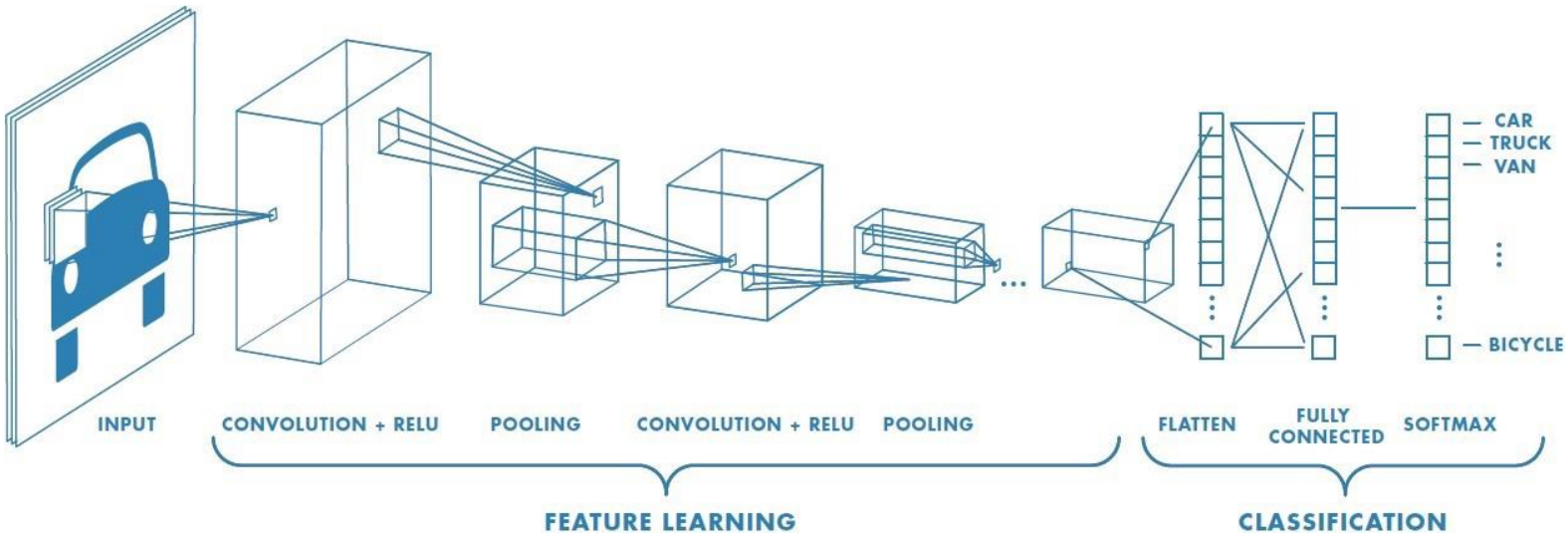
- **Convolutional Neural Networks (convnets,CNN)**
  - **2D convolution –Conv2D**
  - **Padding**
  - **Stride**
  - **Dilation**
  - **Pooling**
  - **Flatten**

Convolutional Neural Networks



- In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.
- The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolutional networks are a specialized type of neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
- They have applications in image and video recognition, recommender systems, image classification, Image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

Kaynak:
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
https://en.wikipedia.org/wiki/Convolutional_neural_network

2D convolution

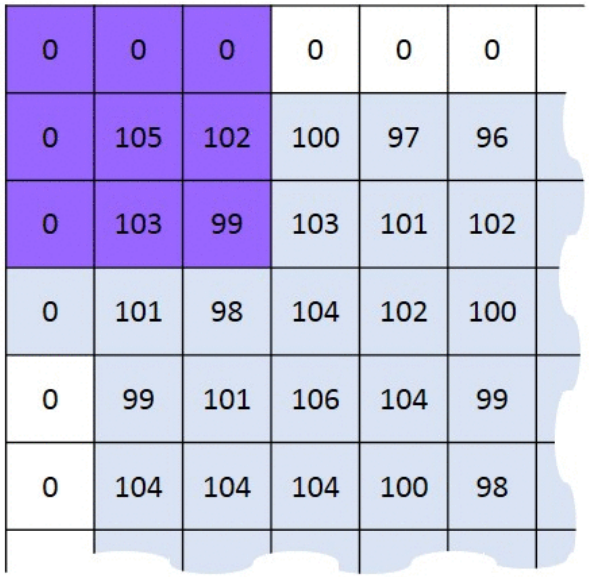
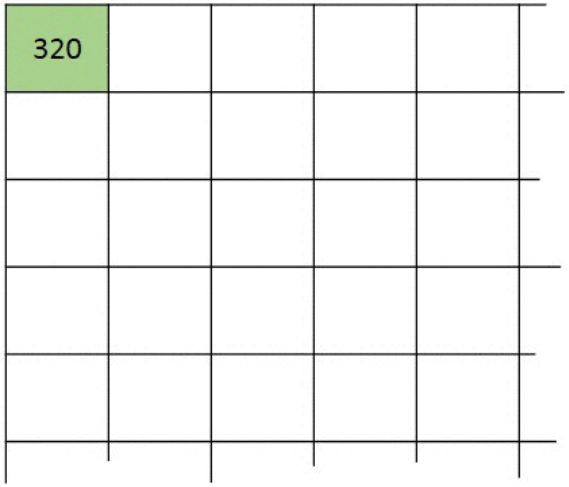


Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

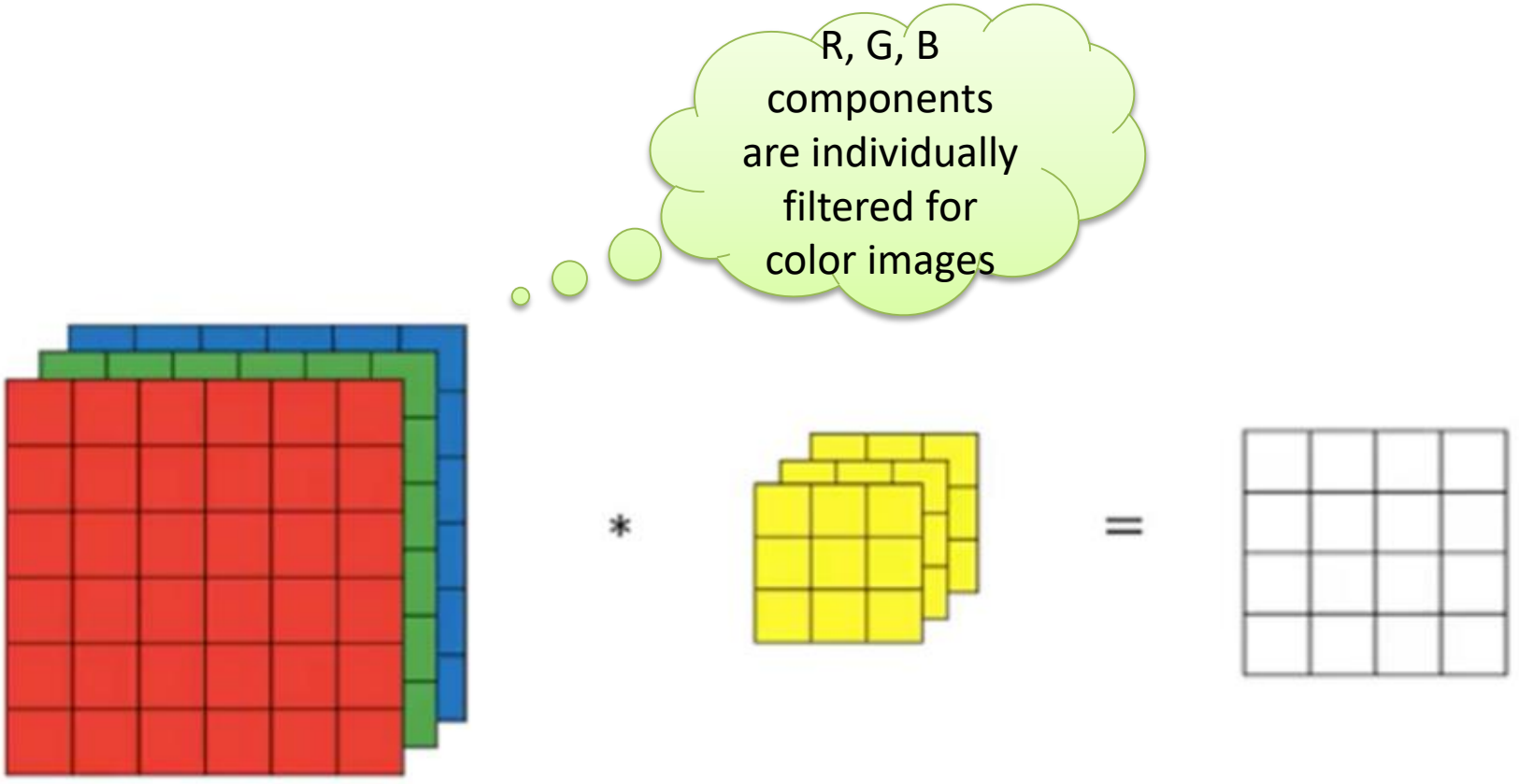
$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 = 320$$



