JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Face Recognition using Transfer learning on a pre-trained model(VGG16)

Indrani Bhunia

Student

Department of Computer Science And Engineering

Techno International New Town

(Formerly known as Techno India College Of Technology)

Abstract : Face recognition finds its major applications in health care and marketing. Among many biometrics like finger print, iris, voice, hand geometry, signature etc., face recognition has attained the most popular research orientation because of high recognition rate, uniqueness and large number of features. Some of the challenges in face recognition task include the differences in illumination, poses, expressions and background. Now-a-days deep learning methods are widely employed and they ensure promising results for image recognition and classification problems. Convolutional Neural Networks automatically learn features at multiple levels of abstraction through back propagation with convolution layers, pooling layers and fully connected layers. In this work, an automated face recognition method using Convolutional Neural Network (CNN) with transfer learning approach is proposed. The CNN with weights learned from pre-trained model VGG-16 on huge ImageNet database is used to train the images from the face database. The extracted features are fed as input to the Fully connected layer and softmax activation for classification. Two publicly available databases of face images - Yale and AT&T are used to test the performance of the proposed method. Experimental results verify that the method gives better recognition results compared to other methods.

1.INTRODUCTION

Facial Recognition can be defined as a bio-metric software application capable of uniquely identifying or verifying a person by comparing and analyzing patterns based on the person's facial contours. Everyone has a unique facial structure. This software is able to analyze your features, match it with information in a database and identify who you are. Facial recognition is a way of recognizing a human face through technology. A facial recognition system uses bio-metrics to map facial features from a photograph or video. It compares the information with a database of known faces to find a match.

Face recognition involves capturing face image from a video or from a surveillance camera. They are compared with the stored database. Face bio-metrics involves training known images, classify them with known classes and then they are stored in the database. When a test image is given to the system it is classified and compared with stored database. Face bio-metrics is a challenging field of research with various limitations imposed for a machine face recognition like variations in head pose, change in illumination, facial expression, aging, occlusion due to accessories etc.

The facial recognition market is expected to grow double. That's because facial recognition has all kinds of commercial applications. It can be used for everything from surveillance to marketing.

2.OBJECTIVES

Create a project using transfer learning solving various problems like Face Recognition, Image Classification, using existing Deep Learning models like VGG16, VGG19, ResNet, etc.

3.OVERVIEW

Human learners appear to have inherent ways to transfer knowledge between tasks. That is, we recognize and apply relevant knowledge from previous learning experience when we encounter new tasks. The more related a new task is to our previous experience, the more easily we can master it.

Transfer learning involves the approach in which knowledge learned in one or more source tasks is transferred and used to improve the learning of a related target task. While most machine learning algorithms are designed to address single tasks, the development of algorithms that facilitate transfer learning is a topic of ongoing interest in the machine-learning community.

