International Journal of Engineering, Management and Humanities (IJEMH)
Volume 2, Issue 4, pp: 162-165 www.ijemh.com

Smart Attendance Using Caffe and Open Face

B.Nageswara Rao
[1,2,3,4]*Student, Lendi Institute of Engineering and Technology,Vizianagaram,AndhraPradesh Corresponding Author: I.Keerthana,B.Aravind,A.Vandana,G.Sindhuja Naidu*

Date of Submission: 06-07-2021 Date of Acceptance: 21-07-2021

ABSTRACT
Human Face Detection has become a major field of interest in current resear ch because there is no deterministic algorithm to find face(s) in a given image further the algorithms t hat exist are very much specific to the kind of image s they would take as input and detect faces. The pro blem is todetect faces in the given, coloured class gr oup photograph. The approach, we take is a mixture of heuristic and known algorithms.The main idea of doing this project is decrease the continuous video c apture for the face recognition attendance, where we provide a single image which contains all the faces the machine learning algorithm will detect all the fa ces atonce and mark the attendance to those persons who are in the picture.Face detection is the key poin t in automatic face recognition system. This project i ntroduces the face detection algorithm with the Mod ule of OpenCV anda detailed analysis of the face det ection and attendance to those faced will be presented.

KEYWORDS: Supervised Learning, Deep Learning, Feature Extraction, Computer Vision

I. INTRODUCTION

The face is one of the easiest ways to distinguish the individual identity of each other. Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity. Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognize a face as individuals. Stage is then replicated and developed as a model for facial image recognition (face recognition) is one of the much-studied biometrics technology and developed by experts. There are two kinds of methods that are currently popular in developed face recognition pattern namely, Eigenface method and Fisher face method. Facial image recognition Eigenface method is based on the reduction of face dimensional space using Principal Component Analysis (PCA) for facial features. The main purpose of the use of PCA on face recognition using Eigen faces was formed (face space) by finding the eigenvector corresponding to the largest eigenvalue of the face image. The area of this project face detection system with face recognition is Image processing.

In this paper i.e, Smart Attendance Using Caffe and Open Face The model we developed is a Supervised learning algorithm that takes a training dataset of faces and then recognizes the faces based on the data set provided to the algorithm for this purpose we used a deep learning framework Caffe and Open Face as our algorithms to obtain the final output.

The algorithm we developed is a deep learning framework. Deep learning is the new big trend in machine learning. It had many recent successes in computer vision, automatic speech recognition and natural language processing .Due to its adverse features we have chosen DL framework for more efficiency and accuracy.

For this we will use some Python code and a popular open source deep learning framework called Caffe to build the classifier. Our classifier will be able to achieve a classification accuracy of 97%.

II. PROBLEM DEFINITION

Our goal is to build a deep learning algorithm capable of recognizing the correct faces for marking the attendance of a particular student without any interference of the professor. In Deep Learning, this type of problem is called classification.(here present or absent for each student).

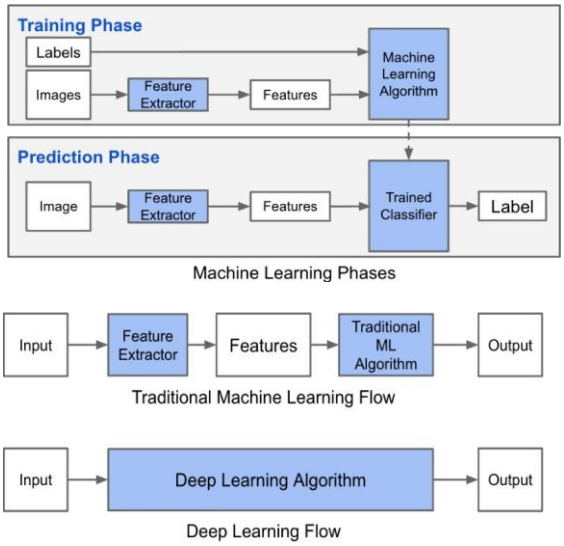
III. CLASSIFICATION USING TRADITIONAL MACHINE LEARNING AND DEEP LEANING

Classification using a machine learning algorithm has 2 phases:

International Journal of Engineering, Management and Humanities (IJEMH)
Volume 2, Issue 4, pp: 162-165 www.ijemh.com

