Capstone Report

Ahmed Hamdy

I. Definition

○ Project Overview

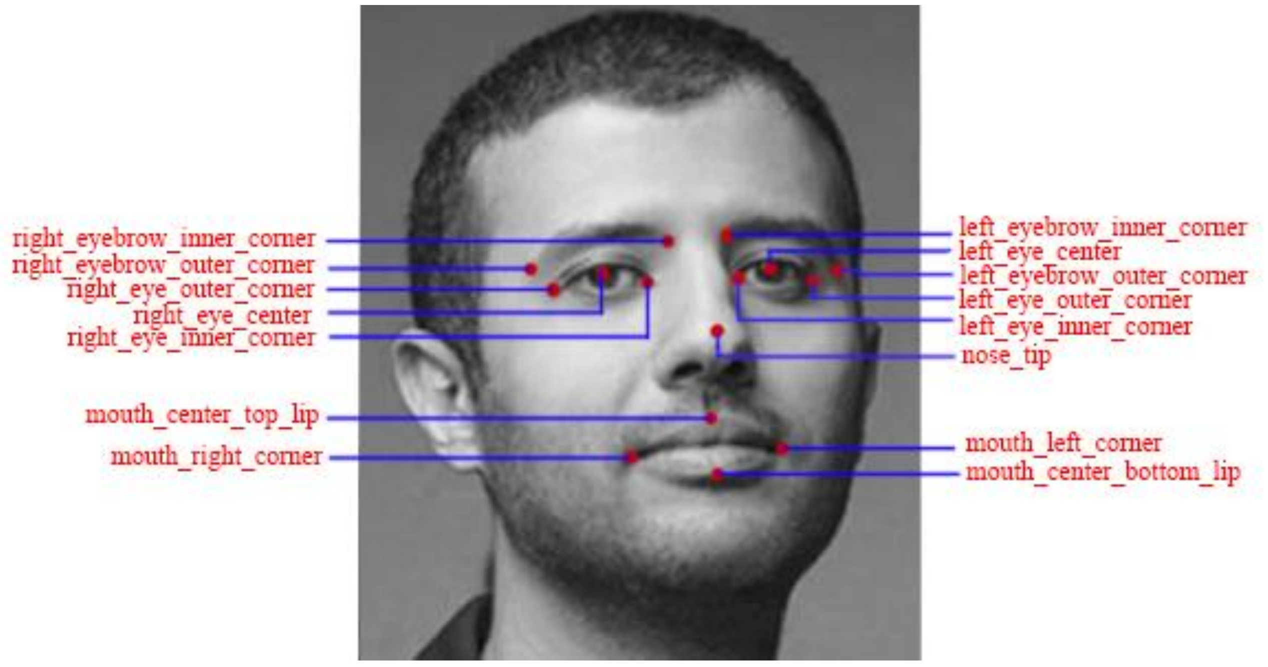
Detecting facial key-points is a very challenging problem and has been studied extensively in recent years [1, 2, 3]. Facial features vary greatly from one individual to another, and even for a single individual, there is a large amount of variation due to 3D pose, size, position, viewing angle, and illumination conditions. Computer vision research has come a long way in addressing these difficulties, but there remain many opportunities for improvement.

Detecting facial key-points can be used as a building block in several applications, such as: tracking faces in images and video, analyzing facial expressions, detecting dysmorphic, facial signs for medical diagnosis, biometrics / face recognition

○ Problem Statement

We want to predict several features on the human face. These features are clearly represented in the picture below (picture is not from the dataset, and it is used only for illustration purpose)

The solution for this problem is using a model that predicts the values for these features using lots of training examples. A CNN is the best candidate to solve such problem since it is the technique that best captures spatial features in images.



○ Metrics

Root mean squared error is used as the evaluation metric. RMSE is very common and is a suitable general-purpose error metric. Compared to the Mean Absolute Error, RMSE punishes large errors:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where $y\hat{}$ is the predicted value and y is the original value.

II. Analysis

○ Data Exploration

Each predicted keypoint is specified by an (x,y) real-valued pair in the space of pixel indices. There are 15 keypoints, which represent the following elements of the face: 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

(Left and right here refers to the point of view of the subject.)

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 input image is given in the last field of the data files and consists of a list of pixels (ordered by row), as integers in (0,255). The images are 96x96 pixels.

There are two abnormalities in the dataset: (1) some data are missing (2) some features are wrongly annotated.

