UDACITY

PROJECT

Object Classification

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Required Files and Tests

The project submission contains the project notebook, called “dlnd_image_classification.ipynb”.

All the unit tests in project have passed.

Preprocessing

The `normalize` function normalizes image data in the range of 0 to 1, inclusive.

The `one_hot_encode` function encodes labels to one-hot encodings.

Neural Network Layers

The neural net inputs functions have all returned the correct TF Placeholder.

The `conv2d_maxpool` function applies convolution and max pooling to a layer.

The convolutional layer should use a nonlinear activation.

This function shouldn’t use any of the tensorflow functions in the tf.contrib or tf.layers namespace.

The `flatten` function flattens a tensor without affecting the batch size.

The `fully_conn` function creates a fully connected layer with a nonlinear activation.

The `output` function creates an output layer with a linear activation.

Neural Network Architecture

The `conv_net` function creates a convolutional model and returns the logits. Dropout should be applied to alt least one layer.

Neural Network Training

The `train_neural_network` function optimizes the neural network.

The `print_stats` function prints loss and validation accuracy.

The hyperparameters have been set to reasonable numbers.

The neural network validation and test accuracy are similar. Their accuracies are greater than 50%.

So the validation accuracy going down seems like a sure sign of overfitting, which is why we are making you print it out. But that doesn't seem to be the case here.

After looking at your code for hours, I still don't know exactly why it won't work. All I could figure out is that there's a problem with your dropout. Without using any dropout (or setting keep_prob to 1), it works. Otherwise, it doesn't. However, everything seems correct, so I have no idea why it isn't working.

A weird architecture like this seemed to work though:

```
def conv_net(x, keep_prob):
    """
    Create a convolutional neural network model
    : x: Placeholder tensor that holds image data.
    : keep_prob: Placeholder tensor that hold dropout keep probability.
    : return: Tensor that represents logits
    """
    # TODO: Apply 1, 2, or 3 Convolution and Max Pool layers
    #    Play around with different number of outputs, kernel size and stride
    # Function Definition from Above:
    x_tensor=conv2d_maxpool(x, conv_num_outputs=32, conv_ksize=(4,4), conv_strides=(2,2), pool_ksize=(4,4), pool_strides
=(2,2))
    x_tensor=tf.contrib.layers.dropout(x_tensor, keep_prob)
    x_tensor=conv2d_maxpool(x, conv_num_outputs=64, conv_ksize=(4,4), conv_strides=(2,2), pool_ksize=(4,4), pool_strides
=(2,2))
    x_tensor=tf.contrib.layers.dropout(x_tensor, keep_prob)
    x_tensor=conv2d_maxpool(x, conv_num_outputs=128, conv_ksize=(4,4), conv_strides=(2,2), pool_ksize=(4,4), pool_stride
s=(2,2))
    x_tensor=tf.contrib.layers.dropout(x_tensor, keep_prob)

    # TODO: Apply a Flatten Layer
    # Function Definition from Above:
    x_tensor=flatten(x_tensor)


    # TODO: Apply 1, 2, or 3 Fully Connected Layers
    #    Play around with different number of outputs
    # Function Definition from Above:
    x_tensor = fully_conn(x_tensor, num_outputs=512)
    x_tensor = fully_conn(x_tensor, num_outputs=256)
    x_tensor = fully_conn(x_tensor, num_outputs=128)

    # TODO: Apply an Output Layer
    #    Set this to the number of classes
    # Function Definition from Above:
    x_tensor = output(x_tensor, num_outputs=10)


    # TODO: return output
    return x_tensor
```

```
"""
DON'T MODIFY ANYTHING IN THIS CELL THAT IS BELOW THIS LINE
"""

##############################
## Build the Neural Network ##
##############################

# Remove previous weights, bias, inputs, etc..
tf.reset_default_graph()

# Inputs
x = neural_net_image_input((32, 32, 3))
y = neural_net_label_input(10)
keep_prob = neural_net_keep_prob_input()

# Model
logits = conv_net(x, keep_prob)

# Name logits Tensor, so that is can be loaded from disk after training
logits = tf.identity(logits, name='logits')

# Loss and Optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y))
optimizer = tf.train.AdamOptimizer().minimize(cost)

# Accuracy
correct_pred = tf.equal(tf.argmax(logits, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32), name='accuracy')

tests.test_conv_net(conv_net)
```

I noticed you modified some of the code where it said 'do not modify below this line'. Perhaps you changed something else somewhere and it's screwing it up. The other thing to try is setting `stddev=0.1` in your weight initializations.

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Student FAQ