National Conference on Engineering Innovations and Solutions (NCEIS – 2018)
International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 4 | Issue 6 | ISSN : 2456-3307

Facial Key-Point Detection and Real-Time Filtering Using Convolutional Neural Network

Abhishek G C[1], Pramukh B V[1], Pranav T V[1], Shravan S Vasista[1], B S Prathibha[2]
[1]Student VIII Sem, Department of Information Science & Engineering, NIE, Mysuru, Karnataka, India
[2]Assisstant Professor, Department of Information Science & Engineering, NIE, Mysuru, Karnataka, India

ABSTRACT

In this paper, an effort is made to combine the knowledge of computer vision techniques and deep learning to build and end-to-end facial keypoint recognition system. Facial keypoints include points around the eyes, nose, and mouth on any face and are used in many applications, from facial tracking to emotion recognition. The partially complete module should be able to take in any image containing faces and identify the location of each face and their facial keypoints. The proposed facial recognition system uses few of the many computer vision algorithms built into the OpenCV library and are implemented at the basic level. This expansive computer vision library is open source and is still growing. The proposed system does real time filtering and facial key point detection. This implementation uses a Convolutional Neural Network to train the system at each step, visualize the loss and learn in the next detection.

Keywords: Convolutional Neural Network, Face Detection, Facial Keypoints, Facial Recognition, Real-time Filtering.

I. INTRODUCTION

Our face plays an elementary role in day to day social interactions. Today, adoption of facial biometric identification technology, facial tracking, emotion recognition have become a global trend covering a wide gamut of applications ranging from business(commercial) to social media and even law enforcement. Some countries and their governments are using facial biometrics for automating the immigration process. As a part of boundary security enforcement, governments are using facial recognition scanners with critical information about criminals and wanted individuals pre-loaded to them. Further driving licences and identity cards contain facial recognition processes for verification and authentication. In the recent past, major companies like Google, Snapchat and Facebook etc, have been using facial detection and recognition systems to tag, blur and identify images real-time. Even companies like Apple, Samsung, LG, Panasonic have also incorporated face recognition, eye detection, emotion recognition systems to their smart devices. All the aforementioned applications demand highly accurate and efficient face recognition technology which can run with minimum loss in real-time.

The existing face recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face. These algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw and use it to detect faces in images from different data sets. Recognition algorithms can be divided into two main approaches, geometric, which looks at distinguishing features, or photometric, which is a statistical approach that distills an image into values and compares the values with templates to eliminate variances. However, these recognition systems do not include any deep learning techniques such as G-CNN and F-CNN[3] (Convolutional Neural Network) to train the image recognition system. The current systems are simply based on image pre-processing, denoising the image and recognizing the facial points. These systems are incapable of learning from each image and hence show fairly lower results when used real-time. Although these methods achieve promising results in terms of recognition rate with constrained scenario datasets such as CMU, FEI, Grimace, Face95 and Yale, these features show considerable degradation in uncontrolled scenario datasets such as FIE and LWF.

Convolutional Neural Networks have achieved substantial success in the field of patttern recognition including object detection and recognition, speech recognition, OCR, action and handwriting recognition due to the fact that it addresses the problems of Multi-Layer Perception Neural Networks. Hence, the proposed system uses a Convolutional Neural Network which is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. The system uses Keras which is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. Tensorflow is an opensource tool for machine learning which contains a wide range of functionality, TensorFlow[6] is mainly designed for deep neural network models.

The main objective of the proposed system is to implement a real-time facial keypoint recognition system using OpenCV library for image processing and Keras with Tensorflow backend for building and training a deep learning CNN.

II. METHODS AND MATERIAL

This facial-keypoint recognition system can be built and run on any platform as it uses an open source Keras module with Tensorflow backend and OpenCV library. The modules can be programmed using Python. The system comprises of three module

1. PRE-PROCESSING AND FACE DETECTION

This module is aimed towards investigating the OpenCV library to process images from a dataset and apply the tool to detect faces in the images.

1.1 Detect Faces Using a Haar Cascade Classifier

At its root face detection is a classification problem - that is a problem of distinguishing between distinct classes of things. With face detection these distinct classes are 1) images of human faces and 2) everything else. This is achieved by using Haar feature-based cascade classifiers.[1]

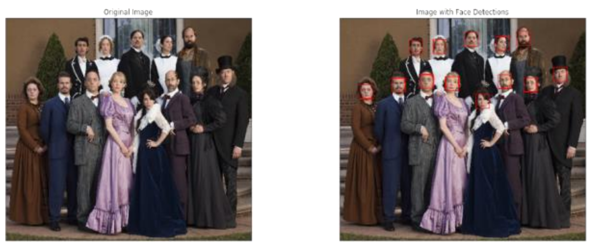


Figure 1.1 Face Detection using Haar Cascasde Classifier

1.2 Adding Eye Detection using pre-trained Haar Cascade Classifier

In this step, pre-trained Haar Cascade[1] Classifiers are used to detect faces in an image and also to recognise the position of the eye in the detected face.[2]

