

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



یادگیری عمیق

جلسه ۱۷

معماری‌های گوناگون شبکه‌های عصبی کانوولوشنال

Various Architectures of Convolutional Neural Networks

کاظم فولادی قلعه

دانشکده مهندسی، پردیس فارابی

دانشگاه تهران

<http://courses.fouladi.ir/deep>

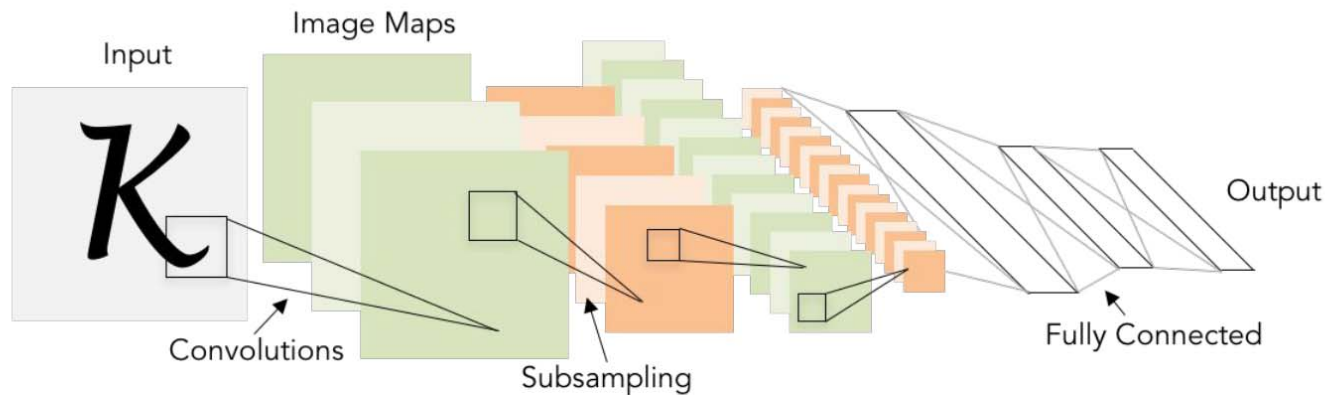
معماری‌های گوناگون
شبکه‌های عصبی کانوولوشنال



معماری‌های پایه

LeNet-5

[LeCun et al., 1998]



Conv filters were 5×5 , applied at stride 1
 Subsampling (Pooling) layers were 2×2 applied at stride 2
 i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Abstract—

Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional Neural Networks, that are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques.

Real-life document recognition systems are composed of multiple modules including field extraction, segmentation, recognition, and language modeling. A new learning paradigm, called Graph Transformer Networks (GTN), allows such multi-module systems to be trained globally using Gradient-Based methods so as to minimize an overall performance measure.

Two systems for on-line handwriting recognition are described. Experiments demonstrate the advantage of global training, and the flexibility of Graph Transformer Networks.

A Graph Transformer Network for reading bank check is also described. It uses Convolutional Neural Network character recognizers combined with global training techniques to provides record accuracy on business and personal checks. It is deployed commercially and reads several million checks per day.

Keywords—Neural Networks, OCR, Document Recognition, Machine Learning, Gradient-Based Learning, Convolutional Neural Networks, Graph Transformer Networks, Finite State Transducers.

NOMENCLATURE

- GT Graph transformer.
- GTN Graph transformer network.
- HMM Hidden Markov model.
- HOS Heuristic oversegmentation.
- K-NN K-nearest neighbor.
- NN Neural network.
- OCR Optical character recognition.
- PCA Principal component analysis.
- RBF Radial basis function.
- RS-SVM Reduced-set support vector method.
- SDNN Space displacement neural network.
- SVM Support vector method.
- TDNN Time delay neural network.
- V-SVM Virtual support vector method.

The authors are with the Speech and Image Processing Services Research Laboratory, AT&T Labs-Research, 100 Schulz Drive Red Bank, NJ 07701. E-mail: {yann.lecun,yoshua,haffner}@research.att.com. Yoshua Bengio is also with the Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, C.P. 6128 Succ. Centre-Ville, 2920 Chemin de la Tour, Montréal, Québec, Canada H3C 3J7.

