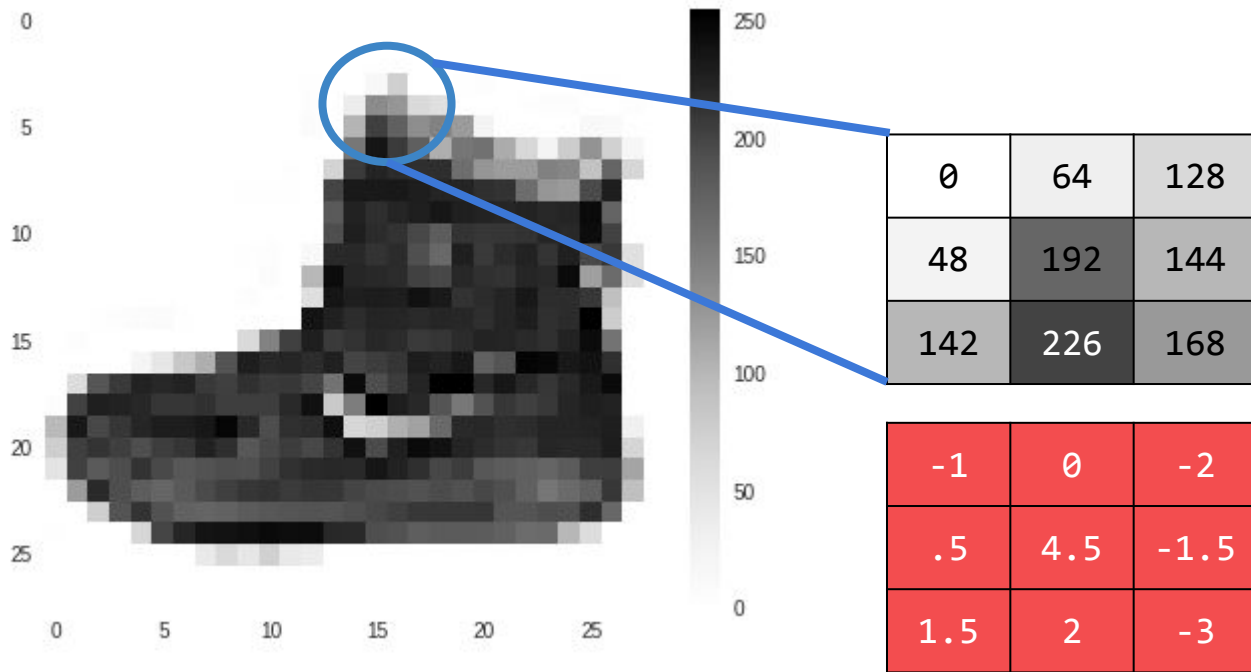


Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



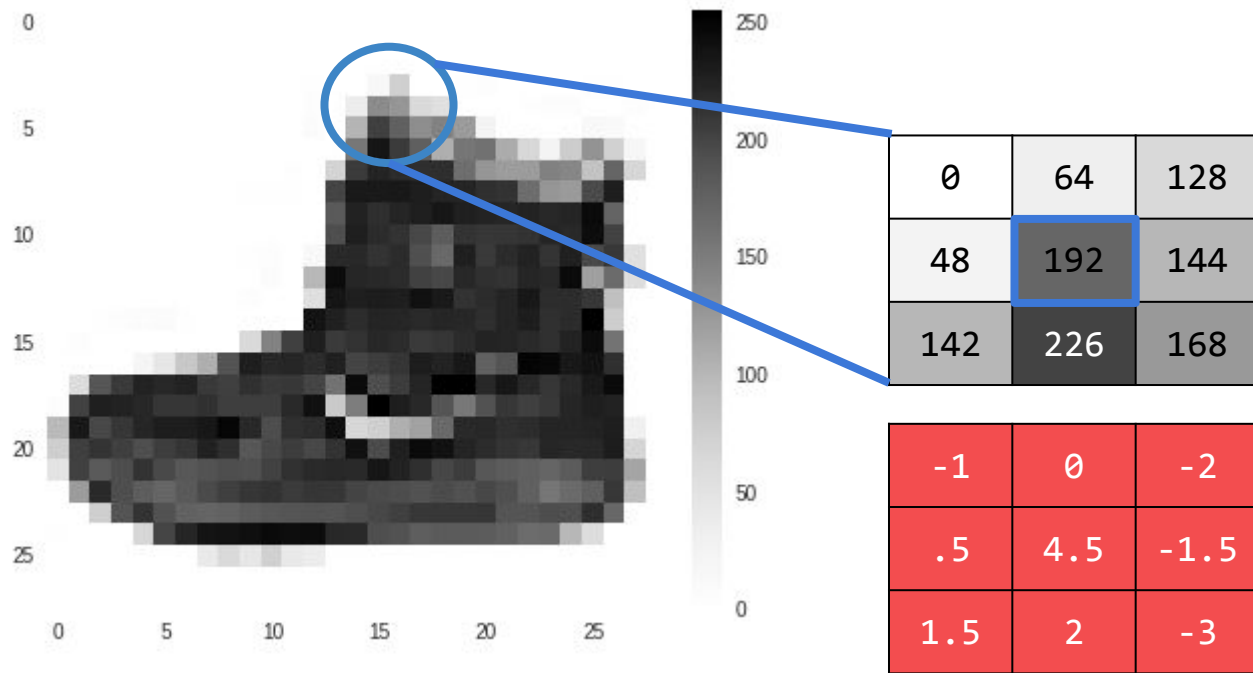
Current Pixel Value is 192

Consider neighbor Values

Filter Definition

CURRENT_PIXEL_VALUE = 192

NEW_PIXEL_VALUE = $(-1 * 0) + (0 * 64) + (-2 * 128) +$
 $(.5 * 48) + (4.5 * 192) + (-1.5 * 144) +$
 $(1.5 * 142) + (2 * 226) + (-3 * 168)$