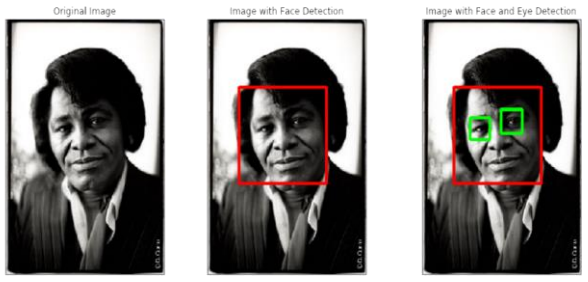


Figure 1.2 Eye Detection using pre-trained Haar Cascade Classifier

1.3 De-noise an Image for Better Face Detection

In the context of face detection, the problem with an image like this is that - due to noise - we may miss some faces or get false detections. Using OpenCV's built in color image de-noising functionality, we de-noise the images enough so that all the faces in the image are properly detected.

1.4 Blur an Image and Perform Edge Detection

Edge detection is a dimension reduction technique - by keeping only the edges of an image we get to throw away a lot of non-discriminating information. And typically the most useful kind of edge-detection is one that preserves only the important, global structures (ignoring local structures that aren't very discriminative). So removing local structures / retaining global structures is a crucial pre-processing step to performing edge detection in an image, and blurring can do just that

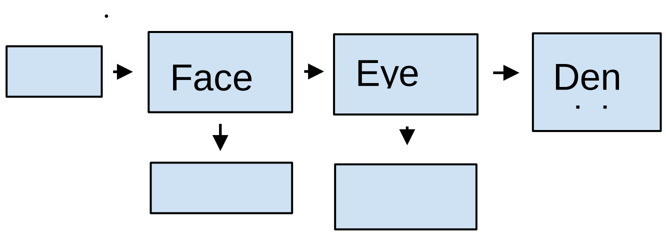


Fig 1.3 Pre-processing and Face Detection

2. TRAINING A CONVOLUTIONAL NEURAL NETWORK TO DETECT FACIAL KEYPOINTS

Once the system is accustomed with face detection, it needs to be trained to perform facial keypoint detection. This is achieved by creating a Sequential CNN using the Keras backend[5]. Images from the Kaggle dataset are used here to train and test the CNN visualise the loss and detect upto 15 facial keypoints. Facial keypoints include facial landmarks around eyes, nose and mouth. This system is aimed towards identifying these facial keypoints real time using the trained CNN.

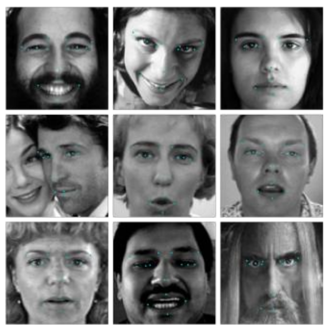


Figure 1.4 Images with Facial Keypoints Detected(using Kaggle Dataset)

3. COMPLETING THE CV PIPELINE

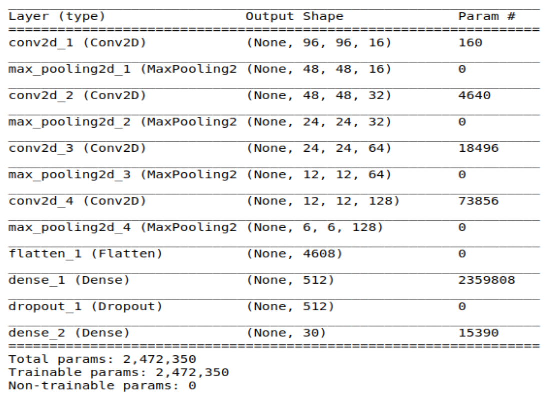
With the work done in Modules 1 and 2 of the project, along with a freshly trained facial keypoint detector, we can now complete the full pipeline. That is given a color image containing a person or persons we can now

1. Detect the faces in this image automatically
2. Predict the facial keypoints in each face detected in the image
3. Mark predicted keypoints on each face detected and use it various application

III. RESULTS AND DISCUSSION

For each training image, there are two landmarks per eyebrow (**four** total), three per eye (**six** total), **four** for the mouth, and **one** for the tip of the nose.

