قسمت اول:
مقدمه ای بر شبکه‌های عصبی و پیاده سازی در

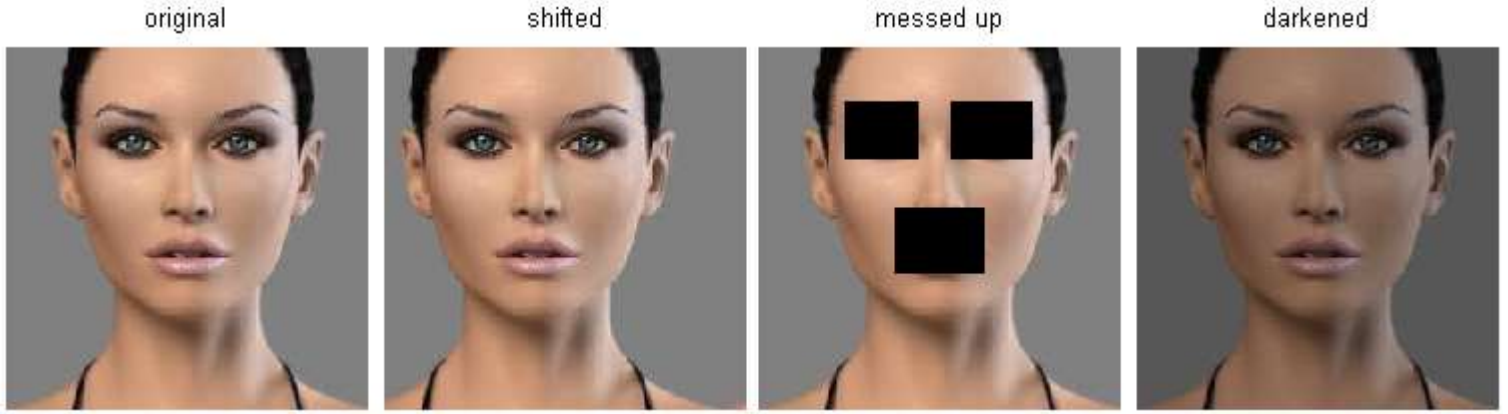




CLASS.VISION

k-Nearest Neighbor on images **never used.**

- distance metrics on level of whole images can be very unintuitive



(all 3 images have same L2 distance to the one on the left)

مقدمه‌ای بر شبکه‌های عصبی

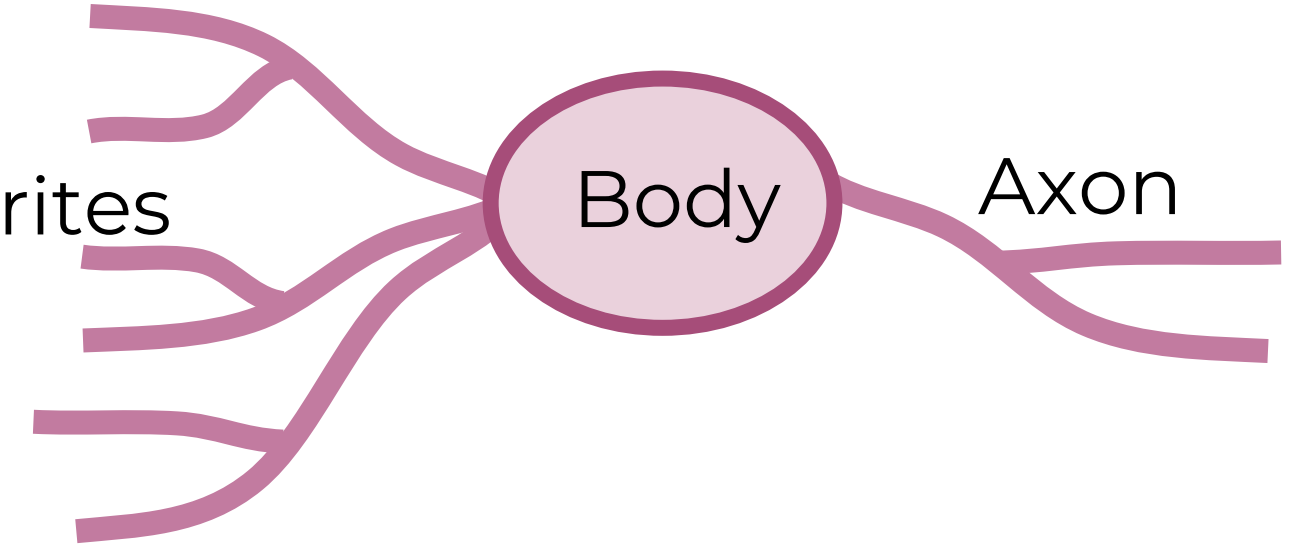
- Neurons and Activation Functions
- Cost Functions
- Gradient Descent
- Backpropagation

مقدمه‌ای بر پرسپترون

Introduction to the Perceptron

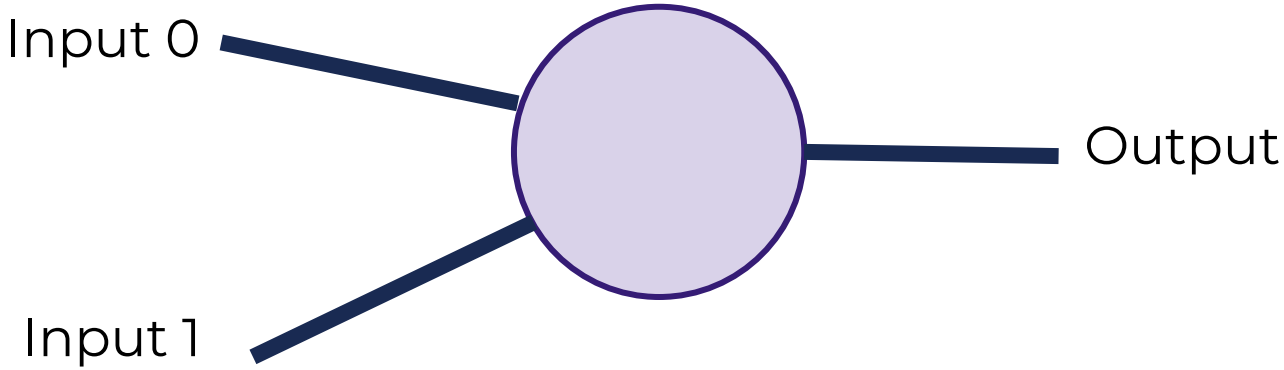
❑ The biological neuron:





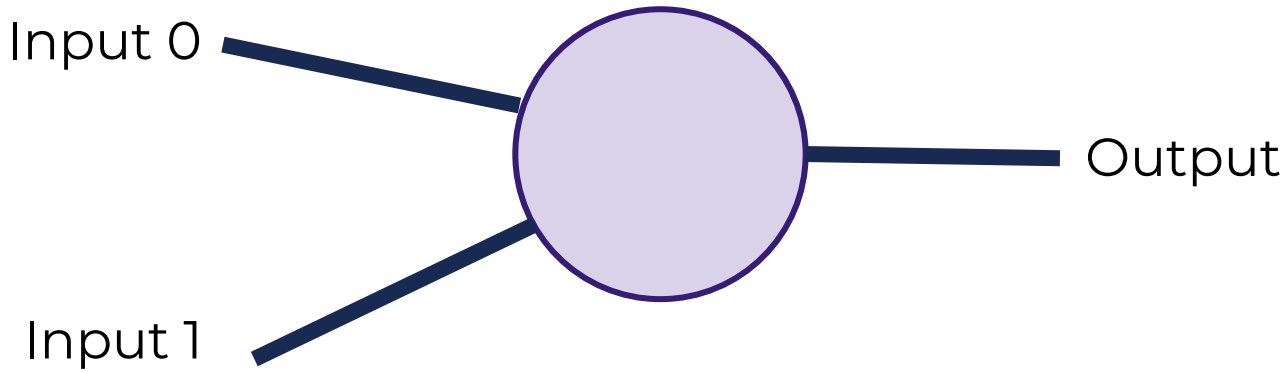
پرسپترون

- نرون‌های مصنوعی هم ورودی و خروجی دارند
- این مدل ساده را پرسپترون می‌نامند.



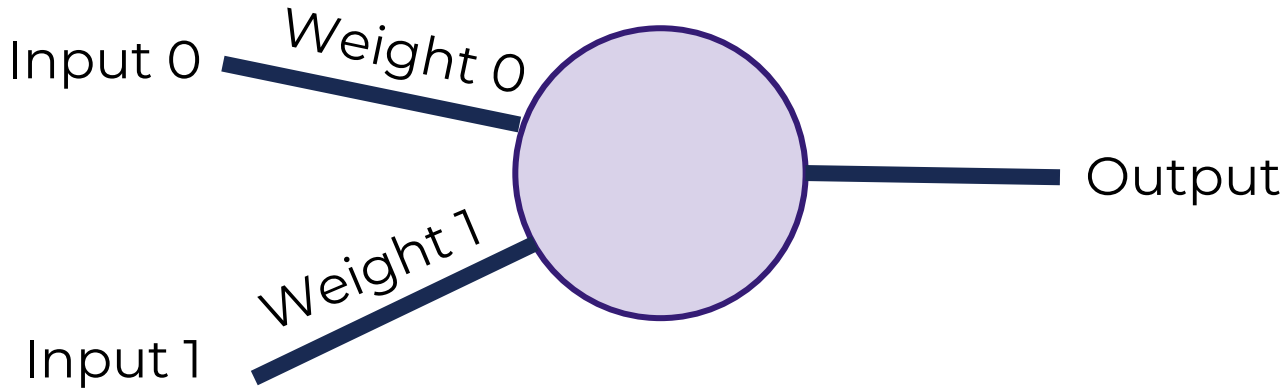
پرسپترون

- ورودی مقادیر ویژگی‌ها (features) هستند



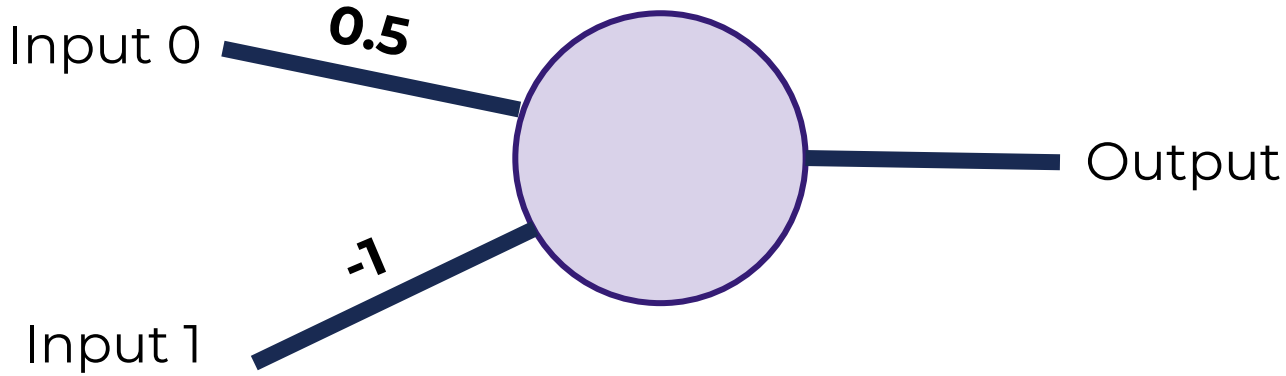
پرسپترون

- ورودی‌ها در وزن‌ها ضرب می‌شوند



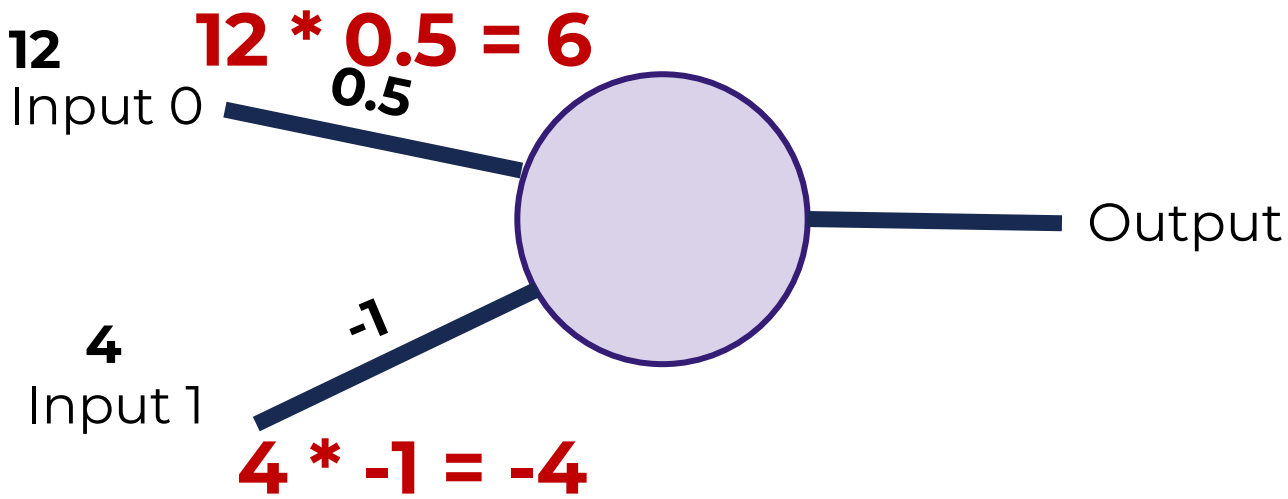
پرسپترون

- وزن‌ها به صورت **تصادفی و رندم** مقداردهی اولیه می‌شوند



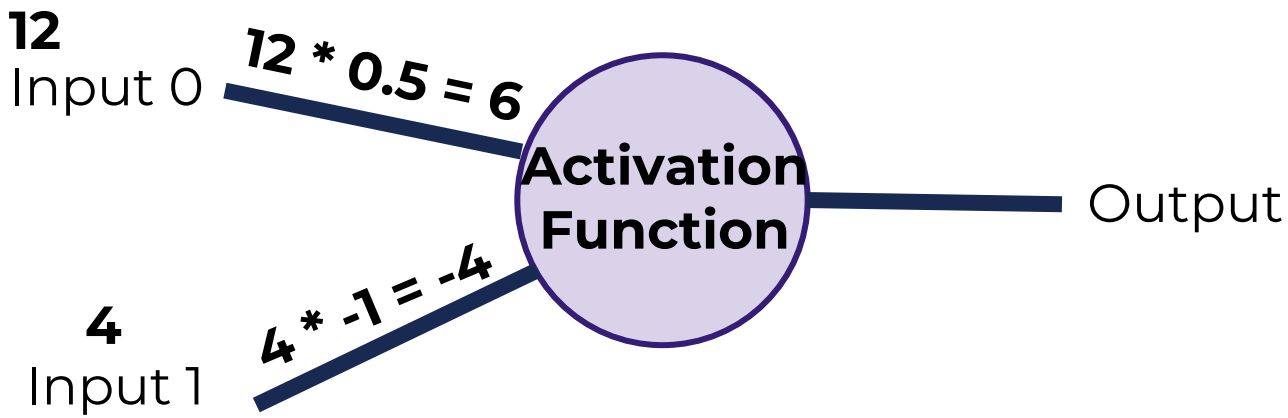
پرسپترون

- ورودی‌ها در وزن‌های متناظر ضرب می‌شوند



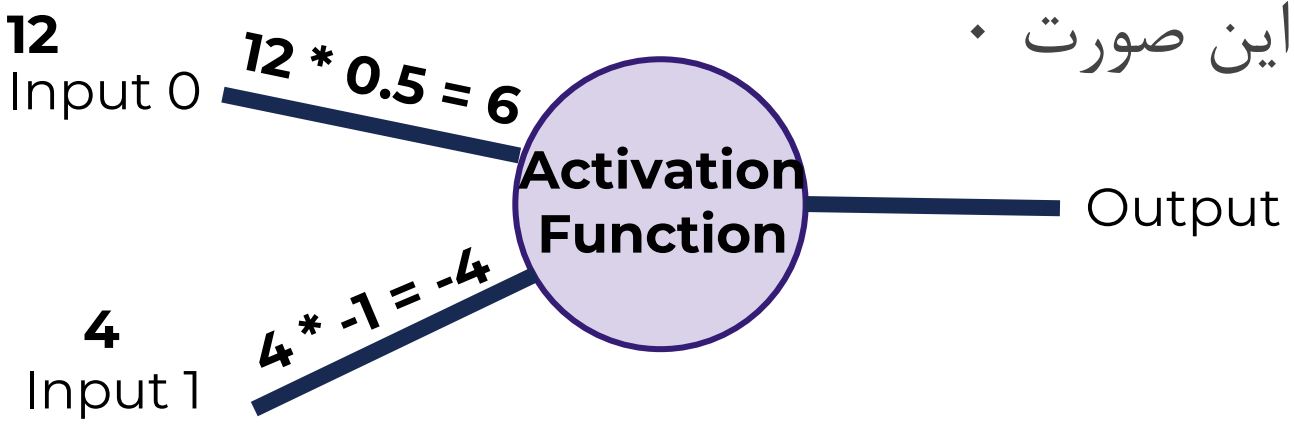
پرسپترون

- یک تابع فعالیت بر روی نتایج اعمال می‌گردد



پرسپترون

- ❑ انتخاب‌های زیادی برای تابع فعالیت داریم
- ❑ ساده ترین تابع فعالیت:
  - اگر مجموع مثبت بود : ۱
  - در غیر این صورت ۰

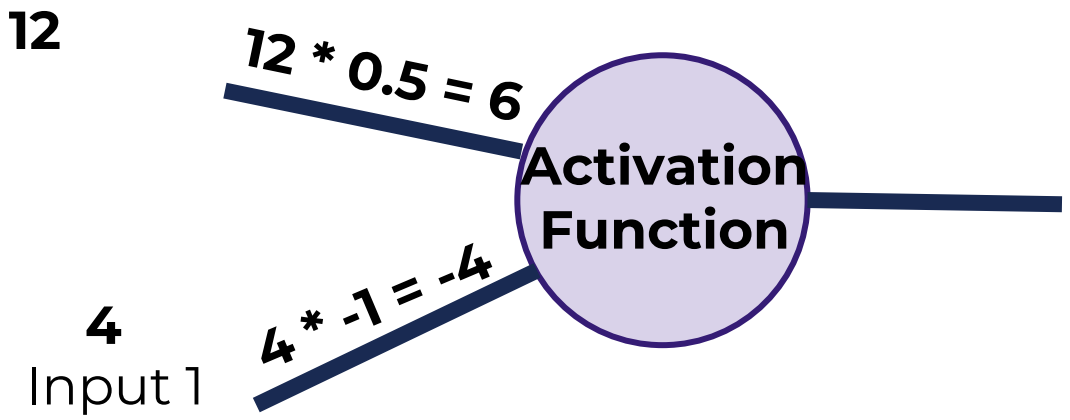


پرسپترون

- در این مثال داریم:

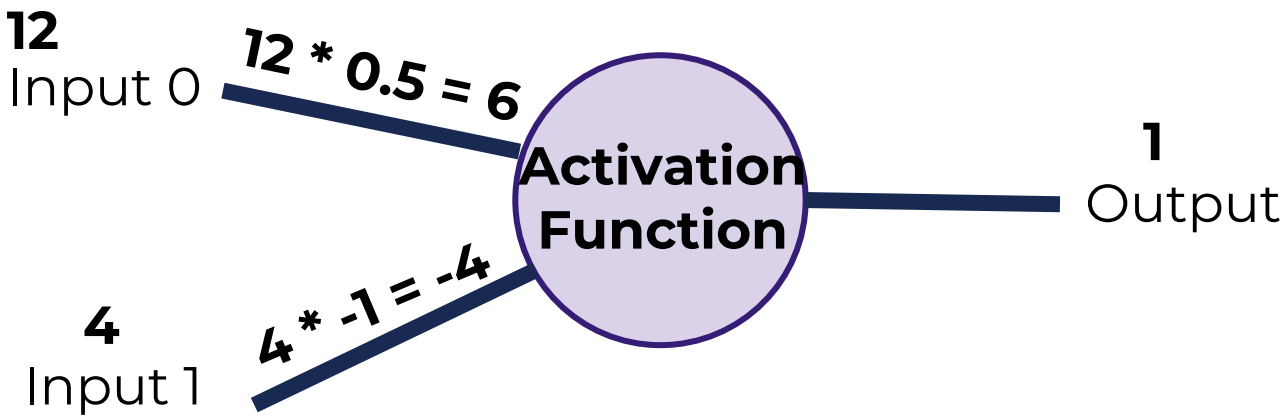
$$6 - 4 = 2$$

پس خروجی برابر است با ۱



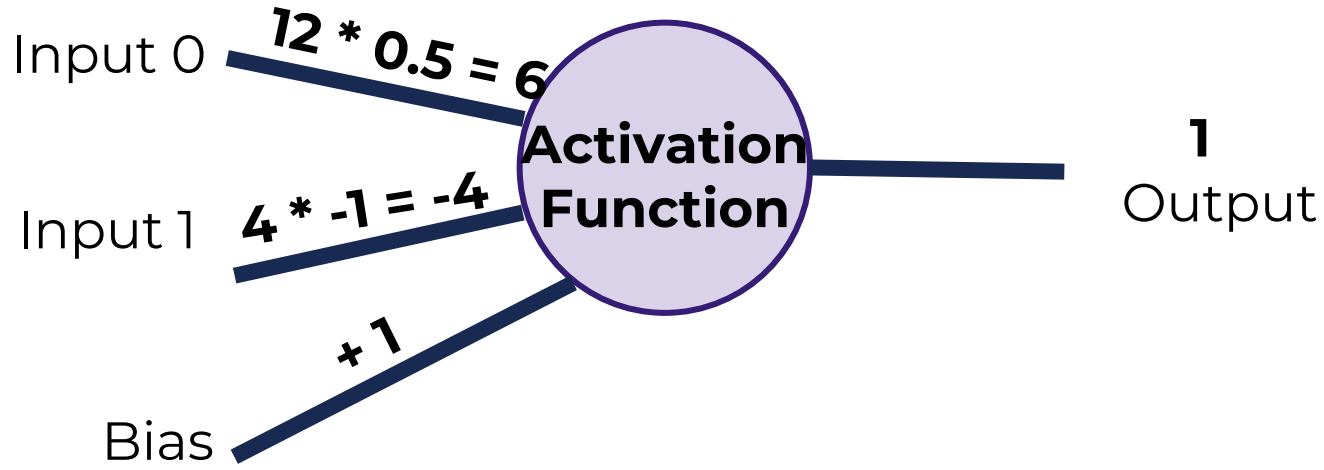
پرسپترون

- یک مشکل! اگر ورودی ۰ باشد؟!
- مستقل از وزن‌ها خروجی ۰ می‌شود!



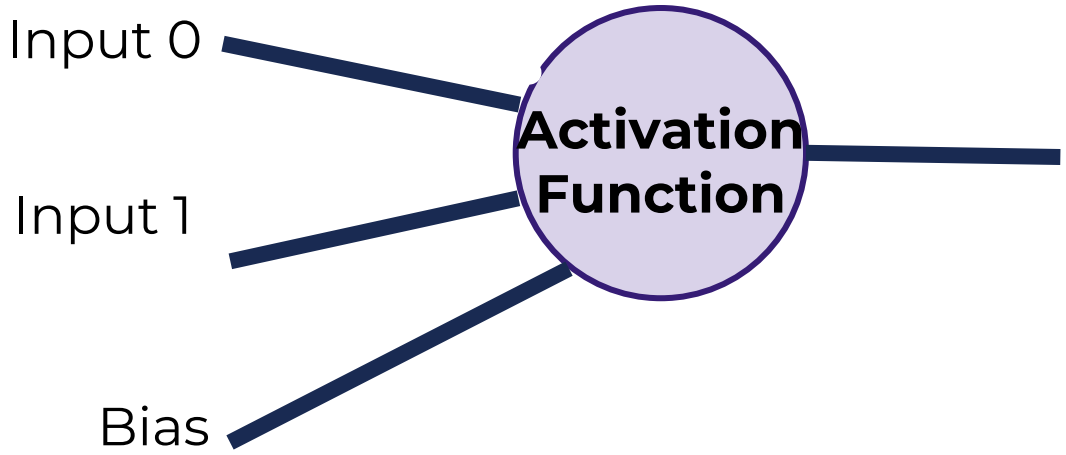
پرسپترون

- این مشکل را با افزودن بایاس رفع می‌کنیم.



پرسپترون

- اگر بخواهیم پرسپترون را به صورت ریاضی بنویسیم...



$$\sum_{i=0}^{n} w_i x_i + b$$

آنچه در بخش اول دیدیم...

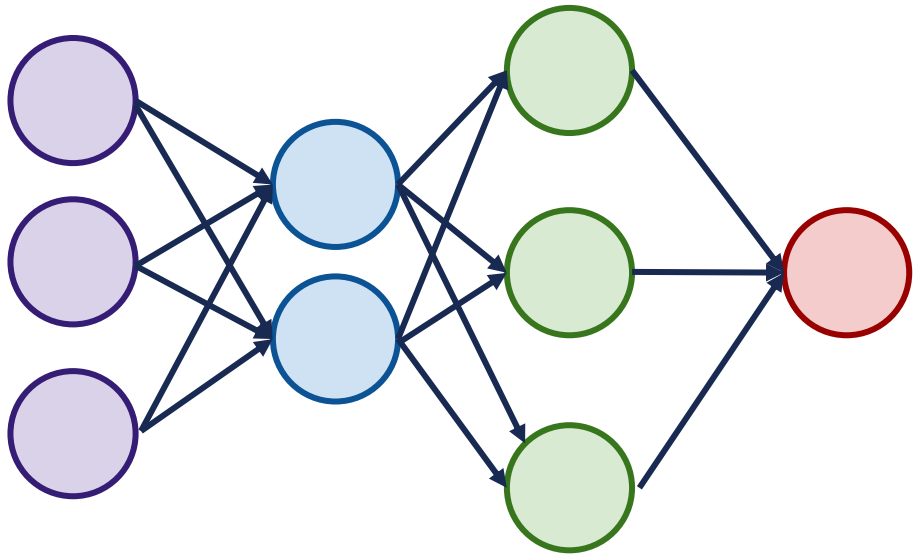
- Biological Neuron
- Perceptron Model
- Mathematical Representation

مقدمه‌ای بر شبکه‌های عصبی

Introduction to Neural Networks

مقدمه‌ای بر شبکه‌های عصبی

- شبکه‌ای از چندین پرسپترون



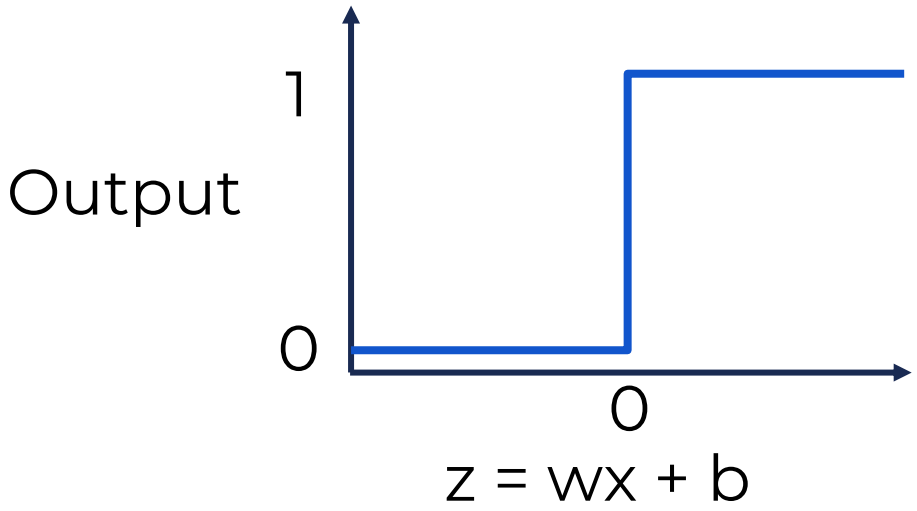
Input Layer. 2 hidden layers. Output Layer

مقدمه‌ای بر شبکه‌های عصبی – لایه‌ها

- **لایه ورودی:**
  - مقادیر حقیقی داده‌ها
- **لایه پنهان:**
  - لایه های بین ورودی و خروجی
  - به ۳ یا بیشتر لایه معمولا شبکه عمیق می‌گوییم.
- **لایه خروجی:**
  - تخمین نهایی شبکه از خروجی است

مقدمه‌ای بر شبکه‌های عصبی – تابع فعالیت

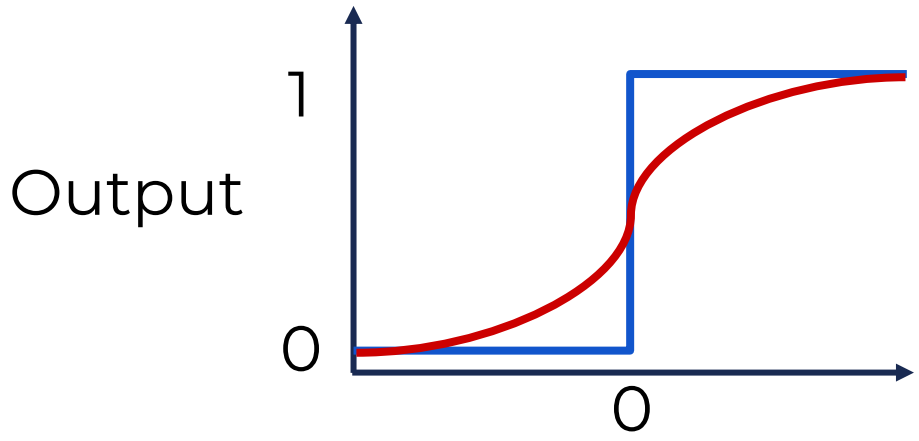
- تابع فعالیت ساده ای که تا کنون دیدیم



- تغییرات کوچک منعکس نمی شوند.

مقدمه‌ای بر شبکه‌های عصبی – تابع فعالیت

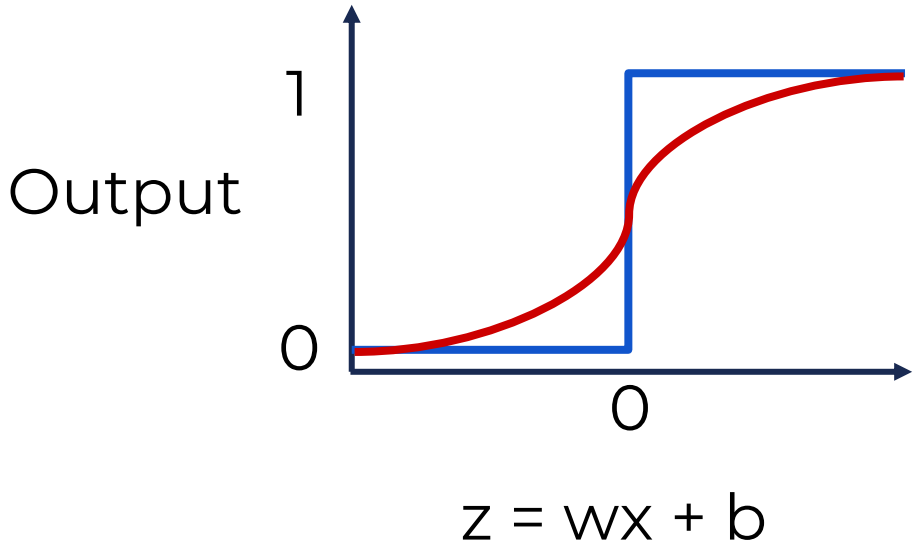
- اگر تابعی پویاتر داشتیم بهتر میشد، مثل تابع قرمز رنگ!



$$z = wx + b$$

مقدمه‌ای بر شبکه‌های عصبی – تابع فعالیت

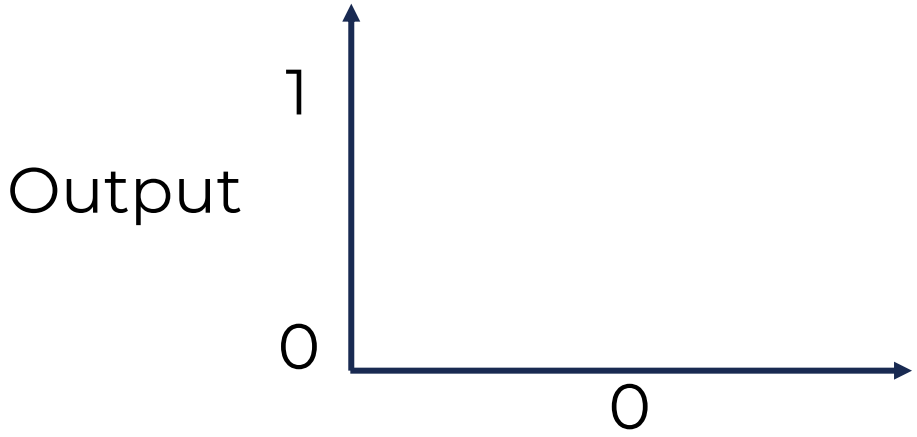
- تابع سیگموئید



$$f(x) = \frac{1}{1 + e^{-(x)}}$$

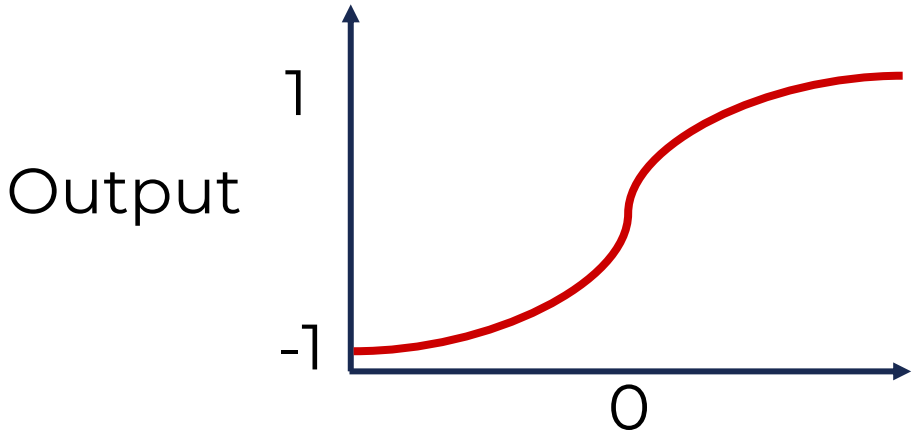
مقدمه‌ای بر شبکه‌های عصبی – تابع فعالیت

- بسته به کار ممکن است از توابع فعالیت متفاوتی استفاده کنیم
- توابع فعالیتی که ممکن است با آنها رو به رو شویم را ببینیم ...



مقدمه‌ای بر شبکه‌های عصبی – تابع فعالیت

- Hyperbolic Tangent: tanh(z)



$$z = wx + b$$

$$\cosh x = \frac{\mathbf{e}^{x} + \mathbf{e}^{-x}}{2}$$

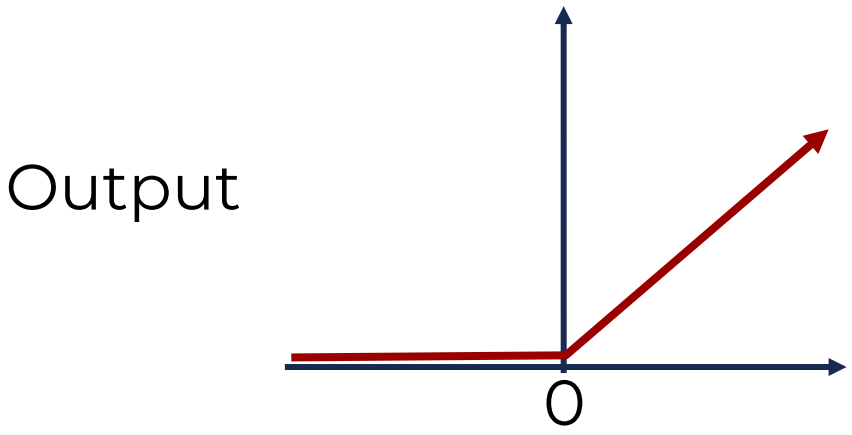
$$\sinh x = \frac{\mathbf{e}^{x} - \mathbf{e}^{-x}}{2}$$

$$\tanh x = \frac{\sinh x}{\cosh x}$$

مقدمه‌ای بر شبکه‌های عصبی – تابع فعالیت

❑ ReLU - واحد یکسوساز خطی

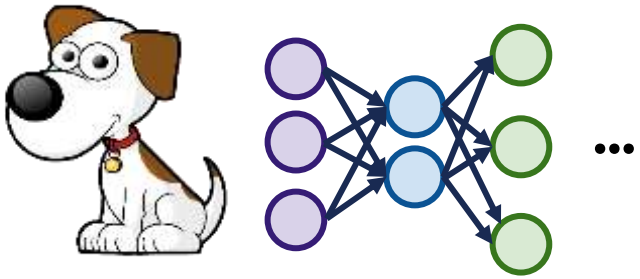
- Rectified Linear Unit



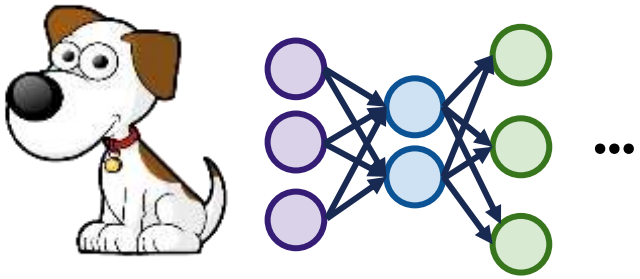
max(0,z)

تابع هزینه

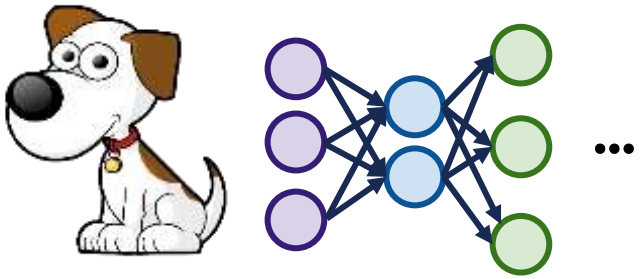
Cost Function



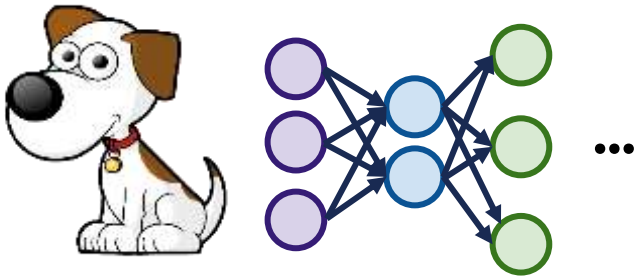
	Y
Cat	0.2
Dog	0.6
human	0.2



		Y	Y’
Cat	○	0.2	0
Dog	○	0.6	1
human	○	0.2	0



	Y	Y’
Cat	0.2	0
Dog	0.6	1
human	0.2	0



	Y		Y’	
Cat	0.2	-	0	0.2
Dog	0.6	-	1	0.4
human	0.2	-	0	0.2

تابع هزینه

- **معیاری برای اندازه گیری بد بودن مدل!**

  بد بودن؟! تفاوت خروجی مدل با خروجی مورد انتظار

- **از علائم و متغیرهای زیر استفاده میکنیم:**

  Y’: نمایانگر خروجی مورد انتظار (true value)

  Y : تخمین حاصل از خروجی شبکه

- **محاسبه a از وزن‌ها و بایاس**

  - w*x + b = **z**
  - σ(**z**) = **y** => Pass z into activation function

تابع هزینه

- تابع هزینه ساده برای طبقه‌بندی:

$$C = \Sigma(y'-y)^2 / n$$

- خطاهای بزرگتر به علت توان ۲ ، برجسته تر هستند.
- این تابع خطا فرآیند آموزش را کند می‌کند.

