Facial Keypoints Detection

By Luis, Siddharth and Rucha

Why this Project?

- Facial recognition is a very popular biometric technique these days.
- Our primary motivation was our interest in applying deep learning models to significant problems with relevant uses.
- We are excited about applying facial keypoints detection to a wide range of applications such as analysing facial expressions, security for phones, face filters, etc.

Introduction

- Detecting facial keypoints is a challenging problem due to varying facial features and image conditions.
- We have tried to solve this problem through our project.
- **Main goal :** To create an algorithm which can predict keypoint positions on face images.
- **Key Points :** We have a set of 15 facial keypoints of eyes, eyebrows, nose and mouth.

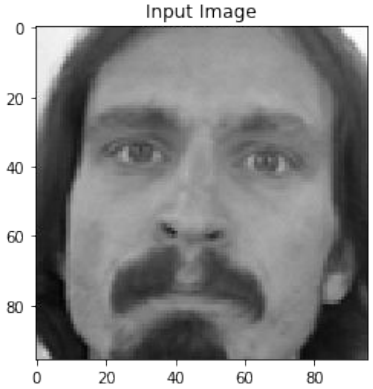
Introduction

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	

- More than one keypoint is assigned to each facial feature in order to be specific about the location and orientation of keypoints.
- **Algorithms :** To predict these 15 key points on face images, deep learning algorithms used are,
  - CNN with 3 and 4-Layers
  - LeNet-5
- Applied these models on real life images to predict facial keypoints.

Our Data Sources

- Kaggle.
- Images from the Internet
- Live Faces



Data Preprocessing

An overview :-

- Importing our data using the kaggle api.
- EDA and Feature Extraction
- Visualizing the input image and Facial Keypoints
- Heatmaps of these facial Keypoints

Importing our Data

- Downloading and Importing the json file containing the username and the key of our kaggle accounts
- Creating a client to host our kaggle API token
- Using the Kaggle API to import the data from the data sources on Kaggle
- Unpacking the data on Google Collab using Pandas

```
train_data.head(18).T.tail()
```

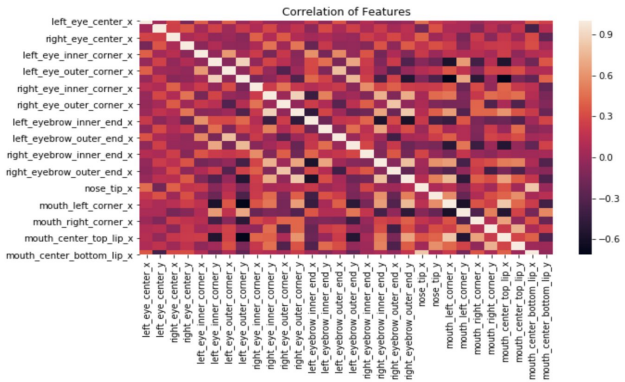
	0	1	2
mouth_center_top_lip_x	43.3126	46.6846	47.2749
mouth_center_top_lip_y	72.9355	70.2666	70.1918
mouth_center_bottom_lip_x	43.1307	45.4679	47.2749
mouth_center_bottom_lip_y	84.4858	85.4802	78.6594
Image	238 236 237 238 240 240 239 241 241 243 240 23...	219 215 204 196 204 211 212 200 180 168 178 19...	144 142 159 180 188 188 184 180 167 132 84 59 ...

