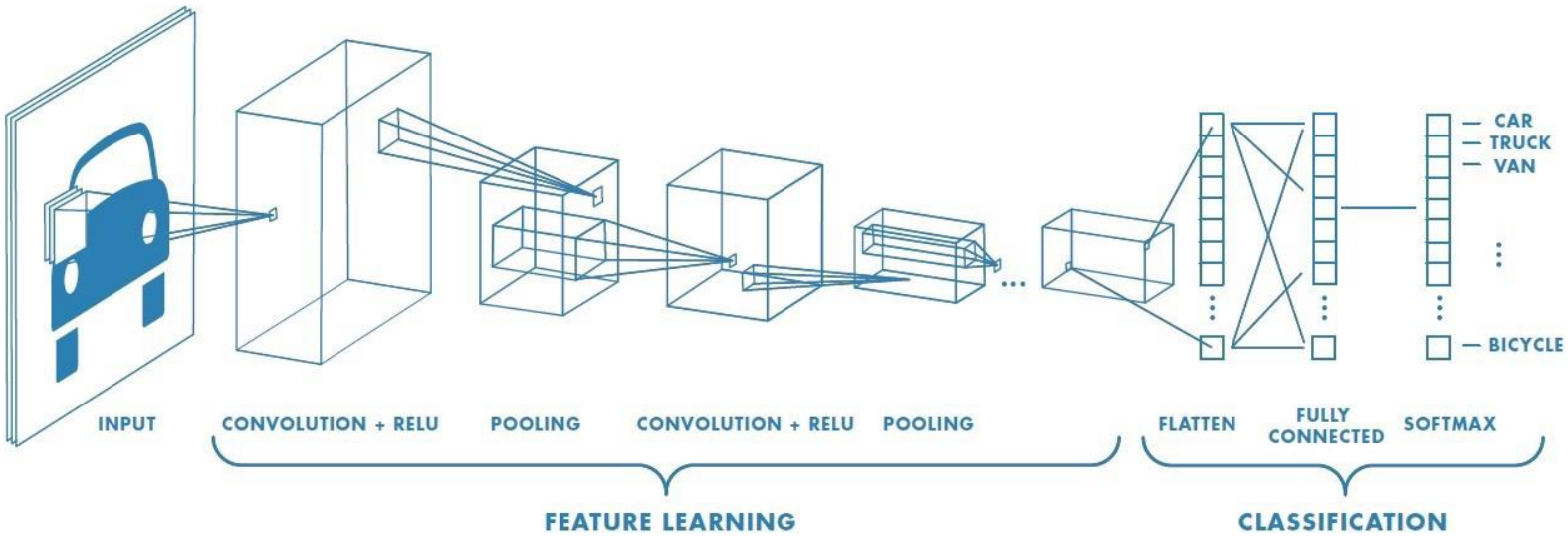
- **Convolutional Neural Networks (convnets,CNN)**
  - **2D convolution**
  - **Padding**
  - **Stride**
  - **Dilation**
  - **Pooling**
  - **Flatten**

