ÖZYEĞİN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

CS 402

2018 Spring

SENIOR PROJECT REPORT

Prediction of popular song success by machine learning

By

Ilayda Gürbak

Supervised By

Melih Kandemir

Declaration of Own Work Statement/ (Plagiarism Statement)

Hereby I confirm that all this work is original and my own. I have clearly referenced/listed all sources as appropriate and given the sources of all pictures, data etc. that are not my own. I have not made any use of the essay(s) or other work of any other student(s) either past or present, at this or any other educational institution. I also declare that this project has not previously been submitted for assessment in any other course, degree or qualification at this or any other educational institution.

Student Name and Surname: Ilayda Gurbak

Signature:

Place, Date:

Abstract

Hit song prediction is one of the popular engineering problems that have been working on in recent years. To predict whether is song will be popular before its launched has important role in music industry. With this information music producers can evaluate their work and learn about important factors, which makes a song more popular in the music market. In this paper, audio- based music dataset is created from official videos on YouTube and they were stored in different popularity categories (low-medium-high) depending on their corresponding view counts. For extracting audio features Mel Frequency Cepstral Coefficient (MFCC) were used in this project. In first part of this project, Convolutional Neural Network (CNN) was trained to achieve image classification problem and for the second part of this project LSTM Neural Network was implemented to achieve image classification.

Contents

I. Introduction

Hit song science is a very popular area for researches in recent years. Everyone wants to know can we predict success of a song before it is released to the music market. Certainly, we cannot be sure if there is a certain pattern or way of compose music that surely makes song a hit without taking closer look to features of an audio. As we know, other things such as commercials and popularity of artist, music video features also taking role of making a song more popular in modern world but in this paper my main focus is only about investigating a relation between audio features of song and it's popularity on one of the most used channel for music, which is YouTube.

If we can get good prediction tool implemented for this issue, it would be beneficial for music lovers and also for the producers.

Since music taste is evolving I wanted to use keep my data more up-to-date. I picked songs from YouTube, which released official music video 2014 – 2017 and also tried to not use song that was too newly released (it would not reach it's peak count yet).

In total, 150 songs were classified in three different categories as following:

Low (0 - million), Medium (million – 100 million), High (100 million+)

To implement my project, I used some python libraries, which are torch, torchvision, matplotlib, librosa, pylab. I used PyTorch machine learning library for my project.

In next part, I will introduce techniques and tools that I used for my project.

II. Background

In this section set of techniques, tools, technologies that are utilized my project can be found.

PyTorch:

PyTorch a python package that provides two high-level features: Tensor computation (like numpy) with strong GPU acceleration.

LibROSA:

I used this package for audio analysis. This library helps to process audio and extract features easily. When I need to convert file type I used this library and also easily extracted duration of song, Fourier transformation in my project. Also it has many other features for audio datasets.

Torch:

Torch is a scientific computing framework with wide support for machine learning algorithms. The goal of Torch is to have maximum flexibility and speed in building machine learning algorithms while making this process simple.

Torchvision:

It is a package consisting popular datasets (cifar10, MNIST etc.), model architectures and common image transformations for computer vision. I used it for visualizing my images.

Matplotlib:

Matplotlib is a library for making 2D plots of arrays in Python.

Python:

I used Python programming language as language of this project.

