



**SAKARYA**  
ÜNİVERSİTESİ

# BSM 310 YAPAY ZEKA

CEMİL ÖZ, İSMAİL ÖZTEL

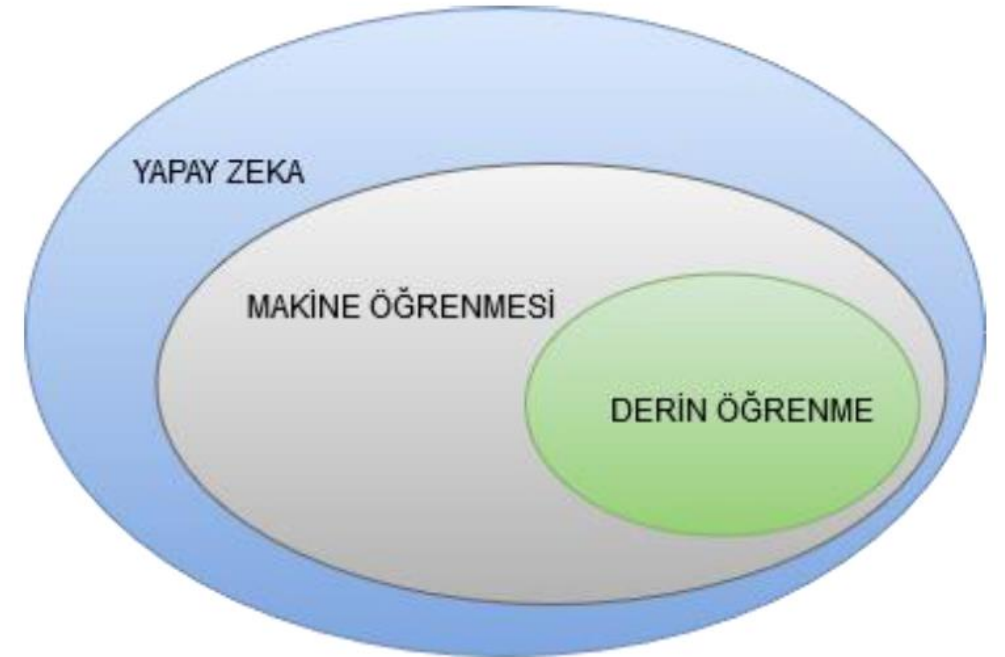
~ KONVOLÜSYONEL SİNİR AĞLARI~

# KONULAR

- Derin öğrenme tanımı
- Sayısal görüntü işlemeye giriş
- Derin ağlarda veri kümesi
- Konvolüsyonel sinir ağları
- Konvolüsyon, Havuzlama, ReLU katmanları
- Derin ağ eğitiminde karşılaşılabilecek sorunlar
- Transfer öğrenme
- Veri çoklama

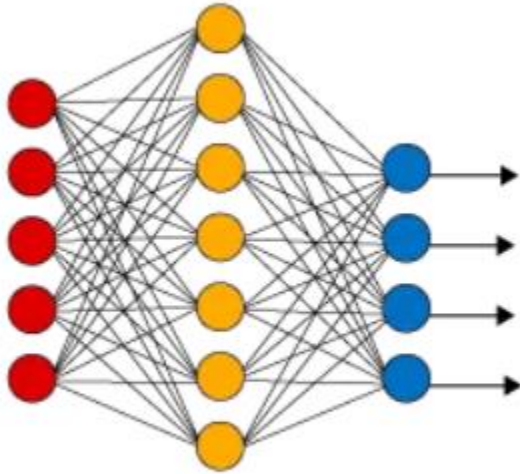
# Derin Öğrenme

- Derin öğrenme, bünyesinde yapay sinir ağı (YSA / ANN) yapısal mimarisini kullanan bir makine öğrenmesi yöntemidir.

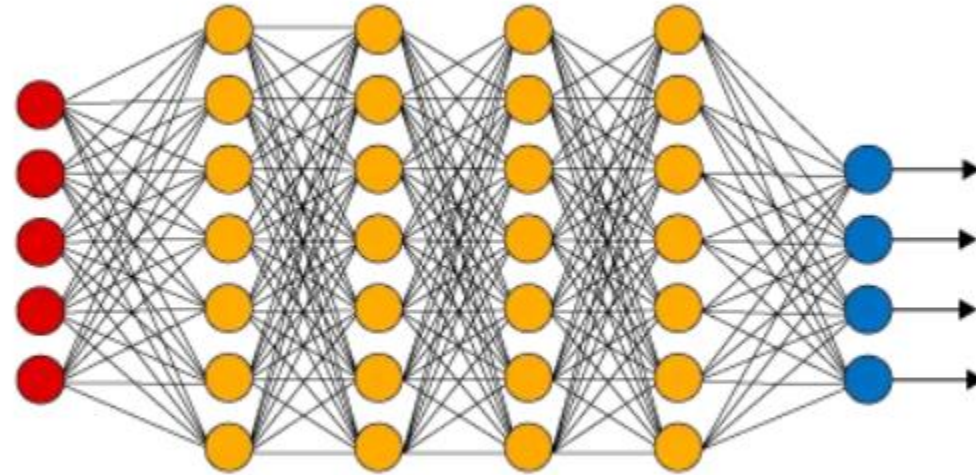


# Derin Öğrenme

Simple Neural Network



Deep Learning Neural Network



● Input Layer

● Hidden Layer

● Output Layer

<https://yapayzeka.ai/python-icin-5-muhtesem-derin-ogrenme-kutuphanesi/>

# Derin Öğrenme- Giriş

- Son yıllarda veri miktarının çok büyük boyutlara ulaşmış olması (Büyük Veri-Big Data)
- Grafiksel İşleme Birimi (GPU) ve Merkezi İşleme Birimi (CPU) gibi donanımların hızlanması / gelişmesi

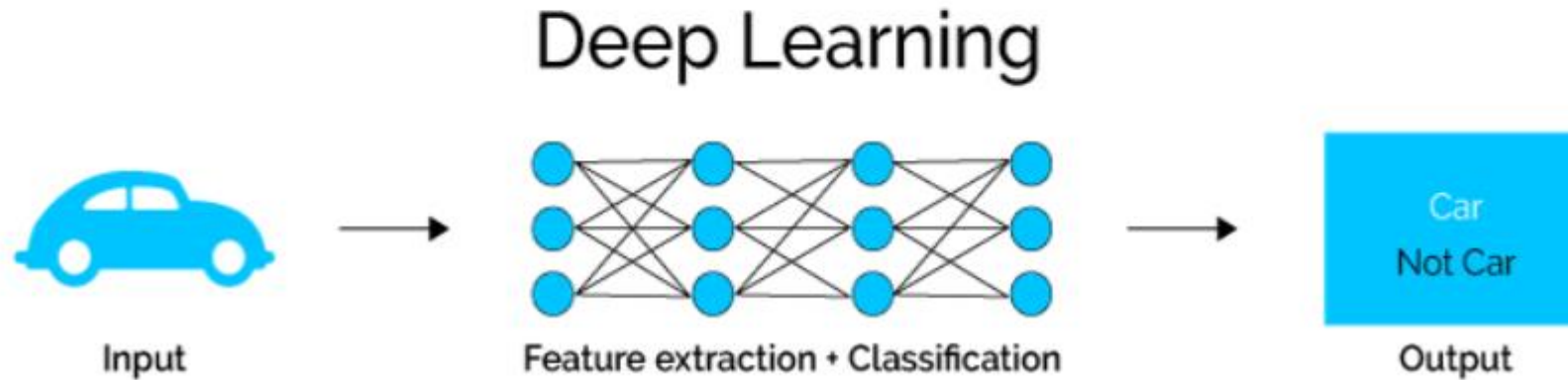
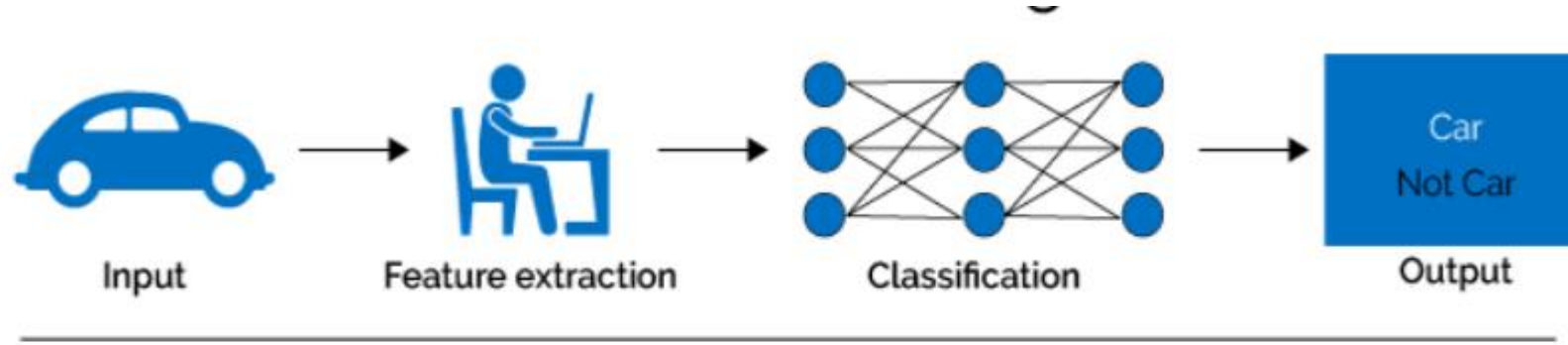
daha fazla sayıda sinir ve katmana sahip YSA'nın eğitilmesi ve çalıştırılmasını mümkün hale getirmiştir. Bu ağlar Derin Sinir Ağları, bu öğrenme de Derin Öğrenme olarak adlandırılır.

