Music Genre Classification

Daniel Amin, Kiwook Kwon, Stephanie Luk

DSE I1910 & CSC 84200, Spring 2020, Prof. Grossberg

Problem Statement

- To build a machine learning model which classifies the genre of a song

Motivation

- Music Information Retrieval (MIR)
  - a field concerned with browsing, searching, and organizing large music collections

- Automatic genre classification
  - can assist or replace humans
  - can provide framework for development and evaluation of features for any type of content-based analysis of musical signals

Why FMA over other datasets?

- Essential qualities for a reference benchmark:
  - Large scale
  - Permissive licensing
  - Available audio
  - Quality audio
  - Metadata rich
  - Easily accessible
  - Future proof and reproducible

dataset[1]	#clips	#artists	year	audio
RWC [12]	465	-	2001	yes
CAL500 [45]	500	500	2007	yes
Ballroom [13]	698	-	2004	yes
GTZAN [46]	1,000	~ 300	2002	yes
MusiClef [36]	1,355	218	2012	yes
Artist20 [7]	1,413	20	2007	yes
ISMIR2004	1,458	-	2004	yes
Homburg [15]	1,886	1,463	2005	yes
103-Artists [30]	2,445	103	2005	yes
Unique [41]	3,115	3,115	2010	yes
1517-Artists [40]	3,180	1,517	2008	yes
LMD [42]	3,227	-	2007	no
EBallroom [23]	4,180	-	2016	no[2]
USPOP [1]	8,752	400	2003	no
CAL10k [44]	10,271	4,597	2010	no
MagnaTagATune [20]	25,863[3]	230	2009	yes[4]
Codaich [28]	26,420	1,941	2006	no
FMA	**106,574**	**16,341**	**2017**	**yes**
OMRAS2 [24]	152,410	6,938	2009	no
MSD [3]	1,000,000	44,745	2011	no[2]
AudioSet [10]	2,084,320	-	2017	no[2]
AcousticBrainz [32]	2,524,739[5]	-	2017	no

[1] Names are clickable links to datasets' homepage.
[2] Audio not directly available, can be downloaded from ballroomdancers.com, 7digital.com, youtube.com.
[3] The 25,863 clips are cut from 5,405 songs.
[4] Low quality 16 kHz, 32 kbit/s, mono mp3.
[5] As of 2017-07-14, of which a subset has been linked to genre labels for the MediaEval 2017 genre task.

Table 1: Comparison between FMA and alternative datasets.

FMA

Problem Description

- Given an audio file (mp3), how well can we correctly classify its root genre
  - Using Mel Frequency Cepstral Coefficients (MFCC)
  - Using mel-spectrograms

Dataset

- Free Music Archive (FMA)
  - 106,574 tracks, 917 GB
  - 161 genres
    - 16 root genres

- dataset = audio + metadata

- Metadata
  - song title, album, artist, per-track genres
  - MFCC features

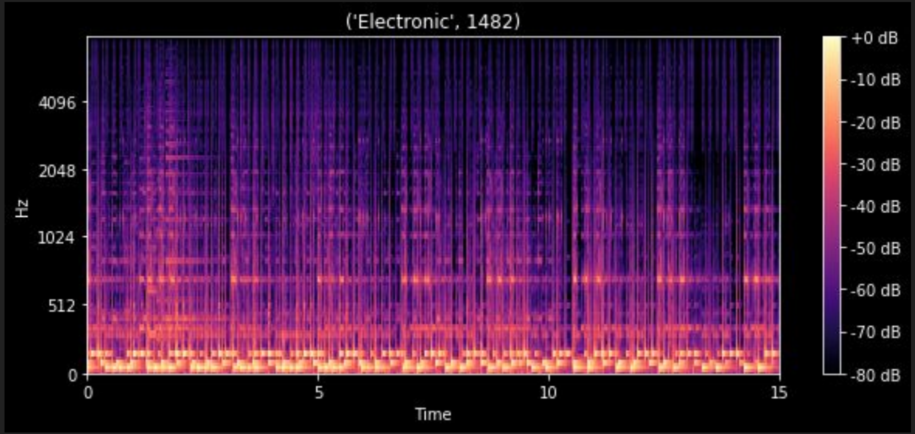
FMA

Small subset

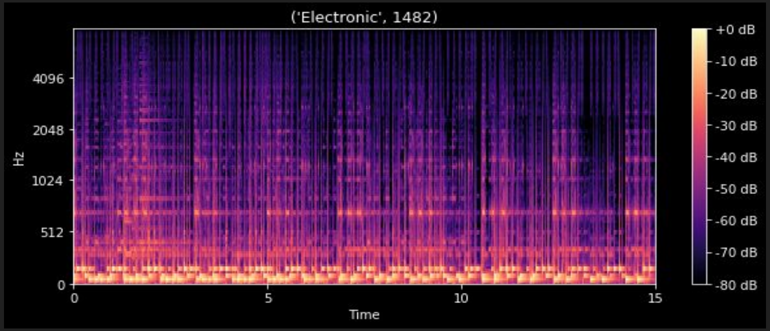
- ~8GB
- 8,000 audio clips (mp3)
    - 30 seconds each
- 8 genres
    - balanced

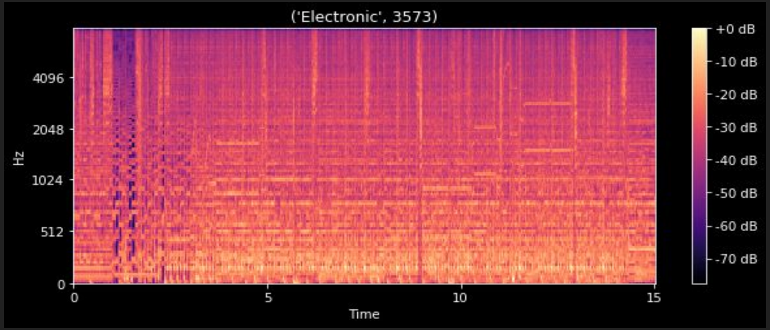
Genre	Track Count
Electronic	1000
Experimental	1000
Folk	1000
Hip-Hop	1000
Instrumental	1000
International	1000
Pop	1000
Rock	1000
Total	**8000**

