Deep-learning: music genre recognition

19th October 2018

Gregoire Virepinte, Margaux Wehr, Zhijian Xu, Yuhua Qiao, Ran Huo

Aim – Learnings & Challenges

Classify mp3 tracks into their respective genres

CNN will learn filters that extract features in the frequency and time domain

Learnings

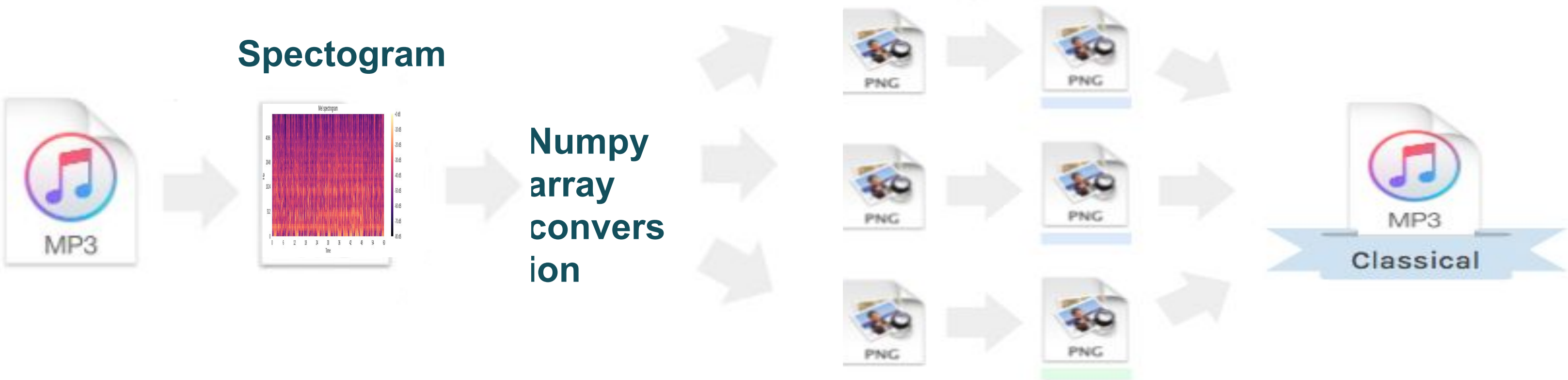
- Converting audio files into visual input
- Pytorch

Challenges

- Large amount of data pre-processing
- Dealing with heavy input size (mp3 format)
- Boundaries between genres can be fuzzy

Overall methodology

Main challenge: transforming audio into usable input for CNN



Data collection

Spectogram

Numpy array convers ion

Slice

Classify using CNN

Vote

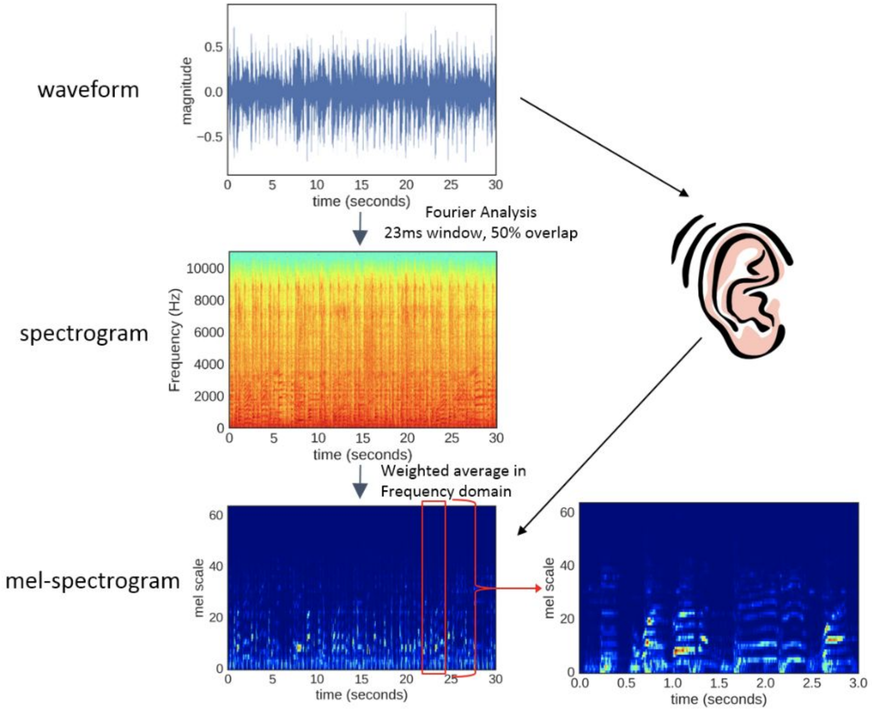
Answer: Transform audio signal into visual output