Output Matrix

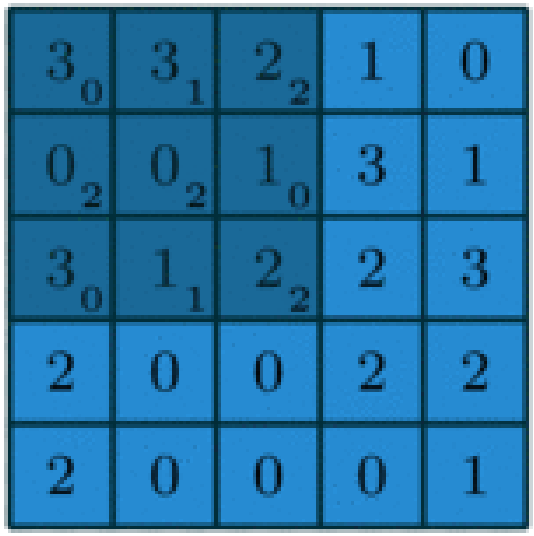
Convolution with horizontal and vertical strides = 1

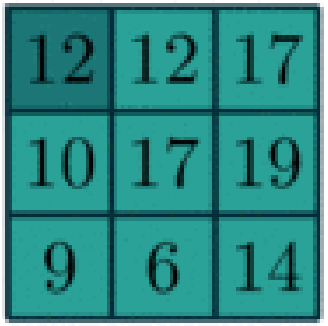
2D convolution



Kaynak: https://medium.com/coinmonks/an-introduction-to-cnns-and-a-step-by-step-model-of-a-digit-recognizer-using-mnist-database-in-f4ea6af06d77

padding





- Computations with the kernel to the edges can be ignored.
- A WxH sized image is transformed into a (W-2) x (H-2) dimensional image as a result of the convolution with a 3x3 kernel.
- Similarly, for a 5x5 kernel, the image becomes a (W-4) x (H-4) dimensional image.

Kernel: $\begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$

Keras: padding=valid

padding

Edge padding can be done to keep the size of the output image equal to input image size.

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

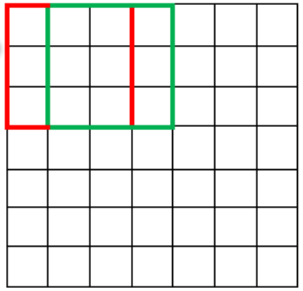
Keras: padding= same

Kaynak: https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/

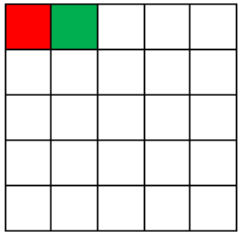
Stride (adım büyüklüğü)

Convolution process is generally performed in 1 pixel steps.

7 x 7 Input Volume



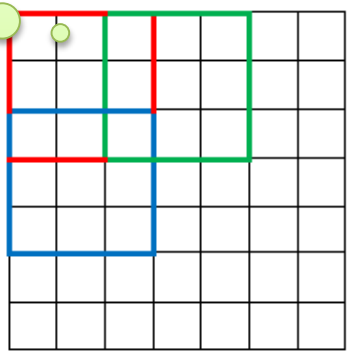
5 x 5 Output Volume



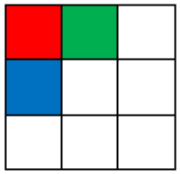
When convolution is performed in 2 steps intervals, the output image is reduced by about half.

However, larger steps can be used to reduce the size of output image.

