NeuroFace: Facial Keypoints Detection using Deep Learning

Rajiv Nayan Choubey

Abstract—In recent years, Facial keypoints/landmarks detection has become a very crucial topic because of its vast application. Its application includes face recognition, emotion detection, augmented reality, etc. The prime objective of Facial Keypoints Detection is to predict the coordinates of crucial landmarks on the face. Due to various types of faces and expressions, it is quite challenging to extract the facial features from faces, which further are used to predict the key points. With the advancements in Deep Learning and Computer Vision, it has become easy to extract the facial features from images of the face. With lots of data, the precision of the predicted coordinates can be improved. In this paper, the author has proposed NeuroFace, a new Deep Convolutional Neural Network Model, and Graphical Interface for the Facial Keypoints detection task. It takes an image of the face as input and predicts 15 key points of the face. Further, the Graphical Interface built on top of the model captures video from the webcam and plots the key points on all the faces present in the video. The proposed model is light-weighted that it gives a real-time prediction at average 18.7 Frames Per Second. The model has been trained and tested on Kaggle's dataset. Using state-of-the-art image augmentation technique for spatial points, it achieved a Root Mean Square Error of 3.35608.

Index Terms—Facial Keypoint Detection, Deep Learning, Kaggle, Deep Convolutional Neural Network.

1 Introduction

The Facial Keypoints/Landmarks are important points on the face, such as coordinates of several locations of eyes, nose, lips, eyebrows, chin, etc. These key points define most of the important features of the face of a person. One can tell a lot about the person just by looking at his/her facial keypoints. Thus, in recent years, many kinds of research have been done on automated Facial Keypoints Detection because of their wide applications. Its applications include Facial Expression Detections, Emotion Detections, Biometrics Analysis, Age & Gender prediction, real-time drowsiness detection, Augmented Reality on Face (like used in Snapchat), and the list goes on. Since the facial keypoints a crucial role in many applications, its precise and fast detection is necessary. Previously, extensive work has been done for the same, which includes advanced Machine Learning (ML) and Deep Learning (DL) methods. Some of the ML methods include probabilistic graphic models to map the relationship between pixels and key points and Markov Random fields [1] to discover constellations that key points could form. But Facial features have lots of complexity and variety in it like different head pose, orientation, skin complexion, and structures, and it gets difficult for basic ML algorithms to learn complex features of the face. With the evolution of Computer Vision along with DL, extracting crucial and complex features from face has become easy. DL needs data to work on, and in today's world, there are data in abundance. Researchers have used several variations of Deep Convolutional Neural Networks [2] to train the model with lots of publicly available data to learn complex facial features and predict facial key points precisely. These researches saw quite a big success in this field. Inspired by these researches, the author proposes a new DCNN with only a few layers for the Facial Keypoints Detection task. The DCNN takes an image of a single face as input and extracts features from it, which is further used to predict the coordinates of 15 key points of the face. To complete the pipeline, the author has also built a Graphical Interface which captures video stream from a webcam and detects all the faces in it using Haar-Cascading [3]. The cropped image of all the faces in the streams is passed to the proposed DCNN one-by-one, which predicts key points in the corresponding faces. The interface then shows the video stream with key points plotted. The author named the complete proposed method as "NeuroFace" [1]. NeuroFace can be tweaked for many applications like real-time face filters and emotion detection. Fig. 1 shows 15 Facial Keypoints detected by NeuroFace on images of face.

(a) (b)

Fig. 1: 15 Predicted Facial Keypoints (shown as dots)

• *R.N. Choubey is with the Department of Computer Science and Engineering, IIIT Naya Raipur, Chhattisgarh, India.*
E-mail: rajiv17100@iiitnr.edu.in