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

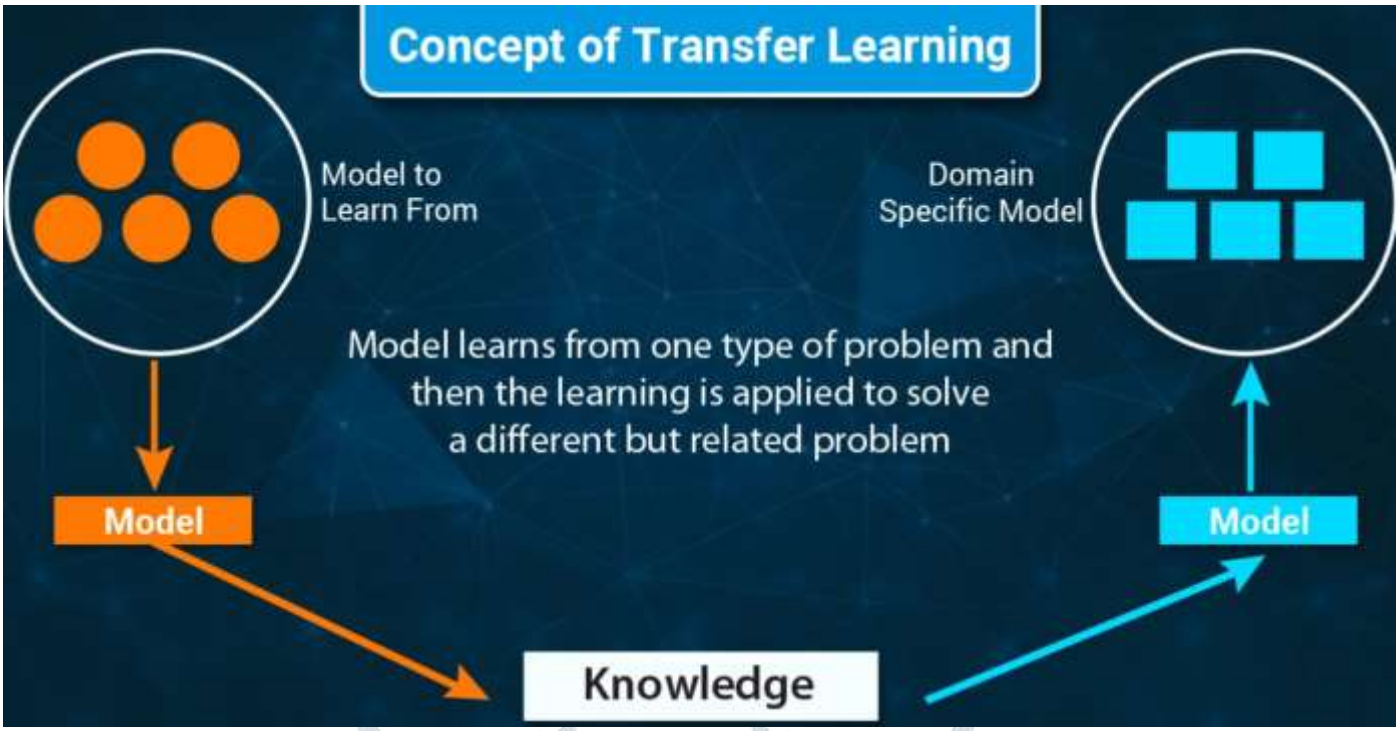
It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

This is very useful since most real-world problems typically do not have millions of labeled data points to train such complex models.

4.REASON FOR USING TRANSFER LEARNING

Many deep neural networks trained on natural images exhibit a curious phenomenon in common: on the first layer they learn features similar to Gabor filters and color blobs. Such first-layer features appear not to specific to a particular datasets or task but are general in that they are applicable to many datasets and tasks. As finding these standard features on the first layer seems to occur regardless of the exact cost function and natural image datasets, we call these first-layer features general. For example, in a network with an N-dimensional soft-max output layer that has been successfully trained towards a supervised classification objective, each output unit will be specific to a particular class. We thus call the last-layer features specific.

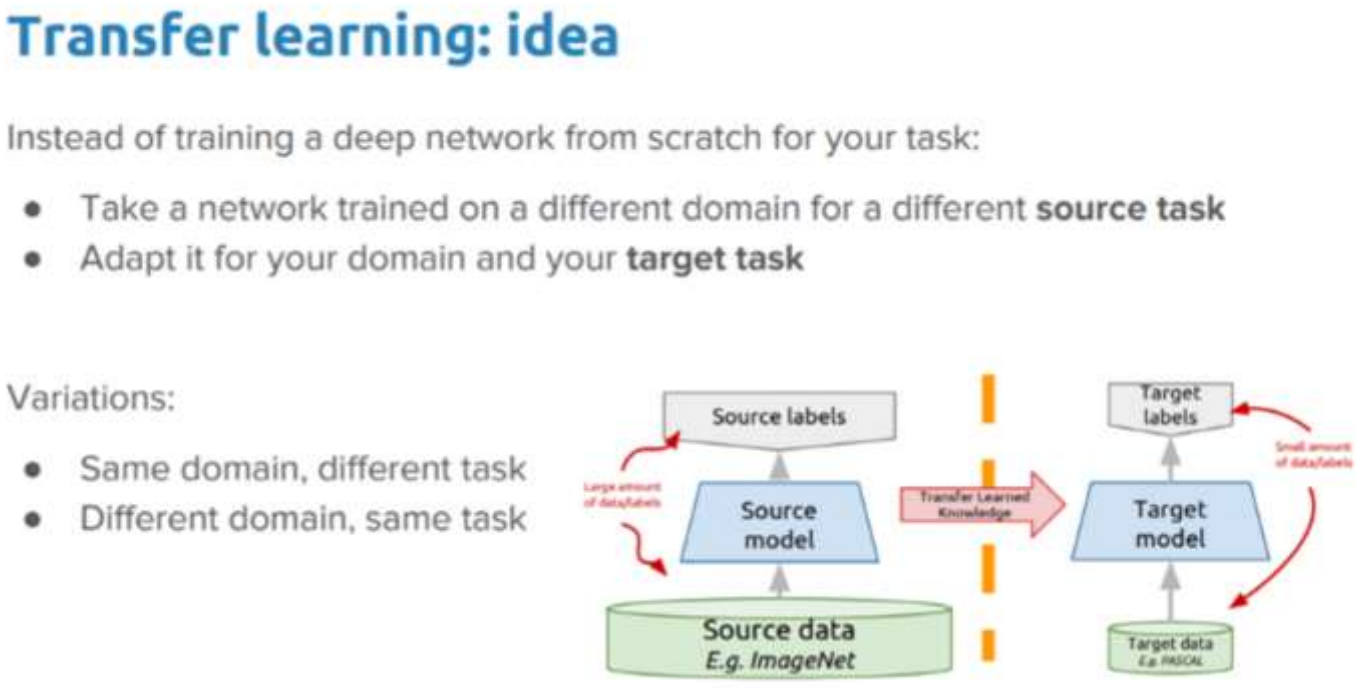
In transfer learning we first train a base network on a base datasets and task, and then we re-purpose the learned features, or transfer them, to a second target network to be trained on a target datasets and task. This process will tend to work if the features are general, that is, suitable to both base and target tasks, instead of being specific to the base task



