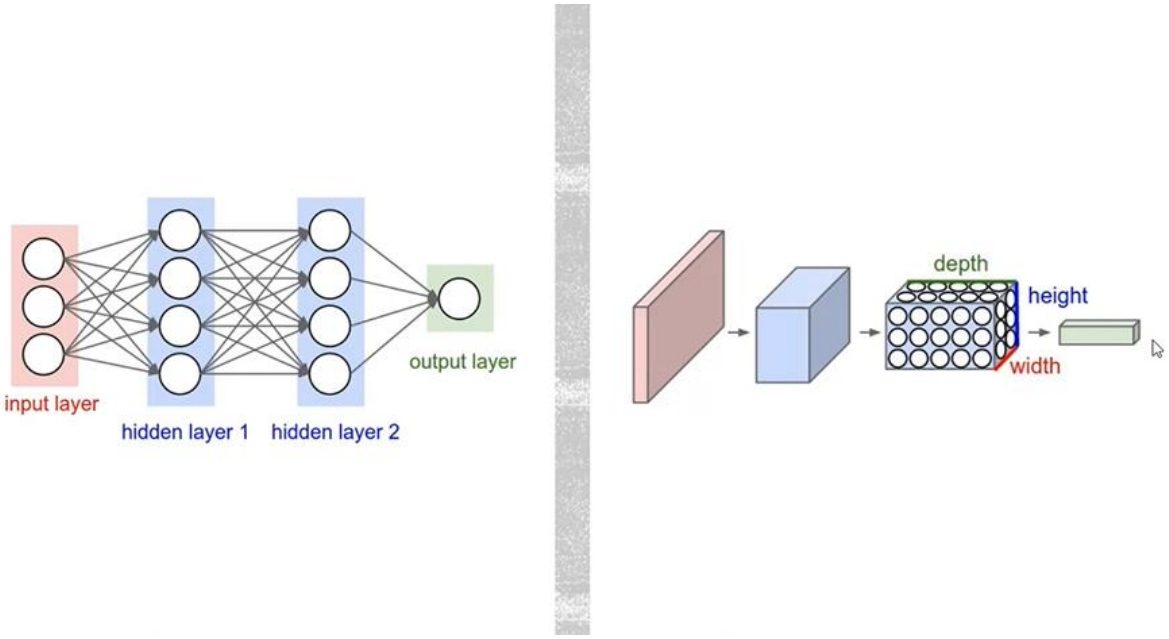
YAPAY SİNİR AĞLARI VE CNN (EVRİŞİMLİ SİNİR AĞLARI)

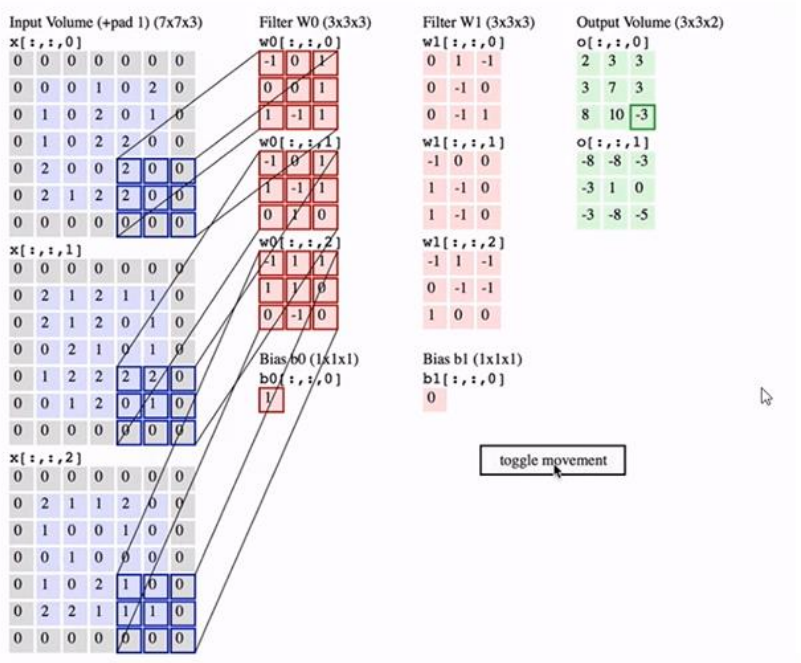


Solda Yapay Sinir Ağları, sağda ise Evrişimli Sinir Ağları (CNN) katmanları vardır.

Nöronları ifade etmek için yapay sinir ağlarında vektörel bir değer varken, evrişimli sinir ağlarında matriksel bir değer vardır.

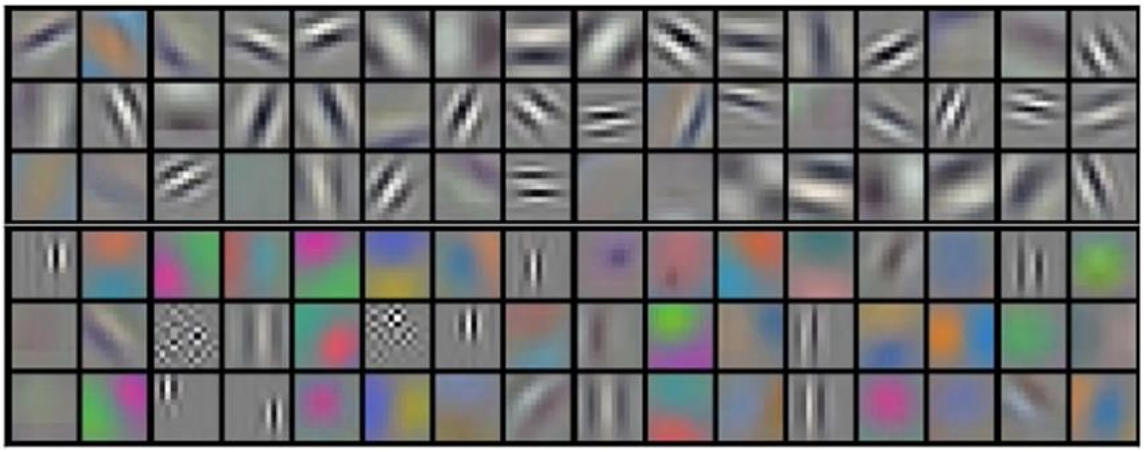
CNN sonunda elde ettiğimiz çıkış bir vektör olacaktır.

CNN'de giriş matrisine evrişim işlemleri yaptırarak çıkıştıra değerler elde ediyoruz.



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

Evrişim İşlemi Uyguladığımızda Elde Edeceğimiz Görüntü Örnekleri



Başlangıçta, basit öznitelik çıkartımı olur. Dikey/Yatay rotasyona sahip kenarları belirler. (Yüksek frekanslar kenarları verir.)

Spesifik renk geçişlerini belirler.

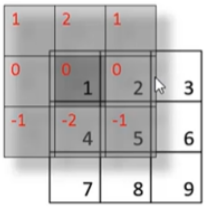
Daha derin katmanlarda, spesifik öznitelikleri öğrenir.