Audio clip & Spectrograms

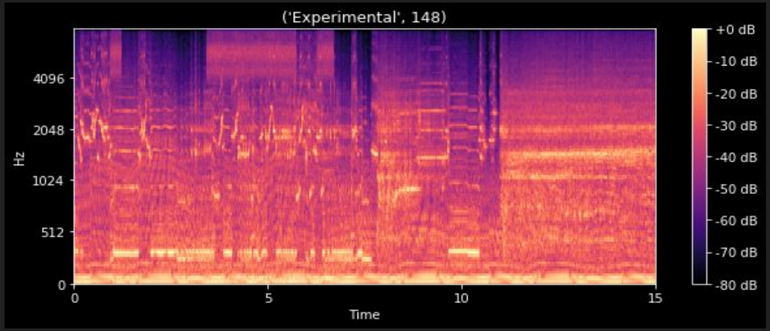


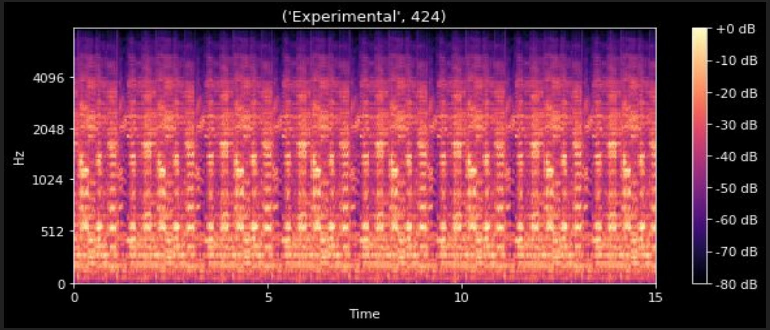
Electronic





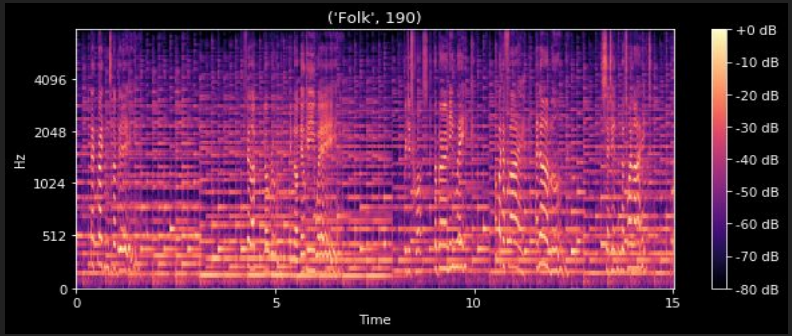
Experimental

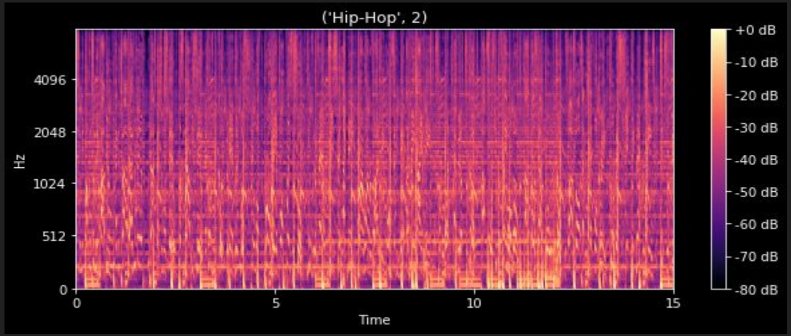


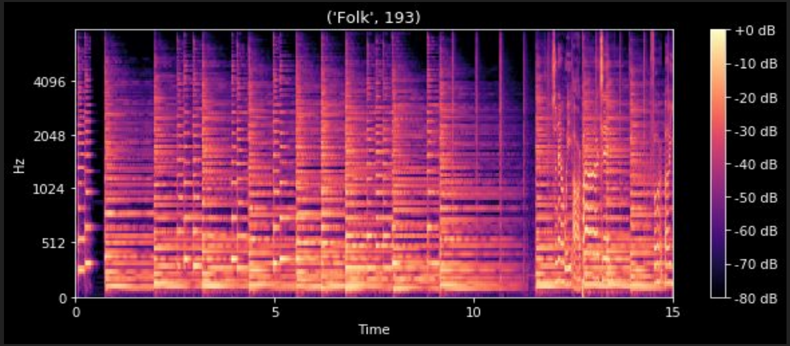


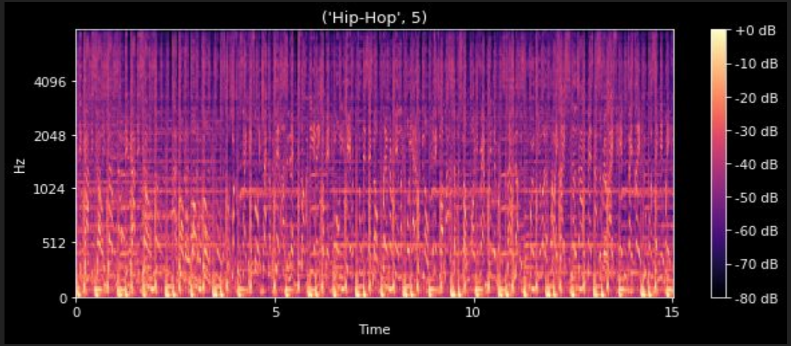
Folk

Hip-Hop


('Folk', 190)
Hz
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

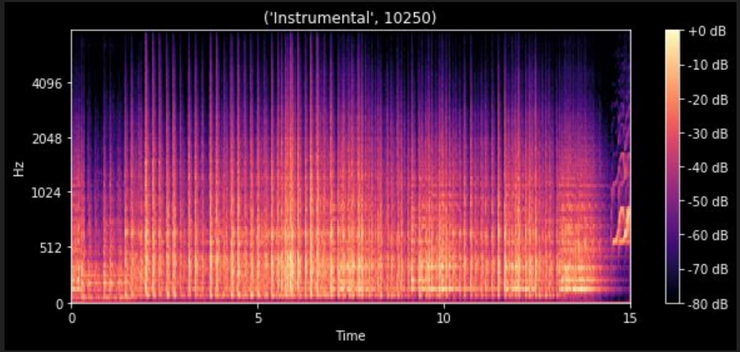

('Hip-Hop', 2)
Hz
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

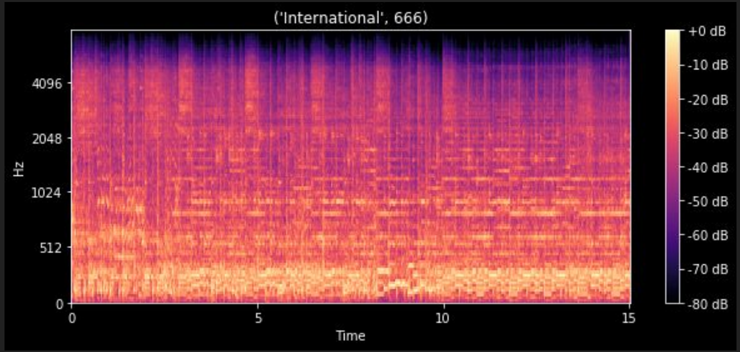

('Folk', 193)
Hz
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

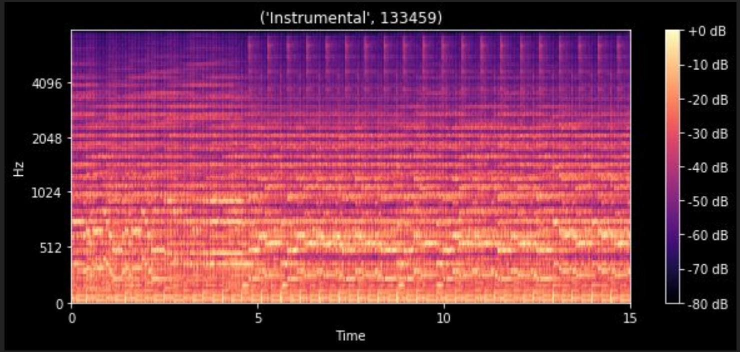

('Hip-Hop', 5)
Hz
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

