Realities of coding

When writing code for neural networks, often the bulk of attention will be on the model architecture -- because we want to find the optimum architecture for accuracy on our training set, and ensure that the accuracy also carries over to validation and testing.

This is good practice, but keep in mind that often we have many more lines of code that handle everything else in our system.

With MNIST, our code looked like this:

```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()
training_images  = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28,28)),
                                    tf.keras.layers.Dense(20, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(training_images, training_labels, epochs=20,
          validation_data=(val_images, val_labels))
```

Note that it only took 3 lines of code to load and prepare our data -- this is because the MNIST dataset was already available to us. This allowed us to import it in one line, and further normalize the images in two lines. As we build real systems, obtaining data may not always be this easy. It might require significant preprocessing to be able to feed it into a neural network for training.

As a result, often, we find that about 80% of our code deals with preparation, only 10% (or less) on our neural network architecture, leaving the other 10% for testing.