Capstone Project

Karim Elgohary

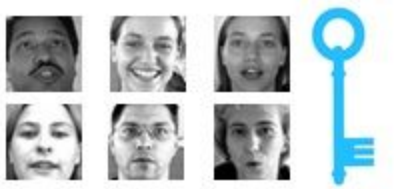
Machine Learning Engineer Nanodegree

November 29th, 2019

Facial Keypoints Detection

Definition

Project Overview



Face detection is a computer technology being used in a variety of Application that identifies human faces also it consider as a classic example of computer vision, the input
Is an image and the output is the location of key points that identifies the face from another. This process consider as psychological process by human eyes but we try to imitate this process.
Face detection is used in biometrics often as part of facial recognition system it is also used in video surveillance, human computer interface and image database management

Problem Statement

The problem is prediction problem ,The objective of this task is to predict keypoint positions on face images,and this is difficult . Detecting facial keypoints is a very challenging problem.
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.

This can be used as a building block in several Application:

1) **Biometric / face recognition**
2) **Analysis facial expressions**
3) **Tracking faces in images and video**

Since this problem is a classical problem in computer vision so the best solution to tackle this, is Convolutional Neural Network (CNN).

Metric

Mean Absolute Error('mae') $$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

mean absolute error (MAE) is a measure of difference between two continuous variables. In our case

the predicted values and the original one.Assume *X* and *Y* are variables of paired observations that express the same phenomenon.Examples of *Y* versus *X* include comparisons of predicted versus observed,and another thing that Mean Absolute Error (MAE) is the average vertical distance between each point and the identity line. MAE is also the average horizontal distance between each point and the identity line. So I chose that metric.

Analysis

Data Exploration

The Dataset from kaggle

Each predicted keypoint is specified by an (x,y) real-valued pair in the space of pixel indices. There are 15 key points, 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 misssing (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.Note that the image in Data frame is string type.

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.The shape of training data is (7049, 31) and the null values is 28.

test.csv: list of 1783 test images. Each row contains ImageId and image data as row-ordered list of pixels The shape of test data is (1783, 2).

Exploratory Visualization

Note that the data which i will train is images and its key point for prediction

The dataset looks like this:

	0	1	2	3	4
left_eye_center_x	66.0336	64.3329	65.0571	65.2257	66.7253
left_eye_center_y	39.0023	34.9701	34.9096	37.2618	39.6213
right_eye_center_x	30.227	29.9493	30.9038	32.0231	32.2448
right_eye_center_y	36.4217	33.4487	34.9096	37.2618	38.042
left_eye_inner_corner_x	59.5821	58.8562	59.412	60.0033	58.5659
left_eye_inner_corner_y	39.6474	35.2743	36.321	39.1272	39.6213
left_eye_outer_corner_x	73.1303	70.7227	70.9844	72.3147	72.5159
left_eye_outer_corner_y	39.97	36.1872	36.321	38.381	39.8845
right_eye_inner_corner_x	36.3566	36.0347	37.6781	37.6186	36.9824
right_eye_inner_corner_y	37.3894	34.3615	36.321	38.7541	39.0949
right_eye_outer_corner_x	23.4529	24.4725	24.9764	25.3073	22.5061
right_eye_outer_corner_y	37.3894	33.1444	36.6032	38.0079	38.3052
left_eyebrow_inner_end_x	56.9533	53.9874	55.7425	56.4338	57.2496
left_eyebrow_inner_end_y	29.0336	28.2759	27.5709	30.9299	30.6722
left_eyebrow_outer_end_x	80.2271	78.6342	78.8874	77.9103	77.7629
left_eyebrow_outer_end_y	32.2281	30.4059	32.6516	31.6657	31.7372
right_eyebrow_inner_end_x	40.2276	42.7289	42.1939	41.6715	38.0354
right_eyebrow_inner_end_y	29.0023	26.146	28.1355	31.05	30.9354
right_eyebrow_outer_end_x	16.3564	16.8654	16.7912	20.458	15.9259
right_eyebrow_outer_end_y	29.6475	27.0589	32.0871	29.9093	30.6722
nose_tip_x	44.4206	48.2063	47.5573	51.8851	43.2995
nose_tip_y	57.0668	55.6609	53.5389	54.1665	64.8895
mouth_left_corner_x	61.1953	56.4214	60.8229	65.5989	60.6714
mouth_left_corner_y	79.9702	76.352	73.0143	72.7037	77.5232
mouth_right_corner_x	28.6145	35.1224	33.7263	37.2455	31.1918
mouth_right_corner_y	77.389	76.0477	72.732	74.1955	76.9973
mouth_center_top_lip_x	43.3126	46.6846	47.2749	50.3032	44.9627
mouth_center_top_lip_y	72.9355	70.2666	70.1918	70.0917	73.7074
mouth_center_bottom_lip_x	43.1307	45.4679	47.2749	51.5612	44.2271
mouth_center_bottom_lip_y	84.4858	85.4802	78.6594	78.2684	86.8712
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 ...	193 192 193 194 194 194 193 192 168 111 50 12 ...	147 148 160 196 215 214 216 217 219 220 206 18...

