MINI PROJECT
(2020-21)

ON

Image Classification

MID-TERM REPORT

Institute of Engineering & Technology

Submitted by-
Saurabh Agarwal
(181500635)
Akash Sharma
(181500059)
Shivansh Agrawal
(181500681)
Raj Gupta
(181500542)

Supervised By: -
Mr. Vinay Agrawal
Assistant Professor
Department of Computer Engineering & Application

Contents

Abstract

In order to allow more flexible and general learning, it is an advantage for artificial systems to be able to discover re-usable features that capture structure in the environment, known as Deep Learning. Techniques have been shown based on convolutional neural networks and stacked Restricted Boltzmann Machines, which are related to some degree with neural processes. An alternative approach using abstract representations, the ARCS Learning Classifier System, has been shown to build feature hierarchies based on reinforcement, providing a different perspective, however with limited classification performance compared to Artificial Neural Network systems. An Abstract Deep Network is presented that is based on ARCS for building the feature network, and introduces gradient descent to allow improved results on an image classification task. A number of implementations are examined, comparing the use of back-propagation at various depths of the system. The ADN system is able to produce classification error of 1.18% on the MNIST dataset, comparable with the most established general learning systems on this task. The system shows strong reliability in constructing features, and the abstract representation provides a good platform for studying further effects such as top-down influences.

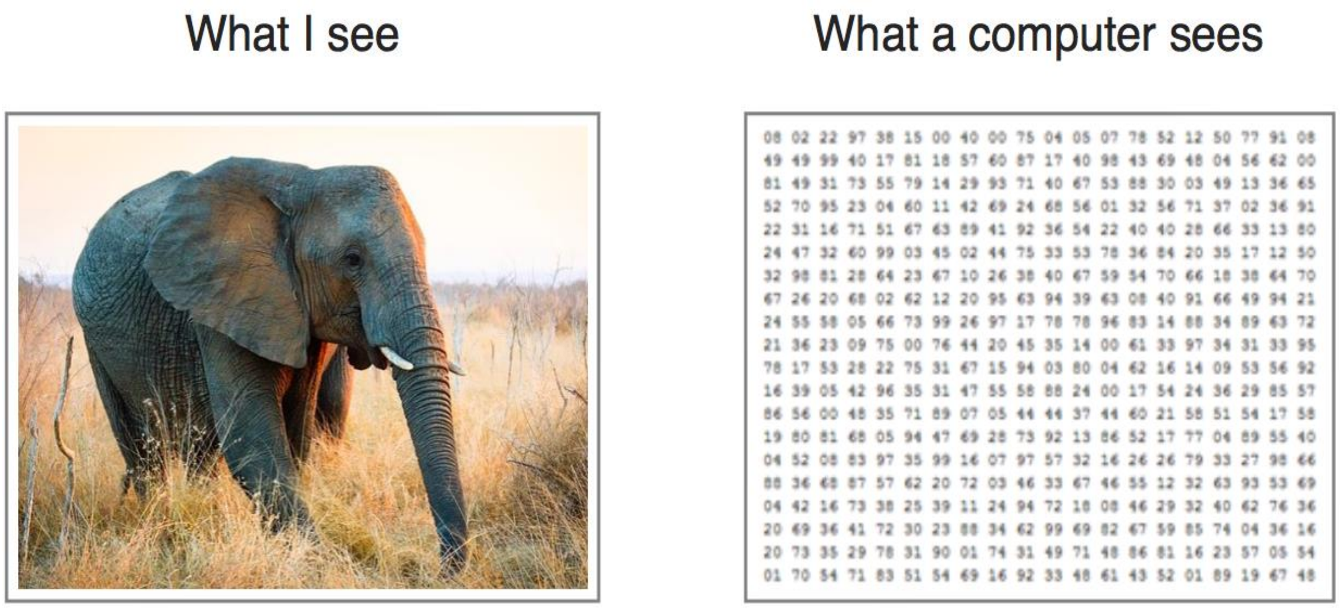
Introduction

Image processing involves some basic operations namely image restoration/rectification, image enhancement, image classification, images fusion etc. Image classification forms an important part of image processing. The objective of image classification is the automatic allocation of image to thematic classes. Two types of classification are supervised classification and unsupervised classification.