# Derin Öğrenme-Giriş

- Derin öğrenme terimi ilk olarak 1986'da Rina Dechter tarafından kullanılmıştır.
- Derin ağların çok kullanılan türleri: Çok katmanlı algılayıcılar(Multi Layer Perceptron-MLP), konvolüsyonel Sinir ağlar (Convolutional Neural Network-CNN), Tekrarlayan Sinir ağları (Recurrent Neural Network-RNN).
- MLP işlemsel veriler üzerinde; CNN bilgisayar görmesinde; ses, metin, zaman serisi gibi ardışık verilerin işlenmesinde ise RNN öne çıkan yöntemlerdendir.

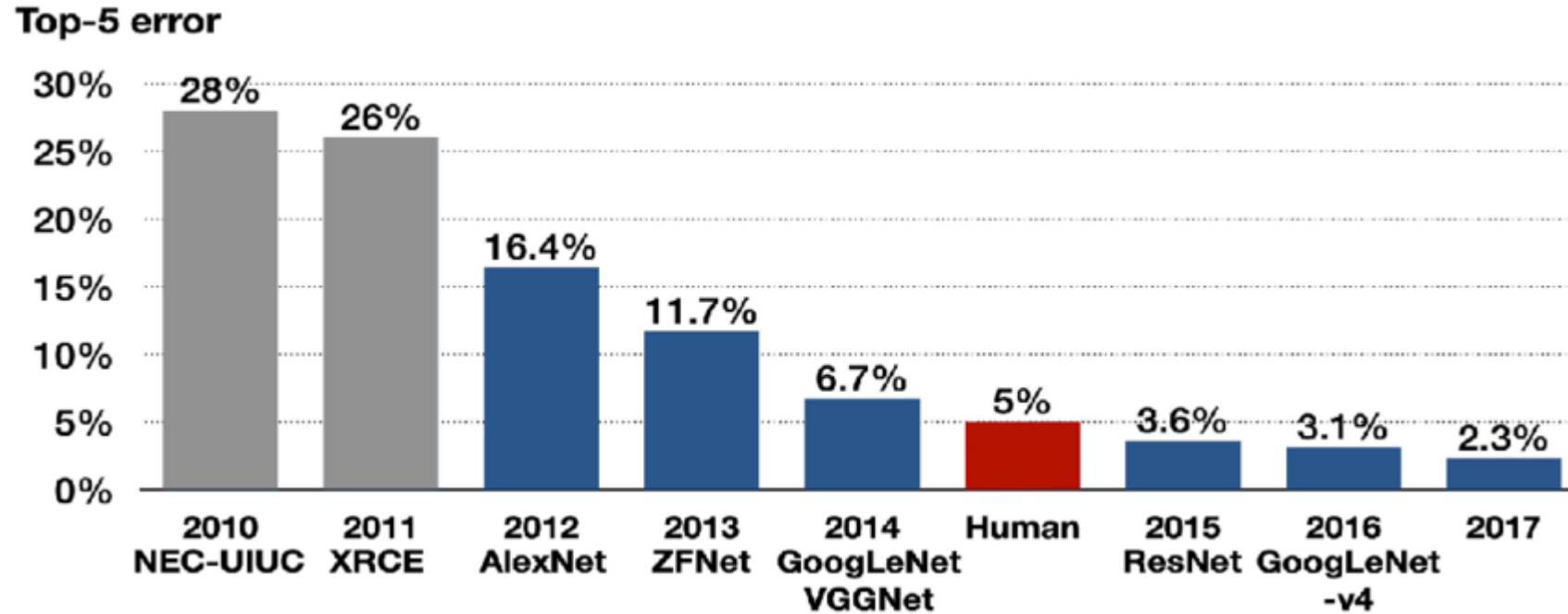
## Derin Öğrenme-Giriş

- Derin öğrenme; ön işleme, boyutsal indirgeme, öznitelik çıkarma ve sınıflandırma aşamalarını tek ağda birleştirir.



<https://www.deeplearning-academy.com/p/ai-wiki-machine-learning-vs-deep-learning>

# Derin Öğrenme-Giriş



[https://www.researchgate.net/publication/346091812\\_Application\\_of\\_Deep\\_Learning\\_in\\_Dentistry\\_and\\_Implantology/figures?lo=1&utm\\_source=google&utm\\_medium=organic](https://www.researchgate.net/publication/346091812_Application_of_Deep_Learning_in_Dentistry_and_Implantology/figures?lo=1&utm_source=google&utm_medium=organic)



# Derin Sinir Ağları Kullanım Alanları



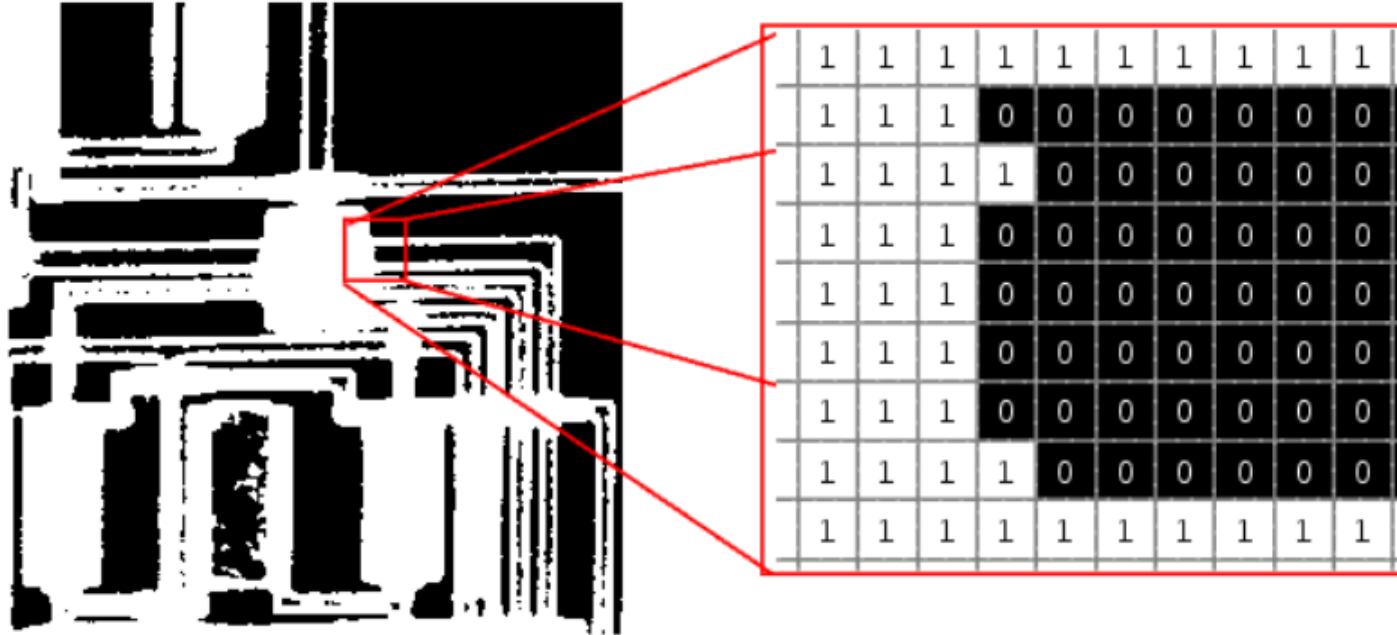
- Derin Sinir Ağları görüntü işleme görüntü sınıflandırma, görüntü bölütleme gibi amaçlarla kullanılabilir.

