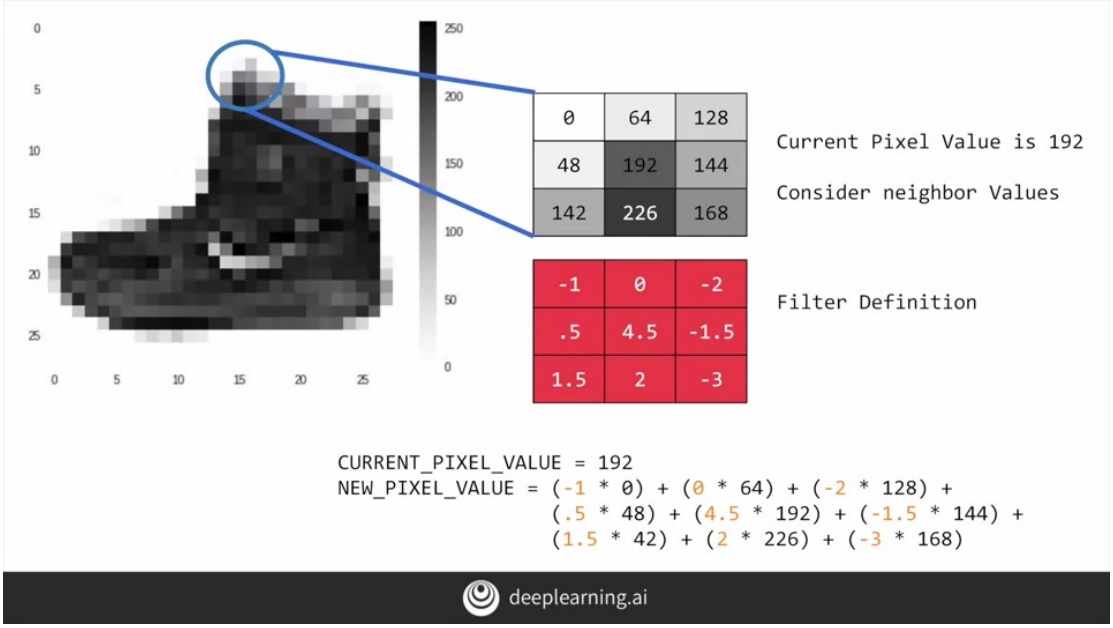
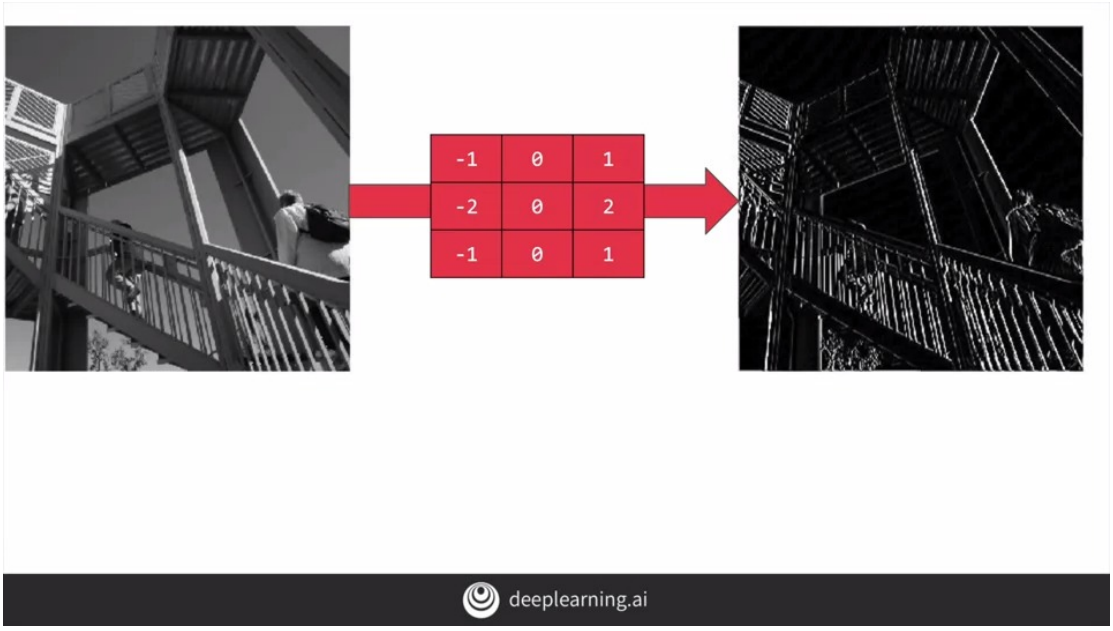
## 3.1 What are convolutions and pooling?

