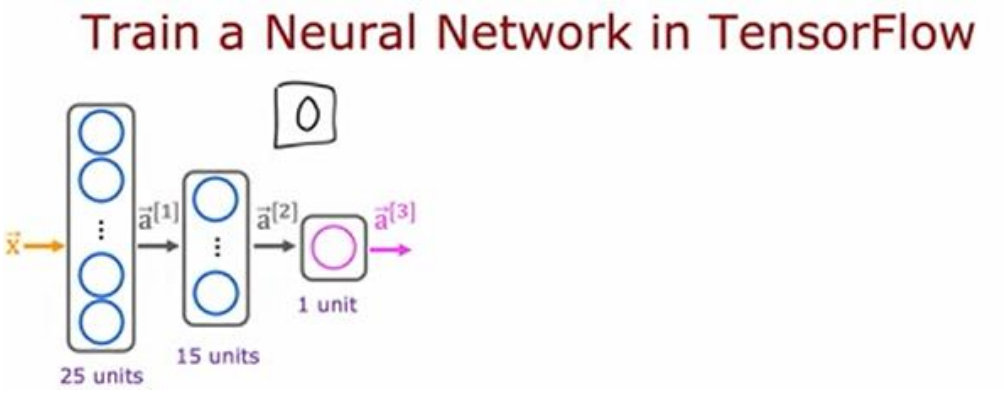
TensorFlow Implementation

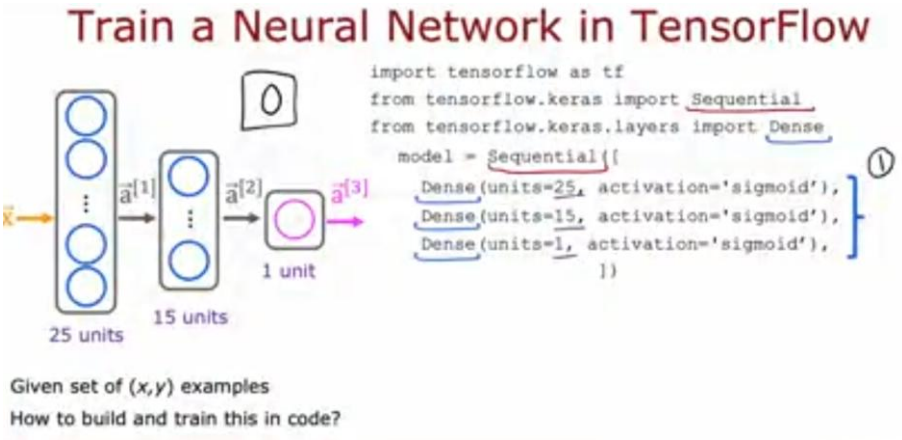
Welcome back to the second week of this course on advanced learning algorithms. Last week you learned how to carry out inference in the neural network. This week, we're going to go over training of a neural network. I think being able to take your own data and train your own neural network on it is really fun. This week, we'll look at how you could do that. Let's dive in. Let's continue with our running example of handwritten digit recognition recognizing this image as zero or a one.



Here we're using the neural network architecture that you saw last week, where you have an input X, that is the image, and then the first hidden layer was 25 units, second hidden layer with 15 units, and then one output unit. If you're given a training set of examples including images X, as was the ground truth label Y, how would you train the parameters of this neural network?

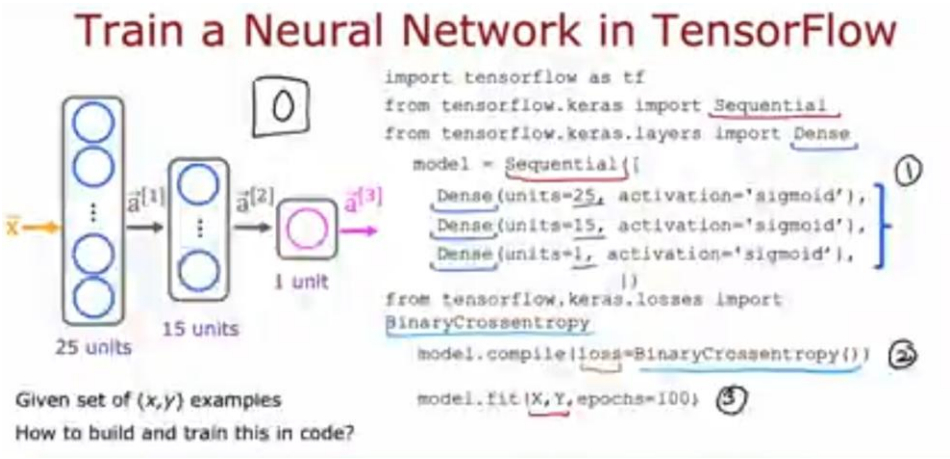
Let me go ahead and show you the code that you can use in TensorFlow to train this network. Then in the next few videos after this, we'll dive into details to explain what the code is actually doing. This is a code you write.

This first part may look familiar from the previous week where you are asking TensorFlow to sequentially string together these three layers of a neural network. The first hidden layer with 25 units and sigmoid activation, the second hidden layer, and then finally the output layer.



Nothing new here relating to what you saw last week. Second step is you're to ask TensorFlow to compile the model. The key step in asking TensorFlow to compile the model is to specify what is the loss function you want to use. In this case we'll use something that goes by the binary crossentropy loss function We'll see more in the next video what this really is.

Then having specified the loss function, the third step is to call the fit function, which tells TensorFlow to fit the model that you specified in step 1 using the loss of the cost function that you specified in step 2 to the dataset That's it. Step 1 is to specify the model, which tells TensorFlow how to compute for the inference



Step 2 compiles the model using a specific loss function, and step 3 is to train the model. That's how you can train a neural network in TensorFlow. As usual, I hope that you people are not just call these lines of code to train the model, but that you also understand what's actually going on behind these lines of code, so you don't just call it without really understanding what's going on.

I think this is important because when you're running a learning algorithm, if it doesn't work initially, having that conceptual mental framework of what's really going on will help you debug whenever things don't work the way you expect. With that, let's go on to the next video where we'll dive more deeply into what these steps in the TensorFlow implementation are actually doing.