5. Working Approach

Neural networks usually try to detect edges in the earlier layers, shapes in the middle layer and some task-specific features in the later layers. In transfer learning, the early and middle layers are used and we only retrain the latter layers. It helps leverage the labeled data of the task it was initially trained on.

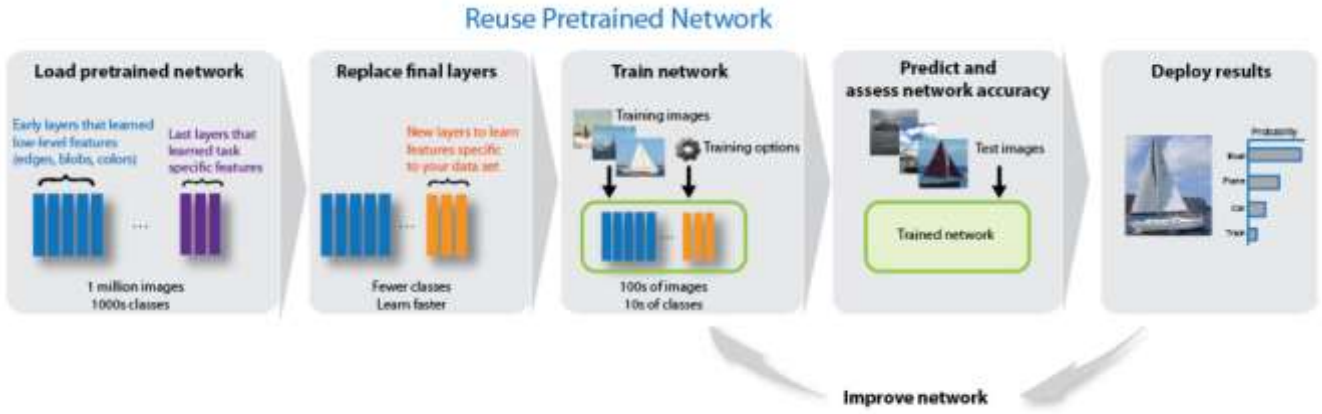
When dealing with transfer learning, we come across a phenomenon called freezing of layers. A layer, it can be a CNN layer, hidden layer, a block of layers, or any subset of a set of all layers, is said to be fixed when it is no longer available to train. Hence, the weights of freezed layers will not be updated during training. While layers that are not freezed follows regular training procedure.



6.Transferring Data Approach

When we use transfer learning in solving a problem, we select a pre-trained model as our base model. Now, there are two possible approaches to use knowledge from the pre-trained model. First way is to freeze a few layers of pre-trained model and train other layers on our new datasets for the new task. Second way is to make a new model, but also take out some features from the

layers in the pre-trained model and use them in a newly created model. In both cases, we take out some of the learned features and try to train the rest of the model. This makes sure that the only feature that may be same in both of the tasks is taken out from the pre-trained model, and the rest of the model is changed to fit new datasets by training.