7 x 7 Input Volume

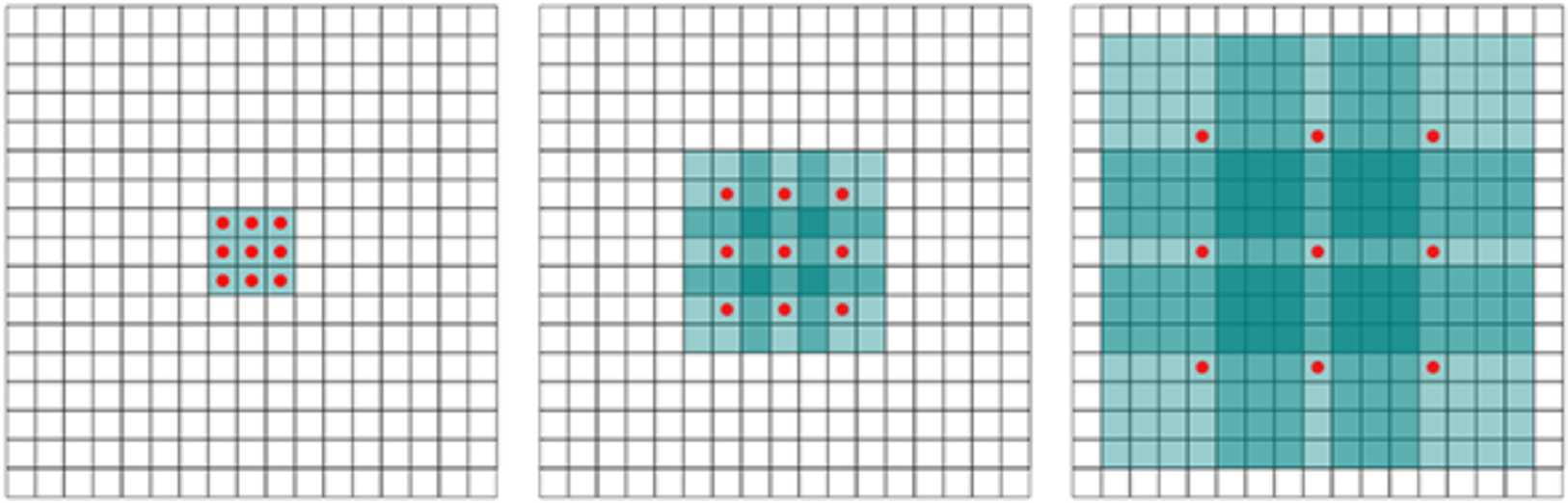


3 x 3 Output Volume



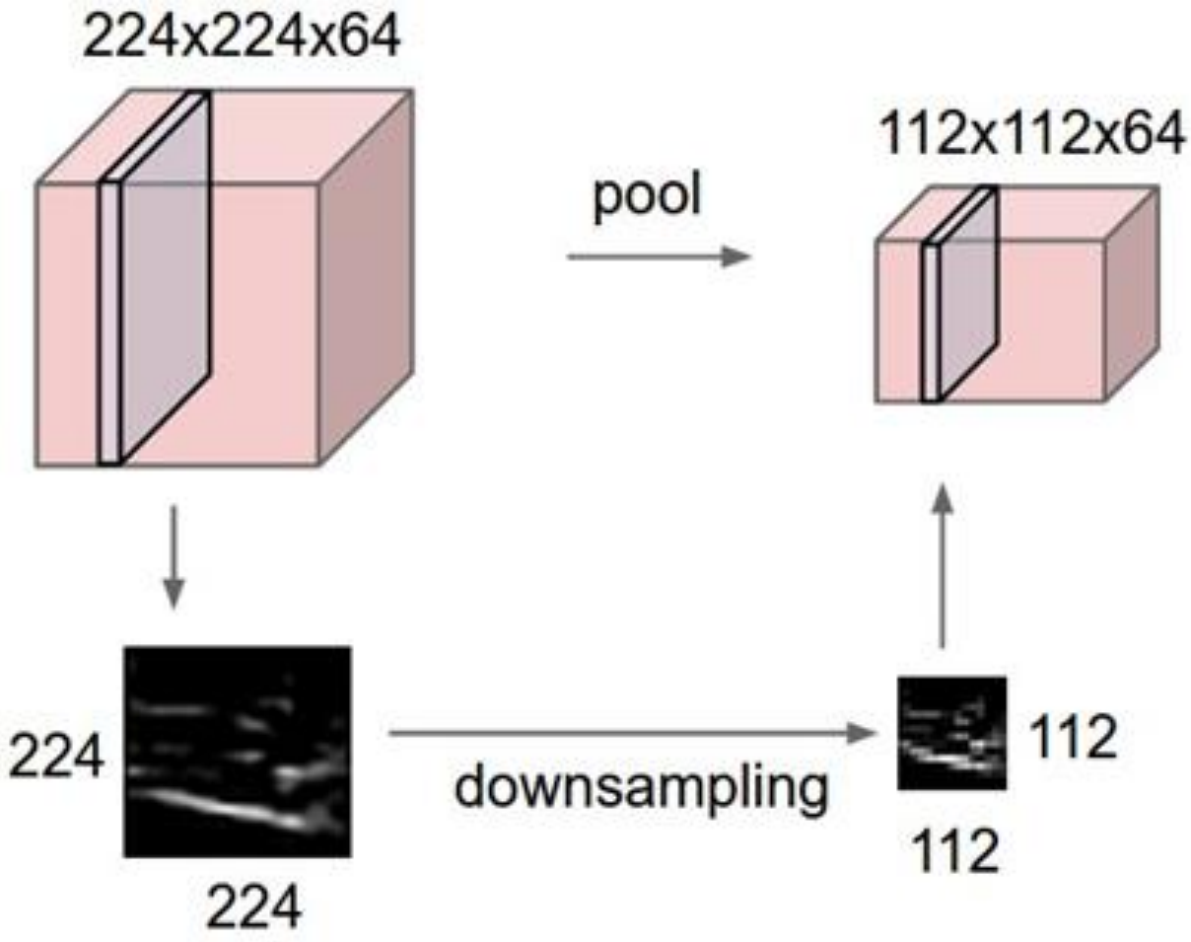
Dilation rate (Genişleme oranı)

- The dilation_rate parameter of the Conv2D class is a 2-digit integer that controls the dilation rate for dilated convolution.
- Dilated convolution is the process applied to the input image only with defined spaces, as shown in the figure below.