I. INTRODUCTION

Over the last several years, machine learning techniques, particularly when applied to neural networks, have played an increasingly important role in the design of pattern recognition systems. In fact, it could be argued that the availability of learning techniques has been a crucial factor in the recent success of pattern recognition applications such as continuous speech recognition and handwriting recognition.

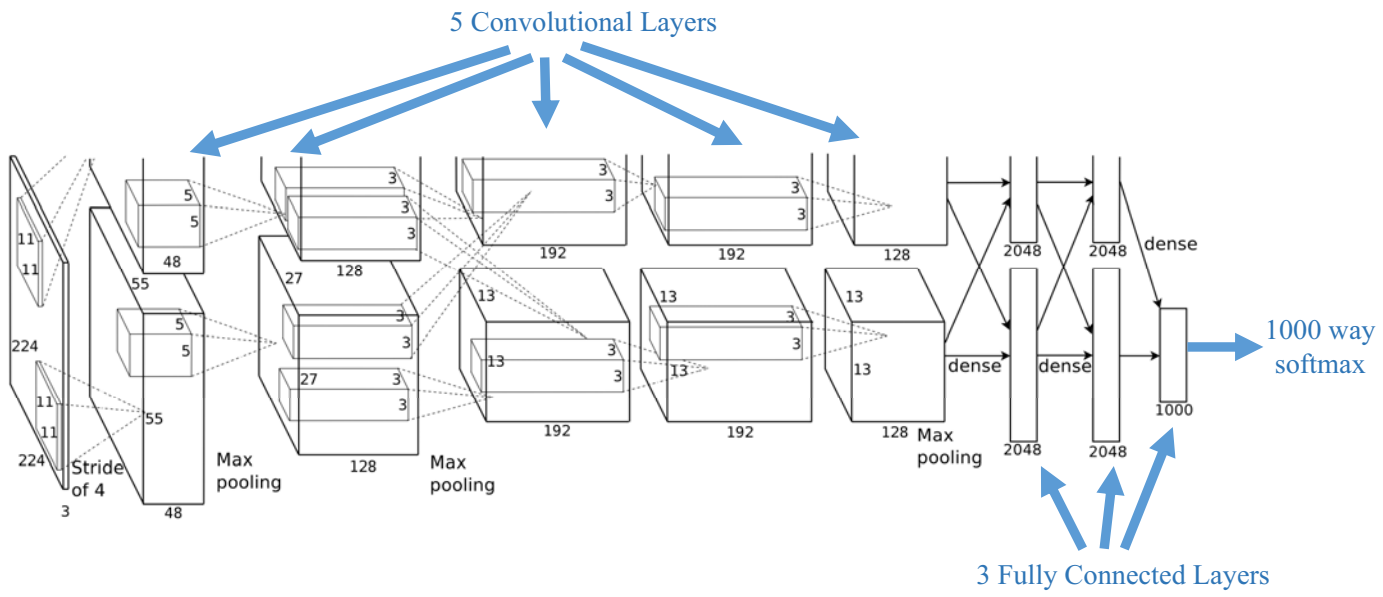
The main message of this paper is that better pattern recognition systems can be built by relying more on automatic learning, and less on hand-designed heuristics. This is made possible by recent progress in machine learning and computer technology. Using character recognition as a case study, we show that hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images. Using document understanding as a case study, we show that the traditional way of building recognition systems by manually integrating individually designed modules can be replaced by a unified and well-principled design paradigm, called *Graph Transformer Networks*, that allows training all the modules to optimize a global performance criterion.

Since the early days of pattern recognition it has been known that the variability and richness of natural data, be it speech, glyphs, or other types of patterns, make it almost impossible to build an accurate recognition system entirely by hand. Consequently, most pattern recognition systems are built using a combination of automatic learning techniques and hand-crafted algorithms. The usual method of recognizing individual patterns consists in dividing the system into two main modules shown in figure 1. The first module, called the feature extractor, transforms the input patterns so that they can be represented by low-dimensional vectors or short strings of symbols that (a) can be easily matched or compared, and (b) are relatively invariant with respect to transformations and distortions of the input patterns that do not change their nature. The feature extractor contains most of the prior knowledge and is rather specific to the task. It is also the focus of most of the design effort, because it is often entirely hand-crafted. The classifier, on the other hand, is often general-purpose and trainable. One of the main problems with this approach is that the recognition accuracy is largely determined by the ability of the designer to come up with an appropriate set of features. This turns out to be a daunting task which, unfortunately, must be redone for each new problem. A large amount of the pattern recognition literature is devoted to describing and comparing the relative

