

## یادگیری ماشین

- یادگیری تقویتی
- یادگیری با ناظارت
- یادگیری بدون ناظارت
- یادگیری نیمه‌ناظارتی
- یادگیری خودناظارتی

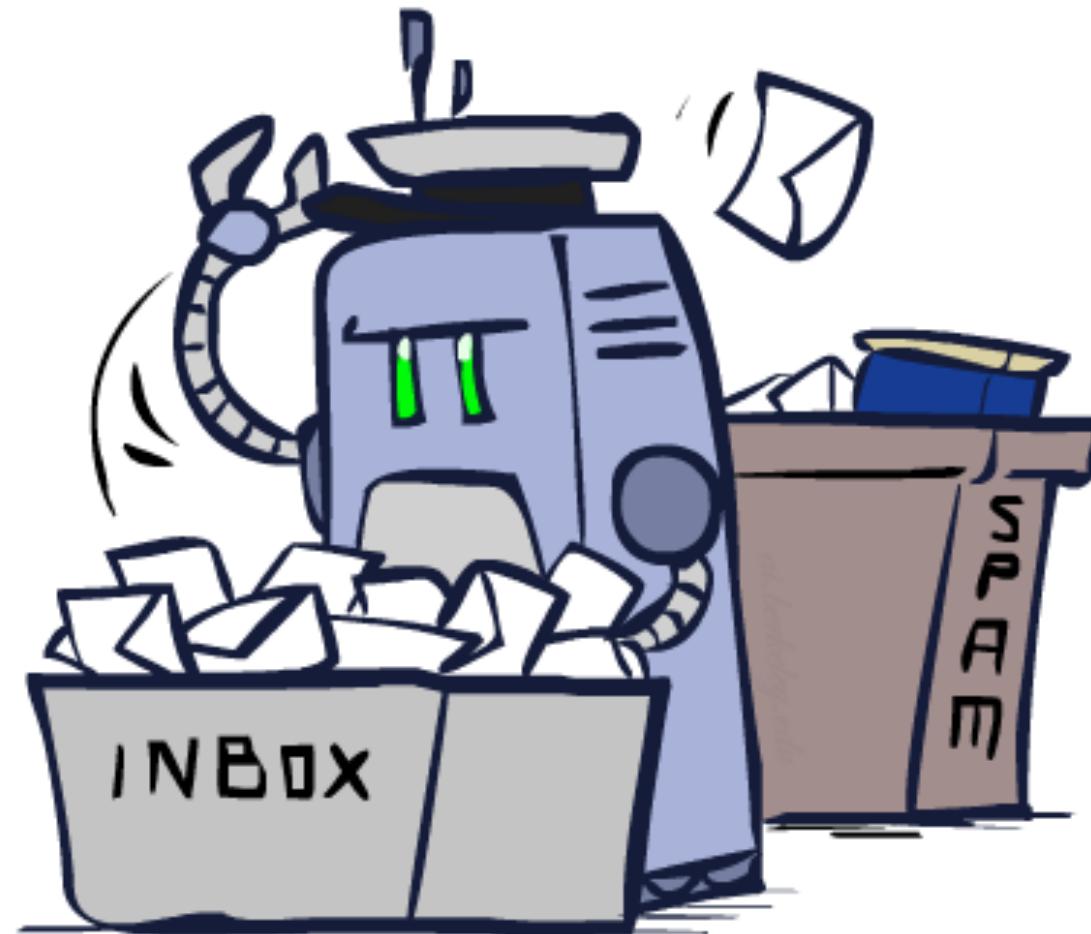
# یادگیری ماشین

## (Machine Learning)

- .1 یادگیری تقویتی (Reinforcement Learning)
- .2 یادگیری باناظارت (Supervised Learning)
- .3 یادگیری بدون ناظارت (Unsupervised Learning)
- .4 یادگیری نیمه‌ناظارتی (Semi-Supervised Learning)
- .5 یادگیری خودناظارتی (Self-Supervised Learning)

# یادگیری بانظارت (Supervised Learning)

دسته‌بندی



# مثال: تشخیص ایمیل تبلیغاتی (Spam)



جناب آقای..  
در ابتدا باید اعتماد شما را در این معامله جلب کنم، چرا که به دلیل ماهیت این موضوع، این مسئله کاملاً محترمانه و سری است....»

«برای حذف شدن از لیست ایمیل‌های آینده، فقط کافیست به این پیام پاسخ دهید و کلمه REMOVE را در عنوان (subject) قرار دهید.»

فقط با ۹۹ دلار (\$)!



باشه، می‌دونم این کاملاً خارج از موضوعه ولی دارم کم کم دیوونه می‌شم!  
یه سیستم قدیمی Dell Dimension XPS گوشه‌ی اتاق داشتم. گفتم بذار ازش استفاده کنم.  
می‌دونم قبل از اینکه بذارمش اونجا، کار می‌کرد.  
ولی الان که وصلش کردم به برق و دکمه‌ی پاور و زدم هیچی نشد!



□ ورودی: یک ایمیل

□ خروجی: ایمیل تبلیغاتی (spam) / عادی (ham)

□ آمده‌سازی (Setup):

- یک مجموعه بزرگ از ایمیل‌ها جمع‌آوری می‌کنیم که هر کدام با برچسب "تبلیغاتی" یا "عادی" مشخص شده‌اند.
- توجه: کسی باید همه این داده‌ها را دستی برچسب‌گذاری کرده باشد!
- هدف این است که مدلی یاد گرفته شود که بتواند برچسب ایمیل‌های جدید و آینده را پیش‌بینی کند.

□ ویژگی‌ها (Features)

- ویژگی‌هایی که برای تصمیم‌گیری در مورد تبلیغاتی یا غیر تبلیغاتی بودن ایمیل استفاده می‌شوند:
  - کلمات: مثلاً وجود کلماتی مثل، رایگان (FREE)، محترمانه، حذف (REMOVE) و ..
  - الگوهای متنی: مثل وجود علامت \$ یا اعداد، یا نوشه‌هایی تماماً با حروف بزرگ
  - خصوصیات غیرمتنی:
    - فرستنده در لیست مخاطبین هست یا نه؟
    - ایمیل برای تعداد زیادی مخاطب ارسال شده یا نه؟

# مثال: تشخیص ارقام دستنویس (Digit Recognition)

0

1

2

1

?

■ ورودی: تصویر مربوط به ارقام دستنویس

■ خروجی: یک عدد بین ۰ تا ۹

■ آماده‌سازی (Setup):

- یک مجموعه بزرگ از تصاویر ارقام دستنویس جمع‌آوری می‌کنیم، که هر تصویر با عدد صحیح مربوط به خودش برچسب‌گذاری شده است.

- توجه: این داده‌ها باید توسط انسان‌ها به صورت دستی برچسب‌گذاری شوند!

- هدف این است که مدلی آموزش داده شود که بتواند برای ارقام دستنویس جدید، عدد صحیح را پیش‌بینی کند.

■ ویژگی‌ها (Features): ویژگی‌هایی که مدل برای تصمیم‌گیری درباره کلاس ارقام استفاده می‌کند:

- پیکسل‌ها: مثلاً پیکسل در موقعیت (6,8) سفید است یا سیاه (در یک تصویر باینری)

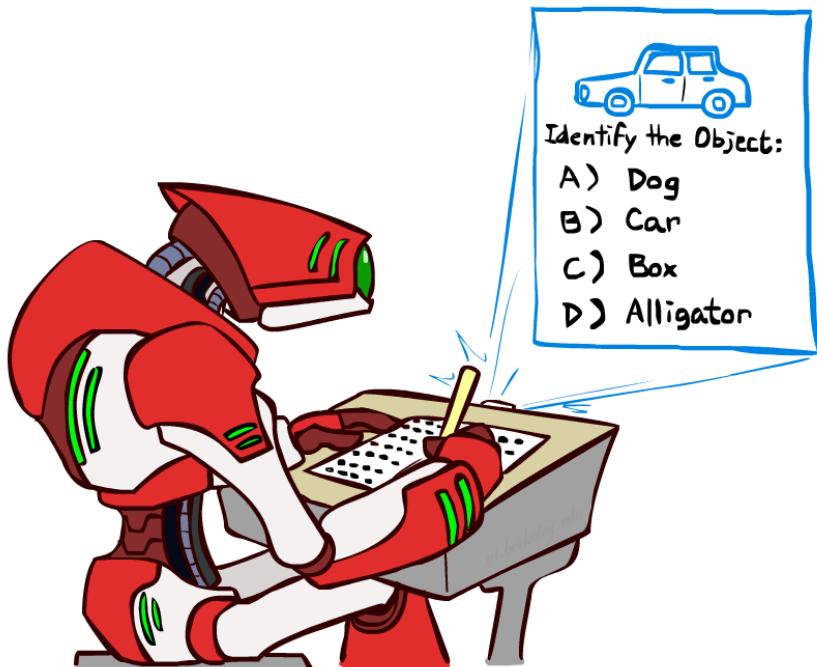
- الگوهای شکلی: تعداد اجزای متصل، نسبت طول به عرض، تعداد حلقه‌ها یا دایره‌ها

# چند مثال دیگر از دسته‌بندی

- دسته‌بندی: هدف پیش‌بینی برچسب‌ها (یعنی کلاس‌ها) برای ورودی‌های داده شده (x) است.
- دسته‌بندی دو تایی (Binary Classification): فقط دو کلاس وجود دارد.
- دسته‌بندی چند کلاسه (Multiclass Classification): بیش از دو کلاس وجود دارد.

## □ مثال‌ها:

- تشخیص پزشکی: ورودی: علائم بیمار، خروجی: بیماری‌های مختلف
- تشخیص کلاهبرداری: ورودی: یک تراکنش مالی، خروجی: کلاهبرداری / عدم کلاهبرداری
- نمره‌دهی خودکار مقاله: ورودی: یک سند متنی، خروجی: نمرات
- ارسال خودکار ایمیل‌های خدمات مشتریان به بخش مربوطه
- تحلیل احساسات در نظرات کاربران
- شناسایی زبان متن
- و بسیاری کاربردهای دیگر...



- دسته‌بندی یکی از فناوری‌های مهم و کاربردی در صنعت است!

# چالش‌ها در یادگیری با ناظارت

$$\text{داده} + \text{مدل} = \text{AI}$$

- در رویکرد سنتی یادگیری ماشین، تمرکز بر انتخاب و بهینه‌سازی مدل بوده است، نه کیفیت داده.

## 1. رویکرد مدل‌محور

- تمرکز اصلی بر بهبود مدل است.

## 2. رویکرد داده‌محور

- تمرکز اصلی بر بهبود کیفیت داده‌هاست.

# چالش‌ها در رویکرد داده‌محور

## 1. کمبود داده:

- کافی نبودن حجم داده‌های آموزشی

## 2. برچسب‌گذاری:

- برچسب‌گذاری نادرست: گاهی اوقات، برچسب‌های داده با خطای انسانی اشتباه ثبت می‌شوند
- هزینه بالا: نیاز به تخصص بالا (در حوزه‌هایی مانند پزشکی یا حقوق)،
- ابهام در تعریف برچسب‌ها: نبود مرز مشخص بین کلاس داده‌ها

## 3. داده‌های با کیفیت پایین:

- وجود نویز، اطلاعات ناقص یا داده‌های نادرست.

## 4. داده‌های نامرتبط یا ویژگی‌های بی‌اثر (Irrelevant Features):

- وجود ویژگی‌هایی که ارتباطی با خروجی هدف ندارند.

## 5. عدم توازن در داده‌های هر کلاس:

- تعداد نمونه‌های یک یا چند کلاس، به طور قابل توجهی کمتر از سایر کلاس‌های است. (منجر به سوگیری مدل)

# چالش‌ها در رویکرد مدل‌محور

## ▪ بیش‌برازش (Overfitting):

- بیش‌برازش زمانی رخ می‌دهد که مدل یادگیری ماشین، آنقدر خود را با داده‌های آموزشی تطبیق می‌دهد که حتی نویزها و جزئیات غیرضروری را نیز به عنوان الگو یاد می‌گیرد.
- عملکرد عالی روی داده‌ی آموزش، ولی افت شدید روی داده‌ی تست (=داده‌های جدید و دیده‌نشده)

## ▪ عامل اصلی ایجاد Overfitting:

- تناسب نداشتن پیچیدگی مدل با حجم داده
- مدل تعداد زیادی پارامتر دارد ولی تعداد داده آموزشی کم هست
  - مثال: شبکه عصبی ۱۰ لایه روی دیتاست ۲۰۰ نمونه‌ای.

# چالش‌ها در رویکرد مدل‌محور

## □ برازش ناکافی (Underfitting):

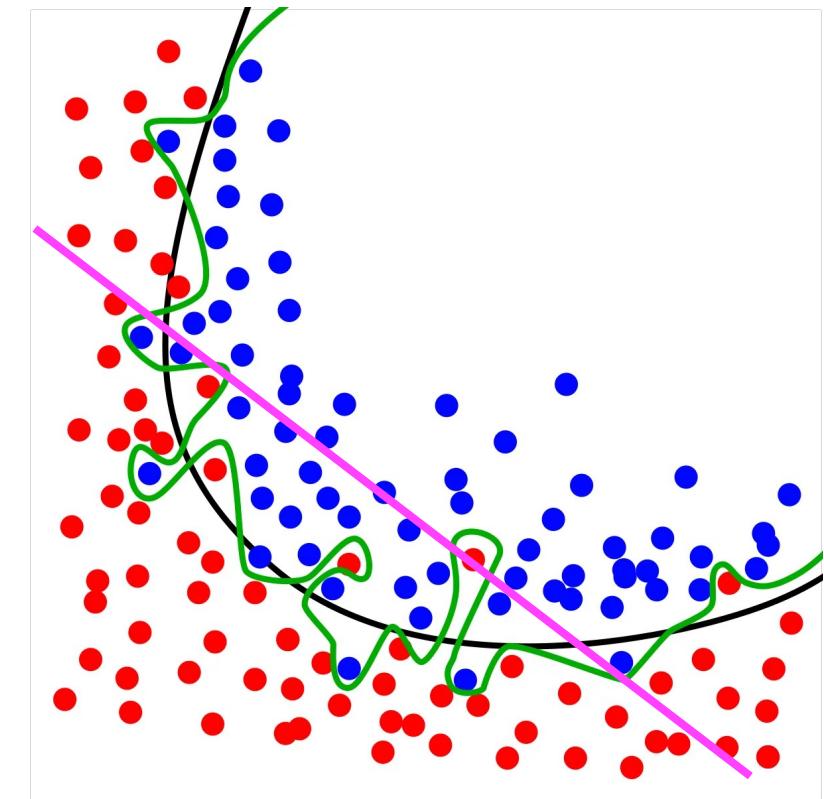
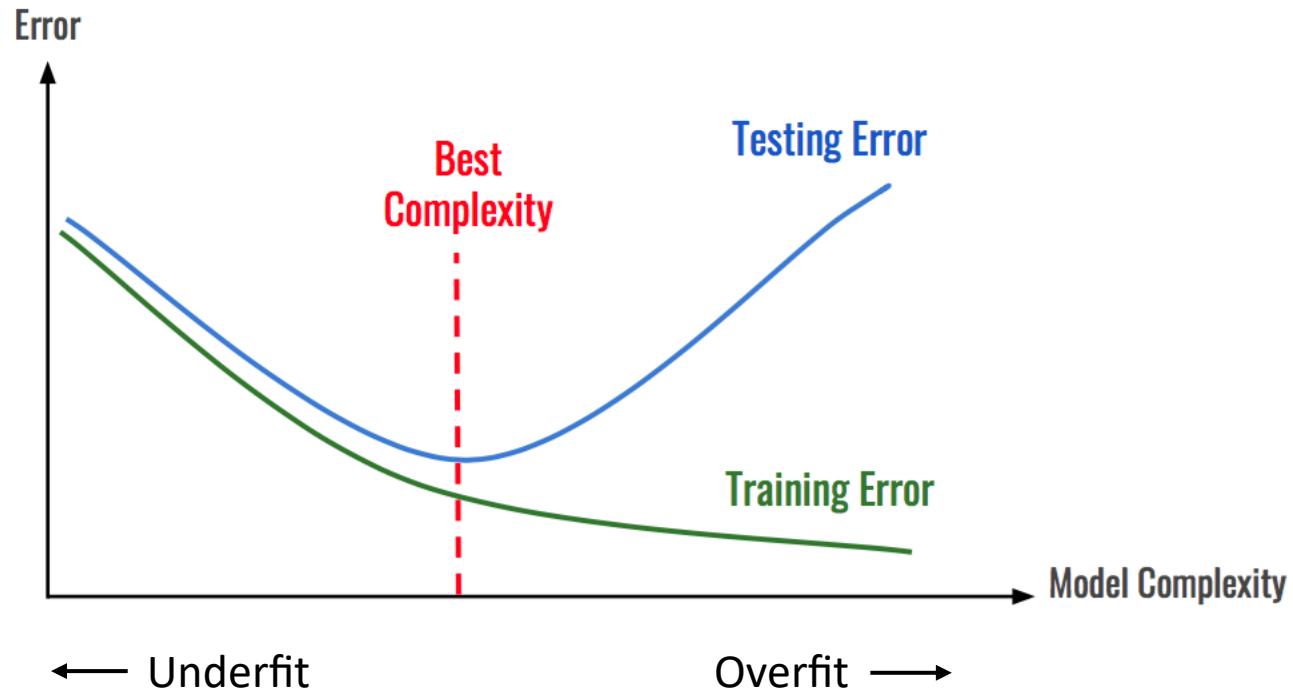
- برازش ناکافی زمانی رخ می‌دهد که مدل یادگیری ماشین، نتواند الگوهای اصلی موجود در داده‌های آموزشی را یاد بگیرد. در این حالت، مدل آنقدر ساده یا ضعیف است که حتی روابط مهم بین ویژگی‌ها و خروجی را نیز درک نمی‌کند.
- هم روی داده‌ی آموزش، و هم روی داده‌ی تست عملکرد ضعیفی دارد