7.Popular Models proposed

Perhaps three of the more popular models are as follows:

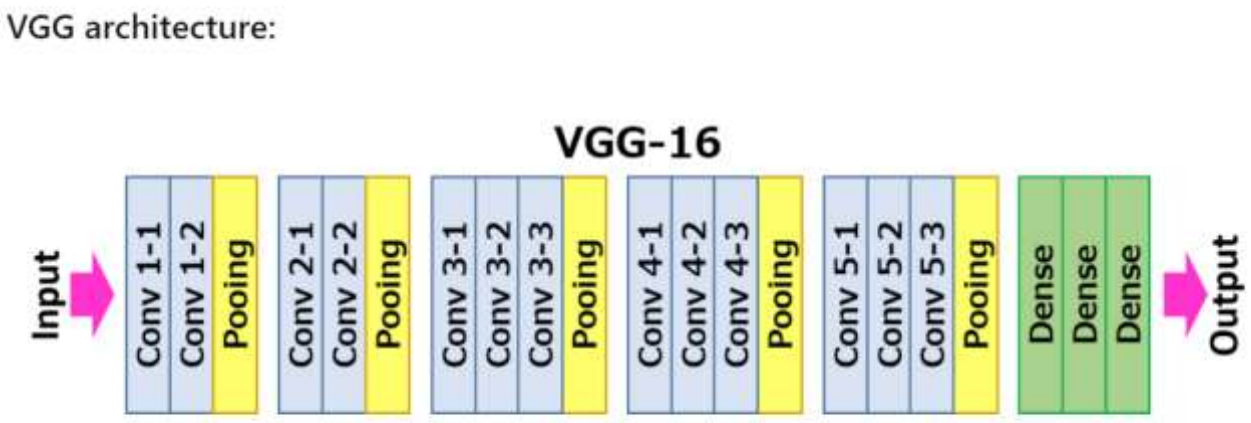
VGG (e.g. VGG16 or VGG19).
GoogLeNet (e.g. InceptionV3).
Residual Network (e.g. ResNet50).

8.**VGG16 Approach**

VGG16 is a convolutional neural network model .The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another.

In this model we have 16 layers in the model.

13 convolve layers
3 dense layers



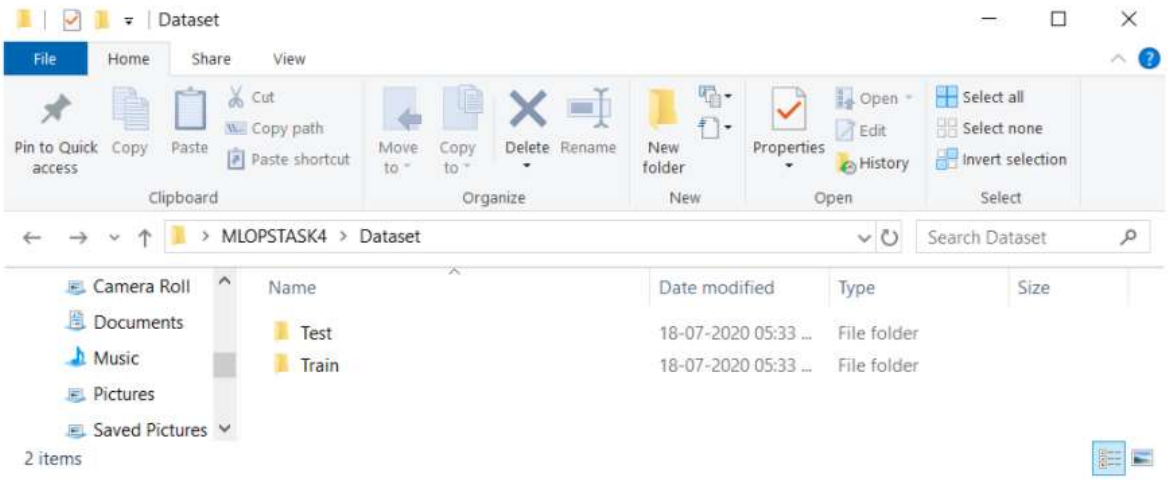
9.**Working Approach**

Pre-requisites :

Keras
Tensorflow
Pillow
numpy
pandas
Jupyter notebook and Python

Then we need to do the following Steps

Create a Workspace(in my case "MLOPSTASK4" for the project.
Create a Folder called “Dataset” inside which create 2sub-folders as “Train” and “Test”. Inside both of these Folders create a sub-folder with the name of the person whose face is to be recognized, this would store images of the particular person.