AlexNet

معماری

ALEXNET: ARCHITECTURE



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet ILSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

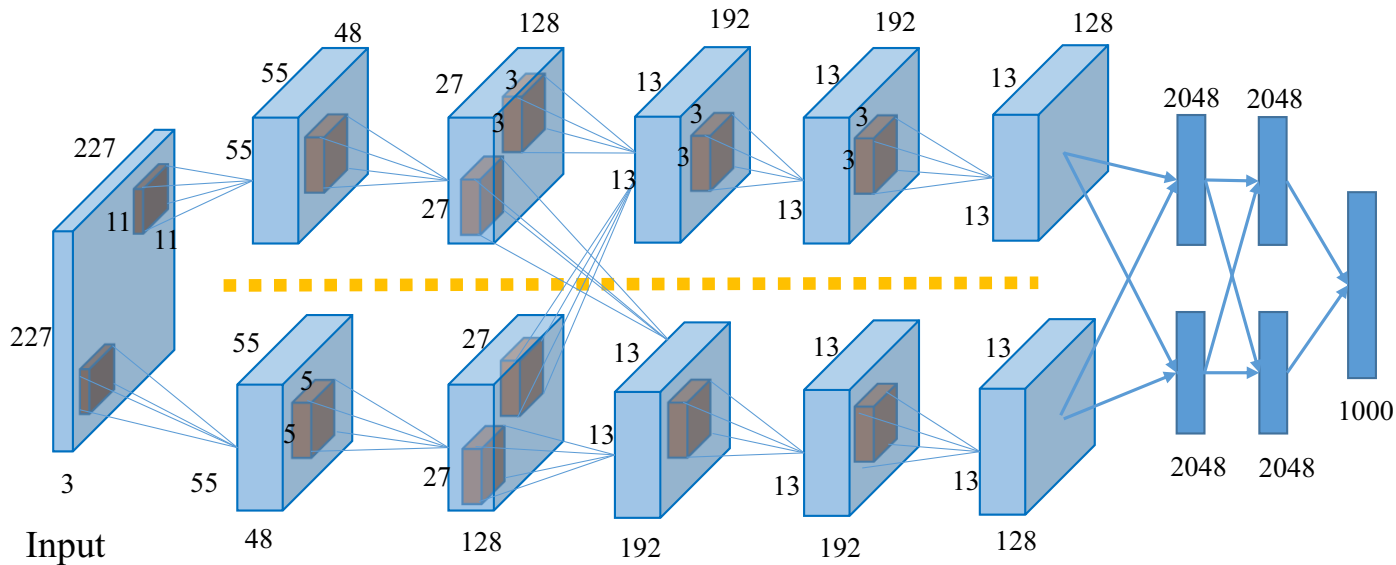
Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don't have. Convolutional neural networks (CNNs) constitute one such class of models [16, 11, 13, 18, 15, 22, 26]. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse.