## □ عامل اصلی ایجاد Underfitting :

- مدل بیش از حد ساده
- مدل قدرت کافی برای یادگیری الگوهای داده را ندارد
  - مثال: معادله‌ی خطی برای مسئله‌ی پیچیده (با رابطه‌ی غیرخطی)

# چالش‌ها در رویکرد مدل‌محور

## Overfitting & Underfitting:

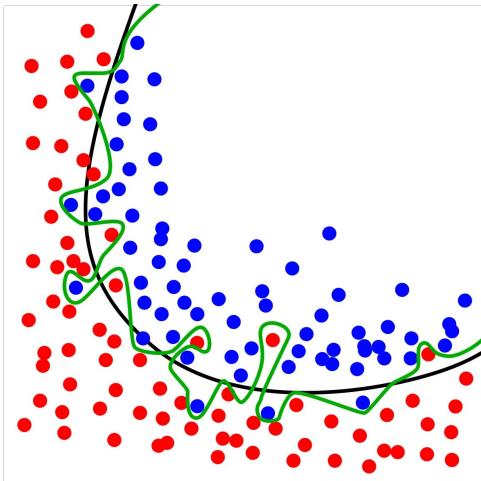


✓ برای غلبه بر چالش‌های داده محو و مدل‌محور باید به درستی مدل انتخاب شود و نحوه ارزیابی آن نیز دقیق باشد.

# انتخاب مدل و ارزیابی آن

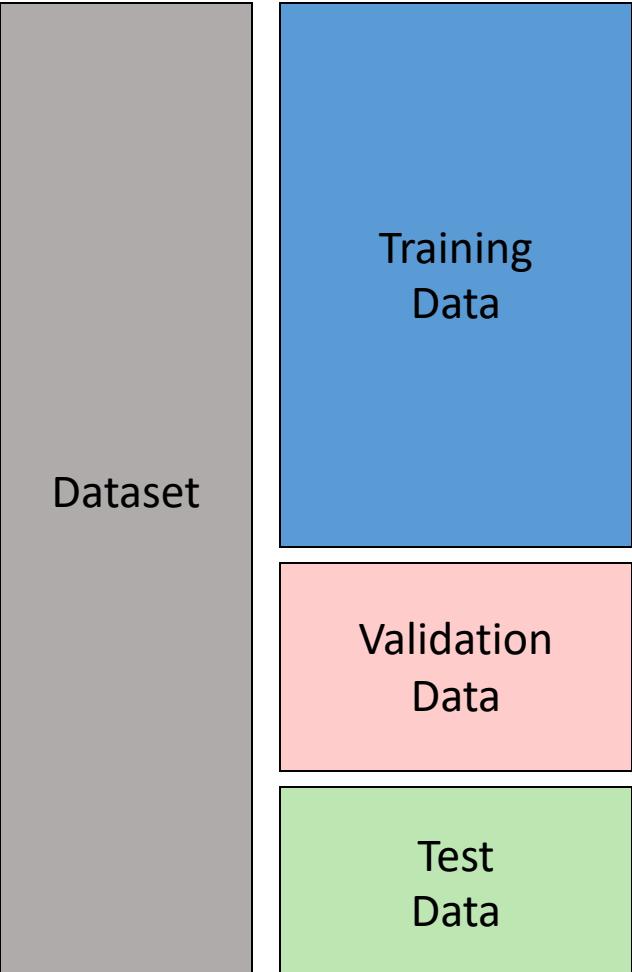
# انتخاب مدل و ارزیابی آن

- هدف، یافتن مدلی است که نه دچار overfitting شود و نه underfitting.
- انتخاب مدل یعنی برگزیدن یک الگوریتم یادگیری (مانند درخت تصمیم، شبکه عصبی مناسب و ...) که بتواند تعمیمدهی مناسبی داشته باشد و روی داده‌های جدید به درستی عمل کند.
- جهت انتخاب مدل، لازم است عملکرد آن بر اساس داده‌هایی که در آموزش استفاده نشده‌اند، ارزیابی شود.



- این ارزیابی معمولاً با تقسیم داده‌ها به سه بخش انجام می‌شود:
  - داده‌ی آموزش (training)
  - داده‌ی اعتبارسنجی (validation)
  - داده‌ی تست (test)

# انتخاب مدل و ارزیابی آن



## □ داده‌ی آموزش (training)

- برای آموزش مدل و بهینه‌سازی پارامترها استفاده می‌شود.

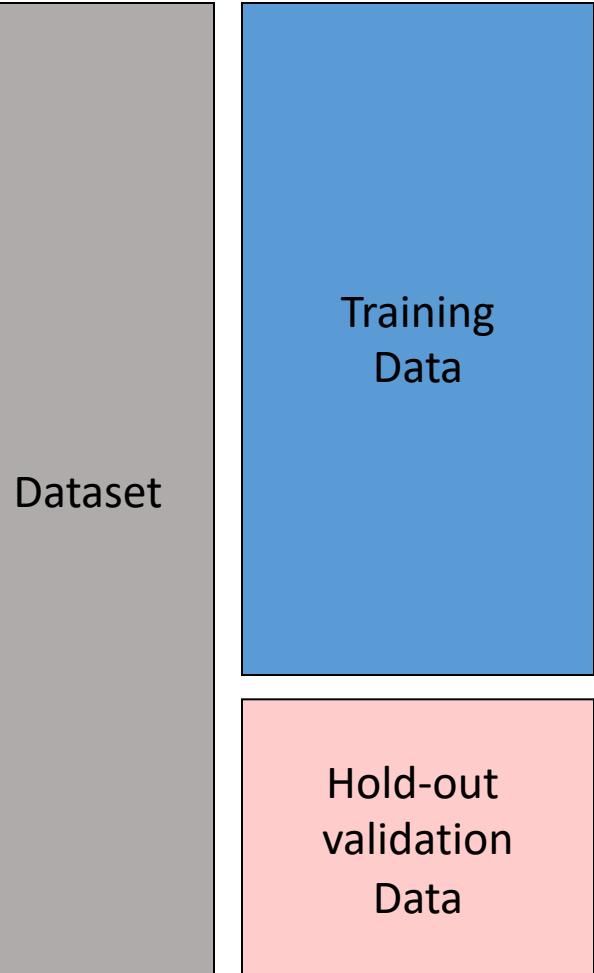
## □ داده‌ی اعتبارسنجی (validation)

- برای مقایسه مدل‌ها و تنظیم ابرپارامترها در حین آموزش به کار می‌رود.

## □ داده‌ی تست (test)

- برای سنجش نهایی عملکرد مدل روی داده‌های دیده‌نشده و تخمین قدرت تعمیم آن استفاده می‌شود.

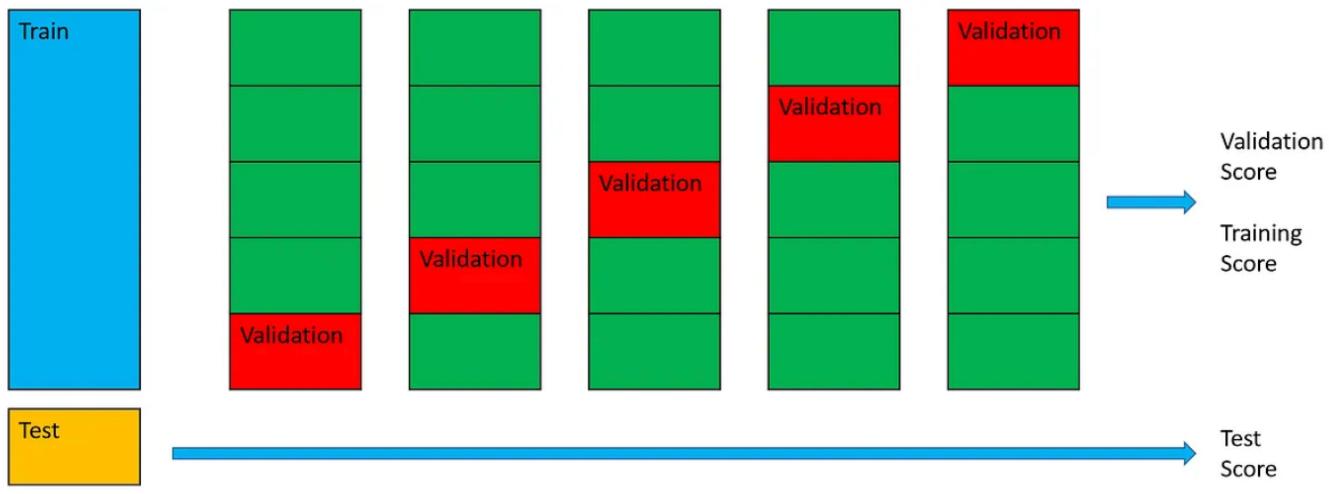
# اعتبارسنجی به روشن Hold-out



■ در این روش، کل داده‌های برچسبدار موجود به دو بخش تقسیم می‌شوند:

1. داده‌ی آموزش (training)■ این بخش از داده‌ها برای آموزش مدل استفاده می‌شود.  
■ یعنی مدل با استفاده از این داده‌ها، پارامترهای داخلی خودش (مثل وزن‌ها یا ضرایب) را یاد می‌گیرد.
2. داده اعتبارسنجی جداشده (Hold-out)■ این بخش از داده‌ها در طول آموزش به مدل نشان داده نمی‌شود.  
■ بعد از آموزش، از این داده‌ها برای ارزیابی عملکرد مدل و تنظیم ابرپارامترها (مثل تعداد لایه‌ها، نرخ یادگیری و ...) استفاده می‌شود.

# اعتبارسنجی به روشن



□ در این روش، کل داده‌ها به  $k$  بخش مساوی تقسیم می‌شوند.

□ سپس مدل  $k$  بار آموزش داده می‌شود؛ هر بار:

- یکی از این  $k$  بخش‌ها به عنوان مجموعه‌ی اعتبارسنجی (Validation) در نظر گرفته می‌شود

- و بقیه‌ی بخش‌ها ( $1-k$  تای دیگر) برای آموزش مدل (Training) استفاده می‌شوند.

□ در پایان، میانگین عملکرد مدل در  $k$  بار آزمایش به عنوان ارزیابی نهایی در نظر گرفته می‌شود.

✓ مناسب مجموعه داده‌های کوچک

# عدم تطابق داده‌ها (Data Mismatch)

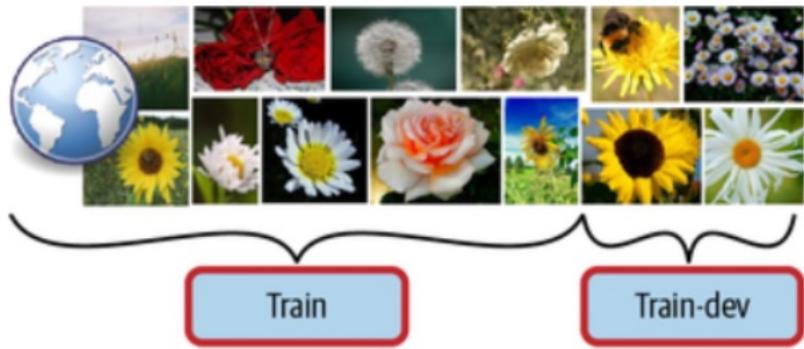
❑ مشکل: داده‌های آموزشی با داده‌های واقعی (در کاربرد نهایی) متفاوت هستند.  
► مثال:

- فرض کنید می‌خواهید یک اپلیکیشن شناسایی گل بسازید.
- میلیون‌ها تصویر گل را از وب دانلود می‌کنید و برای آموزش مدل استفاده می‌کنید.
- اما کاربر قرار است از برنامه روی گوشی استفاده کند و عکس گل‌ها را خودش بگیرد.
- تصاویر وب و تصاویر واقعی کاربر، کیفیت و شرایط نوری و زاویه دوربین کاملاً متفاوتی دارند.



✓ در این شرایط، مهم‌ترین قانون این است که: مجموعه‌ی داده اعتبارسنجی و داده تست باید تا حد امکان نماینده‌ی داده‌های واقعی محیط محصول باشند.

# عدم تطابق داده‌ها (Data Mismatch)



- اگر دقت مدل روی داده اعتبارسنجی dev واقعی پایین باشد
  - overfitting رخ داده است
  - داده‌های اعتبارسنجی dev با داده‌های آموزشی تطابق ندارند.



- اگر دقت مدل روی داده اعتبارسنجی train-dev پایین باشد
  - overfitting رخ داده است

- اگر دقت مدل روی داده اعتبارسنجی train-dev خوب باشد ولی روی داده اعتبارسنجی dev واقعی پایین باشد:
  - داده‌های اعتبارسنجی dev با داده‌های آموزشی تطابق ندارند.
  - در این صورت باید مشکل Data Mismatch را حل کنیم (مثلاً داده‌های آموزشی واقعی‌تر جمع کنیم).

# عملکرد مدل و ارزیابی آن

# MNIST مجموعه داده

□ این مجموعه شامل تصاویر ۲۸×۲۸ پیکسلی از ارقام دستنویس ۰ تا ۹ است که از روی داده‌های خام موسسه NIST استخراج، نرمال‌سازی و به شکل مناسب برای آموزش مدل‌های یادگیری ماشین درآمده‌اند.

0  
1  
2  
3  
4  
5  
6  
7  
8  
9 9

□ این مجموعه شامل:

۶۰۰۰۰ تصویر برای آموزش

۱۰۰۰۰ تصویر برای تست

□ به عنوان مسئله‌ی دسته‌بندی ۱۰ کلاسه شناخته می‌شود.

□ به عنوان معیار پایه‌ای در بسیاری از الگوریتم‌های یادگیری بانظارت و شبکه‌های عصبی استفاده می‌شود.

