So now let's take a look at convolutions and pooling in code.

We don't have to do all the math for filtering and compressing,

we simply define convolutional and pooling layers to do the job for us.

So here's our code from the earlier example, where we defined out a neural

network to have an input layer in the shape of our data, and

output layer in the shape of the number of categories we're trying to define, and

a hidden layer in the middle.

The Flatten takes our square 28 by 28 images and

turns them into a one dimensional array.

To add convolutions to this, you use code like this.

You'll see that the last three lines are the same, the Flatten, the Dense hidden

layer with 128 neurons, and the Dense output layer with 10 neurons.

What's different is what has been added on top of this.

Let's take a look at this, line by line.

Here we're specifying the first convolution.

We're asking keras to generate 64 filters for us.

These filters are 3 by 3, their activation is relu, which means the negative values

will be thrown way, and finally the input shape is as before, the 28 by 28.

That extra 1 just means that we are tallying using a single byte for

color depth.

As we saw before our image is our gray scale, so we just use one byte.

Now, of course, you might wonder what the 64 filters are.

It's a little beyond the scope of this class to define them, but

they aren't random.

They start with a set of known good filters

in a similar way to the pattern fitting that you saw earlier, and

the ones that work from that set are learned over time.

For more details on convolutions and how they work,

there's a great set of resources here.

<https://www.youtube.com/watch?v=ArPaAX_PhIs&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF>