تابع هزینه

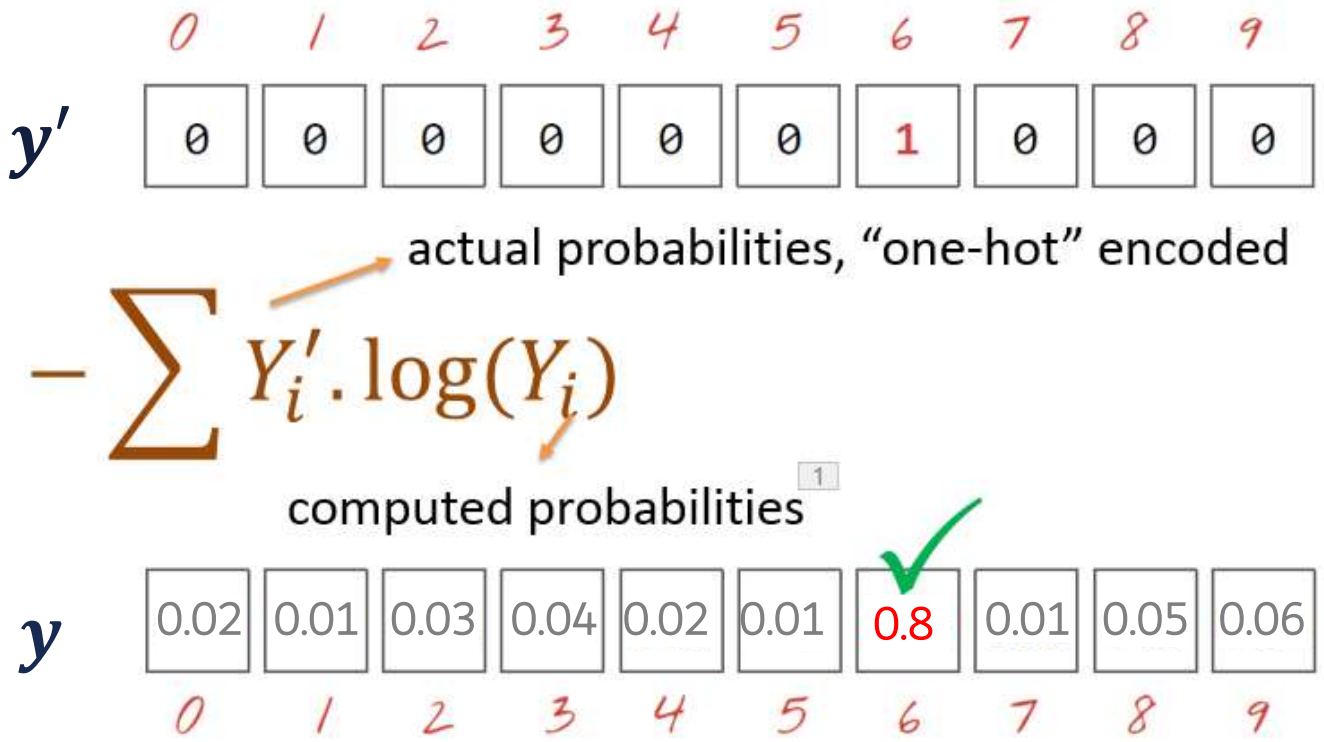
- **Cross Entropy**

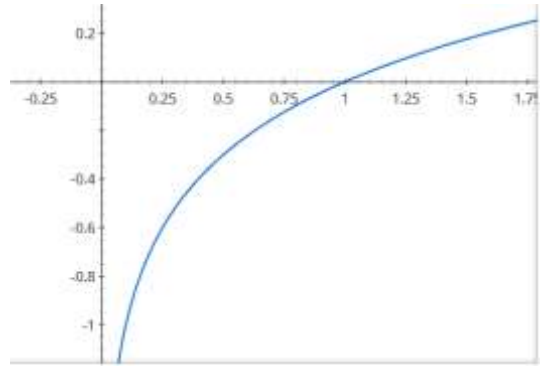
$$C = -\sum_i y_i' \log(y_i)$$

- این تابع اجازه می‌دهد که آموزش تسریع شود.
- هر چه تفاوت بزرگتر باشد، سریعتر نورون می تواند یاد بگیرد.

تابع هزینه

Cross Entropy ❑





Log (x)

تابع هزینه

- تا الان با ۲ جنبه کلیدی یادگیری شبکه های عصبی آشنا شدیم:
  - ✓ نورون ها با تابع فعالیت
  - ✓ تابع هزینه
- ما هنوز یک گام کلیدی را کم داریم! "*یادگیری*"!

بخش چهارم:

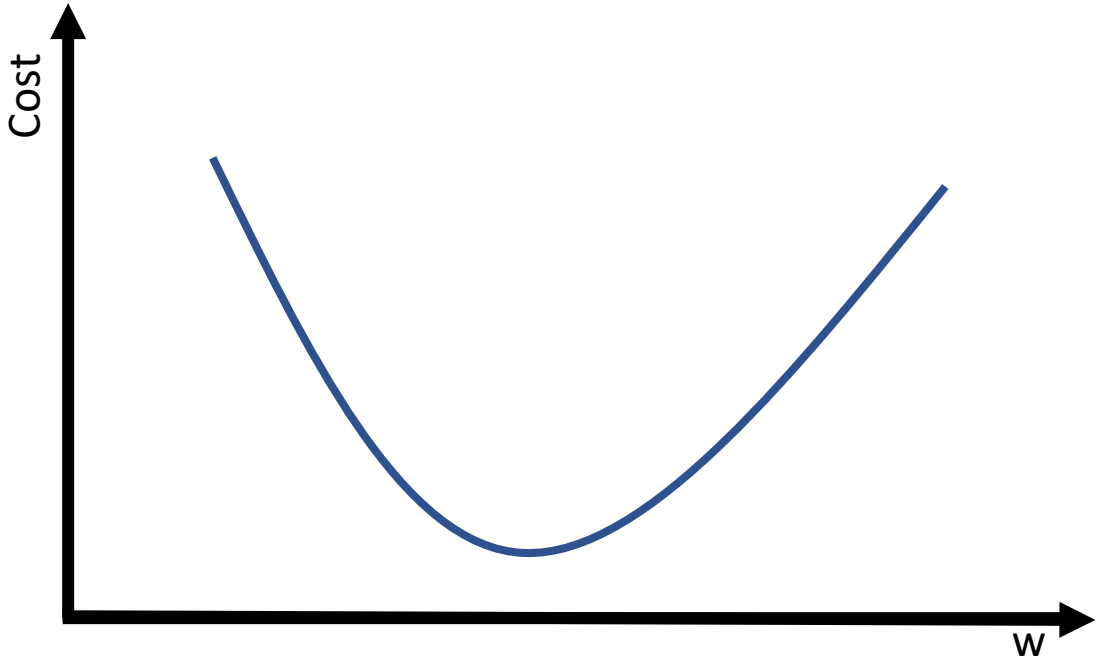
پس‌انتشار خطا و گرادیان کاهشی

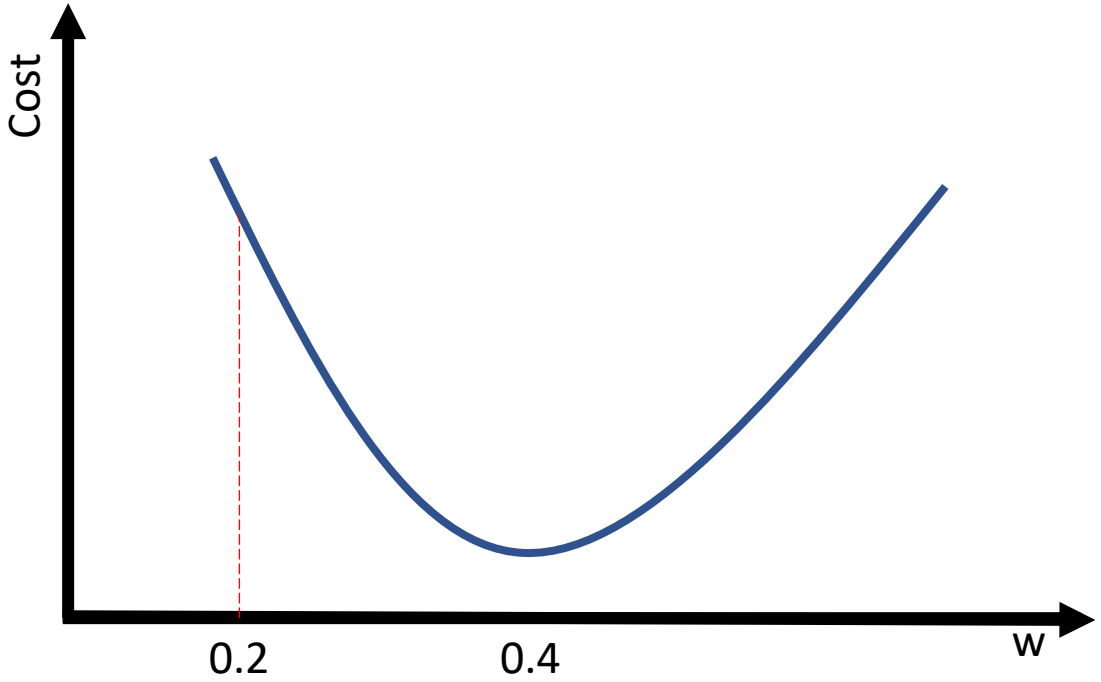
Gradient Descent and Backpropagation

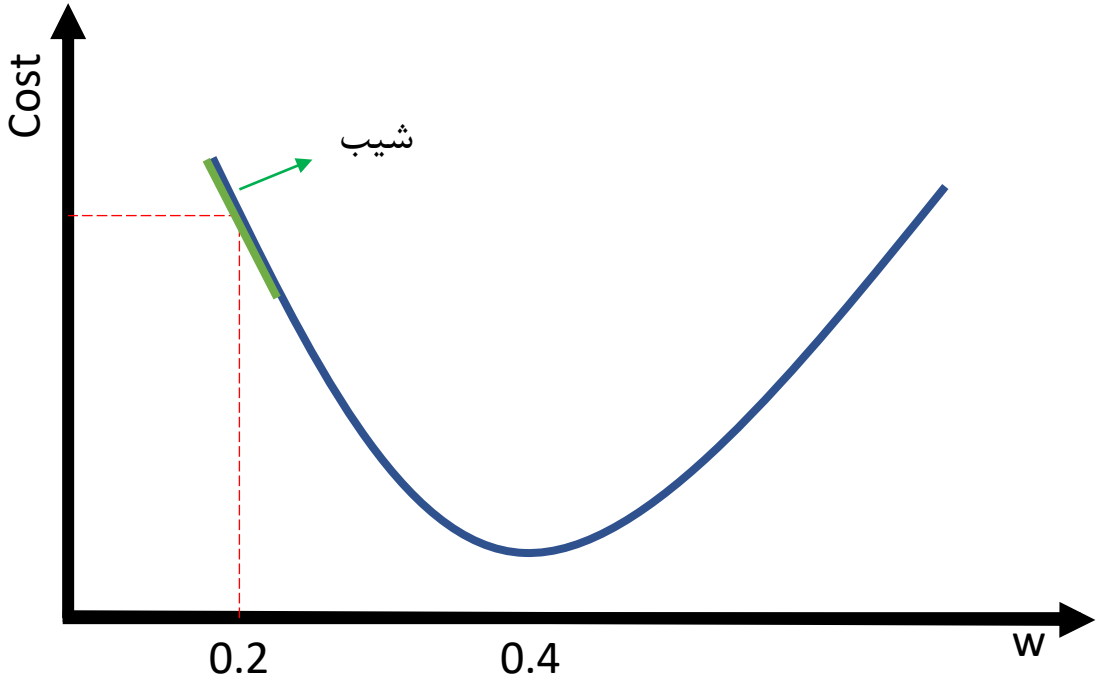
پس‌انتشار خطا و گرادیان کاهشی

- **گرادیان کاهشی** یا Gradient descent
  - یک الگوریتم بهینه سازی برای یافتن حداقل یک تابع است.

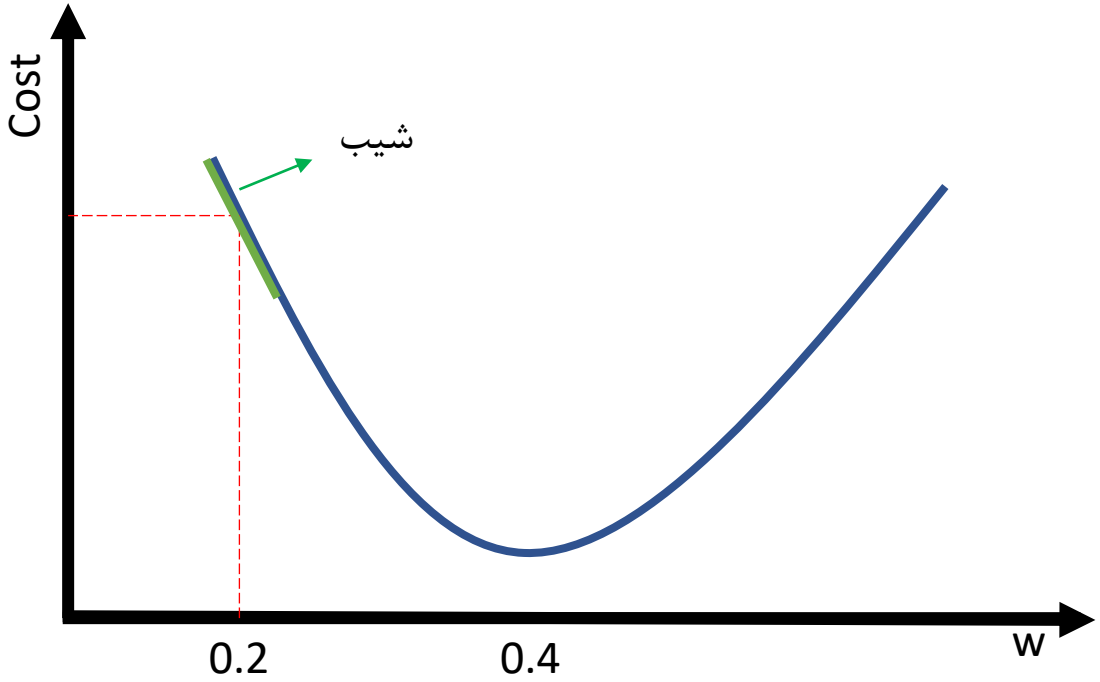
- برای یافتن **کمینه‌ی محلی** یک تابع با استفاده از این الگوریتم، **گام‌هایی** متناسب با **منفی گرادیان** تابع در محل فعلی برداشته خواهد شد.


Cost
w


Cost
0.2
0.4
w

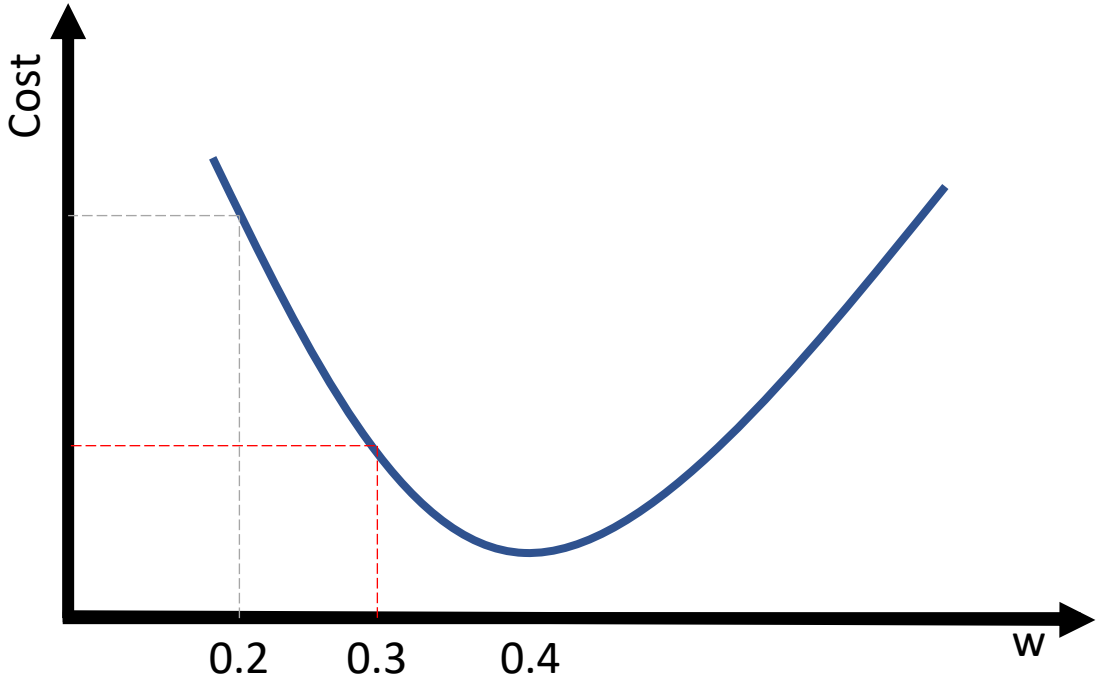


$$W = W \; - \text{شیب}$$



$$W = W \ - \text{شیب}$$

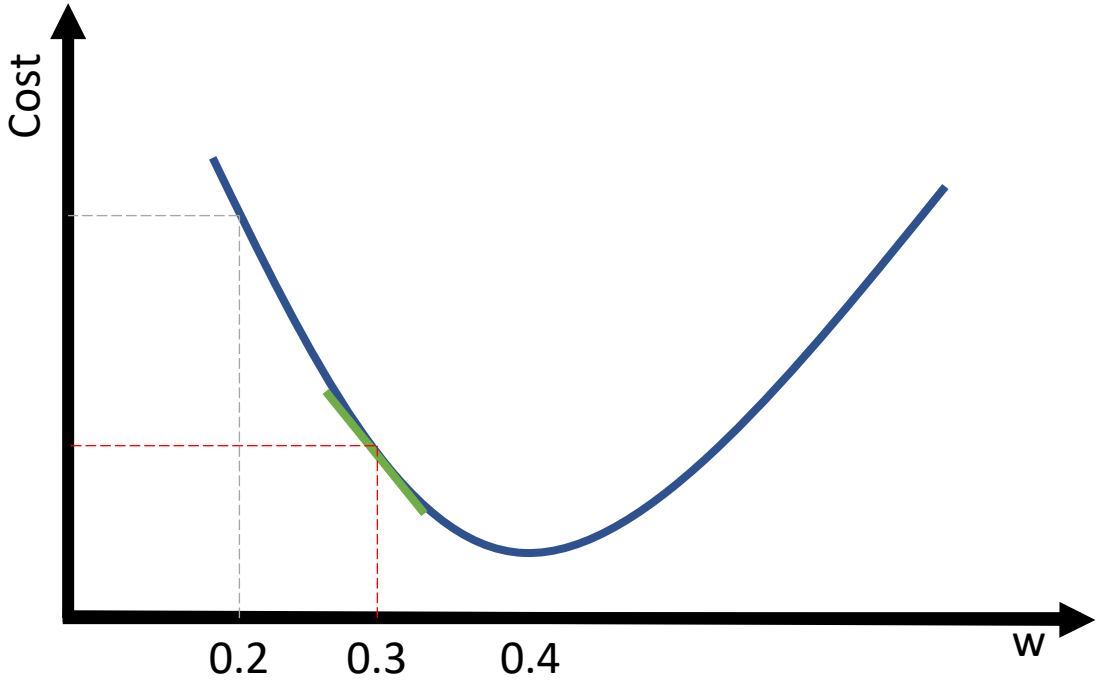
$$W = 0.2 \ - (-0.1)$$



$$W = W \ - \text{شیب}$$

$$W = 0.2 \ - (-0.1)$$

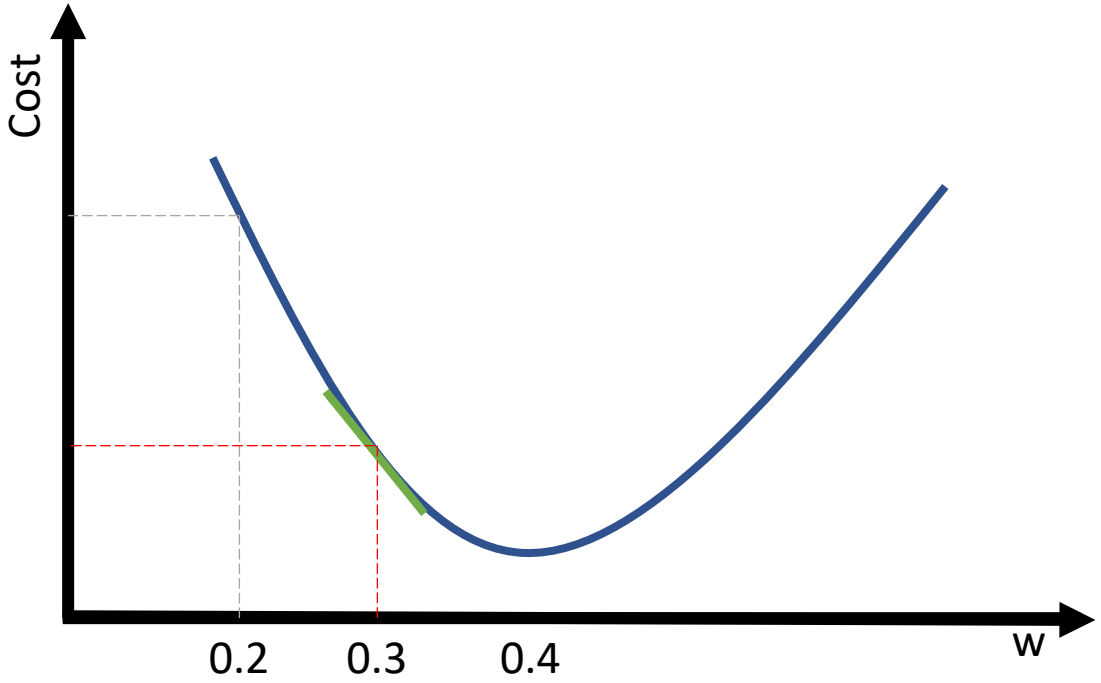
$$W = 0.3$$



$$W = W \ - \text{شیب}$$

$$W = 0.2 \ - (-0.1)$$

$$W = 0.3$$

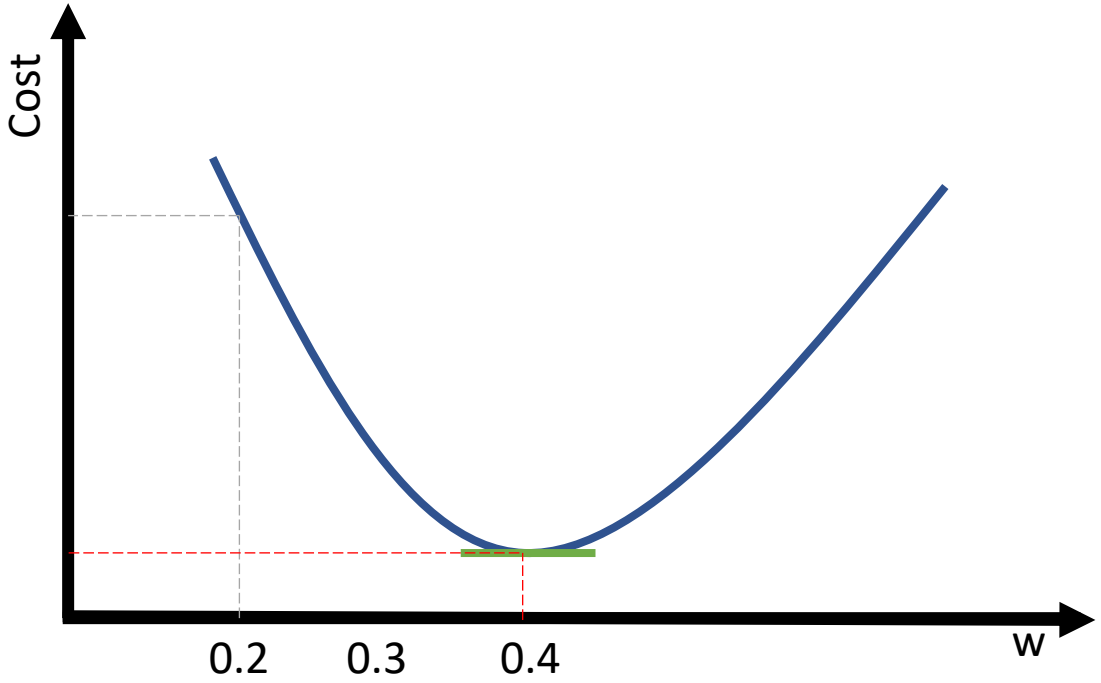


$$W = W \ - \text{شیب}$$

$$W = 0.2 \ - (-0.1)$$

$$W = 0.3$$

$$W = W \ - \text{شیب}$$



$$W = W \ - \text{شیب}$$

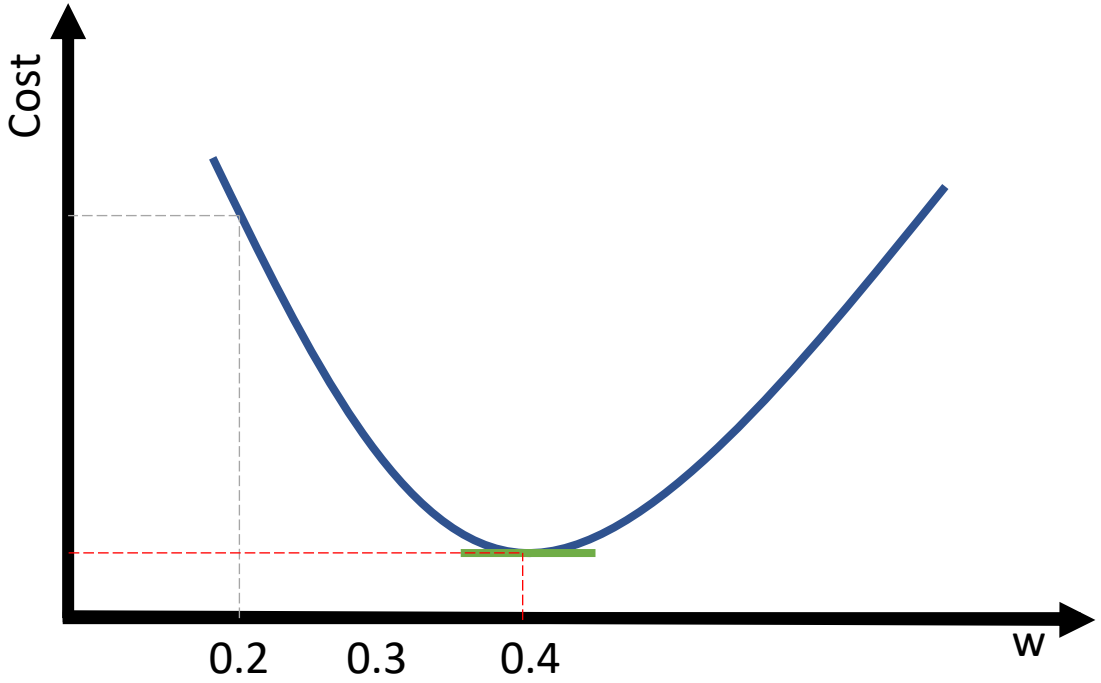
$$W = 0.2 \ - (-0.1)$$

$$W = 0.3$$

$$W = W \ - \text{شیب}$$

$$W = 0.3 \ - (-0.1)$$

$$W = 0.4$$



$$W = W \ - \text{شیب}$$

$$W = 0.2 \ - (-0.1)$$

$$W = 0.3$$

$$W = W \ - \text{شیب}$$

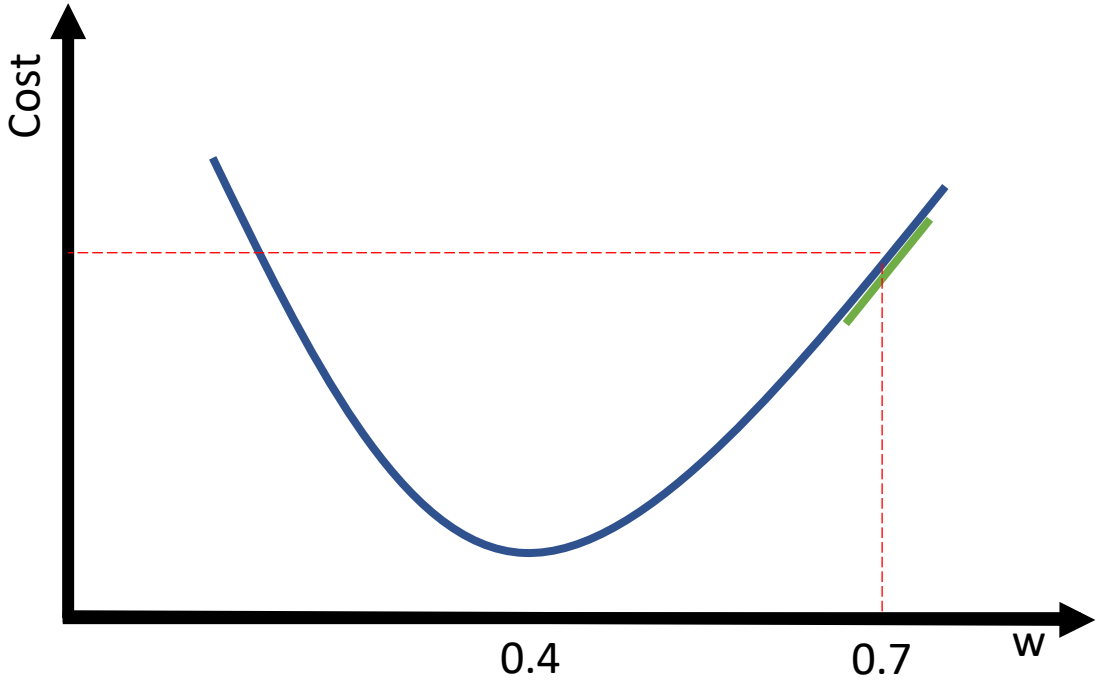
$$W = 0.3 \ - (-0.1)$$

$$W = 0.4$$

$$W = W \ - \text{شیب}$$

$$W = 0.4 \ - 0$$

$$W = 0.4$$

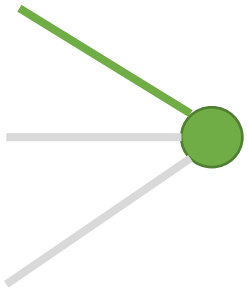


$$W = W \ - \text{شیب}$$

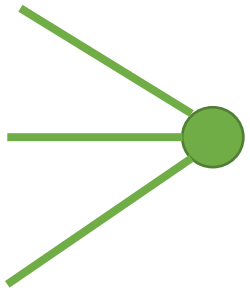
$$W = 0.7 \ - 0.3$$

$$W = 0.4$$

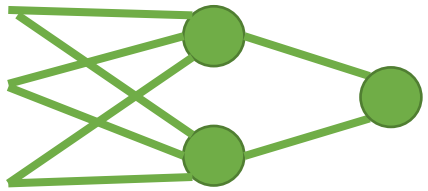
چرا Gradient decent و نه derivative decent



چرا Gradient decent و نه derivative decent

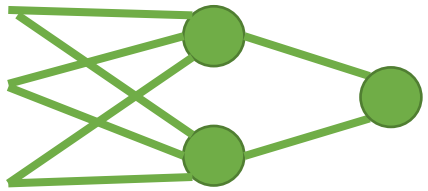


چرا Gradient decent و نه derivative decent



$$C(W_1, W_2, W_3, \dots, W_n)$$

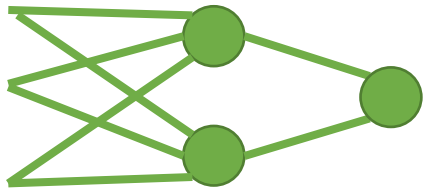
چرا Gradient decent و نه derivative decent



C'

$C(W_1, W_2, W_3, \dots, W_n)$

چرا Gradient decent و نه derivative decent



$C(W_1, W_2, W_3, \dots, W_n)$

~~C'~~

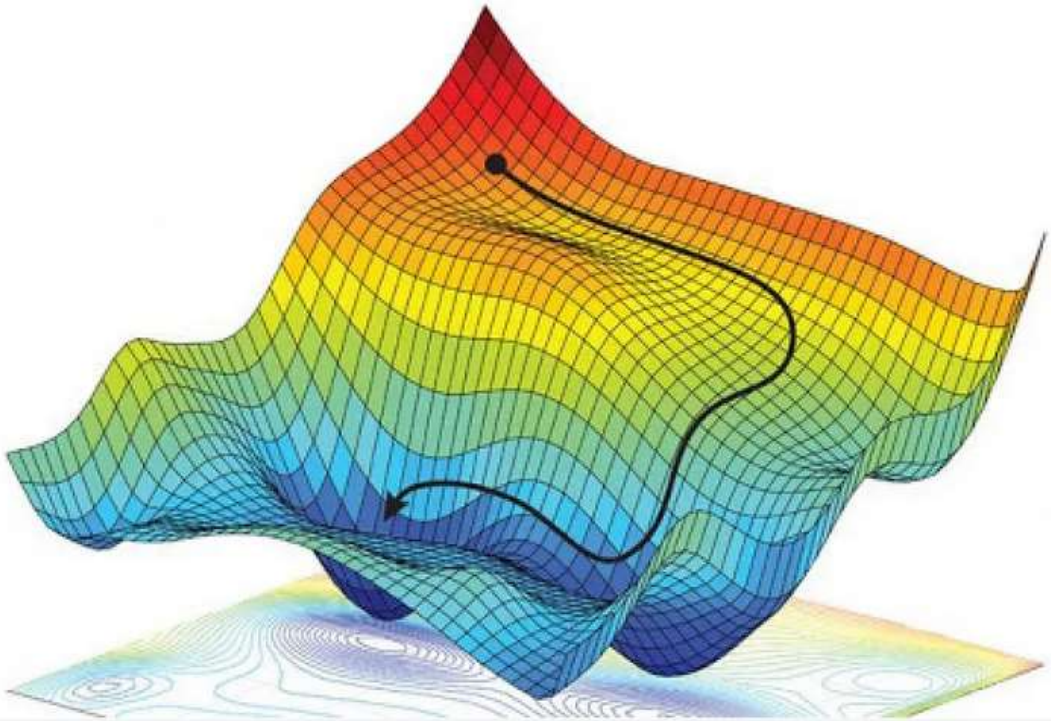
$$\frac{\partial C}{\partial W_1}$$

$$\frac{\partial C}{\partial W_2}$$

$$\vdots$$

$$\frac{\partial C}{\partial W_n}$$

چرا Gradient decent و نه derivative decent



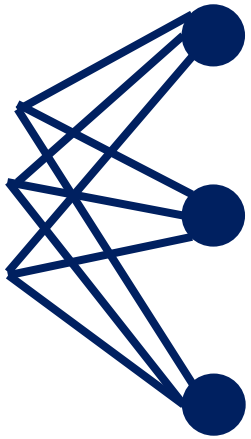
$$\frac{\partial C}{\partial W_1}$$

$$\frac{\partial C}{\partial W_2}$$

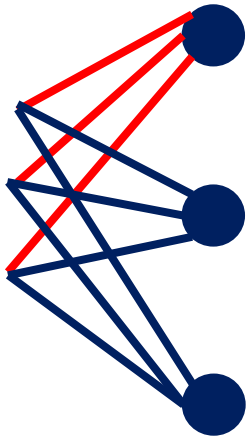
$$\vdots$$

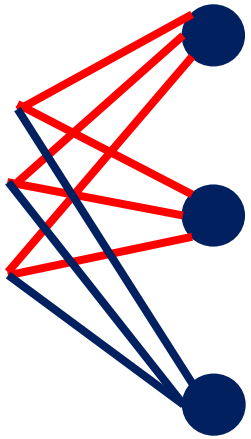
$$\frac{\partial C}{\partial W_n}$$

predicted	Label
0.7	1
0.2	0
0.1	0

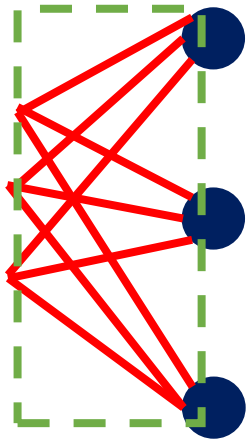


predicted	Label
0.7	1
0.2	0
0.1	0

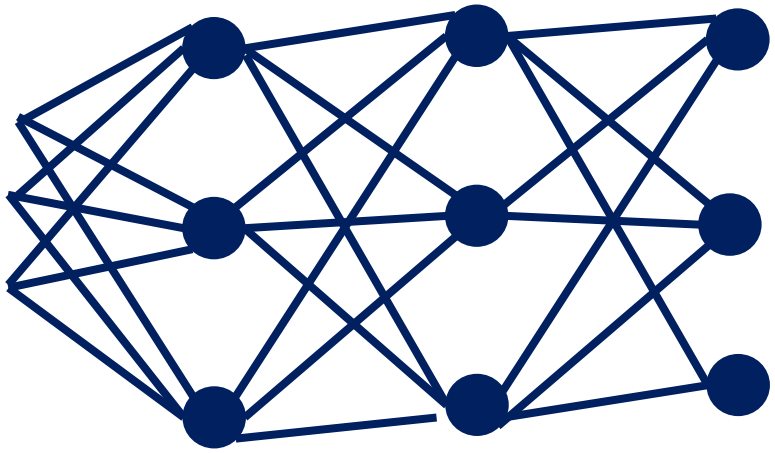




predicted	Label
0.7	1
0.2	0
0.1	0



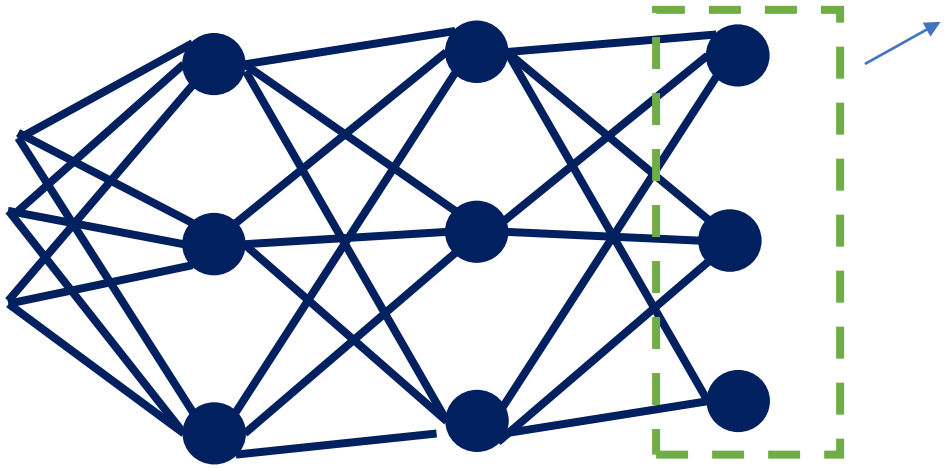
predicted	Label
0.7	1
0.2	0
0.1	0

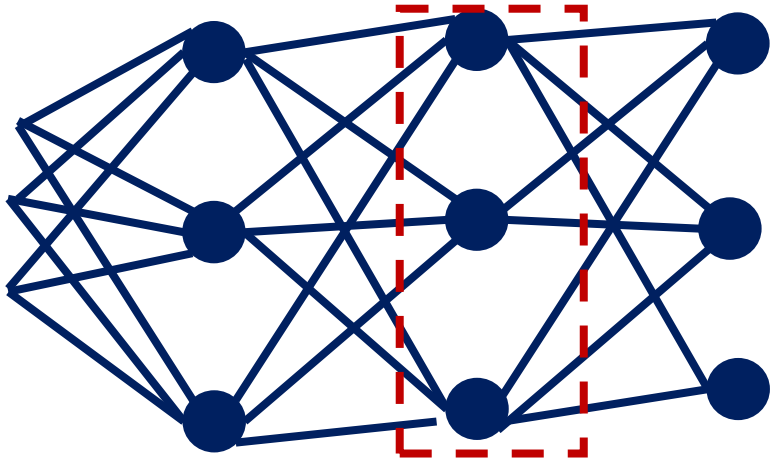


مقدمه ای بر شبکه‌های عصبی و پیاده‌سازی در کتابخانه‌ی Keras

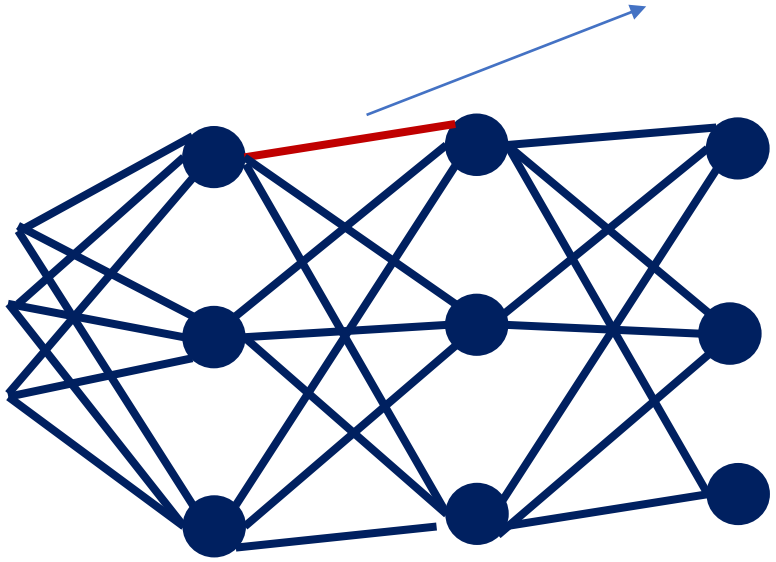
علیرضا اخوان پور

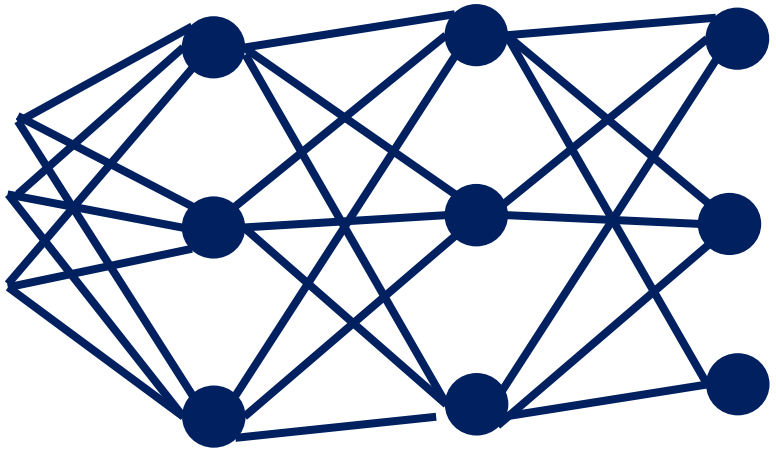
فقط برای لایه آخر علاوه بر خروجی خود نرون ها خروجی
مورد انتظار(لیبل) را نیز داریم