The process of image classification involves two steps, training of the system followed by testing. The training process means, to take the characteristic properties of the images (form a class) and form a unique description for a particular class. The process is done for all classes depending on the type of classification problem; binary classification or multi-class classification. The testing step means to categorize the test images under various classes for which system was trained. This assigning of class is done based on the partitioning between classes based on the training features.

Project Description

Convolutional neural networks (CNN) is a special architecture of artificial neural networks. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. The main task of image classification is acceptance of the input image and the following definition of its class. This is a skill that people learn from their birth and are able to easily determine that the image in the picture is an elephant. But the computer sees the pictures quite differently:



```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.

The image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output. The Convolution layer is always the first. The image (matrix with pixel values) is entered into it. Imagine that the reading of the input matrix begins at the top left of the image. Next the software selects a smaller matrix there, which is called a filter (or neuron, or core). Then the filter produces convolution, i.e. moves along the input image. The filter's task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by 1 unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller than an input matrix.

The network will consist of several convolutional networks mixed with nonlinear and pooling layers. When the image passes through one convolution layer, the output of the first layer becomes the input for the second layer. And this happens with every further convolutional layer.

The nonlinear layer is added after each convolution operation. It has an activation function, which brings nonlinear property. Without this property a network would not be sufficiently intense and will not be able to model the response variable (as a class label).