The Null values of the data :

I will deal with this values in the data preprocessing section so I could train my model

```
left_eye_center_x            10
left_eye_center_y            10
right_eye_center_x           13
right_eye_center_y           13
left_eye_inner_corner_x    4778
left_eye_inner_corner_y    4778
left_eye_outer_corner_x    4782
left_eye_outer_corner_y    4782
right_eye_inner_corner_x   4781
right_eye_inner_corner_y   4781
right_eye_outer_corner_x   4781
right_eye_outer_corner_y   4781
left_eyebrow_inner_end_x   4779
left_eyebrow_inner_end_y   4779
left_eyebrow_outer_end_x   4824
left_eyebrow_outer_end_y   4824
right_eyebrow_inner_end_x  4779
right_eyebrow_inner_end_y  4779
right_eyebrow_outer_end_x  4813
right_eyebrow_outer_end_y  4813
nose_tip_x                    0
nose_tip_y                    0
mouth_left_corner_x        4780
mouth_left_corner_y        4780
mouth_right_corner_x       4779
mouth_right_corner_y       4779
mouth_center_top_lip_x     4774
mouth_center_top_lip_y     4774
mouth_center_bottom_lip_x    33
mouth_center_bottom_lip_y    33
Image                         0
dtype: int64
```

Algorithms and Techniques

For this problem I will use Convolutional Neural Network (CNN) and the reason for that is (CNN) are a special type of Feed-Forward Artificial Neural Networks that are generally used for image detection tasks. It accepts large array of pixels as input to the network and The result is a matrix called the Convolved and in our case it will take the image and extract the feature then produce output in our case is (x,y) real-valued pair in the space of pixel indices.

The following parameters could be tuned for the optimization:

1) **Training Parameters (Training length [number of epoch], Batch size, type of the optimizer)**
2) **Neural Network Architecture (Number of layers, Layer type[fully-connected or pooling])**
3) **The preprocessing .**

- **Using keras and tensorflow as backend**

Benchmark

I will create a simple CNN model (as the reviewer suggested) and I train it on the same data and test it on the same data that I will use it .

So i will improve the model that i will create and get better results.

Methodology

Data Preprocessing

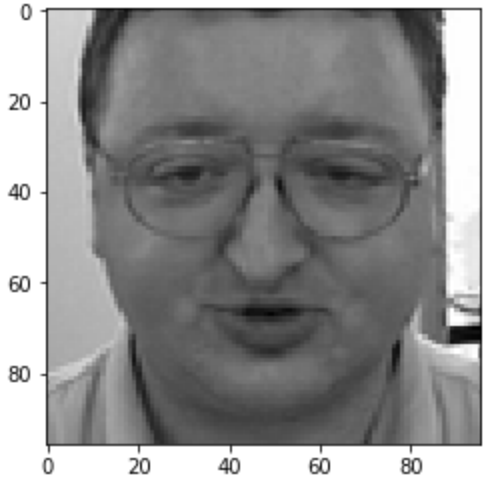
1) **So there are missing values in 28 columns. We can do two things here one remove the rows having missing values and another is the fill missing values with something. I used two options as removing rows will reduce our dataset. I filled the missing values with the previous values in that row.**

