

رسالة محمد



# مبانی بینایی کامپیوتر

مدرس: محمدرضا محمدی

۱۳۹۹

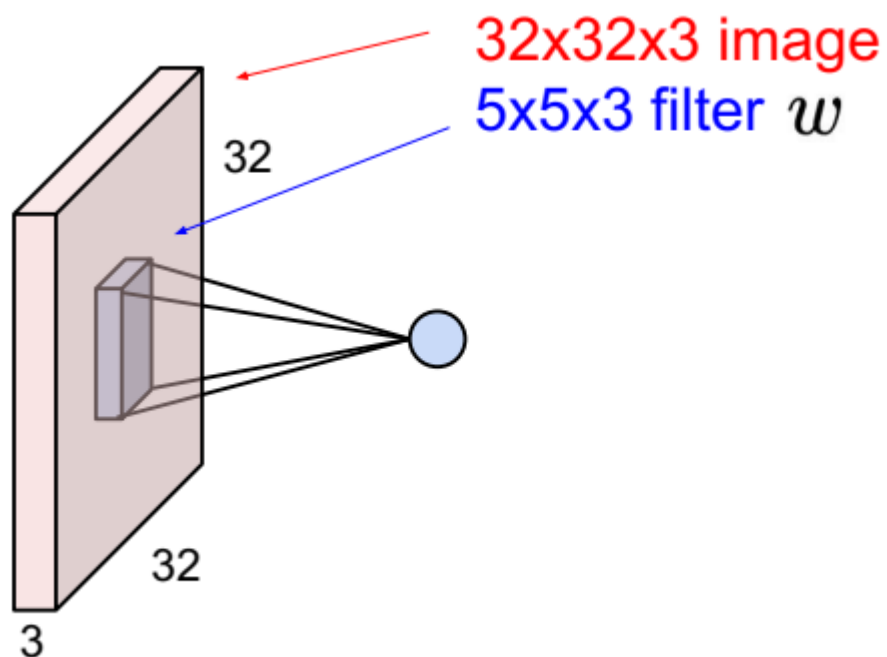
# شبکه‌های عصبی کانولوشنی

Convolutional Neural Networks

# لایه کانولوشنی

خروجی لایه کانولوشنی حاصل فیلتر کردن ماتریس ورودی با فیلتر مربوطه است که به صورت مکانی بر روی آن لغزانده می شود

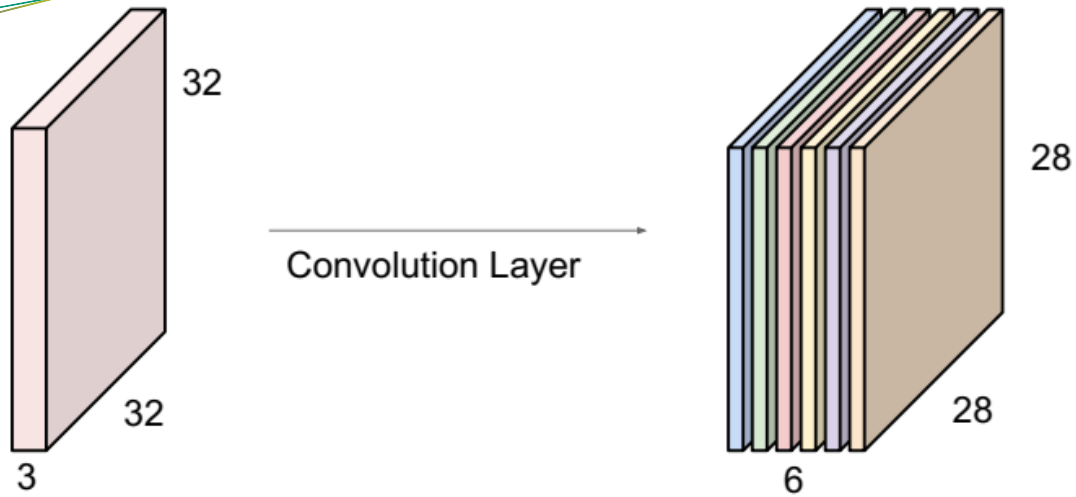
ورودی یک ماتریس ۳ بعدی است



خروجی برابر با ضرب داخلی بین فیلتر و همسایگی مربوطه برای هر پیکسل است که معادل با  $75$  ضرب و جمع است

$$w^T x + b$$

# لایه کانولوشنی



- $W_2 = (W_1 - F + 2P)/S + 1$
- $H_2 = (H_1 - F + 2P)/S + 1$
- $D_2 = K$

- ورودی یک حجم با ابعاد  $W_1 \times H_1 \times D_1$  است

- ابرپارامترهای لایه کانولوشنی عبارتند از:

- تعداد فیلترها  $K$

- اندازه فیلترها  $F$

- اندازه گام  $S$

- مقدار گسترش مرزها  $P$

- خروجی یک حجم با ابعاد  $W_2 \times H_2 \times D_2$  است

- پارامترهای لایه کانولوشنی عبارتند از  $F \cdot F \cdot D_1 \cdot K$  وزن فیلترها و  $K$  بایاس که باید آموزش ببینند

# لایه کانولوشنی در Keras

```
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid',  
                    data_format=None, dilation_rate=(1, 1), activation=None,  
                    use_bias=True, kernel_initializer='glorot_uniform',  
                    bias_initializer='zeros', kernel_regularizer=None,  
                    bias_regularizer=None, activity_regularizer=None,  
                    kernel_constraint=None, bias_constraint=None)
```

**filters:** Integer, the dimensionality of the output space

**kernel\_size:** Specifying the height and width of the 2D convolution window

**activation:** Activation function to use. If you don't specify anything, no activation is applied (see keras.activations)

**strides:** Specifying the strides of the convolution

**padding:** One of “valid” or “same”

# شبیه‌سازی

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 128)	3584
conv2d_1 (Conv2D)	(None, 28, 28, 128)	147584
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 10)	1003530
Total params: 1,154,698		
Trainable params: 1,154,698		
Non-trainable params: 0		

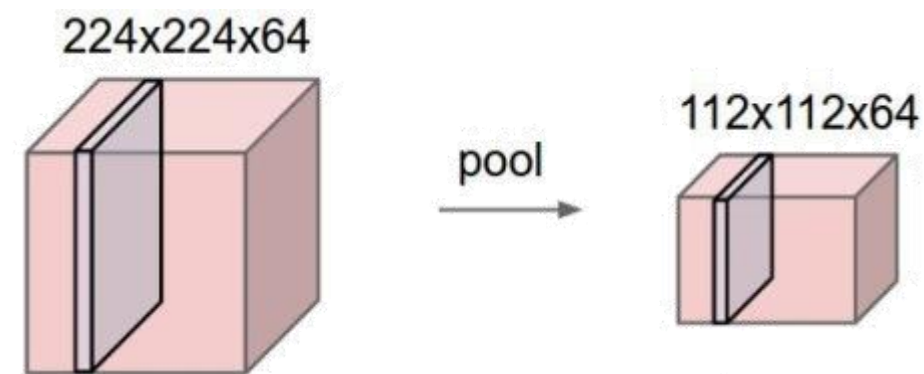
```
Epoch 1/10
500/500 [=====] - 8s 17ms/step - loss: 3.9606 - accuracy: 0.1535 - val_loss: 2.2777 - val_accuracy: 0.1178
Epoch 2/10
500/500 [=====] - 8s 17ms/step - loss: 2.1179 - accuracy: 0.2204 - val_loss: 2.1473 - val_accuracy: 0.1881
Epoch 3/10
500/500 [=====] - 8s 17ms/step - loss: 1.9560 - accuracy: 0.2870 - val_loss: 1.9654 - val_accuracy: 0.3169
Epoch 4/10
500/500 [=====] - 8s 16ms/step - loss: 1.7399 - accuracy: 0.3720 - val_loss: 1.7206 - val_accuracy: 0.3825
Epoch 5/10
500/500 [=====] - 8s 16ms/step - loss: 1.5726 - accuracy: 0.4372 - val_loss: 1.6168 - val_accuracy: 0.4404
Epoch 6/10
500/500 [=====] - 8s 16ms/step - loss: 1.3197 - accuracy: 0.5379 - val_loss: 1.4584 - val_accuracy: 0.5064
Epoch 7/10
500/500 [=====] - 8s 16ms/step - loss: 1.0624 - accuracy: 0.6320 - val_loss: 1.5194 - val_accuracy: 0.5238
Epoch 8/10
500/500 [=====] - 8s 16ms/step - loss: 0.8247 - accuracy: 0.7158 - val_loss: 1.6274 - val_accuracy: 0.5264
Epoch 9/10
500/500 [=====] - 8s 16ms/step - loss: 0.6205 - accuracy: 0.7847 - val_loss: 1.8569 - val_accuracy: 0.5369
Epoch 10/10
500/500 [=====] - 8s 16ms/step - loss: 0.4384 - accuracy: 0.8489 - val_loss: 2.2961 - val_accuracy: 0.5358
```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 30, 30, 128)	3584
conv2d_3 (Conv2D)	(None, 14, 14, 128)	147584
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 10)	250890
Total params: 402,058		
Trainable params: 402,058		
Non-trainable params: 0		

```
Epoch 1/10
500/500 [=====] - 5s 11ms/step - loss: 3.2766 - accuracy: 0.1728 - val_loss: 2.3051 - val_accuracy: 0.1016
Epoch 2/10
500/500 [=====] - 5s 10ms/step - loss: 2.2956 - accuracy: 0.1169 - val_loss: 2.3022 - val_accuracy: 0.1034
Epoch 3/10
500/500 [=====] - 5s 10ms/step - loss: 2.2843 - accuracy: 0.1263 - val_loss: 2.2962 - val_accuracy: 0.1128
Epoch 4/10
500/500 [=====] - 5s 10ms/step - loss: 2.2708 - accuracy: 0.1405 - val_loss: 2.2218 - val_accuracy: 0.2192
Epoch 5/10
500/500 [=====] - 5s 10ms/step - loss: 1.9456 - accuracy: 0.2968 - val_loss: 1.7332 - val_accuracy: 0.3848
Epoch 6/10
500/500 [=====] - 5s 10ms/step - loss: 1.6547 - accuracy: 0.4096 - val_loss: 1.5550 - val_accuracy: 0.4618
Epoch 7/10
500/500 [=====] - 5s 11ms/step - loss: 1.3392 - accuracy: 0.5275 - val_loss: 1.3940 - val_accuracy: 0.5233
Epoch 8/10
500/500 [=====] - 5s 10ms/step - loss: 1.1706 - accuracy: 0.5935 - val_loss: 1.3571 - val_accuracy: 0.5468
Epoch 9/10
500/500 [=====] - 5s 10ms/step - loss: 1.0184 - accuracy: 0.6461 - val_loss: 1.3662 - val_accuracy: 0.5593
Epoch 10/10
500/500 [=====] - 5s 11ms/step - loss: 0.9017 - accuracy: 0.6888 - val_loss: 1.3725 - val_accuracy: 0.5742
```