## دسته بندی دو کلاسه

- ❑ فرض کنید هدف تشخیص تصاویر مربوط به عدد ۵ در مجموعه داده‌ی MNIST است
  - کلاس مثبت: تصویر عدد ۵
  - کلاس منفی: تصاویر سایر ارقام (۰ تا ۹ به جز ۵)
- ❑ مدل باید تصمیم بگیرد که آیا یک تصویر متعلق به عدد ۵ هست یا نه.

# عملکرد مدل و ارزیابی آن

## □ دقت: ارزیابی عملکرد مدل با معیار دقت

- دقت برابر است با نسبت تعداد پیش‌بینی‌های درست به کل پیش‌بینی‌ها.

## □ دقت در مسائلی که توزیع نمونه‌ها بین کلاس‌ها نامتوازن است (Class Imbalance) معیار مناسبی نیست.

## □ به همین دلیل، از معیارهای دیگری مانند:

Precision ▪

Recall ▪

F1 Score ▪

- در ماتریس درهم‌ریختگی استفاده می‌شود.

# ماتریس درهم ریختگی (Confusion Matrix)

		Predicted Class		Sensitivity $\frac{TP}{(TP + FN)}$
		Positive	Negative	
Actual Class	Positive	True Positive (TP) <b>Type II Error</b>	False Negative (FN)	Specificity $\frac{TN}{(TN + FP)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	

		Predicted	
		Positive	Negative
Actual	Positive	5 5 5	5 5
	Negative	6	8 3 9

□ تحلیل عملکرد مدل با استفاده از ماتریس درهم ریختگی

□ دسته‌بندی دوکلاسه: هدف شناسایی تصاویر مربوط به عدد ۵ باشد.

- کلاس مثبت (Positive): تصویر مربوط به عدد ۵
- کلاس منفی (Negative): تصاویر سایر اعداد (به جز ۵)

# معیارهای ارزیابی

$$\text{precision} = \frac{TP}{TP + FP}$$

**Precision** (صحت): دقت پیش‌بینی‌های مثبت

**Sensitivity** یا **Recall** (بازیابی): درصد نمونه‌های مثبتی که به درستی دسته‌بندی شده است.

$$\text{recall} = \frac{TP}{TP + FN}$$

✓ از آن‌جا که معیارهای precision و recall هر یک تنها بخشی از عملکرد مدل را نشان می‌دهند، می‌توان با ترکیب آن‌ها در قالب F1 Score یک شاخص عددی واحد به دست آورد که امکان مقایسه‌ی جامع‌تری میان مدل‌ها را فراهم می‌کند.

**F1 Score**: ترکیب Precision و Recall

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

# Precision/Recall Trade-off

- هرچه نمونه‌ای به سمت راست نزدیک‌تر باشد، مدل اطمینان بیشتری دارد که آن نمونه به کلاس مثبت تعلق دارد (در این مثال: عدد ۵).
- با انتخاب آستانه‌های مختلف برای برش بین پیش‌بینی مثبت و منفی، مقدارهای متفاوتی از precision و recall به دست می‌آید:
  - آستانه‌ی پایین‌تر → Precision بالا ولی Recall پایین‌تر
  - آستانه‌ی بالاتر → Precision پایین‌تر ولی Recall بالا
- بین این دو معیار همیشه باید توازن برقرار کرد.

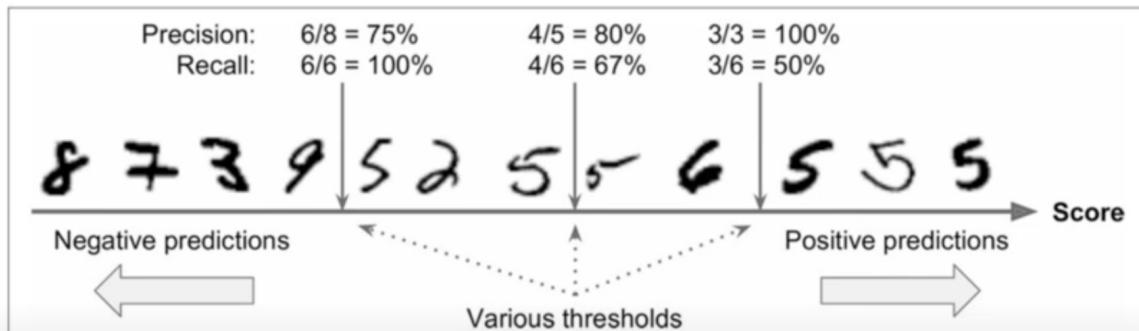
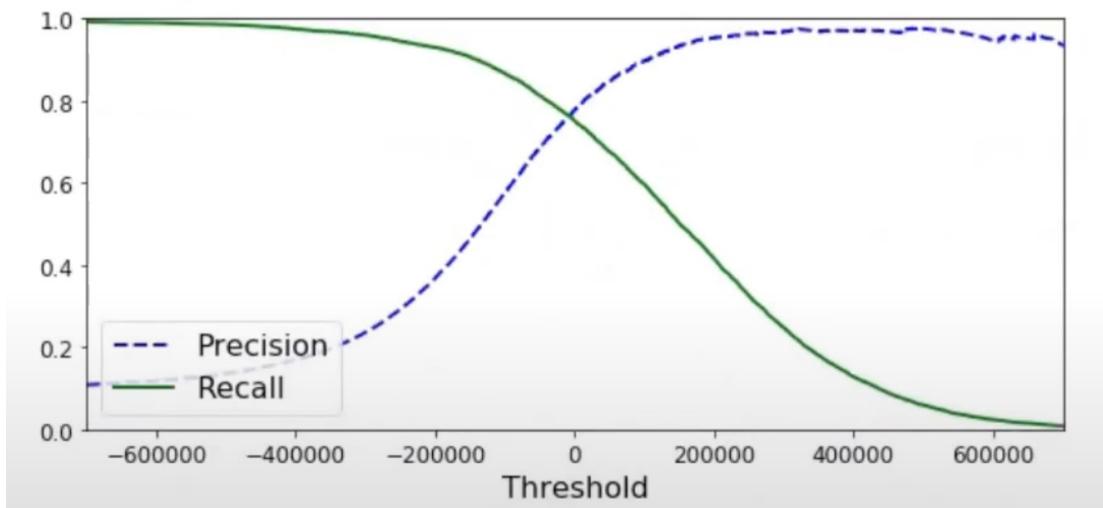


Figure 3-3. In this precision/recall trade-off, images are ranked by their classifier score, and those above the chosen decision threshold are considered positive; the higher the threshold, the lower the recall, but (in general) the higher the precision

# Precision/Recall Trade-off

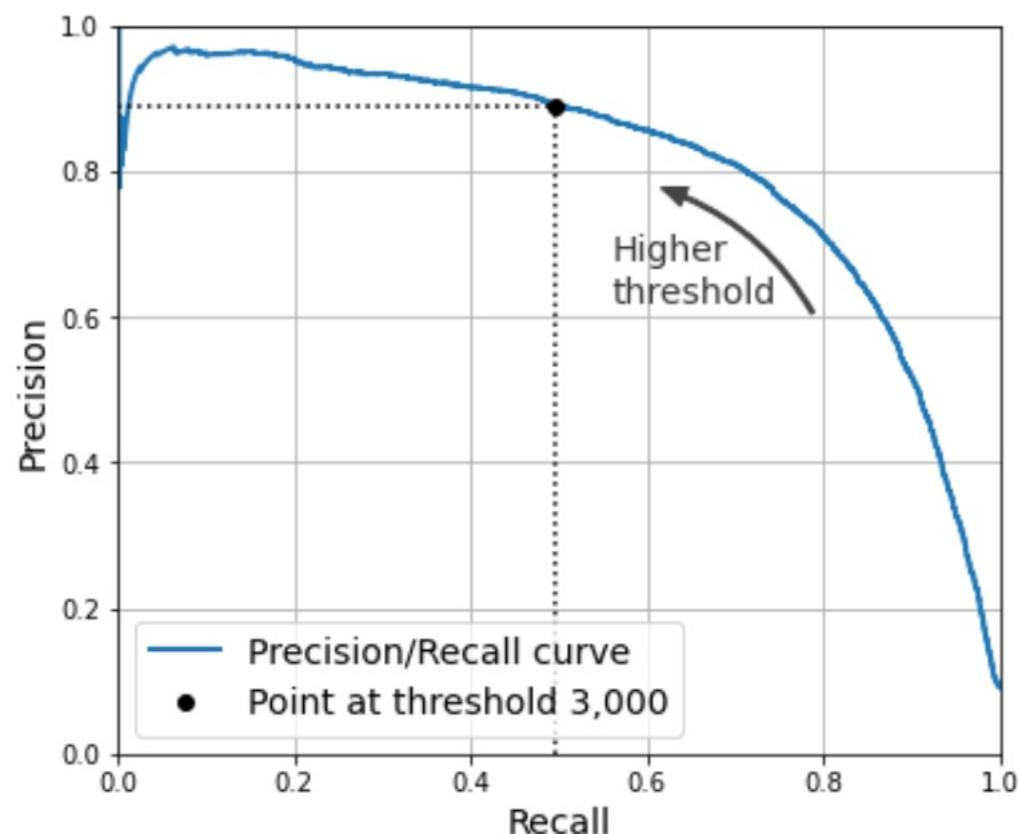


□ نمودار رابطه‌ی معکوس میان precision و recall را با تغییر آستانه نمایش می‌دهد.

□ مقدارهای precision و recall را بر حسب تغییر مقدار threshold ترسیم شده‌اند.

- هرچه آستانه افزایش می‌یابد، مدل سخت‌گیرتر می‌شود:
  - Precision افزایش می‌یابد، زیرا فقط نمونه‌هایی را مثبت در نظر می‌گیرد که امتیاز بالایی دارند.
  - Recall کاهش می‌یابد، چون برخی نمونه‌های مثبت واقعی دیگر از آستانه عبور نمی‌کنند.

# منحنی Precision-Recall



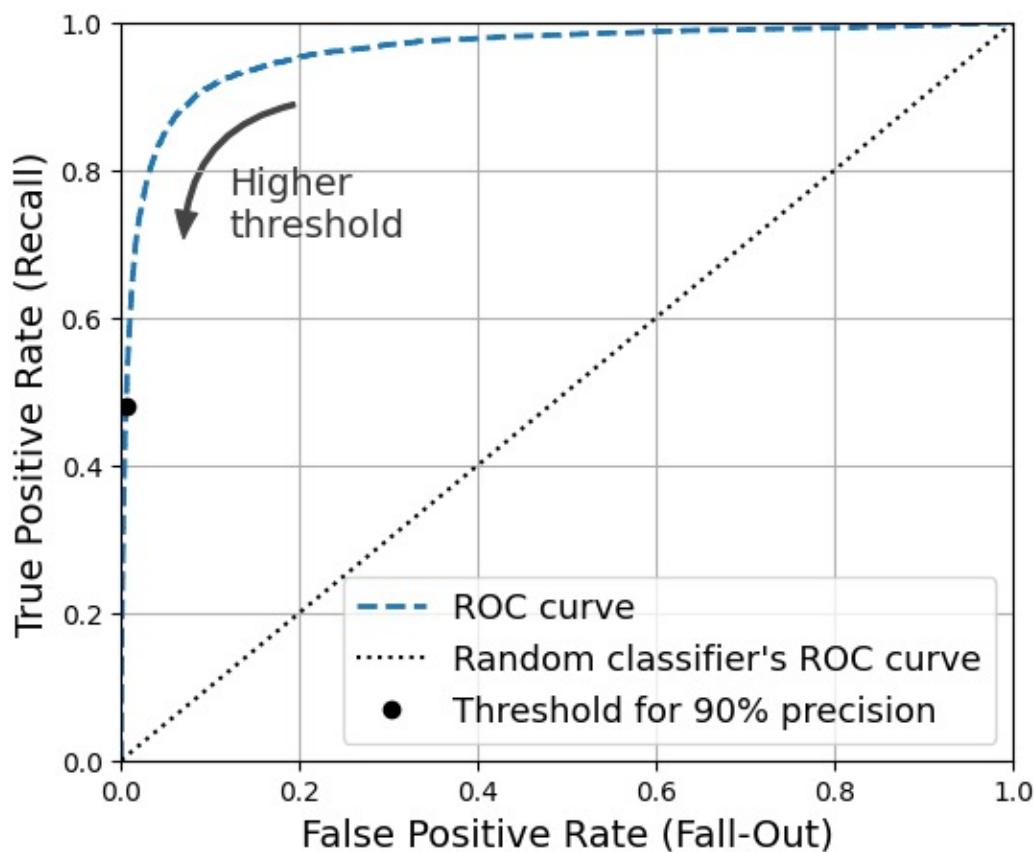
□ محور افقی recall و محور عمودی precision است.

□ هر نقطه روی منحنی نشان‌دهنده‌ی عملکرد مدل برای یک مقدار مشخص از threshold است.

- بخش‌های بالای منحنی نشان‌دهنده‌ی threshold هایی هستند که recall و precision هر دو بالا هستند.

- این نمودار مخصوصاً زمانی مفید است که کلاس مثبت کمیاب (rare) باشد، چون نسبت به تغییرات در داده‌های نامتوازن حساس‌تر از منحنی ROC است.

# منحنی ROC



□ منحنی ROC عملکرد مدل را بر حسب نرخ مثبت واقعی (True Positive Rate = Recall) و نرخ مثبت کاذب (False Positive Rate) نشان می‌دهد.

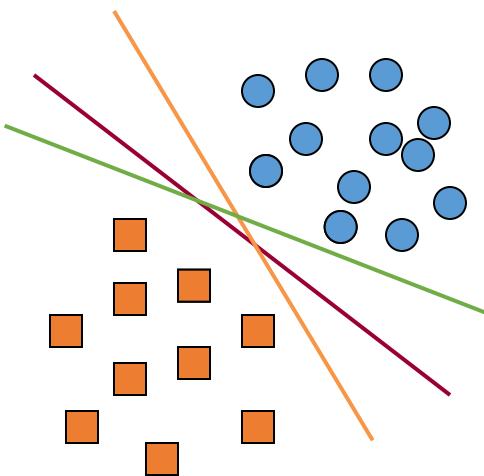
- محور افقی: نرخ مثبت کاذب (FPR)
- محور عمودی: نرخ مثبت واقعی (TPR = Recall)

□ هرچه منحنی به گوشی بالا-چپ نزدیک‌تر باشد، مدل عملکرد بهتری دارد.

□ خط قطری مدل تصادفی را نشان می‌دهد و مدل خوب باید بالای این خط قرار بگیرد.

# مدل‌های رایج در دسته‌بندی

# دسته‌بندهای خطی



▪ بسیاری از مدل‌های یادگیری با ناظارت، تلاش می‌کنند مرزی بین کلاس‌ها تعریف کنند که بتواند آن‌ها را به درستی از یکدیگر تفکیک کنند و دسته‌بندی کنند

▪ در مسئله‌ی دسته‌بندی، هدف این است که بر اساس نمونه‌های ورودی، تصمیم بگیریم که نمونه‌ی جدید به کدام کلاس تعلق دارد.

▪ برای این کار، یک تابع تصمیم (Decision Function) (یک مدل) ساخته می‌شود که مرز جداکننده‌ای بین کلاس‌ها ایجاد می‌کند.