# Sayısal Görüntü İşleme

- Sayısal görüntü, görüntünün  $(x,y)$  koordinatları ve bu koordinatlara ait renk değerinden oluşur (piksel). Piksellerin aldığı renk değerleri bakımından görüntü 3'e ayrılır.
  - Siyah beyaz görüntü
  - Gri seviyeli görüntü
  - Renkli görüntü

# Siyah-Beyaz Görüntü

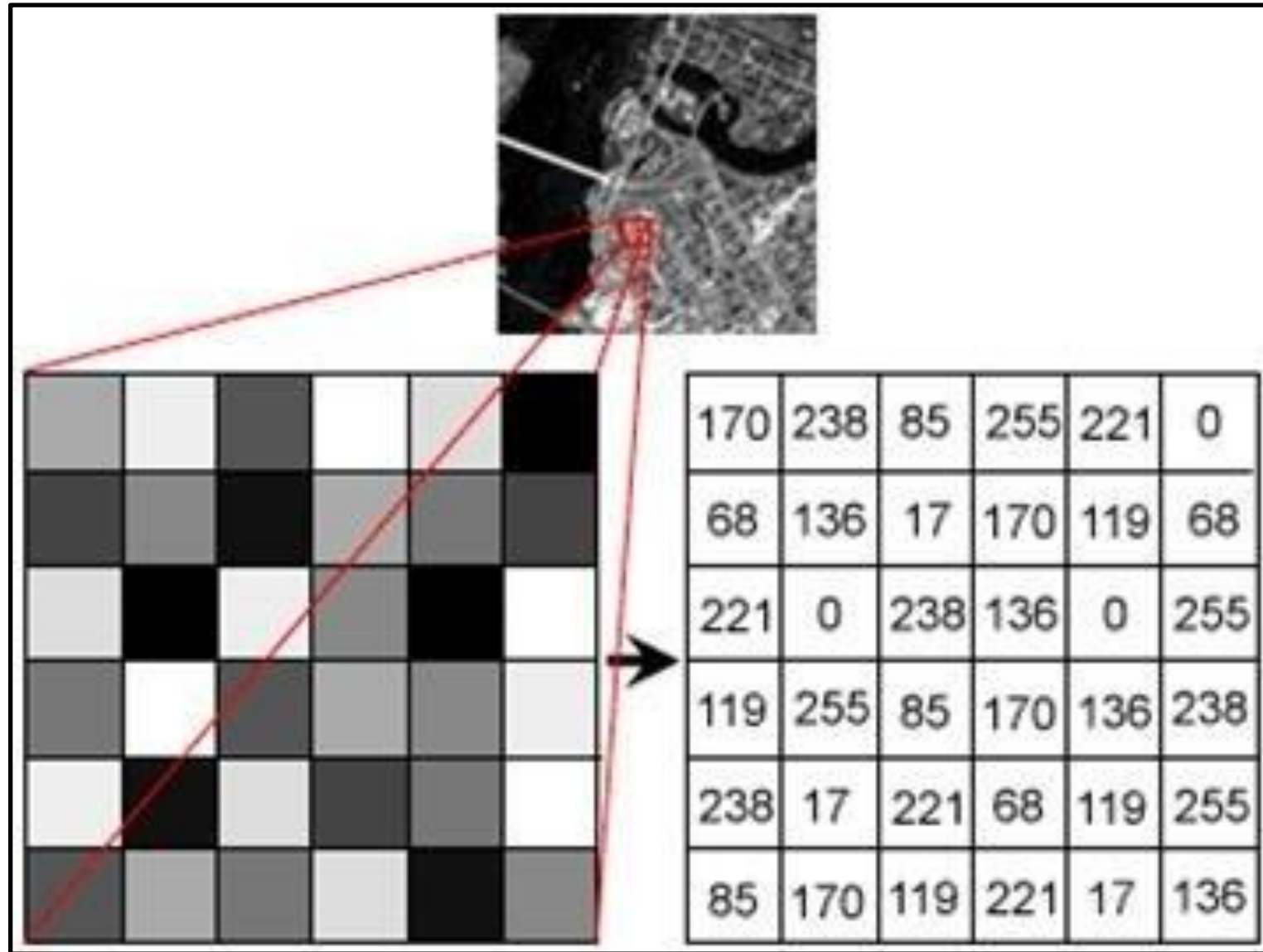
- Sadece siyah ve beyaz olmaz üzere iki piksel değerinden oluşan görüntülerdir



<https://www.mathworks.com/help/images/binary-images.html>

## Gri Seviyeli Görüntü

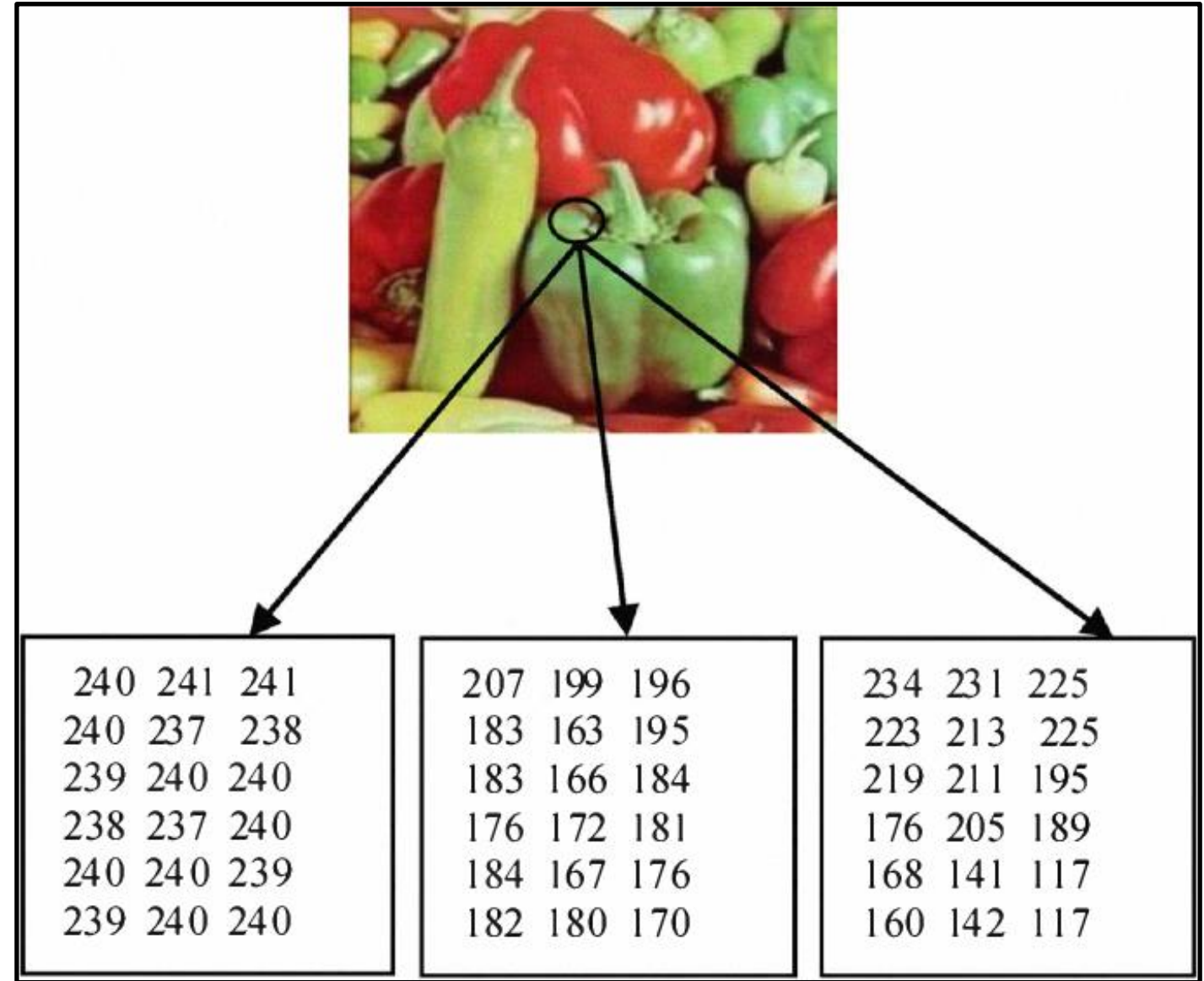
- Piksellerinin tümü gri seviye renklerden oluşur.



Neves et al., Analysis of Emotions From Body Postures Based on Digital Imaging, Third International Conference on Advances in Signal, Image and Video Processing, 2018

# Renkli Görüntü

- Pikselleri kırmızı, yeşil ve mavi renk değerleri içerir.



Karim et al., A new approach for LSB based image steganography using secret key, Computer and Information Technology (ICCIT), 2011

# Sayısal Görüntü İşleme

- Sayısal görüntü işleme, görüntünün onarılması, iyileştirilmesi, anlamlı bölgelere ayrıştırılması gibi amaçlarla gerçekleştirilen klasik veya yapay zekaya dayalı işlemlerdir.
- "Görüntü onarımı" çeşitli sebeplerle bozulan görüntünün tekrar orijinal haline dönüştürülmesi, "görüntü iyileştirme" ise görüntünün daha temiz ve anlaşılır olmasını sağlama amacıyla gürültü giderme kenar belirginleştirme vb. işlemlerdir.

# Sayısal Görüntü İşleme

Distorted Image

## What Is Image Filtering in the Spatial Domain?

Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement.

Filtering is a *neighborhood operation*, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel. (See Neighborhood or Block Processing: An Overview for a general discussion of neighborhood operations.) *Linear filtering* is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

### Convolution

Linear filtering of an image is accomplished through an operation called *convolution*. Convolution is a neighborhood operation in which each output pixel is the weighted sum of neighboring input pixels. The matrix of weights is called the *convolution kernel*, also known as the *filter*. A convolution kernel is a correlation kernel that has been rotated 180 degrees.

For example, suppose the image is

$$A = \begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \end{bmatrix}$$

<https://www.mathworks.com/help/images/ref/imflatfield.html>

# Sayısal Görüntü İşleme

Flat-Field Corrected Image,  $\sigma = 30$

## What Is Image Filtering in the Spatial Domain?

Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement.

