# Convolutional Neural Networks


Alireza Akhavan Pour

AkhavanPour.ir
CLASS.VISION

Thursday, August 16, 2018

- ❑بخش ۱: شبکه های عصبی کانولوشنالی
- ❑بخش ۲: افزایش دادگان
- ❑بخش ۳: بررسی معماری‌های حائز رتبه در طبقه‌بندی تصاویر
- ❑بخش ۴: انتقال یادگیری



# بخش ۱:

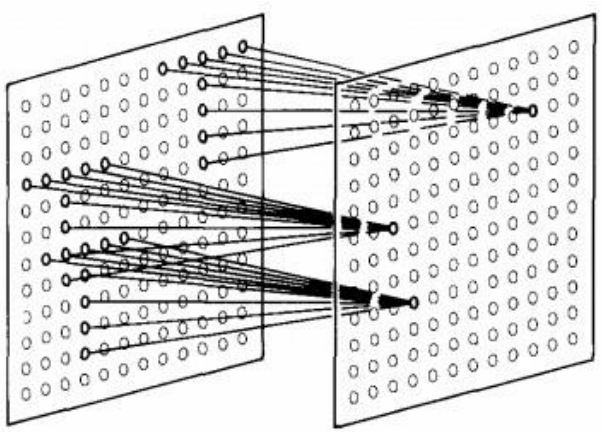
# شبکه های عصبی کانولوشنالی

# Convolutional neural network

A bit of history:

**Neurocognitron**

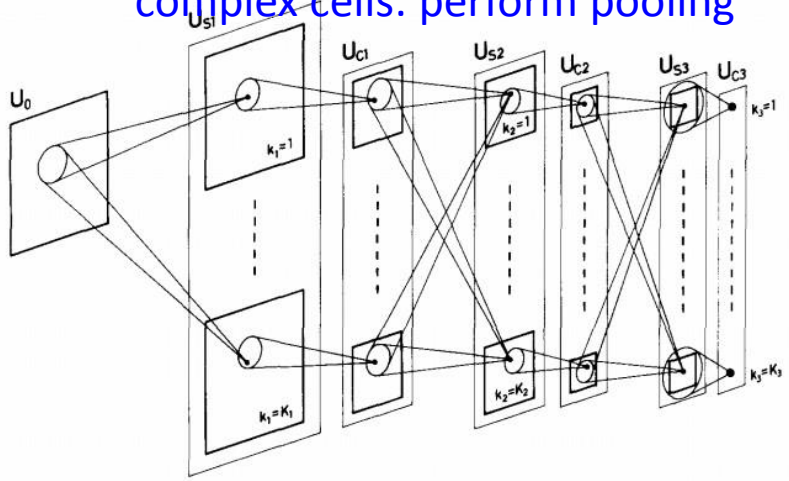
*[Fukushima 1980]*

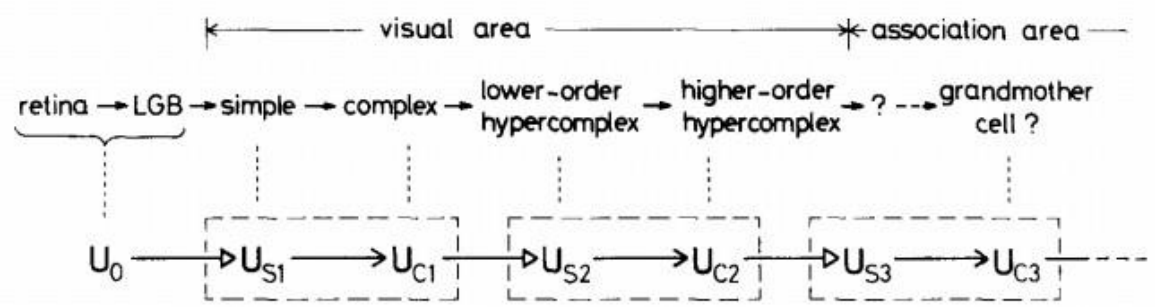


"sandwich" architecture (SCSCSC…)

simple cells: modifiable parameters

complex cells: perform pooling

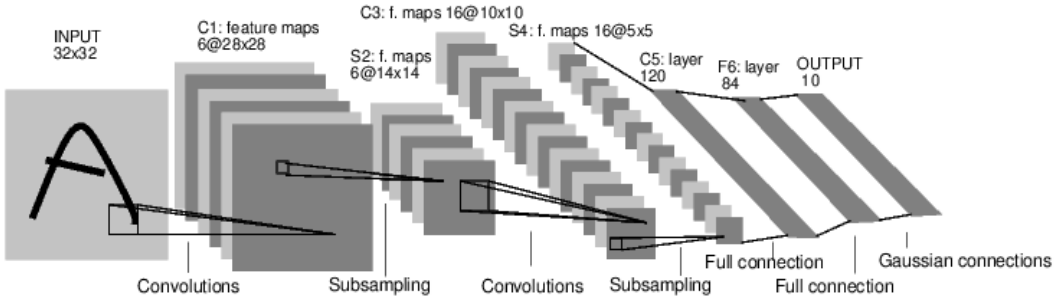


# A bit of history:

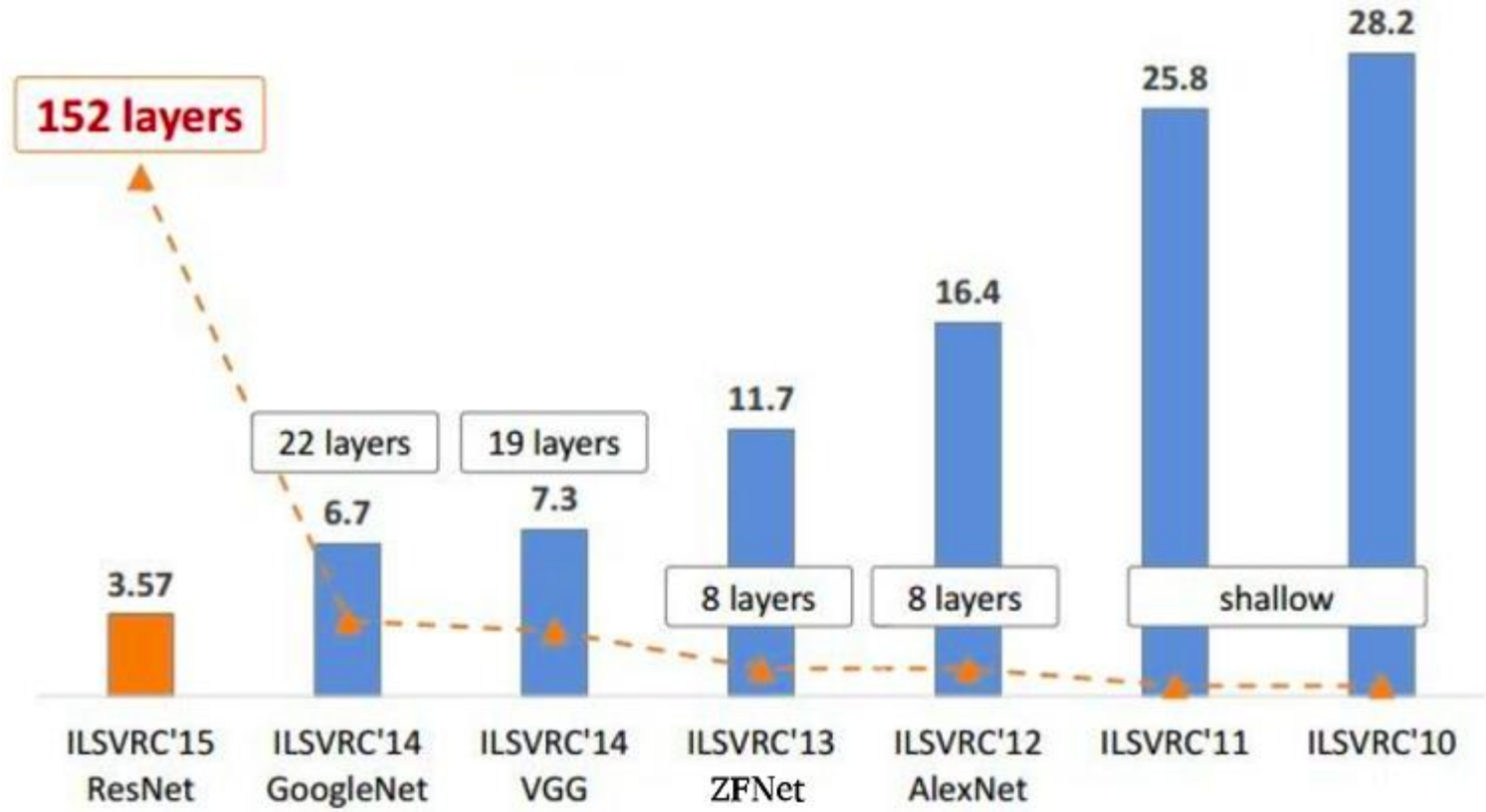
**Gradient-based learning applied to document recognition**

*[LeCun, Bottou, Bengio, Haffner 1998]*

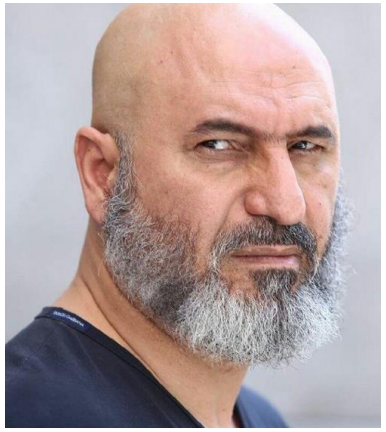


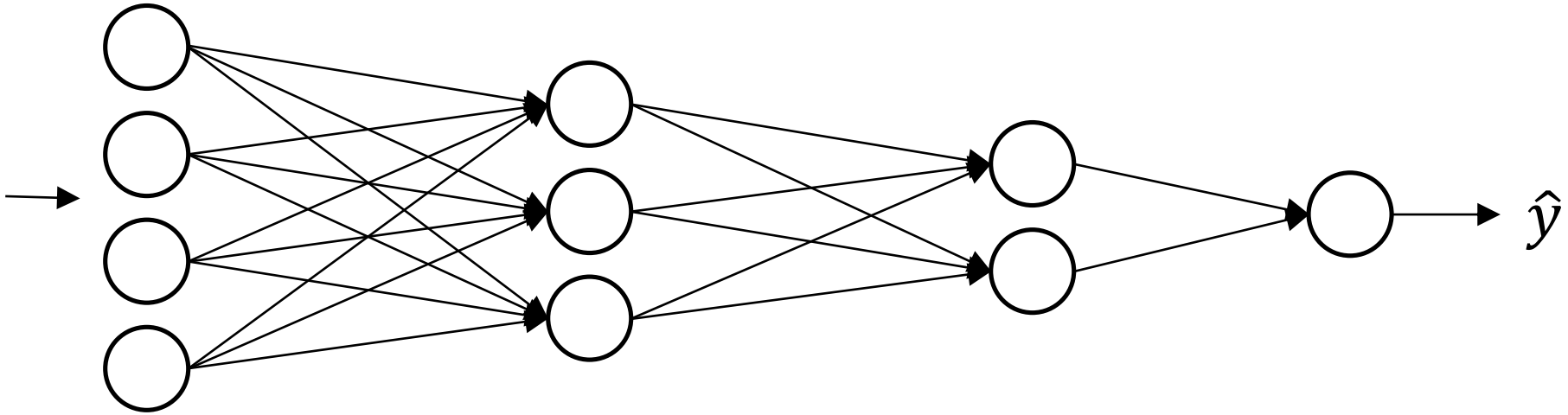

LeNet-5

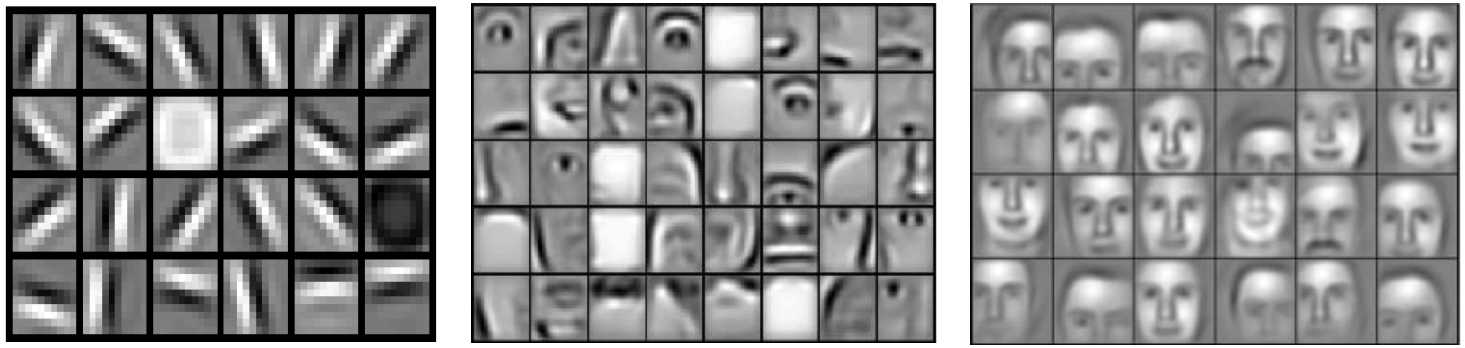
# بعد از ۲۰۱۲...


152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ZFNet
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10

# شهود لزوم سلسله مراتب و عمق شبکه

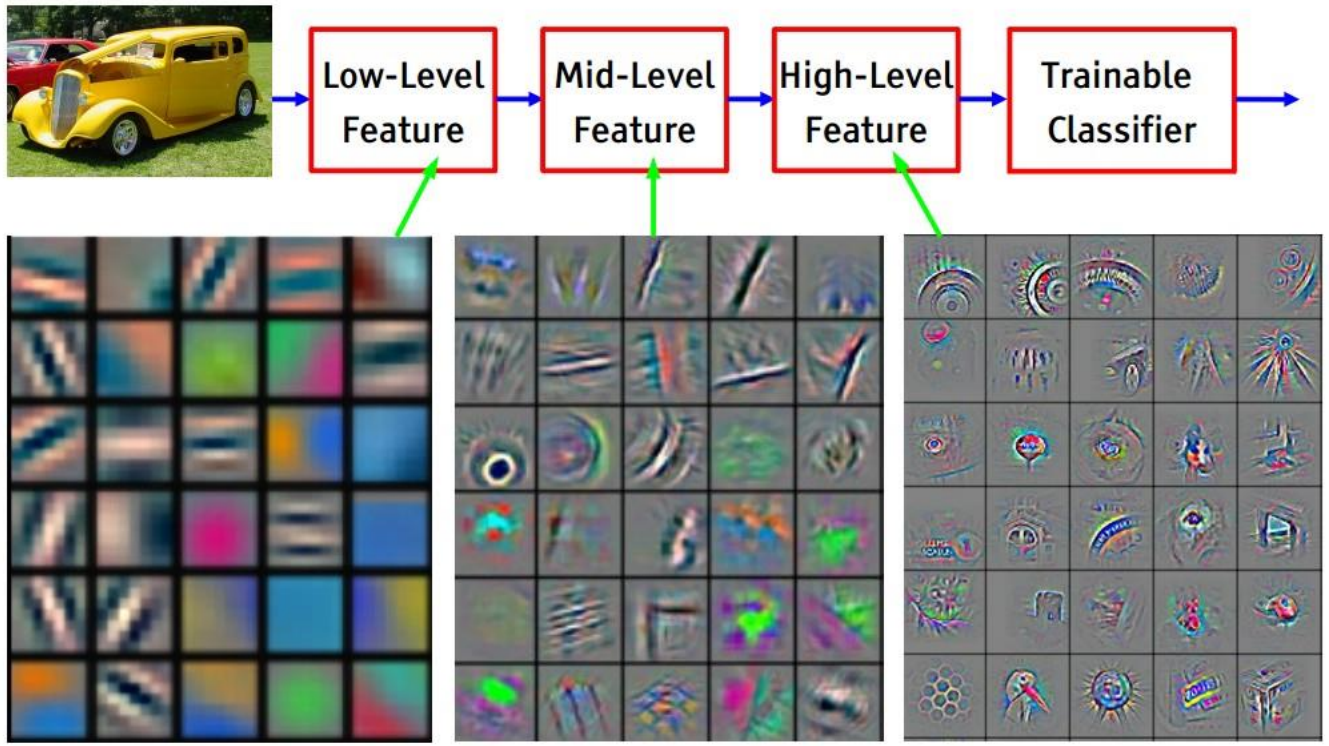




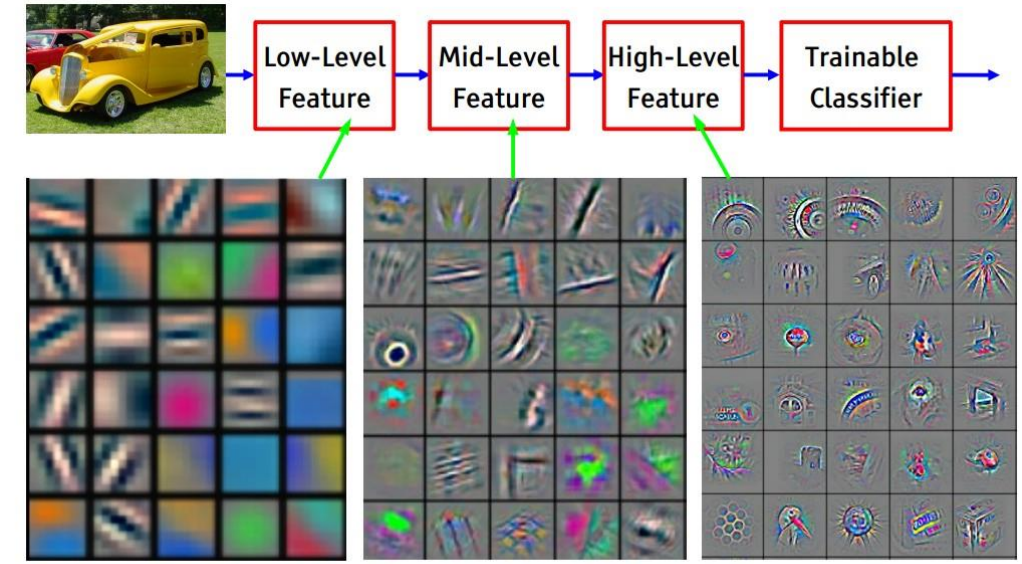
# شهود لزوم سلسله مراتب و عمق شبکه

*[From recent Yann LeCun slides]*



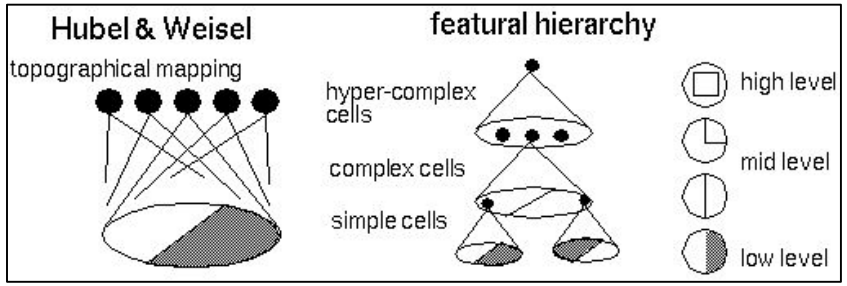

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# شهود لزوم سلسله مراتب و عمق شبکه




