Genre Classification

Tyler Perkins

Problem Description

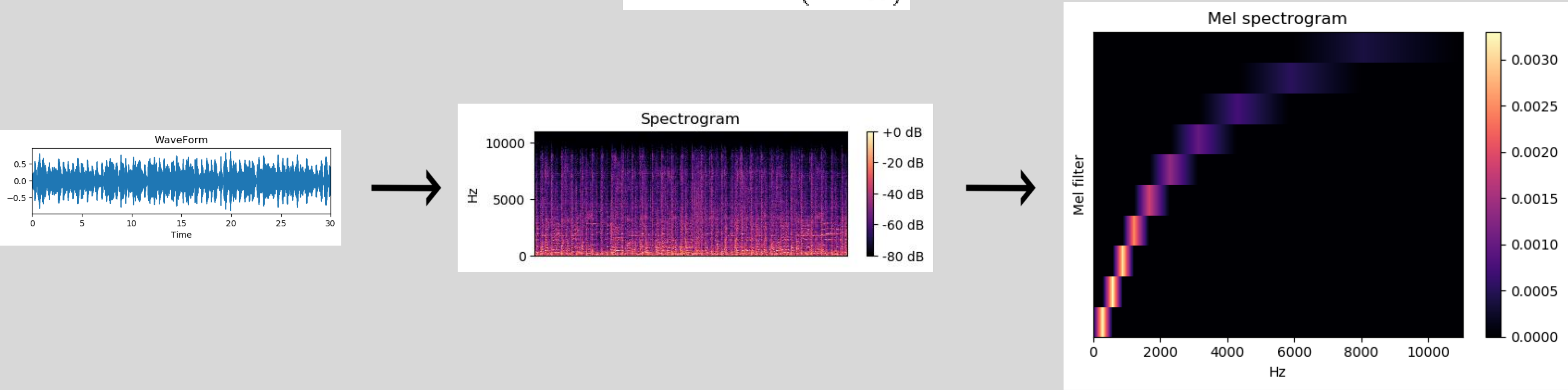
- Being able to classify music by genre is an important feature of any music streaming service
- Whenever new music is uploaded, an algorithm can determine the genre automatically
- This allows music to be organized and discovered more easily

Methods and algorithms

- Using GTZAN genre collection (http://marsyas.info/downloads/datasets.html)
  - Contains 1000 song samples from 10 genres
- Convert audio samples to melspectrograms for features
- Prediction:
  - K-Nearest Neighbor
  - Convolutional Neural Network

Melspectrograms

- Uses librosa package
- Spectrogram: 2D representation of audio using frequency over time
- Mel-Scale: Non-Linear transformation of frequency scale, converts frequency in to “mels”
  - No set formula, something in the form $m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$

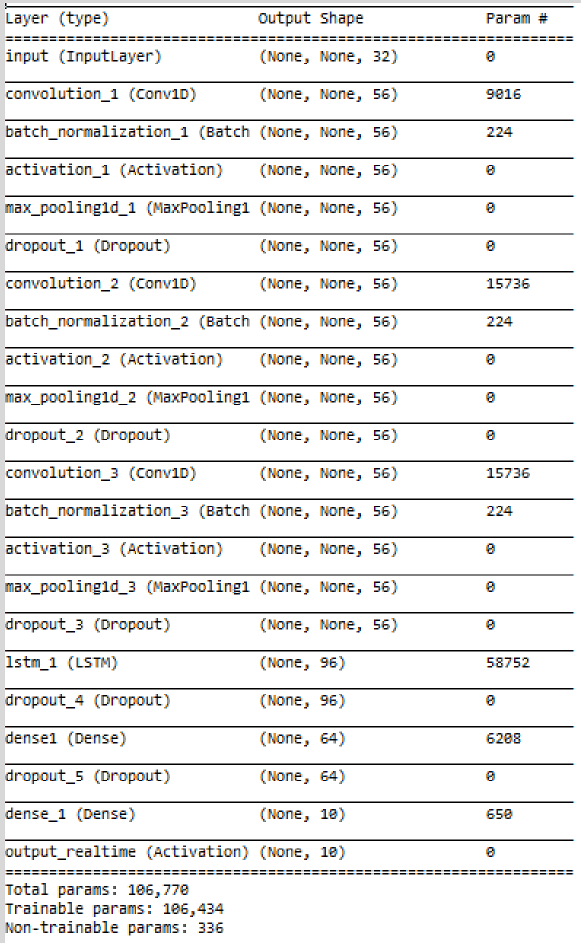


K-Nearest Neighbor

- Supervised Algorithm
- KNN algorithm assumes that similar things exist in close proximity
- Compares distance between features to classify data
- Using scikit learn KNeighborsClassifier