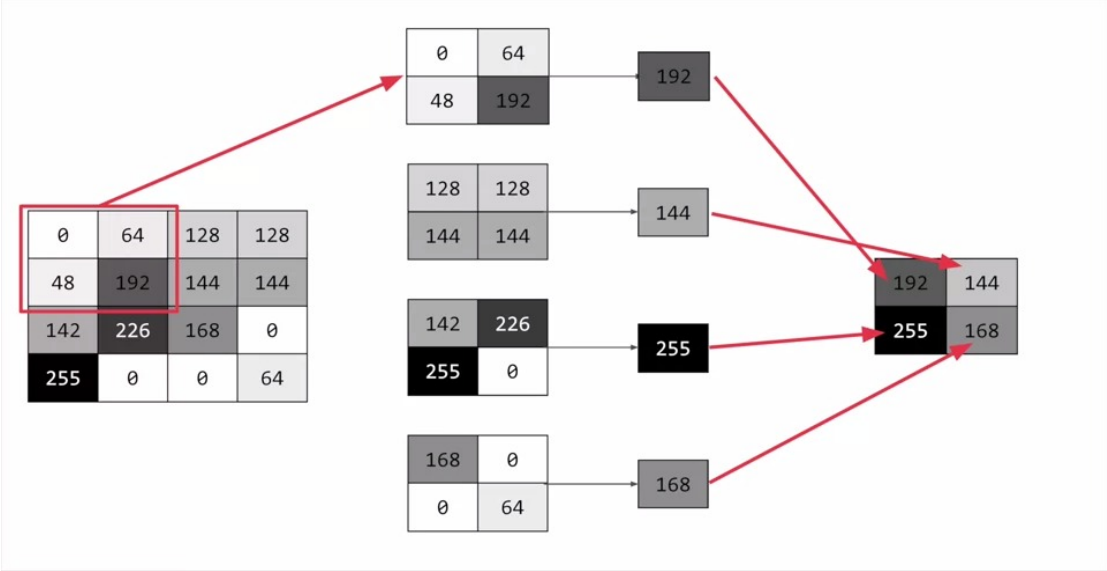
While there are only 784 pixels, it will be interesting to see if there was a way that we could condense the image down to the important features that distinguish what makes it a shoe, or a handbag, or a shirt. **That's where convolutions come in.**
For every pixel, take its value, and take a look at the value of its neighbors.




Repeat this for each neighbor and each corresponding filter value, and would then have the new pixel with the sum of each of the neighbor values multiplied by the corresponding filter value, and that's a convolution.




The idea here is that some convolutions will change the image in such a way that certain features in the image **get emphasized**.




Simply, pooling is a way of compressing an image. A quick and easy way to do this, is to go over the image of four pixels at a time, i.e, the current pixel and its neighbors underneath and to the right of it. Of these four, pick the biggest value and keep just that.

tf.keras.layers.Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

tf.keras.layers.MaxPool2D

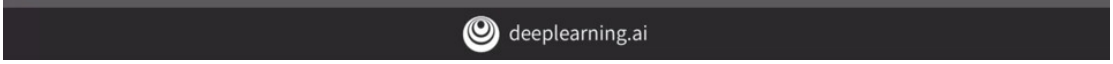
https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D

## 3.2 Implementing convolutional layers

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

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.