Filtering is a *neighborhood operation*, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel. (See Neighborhood or Block Processing: An Overview for a general discussion of neighborhood operations.) *Linear filtering* is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

### Convolution

Linear filtering of an image is accomplished through an operation called *convolution*. Convolution is a neighborhood operation in which each output pixel is the weighted sum of neighboring input pixels. The matrix of weights is called the *convolution kernel*, also known as the *filter*. A convolution kernel is a correlation kernel that has been rotated 180 degrees.

For example, suppose the image is

```
A = [17  24   1   8  15
      23   5   7  14  16
           4   6  13  20  22
      10  12  19  21   3
      11  18  25  26  27]
```

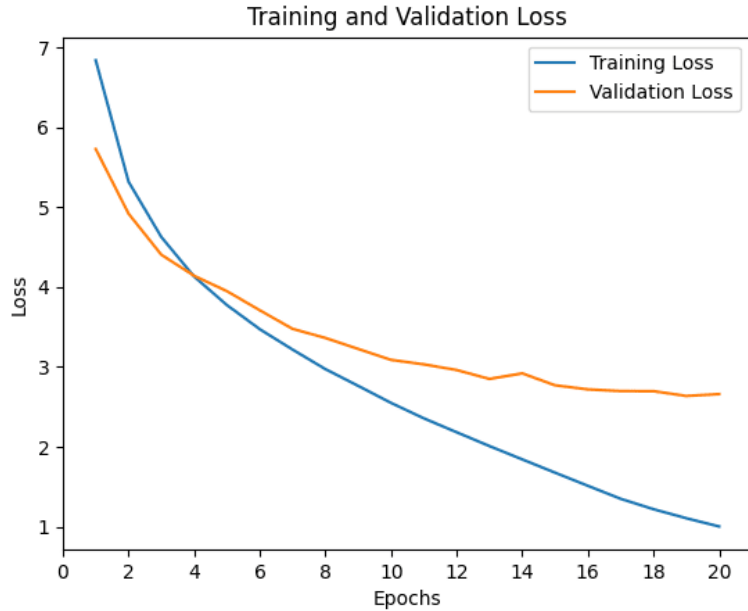
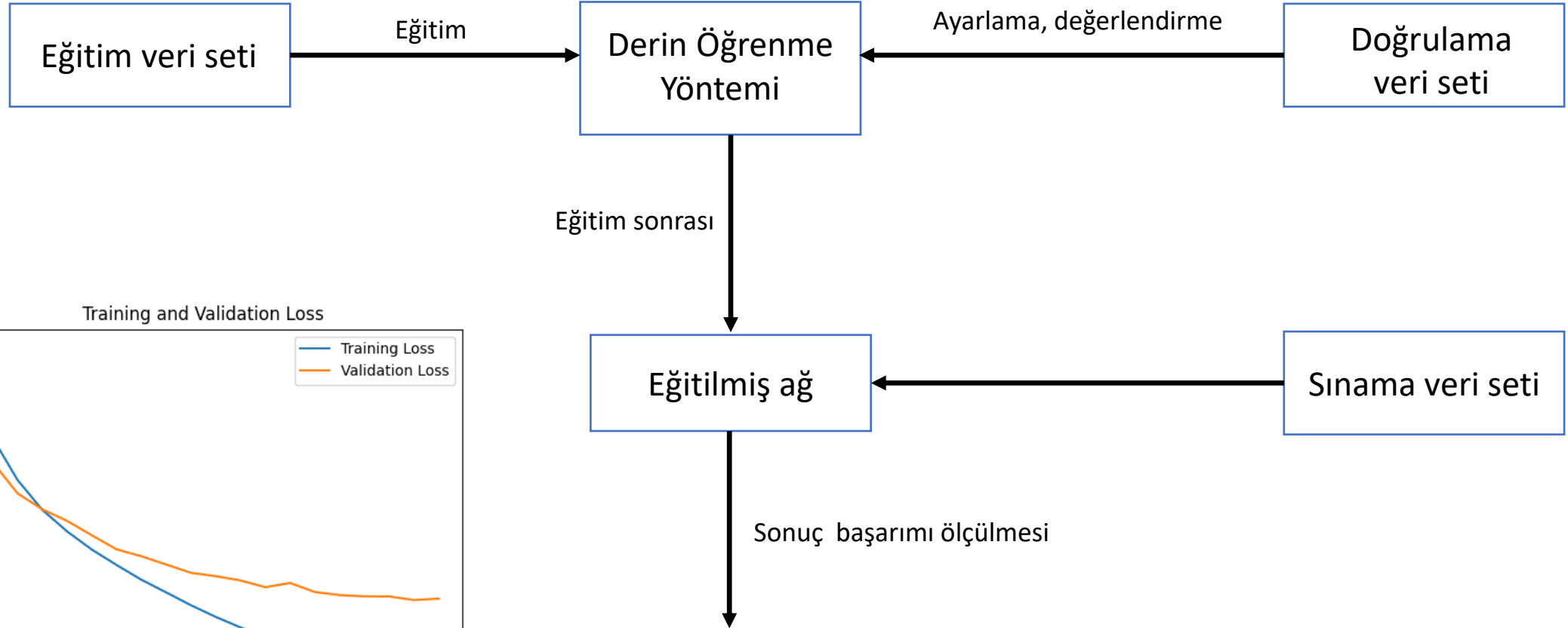
<https://www.mathworks.com/help/images/ref/imflatfield.html>



## Derin Sinir Ağlarda veri kümesi

- YSA'da olduğu gibi derin sinir ağlarda da başarıımı etkileyen önemli faktörlerden biri ağa sunulacak veri kümesidir.
- Veri kümesi genel olarak 3'e bölünür.
  - Eğitim (training),
  - Test (testing)
  - Doğrulama (validation) olarak kullanılır.
- Eğitim kümesi ile ağ eğitilir. Doğrulama veri kümesi eğitim anında başarıımı kontrol etmek için kullanılır. Eğitim sonlandığında ise test veri kümesi ile ağın başarıımı hesaplanır.

# Derin Sinir Ağlarda veri kümesi



## Derin Sinir Ağlarda veri kümesi

- Derin ağların eğitiminde kullanılabilecek el yazısın tanıma, nesne tanıma vb. çok geniş yelpazede açık kaynak,ücretsiz veri kümeleri bulunmaktadır. En bilinenleri aşağıda açıklanmıştır.
- MNIST (Mixed National Institute of Standards and Technology): El yazısı ile yazılmış ve uygun bir şekilde sınıflandırılmış 60.000 eğitim ve 10.000 test görüntüsünden oluşur.
- CIFAR (Canadian Institute for advanced Research): 80 milyon küçük görüntü kümesi içerir. Nesne tanıma amacıyla kullanılır. Uçak, kuş, kedi, vb. görüntüler içerir.
- ImageNet: 14 milyondan fazla görüntü ve 20.000'den fazla sınıf içerir. Sınıflar basit nesnelerden oluşur: araç, yiyecek, ağaç, vb.

# Derin Öğrenme Kütüphaneleri

- Derin öğrenme için geliştirilmiş çok sayıda kütüphane vardır. En yaygın kullanılanları aşağıda açıklanmıştır.
  - TensorFlow: Sayısal hesaplama için kullanılan açık kaynak kodlu bir Python derin öğrenme kütüphanesidir.
  - Theano: Matematik ifadelerini etkili bir şekilde kullanmayı sağlayan Python derin öğrenme kütüphanesidir.
  - Keras: Hem Theano hem de Tensorflow üzerinde çalışan Python ile yazılmış bir derin öğrenme kütüphanesidir.

# Derin Öğrenme Kütüphaneleri

- Torch: Algoritmaları oluşturma konusunda daha işlevsel, kullanıcı dostu arayüzlü bir derin öğrenme kütüphanesidir.
- DL4J (Deep Learning for Java): Keras, TensorFlow gibi açık kaynak kütüphaneleri üzerinde çalışabilen bir Java derin öğrenme kütüphanesidir.
- Caffe: Özellikle CNN’de çok hızlı çalışan modüler ve kullanıcı dostu arayüzlü derin öğrenme kütüphanesidir.