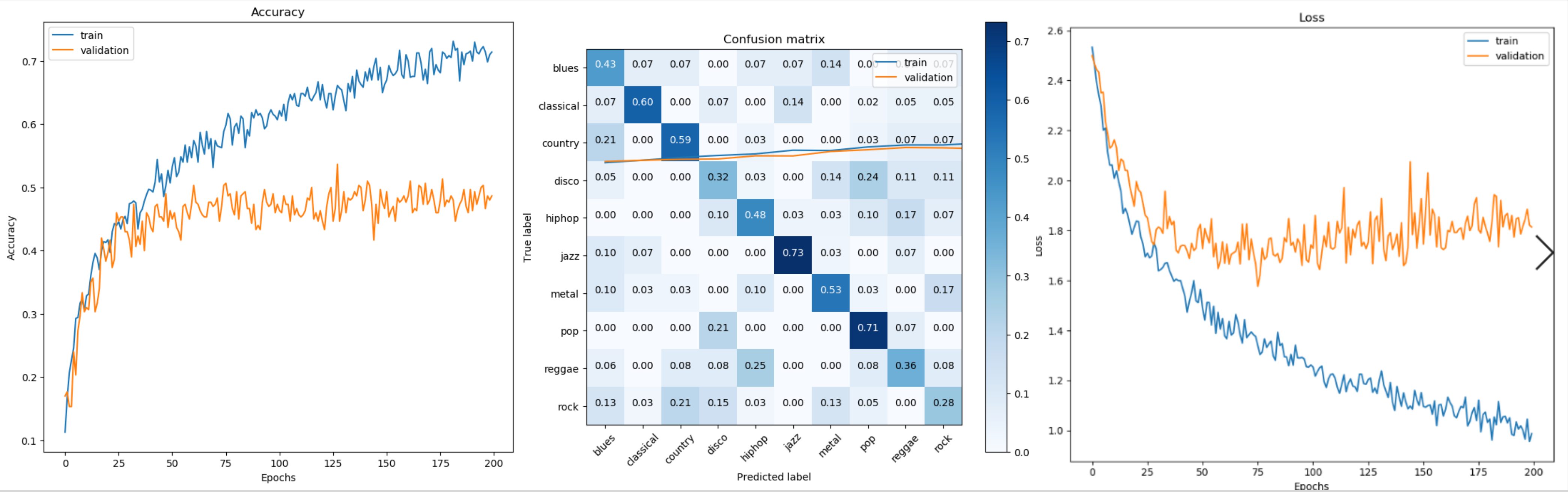
Convolutional Neural Network

- 3 Conv1D layers
- LSTM Layer
- Dense Layer
- Softmax
- Optimizer: Adam with learning rate of .001
- Loss: Categorical Cross Entropy
- 200 epochs



```
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           (None, None, 32)          0
_________________________________________________________________
convolution_1 (Conv1D)       (None, None, 56)          9016
_________________________________________________________________
batch_normalization_1 (Batch (None, None, 56)          224
_________________________________________________________________
activation_1 (Activation)    (None, None, 56)          0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, None, 56)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, None, 56)          0
_________________________________________________________________
convolution_2 (Conv1D)       (None, None, 56)          15736
_________________________________________________________________
batch_normalization_2 (Batch (None, None, 56)          224
_________________________________________________________________
activation_2 (Activation)    (None, None, 56)          0
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, None, 56)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, None, 56)          0
_________________________________________________________________
convolution_3 (Conv1D)       (None, None, 56)          15736
_________________________________________________________________
batch_normalization_3 (Batch (None, None, 56)          224
_________________________________________________________________
activation_3 (Activation)    (None, None, 56)          0
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, None, 56)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, None, 56)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 96)                58752
_________________________________________________________________
dropout_4 (Dropout)          (None, 96)                0
_________________________________________________________________
dense1 (Dense)               (None, 64)                6208
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
_________________________________________________________________
output_realtime (Activation) (None, 10)                0
=================================================================
Total params: 106,770
Trainable params: 106,434
Non-trainable params: 336
```

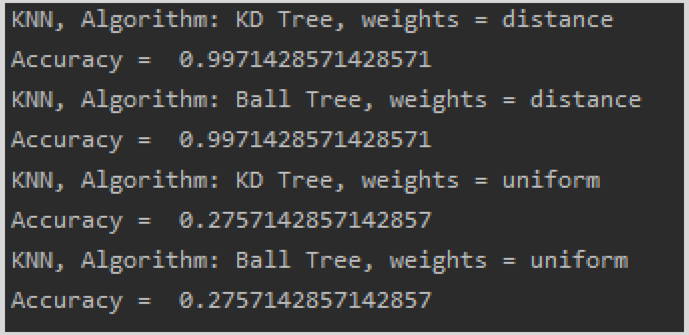
CNN Results



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.43	0.07	0.07	0.00	0.07	0.07	0.14	0.00	[illegible]	0.07
classical	0.07	0.60	0.00	0.07	0.00	0.14	0.00	0.02	0.05	0.05
country	0.21	0.00	0.59	0.00	0.03	0.00	0.00	0.03	0.07	0.07
disco	0.05	0.00	0.00	0.32	0.03	0.00	0.14	0.24	0.11	0.11
hiphop	0.00	0.00	0.00	0.10	0.48	0.03	0.03	0.10	0.17	0.07
jazz	0.10	0.07	0.00	0.00	0.00	0.73	0.03	0.00	0.07	0.00
metal	0.10	0.03	0.03	0.00	0.10	0.00	0.53	0.03	0.00	0.17
pop	0.00	0.00	0.00	0.21	0.00	0.00	0.00	0.71	0.07	0.00
reggae	0.06	0.00	0.08	0.08	0.25	0.00	0.00	0.08	0.36	0.08
rock	0.13	0.03	0.21	0.15	0.03	0.00	0.13	0.05	0.00	0.28

kNN Results

◦ Compared 2 algorithms for computing nearest neighbor and 2 types of weights



```
KNN, Algorithm: KD Tree, weights = distance
Accuracy =  0.9971428571428571
KNN, Algorithm: Ball Tree, weights = distance
Accuracy =  0.9971428571428571
KNN, Algorithm: KD Tree, weights = uniform
Accuracy =  0.2757142857142857
KNN, Algorithm: Ball Tree, weights = uniform
Accuracy =  0.2757142857142857
```