مبانی بینایی کامپیوتر

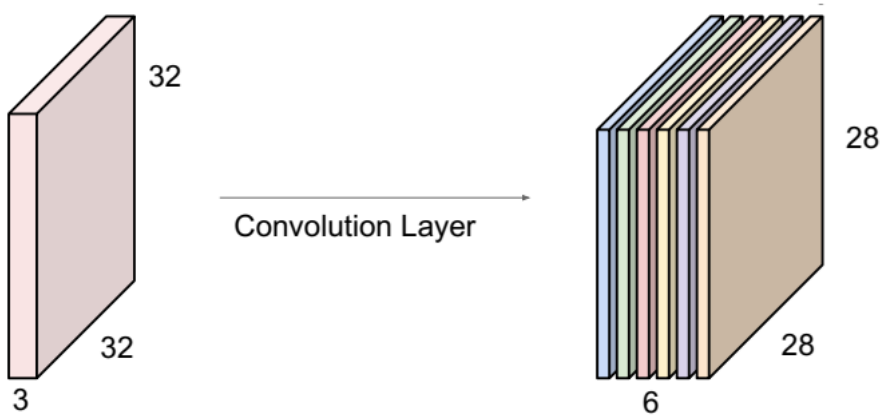
مدرس: محمدرضا محمدی

۱۴۰۱

شبکه‌های عصبی کانولوشنی

Convolutional Neural Networks

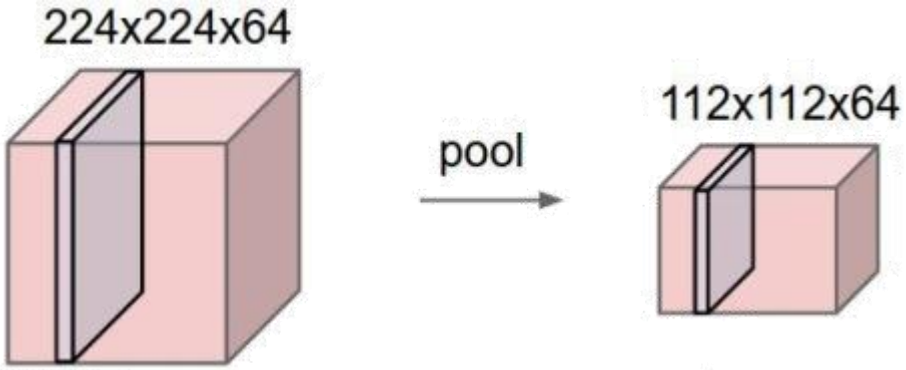
لایه کانولوشنی



- 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
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$
  - $D_2 = K$
- پارمترهای لایه کانولوشنی عبارتند از $F \cdot F \cdot D_1 \cdot K$ وزن فیلترها و K بایاس که باید آموزش ببینند

https://keras.io/api/layers/convolution_layers/convolution2d/

لایه Pooling



- ورودی یک حجم با ابعاد $W_1 \times H_1 \times D_1$ است
- ابرپارامترهای لایه Pooling عبارتند از:
  - نحوه تلفیق
  - 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

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

https://keras.io/api/layers/pooling_layers/max_pooling2d/ , https://keras.io/api/layers/pooling_layers/average_pooling2d/

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”

https://keras.io/api/layers/pooling_layers/max_pooling2d/ , https://keras.io/api/layers/pooling_layers/average_pooling2d/

مقایسه نتایج

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
```

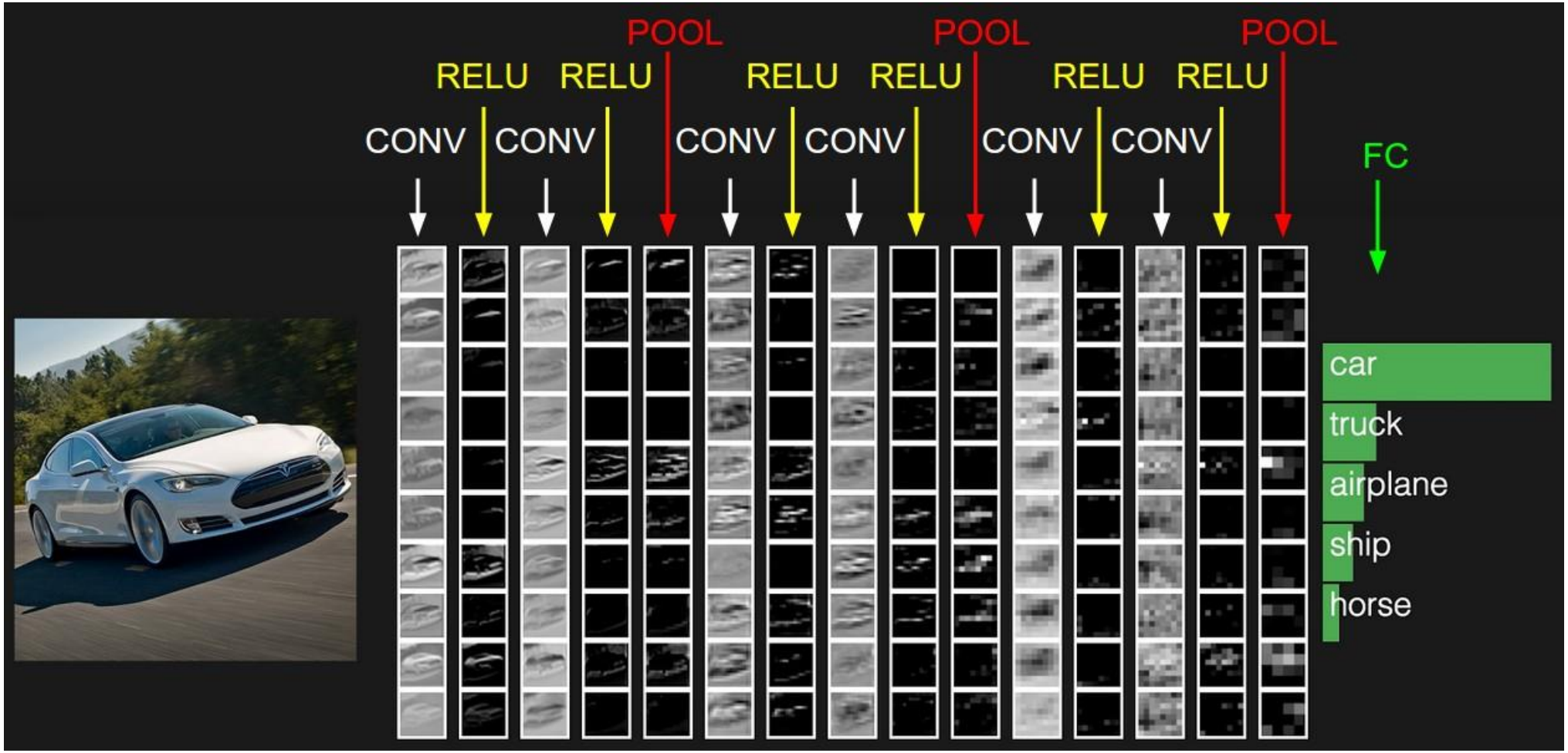
Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 30, 30, 128)	3584
conv2d_3 (Conv2D)	(None, 14, 14, 128)	147584
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 10)	250890

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
```