EVRİŞİM İŞLEMİ

1	2	3
4	5	6
7	8	9

n \ m	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

$$y[0,0] = \sum_j \sum_i x[i,j] \cdot h[0-i, 0-j]$$



$$\begin{aligned}
&= \quad x[-1,-1] \cdot h[1,1] + x[0,-1] \cdot h[0,1] + x[1,-1] \cdot h[-1,1] \\
&\quad + x[-1,0] \cdot h[1,0] \quad + x[0,0] \cdot h[0,0] \quad + x[1,0] \cdot h[-1,0] \\
&\quad + x[-1,1] \cdot h[1,-1] + x[0,1] \cdot h[0,-1] + x[1,1] \cdot h[-1,-1] \\
&= \quad 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 \\
&\quad + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 \\
&\quad + 0 \cdot (-1) + 4 \cdot (-2) + 5 \cdot (-1) \\
&= -13
\end{aligned}$$

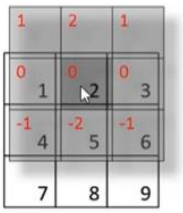
http://www.songho.ca/dsp/convolution/convolution2d_example

1	2	3
4	5	6
7	8	9

n \ m	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

-13	-20	

$$y[1,0] = \sum_j \sum_i x[i,j] \cdot h[1-i, 0-j]$$



$$\begin{aligned}
&= \quad x[0,-1] \cdot h[1,1] + x[1,-1] \cdot h[0,1] + x[2,-1] \cdot h[-1,1] \\
&\quad + x[0,0] \cdot h[1,0] \quad + x[1,0] \cdot h[0,0] \quad + x[2,0] \cdot h[-1,0] \\
&\quad + x[0,1] \cdot h[1,-1] + x[1,1] \cdot h[0,-1] + x[2,1] \cdot h[-1,-1] \\
&= \quad 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 \\
&\quad + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 \\
&\quad + 4 \cdot (-1) + 5 \cdot (-2) + 6 \cdot (-1) \\
&= -20
\end{aligned}$$

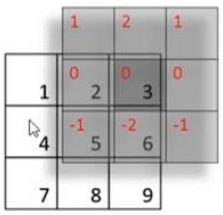
http://www.songho.ca/dsp/convolution/convolution2d_examp

1	2	3
4	5	6
7	8	9

n \ m	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

-13	-20	-17

$$y[2,0] = \sum_j \sum_i x[i,j] \cdot h[2-i, 0-j]$$



$$\begin{aligned}
&= \quad x[1,-1] \cdot h[1,1] + x[2,-1] \cdot h[0,1] + x[3,-1] \cdot h[-1,1] \\
&\quad + x[1,0] \cdot h[1,0] \quad + x[2,0] \cdot h[0,0] \quad + x[3,0] \cdot h[-1,0] \\
&\quad + x[1,1] \cdot h[1,-1] + x[2,1] \cdot h[0,-1] + x[3,1] \cdot h[-1,-1] \\
&= \quad 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 \\
&\quad + 2 \cdot 0 + 3 \cdot 0 + 0 \cdot 0 \\
&\quad + 5 \cdot (-1) + 6 \cdot (-2) + 0 \cdot (-1) \\
&= -17
\end{aligned}$$

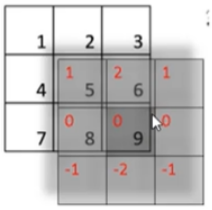
http://www.songho.ca/dsp/convolution/convolution2d_exam

1	2	3
4	5	6
7	8	9

n \ m	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

-13	-20	-17
-18	-24	-18
13	20	17

$$y[2,2] = \sum_j \sum_i x[i,j] \cdot h[2-i, 2-j]$$



$$\begin{aligned}
&= \quad x[1,1] \cdot h[1,1] \quad + x[2,1] \cdot h[0,1] \quad + x[3,1] \cdot h[-1,1] \\
&\quad + x[1,2] \cdot h[1,0] \quad + x[2,2] \cdot h[0,0] \quad + x[3,2] \cdot h[-1,0] \\
&\quad + x[1,3] \cdot h[1,-1] + x[2,3] \cdot h[0,-1] + x[3,3] \cdot h[-1,-1] \\
&= \quad 5 \cdot 1 + 6 \cdot 2 + 0 \cdot 1 \\
&\quad + 8 \cdot 0 + 9 \cdot 0 + 0 \cdot 0 \\
&\quad + 0 \cdot (-1) + 0 \cdot (-2) + 0 \cdot (-1) \\
&= 17
\end{aligned}$$

http://www.songho.ca/dsp/convolution/convolution2d_example

$$(3 \times 1) + (5 \times 1) + (7 \times 1) + (7 \times 0) + (0 \times 0) + (6 \times 0) + (1 \times -1) + (8 \times -1) + (9 \times -1)$$

$$= 3 + 5 + 7 + 0 + 0 + 0 - 1 - 8 - 9 = -3$$

$$(7 \times 1) + (0 \times 1) + (6 \times 1) + (1 \times 0) + (8 \times 0) + (9 \times 0) + (2 \times -1) + (3 \times -1) + (7 \times -1)$$

$$= 7 + 1 + 6 + 0 + 0 + 0 - 2 - 3 - 7 = 2$$

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

*

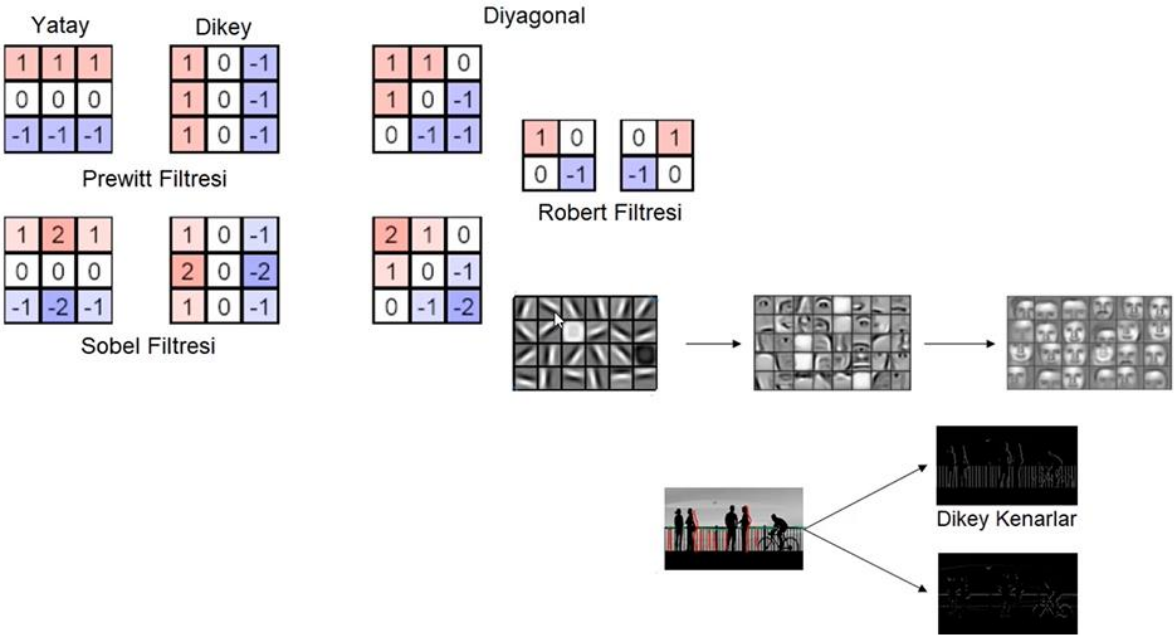
1	0	-1
1	0	-1
1	0	-1

=

-3	2	2	-2	3		
2						

KENAR BULMA (Edge Detection)

Geleneksel yöntemler kenarları Prewit Filtresi, Sobel Fiktresi, Diyagonal, Robert Filtresi gibi matrislerle bu işlemleri yapmaktadır. Fakat Derin Öğrenmede (Evrişimli Sinir Ağları) böyle bir şey yapmamıza gerek yoktur.



Piksel Ekleme (Padding)

Giriş matrisiyle çıkış matrisinin eşit boyutta olmasını istenildiği için, giriş matrisine dolgulama (padding) yapılır.

$giriş = (n \times n) = (6 \times 6)$
$filtre = (f \times f) = (3 \times 3)$
Piksel ekleme var $\rightarrow$ $çıkış = (n + 2p - f + 1) \times (n + 2p - f + 1) = (6 \times 6)$
Piksel ekleme yok $\rightarrow$ $çıkış = (n - f + 1) \times (n - f + 1) = (4 \times 4)$

$$p = \frac{(f-1)}{2}$$

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

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

Adım Kaydırma (Stride)

Bu işlemde, filtreyi görüntü matrisi üzerinde evrişim işlemine alırken filtreyi kaç piksel aralıklarla kaydıracağımızın değeridir. Stride, çıkış matrisinin boyutunu değiştirir.

