Facial Keypoints Detection

The task: The objective of this task is to predict keypoint positions on face images. Detecing facial key points is a very challenging problem.

What type of problem: This keypoint detection problem is a regression problem since we are predicting the coordinates for the landmarks on the face which are real numbers. Detection of facial keypoints is very useful for a number of tasks like facial recognition, detection etc. The keypoints selected in this code uses 15 keypoints representing the various coordinates on the human face.

Dataset: The dataset for this problem was downloaded from Kaggle. In the space of pixel indices, each anticipated keypoint is defined as a (x,y) real-valued pair. The following components of the face are represented by 15 keypoints. The input picture is a list of pixels (sorted by row) stored as numbers in the final field of the data files (0,255). The pictures have a resolution of 96x96 pixels.

```
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   left_eye_center_x          7039 non-null   float64
 1   left_eye_center_y          7039 non-null   float64
 2   right_eye_center_x         7036 non-null   float64
 3   right_eye_center_y         7036 non-null   float64
 4   left_eye_inner_corner_x    2271 non-null   float64
 5   left_eye_inner_corner_y    2271 non-null   float64
 6   left_eye_outer_corner_x    2267 non-null   float64
 7   left_eye_outer_corner_y    2267 non-null   float64
 8   right_eye_inner_corner_x   2268 non-null   float64
 9   right_eye_inner_corner_y   2268 non-null   float64
 10  right_eye_outer_corner_x   2268 non-null   float64
 11  right_eye_outer_corner_y   2268 non-null   float64
 12  left_eyebrow_inner_end_x   2270 non-null   float64
 13  left_eyebrow_inner_end_y   2270 non-null   float64
 14  left_eyebrow_outer_end_x   2225 non-null   float64
 15  left_eyebrow_outer_end_y   2225 non-null   float64
 16  right_eyebrow_inner_end_x  2270 non-null   float64
 17  right_eyebrow_inner_end_y  2270 non-null   float64
 18  right_eyebrow_outer_end_x  2236 non-null   float64
 19  right_eyebrow_outer_end_y  2236 non-null   float64
 20  nose_tip_x                 7049 non-null   float64
 21  nose_tip_y                 7049 non-null   float64
 22  mouth_left_corner_x        2269 non-null   float64
 23  mouth_left_corner_y        2269 non-null   float64
 24  mouth_right_corner_x       2270 non-null   float64
 25  mouth_right_corner_y       2270 non-null   float64
 26  mouth_center_top_lip_x     2275 non-null   float64
 27  mouth_center_top_lip_y     2275 non-null   float64
 28  mouth_center_bottom_lip_x  7016 non-null   float64
 29  mouth_center_bottom_lip_y  7016 non-null   float64
 30  Image                      7049 non-null   object
dtypes: float64(30), object(1)
memory usage: 1.7+ MB
```

Model: I used Convolutional Neural Network for this problem. The model was trained on a set of approx 1700 images and around 300 images were taken for validation purposes.

The images were grayscale and had the shape 96 X 96. The network architecture is shown below.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 96, 96, 32)	288
batch_normalization (BatchNo	(None, 96, 96, 32)	128
max_pooling2d (MaxPooling2D)	(None, 48, 48, 32)	0
conv2d_1 (Conv2D)	(None, 48, 48, 32)	9216
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 32)	0
conv2d_2 (Conv2D)	(None, 24, 24, 64)	18432
batch_normalization_1 (Batch	(None, 24, 24, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 64)	0
conv2d_3 (Conv2D)	(None, 12, 12, 128)	73728
batch_normalization_2 (Batch	(None, 12, 12, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 30)	7710

Total params: 1,290,174
Trainable params: 1,289,726
Non-trainable params: 448

Hyperparameter:

Epochs= 60

batchsize= 32

Optimizer= ADAM

loss= Mean squared error

Eval metrics= Accuracy

Reason for using this algorithm: Convolutional Neural Networks (CNN) are used in a variety of applications. It is, without a doubt, the most widely used deep learning architecture. CNN has become the go-to model for every image-related issue. They outperform the competitors in terms of accuracy. The major benefit of CNN over its predecessors is that it identifies important features without the need for human intervention.CNN is computationally efficient. It performs parameter sharing and employs specific convolution and pooling methods. Thus I chose CNN baseline architecture to solve this problem of Facial keypoints detection.

Results: 85% train accuracy, 77% validation accuracy & 2.9% loss. (Score on Kaggle- 2.5 RMSE)