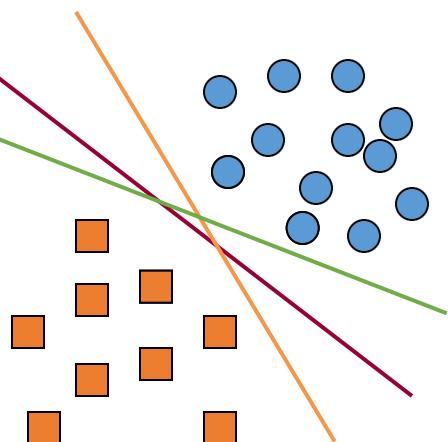
# تابع تصمیم

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{بردار ویژگی‌ها}$$

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \quad \text{بردار وزن‌ها}$$

- در دسته‌بندی، به دنبال یافتن یک مرز تصمیم هستیم که کلاس‌ها را از هم جدا کند.
- این مرز معمولاً با یک تابع به فرم زیر نشان داده می‌شود:

$$d(x) = \mathbf{w}^T \mathbf{x} + w_0$$



- اگر  $d(\mathbf{x}) > 0$  باشد، نمونه به کلاس مثبت تعلق دارد.
- اگر  $d(\mathbf{x}) < 0$  باشد، نمونه به کلاس منفی تعلق دارد.

- اگر فضای ویژگی یک بعدی ( $x_1$ ) باشد
- اگر فضای ویژگی دو بعدی باشد ( $x_1$  و  $x_2$ ) آنگاه مرز تصمیم یک خط، است خواهد بود:

$$d(x) = w_1 x_1 + w_0$$

$$d(x) = w_1 x_1 + w_2 x_2 + w_0$$

- برای بعدهای بالاتر، همین مفهوم تعمیم پیدا می‌کند:

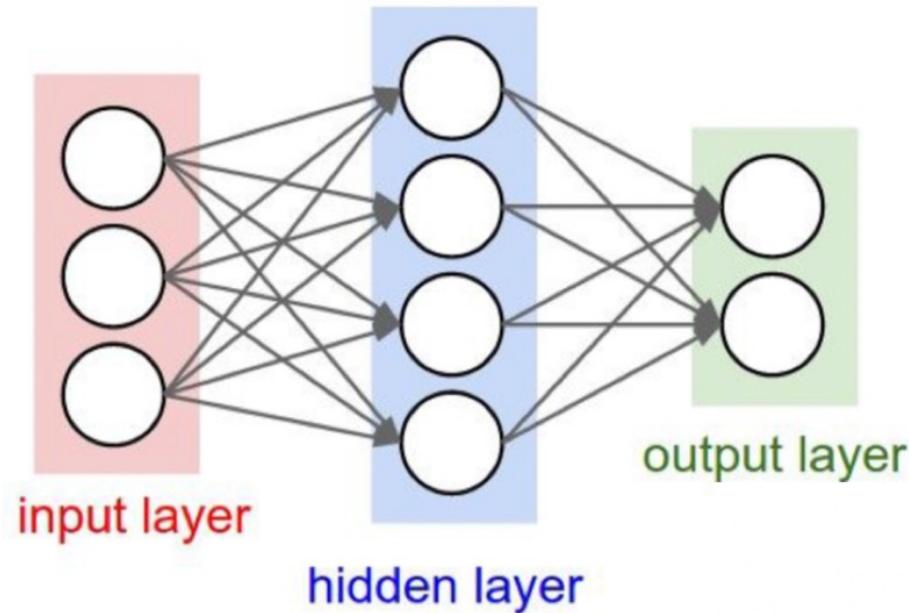
$$d(x) = \sum_{i=1}^n w_i x_i + w_0 = \mathbf{w}^T \mathbf{x} + w_0$$

- هدف پیدا کردن  $\mathbf{w}$  است. چون این پارامترها جهت شب و... خط جداکننده را تعیین می‌کند.

# شبکه‌های عصبی

Neural Networks

# شبکه‌های عصبی



► یک شبکه عصبی مصنوعی از واحدهای محاسباتی به نام نورون‌ها یا عصب‌ها تشکیل شده است.

► نورون‌ها به صورت لایه‌ها در شبکه قرار می‌گیرند.

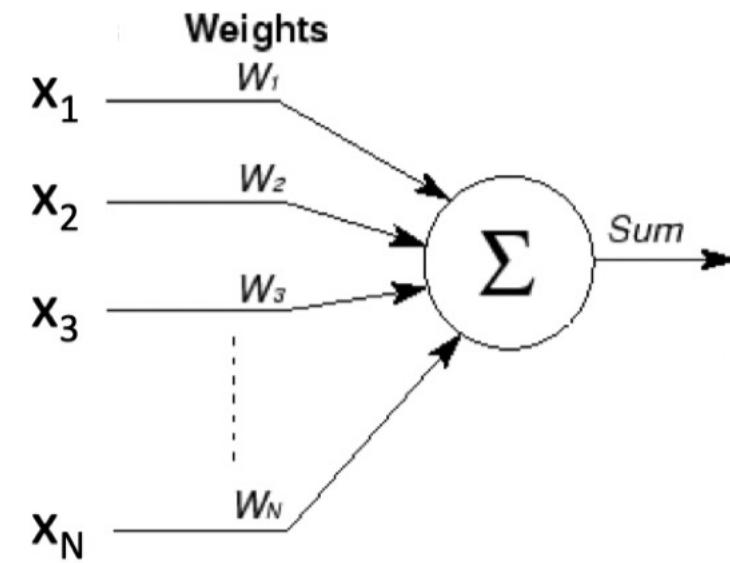
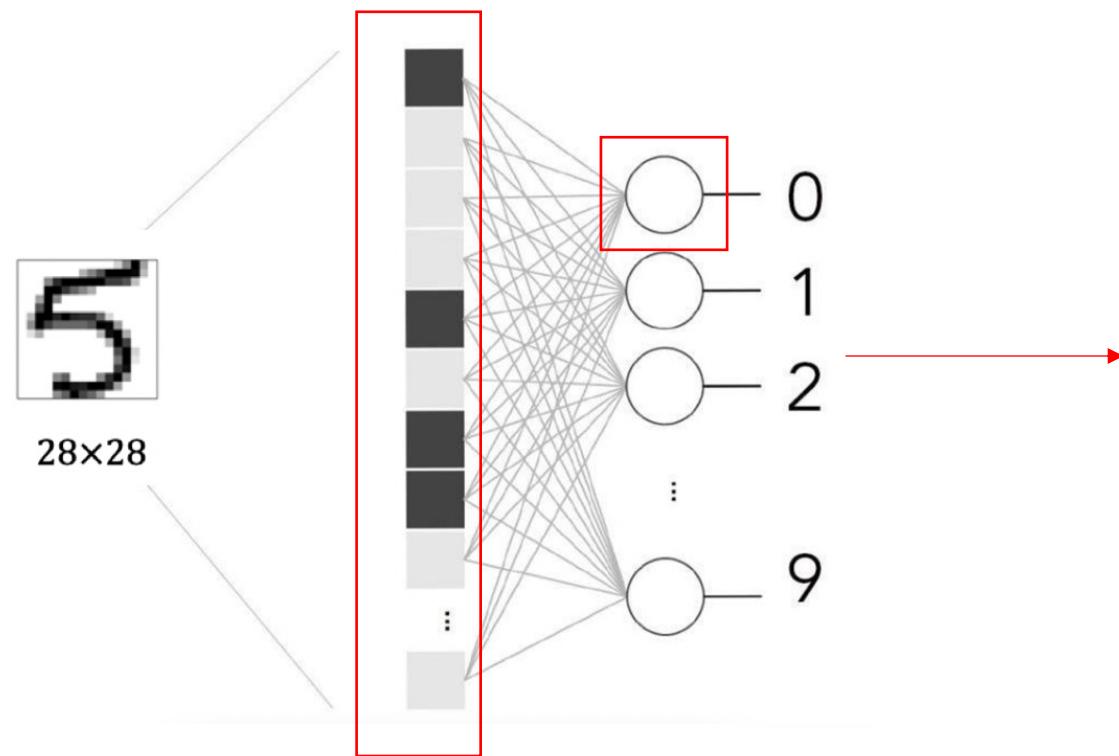
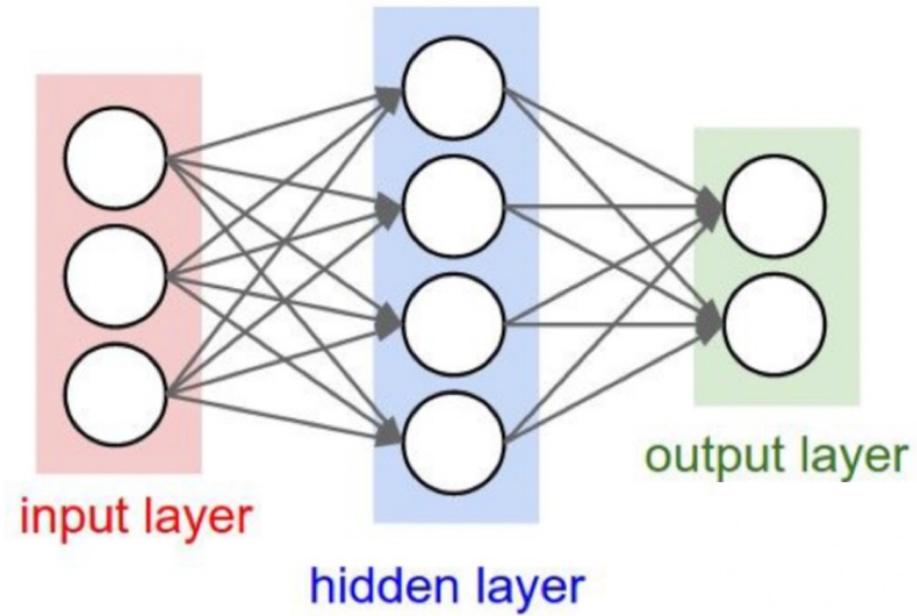
□ لایه‌ها عمدهاً شامل سه نوع هستند:

□ **لایه ورودی (Input Layer)**: هر نورون در این لایه با یک ویژگی ورودی ارتباط دارد.

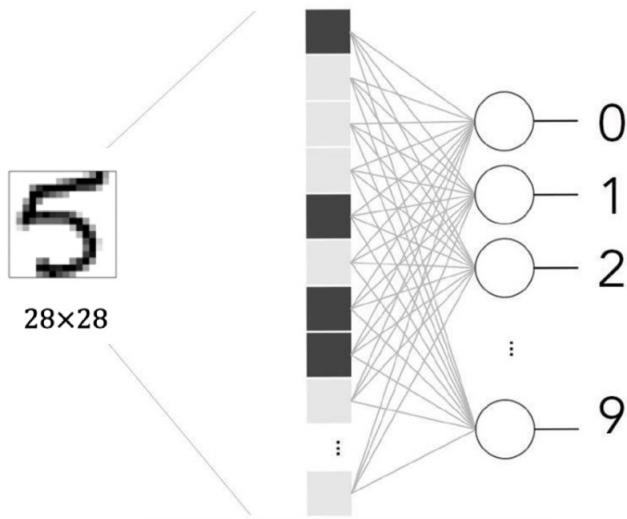
□ **لایه‌های مخفی (Hidden Layers)**: هر نورون در این لایه با همه نورون‌های لایه قبل و بعد ارتباط دارد. (وظیفه پردازش اطلاعات)

□ **لایه خروجی (Output Layer)**: هر نورون در این لایه با یک خروجی ارتباط دارد.

# شبکه‌های عصبی



# دسته بندی تصویر با یادگیری ماشین



• ۳ گام اصلی برای اموزش:

$$y = f(x|\theta) \quad 1. \text{ انتخاب مدل}$$

$$loss = compare(y_{true}, y_{pred} = f(x|\theta)) \quad 2. \text{ معیار ارزیابی}$$

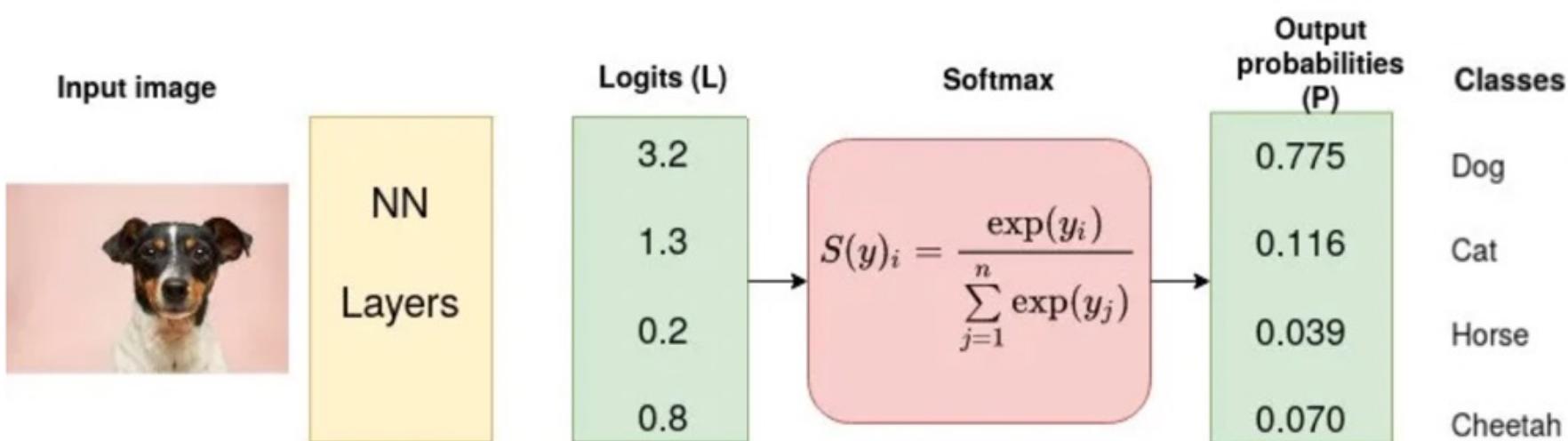
$$\theta^* = \min_{\theta} loss(y_{true}, f(x|\theta)) \quad 3. \text{ بهینه سازی}$$

# Cross-entropy Loss

$$\begin{array}{ccccccc} 5.1 & & & 164.0 & & & 0.87 \\ -1.7 & \xrightarrow{\text{exp}} & 0.87 & \xrightarrow{\text{normalize}} & 0.00 & & \\ 3.2 & & 24.5 & & & & 0.13 \end{array}$$

❑ مثال: چطور وزن ها ( $\theta$ ) به دست اید تا بهترین Logit حاصل شود؟

❑ ابتدا توسط softmax، بردار احتمال logit را به بردار میکنیم.

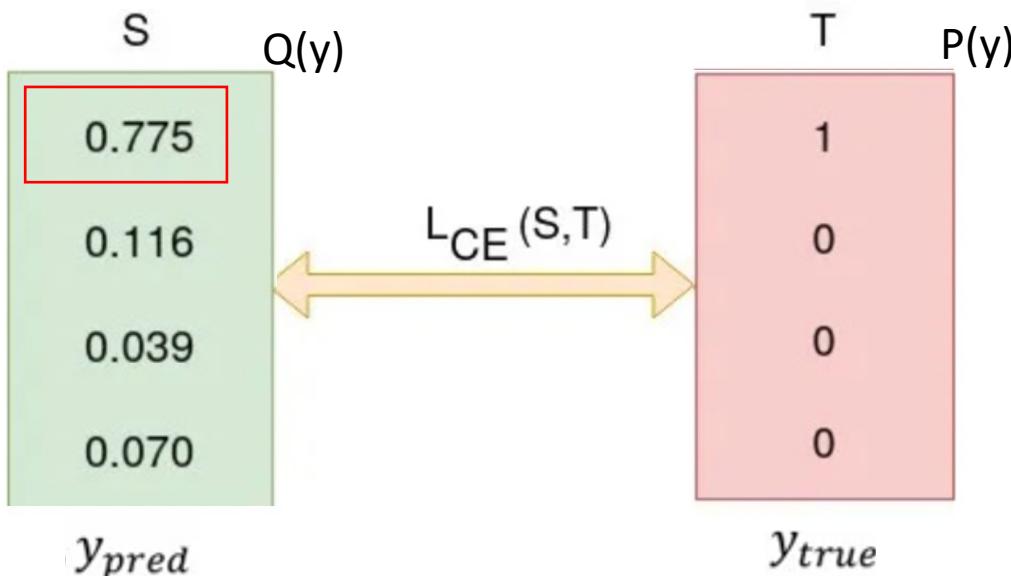


# Cross-entropy Loss

❑ مثال: چطور وزن ها ( $\theta$ ) به دست اید تا بهترین Logit حاصل شود؟

$$loss = compare(y_{true}, y_{pred} = f(x|\theta))$$

❑ سپس بردار احتمال، با بردار برچسب درست (one-hot vector) مقایسه می شود.



