write up

ashish.mishra0011

November 2020

1 Introduction

We are using the convolutional neural network to determine the relationship between the orientation of an ellipse just by giving images as input.

2 Data

We are training our model using simulated data. We have used ellipse of skimage.draw module in python. We have generated the ellipses at different angles. Each angle is determined using x-axis as zero slope angle. We have generated 900 images in which the angle of the ellipses increases iteratively by 0.2°. Each one is a jpeg image having a black background and white filled ellipse.

3 Model

1. We have used filters of size 16,32,64. After iteration through each layer these filters increases. The purpose of these filters is to find the feature values in the input data.

2. Then we use a for loop in which we set the fist CNN layer as our input. As we need to give some input to the first layer. We have used multiple layers from Keras for input:

   (a) Conv2D: This layer converts our input image into a matrix of pixel values. We have used a kernel of size (3,3) that assigns the pixel value to the input image.

(b) Activation: Each node in the neural network gets activated if it gets an input value above a threshold. This threshold value is determined by the activation function. We have used the activation function "ReLu" over here.

(c) BatchNormalization: This layer is used normalize the input values of a batch. For each batch, it calculates mean and deviation and normalizes each batch.

(d) Max Pooling: This layer is basically used to decrease the size of the input volume. The height and width are reduced using pool size and strides.

Note: These layers are only used over inputs.

3. Next comes to the layers which are used over inputs as well as consequent outputs.

   (a) Flatten: This layer converts the input matrix into a vector that can be fed to our model.

   (b) Dense: This layer is used to make the layers interconnected. It's basically used to form interconnection among the layers.

   (c) Activation: Again Relu is used to activate.

   (d) Batch Normalization: Same as used in the previous one.

   (e) Dropout: This layer dropout the connection between the nodes which are unable to activate the next node. Basically, it removes the relations which are not changing our output much.

   (f) Again dense layer and activation layer is used. At last, we need to check if the regression layer still needs to be added or not.

4. Finally we define our model as the input value and the obtained predicted value.

4 Final Regression Code

In this section, we load our images from the directory. Then we rescale the pixel intensities of images from (0,1). We rescale the input angle between (0,1). We divide the training and testing data in 75% and 25% respectively. Our validation data is the same as our training dataset.

(a) We have used the image size as 64,64,3(height,width,depth).

(b) We have used the loss function as squared mean error and the metric(which determines the accuracy or loss in each step) as the squared mean error. Particularly this loss function and metric gives us the minimum loss and maximum accuracy.

(c) We have used the Adam optimizer with learning rate 10^{-3} and batch size about 8 or 16 or 32. We are running the whole program for 200 epochs.

(d) Finally we have calculated the error on our own that comes around to be 10%.