CNN_3-BNDOReg

April 26, 2017

1 CS109B Project Group 26 - Deep Learning

Main Specifications - Ver6
** Data preprocessing:**

- Channel rearrangement
- 14 labels multi label classification
- data augmentation by shift and zoom for 8000 training samples and 2000 test samples
- Centered features

** Main Architecture **

- Multi layer CNN with Batch Normalization Layers and Dropout and 0.1 L2 Kernel Regularization :

- Conv2D - Relu depth 32 and 7x7 kernel - He uniform initialization - 0.1 L2 Kernel Regularization - 30 % Dropout - Batch Norm - MaxPool 2x2 - Conv2D - Relu depth 32 and 5x5 kernel - He uniform initialization - 0.1 L2 Kernel Regularization - 30 % Dropout - Batch Norm - MaxPool 2x2 - Conv2D - Relu depth 64 and 3x3 kernel - He uniform initialization - 0.1 L2 Kernel Regularization - 30 % Dropout - Batch Norm - MaxPool 2x2 - Conv2D - Relu depth 64 and 3x3 kernel - He uniform initialization - 0.1 L2 Kernel Regularization - 30 % Dropout - Batch Norm - FC 128 , Relu , He uniform initialization - 0.1 L2 Kernel Regularization - 50 % Dropout - Batch Norm - FC 64 , Relu , He uniform initialization - 0.1 L2 Kernel Regularization - 50 % Dropout - Batch Norm
- SGD optimzer with 1e-4 learning rate and 0.99 momentum - Binary cross entropy loss function - Batch size 128 - Training convergence after 100 epochs

1.0.1 Import Modules

```
In [1]: 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 import regularizers as reg
        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
```

```
Using TensorFlow backend.
```

Open the Preprocessed Poster Data

```
In [2]: x_train_dict = pickle.load(open('training_num.pik' , 'rb'))
```