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

*[From recent Yann LeCun slides]*

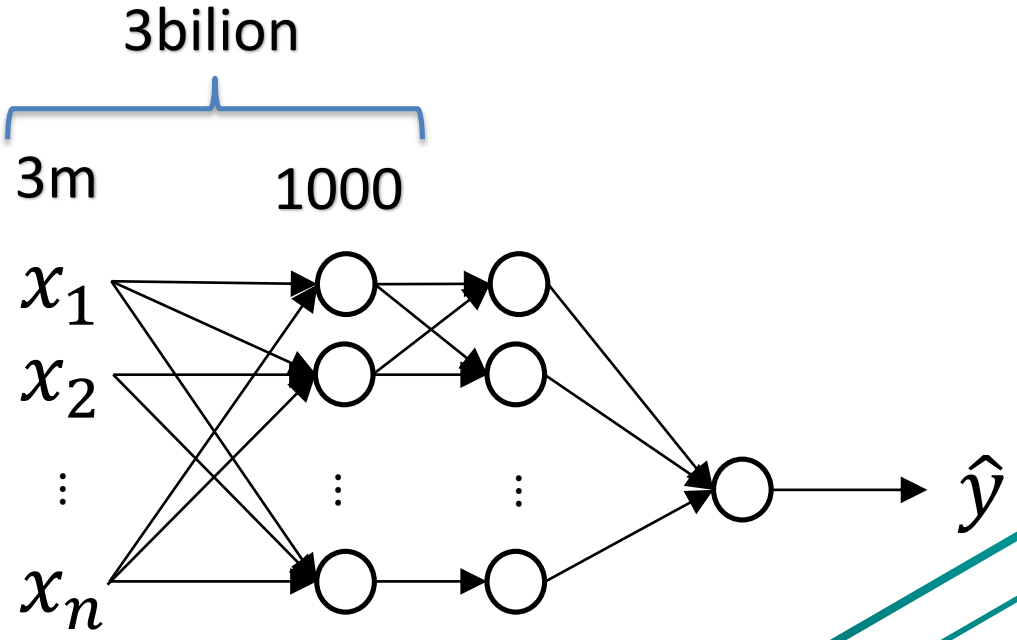


# یادگیری عمیق و تصاویر بزرگ

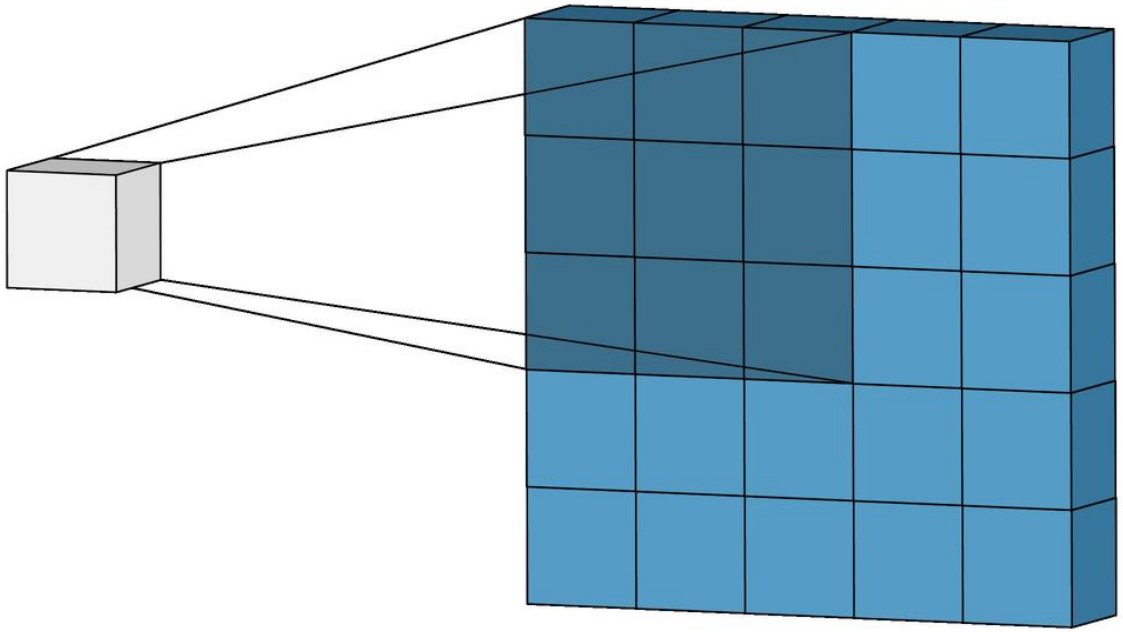
64x64**x3**
**=12288**

1000x1000**x3**
**=3000000 = 3m**

Cat? (0/1)



# مقدمات **convolution** – فیلتر edge detection



| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

# مقدمات convolution – لبه‌یابی

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

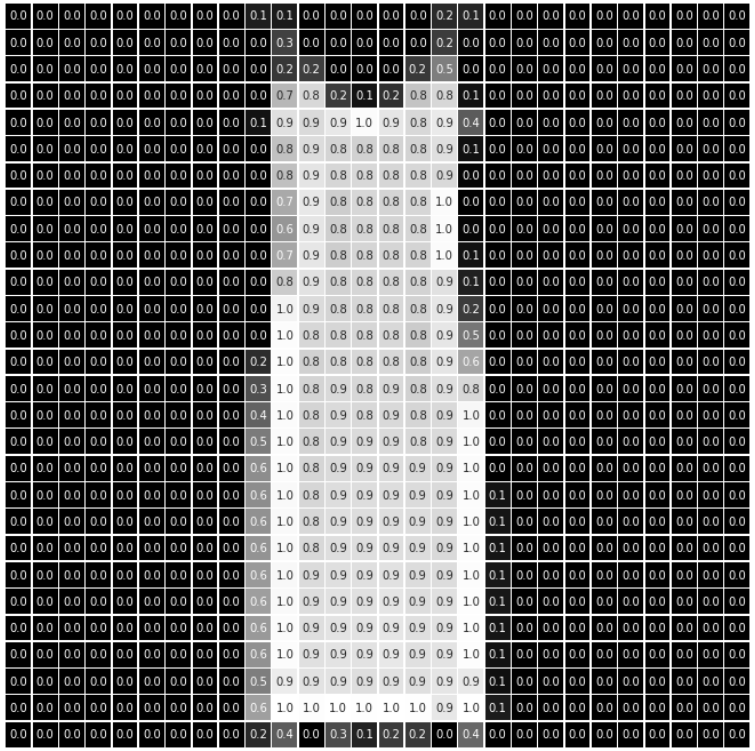
*

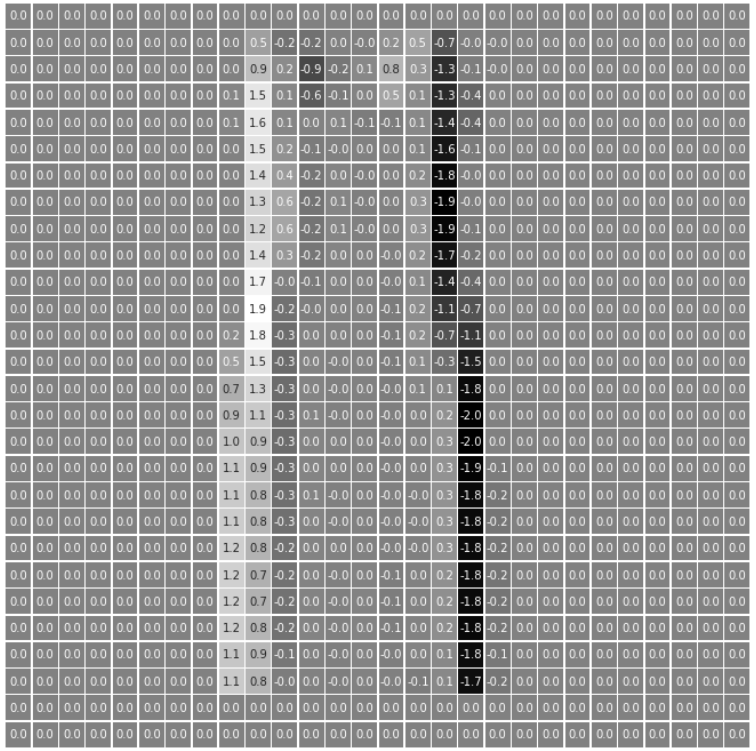
| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

=

| | | | |
|---|---|---|---|
| -5 | -4 | 0 | 8 |
| -10 | -2 | 2 | 3 |
| 0 | -2 | -4 | -7 |
| -3 | -2 | -3 | -16 |

# مقدمات convolution

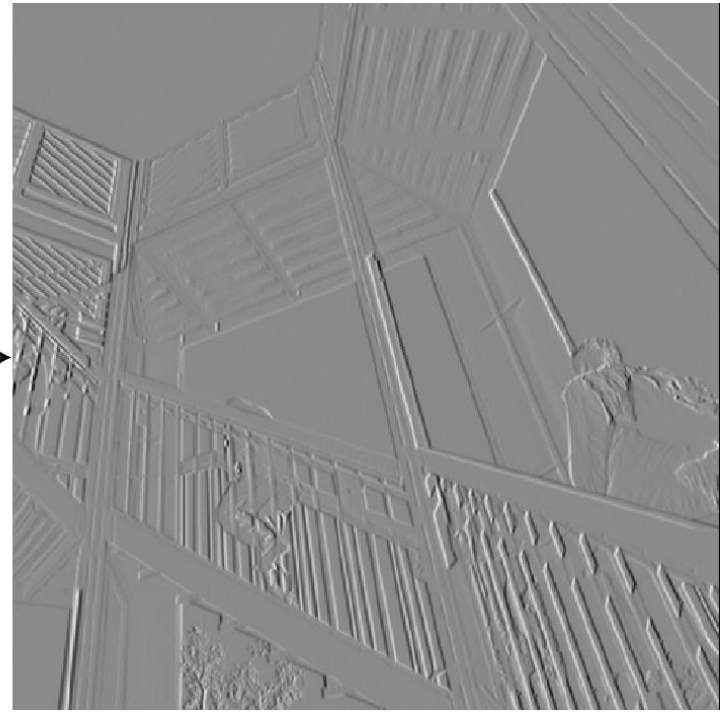


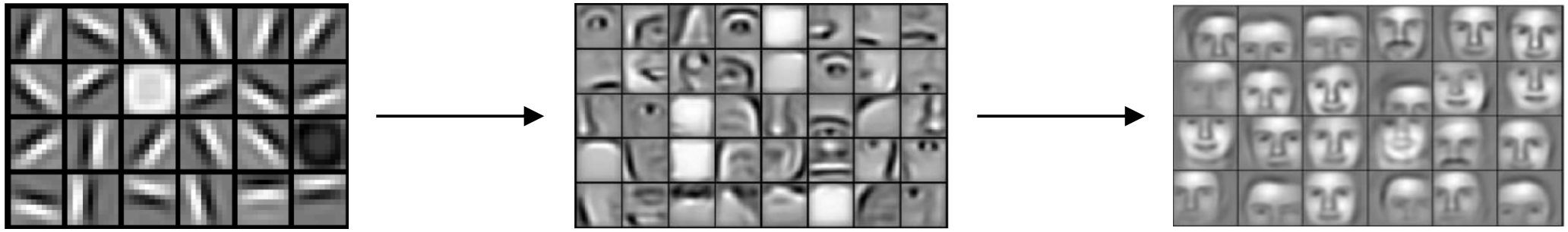
# مقدمات convolution

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Horizontal Sobel kernel



# مقدمات convolution – لبه‌یابی



# یادگیری فیلتر!

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

| | | |
|---|---|---|
| 3 | 0 | -3 |
| 10 | 0 | -10 |
| 3 | 0 | -3 |

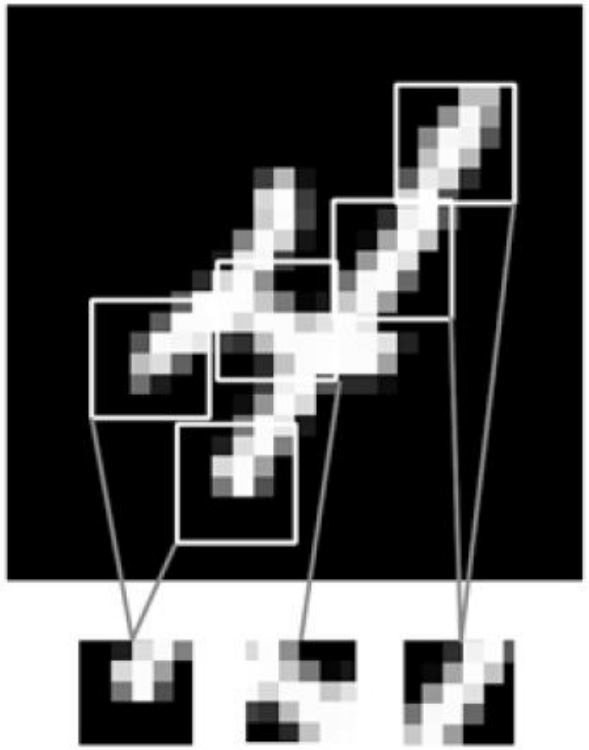
| | | | | | |
|---|---|---|---|---|---|
| 3 | 0 | 1 | 2 | 7 | 4 |
| 1 | 5 | 8 | 9 | 3 | 1 |
| 2 | 7 | 2 | 5 | 1 | 3 |
| 0 | 1 | 3 | 1 | 7 | 8 |
| 4 | 2 | 1 | 6 | 2 | 8 |
| 2 | 4 | 5 | 2 | 3 | 9 |

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

45 ?
70?
73?

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

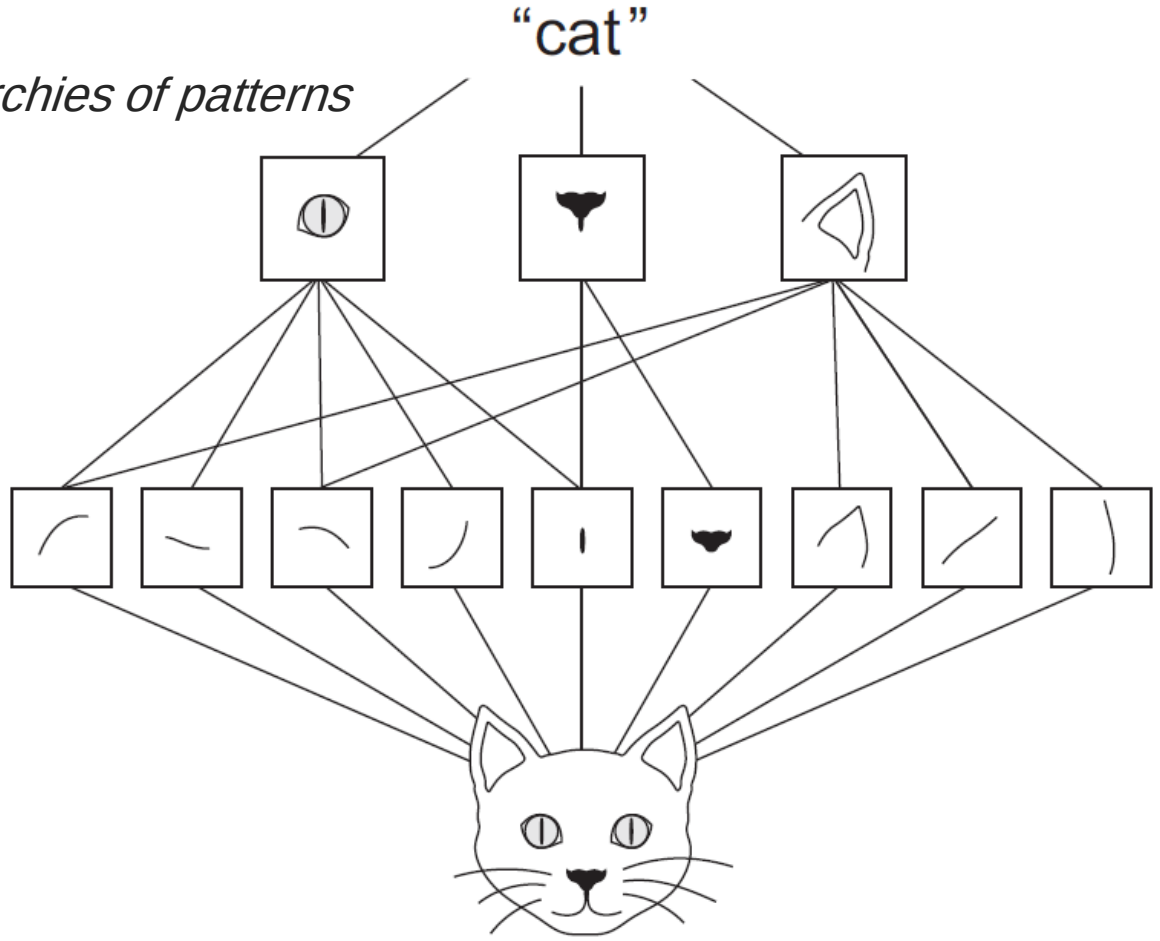
# The convolution operation



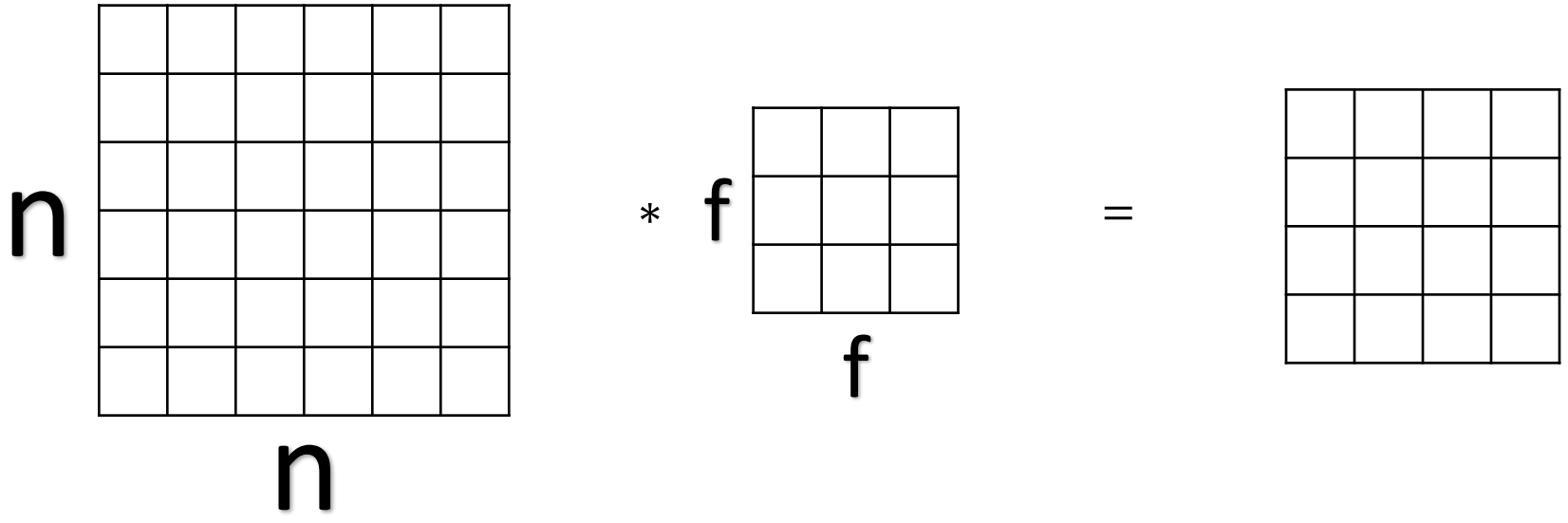
- Dense layers learn global patterns in their input feature space
- Convolution layers learn local patterns

*The patterns they learn are translation invariant*

*They can learn spatial hierarchies of patterns*



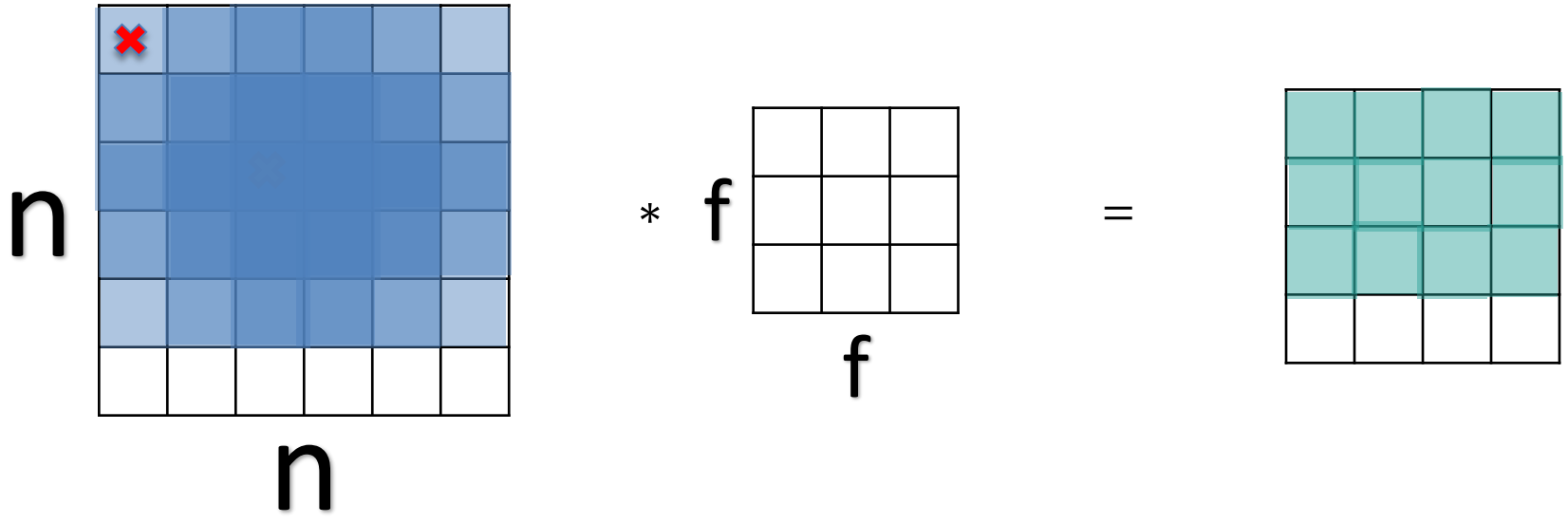
# مشکلات کانولوشن!