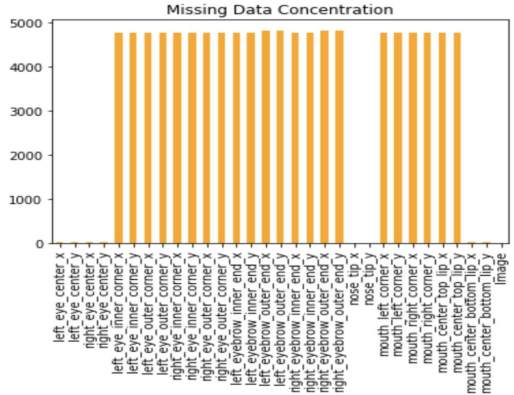
```
!kaggle competitions download -c facial-keypoints-detect
```

```
Warning: Looks like you're using an outdated API Version
Downloading training.zip to /content
 80% 48.0M/60.1M [00:01<00:00, 15.5MB/s]
100% 60.1M/60.1M [00:02<00:00, 31.2MB/s]
Downloading test.zip to /content
 56% 9.00M/16.0M [00:01<00:00, 8.09MB/s]
100% 16.0M/16.0M [00:01<00:00, 13.8MB/s]
Downloading SampleSubmission.csv to /content
  0% 0.00/201k [00:00<?, ?B/s]
100% 201k/201k [00:00<00:00, 168MB/s]
Downloading IdLookupTable.csv to /content
  0% 0.00/843k [00:00<?, ?B/s]
100% 843k/843k [00:00<00:00, 56.2MB/s]
```

EDA and Feature Extraction

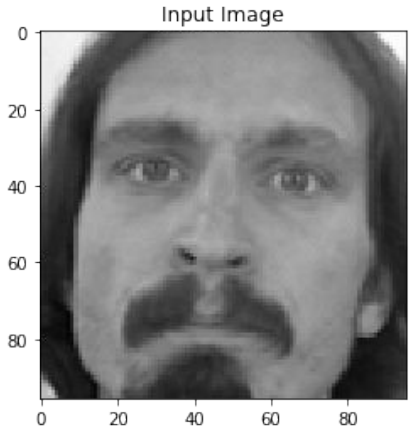
- Extracting the important features
- Calculating and filling the missing values
- Using heatmaps to find correlations between important features.

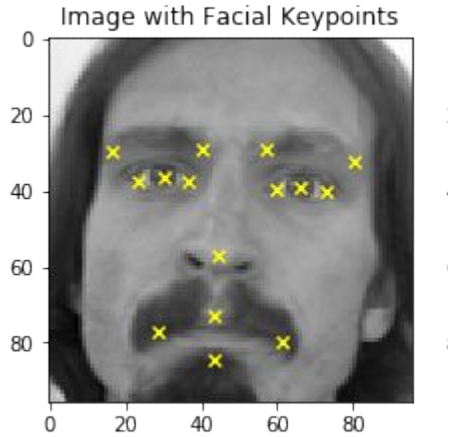




Visualizing the Input Image and its Key Points

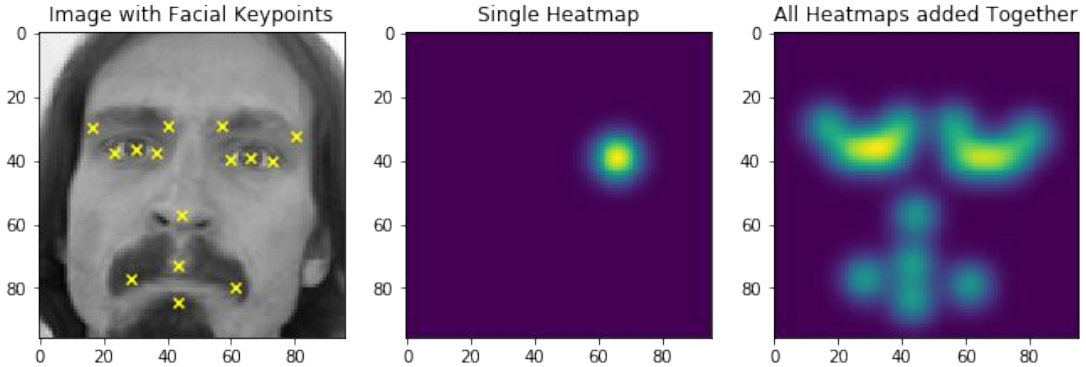
- Creating an numpy array of the image pixels
- Reshaping
- Using Matplotlib to plot these image pixels
- For Key Points dropping the ‘Image’ column and creating a scatter plot according to its coordinates