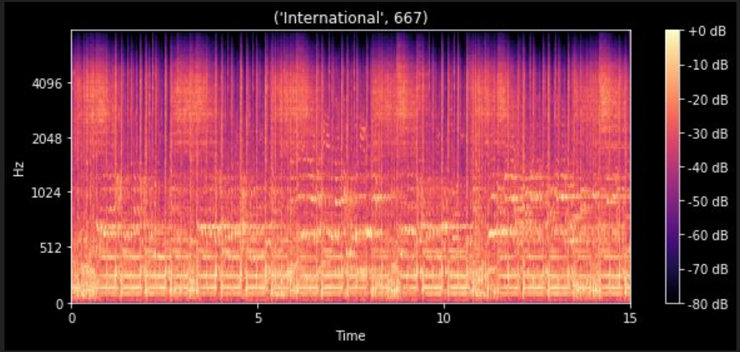
Instrumental

International

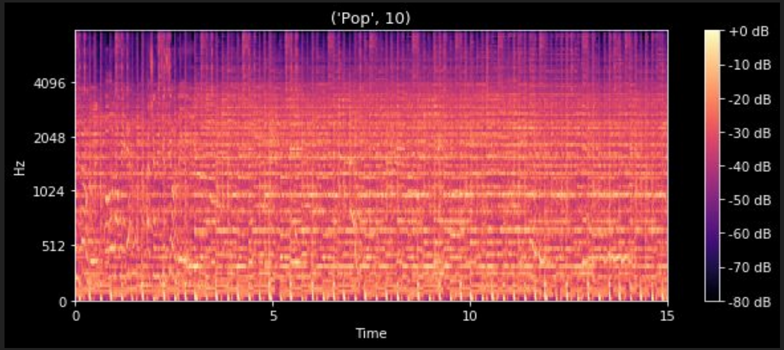


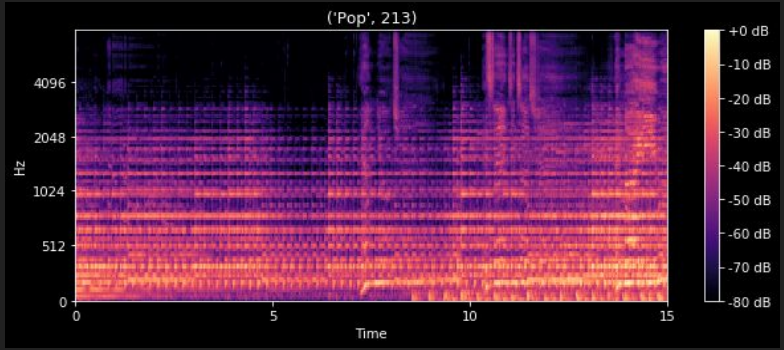




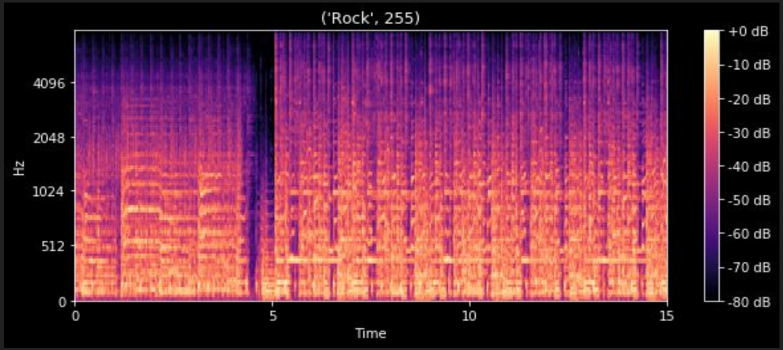


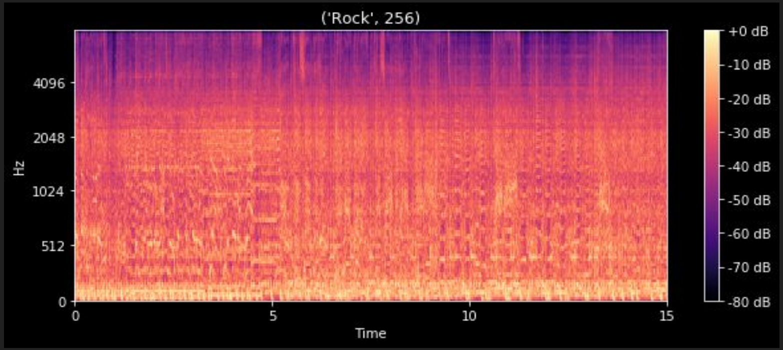
Pop


('Pop', 10)
Hz
4096
2048
1024
512
0
Time
0
5
10
15
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB


('Pop', 213)
Hz
4096
2048
1024
512
0
Time
0
5
10
15
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

Rock


('Rock', 255)
Hz
4096
2048
1024
512
0
Time
0
5
10
15
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB


('Rock', 256)
Hz
4096
2048
1024
512
0
Time
0
5
10
15
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

