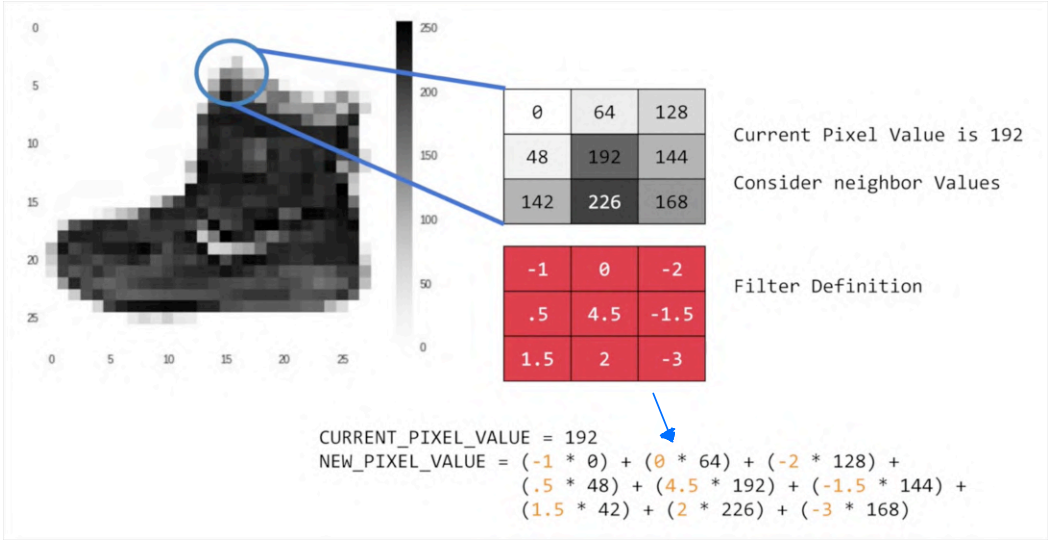
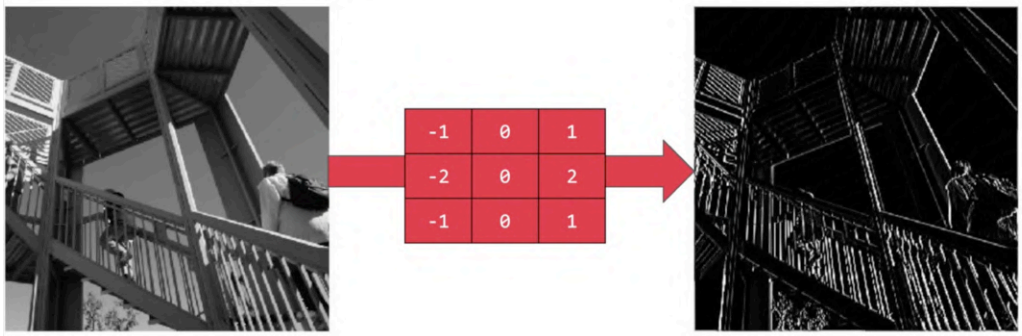
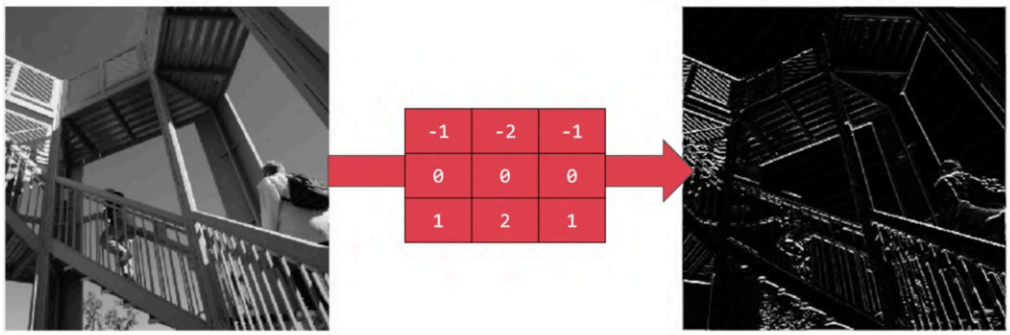
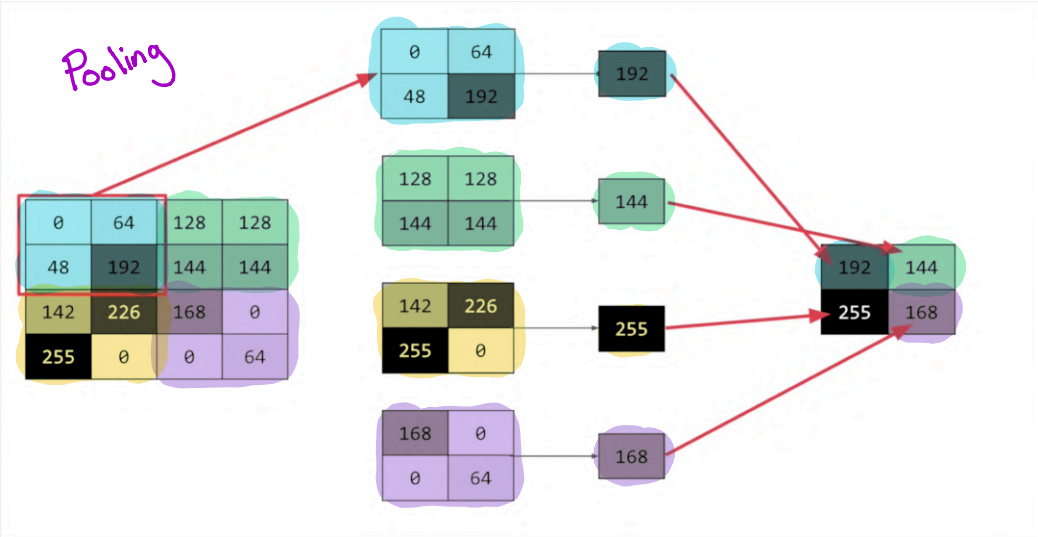