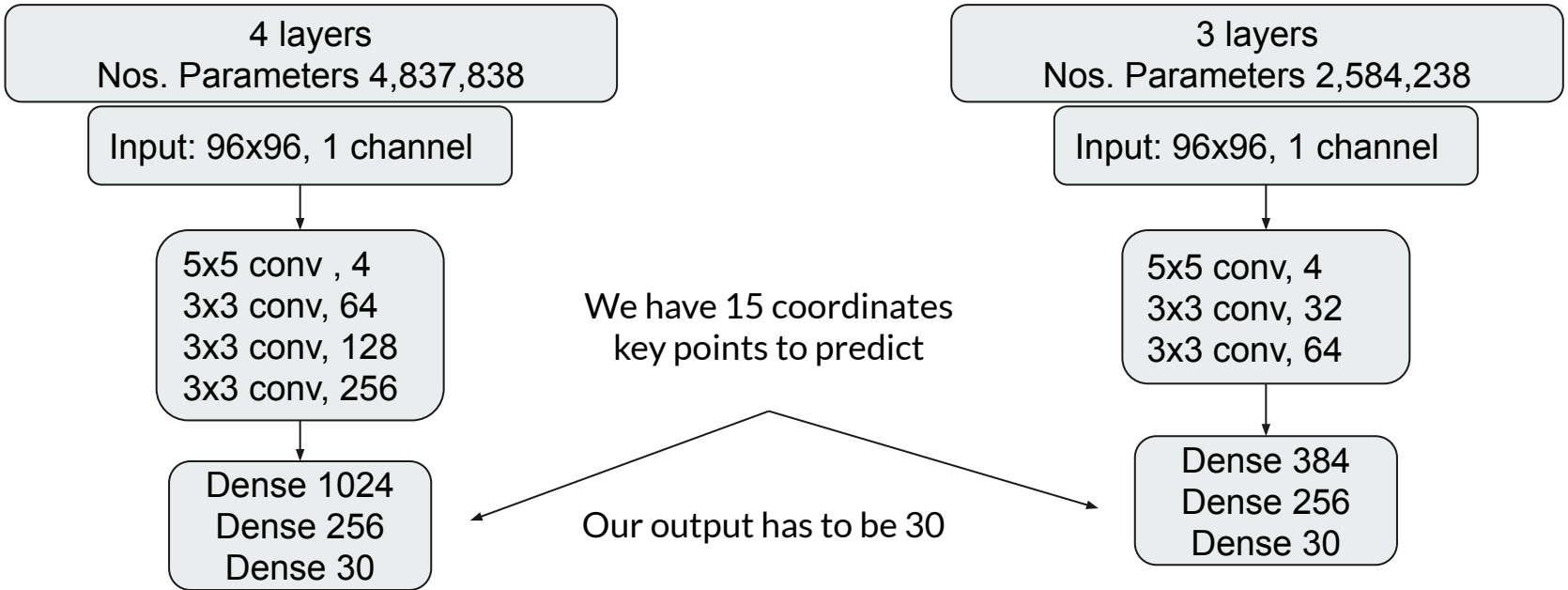
Heatmaps for these Keypoints

- Creating a Gaussian Function
- Using this gaussian function to generate a heat map of a single coordinate as well as all the coordinates.
- Finally plotting the image its keypoints and its heatmap



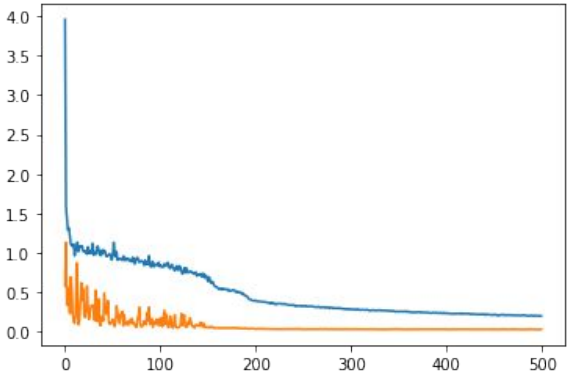
Aleatory CNN configurations

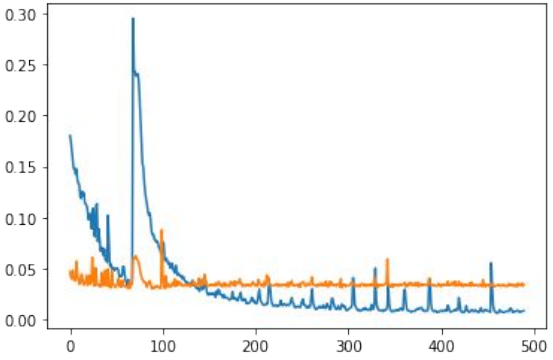
We tried different CNN architectures to find which one could fit better on our data set.