Design Overview

Step 1

Clean Audio Data

Decide size of data

Select features

Step 2

Convert Spectrogram

From Audio data to image data

Pickled spectrograms

Step 3

Run models

5 models

Step 4

Summary

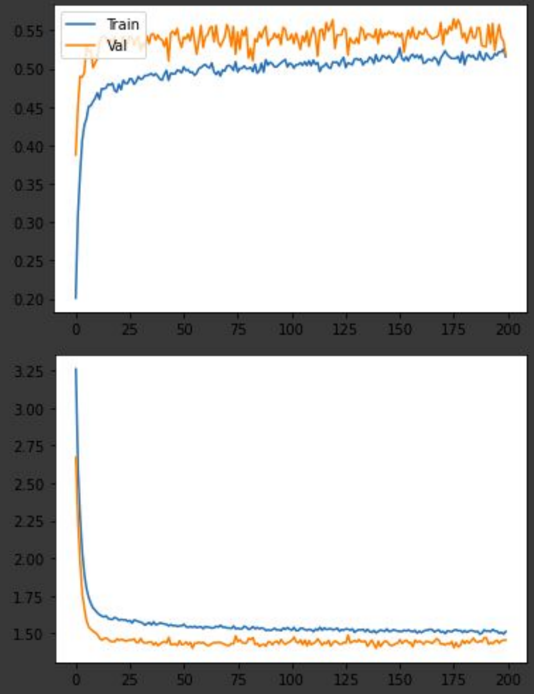
Evaluate our models

Compare with others

Model Architectures:

- Dense Network
- CNN with Augmentation
- VGG16
- VGG16 with Augmentation
- Progressive-resizing VGG16 with Augmentation

Dense Network from Features.csv (mfcc)

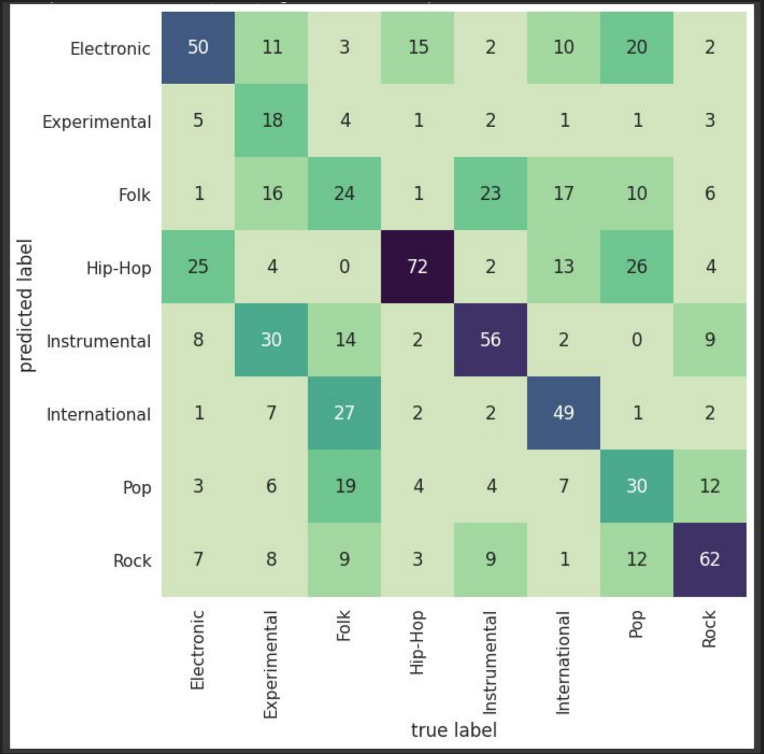



```
Train:
[1.3068895068764688, 0.5868750214576721]
Val:
[1.403822929859l614, 0.5649999976158142]
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 50)                7050
_________________________________________________________________
dropout_5 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_8 (Dense)              (None, 50)                2550
_________________________________________________________________
dropout_6 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_9 (Dense)              (None, 8)                 408
=================================================================
Total params: 10,008
Trainable params: 10,008
Non-trainable params: 0
```

Result

	precision	recall	f1-score	support
Electronic	0.44	0.50	0.47	100
Experimental	0.51	0.18	0.27	100
Folk	0.24	0.24	0.24	100
Hip-Hop	0.49	0.72	0.59	100
Instrumental	0.46	0.56	0.51	100
International	0.54	0.49	0.51	100
Pop	0.35	0.30	0.32	100
Rock	0.56	0.62	0.59	100
accuracy			0.45	800
macro avg	0.45	0.45	0.44	800
weighted avg	0.45	0.45	0.44	800



CNN

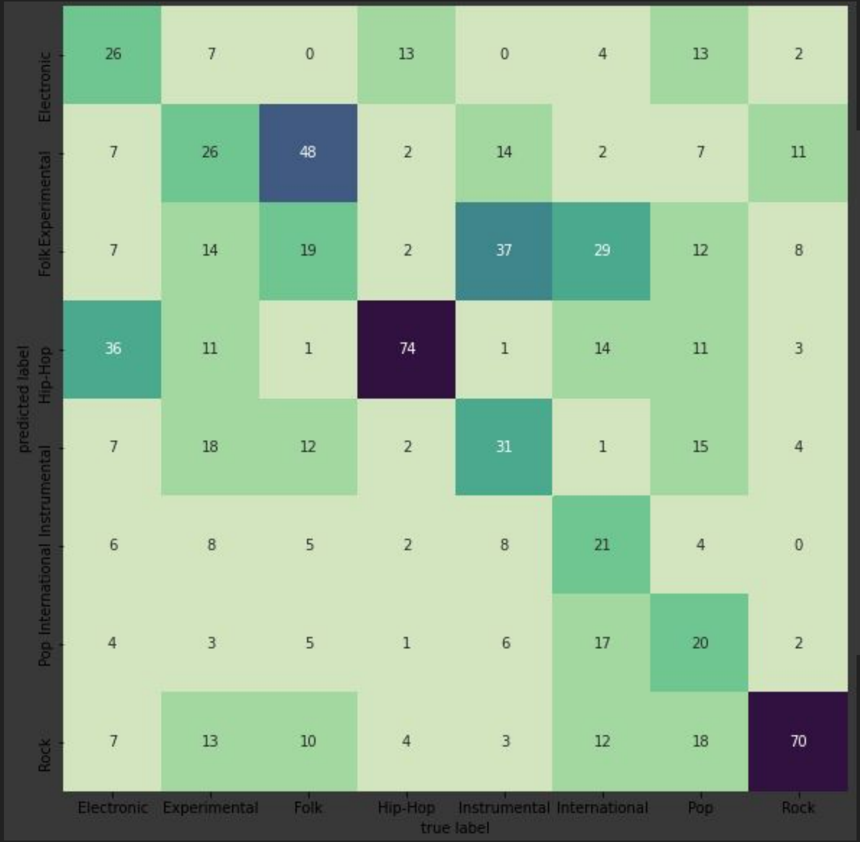
Architecture of CNN

```
activation='relu'
optimizer = keras.optimizers.Adam(lr=0.0001)
metrics=['categorical_accuracy']

model = keras.Sequential()
model.add(Conv2D(32, kernel_size=3, kernel_regularizer=keras.regularizers.l2(), strides=1, activation=activation, input_shape=(128, 400, 1)))
model.add(MaxPool2D(pool_size=(2,4)))
model.add(Conv2D(32, kernel_size=(3,5), kernel_regularizer=keras.regularizers.l2(), strides=1, activation=activation))
model.add(MaxPool2D(pool_size=(2,4)))
model.add(Dense(16, kernel_regularizer=keras.regularizers.l2(), activation=activation))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(8, activation='softmax'))
```