https://colab.research.google.com/drive/10ygqxT8V2_-0fHHLWaeICka6hnIv1Iao?usp=sharing

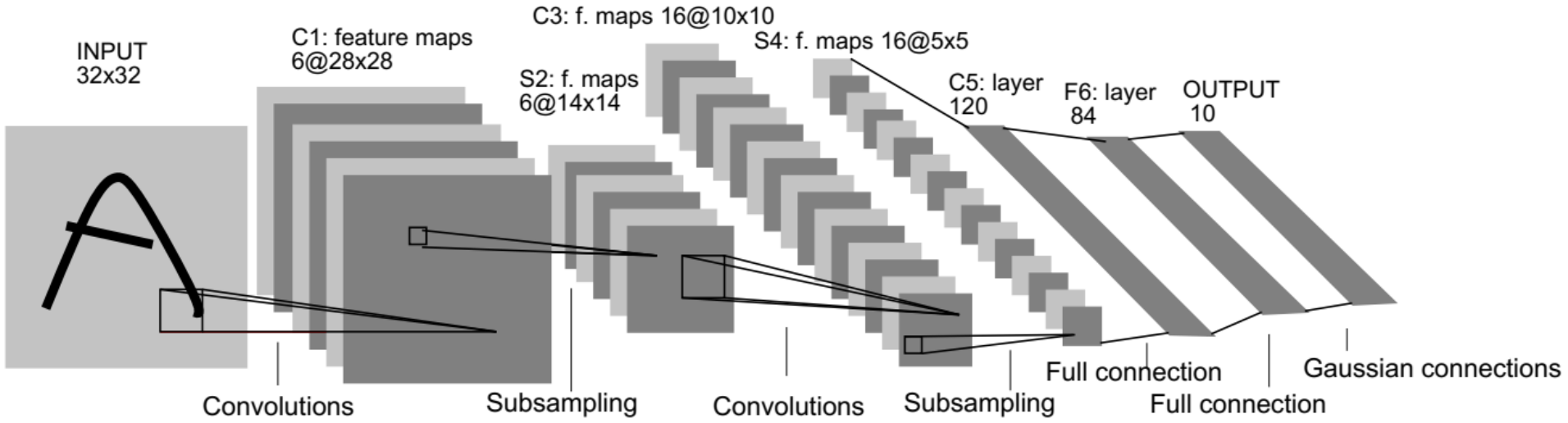
شبکه‌های کانولوشنی برای دسته‌بندی



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

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 کاملا متصل



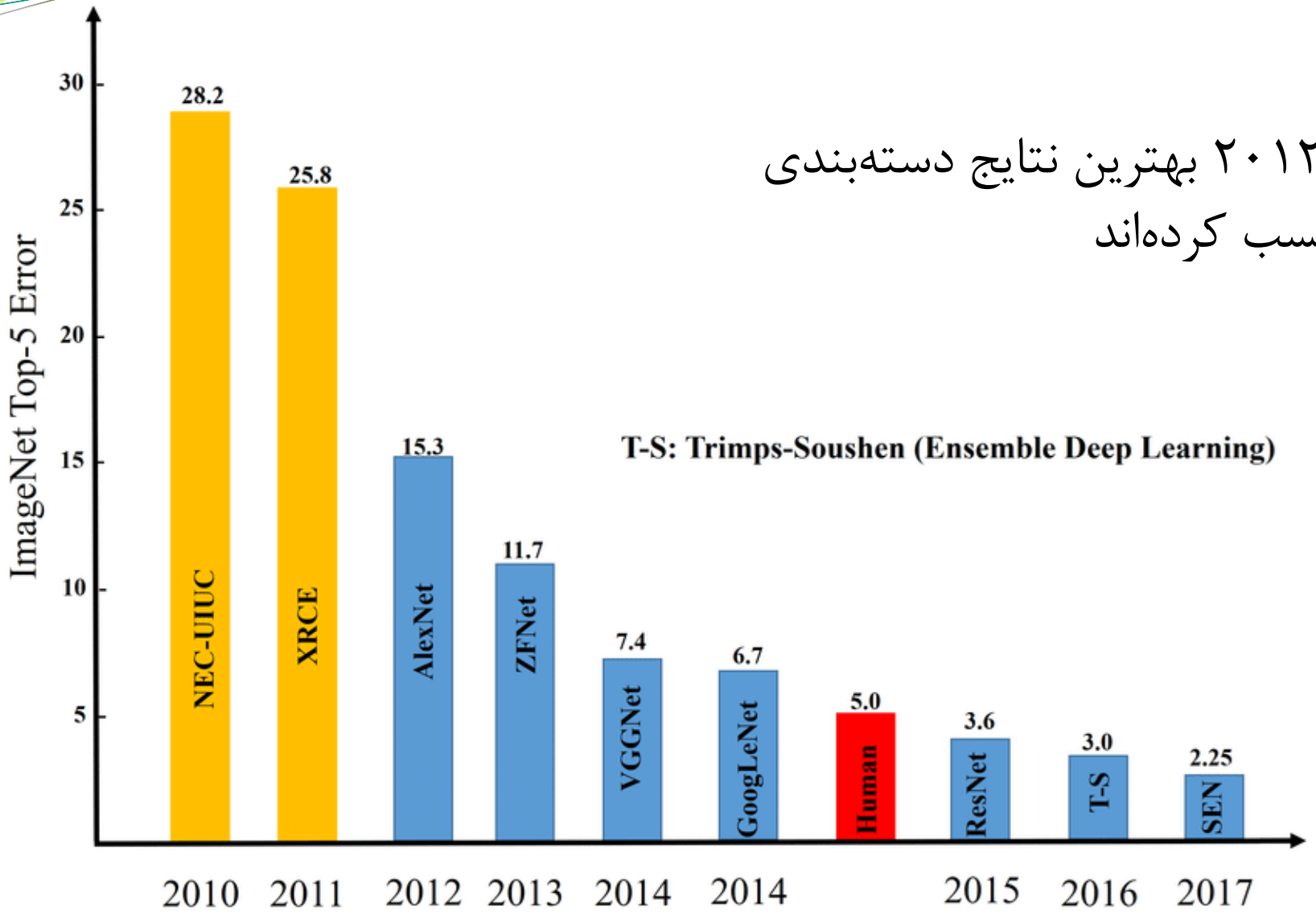
LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998)

