Reshaping

Reshape image data from the basic 2-D matrix format into NHWC format.

Chapter Goals:

- Learn about NHWC format
- Reshape the input data to NHWC

A. NHWC format

As mentioned in the previous chapter, `inputs` has shape `(batch_size, self.input_dim**2)`. However, in order to use the data with our convolutional neural network, we need to get it into NHWC format.

NHWC format has a shape with four dimensions:

1. Number of image data samples (batch size)
2. Height of each image
3. Width of each image
4. Channels per image

The height and width of each image from the dataset is `self.input_dim`, while the number of channels is 1 (since the images are grayscale). The number of samples, i.e. batch size, is unspecified, since we allow for a variable number of input images.

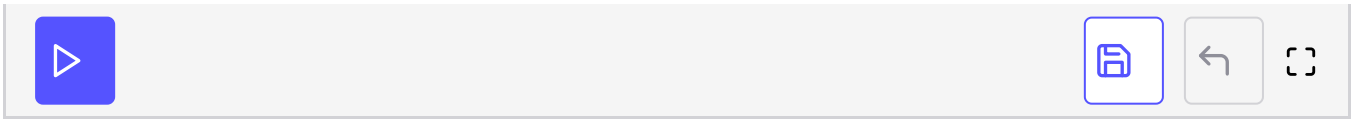
B. Reshaping the data

The function we use to reshape data in TensorFlow is `tf.reshape`. It takes in a tensor and a new shape as required arguments. The new shape must be able to contain all the elements from the input tensor. For example, we could reshape a tensor from original shape `(5, 4, 2)` to `(4, 10)`, since both shapes contain 40 elements. However, we could not reshape that tensor to `(3, 10, 2)` or `(1, 10)`, since those shapes contain 60 and 10 elements, respectively.

We are allowed to use the special value of -1 in at most one dimension of the

new shape. The dimension with -1 will take on the value necessary to allow the new shape to contain all the elements of the tensor. Using the previous example, if we set a new shape of `(1, 10, -1)`, then the third dimension will have size 4 in order to contain the 40 elements. Since the batch size of our input image data is unspecified, we use -1 for the first dimension when reshaping `inputs`.

```
with tf.Session() as sess:
    input_arr = sess.run(inputs)
    reshaped_arr = sess.run(
        tf.reshape(inputs, [-1, 2, 2, 1])
    )
print(repr(input_arr))
print(repr(reshaped_arr))
```



Example usage of tf.reshape.

In the example, the input tensor, `inputs`, has shape `(batch_size, 4)`. We use `tf.reshape` to convert it to a tensor with shape `(batch_size, 2, 2, 1)`.

Time to Code!

In the next several chapters, we'll be working on completing the `model_layers` function of the `MNISTModel` object. The `model_layers` function sets up the layers of the CNN.

We'll first convert the input data, `inputs`, into NHWC format, with 1 channel. In this case, the H and W dimensions will both be `self.input_dim`, while the C dimension is 1. We use -1 in the N dimension to represent an unspecified batch size.

Set `reshaped_inputs` equal to `tf.reshape` with first argument `inputs` and second argument a list/tuple of four integers representing the NHWC shape.

```
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
```

```
    # CNN Layers
    def model_layers(self, inputs, is_training):
        # CODE HERE
        pass
```