1. Implemented at https://github.com/rajivnayanc/Deep_Learning_Projects/tree/master/Facial_Keypoints_Detection

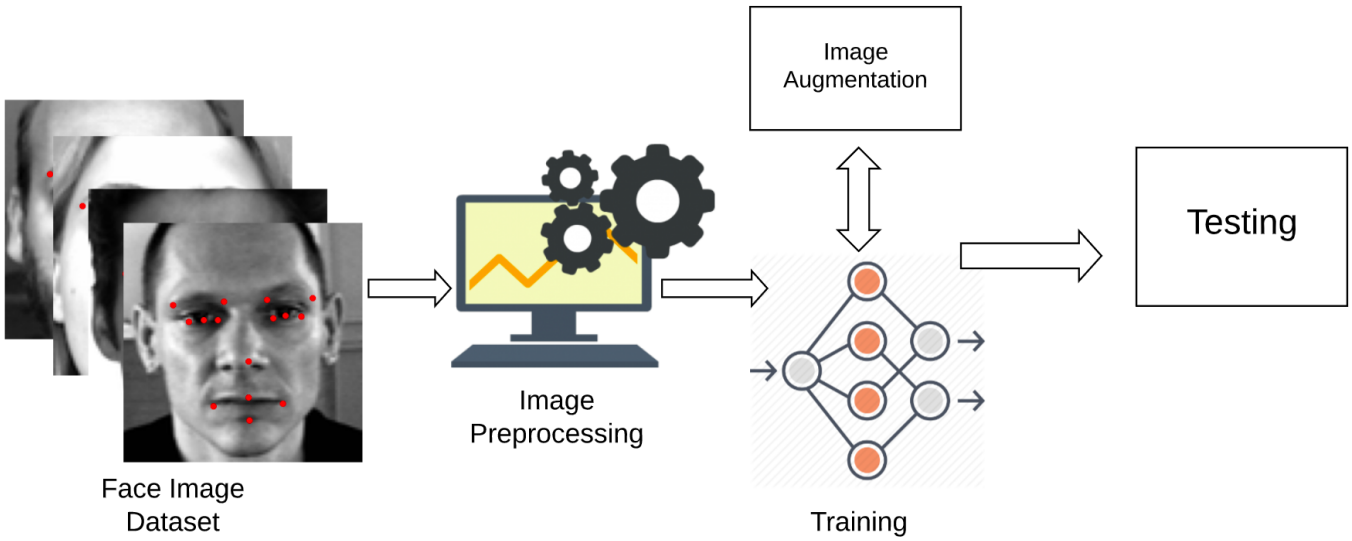


Fig. 2: Flow Chart of training the proposed model

NeuroFace has been built with not-so-deep neural networks for real-time use with little trade-off with precision of the prediction. Thus, the major contributions of the paper are as follows:

- A new DCNN is proposed to have fewer layers for extracting features from an image of a face. This feature is further used to detect 15 key points on face precisely.
- Having less number of layers, the proposed DCNN can give results in less time, which can be used for real-time applications.
- A complete DL pipeline, NeuroFace, is built with Graphical Interface to detect key points from all the faces from a video stream of webcam in real-time.
- Finally, the proposed DCNN has been tested upon various Facial Keypoints dataset and signifies quite a good root mean square score.

This paper is henceforth structured as follows. Section 2 describes some of the previous related works on Facial Keypoints detection. The NeuroFace pipeline is proposed in Section 3. Details of the experiments performed on the proposed model are given in Section 4. Section 5 contains the analysis of observation obtained while experimenting. Finally, the paper concludes in Section 6.

2 Related Works

Facial keypoint detection has been a hot topic for the past few years and lots of studies have been done on the same. Some of those works presented by different researchers include traditional machine learning methods and deep learning methods to extract facial features and discover the relationship between them and the coordinates of key points.

Traditional Machine Learning methods include the Gabor Wavelet feature extraction method to detect up to 20 facial keypoints [4], [5]. It uses a log Gabor Response of a facial keypoints, and the authors demonstrated key points locations on test images of the face. [6] uses feature extraction by surface fitting to surroundings of key points and projecting on the grid. PCA is applied later for dimensionality reduction and later projected and matching from gallery face. Markov Random fields have also been used to discover constellation formations of facial keypoints [1]. These models perform better on aligned faces but not on rotated or sheared faces. In [7], Local evidence aggregation has been used to detect facial keypoints by estimating randomly selected coordinates into single robust prediction.

With the advancement in Deep Learning, recent works are based on it. Deep Neural networks have a high capacity to learn complex features of a face. [8], [9], [10], [11], [12] have used different well structured Deep Convolutional Neural Networks to extract features from images of the face and later used various regressor like Random Forest Regressor, Support Vector Machines, Regression Tree, Neural Networks, etc.

Typically, before detecting facial key points, it is necessary to locate faces in an image. Several techniques have been applied for the same, but some of the widely used methods include Viola-Jones Algorithm [13] based Haar Cascading to detect faces based on skin tone and structure [3]. Nowadays, Deep Neural Networks have achieved far better results in detecting the location of faces accurately with more head rotations and different physical structures and illuminance. FaceNet is one of the state-of-the-art face detection models.

Our proposed DCNN has few layers on the contrary to the above methods, which result in faster training and real-time inference. Taking into account the real-time use of Facial Keypoints Detection, NeuroFace has been proposed by the author.