# لایه Pooling

- لایه Pooling در خروجی لایه‌های کانولوشنی قرار می‌گیرد و پیکسل‌های همسایه را با یکدیگر ترکیب می‌کند تا ابعاد نقشه‌های ویژگی کاهش بیابد
- یکی از دستاوردهای اصلی لایه Pooling کاهش ابعاد نورون‌ها و کاهش تعداد پارامترهای شبکه است
- لایه Pooling بر روی هر نقشه فعالیت به صورت جداگانه اعمال می‌شود
- میانگین و ماکزیمم متداول هستند





# Pooling لایه

1	1	0	5
2	8	2	1
0	0	6	3
3	1	2	5

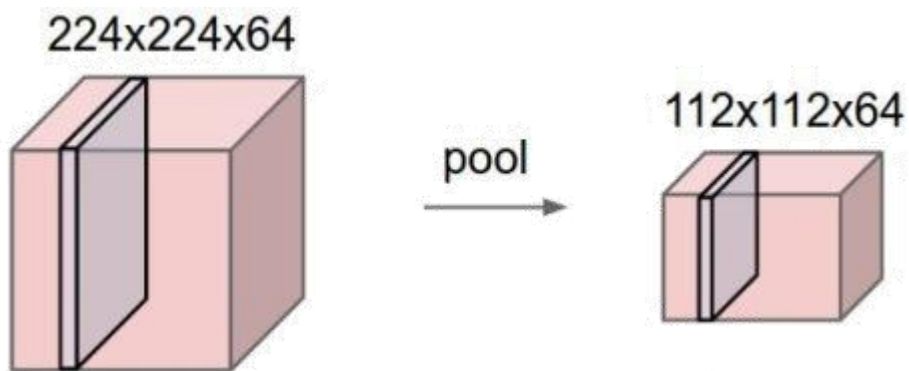
max pool with 2x2 filters and stride 2

8	5
3	6

average pool with 2x2 filters and stride 2

3	2
1	4

# لایه Pooling



- $W_2 = (W_1 - F + 2P)/S + 1$
- $H_2 = (H_1 - F + 2P)/S + 1$
- $D_2 = D_1$

- ورودی یک حجم با ابعاد  $W_1 \times H_1 \times D_1$  است
- ابرپارامترهای لایه Pooling عبارتند از:
  - نحوه ادغام
  - اندازه فیلترها  $F$
  - اندازه گام  $S$
  - مقدار گسترش مرزها  $P$
- خروجی یک حجم با ابعاد  $W_2 \times H_2 \times D_2$  است
- پارمتر ندارد

# لایه Pooling در Keras

```
keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None,  
                           padding='valid', data_format=None)
```

```
keras.layers.AveragePooling2D(pool_size=(2, 2), strides=None,  
                               padding='valid', data_format=None)
```