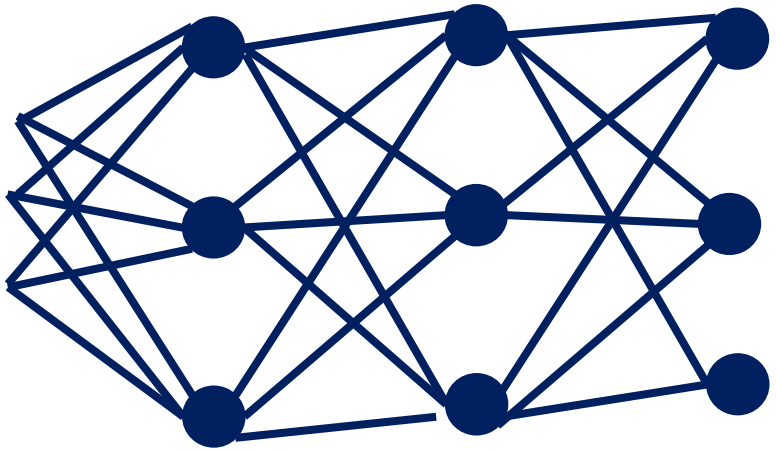


به دنبال روشی هستیم که گرادیان این وزن ها را نسبت
به تابع خطایی نهایی حساب کنیم



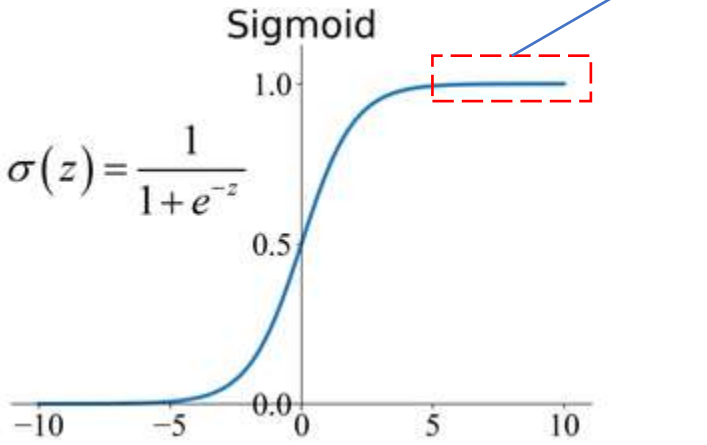


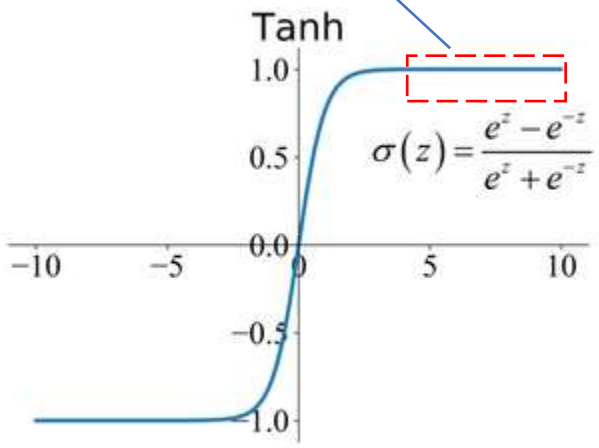
$$\square \times \square \times \square \times \cdots$$

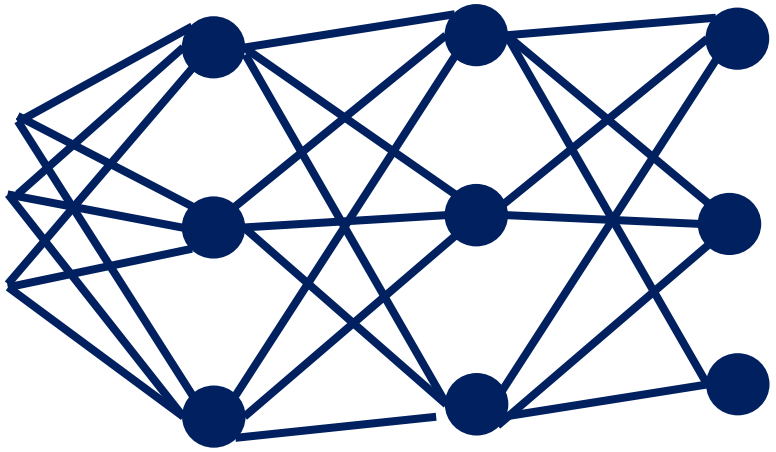


$$\square \times \square \times \square \times \cdots$$

گرادیان(شیب) نزدیک صفر

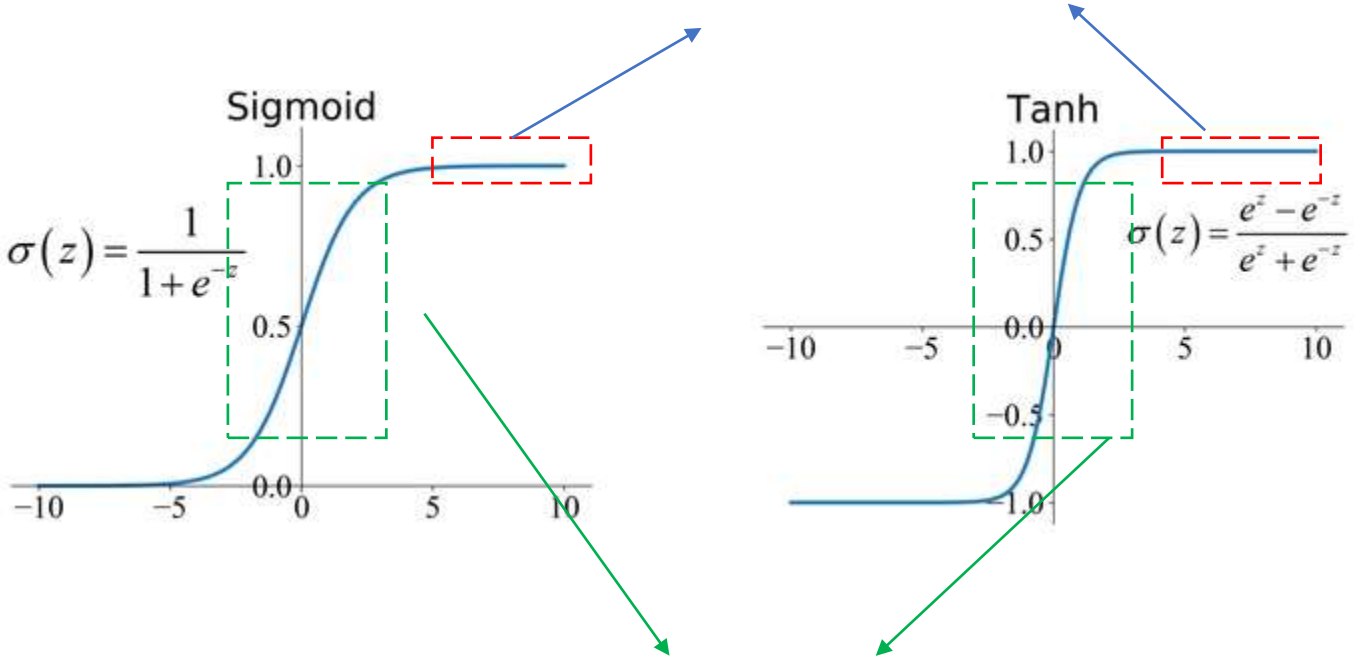




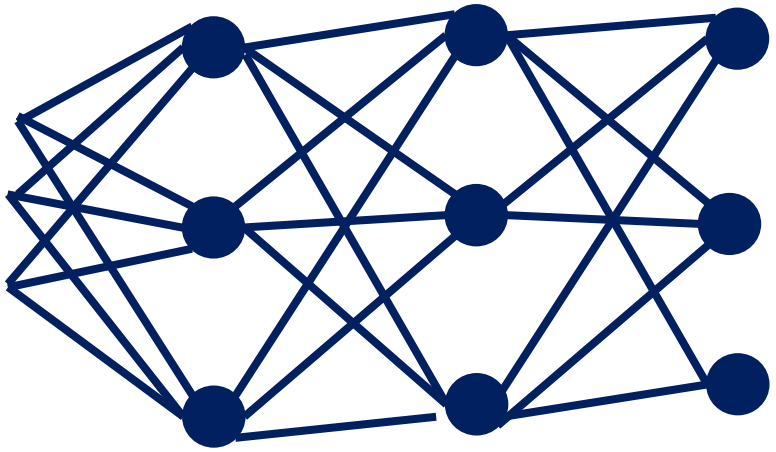


$\square \times \square \times \square \times \cdots$

گرادیان(شیب) نزدیک صفر



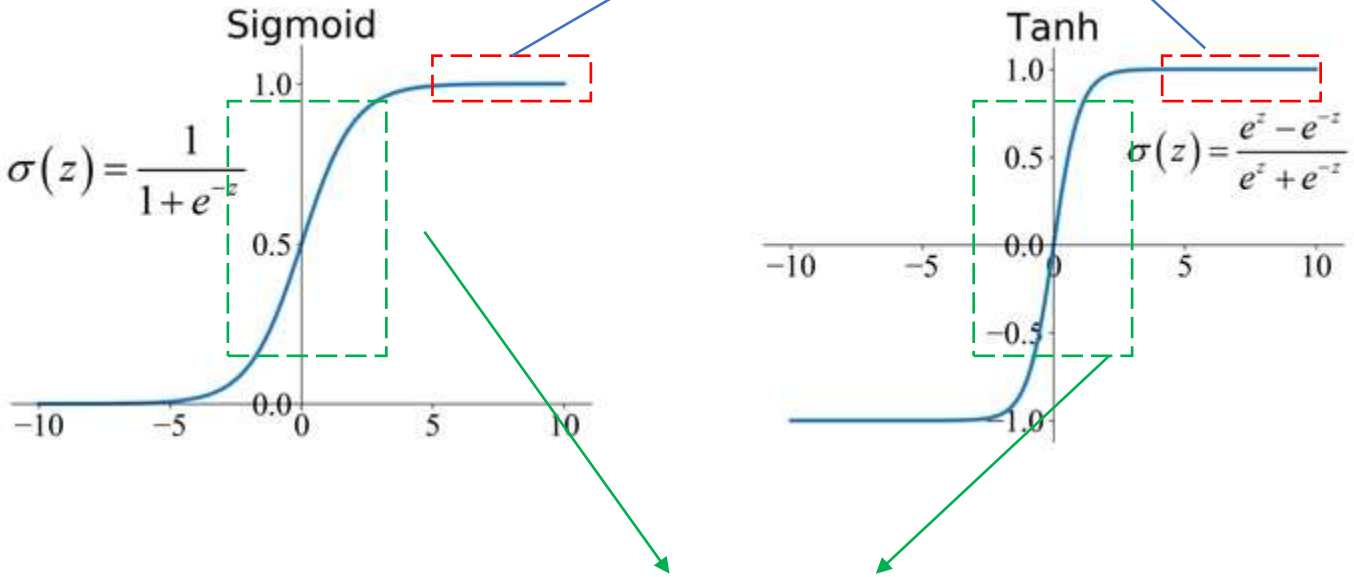
گرادیان مناسب و غیر صفر فقط در برخی از نقاط تابع



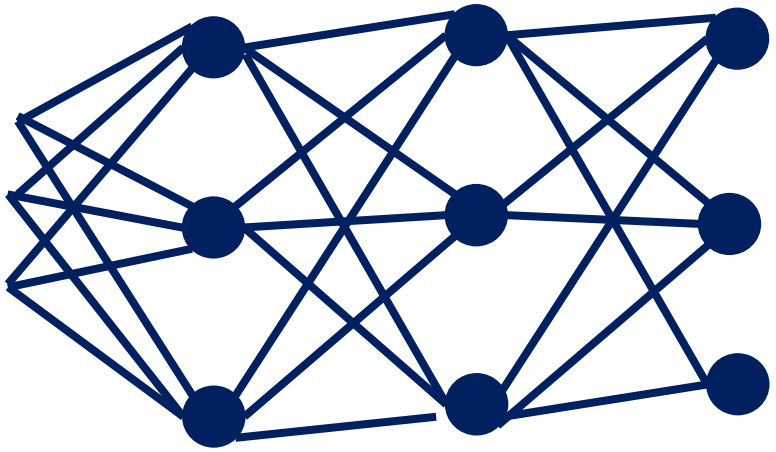
$$W = W - \underbrace{0.01 \text{ x } 0.005 \text{ x } 0.01 \text{ x } \dots \text{ } 0.1}$$

حاصل ضرب تعداد زیادی عدد کوچک نزدیک صفر است

گرادیان(شیب) نزدیک صفر



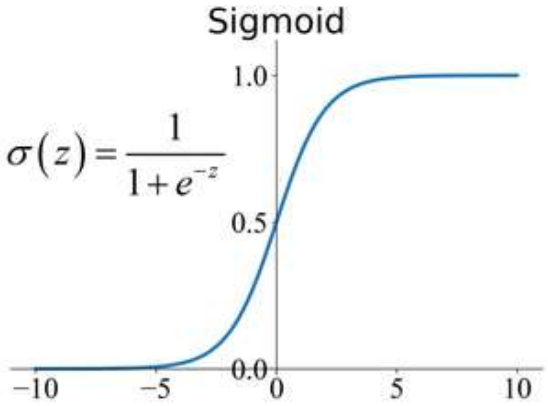
گرادیان مناسب و غیر صفر فقط در برخی از نقاط تابع

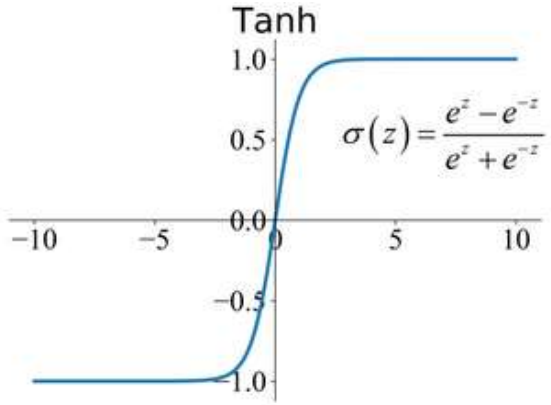


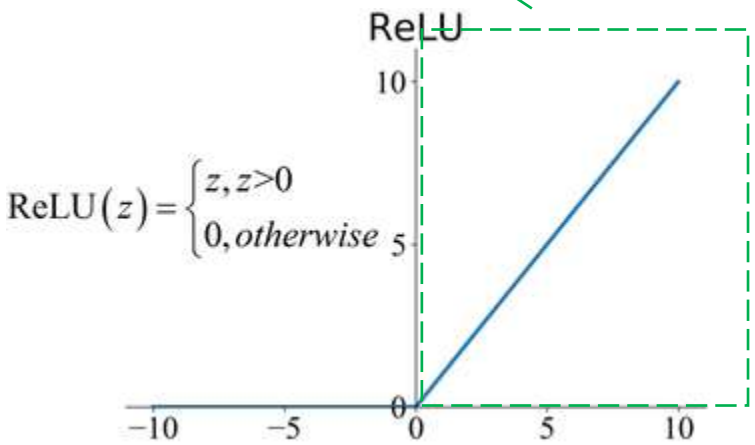
$$W = W - \underbrace{0.01 \text{ x } 0.005 \text{ x } 0.01 \text{ x } \dots \text{ } 0.1}$$

حاصل ضرب تعداد زیادی عدد کوچک نزدیک صفر است

گرادیان ۱ و غیر نزدیک صفر در قسمت مثبت

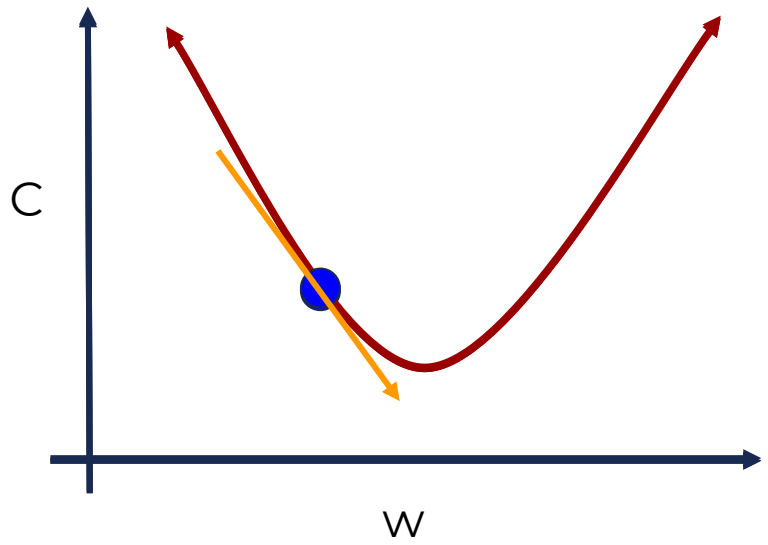






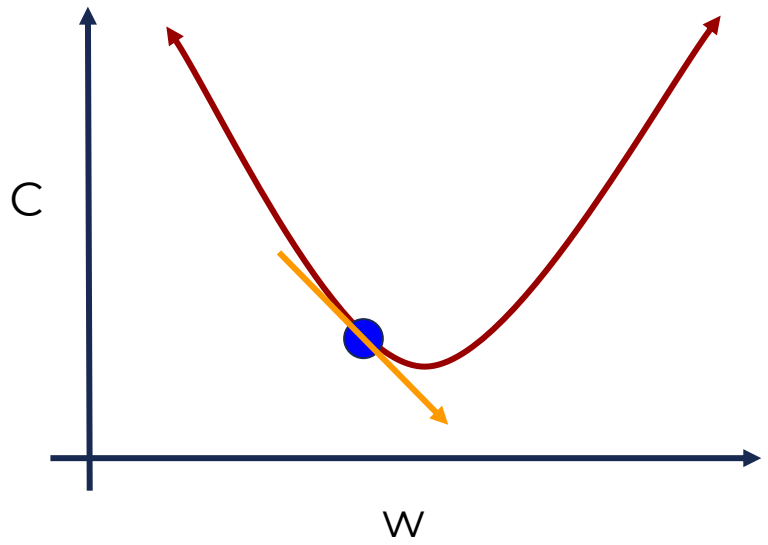
پس‌انتشار خطا و گرادیان کاهشی

- گرادیان کاهشی(در یک بعد)
- Gradient Descent (in 1 dimension)



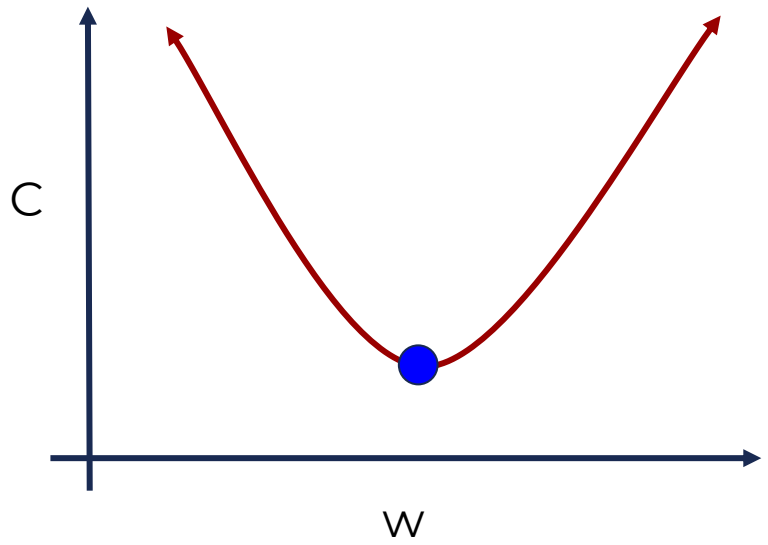
پس‌انتشار خطا و گرادیان کاهشی

- گرادیان کاهشی(در یک بعد)
- Gradient Descent (in 1 dimension)



پس‌انتشار خطا و گرادیان کاهشی

- به صورت بصری می توانیم مقدار انتخابی برای به حداقل رساندن هزینه را ببینیم!



پس‌انتشار خطا و گرادیان کاهشی

- پیدا کردن این کمینه برای ۱ بعد ساده است، اما تقریبا تمام مسائل ما دارای پارامترهای زیادی است، به این معنی که ما باید از جبر خطی built-in در کتابخانه یادگیری عمیق استفاده کنیم.
- با استفاده از گرادیان کاهشی می‌توانیم بهترین پارامترها را برای به حداقل رساندن تابع هزینه بدست آوریم، به عنوان مثال، پیدا کردن بهترین مقادیر برای وزن ورودی‌های نورون.

پس‌انتشار خطا و گرادیان کاهشی

- حالا ما فقط یک مسئله برای حل کردن داریم، چگونه می‌توانیم پارامترهای مطلوب یا وزن‌ها را در کل شبکه تنظیم کنیم؟
- اینجاست که پس‌انتشار خطا یا backpropagation مطرح می‌شود.

پس‌انتشار خطا و گرادیان کاهشی

- پس‌انتشار خطا برای محاسبه سهم خطای هر نورون پس از پردازش یک دسته از داده ها استفاده می شود.
- این روش به شدت بر قاعده زنجیری برای پیدا کردن مشتق یا chain rule تکیه می کند تا از طریق شبکه عقب برود و این خطاها را محاسبه کند.

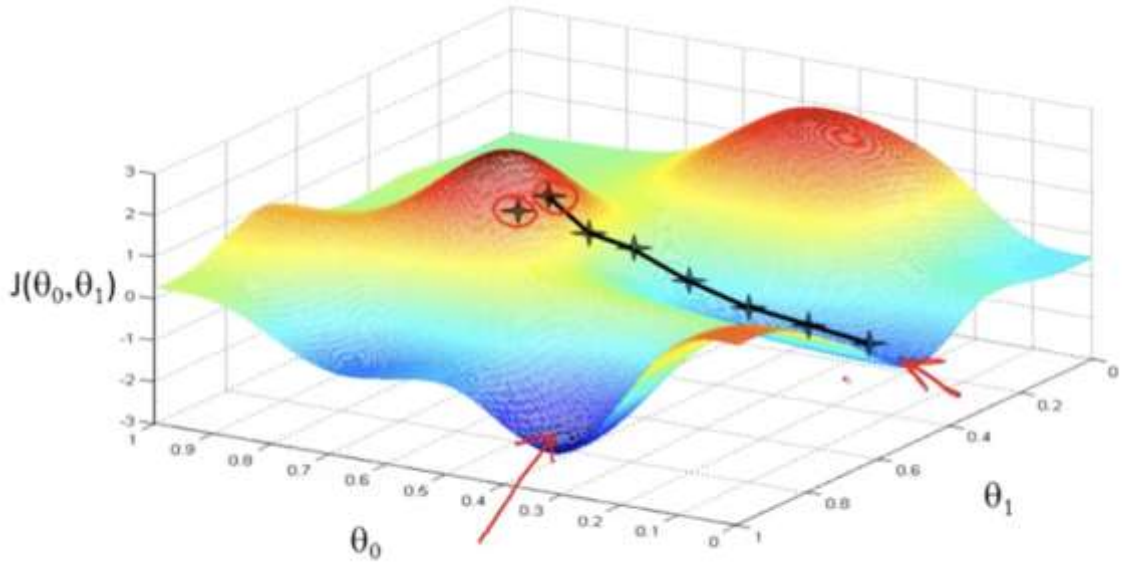
پس‌انتشار خطا و گرادیان کاهشی

- پس‌انتشار خطا با محاسبه خطای خروجی و با بازگشت لایه به لایه از خروجی به سمت ورودی انجام می دهد.
- برای هر مقدار ورودی نیاز به یک خروجی مورد انتظار دارد (یادگیری نظارت شده).

