TensorFlow 3 - Max Pooling Layers in TensorFlow

March 27, 2018

0.0.1 Max Pooling Layers in TensorFlow

If we have a tensor with values `[[1,0],[4,6]]`, after applying a max pooling operation of size `[2x2]`, we get `6` because 6 is the maximum value in the range of `[2x2]`.

The benefit of the max pooling operation is to reduce the size of the input, and allow the neural network to focus on only the most important elements. Max pooling does this by only retaining the maximum value for each filtered area, and removing the remaining values.

We apply **max pooling** to our data by using the `tf.nn.max_pool()` function.

The `tf.nn.max_pool()` function is defined as

```
tf.nn.max_pool(
   input,
   ksize,
   strides,
   padding,
   data_format = 'NHWC',
   name = None
)
```

The function performs max pooling on the input with the size of the filter defined as `ksize` and `strides`. The 2x2 filters with a stride of 2x2 are common i.e. it reduces the size of the input by half.

The `ksize` and `strides` parameters are structured as 4-element lists, with each element corresponding to a dimension of the input tensor(`[batch, height, width, channels]`). Note that the batch and channel dimensions are usally set to `1` for both `ksize` and `strides`.

```
In [2]: import tensorflow as tf
        import numpy as np

        # Construct the input to be 4D (batch_size, height, width, depth)
        # (1, 4, 4, 1)
        x = np.array([
            [0, 1, 0.5, 10],
            [2, 2.5, 1, -8],
            [4, 0, 5, 6],
            [15, 1, 2, 3]], dtype = np.float32).reshape((1,4,4,1))
        # print(x.shape) # (1, 4, 4, 1)
        X = tf.constant(x)
```

```
        def conv2d(input):
            # The output shape is (1, 2, 2, 3) --> output_depth = 3
            # The shape of the filter weight is (height, width, input_depth, output_depth)
            F_W = tf.Variable(tf.truncated_normal([3, 3, 1, 3])) # initialise weights to random
            F_b = tf.Variable(tf.zeros(3)) # initialise bias to zeros
            strides = [1,1,1,1]
            padding = 'VALID'
            # compute the convolution: input * W
            conv = tf.nn.conv2d(input, F_W, strides, padding)
            # add the bias: input * W + bias
            conv = tf.nn.bias_add(conv, F_b)
            # apply the relu activation function
            conv = tf.nn.relu(conv)
            return conv

        def max_pool(input):
            return tf.nn.max_pool(
                        input,
                        ksize = [1,2,2,1], # filter of size 2x2
                        strides = [1,2,2,1], # stride of 2 for vertical and horizontal direction
                        padding = 'VALID')

        # Convolution
        conv = conv2d(X)
        print(conv.shape)

        # Max-pooling
        maxpool = max_pool(conv)
        print(maxpool.shape)
```

```
(1, 2, 2, 3)
(1, 1, 1, 3)
```