**pool\_size:** integer or tuple of 2 integers, factors by which to downscale (vertical, horizontal)

**strides:** Integer, tuple of 2 integers, or None. Strides values. If None, it will default to pool\_size

**padding:** One of “valid” or “same”

# شبیه‌سازی

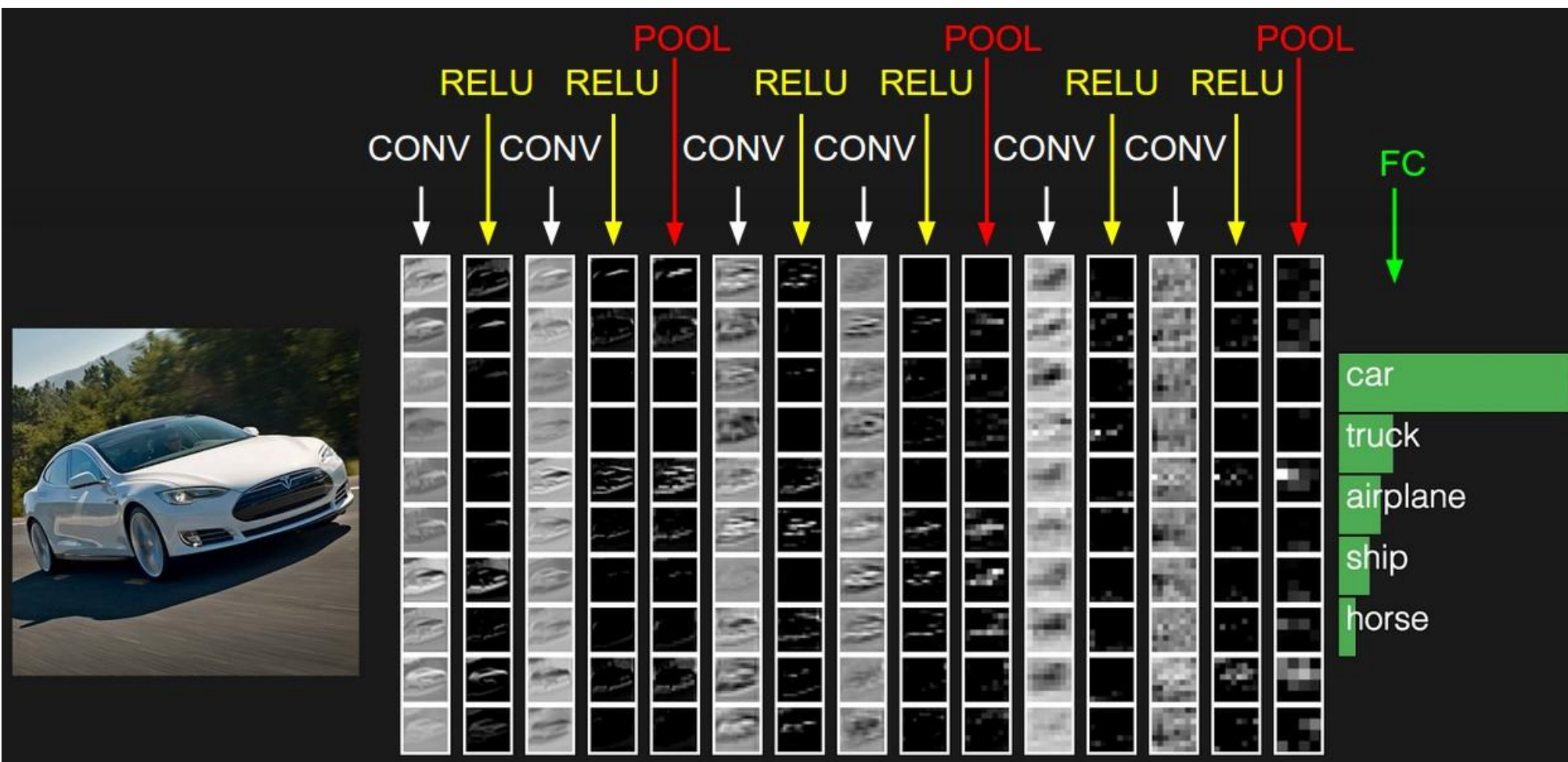
Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 30, 30, 128)	3584
conv2d_5 (Conv2D)	(None, 28, 28, 128)	147584
max_pooling2d (MaxPooling2D)	(None, 14, 14, 128)	0
flatten_2 (Flatten)	(None, 25088)	0
dense_2 (Dense)	(None, 10)	250890
Total params: 402,058		
Trainable params: 402,058		
Non-trainable params: 0		

```
Epoch 1/10
500/500 [=====] - 8s 16ms/step - loss: 2.8436 - accuracy: 0.3582 - val_loss: 1.7058 - val_accuracy: 0.4000
Epoch 2/10
500/500 [=====] - 8s 16ms/step - loss: 1.4645 - accuracy: 0.4832 - val_loss: 1.3355 - val_accuracy: 0.5313
Epoch 3/10
500/500 [=====] - 8s 16ms/step - loss: 1.2001 - accuracy: 0.5833 - val_loss: 1.1815 - val_accuracy: 0.5908
Epoch 4/10
500/500 [=====] - 8s 16ms/step - loss: 1.0611 - accuracy: 0.6356 - val_loss: 1.1972 - val_accuracy: 0.5942
Epoch 5/10
500/500 [=====] - 8s 16ms/step - loss: 0.9638 - accuracy: 0.6700 - val_loss: 1.1687 - val_accuracy: 0.6120
Epoch 6/10
500/500 [=====] - 8s 16ms/step - loss: 0.8891 - accuracy: 0.6923 - val_loss: 1.1796 - val_accuracy: 0.6126
Epoch 7/10
500/500 [=====] - 8s 16ms/step - loss: 0.8143 - accuracy: 0.7190 - val_loss: 1.1703 - val_accuracy: 0.6286
Epoch 8/10
500/500 [=====] - 8s 16ms/step - loss: 0.7423 - accuracy: 0.7408 - val_loss: 1.1978 - val_accuracy: 0.6314
Epoch 9/10
500/500 [=====] - 8s 16ms/step - loss: 0.6749 - accuracy: 0.7669 - val_loss: 1.2484 - val_accuracy: 0.6305
Epoch 10/10
500/500 [=====] - 8s 16ms/step - loss: 0.6164 - accuracy: 0.7855 - val_loss: 1.3708 - val_accuracy: 0.6074
```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 30, 30, 128)	3584
conv2d_3 (Conv2D)	(None, 14, 14, 128)	147584
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 10)	250890
Total params: 402,058		
Trainable params: 402,058		
Non-trainable params: 0		