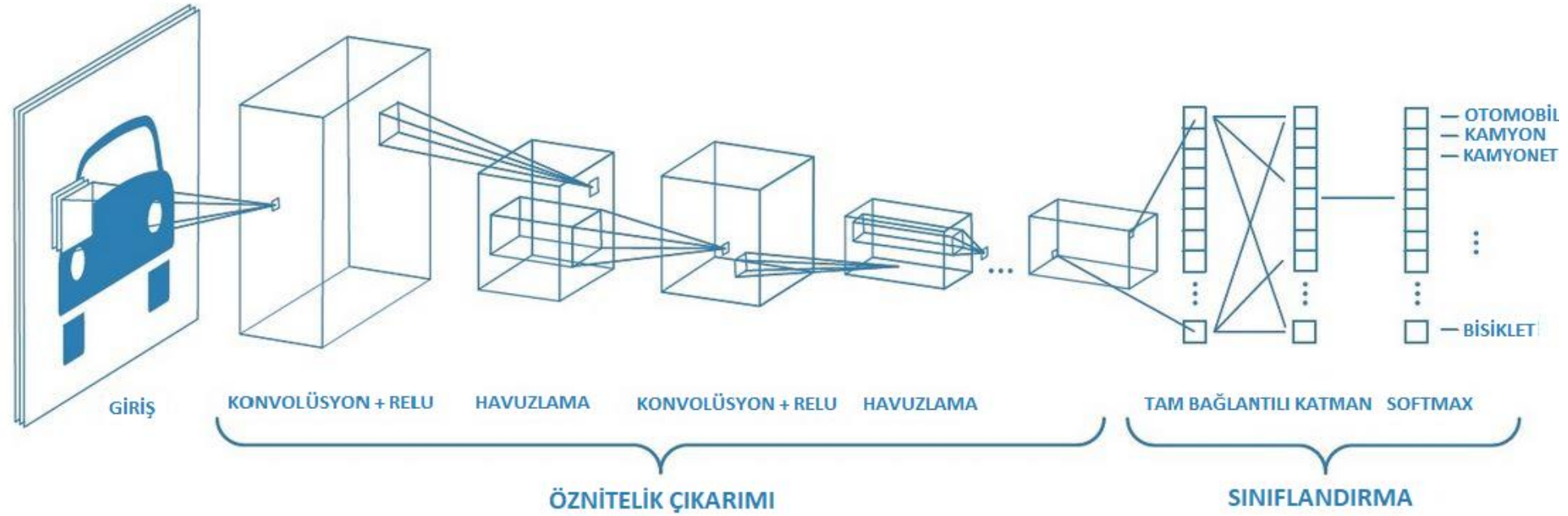
## Derin Öğrenme için kullanılan programlama dilleri

- Derin öğrenme için genellikle kullanılan programlama dilleri aşağıda sıralanmıştır.
- Python: En esnek dillerden biridir. Basitliği sayesinde yapay zeka alanında en sık kullanılan dillerden biridir.
- C++: Donanım düzeyinde programlama yeteneği sayesinde derin öğrenme projelerine uygundur.
- Java: Algoritma kodlamada kolaylığı ile derin öğrenme projelerine uygundur.
- R dili: İstatistiksel modellerin üretilmesinde esneklik sağlar. Oldukça yeni bir dil olmasına rağmen akademik çalışmalarda ve sektörde yaygınlığı artmaktadır.

# Konvolüsyonel Sinir Ağları (Convolutional Neural Networks - CNN)

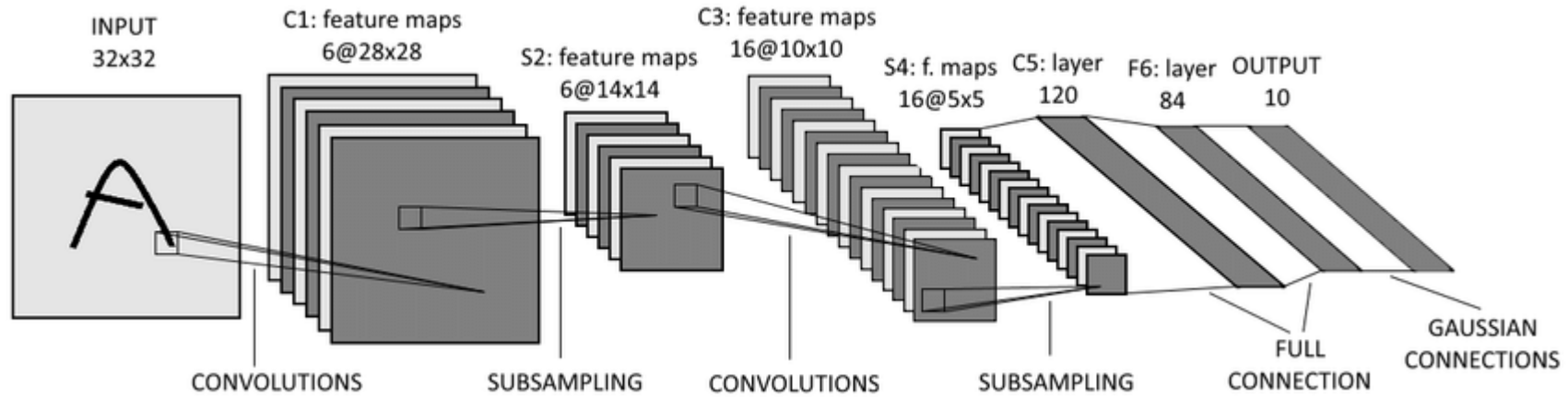
- YSA'lara göre daha fazla sinir ve katmana sahiptir.
- Çalışma şekilleri farklıdır. Örneğin YSA'lar şekil tanıma yaparken alt parçalar yerine şeklin tamamını tanımaya çalışır, CNN önce şeklin alt parçalarını bulur ardından bu şekiller birleştirilerek şeklin tamamı bulunur.
- CNN konvolüsyon (evrişim), havuzlama(pooling) ve tam bağlı (fully-connected), vb. katmanlardan oluşur.
- CNN'e ait ilk konvolüsyon katmanı görüntüde kenar bulma, köşe bulma vb. genel özellikleri bulmaya yönelik çalışır. Sinir ağının sonuna doğru ise probleme özgü ayırt edici özellikler elde edilir.

# Konvolüsyonel Sinir Ağları (CNN)





# Konvolüsyonel Sinir Ağları (CNN)



- Yukarıdaki şekil 1998 yılında YannLeCun ve ekibi tarafından geliştirilen LeNet ağını gösterir. Bu ağ el yazımı rakam tanıma için geliştirilmiştir.
- Bu ağın sinir sayısı ve bağlantı sayısı 60 bini bulmaktadır. Bu sayı YSA ile kıyaslandığında çok önemli ölçüde artmış bununla birlikte başarımlar da önemli ölçüde iyileşmiştir.

# Konvolüsyon

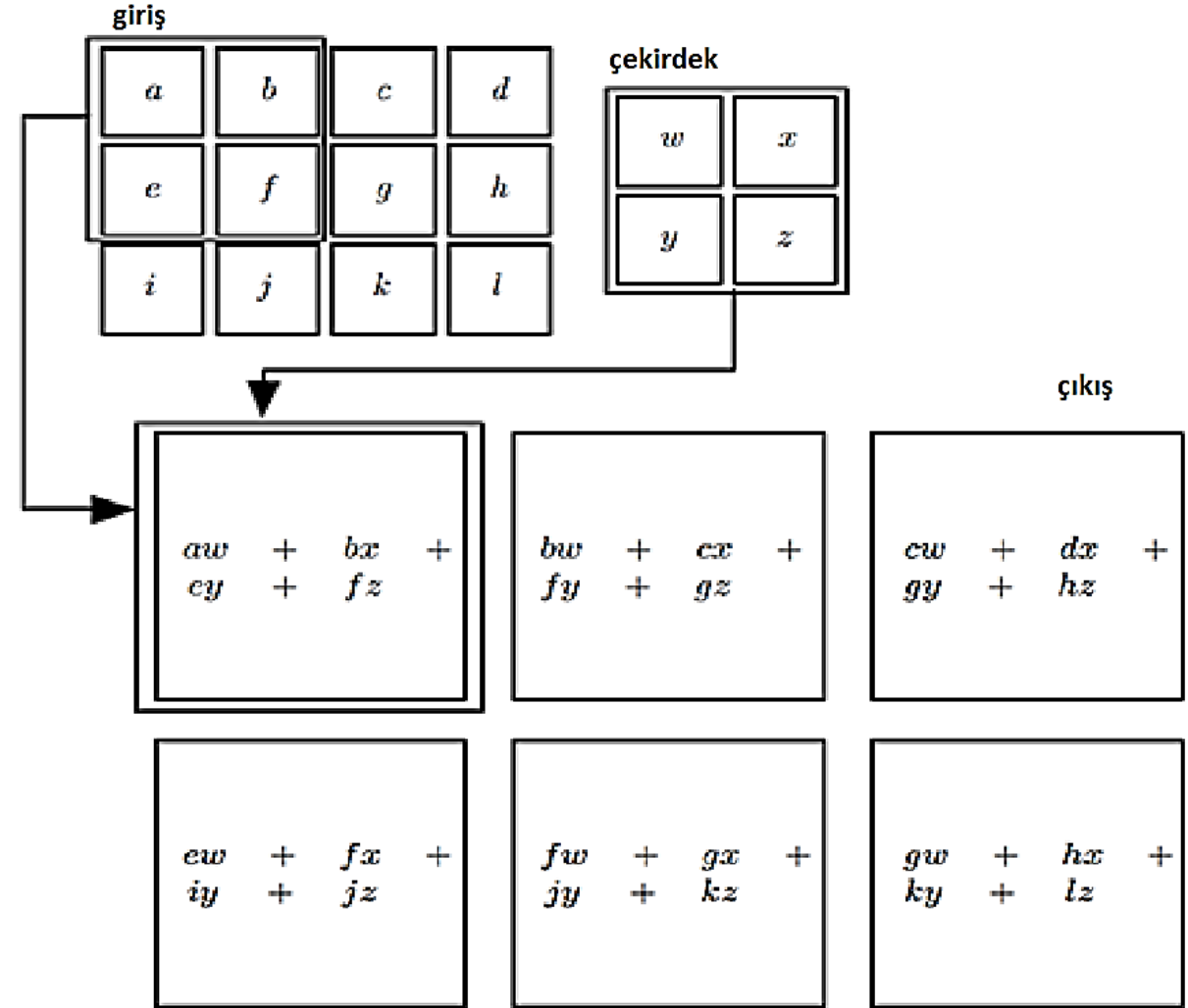
- Toplama, çıkarma, türev gibi matematiksel bir işlemdir.
- Birincil amacı giriş olarak verilen görüntülerin özellik haritalarını çıkarmaktır.
- Bu işlem girdi olarak genellikle iki boyutlu veri dizilerini ve bu diziler üzerinde dolaşabilecek daha küçük iki boyutlu çekirdek dizilerini parametre olarak alır.
- Konvolüsyon işlemi “ \* ” ile gösterilir ve matematiksel ifadesi:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$$

- $I$ ,  $m \times n$  boyutlu görüntü için girişi;  $K$  çekirdek verilerini,  $s$  ise  $i, j$  noktalarındaki işlem sonucunu temsil etmektedir.