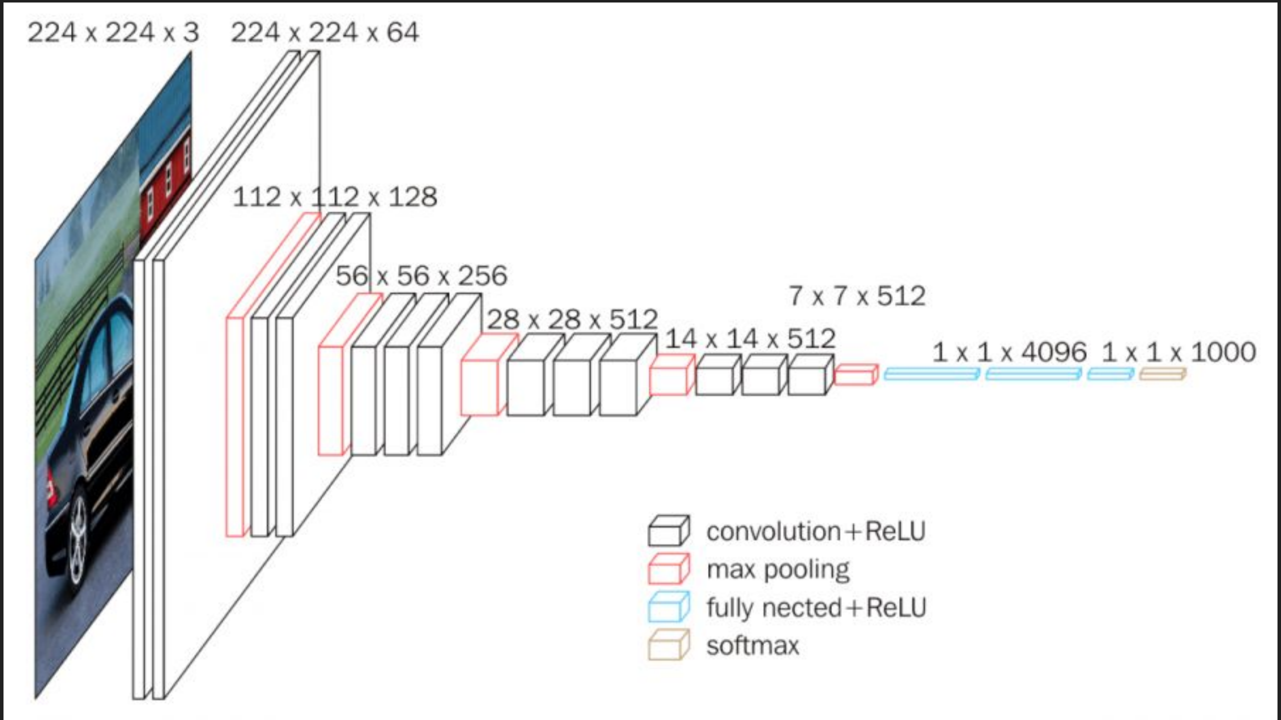
Validation set: 49.25%

CNN Results



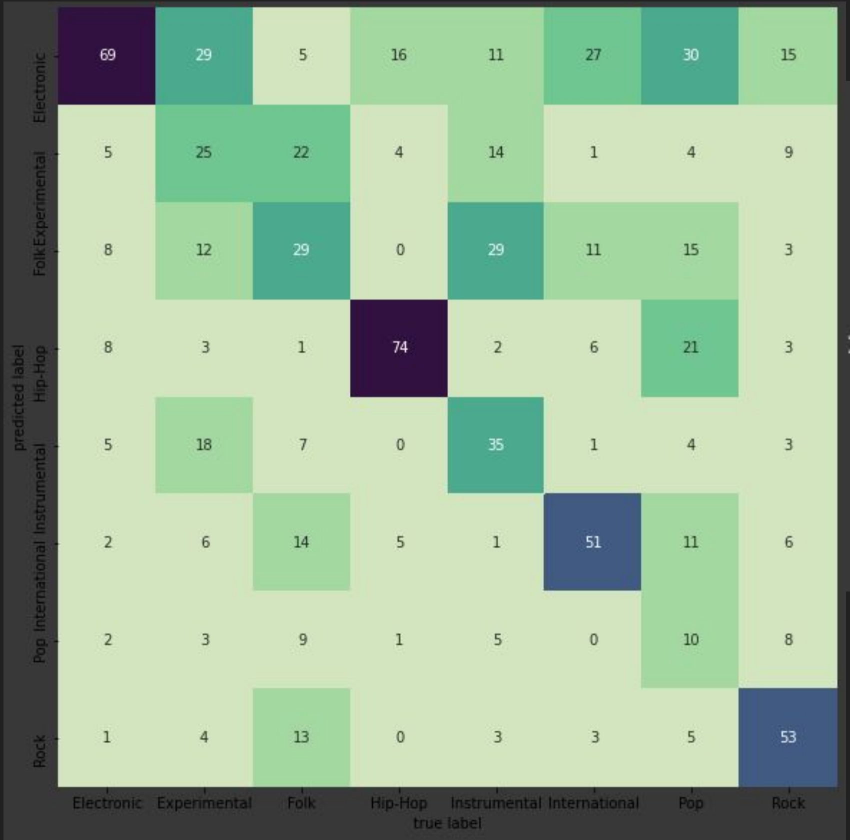
	precision	recall	f1-score	support
Electronic	0.40	0.26	0.32	100
Experimental	0.22	0.26	0.24	100
Folk	0.15	0.19	0.17	100
Hip-Hop	0.49	0.74	0.59	100
Instrumental	0.34	0.31	0.33	100
International	0.39	0.21	0.27	100
Pop	0.34	0.20	0.25	100
Rock	0.51	0.70	0.59	100
accuracy			0.36	800
macro avg	0.36	0.36	0.34	800
weighted avg	0.36	0.36	0.34	800