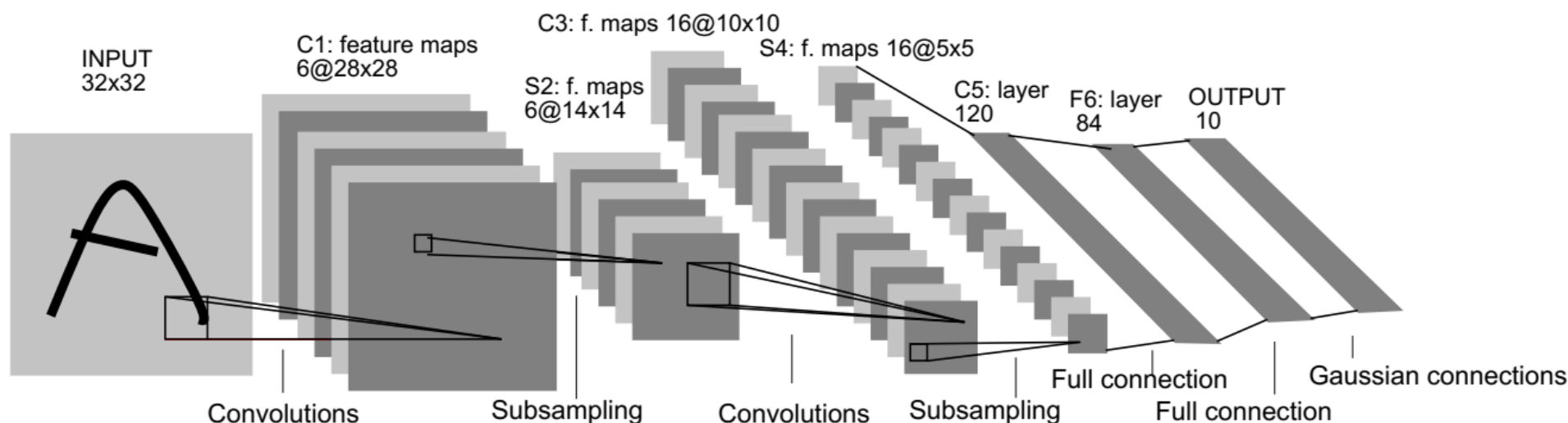
```
Epoch 1/10
500/500 [=====] - 5s 11ms/step - loss: 3.2766 - accuracy: 0.1728 - val_loss: 2.3051 - val_accuracy: 0.1016
Epoch 2/10
500/500 [=====] - 5s 10ms/step - loss: 2.2956 - accuracy: 0.1169 - val_loss: 2.3022 - val_accuracy: 0.1034
Epoch 3/10
500/500 [=====] - 5s 10ms/step - loss: 2.2843 - accuracy: 0.1263 - val_loss: 2.2962 - val_accuracy: 0.1128
Epoch 4/10
500/500 [=====] - 5s 10ms/step - loss: 2.2708 - accuracy: 0.1405 - val_loss: 2.2218 - val_accuracy: 0.2192
Epoch 5/10
500/500 [=====] - 5s 10ms/step - loss: 1.9456 - accuracy: 0.2968 - val_loss: 1.7332 - val_accuracy: 0.3848
Epoch 6/10
500/500 [=====] - 5s 10ms/step - loss: 1.6547 - accuracy: 0.4096 - val_loss: 1.5550 - val_accuracy: 0.4618
Epoch 7/10
500/500 [=====] - 5s 11ms/step - loss: 1.3392 - accuracy: 0.5275 - val_loss: 1.3940 - val_accuracy: 0.5233
Epoch 8/10
500/500 [=====] - 5s 10ms/step - loss: 1.1706 - accuracy: 0.5935 - val_loss: 1.3571 - val_accuracy: 0.5468
Epoch 9/10
500/500 [=====] - 5s 10ms/step - loss: 1.0184 - accuracy: 0.6461 - val_loss: 1.3662 - val_accuracy: 0.5593
Epoch 10/10
500/500 [=====] - 5s 11ms/step - loss: 0.9017 - accuracy: 0.6888 - val_loss: 1.3725 - val_accuracy: 0.5742
```