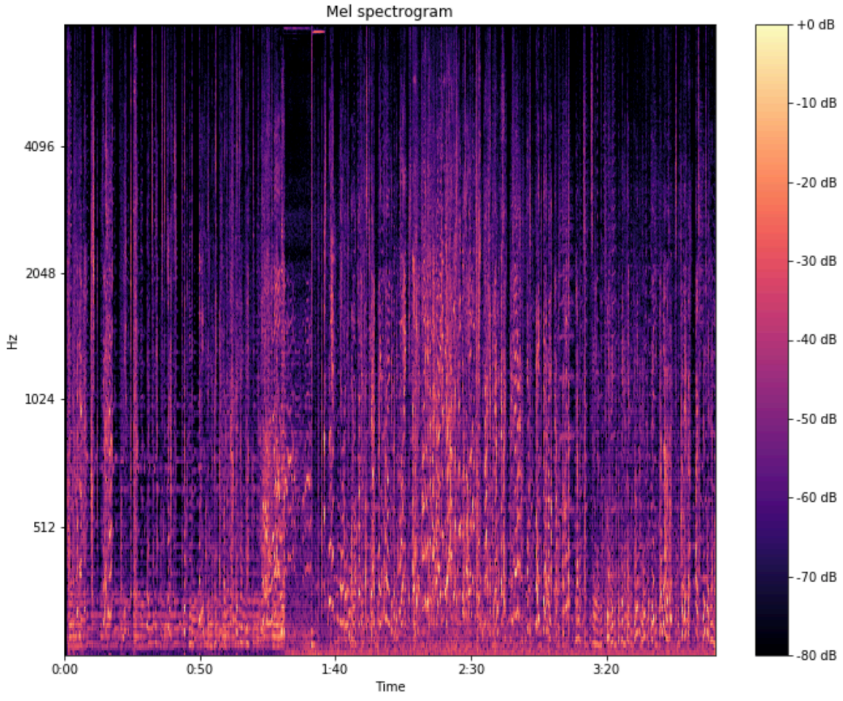
Problem Statement

- PROJECT SCOPE

This project wasn't unexplored area, yet I have started implementing from scratch, I did not contribute as part of other project. My supervisor, Melih Kandemir, suggested this project. After I read papers about this problem, I came up with solution approach by his suggestions. In first part of the project I worked on creating my dataset and CNN implementation and in second part I worked on LSTM implementation.

