Realities of Coding Neural Networks

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

This is good practice, but do keep in mind that often you will have many more lines of code to handle everything else in your system (e.g., data collection and preparation).

With MNIST, your 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 your data -- this is because the MNIST dataset was already available to you, so you could get it with 1 line, and then it was just another 2 lines to normalize the images. As you build real systems, getting the data may not always be this easy -- and it might require significant coding to get it into a place where you can train a neural network with it. Going into the specifics of this process for every possible data source is beyond the scope of this course, but as you continue to learn neural networks, it's good to practice with different types of data, so you can prepare them for ingestion into your neural network architecture.

In this course we will explore three main types of data in the context of TinyML: audio, visual, and motion. As you will explore later, we often need to include pre and post processing steps to this data to get good results from our machine learning models.

Indeed, often you'll find about 80% of your code might be in the preparation, and only 10% (or less) in your neural network architecture, leaving the other 10% for testing and other tidy up.