2) **As image column values are in string format and there are also some missing values(the space) so we have to split the string by space and append it and also handling missing values (with zeros)**

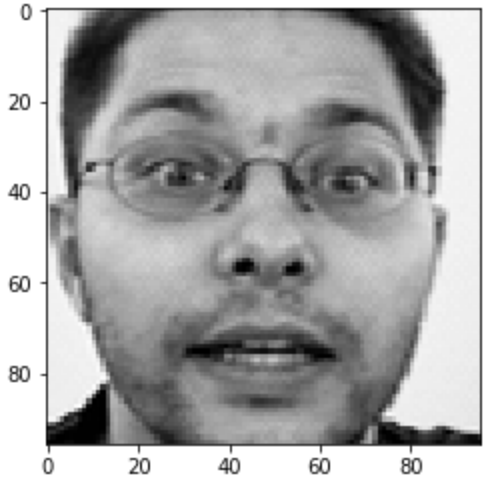
```
Train['Image'].dtype
```

```
dtype('O')
```

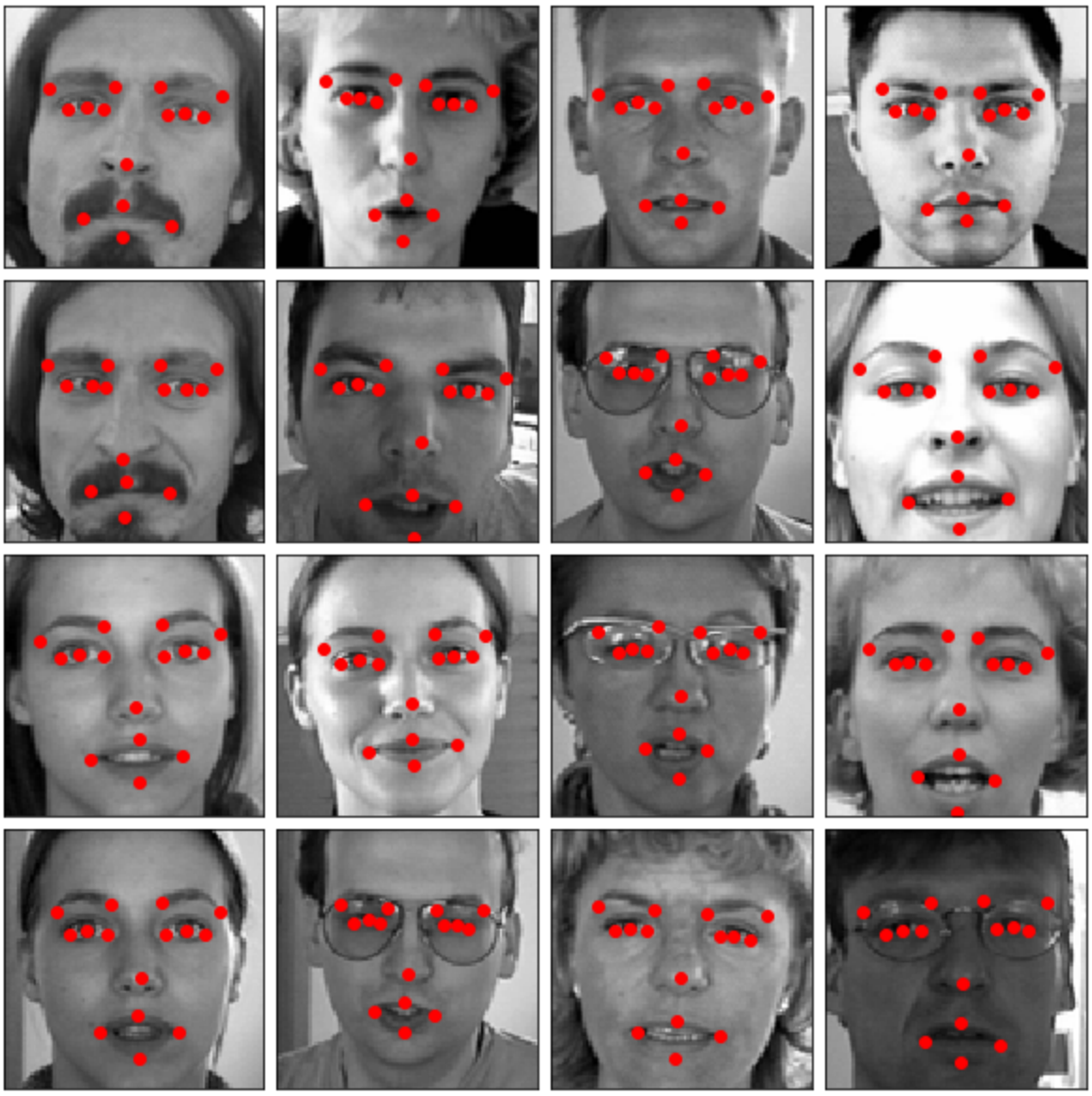
3) **Reshaping the image to (96,96) ,the image will look like :**