0 64 128
48 192 144
142 226 168
Current Pixel Value is 192
Consider neighbor Values
-1 0 -2
.5 4.5 -1.5
1.5 2 -3
Filter Definition
CURRENT_PIXEL_VALUE = 192
NEW_PIXEL_VALUE = (-1 * 0) + (0 * 64) + (-2 * 128) +
(.5 * 48) + (4.5 * 192) + (-1.5 * 144) +
(1.5 * 42) + (2 * 226) + (-3 * 168)


-1 0 1
-2 0 2
-1 0 1


-1 -2 -1
0 0 0
1 2 1



NN

```
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)
])
```

CNN

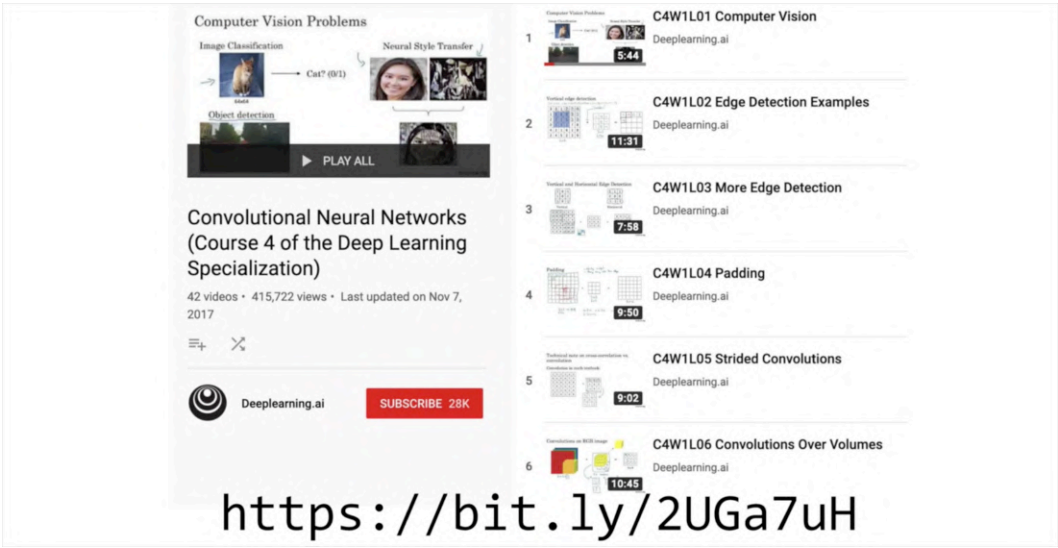
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), activation='relu',
                         input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),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

64 filters that are 3x3

```
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')
])
```