Converting audio to image recognition

- A spectrogram is a space, frequency and amplitude representation. It's commonly used to visualize these three attributes of a signal

- **'Divide and conquer':** split spectogram into 3s segments

- We use **mel-spectrogram** as the input to the **CNN**



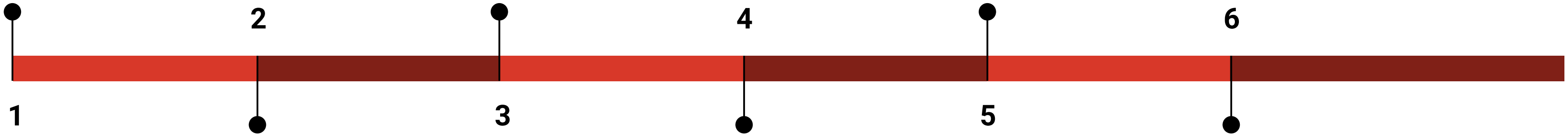
Source: *Convolutional Neural Network Achieves Human-level Accuracy in Music Genre Classification*, Mingwen Dong

Data collection

- 106,574 of Creative Commons-licenced tracks from 16,341 artists and 14,854 album
- Dataset: **8,000 30-second clips from Free Music Archive.** The clips belong to 8 top genres, balanced with 1,000 clips per genre.
- Our goal in this notebook will be t**o classify the genre of an unknown track among the 8 genres that are present in our dataset.**

Data source: https://github.com/mdeff/fma

Data pre-processing



1

Load the data in subset

2

Link each track to a genre, avoid subgenres

Rap
Rap
Hip-Hop thug rap

3

Sample 1000 tracks out of the 8000 tracks to reduce computational power required

4

Convert mp3 file to spectogram

5

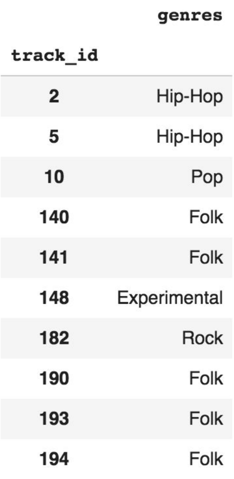
Save as numpy array to txt file for each track

6

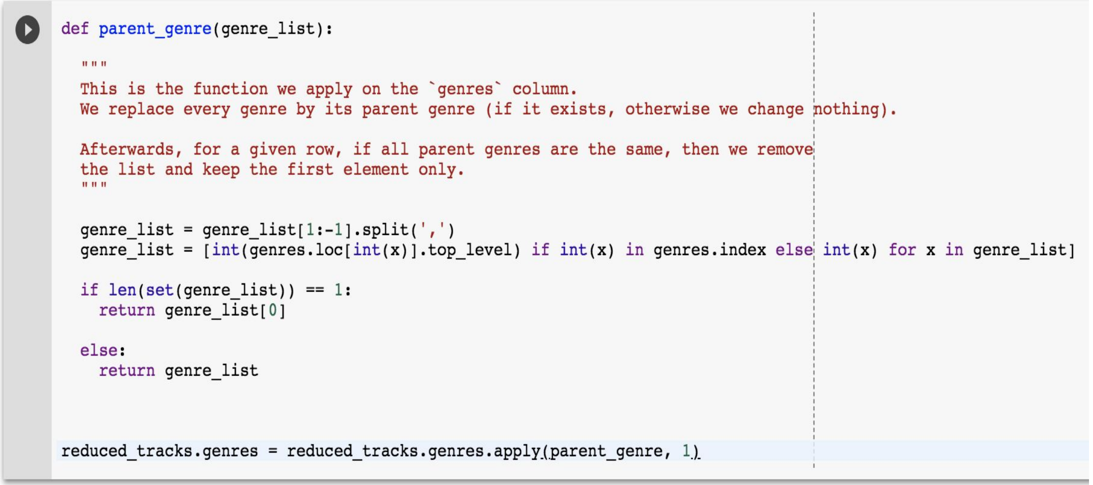
Slice numpy array to convert each 30s clip into 10 clips of 3s

Deep-dive: finding the right genre

- Some songs have **several genres** associated to them
- In the end, the goal of this section was to only keep the "**parent**" genre, which is supposed to be unique, for every track we have in our dataset

