AAI 628-WS Final Report

Facial Keypoints Detection

Abstract :

Facial Key Points Detection is a significant and difficult subject in the disciplines of computer vision and machine learning. There is difficulty capturing keypoints on the face due to complicated influences from original photos, very varied facial traits from person to person, and little advice to suitable algorithms. It entails forecasting the co-ordinates of the Facial Keypoints, such as the nose tip, the center of the eyes, and so on, for a particular face. As deep learning advances, more and more ways based on it are offered for addressing linked difficulties, ushering in a slew of revolutionary improvements.

In this project, I proposed a simplified method based on Convolution Neural Network for solving the problem of detecting facial key points. We are given a list of 96×96-pixel 8-bit gray level images with their corresponding (x, y) coordinates of 15 facial keypoints. Initially I located the keypoints in each image using Deep Convolutional Neural Network algorithm. The algorithm trains the facial keypoints data and assess its performance on the test set to evaluate predictions.

Introduction :

A. Problem Statement:

The purpose of this research is to develop an algorithm that can predict the positions of facial keypoints on face photos with high accuracy. When given a raw facial image, my goal is to find 15 sets of facial keypoints such as the eyes, brows, nose, and mouth. I used deep structures for facial key point identification, which can learn well from diverse faces and overcome the variance between faces of different people or conditions to a large extent. The Deep Convolutional Neural Network technique is employed in this research. We used Convolutional Neural Network to train our model using the given images and then tested our predictions on the other images. As a result, we can see the actual and predicted keypoints on face images.

B. Project Flow:

Understanding the dataset: In this dataset, we are provided with the training and testing data folders that contain face images with image data as row-ordered list of pixels and the coordinates for facial keypoints.

Data Pre-processing: This step involved importing the images and using their paths to access these images and apply different augmentation techniques. So, this data pre-processing step will include loading and reading the dataset and making it ready for training.

Training: For training, the algorithm used is Convolutional Neural Network (CNN). Other optimization algorithms like Adam or Stochastic Gradient Descent is be used to minimize the error associated with this data.
Testing: After training our model using CNN, we evaluate our predictions using the test dataset. This will let us know how our model will be training and decide on whether we should modify its structure or hyperparameters. Then we use this final trained model to predict facial keypoints on the images.

Implementation :

This project is done in various stages illustrated in the project flow. I have used a deep learning python library PyTorch to implement this project code. The Project was Implemented on Google Colab and its GPU aided in optimizing and accelerating our deep learning frameworks.

Dataset Description :

I have used the dataset from Kaggle Competition – Facial Key Points Detection. There are 15 Facial Keypoints per face image like 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 and mouth center bottom lip.

The data set consists of a list of 7049 two-dimensional 8- bit graylevel training images with their corresponding (x, y) coordinates of the 15 facial keypoints. Each input image is represented by a size of 96 × 96 pixel, with pixel values in range of [0, 255]. The given training set is a huge matrix of size 7049 × 31, where each row corresponds to one image, the first 30 target columns give the (x, y) values for each of the 15 facial keypoints, and each entry in the last column is a long list of 9216 numbers representing the pixel matrix of each image melted row by row.

In some examples, some of the target keypoint positions are missing (encoded as missing entries in the csv, i.e., with nothing between two commas).

The test dataset consists of 1783 images with no target information. The Kaggle submission file consists of 27124 Facial Keypoint co-ordinates, which are to be predicted. The Kaggle submissions are scored based on the Root Mean Squared Error (RMSE).

EDA and Feature Engineering :

Importing the data:

Data was imported through drive link to import css and folders from the data source on Kaggle.

Data Pre-processing:

Using Pandas, create data frames for csv files and extract features which can be used in the analysis.

Since data contains some missing values, this step involves calculating, visualizing and replacing those missing values for every feature.This can be understood better in the graph below(Fig 1).
I have also used heat maps by Seaborn for finding the correlation between features as shown (Fig 2).
Then we spilt the training into keypoints and images. Each row of keypoints data contains the (x, y) coordinates for 15 keypoints, while that of images data contains the row-ordered list of pixels.