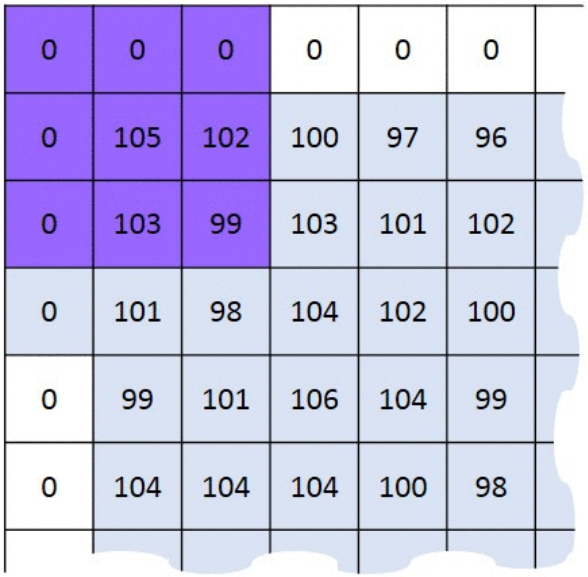
Convolutional Neural Networks



- Bir Evrişimsel Sinir Ağı (ConvNet / Convolutional neural networks -CNN), bir girdi görüntüsünü alıp, görüntüdeki çeşitli görünüşleri/nesneleri birbirinden ayırabilen Derin Öğrenme algoritmasıdır.
- Evrişimli sinir ağları, temel olarak görüntüleri sınıflandırmak (örneğin gördüklerini isimlendirmek), benzerlikle kümelemek (fotoğraf arama) ve sahnelerde nesne tanıma yapmak için kullanılan derin yapay sinir ağlarıdır.

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

2D convolution



0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

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

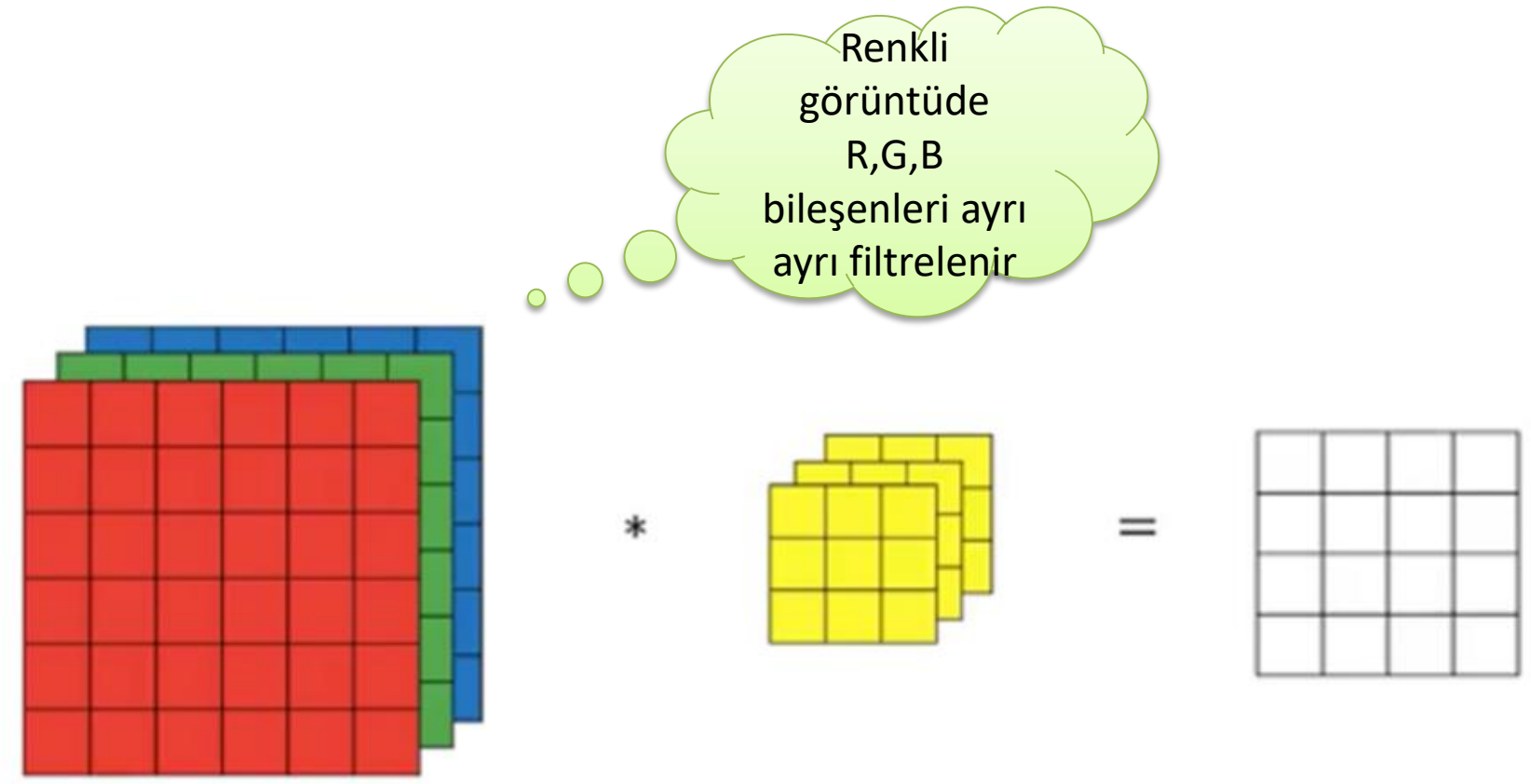
$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 = 320$$

