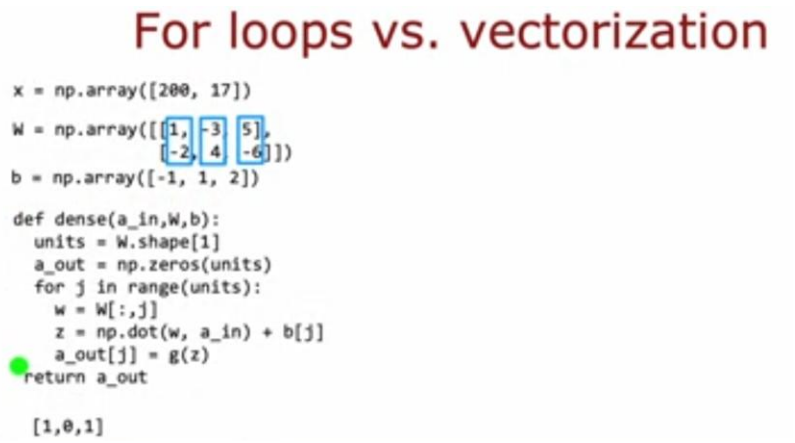
How are neural networks implemented effectively?

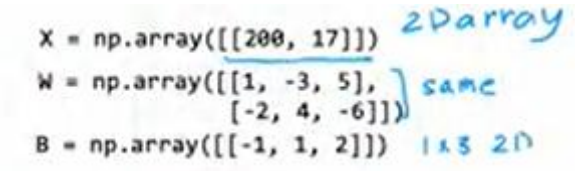
One of the reasons that deep learning researchers have been able to scale up neural networks, and thought really large neural networks over the last decade, is because neural networks can be vectorized. They can be implemented very efficiently using matrix multiplications.

It turns out that parallel computing hardware, including GPUs, but also some CPU functions are very good at doing very large matrix multiplications. In this video, we'll take a look at how these vectorized implementations of neural networks work. Without these ideas, I don't think deep learning would be anywhere near a success and scale today.

Here on the left is the code that you had seen previously of how you would implement forward prop, or forward propagation, in a single layer. X here is the input, W, the weights of the first, second, and third neurons, say, parameters B, and then this is the same code as which we saw before.



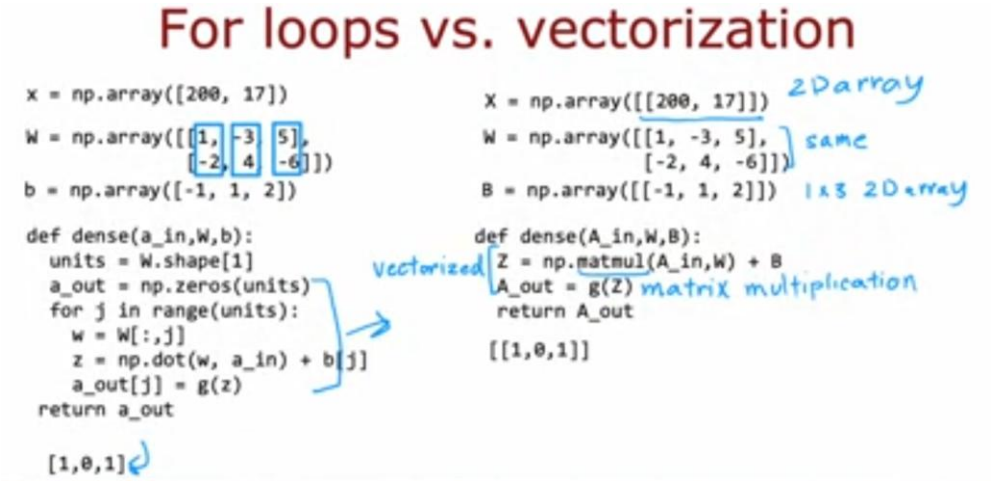
This will output three numbers, say, like that. If you actually implement this computation, you get 1, 0, 1. It turns out you can develop a vectorized implementation of this function as follows. Set X to be equal to this. Notice the double square brackets. This is now a 2D array, like in TensorFlow. W is the same as before, and B, I'm now using B, is also a one by three 2D array.



Then it turns out that all of these steps, this for loop inside, can be replaced with just a couple of lines of code, Z equals np.matmul. Matmul is how NumPy carries out matrix multiplication. Where now X and W are both matrices, and so you just multiply them together.

It turns out that this for loop, all of these lines of code can be replaced with just a couple of lines of code, which gives a vectorized implementation of this function. You compute Z, which is now a matrix again, as numpy.matmul between A in and W, where here A in and W are both matrices, and matmul is how NumPy carries out a matrix multiplication.

It multiplies two matrices together, and then adds the matrix B to it. Then A out is equal to the activation function g, that is the sigmoid function, applied element-wise to this matrix Z, and then you finally return A out.



This is what the code looks like. Notice that in the vectorized implementation, all of these quantities, x, which is fed into the value of A in as well as W, B, as well as Z and A out, all of these are now 2D arrays. All of these are matrices. This turns out to be a very efficient implementation of one step of forward propagation through a dense layer in the neural network. This is code for a vectorized implementation of forward prop in a neural network. But what is this code doing and how does it actually work? What is this matmul actually doing?