

Assignment No. 4

Aim: Develop face recognition system using CNN. Create a dataset of minimum 20 students from your class. Check and validate the accuracy of the model.

☑ Apply dimensionality reduction on input image and plot the change in accuracy of system.

Objectives:

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

Theory:

Dataset creation steps

1. Articulate the problem early.
2. Establish data collection mechanisms.
3. Format data to make it consistent.
4. Reduce data.
5. Complete data cleaning.
6. Decompose data.
7. Rescale data.
8. Discretize data.

Image Augmentation

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Image data augmentation is used to expand the training dataset in order to improve the performance and ability of the model to generalize.

The intent is to expand the training dataset with new, plausible examples. This means, variations of the training set image that are likely to be seen by the model. For example, a horizontal flip of a picture of a cat may make sense, because the photo could have been taken from the left or right. A vertical flip of the photo of a cat does not make sense and would probably not be appropriate given that the model is very unlikely to see a photo of an upside-down cat.

Libraries for image augmentation

There are a lot of image augmentations packages

- skimage
- opencv
- imgaug
- Albumentations
- Augmentor

- Keras(ImageDataGenerator class)

Fit_generator, validate_generator, predict_generator

fit_generator

```
fit_generator(  
    generator, steps_per_epoch=None, epochs=1, verbose=1, callbacks=None,  
    validation_data=None, validation_steps=None, validation_freq=1,  
    class_weight=None, max_queue_size=10, workers=1, use_multiprocessing=False,  
    shuffle=True, initial_epoch=0  
)
```

Fits the model on data yielded batch-by-batch by a Python generator.

predict_generator

```
predict_generator(  
    generator, steps=None, callbacks=None, max_queue_size=10, workers=1,  
    use_multiprocessing=False, verbose=0  
)
```

Generates predictions for the input samples from a data generator.

evaluate_generator

```
evaluate_generator(  
    generator, steps=None, callbacks=None, max_queue_size=10, workers=1,  
    use_multiprocessing=False, verbose=0  
)
```

Evaluates the model on a data generator.

Code:

MTCNN

```
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from matplotlib.patches import Circle
from mtcnn.mtcnn import MTCNN
from PIL import Image
from numpy import asarray
def draw_faces(filename, result_list):
    data = plt.imread(filename)
    for i in range(len(result_list)):
        # get coordinates
        x1, y1, width, height = result_list[i]['box']
        x2, y2 = x1 + width, y1 + height
        if x1 < 0:
            x1 = 0
        if x2 < 0:
            x2 = 0
        if y1 < 0:
            y1 = 0
        if y2 < 0:
            y2 = 0
        plt.subplot(1, len(result_list), i+1)
        plt.axis('off')
        plt.imshow(data[y1:y2, x1:x2])
        cv2.imwrite(filename, data[y1:y2, x1:x2])
    plt.show()
```

```
import glob
import cv2
path = glob.glob("FaceRecog/*.jpg")
cv_img = []
for img in path:
    filename = img
    image = Image.open(filename)
    image = image.convert('RGB')
    pixels = asarray(image)
    detector = MTCNN()
    faces = detector.detect_faces(pixels)
    draw_faces(filename, faces)
```