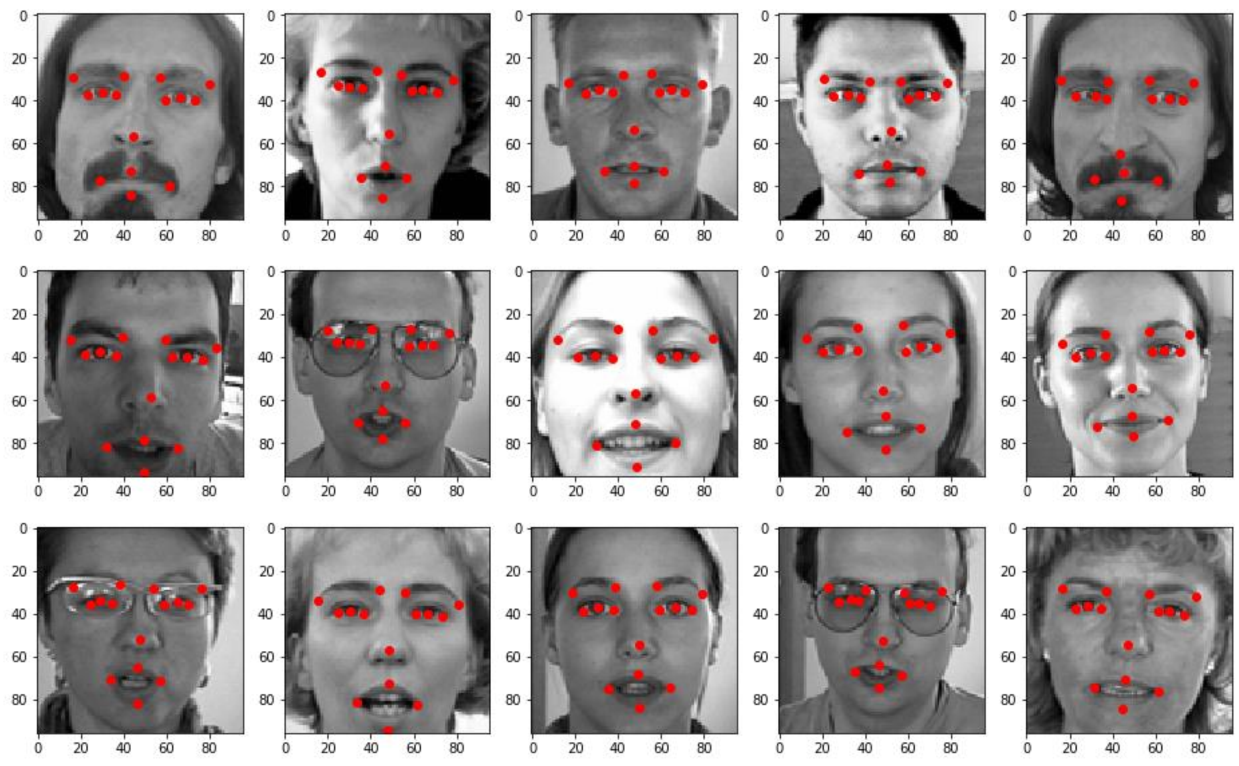
Data files:

training.csv: list of training 7049 images. Each row contains the (x,y) coordinates for 15 keypoints, and image data as row-ordered list of pixels.

Dataset is taken from a Kaggle.com : https://www.kaggle.com/c/facial-keypoints-detection/data

○ Exploratory Visualization

Here are the first 15 pictures in the dataset with the target features displayed over them



○ Algorithm and Techniques

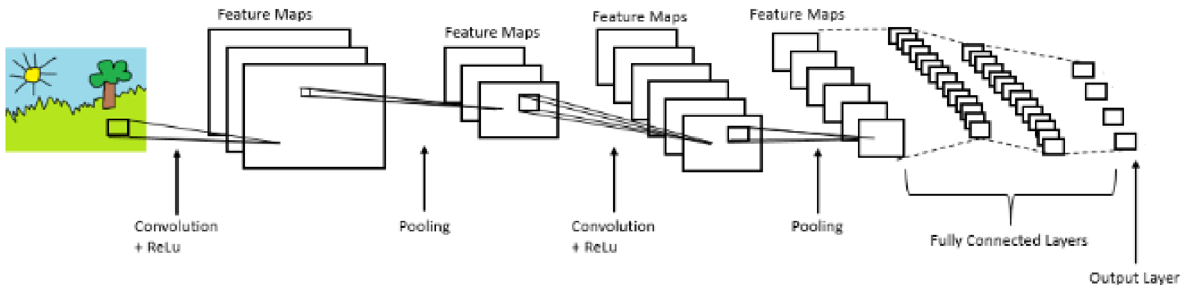
Our dataset consists of images, and the best way to extract spatial information in images datasets is by using Convolution Neural Network.