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

$$s = 2 \rightarrow Adım\ Kaydırma\ (stride)$$

$$\left(\frac{(n+2p-f)}{s}+1\right)\times\left(\frac{(n+2p-f)}{s}+1\right)$$

$$n = 9\times 9$$

$$f = 3\times 3$$

$$\left(\frac{(9+2.0-3)}{2}+1\right)\times\left(\frac{(9+2.0-3)}{2}+1\right)$$

$$= (5\times 5)\ Çıkış\ matirisi\ boyutu$$

$n = 5\times 5$
$f = 3\times 3$
$s = 2$
$p = 1$

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

5	13	14
17	42	35
16	32	15

3x3

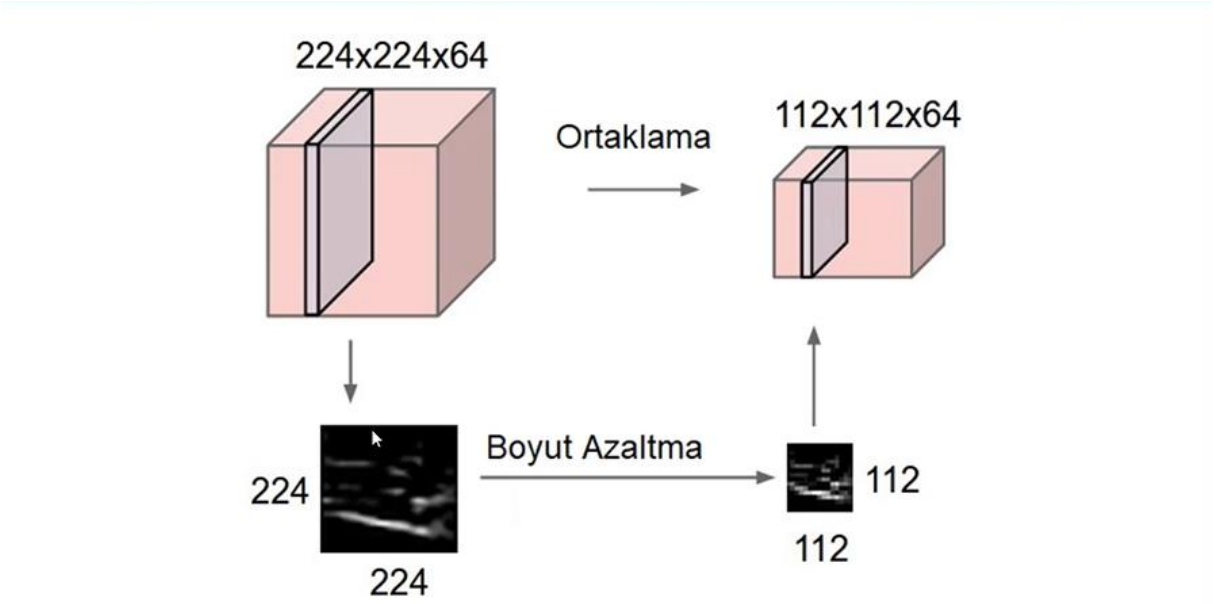
$$\left(\frac{(n+2p-f)}{s}+1\right)\times\left(\frac{(n+2p-f)}{s}+1\right) = \left(\frac{(5+2.1-3)}{2}+1\right)\times\left(\frac{(5+2.1-3)}{2}+1\right) = (3)\times(3)$$

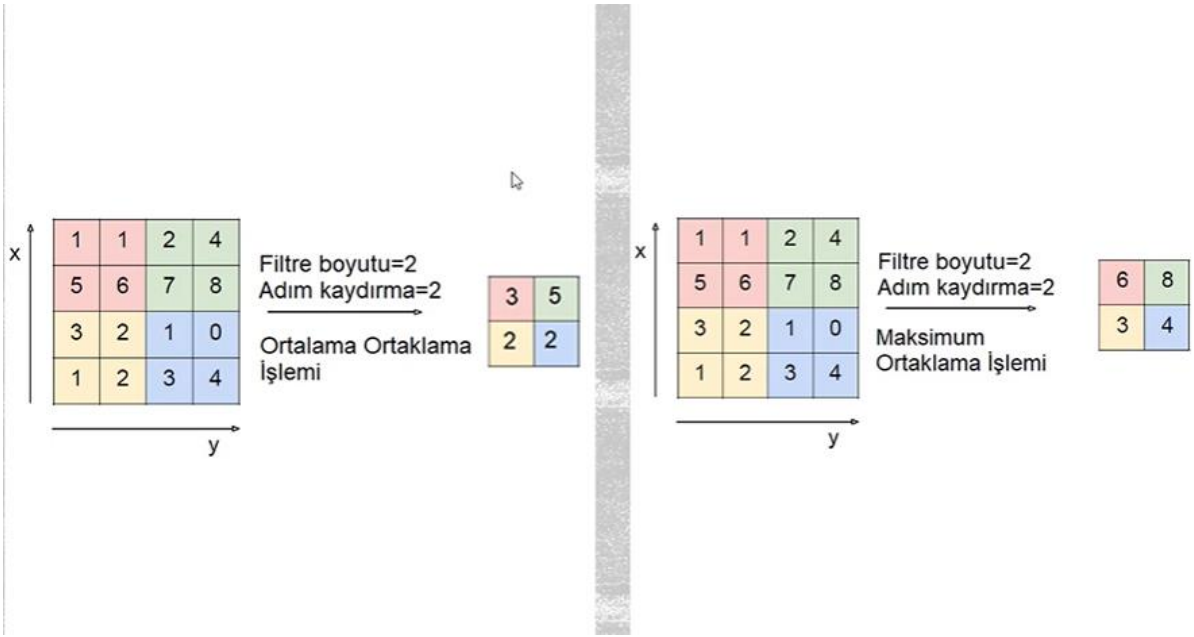
$n = 5\times 5$
$f = 3\times 3$
$s = 2$
$p = 3$

$$\left(\frac{(5+2.3-3)}{2}+1\right)\times\left(\frac{(5+2.3-3)}{2}+1\right) = (5)\times(5)$$

ORTAKLAMA (POOLING)

Evrişim ağlarından çıkan öznitelik haritalarının boyutunu küçültmek için kullanılan bir işlemdir.





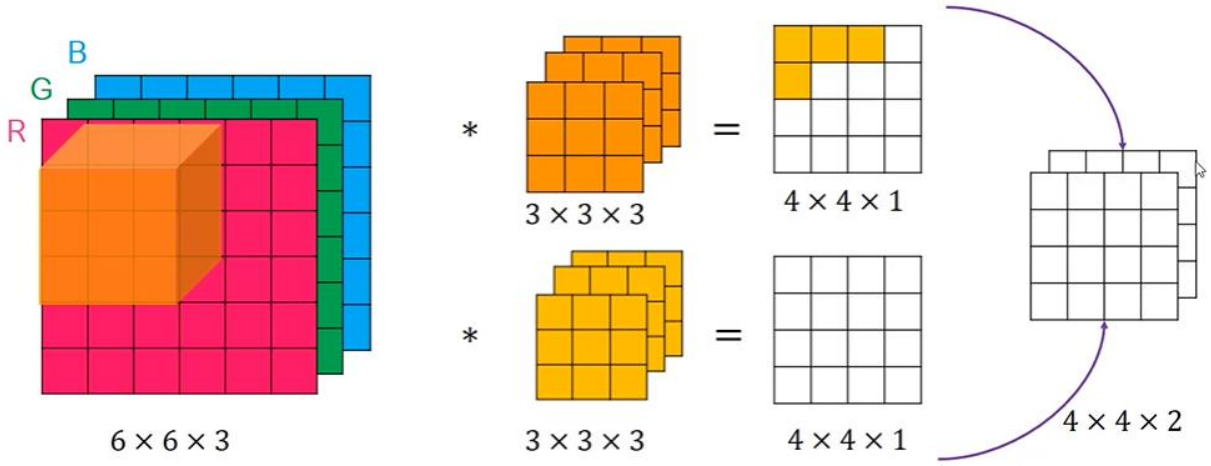
ÇOK KANALLI EVRİŞİM İŞLEMİ (MULTI CHANNEL CONVOLOTION OPERATION)

Bir matris birden çok kanaldan oluşuyorsa buna tensör denmektedir.

