Logits

Use a fully-connected layer to extract multiclass logits from the CNN.

Chapter Goals:

- Obtain the logits for each digit class

A. Multiclass logits

Since there are 10 possible digits an MNIST image can be, we use a 10 neuron fully-connected layer to obtain the logits for each digit class. The logits are the output of the `model_layers` function.

The rest of the model follows the standard format for multiclass classification:

- Softmax applied to the logits to convert them into per class probabilities
- The `labels` are one-hot vectors, where the "hot index" corresponds to the digit in the MNIST image
- Softmax cross entropy to calculate loss

Time to Code!

In this chapter, we'll create the helper function, `get_logits`, which obtains logits from the previous chapter's `dropout`.

We use a final fully-connected layer to obtain our logits, which we return as the output of our function.

Set `logits` equal to `tf.layers.dense` applied with `dropout` as the inputs, `self.output_size` as the output size, and `name` equal to `'logits'`.
Then return `logits`.

```
import tensorflow as tf

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

```
    self.output_size = output_size

# Get logits from the dropout layer
def get_logits(self, dropout):
    # CODE HERE
    pass

# CNN Layers
def model_layers(self, inputs, is_training):
    reshaped_inputs = tf.reshape(
        inputs, [-1, self.input_dim, self.input_dim, 1])
    # Convolutional Layer #1
    conv1 = tf.layers.conv2d(
        inputs=reshaped_inputs,
        filters=32,
        kernel_size=[5, 5],
        padding='same',
        activation=tf.nn.relu,
        name='conv1')
    # Pooling Layer #1
    pool1 = tf.layers.max_pooling2d(
        inputs=conv1,
        pool_size=[2, 2],
        strides=2,
        name='pool1')
    # Convolutional Layer #2
    conv2 = tf.layers.conv2d(
        inputs=pool1,
        filters=64,
        kernel_size=[5, 5],
        padding='same',
        activation=tf.nn.relu,
        name='conv2')
    # Pooling Layer #2
    pool2 = tf.layers.max_pooling2d(
        inputs=conv2,
        pool_size=[2, 2],
        strides=2,
        name='pool2')
    # Dense Layer
    hwc = pool2.shape.as_list()[1:]
    flattened_size = hwc[0] * hwc[1] * hwc[2]
    pool2_flat = tf.reshape(pool2, [-1, flattened_size])
    dense = tf.layers.dense(pool2_flat, 1024,
        activation=tf.nn.relu, name='dense')
    # Apply Dropout
    dropout = tf.layers.dropout(dense, rate=0.4,
        training=is_training)
    # Get and Return Logits
    return self.get_logits(dropout)
```