320				

Output Matrix

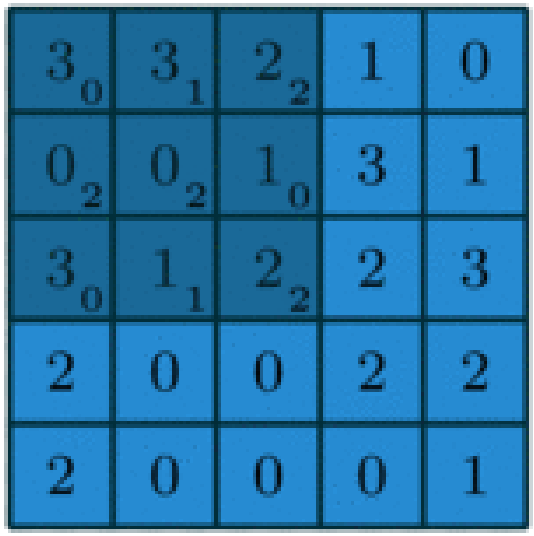
Convolution with horizontal and vertical strides = 1

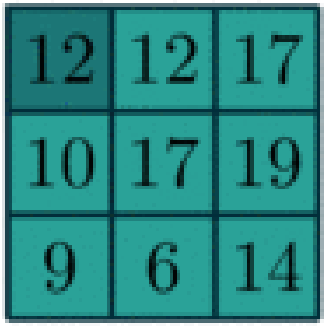
2D convolution



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





- Kernelin kenarlara geldiği hesaplamalar ihmal edilebilir.
- WxH boyutlu bir görüntü 3x3 kernel ile konvolüsyon işlemi sonucu (W-2)x(H-2) boyutlu bir görüntüye dönüşür.
- Benzer şekilde 5x5 kernel için görüntü (W-4)x(H-4) boyutlu bir görüntüye dönüşür.

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

Keras: padding=valid

padding

Çıkış görüntüsünün boyutunu sabit tutmak için kenar dolgulama yapılabilir.

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

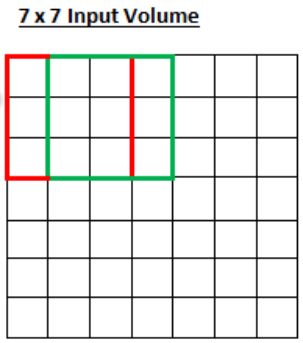
114				

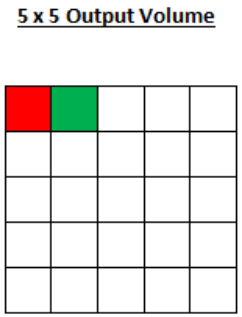
Keras: padding= same

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

Stride (adım büyüklüğü)

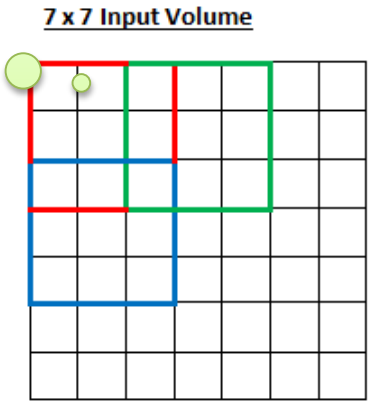
Konvolüsyon işlemi genel de 1 piksel adımlarla gerçekleştirilir.