3 Proposed Solutions and Methodology

In this section, the author has proposed and explained a new Deep Convolutional Neural Network Architecture for Facial Keypoints Detection Task. The author has also explained the

working of Graphical Interface as a complete pipeline for the proposed solution, NeuroFace. Major steps for the proposed solution to train the proposed architecture are shown in Fig. 2.

Firstly, images of the face along with annotated keypoints are collected as the dataset.In data preprocessing phase, the dataset is then divided into X_{train}, X_{val} and X_{test} and ground truth vectors containing all the coordinates of key points as Y_{train}, Y_{val} and Y_{test}. All the data points having incomplete ground truth vectors are removed for consistency. If we fill the null values with some random numbers or mean, it won't represent the actual location perfectly due to a variety of physical structures and orientations of faces in training images. Next, we need to reshape the images into the desired dimension which proposed DCNN accepts. Afterwards, the images need to be converted to gray-scale and then normalized with mean (μ) = 0.5 and standard deviation(σ) = 0.5 for better and stable training of the DCNN. Let the input image be I. After I is converted to grayscale, $I_{ij} \in [0, 255]$. The image can be normalized using equation 1.

$$I_{ij}^{'} = \frac{I_{ij} - \mu}{\sigma} \tag{1}$$

Where, $I_{ij}^{'}$ represents new normalized image.

Normalizing images with $\mu = 0.5$ and $\sigma = 0.5$ brings every pixel in range of -1 and 1. Thus $I_{ij}^{'} \in [-1, 1]$ or $I^{'} \in \mathcal{N}(0.5, 0.5)$. To recover back the original image from the normalized image, equation 2.

$$I_{ij} = I_{ij}^{'} * \sigma + \mu \tag{2}$$

Here,$I_{ij}^{'}$ and I_{ij} represents normalized and recovered images respectively.

To extract the features from the preprocessed face image, the proposed DCNN architecture consists of 5 layers of convolutional layers followed by 1 layer of fully connected neural network for predicting the coordinates of facial keypoints from the extracted images. An overview of architecture of the proposed model is as follows: $INPUT- > [CONV- > RELU- > BATCHNORM- > POOL] * 4- > CONV- > RELU- > BATCHNORM- > FLATTEN- > FCNN$. The model takes a normalized single channel or gray-scale image as an input of dimension accepted by the DCNN. It follows a series of Convolutional, ReLU, Batch Normalization and Max Pooling layers to extract features from the face image. The extracted features are then flattened and passed to the fully connected neural network for prediction of Facial Keypoints. This few layers deep network is able to map complex facial features into the location of facial keypoints efficiently. The model takes a normalized single channel or gray-scale image as an input of dimension accepted by the DCNN. It follows a series of Convolutional, ReLU, Batch Normalization and Max Pooling layers to extract features from the face image. The extracted features are then flattened and passed to the fully connected neural network for prediction of Facial Keypoints. This few layers deep network is able to map complex facial features into the location of facial key points efficiently.

Deep Learning models require lots of data but due to less number of training image, Image Augmentation plays a crucial role. In order to bring vividity in training images, they are rotated, flipped, sheared and noise is added. With these augmentations, even with less training data, more generality is introduced into the trained model. But there is a difficulty in augmenting the images. The ground truth facial keypoints have spatial importance. So, for example, if images are transformed, then the ground truth coordinates should also be transformed along with it. These can be achieved by plotting the key points on an array of dimension similar to the face image and then applying the same transformation on it too. This makes image and corresponding ground truth to have the same transformation and, thus, the spatial location of keypoints is preserved.

Now, in order to train the proposed model or learn the mapping between an input face image and corresponding key points, network parameters (Θ) need to be optimized. The mapping is represented as $F(I, \Theta) = \hat{Y}$. This is done by minimizing the value of loss function which in this case is Mean Square Error(MSE) loss represented in equation 3. MSE Loss is used when the task is regression. In this task, real numbers(coordinates) is predicted by the models.

$$L_{MSE}(\Theta) = \frac{\sum_1^N (Y - \hat{Y})^2}{N} \tag{3}$$

Here, N represents Number of Training Samples, Y represents true coordinates of facial key points and $\hat{Y}$ represents predicted coordinates of facial key points.