Kaynak: https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/

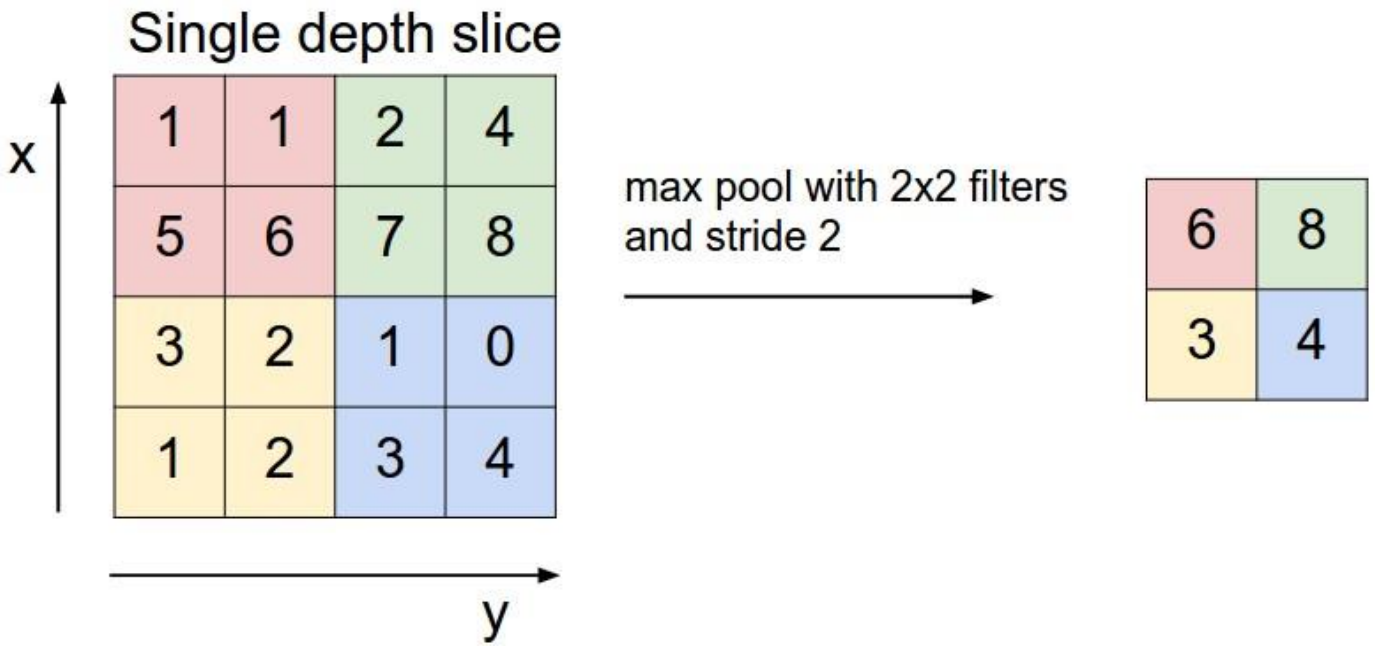
Pooling (birleştirme)



