E-ATTENDANCE

SMART FACE RECOGNITION

Edwin Jose And Greeshma M
Department of Electronics
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

TASK ONE:

SIMPLE FACE RECOGNITION AND UNDERSTANDING CONVOLUTIONAL NEURAL NETWORK

IMPLEMENTATION

DOWNLOADING DATASETS

TRAINING

- LOADING IMAGES
- IMAGE AUGMENTATION
- MODEL CREATION
- MODEL FITTING

IMPLEMENTATION

VALIDATION

PLOTTING PARAMETER

TESTING WITH RANDOM DATA

SAMPLE DATASET

FACES

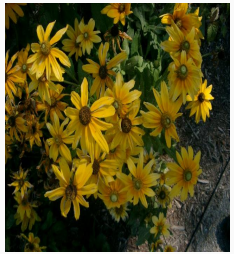





NON FACES








LOADING IMAGES

```
w=28
d=28
# grab the image paths and randomly shuffle them

#random.shuffle(imagePaths)
imagePaths = "data/face"
random.seed(42)
# loop over the input images
for imagePath in os.listdir(imagePaths):
    print(imagePath);
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePaths+"/"+imagePath)
    image = cv2.resize(image,(w, d))
    image = img_to_array(image)
    data.append(image)
    # extract the class label from the image path and update the
    # labels list
    label = 1
    labels.append(label)
##for nonfaces
imagePaths = "data/notface"
random.seed(42)
# loop over the input images
for imagePath in os.listdir(imagePaths):
    print(imagePath);
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePaths+"/"+imagePath)
    image = cv2.resize(image,(w,d))
    image = img_to_array(image)
    data.append(image)
    # extract the class label from the image path and update the
    # labels list
    label = 0
    labels.append(label)
##for nonfaces label is zero
```

IMAGE PROCESING

Image Array Manipulation

------pixel intensities range [0,1]

Partition of Data Matrix using skLearn

-------training(75%)

-------testing(25%)

IMAGE AUGMENTATION

```
# scale the raw pixel intensities to the range [0, 1]
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)

# partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
(trainX, testX, trainY, testY) = train_test_split(data,labels, test_size=0.25, random_state=42)

# convert the labels from integers to vectors
trainY = to_categorical(trainY, num_classes=2)
testY = to_categorical(testY, num_classes=2)

# construct the image generator for data augmentation0
aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
    horizontal_flip=True, fill_mode="nearest")

# initialize the model
print("[INFO] compiling model...")
```