معماری‌های CNN

CNN Architectures

معماری‌های CNN

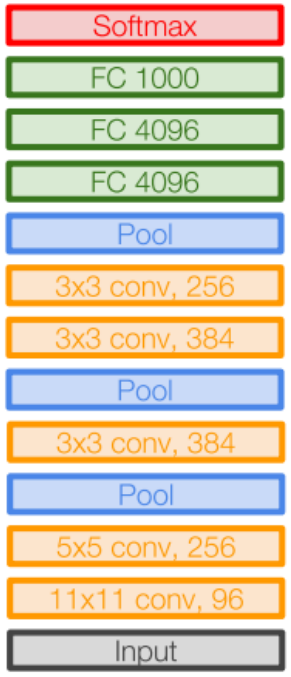
- معماری‌های مختلف CNN از سال ۲۰۱۲ بهترین نتایج دسته‌بندی تصویر در چالش ImageNet را کسب کرده‌اند

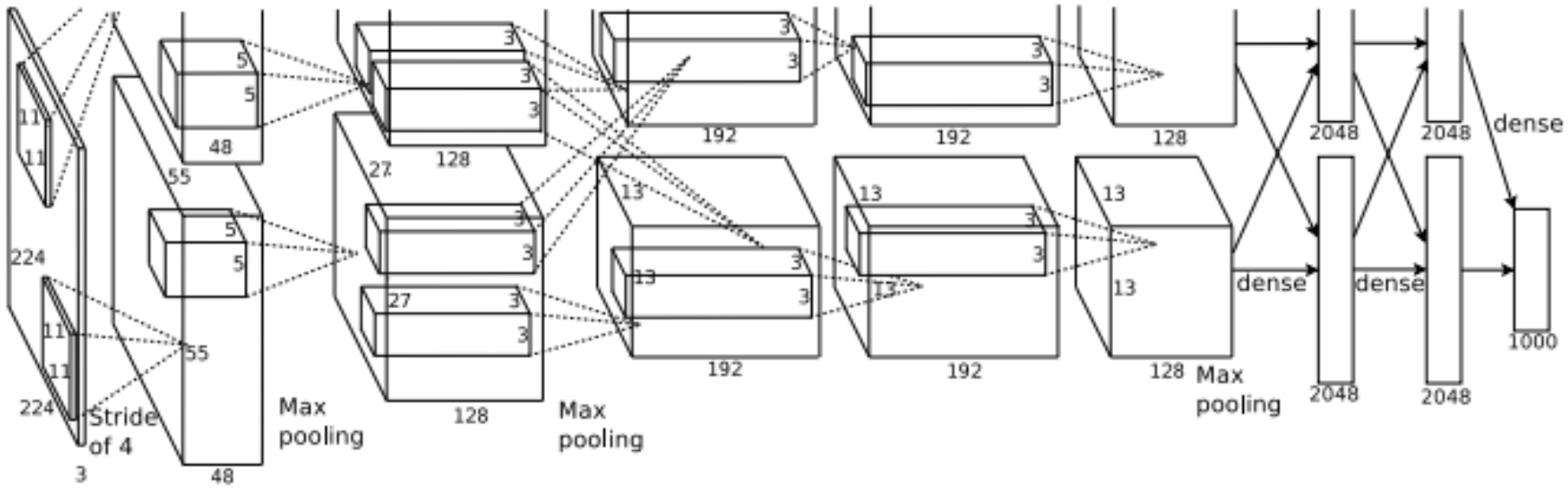


Alyafeai, Zaid, and Lahouari Ghouti. "A fully-automated deep learning pipeline for cervical cancer classification." *Expert Systems with Applications* (2020)

