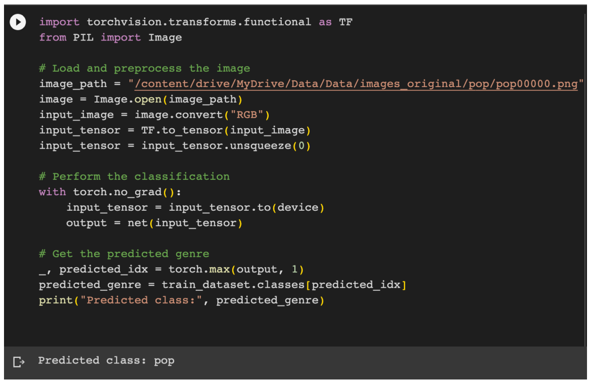


Figure 14: Confusion Matrix for the final CNN model

Figure 14 is the confusion matrix shown for the final CNN that shows how accurately the specific genres would be predicted. The model gave a test accuracy of 72%

Looking at Figure 14 it can be seen that the latest neural network model exhibits characteristics that shows that it performs classification well. The darker colours forming a diagonal shape represents correct classifications. We can deduce that the model can learn distinct features and patterns from the dataset and effectively differentiate between them. There are minimal deep colour off-diagonal boxes indicating minimal confusion between classification of the different classes.

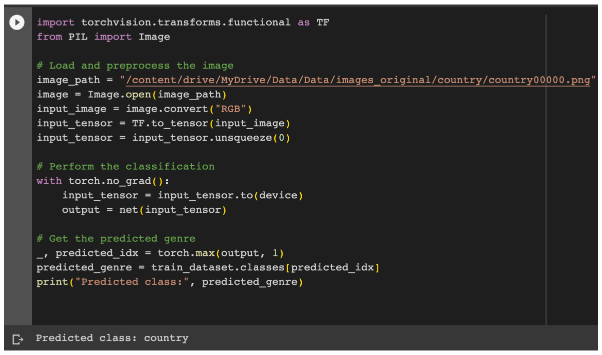
Figures 15 and 16 below shows how the final model successfully classifies mel spectrogram images into the respective genres.



```
import torchvision.transforms.functional as TF
from PIL import Image

# Load and preprocess the image
image_path = "/content/drive/MyDrive/Data/Data/images_original/pop/pop00000.png"
image = Image.open(image_path)
input_image = image.convert("RGB")
input_tensor = TF.to_tensor(input_image)
input_tensor = input_tensor.unsqueeze(0)

# Perform the classification
with torch.no_grad():
    input_tensor = input_tensor.to(device)
    output = net(input_tensor)

# Get the predicted genre
_, predicted_idx = torch.max(output, 1)
predicted_genre = train_dataset.classes[predicted_idx]
print("Predicted class:", predicted_genre)
```

```
Predicted class: pop
```

Figure 15: Classifying a pop music spectrogram image



```
import torchvision.transforms.functional as TF
from PIL import Image

# Load and preprocess the image
image_path = "/content/drive/MyDrive/Data/Data/images_original/country/country00000.png"
image = Image.open(image_path)
input_image = image.convert("RGB")
input_tensor = TF.to_tensor(input_image)
input_tensor = input_tensor.unsqueeze(0)

# Perform the classification
with torch.no_grad():
    input_tensor = input_tensor.to(device)
    output = net(input_tensor)

# Get the predicted genre
_, predicted_idx = torch.max(output, 1)
predicted_genre = train_dataset.classes[predicted_idx]
print("Predicted class:", predicted_genre)
```

```
Predicted class: country
```

Figure 16: Classifying a country music spectrogram image

10 CONCLUSIONS

We performed music genre classification using the GTZN dataset. We started off with a base code that gave an accuracy of 10% using a feedforward network. This base was modified to a CNN which dramatically boosted the accuracy to 48%. We built on the CNN base model to produce a model that returned a test accuracy of 72%. The latest model gives a satisfactory level of classification making very accurate predictions.