The pooling layer follows the nonlinear layer. It works with width and height of the image and performs a down sampling operation on them. As a result the image volume is reduced. This means that if some features (as for example boundaries) have already been identified in the previous convolution operation, then a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.

After completion of a series of convolutional, nonlinear and pooling layers, it is necessary to attach a fully connected layer. This layer takes the output information from convolutional networks. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the amount of classes from which the model selects the desired class. A fragment of the code of this model written in Python will be considered further in the practical part.

Hardware Requirements

- Processor: Intel dual core or above
- Processor Speed: 1.0GHZ or above
- RAM: 1 GB RAM or above
- Hard Disk: 20 GB hard disk or above

Software Requirements

- Language: Python
- Framework: Keras
- Libraries: Numpy, MatplotLib, Pandas
- Dataset: Celebrity images (41 categories)

Process

To create such model, it is necessary to go through the following phases:

- Model construction
- Model training
- Model testing
- Model evaluation

Model construction depends on machine learning algorithms. In this project case, it was neural networks. Such an algorithm looks like:

- begin with its object: model = Sequential()
- Then consist of layers with their types: model.add (*type_of_layer ()*)
- after adding a sufficient number of layers the model is compiled. At this moment Keras communicates with TensorFlow for construction of the model. During model compilation it is important to write a loss function and an optimizer algorithm.
- It looks like: model.compile (loss= 'name_of_loss_function', optimizer= 'name_of_opimazer_alg') the loss function shows the accuracy of each prediction made by the model.