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

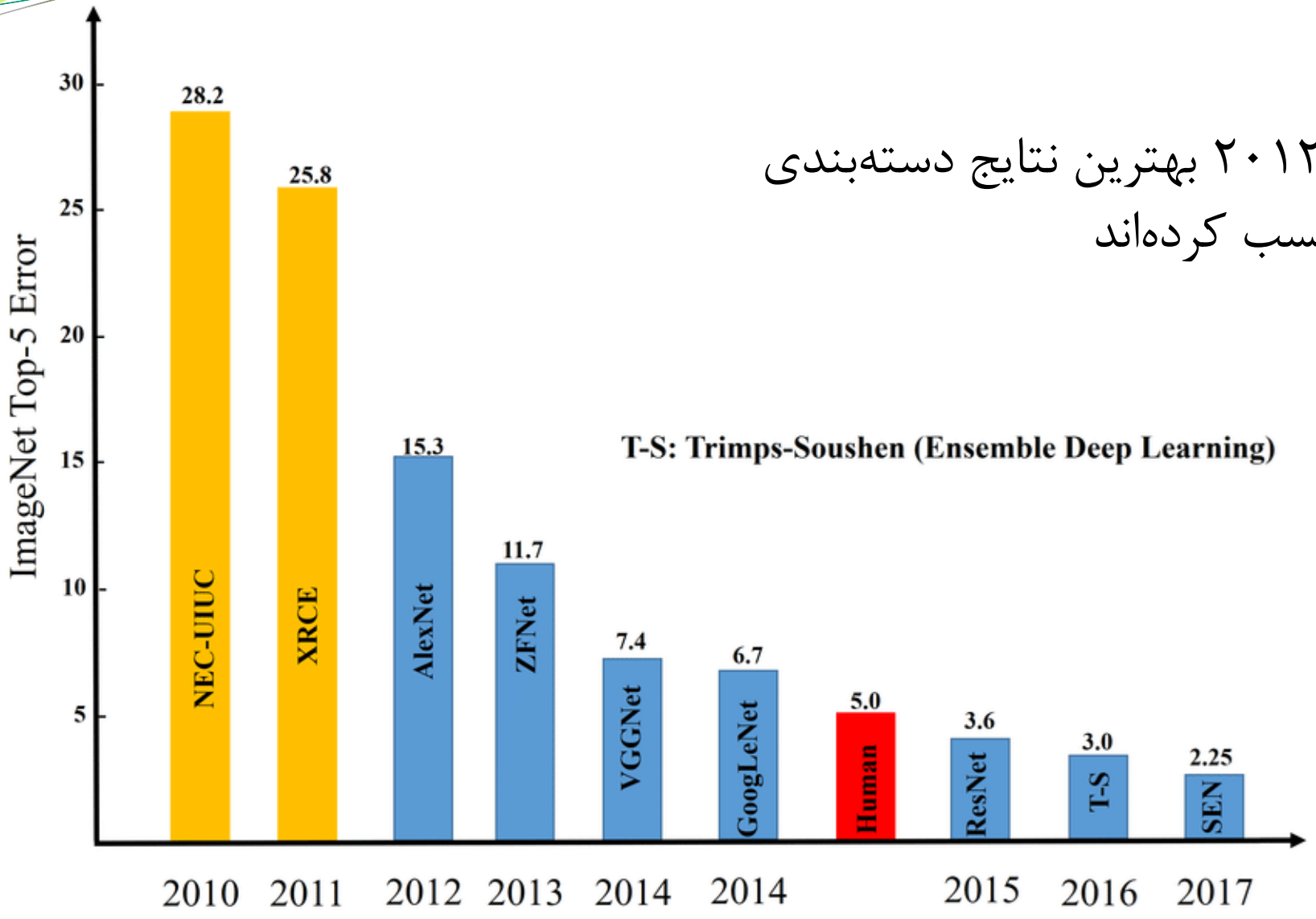




Hinton, Geoffrey E., Alex Krizhevsky, and Ilya Sutskever. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* (2012).