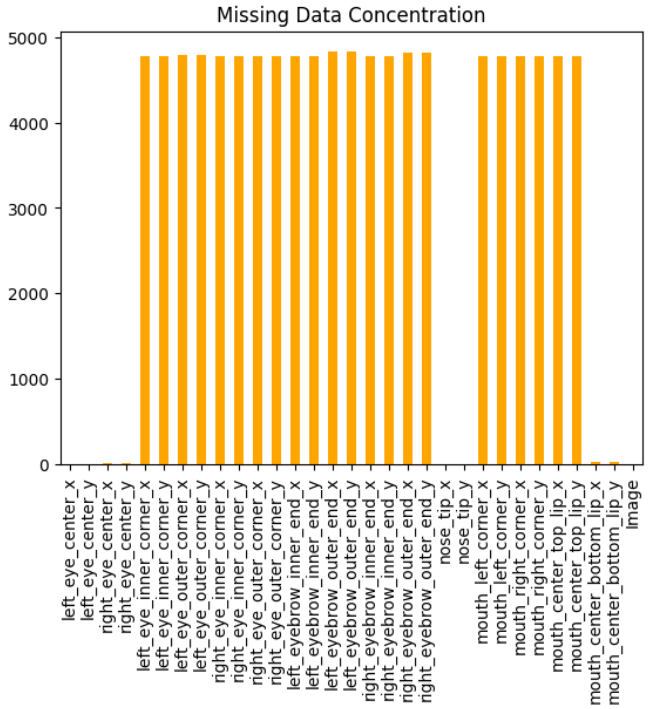


Fig 1 : Missing Data

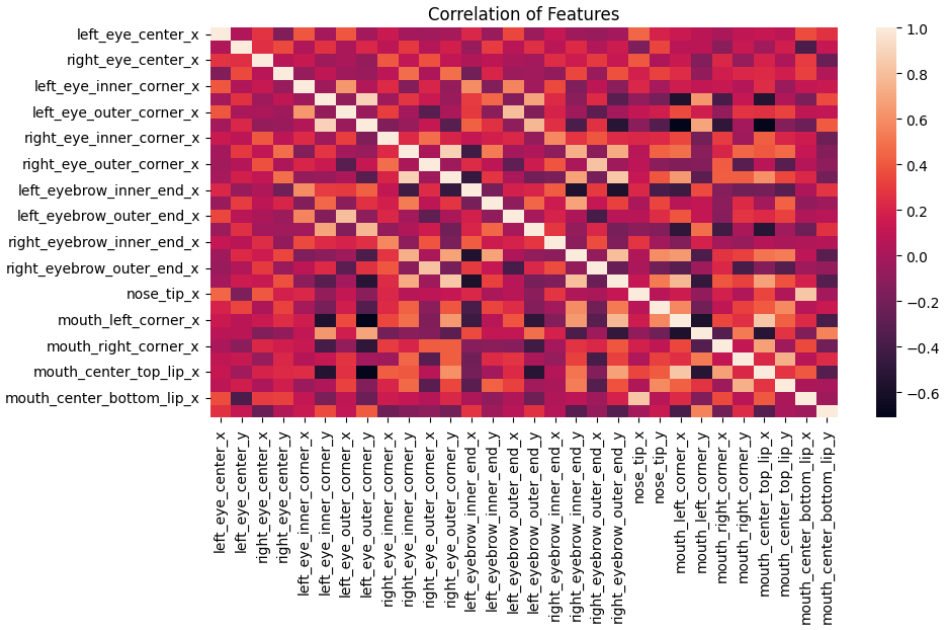
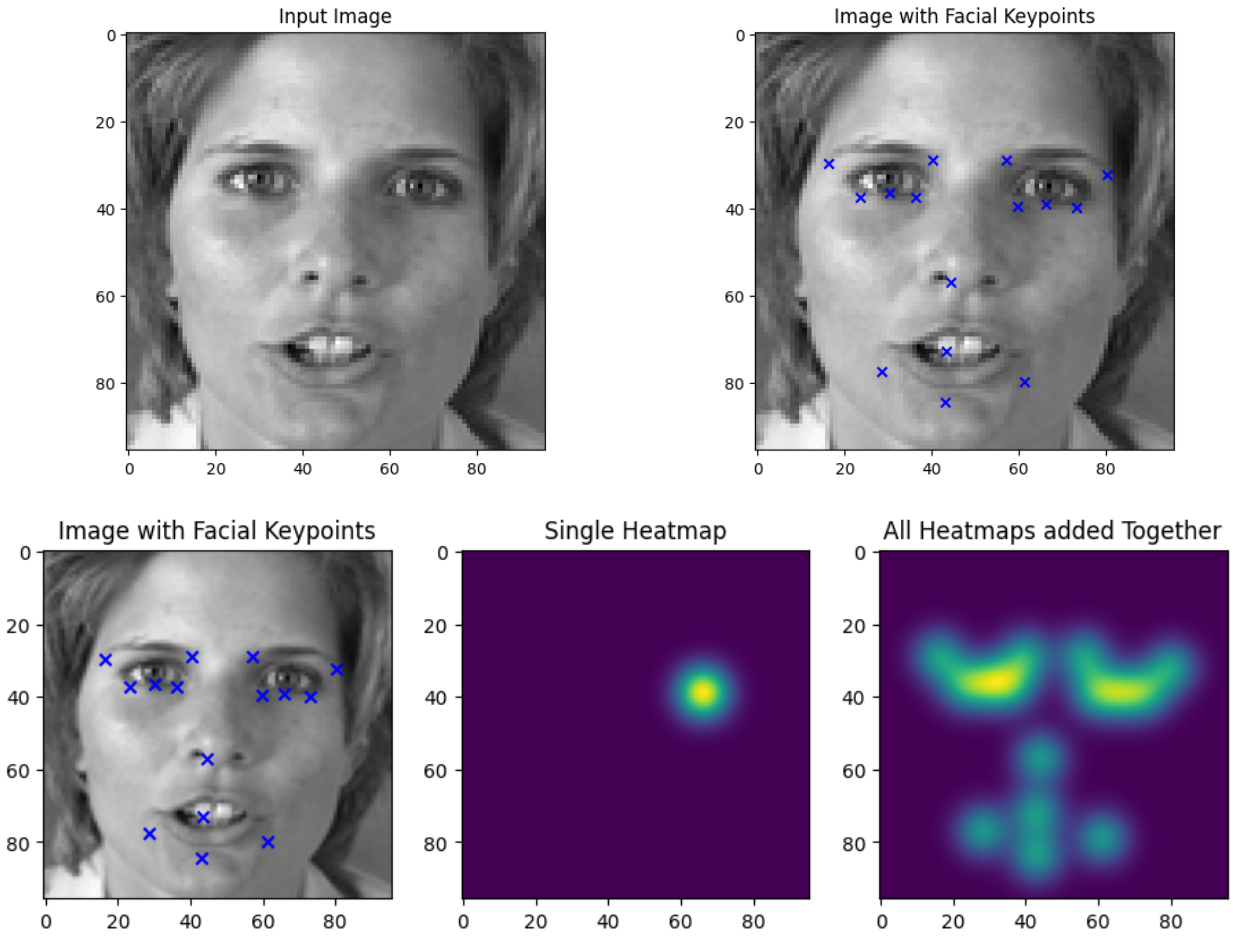