Evrişimli Sinir Ağlarında birden çok çıkışlı kanallara tensör denmektedir.

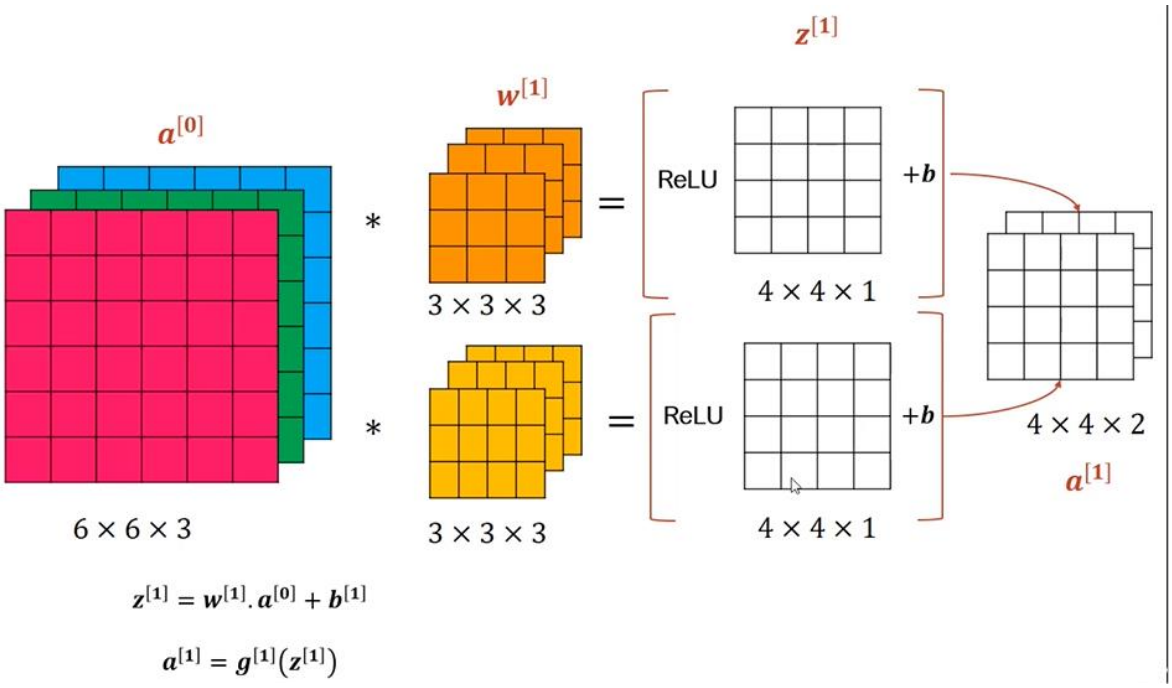
Görüntü kanal sayısı ile, filtrenin kanal sayısı eşit olmalıdır.

Her filtre, evrişimle görüntü üzerinde ayrı ayrı işleme alınmaktadır. Sonrasında çıkışlar birleştirilmektedir. (Çıkış Tensörü, bir öznitelik çıkartma işlemi gerçekleştirmiş oldu.)



$$n \times n \times \boldsymbol{n_C} * f \times f \times \boldsymbol{f_c} = n - f + 1 \times n - f + 1 \times n_C'$$

TEK KATMANLI EVRİŞİMLİ SİNİR AĞI



Bu filtreler, özniteliklerin hesaplanmasıdır. (renk bulma, doku bulma, ışık gibi bilgileri bu filtreler sayesinde bulabiliyoruz.)

Filtre boyutu : $f^{[l]}$
Piksel doldurma : $p^{[l]}$
Adım kaydırma : $s^{[l]}$
Filtre (kanal) sayısı : $n_C^{[l]}$

Her bir filtrenin boyutu : $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$

Aktivasyon fonksiyonu : $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$

Ağırlıklar : $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}$

Bias : $n_C^{[l]} \rightarrow \left(1, 1, 1, n_C^{[l]}\right)$

Giriş : $n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$

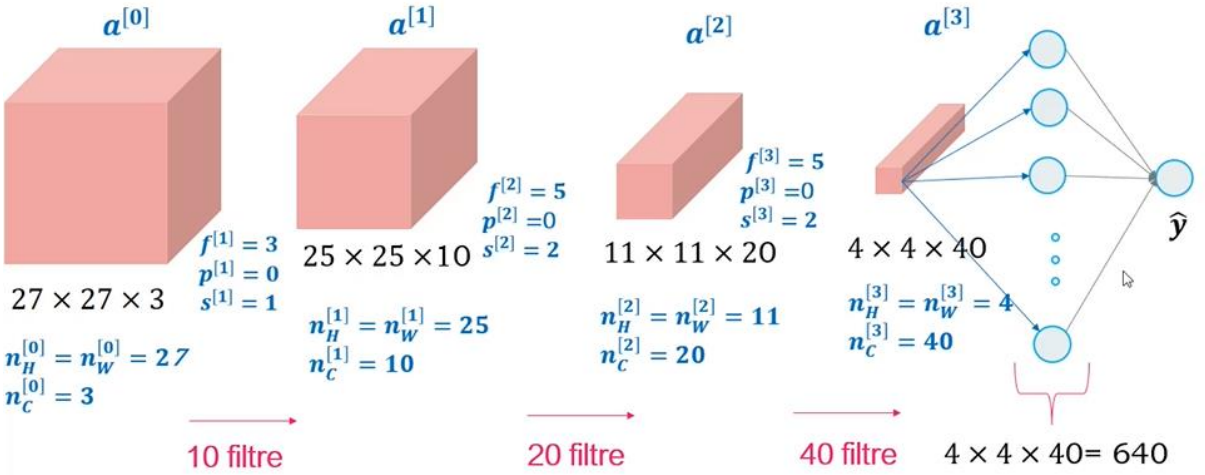
Çıkış : $n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$

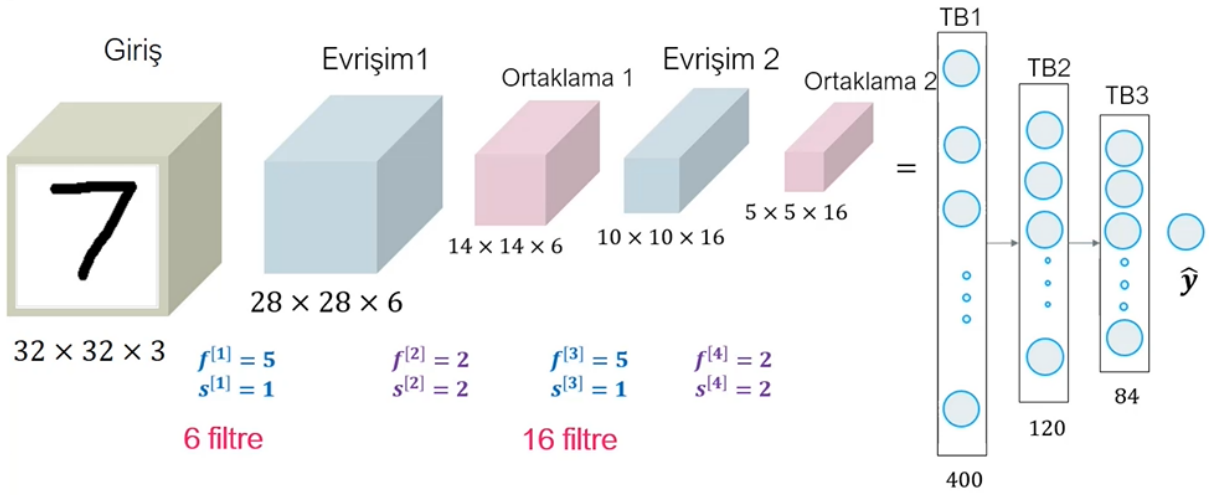
$n_{H,W}^{[l]} : \left[\frac{n_H^{[l-1]} + 2p - f^{[l]}}{s^{[l]}} + 1\right]$