The loss is optimized using the Adam Optimizer with normal backpropagation. Mathematically, the weights are updated according to the equation 4.

$$\begin{aligned} initially, m_0 = 0, v_0 = 0 \\ m_{t+1} \leftarrow \beta_1 m_t + (1 - \beta_2) \nabla_{\theta_j} L_{MSE}(\Theta) \\ v_{t+1} \leftarrow \beta_2 v_t + (1 - \beta_2) \nabla_{\theta_j} L_{MSE}(\Theta)^2 \\ \theta_j \leftarrow \theta_j - \frac{\alpha}{\sqrt{v_{t+1}} + 10^{-5}} m_{t+1} \end{aligned} \tag{4}$$

Here, m and v are first and second moment vectors initialized with 0. β_1 and β_2 are exponential decay factors for first and second moments. $\nabla_{\theta_j} L_{MSE}(\Theta)$ represents gradient of Loss function with respect to weight θ_j. Optimizing the loss function for several epochs will train the model to learn the mapping function $F(i, \Theta)$. Augmented Images can be served while training by freshly sampling new face images and applying augmentations on it. These produce more variety of images rather than augmenting data beforehand only. After training, the model is evaluated on Validation data (X_{val}) and Test data (X_{test}).

Finally, after complete training of the model, the complete pipeline, NeuroFace, to capture video stream from the webcam and displaying the detected Facial Keypoints on every face present in the video is built. Detailed Flow chart of working of NeuroFace is shown in Fig. 3.

Here, frames are extracted from webcam video streams and converted to gray-scale. On the converted gray-scale image Viola-Jones face detection algorithm is applied to detect all the faces and get the coordinates of the bounding box of the faces. Viola-Jones Algorithm applies several filters as Haar Cascades to detect the presence of the face based

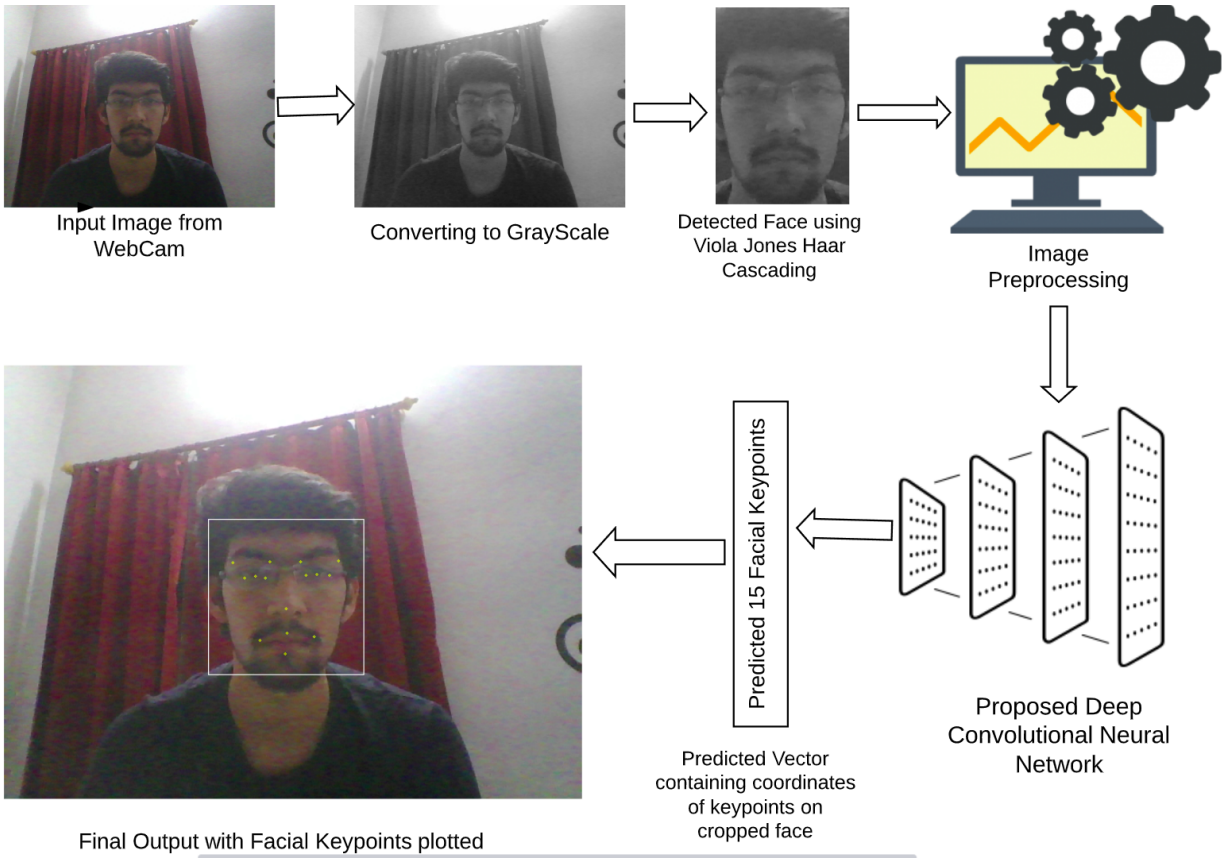


Fig. 3: Flow Chart of complete working of NeuroFace (Predicted Keypoints shown as dots on face)

