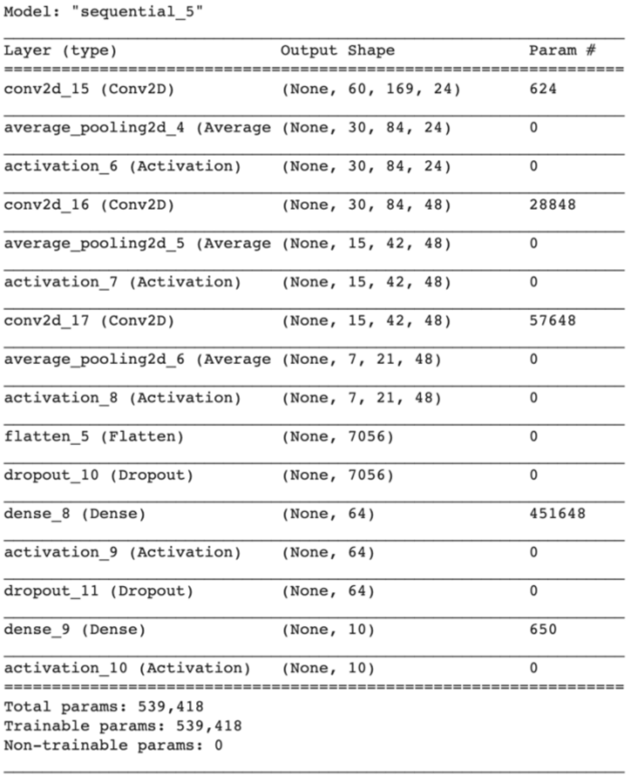
1. Final Training Results

Since I was not yet satisfied with the results I obtained, I decided to continue along the way in the research from BAHULEYAN, Hareesh on Music Genre Classification using Machine Learning Techniques[1]. In fact, I extracted the spectrograms from each audio tracks, which can be treated as images and used by the Convolutional Neural Network to predict the genre label. Some of the spectrogram samples can be found in the folder Spectrogram_Samples. The model follows a standard CNN structure implementing different layers from the keras library. We started by importing a sequential model and adding a 2D-convolutional layer with a kernel size of (5, 5) to extract the spectrogram's features. It is followed by a rectified linear unit as activation function and a max pooling layer with a pool-size of (2, 2) to reduce the matrix size for following layers. Finally, a dropout layer is added to randomly ignore neurons in order to prevent overfitting. I opted to use 3 convolutional layers where the second and third layers follows the same architecture. We then flatten the matrix into a single vector with a dropout rate of 0.5 and pass it into 2 dense layers followed by a dropout layer to accurately categorize the image. We compiled the model using an Adam optimizer and a categorical cross-entropy loss function. The model was set to train for 90 epochs; however, Early Stopping was implemented to monitor the validation loss with a patience of 20, a common number of epochs before early stoppage.



```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_15 (Conv2D)           (None, 60, 169, 24)       624       
_________________________________________________________________
average_pooling2d_4 (Average (None, 30, 84, 24)        0         
_________________________________________________________________
activation_6 (Activation)    (None, 30, 84, 24)        0         
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 30, 84, 48)        28848     
_________________________________________________________________
average_pooling2d_5 (Average (None, 15, 42, 48)        0         
_________________________________________________________________
activation_7 (Activation)    (None, 15, 42, 48)        0         
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 15, 42, 48)        57648     
_________________________________________________________________
average_pooling2d_6 (Average (None, 7, 21, 48)         0         
_________________________________________________________________
activation_8 (Activation)    (None, 7, 21, 48)         0         
_________________________________________________________________
flatten_5 (Flatten)          (None, 7056)              0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 7056)              0         
_________________________________________________________________
dense_8 (Dense)              (None, 64)                451648    
_________________________________________________________________
activation_9 (Activation)    (None, 64)                0         
_________________________________________________________________
dropout_11 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_9 (Dense)              (None, 10)                650       
_________________________________________________________________
activation_10 (Activation)   (None, 10)                0         
=================================================================
Total params: 539,418
Trainable params: 539,418
Non-trainable params: 0
_________________________________________________________________
```