- لبه ها به اندازه ی پیکسل های درونی در کانولوشن شرکت نمیکنند
- خروجی کوچک میشود

$$n-f+1$$

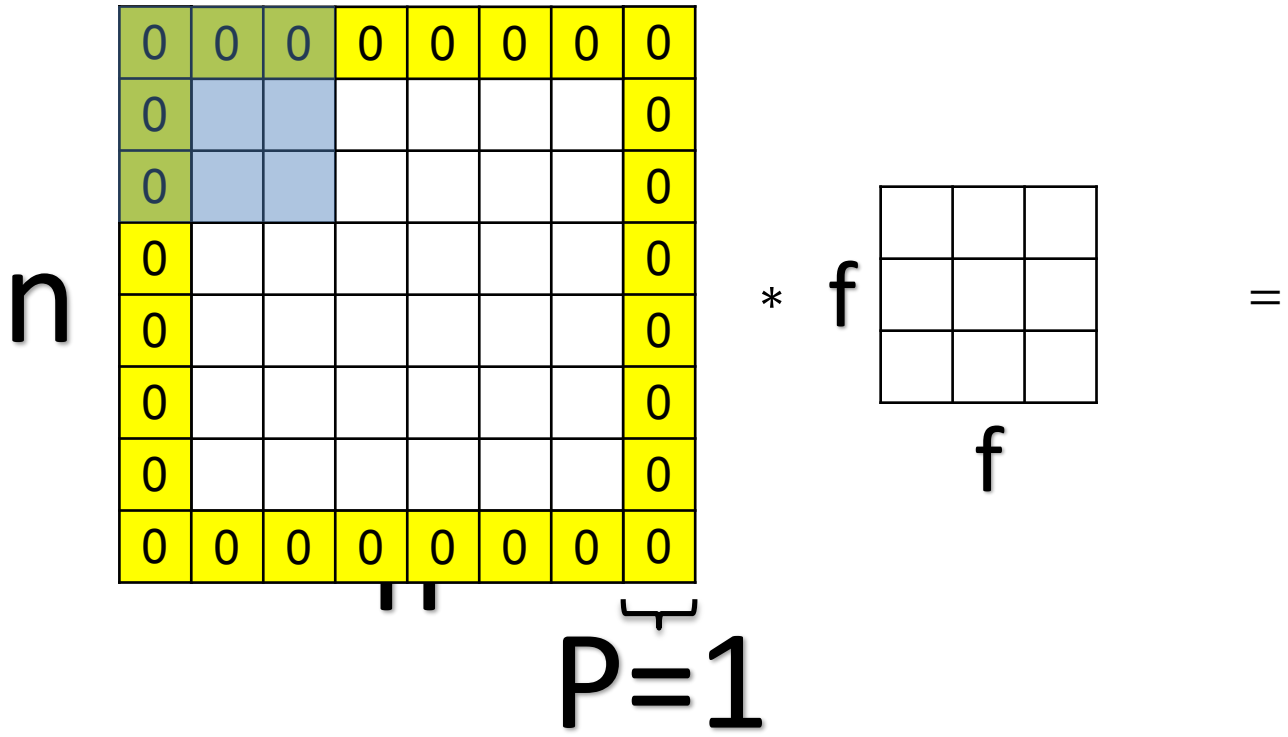
# مشکلات کانولوشن!



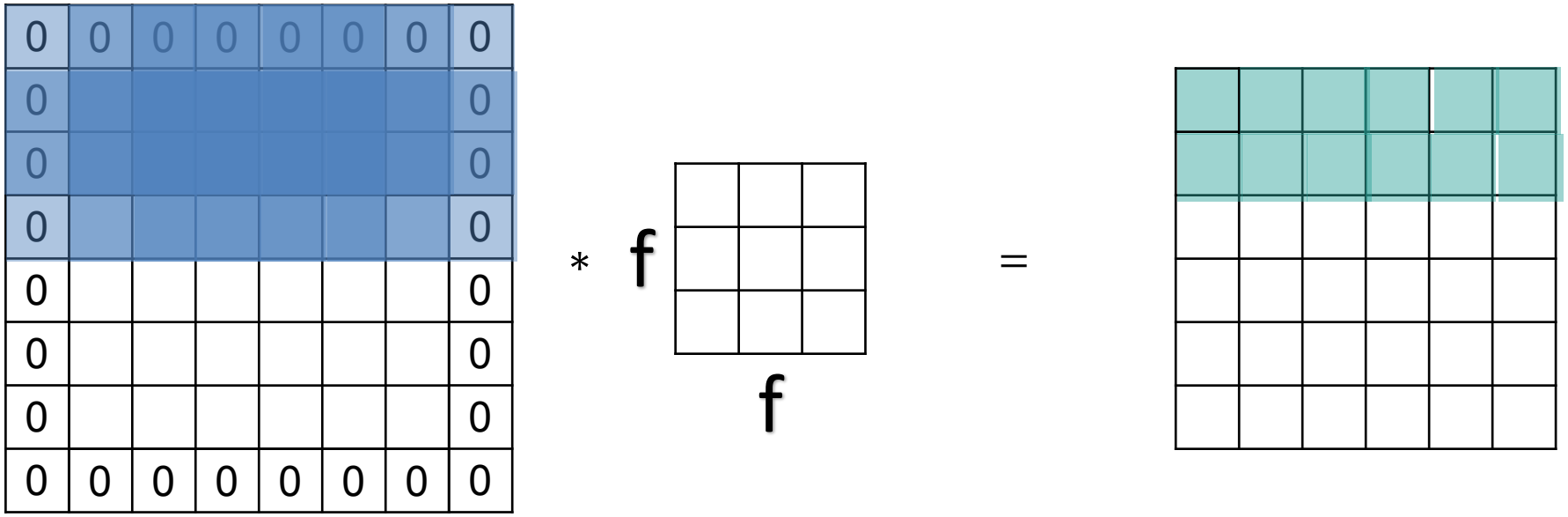

- لبه ها به اندازه ی پیکسل های درونی در کانولوشن شرکت نمیکنند
- خروجی کوچک میشود

$$n-f+1$$

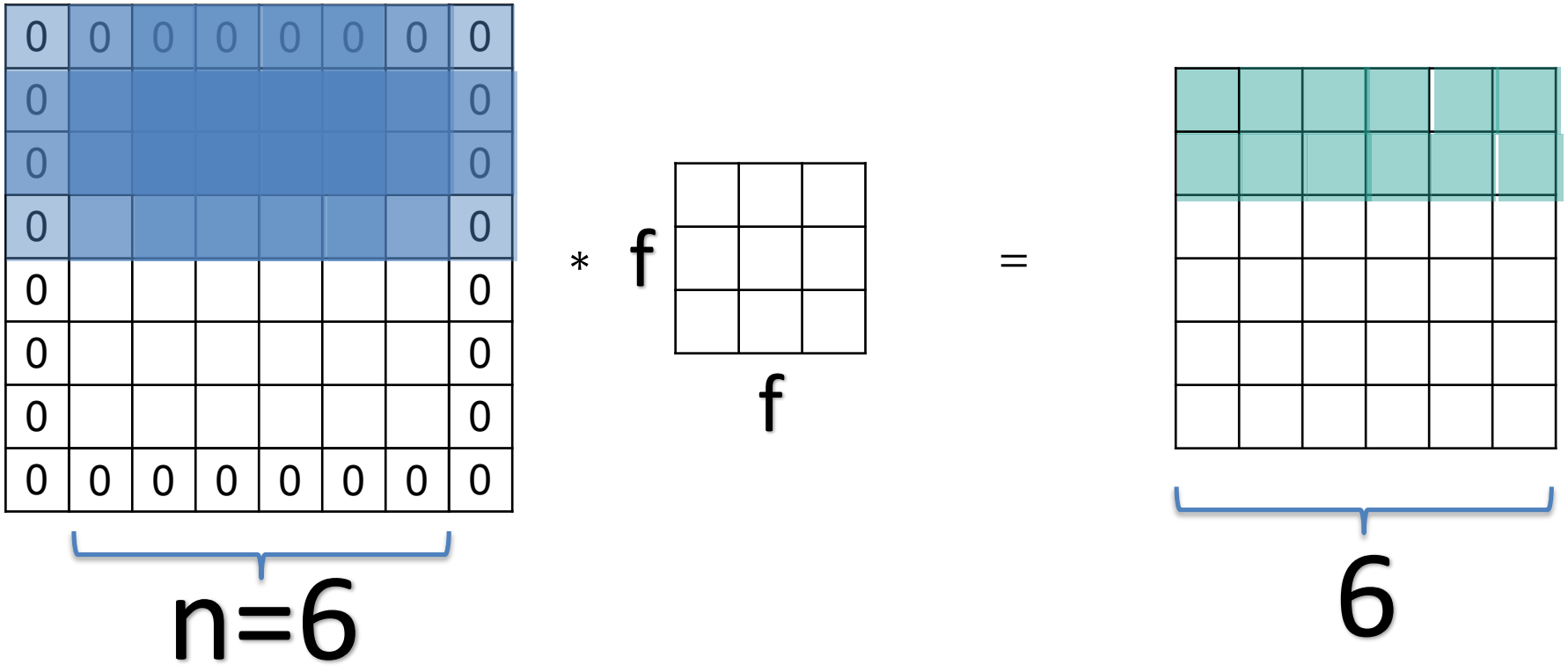
# راه کار؟ padding



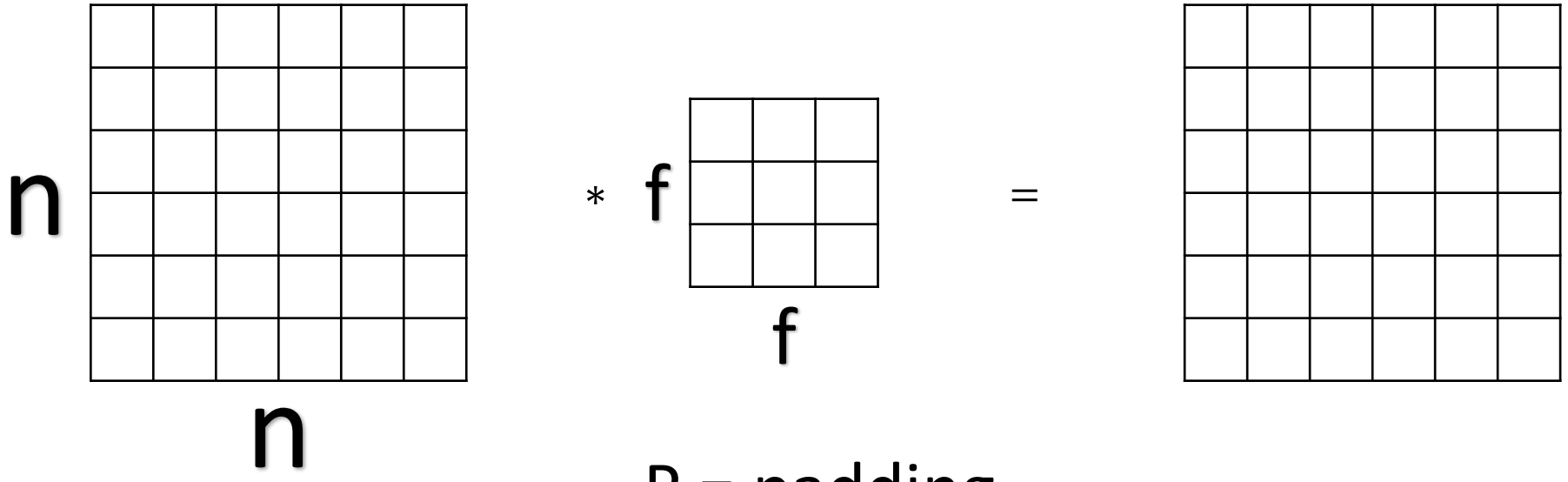
# padding



# padding



# padding




P = padding

$$n+2p-f+1$$

# کانولوشن Valid و Same

"Valid": nxn * fxf -> n-f+1 x n-f+1

6x6 * 3x3 -> 4 x 4

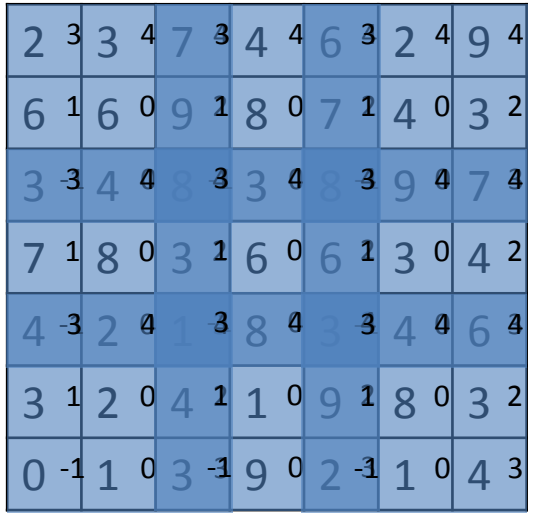
"Same": Pad so that output size is the same as the input size.

n+2p-f+1 x n+2p-f+1

$$n + 2p - f + 1 = n \Rightarrow p = \frac{f-1}{2}$$

برای مثال قبل P=1

# Stride در کانولوشن




*

| | | |
|---|---|---|
| 3 | 4 | 4 |
| 1 | 0 | 2 |
| -1 | 0 | 3 |

=

| | | |
|---|---|---|
| | | |
| | | |
| | | |

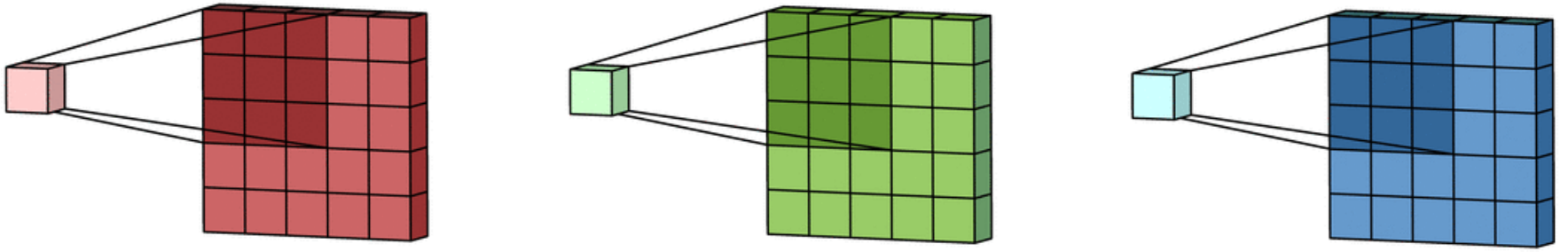
# خلاصه کانولوشن و خروجی آن

$n \times n$ image $\quad f \times f$ filter

padding $p$ $\quad$ stride $s$

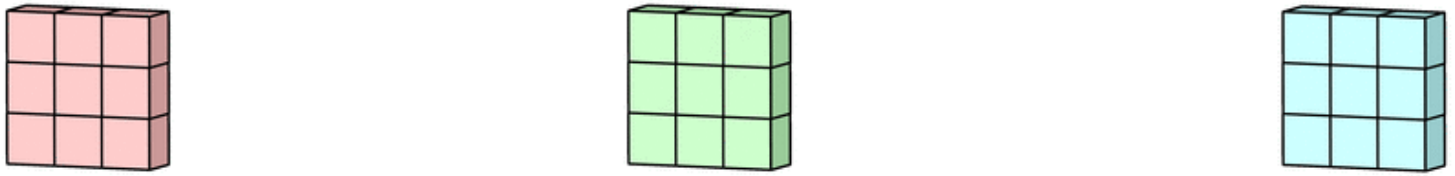
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