Step 1: Preparation of dataset.

```
In [4]: import cv2
        import numpy as np

        # Load HAAR face classifier
        face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

        # Load functions
        def face_extractor(img):
            # Function detects faces and returns the cropped face
            # If no face detected, it returns the input image

            gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray, 1.3, 5)

            if faces is ():
                return None

            # Crop all faces found
            for (x,y,w,h) in faces:
                cropped_face = img[y:y+h, x:x+w]

            return cropped_face

        # Initialize Webcam
```

```
# Initialize Webcam
cap = cv2.VideoCapture(0)
count = 0

# Collect 100 samples of your face from webcam input
while True:

    ret, frame = cap.read()
    if face_extractor(frame) is not None:
        count += 1
        face = cv2.resize(face_extractor(frame), (224, 224))
        #face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

        # Save file in specified directory with unique name
        file_name_path = 'C://Users//pauls//Desktop//MLOPSTASK4//Dataset//Train//Sayan//' + str(count) + '.jpg'
        cv2.imwrite(file_name_path, face)

        # Put count on images and display live count
        cv2.putText(face, str(count), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
        cv2.imshow('Face Cropper', face)

    else:
        print("Face not found")
        pass
```

```
    if face_extractor(frame) is not None:
        count += 1
        face = cv2.resize(face_extractor(frame), (224, 224))
        #face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

        # Save file in specified directory with unique name
        file_name_path = 'C://Users//pauls//Desktop//MLOPSTASK4//Dataset//Train//Sayan//' + str(count) + '.jpg'
        cv2.imwrite(file_name_path, face)

        # Put count on images and display live count
        cv2.putText(face, str(count), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
        cv2.imshow('Face Cropper', face)

    else:
        print("Face not found")
        pass

    if cv2.waitKey(1) == 13 or count == 100: #13 is the Enter Key
        break

cap.release()
cv2.destroyAllWindows()
print("Collecting Samples Complete")
```

```
Collecting Samples Complete
```

First we take 100 images from the required person and split it to 2 sets Train and Test. Train set consists of 80 images while Test set contains 20 images(We need to segregate manually)