on skin colour change between various features elements of the face and skin structures. This algorithm does not work perfectly on the rotated face above 20° but produces results faster. Moving further, the face regions are cropped and resized into the dimension accepted by the proposed DCNN and then it is normalized. All the normalized face images are passed into DCNN and it produces vectors containing coordinates of key points of all the faces. The coordinates are then mapped into the original image containing all faces and plotted over corresponding faces. Having fewer layers and using fast face detection algorithm, it produces real-time output for the Facial Keypoints Detection Task.

4 Experiments

This section consists of a description of the dataset used, hardware and software used, architectural detail of proposed DCNN model and implementation details of NeuroFace.

4.1 Dataset used

The data provided by Kaggle's Facial Keypoints Detection Challenge [14] competition is used and the final evaluation of the trained model is done through Kaggle Platform. The dataset consists of 7,049 training images along with (x,y) coordinates of 15 Facial Keypoints (refer Table 1) and 1,783 test images without annotation. Each data point of training dataset contains a 96x96 dimension grayscale face image and an output vector of length 30 representing (x,y) coordinates of corresponding 15 key points. Details of Facial Keypoints annotated in the dataset is stated in Table 1. Fig. 4 shows samples of Annotated Face Images from X_{train}

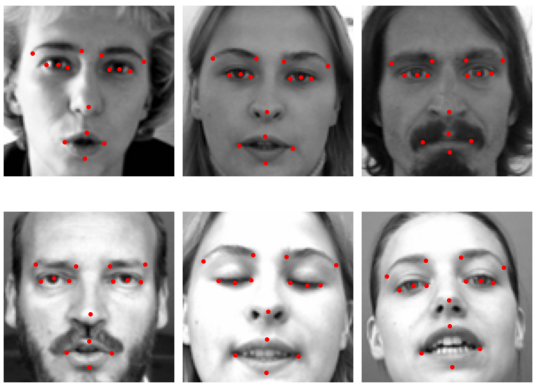


Fig. 4: Samples of Annotated Face Images from Training Set

After removing data points containing null values, the training data is divided into the training split (80 %)(X_{train}, Y_{train}) to train our data while applying image augmentation and validation split (20%) (X_{val}, Y_{val}) to validate our proposed network. The test data (X_{test}) is used for the final evaluation when the submission is submitted on Kaggle's Facial Keypoint Detection Challenge.