# کانولوشن بر روی تصویر رنگی (RGB)



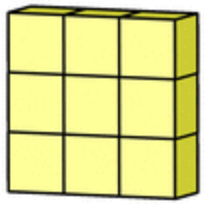
به ازای هر کانال رنگی یک فیلتر

# کانولوشن بر روی تصویر رنگی (RGB)



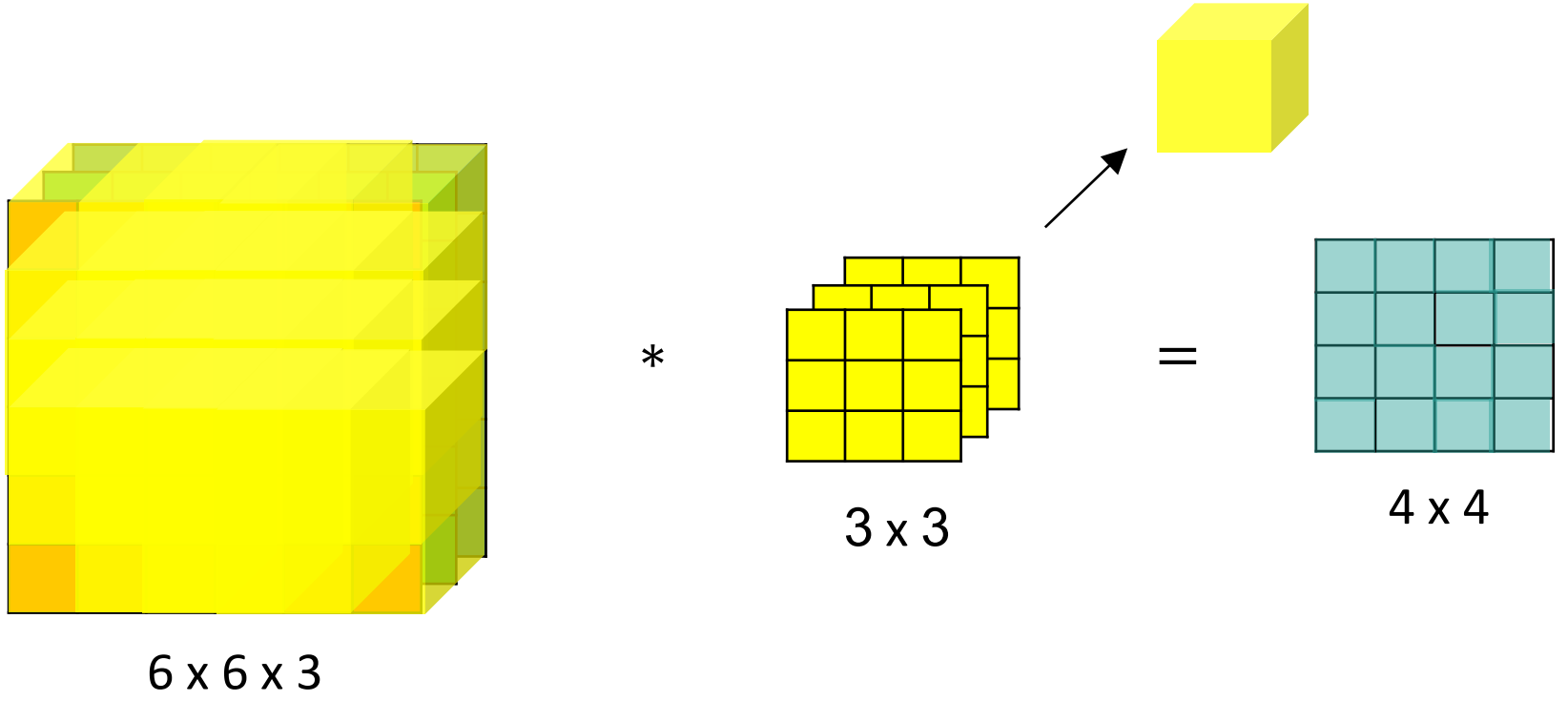
جمع نظیر به نظیر feature map یا activation map

# کانولوشن بر روی تصویر رنگی (RGB)

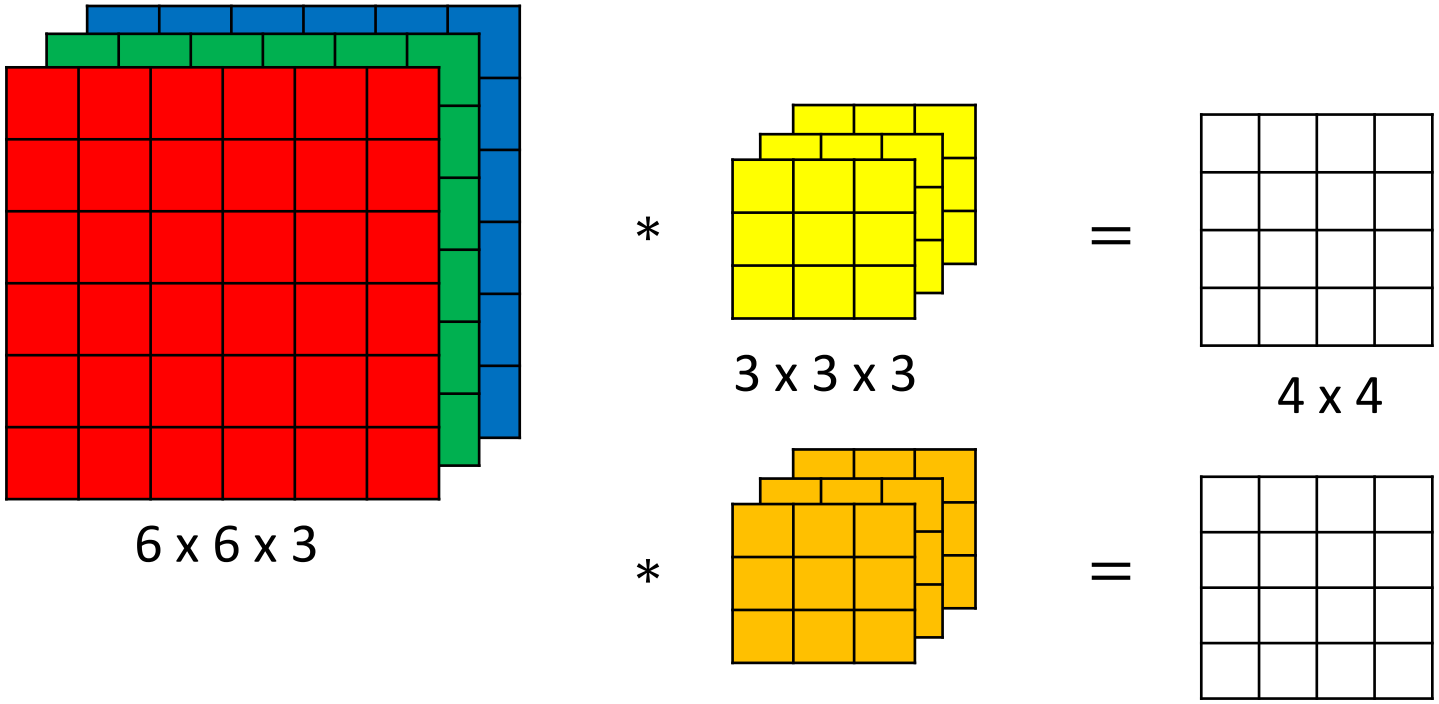


+ بایاس

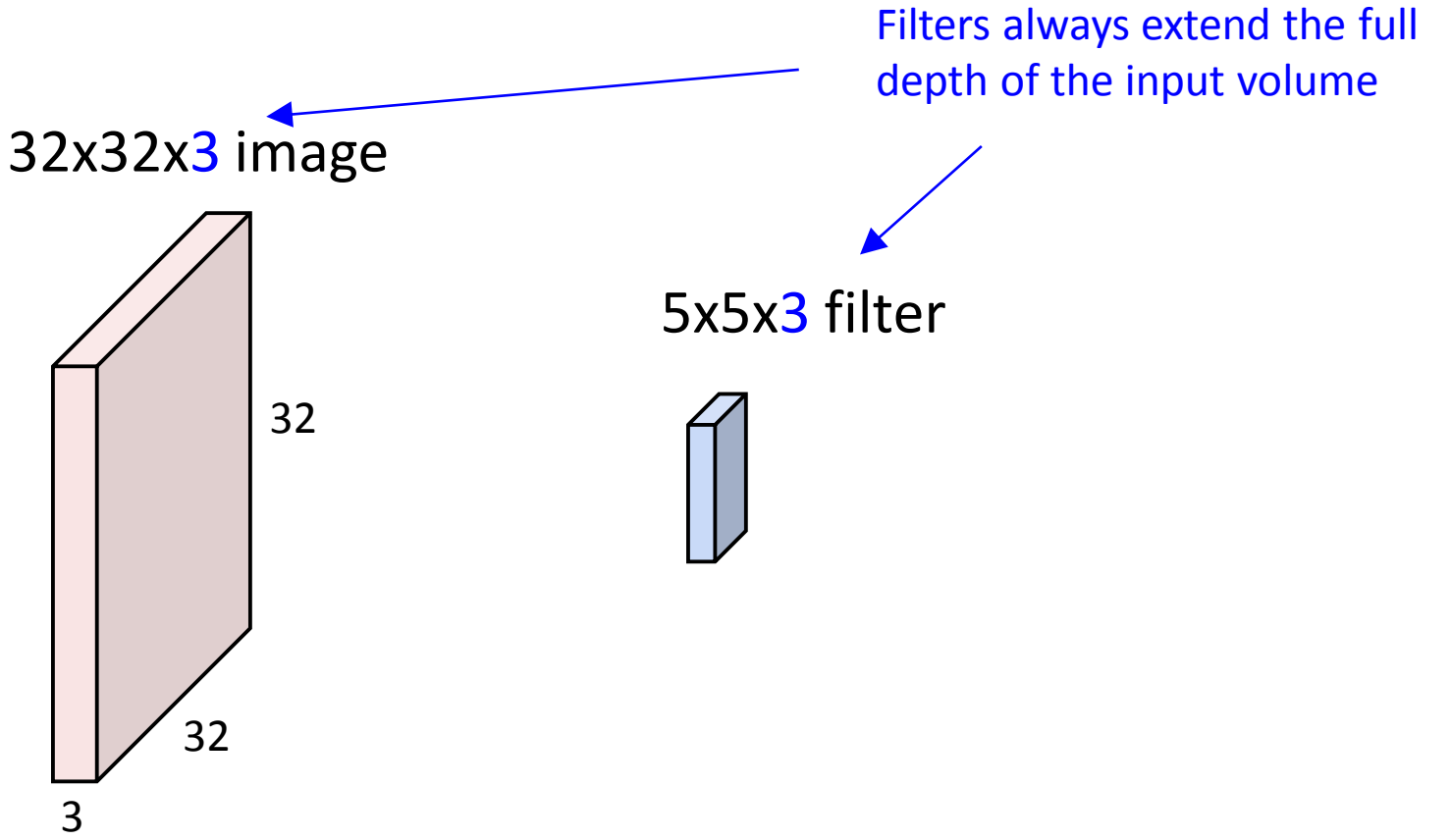
# کانولوشن بر روی تصویر رنگی (RGB)



# چندین فیلتر...

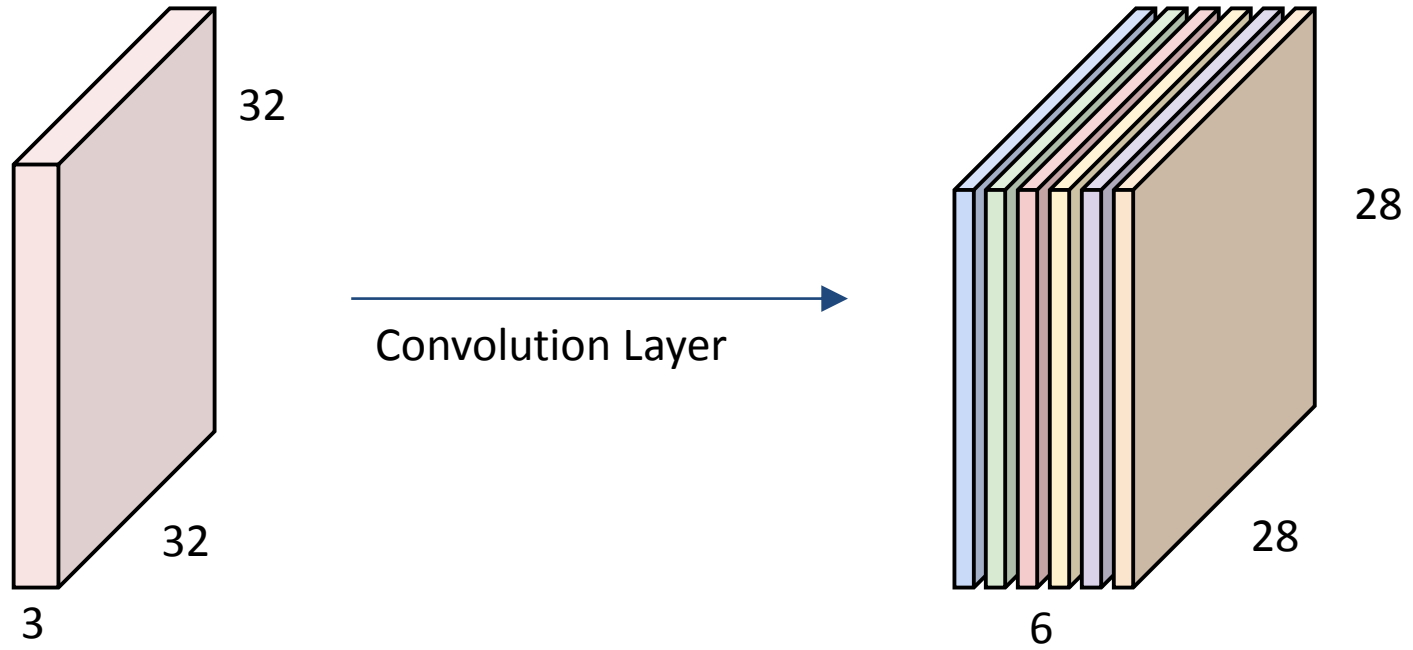


# چندین فیلتر...


Filters always extend the full depth of the input volume
32x32x3 image
32
32
3
5x5x3 filter

# چندین فیلتر...

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:




We stack these up to get a "new image" of size 28x28x6!

# محاسبه تعداد پارامتر

Number of parameters in one layer

If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?

تعداد پارامترهای هر فیلتر 3x3x3+1 = 28

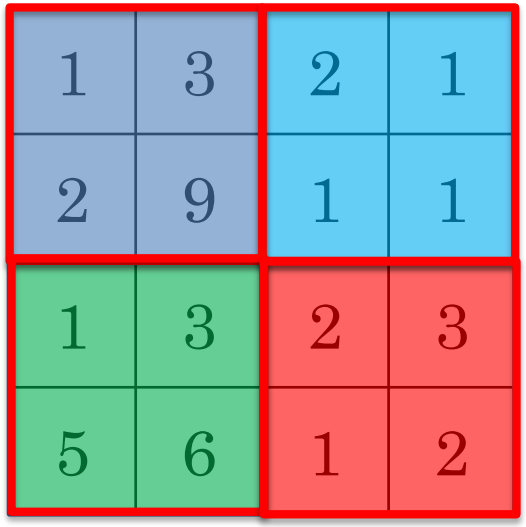
28x10 = 280

# یک شبکه کانولوشنال ساده

# انواع لایه در شبکه‌های کانولوشنالی

- Convolution (Conv)
- Pooling (Pool)
- Fully Connected (FC)

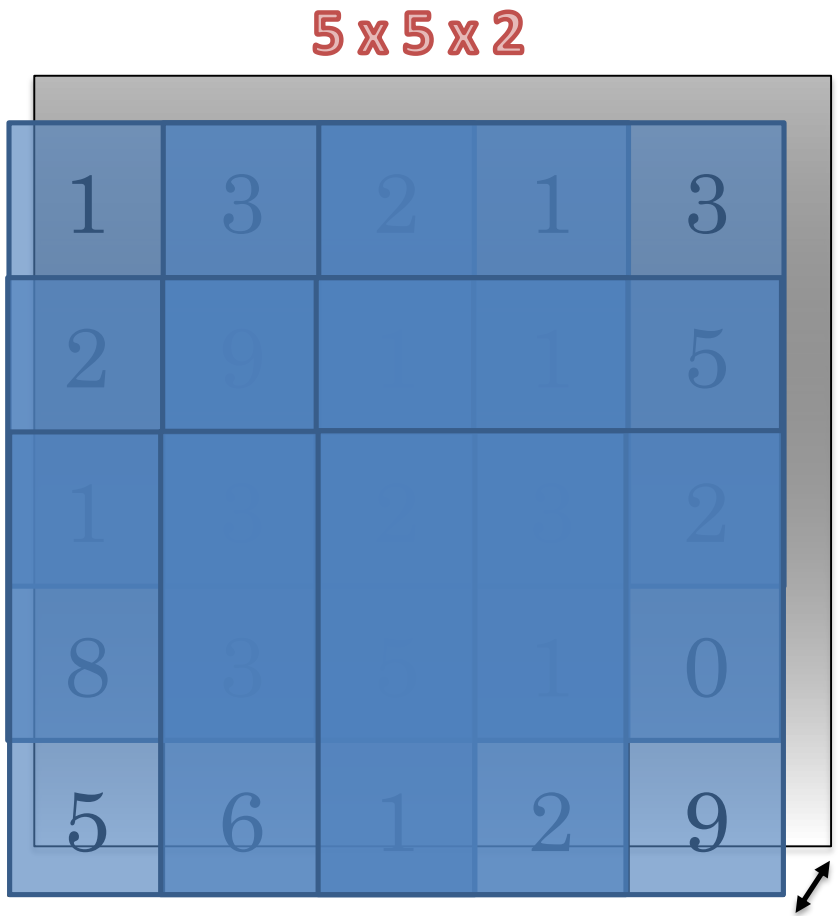
# لایه‌ی pooling




| | |
|---|---|
| 9 | 2 |
| 6 | 3 |

- دو hyper parameter دارد: s و f
- هیچ پارامتری برای آموزش ندارد

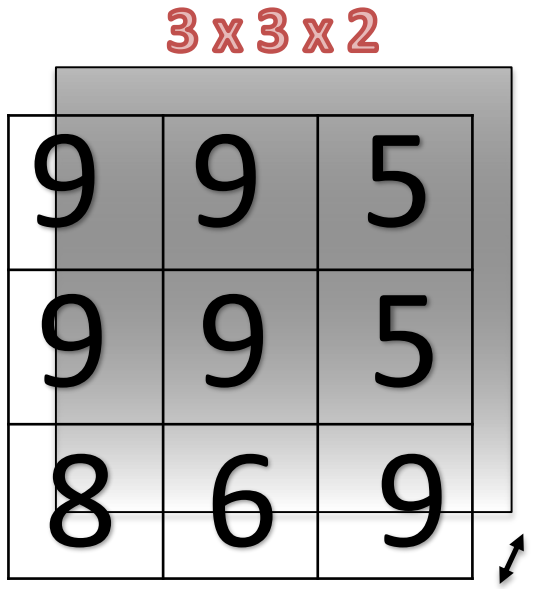
# لایه‌ی max pooling – pooling



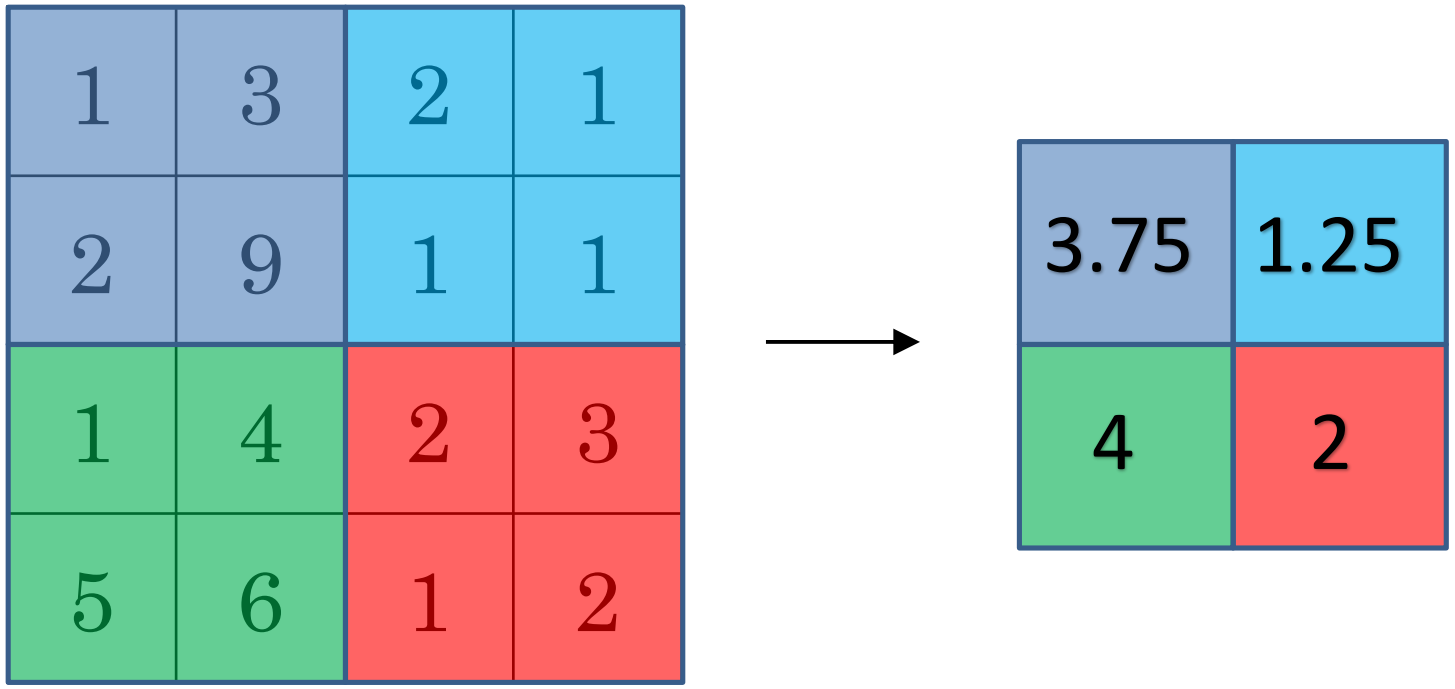

5 x 5

f = 3

S = 1



# Average pooling



# خلاصه pooling

Hyperparameters:

f : filter size

s : stride

Max or average pooling

f = 2, s = 2
f = 3, s = 2

# مثالی از یک شبکه کانولوشنالی (LeNet-5)

# سایز هر لایه و تعداد پارامترها

| | Activation shape | Activation Size | # parameters* |
|---|---|---|---|
| Input: | (32,32,3) | 3,072 | 0 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# چرا کانولوشن؟

$$76\ x\ 6\ =\ 456(5\ x\ 5\ x\ 3)\ +\ 1\ =76$$

$$3072\ \times 4704 = 14450688$$
$$\approx\ \ 14.5\ m$$

| | | | | | |
|---|---|---|---|---|---|
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |

*

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

=

| | | | |
|---|---|---|---|
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |

**Parameter sharing:** A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

**Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

# شبکه‌های کانولوشنالی در کراس – ارقام فارسی


**06_ConvolutionalNeuralNetwork-Hoda-Keras.ipynb**

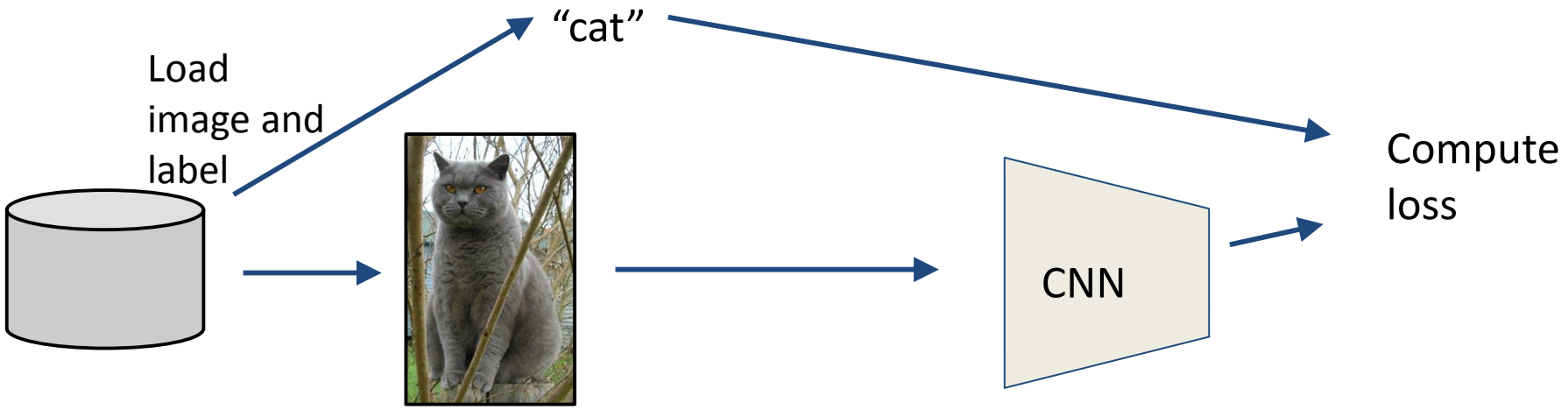
# شبکه‌های کانولوشنالی در کراس – چالش گربه / سگ


**07_CNN-cat_Vs_dog.ipynb**

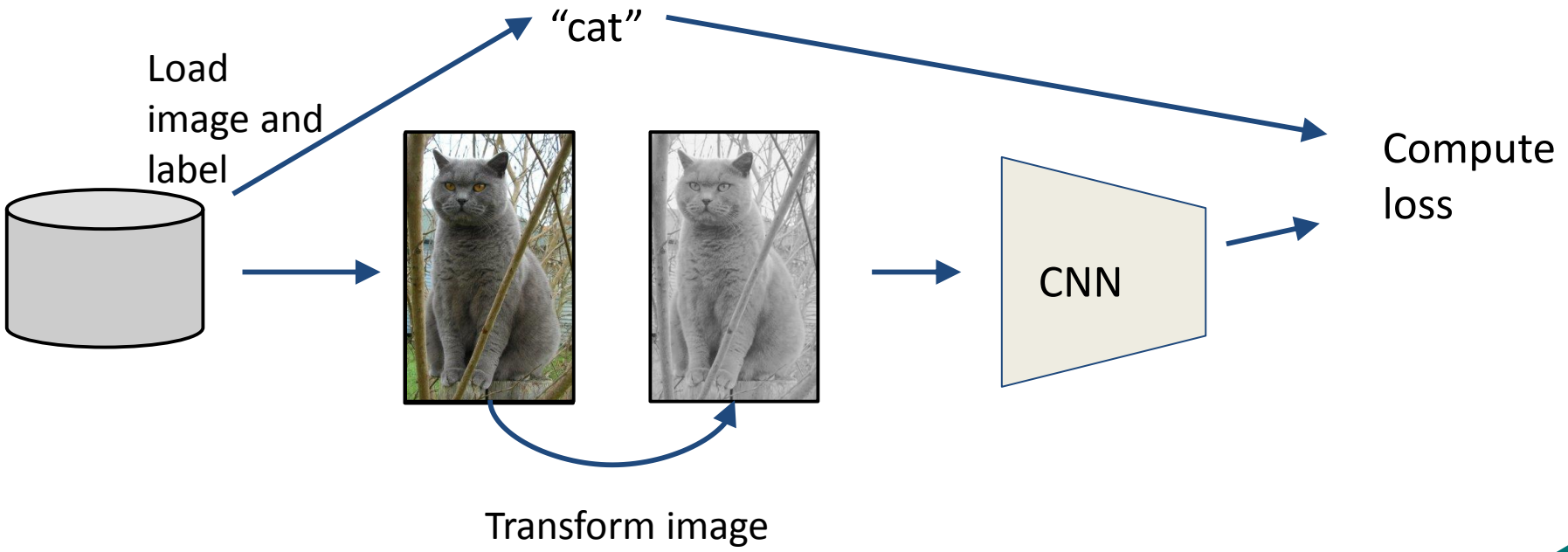
# بخش ۲:

# افزایش دادگان
# Data Augmentation

# Data Augmentation

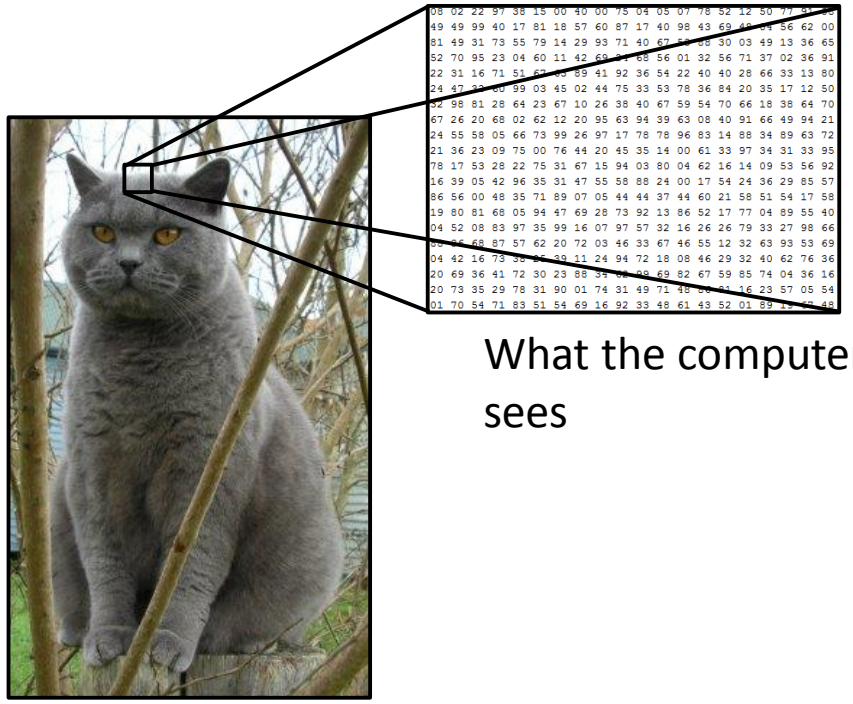


# Data Augmentation



# Data Augmentation

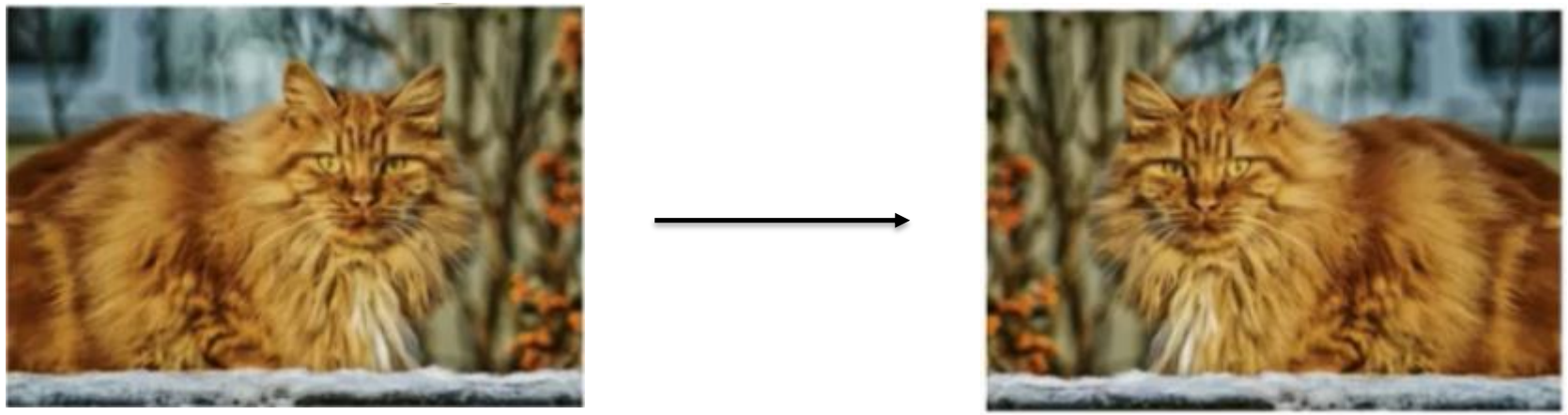
- Change the pixels without changing the label
- Train on transformed data
- VERY widely used



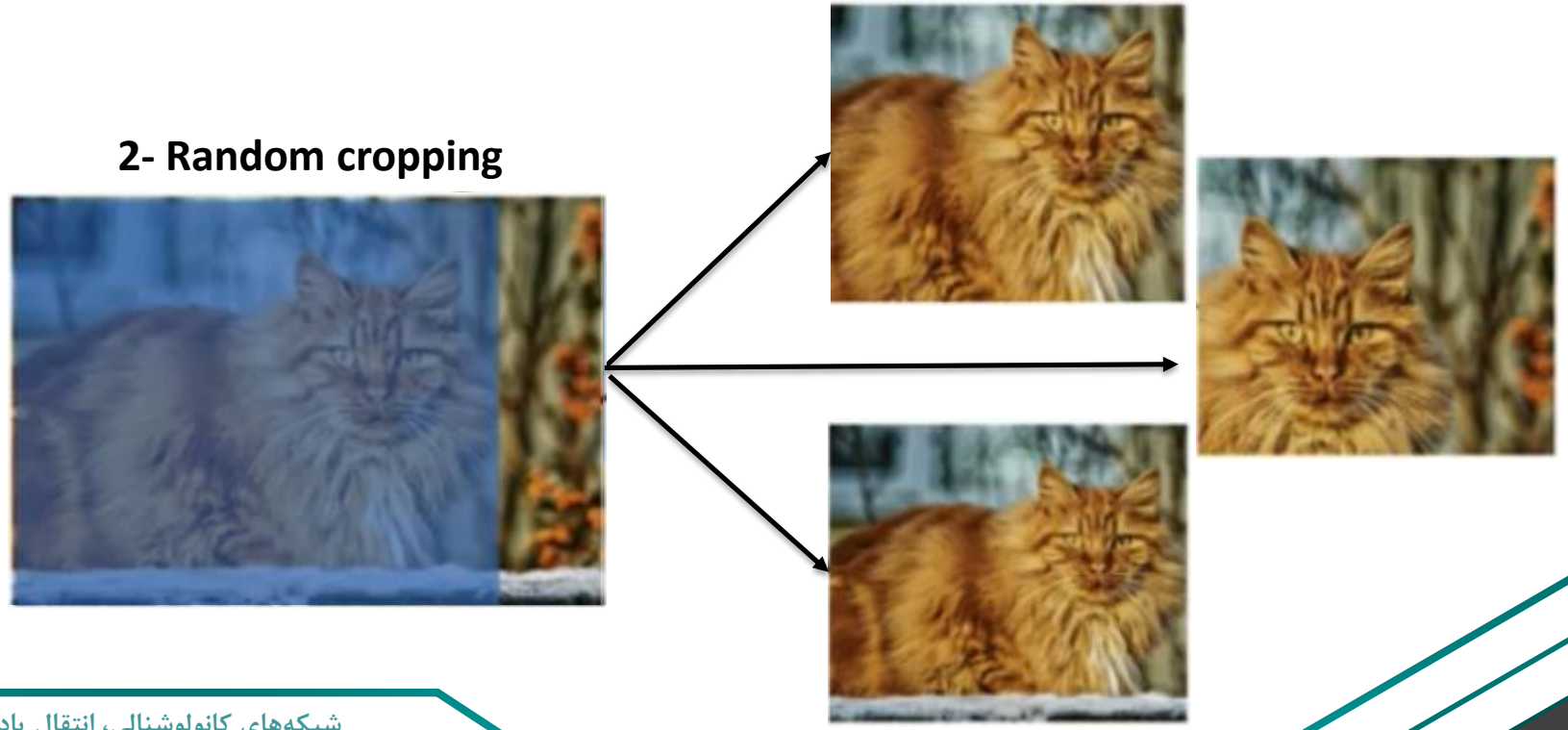
What the computer sees

# Common augmentation method

## 1- Mirroring(Horizontal flips)

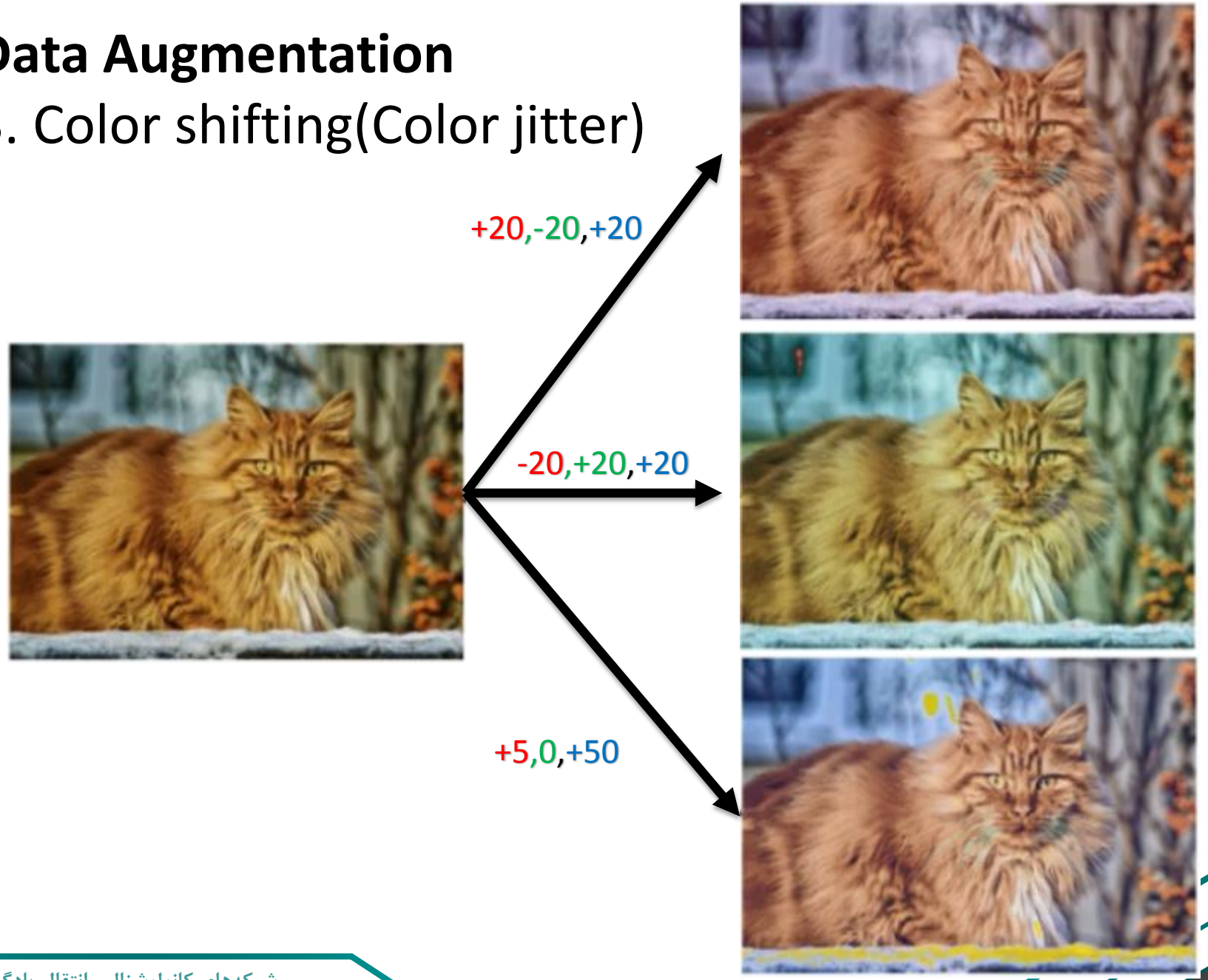


## 2- Random cropping



# Data Augmentation

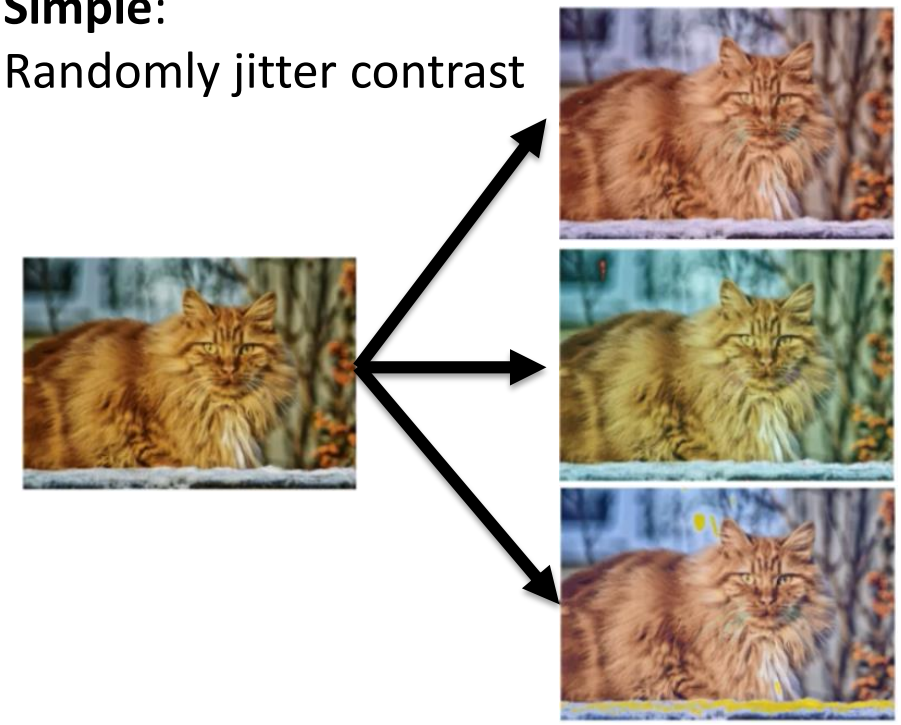
## 3. Color shifting(Color jitter)