Step 2: Model Creation.

```
from keras.applications import VGG16

rows = 224
cols = 224

model = VGG16(weights = 'imagenet', include_top = False, input_shape = (rows, cols, 3))
for (i,layer) in enumerate(model.layers):
    print(str(i) + " " + layer.__class__.__name__, layer.trainable)

from keras.applications import VGG16

rows = 224
cols = 224

model = VGG16(weights = 'imagenet', include_top = False, input_shape = (rows, cols, 3))

for layer in model.layers:
    layer.trainable = False

for (i,layer) in enumerate(model.layers):
    print(str(i) + " " + layer.__class__.__name__, layer.trainable)

def addlayer(bottom_model, num_classes):
    """creates the head of the model that will bw placed on top of the bottom layers"""
```

```
def addlayer(bottom_model, num_classes):
    """creates the head of the model that will bw placed on top of the bottom layers"""
    top_model = bottom_model.output
    top_model = GlobalAveragePooling2D()(top_model)
    top_model = Dense(1024,activation='relu')(top_model)
    top_model = Dense(1024,activation='relu')(top_model)
    top_model = Dense(512,activation='relu')(top_model)
    top_model = Dense(num_classes,activation='softmax')(top_model)
    return top_model

from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, GlobalAveragePooling2D
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.layers.normalization import BatchNormalization
from keras.models import Model

num_classes = 2

FC_Head = addlayer(model, num_classes)
modelnew = Model(inputs=model.input, outputs=FC_Head)
print(modelnew.summary())

from keras.preprocessing.image import ImageDataGenerator

train_data_dir = 'C://Users//pauls//Desktop//MLOPSTASK4//Dataset//Train//'
```

```
from keras.preprocessing.image import ImageDataGenerator

train_data_dir = 'C://Users//pauls//Desktop//MLOPSTASK4//Dataset//Train//'
validation_data_dir = 'C://Users//pauls//Desktop//MLOPSTASK4//Dataset//Test//'

train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=20,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
validation_datagen = ImageDataGenerator(rescale=1./255)

train_batchsize = 16
val_batchsize = 10

train_generator = train_datagen.flow_from_directory(train_data_dir,
                                                    target_size=(rows, cols),
                                                    batch_size=train_batchsize,
                                                    class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(validation_data_dir,
                                                    target_size=(rows, cols),
                                                    batch_size=val_batchsize,
```

```
validation_generator = validation_datagen.flow_from_directory(validation_data_dir,
                                                    target_size=(rows, cols),
                                                    batch_size=val_batchsize,
                                                    class_mode='categorical',
                                                    shuffle=False)

from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint = ModelCheckpoint("face_recog.h5", monitor="val_loss", mode="min", save_best_only = True, verbose=1)
earlystop = EarlyStopping(monitor= 'val_loss', min_delta = 0, patience = 3, verbose = 1, restore_best_weights = True)
callbacks = [earlystop, checkpoint]

modelnew.compile(loss='categorical_crossentropy', optimizer=RMSprop(lr=0.001),metrics=['accuracy'])

nb_train_samples=1190
nb_validation_samples=170
epochs=3
batch_size=16

history = modelnew.fit_generator(train_generator,
                                 steps_per_epoch=nb_train_samples // batch_size,
                                 epochs=epochs,
                                 callbacks=callbacks,
                                 validation_data=validation_generator,
```

```
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint = ModelCheckpoint("face_recog.h5", monitor="val_loss", mode="min", save_best_only = True, verbose=1)
earlystop = EarlyStopping(monitor= 'val_loss', min_delta = 0, patience = 3, verbose = 1, restore_best_weights = True)
callbacks = [earlystop, checkpoint]

modelnew.compile(loss='categorical_crossentropy', optimizer=RMSprop(lr=0.001),metrics=['accuracy'])

nb_train_samples=1190
nb_validation_samples=170
epochs=3
batch_size=16

history = modelnew.fit_generator(train_generator,
                                 steps_per_epoch=nb_train_samples // batch_size,
                                 epochs=epochs,
                                 callbacks=callbacks,
                                 validation_data=validation_generator,
                                 validation_steps=nb_validation_samples // batch_size)
modelnew.save("face_recog.h5")
```

```
                                callbacks=callbacks,
                                validation_data=validation_generator,
                                validation_steps=nb_validation_samples // batch_size)
modelnew.save("face_recog.h5")
```

```
0 InputLayer False
1 Conv2D True
2 Conv2D True
3 MaxPooling2D True
4 Conv2D True
5 Conv2D True
6 MaxPooling2D True
7 Conv2D True
8 Conv2D True
9 Conv2D True
10 MaxPooling2D True
11 Conv2D True
12 Conv2D True
13 Conv2D True
14 MaxPooling2D True
15 Conv2D True
16 Conv2D True
17 Conv2D True
18 MaxPooling2D True
0 InputLayer False
```

```
17 Conv2D True
18 MaxPooling2D True
0 InputLayer False
1 Conv2D False
2 Conv2D False
3 MaxPooling2D False
4 Conv2D False
5 Conv2D False
6 MaxPooling2D False
7 Conv2D False
8 Conv2D False
9 Conv2D False
10 MaxPooling2D False
11 Conv2D False
12 Conv2D False
13 Conv2D False
14 MaxPooling2D False
15 Conv2D False
16 Conv2D False
17 Conv2D False
```

```
17 Conv2D False
18 MaxPooling2D False
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
```