Before model training it is important to scale data for their further use. After model construction it is time for model training. In this phase, the model is trained using training data and expected output for this data.

It looks this way: model.fit (training_data, expected_output).

Progress is visible on the console when the script runs. At the end it will report the final accuracy of the model. Once the model has been trained it is possible to carry out model testing. During this phase a second set of data is loaded. This data set has never been seen by the model and therefore its true accuracy will be verified. After the model training is complete, and it is understood that the model shows the right result, it can be saved by: model.save ("name_of_file.h5").

Finally, the saved model can be used in the real world. The name of this phase is model evaluation. This means that the model can be used to evaluate new data.

Coding

Importing the libraries:-

- import numpy as np
- import matplotlib.pyplot as plt
- import os
- import keras
- from keras.preprocessing.image import ImageDataGenerator
- from keras import Sequential, layers, models
- from keras import optimizers
- from keras.models import Model
- import itertools

Import the Dataset:-

- train_it = "C: /Users/DEMD/Desktop/data/datasets/train"
- test_it = "C: /Users/DEMD/Desktop/data/datasets/validation"

Rescaling the image through ImageDataGenerator:-

- Datagen = ImageDataGenerator(rescale = 1./255)
- train_it = datagen.flow_from_directory (train_it, target_size = (64, 64), batch_size = 32, class_mode = 'categorical')
- test_it = datagen.flow_from_directory (test_it, target_size = (64, 64), batch_size = 32, class_mode = 'categorical')

Define the Model layers:-

- model = models.Sequential ()
- model.add(layers.Conv2D(32,(3,3),strides = 1,padding = 'same',activation = 'relu',input_shape = (64,64,3)))
- model.add (layers.MaxPooling2D((2,2)))
- model.add (layers.Conv2D(64,(3,3),strides = 1,activation = 'relu'))
- model.add (layers.MaxPooling2D((2,2)))
- model.add(layers.Flatten())
- model.add(layers.Dense(64,activation = 'relu'))
- model.add(layers.Dense(41,activation = 'softmax'))

Compile the Model:-