VGG16



Sources:
https://neurohive.io/en/popular-networks/vgg16/

VGG16 Test set with Augmentation



	precision	recall	f1-score	support
Electronic	0.34	0.69	0.46	100
Experimental	0.30	0.25	0.27	100
Folk	0.27	0.29	0.28	100
Hip-Hop	0.63	0.74	0.68	100
Instrumental	0.48	0.35	0.40	100
International	0.53	0.51	0.52	100
Pop	0.26	0.10	0.14	100
Rock	0.65	0.53	0.58	100
accuracy			0.43	800
macro avg	0.43	0.43	0.42	800
weighted avg	0.43	0.43	0.42	800

Progressive-resizing VGG16

1. 42x100

   Frozen weights: VGG16

   Flatten

   Dense(16)

   Dropout(0.3)

   Dense(8)

   On Validation reached 43%

Progressive-resizing VGG16 with Augmentation

1. 42x100

   Frozen weights: VGG16

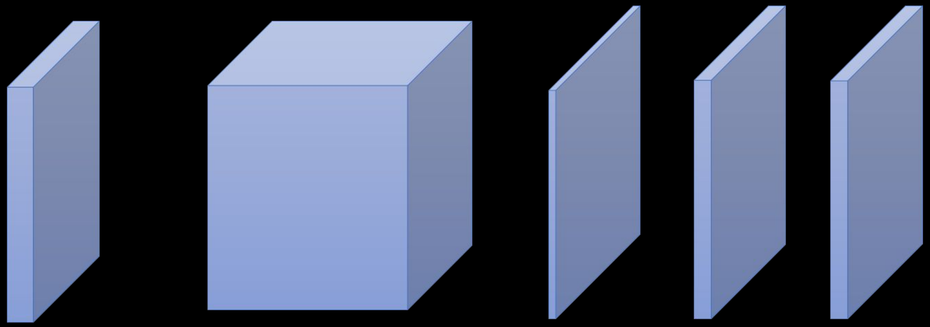
   Flatten

   Dense(16)

   Dropout(0.3)

   Dense(8)