MODEL

```
import pandas as pd
import tensorflow as tf
from tensorflow.keras import models, Sequential, layers, preprocessing
import keras
import os
import mtcnn

file_names=os.listdir("FaceRecog")
NameArray=[]
for name in file_names:
    category=name.split('.')[0]
    if category=='gourishankar':
        NameArray.append('Gourishankar')
    elif category=='Aditya_Panchwagh':
        NameArray.append("Aditya")
    elif category=="Dhananjay_Jha":
        NameArray.append("Dhananjay")
    elif category=='Habil_Bhagat':
        NameArray.append("Habil")
    elif category=="Karan_Mahajan":
        NameArray.append("Karan")
    elif category=='Kartik_Jawanjal':
        NameArray.append("Kartik")
    elif category=="Krish_Shah":
        NameArray.append("Krish")
    elif category=='Manas_Oswal':
        NameArray.append("Manas")
    elif category=="Mayank_Modi":
        NameArray.append("Mayank")
    elif category=='Shubham_Pagare':
        NameArray.append("Shubham")
    elif category=="Vishal_Kasa":
        NameArray.append("Kasa")

train=pd.DataFrame({
    'filename':file_names,
    'category':NameArray
})

from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator,load_img
train_df,validate_df = train_test_split(train,test_size=0.2, random_state=0)
train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
training = preprocessing.image.ImageDataGenerator(rotation_range=5, rescale=1./255, shear_range=0.1, zoom_range=0.2, horizontal_flip=True, width_shift_range=0.1, height_shift_range=0.1)
```

```

trainingdata = training.flow_from_dataframe(train_df, "FaceRecog", x_col='filename', y_col='category', target_size=(224,224), class_mode='categorical')
validation = ImageDataGenerator(rotation_range=5, rescale=1./255, shear_range=0.1, zoom_range=0.2, horizontal_flip=True, width_shift_range=0.1, height_shift_range=0.1)
validationdata = validation.flow_from_dataframe(validate_df, "FaceRecog", x_col='filename', y_col='category', target_size=(224,224), class_mode='categorical')

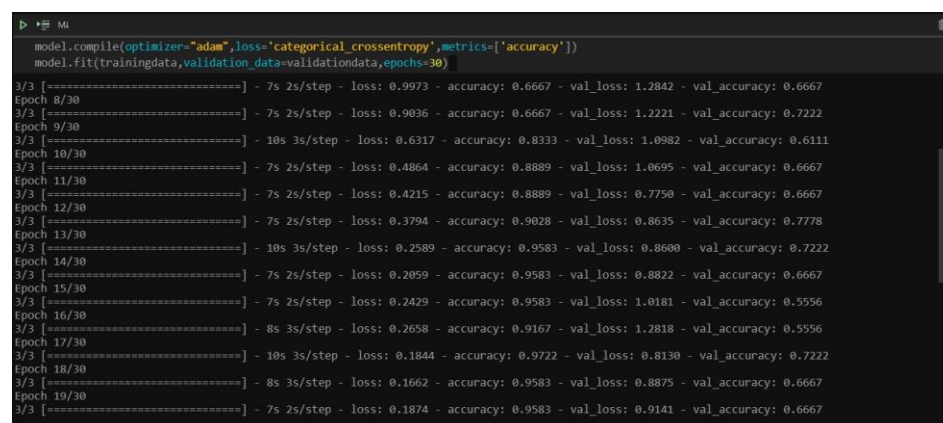
from tensorflow.keras.applications.vgg16 import VGG16
base = VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))
base.trainable = False
model = models.Sequential()
model.add(base)
model.add(layers.Flatten())
model.add(layers.Dense(400, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(trainingdata, validation_data=validationdata, epochs=30)

_, validation_acc = model.evaluate(validationdata, verbose=0)
print(validation_acc)