# Konvolüsyon

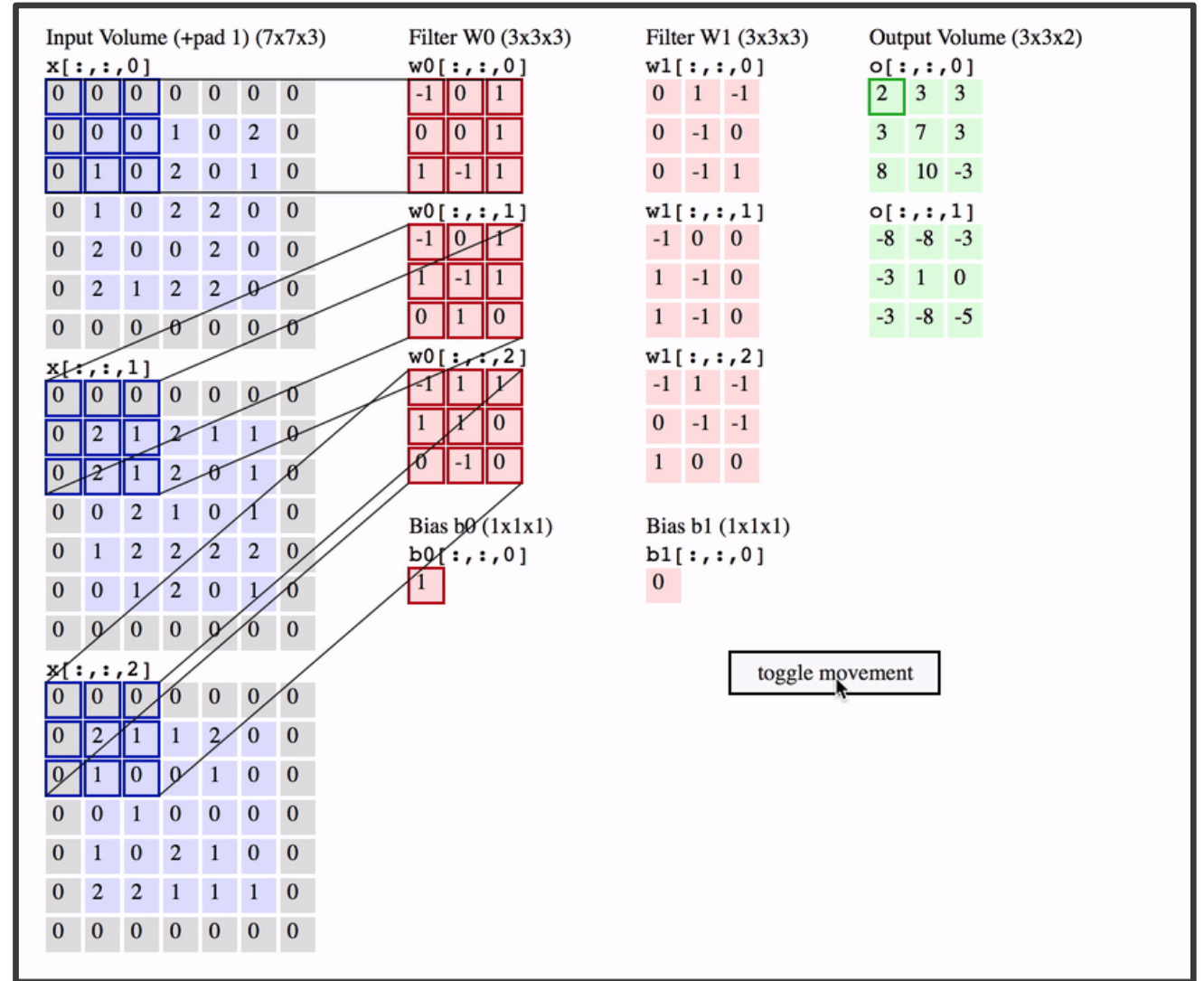
- kayma oranı (stride) 1
- görüntü kenarlarına ekleme yapma (zero padding)



Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.

# Konvolüsyon

- stride 2
- padding 1



CS231n Course Materials, "CS231n Convolutional Neural Networks for Visual Recognition." <https://skymind.ai/wiki/convolutional-network>.

## Sıfır Dolgu (padding)

- Giriş matrisinde çerçeve dışına sıfır ekleme işlemidir.
- Dolgu işlemi ile giriş boyutu ihtiyaca göre ayarlanabilir.
- Genellikle giriş boyutunun çıkışta ayarlanması gerektiği durumda kullanılır.
- Aşağıdaki şekilde 3x3 boyutlu bir matrise her kenardan sıfır dolgu uygulanarak matris 5x5 olarak büyütülmüştür.
- Sıfır dolgu sadece bir kenardan ya da iki kenardan da yapılabilir.

1	2	1
4	5	1
6	8	1



0	0	0	0	0
0	1	2	1	0
0	4	5	1	0
0	6	8	1	0
0	0	0	0	0

# Farklı Filtrelerin Konvolüsyona Etkileri

Özgün görüntü



İşlem	Filtre			Evrişmiş Görüntü
Kenar Algılama	-1	-1	-1	
Keskinleştirme	0	-1	0	
Gaussian Bulanıklık	0.0625	0.125	0.0625	

# Evriřim (Konvolüsyon)



Mathworks, "Visualize Activations of a Convolutional Neural Network. <https://ww2.mathworks.cn/help/nnet/examples/visualize-activations-of-a-convolutional-neural-network.html>.

# Evriřim (Konvolüsyon)



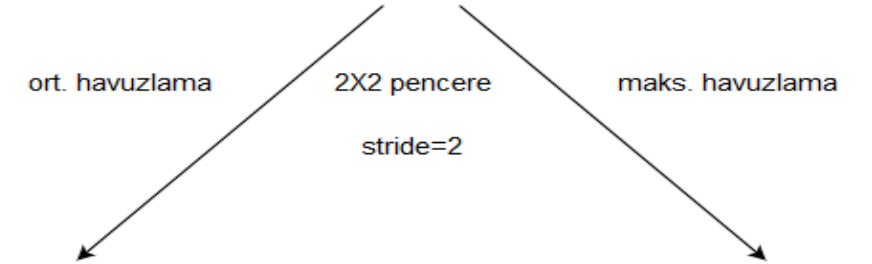
Mathworks, "Visualize Activations of a Convolutional Neural Network. <https://ww2.mathworks.cn/help/nnet/examples/visualize-activations-of-a-convolutional-neural-network.html>.



# Havuzlama (Pooling) Katmanı

- Bir sonraki konvolüsyon katmanı için giriş boyutunu azaltır, bilgi kaybına yol açar.
- Hesaplama yükünü azaltır.
- Ezberlemeyi önler.