✓ برای مقایسه دو بردار احتمالی از  $D_{KL}$  استفاده می شود

$$D_{KL}(P, Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

$$L_{CE} = -\log Q(y_{pred \ (of \ true \ class)})$$

Loss زمانی ۰ می شود که مقدار پیش بینی شده برای کلاس درست ۱ باشد

# Cross-entropy Loss

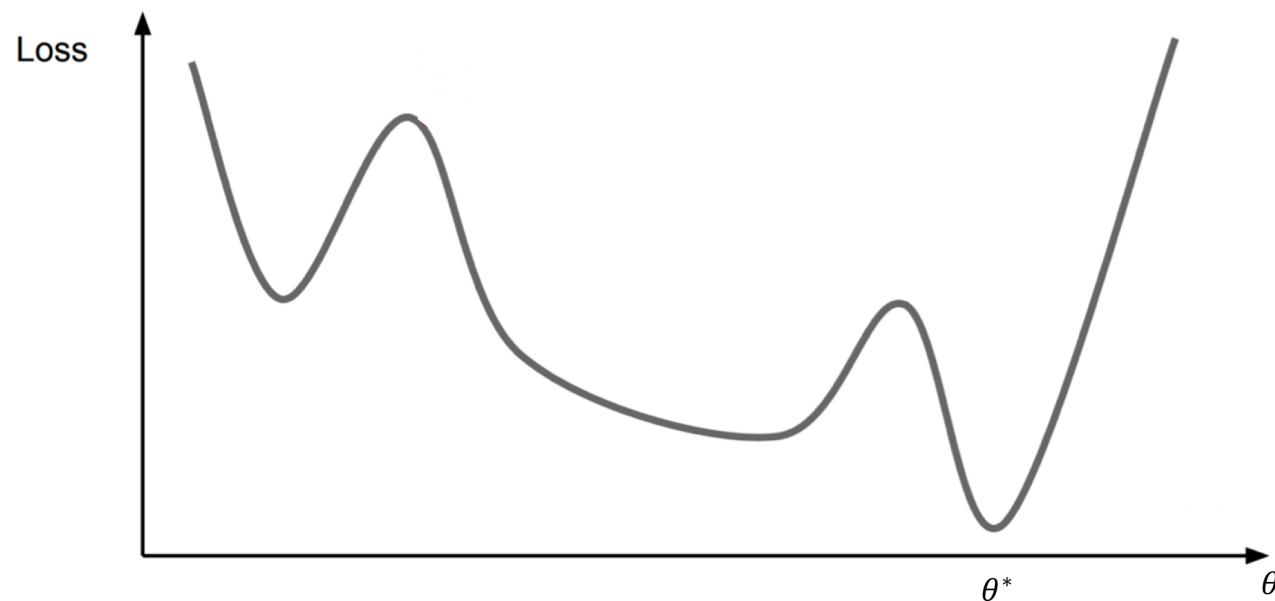
❑ مقدار Cross-entropy Loss  $\max$  و  $\min$  در  $+\infty$  و  $0$  ✓

❑ مقدار اولیه Cross-entropy Loss  $= (c \log c) \checkmark$  تعداد کلاس ها

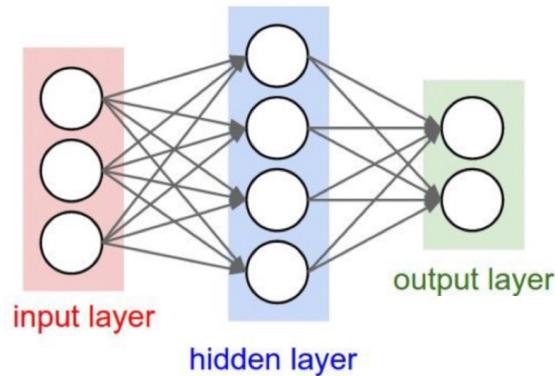
## بهینه سازی

$$\theta^* = \min_{\theta} \text{loss}(y_{true}, f(x|\theta))$$

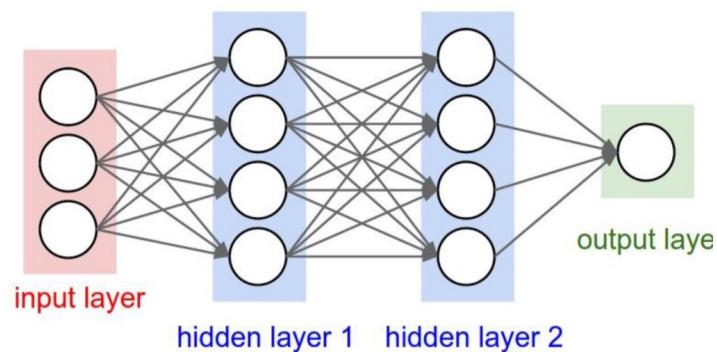
□ یافتن  $\theta^*$  با استفاده از روش Gradient Descent



# شبکه‌های عصبی



- شبکه عصبی یک لایه، شامل یک لایه مخفی است.

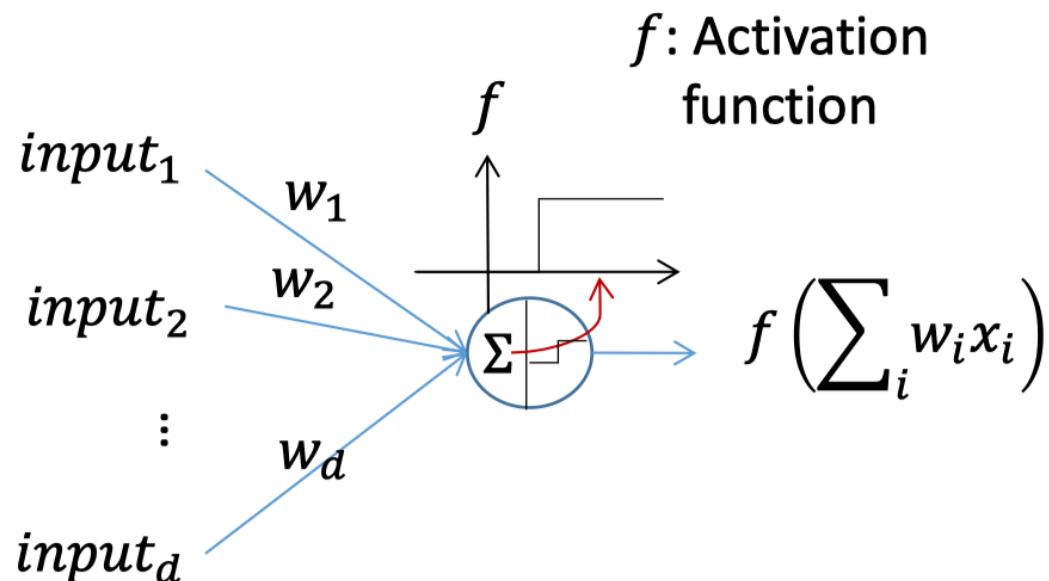


- شبکه عصبی دو لایه، شامل دو لایه مخفی است.

- ✓ یک شبکه عصبی شامل تعدادی لایه خطی و توابع فعال سازی غیرخطی است.

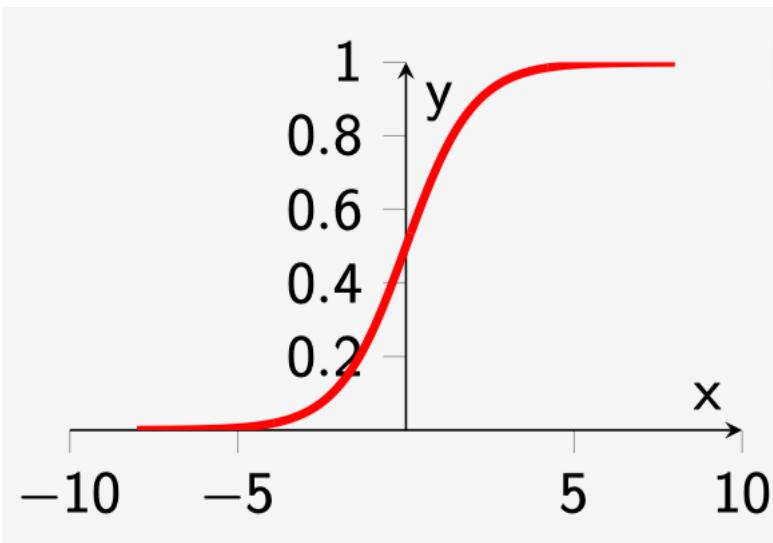
# توابع فعال‌سازی

- توابع فعال‌سازی وظیفه اضافه کردن مولفه غیرخطی به شبکه را دارند.
- این توابع موجب می‌شوند شبکه‌ها الگوها و ویژگی‌های پیچیده‌تری را یاد بگیرد.



# توابع فعالسازی

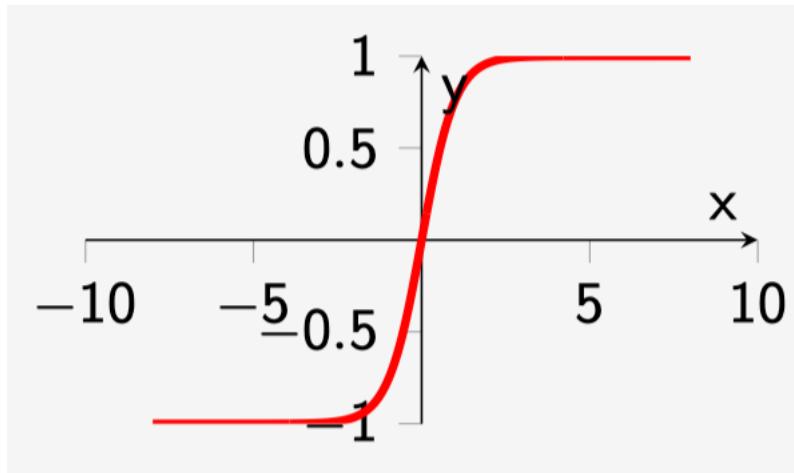
## Sigmoid activation function $\square$



- ✓ در بازه  $(0,1)$  قرار دارد. (می‌تواند به عنوان تابع احتمال قابل استفاده باشد)
- ✓ اگر ورودی به سمت مقادیر خیلی بزرگ (مثبت یا منفی) حرکت کند، خروجی به مقدار یک (برای مقادیر بزرگ مثبت) یا صفر (برای مقادیر بزرگ منفی) همگرا می‌شود. (در بازه  $0$  تا  $1$  اشباع می‌شود.)

# توابع فعال‌سازی

Hyperbolic tangent activation function  $\square$

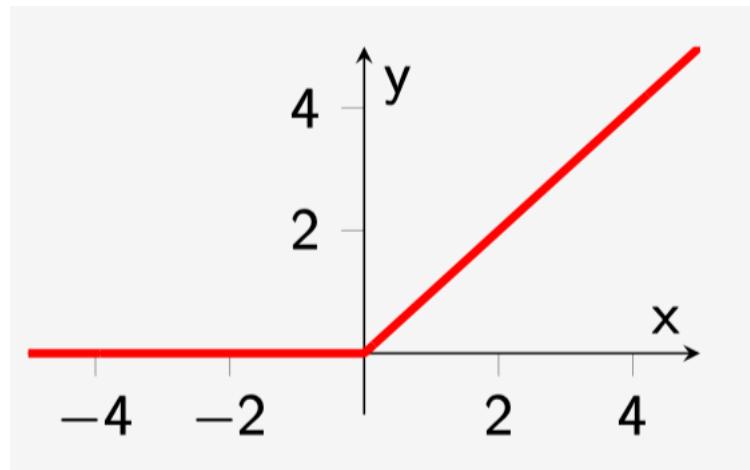


✓ اگر ورودی به سمت مقادیر خیلی بزرگ (مثبت یا منفی) حرکت کند، خروجی به مقدار یک (برای مقادیر بزرگ مثبت) یا منفی یک (برای مقادیر بزرگ منفی) همگرا می‌شود. (در بازه  $-1 \leq y \leq 1$  اشباع می‌شود.)

✓ در بازه  $(-1, 1)$  دارد.

# توابع فعال‌سازی

Rectified linear unit activation function (ReLU)  $\square$

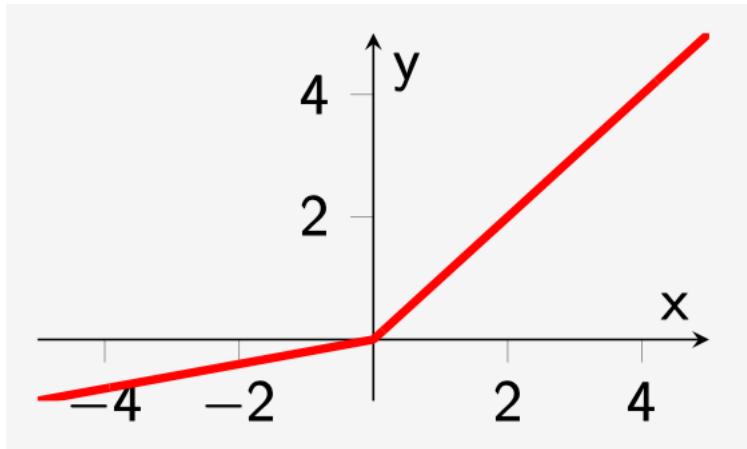


✓ تمام مقادیر منفی بلافاصله صفر می‌شوند که توانایی مدل برای آموزش داده‌ها را کاهش می‌دهد.

- ✓ محاسبه سریع
- ✓ برای مقادیر بزرگ مثبت اشباع نمی‌شود.

# توابع فعال‌سازی

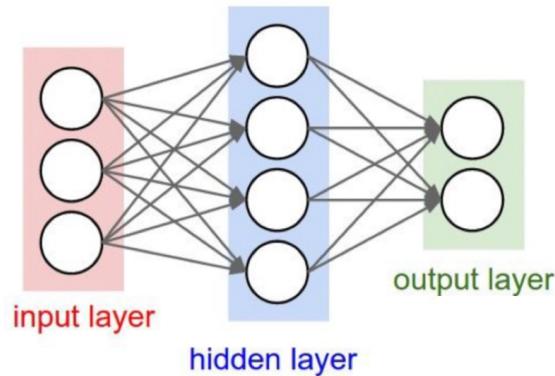
Leaky ReLU activation function  $\square$



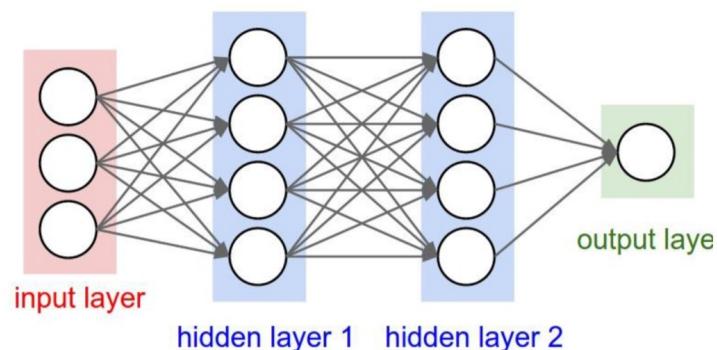
✓ به افزایش دامنه عملکرد ReLU کمک می‌کند. (با افزودن یک تابع خطی برای مقادیر منفی)

- ✓ محاسبه سریع
- ✓ برای مقادیر بزرگ اشباع نمی‌شود.