# Data Augmentation

## 3. Color shifting(Color jitter)

**Simple**:
Randomly jitter contrast



برای مطالعه بیشتر ☺

**Complex (PCA color augmentation)**:

1. Apply PCA to all [R, G, B] pixels in training set
2. Sample a “color offset” along principal component directions

(As seen in *[Krizhevsky et al. 2012],* ResNet, etc)

## Data Augmentation

4. Get creative!

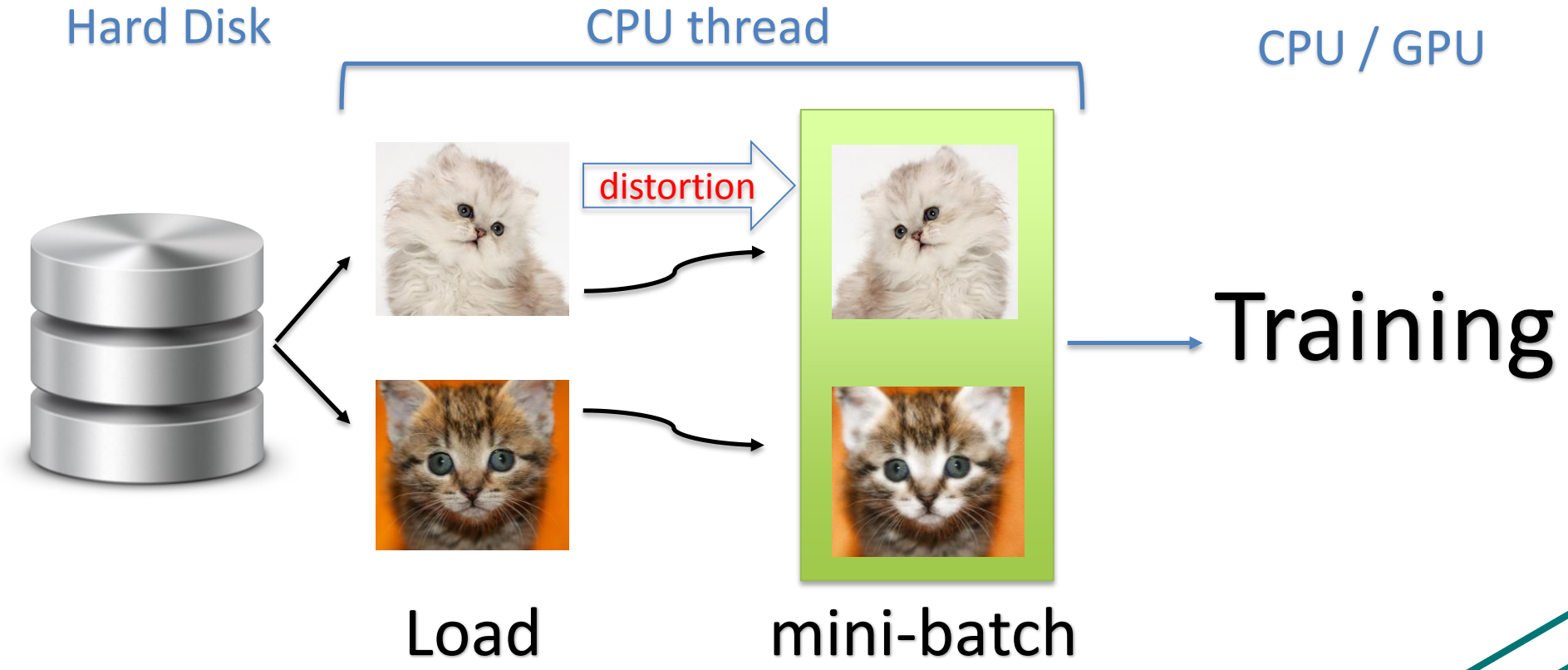
Random mix/combinations of :

- translation
- rotation
- stretching
- shearing, 
- lens distortions, … (go crazy)

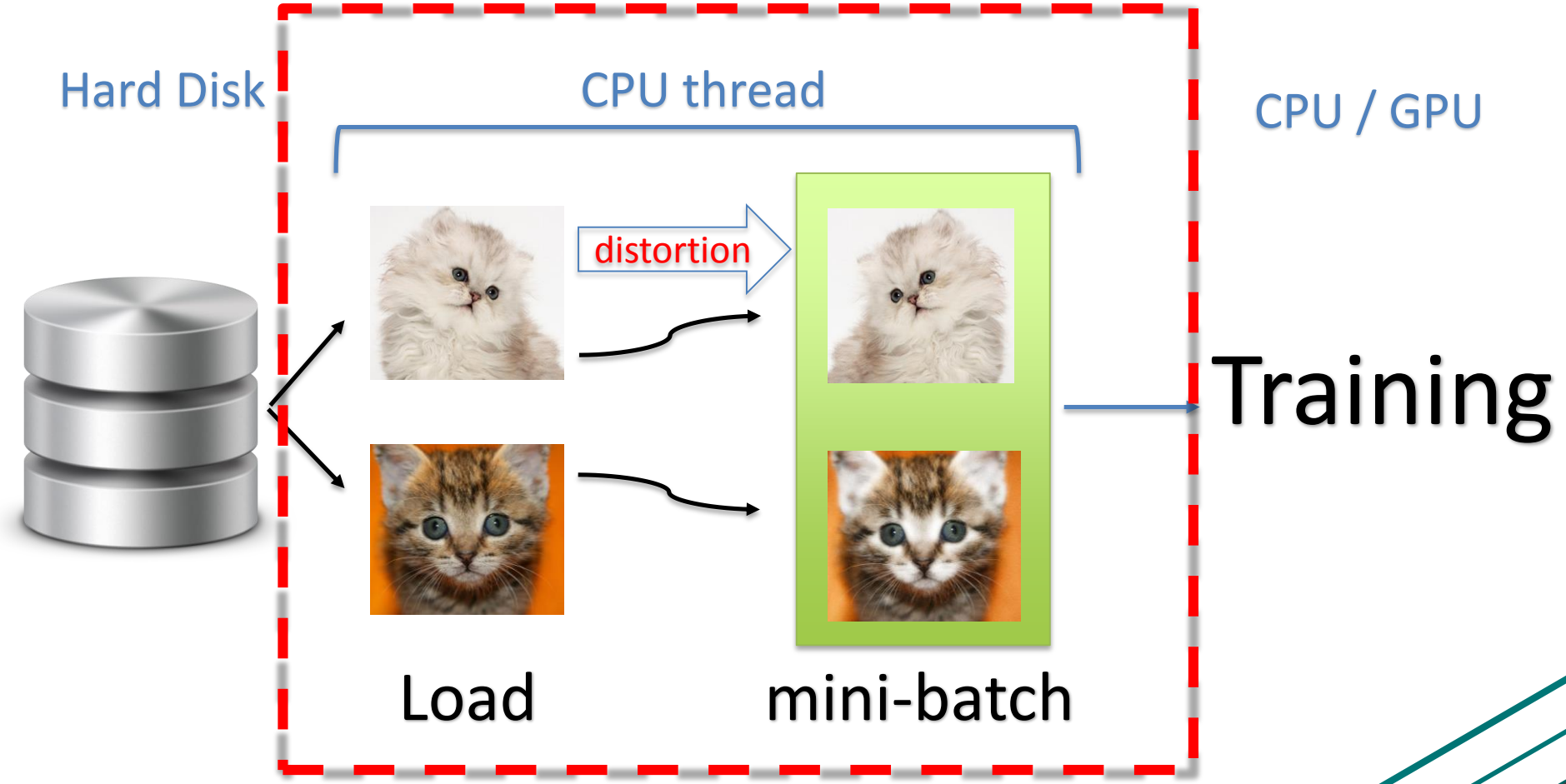
# Bottlenecks

to be aware of

# Implementation distortions during training



# Implementation distortions during training



# Data Augmentation


**08_data_augmentation.ipynb**

# لود مدل ذخیره شده در کراس


**09_Load_trained_model_in_keras.ipynb**

# بخش ۳:

# مطالعه موردی

## (بررسی معماری‌های حائز رتبه در طبقه بندی تصاویر)

# Case Study

# Classic networks:

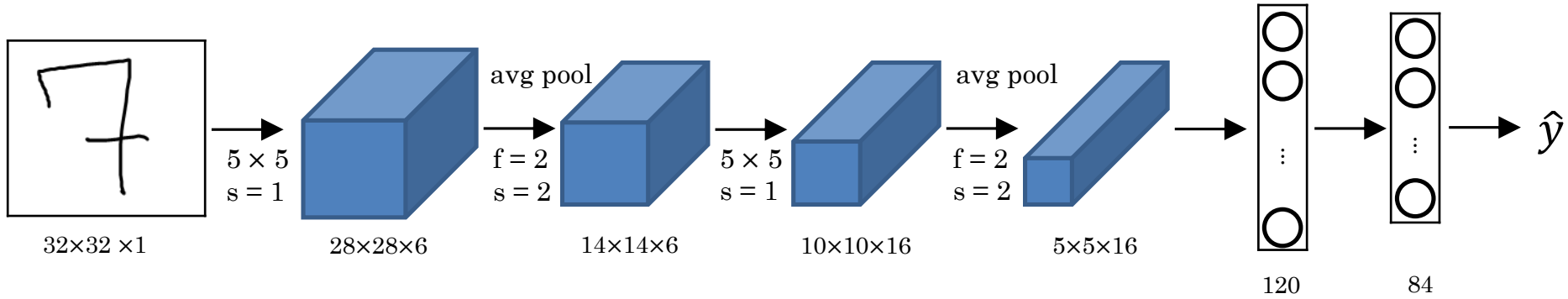
- LeNet-5
- AlexNet
- ZFNet
- VGG

ResNet

Inception

# LeNet

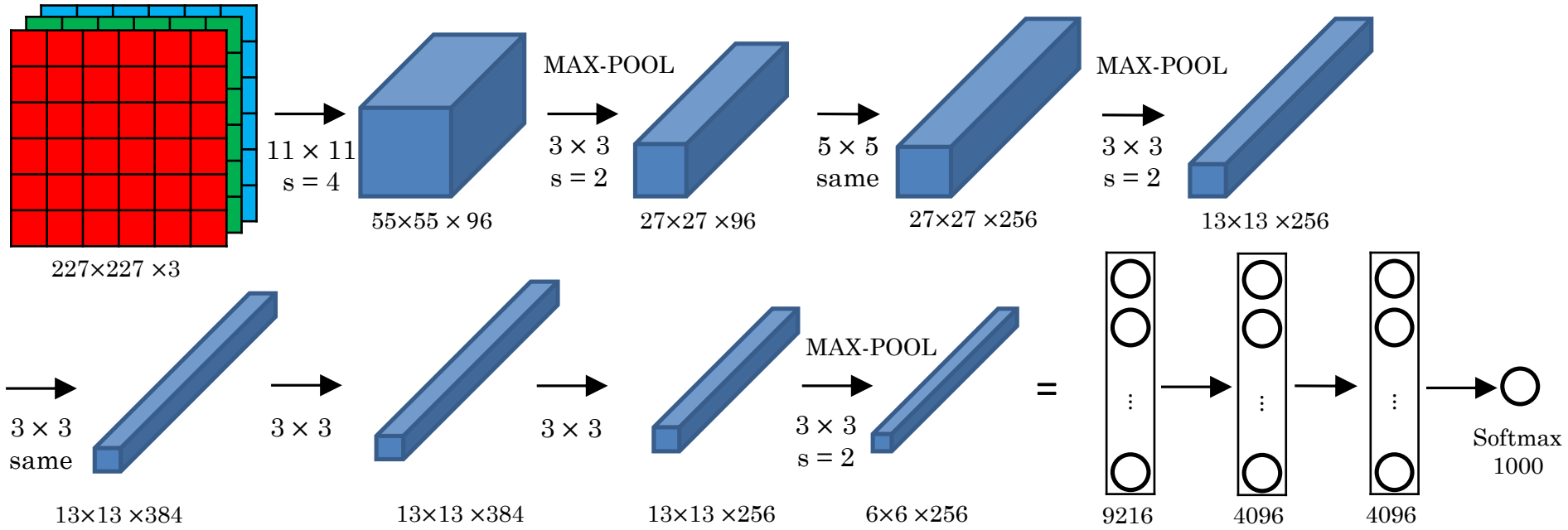
# LeNet - 5




[LeCun et al., 1998. Gradient-based learning applied to document recognition]

# AlexNet

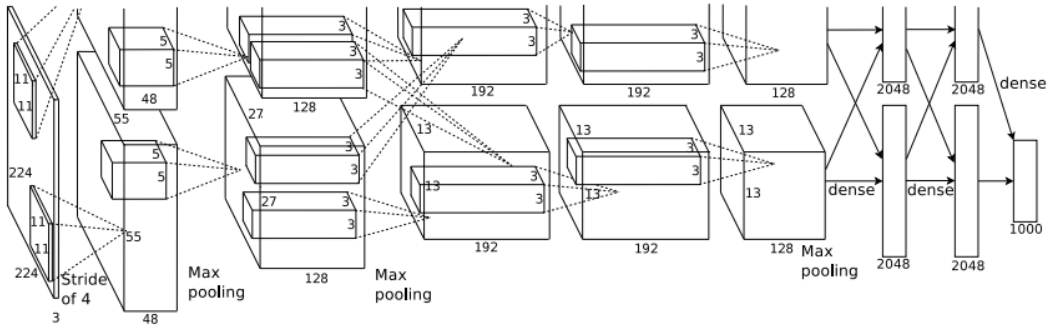
AlexNet




[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

# Case Study: AlexNet

*[Krizhevsky et al. 2012]*



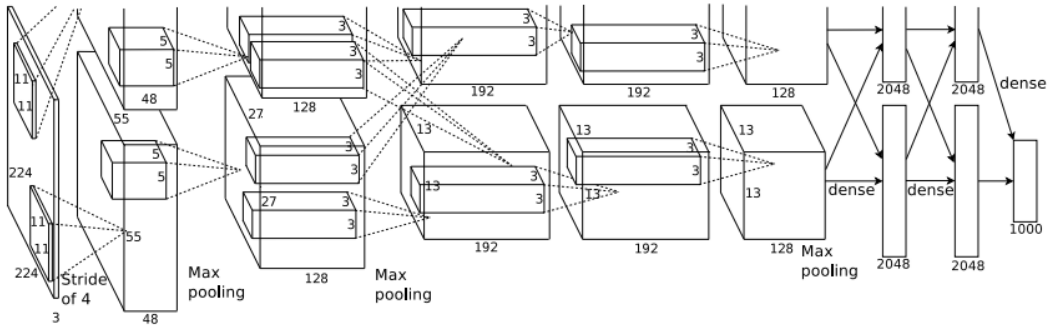

Input: 227x227x3 images

**First layer** (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**

Q: What is the total number of parameters in this layer?

# Case Study: AlexNet

*[Krizhevsky et al. 2012]*

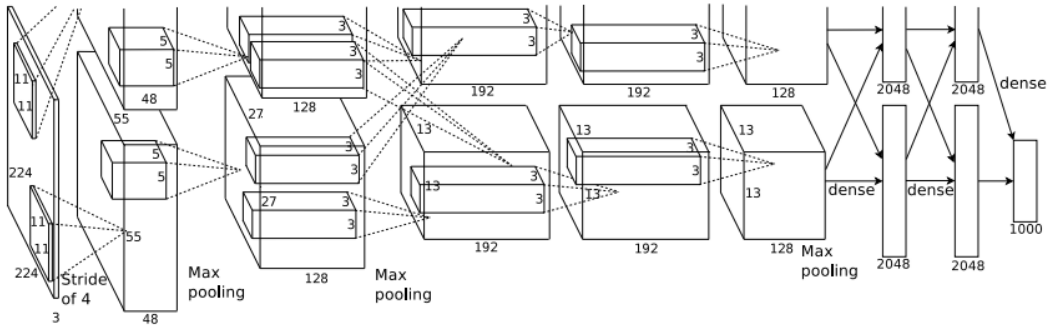



Input: 227x227x3 images

**First layer** (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3)*96 = **35K**

# Case Study: AlexNet

*[Krizhevsky et al. 2012]*



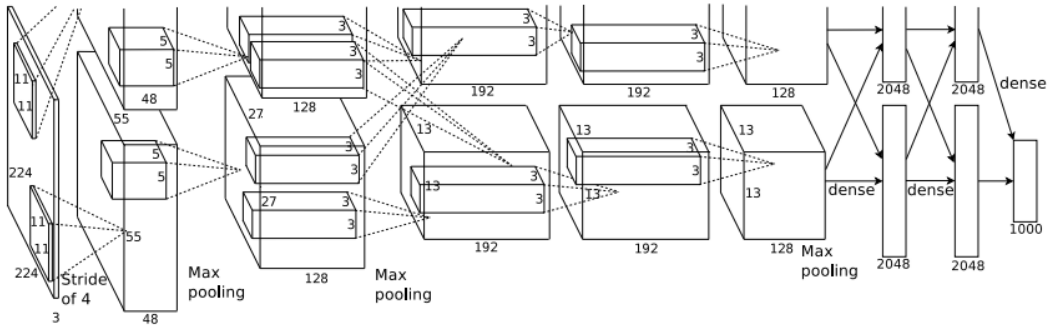

Input: 227x227x3 images
After CONV1: 55x55x96

**Second layer** (POOL1): 3x3 filters applied at stride 2

Q: what is the output volume size? Hint: (55-3)/2+1 = 27

# Case Study: AlexNet

*[Krizhevsky et al. 2012]*



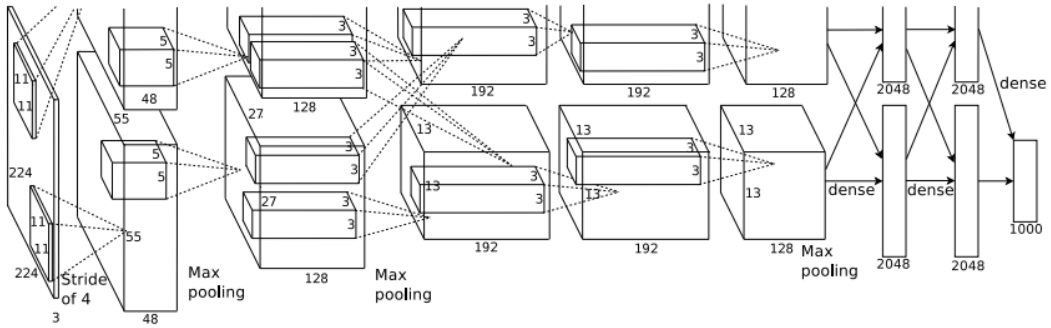

Input: 227x227x3 images
After CONV1: 55x55x96

**Second layer** (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

Q: what is the number of parameters in this layer?