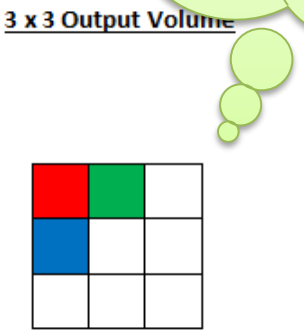




Konvolüsyon 2 adım aralıklarla gerçekleştirildiğinde çıkış görüntüsü yaklaşık yarıya düşer

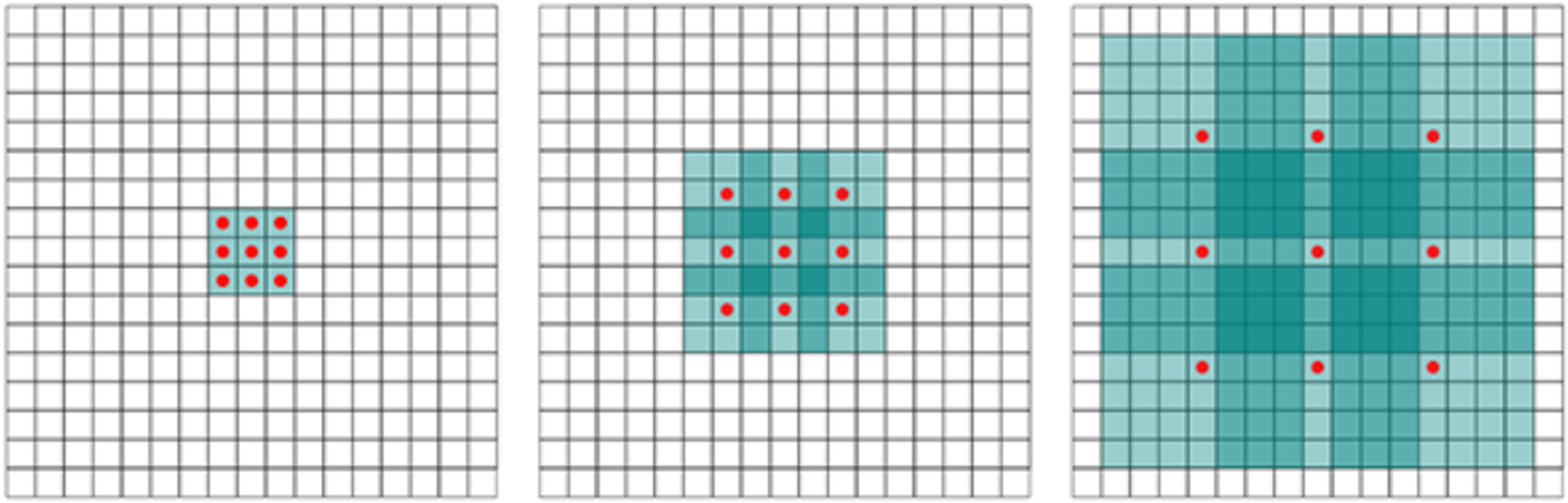
Ancak çıkış görüntüsünü seyrelmek için daha büyük adımlar kullanılabilir.