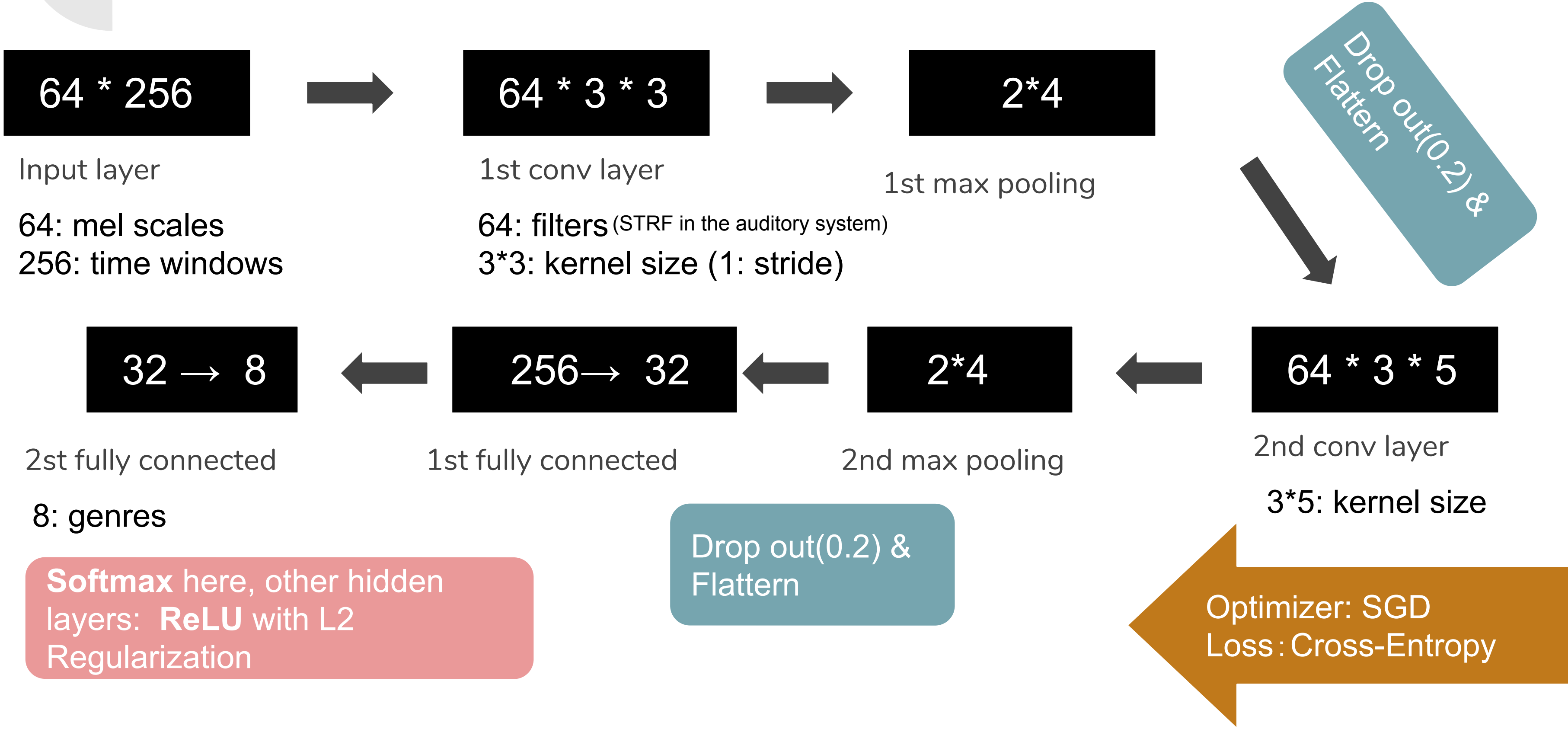


	genres
track_id	
2	Hip-Hop
5	Hip-Hop
10	Pop
140	Folk
141	Folk
148	Experimental
182	Rock
190	Folk
193	Folk
194	Folk



```
def parent_genre(genre_list):

  """
  This is the function we apply on the `genres` column.
  We replace every genre by its parent genre (if it exists, otherwise we change nothing).

  Afterwards, for a given row, if all parent genres are the same, then we remove
  the list and keep the first element only.
  """

  genre_list = genre_list[1:-1].split(',')
  genre_list = [int(genres.loc[int(x)].top_level) if int(x) in genres.index else int(x) for x in genre_list]

  if len(set(genre_list)) == 1:
    return genre_list[0]

  else:
    return genre_list



reduced_tracks.genres = reduced_tracks.genres.apply(parent_genre, 1)
```

CNN: Network Architecture



64 * 256

Input layer

64: mel scales
256: time windows

64 * 3 * 3

1st conv layer

64: filters (STRF in the auditory system)
3*3: kernel size (1: stride)