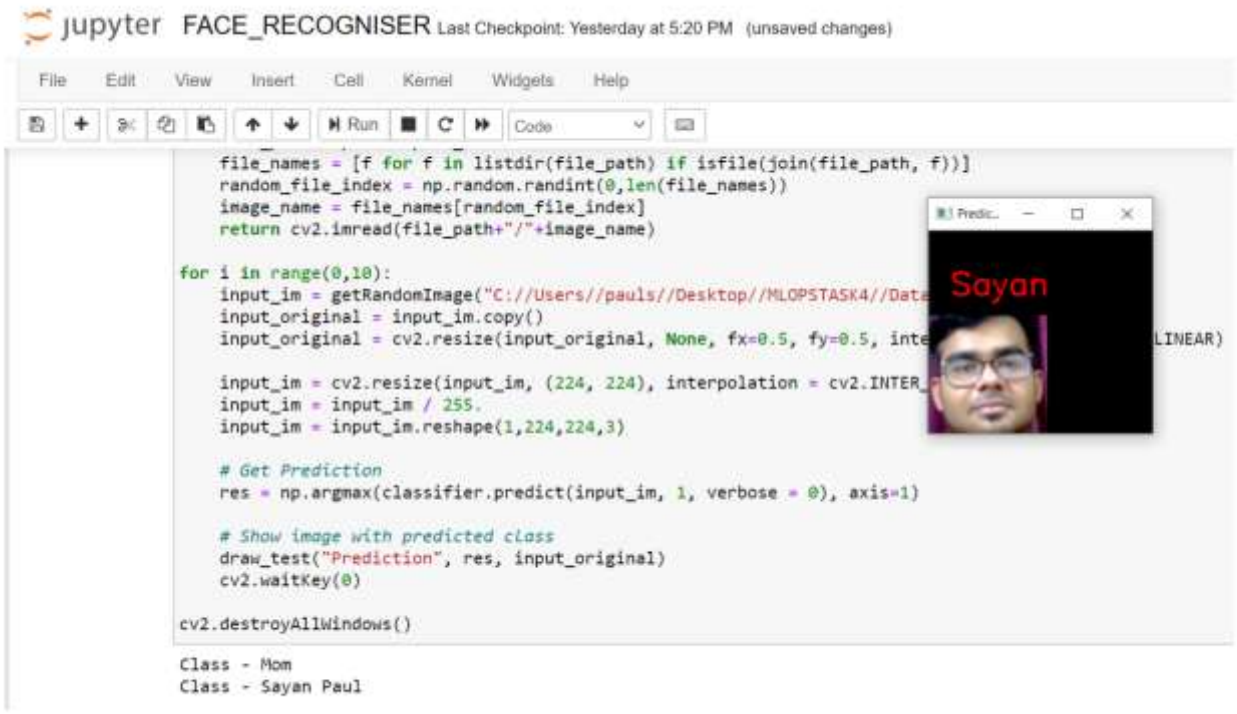
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0

```
Non-trainable params: 14,714,688
_________________________________________________________________
None
Found 160 images belonging to 2 classes.
Found 40 images belonging to 2 classes.
Epoch 1/3
74/74 [==============================] - 309s 4s/step - loss: 0.2813 - accuracy: 0.9062 - val_loss: 2.8372e-06 - val_accuracy:
1.0000

Epoch 00001: val_loss improved from inf to 0.00000, saving model to face_recog.h5
Epoch 2/3
74/74 [==============================] - 316s 4s/step - loss: 0.1208 - accuracy: 0.9949 - val_loss: 1.6928e-06 - val_accuracy:
1.0000

Epoch 00002: val_loss improved from 0.00000 to 0.00000, saving model to face_recog.h5
Epoch 3/3
74/74 [==============================] - 319s 4s/step - loss: 2.5478e-06 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accur
acy: 1.0000

Epoch 00003: val_loss improved from 0.00000 to 0.00000, saving model to face_recog.h5
```

Here we are doing only two epochs as the data set is very small and the model is being trained with only 2 separate faces. Model after training is giving the accuracy of 100% it is only because all the training images are taken at the same time and it is trained with 2 different faces(Small dataset).