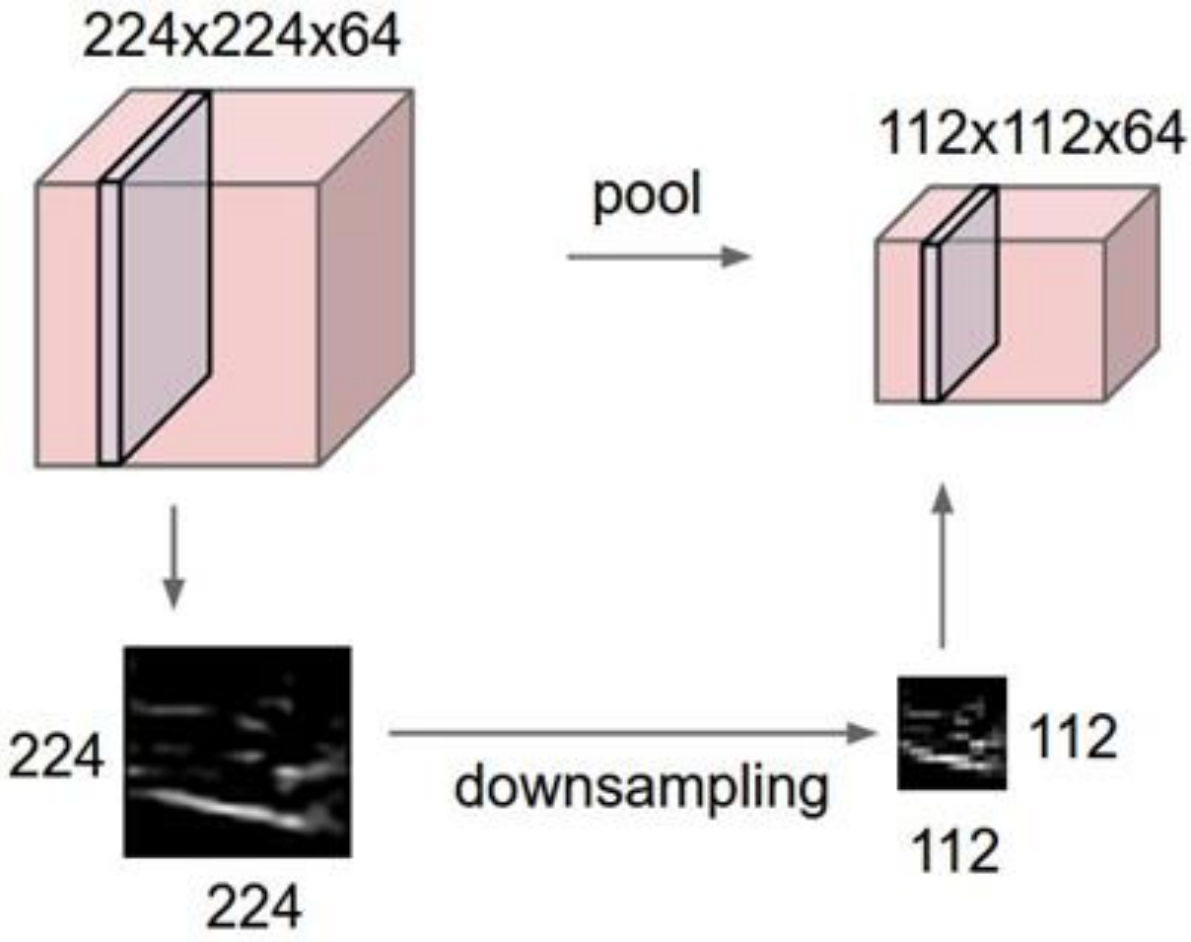
Genişleme oranı (Dilation rate)

- Conv2D sınıfının dilation_rate parametresi, dilate konvolüsyon için dilation oranını kontrol eden bir 2'li tam sayıdır.
- Dilate konvolüsyon, yukarıda şekilde gösterildiği gibi, yalnızca tanımlanan boşluklarla giriş görüntüsüne uygulanan konvolüsyon işlemidir.