# شبکه‌های کانولوشنی برای دسته‌بندی



# شبکه LeNet-5

- شبکه LeNet-5 در سال ۱۹۹۸ برای شناسایی اعداد و حروف دستنویس پیشنهاد شد
- این شبکه تنها دارای ۵ لایه آموزشی است: ۲ لایه کانولوشنی و ۳ لایه کاملاً متصل



<https://colab.research.google.com/drive/1Fv3SuHgquWt5Yi4rbp4wKkrqsGQX2g9K?usp=sharing>

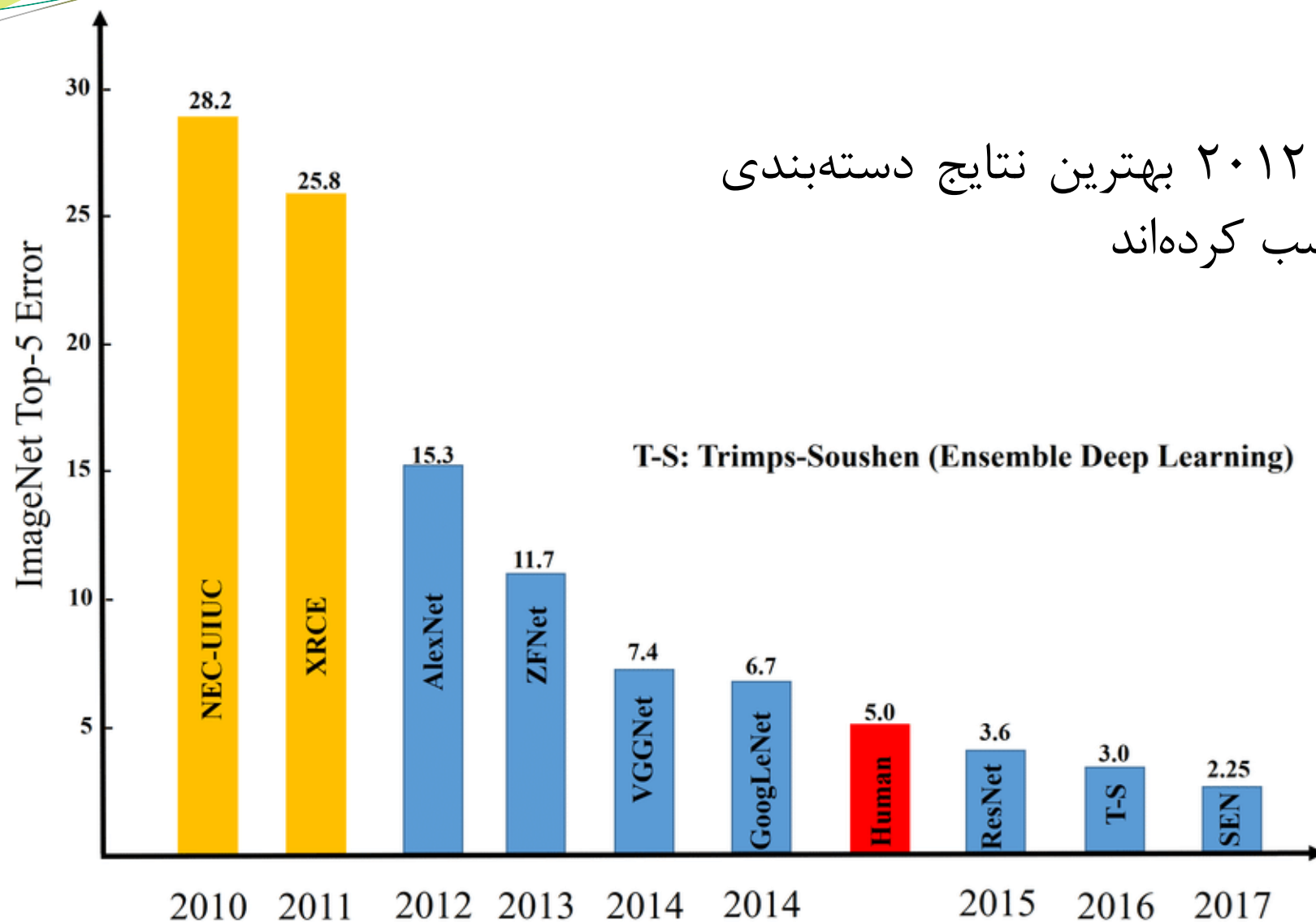
[https://colab.research.google.com/drive/1VzHF2z9hek1z9wLUw\\_3io6hpXW-vs-TW?usp=sharing](https://colab.research.google.com/drive/1VzHF2z9hek1z9wLUw_3io6hpXW-vs-TW?usp=sharing)