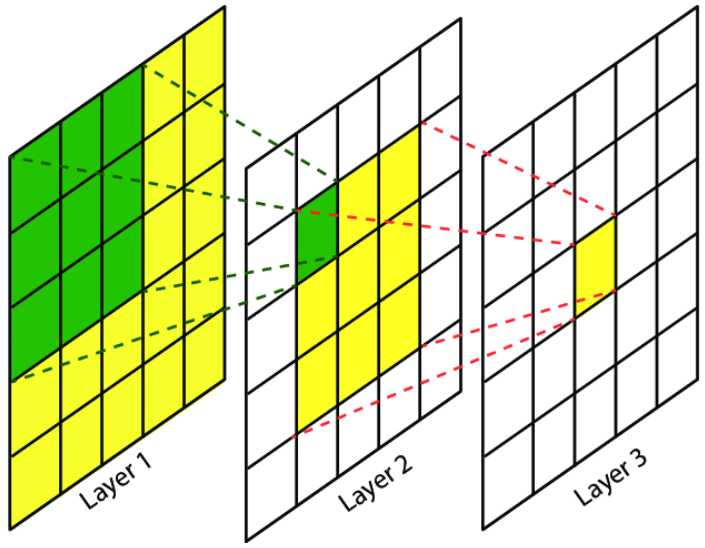
معماری‌های CNN

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



VGG

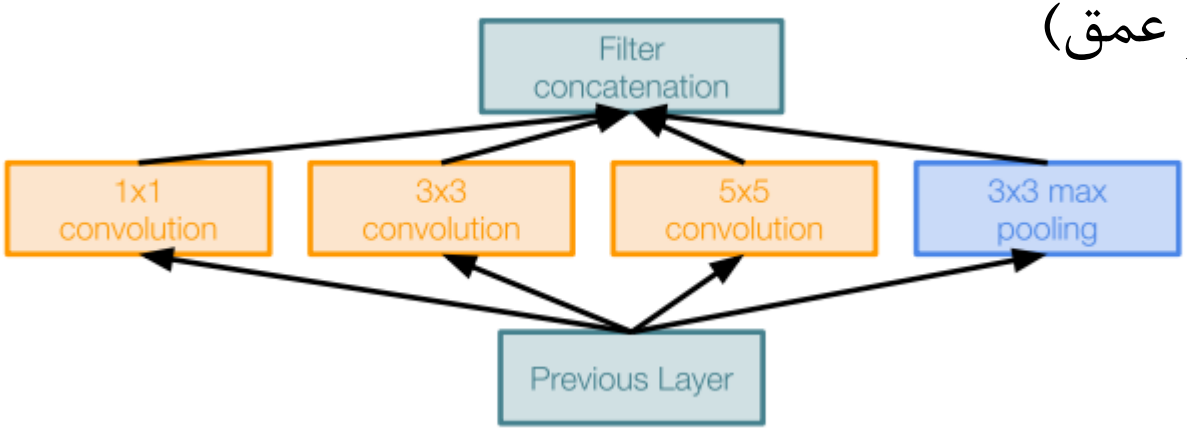
- معماری VGG در سال ۲۰۱۴ تیم دوم مسابقه ILSVRC’14 شد
- فیلترهای کوچکتر و لایه‌های بیشتر
- در این معماری ابعاد تمام فیلترها ۳×۳ با گام ۱ است و تعداد لایه‌های آموزشی از ۸ لایه به ۱۶ و ۱۹ لایه افزایش یافته است



Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

GoogLeNet

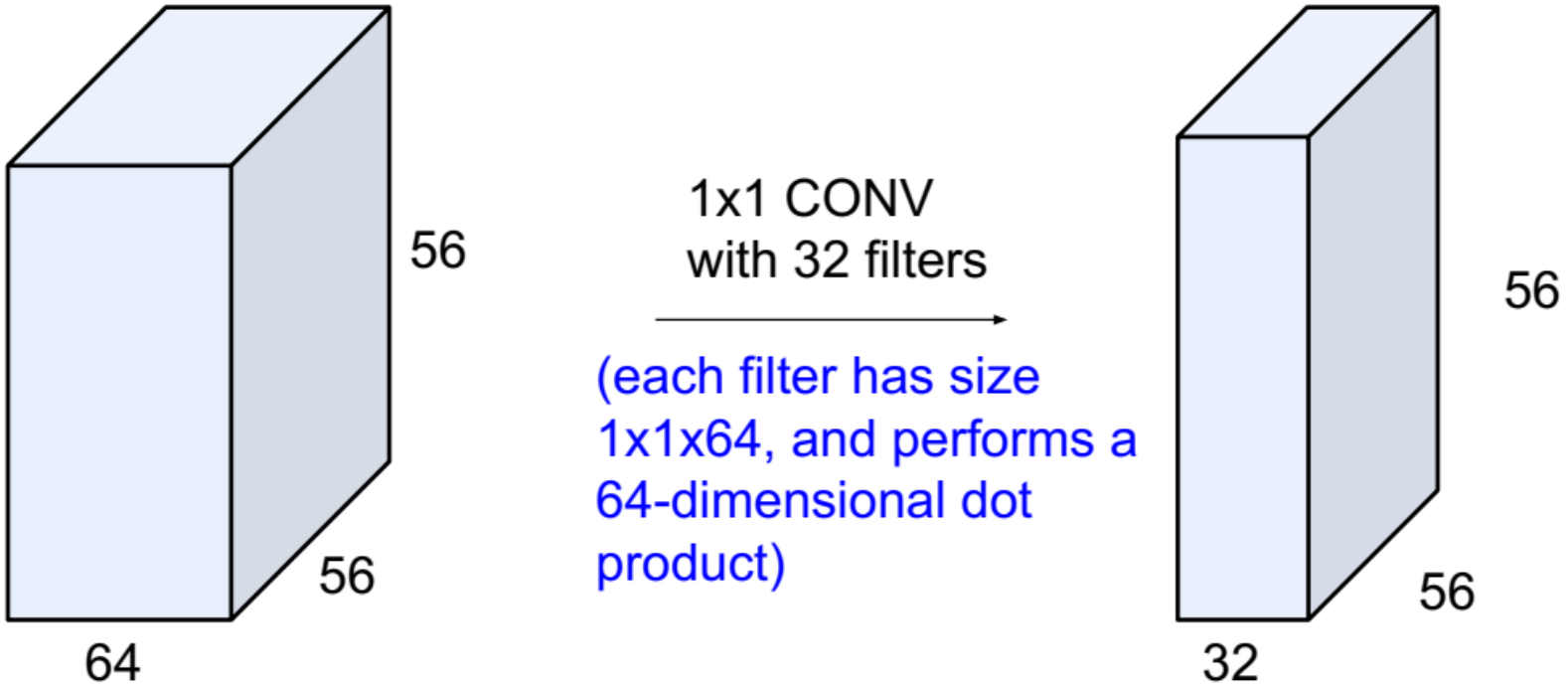
- شبکه GoogLeNet برنده مسابقه ILSVRC’14 با خطای ۶.۷٪ شد
- شبکه عمیق‌تر با پارامترهای کمتر
- فیلترهای هم‌عرض (موازی) تحت عنوان Inception Module معرفی شدند
  - کانولوشن‌های دارای ابعاد مختلف
  - عملیات Pooling
- سپس، خروجی تمام فیلترها به هم الحاق می‌شوند (در عمق)