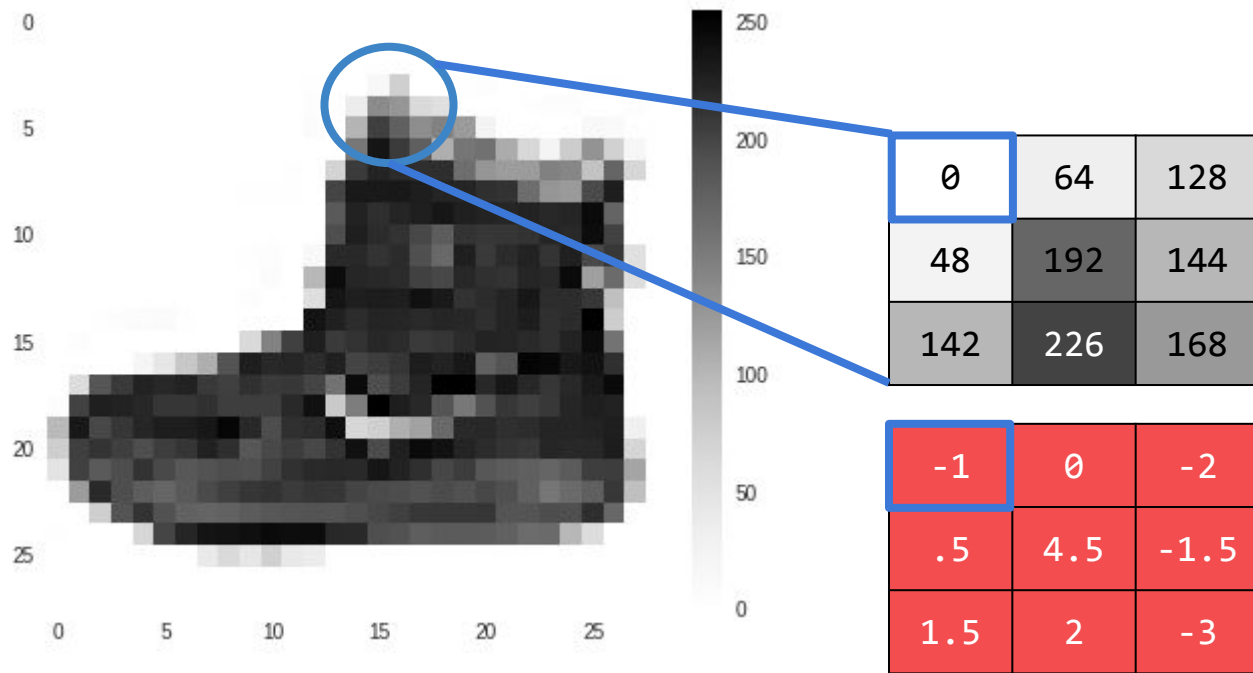
Current Pixel Value is 192

Consider neighbor Values

Filter Definition

CURRENT_PIXEL_VALUE = 192

NEW_PIXEL_VALUE = $(-1 * 0) + (0 * 64) + (-2 * 128) +$
 $(.5 * 48) + (4.5 * 192) + (-1.5 * 144) +$
 $(1.5 * 142) + (2 * 226) + (-3 * 168)$