معماری‌های CNN

CNN Architectures

# معماری‌های CNN

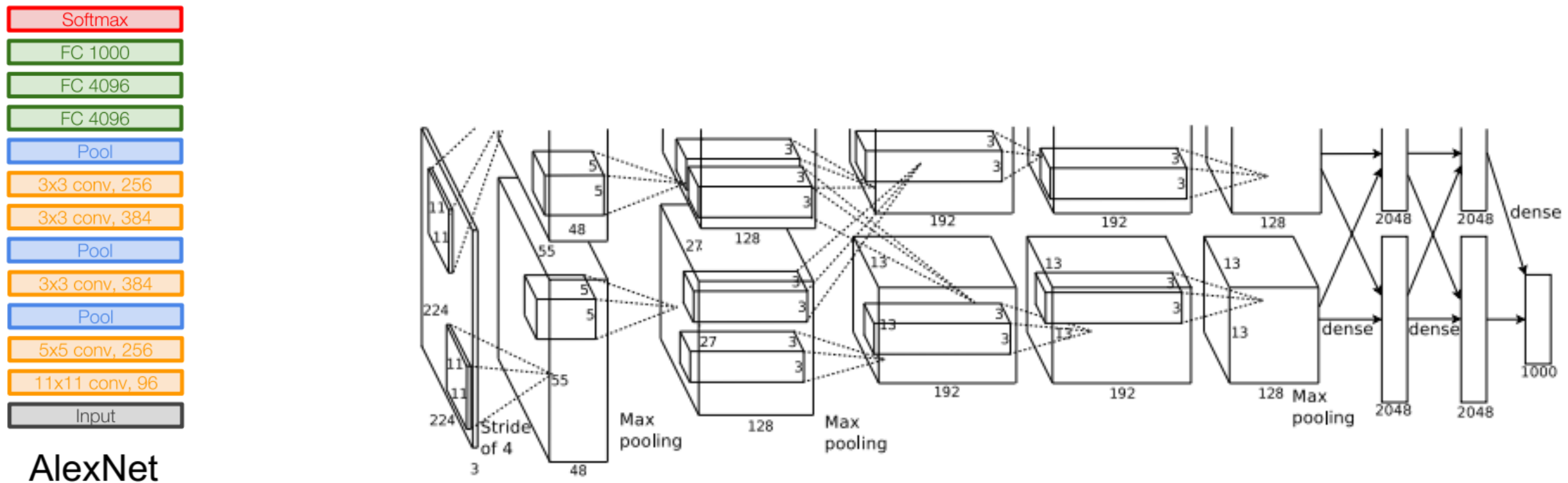
- معماری‌های مختلف CNN از سال ۲۰۱۲ بهترین نتایج دسته‌بندی تصویر در چالش ImageNet را کسب کرده‌اند