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

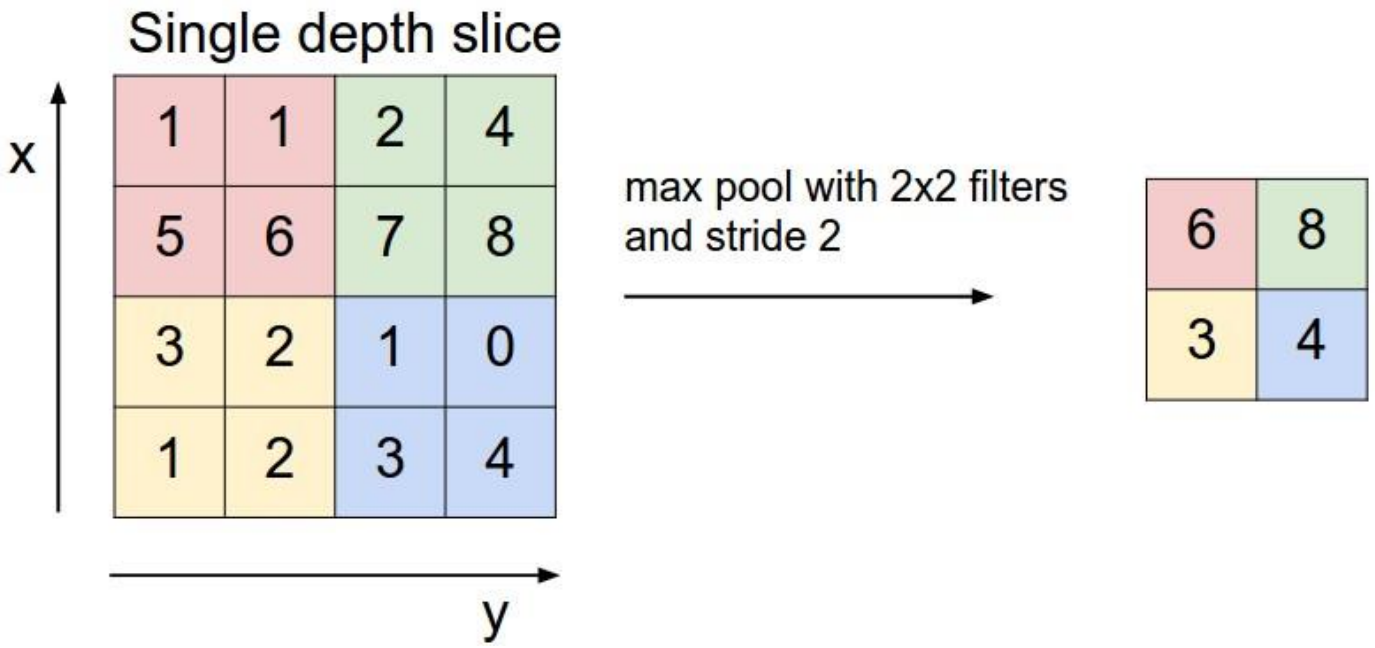
Pooling (birleştirme)



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

Pooling (birleştirme)