playground.tensorflow.org

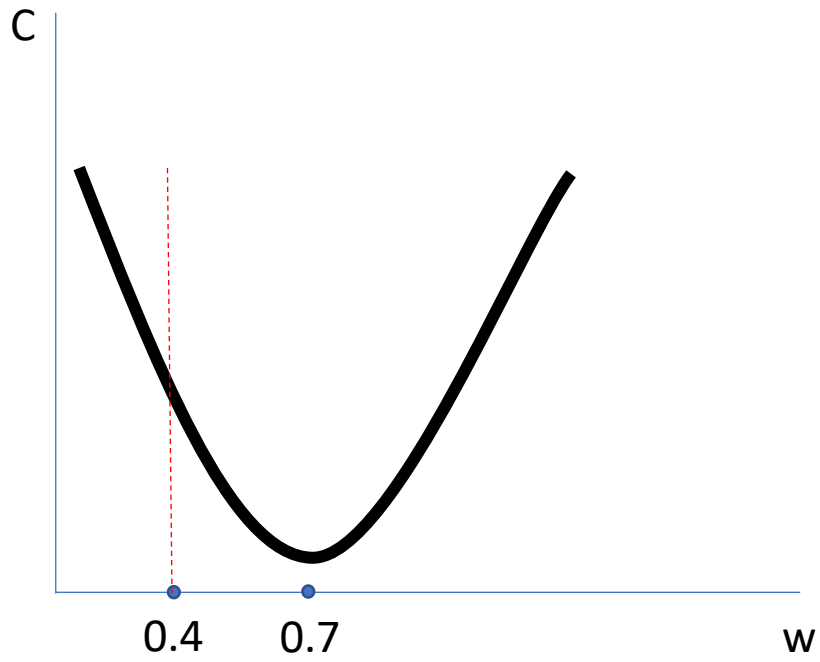
Learning rate

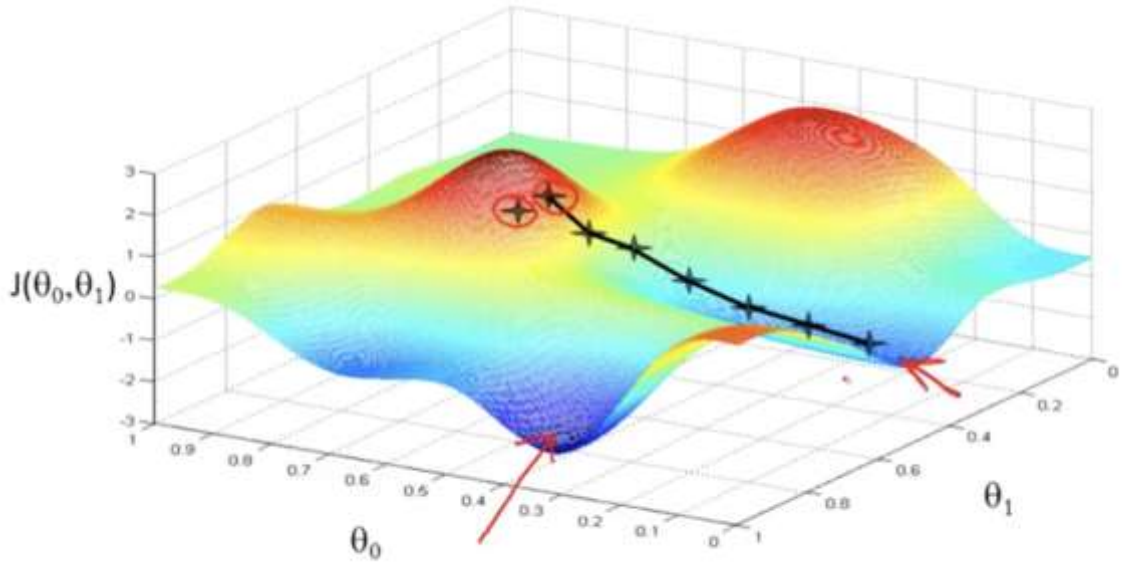
$$w = w - \nabla_w$$



Learning rate

$$w = w - \nabla_w$$

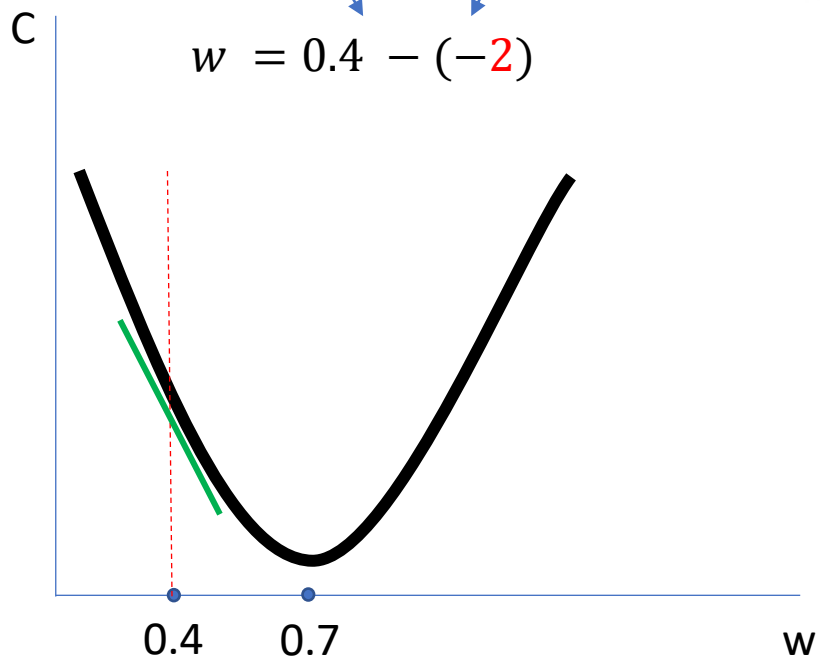


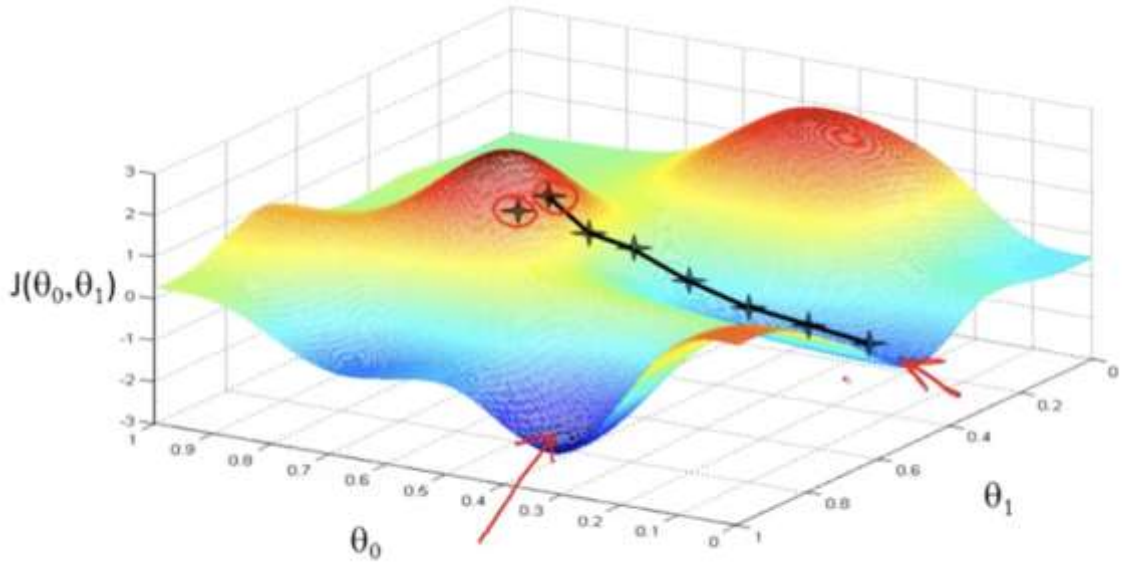


Learning rate

$$w = w - \nabla_w$$

$$w = 0.4 - (-2)$$

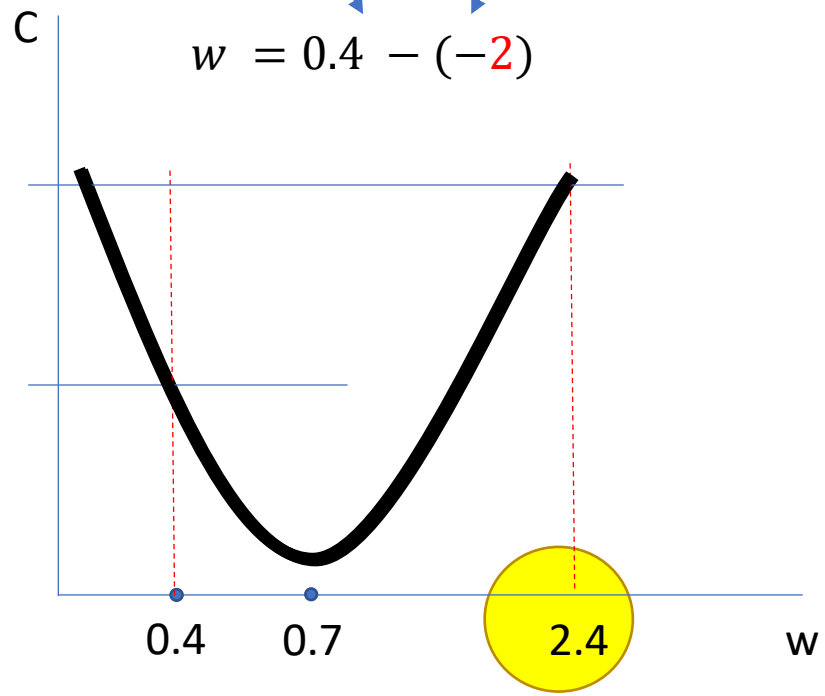


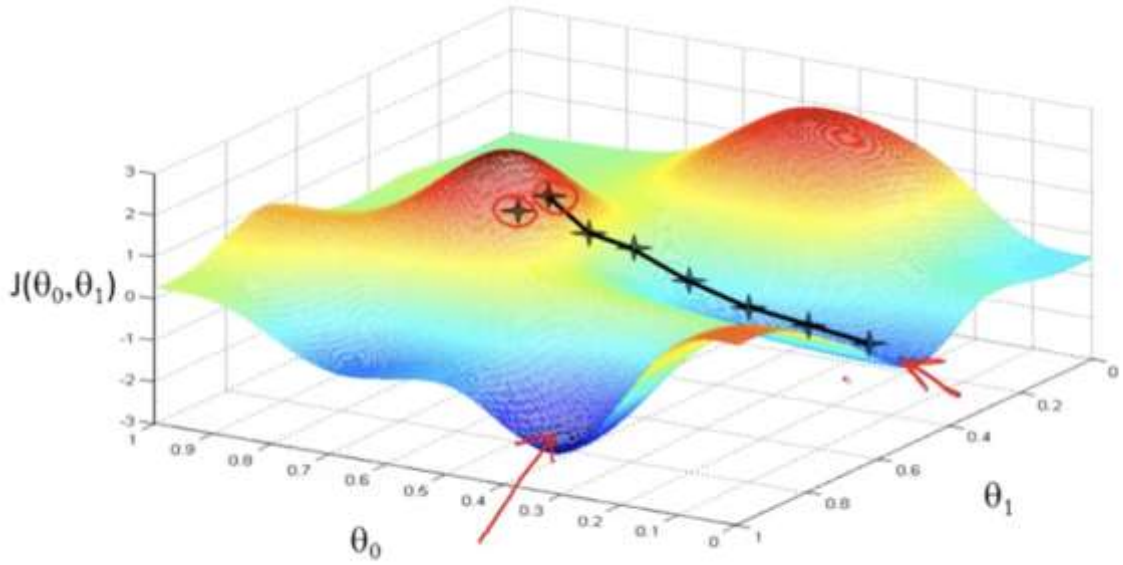


Learning rate

$$w = w - \nabla_w$$

$$w = 0.4 - (-2)$$

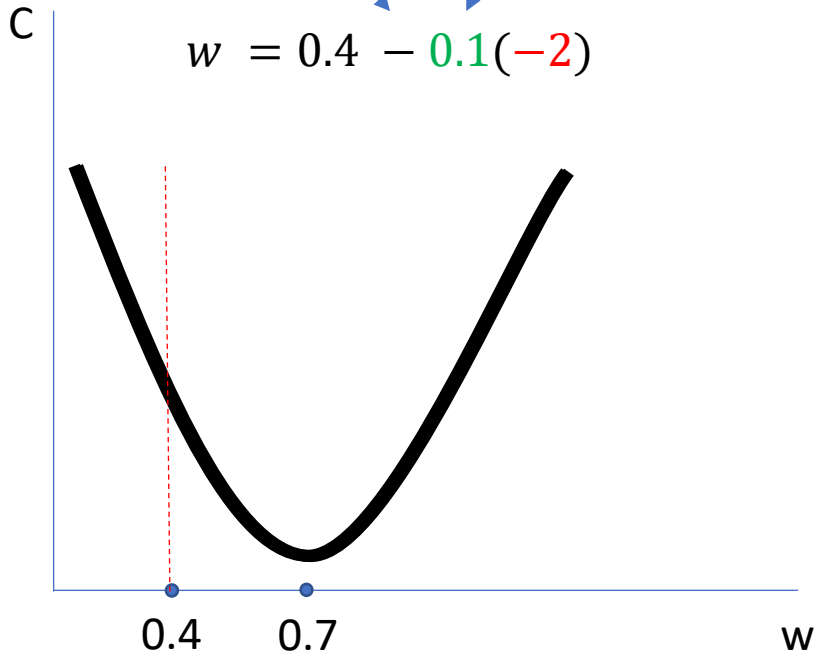


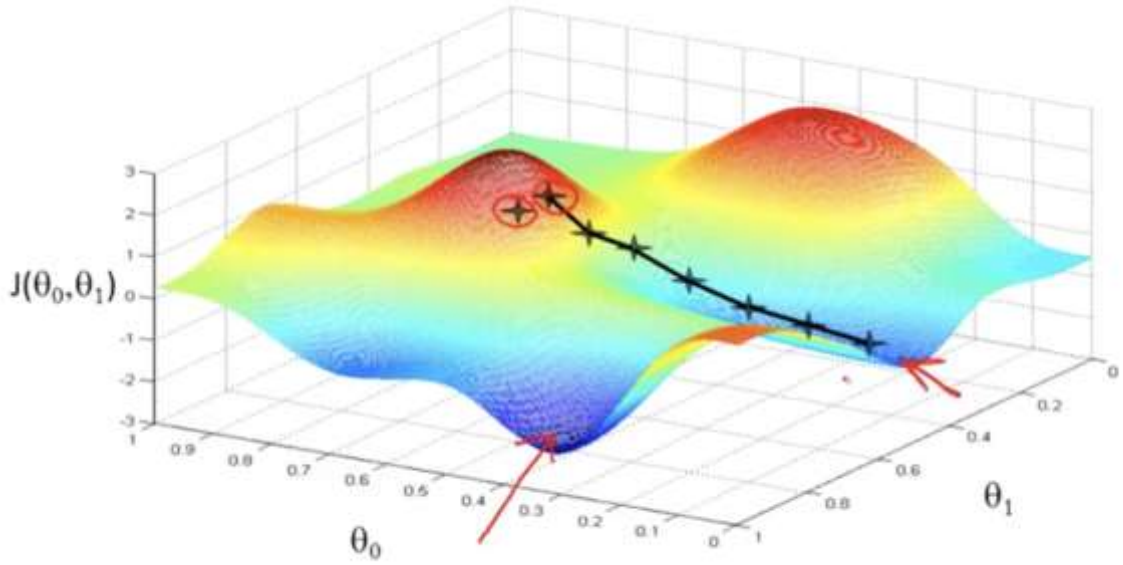


Learning rate

$$w = w - \mu \nabla_w$$

$$w = 0.4 - 0.1(-2)$$

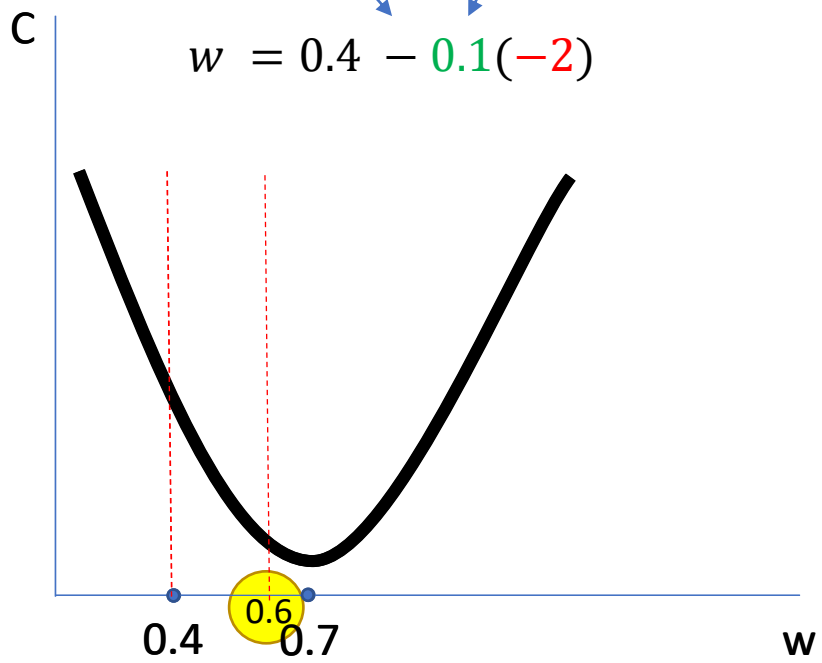




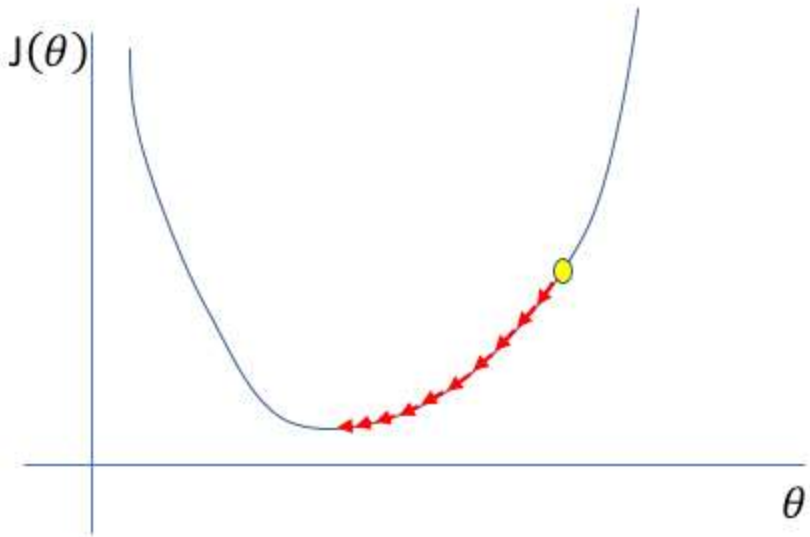
Learning rate

$$w = w - \mu \nabla_w$$

$$w = 0.4 - 0.1(-2)$$

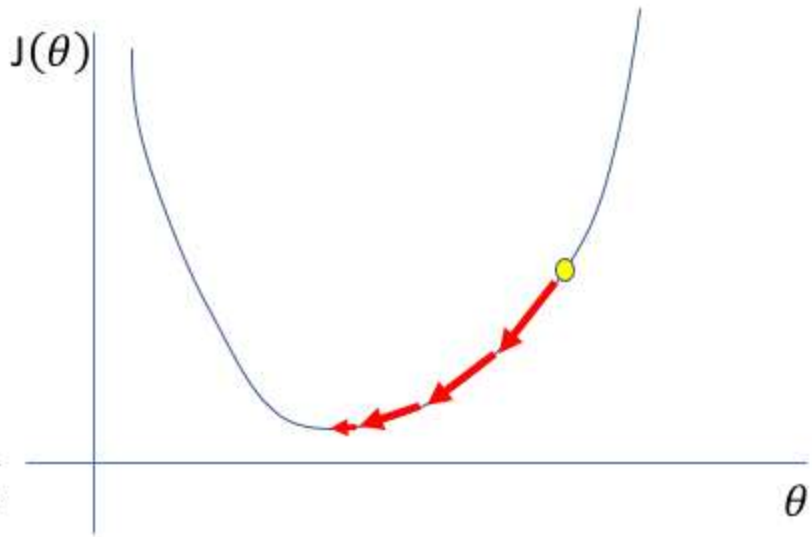


Too low



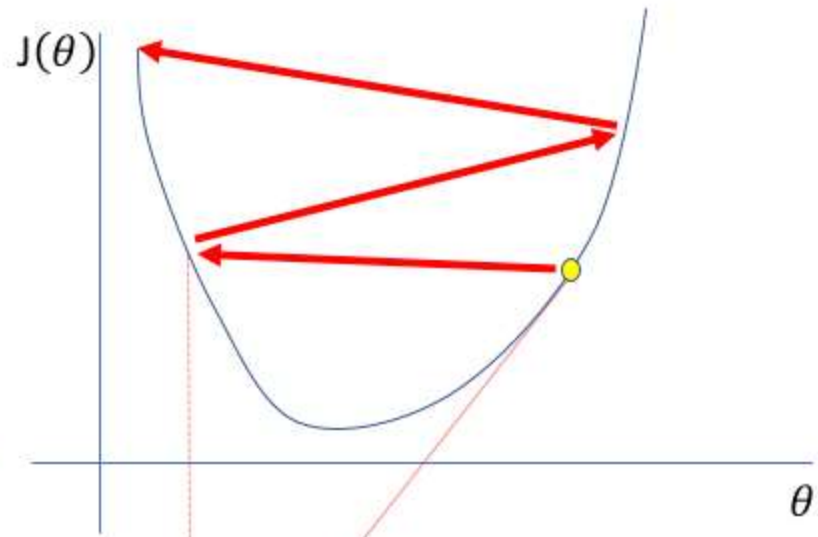
A small learning rate requires many updates before reaching the minimum point

Just right

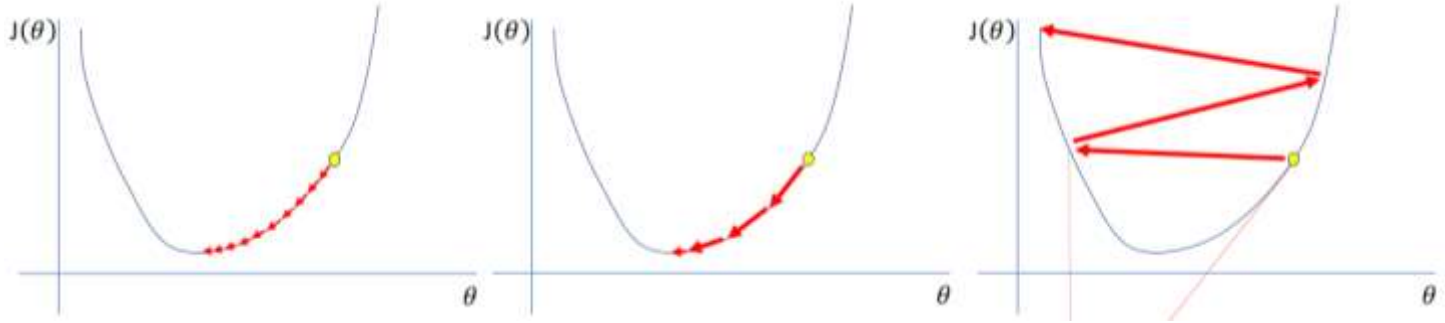


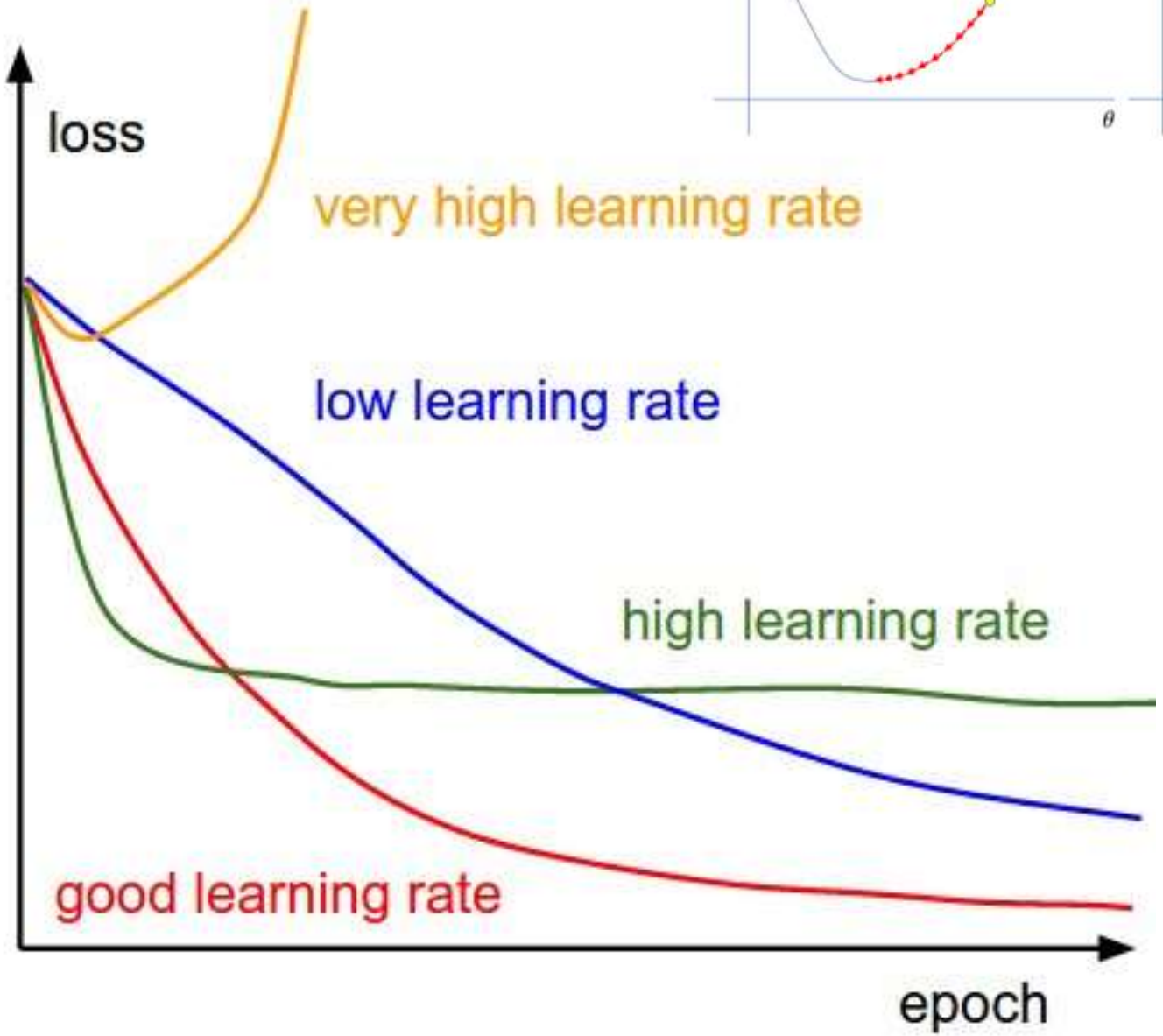
The optimal learning rate swiftly reaches the minimum point

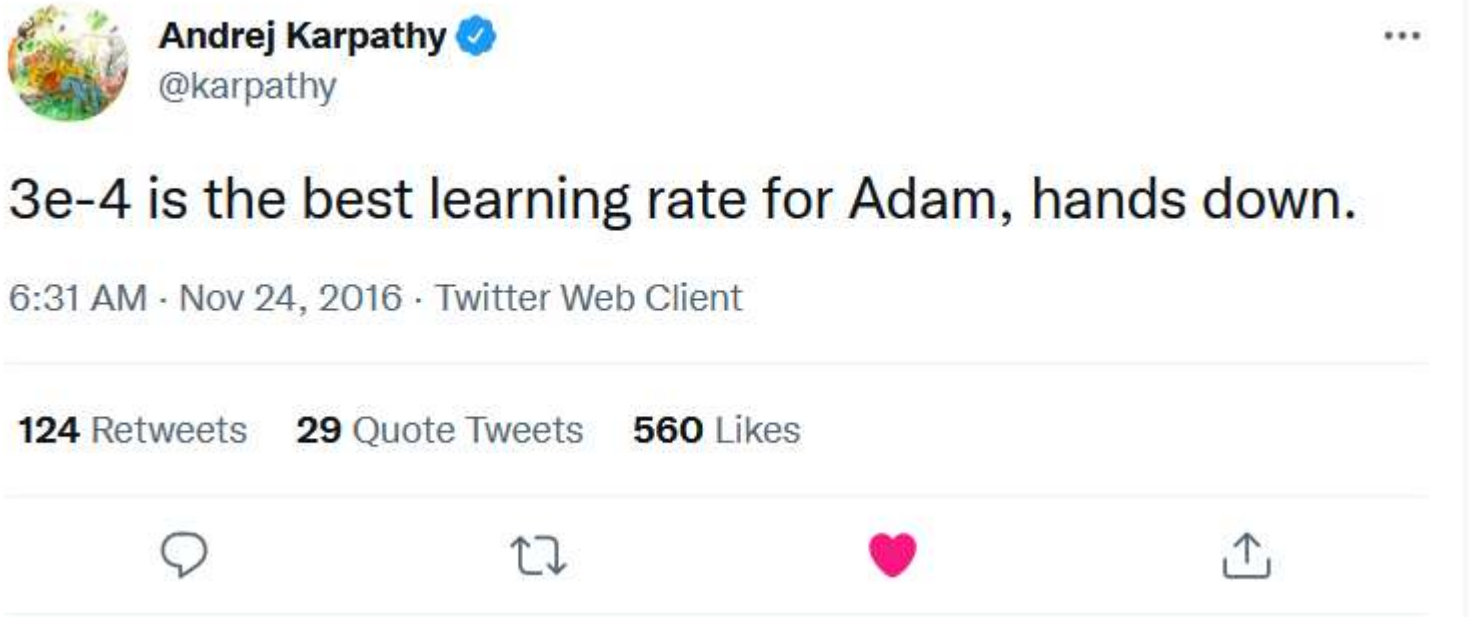
Too high

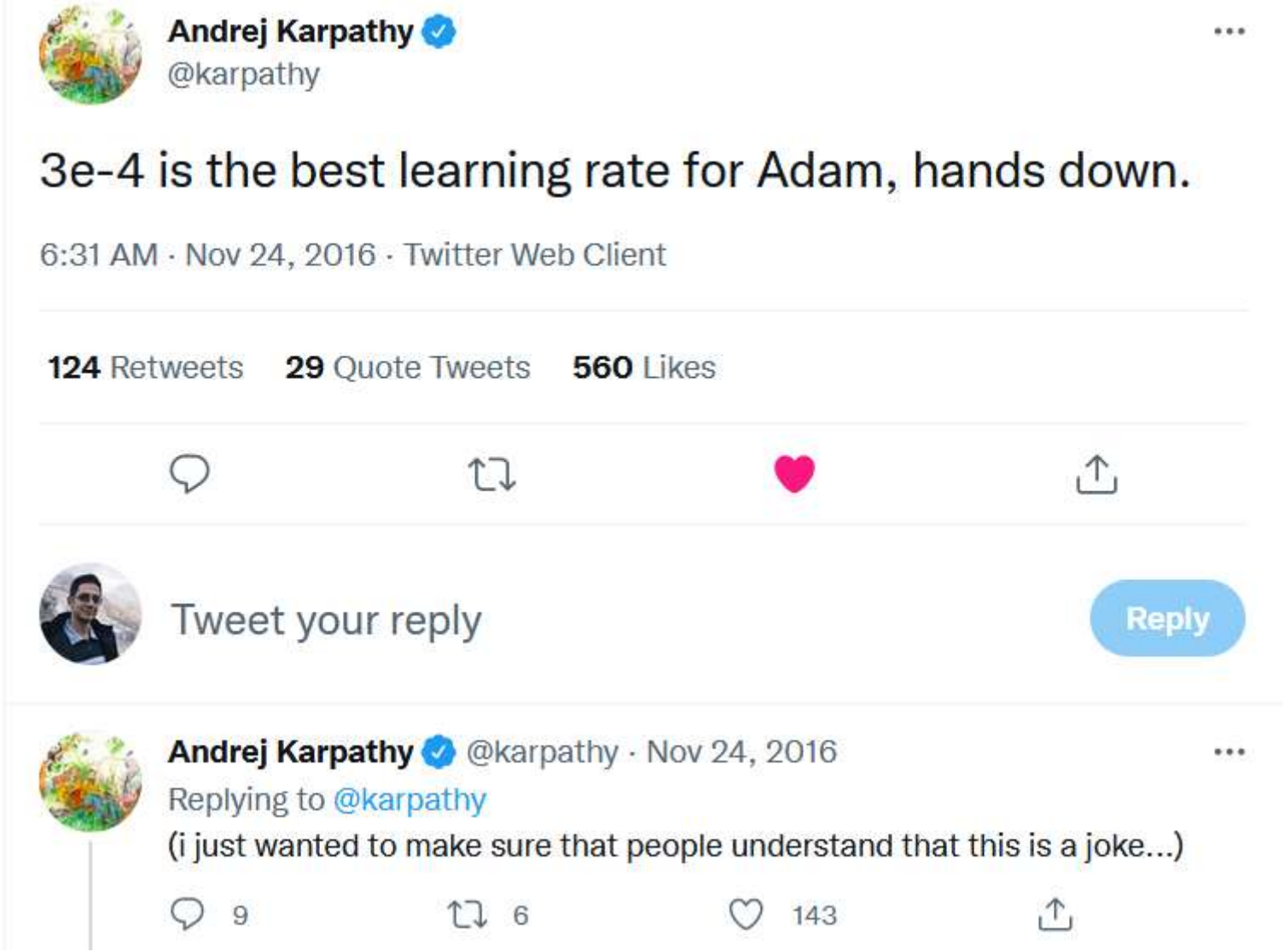


Too large of a learning rate causes drastic updates which lead to divergent behaviors


J(θ)
θ
J(θ)
θ
J(θ)
θ


loss
very high learning rate
low learning rate
high learning rate
good learning rate
epoch


Andrej Karpathy
@karpathy
3e-4 is the best learning rate for Adam, hands down.
6:31 AM · Nov 24, 2016 · Twitter Web Client
124 Retweets
29 Quote Tweets
560 Likes


Andrej Karpathy
@karpathy
3e-4 is the best learning rate for Adam, hands down.
6:31 AM · Nov 24, 2016 · Twitter Web Client
124 Retweets
29 Quote Tweets
560 Likes
Tweet your reply
Reply
Andrej Karpathy @karpathy · Nov 24, 2016
Replying to @karpathy
(i just wanted to make sure that people understand that this is a joke...)
9
6
143

More about optimizers

- https://keras.io/api/optimizers/
- https://www.aparat.com/v/p98WT?playlist=325585

Callbacks

- https://www.aparat.com/v/Tdbck?playlist=342711

بخش پنجم:

پیاده سازی در کراس

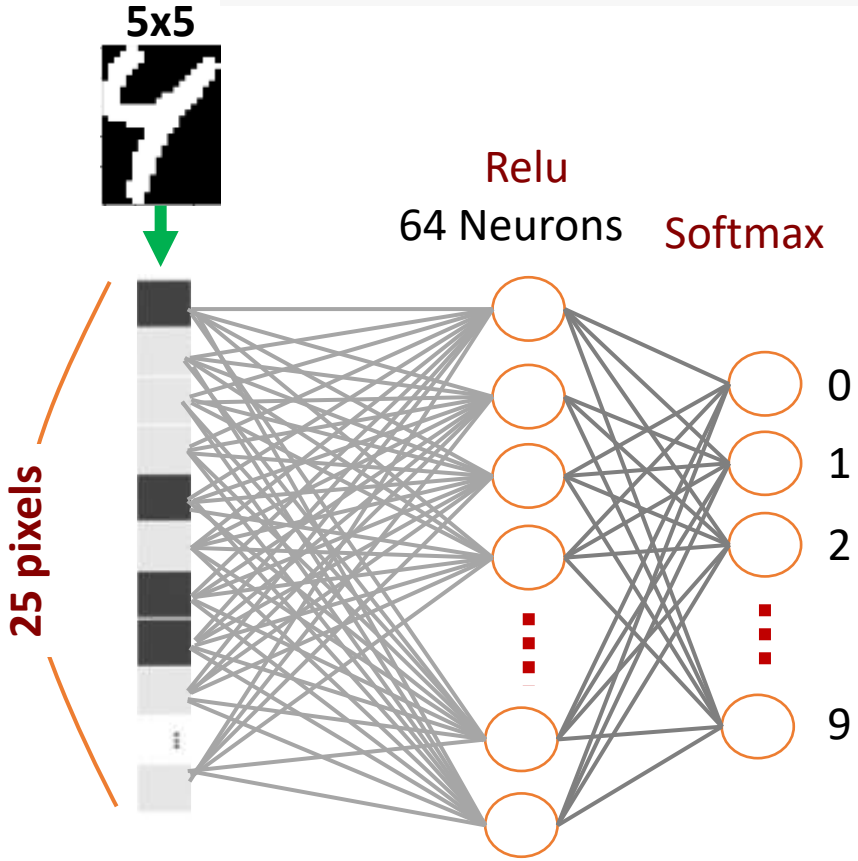
Implementation with Keras!

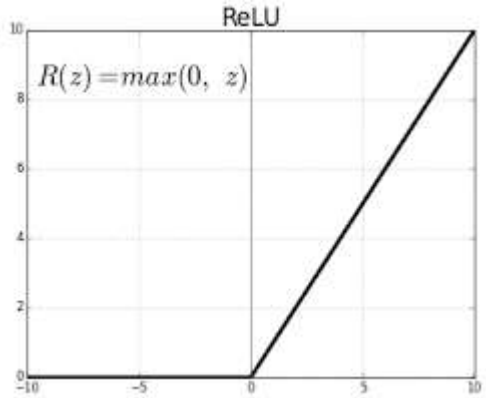




پیاده سازی در کراس- معماری مدل

```
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=25))
model.add(Dense(10, activation='softmax'))
```



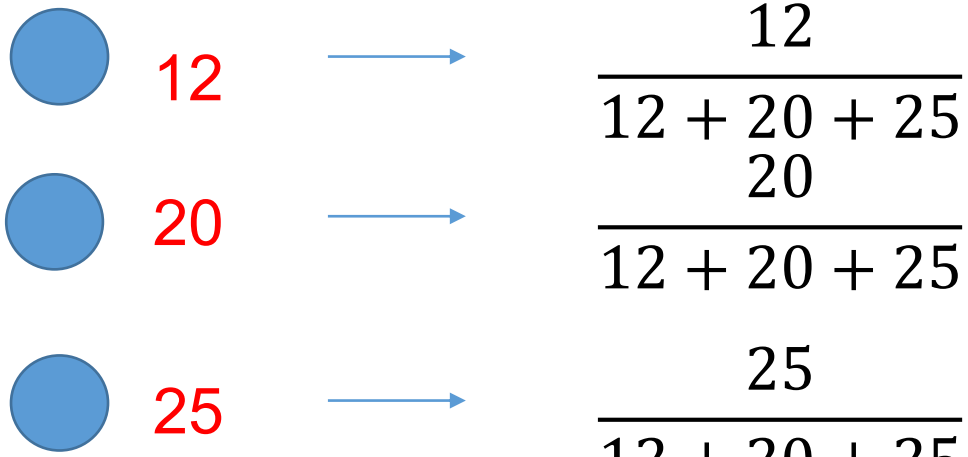


$$softmax(L_n) = \frac{e^{L_n}}{||e^{L}||}$$

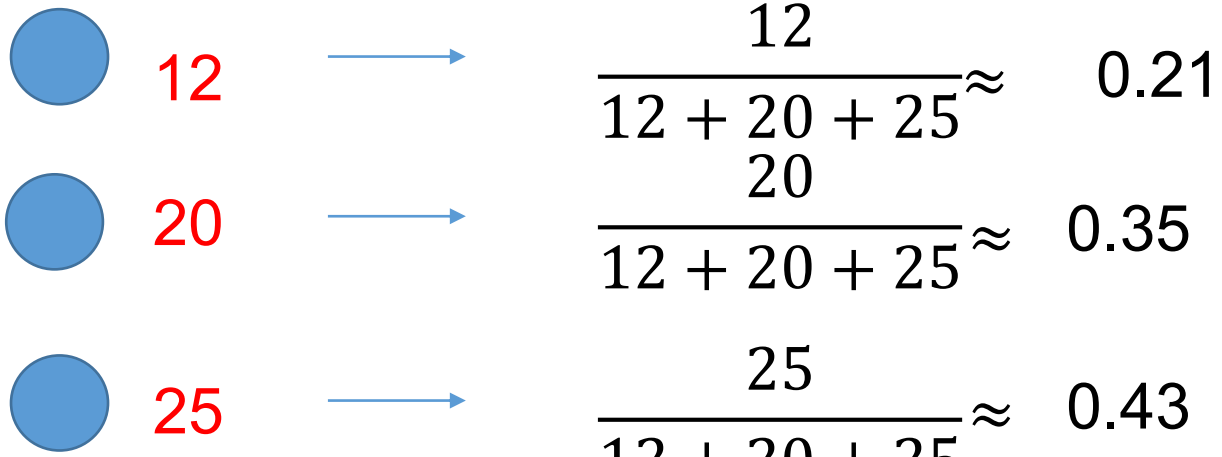
تابع Softmax

- 12
- 20
- 25

تابع Softmax



تابع Softmax





12

$$\frac{12}{12+20+25} \approx 0.21$$

One-hot label

0



20

$$\frac{20}{12+20+25} \approx 0.35$$

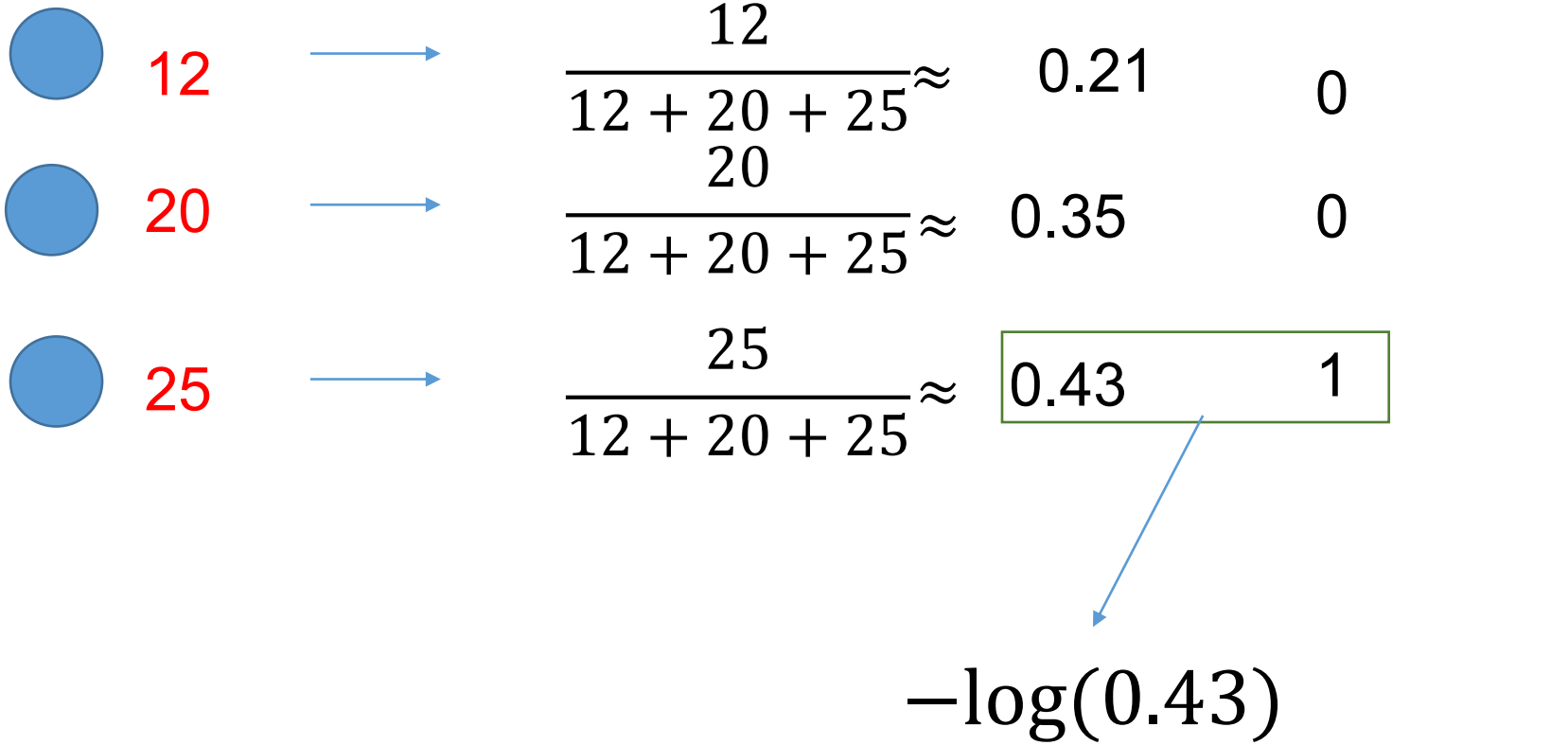
0



25

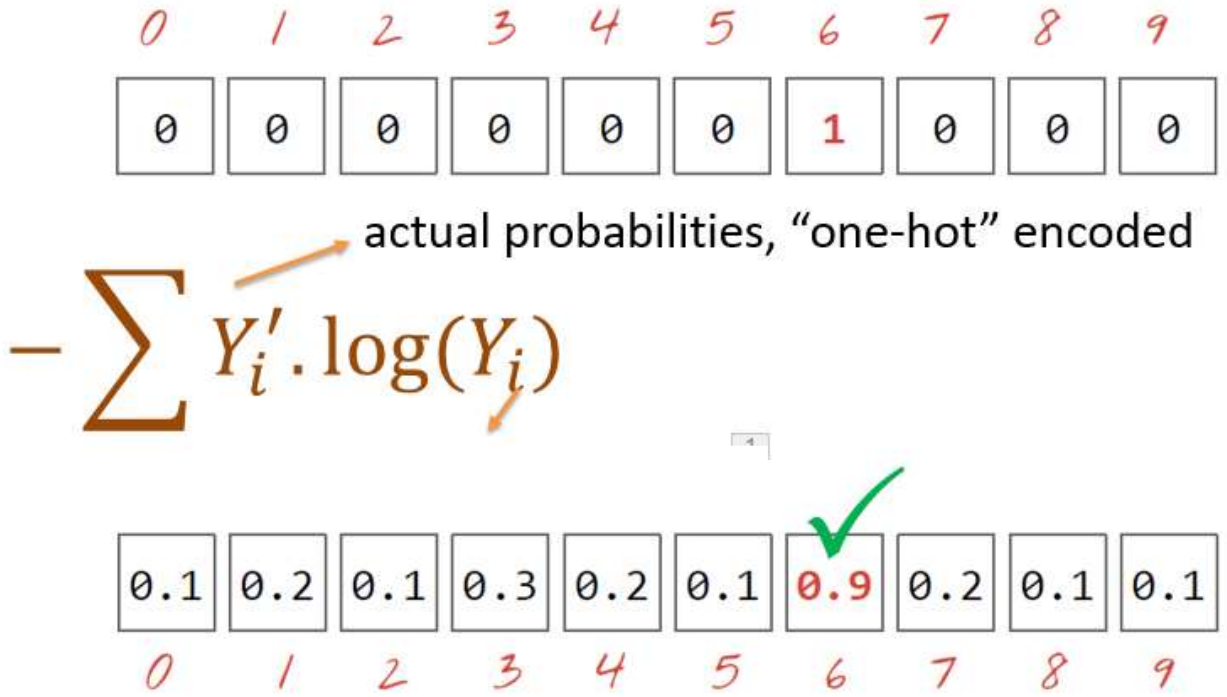
$$\frac{25}{12+20+25} \approx 0.43$$

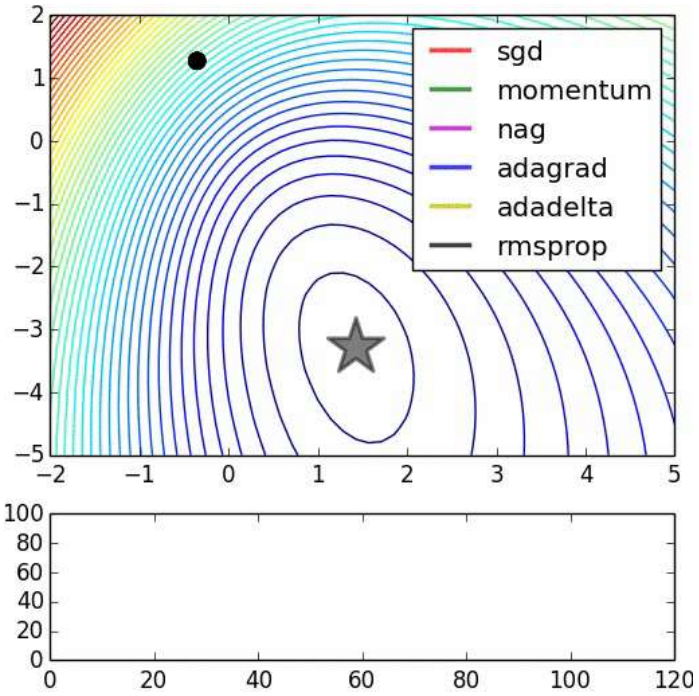
1


One-hot label
12
20
25
$\frac{12}{12+20+25} \approx$ 0.21 0
$\frac{20}{12+20+25} \approx$ 0.35 0
$\frac{25}{12+20+25} \approx$ 0.43 1
$-\log(0.43)$

پیاده سازی در کراس – تابع هزینه و بهینه سازی

```
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```





DeepLearning.AI

Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization

11 videos • Total 92 minutes

Mini-batch Gradient Descent • 11 minutes • **Preview module**

Understanding Mini-batch Gradient Descent • 11 minutes

Exponentially Weighted Averages • 5 minutes

Understanding Exponentially Weighted Averages • 9 minutes

Bias Correction in Exponentially Weighted Averages • 4 minutes

Gradient Descent with Momentum • 9 minutes

RMSprop • 7 minutes

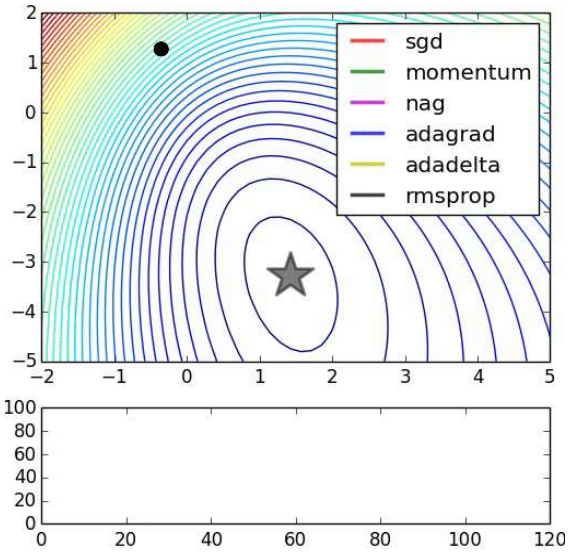
Adam Optimization Algorithm • 7 minutes

Learning Rate Decay • 6 minutes

The Problem of Local Optima • 5 minutes

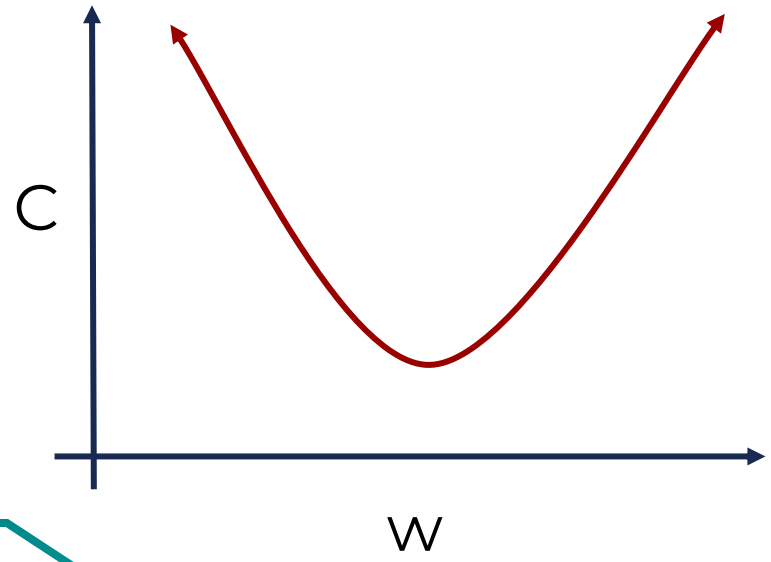
Link

انواع بهینه سازها



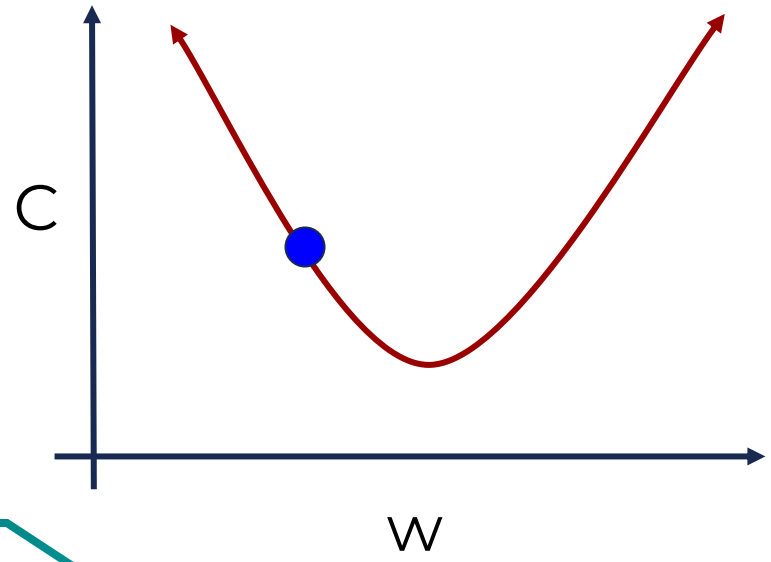
پیاده سازی در کراس:
تعیین داده آموزشی، اندازه mini-batch و تعداد epoch

```
model.fit(x_train, y_train,
          epochs=30,
          batch_size=64)
```



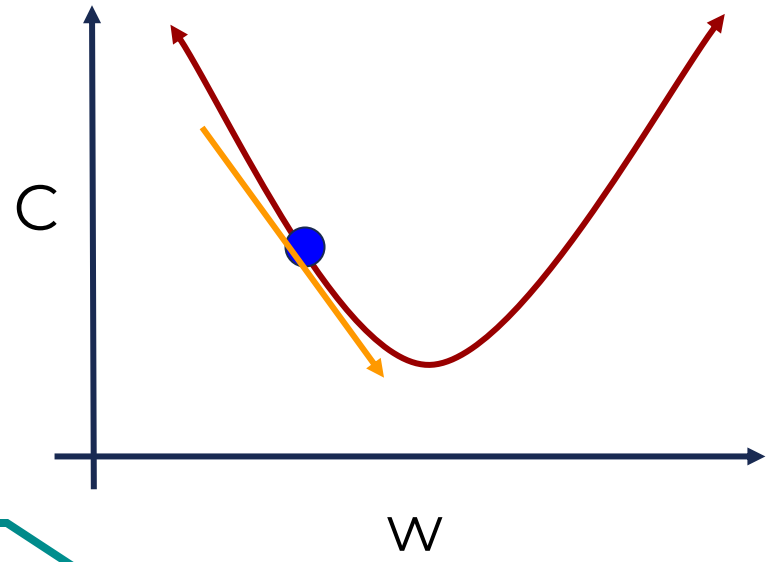
پیاده سازی در کراس:
تعیین داده آموزشی، اندازه mini-batch و تعداد epoch