- adam = keras.optimizers.Adam (learning_rate = 0.001)
- model.compile (optimizer = adam,loss = keras.losses.CategoricalCrossentropy (from_logits = True),metrics = ['accuracy'])

Fit to the Model:-

- history = model.fit_generator (train_it,steps_per_epoch = 60,epochs =80, validation_data = test_it,validation_steps = 25)

Evaluate the Test Set:-

- y_pred = model.evaluate(test_it)
- print ('Test loss:', y_pred[0])
- print ('Test accuracy:', y_pred[1])

Confusion matrix:-

- from sklearn.metrics import classification_report, confusion_matrix
- Y_pred = model.predict_generator (test_it)
- y_pred = np.argmax (Y_pred, axis = 1)
- print('Confusion Matrix')
- print(confusion_matrix(test_it.classes,y_pred))
- print('classification report')
- target_names=['Adrian Broody','Adriana Barazza','Ali Latter','Amena Khan','Andrew Lincolen', 'Angella Basset','Anna_Hathaway','Anthony Hopkins','arnold_schwarzenegger','Aron Judge','Aron Paul','Barabara Palvin','Barbra Streisand','Barry Pepper','Bella Hadid','bella thorne','ben_afflek','Beyonce Knowles','Buffon','De_Niro','Donnie Yen','dwayne_johnson','elton_john','Ennio Morricone','Fan Bingbing','Jackie chan','jerry_seinfeld','JetLee','kate_beckinsale','keanu_reeves','lauren_cohan','madonna','Maldini','mindy_kaling','Pirlo','Roberto_Benigni','simon_pegg','sofia_vergara','Tom_Hanks','Totti','will_smith']
- print(classification_report(test_it.classes,y_pred,target_names=target_names))

Plot the Confusion matrix:-

- cm = confusion_matrix(test_it.classes,y_pred)
- def plot_confusion_matrix(cm,classes,normalize = False,title = 'confusion matrix',cmap = plt.cm.Blues):
    - plt.imshow(cm,interpolation = 'nearest',cmap = cmap)
    - plt.title(title)
    - plt.colorbar()
    - tick_marks = np.arange(len(classes))

    - plt.xticks(tick_marks,classes,rotation = 45)
    - plt.yticks(tick_marks,classes)
    - if normalize:
        - cm = cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
        - print('normalized confusion matrix')
    - else:
        - print('without normalized confusion matrix')
    - print(cm)
    - thresh = cm.max()/2
