Rishabh Singh	LY- IS-3	2193192

Assignment - 04

Aim: Develop a face recognition system using CNN. Create a dataset of a minimum of 20 students from your class. Check and validate the accuracy of the model. Apply dimensionality reduction on the input image and plot the change in the accuracy of the system.

Objectives:

1. To learn data set creation.
2. To learn data normalization.

Theory:

1) *Dataset creation step:*
    1. Use google teachable machine to take fast snapshots of the student. Took 30 photos of each student exactly.
    2. Some with specs, some without specs, with different facial angles.
    3. Combined data from both batch-A and batch-B.
    4. Split the dataset between test and train 21 photos to train each user/student's picture and give 9 photos to test dataset.
    5. Create the named folder in both train and test.
    6. Data splitting is done manually.
    7. Now we have the final dataset ready.

2) *Image Augmentation:* An image classifier usually performs better when trained on significantly more images. A common problem in image classification models occurs when the model fails to correctly classify an image only because it was not trained on a different orientation of the same image. This can be overcome by feeding multiple possible image orientations and transformations to the model for training. However, in reality, gathering such diverse data might require more time, resources, and expertise and could be costly for a company. In such cases, image data augmentation is a popular choice for adding diversity to the existing dataset by using one or more augmentation techniques to generate various images for training. Although several Python libraries support multiple augmentation techniques, not all techniques are relevant and appropriate to train the model. A user needs to know which augmentations would help generate realistic additional data for training the model.

We can Augment the image data using various techniques, it includes:

- Augmenting image data using Geometric transformations such as flipping, cropping, rotating, zooming, etc.
- Augmenting image data by using Color transformations such as by adjusting brightness, darkness, sharpness, saturation, etc.
- Augmenting image data by random erasing, mixing images, etc.

3) *Libraries for image augmentation*

- **Scikit image:** scikit-image is an open-source library comprising a collection of easy-to-use algorithms for image processing. It is built on scipy.npimage, and it aims to be the reference library for scientific image analysis and image manipulation.
- **Augmentor:** Augmentor is an image augmentation library in Python for machine learning. It aims to be a standalone library that is platform and framework independent, allowing for convenient and fine-grained control over augmentation.
- **Torchvision:** The TorchVision package is part of the PyTorch project, a popular open-source Python machine learning framework. It consists of many sample datasets, model architectures, and typical image transformations for computer vision. Developed by the Facebook AI team, this package was intentionally kept separate from PyTorch to keep it lean and lightweight while still working seamlessly with PyTorch.
- **imgaug:** imgaug is a library for image augmentation in machine learning experiments. It supports various augmentation techniques, allowing users to easily combine and execute them in random order or on multiple CPU cores. It has a simple yet powerful stochastic interface that can augment images, landmarks, bounding boxes, heatmaps, and segmentation maps.
- **OpenCV:** Open Source Computer Vision) is perhaps the most famous open-source library for computer vision tasks. It focuses on image processing, video capture, and analysis (including features like face detection and object detection) and also helps to provide a common infrastructure for computer vision applications.

4) *Fit generator:*

When we call the .fit_generator() function it makes assumptions:

- Keras is first calling the generator function(data augmentation)
- Generator function(data augmentation) provides a batch_size of 32 to our .fit_generator() function.

- our **.fit_generator()** function first accepts a batch of the dataset, then performs backpropagation on it, and then updates the weights in our model.
- For the number of epochs specified(10 in our case) the process is repeated.