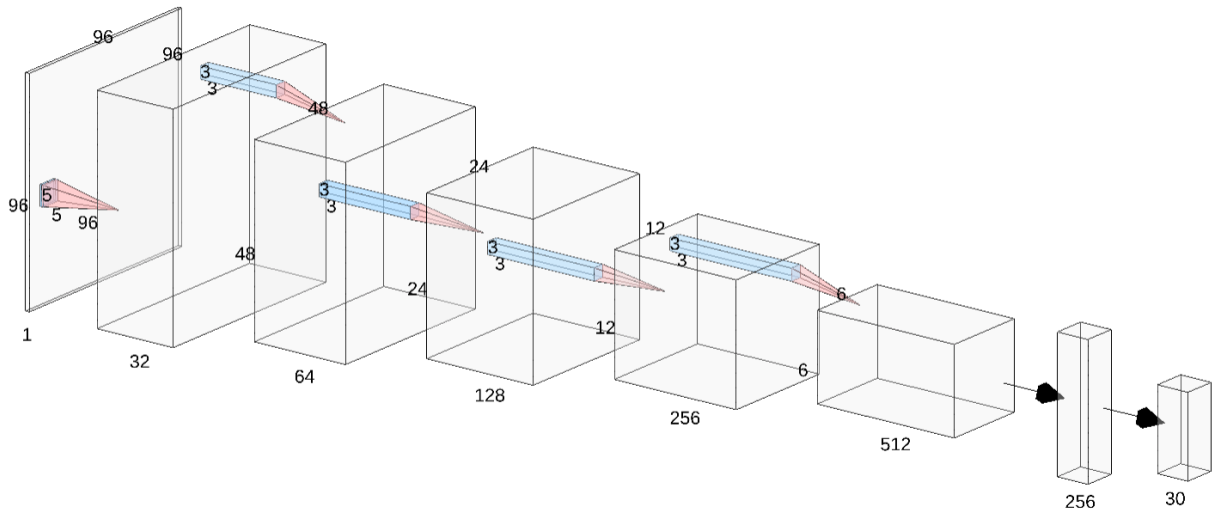


Fig. 5: Proposed Deep Convolutional Neural Network Architecture

TABLE 1: 15 Facial Keypoints Details

Left Eye Center	Right Eye Center
Left Eye Inner Corner	Left Eye Outer Corner
Right Eye Inner Corner	Right Eye Outer Corner
Left Eyebrow Inner End	Left Eyebrow Outer End
Right Eyebrow Inner End	Right Eyebrow Outer End
Nose tip	Mouth Left Corner
Mouth Right Corner	Mouth Center top Lip
Mouth Center Bottom Lip	

TABLE 2: Detailed Architectural Design of Proposed Model

S.No	Operations	Activation Function	Output Shape
1	Input Grayscale Image	-	1x96x96
2	Conv2d 5x5	ReLU	32x96x96
3	BatchNorm	-	32x96x96
4	MaxPool 2x2	-	32x48x48
5	Conv2d 3x3	ReLU	64x48x48
6	BatchNorm	-	64x48x48
7	MaxPool 2x2	-	64x24x24
8	Conv2d 3x3	ReLU	128x24x24
9	BatchNorm	-	128x24x24
10	MaxPool 2x2	-	128x12x12
11	Conv2d 3x3	ReLU	256x12x12
12	BatchNorm	-	256x12x12
13	MaxPool 2x2	-	256x6x6
14	Conv2d 3x3	ReLU	512x6x6
15	BatchNorm	-	512x6x6
16	Flattening	-	1x18432
17	Linear	LeakyReLU	1x256
18	Dropout (p=0.5)	-	1x256
19	Linear	-	1x30

4.2 Hardwares and Softwares Used

The experimental setup consists of Intel Core i7 7700 8 Core 2.83Hz CPU. Nvidia GTX 1080Ti 15GB Graphics Processing Unit is used for acceleration of training CNNs. The model has been implemented on Python 3.7 using PyTorch [15] 1.4.0 as Deep Learning Framework. Sklearn, Numpy, Matplotlib are also used for basic machine learning tasks, mathematical calculations and visualization. To implement the end-to-end pipeline, NeuroFace, OpenCV 4.3.0 is used to capture the webcam video stream and do most of the image processing task like converting to grayscale, cropping facial region etc.

4.3 Implementation Details

The training dataset is preprocessed before training. Initially, 7049 images are present in the training data. After removing data points with null values, 2140 images are available for training. It is split into X_{train} and Y_{train}. X_{train} contains 1700 face images and X_{val} contains 440 face images. Each image is resized into the dimension of 96x96 as a grayscale image. Afterwards, each face images are normalized with mean(μ)= 0.5 and standard deviation(σ) = 0.5.

The proposed architecture of Deep Convolutional Neural Network is shown in Fig. 5. The model takes a grayscale normalized image with one channel and after series of operations at different hidden layers, it produces a 30 length vector containing the (x,y) coordinates of 15 facial key points(Table 1). Complete detailed architecture of the model is described in Table 2. During training, the augmented images were served using 20 threads to speed up the process. The Adam Optimizer used for reducing the MSE Loss function had β_1 and β_2 set as 0.9 and 0.999 respectively. The model was trained for 60 epochs with initial learning rate of 10^{-4}. The learning rate was reduced by a factor of 0.5 after every 15 epochs.

The trained model is put into use in NeuroFace. Firstly, for each grayscaled frame of the webcam video stream, the location of faces are calculated using Viola-Jones Haar Cascades. Using the face coordinates, the facial regions are cropped and scaled to 96x96 dimension. All the cropped face images are normalized and passed to trained DCNN model one by one. For each facial region, coordinates of facial keypoints are predicted and mapped to the original webcam image. An output window of the NeuroFace is shown in Fig. 6.

