CAPSTONE REPORT-MLND-Vidit Shah

REPORT

In this report I will describe the steps taken and approach used for building My project.

Let’s Begin.

HELPER FUNCTIONS:

Apart from my own I would first like to thank HVASS LABORATORIES open sourcing his code.I have used his plotting functions to describe and show weights

BUILDING MODEL

1.First we describe our parameters and Hyper parameters. To get lay man idea difference between parameter and hyper parameter ,Hyper parameter affects performance of our model and often tuned(changed).For example- Batch size, epoch and learning rate. On the other hand Parameters are fixed image sizes and few constants we need.

2.Now we describe Weights and Biases for our model

```
def new_weights(shape):
   return tf.Variable(tf.truncated_normal(shape, stddev=0.05))
```

Above, is an example of describing weights.

Shape takes 4 dimensional tensor having values of following parameters

```
    shape = [filter_size, filter_size, num_input_channels, num_filters]
```

I have described bias similarly.

```
def new_biases(length):
   return tf.Variable(tf.constant(0.05, shape=[length]))
```

3.Now we describe our main Convolutional Layer where all the calculation takes place.

```
  layer = tf.nn.conv2d(input=input,filter=weights,strides=[1, 1, 1, 1],
padding='SAME')
```

tf.nn.conv2d takes input image,weights,numper of pixel it needs to move(stride) and padding as parameter and handles all the complex calculation behind it.

Later Max pooling comes in the picture tf.nn.max_pool handles all the operation for max pooling.

After consecutive convolutional ,poling and activation we need to flatten our dimension from 4d to 2d for prediction purposes.

Hence we reach to step five

5. Flattening Layer

num_features = layer_shape[1:4].num_elements()

layer_flat = tf.reshape(layer, [-1, num_features])

With help of tf.reshape we reshape our image to desired dimension and forward it to fully connected layer

6.FULLY CONNECTED LAYER

The output from Fully Connected Layer is fed into softmax function and predictions is made with help of tf.argmax() function and thus, now we need to Calculate Error

7.Error

Error Is Calculated With help of Cross Entropy and Gradient Descent Optimizer taking in consideration of Learning Rate between Predicted output and True output.

Now we are done with preparing the Model. We define our paths to images where they are located and split them into training and testing Data.

RUNNING THE MODEL:

To run the model we first initialize all the variables and define a Session.

We pass our model inside session and print out the loss .Our goal is to minimize the Loss and increase the accuracy and not to allow the model to over fit.

After running we test images and see whether the model trained is performing well or not.