AlexNet

معماری: برای طبقه‌بندی تصاویر ImageNet

ALEXNET: ARCHITECTURE

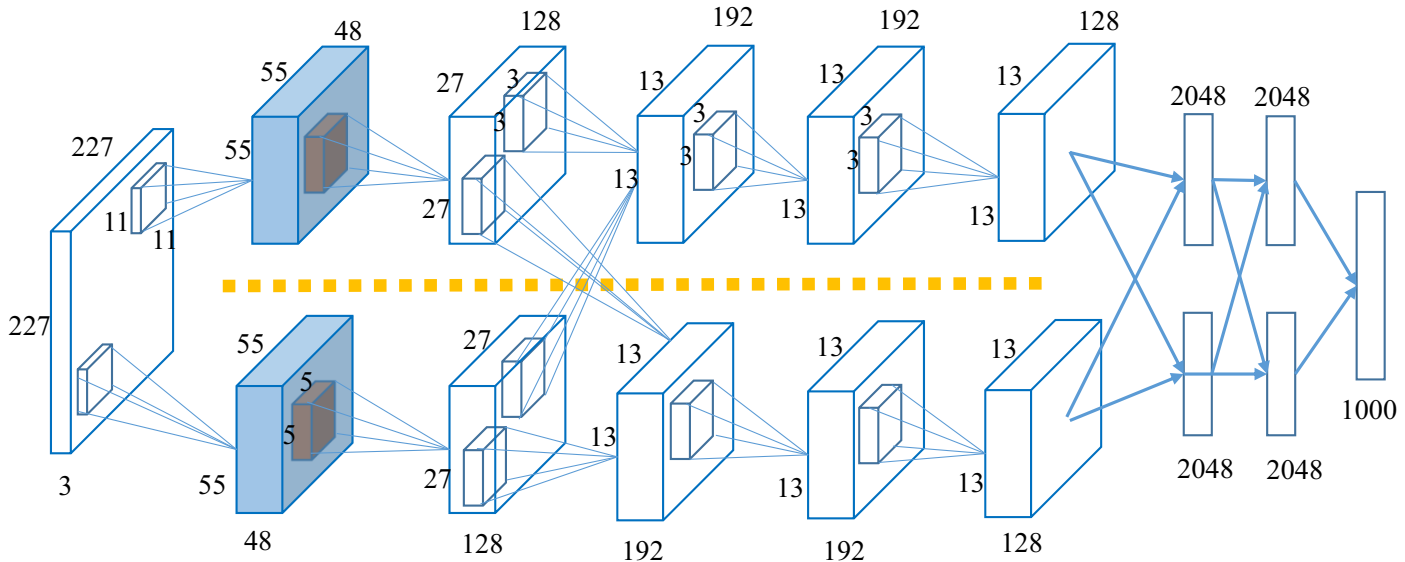


AlexNet

معماری: تعداد پارامترها

ALEXNET: ARCHITECTURE

- $55 \times 55 \times 96 = 290,400$ neurons, each having $11 \times 11 \times 3 = 363$ weights + 1 bias
- $290400 \times 364 = 105,705,600$ parameters in first layer alone if fully connected.



