**Project 2 - Perform Facial Recognition with Deep Learning in Keras Using CNN**

**Problem Statement**:  
Facial recognition is a biometric alternative that measures unique characteristics of a human  
face. Applications available today include flight check in, tagging friends and family members in  
photos, and “tailored” advertising. You are a computer vision engineer who needs to develop a  
face recognition programme with deep convolutional neural networks.

**Objective**: Use a deep convolutional neural network to perform facial recognition using Keras.

**Dataset Details**:  
ORL face database composed of 400 images of size 112 x 92. There are 40 people, 10 images  
per person. The images were taken at different times, lighting and facial expressions. The faces  
are in an upright position in frontal view, with a slight left-right rotation.

**Link to the Dataset**: https://www.dropbox.com/s/i7uzp5yxk7wruva/ORL\_faces.npz?dl=0

**Prerequisites**:  
Keras  
Scikit Learn

**Steps to be followed**:  
**1. Input the required libraries**

Libraries such as numpy is used to load the dataset; sklearn for split training data and build confusion metrics; kereas is used for modelling.

**2. Load the dataset after loading the dataset, you have to normalize every image**

ORL\_faces.npz dataset is loaded to identified that there are 4 sets of data: testY – 160 of test response, testX – 160 of test image input; trainX – 240 of training image input; trainY – 240 of training response.

**3. Split the dataset**

The 240 of training dataset is split into 2 sets of data: one for training and the other for validation when building the model. The 160 of test data is used to understand how good the model is used for prediction/classification.

**4. Transform the images to equal sizes to feed in CNN**

As the image data came in as single dimension arrays of 10304 elements, they are transformed to be 112 x 92 images with single channel.

**5. Build a CNN model that has 3 main layers**

A CNN model is constructed using 1 convolution layer of 32 filters, followed by a pooling layer. Then it is flattened before adding 2 hidden FC layers and 1 output layer.

**6. Train the model**

The model is complied using Adam optimizer, categorical cross entropy and accuracy as the metrics. Bath size of 32 and 100 epochs are used to perform the training. The model accuracy when evaluated against the test dataset is around 93%.

**7. Plot the result**

A confusion metrics is constructed to understand the prediction of the model which illustrates that most of the prediction is accurate except for a few labelled as 4, 9, 14, 15, 18.

**8. Iterate the model until the accuracy is above 90%**

As the model already achieved more than 90% accuracy, the step may be viewed as how to stop training once the accuracy is above 90%. Plots of model accuracy and loss are performed on the history of the trained model to identify that around 30 epochs are sufficient to achieve 90%. This is further checked by running a new model with same set of layers and a callback to confirmed the understanding.