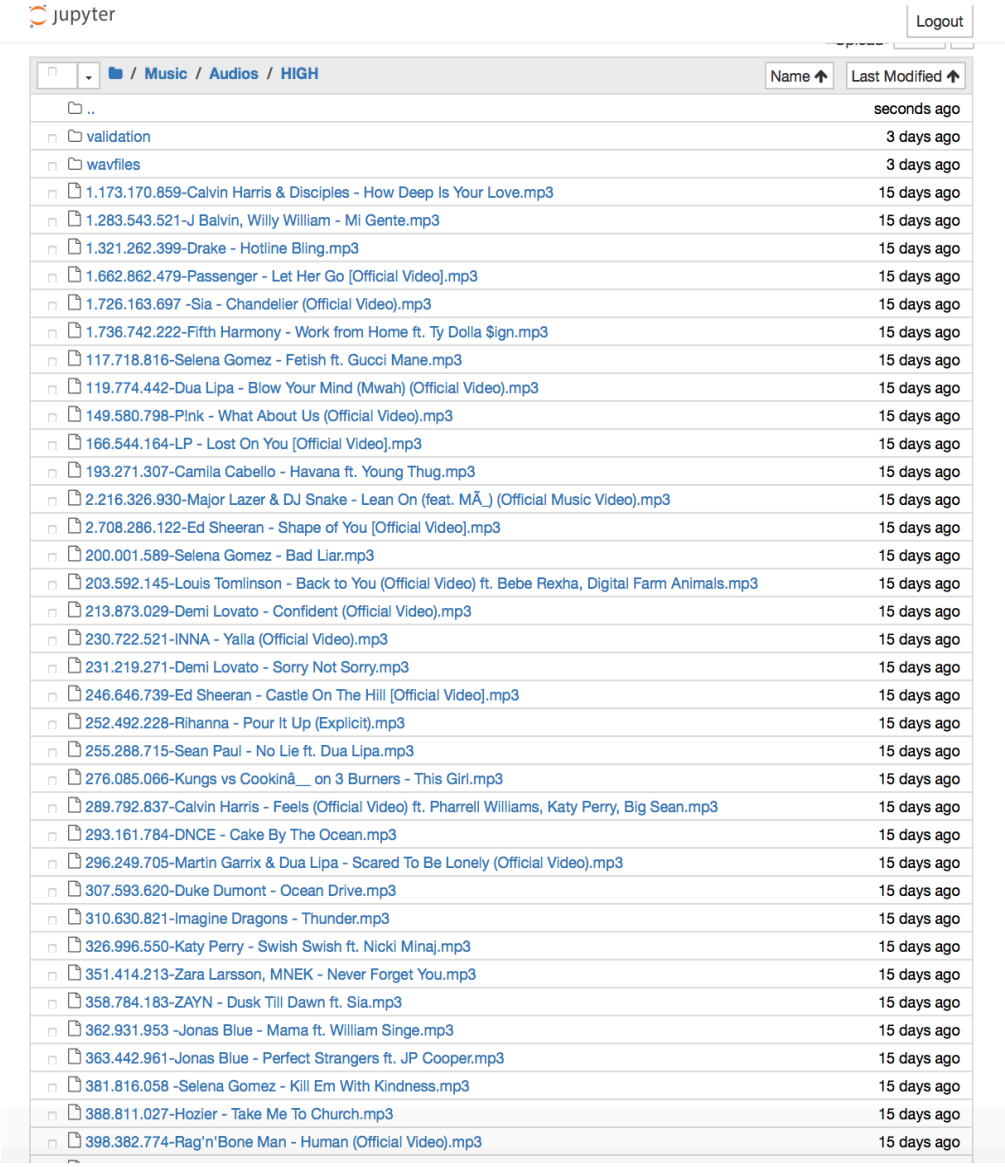
- ENGINEERING PROBLEM

Engineering problems to be solved for this problem were using music files as manageable data, extracting important features then training a classifier to solve the problem. First question was how should I use my audio data? To deal with audio data libROSA was suitable library. Firstly, I tried to implement algorithm that converts audio to image, which plots dB-kHz but my samples looked really similar to each other so that I looked for another way to represent my data. Then I learned about Mel-spectrograms. I plotted my graphs as (x=time(s), y=frequency(Hz)) .



You can see above, MFCC of the song "Camila Cabello- Havana" is from high popularity list.

After preparing my dataset, I learned about how to use data load. I created 3 classes for my dataset, which are “low-medium-high”. For low, I picked songs that have view count 0-million. For medium, I picked songs that have view count million-100 million. For high, I picked songs that have view more than 100 million. Then I stored them as follows.



- ASSUMPTIONS&CONSTRAINTS

Firstly, I assumed I could extract meaningful features from audio data. Secondly, I assumed libraries would work well with the Python version, which I installed on my laptop. Thirdly, I assumed CPU power would be enough for my dataset so that I did not use CUDA for running my neural net on GPU.

Constraints, data that I have used were really limited for training neural network.

III. Solution Approach

- TOOLS AND TECHNIQUES

I used PyTorch framework, python as programming language for implementation. As solution of this problem firstly, I prepared my own music dataset. I downloaded 50 songs per three popularity categories that I have created. Three classes were “low”, “medium” and “high”. For each class I separated 45 songs for training and 5 songs for testing data. Then I extracted MFCC of every song and stored them in these categories. After that, I treated my problem as image classification problem. I followed PyTorch tutorials to learn how to create convolutional neural network and train it for my problem. And in second part, I used LSTM neural network for image classification.
To use my image dataset I transformed it to tensor (transforms.Totensor()), scaled my images and also normalized them. I loaded my dataset by dataloader.

```
transform = transforms.Compose([
            transforms.Grayscale(num_output_channels=1),
            transforms.Resize((28, 28)),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
```

```
trainset = dset.ImageFolder(root="/Users/ilayda/Music/data/train_set/",transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=100,shuffle=True,  num_workers=1)

testset = dset.ImageFolder(root='/Users/ilayda/Music/data/test_set/',transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=100,shuffle=True,     num_workers=1)
```

```
classes=('low','medium','high')
```