# Case Study: AlexNet

*[Krizhevsky et al. 2012]*

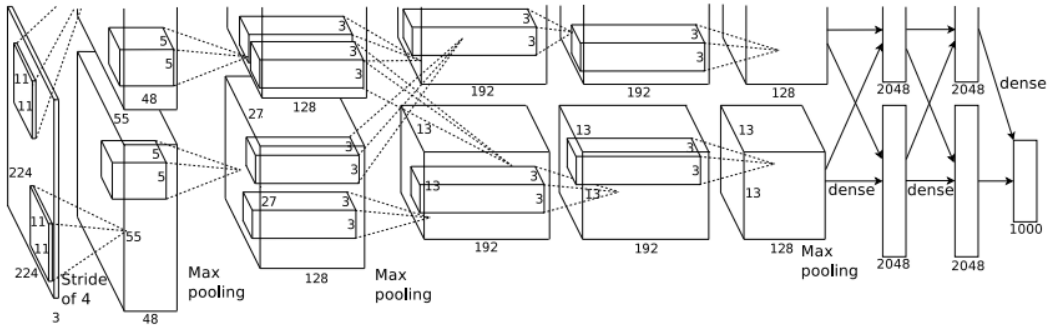



Input: 227x227x3 images
After CONV1: 55x55x96

**Second layer** (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!

# Case Study: AlexNet

*[Krizhevsky et al. 2012]*

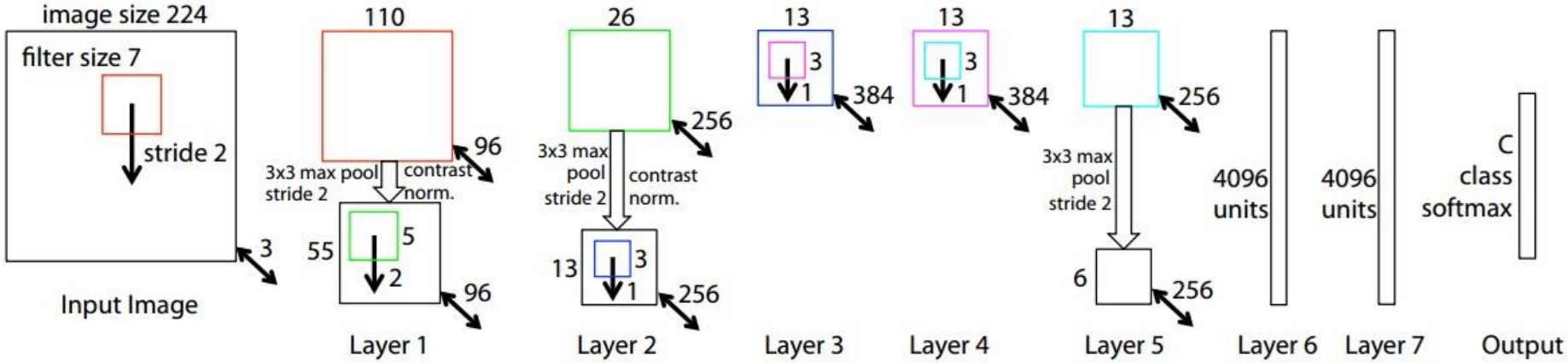



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

**Details/Retrospectives:**
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
**- 7 CNN ensemble: 18.2% -> 15.4%**

# ZFNet

# Case Study: ZFNet




AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

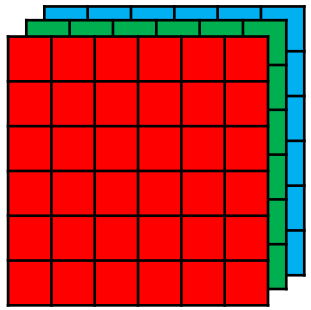
ImageNet top 5 error: 15.4% -> 14.8%

*[Zeiler and Fergus, 2013]*

# VGG

# VGG - 16

CONV = 3×3 filter, s = 1, same  MAX-POOL = 2×2 , s = 2



224×224 ×3

→ [CONV 64] ×2 → 224×224×64 → POOL → 112×112 ×64 → [CONV 128] ×2 → 112×112 ×128 → POOL → 56×56 ×128

→ [CONV 256] ×3 → 56×56 ×256 → POOL → 28×28 ×256 → [CONV 512] × 3 → 28×28 ×512 → POOL → 14×14×512

→ [CONV 512] ×3 → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

# Case Study: VGGNet

*[Simonyan and Zisserman, 2014]*

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

TOTAL memory: 24M * 4 bytes ~= 93MB / image
(only forward! ~*2 for bwd)
TOTAL params: 138M parameters

best model

| 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 **conv1-256** | 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** |
| 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** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

Table 2: **Number of parameters** (in millions).

| Network | A,A-LRN | B | C | D | E |
|---|---|---|---|---|---|
| Number of parameters | 133 | 133 | 134 | 138 | 144 |

11.2% top 5 error in ILSVRC 2013 -> 7.3% top 5 error

# Classic networks:

- LeNet-5 ✓
- AlexNet ✓ **7 CNN ensemble: 15.4%** top 5 error
- ZFNet ✓ 14.8% top 5 error
- VGG ✓ 7.3% top 5 error

Inception

ResNet

**اطلاعات بیشتر:**

جلسه ۱۶ - بررسی معماری‌های حائز رتبه در طبقه بندی تصاویر:

http://blog.class.vision/winter-96-97-syllabus/

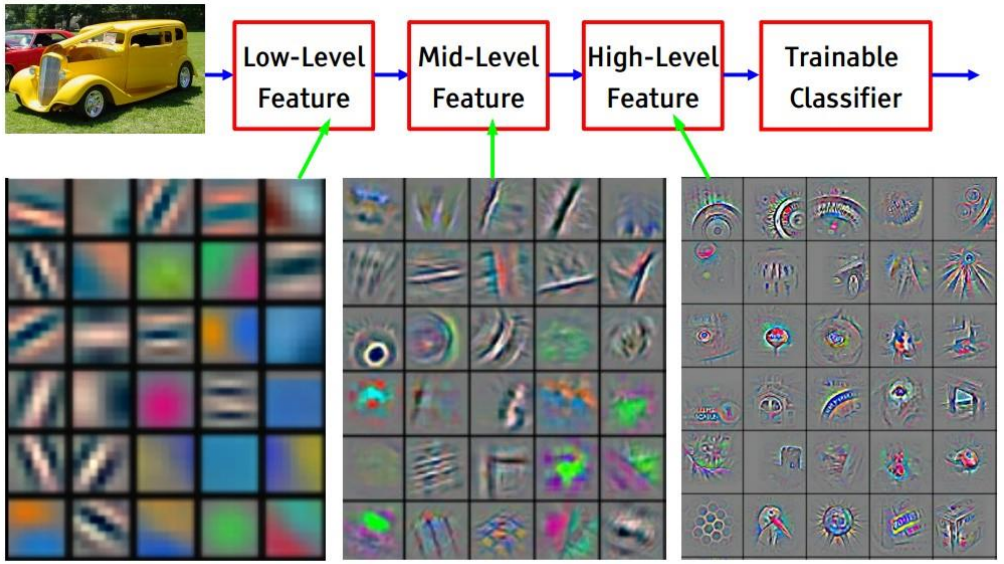
# استفاده از مدل‌های از قبل آموزش داده شده در کراس


**11_using-a-pretrained-convnet.ipynb**

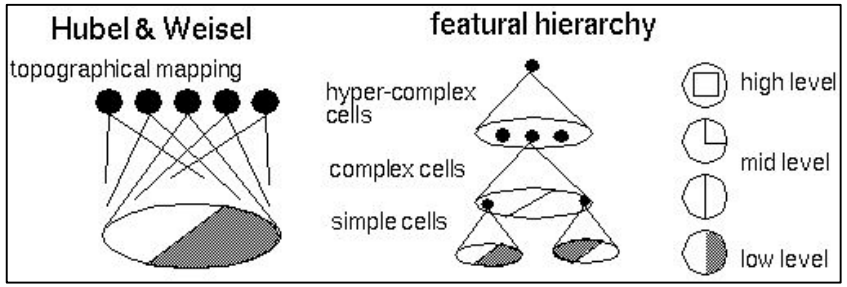
# بخش ۴:

# انتقال یادگیری
# Transfer Learning

## Preview

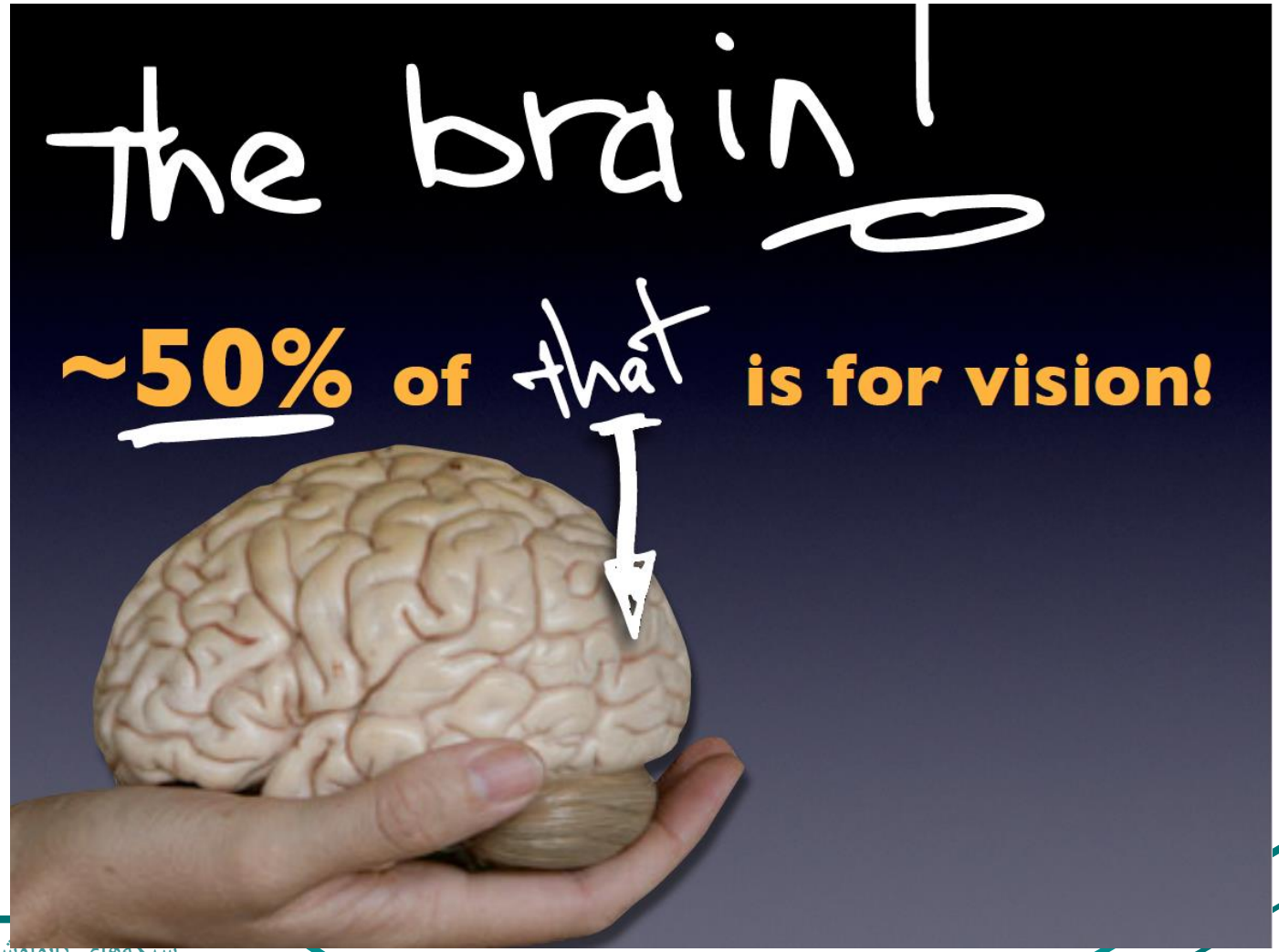



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



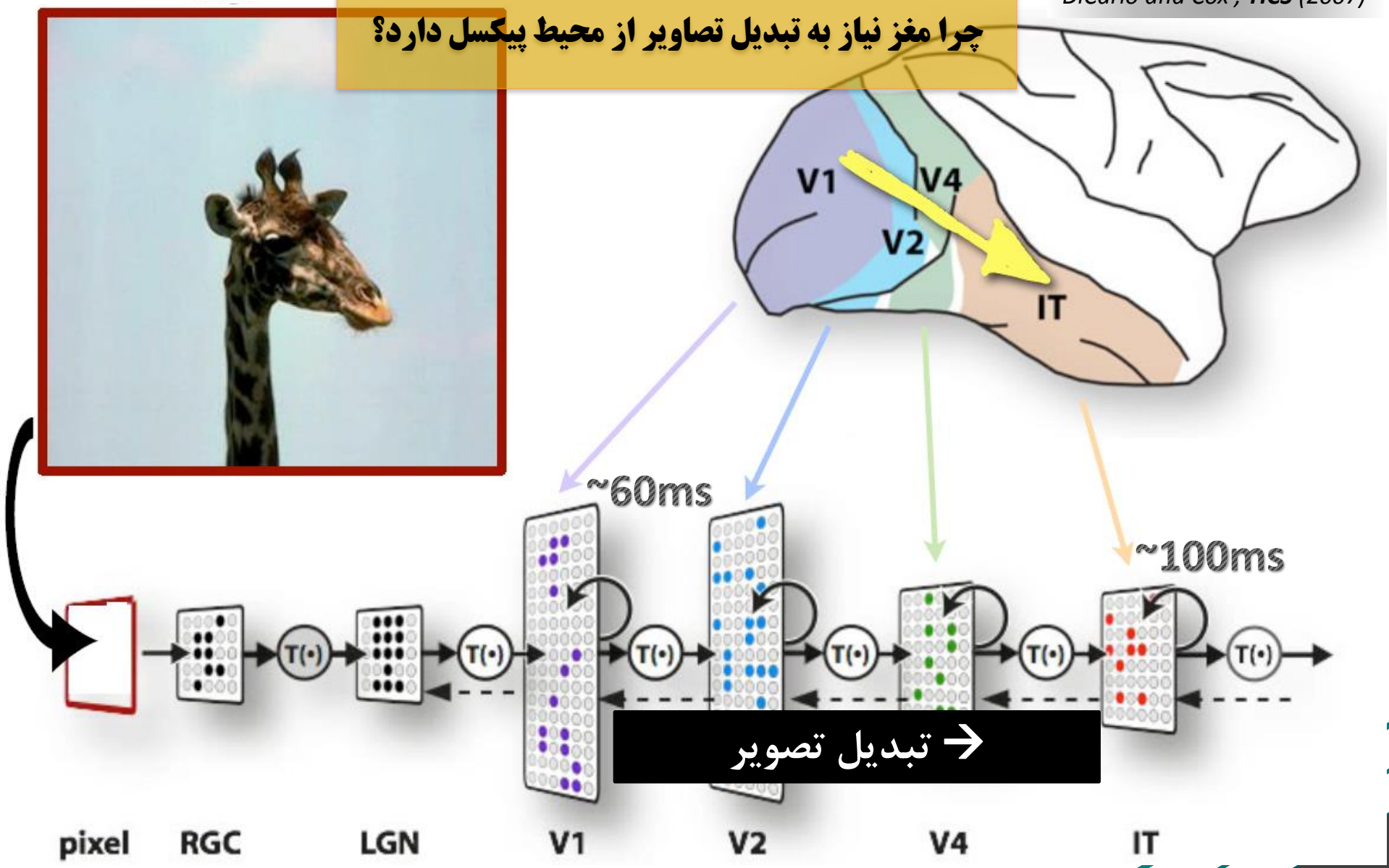

*[From recent Yann LeCun slides]*

# نگاهی به فرآیند بینایی در مغز!



## چرا مغز نیاز به تبدیل تصاویر از محیط پیکسل دارد؟

DiCarlo and Cox , **TICS** (2007)



## تغییرات اشیاء

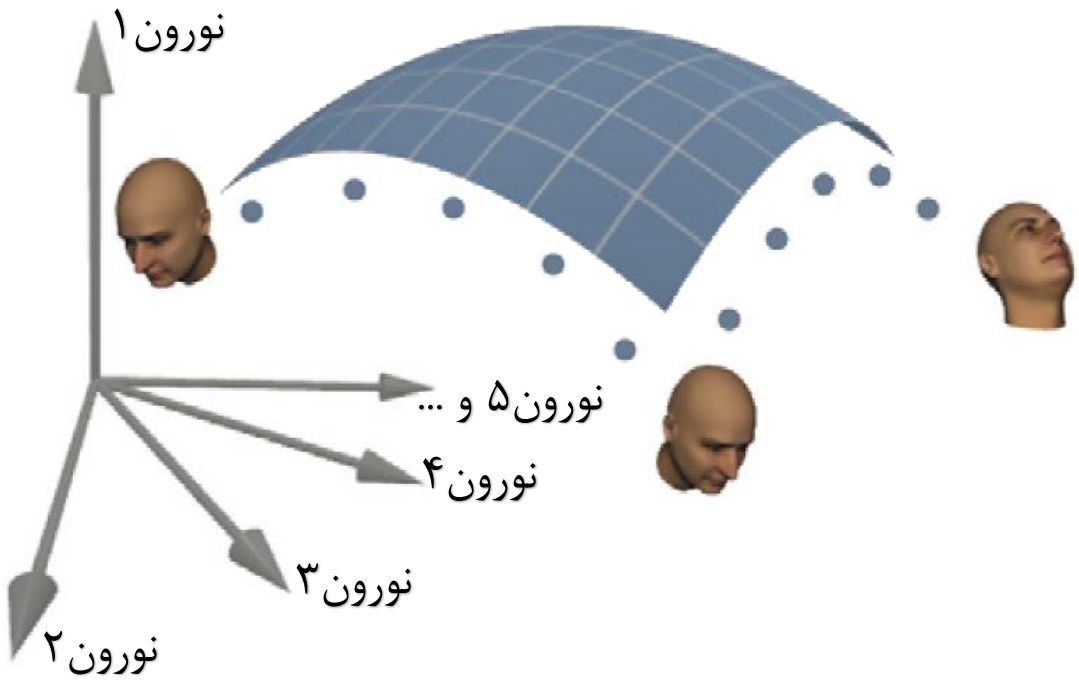
اندازه، زاویه‌ی دید، تغییرات نور و موقعیت در صفحه

درهم ریختگی (Clutter, occlusion)