The solution will be a model of consecutive convolutional and max pooling layers such that the area of the image gradually decreases while its depth increases and carrying on only the important information that help in the prediction of the required features.

Batch normalization layers are used to increase the speed of training. And drop out layers are used to avoid overfitting.

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel**.** Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

Unlike NLP that loses the spatial information of the image when the image pixels is flattened, the fact that CNN uses kernels (which has area) that slide over the image as is, is what makes CNN powerful in capturing spatial information, which is suitable for image analysis.



Sometimes filter does not fit perfectly fit the input image. We have two options: **pad** the picture with zeros (zero-padding) so that it fits, or, drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is ***F(x) = max(0,x).*** Why ReLU is important: ReLU's purpose is to introduce non-linearity.

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types: Max Pooling, Average Pooling, Sum Pooling

Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

We **normalize** the input layer by adjusting and scaling the activations. For example, when we have features from 0 to 1 and some from 1 to 1000, we should normalize them to speed up learning.

Batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

○ Benchmark

I will compare my model against a model I have made which is a multilayer perceptron neural network which has 4 layers: input layer = 9216 neurons, two 512 neurons hidden layers, output layer = 30 (one neuron for each feature that the model will predict)

This model gave an MSE = 27.85 which I predict to decrease significantly after using a CNN model.

Benchmark model link : https://github.com/Ahmed-H-N/machine-learning/blob/master/ML%20nd/projects/capstone/facial_keypoints_detection/Capstone%20-%20Facial%20Keypoints%20Detection%20(benchmark%20model).ipynb

III. Methodology

Data Preprocessing

- Missing data are replaced by an existing value of a previous feature (in the same row)

- Wrongly annotated pictures are kept as is

- Features were provided as a long string, so a splitting is done to separate each feature into a single string, then each string is converted to float.

Implementation

- We defined a sequential model.

- We defined 12 group of layers that consists of conv2D, LeakyReLU, BatchNormalization, MaxPool2D.

- We flattened the last layer and added a Dense, DropOut, Dense layers.

- Last Layer has 30 neurons (a single neuron for each predict value)

- We compiled the model using optimizer='adam', loss='mean_squared_error', metrics=['mse']

- Then at last, the model is trained with epochs = 50, batch_size = 256, validation_split = 0.2

I used Jupyter notebook to code the model.

I first started implementing the model on my CPU, which took ages and couldn't even finish one epoch that's where I realized that I must switch to GPU

I made use of the Udacity GPU (provided in the deep learning section) to be able to train such a high parameter model.

Refinement

The main building block of the layers of the model is the following goup of layers: conv2D, LeakyReLU, BatchNormalization, MaxPool2D.

First model has 4 of these groups. Second and third models have 8 and 12 respectively.

So, I increased the number of layers and watched the effect on MSE, also I added a dropout layer at the final dense layer. And the results are documented below.

It is clear that adding more hidden layers led to better results. But adding a dropOut layer increased the error, and the error keeps increasing with the increasing the probability of the dropped neurons from 0.1 to 0.3, this imply that all the neurons has a great composite effect in training, and a small stoppage of them has affected the model badly. So, the chosen model is denoted with green in the table below.

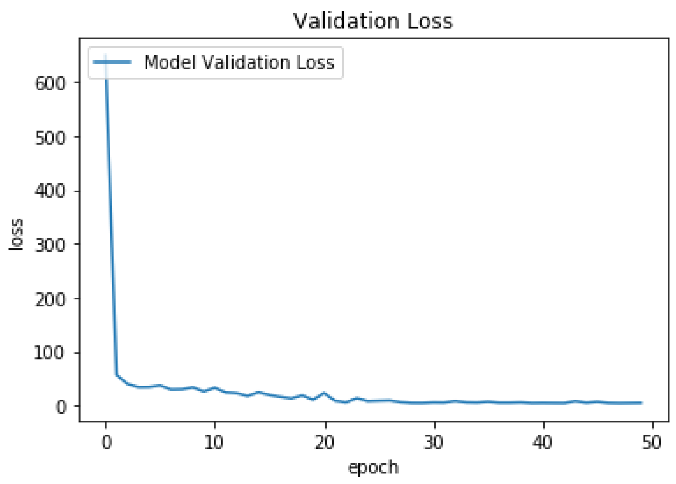
	4 groups	**8 groups**	**12 groups**	**dropout = 0.1**	**dropout = 0.3**
Val_MSE	2.63	2.85	1.19	5.78	14.23