Step 3: Testing Model

```
from keras.models import load_model

classifier = load_model('face_recog.h5')

import os
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join

person_dict = {"[0]": "Mom", "[1]": "Sayan"}
person_dict_n = {"Mom": "Mom", "Sayan": "Sayan Paul"}

def draw_test(name, pred, im):
    person = person_dict[str(pred)]
    BLACK = [0,0,0]
    expanded_image = cv2.copyMakeBorder(im, 80, 0, 0, 100 ,cv2.BORDER_CONSTANT,value=BLACK)
    cv2.putText(expanded_image, person, (20, 60) , cv2.FONT_HERSHEY_SIMPLEX,1, (0,0,255), 2)
    cv2.imshow(name, expanded_image)

def getRandomImage(path):
    """function loads a random images from a random folder in our test path """
    folders = list(filter(lambda x: os.path.isdir(os.path.join(path, x)), os.listdir(path)))
    random_directory = np.random.randint(0,len(folders))
    path_class = folders[random_directory]
    print("Class - " + person_dict_n[str(path_class)])
    file_path = path + path_class
    file_names = [f for f in listdir(file_path) if isfile(join(file_path, f))]
    random_file_index = np.random.randint(0,len(file_names))
    image_name = file_names[random_file_index]
    return cv2.imread(file_path+"/"+image_name)

for i in range(0,10):
    input_im = getRandomImage("C://Users//pauls//Desktop//MLOPSTASK4//Dataset//Test//")
    input_original = input_im.copy()
    input_original = cv2.resize(input_original, None, fx=0.5, fy=0.5, interpolation = cv2.INTER_LINEAR)

    input_im = cv2.resize(input_im, (224, 224), interpolation = cv2.INTER_LINEAR)
    input_im = input_im / 255.
    input_im = input_im.reshape(1,224,224,3)

    # Get Prediction
    res = np.argmax(classifier.predict(input_im, 1, verbose = 0), axis=1)

    # Show image with predicted class
    draw_test("Prediction", res, input_original)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```



(Name, Image, photo can be variable and subjected to change, taken value as random)

10.Conclusion:

By the above experiments, we are providing methods to apply deep learning techniques for smaller databases using transfer learning and triple siamese network. We have achieved state of the art recognition rate for some of the databases. Till now no one had Incorporated transfer learning for face recognition tasks

We have also discussed that how pre-trained network can be harnessed as feature extractor for rare class classification. Now we have used it for face recognition task but this can find application in tasks including race cancer classification, rare species recognition, one shot learning to name a few. Even when we don't have pre-trained network from the same domain, still these methods can be applied to transfer weights from network originally trained for different task too. We also demonstrated two different modalities for feature visualization. Before applying any machine learning techniques it is imperative to know that our extracted features are actually classifiable or not. However you may not be able to visualize every time but our visualization using reshaped features in form of images and t-SNE plots are strong and effective way to test whether your feature extractor will work or not. We are providing very efficient algorithms for enrolling new subjects. In any deep learning methods you always need to train your network on new data as well. But our algorithm allows generalization to the extent that you can directly generate the embedding and recognize new subjects as well. Also for impostor recognition we are providing a novel approach to set the clear threshold. We can also think this method as kernel learning algorithm. Because the similarity matrix obtained is positive semi-definite and can be used as custom kernel for SVM based classification. N-fold recognition accuracy makes our results more robust and has included all types of cases. Although we are providing

top-5 best accuracy also because when we do recognition for only one image then task becomes more difficult as we have very less flexibility to variation. Top-5 best accuracy is justified as whenever such system will be employed anywhere we can make sure that the training happens on really best images. In quoting overall average accuracy, we have some lower recognition rates pulling the results bit lower, showing that training images were not so clear in that particular set.

11.References:

[1] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. "learning and transferring mid-level image representations using convolutional neural networks.". Computer Vision and Pattern Recognition, 2014. [2] Faizan Ahmad, Aaima Najam, and Zeeshan Ahmed. image-based face detection and recognition: State of the art. IJCSI International Journal of Computer Science Issues,, pages Vol. 9, Issue. 6, No. 1., 2013. [3] J. Philbin F. Schroff, D. Kalenichenko. facenet: A unified embedding for face recognition and clustering. Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pages 815–823, 2015. [4] Bromley, J.and Bentz, J. W. Bottou, L.and Guyon, I.and LeCun, Y.and Moore, C., E. Sckinger, Shah, and R. "signature verification using a "siamese" time delay neural network". IJPRAI, 7(4):669–688, 1993. [5] Vo, J.and N. N., and Hays. "localizing and orienting street views using overhead imagery". In European Conference on Computer Vision, pages 494–509, 2016. [6] Ahmed, E., Jones, M., Marks, and T.K. "an improved deep learning architecture for person re-identification". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3908– 3916, 2015. [7] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. "siamese neural networks for one-shot image recognition ". In ICML Deep Learning workshop, 2015.