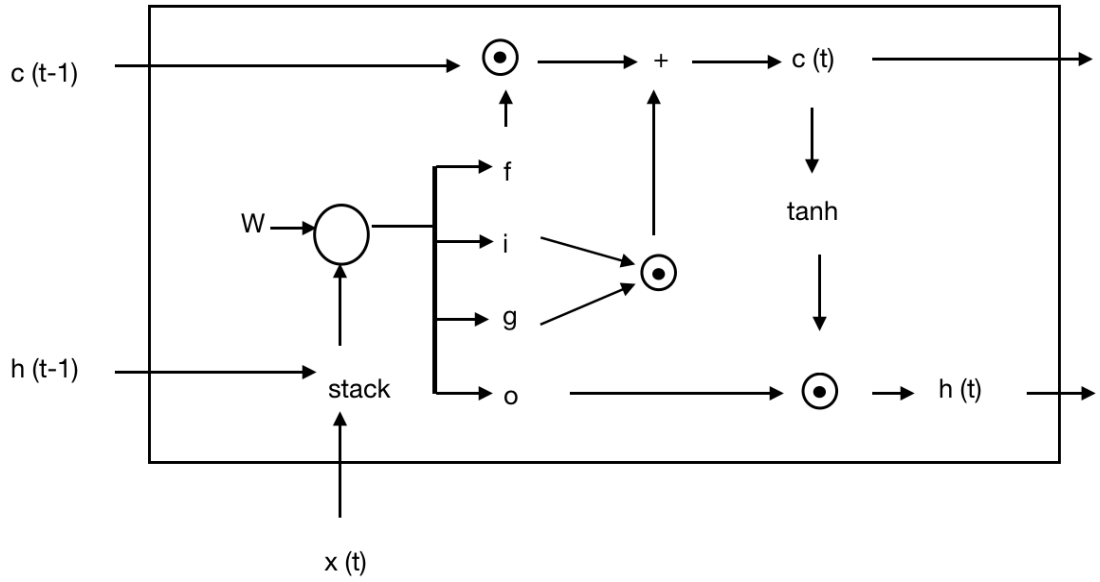
Then used matplotlib and torchvision for plotting images on my notebook.
I defined convolution neural network (CNN), defined loss function and optimizer and trained the network by looping over on data iterator and fed the inputs into network then optimize them. After training I loaded test data and printed accuracy of the network as first part of my project.

In second part, I have implemented 2 layers LSTM neural network. Trained my network with 100 epochs. I split my dataset into training and testing by 45 to 5. In first time the accuracy of my model was 26%, in second time it was 45% and in third it reached up to 60% on 15 images test dataset.

Each time when I was splitting my data, I picked most distant ones (depending on their view count) when it was most distant on test dataset, in third time, I got the best accuracy.

LSTM is a type of RNN and used for time series. In my project, I used it for image classification problem. My data was spectrogram of an audio, 28*28. Every time LSTM model was reading a row, as it is a time series, after reading through all rows (for 28 times) prediction was made. It concludes which class belongs to this picture.

LSTM CELL



- KNOWLEDGE AND SKILL SET
  - CS 102
  - CS 201
  - CS 302
  - CS 466 (Intro to Deep Learning)

In addition to these courses, I have read lecture notes of Stanford University CS231 and also follow lecture videos.

- ENGINEERING STANDARDS
  - PEP-8 while using Python.

IV. Results and Discussion

I started my project by learning basics of python (such as data types, sets, numpy array, array indexing). I got familiar with the Jupyter notebook environment. Then I started learning machine-learning concepts.

In the first part of project, the problem that confused me most about this project was how to use audio data in python code. I did not have background information about audio processing. After looking up some other projects, which used audio dataset I learned about Fourier transformation. I found librosa library to convert audio to raw data and get MFCC by taking Fourier transform.

After learning about how to feed my data in the system, I achieved to have simple implementation of CNN on my data. Success rate of network was only 33%.

In second part of project, I was focused on getting higher accuracy on test set. With LSTM, I split my dataset three times into test and training. It reached 26%, 45% and 60% accuracy as following.

In validation process, I notice when I pick songs for test dataset, I was getting better accuracy if I pick songs that have more variation in view count.