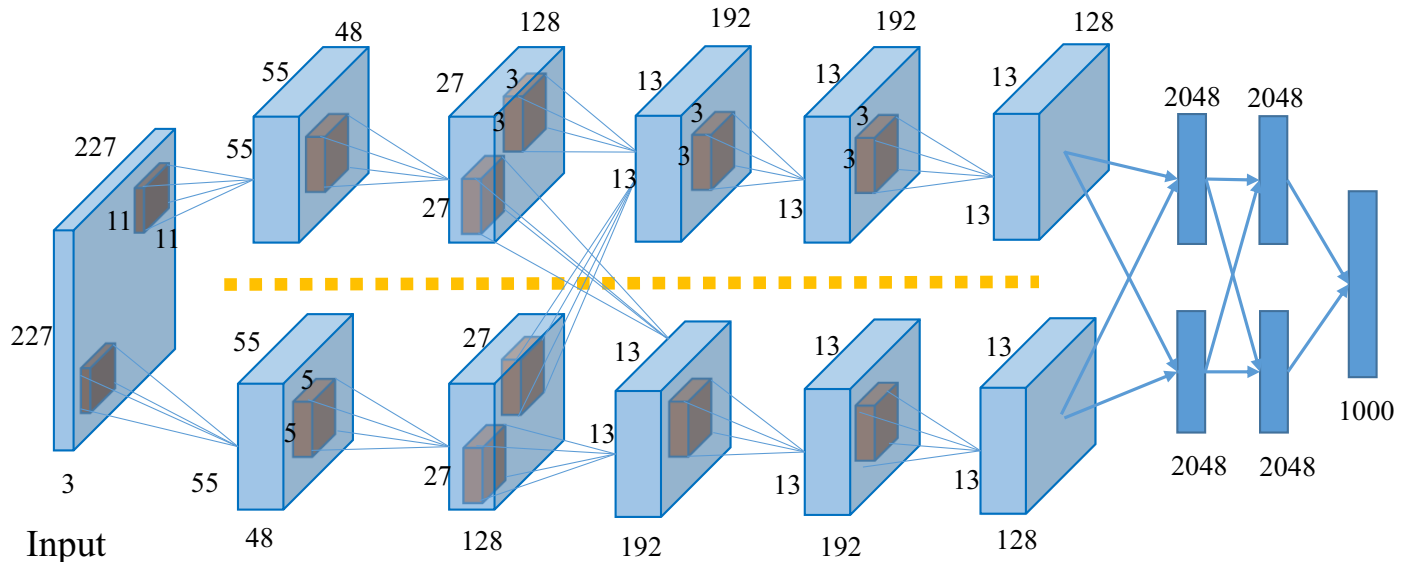
AlexNet

معماری: جزئیات لایه‌ها

ALEXNET: ARCHITECTURE

for layer details refer to:

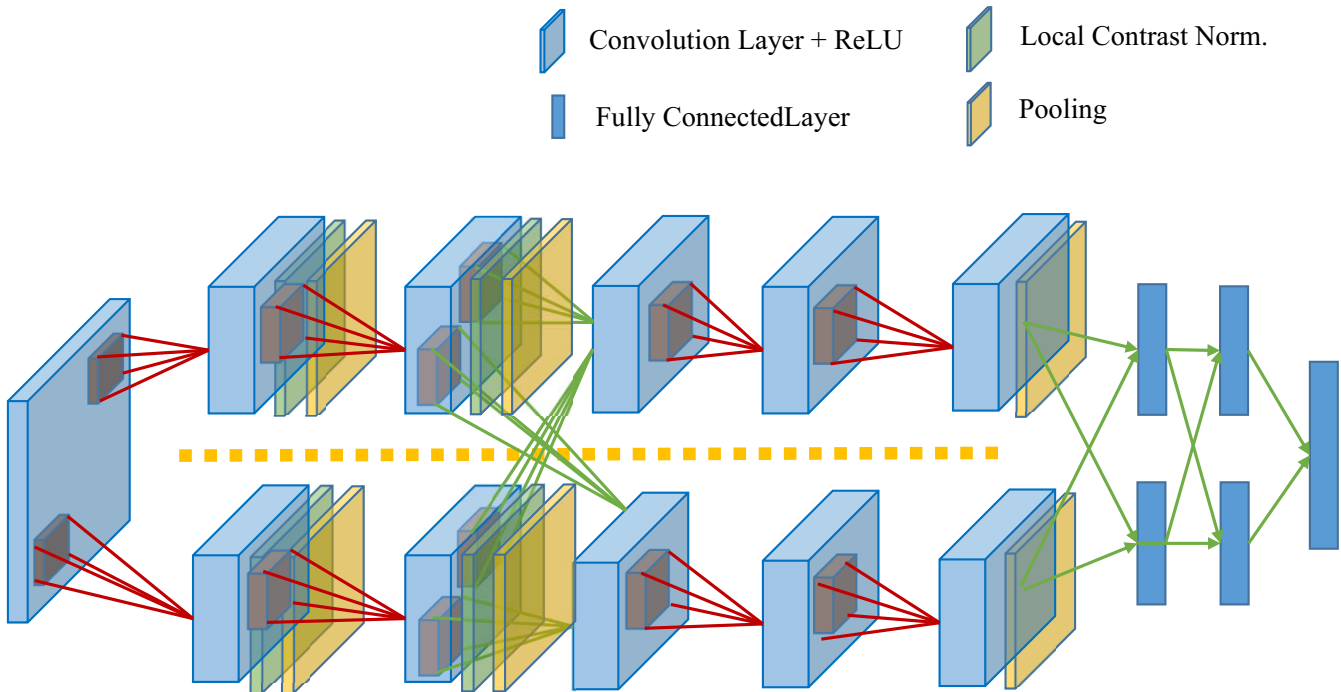
https://github.com/BVLC/caffe/blob/master/models/bvlc_alexnet/deploy.prototxt



AlexNet

معماری: نمایش لایه‌های جانبی

ALEXNET: ARCHITECTURE



AlexNet

معماری

AlexNet

[Krizhevsky et al. 2012]

Architecture:

CONV1

MAX POOL1

NORM1

CONV2

MAX POOL2

NORM2

CONV3

CONV4

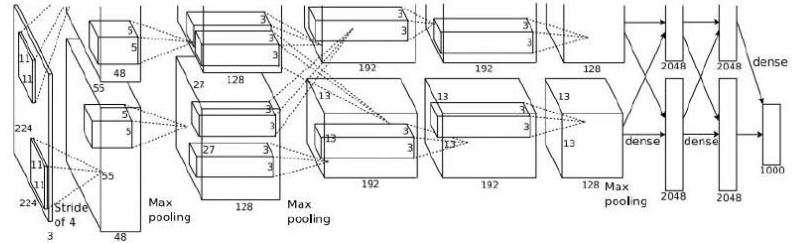
CONV5

Max POOL3

FC6

FC7

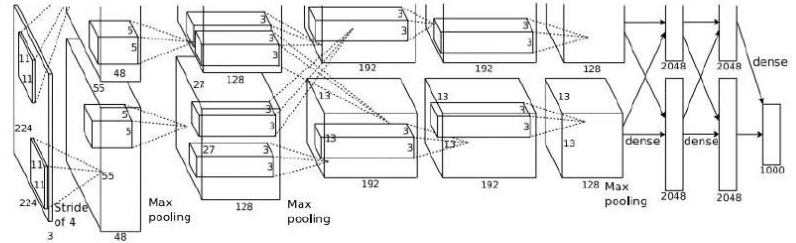
FC8



AlexNet

AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

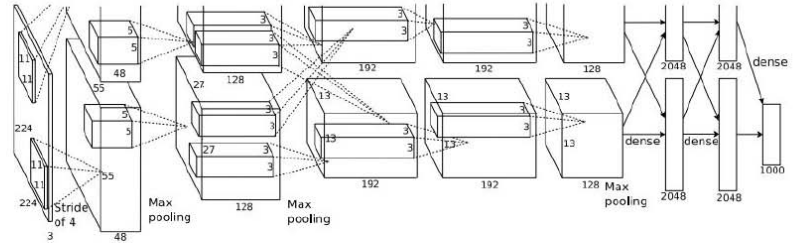
=>

Q: what is the output volume size? Hint: $(227-11)/4+1 = 55$

AlexNet

AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

=>

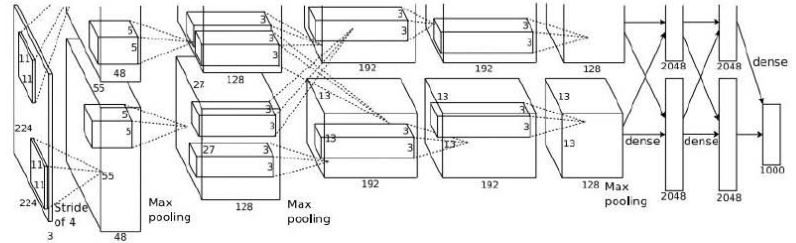
Output volume **[55x55x96]**

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

AlexNet

AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

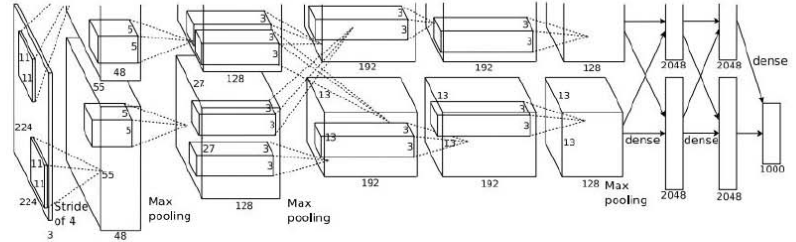
=>

Output volume **[55x55x96]**Parameters: $(11 \cdot 11 \cdot 3) \cdot 96 = 35K$

AlexNet

AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images
 After CONV1: 55x55x96

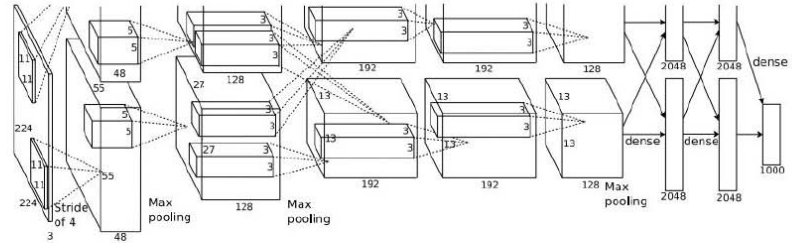
Second layer (POOL1): 3x3 filters applied at stride 2

