In [ ]:

```
import requests
import json
import time
import itertools
import wget
import os
import pickle
import numpy as np

import random
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
from sklearn.cluster.bicluster import SpectralCoclustering
from sklearn.metrics import precision_recall_curve
import scipy

sns.set_style('white')
import tensorflow as tf
import pandas as pd
import keras
from keras.optimizers import SGD, Adam
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
import keras.initializers as init
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as K
from keras.models import load_model
```

In [ ]:

```
x_train_dict = pickle.load(open('training_num.pik' , 'rb'))
```

In [ ]:

```
x_train_raw = x_train_dict['images']

x_train = np.array(x_train_raw)

print x_train.shape
```

In [ ]:

```
img_rows = x_train.shape[1]

img_cols = x_train.shape[2]

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 3, img_rows, img_cols)
    input_shape = (3, img_rows, img_cols)

else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 3)
    input_shape = (img_rows, img_cols, 3)
```

In [ ]:

```
x_train = x_train.astype('float32')

x_train /= 255

print 'x_train shape:', x_train.shape
print  x_train.shape[0], 'train samples'
```

In [ ]:

```
y_raw = pd.read_csv('Genres_labels_All_cleaned.csv')

y_train = y_raw.iloc[:, 1:-1].values

num_classes = y_train.shape[1]
```

In [ ]:

```
datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range = 0.5,
    fill_mode = 'wrap')

datagen.fit(x_train)
```

In [ ]:

```
# create an empty network model
model = Sequential()

model.add(Dense(64, activation='relu', input_shape=input_shape))
# this is our hidden layer

model.add(Dense(64, activation='relu'))
# and an output layer

model.add(Dense(8, activation='softmax'))

# prints out a summary of the model architecture
model.summary()
```

In [ ]:

```
ada = Adam(lr=0.01)
model.compile(loss='binary_crossentropy',
              optimizer=ada,
              metrics=['accuracy'])
```

In [ ]:

```
batch_size = 64
epochs = 20
```

In [ ]:

```
history = model.fit_generator(datagen.flow(x_train, y_train,
batch_size=batch_size),
                              steps_per_epoch=len(x_train) / batch_size,

                              epochs=epochs)
```

```
In [ ]: plt.plot(history.history['acc'])
        plt.xlabel("epoch")
        plt.ylabel("accuracy")
```

```
In [ ]: import h5py as h5py
```

```
In [ ]: model.save('mlp_var1.h5')
```

```
In [ ]: Acc_mlp_var1 = pd.DataFrame(history.history['acc'] , columns = ['Accuracy'])
```

```
In [ ]: Acc_mlp_var1.to_csv('mlp_v1.csv')
```