```
model.fit(x_train, y_train,
          epochs=30,
          batch_size=64)
```



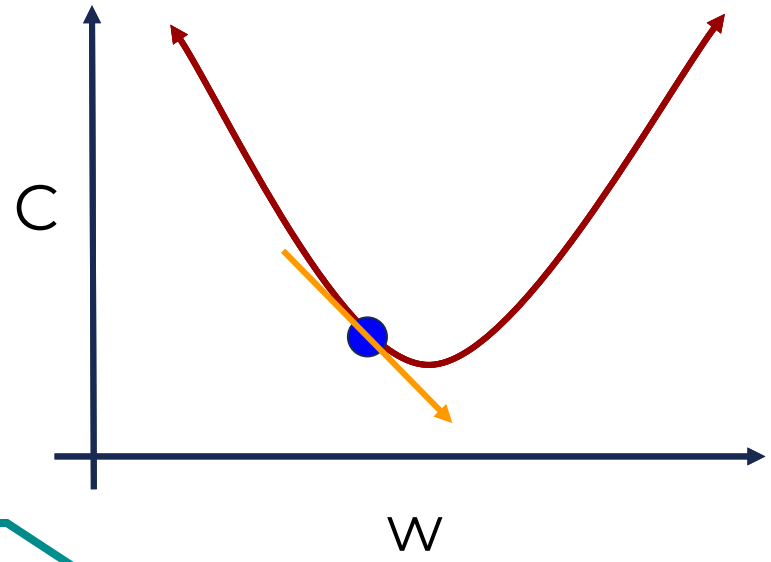
پیاده سازی در کراس:
تعیین داده آموزشی، اندازه mini-batch و تعداد epoch

```
model.fit(x_train, y_train,
          epochs=30,
          batch_size=64)
```



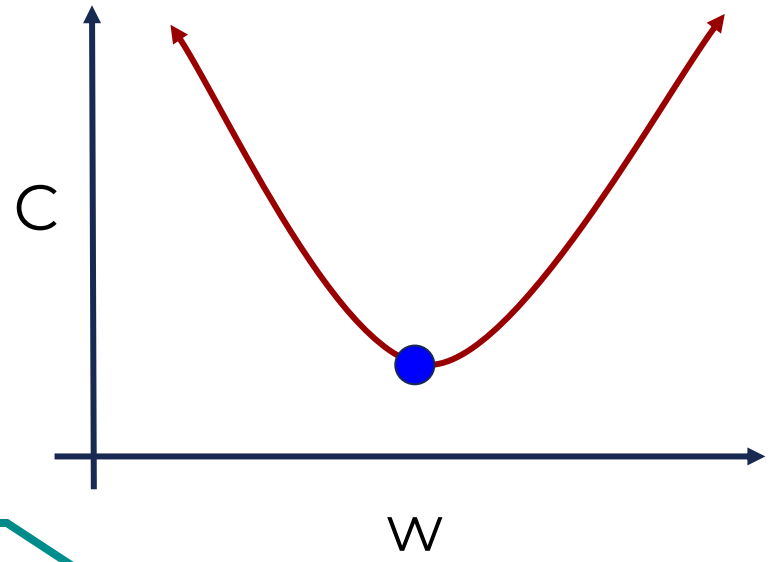
پیاده سازی در کراس:
تعیین داده آموزشی، اندازه mini-batch و تعداد epoch

```
model.fit(x_train, y_train,
          epochs=30,
          batch_size=64)
```



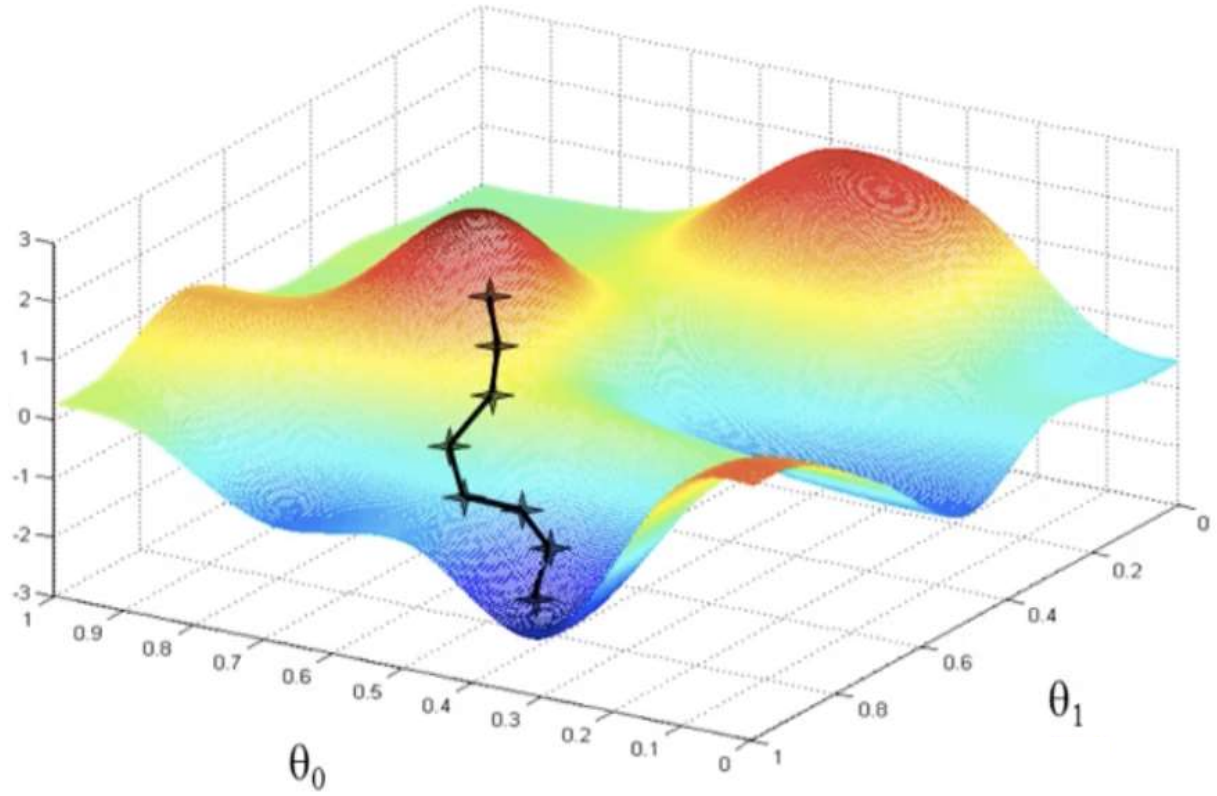
پیاده سازی در کراس:
تعیین داده آموزشی، اندازه mini-batch و تعداد epoch