- **Training phase:** In this phase, we train a machine learning algorithm using a dataset comprised of the images and their corresponding labels. The training phase for an image classification problem has 2 main steps:
- **Feature Extraction:** In this phase, we utilize domain knowledge to extract new features that will be used by the machine learning algorithm. HoG and SIFT are examples of features used in image classification.
- **Model Training:** In this phase, we utilize a clean dataset composed of the images' features and the corresponding labels to train the machine learning model
- **Prediction phase:** In this phase, we utilize the trained model to predict labels of unseen images.In the prediction phase, we apply the same feature extraction process to the new images and we pass the features to the trained machine learning algorithm to predict the label.

The main difference between traditional machine learning and deep learning algorithms is in the feature engineering. In traditional machine learning algorithms, we need to hand-craft the features. By contrast, in deep learning algorithms feature engineering is done automatically by the algorithm. Feature engineering is difficult, time-consuming and requires domain expertise. The promise of deep learning is more accurate machine learning algorithms compared to traditional machine learning with less or no feature engineering.



Machine Learning Phases Vs Deep Learning Flow

IV. CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks have different types of models developed recently in that we have used the CAFFE and OPEN FACE for our algorithm

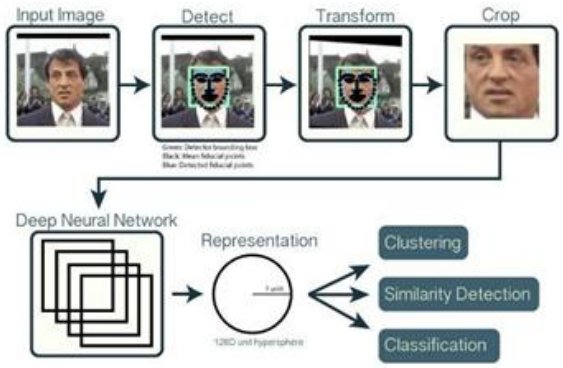
CAFFE Overview

Caffe is a deep learning framework developed by Berkeley Vision and Learning Centre (BVLC), released under the BSD 2-Clause license. Nvidia Digits is based on Caffe and can be used as GUI and convenient interface for multi-GPU systems. The Caffe framework offers more flexible CNN architectures than any other DL framework and is highly optimized for speed by CUDA and CuDNN support. It is written in C++ and has Python and Matlab bindings.

There are 4 steps in training a CNN using Caffe:

- **Step 1 - Data preparation**: In this step, we clean the images and store them in a format that can be used by Caffe. We will write a Python script that will handle both image pre-processing and storage.
- **Step 2 - Model definition:** In this step, we choose a CNN architecture and we define its parameters in a configuration file with extension **.prototxt.**
- **Step 3 - Solver definition:** The solver is responsible for model optimization. We define the solver parameters in a configuration file with extension .prototxt.
- **Step 4 - Model training:** We train the model by executing one Caffe command from the terminal. After training the model, we will get the trained model in a file with extension .caffemodel.

After the training phase, we will use the .caffe trained model to make predictions of new unseen data. We will write a Python script to this.



International Journal of Engineering, Management and Humanities (IJEMH)
Volume 2, Issue 4, pp: 162-165 www.ijemh.com

SOURCE CODE

1. TO EXTRACT EMBEDDINGS FROM THE IMAGE

```
from imutils import paths import face_recognition
import pickle
import cv2 import os
#get paths of each file in folder named Images
#Images here contains my data(folders of various
persons)
imagePaths = list(paths.list_images('Images'))
knownEncodings = [] knownNames = []
# loop over the image paths for (i, imagePath) in
enumerate(imagePaths):
# extract the person name from the image path
name = imagePath.split(os.path.sep)[-2]

# load the input image and convert it from BGR
(OpenCV ordering)
# to dlib ordering (RGB)
image = cv2.imread(imagePath) rgb =
cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)
#Use Face_recognition to locate faces boxes =
face_recognition.face_locations(rgb,model
='hog')
# compute the facial embedding for the face
encodings = face_recognition.face_encodings(rgb,
boxes)
# loop over the encodings for encoding in
encodings:
knownEncodings.append(encoding)
knownNames.append(name)
#save emcodings along with their names in
dictionary data
data = {"encodings": knownEncodings, "names":
knownNames}
#use pickle to save data into a file for later use
f = open("face_enc", "wb")
f.write(pickle.dumps(data)) f.close()
```