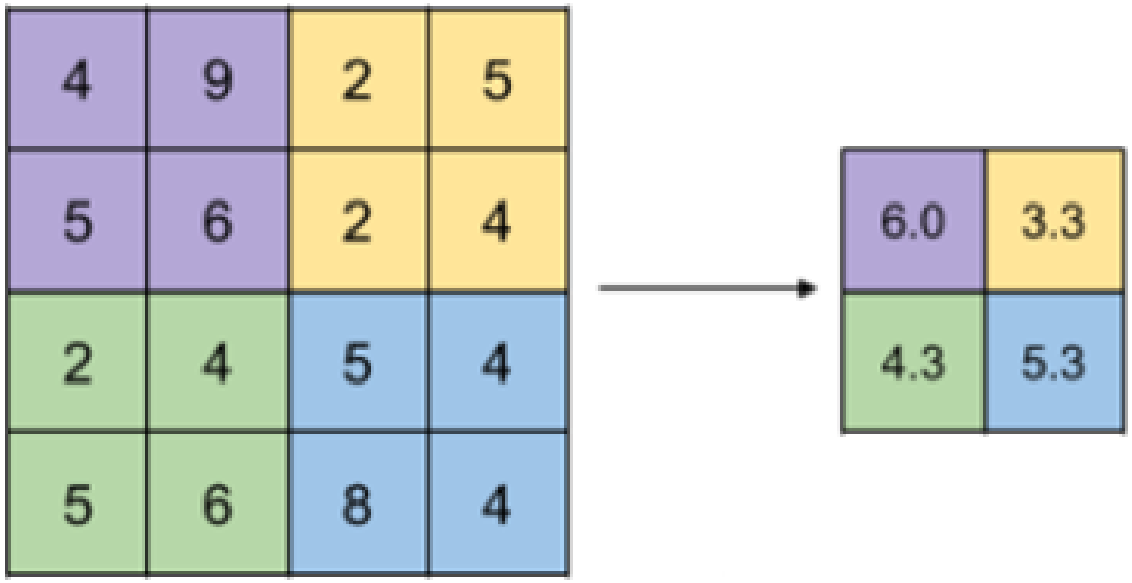
- Max Pooling



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

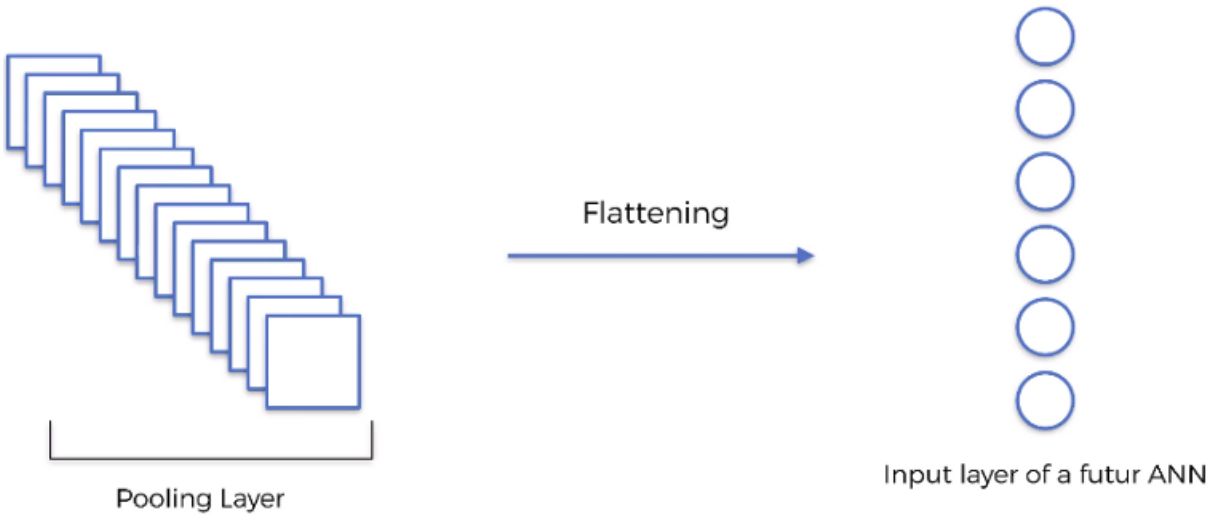
Pooling (birleştirme)

