Using Keras for tasks in image recognition, face recognition, and embedding

Objective: To Program a network in Keras with a triplet loss function and recognise the given image or digit.

Approach:

- MNIST datasets is loaded and separated into train and test data using keras.
- The triplet loss is determined by adding square of anchor – positive and square of anchor-negative. (From given formula)
- Here , Anchor and positive are similar images and negative is a different image. They are parsed to embedding model.
- The embedding model is representing the image in a 128 dimensional embedding. An embedding is the collective name for mapping input features to vectors. It is a convolution neural network with different layers using Relu function.
- There are totally three layers used in the embedding function. The first two layers used rely function and the final layer used softmax function. Since the accuracy for linear function is very low compared to softmax function.
- The output of the embedding model is passed to complete_model function that identifies loss and compiles the model.
- Later the model is fit and the accuracy is determined.

Result:

MNIST DATASET:

- The model is predicted using X_train.
- The X_train and y_train passed to the final layer of embedding model. (Softmax function)
- The model is fit and the accuracy for the dataset is 85 percent.

EMNIST DATASET:

- The dataset is stored and they are reshaped accordingly. Once again the MNIST dataset are loaded and stored to concatenate with the letters. They are reshaped once again accordingly.
- The datasets are trained ,embedded and modelled.
- X_train is predicted and y_train passed to the final layer of embedding(softmax).
- The model is fit and the accuracy determined is 80 percent. The test image is loaded , predicted and plotted.

Challenges:

1. Concatenating the digit and letter was challenging due to different size of dataset(dataset.shape).

2. The letters had to be rotated and flipped to ensure that they are predicted correctly.

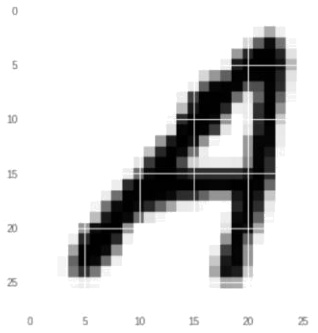

Fig: EMNIST-letter plot

Reference:

[1] https://hackernoon.com/building-a-facial-recognition-pipeline-with-deep-learning-in-tensorflow-66e7645015b8
[2] https://github.com/keras-team/keras/issues/10417
[3] https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526
[4] https://towardsdatascience.com/tutorial-alphabet-recognition-deeplearning-opencv-97e697b8fb86
[5] https://jakevdp.github.io/PythonDataScienceHandbook/03.06-concat-and-append.html
https://stackoverflow.com/questions/52449207/how-to-merge-different-dimensions-arrays-in-python
[6] https://stackoverflow.com/questions/51125969/loading-emnist-letters-dataset
[7] https://weehourstechnology.com/2018/08/26/the-emnist-dataset-handwriting-recognition-in-deep-machine-learning/

Pre-trained deep convolutional neural network face model:

Approach:

To learn pictures of someone, Deep learning tasks expect to be fed multiple instances of a custom class.

-Constructing VGG Face model:

- Architecture of this model is convolutional neural network.
- This model is a pertained implementation of **Mask R-CNN** technique on Python and Keras.
- Segmentation masks and generates bounding boxes are created for each instance of an image.

-VGG-Face has been designed through the following method:

- **vgg-face-keras-fc**- First convert the vgg-face Caffe model to a mxnet model, and then convert it to a keras model
- **vgg-face-keras**- Directly convert the vgg-face model to a keras model.

-VGG_face_weights which has pre-trained weights is been loaded.

-Loading test images as vectors, to decide if the loaded images are similar we have used **cosine similarity distance** to compute the distance between the vectors.

Results:

- Based on the above cosine similarity distance, the loaded test images are verified to be same. This concludes that the face reorganization using VGG Face model is obtained.
- Images of 3 different actors are downloaded and pre-processed (i.e. Hands and other extra features which will mislead are manually cropped.)-Total 15 Images are taken
- **'from sklearn.cluster import KMeans'** this function is used to cluster the similar images.
- **'from sklearn.decomposition import PCA' is mainly used to reduce** the dimensionality of a data set consisting of many variables correlated with each other and it also used in summarising and visualizing clusters.(i.e. scatter plot)
- **'from matplotlib.offsetbox import (OffsetImage, AnnotationBbox)'** this is used in visualizing image clusters in the scatter plot.

Challenges:

Plotting the images on the graph was little challenging because of the following:

- Scales of the graphs were different.
- Since the images were downloaded manually fitting them into the graphs were challenging.

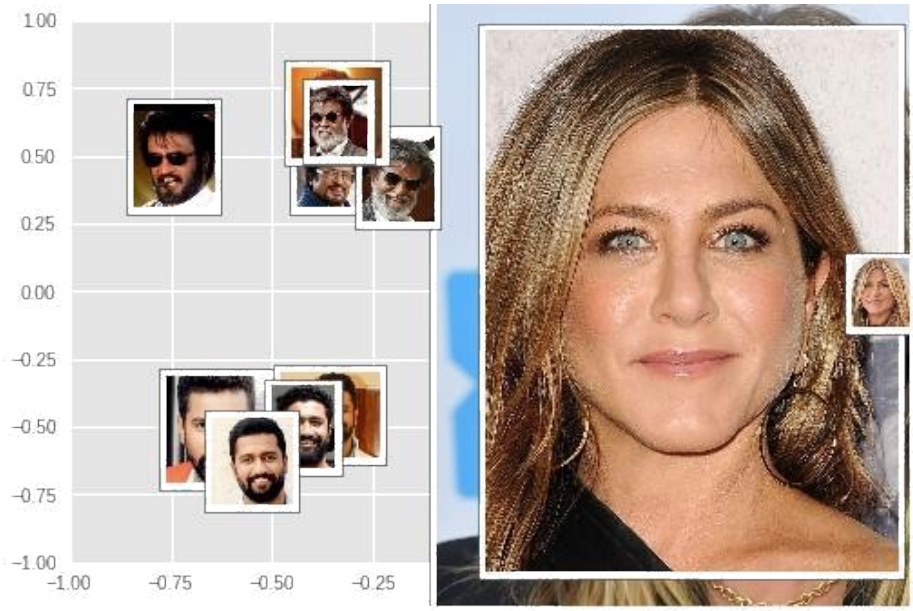

Fig: Final Image clustering plot

Reference:

[1] https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/
[2] https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial
[3] http://www.robots.ox.ac.uk/~vgg/data/vgg_face/