```
model.fit(x_train, y_train,
          epochs=30,
          batch_size=64)
```



Gradient decent

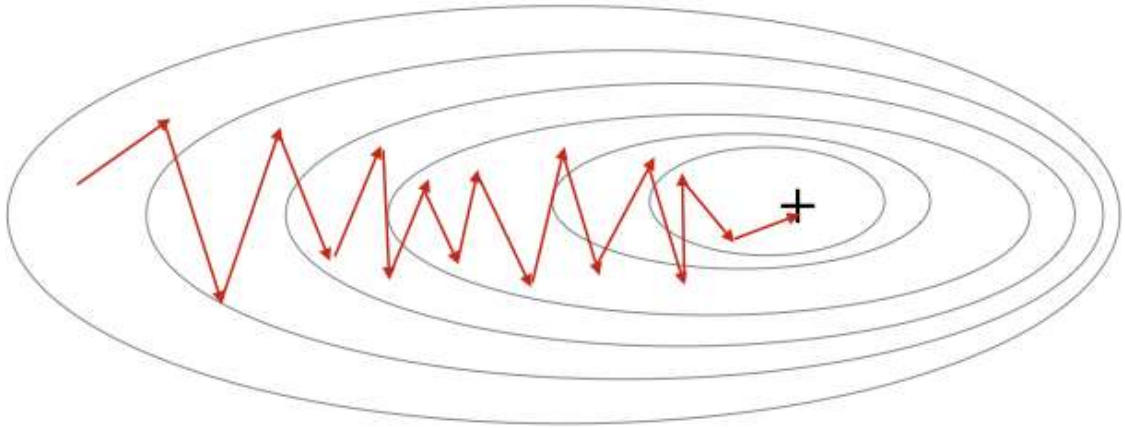
$J(\theta_0,\theta_1)$



$$W = W - \alpha\, dW$$

$$b = b - \alpha\, db$$

Stochastic Gradient Descent



Repeat Until Convergence {

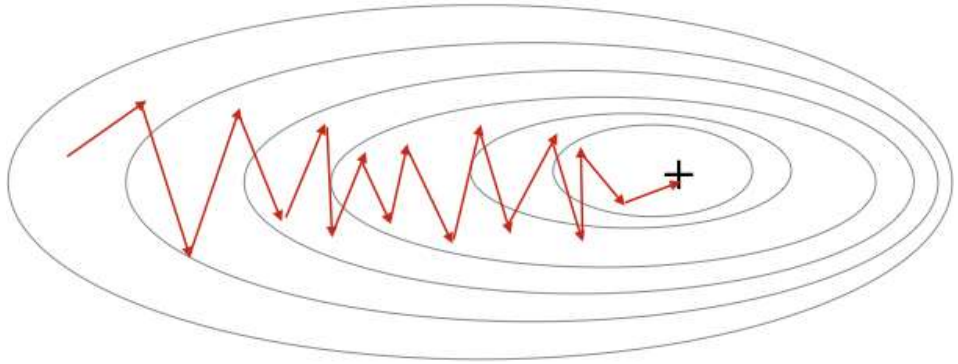
for i = 1...m {

$$\omega \leftarrow \omega - \alpha * \nabla_w L_m(w)$$

}

}

Stochastic Gradient Descent



Repeat Until Convergence {

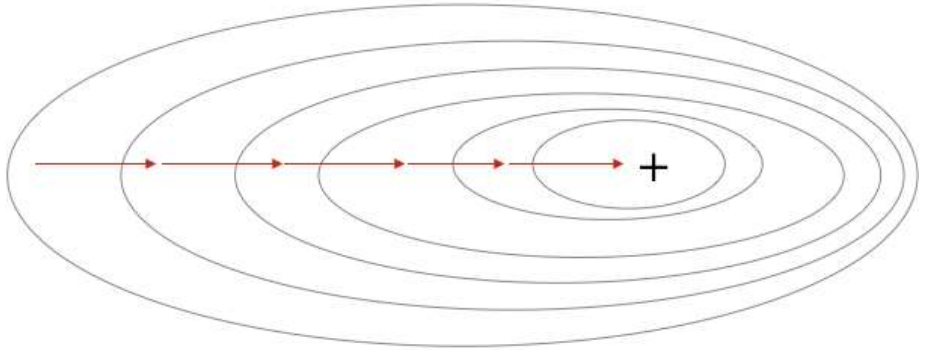
for i = 1...m {

$$\omega \leftarrow \omega - \alpha * \nabla_w L_m(w)$$

}

}

(Batch) Gradient Descent



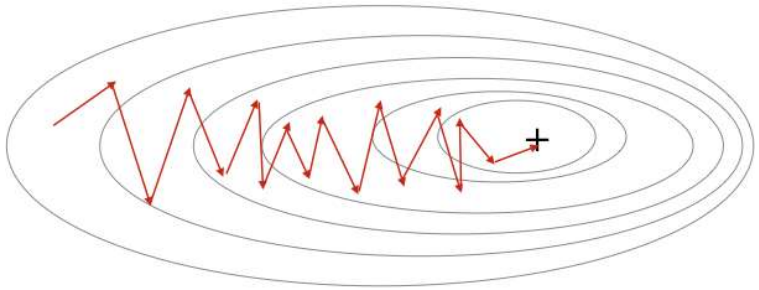
Repeat Until Convergence {

$$\omega \leftarrow \omega - \alpha * \nabla_w \sum_{1}^{m} L_m(w) \text{ /m}$$

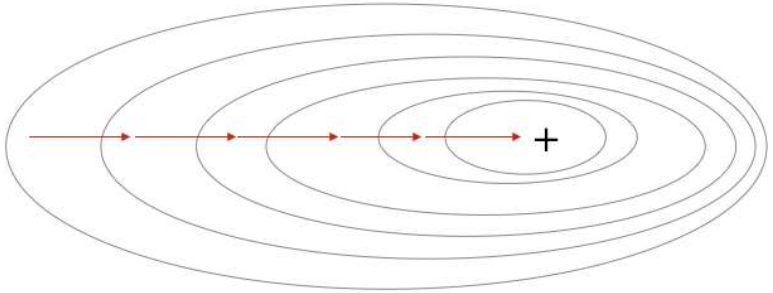
}

S.NO.	Batch Gradient Descent	Stochastic Gradient Descent
1.	Computes gradient using the whole Training sample	Computes gradient using a single Training sample
2.	Slow and computationally expensive algorithm	Faster and less computationally expensive than Batch GD
3.	Not suggested for huge training samples.	Can be used for large training samples.
4.	Deterministic in nature.	Stochastic in nature.
5.	Gives optimal solution given sufficient time to converge.	Gives good solution but not optimal.
6.	No random shuffling of points are required.	The data sample should be in a random order, and this is why we want to shuffle the training set for every epoch.
7.	Can’t escape shallow local minima easily.	SGD can escape shallow local minima more easily.
8.	Convergence is slow.	Reaches the convergence much faster.

Stochastic Gradient Descent

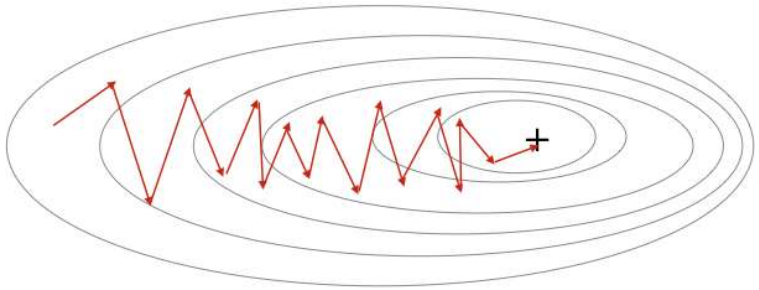


(Batch) Gradient Descent

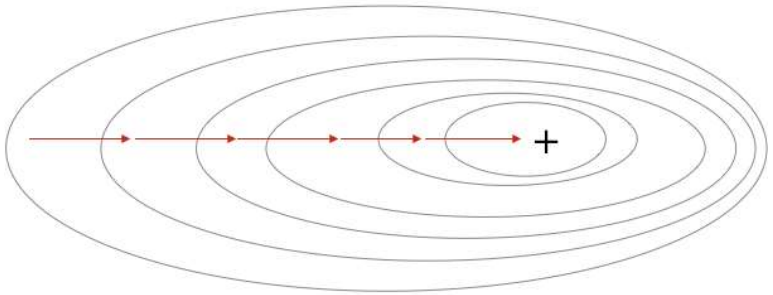


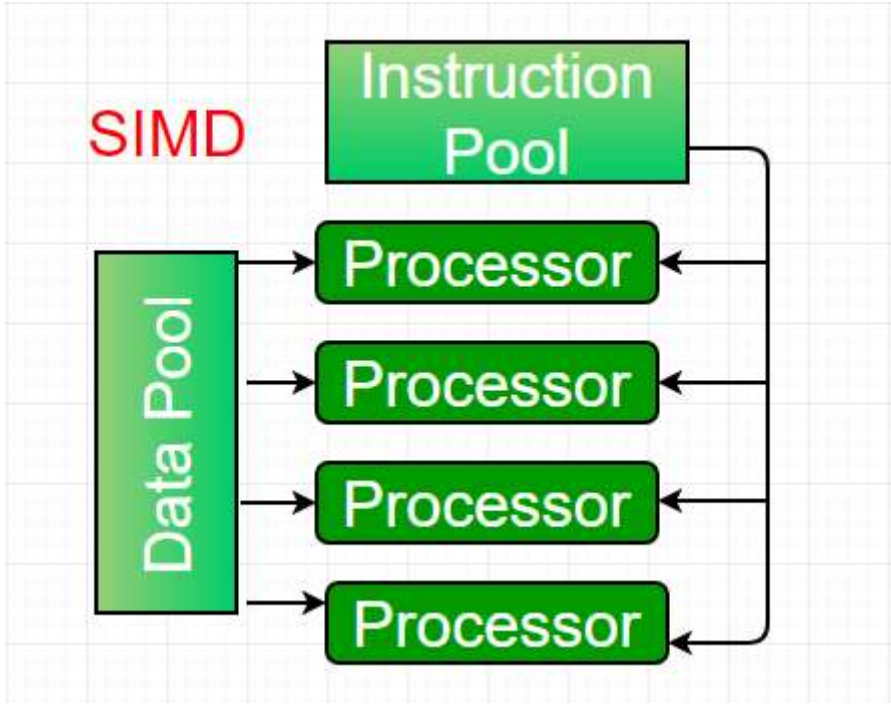


Stochastic Gradient Descent

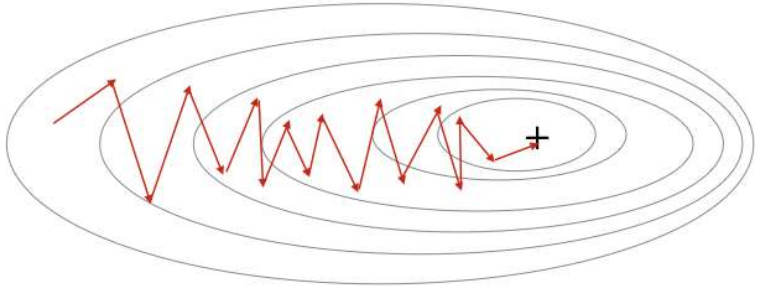


(Batch) Gradient Descent

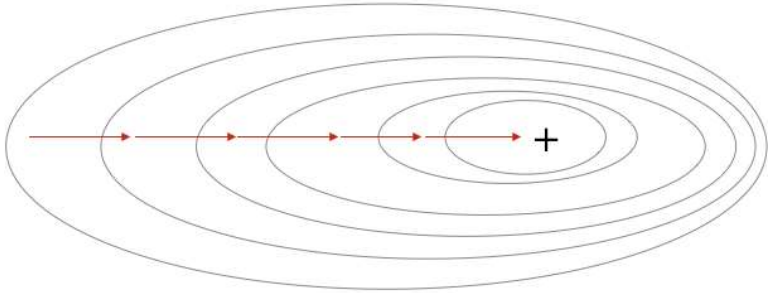




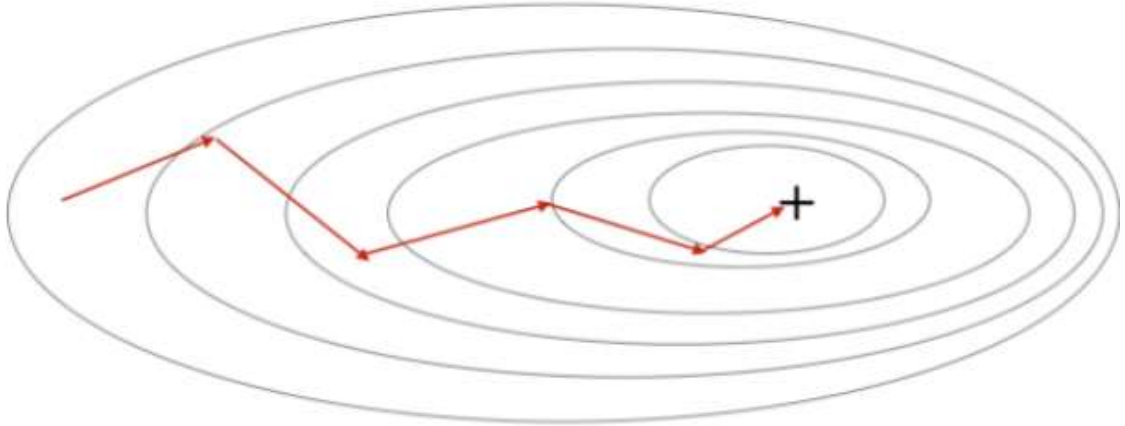
Stochastic Gradient Descent



(Batch) Gradient Descent



Mini-Batch Gradient Descent

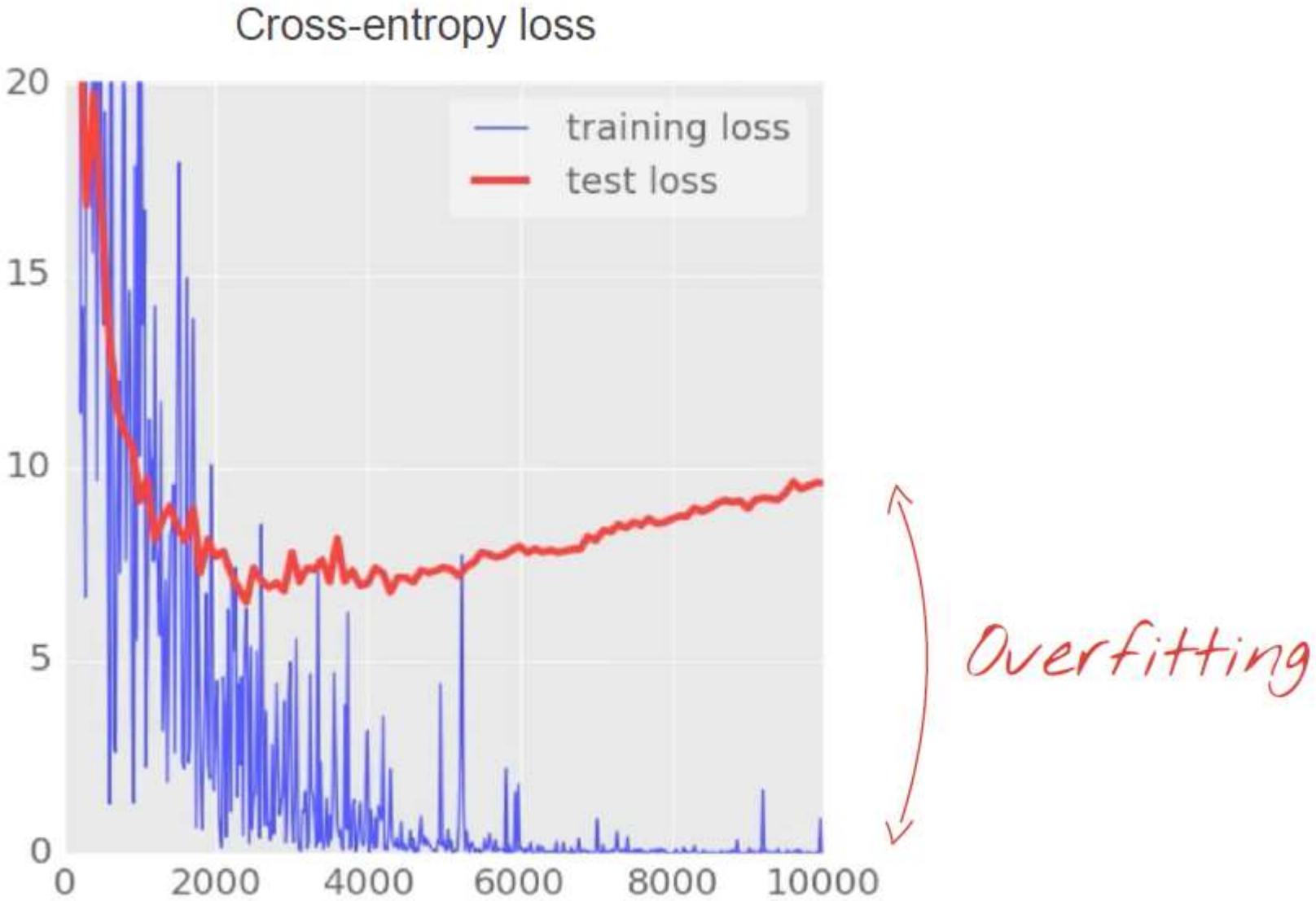


پیاده سازی شبکه عصبی در کراس – ارقام فارسی

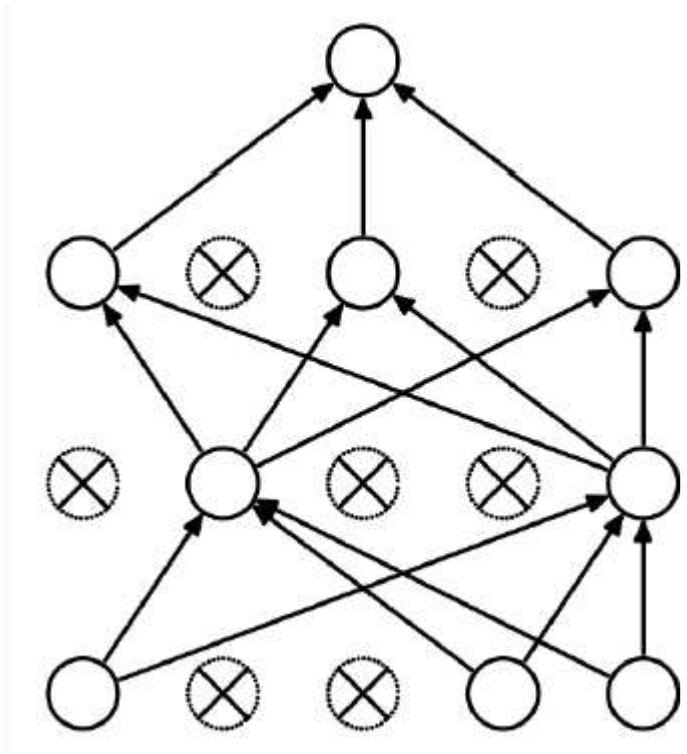


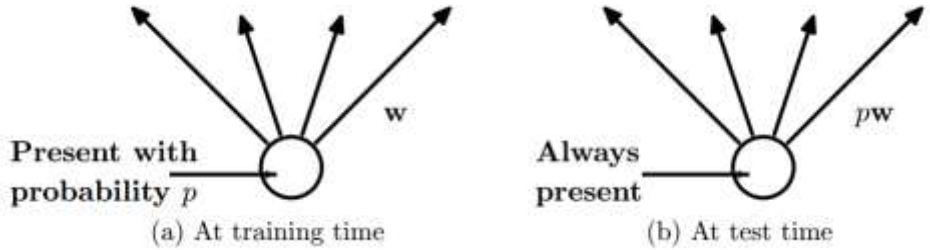
01-Simple-neural-network(mlp)-in-keras.ipynb

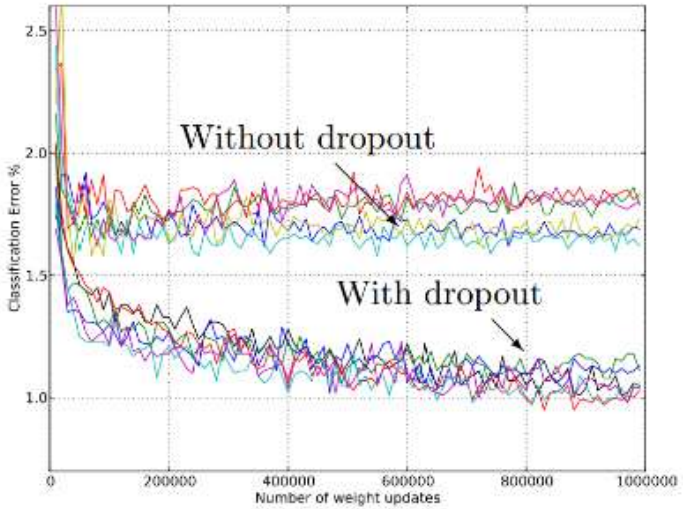
پیاده سازی در کراس- بیش برازش



پیاده سازی در کراس- Dropout







Main Idea: approximately combining exponentially many different neural network architectures efficiently

*Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, **JMLR** (2014)*

بیش برازش داده‌ها و لایه‌ی Dropout در کراس



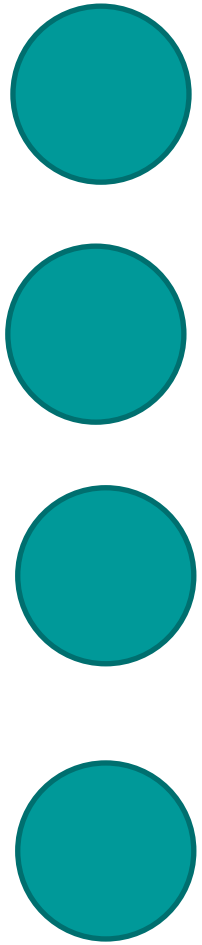
03_Dropout.ipynb

شبکه های عصبی کانولوشنالی

Convolutional Neural Networks


bedroom
bathroom
area
age

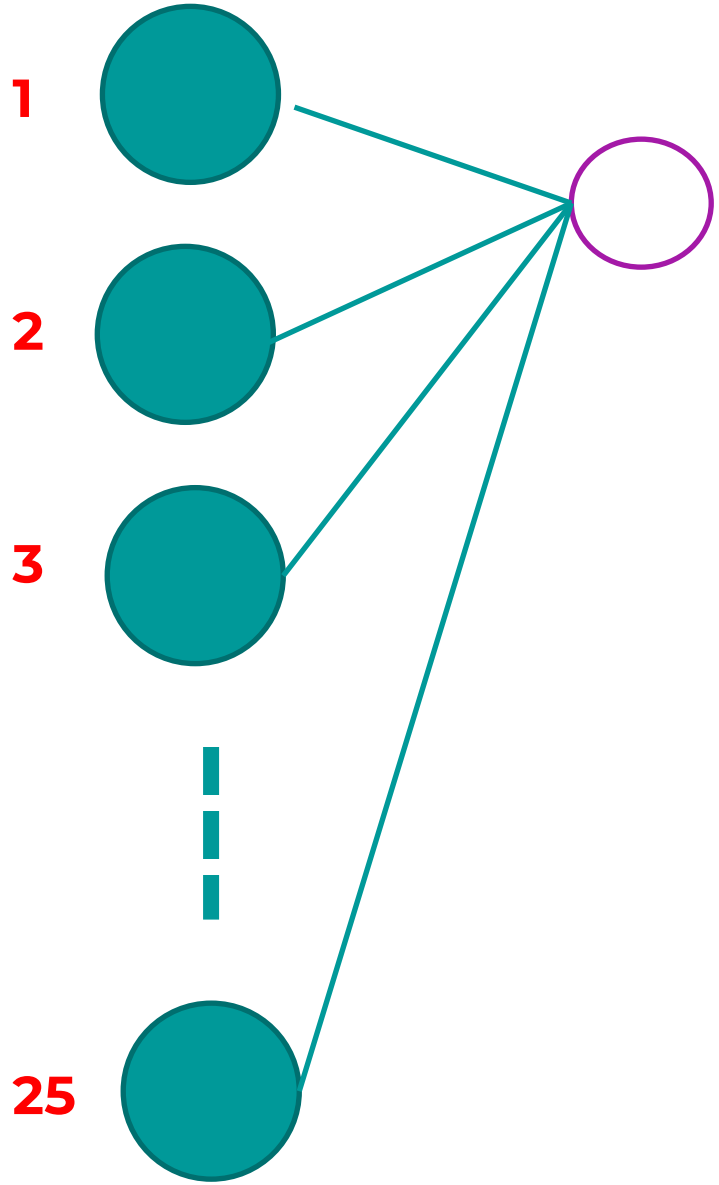

age
area
bathroom
bedroom



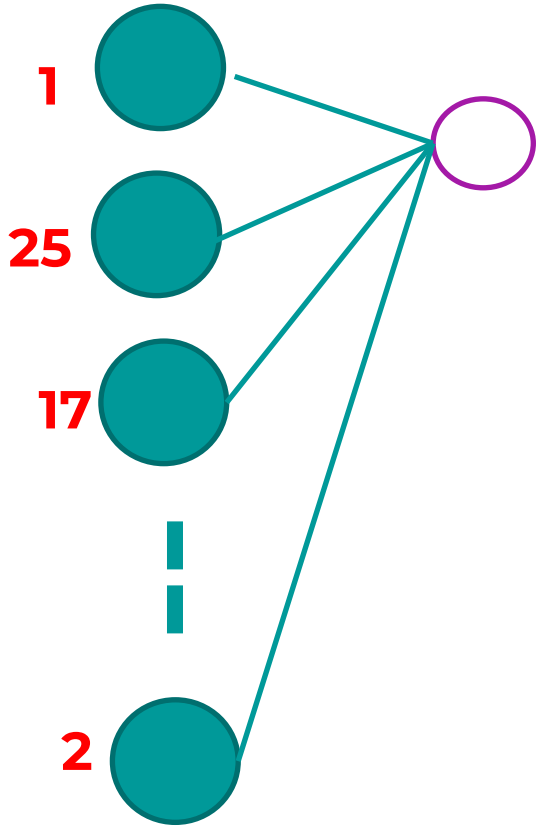
area w_1 + age w_2 + # bathroom w_3 + # bedroom w_4

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



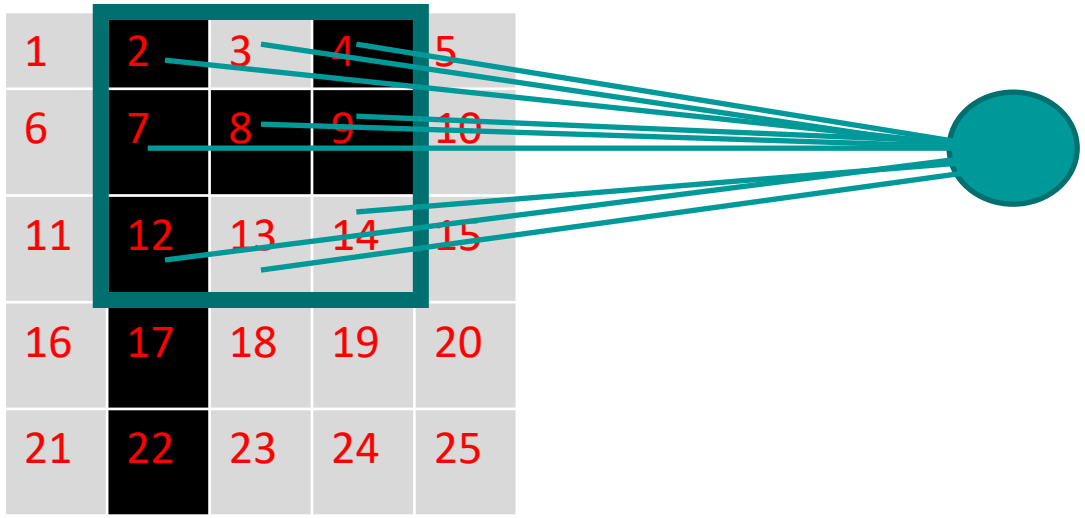
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



Pixel 1 w1 + pixel25 w2 + Pixel 3 w3 + ...

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

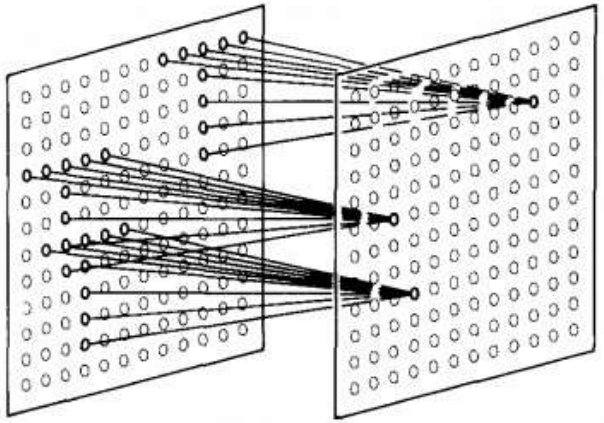
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



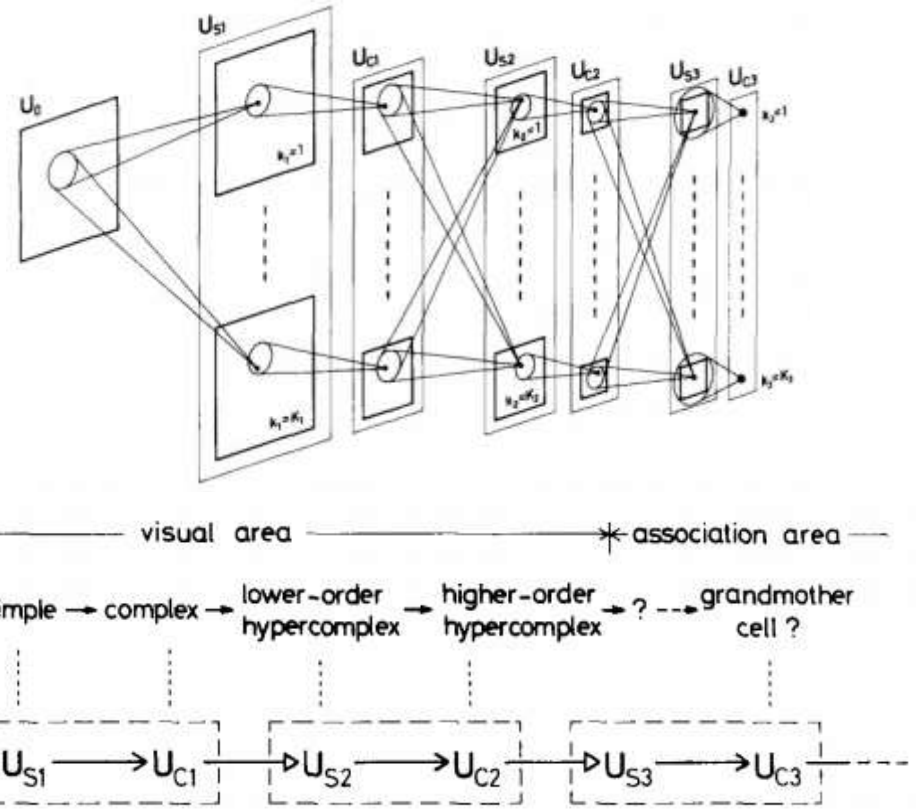
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

A bit of history:

Neurocognitron
[Fukushima 1980]



"sandwich" architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling

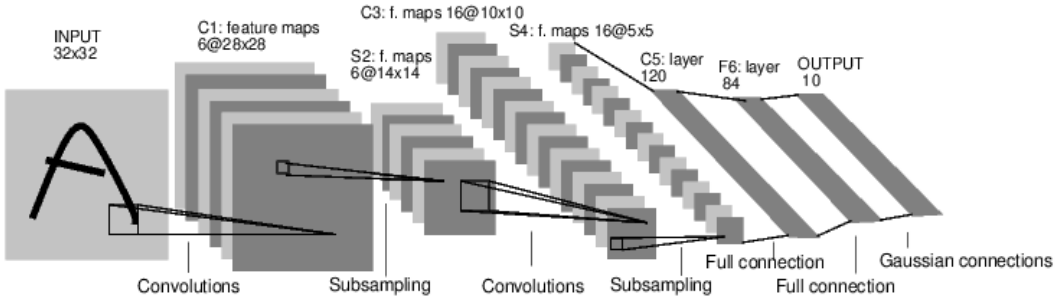


A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]





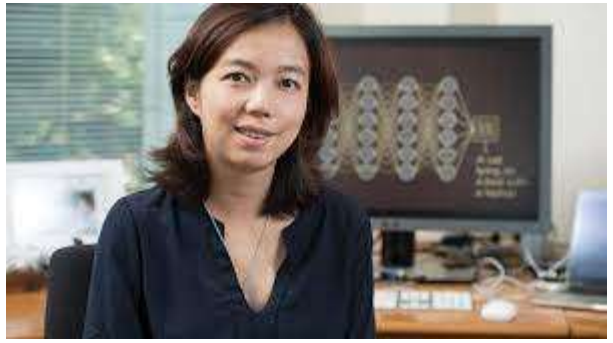
LeNet-5

ImageNet



https://class.vision/blog/what-is-imagenet/

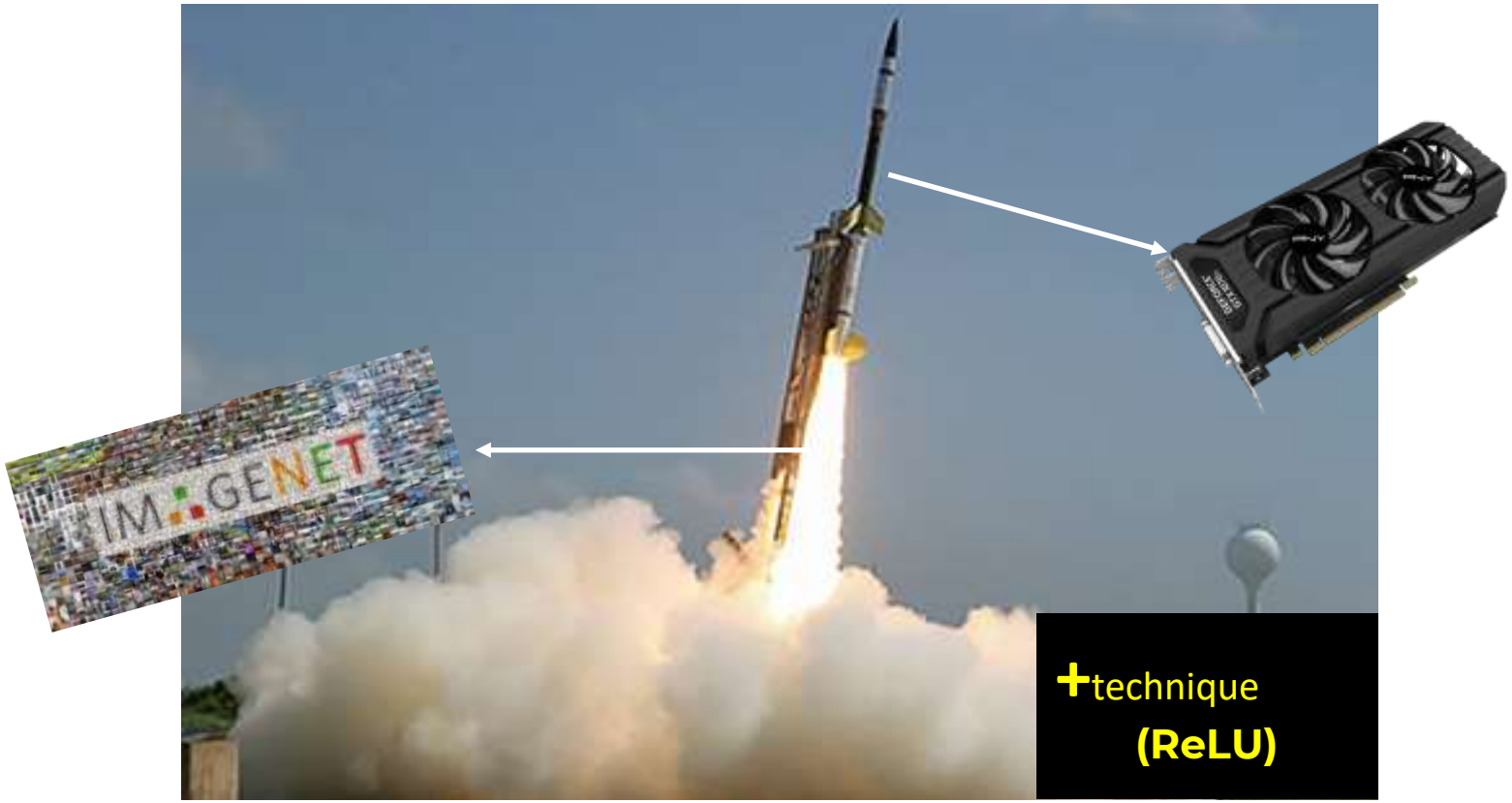
Fei-Fei Li

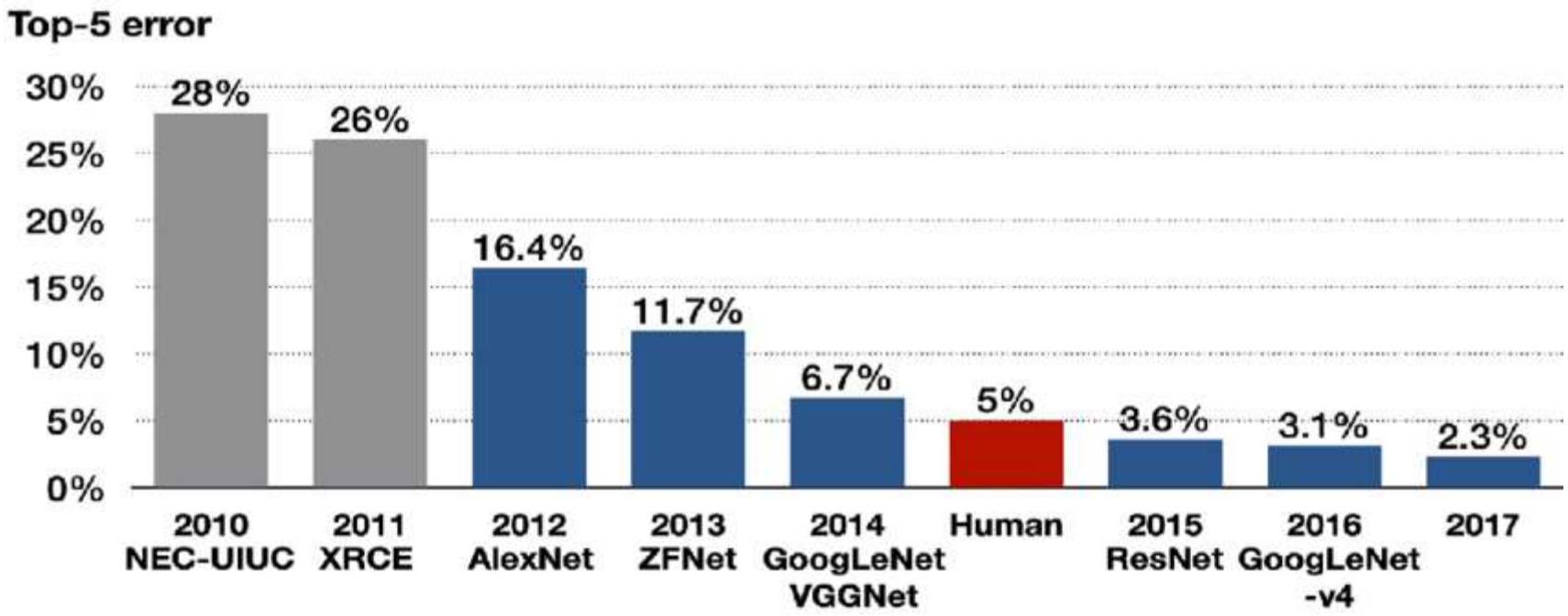


Fei-Fei Li

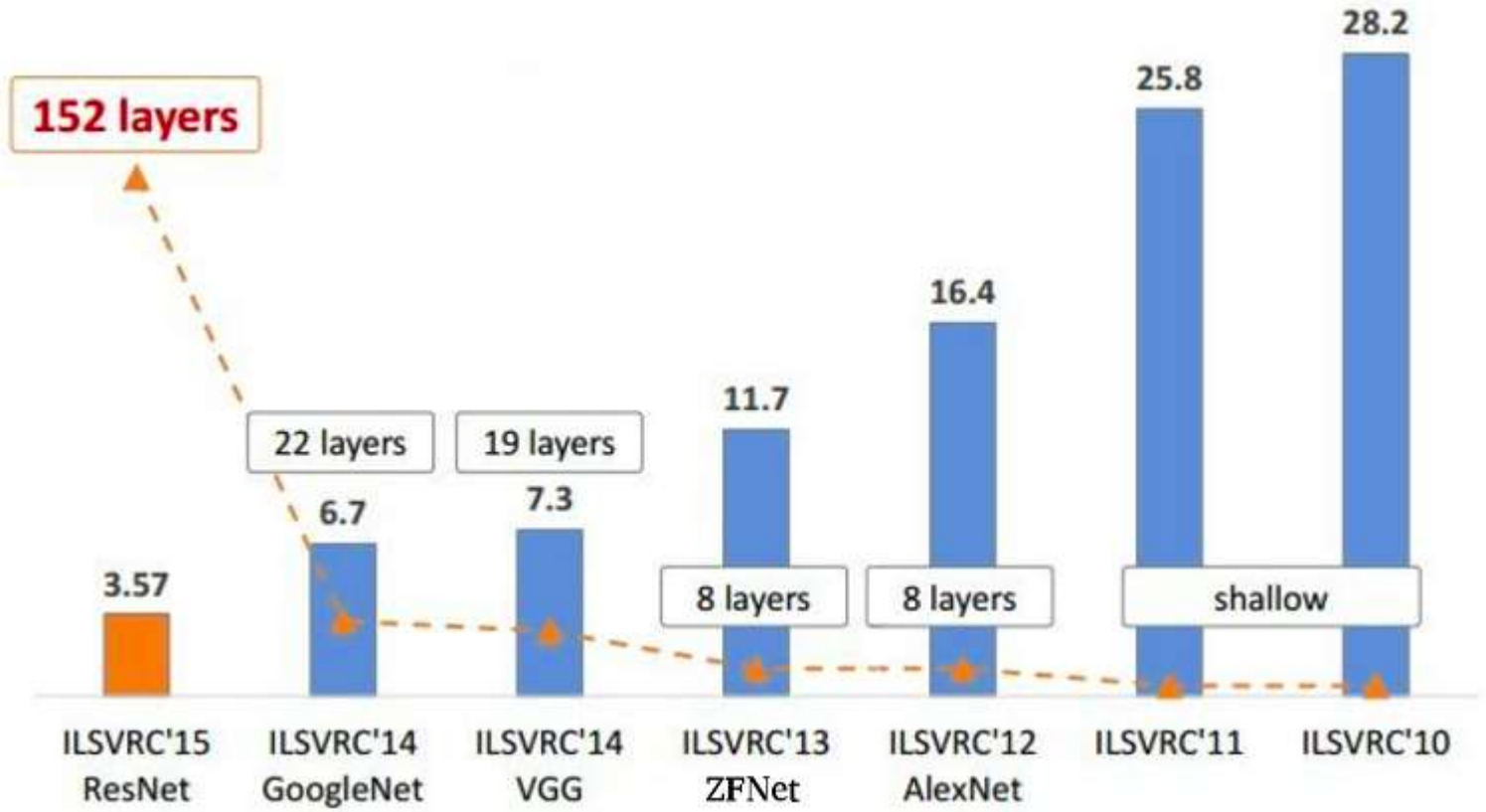


GPU+Data+Relu

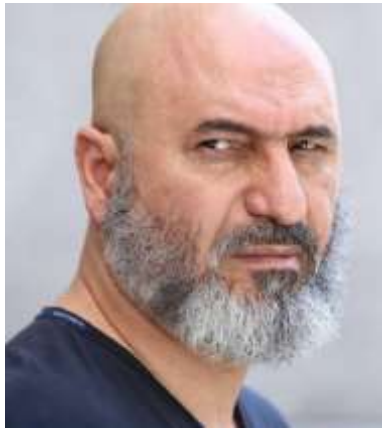


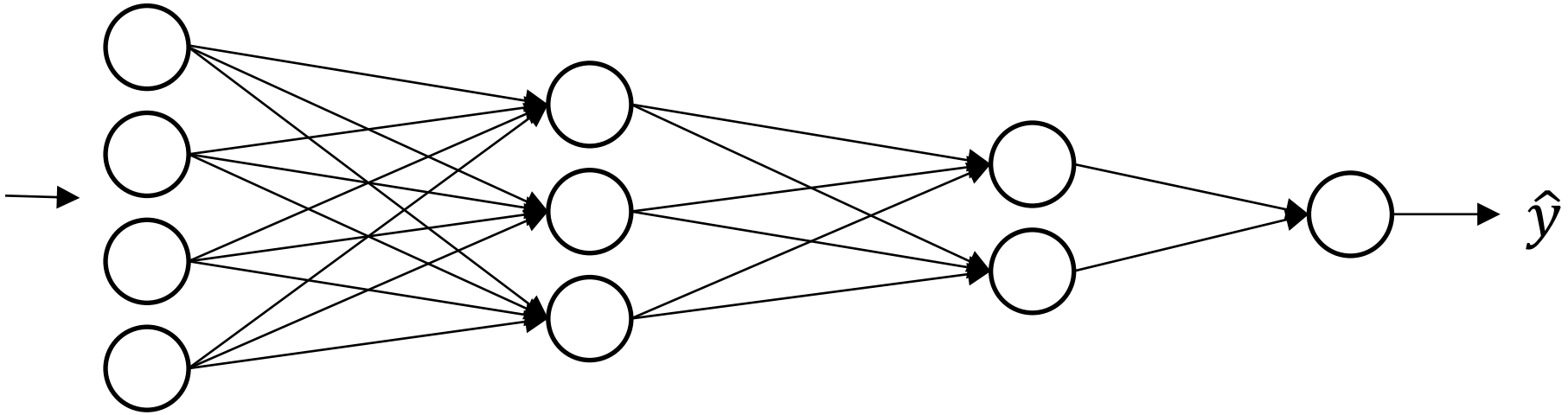

Top-5 error
30%
25%
20%
15%
10%
5%
0%
28%
26%
16.4%
11.7%
6.7%
5%
3.6%
3.1%
2.3%
2010 NEC-UIUC
2011 XRCE
2012 AlexNet
2013 ZFNet
2014 GoogLeNet VGGNet
Human
2015 ResNet
2016 GoogLeNet -v4
2017

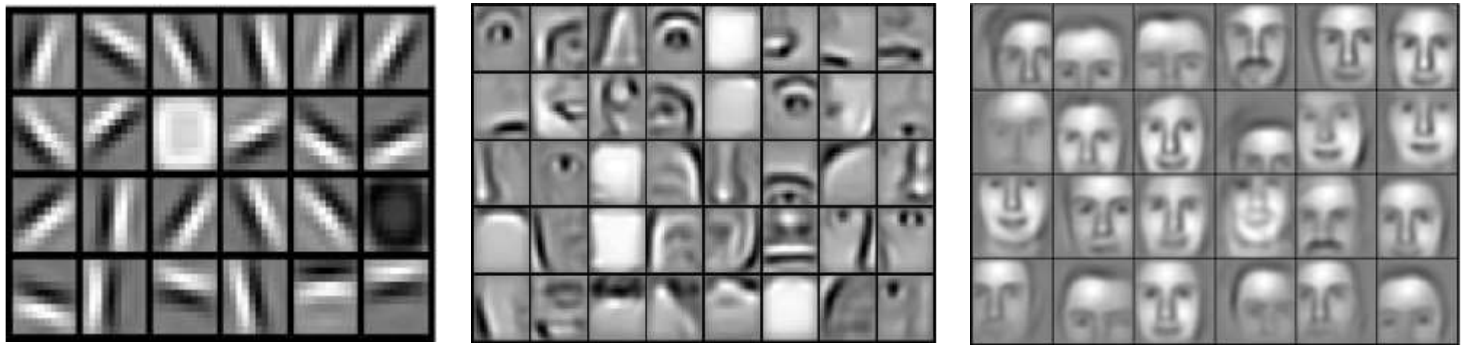
بعد از ۲۰۱۲...

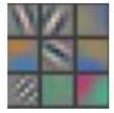

152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ZFNet
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10

شهود لزوم سلسله مراتب و عمق شبکه



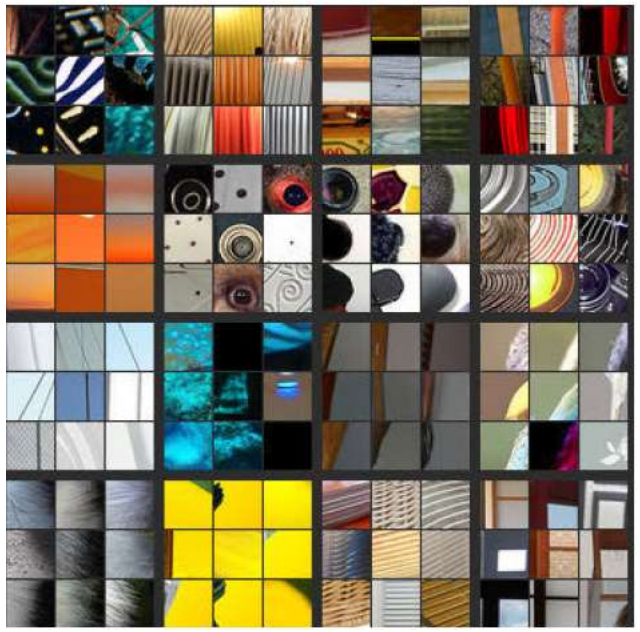






Layer 1

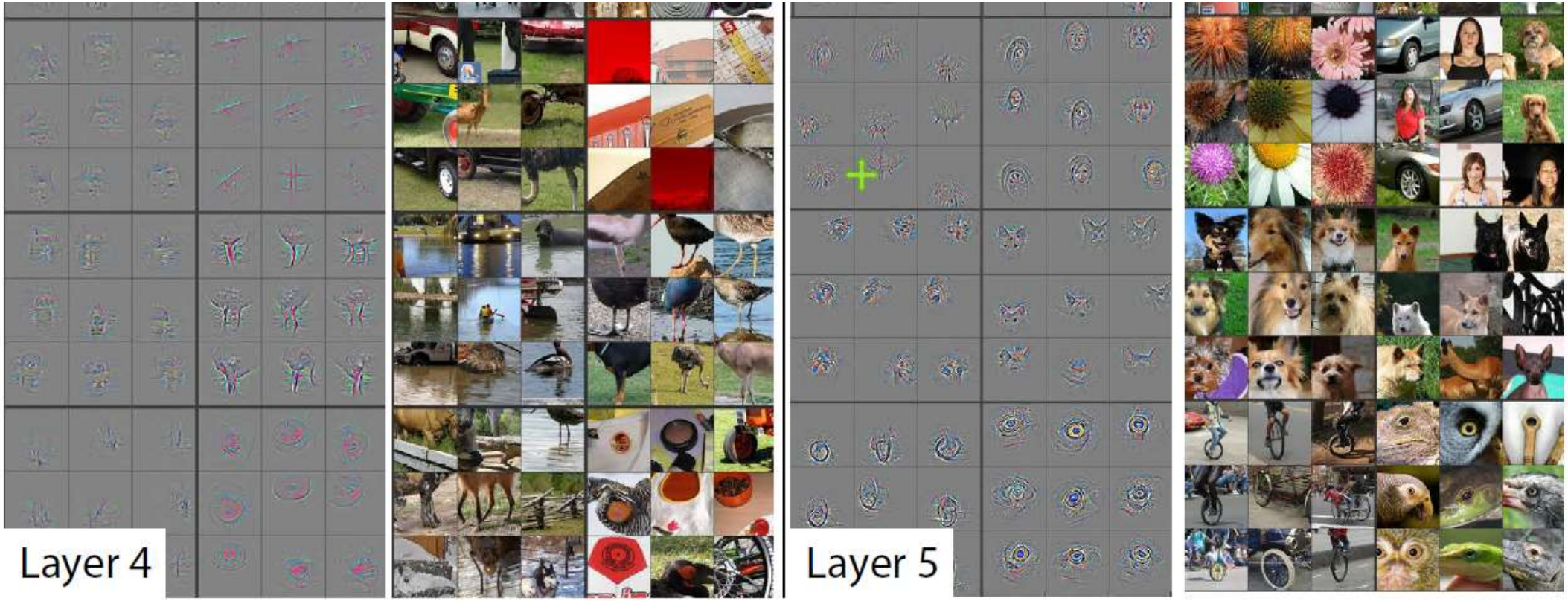




Layer 2

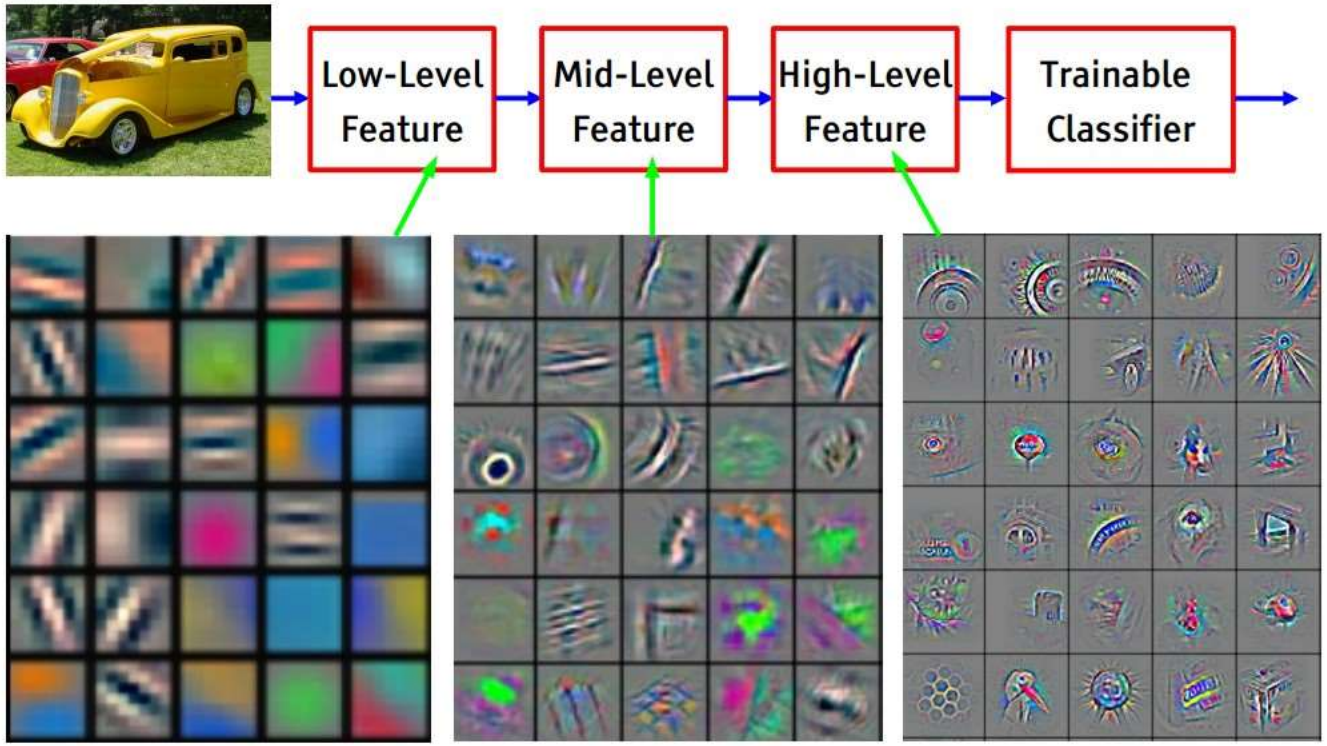
Layer 3




Layer 4
Layer 5

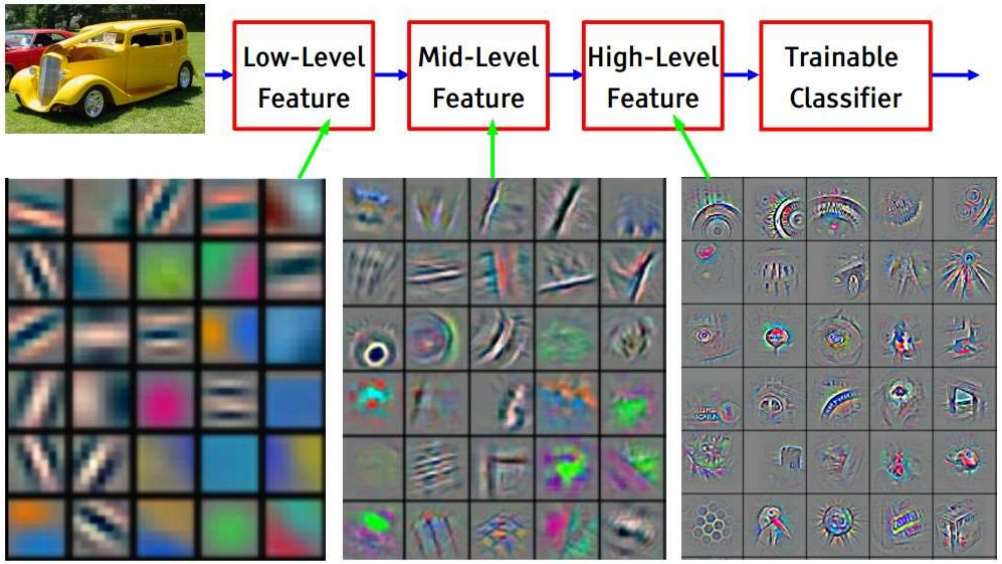
شهود لزوم سلسله مراتب و عمق شبکه

[From recent Yann LeCun slides]



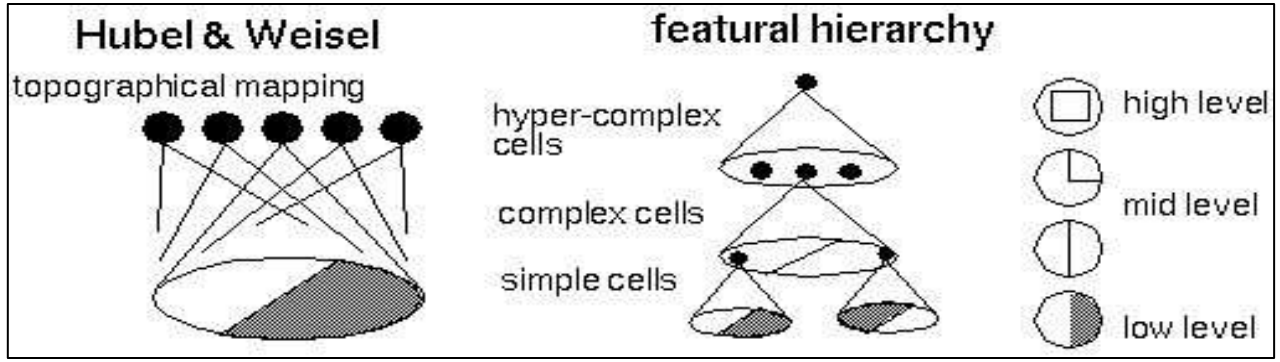
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

شهود لزوم سلسله مراتب و عمق شبکه



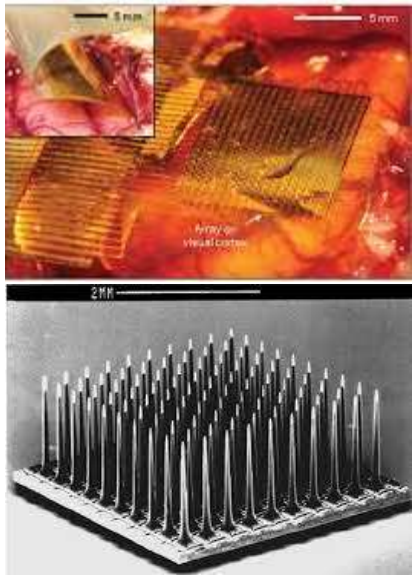
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

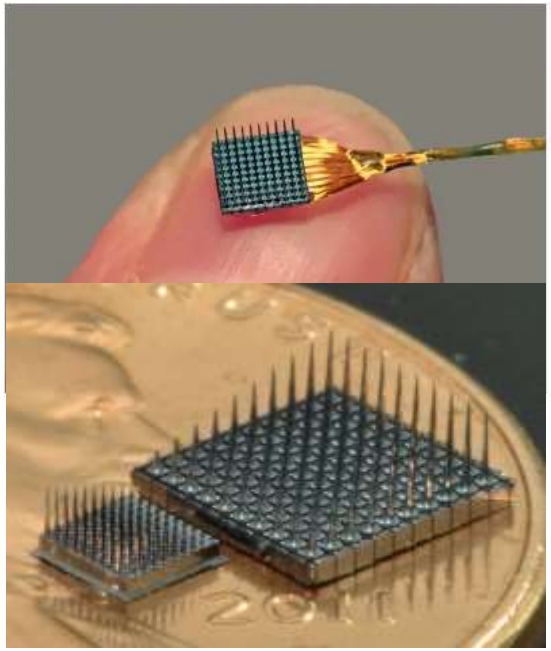
[From recent Yann LeCun slides]



آزمایش و یافته های هابل و ویزل







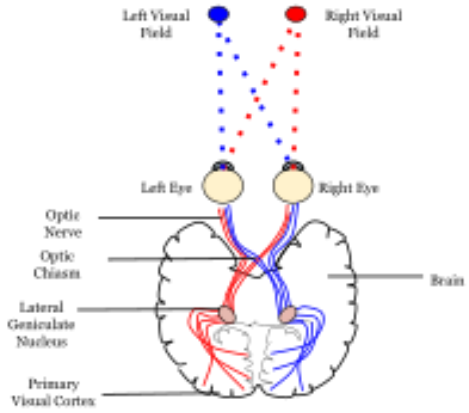
آزمایش و یافته های هابل و ویزل

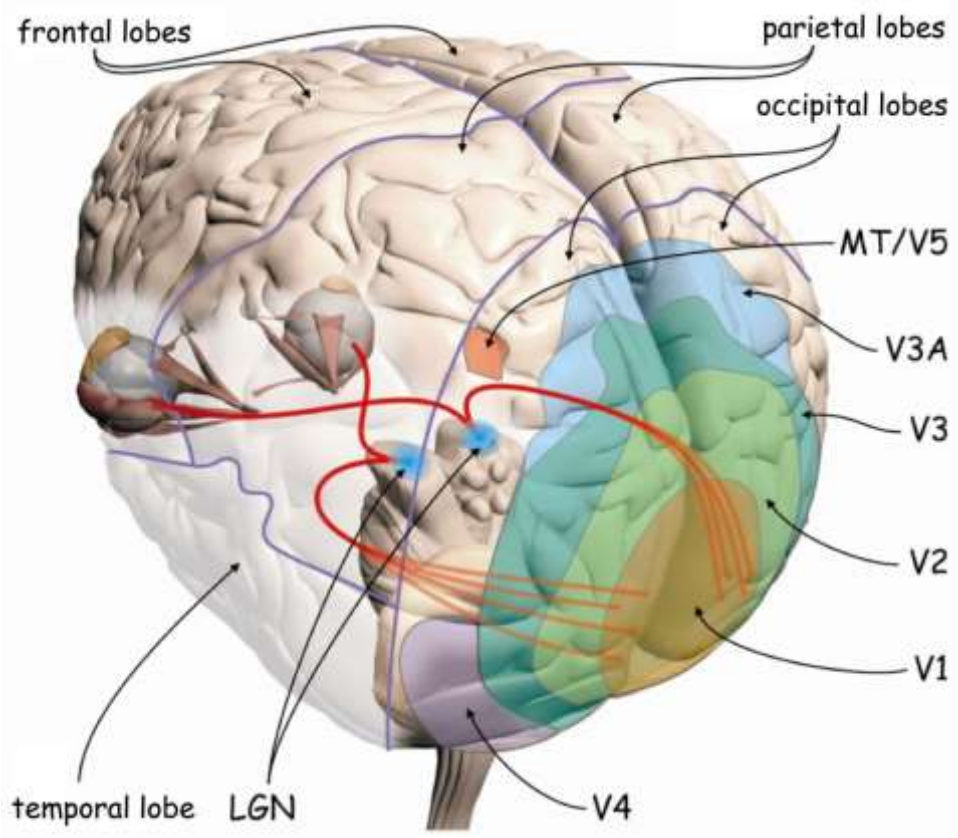









Left Visual Field
Right Visual Field
Left Eye
Right Eye
Optic Nerve
Optic Chiasm
Brain
Lateral Geniculate Nucleus
Primary Visual Cortex

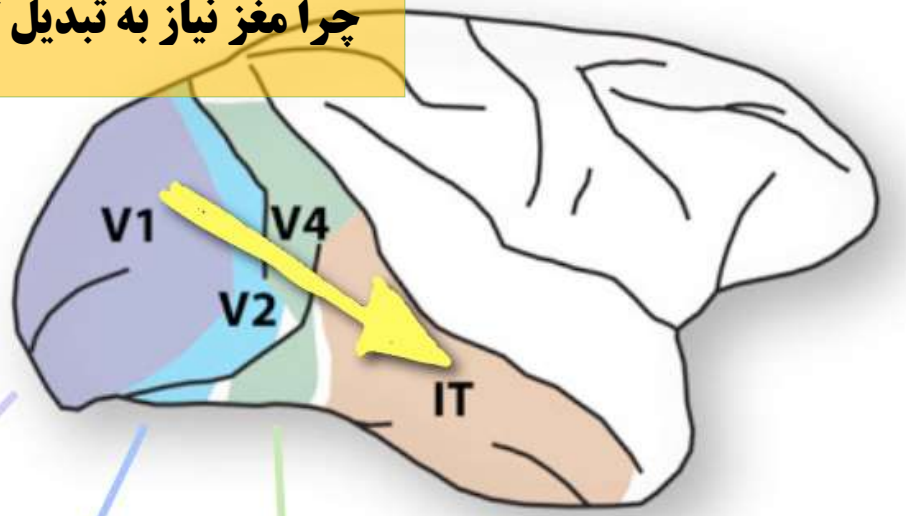

frontal lobes
parietal lobes
occipital lobes
MT/V5
V3A
V3
V2
V1
V4
temporal lobe
LGN

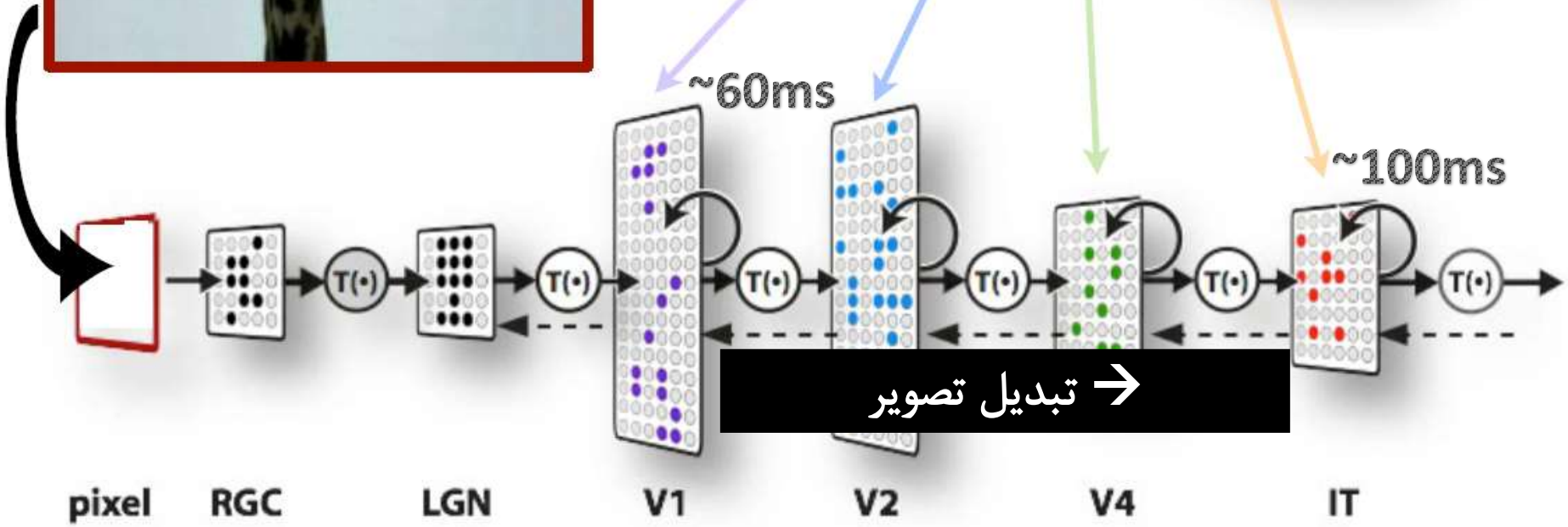
مسیر قدامی بینایی در مغز

چرا مغز نیاز به تبدیل تصاویر از محیط پیکسل دارد؟

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

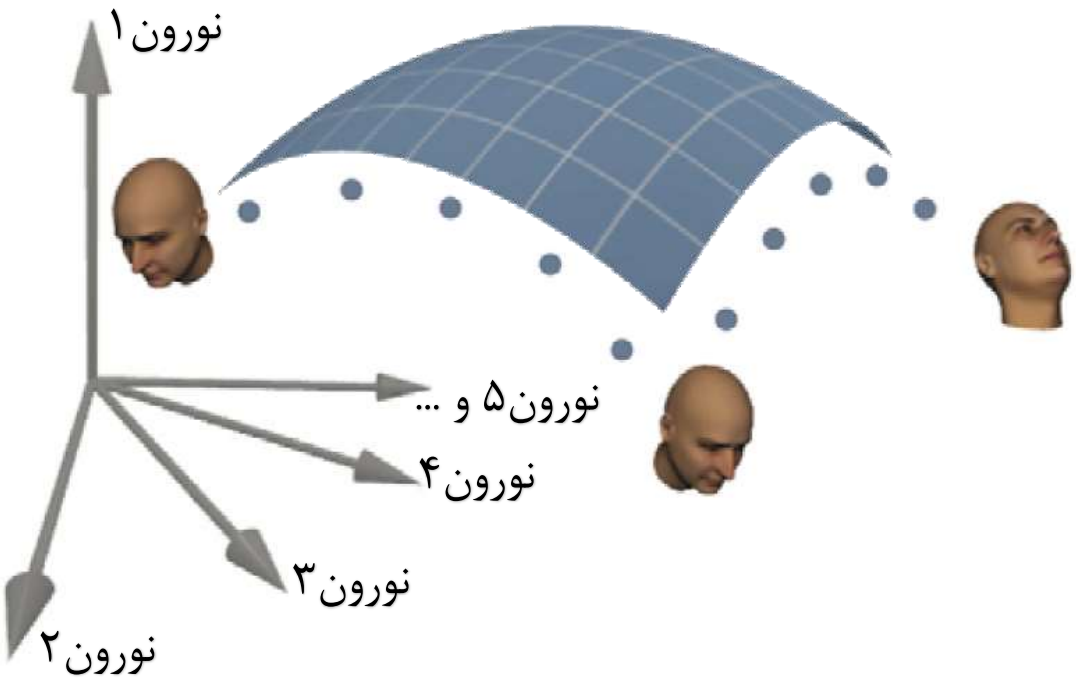






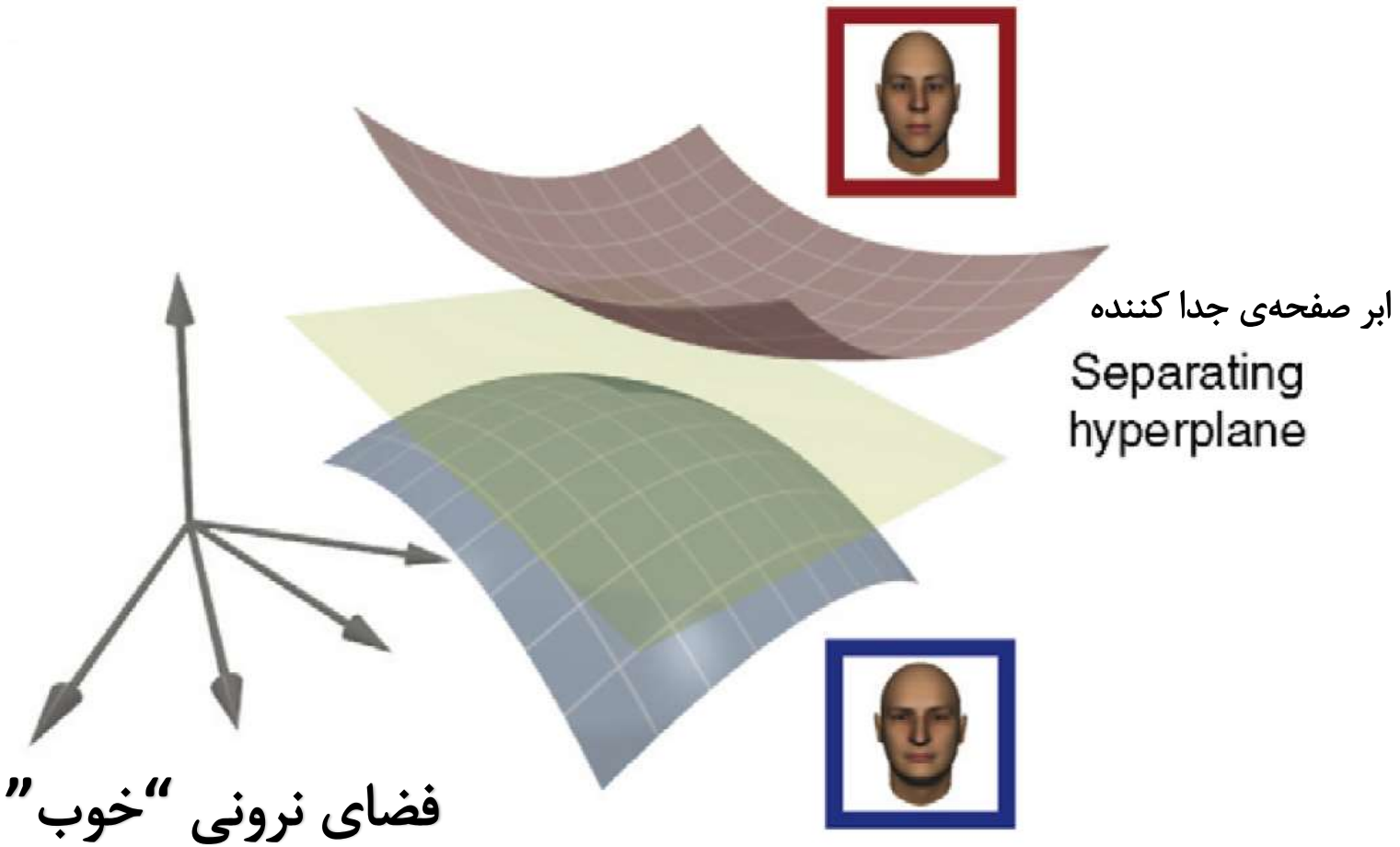
تبدیل تصویر ←

فضای نرونی و خمینه (manifold)

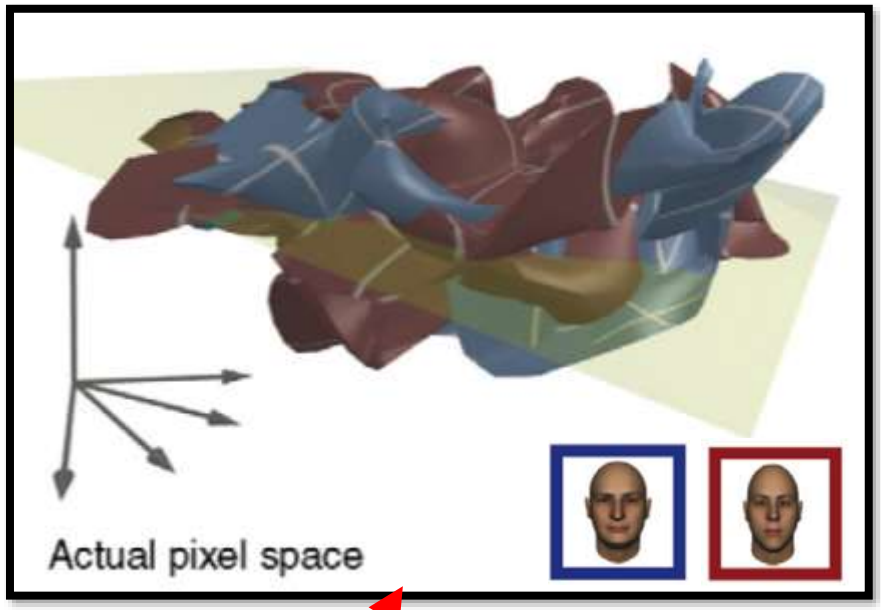