# شبکه‌های عصبی



- شبکه عصبی یک لایه، شامل یک لایه مخفی است.

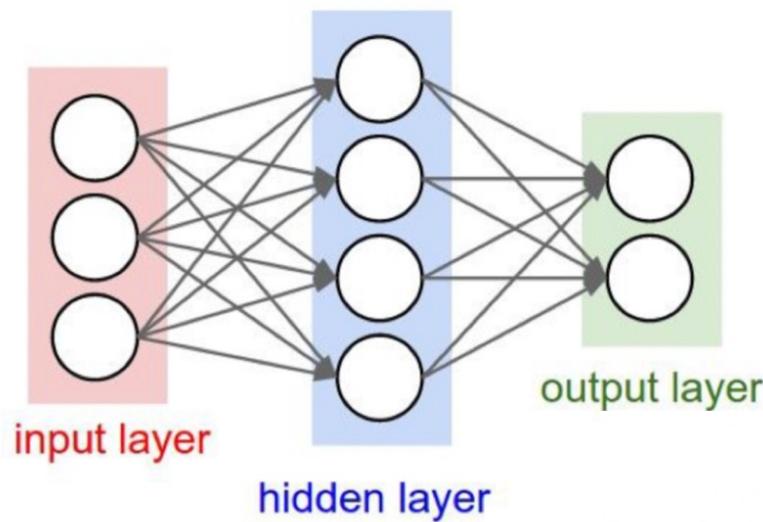


- ❑ شبکه عصبی دو لایه، شامل دو لایه مخفی است.

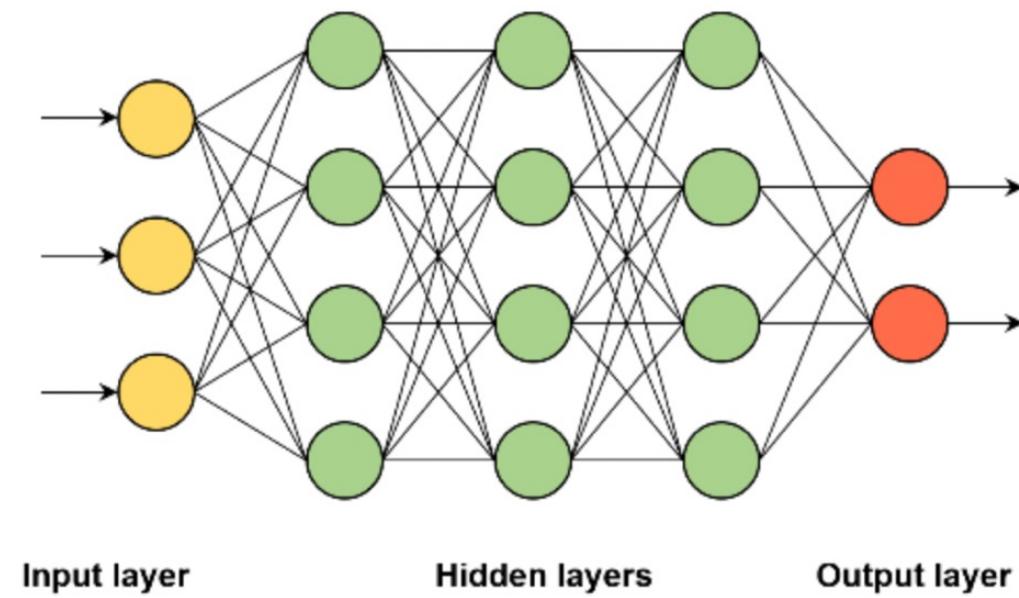
- ✓ یک شبکه عصبی شامل تعدادی لایه خطی و توابع فعال سازی غیرخطی است.

# شبکه‌های عصبی عمیق

- شبکه عصبی عمیق، شبکه عصبی با چند لایه مخفی است.

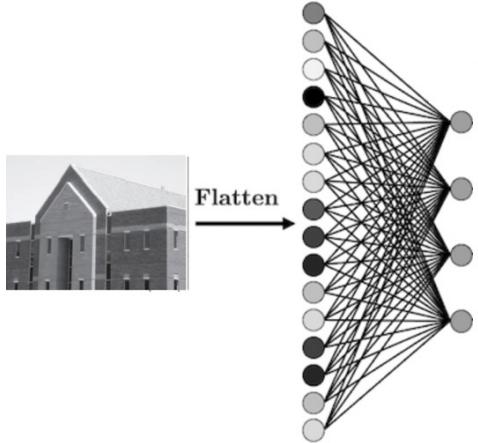


شبکه عصبی



شبکه عصبی عمیق

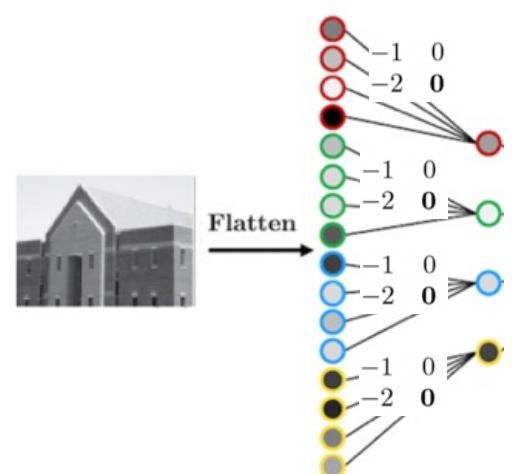
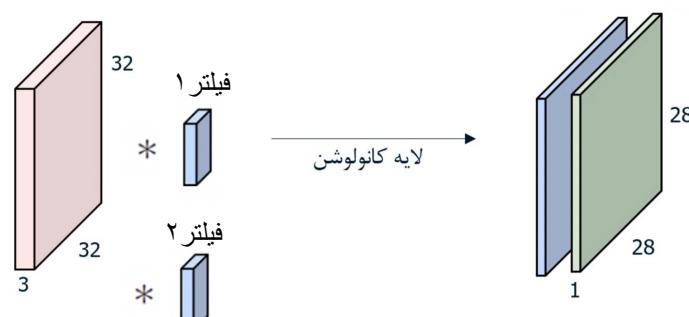
# شبکه‌های عصبی عمیق



□ لایه کاملاً متصل (Fully Connected)

$$\text{Input Image} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \text{Output Image}$$

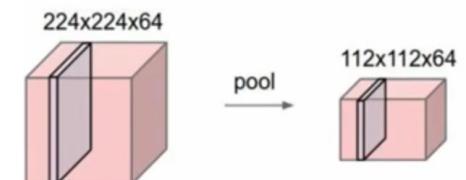
$$\text{Input Image} * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \text{Output Image}$$



اشتراك گذاري وزن ها = parameter sharing

□ لایه کانولوشنی (Convolutional)

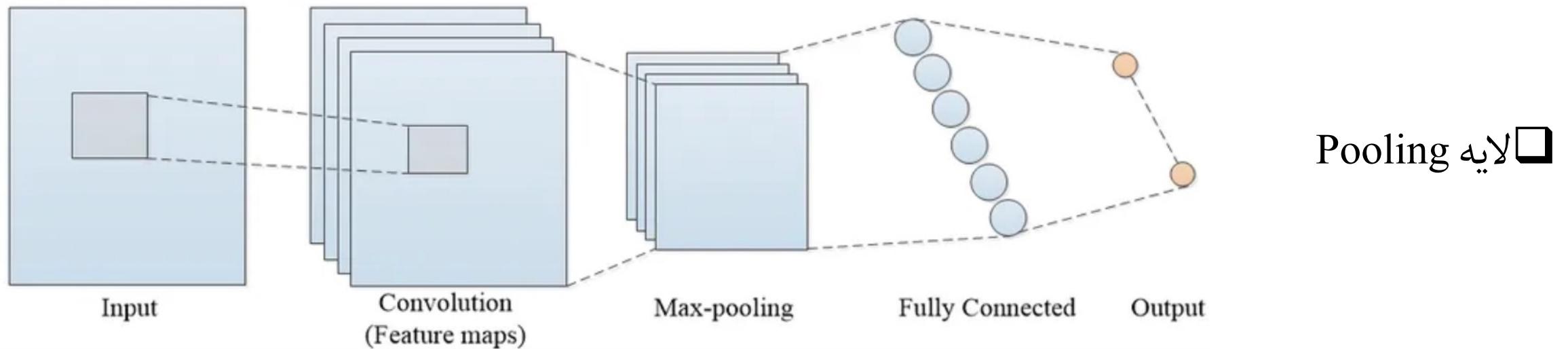
□ لایه Pooling



# شبکه‌های عصبی عمیق

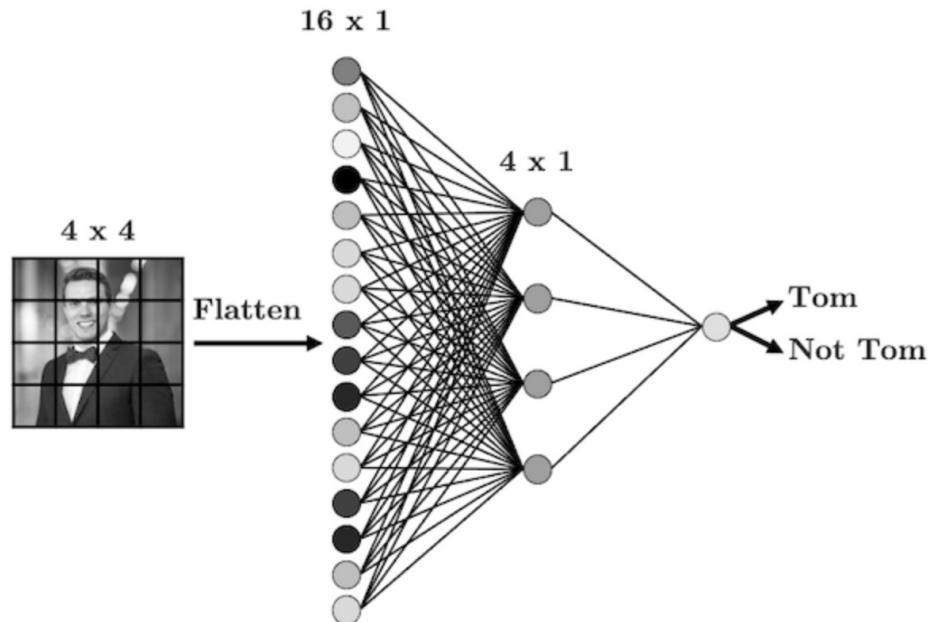
□ لایه کاملاً متصل (Fully Connected)

□ لایه کانولوشنی (Convolutional)



□ لایه Pooling

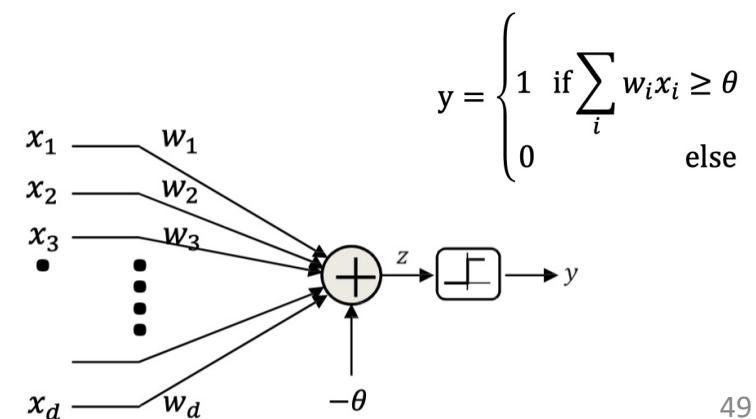
# لایه کاملاً متصل (Fully Connected Layer)



$$\text{Total Parameters} = (16 \times 4 + 4) + (4 \times 1 + 1) = 73$$

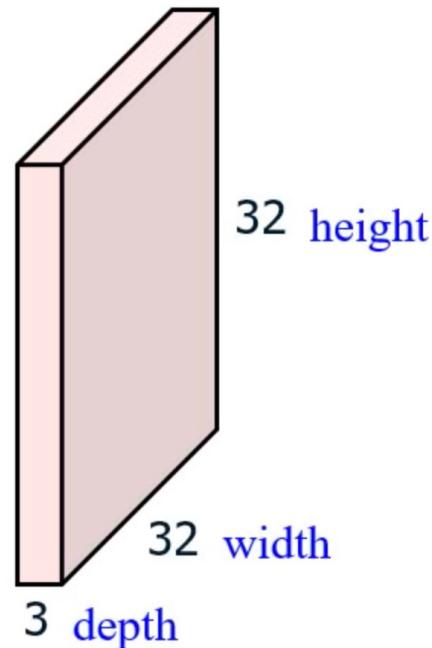
❑ هر نورون در یک لایه به همه نورون‌ها در لایه‌های قبل و بعد خود متصل است.

- ✓ تعداد زیاد پارامترها و محاسبات سنگین
- ✓ ترتیب ویژگی‌ها مهم نیست.

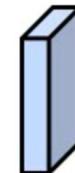


# لایه کانولوشنی (Convolution Layer)

32x32x3 image



5x5x3 filter



5  $\times$  5  $\times$  1 تصویر

1	1	1	0	0
0	1	1	1	0
0	0	1 $\times$ 1	1 $\times$ 0	1 $\times$ 1
0	0	1 $\times$ 0	1 $\times$ 1	0 $\times$ 0
0	1	1 $\times$ 1	0 $\times$ 0	0 $\times$ 1

