**Chapter 2**

During the work on this project, about 1 GB of data was collected with photographs, these photos were divided into 5 target groups that will be used to identify people by gender and age.  
In total, about 80,000 different photos were collected and edited in the public domain on the Internet, as well as a screen for scrubbing deleted photos from social networks was compiled.  
  
A test system was also created on the basis of which, on a small amount of data, our neural network is trained at the moment based on a primitive algorithm, but I remember this algorithm tests such important aspects as access to a wireless ip-camera as well as the speed of decision making and the power of a potential computer server handler.   
As a result of testing the system, the need to increase the accuracy of decision making of the neural network algorithm was revealed.   
at the moment, our script, written in Python using machine learning libraries, as well as neural networks, is able to connect to a wireless ip-camera to process and receive a video stream and information about the time interval, converting this information into a label and determining the gender and age of a potential person, which gets recorded.  
all this functionality occurs in real time, it allows you to accurately determine the number of people, as well as their gender and age, who come to a potential store, a potential area of ​​interest.  
A lot of work was done to prepare the data for its cleaning and also to divide this group of photos into potentially target characteristics such as gender and age, and also sorting and removing irrelevant data for our area as well as data that would create unnecessary noise and bias in the decision-making system.  
As a result of the scraper's work and downloading cheek photos from social networks, about 20,000 photos were obtained, which were subsequently cropped and distributed according to targeted features, and about 60,000 more photos were found on the Internet in open access, which were also cleaned and divided into appropriate clusters and For further use when teaching neural.   
so at the moment we have about 80,000 photos available that can be used to build a neural network, which is equivalent to one and a half GB of data. An efficient scraper script that can potentially extract a large number of photos at the proper time intervals which can then also be used as training our artificial model.   
A route was laid a channel for communication between an IP camera installed in a potential place of interest and a server by a processor information running in a neural network with a record of time intervals as well as information about how many of which age group entered at certain intervals.  
We will conduct a deep analysis of existing offers on the market as a result of which some changes were made to the initial concept of the system's operation, as well as an analysis of the effectiveness of decision-making, as well as a test for the erroneousness of decision-making in our system.  
Then you can familiarize yourself with the source code with the copier as well as the systems for connecting to a wireless IP camera and the initial test system for recognizing a potential category of persons of interest by age and gender.

Cropper:

from PIL import Image  
from autocrop import Cropper  
  
cropper = Cropper()  
  
for i in range(0, 3000):  
  
 try:  
 # Get a Numpy array of the cropped image  
 cropped\_array = cropper.crop('#smile/' + str(i)+ '\_\_.jpg')  
 # Save the cropped image with PIL  
  
 cropped\_image = Image.fromarray(cropped\_array)  
 print(str(i) + ": Face found\n")  
 cropped\_image.save('cropped/' + str(i)+ '\_3.jpg')  
  
 except:  
 print(str(i) + ": Face not found\n")

Renamer:

import os  
  
  
# Function to rename multiple files  
def main():  
 for count, filename in enumerate(os.listdir("#smile")):  
 dst = str(count) + "\_\_.jpg"  
 src = '#smile/' + filename  
 dst = '#smile/' + dst  
  
 # rename() function will  
 # rename all the files  
 os.rename(src, dst)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Calling main() function  
 main()

Scrapper:

import cv2  
import sys  
  
# imagePath = sys.argv[1]  
for i in range(0, 3000):  
 try:  
 image = cv2.imread('#smile/' + str(i) + '\_\_.jpg')  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
  
 faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")  
 faces = faceCascade.detectMultiScale(  
 gray,  
 scaleFactor=1.3,  
 minNeighbors=3,  
 minSize=(30, 30)  
 )  
  
 print("[INFO] Found {0} Faces.".format(len(faces)))  
  
 for (x, y, w, h) in faces:  
 cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)  
 roi\_color = image[y:y + h, x:x + w]  
 print("[INFO] Object found. Saving locally.")  
 cv2.imwrite('cropped/' + str(w) + str(h) + '\_faces.jpg', roi\_color)  
  
 status = cv2.imwrite('detected/' + str(i) + '1\_.jpg', image)  
 print("[INFO] Image faces\_detected.jpg written to filesystem: ", status)  
 except:  
 print("Error")

Part of remote cam code and server code: (full version in git hub repo):

import numpy as np  
import requests  
import cv2  
import pickle  
  
face\_cascade = cv2.CascadeClassifier('cascades/data/haarcascade\_frontalface\_alt2.xml')  
recognizer = cv2.face.LBPHFaceRecognizer\_create()  
recognizer.read("trainner.yml")  
  
  
labels = {}  
  
with open("labels.pickle", 'rb') as f:  
 og\_labels = pickle.load(f)  
 labels = {v: k for k, v in og\_labels.items()}  
  
cap = cv2.VideoCapture(0)  
  
while (True):  
  
 # Capture frame-by-frame  
 ret, frame = cap.read()  
  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)  
  
 for (x, y, w, h) in faces:  
 #print(x, y, w, h)  
 roi\_gray = gray[y:y+h, x:x+w]  
 roi\_color = frame[y:y + h, x:x + w]  
  
 id\_, conf = recognizer.predict(roi\_gray)  
 if conf >= 50: #and conf <= 85:  
 print(id\_)  
 print(labels[id\_])  
 font = cv2.FONT\_HERSHEY\_SIMPLEX  
 name = labels[id\_]  
 color = (255, 255, 255)  
 stroke = 2  
 cv2.putText(frame, name, (x, y), font, 1, color, stroke, cv2.LINE\_AA)  
  
  
 img\_item = "my-image.png"  
 cv2.imwrite(img\_item, roi\_gray)  
  
 color = (255, 0, 0) #BGR  
 stroke = 2  
 end\_cord\_x = x + w  
 end\_cord\_y = y + h  
 cv2.rectangle(frame, (x, y), (end\_cord\_x, end\_cord\_y), color, stroke)  
  
  
 # Display the resulting frame  
 cv2.imshow('frame', frame)  
 if cv2.waitKey(20) & 0xFF == ord('q'):  
 break  
  
# When everything done, release the capture  
cap.release()  
cv2.destroyAllWindows()