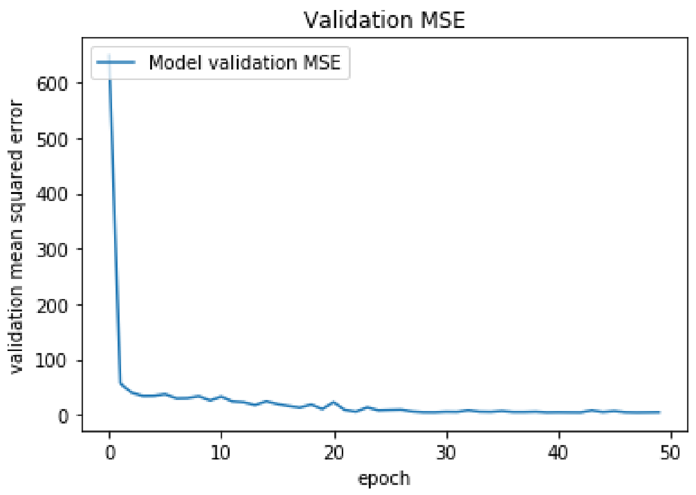
IV. Results

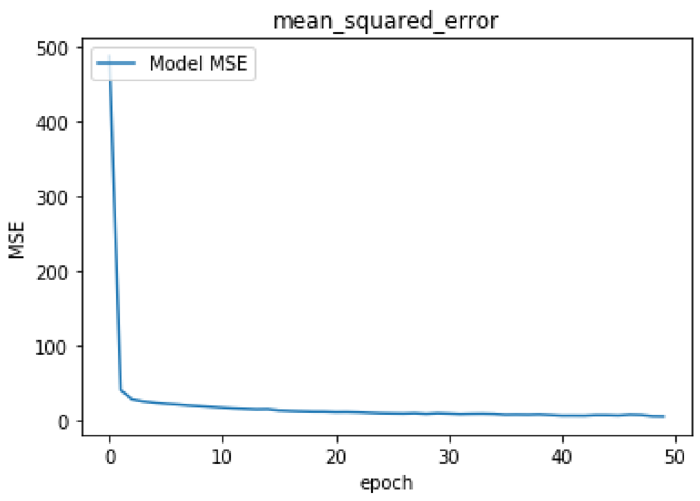
○ Model Evaluation and Validation

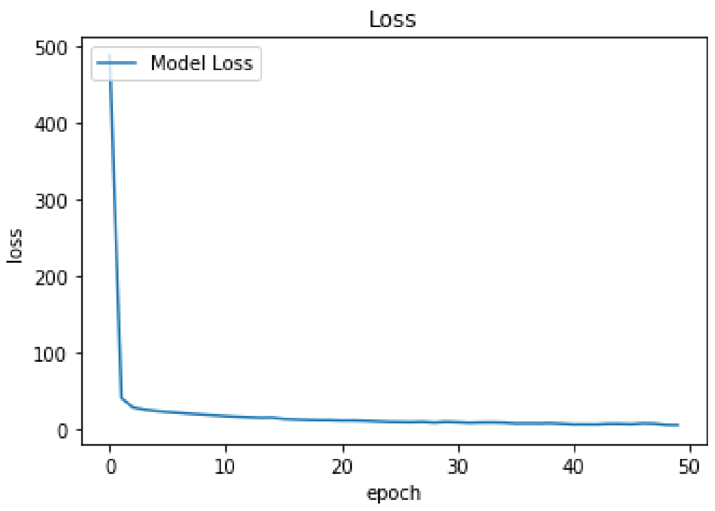
- It is clear from the figures below that the model only needed few epochs to reach to a significant low error

- The model is also tested with test data (unseen data) and performed very well with MSE = 4.96. So, the model generalizes well.









○ Justification

It is clear from the below table that our model has significant less error than the benchmark model by a factor of about 5.