11 REPRODUCING RESULTS

11.1 Instructions

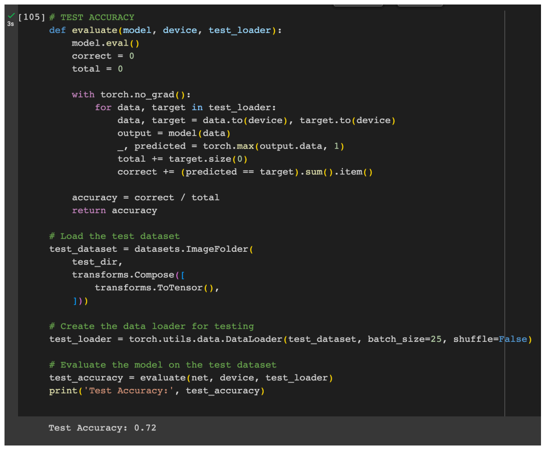
The results for the baseline and final models can be reproduced using the following steps:

- The notebooks for both the baseline and final model can be run using Google Collab.
- You need to mount your drive at /content/drive
- After that you should load the dataset from Kaggle in a folder called Data.
- The train, testing and validation data are automatically split by the code.
- The model will then be trained and accuracy produced.

11.2 Hyperparameters

The model is a CNN with 5 convolutional layers and batch normalisation, using the ReLu activation function and the softmax function in the fully connected layer. The SGD optimiser was used with a learning rate of 0.0007 and momentum of 0.9 as well as a scheduler with a step size of 10 and gamma value of 0.01. A dropout rate of 0.5 is used as well as L2 regularization with lambda as 0.002. The final test accuracy for this model is 0.72 and below is the validation and training accuracy and loss.

11.3 Test results



```
[105] # TEST ACCURACY
      def evaluate(model, device, test_loader):
          model.eval()
          correct = 0
          total = 0

          with torch.no_grad():
              for data, target in test_loader:
                  data, target = data.to(device), target.to(device)
                  output = model(data)
                  _, predicted = torch.max(output.data, 1)
                  total += target.size(0)
                  correct += (predicted == target).sum().item()

          accuracy = correct / total
          return accuracy

      # Load the test dataset
      test_dataset = datasets.ImageFolder(
          test_dir,
          transforms.Compose([
              transforms.ToTensor(),
          ]))

      # Create the data loader for testing
      test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=25, shuffle=False)

      # Evaluate the model on the test dataset
      test_accuracy = evaluate(net, device, test_loader)
      print('Test Accuracy:', test_accuracy)
```

```
Test Accuracy: 0.72
```

REFERENCES

[1] Ahmet Elbir and Nazan Aydin. 2020. Music genre classification and music recommendation by using deep learning. 56, 12 (June 2020), 627–629. DOI:https://doi.org/10.1049/el.2019.4202

[2] Satti Vishnupriya and K. Meenakshi. 2018. Automatic Music Genre Classification using Convolution Neural Network. (January 2018). DOI:https://doi.org/10.1109/iccci.2018.8441340

[3] Beatrix Benko, Lina Teichmann. 2023. Music classification and generation with spectrograms — Neuromatch Academy: Deep Learning. Neuromatch.io. Retrieved May 21, 2023 from https://deeplearning.neuromatch.io/projects/ComputerVision/spectrogram_analysis.html

[4] StudyGyaan. 2021. GTZAN Dataset - Music Genre Classification using Python. YouTube. Retrieved May 22, 2023 from https://www.youtube.com/watch?v=2mCfP6mpQpo

[5] Nirmal Kumar R and Shajee Mohan. 2020. Music Genre Classification using Spectrograms. (December 2020). DOI:https://doi.org/10.1109/picc51425.2020.9362364

[6] Andrada. 2020. GTZAN Dataset - Music Genre Classification. Kaggle.com. Retrieved May 24, 2023 from https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[7] 2017. What is PyTorch? NVIDIA Data Science Glossary. Retrieved May 24, 2023 from https://www.nvidia.com/en-us/glossary/data-science/pytorch/#:~:text=PyTorch%20is%20a%20fully%20featured,developers%20to%20learn%20and%20use.

[8] Chablani, M. (2017) *Batch normalization*, *Medium*. Available at: https://towardsdatascience.com/batch-normalization-8a2e585775c9#:~:text=Using%20batch%20normalization%20allows%20us,difficult%20when%20creating%20deeper%20networks.

[9] Brownlee, J. (2020) *How to control neural network model capacity with nodes and layers*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/how-to-control-neural-network-model-capacity-with-nodes-and-layers/#:~:text=The%20number%20of%20layers%20in%20a%20model%20is%20referred%20to,a%20vast%20number%20of%20nodes.

[10] James Le. The 4 Convolutional Neural Network Models That Can Classify Your . *Towards Data Science*. Retrieved May 24, 2023 from https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images-9fe7f3e5399d

[11] Brownlee, J. (2020) *How to control neural network model capacity with nodes and layers*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/how-to-control-neural-network-model-capacity-with-nodes-and layers/#:~:text=The%20number%20of%20layers%20in%20a%20model%20is%20referred%20to,a%20vast%20number%20of%20nodes.