   On Validation reached 43%

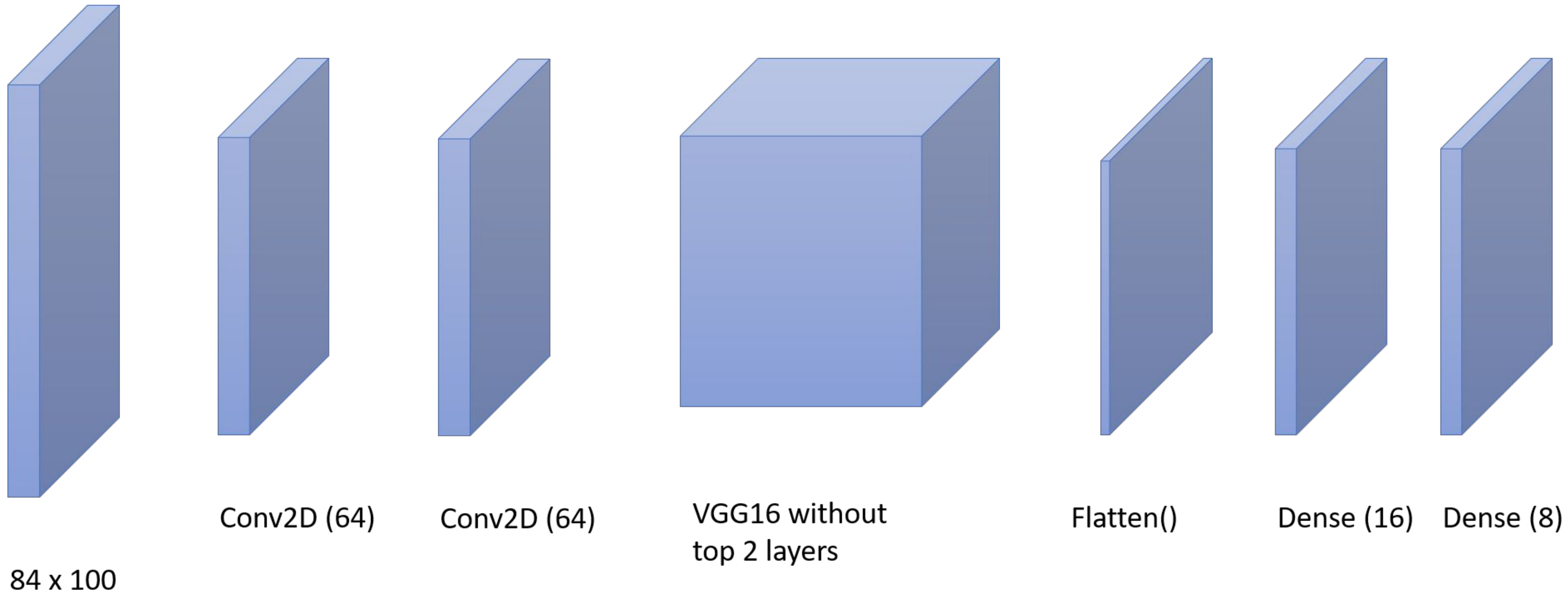


Progressive-resizing VGG16 with Augmentation

2. 84x100

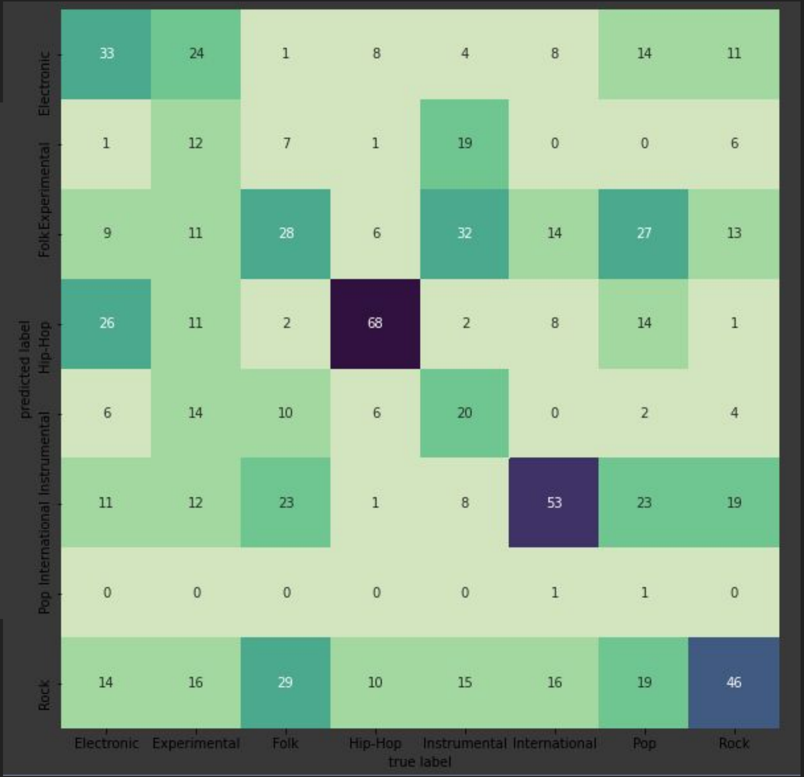
New Conv2D(64)	- Trainable
New Conv2D(64)	- Trainable
VGG16 layers without top 2 layers	- FROZEN
Flatten()	- PRETRAINED (frozen)
Dense(16)	- PRETRAINED (frozen)
Dropout(0.3)	- PRETRAINED (frozen)
Dense(8)	- PRETRAINED (frozen)

On Validation reached 40%


Conv2D (64)
Conv2D (64)
VGG16 without top 2 layers
Flatten()
Dense (16)
Dense (8)
84 x 100

Progressive VGG16 with Augmentation Test set

	precision	recall	f1-score	support
Electronic	0.32	0.33	0.33	100
Experimental	0.26	0.12	0.16	100
Folk	0.20	0.28	0.23	100
Hip-Hop	0.52	0.68	0.59	100
Instrumental	0.32	0.20	0.25	100
International	0.35	0.53	0.42	100
Pop	0.50	0.01	0.02	100
Rock	0.28	0.46	0.35	100
accuracy			0.33	800
macro avg	0.34	0.33	0.29	800
weighted avg	0.34	0.33	0.29	800



Performance Summary

Author	Model	Accuracy	F1 Score
Priya Dwivedi	CRNN	0.44125	0.44
Priya Dwivedi	CNN_RNN_parallel	0.44375	0.44
The FMA author	Baseline	0.12	0.13
Ours	Dense	**0.45**	**0.44**
Ours	CNN with augmentation	0.36	0.34
Ours	VGG16 with augmentation	0.43	0.42
Ours	Progressive VGG16 with augmentation	0.33	0.29

Results

- Deep learning models can extract useful features from mel-spectrograms
  - input - spectrogram images

- Deep learning models do not seem to perform better than baseline models using MFCC features
  - input - audio features (mfcc)

Achievements

- We learned how to preprocess audio data
  - Understood audio features (mfcc)
  - Converted the mel-spectrograms into npz, pickle files
- We conducted many interesting models
  - More than 40 models with various hyperparameters and architectures
  - Researched state of the art models to apply our problem
- We became skillful for tools and environments to run deep learning models
  - How to collaborate with team members on Github

Challenges / Discussion

- Computing power
  - Not enough memory to handle big size image data (>25 GB)
  - About 40 minutes for one epoch
  - More storage to use a larger dataset (22GB, 93GB, and 879 GB)
    - Solution may be cloud computing such as AWS
- Need more data
  - Insufficient sample size → 1000 samples per genre is still a small sample
  - Low test accuracy for more robust models
    - May be due to the limited dataset (8,000 audio tracks)
- Music Genre Recognition (MGR)
  - Interplay of cultures, artists, and market
  - Boundaries between genres still remain fuzzy

Github - Commits

April 14, 2020 – May 14, 2020

Period: 1 month

Overview

7 Active Pull Requests

0 Active Issues

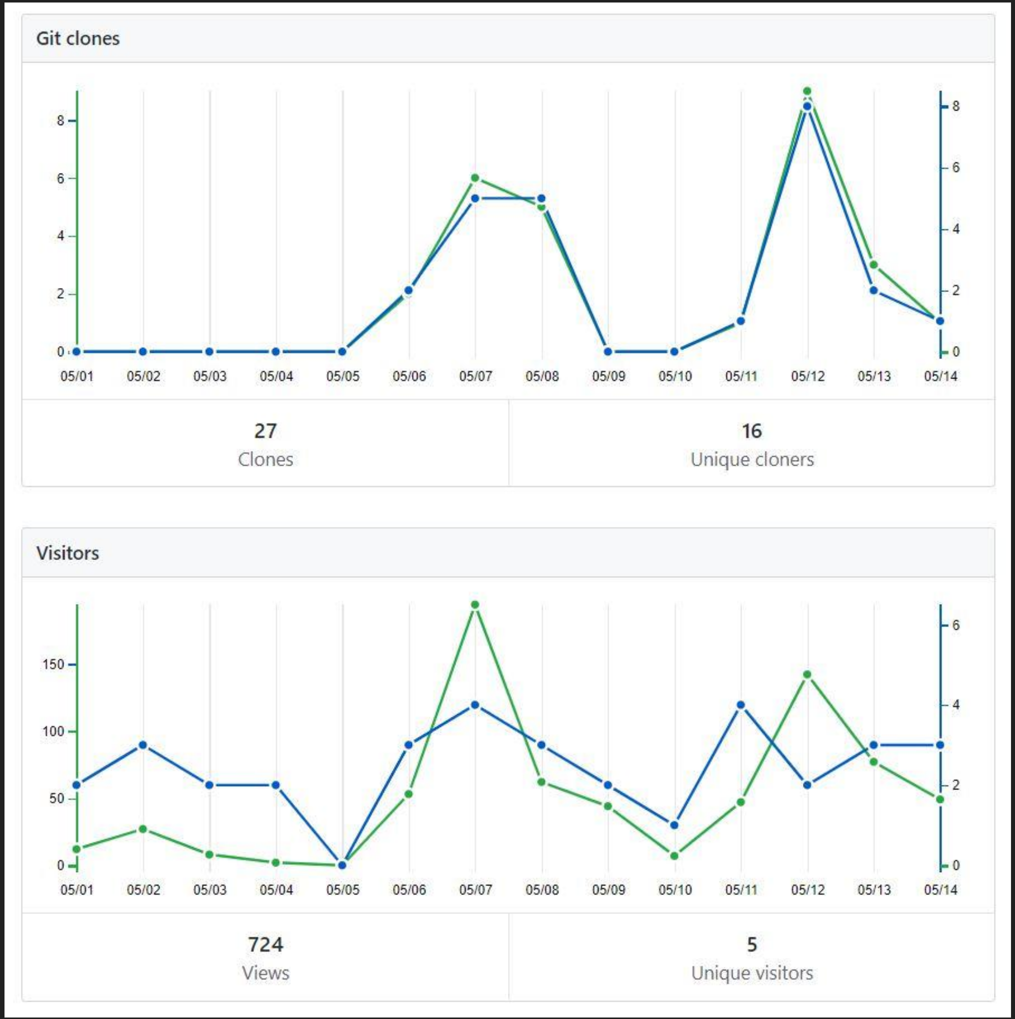
7	0	0	0
Merged Pull Requests	Proposed Pull Requests	Closed Issues	New Issues

