TensorFlow Convolution Layer

Let's examine how to implement a CNN in TensorFlow.

TensorFlow provides the **tf.nn.conv2d()** and **tf.nn.bias_add()** functions to create your own convolutional layers.

```python
# Output depth
k_output = 64

# Image Properties
image_width = 10
image_height = 10
color_channels = 3

# Convolution filter
filter_size_width = 5
filter_size_height = 5

# Input/Image
input = tf.placeholder(
    tf.float32,
    shape=[None, image_height, image_width, color_channels])

# Weight and bias
weight = tf.Variable(tf.truncated_normal(
    [filter_size_height, filter_size_width, color_channels, k_output]))
bias = tf.Variable(tf.zeros(k_output))

# Apply Convolution
conv_layer = tf.nn.conv2d(input, weight, strides=[1, 2, 2, 1], padding='SAM
# Add bias
conv_layer = tf.nn.bias_add(conv_layer, bias)
# Apply activation function
conv_layer = tf.nn.relu(conv_layer)
```

The code above uses the **tf.nn.conv2d()** function to compute the convolution with `weight` as the filter and `[1, 2, 2, 1]` for the strides. TensorFlow uses a stride for each `input` dimension, `[batch, input_height, input_width, input_channels]`. We are generally always going to set the stride for `batch` and `input_channels` (i.e. the first and fourth element in the `strides` array) to be `1`.

You'll focus on changing `input_height` and `input_width` while setting `batch` and `input_channels` to 1. The `input_height` and `input_width` strides are for striding the filter over `input`. This example code uses a stride of 2 with 5x5 filter over `input`.

The **tf.nn.bias_add()** function adds a 1-d bias to the last dimension in a matrix.