4) **We will repeat the step 1,2 on the test data, the test image will look like:**



5) I will split the training data for training and testing for evaluation and the reason for that is the test file have not label to evaluate the model
6) The train data shape (6549, 96, 96, 1), The test data shape for evaluation is (500, 96, 96, 1)
7) The original data with the key points and data after preprocessing



Implementation

As i said that i created a deep model of CNN that model have several layers :

1) **Conv2D: it 2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs. (i used 12 layers of Conv2D)**

2) **MaxPool2D: it is a sample-based discretization process. The objective is to down-sample an input representation .(I used 5 layers of MaxPool2D).**

3) **BatchNormalization: The layer will transform inputs so that they are standardized, meaning that they will have a mean of zero and a standard deviation of one. During training, the layer will keep track of statistics for each input variable and use them to standardize the data. (i used 12 layers of BatchNormalization)**

4) **Flatten: collapses the spatial dimensions of the input into the channel dimension. (i used 1 layer)**

5) **Dense:it is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer(i used 2 layers)**

6) **Dropout: A Simple Way to Prevent Neural Networks from Overfitting (i used 1 layer)**
7) **The activation layer is LeakyReLU**

Then i compiled the model and evaluate it , the metric is ‘mae’, the loss function is (difference between output and target variable.)and it is ‘mean_squared_error’, the optimizer update the weight parameters to minimize the loss function and i use ‘adam’ and it is the best combines with loss function

Refinement

I experimented the model with the data and i get bad accuracy so I try to improve it by increasing the number of epochs from 50 to 150 and I increased the the hidden layer as i explained in the implementation section and i get better results .

Model Evaluation and Validation

The final model that I created was the best one after my experiments and *reasonable and aligning with solution expectations* the final parameters were :

Conv2D: 1) the filters of each Conv2D layers size were[32,32,64,64,96,96,128,128,256,256,512,512]

2) the kernel size is (3,3)

3)The padding is the same

4) use_bias is False

5) The activation function: is LeakyReLU,and the reason for that is to prevent the neural network from death and that happen when we did not normalize the photo and there are values will be zero so we use it to prevent it .