Figure 1.1 Convolutional Neural Network Implementation

Training and Validation Results

Training loss: 0.3873

Training accuracy: 0.8702

Validation loss: 0.9925

Validation accuracy: 0.7520

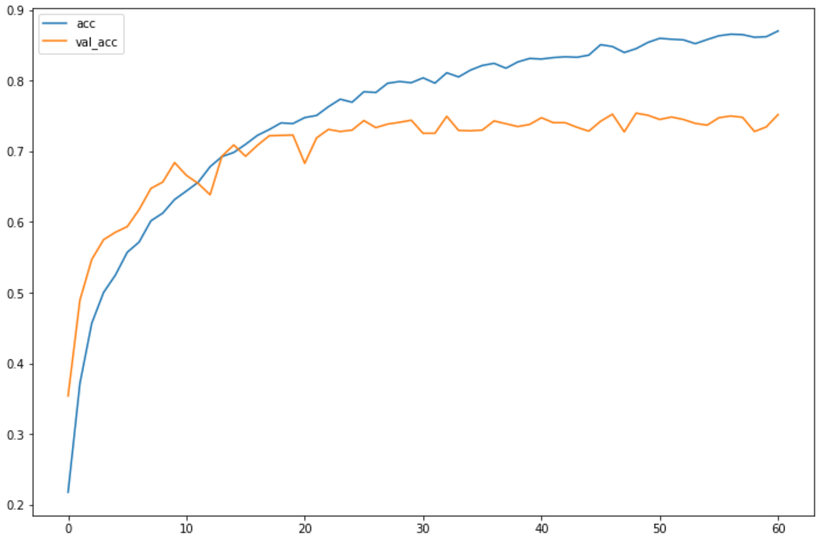


Figure 1.2 Training and Validation Accuracy over 61 epochs

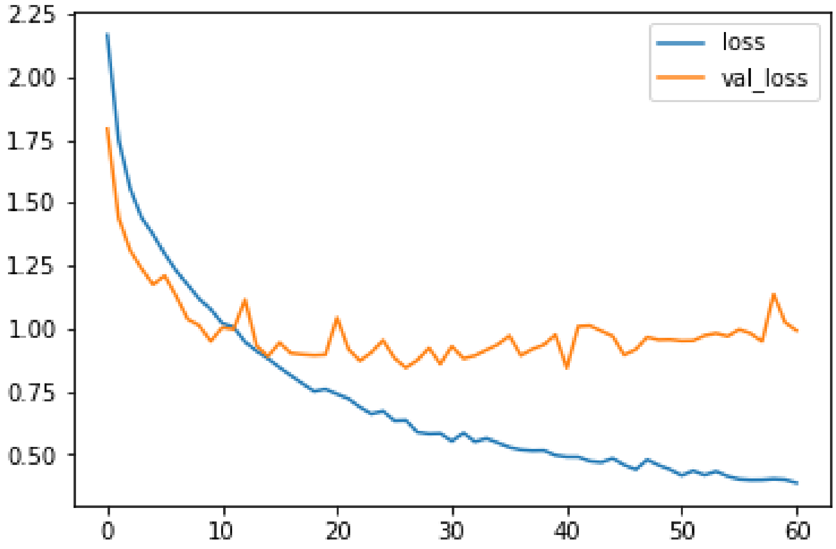
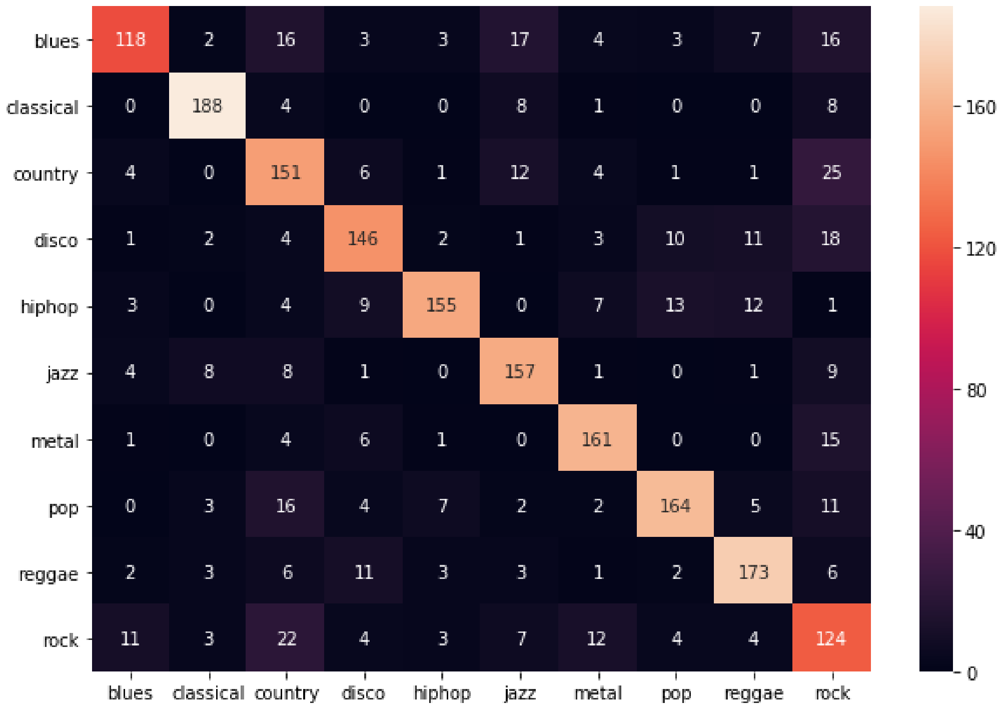


