Multiple Layers

Discover the nuances of increasing the size and depth of a CNN.

Chapter Goals:

- Add an additional convolution and max pooling layer to the CNN

A. Adding extra layers

Like all neural networks, CNNs can benefit from additional layers. The additional layers allow a CNN to essentially stack multiple filters together for use on the image data. However, similar to building any neural network, we need to be careful of how many additional layers we add. If we add too many layers to a model, we run the risk of having it overfit to the training data and therefore generalizing very poorly. Furthermore, each additional layer adds computational complexity and increases training time for our model.

Since the MNIST images are pretty simple and only have one channel, we won't need more than one additional convolution and max pooling layer. However, in the next two sections of this course you will build CNNs for classifying much more complex images, necessitating the use of many more convolution layers.

B. Increased filters

We usually increase the number of filters in a convolution layer the deeper it is in our model. In this case, our second convolution layer has 64 filters, compared to the 32 filters of the first convolution layer. The deeper the convolution layer, the more detailed the extracted features become. For example, the first convolution layer may have filters that extract features such as lines, edges, and curves. When we get to the second level, the filters of the convolution layer could now extract more distinguishing features, such as the sharp angle of a 7 or the intersecting curves of an 8.

Time to Code!

We're going to add another convolution layer to the model, which will take in

the output from the previous pooling layer.

Set `conv2` equal to `tf.layers.conv2d` applied with `pool1` as the inputs. The function will use `64` filters, a kernel size of `[5, 5]`, `'same'` padding, and `tf.nn.relu` activation. We'll also set the `name` argument to `'conv2'`.

We apply max pooling to the output of the added convolution layer.

Set `pool2` equal to `tf.layers.max_pooling2d` applied with `conv2` as the inputs. The function will use a pooling size of `[2, 2]` and a stride size of 2, both horizontally and vertically. We'll also set the `name` argument to `'pool2'`.

```
import tensorflow as tf

class MNISTModel(object):
    # Model Initialization
    def __init__(self, input_d
        self.input_dim = input_
        self.output_size = outp

    # CNN Layers
    def model_layers(self, inpu
        reshaped_inputs = tf.re
            inputs, [-1, self.i
        # Convolutional Layer #
        conv1 = tf.layers.conv1
            inputs=reshaped_inp
            filters=32,
            kernel_size=[5, 5],
            padding='same',
            activation=tf.nn.re
            name='conv1')
        # Pooling Layer #1
        pool1 = tf.layers.max_p
            inputs=conv1,
            pool_size=[2, 2],
            strides=2,
            name='pool1')
        # CODE HERE
```