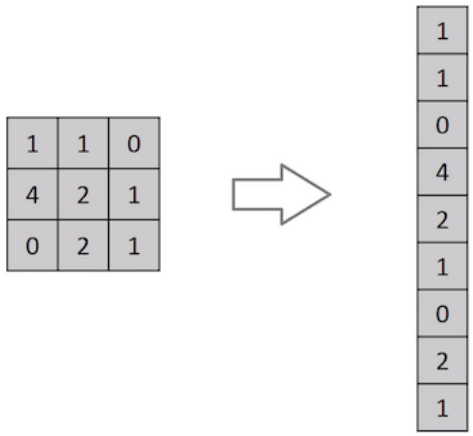
- Average Pooling



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

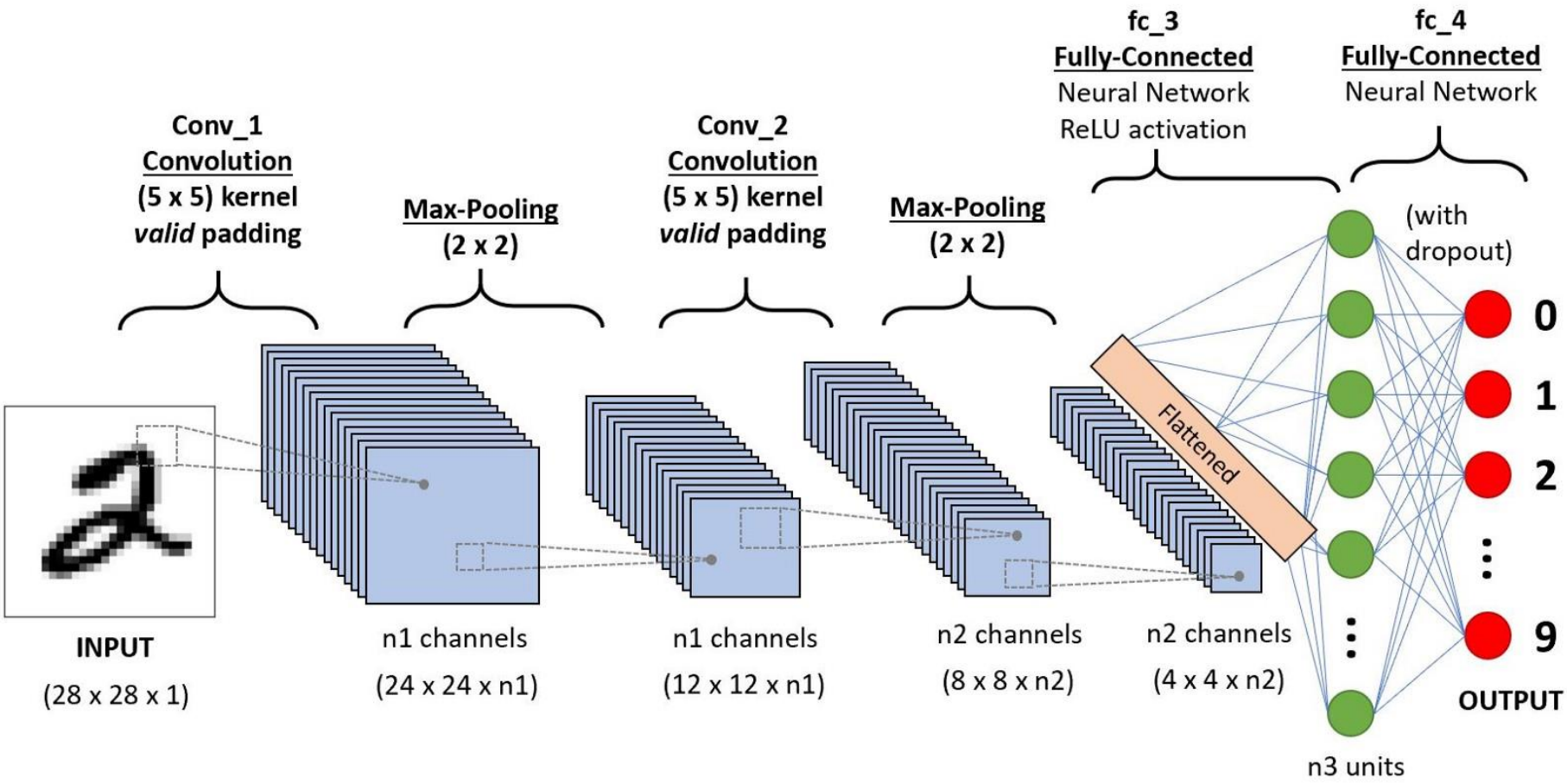
Flatten (düzleştirme)





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

Örnek: mnist veri seti



Örnek: mnist veri seti

```
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()
```

Örnek: mnist veri seti

```
_________________________________________________________________
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
```

Örnek: mnist veri seti

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
```