1	7	3	9	0	6	0	8
2	0	1	4	1	0	7	0
1	2	3	7	2	1	3	9
0	0	4	2	5	6	0	9
9	1	8	3	0	2	3	8
7	9	0	2	1	5	4	2
5	4	1	1	4	7	8	7
2	3	8	3	8	0	5	6



2	4	1	3
0	4	3	5
6	3	2	4
3	3	4	6

7	9	6	8
2	7	6	9
9	8	5	8
5	8	8	8

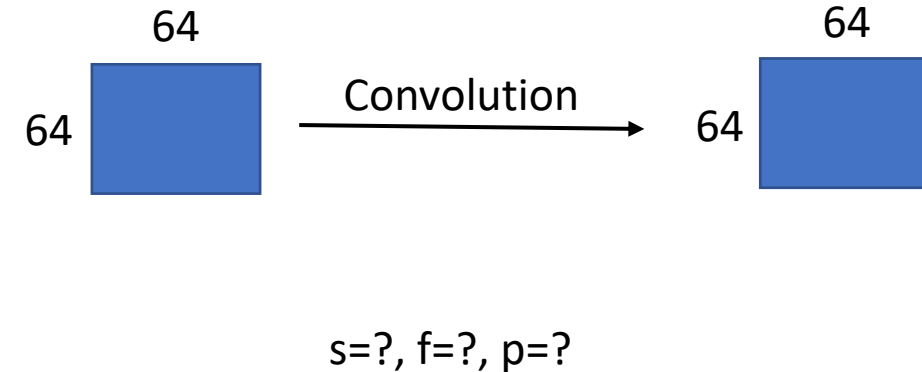
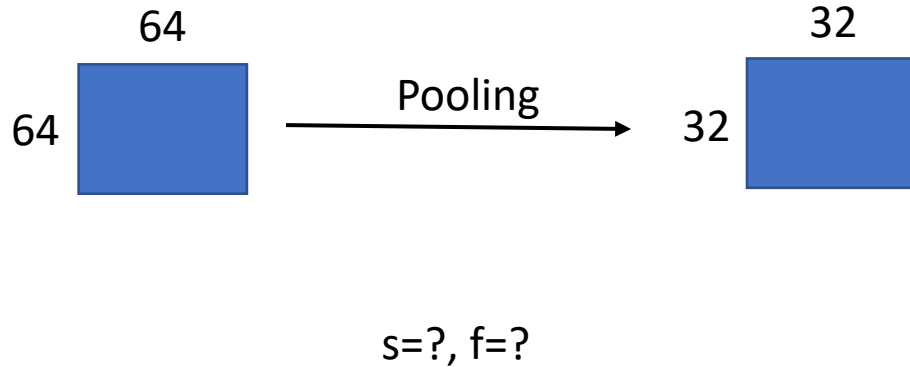
# Katmanların çıkış boyutlarının hesaplanması

Katman	Çıkış boyutu hesaplama formülü
Havuzlama (Pooling)	$\frac{w - f}{s} + 1$
Evrişim (Convolution)	$\frac{w - f + 2p}{s} + 1$

- w: giriş boyutu
- f: filtre boyutu
- s: adım sayısı
- p: padding sayısı

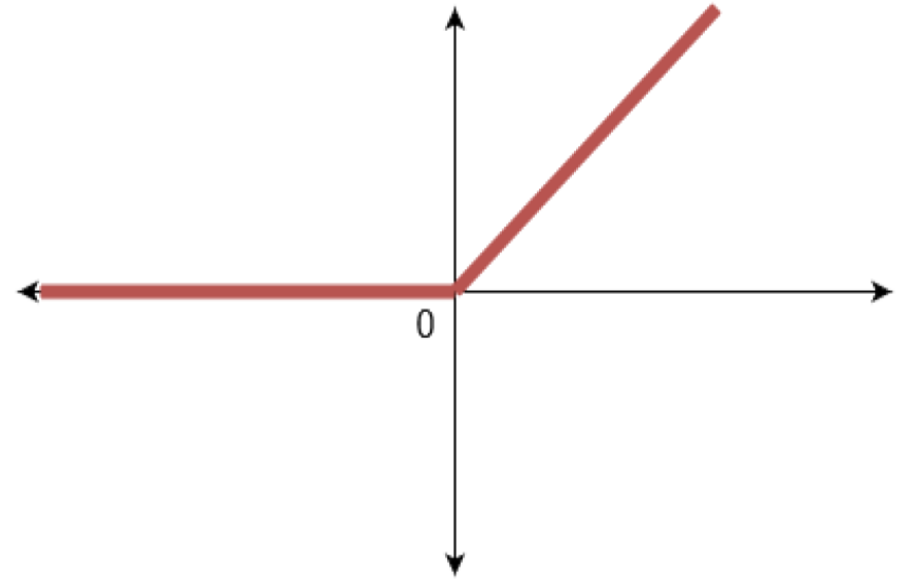
# Katmanların çıkış boyutlarının hesaplanması

Katman	Çıkış boyutu hesaplama formülü
Havuzlama (Pooling)	$\frac{w - f}{s} + 1$
Konvolüsyon (Convolution)	$\frac{w - f + 2p}{s} + 1$



## ReLU Katmanı

- $G(z) = \max\{0, z\}$
- ReLU'nun giriş parametresi olarak aldığı değer negatif veya sıfır ise sıfır, değilse kendisini döndürür.
- ReLU'yu diğer aktivasyon fonksiyonlarından ayıran en önemli özellik, diğerlerine göre daha hızlı çalışmasıdır.
- Bu katmanın kullanılması ile ağ daha hızlı bir eğitim gerçekler.



# Alıştırma

1	2	3
4	5	6
7	8	0

giriş  
görüntüsü

konvolüsyon  
s=1, p=1  
→  
 $\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$

?

ReLU  
→

?

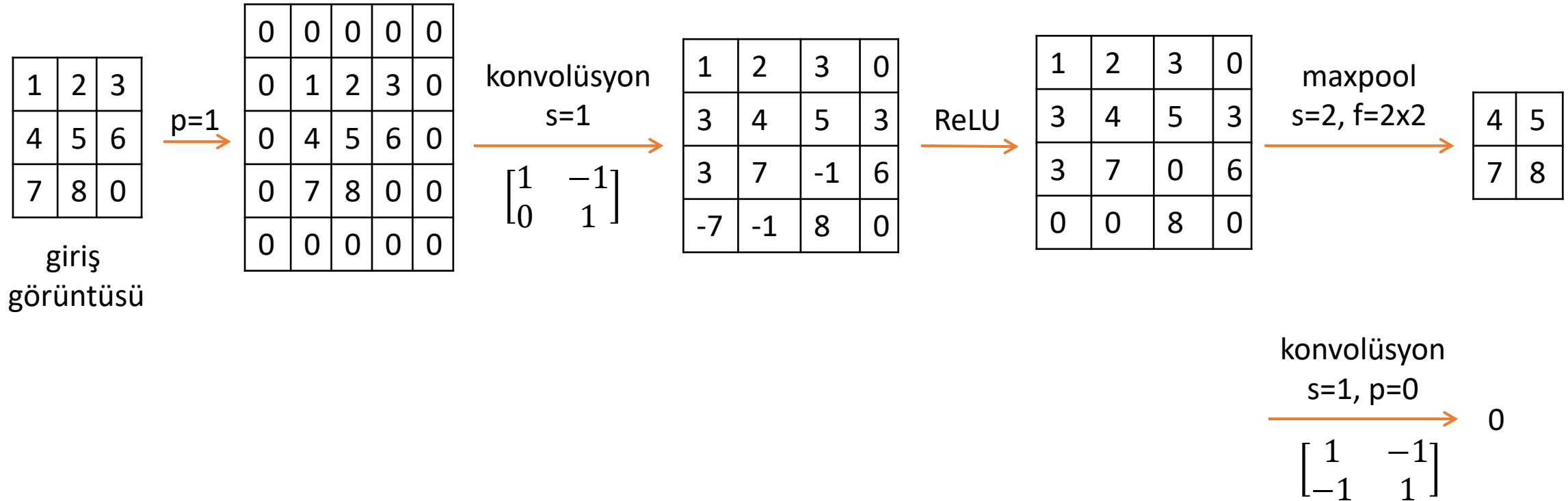
maxpool  
s=2, f=2x2  
→

?

konvolüsyon  
s=1, p=0  
→  
 $\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$

?

# Alıştırma



# Ağın eğitiminde ortaya çıkabilecek sorunlar

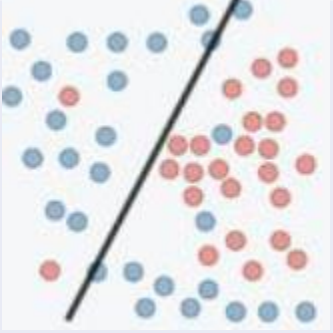
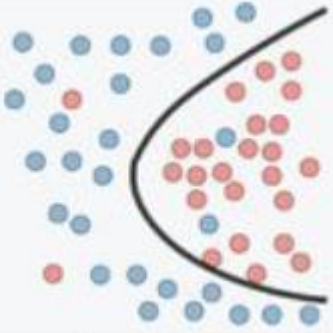
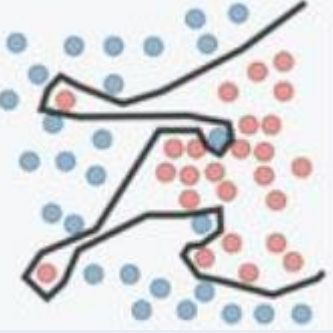

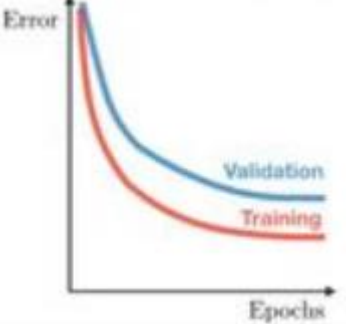

- Ağ eğitilirken karşılaşılabilecek sorunlar
  - Ezberleme (Overfitting)
  - Yanlış öğrenme (Underfitting)
- Ezberleme ve yanlış öğrenme iki sebeple olabilir.
  - Veri kümesi ile ilgili nedenler,
  - Eğitim süresi ile ilgili nedenler

## Ağın eğitiminde ortaya çıkabilecek sorunlar

- Ezberleme, ağın eğitim setine doğru cevap verip test verilerine genelleme yapamayıp düzgün sonuç üretememesidir.
  - Eğitim uzay yeterli genişliğe sahip değilse ezberleme ortaya çıkar.
  - Örnek: Bir ağın kuş cinslerini tanımak üzere eğitildiğini düşünelim. 50 bin görüntü ile ağ eğitilsin. Bu görüntülerin yarısı kanarya, diğer yarısı ise kanarya benzeri çeşitli kuşlara ait olsun. Bu durumda veri kümesi dengesiz dağıldığından ezberleme ihtimali olasıdır.
  - Ezberlemeyi önlemek için eğitim veri kümesi genişletilmelidir.
  - Aynı verilerle çok fazla eğitim yapıldığında da ezberleme sorunu ortaya çıkar.