Fig 2 : Correlation of Features

Visualising the Input Image:



Algorithms :

Convolutional Neural Network - In CNN, I have applied four convolutional layers. Each convolutional layer is defined using the Conv2d nn.Module method which creates a set of convolutional filters. The first argument for this method is the number of input channels, the second argument is the number of output channels, and the third argument is the kernel size. In this case, we want 5x5 sized convolutional filters, so used a kernel size of 5. After the convolutional layer, apply a Rectified Linear Unit (ReLU) activation function to introduce non-linearity .Then I added a max pooling operation to reduce the spatial dimensionality of the output.

To limit covariate shift and stabilize the distribution of inputs to each layer, apply batch normalization which normalizes the activations of each layer by transforming the inputs to have zero mean and unit variance. Additionally, to prevent over-fitting, I used to add a dropout layer.

The second, third, and fourth convolutional layers are defined in the same way as the first layer, with different input and output channels as needed for each layer. I also applied ReLU activation, max pooling, batch normalization, and dropout in the same way as the first layer.

The final output layer is a fully connected layer that takes the flattened output from the previous layers as input and transforms it into the desired number of output classes for the network.

LeNet - LeNet is a neural network architecture that includes two convolutional layers. The first layer is created using the Conv2d nn.Module method with input channels, output channels, and kernel size as arguments. For the convolutional filters, we want a 5 x 5 size, so we specify a kernel size of 5.

The sigmoid activation function is applied to this layer, and the max pooling operation is added as the last element in the layer definition. The second convolutional layer is defined in a similar manner to the first one. The final output layer is a fully connected layer. I have trained my model using these deep network architectures.