Excluding merges, **4 authors** have pushed **50 commits** to master and **50 commits** to all branches. On master, **0 files** have changed and there have been **0 additions** and **0 deletions**.

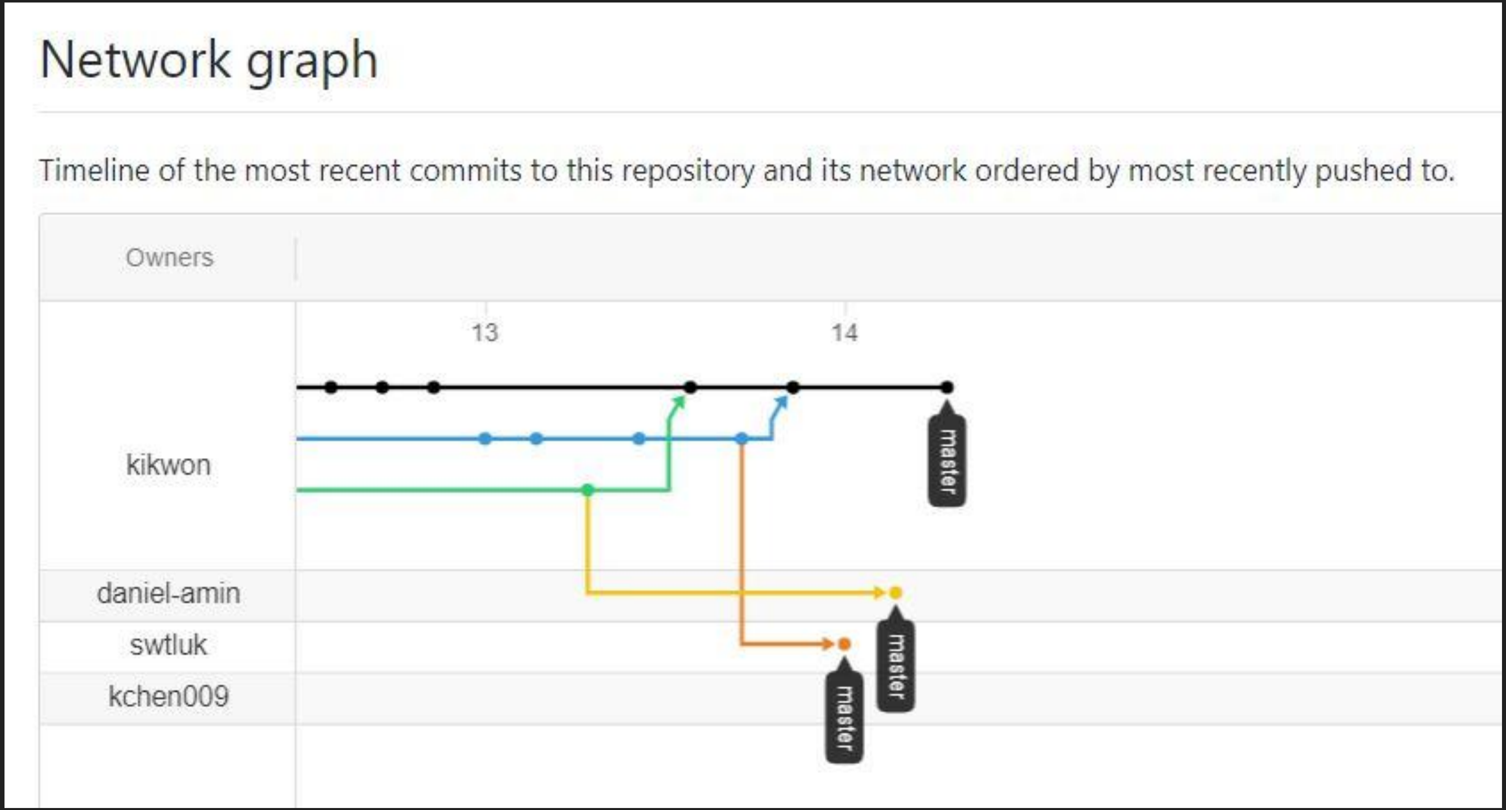
Github - Traffic

Green - total

Blue - unique



Github - Network Graph



References

- Bahuleyan, H., 2020. Music Genre Classification Using Machine Learning Techniques. [online] arXiv.org. Available at: <https://arxiv.org/abs/1804.01149> [Accessed 13 May 2020].
- Bilogur, Aleksey. 2019. Boost your CNN image classifier performance with progressive resizing in Keras. [online]. Available at: <https://towardsdatascience.com/boost-your-cnn-image-classifier-performance-with-progressive-resizing-in-keras-a7d96da06e20> [Accessed 14 May 2020]
- Defferrard, M., Benzi, K., Vandergheynst, P. and Bresson, X., 2020. FMA: A Dataset For Music Analysis. [online] arXiv.org. Available at: <https://arxiv.org/abs/1612.01840> [Accessed 23 April 2020]
- Dong, Mingwen. 2018. Convolutional Neural Network Achieves Human-level Accuracy in Music Genre Classification. [online] arXiv.org. Available at: <https://arxiv.org/pdf/1802.09697.pdf> [Accessed 30 April 2020]
- Dwivedi, P. Deep Learning for Music Genre Recognition.[online] Available at: <https://github.com/priya-dwivedi/Music_Genre_Classification> [Accessed 12 May 2020]

Thank you