# Ağın eğitiminde ortaya çıkabilecek sorunlar

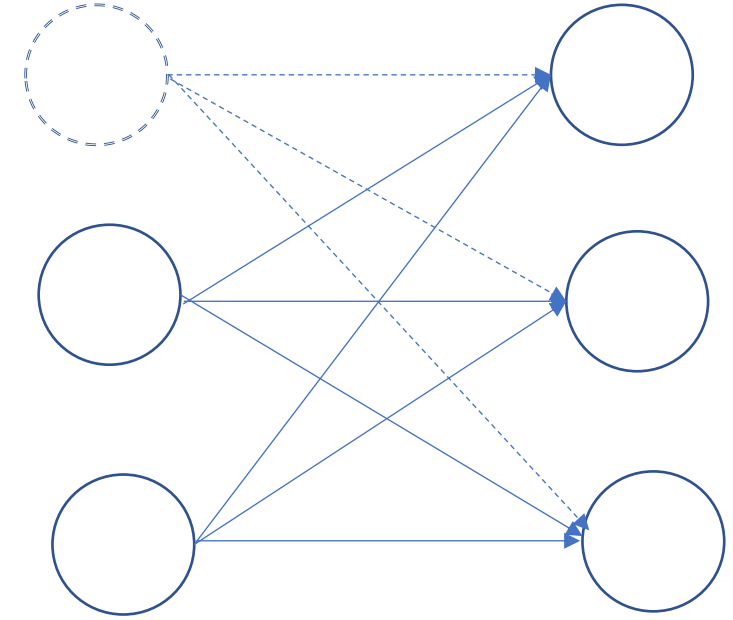
	Underfitting (Yanlış öğrenme)	Öğrenme	Overfitting (Ezberleme)
Belirtiler	<ul style="list-style-type: none"><li>Yüksek eğitim hatası</li><li>Eğitim hatası ile test hatası birbirine yakın</li></ul>	<ul style="list-style-type: none"><li>Eğitim hatası test hatasından az miktarda düşüktür.</li></ul>	<ul style="list-style-type: none"><li>Çok düşük eğitim hatası</li><li>Eğitim hatası test hatasından çok çok düşüktür.</li></ul>
Sınıflandırma gösterimi			
Derin Öğrenme gösterimi			
Çözüm için yapılabilecekler	<ul style="list-style-type: none"><li>Model karmaşıktırılabilir</li><li>Daha uzun süre eğitim yapılabilir</li></ul>		<ul style="list-style-type: none"><li>Daha fazla veri eklenebilir</li></ul>

# Seyreltme

- Eğitimde ortaya çıkan ezberleme gibi sorunları önlemek için seyreltme başvurulabilecek bir başka yöntemdir.
- Seyreltme ağ boyutunu azaltarak eğitimin iyileşmesine yardımcı olur. Ağların daha genelleştirilmiş sonuçlar vermesine katkı sağlar.
- Seyreltme 3 şekilde yapılabilir:
  - Sinir Seyreltme (Dropout)
  - Bağlantı Seyreltme (Dropconnect)
  - Tümünü Seyreltme (DropAll)

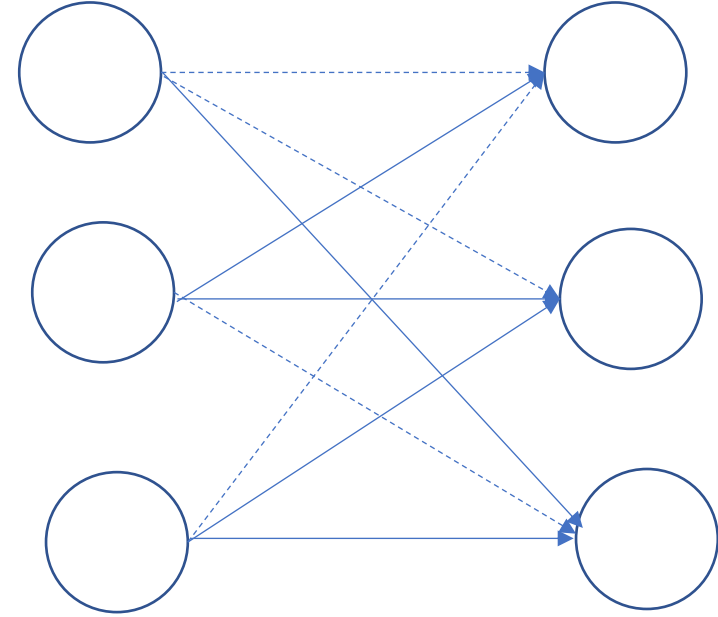
## Dropout (Sinir seyreltme) katmanı:

- Dropout katmanı, ağlarda aşırı uydurma (overfitting) önüne geçmek için kullanılır.
- Her eğitim aşaması için belli bir oranda rastgele seçilen sinirler yok sayılır. O sinirden sonraki katmandaki sinirlere bağlantı yapılmaz.



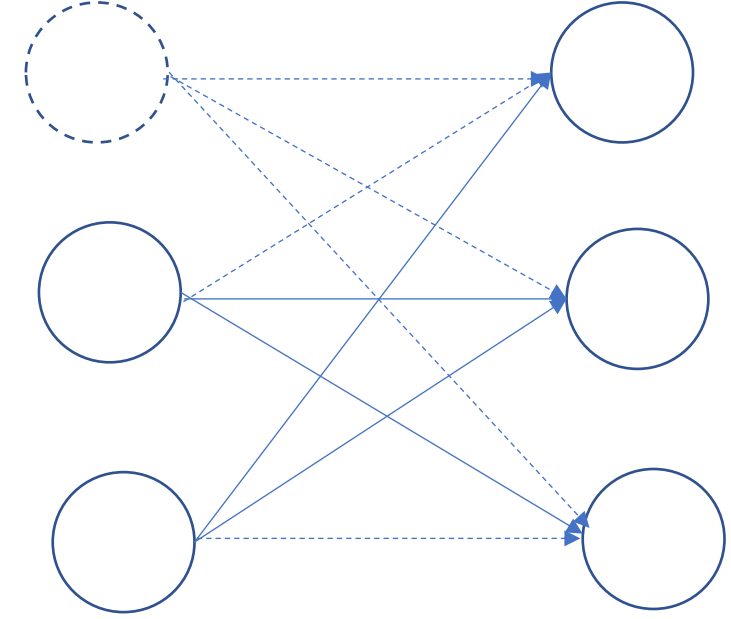
## DropConnect (Bağlantı seyreltme) katmanı:

- Sinir seyreltme işlemindeki sinirlerin sonraki katman sinirlerine bağlanmaması yerine belli bir oranda rastgele seçilen ağırlıklar sıfır yapılarak sonraki katmandaki bazı sinirlere bağlantılar koparılır.



## DropAll (Tümünü seyreltme) katmanı:

- Sinir seyreltme ve bağlantı seyreltmenin belli bir oranda rastgele seçilen sinir ve bağlantılara uygulanması işlemidir.



## Tam Bağlantılı Katman (Fully Connected):

- Softmax ile birlikte sınıflandırma işlevlerini yerine getiren katmandır.
- Kendinden önceki katmanın tüm elemanlarıyla bağlantılıdır ve sınıflandırmadan önceki aşama için, sınıf etiketlerine dair olasılıksal değerler içerir.
- Bu katmandan elde edilen sayısal değerlerin hangi sınıf ile ilişkili olduğuna softmax katmanı karar verir ve bu olasılıksal değerleri  $[0,1]$  aralığına çeker.
- Değerler incelendiğinde; en büyük değer hangi sınıfa ait ise, ağ o sınıf etiketini çıktı olarak üretir.

## Geri Yayılım Algoritması (Backpropagation)

- Yapay sinir ağlarında kullanılan eğitim yöntemidir.
- Geri yayılım yaklaşımı ile güncellemeleri ağ üzerinde gerçekleştirir.
- Güncellemeler ağın çıktısı ile beklenen çıktı arasındaki hatayı minimize etmeyi amaçlar. Bu hata değerleri en küçük ortalamaların karesi (Least Mean Square - LMS) hatası ile hesaplanabilir.

$$J(w) = \frac{1}{2} \sum_{k=1}^c (t_k - z_k)^2$$

- Denklemden  $z$  tahmin edilen sonuç vektörünü,  $t$  ise olması gereken sınıflandırma sonuç vektörünü göstermektedir.  $c$  vektör uzunluğunu,  $w$  ise ağdaki ağırlıkları sembolize etmektedir.