Kaynak: http://cs231n.github.io/convolutional-networks/

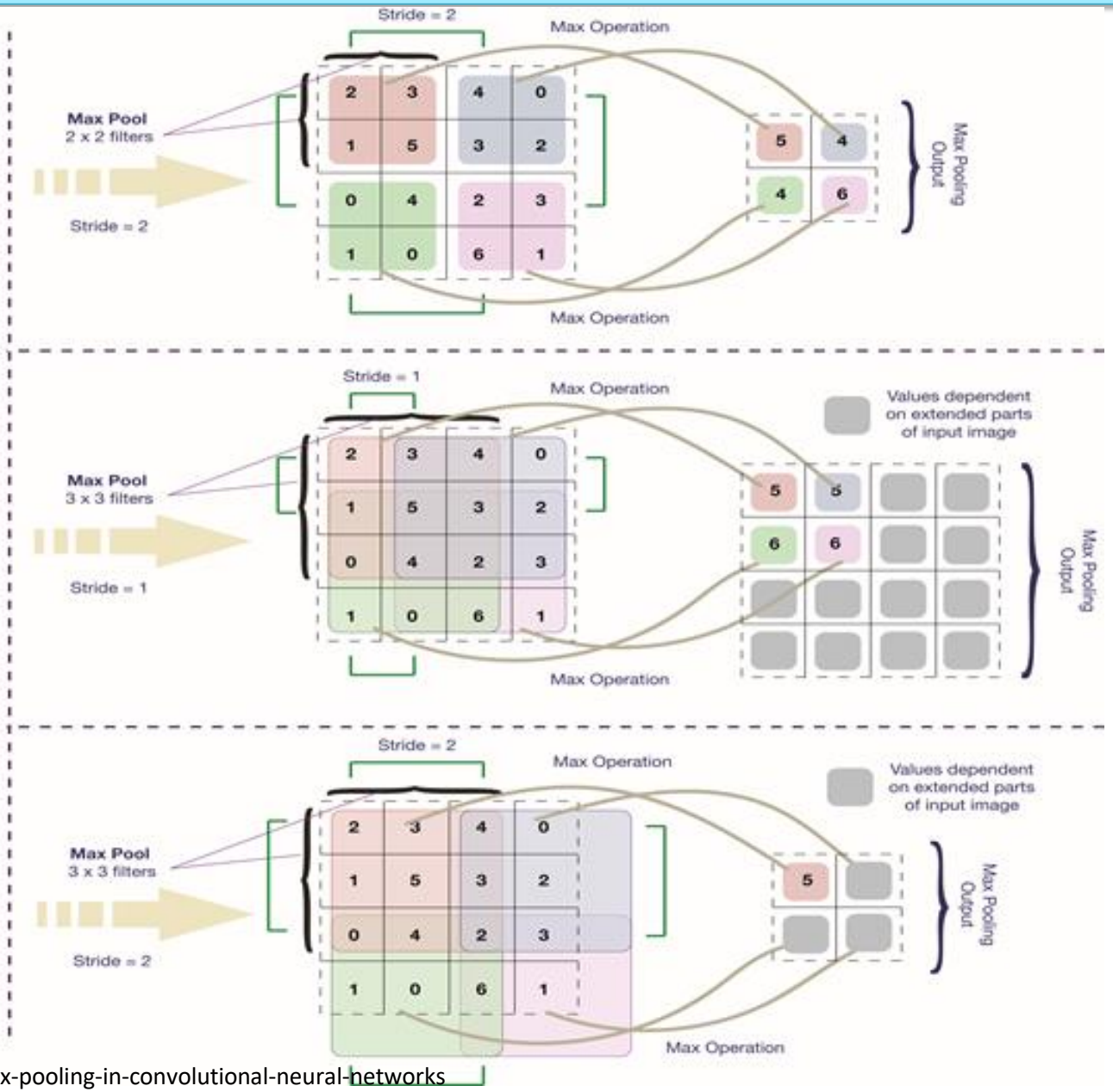
Pooling (birleştirme)

- Max Pooling



Kaynak:http://www.jessicayung.com/explaining-tensorflow-code-for-a-convolutional-neural-network/

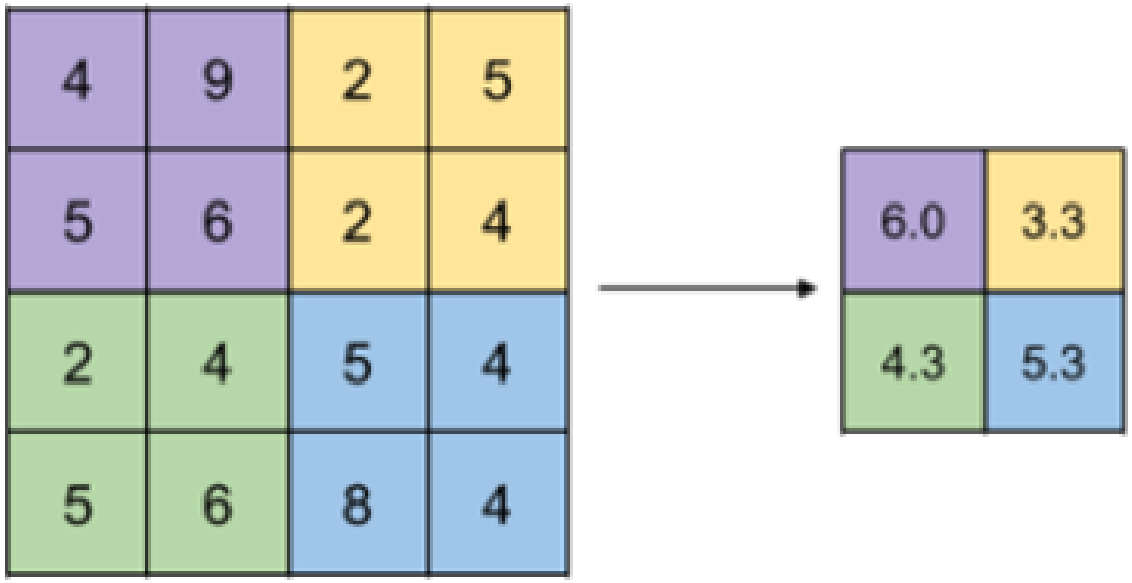
Pooling (birleştirme)