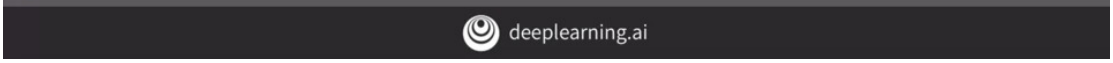
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), activation='relu',
                         input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```



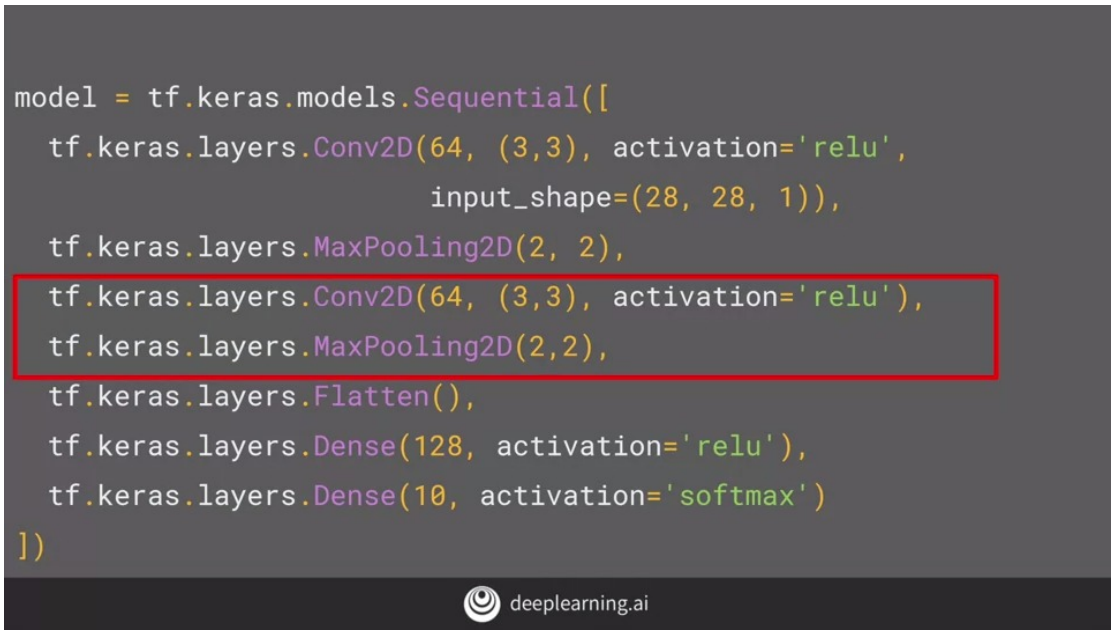

You'll see that the last three lines are the same, the **Flatten**, the **Dense hidden layer** with 128 neurons, and the **Dense output layer** with 10 neurons.
What's different is what has been added on top of this.
We're asking keras to generate **64 filters** for us. These filters are **3 by 3**, 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.
https://www.youtube.com/playlist?list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF

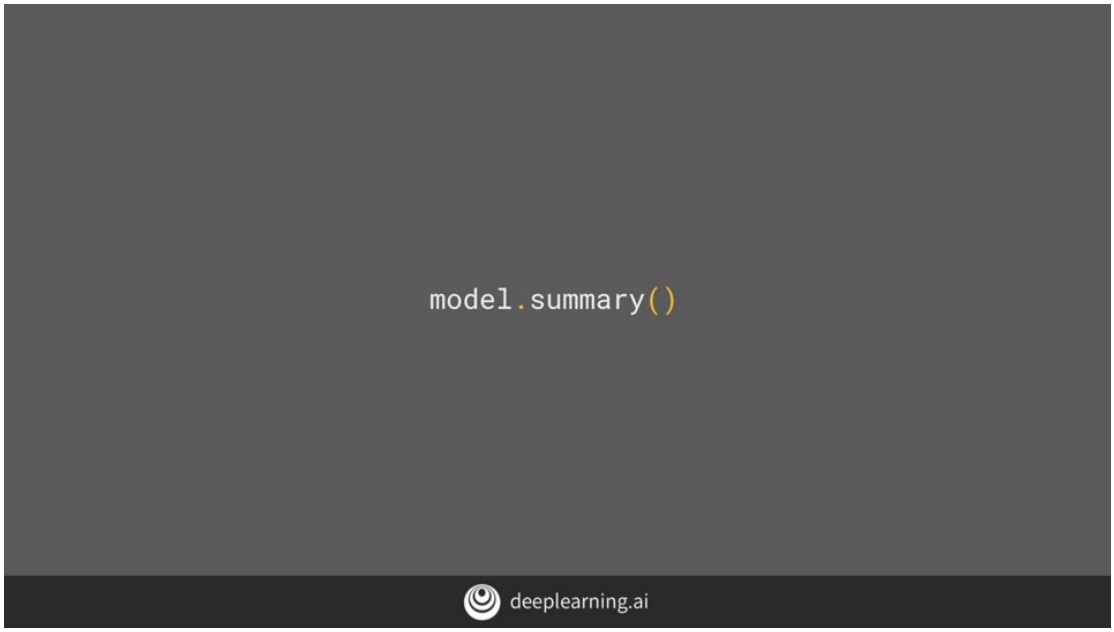
## 3.3 Implementing pooling layers

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), activation='relu',
                         input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```