	Validation MSE	**Improvement**	**Testing MSE**	**Improvement**
Benchmark	27.85	x4.7	18.31	x3.7
Our Model	5.91		4.96	

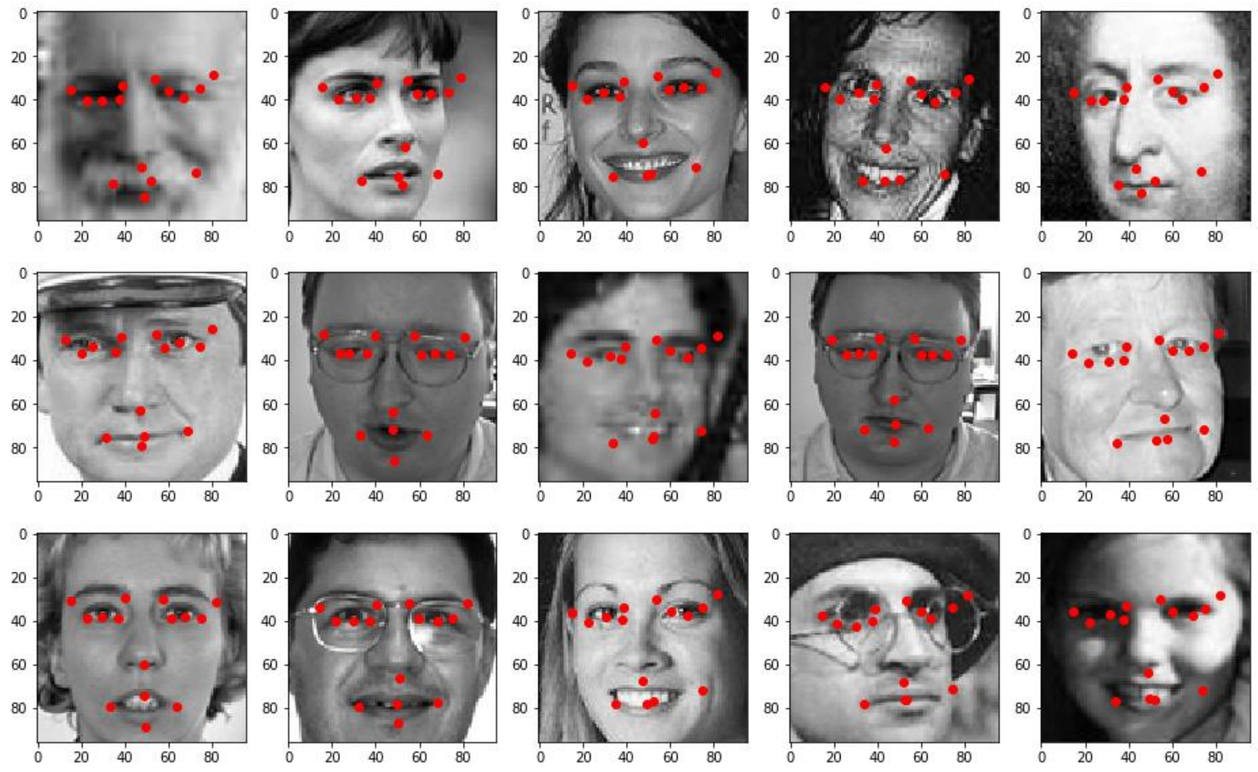
V. Conclusion

○ Free-Form Visualization

We tested the model on some unseen data to see how the model behave. The model did very well in predicting the eyebrow corners and most of the mouth features, however, the nose tip has never been predicted well.

It may be that the eyebrow has a high contrast with surrounding skin while the nose tip does not have this advantage. That's why the first is usually predicted well and the latter is badly reported.

Another possibility for the error is that the model has learnt from some pictures that are wrongly annotated, this extremely affects the accuracy of the model.



- ## Reflection

  In this project, I tried to predict important facial features so that it can be used in several application. I started by searching for a dataset, and found it on Kaggle. I explored the data and found some abnormalities that I dealt with some of them. Then I defend a CNN model which is very good with images because it captures the spatial information very well. Finally, I trained the model and tested it and compared the results with a benchmark model

  The hardest decision I had to take while implementing the solution is what to do with abnormalities of the dataset? I thought of manually correct the annotations but that would take ages since the missing and incorrect data sum up to about 4000 cells! So, I just made dummy copying to make the model work with these abnormalities and the results were both good and bad. Good at predicting the eyebrows and bad at predicting the nose tip.

- ## Improvement

  The model itself is behaving very good and stable, but a large improvement can be done to the model if the dataset abnormalities are solved. I.e. the missing data are appropriately filled, and the wrongly annotated data are corrected