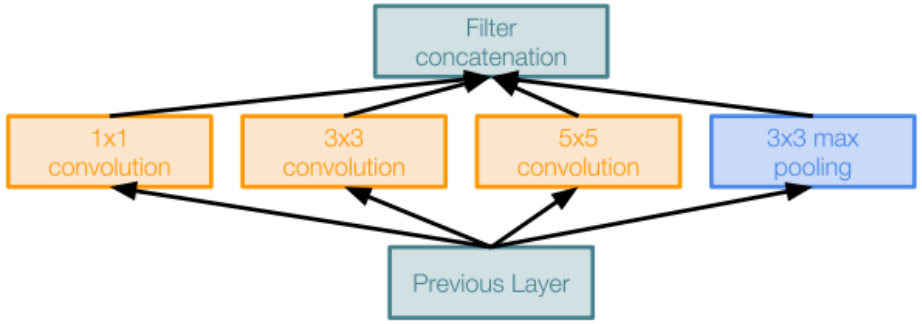
Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. (2015).

کانولوشن ۱×۱

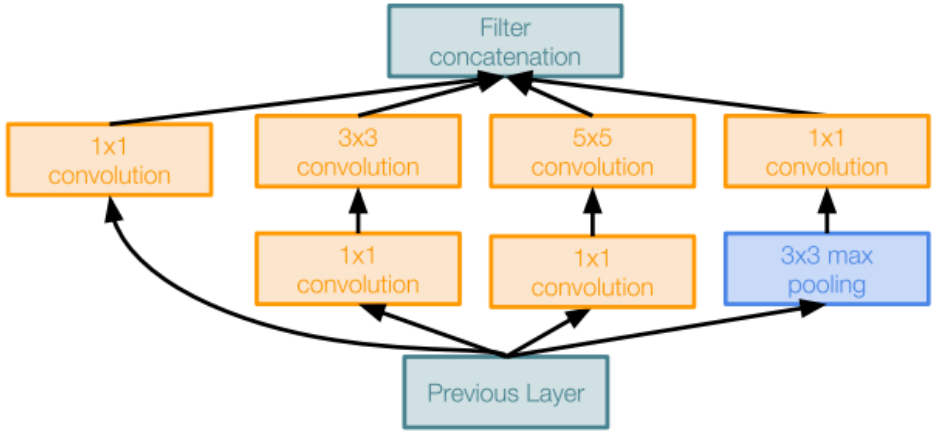
- ابعاد مکانی حفظ می‌شود و عمق کاهش می‌یابد