Specify the training/test split

```
In [3]: train_split = 8000
```

```
In [4]: x_train_raw = x_train_dict['images']

        x_train = np.array(x_train_raw)[:train_split,: ,: , :]
        x_test  = np.array(x_train_raw)[train_split:,: ,: , :]

        print x_train.shape
```

```
(8000, 128, 85, 3)
```

```
In [5]: 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 [6]: x_train = x_train.astype('float32')
        x_test  = x_test.astype('float32')

        x_train /= 255.0
        x_test  /= 255.0

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

        print 'x_test shape:', x_test.shape
        print  x_test.shape[0], 'test samples'
```

```
x_train shape: (8000, 128, 85, 3)
8000 train samples
x_test shape: (1988, 128, 85, 3)
1988 test samples
```

```
In [7]: y_raw = pd.read_csv('Genres_labels_All_cleaned.csv')

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

        num_classes = y_train.shape[1]

        print 'number of classes:  ' , num_classes
```

```
number of classes:   14
```

```
In [8]: 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)

        datagen.fit(x_test)
```

```
In [9]: # Specify regularization parameter
        reg_par = 0.1

In [10]: # create an empty network model
         modelc = Sequential()

         # --- input layer ---
         modelc.add(Conv2D(32, kernel_size=(7, 7), activation='relu', input_shape=input_shape
                          kernel_initializer = init.he_normal(109),
                          kernel_regularizer = reg.l2(reg_par)))

         # ------Dropout -------
         modelc.add(Dropout(0.3))

         # -----Batch Normalization -----
         modelc.add(BatchNormalization())

         # --- max pool ---
         modelc.add(MaxPooling2D(pool_size=(2, 2)))


         # ----------------------------------------------------------------------------------

         # ---- Conv Layer ---
         modelc.add(Conv2D(32, kernel_size=(5, 5), activation='relu',
                          kernel_initializer = init.he_normal(109),
                          kernel_regularizer = reg.l2(reg_par)))

         # ------Dropout -------
         modelc.add(Dropout(0.3))

         # -----Batch Normalization -----
         modelc.add(BatchNormalization())

         # --- max pool ---
         modelc.add(MaxPooling2D(pool_size=(2, 2)))


         # ----------------------------------------------------------------------------------

         # --- Conv layer ---
         modelc.add(Conv2D(64, kernel_size=(3, 3), activation='relu',
                          kernel_initializer = init.he_normal(109),
                          kernel_regularizer = reg.l2(reg_par)))

         # ------Dropout -------
         modelc.add(Dropout(0.3))
```

```
# -----Batch Normalization -----
modelc.add(BatchNormalization())

# --- max pool ---
modelc.add(MaxPooling2D(pool_size=(2, 2)))

# ----------------------------------------------------------------------------------

# --- Conv layer ---
modelc.add(Conv2D(64, kernel_size=(3, 3), activation='relu' ,
                 kernel_initializer = init.he_normal(109),
                 kernel_regularizer = reg.l2(reg_par)))

# ------Dropout -------
modelc.add(Dropout(0.3))

# -----Batch Normalization -----
modelc.add(BatchNormalization())

# --- max pool ---
modelc.add(MaxPooling2D(pool_size=(2, 2)))

#-----------------------------------------------------------------------------------

# flatten for fully connected classification layer
modelc.add(Flatten())

# --- fully connected layer ---
modelc.add(Dense(128, activation='relu',
                kernel_initializer = init.he_normal(109),
                kernel_regularizer = reg.l2(reg_par)))

# ------Dropout -------
modelc.add(Dropout(0.5))

# -----Batch Normalization -----
modelc.add(BatchNormalization())

# ----------------------------------------------------------------------------------

# --- fully connected layer ---
modelc.add(Dense(64, activation='relu' ,
                kernel_initializer = init.he_normal(109),
                kernel_regularizer = reg.l2(reg_par)))

# ------Dropout -------
modelc.add(Dropout(0.5))
```

```
# -----Batch Normalization -----
modelc.add(BatchNormalization())

# -----------------------------------------------------------------------------------

# --- classification ---
modelc.add(Dense(num_classes, activation='sigmoid'))

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

```
-----------------------------------------------------------------
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 122, 79, 32)       4736

-----------------------------------------------------------------
dropout_1 (Dropout)          (None, 122, 79, 32)       0

-----------------------------------------------------------------
batch_normalization_1 (Batch (None, 122, 79, 32)       128

-----------------------------------------------------------------
max_pooling2d_1 (MaxPooling2 (None, 61, 39, 32)        0

-----------------------------------------------------------------
conv2d_2 (Conv2D)            (None, 57, 35, 32)        25632

-----------------------------------------------------------------
dropout_2 (Dropout)          (None, 57, 35, 32)        0

-----------------------------------------------------------------
batch_normalization_2 (Batch (None, 57, 35, 32)        128

-----------------------------------------------------------------
max_pooling2d_2 (MaxPooling2 (None, 28, 17, 32)        0

-----------------------------------------------------------------
conv2d_3 (Conv2D)            (None, 26, 15, 64)        18496

-----------------------------------------------------------------
dropout_3 (Dropout)          (None, 26, 15, 64)        0

-----------------------------------------------------------------
batch_normalization_3 (Batch (None, 26, 15, 64)        256

-----------------------------------------------------------------
max_pooling2d_3 (MaxPooling2 (None, 13, 7, 64)         0

-----------------------------------------------------------------
conv2d_4 (Conv2D)            (None, 11, 5, 64)         36928

-----------------------------------------------------------------
dropout_4 (Dropout)          (None, 11, 5, 64)         0

-----------------------------------------------------------------
batch_normalization_4 (Batch (None, 11, 5, 64)         256

-----------------------------------------------------------------
max_pooling2d_4 (MaxPooling2 (None, 5, 2, 64)          0

-----------------------------------------------------------------
flatten_1 (Flatten)          (None, 640)               0

-----------------------------------------------------------------
```