2*4

1st max pooling

Drop out(0.2) &
Flattern

64 * 3 * 5

2nd conv layer

3*5: kernel size

2*4

2nd max pooling

256→ 32

1st fully connected

Drop out(0.2) &
Flattern

32 → 8

2st fully connected

8: genres

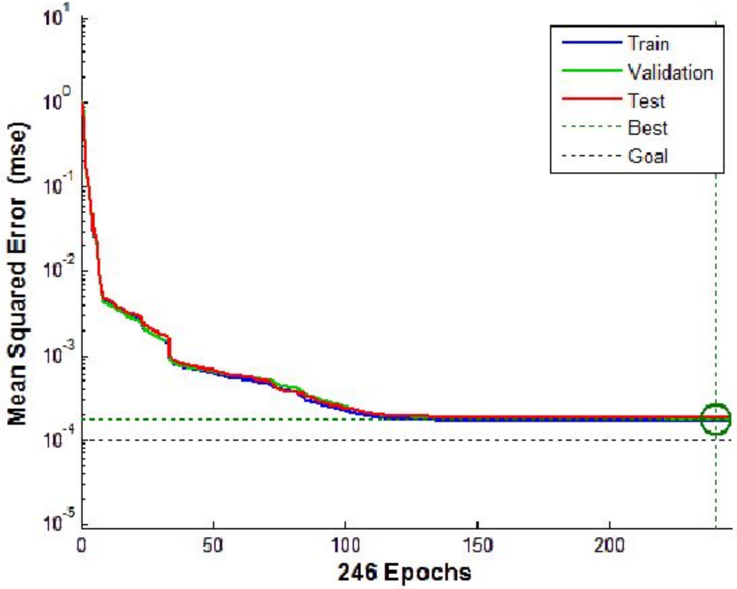
Softmax here, other hidden layers: **ReLU** with L2 Regularization

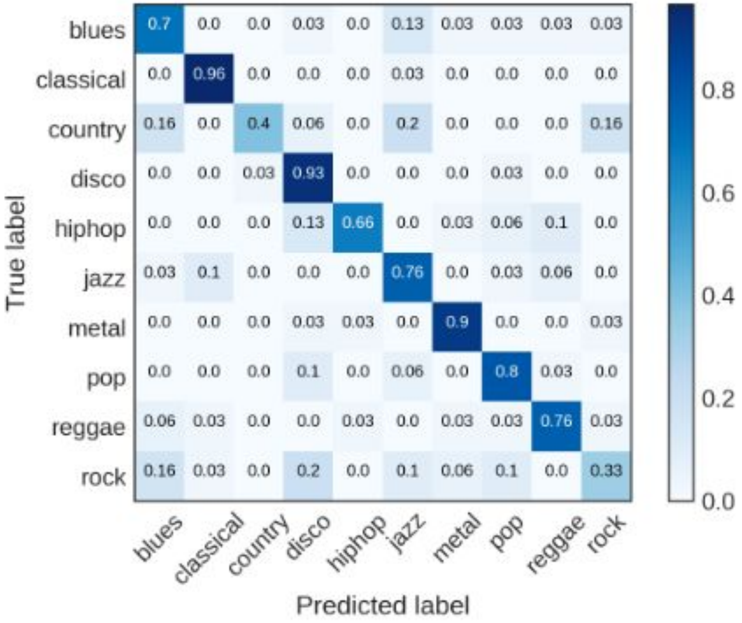
Optimizer: SGD
Loss : Cross-Entropy

CNN: Train and Prediction

We set training set, validation set, test set = 7:2:1

Output for the time being:





	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.7	0.0	0.0	0.03	0.0	0.13	0.03	0.03	0.03	0.03
classical	0.0	0.96	0.0	0.0	0.0	0.03	0.0	0.0	0.0	0.0
country	0.16	0.0	0.4	0.06	0.0	0.2	0.0	0.0	0.0	0.16
disco	0.0	0.0	0.03	0.93	0.0	0.0	0.0	0.03	0.0	0.0
hiphop	0.0	0.0	0.0	0.13	0.66	0.0	0.03	0.06	0.1	0.0
jazz	0.03	0.1	0.0	0.0	0.0	0.76	0.0	0.03	0.06	0.0
metal	0.0	0.0	0.0	0.03	0.03	0.0	0.9	0.0	0.0	0.03
pop	0.0	0.0	0.0	0.1	0.0	0.06	0.0	0.8	0.03	0.0
reggae	0.06	0.03	0.0	0.0	0.03	0.0	0.03	0.03	0.76	0.03
rock	0.16	0.03	0.0	0.2	0.0	0.1	0.06	0.1	0.0	0.33

Improvement

1

Use more data

Run the CNN on the whole dataset, instead of the reduced one

2

Add validation set

Repeat the procedure until classification accuracy on the cross-validation data set doesn’t improve anymore

3

Combine the prediction

Combine 10 predictions ot 10 3-sec segments of one song → probability

Deep learning vs traditional methods

CNN

- Widely used for image classification
- Non-parametric
- No need to input features
- Hard to interpret the model

Random Forest

- "Worry-free" approach, no real hyperparameters to tune
- Non-parametric
- Limit instability of single trees by averaging predictions, limit overfitting
- View feature importance

Train set accuracy	98%
Test set accuracy	51%