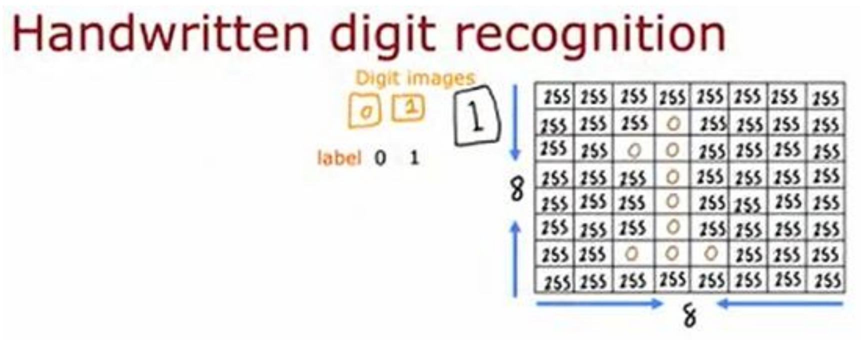
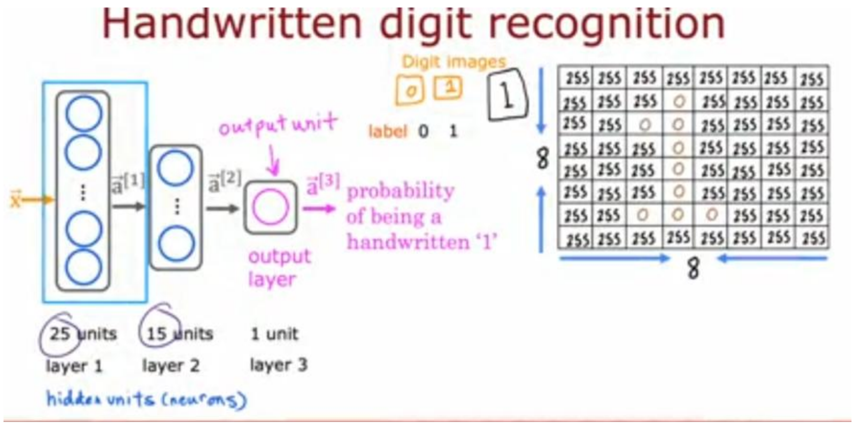
Inference: making predictions (forward propagation)

Let's take what we've learned and put it together into an algorithm to let your neural network make inferences or make predictions. This will be an algorithm called forward propagation. Let's take a look. I'm going to use as a motivating example, handwritten digit recognition. And for simplicity we are just going to distinguish between the handwritten digits zero and one. So it's just a binary classification problem where we're going to input an image and classify, is this the digit zero or the digit one? And you get to play with this yourself later this week in the practice lab as well.



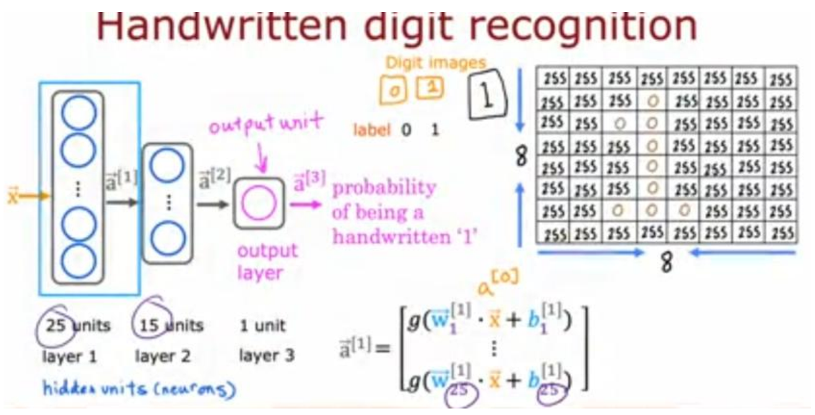
255	255	255	255	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	0	255	255	255
255	255	255	255	255	255	255	255

For the example of the slide, I'm going to use an eight by eight image. And so this image of a one is this grid or matrix of eight by eight or 64 pixel intensity values where 255 denotes a bright white pixel and zero would denote a black pixel. And different numbers are different shades of gray in between the shades of black and white. Given these 64 input features, we're going to use the neural network with two hidden layers. Where the first hidden layer has 25 neurons or 25 units. Second hidden layer has 15 neurons or 15 units. And then finally the output layer or outputs unit, what's the chance of this being 1 versus 0?.



So let's step through the sequence of computations that in your neural network will need to make to go from the input X, this eight by eight or 64 numbers to the predicted probability a3. The first computation is to go from X to a1, and that's what the first layer of the first hidden layer does. It carries out a computation of a super strip square bracket 1 equals this formula on the right.