Now my project meets basic requirements although the success rate needs to increase. I need to train my network with more data as my future work. It will be generalizable when there is more variety in dataset.

V. Related Work

There are many other researches about this topic. Those papers are working with different datasets and approaches but most of them have high rate of success.

One of the tools used for this purpose is “Dance hit prediction”

Billboard 2015 Hot Dance/Electronic Songs

1. "Lean On" by Major Lazer & DJ Snake Featuring M0 — **82%**
2. "Where Are U Now" by Skrillex & Diplo With Justin Bieber — **72%**
3. "Hey Mama" by David Guetta Featuring Nicki Minaj, Bebe Rexha & Afrojack — **72%**
4. "You Know You Like It" by DJ Snake & AlunaGeorge — **63%**
5. "Waves" by Mr. Probz — **68%**
6. "Outside" by Calvin Harris Featuring Ellie Goulding — **82%**
7. "Prayer In C" by Lillywood & Robin Schulz — **65%**
8. "Blame" by Calvin Harris Featuring John Newman — **88%**
9. "How Deep Is Your Love" by Calvin Harris & Disciples — **62%**
10. "I Want You To Know" by Zedd Featuring Selena Gomez — **89%**

Figure 1. It is taken from https://motherboard.vice.com/en_us/article/bmvxvm/a-machine-successfully-predicted- the-hit-dance-songs-of-2015

As we can see their prediction achieves high rate of success. For their implementation they used SVM as model and Echo Net. Another paper about hit song prediction is “Hit Song Prediction For Pop Music By Siamese CNN with Ranking Loss” by Lang-Chi Yu, Yi-Hsuan Yang, Yun-Ning Hung and Yi-An Chen. In this paper they train CNN which treats it as ranking problem. They use commercial dataset with daily play-counts to train a multi-objective Siamese CNN model with Euclidean loss and pairwise ranking loss to learn from audio.

To sum up, there are number of relative research papers and tools focusing on this subject, which are implemented in variety of python tools and different machine learning methods.

VI. Conclusion and Future Work

- **ACHIVEMENTS**

I prepared small dataset for implemented prototype. 150 songs in total were stored in 3 categories and trained neural network for supervised image classification problem. With CNN, I have achieved 33% accuracy on test data. With LSTM, I split my dataset three times into test and training. It reached 26%, 45% and 60% accuracy as following.

- **ECONOMIC AND SOCIAL IMPACTS**

Project does not have social impacts yet as economic impact if tool will be developed to achieve higher percentage of prediction then music producers can purchase it as a service.

- **EFFECT**

  - **Economics**:

    If I can achieve high success rate with popularity prediction for songs, it would have impact on music industry.

  - **Impact on Environment**:

    It doesn't have impact on environment.

  - **Sustainability**

    Sustainability can be increased by machine learning, if we train the tool with bigger number of data and keep it up-to-date since music taste is changing by time then it is possible to get more sustainable solution for the problem.

- **Manufacturability**:

It can be developed to a web tool and can be purchased by music producers.

- **Ethics**:

Correctness of the tool is ethical problem of this project. If it doesn't work successfully, it can lead people to wrong answers. Another ethical problem is if this tool would be used for profiting then it may require permission from song owners.

- **Health**:

It has no affect on health.

- **Security**:

There is no security issue related to my project.

- **Social and political issues**:

It does not have any impact on social and political issues.

- **FUTURE WORK**

In future, my first aim is to increase the accuracy of this tool. For this issue, I will use more data; also other metrics can be added in dataset such as video of the music video, name of composer, lyrics, genre, released date.

Acknowledgements

I would like to thank Melih Kandemir for his guidance on my project.