    - for i,j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        - plt.text(j,i,cm[i,j],horizontalalignment ="center",color = "white" if cm[i,j] > thresh else "black")
    - plt.tight_layout()
    - plt.ylabel('true label')
    - plt.xlabel('predicted label')
    - cm_plot_labels = ['Adrian Broody','Adriana Barazza','Ali Latter','AmenaKhan','AndrewLincolen','Angella Basset','Anna_Hathaway','Anthony Hopkins','arnold_schwarzenegger','Aron Judge','Aron Paul','Barabara Palvin','Barbra Streisand','Barry Pepper','Bella Hadid','bella thorne','ben_afflek','Beyonce Knowles','Buffon','De_Niro','Donnie Yen','dwayne_johnson','elton_john','Ennio Morricone','Fan Bingbing','Jackie chan','jerry_seinfeld','Jet Lee','kate_beckinsale', 'keanu_reeves','lauren_cohan','madonna','Maldini','mindy_kaling','Pirlo','Roberto_Benigni','simon_pegg','sofia_vergara','Tom_Hanks','Totti','will_smith']
- plt.figure(figsize = (30,35))
- plot_confusion_matrix(cm,cm_plot_labels,title = 'confusion_matrix')

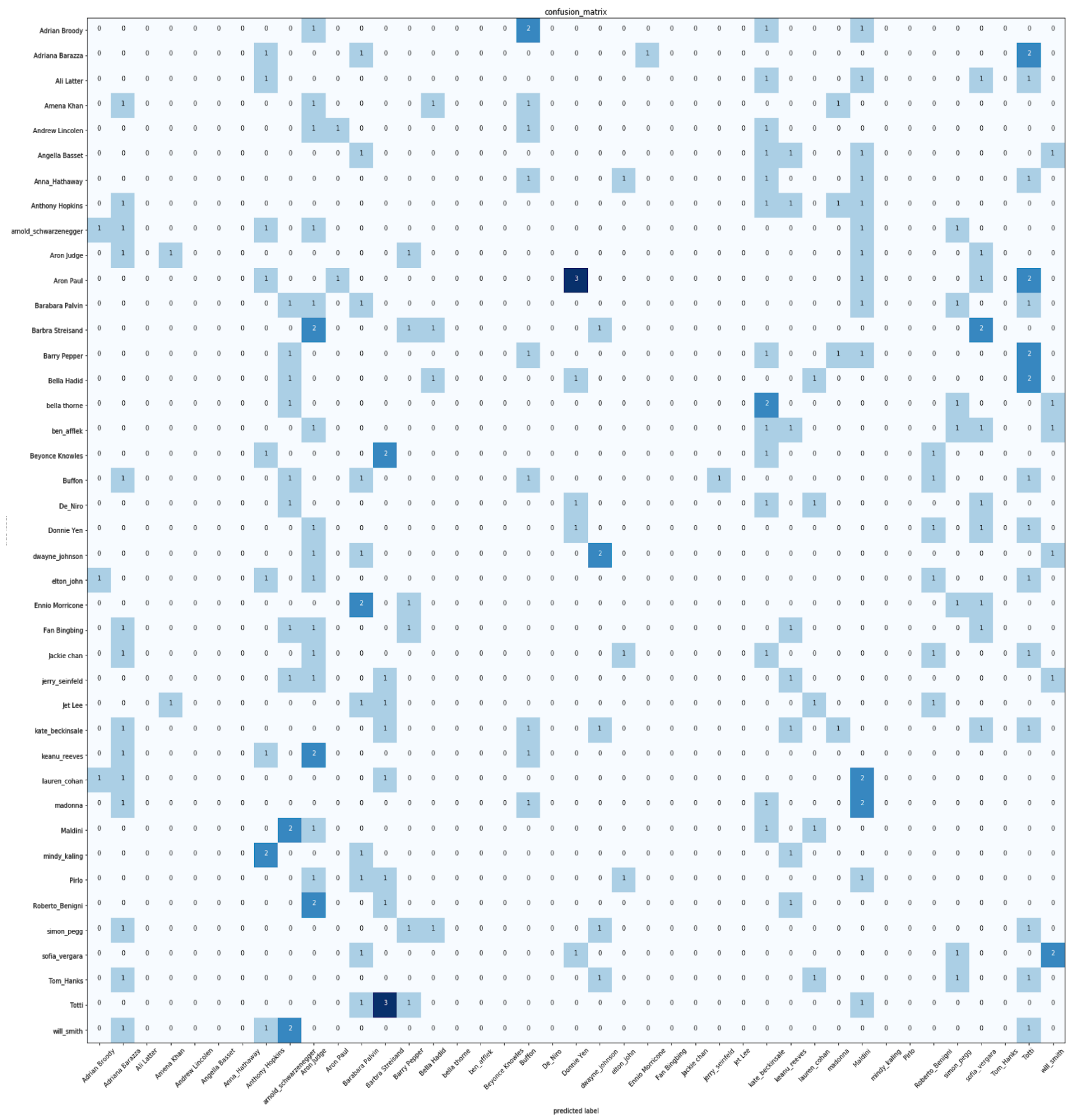
Plot the Model Accuracy:-

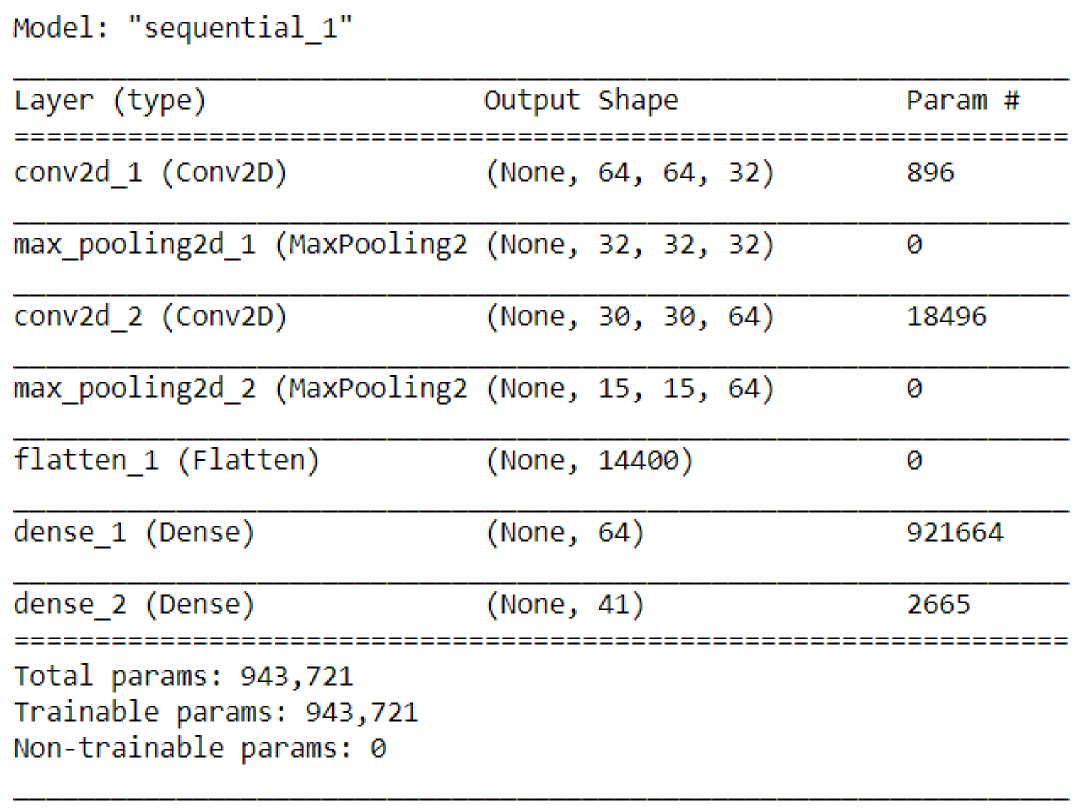
- plt.plot(history.history['accuracy'])
- plt.plot(history.history['val_accuracy'])
- plt.title('model accuracy')
- plt.ylabel('accuracy')
- plt.xlabel('epoch')
- plt.legend(['train', 'test'], loc='upper left')
- plt.show()

Plot the Model the Loss:-

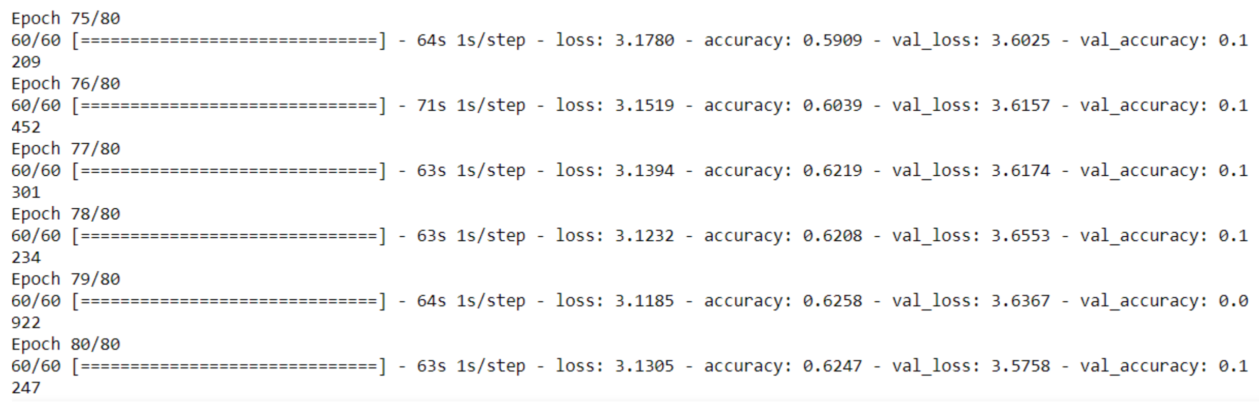
- plt.plot(history.history['loss'])
- plt.plot(history.history['val_loss'])
- plt.title('model loss')
- plt.ylabel('loss')
- plt.xlabel('epoch')
- plt.legend(['train', 'test'], loc='upper left')
- plt.show()

Screenshots

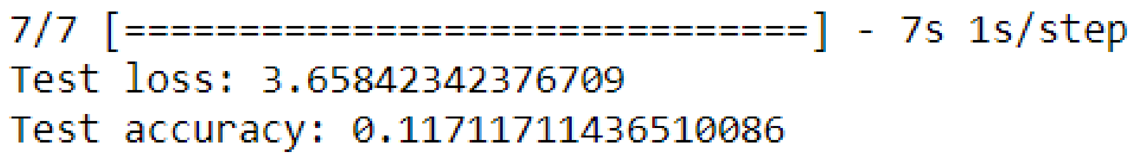




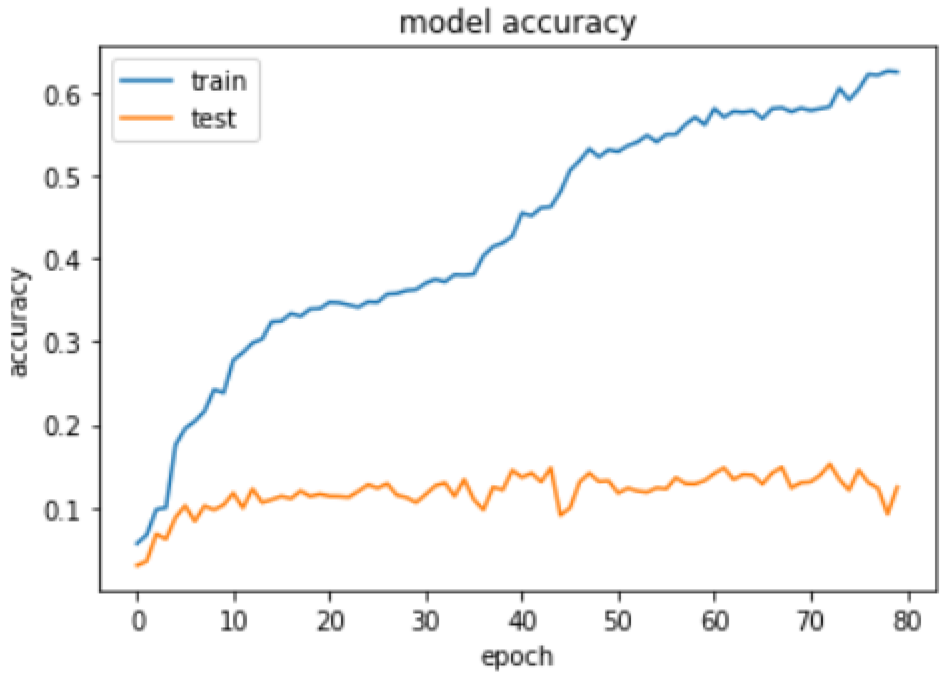
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 14400)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                921664
_________________________________________________________________
dense_2 (Dense)              (None, 41)                2665
=================================================================
Total params: 943,721
Trainable params: 943,721
Non-trainable params: 0
_________________________________________________________________
```