# Invariant object recognition

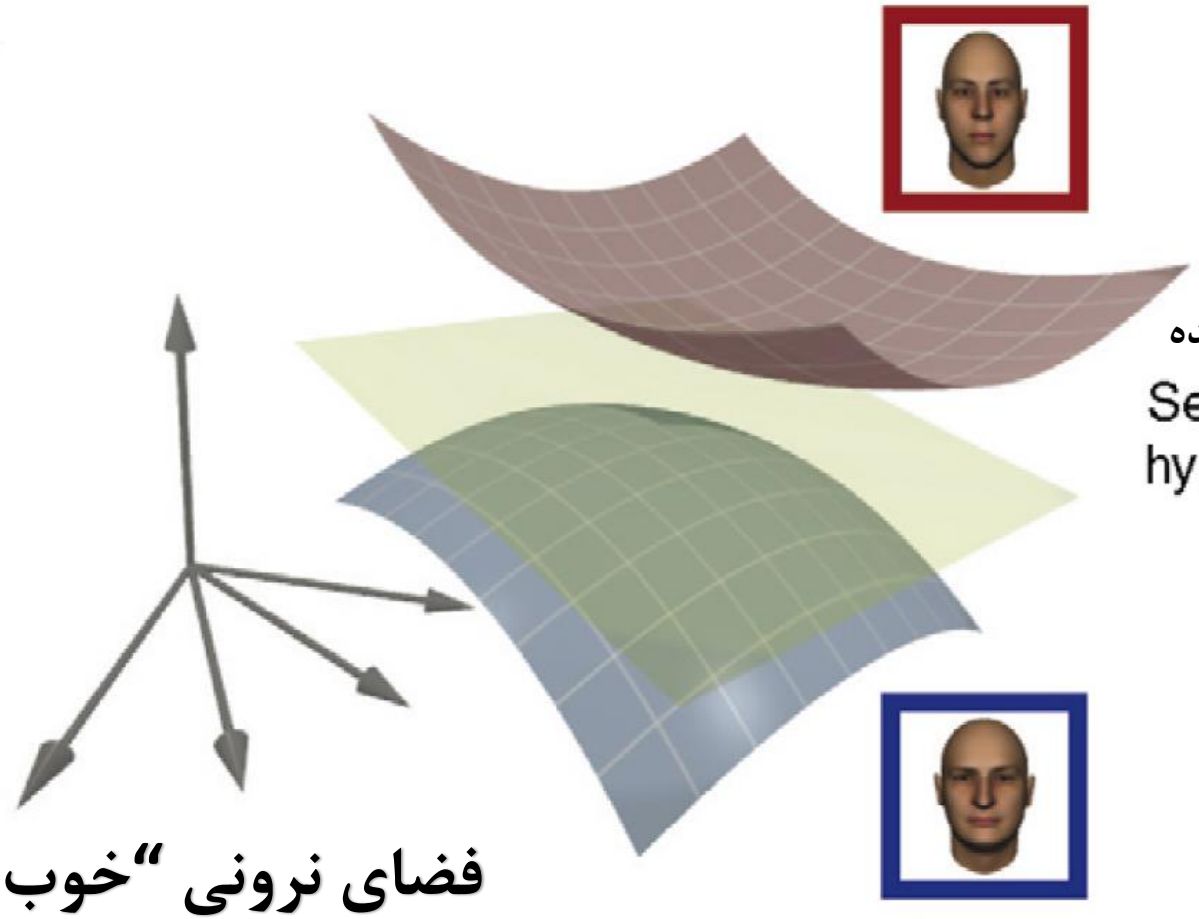
درون دسته‌ای
(intraclass)

# فضای نرونی و خمینه (manifold)




DiCarlo and Cox , **TICS** (2007)

# فضای نرونی خوب

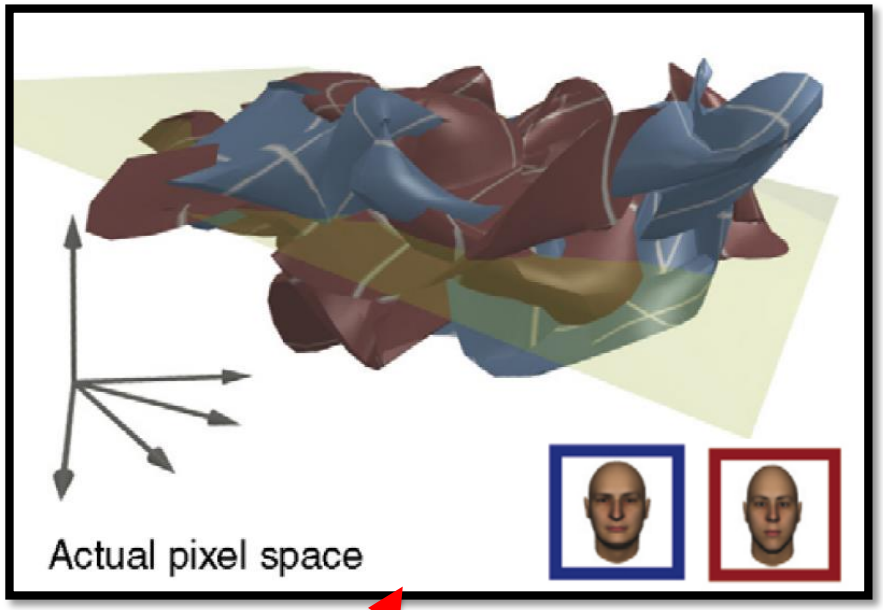


ابر صفحه‌ی جدا کننده

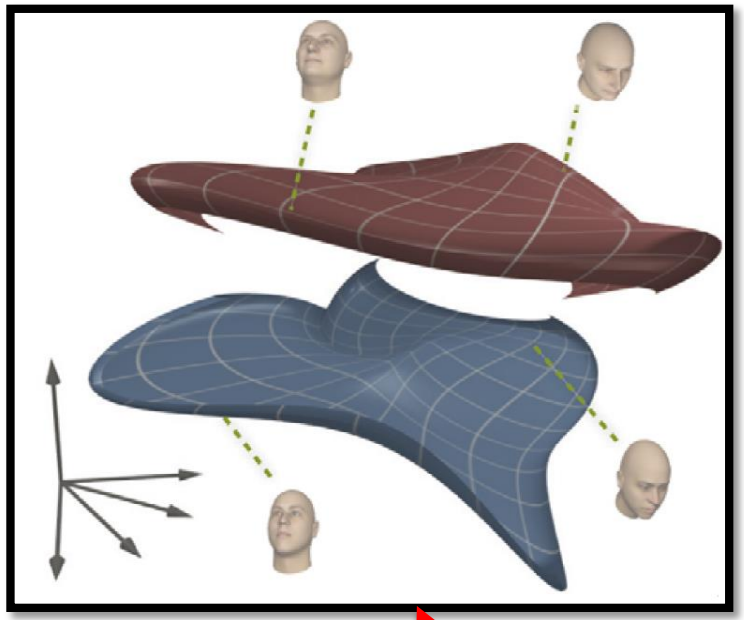
Separating hyperplane

فضای نرونی "خوب"

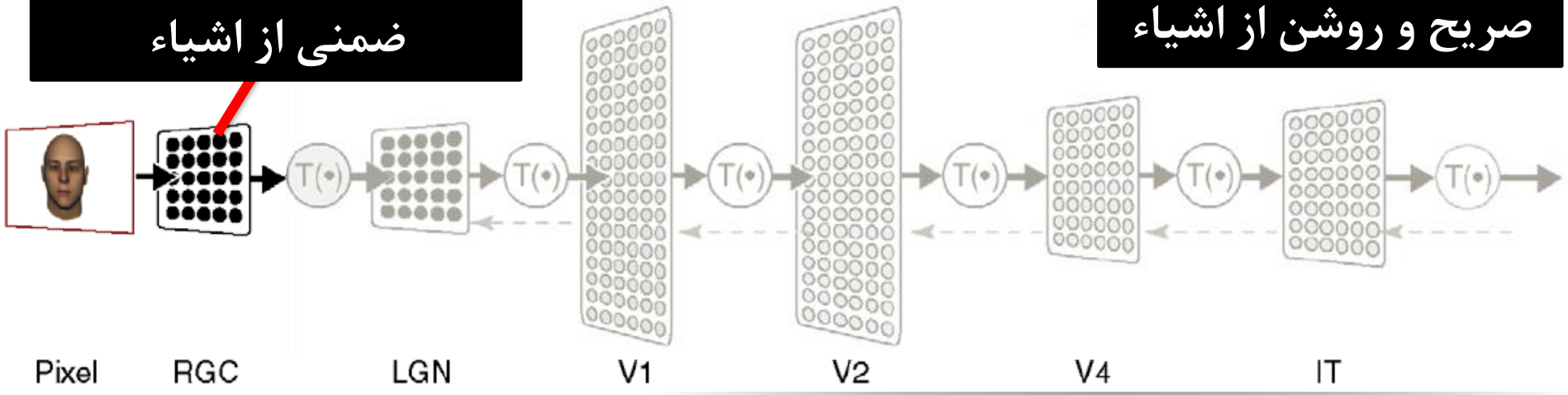
# تبدیل فضای نورونی در مغز




در هم تنیده، اطلاعات ضمنی از اشیاء



جداپذیر، اطلاعات صریح و روشن از اشیاء




*DiCarlo and Cox , **TICS** (2007), DiCarlo, Zoccolan and Rust, **Neuron** (2012)*

# Transfer Learning

“You need a lot of a data if you want to train/use CNNs”

# Transfer Learning

“You need a lot of a data if you want to train/use CNNs”

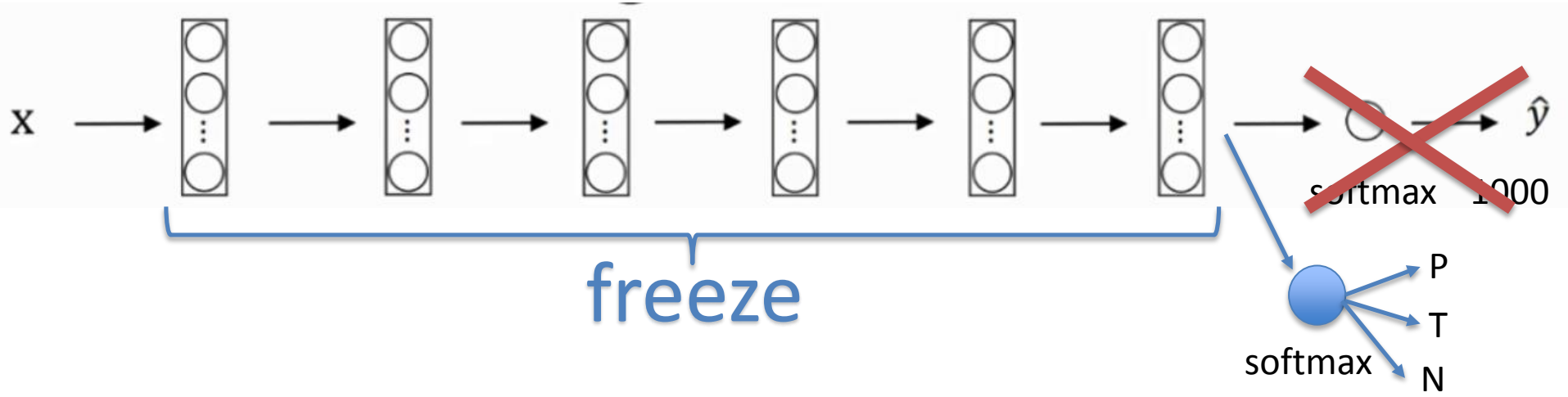
# Transfer Learning

Persian Cat

Tiger cat

?

Neither



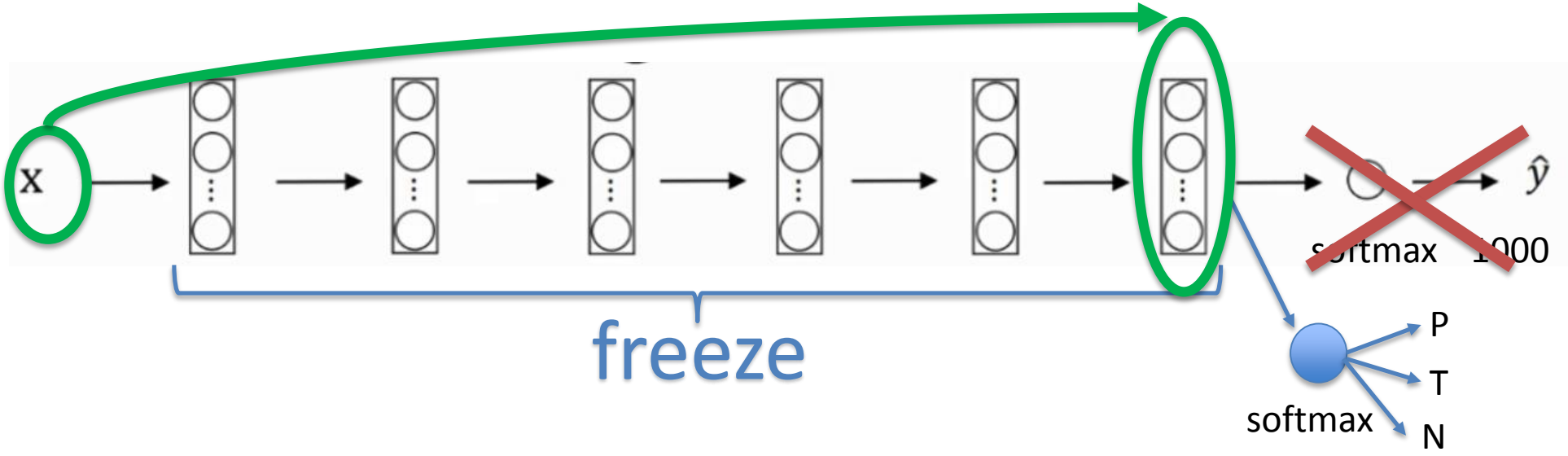
# Transfer Learning

Persian Cat

Tiger cat

?

Neither




- میتوان از قبل خروجی لایه را برای تصاویر آموزشی حساب کرد.

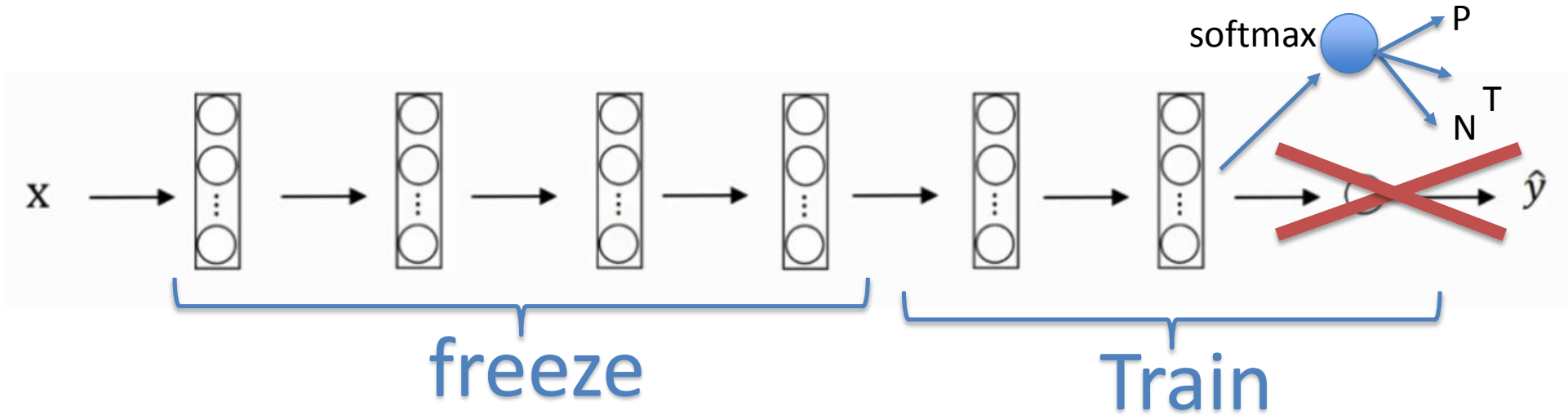
# Transfer Learning

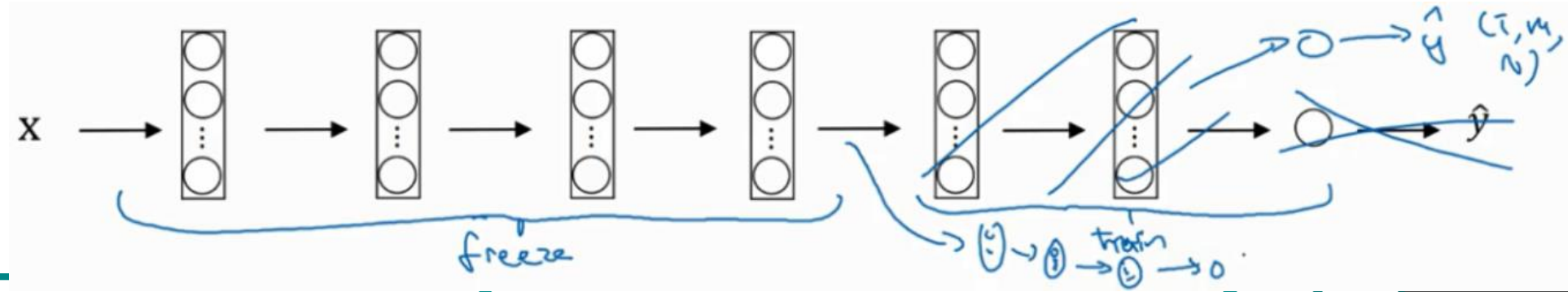
Persian Cat

Tiger cat

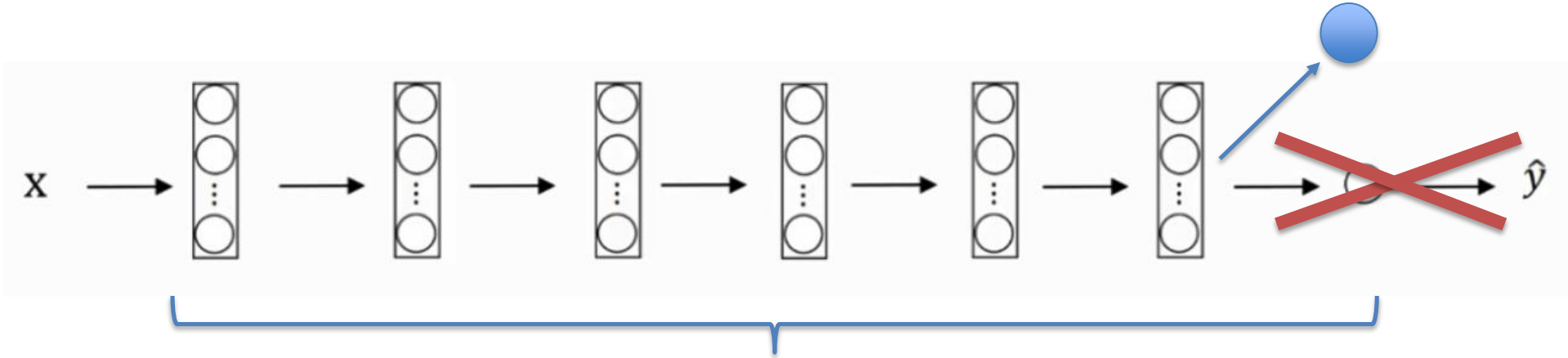
?

Neither



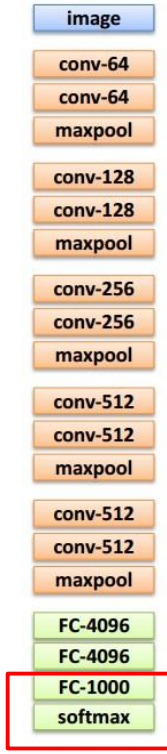



# Transfer Learning



# Transfer Learning with CNNs


1. Train on Imagenet




2. Small dataset: **feature extractor**

Freeze these

Train this


3. Medium dataset: **finetuning**

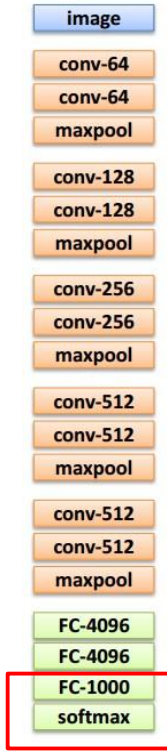
more data = retrain more of the network (or all of it)

Freeze these

Train this

# Transfer Learning with CNNs


1. Train on Imagenet




2. Small dataset: **feature extractor**

Freeze these

Train this


3. Medium dataset: **finetuning**

more data = retrain more of the network (or all of it)

Freeze these

Train this

# Transfer Learning(1)


**12_Transfer_learning_feature_extraction.ipynb**

# Transfer Learning(2)


**13_Transfer_learning_Fine_tuning.ipynb**

# منابع


- *https://www.coursera.org/specializations/deep-learning*
- http://cs231n.stanford.edu/
- https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1
- http://blog.class.vision/winter-96-97-syllabus/
- *https://www.slideshare.net/Alirezaakhavanpour/presentation10-68048331*
- *https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/*
- *https://www.datacamp.com/courses/machine-learning-with-tree-based-models-in-python*
- *https://www.amazon.com/Deep-Learning-Python-Francois-Chollet/dp/1617294438*