References

[1] https://chatbotslife.com/how-neural-networks-work-ff4c7ad371f7

[2] http://myinspirationinformation.com/uncategorized/audio-signals-in-python/

[3] https://github.com/despoisj/DeepAudioClassification

[4] https://librosa.github.io/librosa/

[5] https://github.com/temerick/pytorch-lnl/blob/master/PyTorch lunch and learn.ipynb

[6] http://cs231n.github.io/python-numpy-tutorial/

[7] http://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[8] https://stackoverflow.com/questions/43441673/trying-to-load-a-custom-dataset-in-pytorch

[9] http://dorienherremans.com/dance/index.php

[10] https://stackoverflow.com/questions/12201577/how-can-i-convert-an-rgb-image-into-grayscale-in-python?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

Appendix

TRAINING AND TESTING – LSTM IMAGE CLASSIFICATION

In [1]:
```
import torch
import torchvision
import torchvision.datasets as dset
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np

from torch.autograd import Variable
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data as data
```

In [2]:
```
transform = transforms.Compose([
            transforms.Grayscale(num_output_channels=1),
            transforms.Resize((28, 28)),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
```

In [3]:
```
trainset = dset.ImageFolder(root="/Users/ilayda/Music/data/train_set/",transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=100,shuffle=True,  num_workers=1)

testset = dset.ImageFolder(root='/Users/ilayda/Music/data/test_set/',transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=100,shuffle=True,     num_workers=1)
```

In [4]:
```
classes=('low','medium','high')
```

In [5]:
```
# Hyper Parameters
sequence_length = 28
input_size = 28
hidden_size = 128
num_layers = 2
num_classes = 3
batch_size=100
num_epochs = 100
learning_rate = 0.01
```

In [6]:

```
# RNN Model (Many-to-One)
class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(RNN, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size,num_classes)

    def forward(self, x):
        # Set initial states
        h0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size))
        c0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size))

        # Forward propagate RNN
        out, _ = self.lstm(x, (h0, c0))

        # Decode hidden state of last time step
        out = self.fc(out[:, -1, :])
        return out

rnn = RNN(input_size, hidden_size, num_layers, num_classes)
```

In [7]:
```
# Loss and Optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
```

In [8]:
```
# Train the Model
for epoch in range(num_epochs):
    for i, (images, labels) in enumerate(trainloader):
        labels -= 1
        images = Variable(images.view(-1, sequence_length, input_size))
        labels = Variable(labels)

        # Forward + Backward + Optimize
        optimizer.zero_grad()
        outputs = rnn(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        if epoch % 1 == 0:
            print ('Epoch [%d/%d], Step [%d/%d], Loss: %.4f'
                   %(epoch+1, num_epochs, i+1, len(trainset)//batch_size, loss.data[0]))
```

```
Epoch [91/100], Step [2/1], Loss: 0.6202
Epoch [92/100], Step [1/1], Loss: 0.6098
Epoch [92/100], Step [2/1], Loss: 0.5696
Epoch [93/100], Step [1/1], Loss: 0.5288
Epoch [93/100], Step [2/1], Loss: 0.5454
Epoch [94/100], Step [1/1], Loss: 0.6151
Epoch [94/100], Step [2/1], Loss: 0.4363
Epoch [95/100], Step [1/1], Loss: 0.5466
Epoch [95/100], Step [2/1], Loss: 0.5155
Epoch [96/100], Step [1/1], Loss: 0.4991
Epoch [96/100], Step [2/1], Loss: 0.5057
Epoch [97/100], Step [1/1], Loss: 0.5107
Epoch [97/100], Step [2/1], Loss: 0.3135
Epoch [98/100], Step [1/1], Loss: 0.4552
Epoch [98/100], Step [2/1], Loss: 0.5510
Epoch [99/100], Step [1/1], Loss: 0.3860
Epoch [99/100], Step [2/1], Loss: 0.4004
Epoch [100/100], Step [1/1], Loss: 0.3586
Epoch [100/100], Step [2/1], Loss: 0.4281
```

In [9]:
```
# Test the Model
correct = 0
total = 0
for images, labels in testloader:
    labels -= 1
    images = Variable(images.view(-1, sequence_length, input_size))


    outputs = rnn(images)
    _, predicted = torch.max(outputs.data, 1)
    total += labels.size(0)
    correct += (predicted == labels).sum()

print('Test Accuracy of the model on the 15 test images: %d %%' % (100 * correct / total))

# Save the Model
torch.save(rnn.state_dict(), 'rnn.pkl')
```

```
Test Accuracy of the model on the 15 test images: 60 %
```