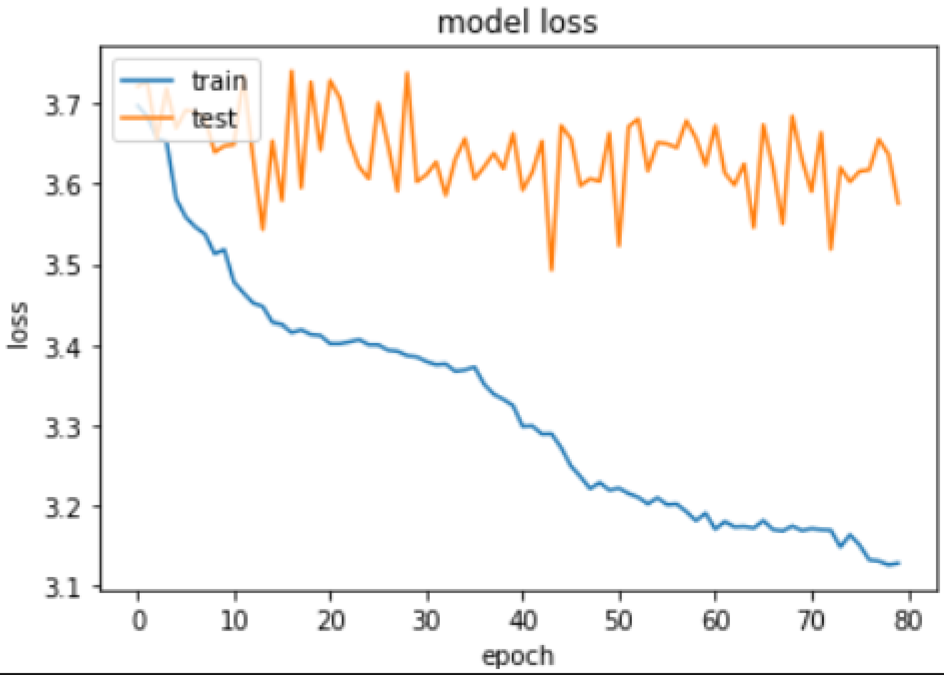


```
Epoch 75/80
60/60 [==============================] - 64s 1s/step - loss: 3.1780 - accuracy: 0.5909 - val_loss: 3.6025 - val_accuracy: 0.1
209
Epoch 76/80
60/60 [==============================] - 71s 1s/step - loss: 3.1519 - accuracy: 0.6039 - val_loss: 3.6157 - val_accuracy: 0.1
452
Epoch 77/80
60/60 [==============================] - 63s 1s/step - loss: 3.1394 - accuracy: 0.6219 - val_loss: 3.6174 - val_accuracy: 0.1
301
Epoch 78/80
60/60 [==============================] - 63s 1s/step - loss: 3.1232 - accuracy: 0.6208 - val_loss: 3.6553 - val_accuracy: 0.1
234
Epoch 79/80
60/60 [==============================] - 64s 1s/step - loss: 3.1185 - accuracy: 0.6258 - val_loss: 3.6367 - val_accuracy: 0.0
922
Epoch 80/80
60/60 [==============================] - 63s 1s/step - loss: 3.1305 - accuracy: 0.6247 - val_loss: 3.5758 - val_accuracy: 0.1
247
```



```
7/7 [==============================] - 7s 1s/step
Test loss: 3.65842342376709
Test accuracy: 0.11711711436510086
```


model accuracy
train
test
accuracy
epoch
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80


model loss
train
test
loss
epoch
3.7
3.6
3.5
3.4
3.3
3.2
3.1
0
10
20
30
40
50
60
70
80

<u>Conclusion</u>

“Image Classification” model developed to classify the celebrity image. I used Python Language with keras framework to implement this model. I used Convolutional Neural Networks (CNN) for image classification using images on Celebrity images data sets. This dataset used both training and testing purposes using CNN. It provides the accuracy rate of 65% on the training set and 14% on the test set. Images used in the training purpose are 64*64 and RGB color images. The future enhancement will focus on classifying the object detection in a video with better accuracy.

References

- https://keras.io/
- www.medium.com
- www.towardsdatascience.com
- https://www.youtube.com/