The system is trained to predict these facial keypoints using the following Convolutional Neural Network architecture



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 96, 96, 16)        160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 32)        4640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               2359808
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 30)                15390
=================================================================
Total params: 2,472,350
Trainable params: 2,472,350
Non-trainable params: 0
```

Figure 2.1 CNN Architecture with total trainable parameters

The validation of MSE(Mean Squared Error) for different Convolutional Neural Networks for recognition and detection of facial keypoints is obtained as follows

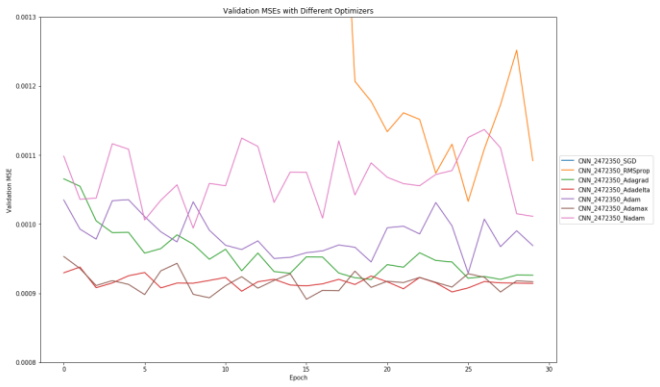


Chart 2.1 Plot of Validation MSE with different Optimisers

The training and validation loss for the designed neural network is as follows

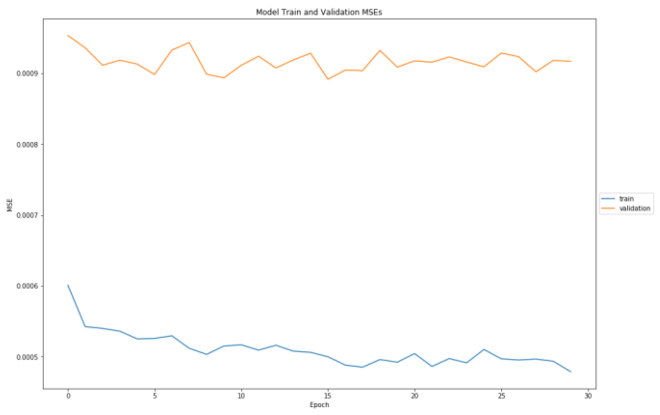


Chart 2.2 Plot of Train and Validation MSE

To reduce overfitting and increase the generalization abilities of the model I experimented with **four different level of dropout rates [0.25, 0.3, 0.4, 0.5] between the last two denses layers** and got the best generalization results with a **dropout rate of 0.5**.

Finally, as a final hyperparameter tuning step, as shown above, **all the optimizers provided by Keras** were implemented and ended up selecting the one with the **lowest validation MSE using 20% of the training data as validation data**. Best results were obtained using the Adamax and Adadelta optimizers and for the proposed implementation **Adamax optimizer is chosen for the final model**.

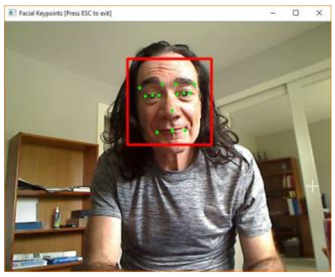


Figure 2.2 Facial Keypoint Detector with Real-time filtering

IV. CONCLUSION

The proposed system is able to detect and mark facial keypoints in any image. It has a similar but better performance when compared to older techniques. Also, the older techniques do not include facial keypoint detectors for facial biometrics. This system is capable to functioning real time and hence can be used in wide range of aforementioned applications like Facebook, Snapchat and Google. The Network architectures proposed in this paper effectively reduces processing time, improves accuracy of the CNN running on low-end GPUs or even on CPUs. Also, the system proposed here has room for change. The OpenCV library which is used for image processing is an expanding library and hence this system is built to incorporate any changes in the library. Further enhancements for this paper would be to expand the network architecture to larger datasets, improve performance and accuracy and make the system run efficiently even on low computing devices and provide range of applications for facial biometrics.

V. REFERENCES

[1] "**Real-time eye state detection system using haar cascade classifier and circular hough transform**" by Norma Latif Fitriyani & Chuan-Kai Yang, Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan and Muhammad Syafrudin, Department of Industrial and Systems

Engineering, Dongguk University, Seoul, South Korea. Added to IEEE Xplore: 29 December 2016

[2] **"An approach to face detection and alignment using hough transformation with convolution neural network"** by Oshin Misra & Ajit Singh, CS&E Department B.T.K.I.T Dwarahat Almora. Published in: Advances in Computing, Communication, & Automation (ICACCA) (Fall), International Conference on 30 Sept.-1 Oct. 2016. Added to IEEE Xplore: 21 November 2016.

[3] **"G-CNN and F-CNN: Two CNN based architectures for face recognition"** by A Vinay, Desanur Naveen Reddy & Abhishek C. Sharma, PES University Bengaluru, India. Published in: Big Data Analytics and Computational Intelligence (ICBDAC), 2017 International Conference on 23-25 March 2017. Added to IEEE Xplore: 19 October 2017

[4] **"J Learning and Transferring mid-level image representation using convolutional neural networks"** by Oquab, M.Bottou, Laptev and I.Sivic in CVPR(2014)

[5] **"A Facial Expression Recognition System Using Convolutional Networks"** by Lopes, Andre Teixeira, Edilson de Aguiar and Thiago Oliveira-Santos in 28th SIBGRAPI Conference on Graphics, Patterns and Images, IEEE(2015)

[6] **"Facial Expression Recognition Based on Tensorflow Platform"**, ITM Web Conferences, 2017