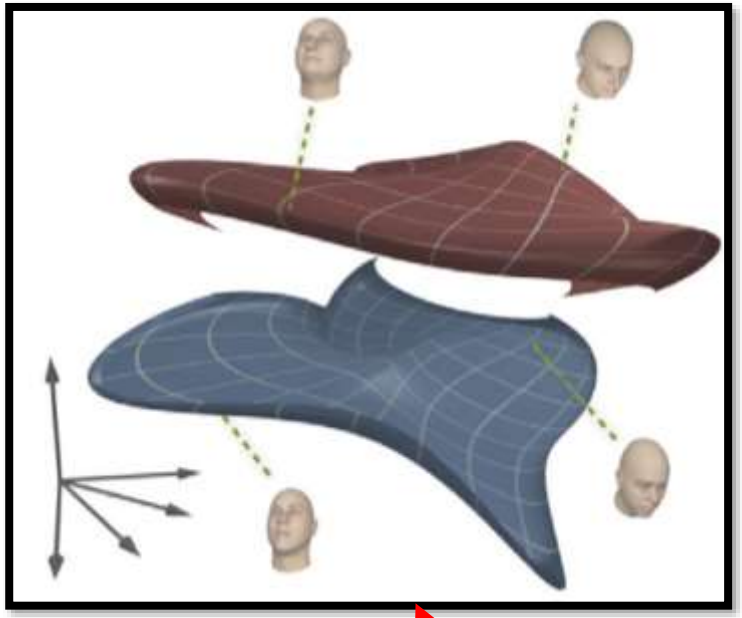
*DiCarlo and Cox , **TICS** (2007)*

فضای نرونی خوب



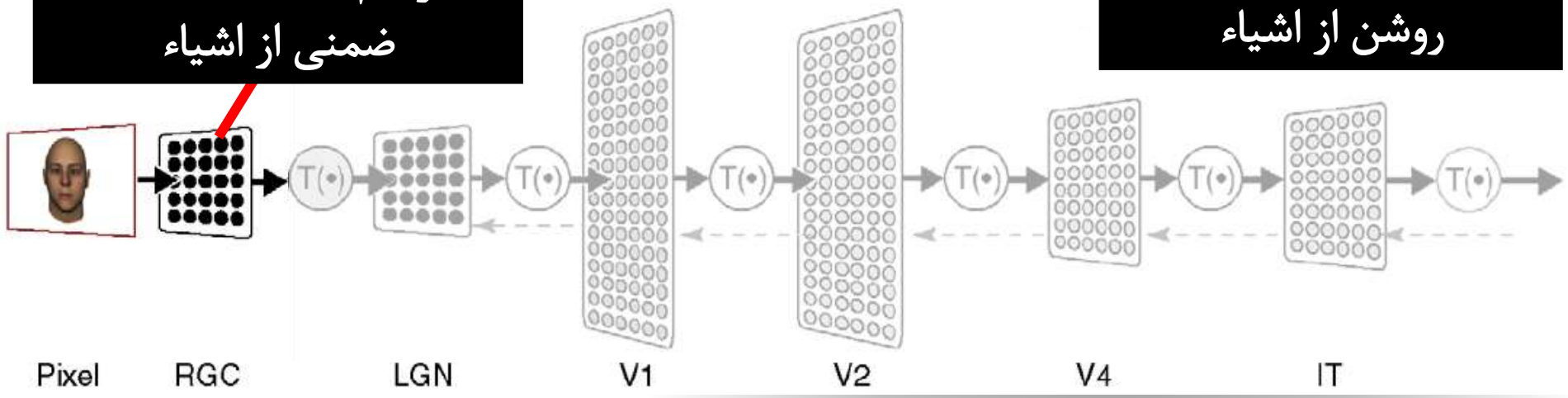
تبدیل فضای نرونی در مغز





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)*

یادگیری عمیق و تصاویر بزرگ

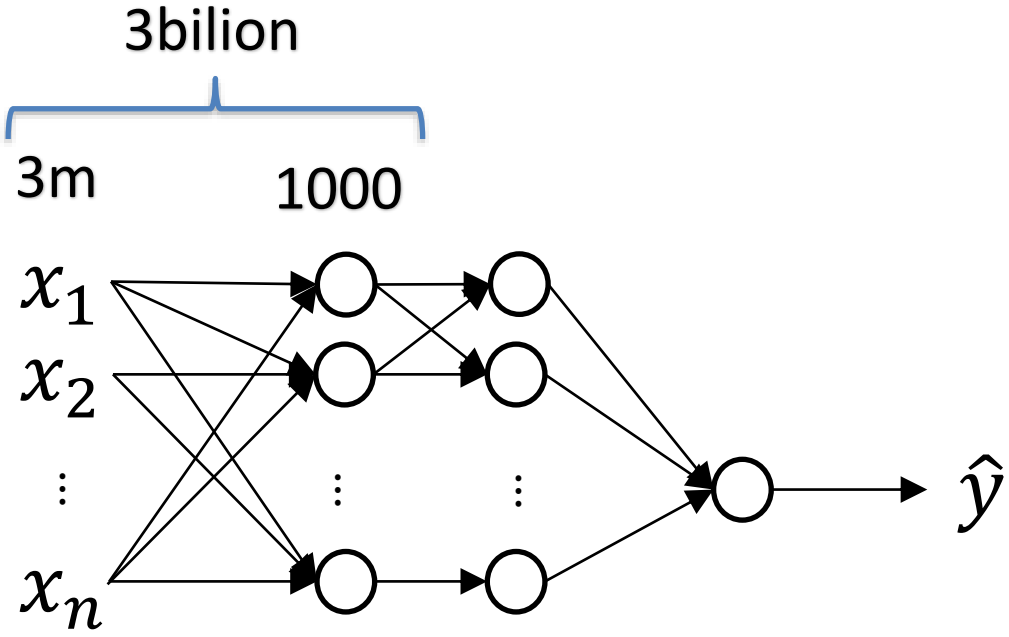


64x64**x3**
=12288

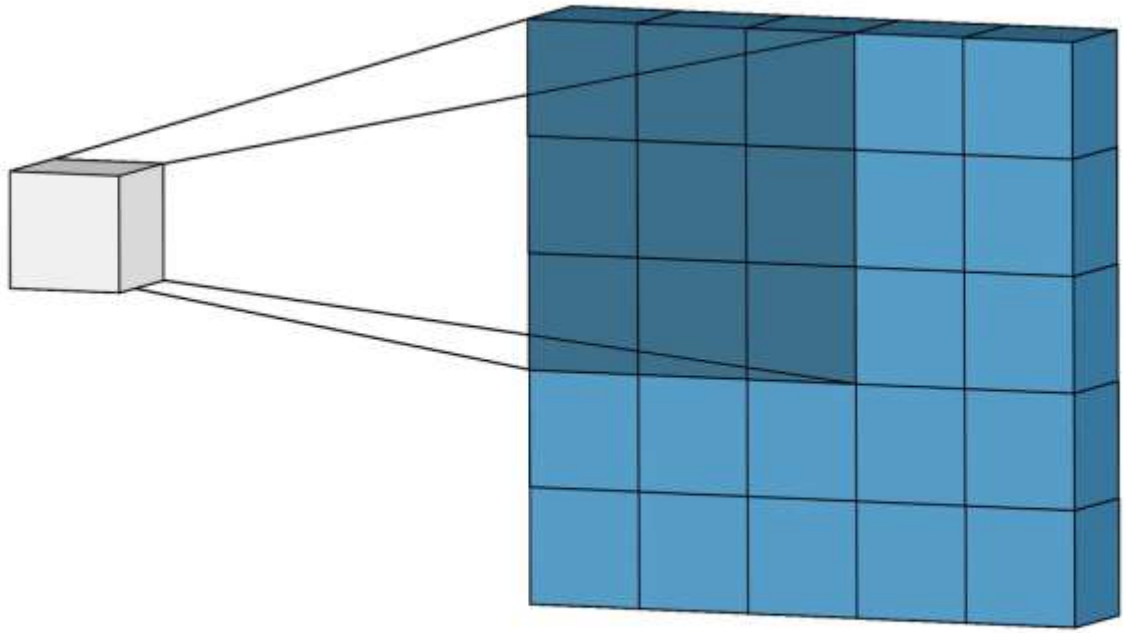


1000x1000**x3**
=3000000 = 3m

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

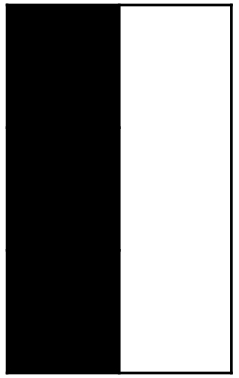
=

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

مقدمات convolution – لبه‌یابی

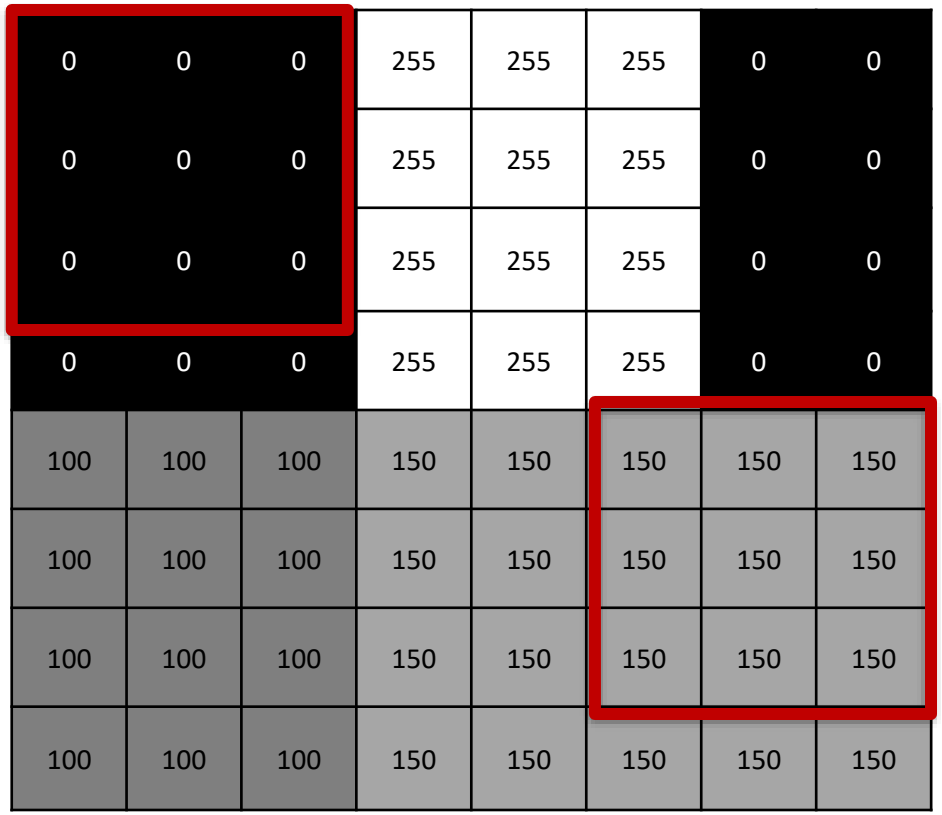
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150

-1	0	+1
-1	0	+1
-1	0	+1



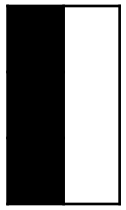
مقدمات convolution – لبه‌یابی

- روی تصویر یکنواخت: 0
  (عدم مشاهده پترن)



مقدمات convolution – لبه‌یابی

0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150



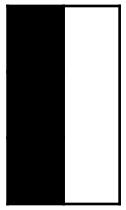
روی تصویر یکنواخت: 0
(عدم مشاهده پترن)

عدد مثبت بزرگ: 765
(مشاهده پترن به صورت قوی)

مقدمات convolution – لبه‌یابی

0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150

روی تصویر یکنواخت: 0
(عدم مشاهده پترن)



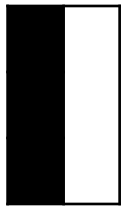
عدد مثبت بزرگ: 765
(مشاهده پترن به صورت قوی)

عدد مثبت نسبتا کوچک: 150
(مشاهده پترن به صورت ضعیف)

مقدمات convolution – لبه‌یابی

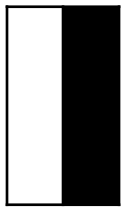
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150
100	100	100	150	150	150	150	150

روی تصویر یکنواخت: 0
(عدم مشاهده پترن)



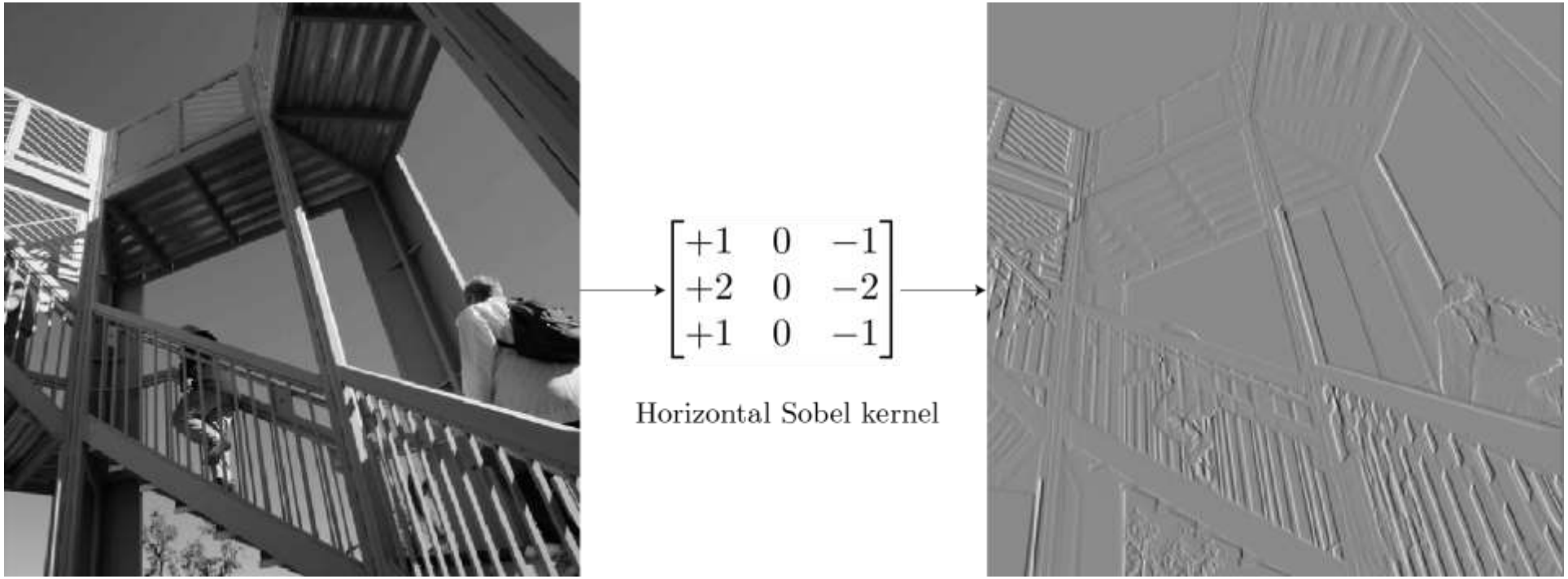
عدد مثبت بزرگ: 765
(مشاهده پترن به صورت قوی)

عدد مثبت نسبتا کوچک: 150
(مشاهده پترن به صورت ضعیف)

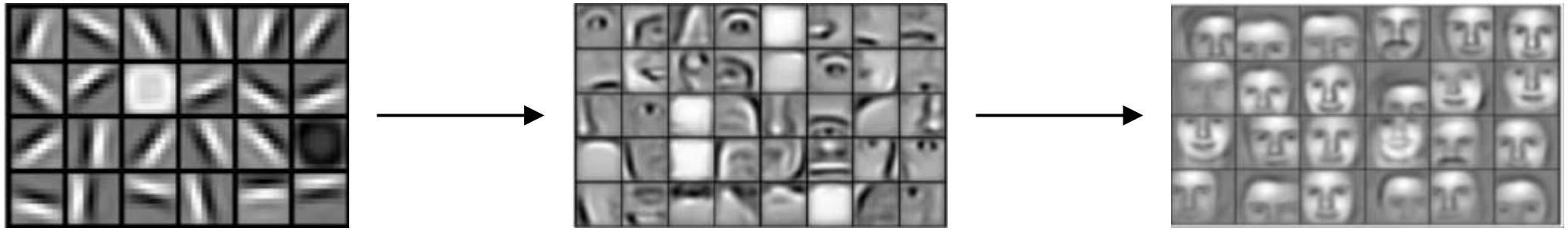


عدد مثبت نسبتا کوچک: -765
(مشاهده معکوس پترن)

مقدمات convolution



مقدمات convolution – لبه‌یابی





یادگیری فیلتر!

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

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

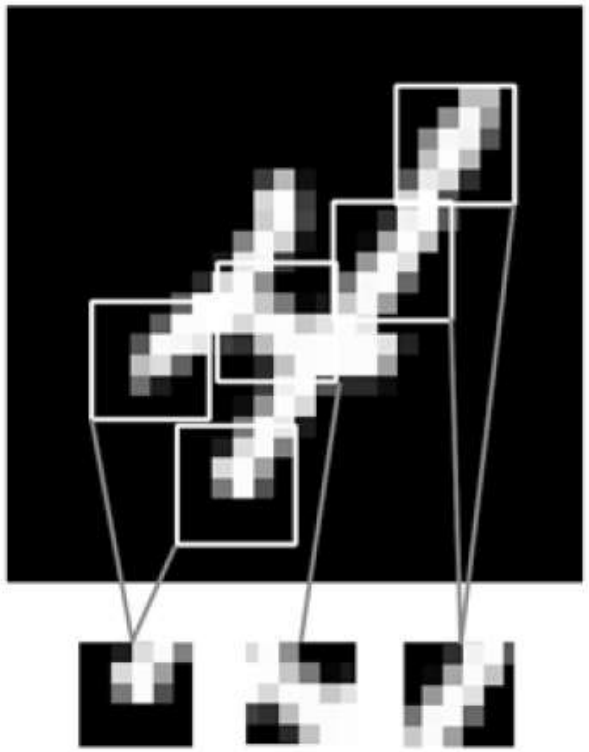
3	0	-3
10	0	-10
3	0	-3

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?

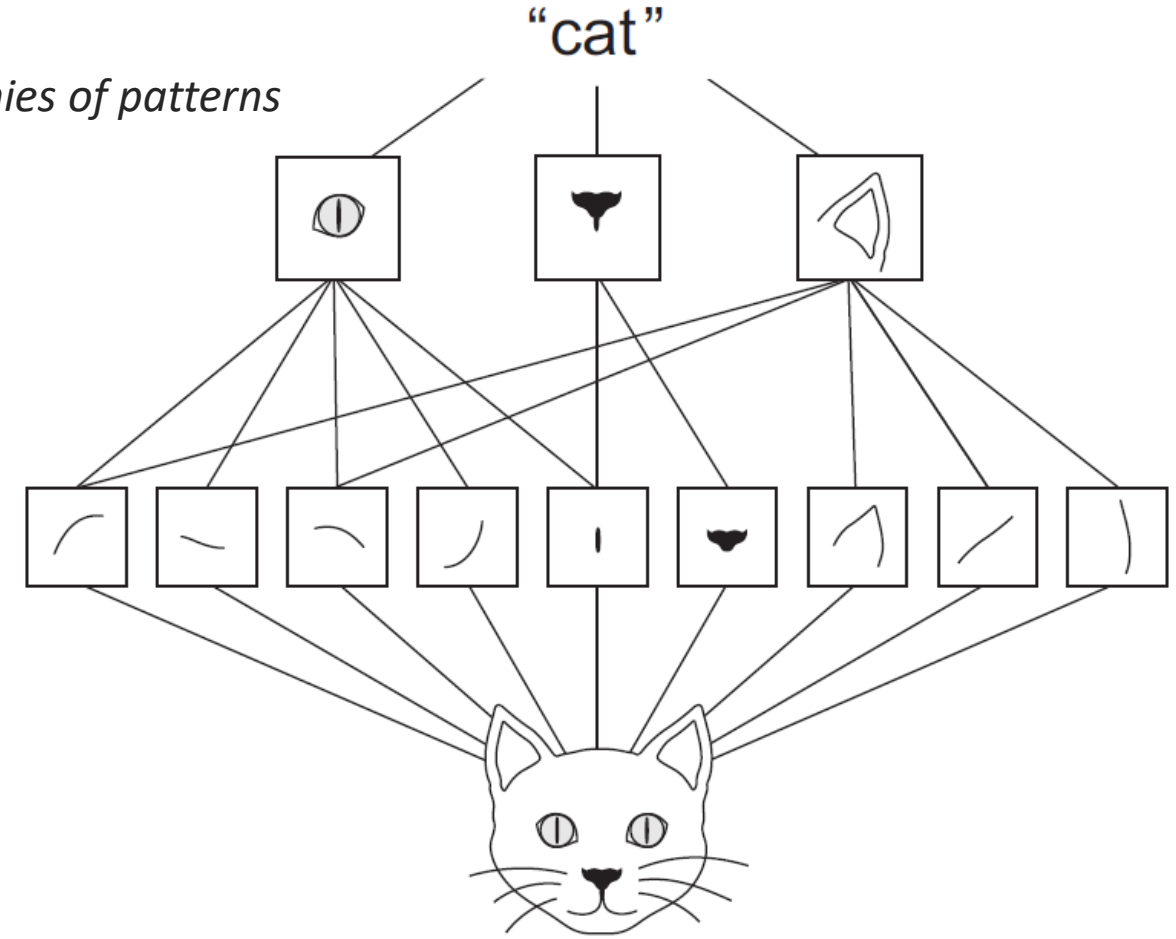
The convolution operation



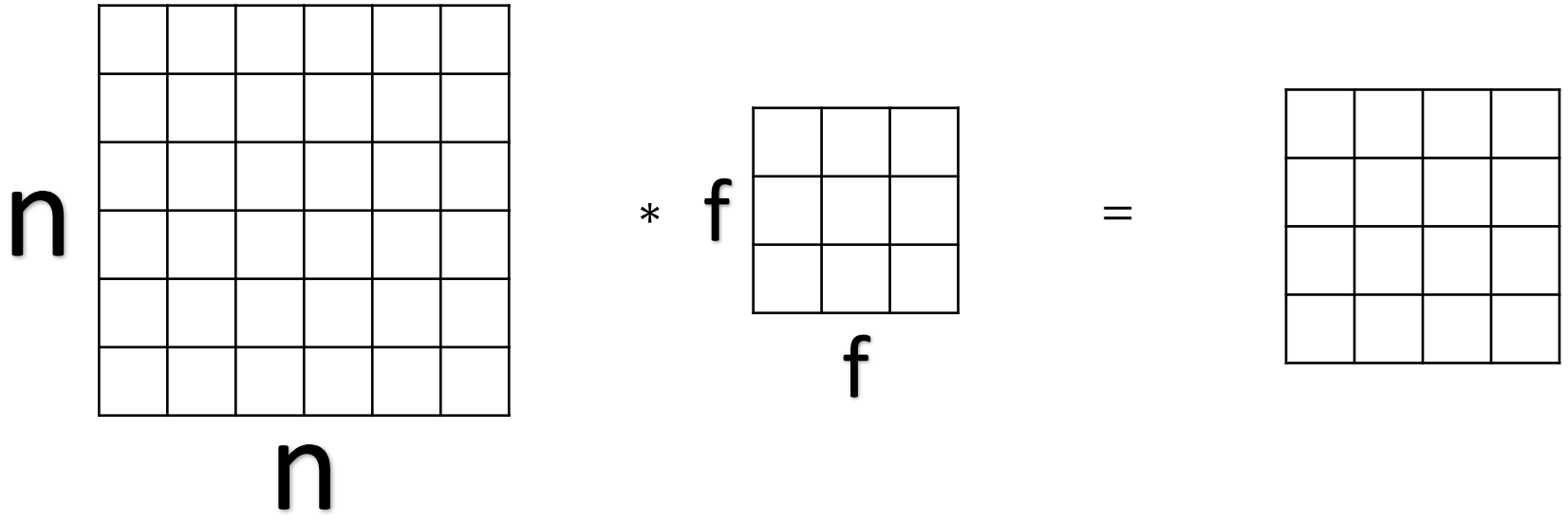
- Dense layers learn global patterns in their input feature space
- Convolution layers learn local patterns

The patterns they learn are translation invariant

They can learn spatial hierarchies of patterns



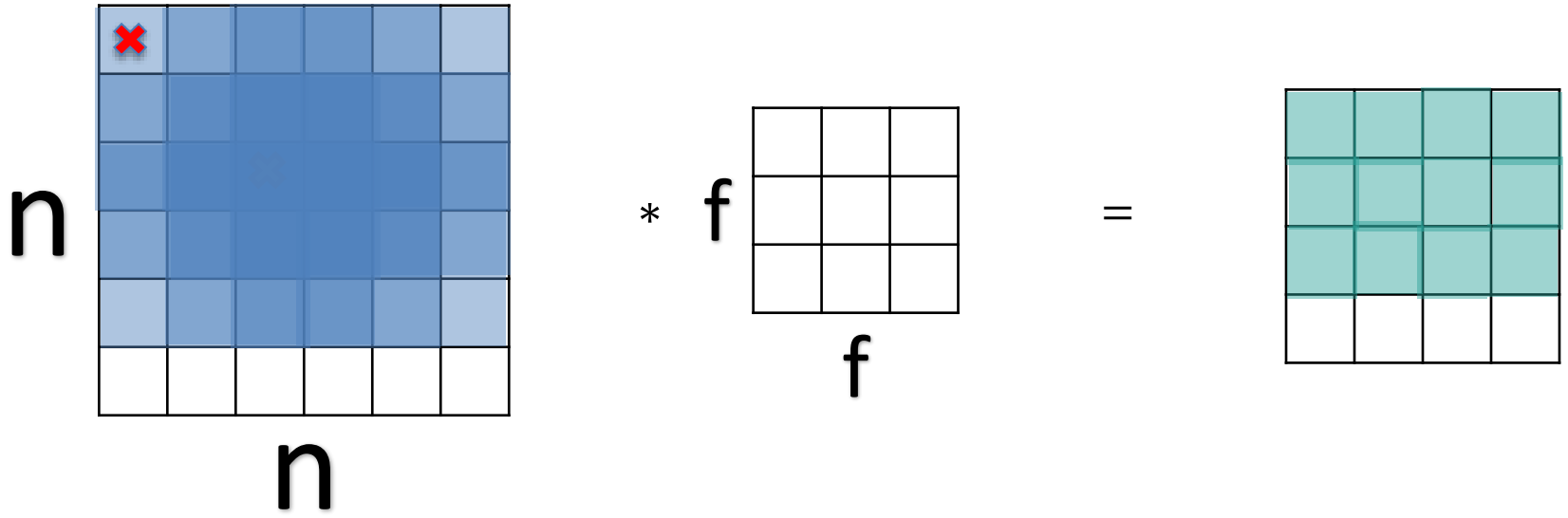
مشکلات کانولوشن!



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

$$n-f+1$$

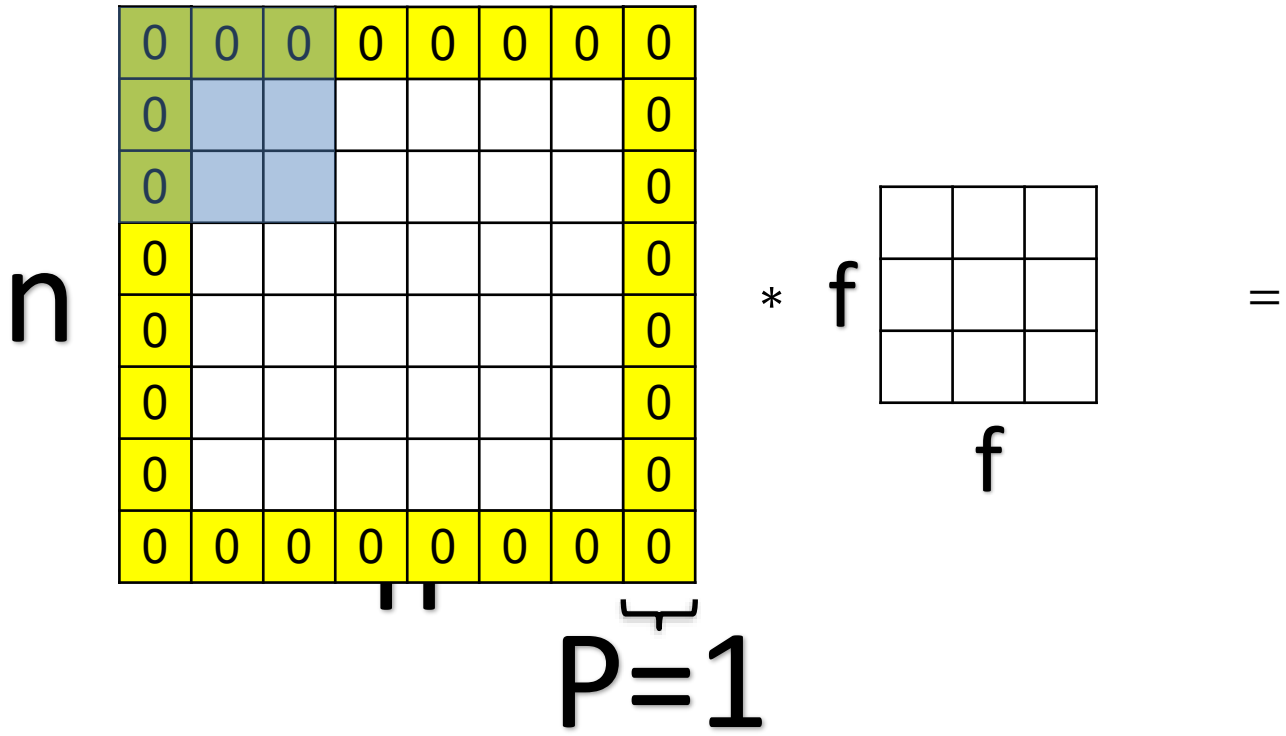
مشکلات کانولوشن!



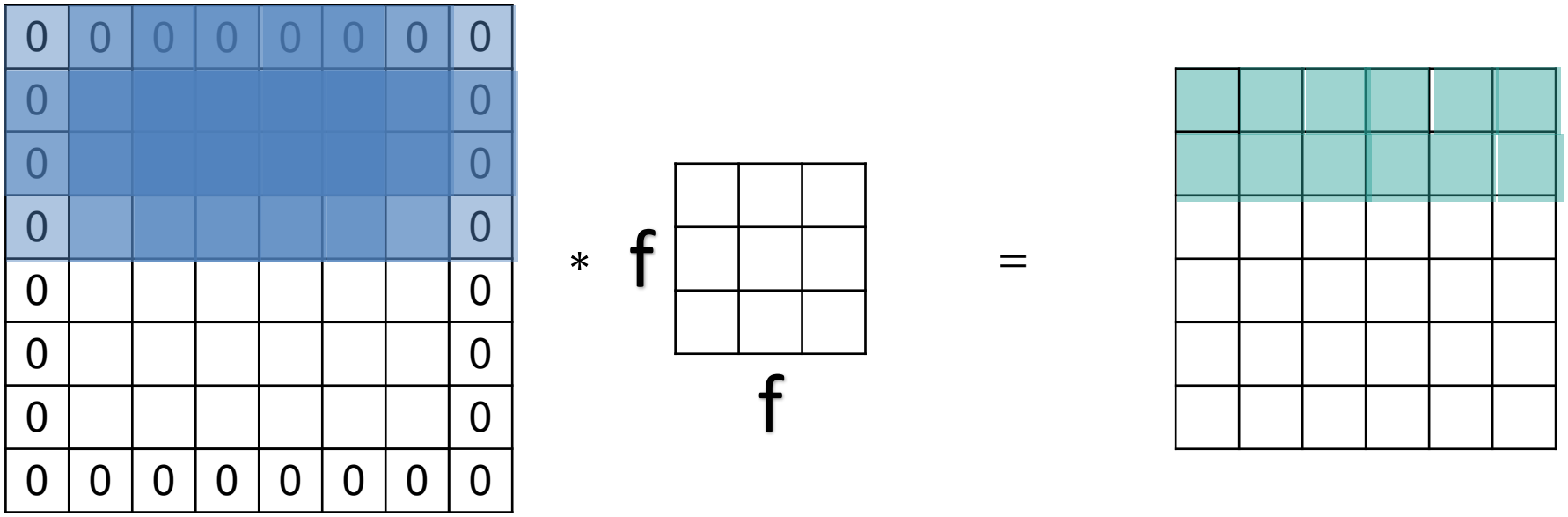
- لبه ها به اندازه ی پیکسل های درونی در کانولوشن شرکت نمیکنند
- خروجی کوچک میشود

$$n-f+1$$

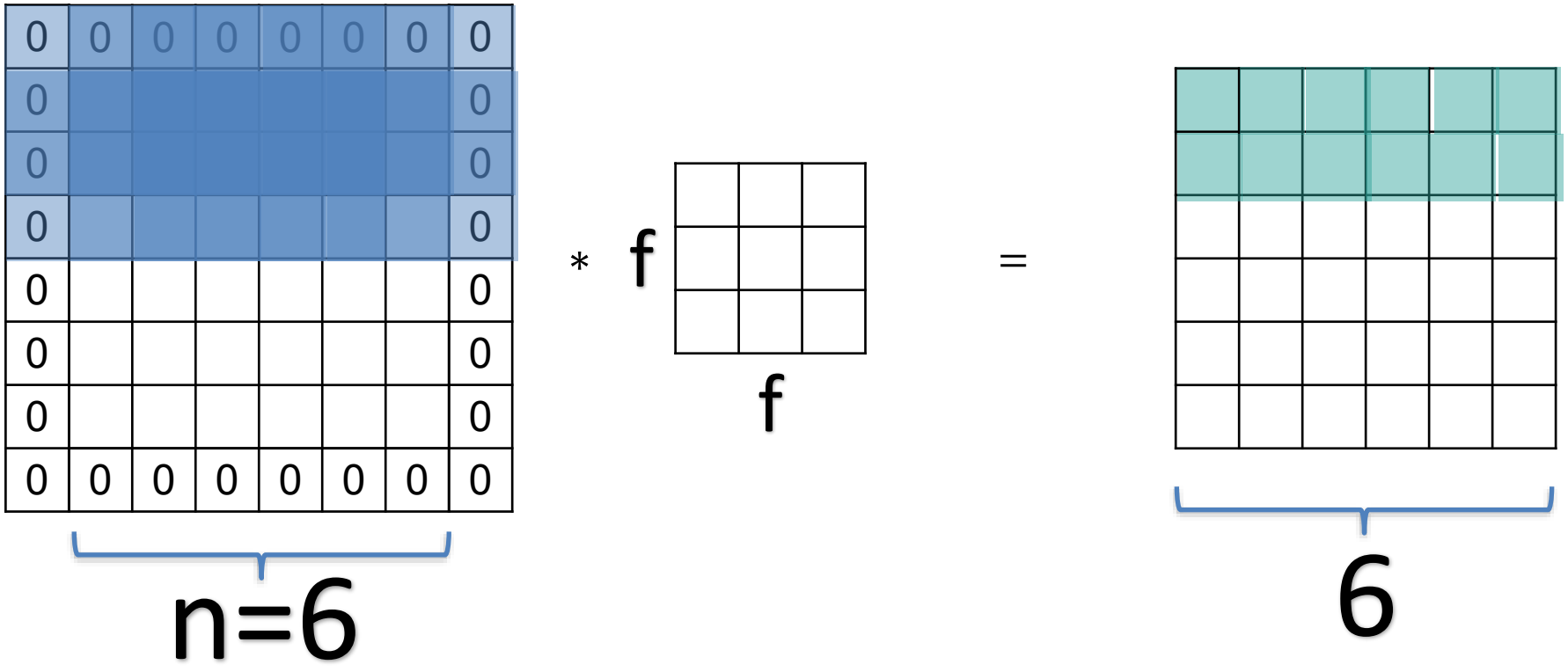
راه کار؟ padding



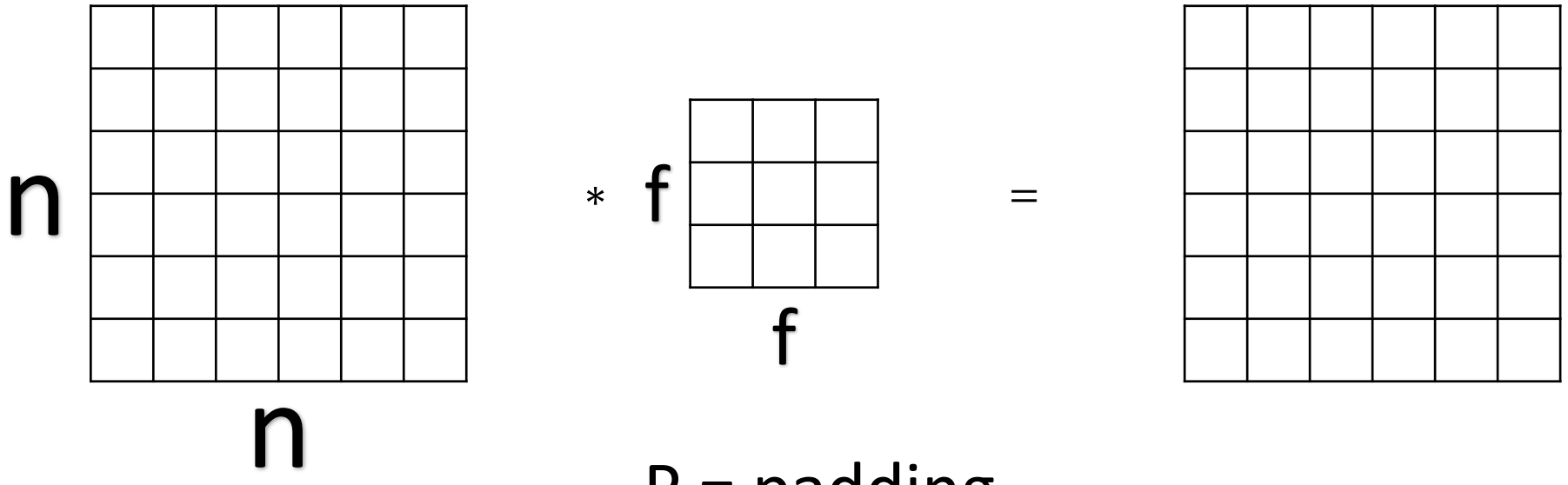
padding



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

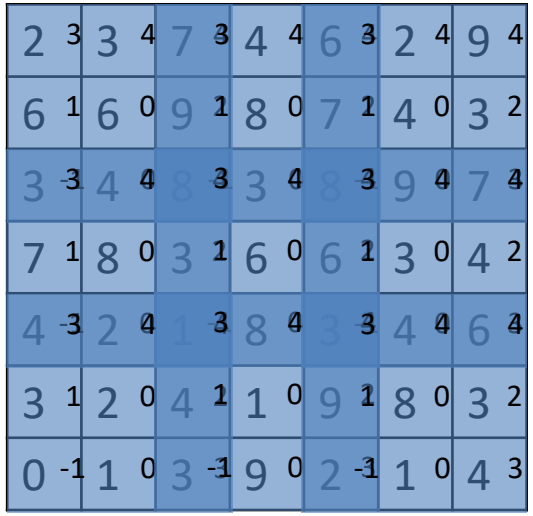
"Same": Pad so that output size is the same as the input size.

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

=

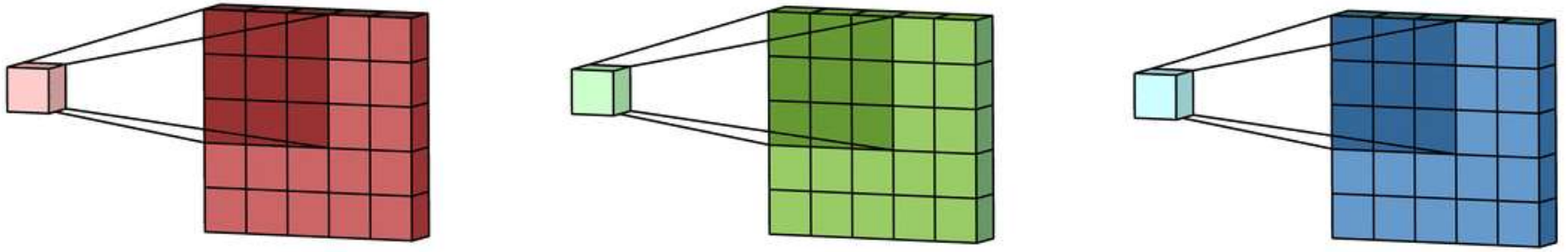
خلاصه کانولوشن و خروجی آن

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

padding p $\quad$ stride s

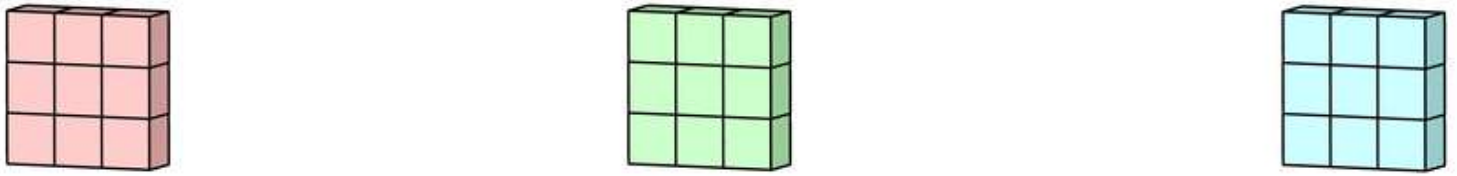
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

کانولوشن بر روی تصویر رنگی (RGB)



به ازای هر کانال رنگی یک فیلتر

کانولوشن بر روی تصویر رنگی (RGB)



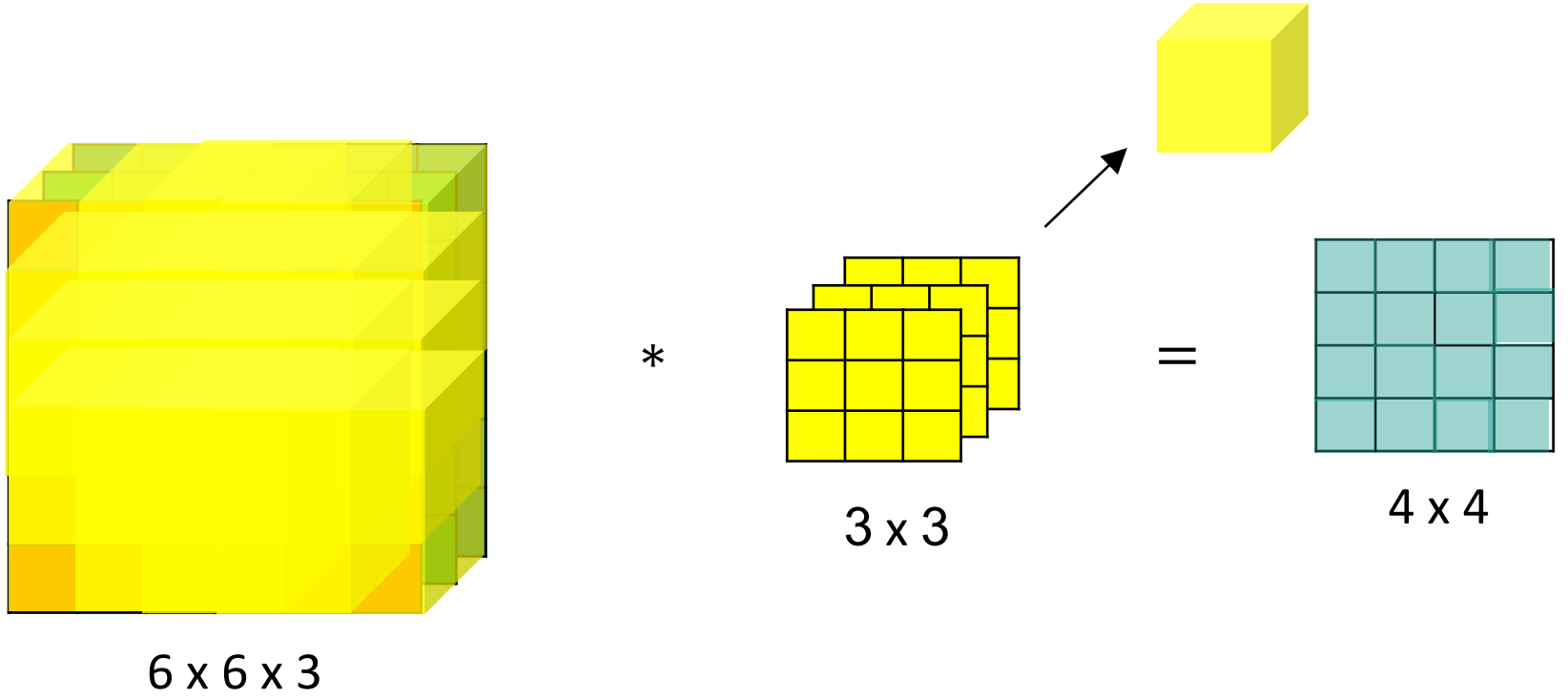
جمع نظیر به نظیر feature map یا activation map

کانولوشن بر روی تصویر رنگی (RGB)

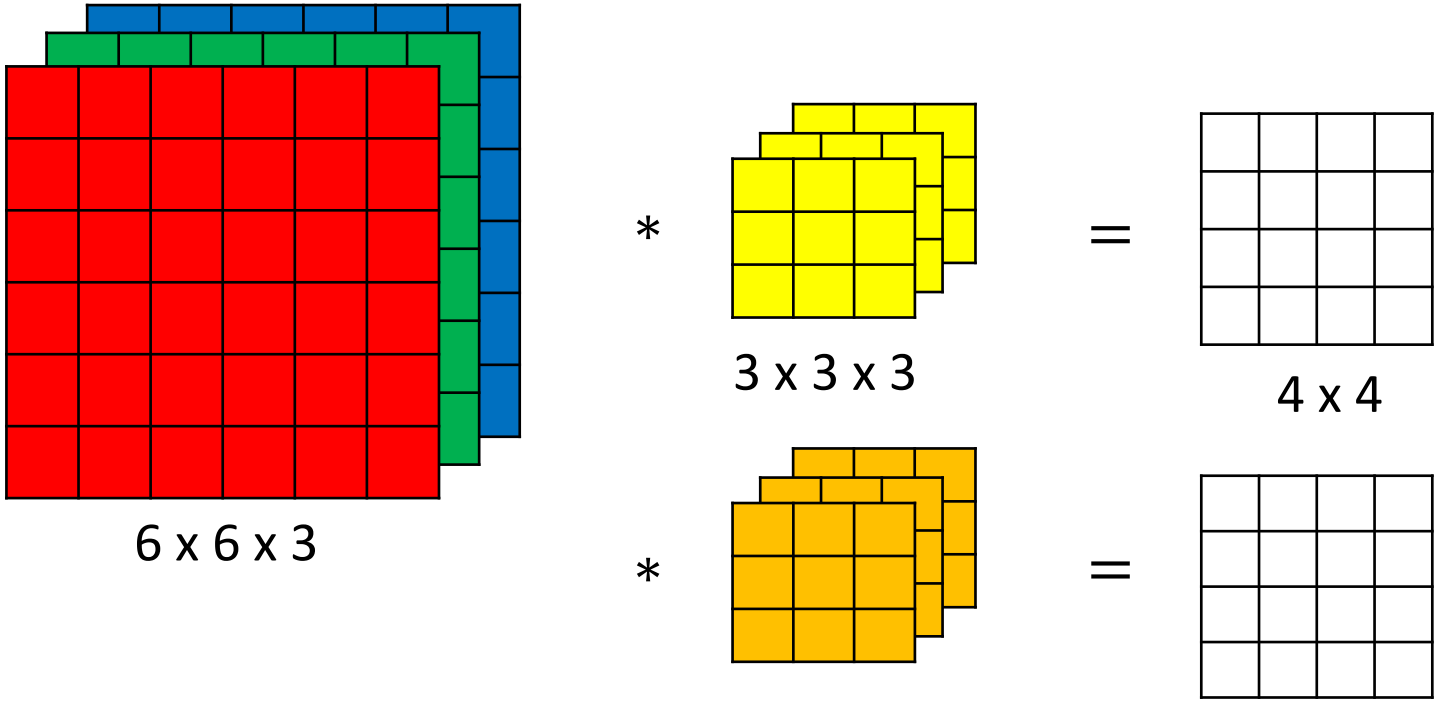


+ بایاس

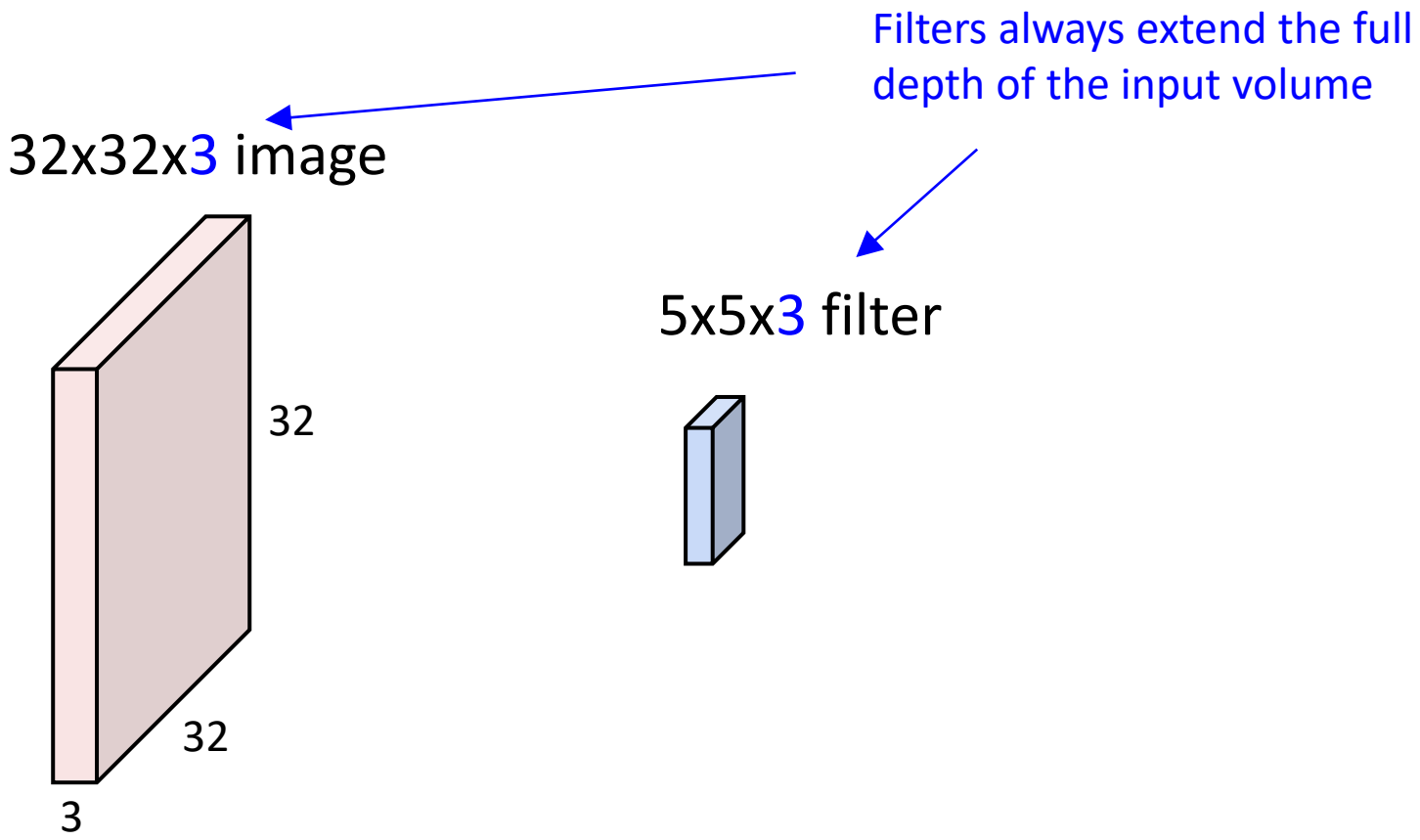
کانولوشن بر روی تصویر رنگی (RGB)



چندین فیلتر...

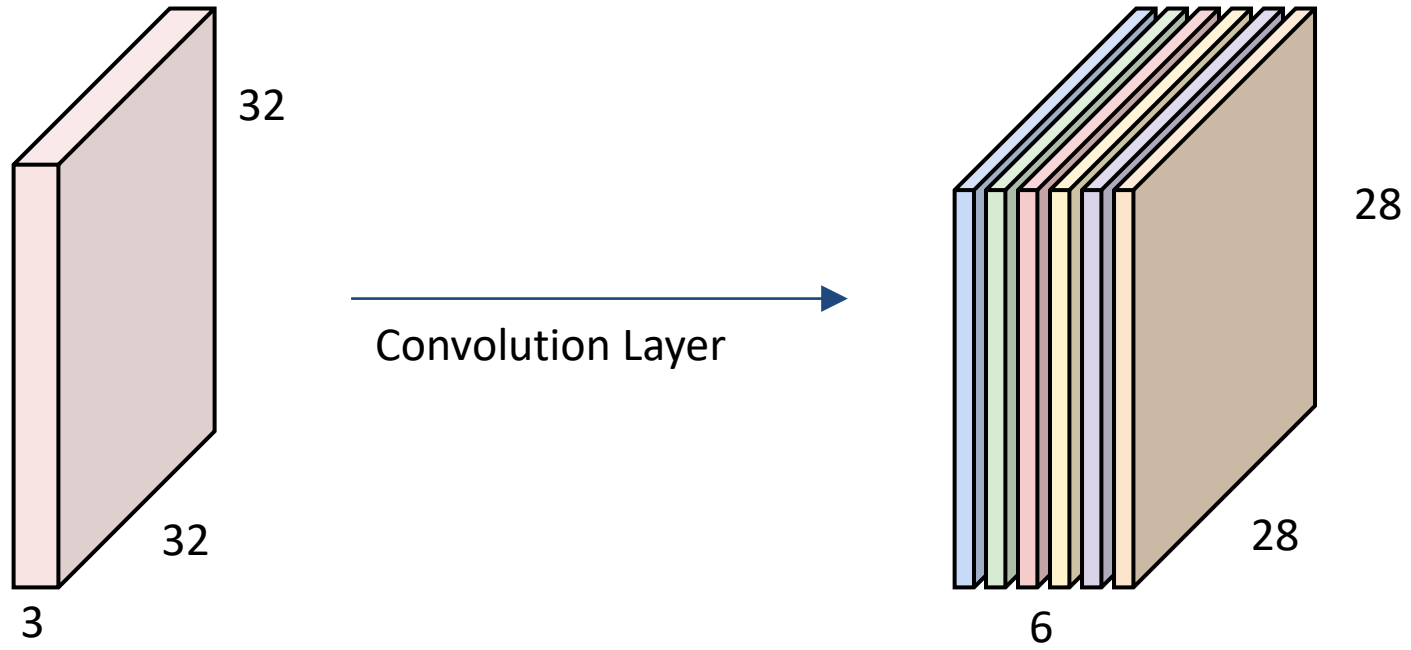


چندین فیلتر...


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

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

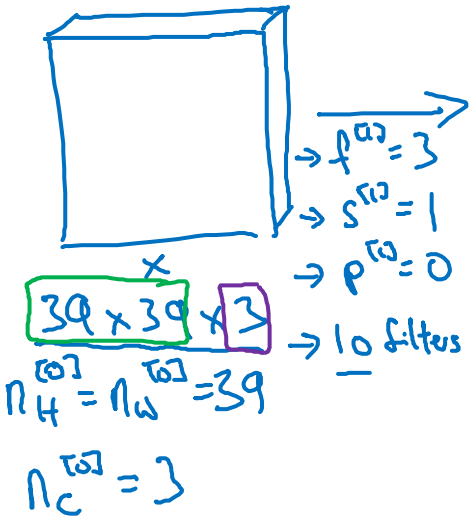
Number of parameters in one layer

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



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

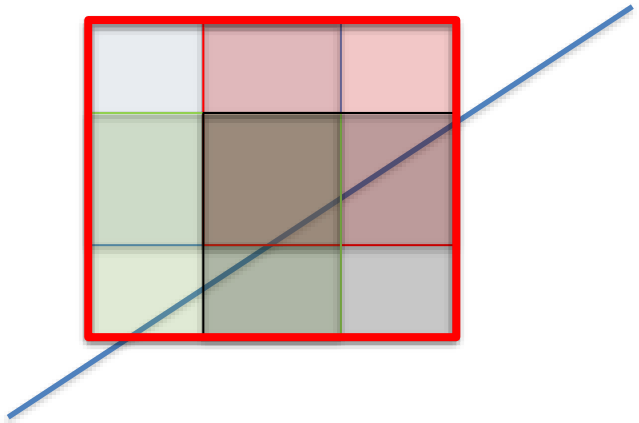
$$\frac{39+0-3}{1}+1=37$$

انواع لایه در شبکه‌های کانولوشنالی

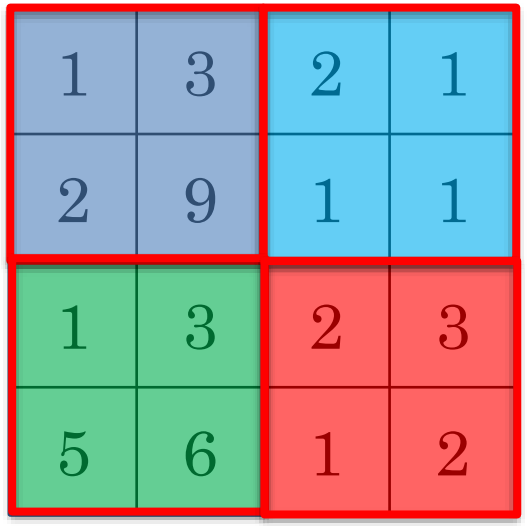
- ❑Convolution (Conv)
- ❑Pooling (Pool)
- ❑Fully Connected (FC)

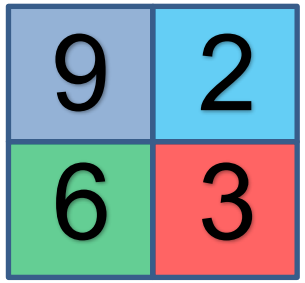
لایه‌ی pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2



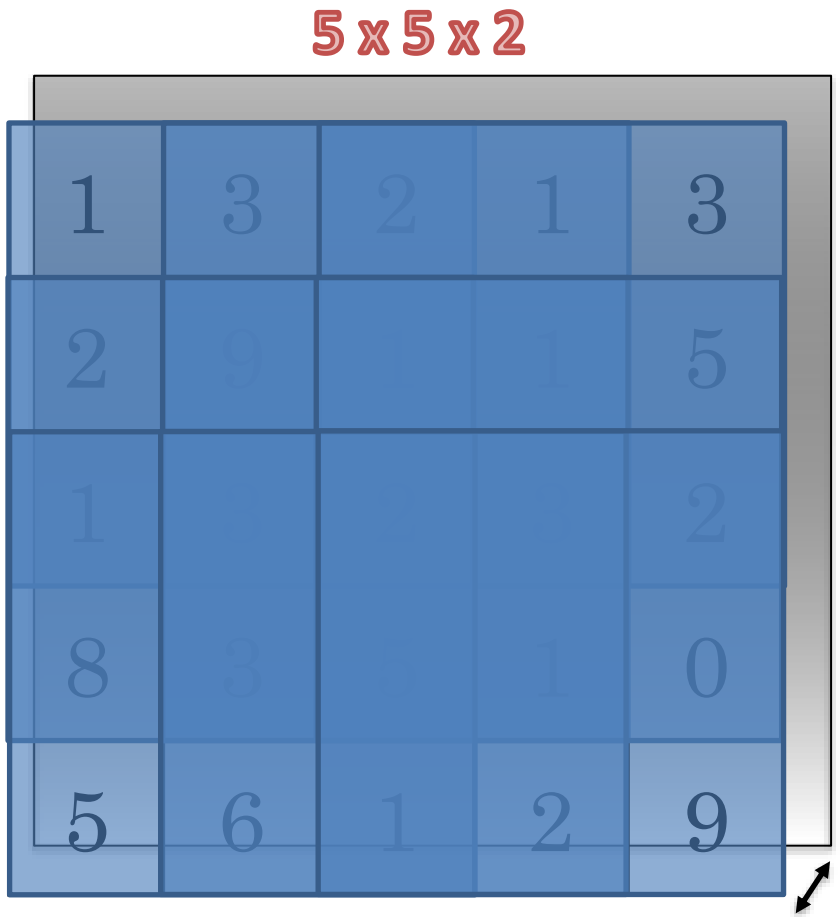
لایه‌ی pooling

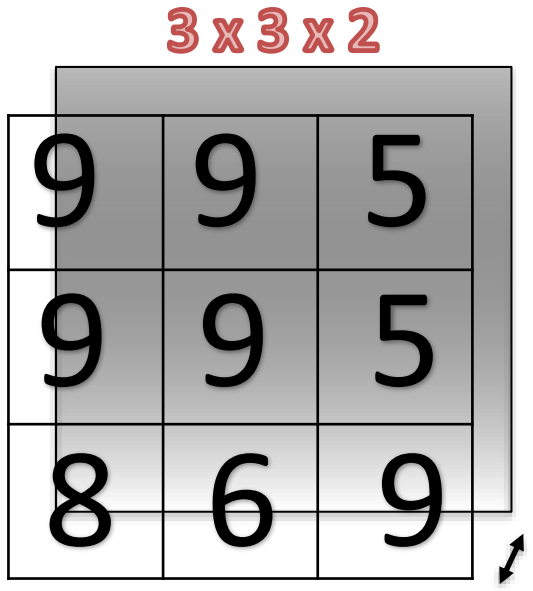




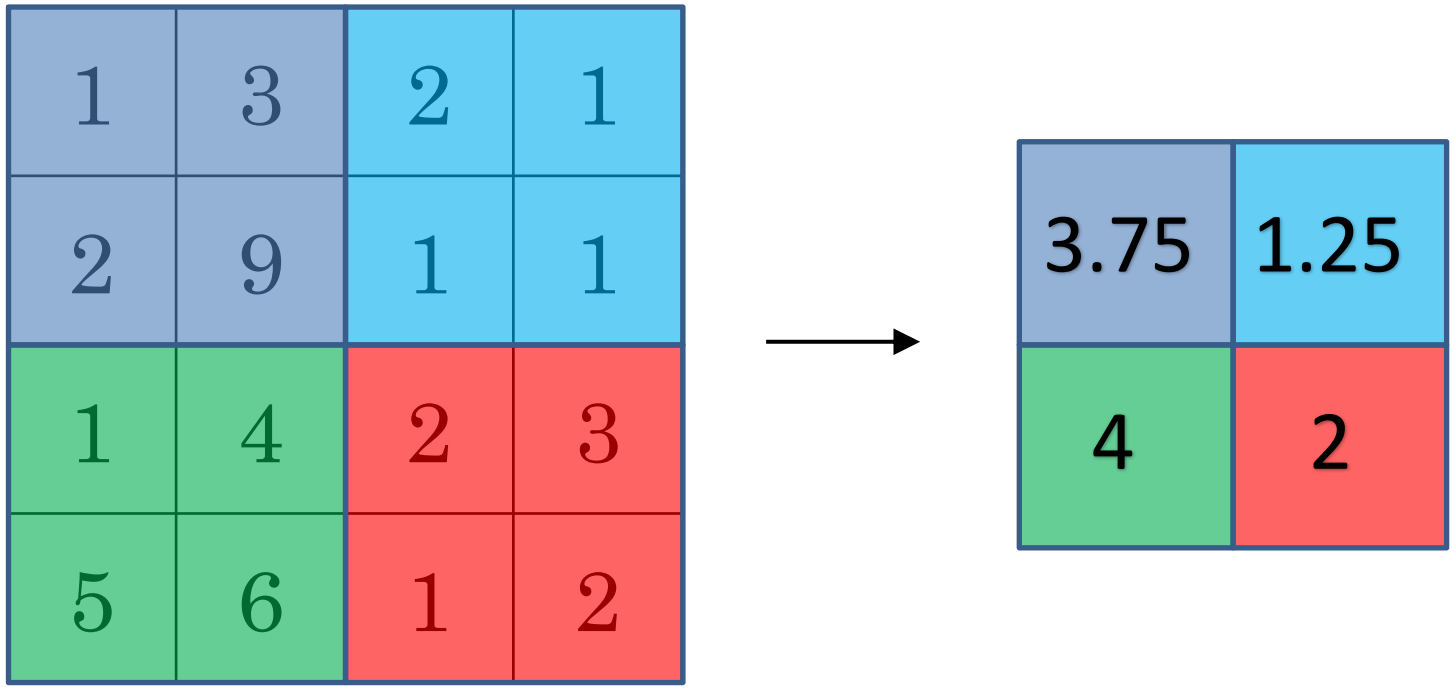
- ❑ دو hyper parameter دارد: s و f
- ❑ هیچ پارامتری برای آموزش ندارد

max pooling – pooling لایه‌ی





Average pooling



خلاصه pooling

Hyperparameters:

f : filter size

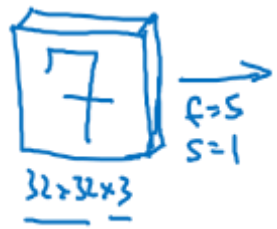
s : stride

Max or average pooling

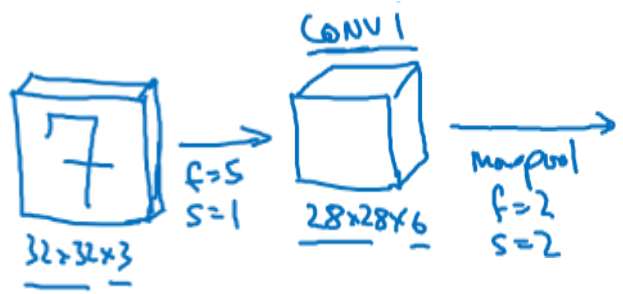
f = 2, s = 2
f = 3, s = 2

مثالی از یک شبکه کانولوشنالی (LeNet-5)

#of filters: 6

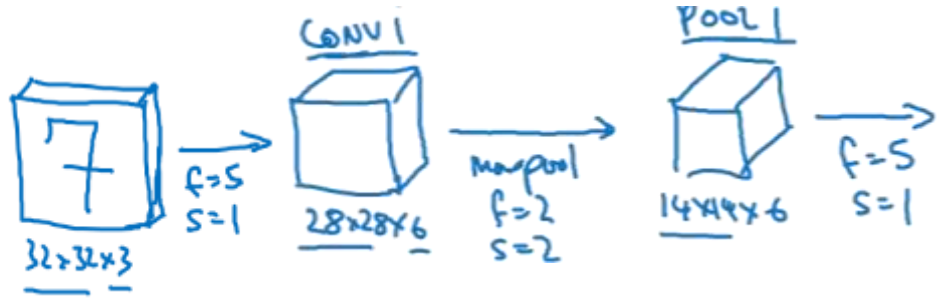


مثالی از یک شبکه کانولوشنالی (LeNet-5)

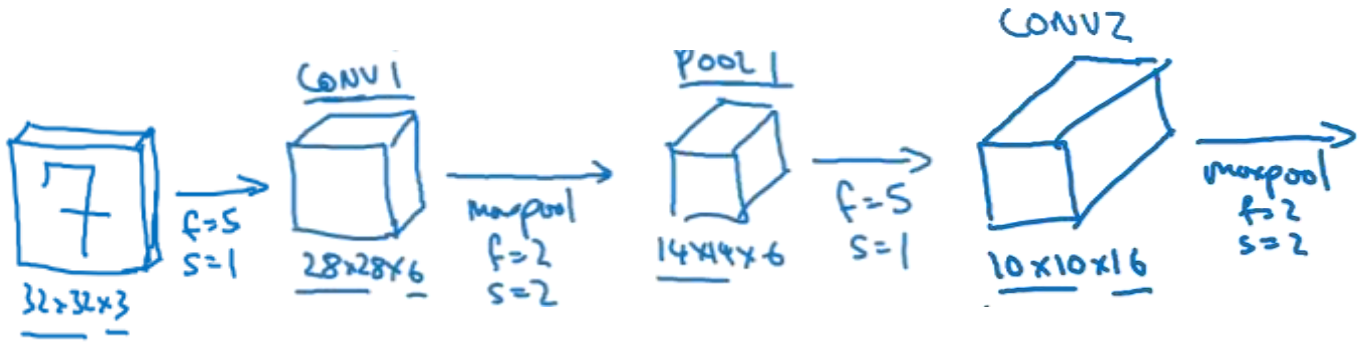


مثالی از یک شبکه کانولوشنالی (LeNet-5)

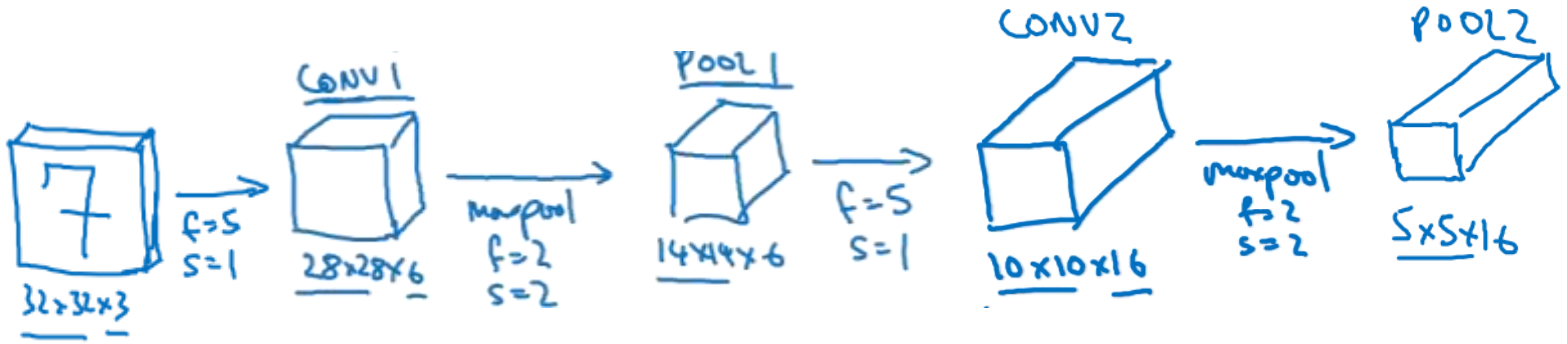
#of filters: 16



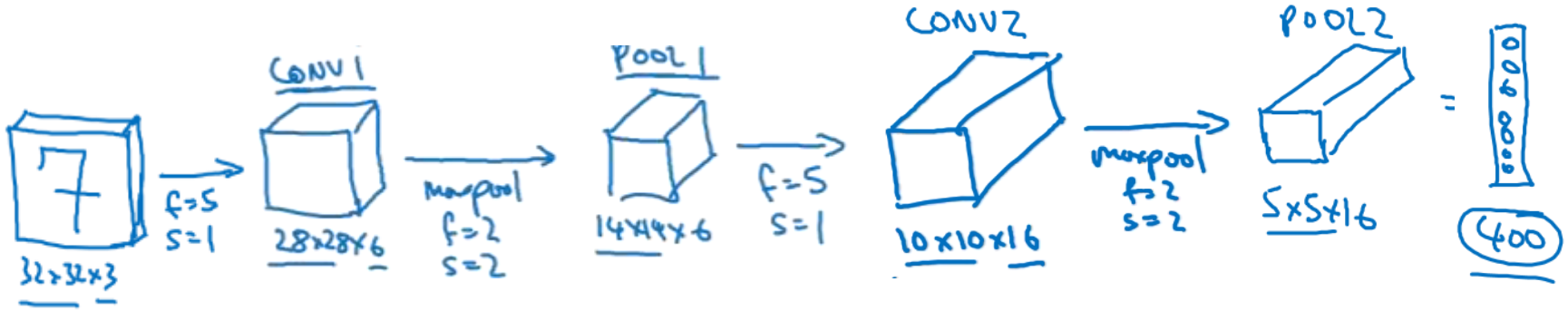
مثالی از یک شبکه کانولوشنالی (LeNet-5)



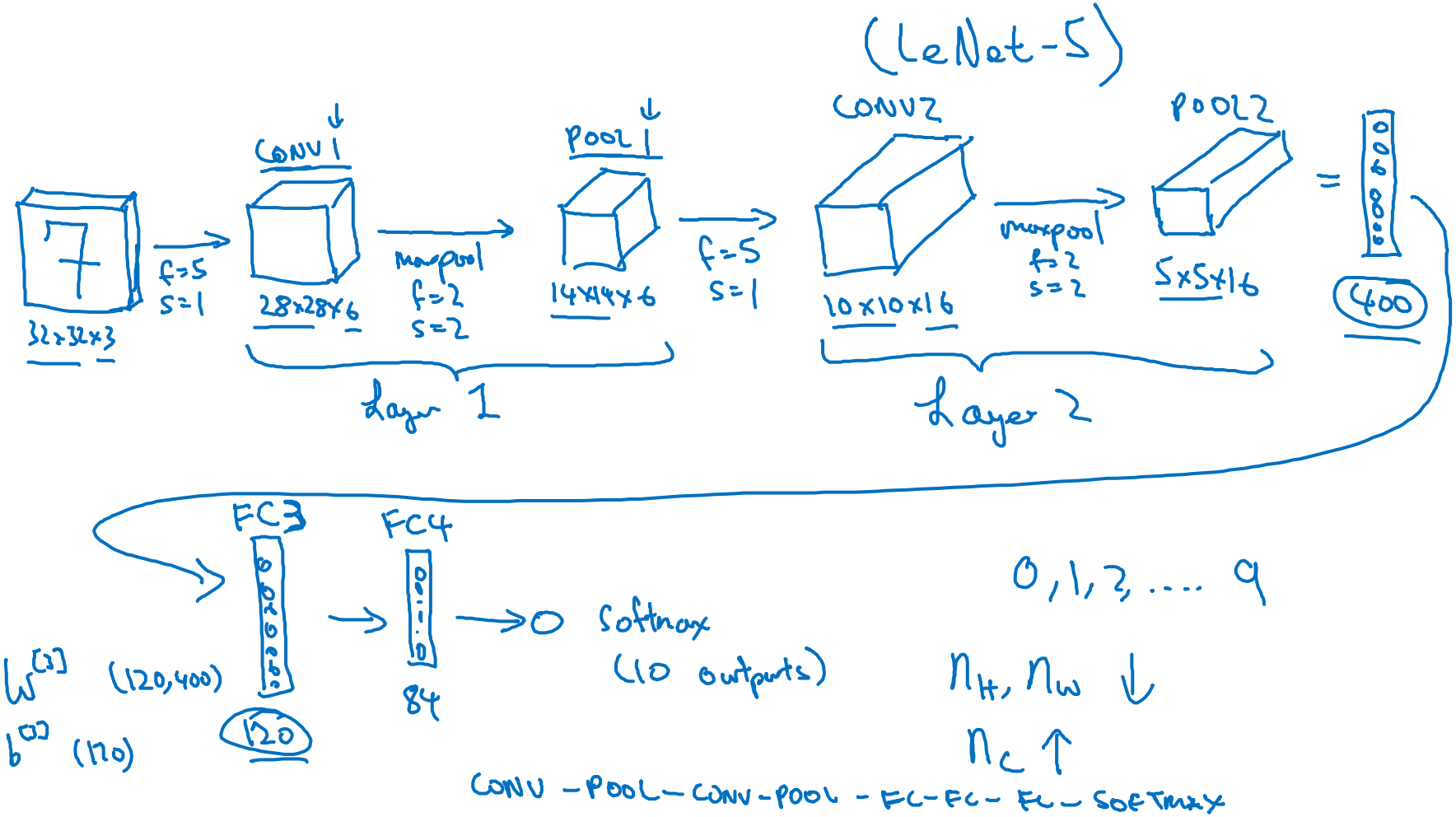
مثالی از یک شبکه کانولوشنالی (LeNet-5)



مثالی از یک شبکه کانولوشنالی (LeNet-5)



مثالی از یک شبکه کانولوشنالی (LeNet-5)



سایز هر لایه و تعداد پارامترها

#of filters: 8

	Activation shape	Activation Size	# parameters*
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)			
POOL1			
CONV2 (f=5, s=1)			
POOL2			
FC3			
FC4			
Softmax			

سایز هر لایه و تعداد پارامترها

#of filters: 16 #of neurons: 120

	Activation shape	Activation Size	# parameters*
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	608
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)			
POOL2			
FC3			
FC4			
Softmax			