# AlexNet

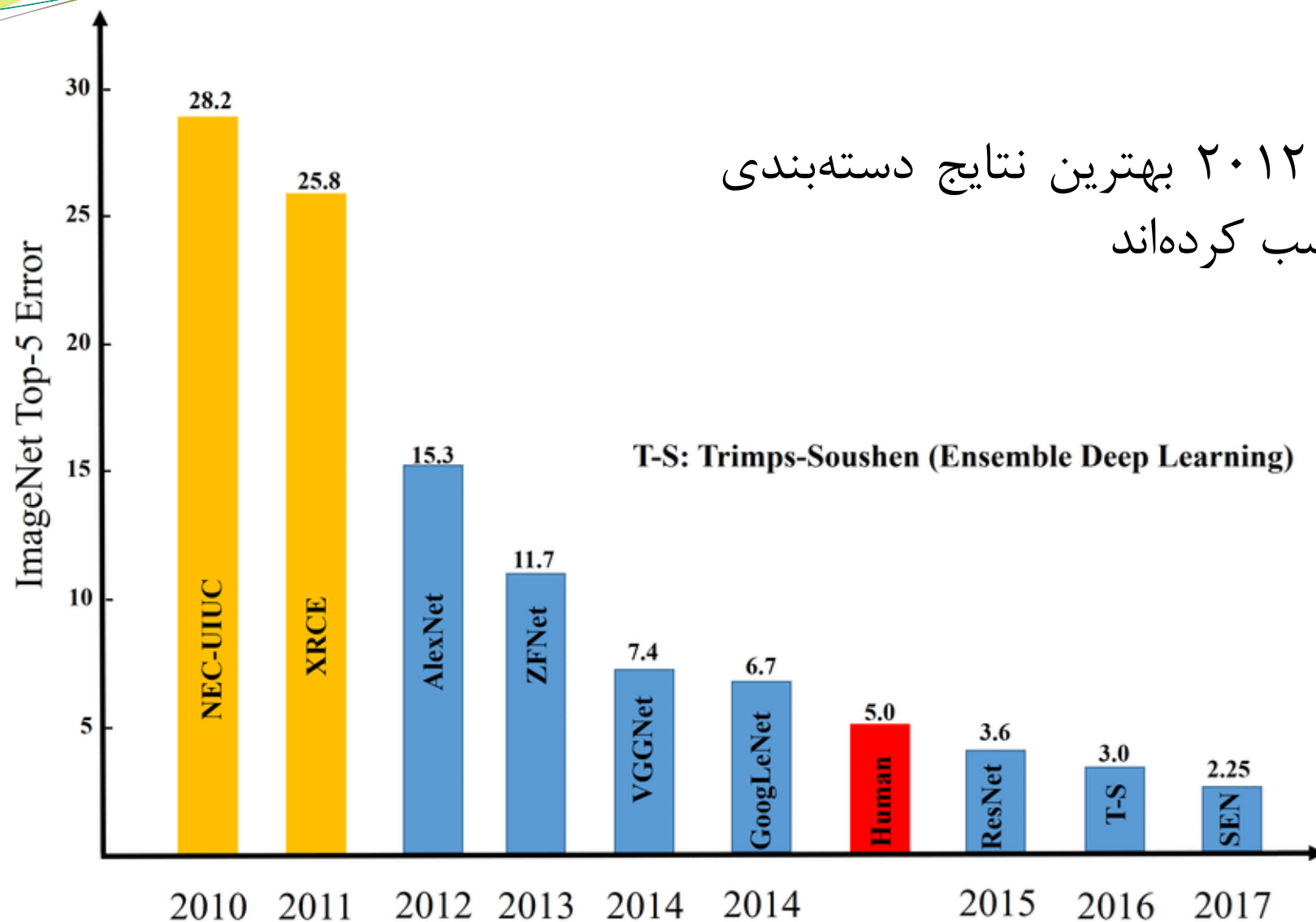
- شبکه AlexNet یک شبکه دارای ۸ لایه آموزشی است که در سال ۲۰۱۲ پیشنهاد شد و توانست خطای top-5 در چالش ILSVRC'12 را به ۱۵.۳٪ کاهش دهد



AlexNet

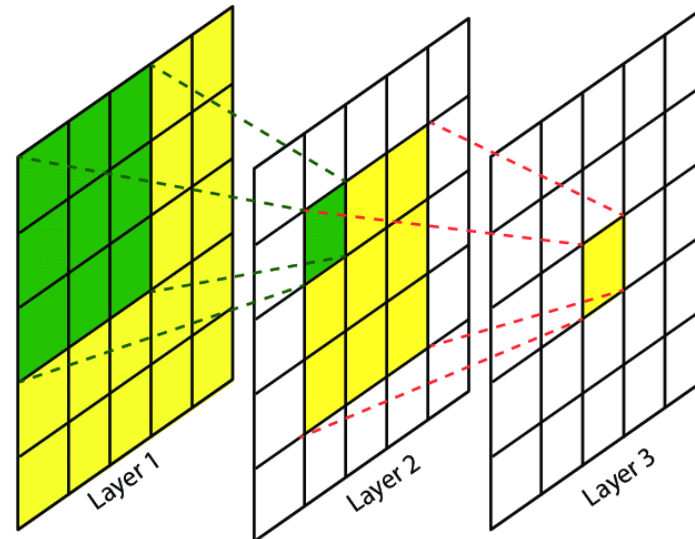
# معماری‌های CNN

- معماری‌های مختلف CNN از سال ۲۰۱۲ بهترین نتایج دسته‌بندی تصویر در چالش ImageNet را کسب کرده‌اند



# VGG

- معماری VGG در سال ۲۰۱۴ تیم دوم مسابقه ILSVRC'14 شد
- فیلترهای کوچکتر و لایه‌های بیشتر
- در این معماری ابعاد تمام فیلترها  $3 \times 3$  با گام ۱ است و تعداد لایه‌های آموزشی از ۸ لایه به ۱۶ و ۱۹ لایه افزایش یافته است



# GoogLeNet

- شبکه GoogLeNet برنده مسابقه ILSVRC'14 با خطای ۶.۷٪ شد
- شبکه عمیق تر با پارامترهای کمتر
- فیلترهای هم‌عرض (موازی) تحت عنوان Inception Module معرفی شدند
  - کانولوشن‌های دارای ابعاد مختلف
  - عملیات Pooling
- سپس، خروجی تمام فیلترها به هم الحاق می‌شوند (در عمق)