Fig. 6: NeuroFace Sample Window processing Webcam Stream

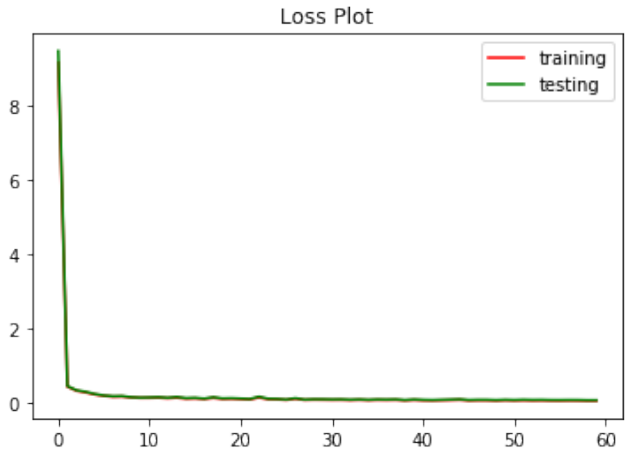


Fig. 7: Loss plot of training the model for every epoch

Fig. 8: Predicted Outputs from Validation Set with ground truth available

5 Results and Analysis

5.1 Performance Parameters

- **Root Mean Square Error (RMSE)**: RMSE is used to find the average error between predicted coordinates and true coordinates of the Facial Keypoints. RMSE can be calculated using equation 5.

$$RMSE = \sqrt{\frac{\sum_1^N (Y - \hat{Y})^2}{N}} \tag{5}$$

  Here, Y and $\hat{Y}$ represent true and predicted facial key points coordinates respectively.N represents total number of training points.
- **Training Time per epoch**: Training time per epoch represents the time taken by model to feed forward and backpropagate error for the complete dataset exactly once.
- **Frames Per Second (FPS)**: For the real-time Facial Keypoint detections, FPS plays an important role. It shows how many Frames are processed by the model per second.

5.2 Analysis

The proposed architecture was trained with and without Image Augmentation. Table 3 shows the time taken for each epoch while training the model. It can be observed from Table 3 that Image Augmentation mode takes more time to train. Both the models were evaluated using X_{test} on Kaggle's Platform. Table 4 shows the RMSE score of two modes of training. On comparing Table 3 and 4, it can be seen that, though it took more time for the model to train with Image Augmentation but it provides better results with less RMSE Score.

TABLE 3: Training time per epoch for different modes

Mode	Training Time per epoch
Without Image Augmentation	2min 3secs
With Image Augmentation	12min 34secs

TABLE 4: RMSE of two modes of training

Mode	RMSE Score
Without Image Augmentation	4.29480
With Image Augmentation	3.35608

Fig. 7 represents MSE Loss Plot of training the model for every epoch for both training and validation data. It can be observed from the Fig. 7 that the model learning curve converges quickly. Also, the curve for training and validation data overlaps proving there's no over-fitting. The final RMSE Score on Kaggle is 3.35608. The trained model is able to predict the coordinates of Facial Keypoints precisely. Fig 8 and Fig. 9 shows some predicted outputs from Training Data and Test Data respectively. Fig. 8 and Fig. 9 shows predicted and actual keypoints as red and yellow dots respectively.

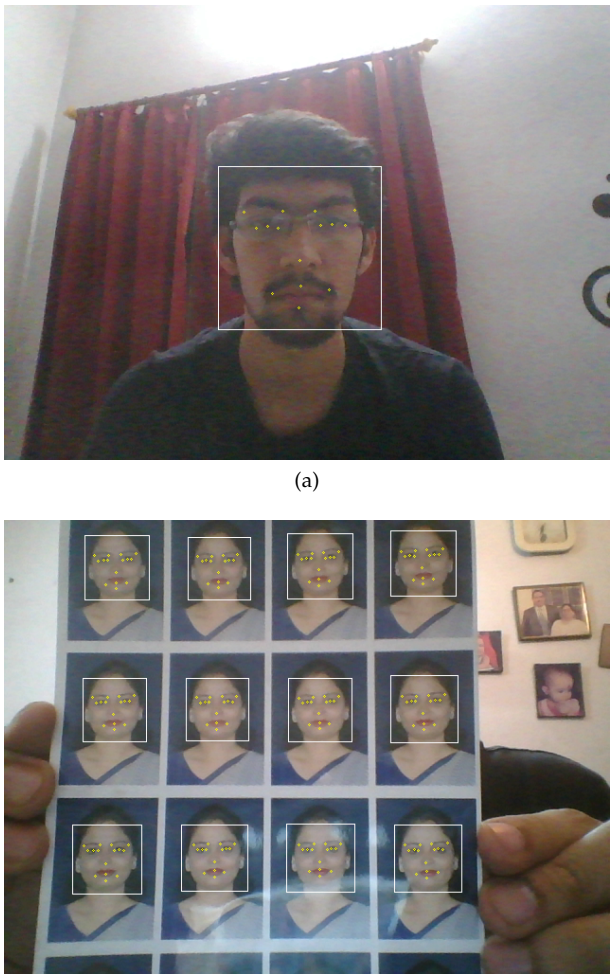
Finally, the complete proposed Pipeline, NeuroFace, was built and evaluated with FPS and number of faces present in video stream. Table 5 shows observed data for the same. It can be seen that the NeuroFace performs quite fast on enough number of faces with 18.87 Average FPS.