4 layers CNN Configuration

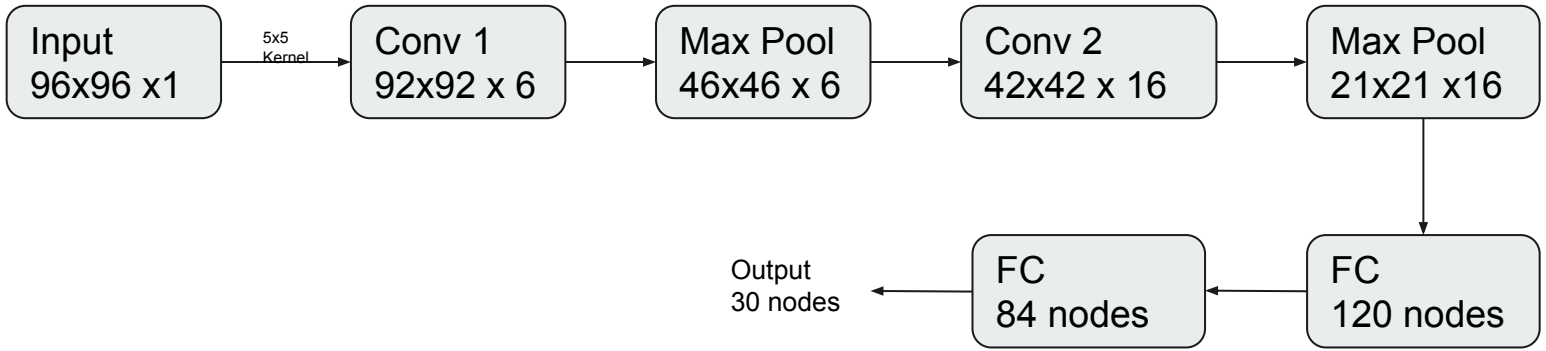
	Training Error	Validation Error
BN & DP	0.19	0.031
BN & -DP	0.01	0.021
-BN & DP	0.35	0.060
-BN & -DP	0.008	0.034



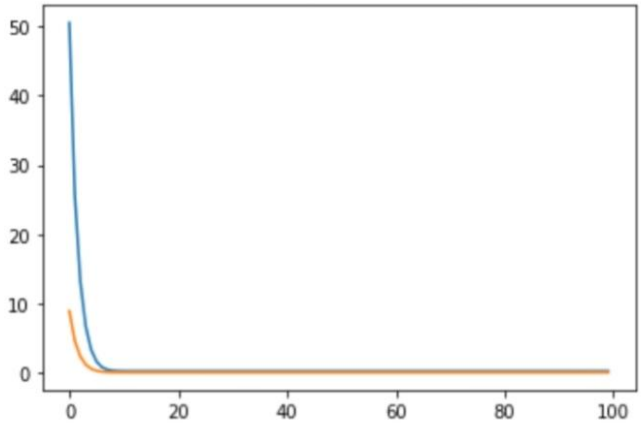


Used 500 epochs on all models

LeNet-5


Input
96x96 x1
5x5
Kernel
Conv 1
92x92 x 6
Max Pool
46x46 x 6
Conv 2
42x42 x 16
Max Pool
21x21 x16
FC
120 nodes
FC
84 nodes
Output
30 nodes

LeNet-5 Results

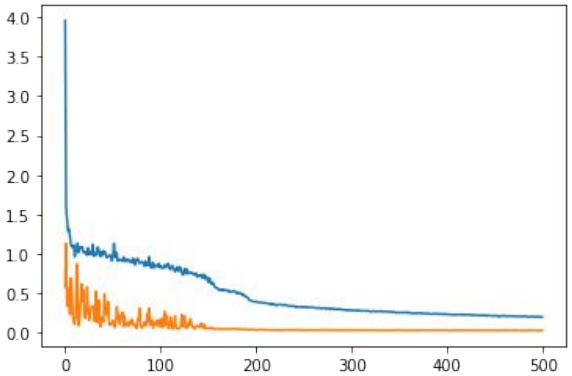


Epoch	Training error	Validation error
10	0.25154532	0.057084619
20	0.23186032	0.056160308
100	0.23151493	0.056994468