https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks

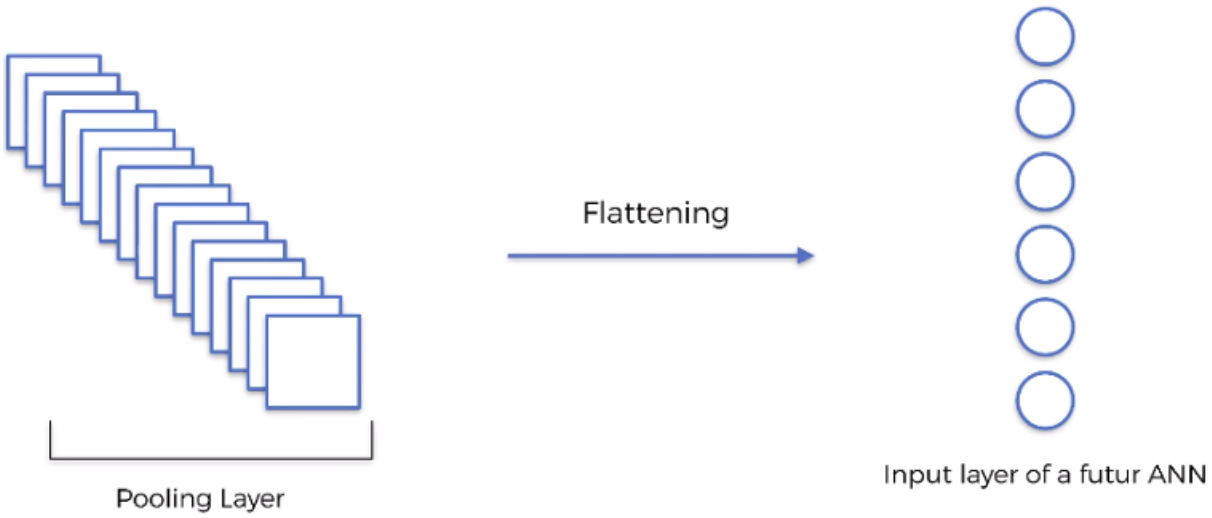
Pooling (birleştirme)

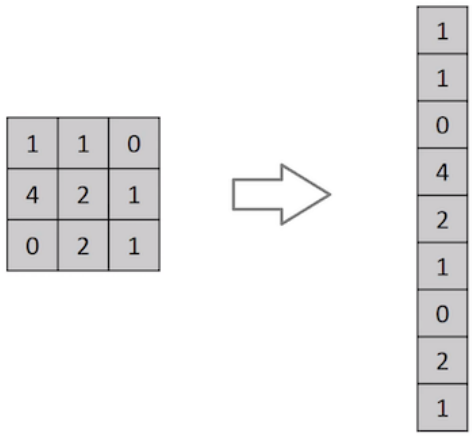
- Average Pooling



Kaynak: https://medium.com/coinmonks/an-introduction-to-cnns-and-a-step-by-step-model-of-a-digit-recognizer-using-mnist-database-in-f4ea6af06d77

