IP PROJECT

Kenny: main.py 1-80

import os.path

import datetime

import pickle

import tkinter as tk

import cv2

from PIL import Image, ImageTk

import face\_recognition

import util

from test import test

class App:

def \_\_init\_\_(self):

self.main\_window = tk.Tk()

self.main\_window.geometry("1200x520+350+100")

self.login\_button\_main\_window = util.get\_button(self.main\_window, 'login', 'green', self.login)

self.login\_button\_main\_window.place(x=750, y=200)

self.logout\_button\_main\_window = util.get\_button(self.main\_window, 'logout', 'red', self.logout)

self.logout\_button\_main\_window.place(x=750, y=300)

self.register\_new\_user\_button\_main\_window = util.get\_button(self.main\_window, 'register new user', 'gray',

self.register\_new\_user, fg='black')

self.register\_new\_user\_button\_main\_window.place(x=750, y=400)

self.webcam\_label = util.get\_img\_label(self.main\_window)

self.webcam\_label.place(x=10, y=0, width=700, height=500)

self.add\_webcam(self.webcam\_label)

self.db\_dir = './db'

if not os.path.exists(self.db\_dir):

os.mkdir(self.db\_dir)

self.log\_path = './log.txt'

def add\_webcam(self, label):

if 'cap' not in self.\_\_dict\_\_:

self.cap = cv2.VideoCapture(2)

self.\_label = label

self.process\_webcam()

def process\_webcam(self):

ret, frame = self.cap.read()

self.most\_recent\_capture\_arr = frame

img\_ = cv2.cvtColor(self.most\_recent\_capture\_arr, cv2.COLOR\_BGR2RGB)

self.most\_recent\_capture\_pil = Image.fromarray(img\_)

imgtk = ImageTk.PhotoImage(image=self.most\_recent\_capture\_pil)

self.\_label.imgtk = imgtk

self.\_label.configure(image=imgtk)

self.\_label.after(20, self.process\_webcam)

def login(self):

label = test(

image=self.most\_recent\_capture\_arr,

model\_dir='/home/phillip/Desktop/todays\_tutorial/27\_face\_recognition\_spoofing/code/face-attendance-system/Silent-Face-Anti-Spoofing/resources/anti\_spoof\_models',

device\_id=0

)

if label == 1:

name = util.recognize(self.most\_recent\_capture\_arr, self.db\_dir)

if name in ['unknown\_person', 'no\_persons\_found']:

util.msg\_box('Ups...', 'Unknown user. Please register new user or try again.')

else:

util.msg\_box('Welcome back !', 'Welcome, {}.'.format(name))

with open(self.log\_path, 'a') as f:

f.write('{},{},in\n'.format(name, datetime.datetime.now()))

f.close()

else:

util.msg\_box('Hey, you are a spoofer!', 'You are fake !')

***Aditya: util.py 1-74***

import os

import pickle

import tkinter as tk

from tkinter import messagebox

import face\_recognition

def get\_button(window, text, color, command, fg='white'):

button = tk.Button(

window,

text=text,

activebackground="black",

activeforeground="white",

fg=fg,

bg=color,

command=command,

height=2,

width=20,

font=('Helvetica bold', 20)

)

return button

def get\_img\_label(window):

label = tk.Label(window)

label.grid(row=0, column=0)

return label

def get\_text\_label(window, text):

label = tk.Label(window, text=text)

label.config(font=("sans-serif", 21), justify="left")

return label

def get\_entry\_text(window):

inputtxt = tk.Text(window,

height=2,

width=15, font=("Arial", 32))

return inputtxt

def msg\_box(title, description):

messagebox.showinfo(title, description)

def recognize(img, db\_path):

# it is assumed there will be at most 1 match in the db

embeddings\_unknown = face\_recognition.face\_encodings(img)

if len(embeddings\_unknown) == 0:

return 'no\_persons\_found'

else:

embeddings\_unknown = embeddings\_unknown[0]

db\_dir = sorted(os.listdir(db\_path))

match = False

j = 0

while not match and j < len(db\_dir):

path\_ = os.path.join(db\_path, db\_dir[j])

file = open(path\_, 'rb')

embeddings = pickle.load(file)

match = face\_recognition.compare\_faces([embeddings], embeddings\_unknown)[0]

j += 1

if match:

return db\_dir[j - 1][:-7]

else:

return 'unknown\_person'

Fouzan: main.py 82-155

import os

import pickle

import tkinter as tk

from tkinter import messagebox

import face\_recognition

def get\_button(window, text, color, command, fg='white'):

button = tk.Button(

window,

text=text,

activebackground="black",

activeforeground="white",

fg=fg,

bg=color,

command=command,

height=2,

width=20,

font=('Helvetica bold', 20)

)

return button

def get\_img\_label(window):

label = tk.Label(window)

label.grid(row=0, column=0)

return label

def get\_text\_label(window, text):

label = tk.Label(window, text=text)

label.config(font=("sans-serif", 21), justify="left")

return label

def get\_entry\_text(window):

inputtxt = tk.Text(window,

height=2,

width=15, font=("Arial", 32))

return inputtxt

def msg\_box(title, description):

messagebox.showinfo(title, description)

def recognize(img, db\_path):

# it is assumed there will be at most 1 match in the db

embeddings\_unknown = face\_recognition.face\_encodings(img)

if len(embeddings\_unknown) == 0:

return 'no\_persons\_found'

else:

embeddings\_unknown = embeddings\_unknown[0]

db\_dir = sorted(os.listdir(db\_path))

match = False

j = 0

while not match and j < len(db\_dir):

path\_ = os.path.join(db\_path, db\_dir[j])

file = open(path\_, 'rb')

embeddings = pickle.load(file)

match = face\_recognition.compare\_faces([embeddings], embeddings\_unknown)[0]

j += 1

if match:

return db\_dir[j - 1][:-7]

else:

return 'unknown\_person'