```
model = Sequential()
model.add(layers.Conv2D(8, (5, 5), activation='relu',
                        input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(16, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(120, activation='relu'))
model.add(Dropout(0.5))
model.add(layers.Dense(84, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 28, 28, 8)         608
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 14, 14, 8)         0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 10, 10, 16)        3216
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 5, 5, 16)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_12 (Dense)             (None, 120)               48120
_________________________________________________________________
dropout_6 (Dropout)          (None, 120)               0
_________________________________________________________________
dense_13 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_14 (Dense)             (None, 10)                850
=================================================================
Total params: 62,958
Trainable params: 62,958
Non-trainable params: 0
_________________________________________________________________
```

شبکه‌های کانولوشنالی در کراس – ارقام فارسی



06_ConvolutionalNeuralNetwork-Hoda-Keras.ipynb

منابع

- http://cs231n.stanford.edu/
- https://www.csee.umbc.edu/courses/671/fall01/class-notes/k-nn.ppt
- https://docs.google.com/presentation/d/1kytN5gqrZe0Bcu3hExjZpA4PQipaFEKn26S-JhqncXk/edit#slide=id.g1ff36e20db_0_450
- https://www.slideshare.net/Alirezaakhavanpour/tensorflow-71395844
- https://www.datacamp.com/courses/convolutional-neural-networks-for-image-processing