Inception Module



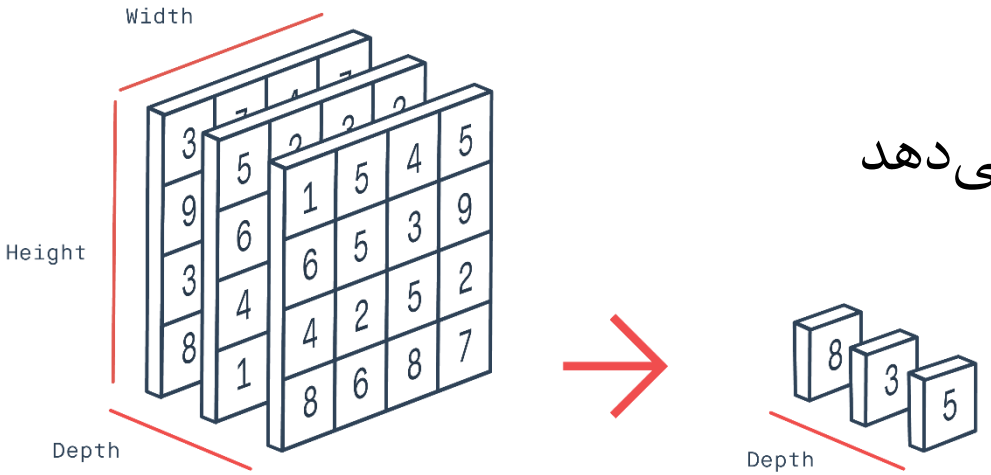
Naive Inception module

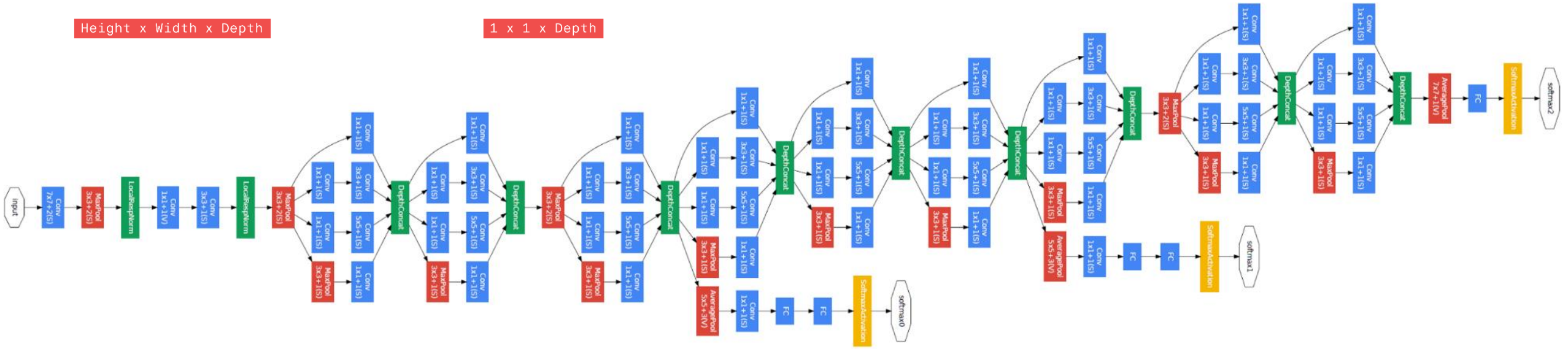


Inception module with dimension reduction

Global Average Pooling

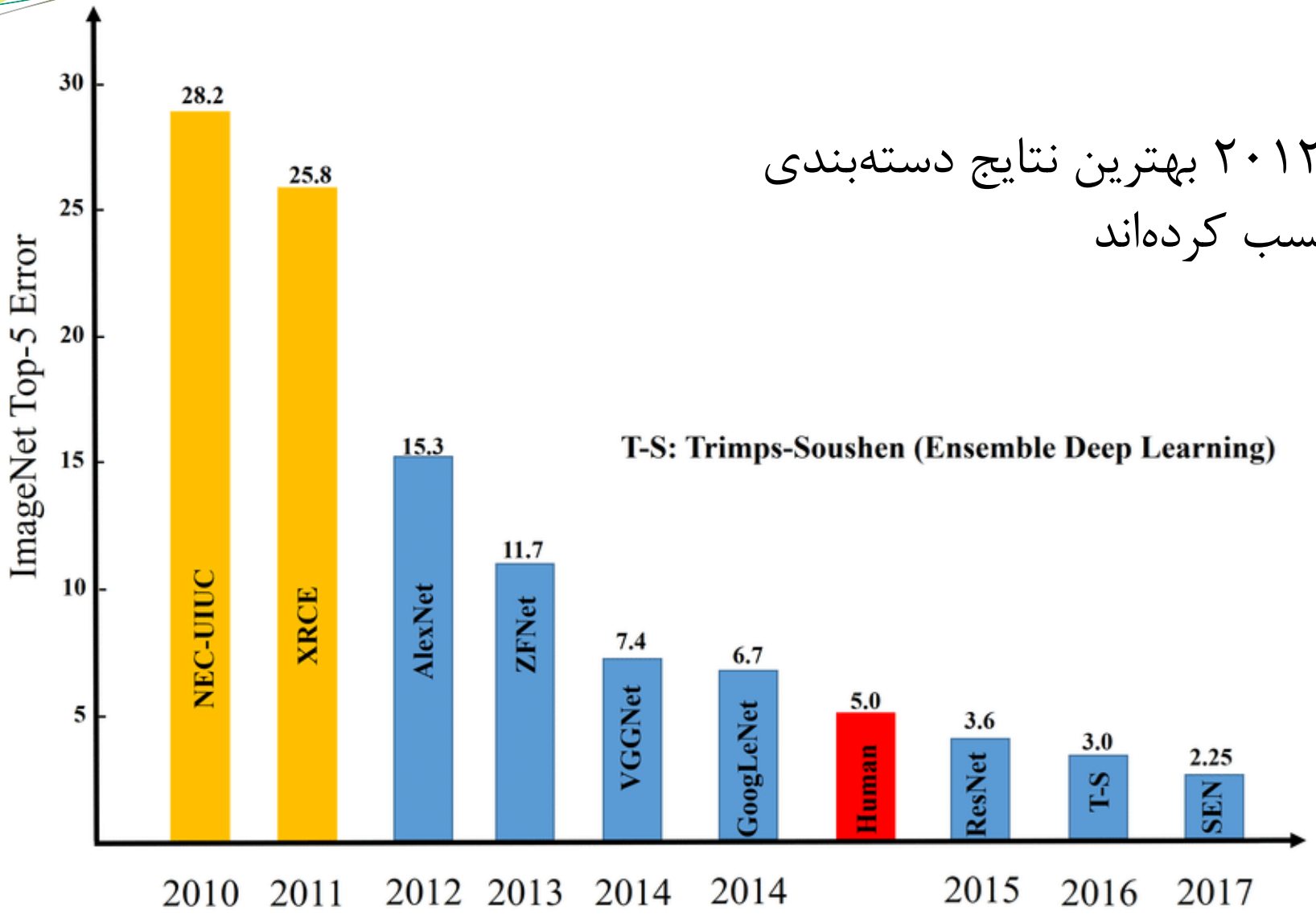
- بر خلاف لایه Flatten، تعداد پارامترهای مدل را بسیار کاهش می‌دهد