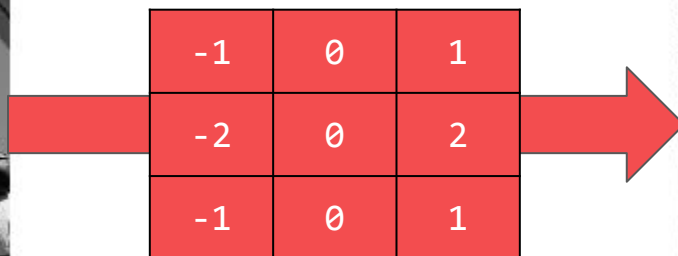
Current Pixel Value is 192

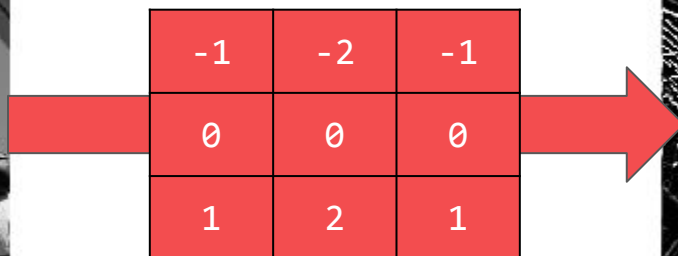
Consider neighbor Values

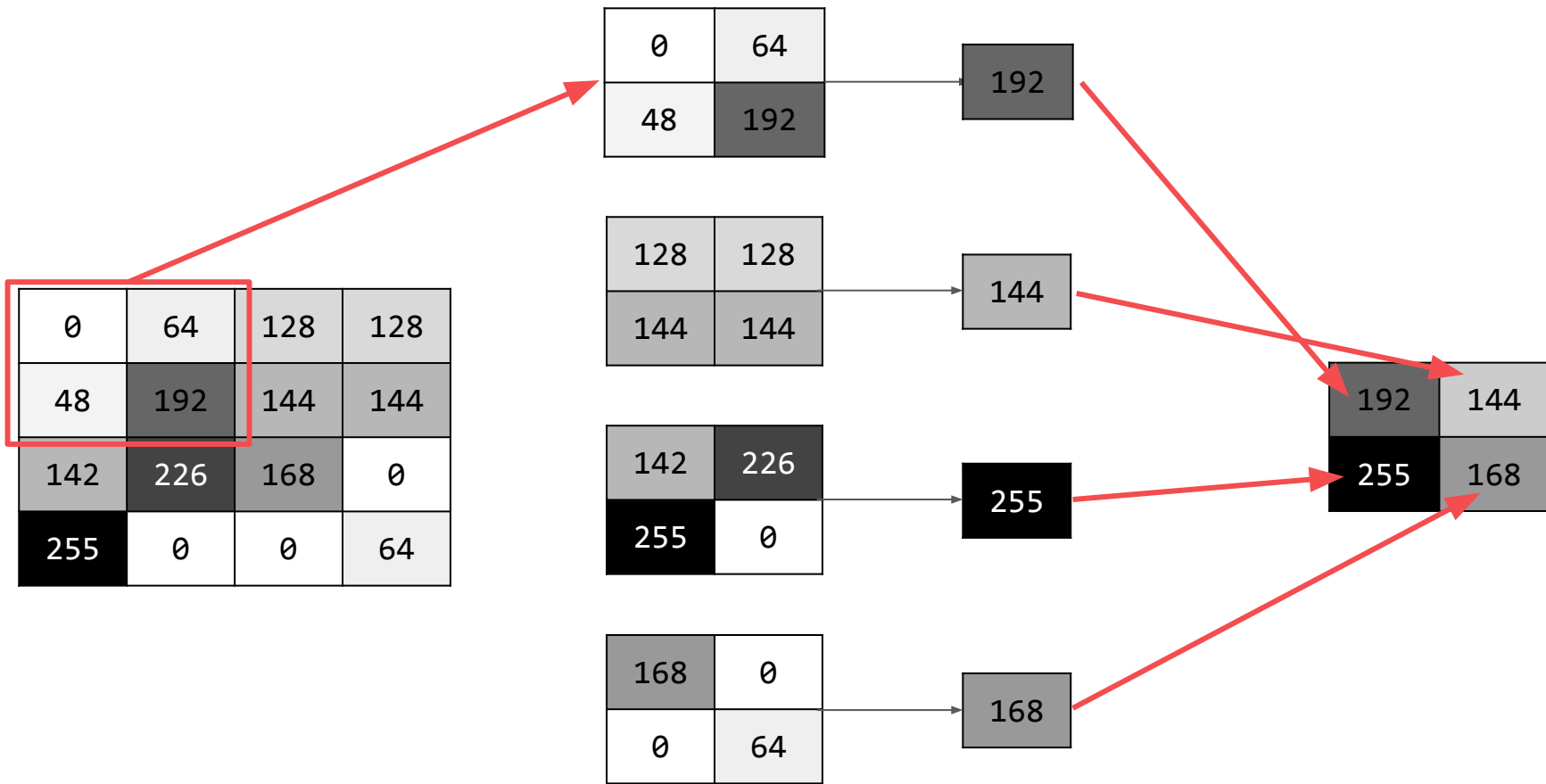
Filter Definition

CURRENT_PIXEL_VALUE = 192

$$\begin{aligned} \text{NEW_PIXEL_VALUE} = & (-1 * 0) + (0 * 64) + (-2 * 128) + \\ & (.5 * 48) + (4.5 * 192) + (-1.5 * 144) + \\ & (1.5 * 142) + (2 * 226) + (-3 * 168) \end{aligned}$$







```
model = tf.keras.Sequential([  
    tf.keras.Input(shape=(28, 28)),  
    tf.keras.layers.Flatten(),  
    tf.keras.layers.Dense(128, activation=tf.nn.relu),  
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)  
])
```




```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(28, 28, 1)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    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.Sequential([
    tf.keras.Input(shape=(28, 28, 1)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    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')
])
```

Computer Vision Problems

Image Classification



Cat? (0/1)

Neural Style Transfer



Object detection



PLAY ALL

Convolutional Neural Networks (Course 4 of the Deep Learning Specialization)

42 videos • 415,722 views • Last updated on Nov 7, 2017



DeepLearning.AI

SUBSCRIBE 28K

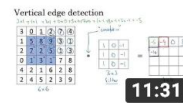
1



C4W1L01 Computer Vision

Deeplearning.ai

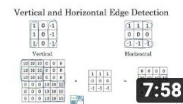
2



C4W1L02 Edge Detection Examples

Deeplearning.ai

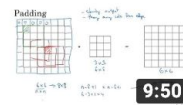
3



C4W1L03 More Edge Detection

Deeplearning.ai

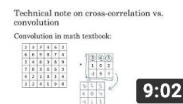
4



C4W1L04 Padding

Deeplearning.ai

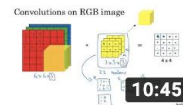
5



C4W1L05 Strided Convolutions

Deeplearning.ai

6



C4W1L06 Convolutions Over Volumes

Deeplearning.ai

<https://bit.ly/2UGa7uH>



deeplearning.ai

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(28, 28, 1)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    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.Sequential([
    tf.keras.Input(shape=(28, 28, 1)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    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 |

| 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 |

| 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 |