Bir Evrişimli Sinir Ağı için Gereken Katmanlar

- Evrişim Katmanı (Aktivasyon fonksiyonu, Bias değeri)
- Ortaklama Katmanı (Maksimum ya da ortalama ortaklama)
- Tam/Tüm Bağlantı Katmanı (Klasik yapay sinir ağı bağlantıları)

Evrişimli Sinir Ağı Örnekleri





Çıkışta 10 sınıflı rakam tanıma yapılacaksa: softmax(10)

$n_H, n_W \downarrow$
$n_C \uparrow$

LeNet

LeNet 5 – 1980'lerden - 1998

PROC. OF THE IEEE, NOVEMBER 1998

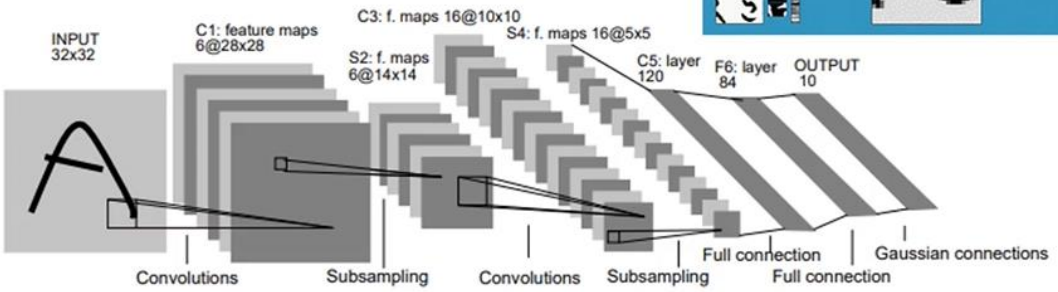


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

AlexNet

AlexNet - 2012

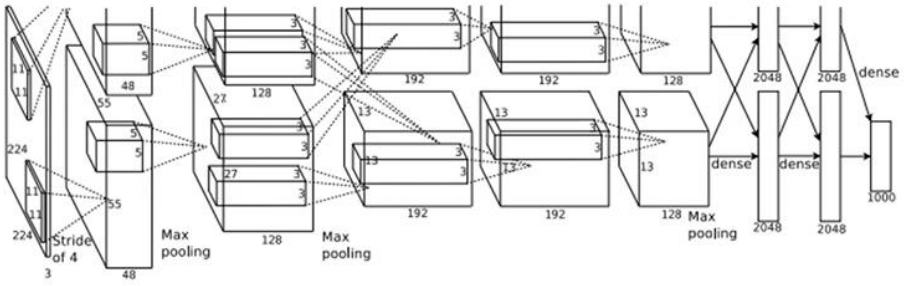
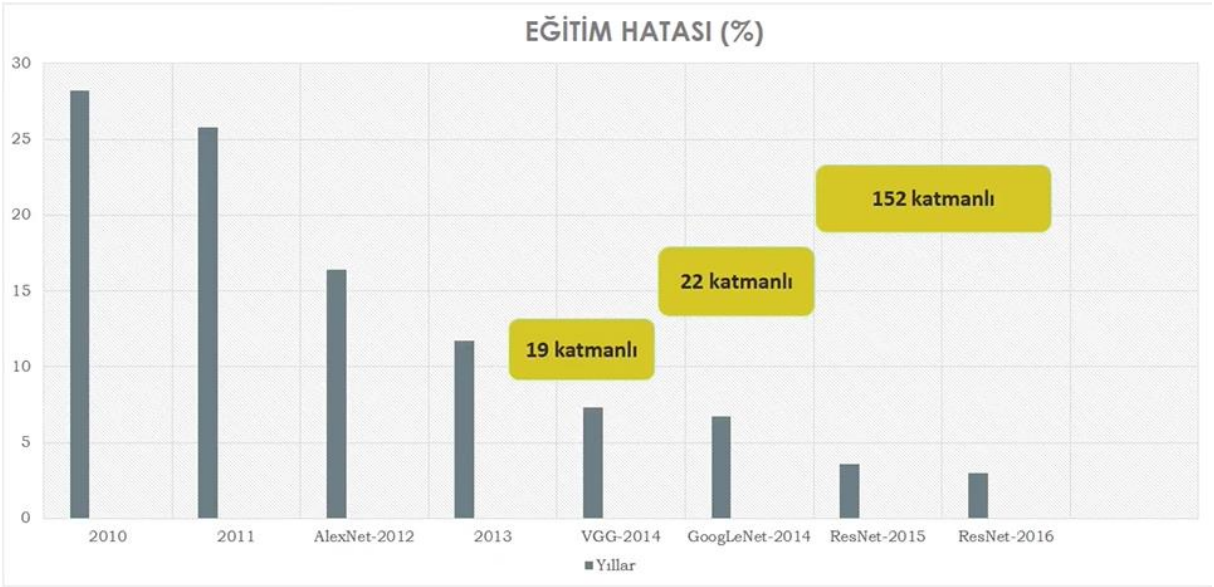


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf



VGG16

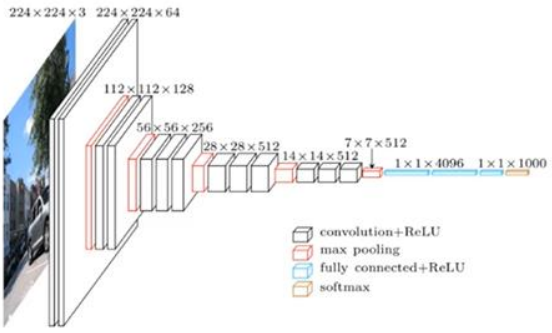
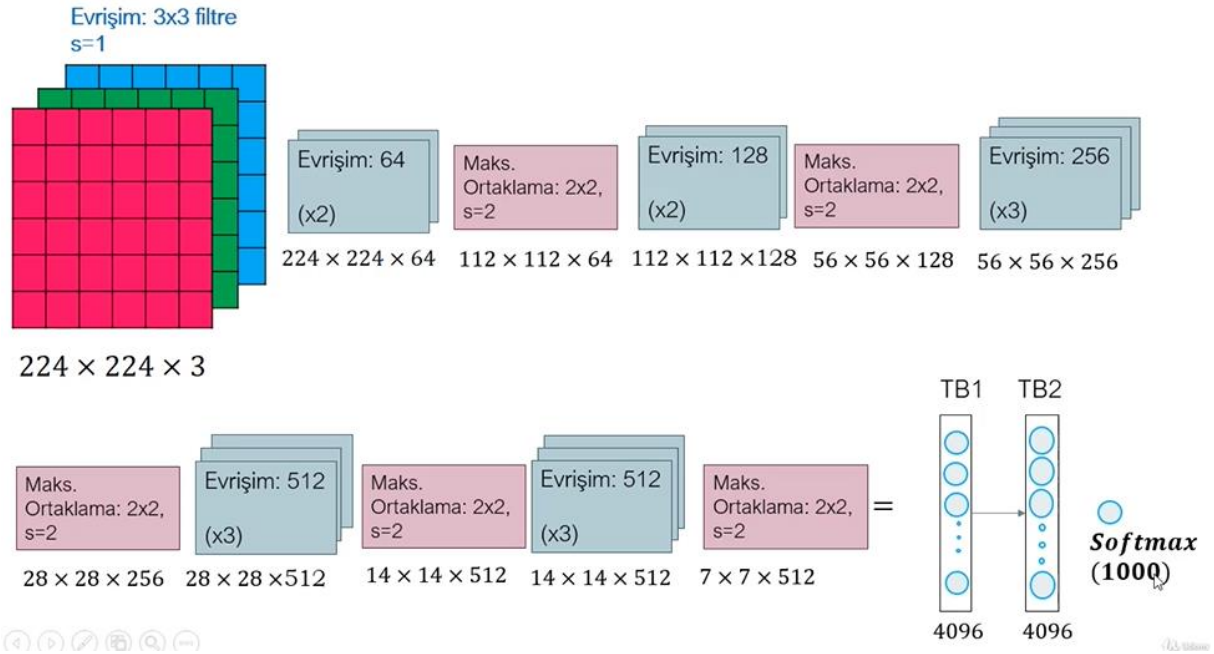
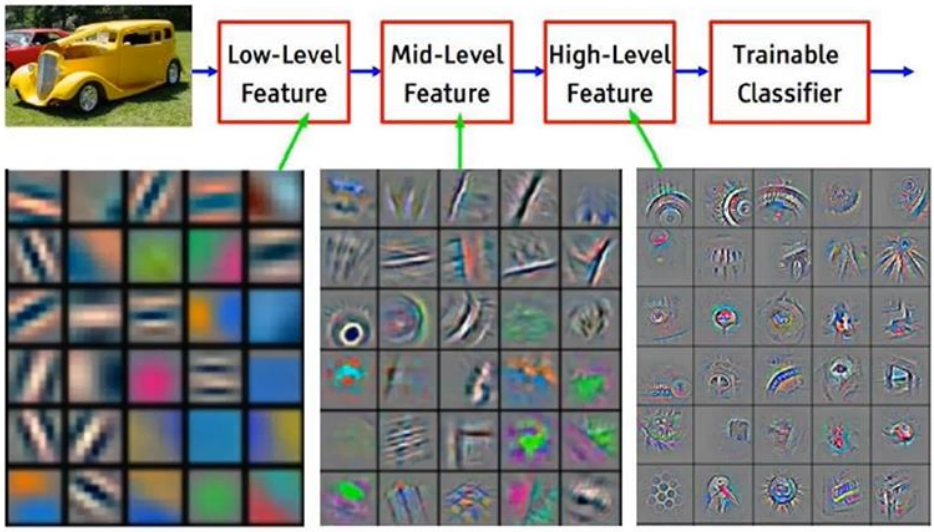


Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

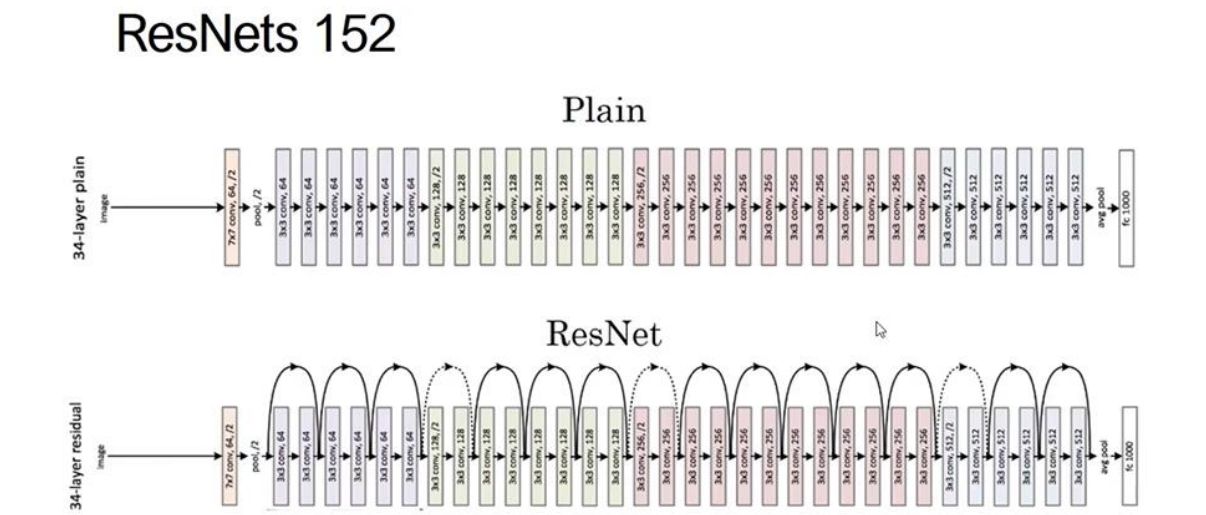
https://arxiv.org/pdf/1409.1556.pdf



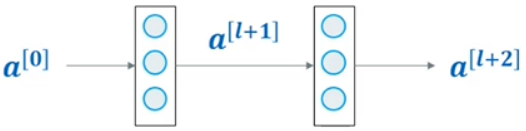


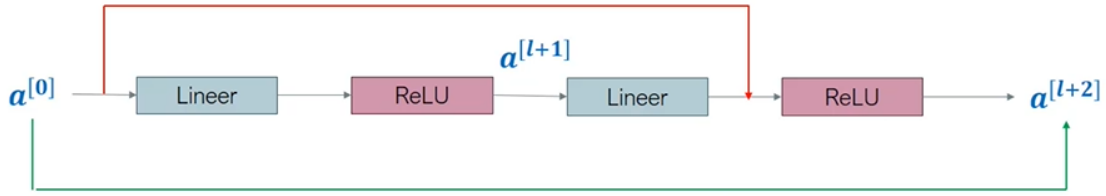
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

ResNet



https://arxiv.org/pdf/1512.03385.pdf





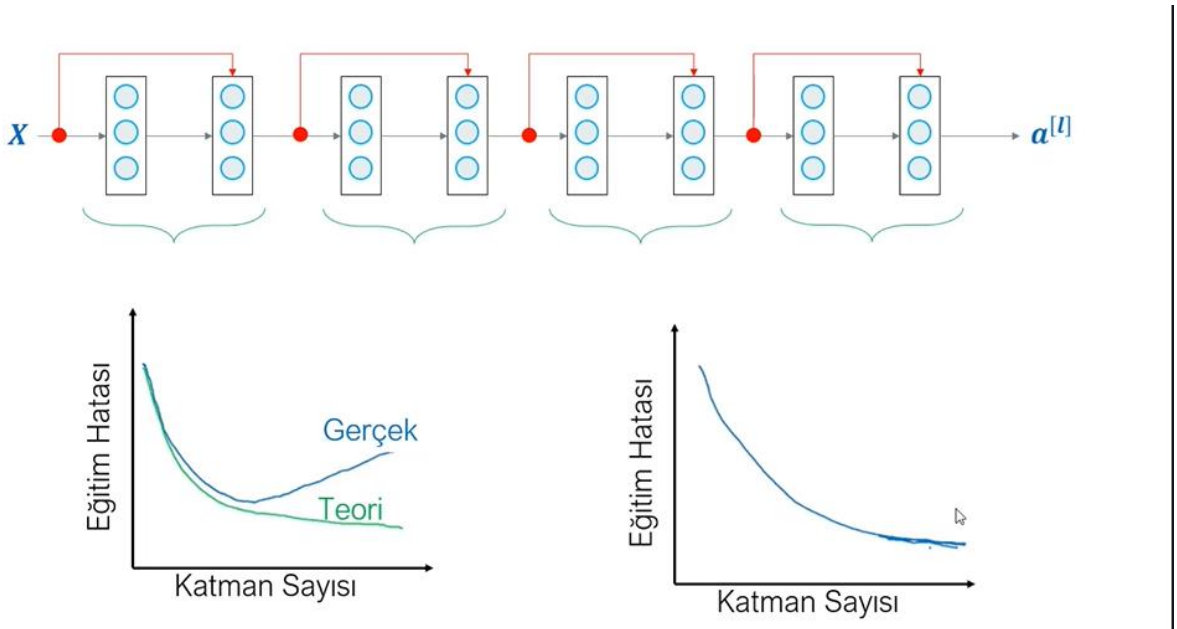
$$z^{[l+1]} = W^{[l+1]}\, a^{l} + b^{[l+1]} \qquad z^{[l+2]} = W^{[l+2]}\, a^{[l+1]} + b^{[l+2]}$$

$$a^{[l+1]} = g(z^{[l+1]}) \qquad a^{[l+2]} = g(z^{[l+2]}) \qquad a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

a Değeri Nedir?

Bir sinir ağı katmanında, girdileri ağırlıklarla çarpıp bias değerini ekleyerek bir ön hesaplama yaparız; buna z değeri denir.

Bu z değerine, ReLU gibi bir aktivasyon fonksiyonu uyguladığımızda elde ettiğimiz sonuca ise a değeri denir. Bu a değeri, sonraki katmana gönderilen bilgi veya nihai çıktı olur.



https://arxiv.org/pdf/1512.03385.pdf

Normalde (ResNet'den önce), daha fazla katman koyduğumuzda hata miktarı artıyor, model öğrenemez hale geliyordu. ResNet bu soruna çözüm bulmuştur. Daha fazla katmanla daha iyi öğrenmenin yolunu açmıştır.