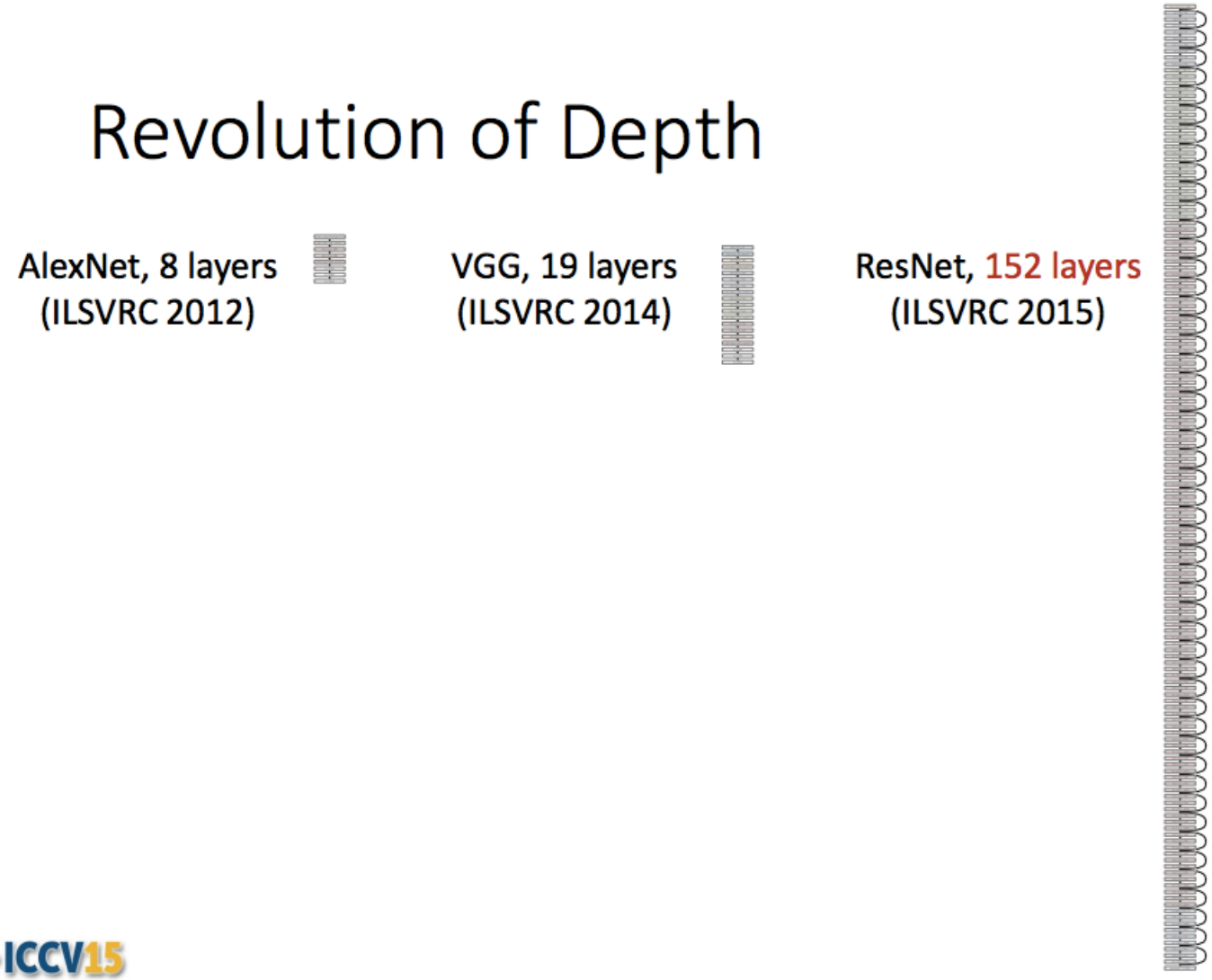
معماری‌های CNN

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



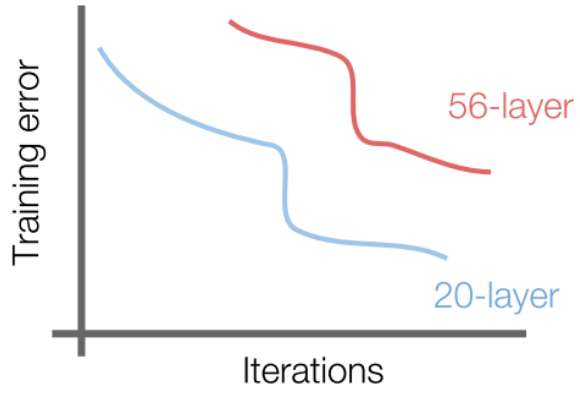
ResNet

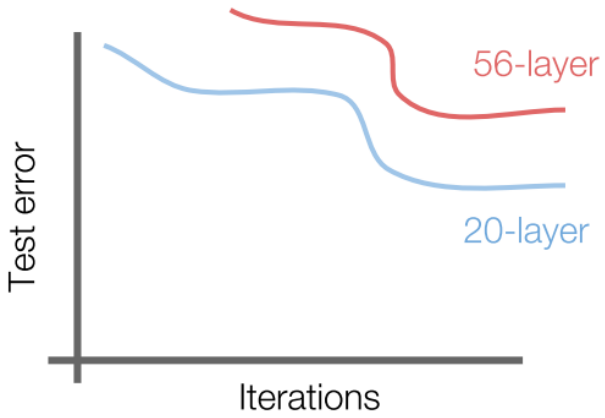
- شبکه ResNet برنده مسابقه ILSVRC’15 با خطای ۳.۵۷٪ شد
- با ۱۵۲ لایه، انقلابی در عمق شبکه‌های کانولوشنی به وجود آورد



He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. (2016).

ResNet





- اگر تعداد لایه‌های کانولوشنی ساده را بسیار زیاد کنیم چه اتفاقی می‌افتد؟
- چرا شبکه عمیق‌تر هم در آموزش و هم در آزمون عملکرد ضعیف‌تری دارد؟
  - البته مشکل از overfitting نیست!
- فرضیه: مشکل در مسئله بهینه‌سازی است
  - بهینه‌سازی مدل‌های عمیق‌تر دشوارتر است
- عملکرد مدل‌های عمیق‌تر باید حداقل به خوبی مدل‌های با عمق کمتر باشد
  - می‌توان وزن‌های مدل کم‌عمق را به لایه‌های نخست شبکه عمیق کپی کرد و لایه‌های اضافی را به گونه‌ای تنظیم کرد که نگاشت همانی را انجام دهند
- ایده ResNet آن است که لایه‌های شبکه بجای آموختن نگاشت مطلوب، باقی‌مانده آن را یاد بگیرند