Result comparison between CNN's

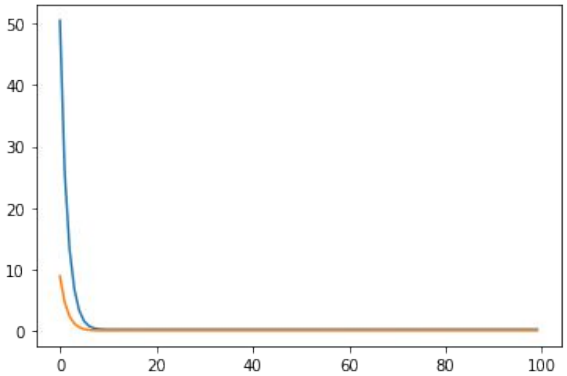
4 layers CNN

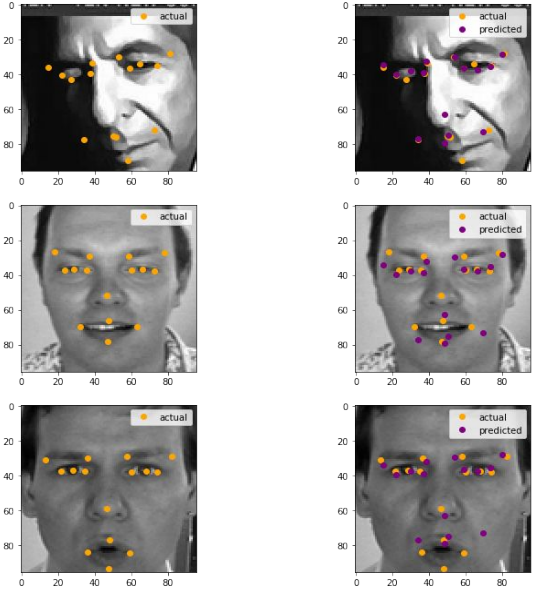
Validation Error: 0.031



LeNet

Validation Error: 0.057





Application to real life images

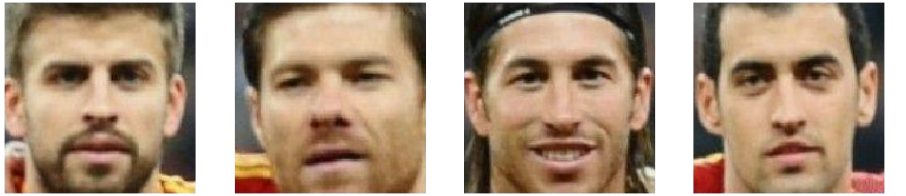
Feed our model with different images

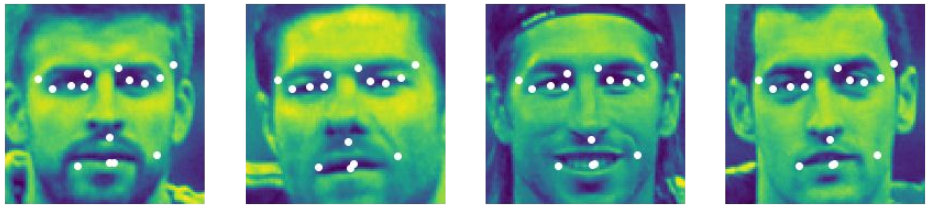
Problem: Our model was trained for an input of one channel and 96x96 dimension

Workaround:

- Find the faces on the images and extract them.
- OpenCV
- Haarcascade_frontalface_default dataset

Application to real life images





Future Scope

- Try to translate our model to recognize emotions based on the distribution of the key points detected.
- Try to finish the implementation of our model into live footage. So far we only used an existing model but the idea is do ourselves.
- Try to re-design our model to detect faces rather than using the pre-existing models like haar cascade.

Thank You