Evaluation :

- To evaluate the performance of the models created, we will be applying them to training and testing data, with the objective of minimizing the error through iterative improvements.
- In order to achieve this, I have opted to utilize the Adam optimization algorithm, which is considered to be the most suitable for deep learning frameworks. To calculate the loss function, I have chosen the Mean Squared Error (MSE) approach.
- As part of data preparation process, I have utilized the train test split function to separate the target variable (Keypoints) from Images dataset. Additionally, created a DataLoader using the torch.utils library, which will be utilized to load batches of data for both training and validation purposes.
- By implementing this approach, we aim to efficiently manage large datasets and ensure that models are effectively trained and evaluated using the available data.

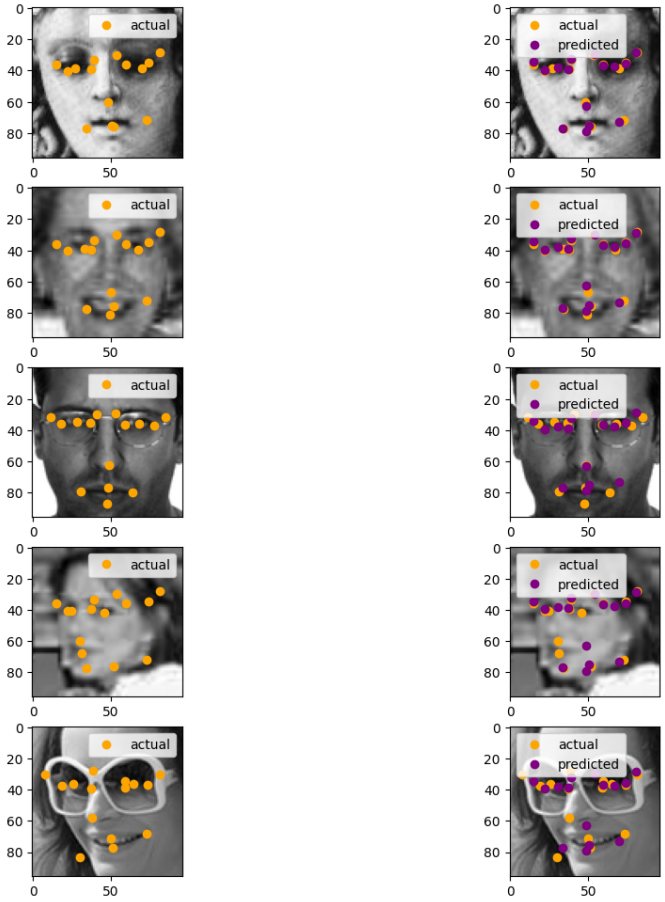
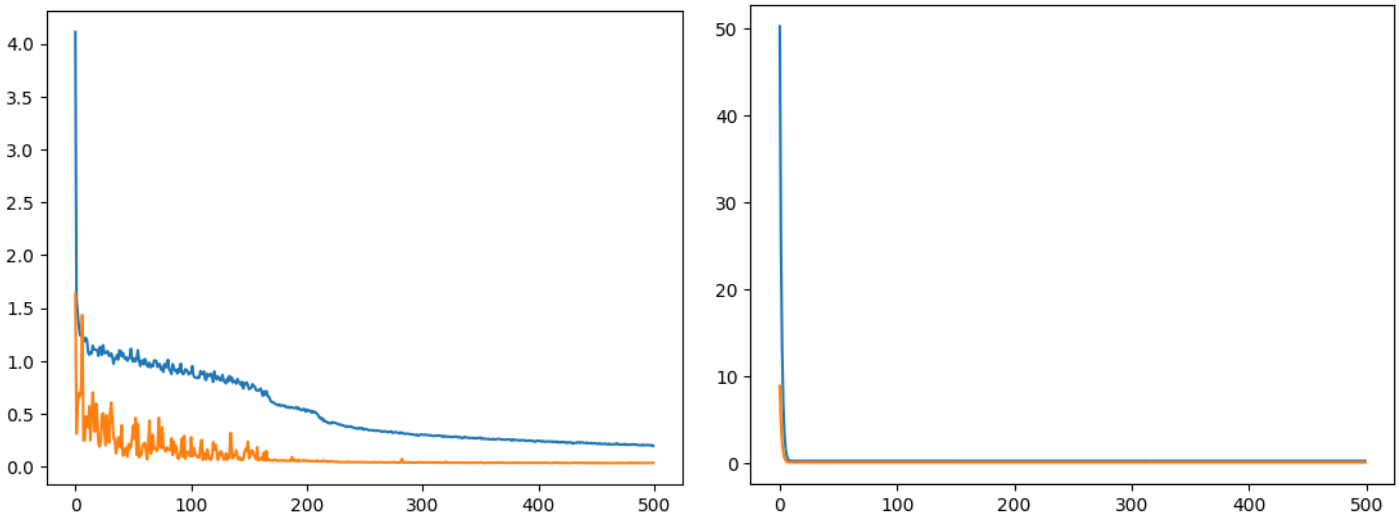


Fig 3. Actual and Predicted Keypoints

Results :

From our above trained model and predicting evaluations, we have plotted the graph of training and validation error for both CNN and LeNet as shown in the figure respectively.