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

Dense : the activation is ‘ relu’ and its filter size were[512,30]

The filter size in the last layer 30 because it is the input which we expected

Dropout is 0.1

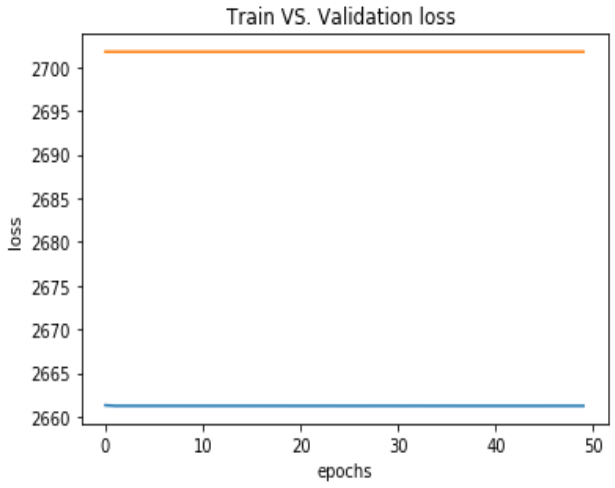
The model was enough robust for that problem based on the unseen data

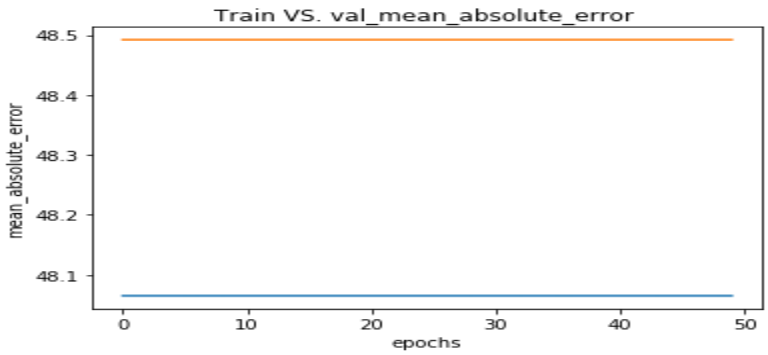
Justification

The results of the benchmark (a simple model of CNN) were:

Test loss= 2714.2135703125

Test accuracy= 48.558382629394529



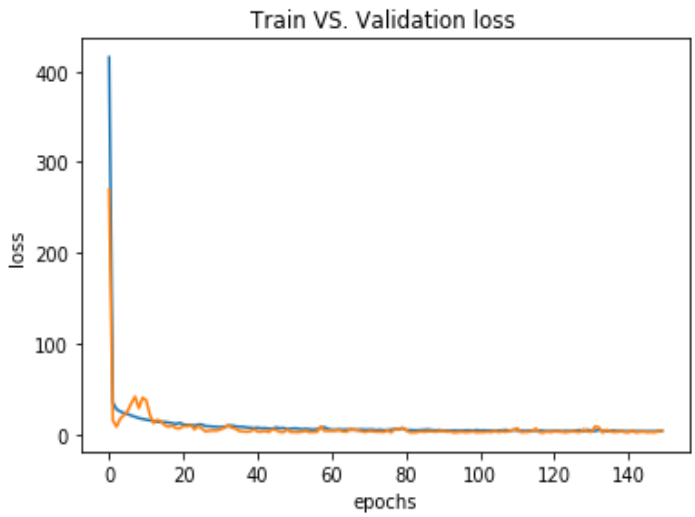


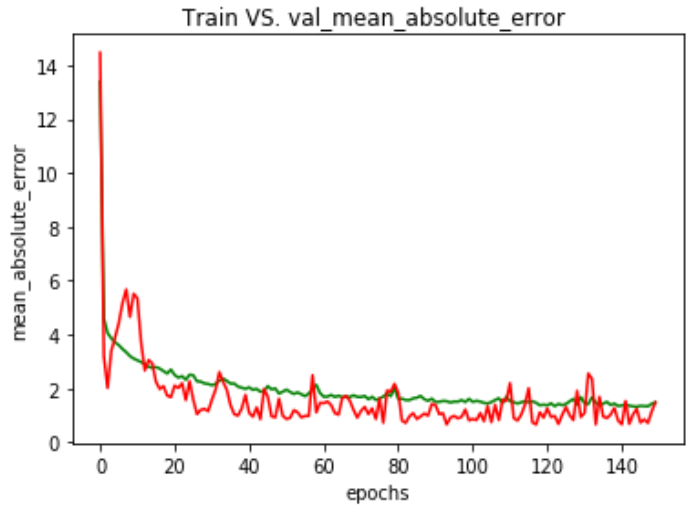
The results of the model were:

Test loss= 4.5584375686645506

Test accuracy= 1.5409615421295166

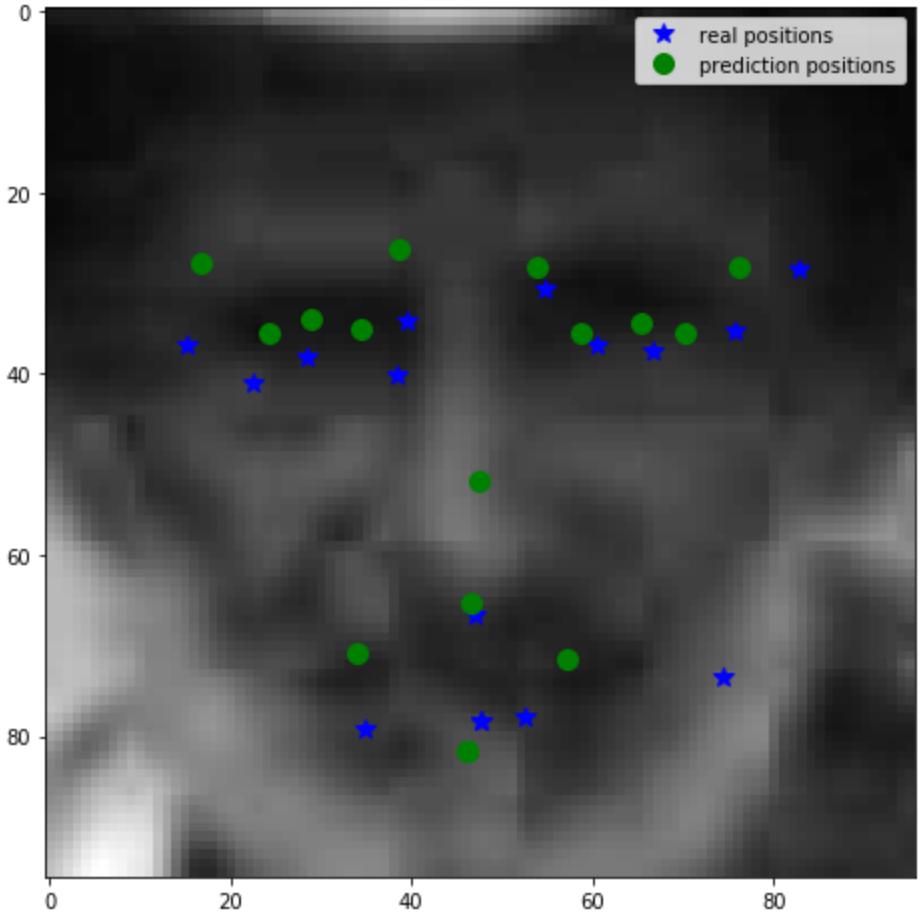
Note that the metric is mae that mean our target to minimize it .





Free-Form Visualization

The model prediction in green point while the original are blue stars



Reflection

The following process were taken to complete the process :

1) **Downloading the dataset from kaggle**
2) **Explore the data**
3) **Preprocessing the data as i explained in above section**
4) **Split the train data for training and evaluate**
5) **Create the model of the CNN and choose the right parameters for the model**

- **As this is the second time i deal with the CNN i find it interesting in**

  Tuning the parameters to get the best solution and it was a hard problem that I deal with.

- **Second interesting thing is the preprocessing the data and deal with image in that shape and converting the spaces to zeros and I take time to understand this problem.**

Improvement

There are a few ways I suggest to improve the model:

1) **Tuning and experimenting with different number of epoch**
2) **Using different optimizers**
3) **Tuning and experimenting with different size of batch size**

So there are *here further improvements as we see*
I want to learn the yolo framework and and OpenCv to improve That model .