CREATING MY DATASET

1) Mp3 data to wav files

In [699]:
```
import torch
import torch.nn as nn
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.autograd import Variable
import librosa
import librosa.display
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pylab

from IPython.display import Audio, display
import matplotlib.pyplot as plt

import numpy as np
import pydub
```

In [1262]:
```
songname ="2.216.326.930-Major Lazer & DJ Snake - Lean On (feat. MÃ_) (Official Music Vid
```

In [1263]:
```
y, sr = librosa.load('/Users/ilayda/Music/Audios/HIGH/'+songname+'.mp3', mono=True, durat
```

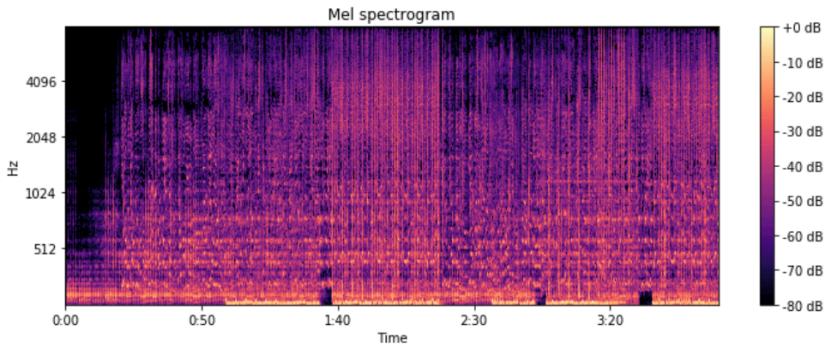
In [1264]:
```
y.shape, sr
```

Out[1264]:
```
((5292000,), 44100)
```

In [1265]:
```
librosa.core.get_duration(y, sr)
```

Out[1265]:
```
120.0
```

In [1266]:
```
S = librosa.feature.melspectrogram(y=y, n_mels=128*4, fmax=8000)
plt.figure(figsize=(10,4))
librosa.display.specshow(librosa.power_to_db(S, ref=np.max),
                         y_axis='mel', fmax=8000,
                         x_axis='time')
plt.colorbar(format='%+2.0f dB')
plt.title('Mel spectrogram')
plt.tight_layout()
plt.show()
S.shape
```



Out[1266]:
```
(512, 10336)
```

In [1267]:
```
df = pd.DataFrame(S)
df.describe()
```

2) Plotting Mel-Spectrogram from wav files and saving them as dataset (using librosa audio library)

In [237]:
```
import torch
import torch.nn as nn
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.autograd import Variable
import librosa
import librosa.display
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import random
```

In [419]:
```
%time
songname="2.216.326.930-Major Lazer & DJ Snake - Lean On (feat. MÃ_) (Official Music Video
base_path = '/Users/ilayda/Music/Audios/HIGH/wavfiles/'
filename = songname+'.wav'
y, sr = librosa.load(base_path + filename, mono=True,  sr=None)
```

```
CPU times: user 6 µs, sys: 13 µs, total: 19 µs
Wall time: 69.9 µs
```

In [420]:
```
%time
S = librosa.feature.melspectrogram(y=y, n_mels=128*4, fmax=8000)
plt.figure(figsize=(10,8))
librosa.display.specshow(librosa.power_to_db(S, ref=np.max),
                         y_axis='mel', fmax=8000,
                         x_axis='time')
plt.colorbar(format='%+2.0f dB')
plt.title('Mel spectrogram')
plt.tight_layout()
S.shape
plt.savefig("/Users/ilayda/Music/Audios/HIGH/spectrograms/" +  filename.strip('.wav') + '.
```

```
CPU times: user 8 µs, sys: 3 µs, total: 11 µs
Wall time: 18.8 µs
```