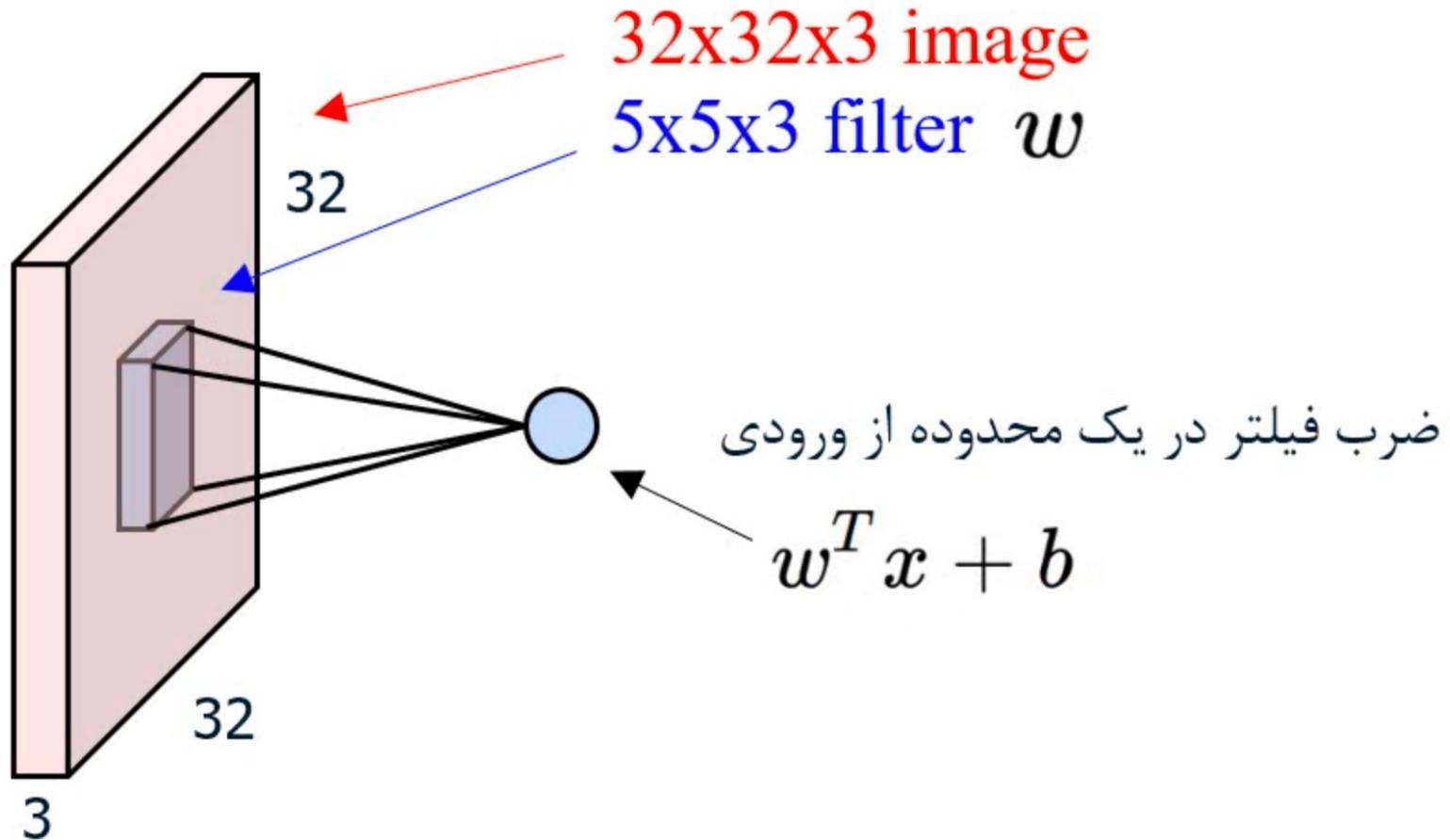
Image

4	3	4
2	4	3
2	3	4

Convolved  
Feature

خروجی 3  $\times$  3  $\times$  1 فیلتر 50 تصویر

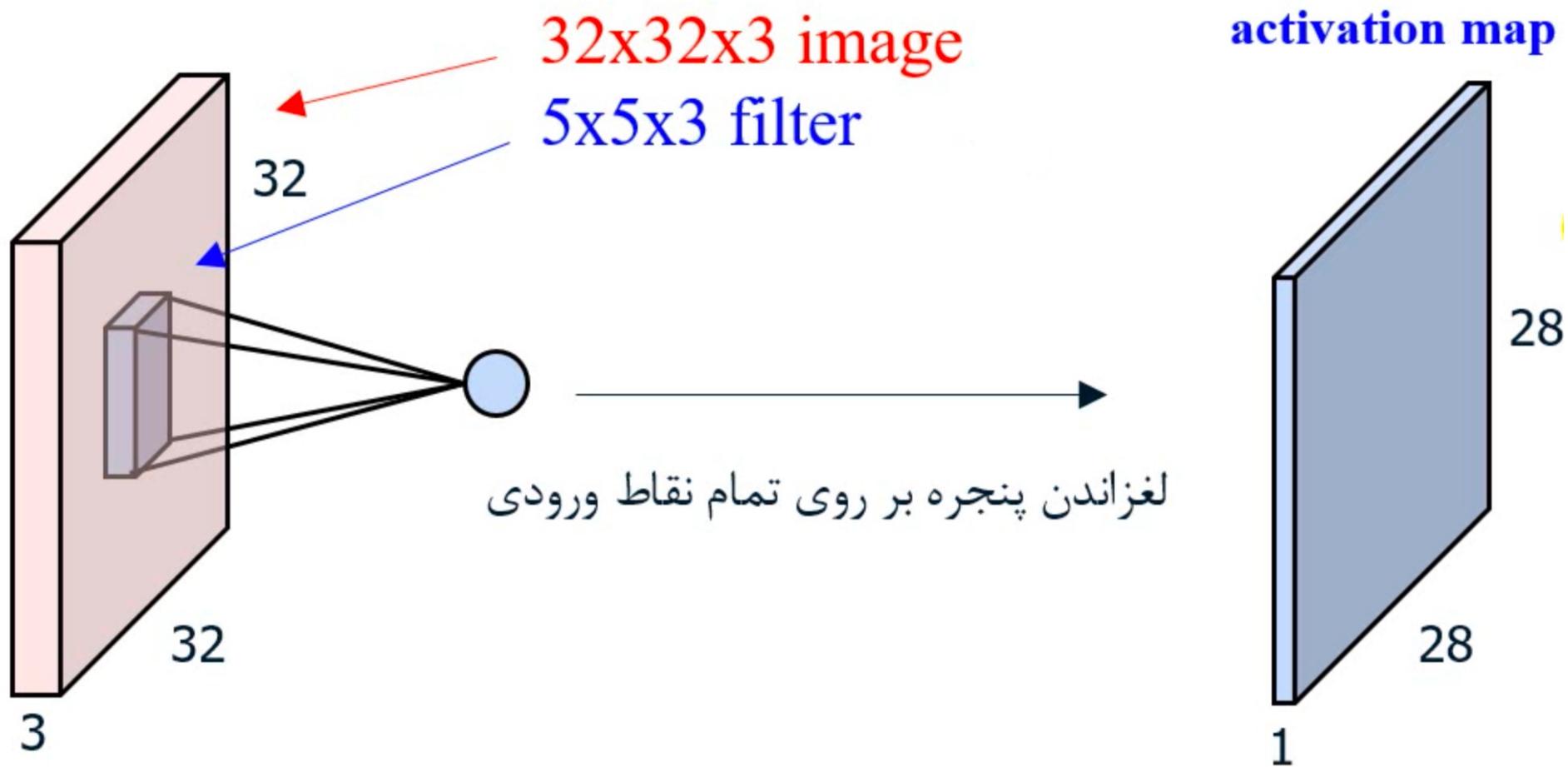
# لایه کانولوشنی (Convolution Layer)



❑ هر نورون در لایه مخفی فقط به قسمت کوچکی از تصویر نگاه می‌کند.

✓ پیکسل‌هایی که از هم دور هستند احتمالاً به هم مرتبط نیستند و بنابراین نیازی به اتصال ندارند.

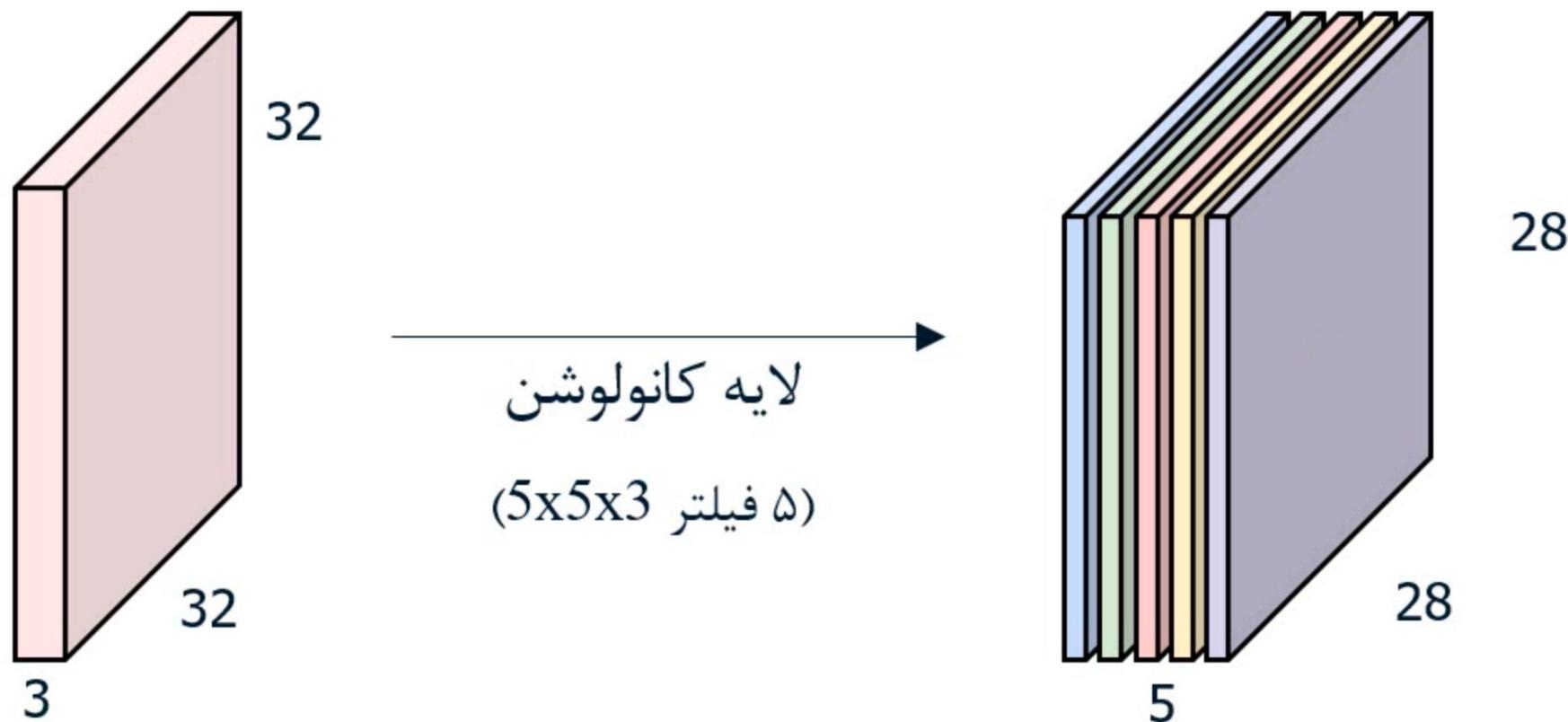
# لایه کانولوشنی (Convolution Layer)



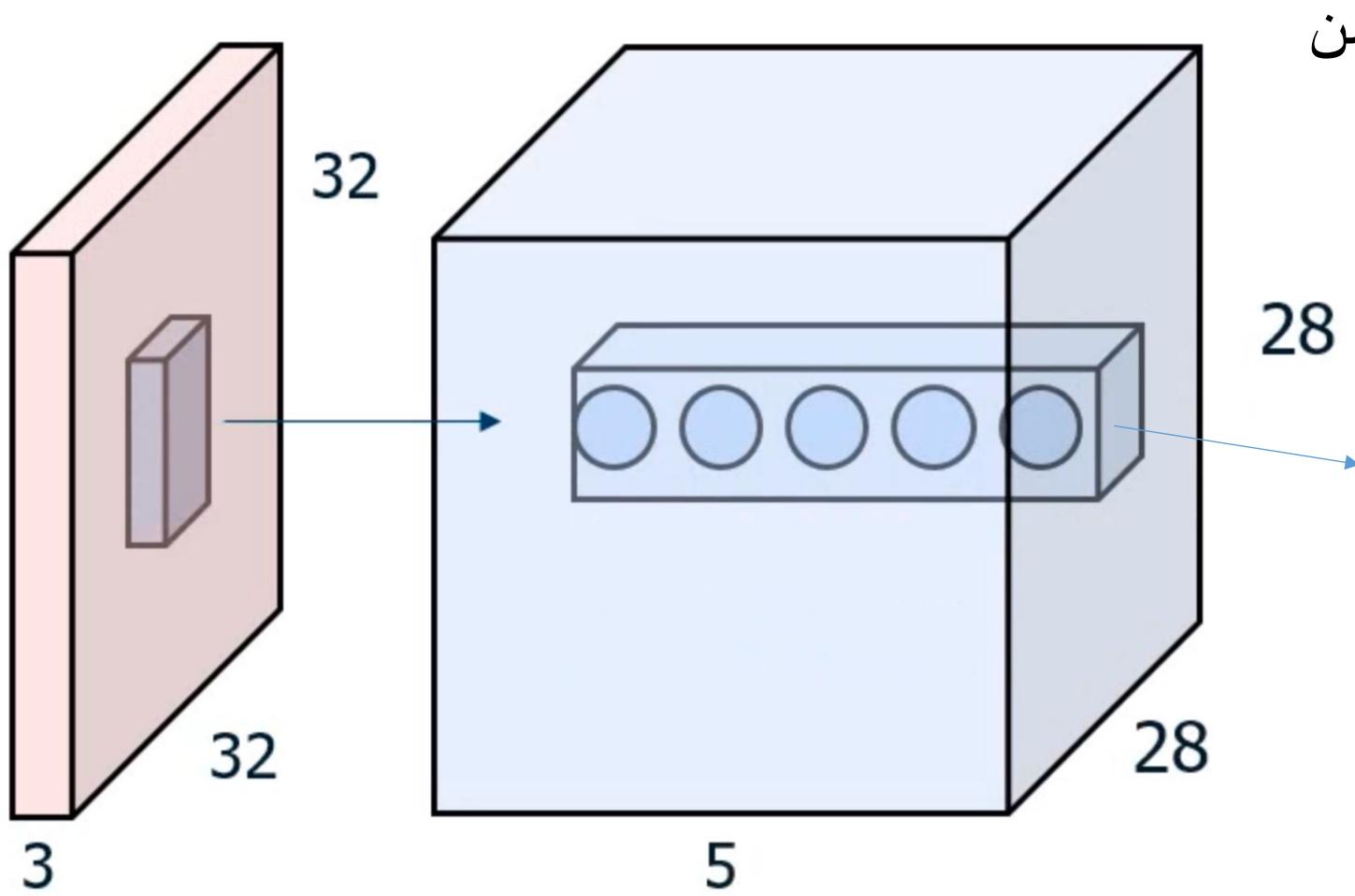
# لایه کانولوشنی (Convolution Layer)



# لایه کانولوشنی (Convolution Layer)



# لایه کانولوشنی (Convolution Layer)

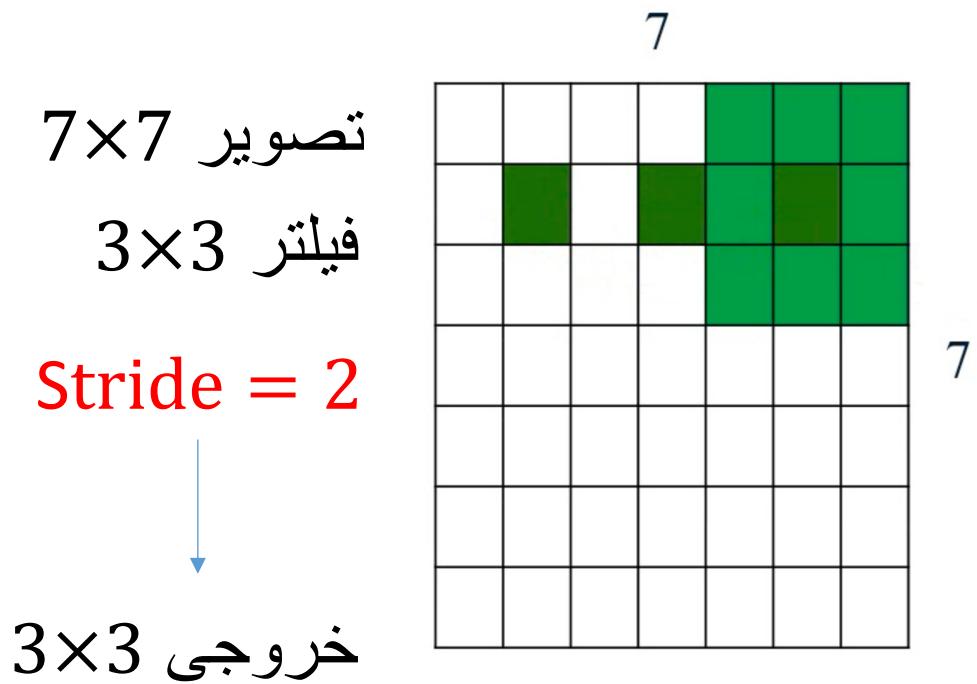
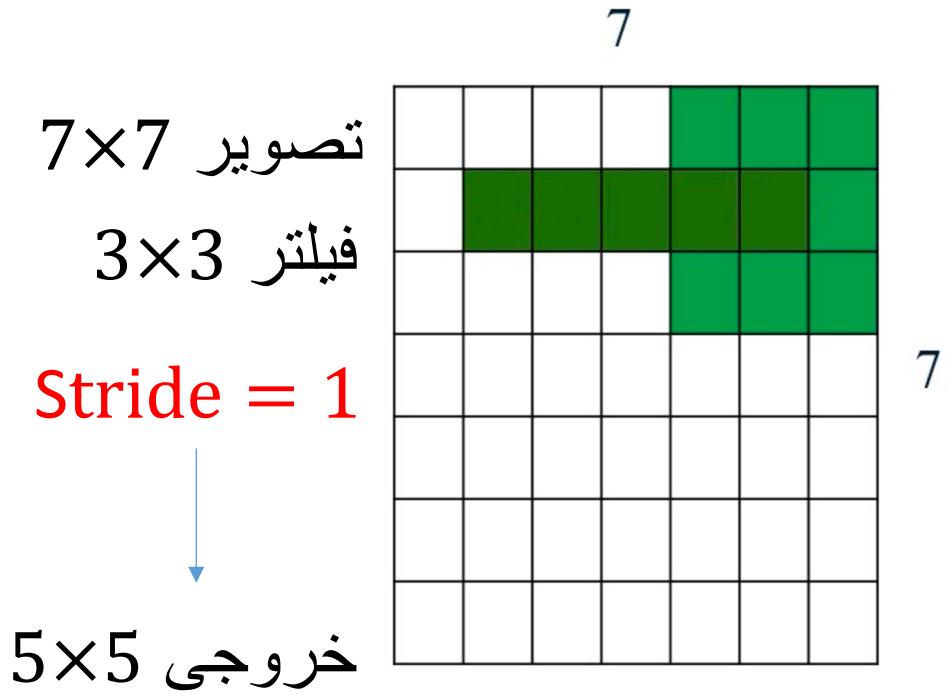