Notice that a one has 25 numbers because this hidden layer has 25 units. Which is why the parameters go from w1 through w25 as well as b1 through b25. And I've written x here but I could also have written a0 here because by convention the activation of layer zero, that is a0 is equal to the input feature value x. So let's just compute a1.

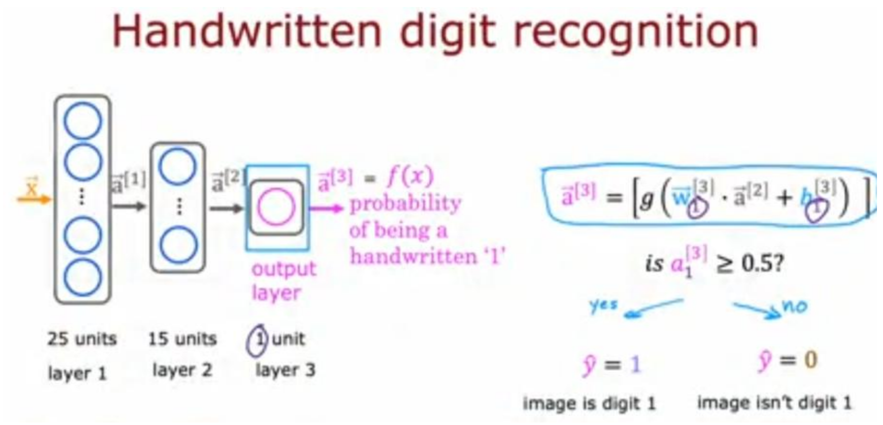


255	255	255	255	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	0	0	0	255	255	255
255	255	255	255	255	255	255	255

$$\vec{a}^{[1]} = \begin{bmatrix} g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}) \\ \vdots \\ g(\vec{w}_{25}^{[1]} \cdot \vec{x} + b_{25}^{[1]}) \end{bmatrix}$$

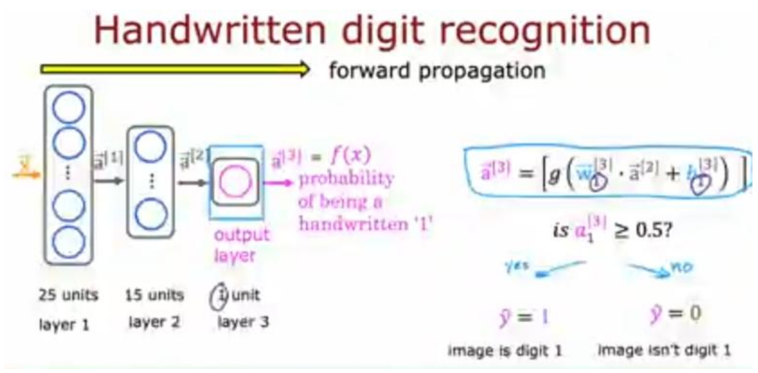
The next step is to compute a2. Looking at the second hidden layer, it then carries out this computation where a2 is a function of a1 and it's computed as the safe point activation function applied to w dot product a1 plus the corresponding value of b. Notice that layer two has 15 neurons or 15 units, which is why the parameters Here run from w1 through w15 and b1 through b15.

Now we've computed a2. The Final step is then to compute a3 and we do so using a very similar computation. Only now, this third layer, the output layer has just one unit, which is why there's just one output here. So a3 is just a scalar.



And finally you can optionally take a3 subscript one and threshold it at 4.5 to come up with a binary classification label. Is this the digit 1? Yes or no? So the sequence of computations first takes x and then computes a1, and then computes a2, and then computes a3, which is also the output of the neural networks. You can also write that as f(x).

So remember when we learned about linear regression and logistic regression, we use f(x) to denote the output of linear regression or logistic regression. So we can also use f(x) to denote the function computed by the neural network as a function of x.



Because this computation goes from left to right, you start from x and compute a1, then a2, then a3. This album is also called forward propagation because you're propagating the activations of the neurons. So you're making these computations in the forward direction from left to right. And this is in contrast to a different algorithm called backward propagation or back propagation, which is used for learning.

And that's something you learn about next week. And by the way, this type of neural network architecture where you have more hidden units initially and then the number of hidden units decreases as you get closer to the output layer. There's also a pretty typical choice when choosing neural network architectures.

And you see more examples of this in the practice lab as well. So that's neural network inference using the forward propagation algorithm. And with this, you'd be able to download the parameters of a neural network that someone else had trained and posted on the Internet. And you'd be able to carry out inference on your new data using their neural network. Now that you've seen the math and the algorithm, let's take a look at how you can actually implement this in tensorflow.