MODEL CREATION

2 convolutional neural network with ReLU activation function

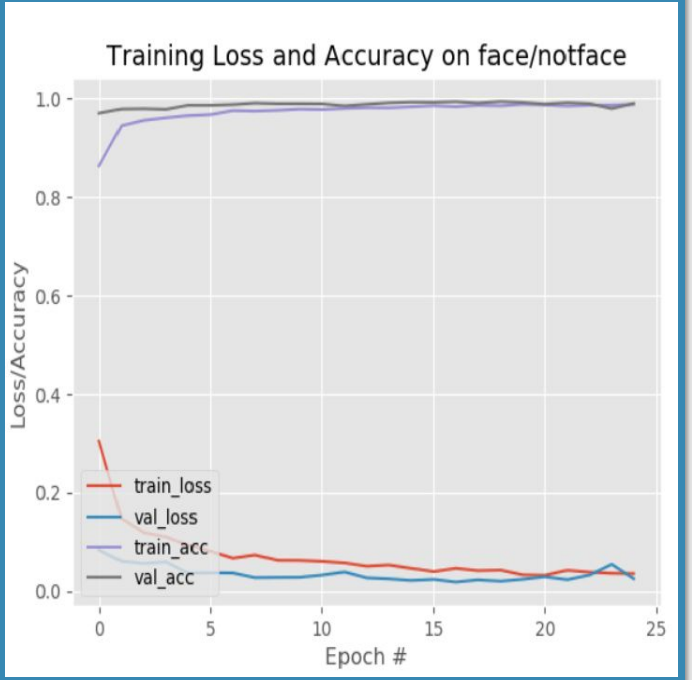
Max Pooling layer

Fully connected layer with ReLu activation

Decision making layer with Softmax function as activation layer

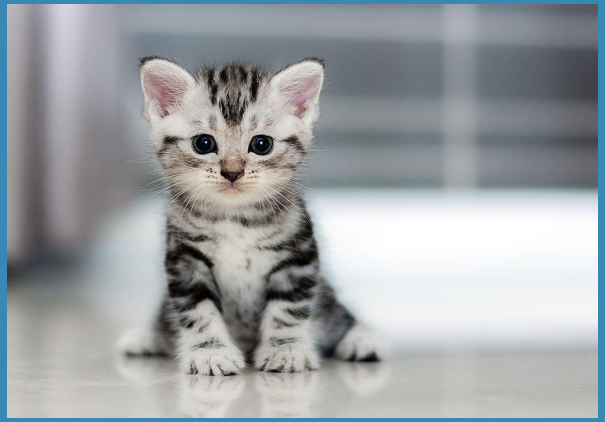
```
###model
# import the necessary packages
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras import backend as K

def build(width, height, depth, classes):
        # initialize the model
        model = Sequential()
        inputShape = (height, width, depth)
        # if we are using "channels first", update the input shape
        if K.image_data_format() == "channels_first":
            inputShape = (depth, height, width)

        # first set of CONV => RELU => POOL layers
        model.add(Conv2D(20, (5, 5), padding="same",
            input_shape=inputShape))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

        # second set of CONV => RELU => POOL layers
        model.add(Conv2D(50, (5, 5), padding="same"))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

        # first (and only) set of FC => RELU layers
        model.add(Flatten())
        model.add(Dense(500))
        model.add(Activation("relu"))

        # softmax classifier
        model.add(Dense(classes))
        model.add(Activation("softmax"))

        # return the constructed network architecture
        return model
```

TRAINING




```
model = build(width=28, height=28, depth=3, classes=2)
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# train the network
print("[INFO] training network...")
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=BS),
    validation_data=(testX, testY), steps_per_epoch=len(trainX) // BS,
    epochs=EPOCHS, verbose=1)

# save the model to disk
print("[INFO] serializing network...")
model.save("face.model")

# plot the training loss and accuracy
plt.style.use("ggplot")
plt.figure()
N = EPOCHS
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["acc"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on face/notface")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.show()
plt.savefig("graph")
```

TESTING



OUTPUT:

[INFO] loading network...
Not face: 99.99%

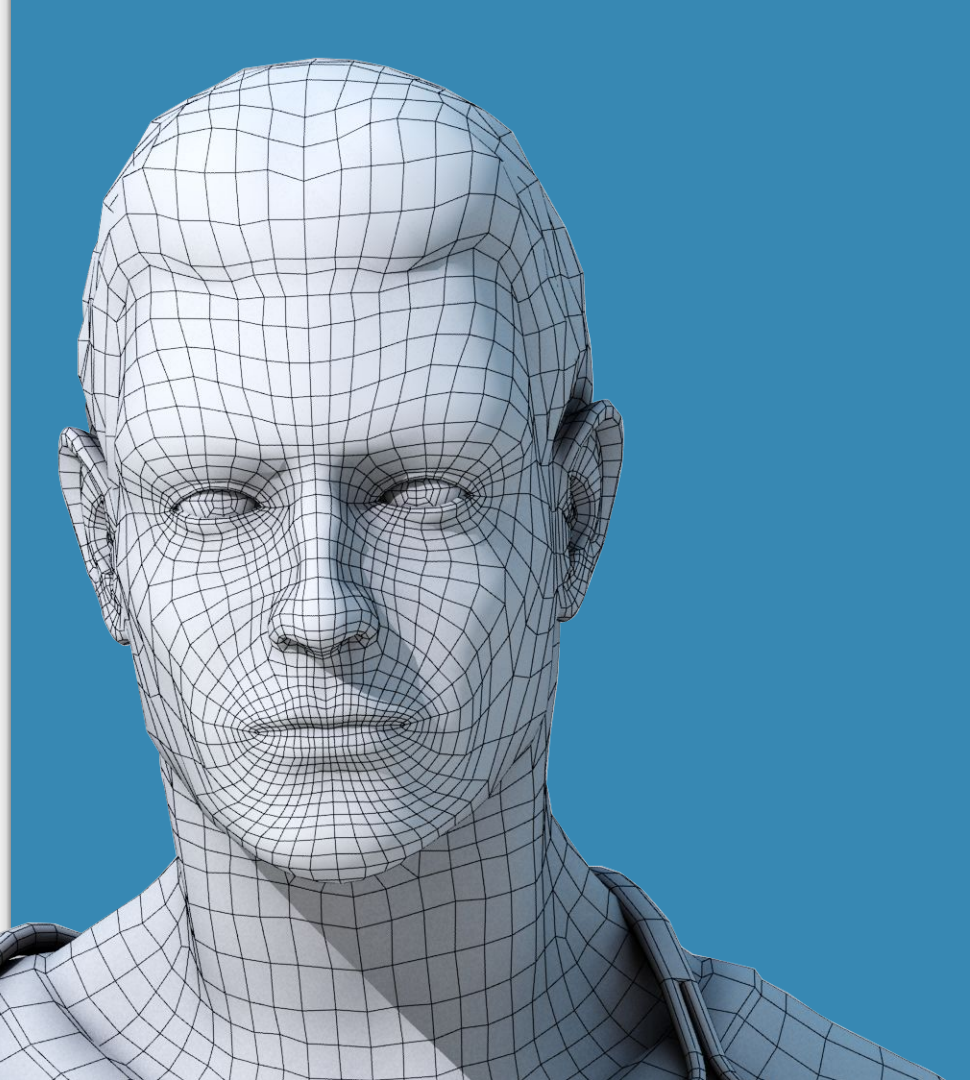
```
# USAGE
# python test_network.py --model face_not_face.model --image images/examples/face_01.png

# import the necessary packages
from keras.preprocessing.image import img_to_array
from keras.models import load_model
import numpy as np
import argparse
import imutils
import cv2

# construct the argument parse and parse the arguments
ima="cat.jpg"


# load the image
image = cv2.imread(ima)
orig = image.copy()

# pre-process the image for classification
image = cv2.resize(image, (28, 28))
image = image.astype("float") / 255.0
image = img_to_array(image)
image = np.expand_dims(image, axis=0)

# load the trained convolutional neural network
print("[INFO] loading network...")
model = load_model("face.model")

# classify the input image
(x, face) = model.predict(image)[0]

# build the label
label = "face" if face > x else "Not face"
proba = face if face > x else x
label = "{}: {:.2f}%".format(label, proba * 100)
print(label);
# draw the label on the image
output = imutils.resize(orig, width=400)
cv2.putText(output, label, (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 2)

# show the output image
cv2.imshow("Output", output)
cv2.waitKey(0)
```

PENDING IMPLEMENTATION in task one:

- Dataset refining program
  False images should be removed from nonface dataset
- Confusion Matrix
  Program to display confusion matrix

FURTHER TASKS

- Face Detection With Face Localisation
- Multiple Face Detection
- Gender Detection(Optional)
- Face Recognition



STEPS FOR FURTHER DEVELOPMENT

KNOWLEDGE DEVELOPMENT

- Understanding architectures
- Rcnn and ssd example reviews
- Facenet implementation using MTCNN

Divide and Conquer

Our Coding Approach

https://github.com/edwinjose900/FR_CUSAT/projects/1
https://github.com/edwinjose900/FR_CUSAT

feedback