ResNet, Vanish Gradient (Gradyanların yok olması) problemine çözüm getirmiştir. (Gradyanlar kaybolursa öğrenme azalır, 1. Grafikteki gibi teori yerine gerçek grafik çizgisi haline evrilir.)

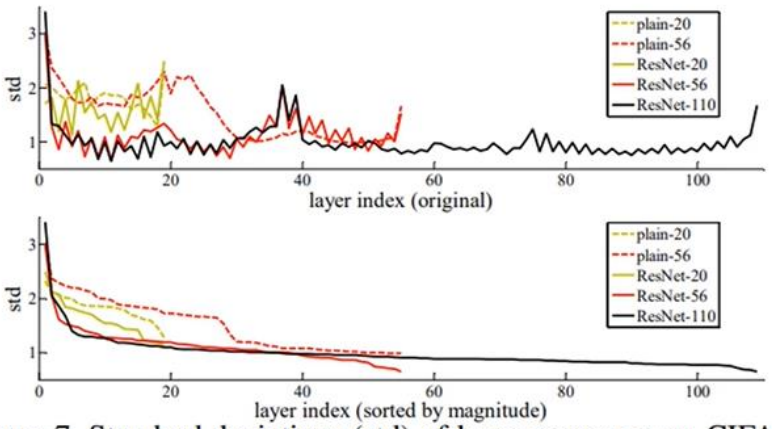
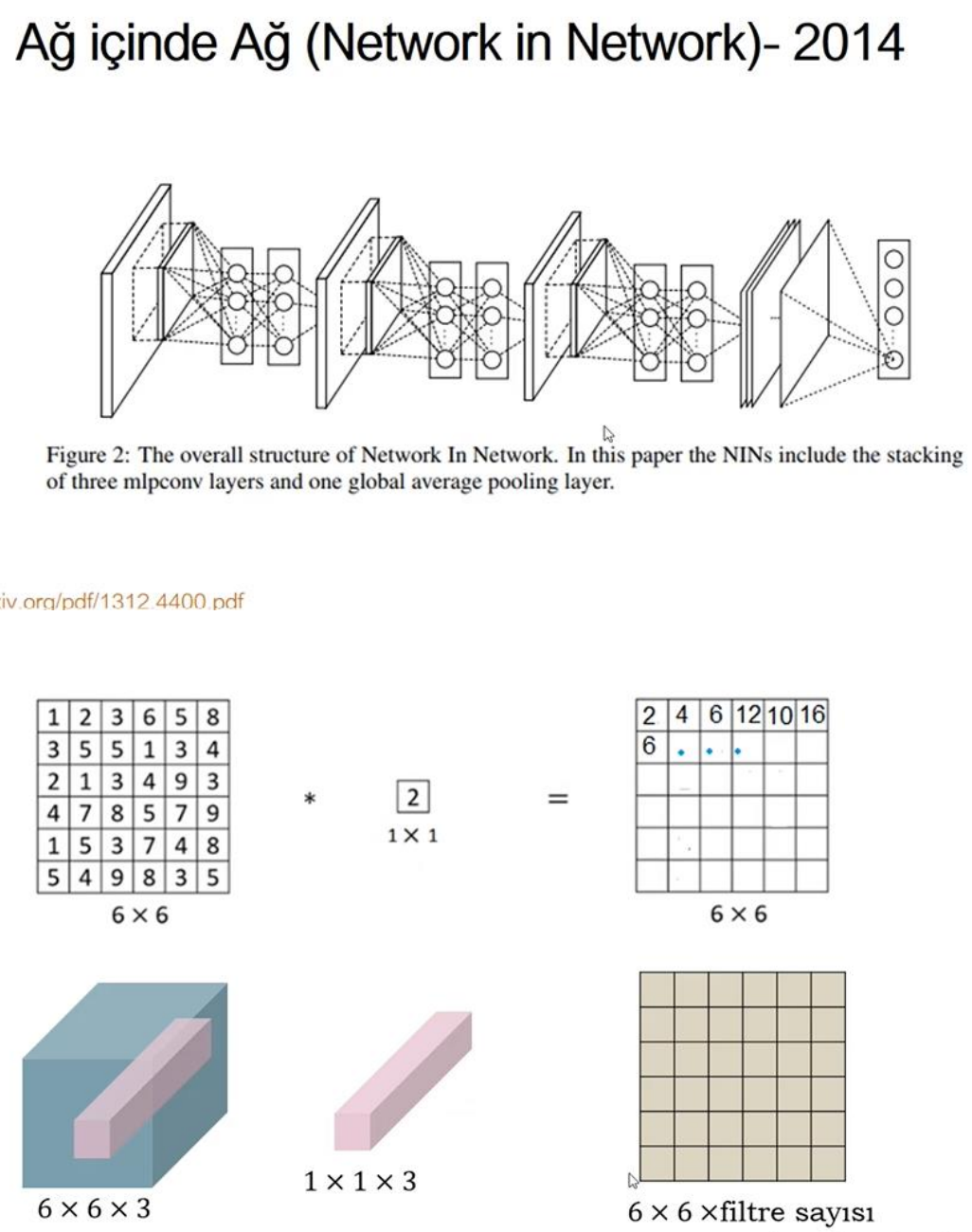


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

https://arxiv.org/pdf/1512.03385.pdf

Ağ İçinde Ağ (Network in Network)

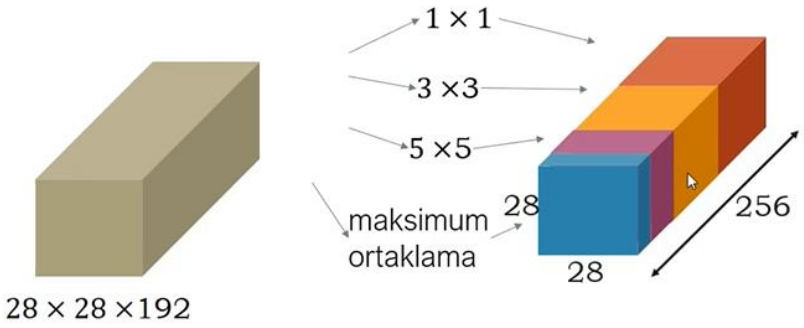


Bire bir evrişim işlemi bir şey ifade etmezken, bunu tensör olarak ifade ettiğimiz zaman parametre hesabı bakımından baktığımızda 1x1 filtre fark yaratmaktadır. Bir önceki katmana bakarak hesap yaptığımız için, boyut azaltmış (dimensionality reduction) olmaktayız.

1x1 evrişim filtresi, resmin veya bilginin büyüklüğünü (eni-boyu) değil, derinliğini (kanal sayısını) kontrol eder. Yani, kanalları azaltarak boyutu küçültmemizi sağlar.

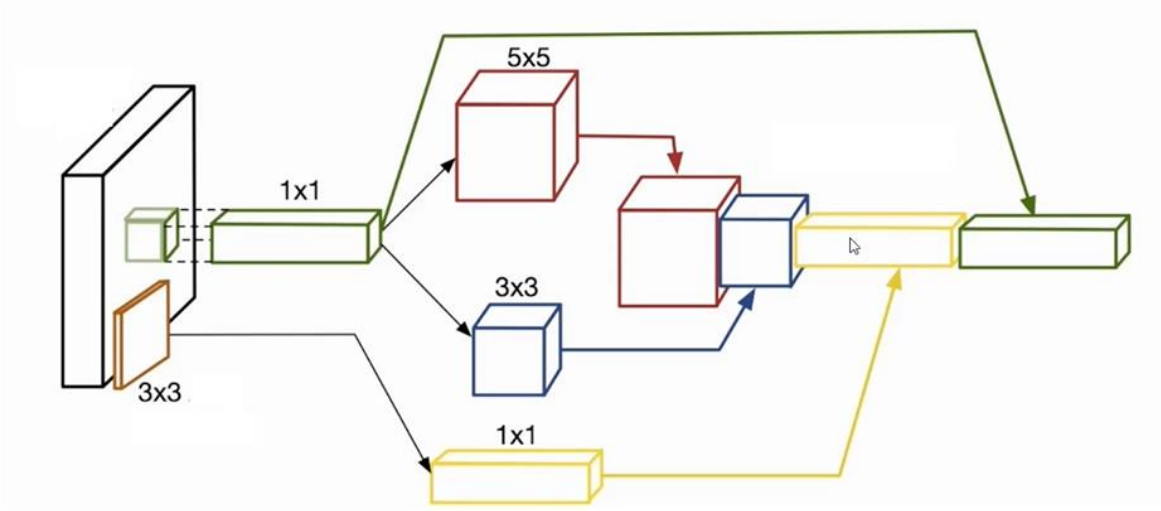
Normalde evrişim işlemlerinde filtre boyutumuzu 5x5, 11x11 gibi tek değerler seçebiliyorduk ama artık parametre sayımızı azaltabilir olduğumuza göre birden fazla

filtreyi aynı anda uygulayıp daha az sayıda parametre hesabı yapmanın yolunu veriyor bize inception yöntemi.