For better understanding of the results, we have visualized our outputs.

As we can see in, we have plotted actual and predicted keypoints on the images.

Now we are going to use test dataset for predictions and visualizing test outputs as shown in figure below.

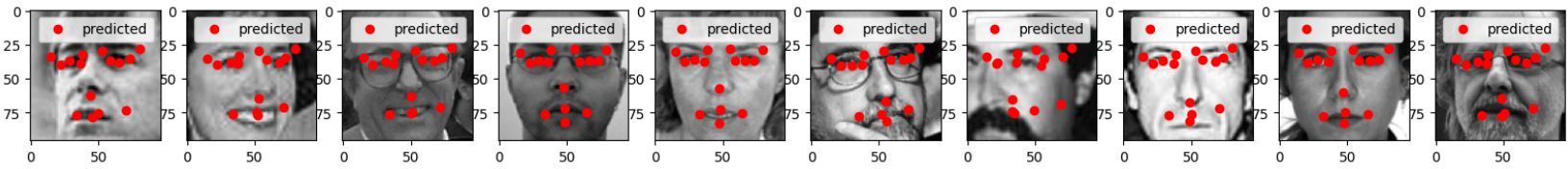


Fig 4. Predicted Keypoints for Test Dataset

Considering the results obtained for both networks we can observe how the LeNet5 network converges considerably faster in comparison with the random CNN of 4 layers. After the 15th epoch approximately the LeNet converges showing not improvement afterwards.

However in the case of the CNN, we can see how the model accuracy bounces around during the first 150 epochs, after- wards the model stabilizes the error and it slowly decreases it. Around the epoch 350 the validation error stops decreasing but the training error keeps decreasing, at this point we can stop the training other ways it may cause overfitting.

As conclusion, if we are looking for accuracy over speed then we will choose the CNN 4 layers model, but if what we are looking for is speed and we do not mind increasing the MSE then LeNet would be the choice. Note that the MSE reduction obtained by choosing the CNN 4 layers over the LeNet is from 0.0569 to 0.031

Conclusion :

In this project, I have focused on the task of detecting facial keypoints when given raw facial images. Specifically, for a given 96 x 96 image, we would predict 15 sets of (x,y) co-ordinates for facial keypoints.

A deep structure, Convolutional Neural Network is successfully applied for Facial Key Points Detection problem. We further explored another deep model, LeNet to fit the requirements for detecting facial keypoints.

Experiments which conducted on real-world Kaggle dataset have shown the effectiveness of deep structures. Comparing both the models, LeNet works better.

Future Scope :

Given enough computation resources and time, CNN can be further modified by experimenting with different initialization schemes, different activation functions, different number of filters and kernel size, different number of layers, switching from LeNet to VGGNet, introducing Residual Networks, etc.

Different image pre-processing approaches like histogram stretching, zero centering, etc. can be tried to check which approaches improve the model's accuracy.

Different resolution can greatly affect the results of the facial keypoints detection, thus what we can do is try to reduce the resolution of our given raw images to see the variance of the performance to further evaluate my model.

References :

1. D. Vukadinovic and M. Pantic. Fully automatic facial feature point detection using gabor feature based boosted classifiers. In Systems, Man and Cybernetics, 2005 IEEE International Conference on, volume 2, pages 1692–1698. IEEE, 2005.

2. E.-J. Holden and R. Owens. Automatic facial point detection. In Proc. Asian Conf. Computer Vision, volume 2, page 2, 2002.

3. A.S.Mian, M.Bennamoun and R.Owens Key point detection and local feature matching for textured 3d face recognition. International Journal of Computer Vision, 79(1):1–12, 2008.

4. M.Valstar, B.Martinez, X.Binefa and M.Pantic Facial point detection using boosted regression and graph models. In Computer Vision and Pat- tern Recognition (CVPR), 2010 IEEE Conference on, pages 2729–2736. IEEE, 2010.

5. B. Martinez, M. F. Valstar, X. Binefa, and M. Pantic. Local evidence aggregation for regression-based facial point detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(5):1149–1163, 2013.