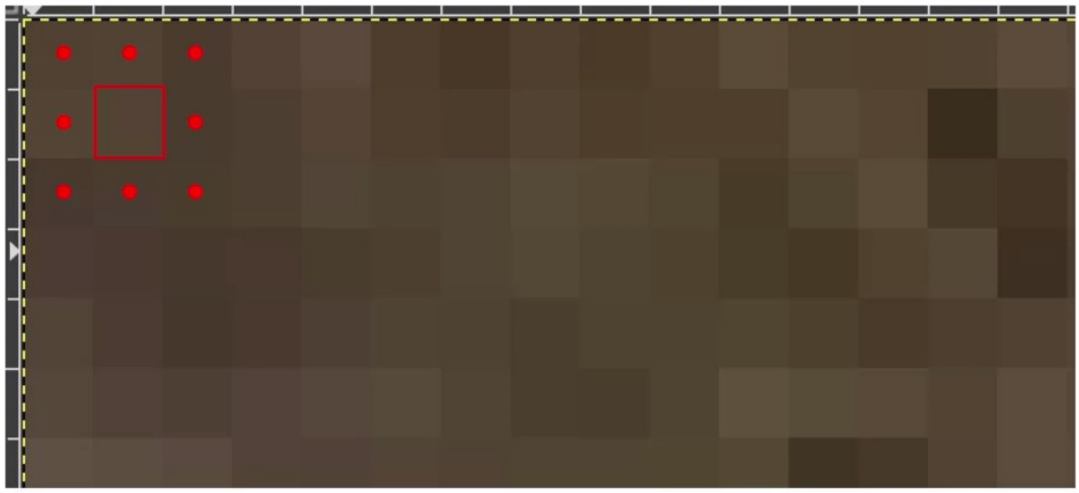
then create a pooling layer. It's max-pooling because we're going to take the maximum value. We're saying it's a two-by-two pool, so for every four pixels, the biggest one will survive as shown earlier.



| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_12 (Conv2D) | (None, 26, 26, 64) | 640 |
| max_pooling2d_12 (MaxPooling | (None, 13, 13, 64) | 0 |
| conv2d_13 (Conv2D) | (None, 11, 11, 64) | 36928 |
| max_pooling2d_13 (MaxPooling | (None, 5, 5, 64) | 0 |
| flatten_5 (Flatten) | (None, 1600) | 0 |
| dense_10 (Dense) | (None, 128) | 204928 |
| dense_11 (Dense) | (None, 10) | 1290 |

deeplearning.ai



deeplearning.ai

## 3.4 Improving the Fashion classifier with convolutions

Course 1 - Part 6 - Lesson 2 - Notebook.ipynb
By passing filters over an image to reduce the amount of information, they then allowed the neural network to effectively extract features that can distinguish one class of image from another.
You also saw how pooling compresses the information to make it more manageable.

For every image, 64 convolutions are being tried, and then the image is compressed and then another 64 convolutions, and then it's compressed again, and then it's passed through the DNN, and that's for 60,000 images that this is happening on each epoch.

The Keras API gives us each convolution and each pooling and each dense, etc. as a layer. So with the layers API, I can take a look at each layer's outputs, so I'll create a list of each layer's output. I can then treat each item in the layer as an individual activation model if I want to see the results of just that layer.

## 3.5 Walking through convolutions

Convolutions Sidebar.ipynb
Under the hood, TensorFlow is trying different filters on your image and learning which ones work when looking at the training data. As a result, when it works, you'll have greatly reduced information passing through the network, but because it isolates and identifies features, you can also get increased accuracy.

Lode's Computer Graphics Tutorial Image Filtering
https://lodev.org/cgtutor/filtering.html