Fig. 9: Predicted Outputs from Test Set with no ground truth available

TABLE 5: Number of Faces vs FPS of NeuroFace

#Faces in Photos	FPS
1-4	20-25
4-10	14-20
10-15	10-14
>15	8

Some of the sample outputs of NeuroFace is shown in Fig.10 (Predicted Keypoints are shown as yellow dots on the face(s)). From Fig. 10b, it can be observed that MultiFace Keypoints Detection is performed perfectly by the NeuroFace



(a)

(b)

Fig. 10: Outputs of NeuroFace

6 Conclusion

The paper presented a new Deep Learning pipeline, NeuroFace, for the task of Facial Keypoints detection. In NeuroFace, author has built a new DCNN architecture with few layers which is able to detect (x,y) coordinates of 15 facial keypoints at high speed. The new DCNN model achieved RMSE score of 3.35608 on Kaggle's Facial Keypoints Detection Challenge. Author has also built a Graphical Interface to work with real-time video stream from webcam and detect facial keypoints for all the faces present in it. The NeuroFace is able to process the image at average 18.87 FPS which is quite a good result for real-time applications.

Acknowledgments

The author would like to thank Dr. Muneendra Ojha for his constant support in making of this project. The author would also like to extend a big thanks to Google Colaboratory for giving free Graphics Processing Units for use.

References

[1] M. Valstar, B. Martinez, X. Binefa, and M. Pantic, "Facial point detection using boosted regression and graph models," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 2729–2736.

[2] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8614–8618.

[3] R. Padilla, C. Costa Filho, and M. Costa, "Evaluation of haar cascade classifiers designed for face detection," *World Academy of Science, Engineering and Technology*, vol. 64, pp. 362–365, 2012.

[4] D. Vukadinovic and M. Pantic, "Fully automatic facial feature point detection using gabor feature based boosted classifiers," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 2. IEEE, 2005, pp. 1692–1698.

[5] E.-J. Holden and R. Owens, "Automatic facial point detection," in *Proc. Asian Conf. Computer Vision*, vol. 2, no. 731-736, 2002, p. 2.

[6] A. S. Mian, M. Bennamoun, and R. Owens, "Keypoint detection and local feature matching for textured 3d face recognition," *International Journal of Computer Vision*, vol. 79, no. 1, pp. 1–12, 2008.

[7] B. Martinez, M. F. Valstar, X. Binefa, and M. Pantic, "Local evidence aggregation for regression-based facial point detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 5, pp. 1149–1163, 2012.

[8] N. Tripatarasit, "Facial keypoints detection with pytorch," https://medium.com/diving-in-deep/facial-keypoints-detection-with-pytorch-86bac79141e4, May 2019.

[9] N. Agarwal, A. Krohn-Grimberghe, and R. Vyas, "Facial key points detection using deep convolutional neural network-naimishnet," *arXiv preprint arXiv:1710.00977*, 2017.

[10] S. Zhang and C. Meng, "Facial keypoints detection using neural network," *Stanford Report*, p. 1, 2016.

[11] S. Shi, "Facial keypoints detection," *arXiv preprint arXiv:1710.05279*, 2017.

[12] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3476–3483.

[13] O. H. Jensen, "Implementing the viola-jones face detection algorithm," Master's thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2008.
[14] Kaggle, "Facial keypoints detection," https://www.kaggle.com/c/facial-keypoints-detection/data, May 2017.
[15] N. Ketkar, "Introduction to pytorch," in *Deep learning with python*. Springer, 2017, pp. 195–208.

Rajiv Nayan Choubey is currently pursuing his under-graduate degree in Computer Science and Engineering from Dr. SPM International Institute of Information Technology. He is currently in the third year of his graduate study and will be graduating in 2021. He has to his credit for publishing one research paper.