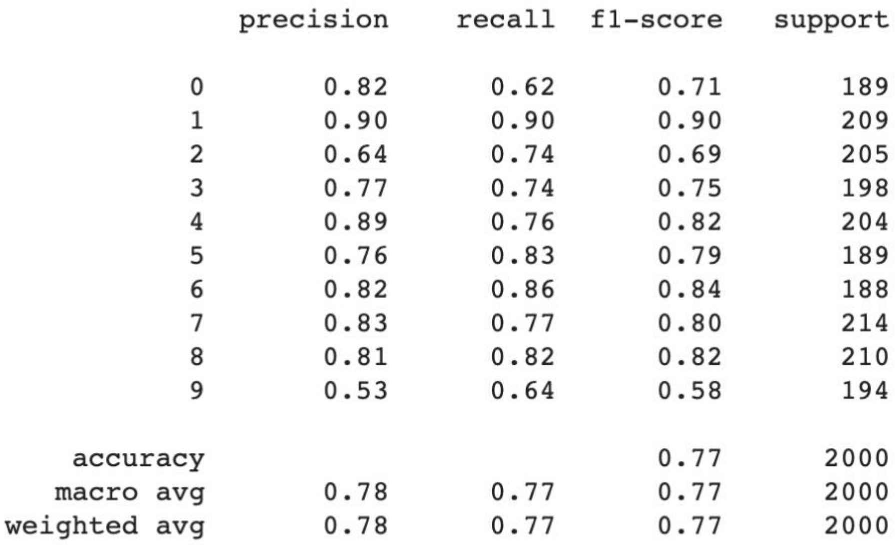
Figure 1.3 Training and Validation Loss over 61 epochs

Confusion Matrix



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	118	2	16	3	3	17	4	3	7	16
classical	0	188	4	0	0	8	1	0	0	8
country	4	0	151	6	1	12	4	1	1	25
disco	1	2	4	146	2	1	3	10	11	18
hiphop	3	0	4	9	155	0	7	13	12	1
jazz	4	8	8	1	0	157	1	0	1	9
metal	1	0	4	6	1	0	161	0	0	15
pop	0	3	16	4	7	2	2	164	5	11
reggae	2	3	6	11	3	3	1	2	173	6
rock	11	3	22	4	3	7	12	4	4	124

Classification Report



```
              precision    recall  f1-score   support

           0       0.82      0.62      0.71       189
           1       0.90      0.90      0.90       209
           2       0.64      0.74      0.69       205
           3       0.77      0.74      0.75       198
           4       0.89      0.76      0.82       204
           5       0.76      0.83      0.79       189
           6       0.82      0.86      0.84       188
           7       0.83      0.77      0.80       214
           8       0.81      0.82      0.82       210
           9       0.53      0.64      0.58       194

    accuracy                           0.77      2000
   macro avg       0.78      0.77      0.77      2000
weighted avg       0.78      0.77      0.77      2000
```

From the research report from BAHULEYAN, Hareesh (2018) on Music Genre Classification using Machine Learning Techniques[1], a method utilizing the audio track spectrograms and deep learning techniques resulted in far better results. Indeed, the test results using the CNN model were much better:

Test loss: 0.8917159118652344

Test accuracy: 0.7685

In addition, from Derek A. Huang, Arianna A. Serafini, and Eli J. Pugh's research on Music Genre Classification2, significant performance increases across all models were obtained from proper data processing, using a 64 mel-bins and a window length of 512 samples with an overlap of 50% between windows as well as a log-scaling using the formula log(X2). I sampled a contiguous 2-second window at 14 random locations from each audio track in order to increase our data size from 1000 clips to 14000 clips of two seconds each. The results between 8000 clips and 14000 clips in our dataset were tremendous, resulting in an almost 7% accuracy difference. For further steps, I believe that through further preprocessing and optimization, it is possible to get 80-90% accuracies.

2. Final Demonstration Proposal

For my final product and final integration approach, I will create a landing page type website to demo my model and results. I will use the technology stack composed of Vue.js for the frontend and Flask for the backend since I believe the back-end component is relatively simple and Flask is suitable to import my training weights. As for past experiences with the technologies, I have been learning Flask during the past month following along the Flasky: Flask Web Development book by Miguel Grinberg (https://flaskbook.com/). I have also been working with Vue.js for the frontend component of other projects.

[1] BAHULEYAN, Hareesh. (2018). Music Genre Classification using Machine Learning Techniques. eprint arXiv:1804.01149.

[2] HUANG, Derek A., Serafini, Arianna A., PUGH, Eli J. (2019). Music Genre Classification. eprint.