```

Results:



```

model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(trainingdata, validation_data=validationdata, epochs=30)
3/3 [=====] - 7s 2s/step - loss: 0.9973 - accuracy: 0.6667 - val_loss: 1.2842 - val_accuracy: 0.6667
Epoch 8/30
3/3 [=====] - 7s 2s/step - loss: 0.9036 - accuracy: 0.6667 - val_loss: 1.2221 - val_accuracy: 0.7222
Epoch 9/30
3/3 [=====] - 10s 3s/step - loss: 0.6317 - accuracy: 0.8333 - val_loss: 1.0982 - val_accuracy: 0.6111
Epoch 10/30
3/3 [=====] - 7s 2s/step - loss: 0.4864 - accuracy: 0.8889 - val_loss: 1.0695 - val_accuracy: 0.6667
Epoch 11/30
3/3 [=====] - 7s 2s/step - loss: 0.4215 - accuracy: 0.8889 - val_loss: 0.7750 - val_accuracy: 0.6667
Epoch 12/30
3/3 [=====] - 7s 2s/step - loss: 0.3794 - accuracy: 0.9028 - val_loss: 0.8635 - val_accuracy: 0.7778
Epoch 13/30
3/3 [=====] - 10s 3s/step - loss: 0.2589 - accuracy: 0.9583 - val_loss: 0.8600 - val_accuracy: 0.7222
Epoch 14/30
3/3 [=====] - 7s 2s/step - loss: 0.2059 - accuracy: 0.9583 - val_loss: 0.8822 - val_accuracy: 0.6667
Epoch 15/30
3/3 [=====] - 7s 2s/step - loss: 0.2429 - accuracy: 0.9583 - val_loss: 1.0181 - val_accuracy: 0.5556
Epoch 16/30
3/3 [=====] - 8s 3s/step - loss: 0.2658 - accuracy: 0.9167 - val_loss: 1.2818 - val_accuracy: 0.5556
Epoch 17/30
3/3 [=====] - 10s 3s/step - loss: 0.1844 - accuracy: 0.9722 - val_loss: 0.8130 - val_accuracy: 0.7222
Epoch 18/30
3/3 [=====] - 8s 3s/step - loss: 0.1662 - accuracy: 0.9583 - val_loss: 0.8875 - val_accuracy: 0.6667
Epoch 19/30
3/3 [=====] - 7s 2s/step - loss: 0.1874 - accuracy: 0.9583 - val_loss: 0.9141 - val_accuracy: 0.6667

```

```

3/3 [=====] - 7s 2s/step - loss: 0.1874 - accuracy: 0.9583 - val_loss: 0.9141 - val_accuracy: 0.6667
Epoch 20/30
3/3 [=====] - 10s 3s/step - loss: 0.1647 - accuracy: 0.9861 - val_loss: 0.7870 - val_accuracy: 0.7222
Epoch 21/30
3/3 [=====] - 8s 3s/step - loss: 0.1514 - accuracy: 0.9583 - val_loss: 0.7556 - val_accuracy: 0.7222
Epoch 22/30
3/3 [=====] - 7s 2s/step - loss: 0.1013 - accuracy: 0.9861 - val_loss: 0.7539 - val_accuracy: 0.6667
Epoch 23/30
3/3 [=====] - 7s 2s/step - loss: 0.1262 - accuracy: 0.9861 - val_loss: 0.6245 - val_accuracy: 0.7222
Epoch 24/30
3/3 [=====] - 8s 3s/step - loss: 0.1059 - accuracy: 0.9861 - val_loss: 0.7109 - val_accuracy: 0.6111
Epoch 25/30
3/3 [=====] - 11s 4s/step - loss: 0.0766 - accuracy: 0.9861 - val_loss: 0.9960 - val_accuracy: 0.7222
Epoch 26/30
3/3 [=====] - 11s 4s/step - loss: 0.0996 - accuracy: 0.9583 - val_loss: 0.9721 - val_accuracy: 0.6111
Epoch 27/30
3/3 [=====] - 8s 3s/step - loss: 0.0942 - accuracy: 1.0000 - val_loss: 0.9337 - val_accuracy: 0.7222
Epoch 28/30
3/3 [=====] - 7s 2s/step - loss: 0.0698 - accuracy: 0.9861 - val_loss: 0.7403 - val_accuracy: 0.6667
Epoch 29/30
3/3 [=====] - 11s 4s/step - loss: 0.0663 - accuracy: 1.0000 - val_loss: 0.6007 - val_accuracy: 0.7778
Epoch 30/30
3/3 [=====] - 8s 3s/step - loss: 0.0881 - accuracy: 0.9861 - val_loss: 0.7000 - val_accuracy: 0.6667
tensorflow.python.keras.callbacks.history at 0x14c6e916398

```

```

▶ ▶ M
_, validation_acc = model.evaluate(validationdata, verbose=0)
print(validation_acc)

0.7777777910232544

```

Conclusion:

Thus, we have understood how to create Face recognition system is used and how it is programmed in TensorFlow.

Plus we have also learned to use various face detection algorithms