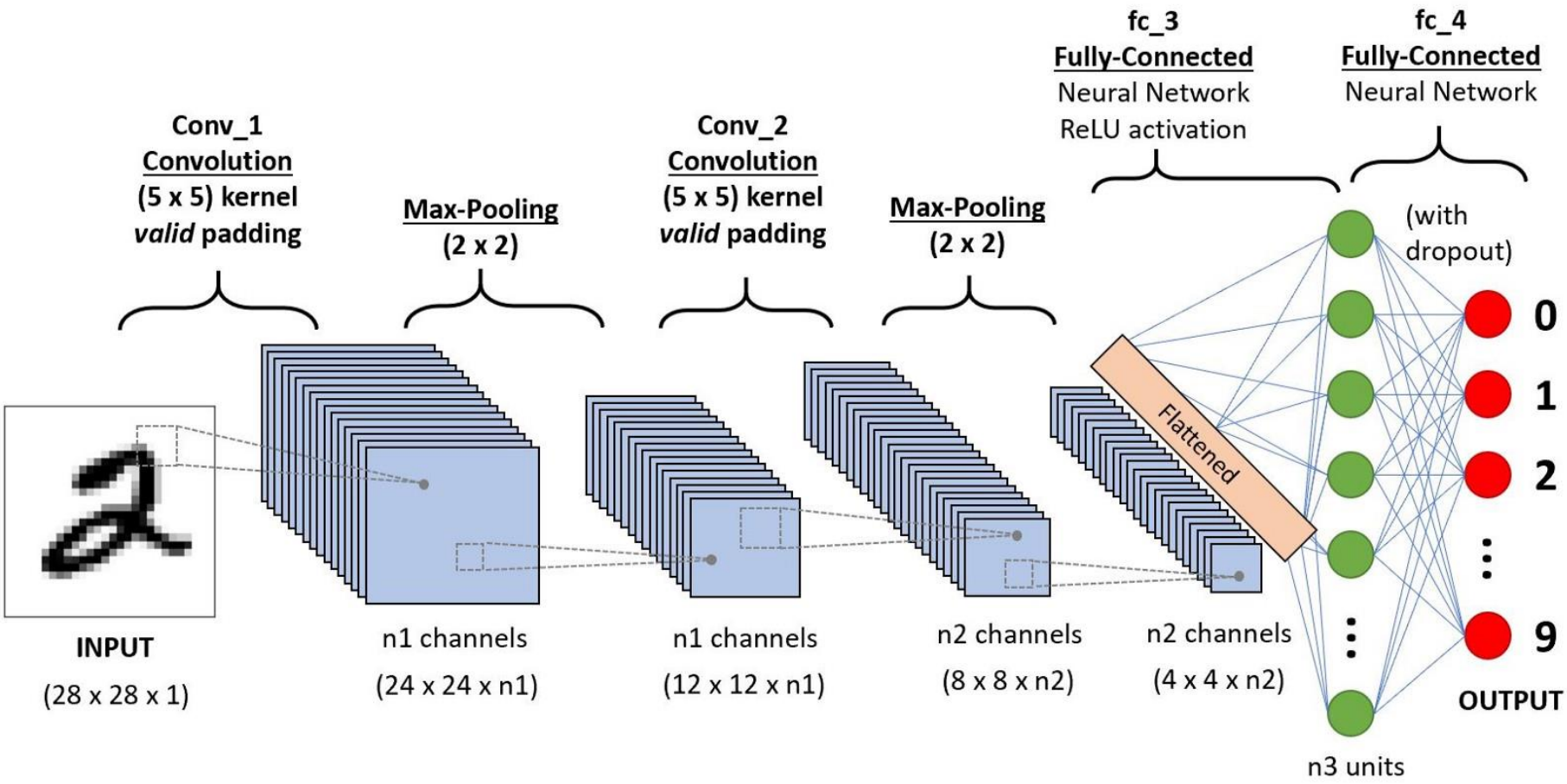
Flatten (düzleştirme)





Kaynak: https://medium.com/bbm406f18/week-3-predicting-the-pieces-of-furniture-in-living-rooms-af1de0006a56

Example: mnist dataset



Example: mnist dataset

```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32,
                        (3, 3),
                        activation='relu',
                        input_shape=(28,28, 1)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64,
                        (3, 3),
                        activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64,
                        (3, 3),
                        activation='relu'))


model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(10, activation='softmax'))


model.summary()
```

Example: mnist dataset

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_36 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_4 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_8 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

Example: mnist dataset

Kodun devamı:

```
39 from keras.datasets import mnist
40 from keras.utils import to_categorical
41
42 (train_images, train_labels), (test_images, test_labels) =\
43  mnist.load_data()
44
45 train_images = train_images.reshape((60000, 28, 28, 1))
46 train_images = train_images.astype('float32') / 255
47 test_images = test_images.reshape((10000, 28, 28, 1))
48 test_images = test_images.astype('float32') / 255
49 train_labels = to_categorical(train_labels)
50 test_labels = to_categorical(test_labels)
51
52 model.compile(optimizer='rmsprop',
53               loss='categorical_crossentropy',
54               metrics=['accuracy'])
55
56 model.fit(train_images,
57           train_labels,
58           epochs=5,
59           batch_size=64)
60
61 test_loss, test_acc = model.evaluate(test_images, test_labels)
62 print("test_acc=",test_acc)
```

Örnek: mnist veri seti

```
_________________________________________________________________
Epoch 1/5
60000/60000 [==============================] - 45s 747us/step - loss: 0.1628 - acc: 0.9492
Epoch 2/5
60000/60000 [==============================] - 45s 751us/step - loss: 0.0479 - acc: 0.9851
Epoch 3/5
60000/60000 [==============================] - 55s 911us/step - loss: 0.0330 - acc: 0.9896
Epoch 4/5
60000/60000 [==============================] - 52s 873us/step - loss: 0.0249 - acc: 0.9923
Epoch 5/5
60000/60000 [==============================] - 48s 808us/step - loss: 0.0197 - acc: 0.9942
10000/10000 [==============================] - 3s 293us/step
test_acc= 0.9915
```

Example: Cifar10 dataset

- **from** keras.datasets **import** cifar10
- (x_train, y_train), (x_test, y_test) = cifar10.load_data()