#the below was copied from <https://www.analyticsvidhya.com/blog/2019/04/build-first-multi-label-image-classification-model-python/>

#citation: Wei-Ta Chu and Hung-Jui Guo, “Movie Genre Classification based on Poster Images with Deep Neural Networks,” Proceedings of International Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes, pp. 39-45, 2017. (in conjunction with ACM Multimedia 2017)

#imports

import keras

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D, MaxPooling2D

from keras.utils import to\_categorical

from keras.preprocessing import image

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from tqdm import tqdm

%matplotlib inline

#read meta data and class labels

train = pd.read\_csv('multi\_label\_train.csv') # reading the csv file

#read in images

train\_image = []

for i in tqdm(range(train.shape[0])):

img = image.load\_img('Multi\_Label\_dataset/Images/'+train['Id'][i]+'.jpg',target\_size=(400,400,3))

img = image.img\_to\_array(img)

img = img/255

train\_image.append(img)

X = np.array(train\_image)

#read labels for images

y = np.array(train.drop(['Id', 'Genre'],axis=1))

#split into train and test. Note that the samples should be split into training and testing by image, not by pixel. The testing should be on images whose pixels were not used for training. Make separate train, test, validate folders

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42, test\_size=0.1)

#Build model (total of 25 output labels possible)

model = Sequential()

model.add(Conv2D(filters=16, kernel\_size=(5, 5), activation="relu", input\_shape=(400,400,3)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(filters=32, kernel\_size=(5, 5), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(filters=64, kernel\_size=(5, 5), activation="relu"))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(filters=64, kernel\_size=(5, 5), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(25, activation='sigmoid'))

model.summary()

#”compile” the model

#Note the binary\_crossentropy as the loss function

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

#train the model

model.fit(X\_train, y\_train, epochs=10, validation\_data=(X\_test, y\_test), batch\_size=64)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#using the model to label unknown:

#Load image to classify:

img = image.load\_img('GOT.jpg',target\_size=(400,400,3))

img = image.img\_to\_array(img)

img = img/255

#Predict the labels:

classes = np.array(train.columns[2:])

proba = model.predict(img.reshape(1,400,400,3))

top\_3 = np.argsort(proba[0])[:-4:-1]

for i in range(3):

print("{}".format(classes[top\_3[i]])+" ({:.3})".format(proba[0][top\_3[i]]))

plt.imshow(img)