نمایش دیگری از لایه کانولوشن

- ✓ ۵ دیدگاه متفاوت از یک نقطه از تصویر ورودی

# Convolution stride

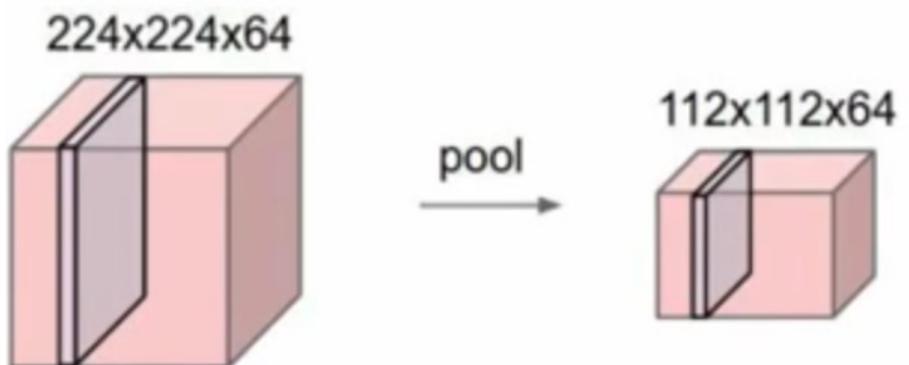
- جایه جایی پنجره با گام بزرگتر



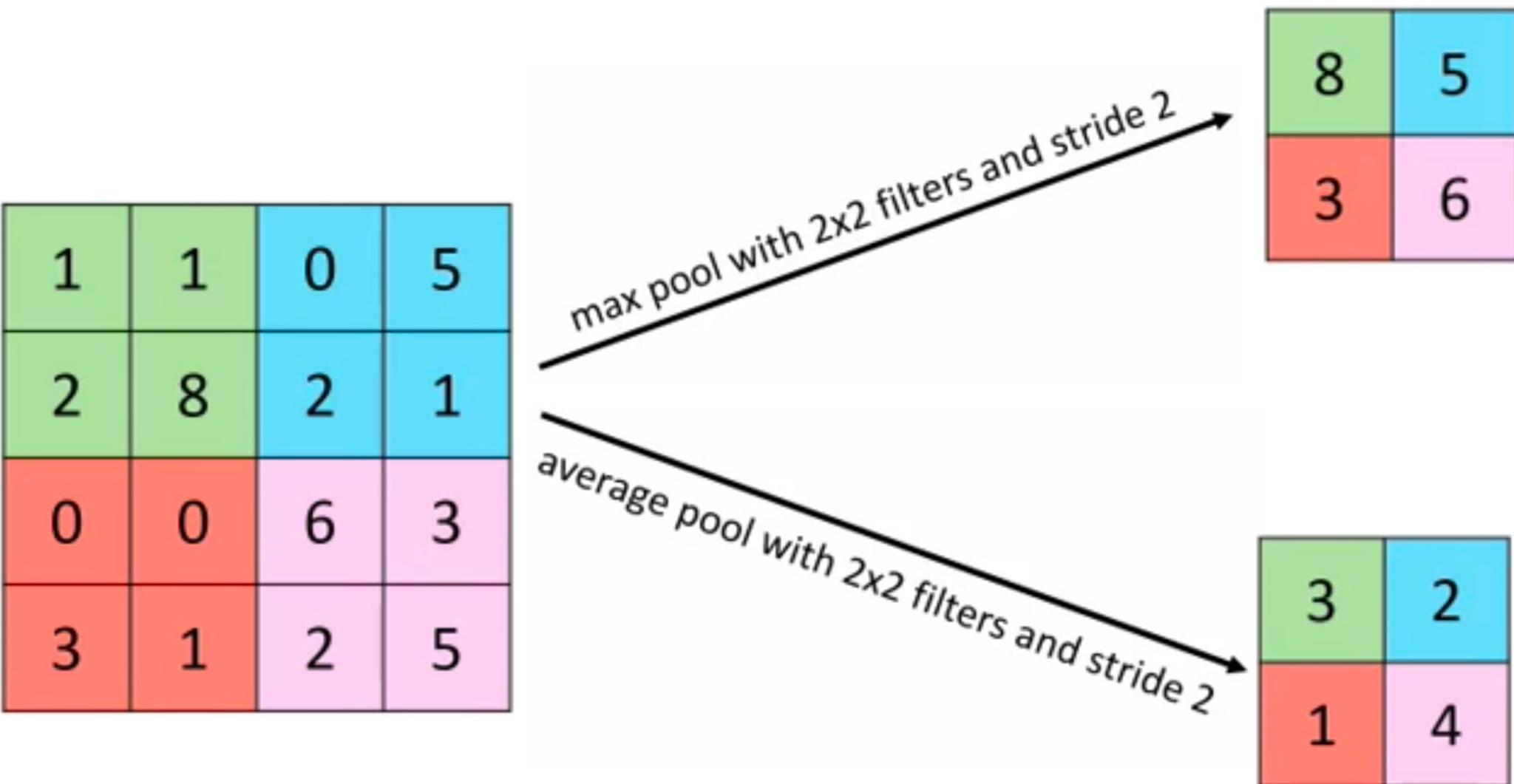
✓ هدف از stride: کاهش سایز ابعاد مکانی

# لایه Pooling

- هر چه تعداد فیلترها زیاد می‌شود حجم پردازش بیشتر می‌شود
- ❑ هدف از لایه Pooling: کاهش سایز ابعاد مکانی، کاهش تعداد پارامترها و در نتیجه کاهش حجم پردازش
- ❑ عملیات AveragePooling و MaxPooling دو روش متداول هستند



# Pooling لایه

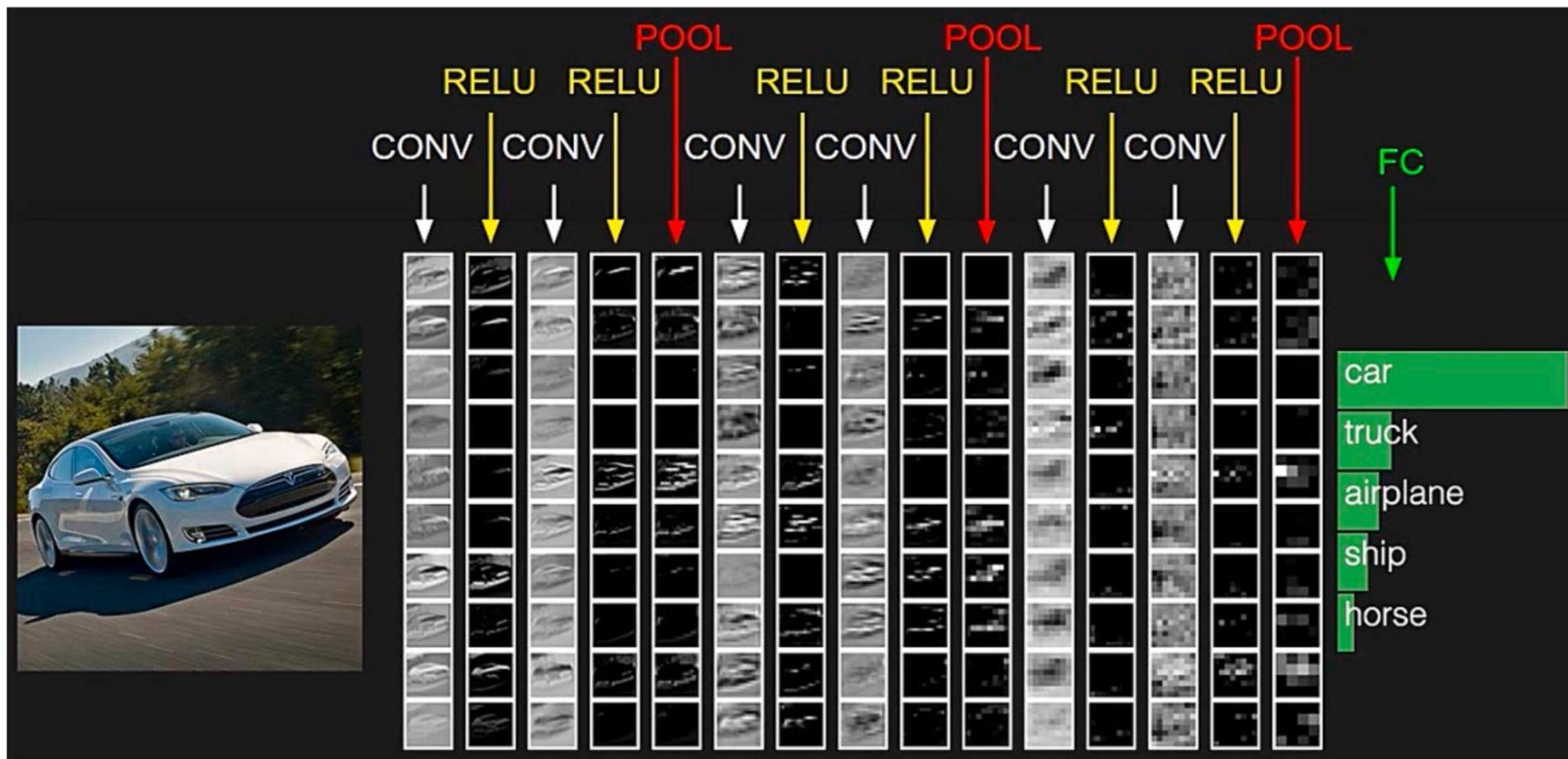


# Hyper Parameter vs. Parameter

- پارامترها وزن‌هایی هستند که شبکه یاد می‌گیرد.
- هایپرپارامترها، پارامترهایی هستند که طراح باید به شبکه دهد:
  1. تعداد و اندازه فیلترها
  2. اندازه Stride
  3. نحوه Padding
  4. (عملیات Pooling، اندازه فیلتر و اندازه Pooling)

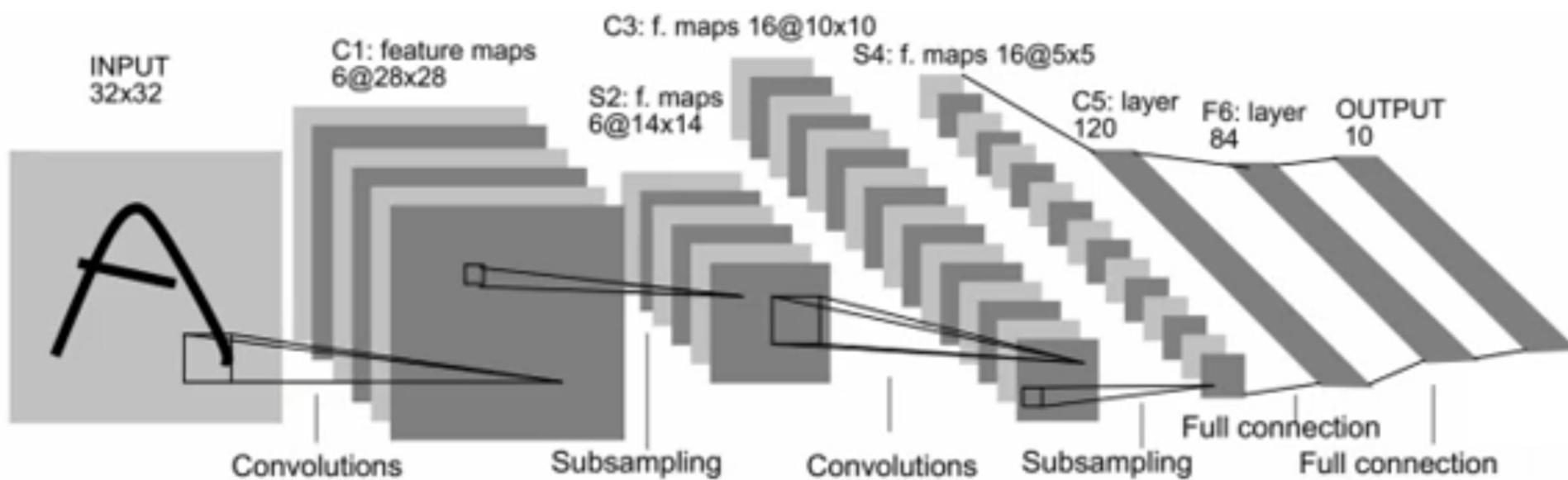
# شبکه کانولوشنی

مثال:



# LeNet-5

- شبکه LeNet-5 (سال ۱۹۹۸):
- ۵ لایه آموزشی (۲ لایه کانولوشنی - ۳ لایه کاملاً متصل)



# K نزدیکترین همسایه

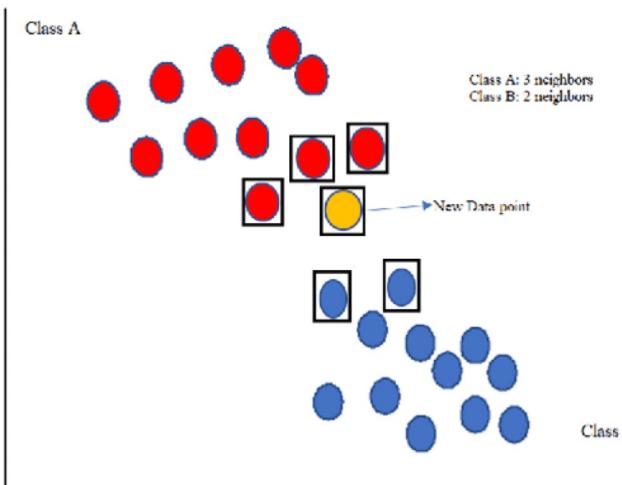
K-Nearest Neighbors (KNN)

# دسته‌بند K-Nearest Neighbors (KNN)

دسته بند  $k$  نزدیکترین همسایه:

## نحوه عملکرد:

- فاصله نقطه تست از همه نقاط داده‌های آموزشی محاسبه می‌شود (مثلاً با استفاده از فاصله اقلیدسی).
- نزدیکترین  $k$  نقاط به نقطه تست انتخاب می‌شوند.
- رای‌گیری می‌کنیم (رای اکثریت).



# يادگیری بدون نظارت

Unsupervised Learning

# K-means