Grayscale: 1
Color: 3

We're saying it's a two-by-two pool, so for every four pixels, the biggest one will survive



https://bit.ly/2UGa7uH

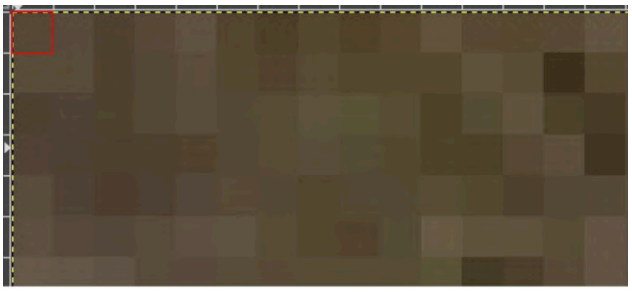
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), activation='relu',
                         input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),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

```
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')
])
```

```
model.summary()
```

```
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
=================================================================
```

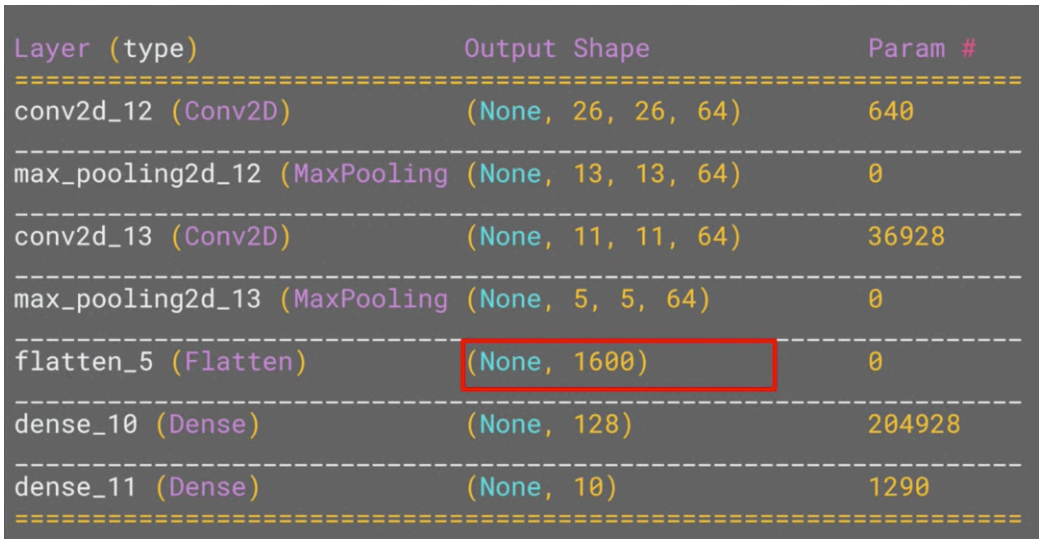
When you first look at this, it can be a little bit confusing and feel like a bug. After all, isn't the data 28 by 28, so y is the output, 26 by 26. The key to this is remembering that the filter is a three by three filter. Consider what happens when you start scanning through an image starting on the top left. So, for example with this image of the dog on the right, you can see zoomed into the pixels at its top left corner. You can't calculate the filter for the pixel in the top left, because it doesn't have any neighbors above it or to its left. In a similar fashion, the next pixel to the right won't work either because it doesn't have any neighbors above it. So, logically, the first pixel that you can do calculations on is this one, because this one of course has all eight neighbors that a three by three filter needs. This when you think about it, means that you can't use a one pixel margin all around the image, so the output of the convolution will be two pixels smaller on x, and two pixels smaller on y. If your filter is five-by-five for similar reasons, your output will be four smaller on x, and four smaller on y. So, that's y with a three by three filter, our output from the 28 by 28 image, is now 26 by 26, we've removed that one pixel on x and y, and each of the borders. So, next is the first of the max-pooling layers.



```
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
=================================================================
```

```
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
=================================================================
```

```
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
=================================================================
```



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

Remember we specified it to be two-by-two, thus turning four pixels into one, and having our x and y. So, now our output gets reduced from 26 by 26, to 13 by 13. The convolutions will then operate on that, and of course, we lose the one pixel margin as before, so we're down to 11 by 11, add another two-by-two max-pooling to have this rounding down, and went down, down to five-by-five images. So, now our dense neural network is the same as before, but it's being fed with five-by-five images instead of 28 by 28 ones. But remember, it's not just one compress five-by-five image instead of the original 28 by 28, there are a number of convolutions per image that we specified, in this case 64. So, there are 64 new images of five-by-five that had been fed in. Flatten that out and you have 25 pixels times 64, which is 1600. So, you can see that the new flattened layer has 1,600 elements in it, as opposed to the 784 that you had previously. This number is impacted by the parameters that you set when defining the convolutional 2D layers.