2. TO DETECT FACES (Training the Model)

```
from sklearn.preprocessing import LabelEn coder
from sklearn.svm import SVC import argparse
import pickle
# construct the argument parser and parse the
arguments
ap = argparse.ArgumentParser() ap.add_argument("-
e", "-- embeddings", required=True,
help="path to serialized db of facial embeddings")
ap.add_argument("-r", "-- recognizer",
required=True,

help="path to output model trained to recognize
faces")
ap.add_argument("-l", "-- le", required=True,
help="path to output label encoder") args =
vars(ap.parse_args())

# load the face embeddings
print("[INFO] loading face embeddings...") data =
pickle.loads(open(args["embeddings "], "rb").read())

# encode the labels
print("[INFO] encoding labels...") le =
LabelEncoder()
labels = le.fit_transform(data["names"])
# train the model used to accept the 128-d
embeddings of the face and
# then produce the actual face recognition
print("[INFO] training model...") recognizer =
SVC(C=1.0, kernel="linear", probability=True)
recognizer.fit(data["embeddings"], labels)

import os

#find path of xml file containing haarcascade file
cascPathface = os.path.dirname( cv2._file_) +
"/data/haarcascade_frontalface_alt2.xml" # load the
harcaascade in the cascade classifier
faceCascade = cv2.CascadeClassifier(cascPathface)
# load the known faces and embeddings saved in
last file
data = pickle.loads(open('face_enc', "rb").read())
#Find path to the image you want to detect face and
pass it here
image = cv2.imread(Path-to-img) rgb =
cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
#convert image to Greyscale for haarcascade
gray = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(gray,
scaleFactor=1.1, minNeighbors=5, minSize=(60,
60), flags=cv2.CASCAD

# write the actual face recognition model to disk
f = open(args["recognizer"], "wb")
f.write(pickle.dumps(recognizer)) f.close()

# write the label encoder to disk f = open(args["le"],
"wb") f.write(pickle.dumps(le)) f.close()
```

3. FACE RECOGNITION IN IMAGES

```
import face_recognition import imutils
import pickle import time import cv2

E_SCALE_IMAGE)

# the facial embeddings for face in input encodings
```

```
= face_recognition.face_encodings(rgb) names = []
# loop over the facial embeddings incase # we have
multiple embeddings for multiple fcaes
for encoding in encodings:
#Compare encodings with encodings in
matches =
face_recognition.compare_faces(data["enco dings"],
encoding)
#set name =inknown if no encoding matches
name = "Unknown"
# check to see if we have found a match if True in
matches:
#Find positions at which we get True and store them
matchedIdxs = [i for (i, b) in enumerate(matches) if
b]
counts = {}
# loop over the matched indexes and maintain a
count for
# each recognized face face for i in matchedIdxs:

name = data["names"][i]
counts[name] = counts.get(name, 0) + 1 name =
max(counts, key=counts.get) names.append(name)
for ((x, y, w, h), name) in zip(faces, names):
# rescale the face coordinates
# draw the predicted face name on the image
cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255,
0), 2)
cv2.putText(image, name, (x, y),
cv2.FONT_HERSHEY_SIMPLEX,
0.75, (0, 255, 0), 2)

cv2.imshow("Frame", image) cv2.waitKey(0)
```

V. OBESERVATIONS

From the above model we can easily mark the attendance of a person automatically with out any human intervention. This model can be used by professors in a college,employees in a company,staff in an organization etc. It works in a efficient way like a unique biometric system does without any high configuration systems

REFERENCES

[1]. Beymer, D. and Poggio, T. (1995) Face Recognition From One Example View, A.I. Memo No. 1536, C.B.C.L. Paper No. 121. MIT

[2]. Adelson, E. H., and Bergen, J. R. (1986) The Extraction of Spatio-Temporal Energy in Human and Machine Vision, Proceedings of Workshop on Motion: Representation and Analysis (pp. 151-155) Charleston, SC; May 7-9

[3]. Heisele, B. and Poggio, T. (1999) Face Detection. Artificial Intelligence Laboratory. MIT