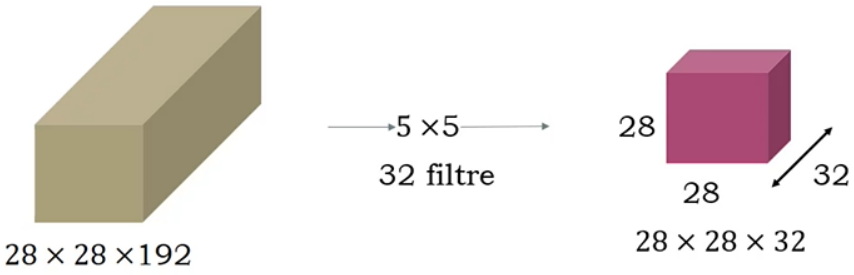
https://arxiv.org/pdf/1312.4400.pdf

https://www.deeplearning.ai/



Bunu yapmamızdaki amaç ise

Yalnızca bu işlem adımı için (28x28x32)x(5x5x192)=120 milyon parametre hesaplanması gerekir.



https://arxiv.org/pdf/1312.4400.pdf

https://www.deeplearning.ai/

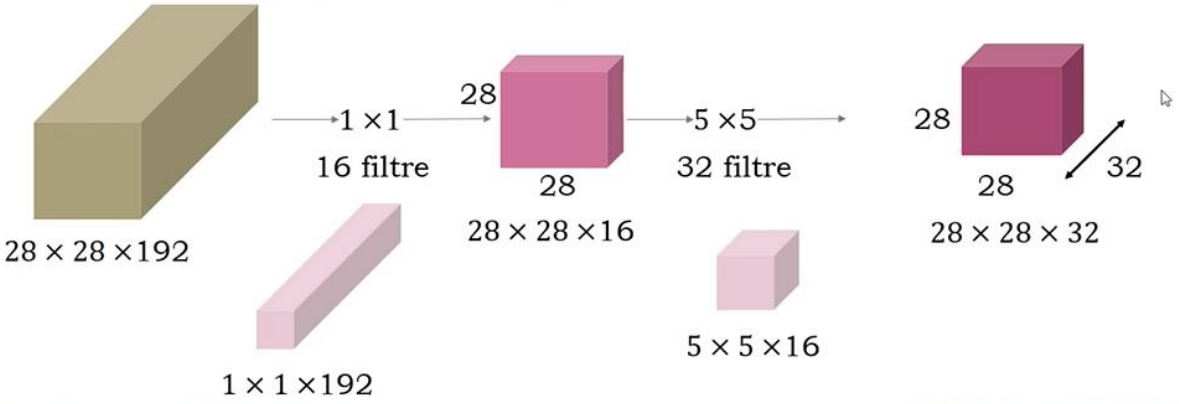
Bu koşulda; 1x1 evrişim katmanında: (28x28x16)x(1x1x192)=2,4 milyon parametre

5x5 evrişim katmanında: (28x28x32)x(5x5x16)=10 milyon parametre

Toplamda 12.4 milyon parametre

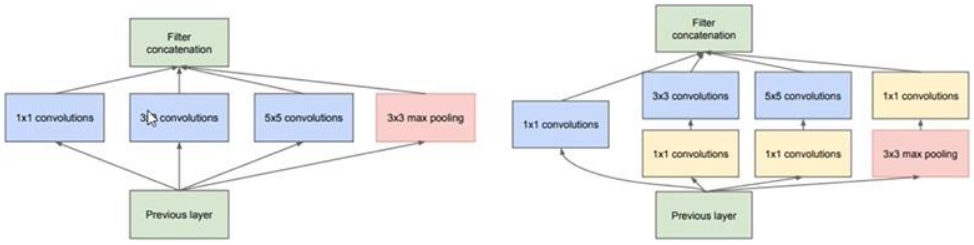
~10 Kat Daha Az!

İlk duruma göre yaklaşık **10 kat daha az** parametre hesabı son derece çarpıcıdır.



https://arxiv.org/pdf/1312.4400.pdf

https://www.deeplearning.ai/



(a) Inception module, naïve version (b) Inception module with dimension reductions

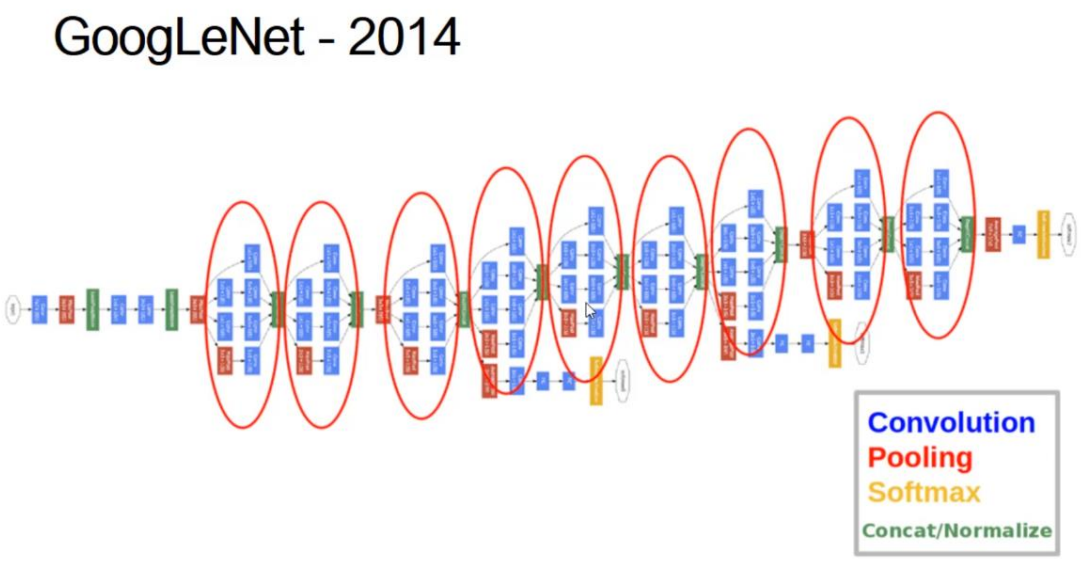
Figure 2: Inception module

https://arxiv.org/pdf/1312.4400.pdf

Inception modeli, daha derin olmasına rağmen daha az sayıda parametre hesabı barındırır.

1x1 evrişim (1x1 convolution) işlemi, derin öğrenme mimarilerinde "darboğaz katmanı" (bottleneck layer) olarak adlandırılır.

GoogleNet



https://arxiv.org/pdf/1409.4842.pdf

GoogleNet, her 5x5, 11x11 vs. filtelerden önce; 1x1 darboğaz işlemini 9 kez kullanır. (9 tane inception bloğundan oluşur)

Aktivasyon fonksiyonu olarak RELU kullanılır.

Ortaklama olarak Max. Pooling işlemi uygulanır.

Son çıkıştan önce, 4. Ve 6. Katmanda softmax çıkışları bulunur.

Inception Model Örneği