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

AlexNet

AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images
 After CONV1: 55x55x96

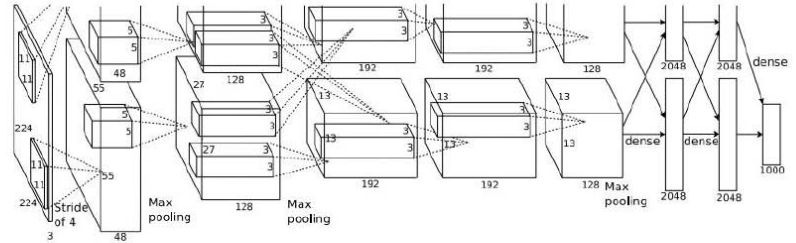
Second layer (POOL1): 3x3 filters applied at stride 2
 Output volume: 27x27x96

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

AlexNet

AlexNet

[Krizhevsky et al. 2012]



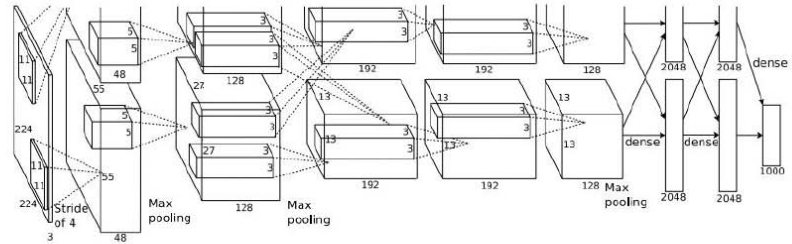
Input: 227x227x3 images
 After CONV1: 55x55x96

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

AlexNet

AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

After CONV1: 55x55x96

After POOL1: 27x27x96

...

AlexNet

AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

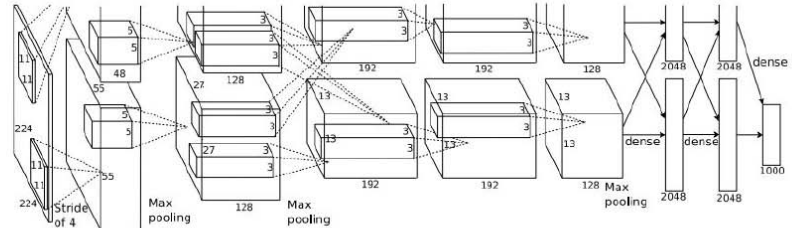


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

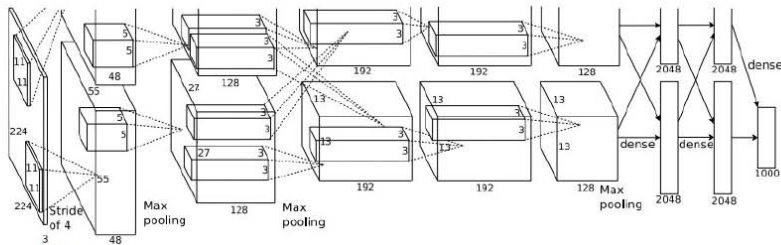
AlexNet

AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

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

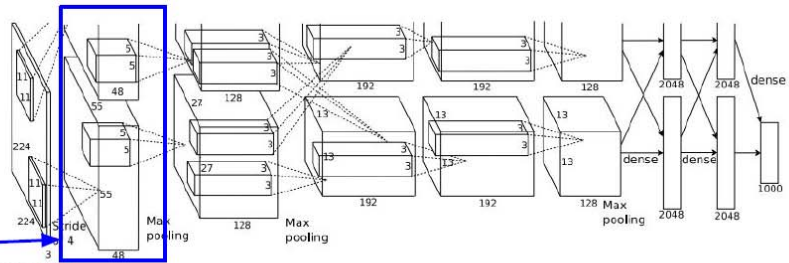
Details/Retrospectives:

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

AlexNet

AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

[55x55x48] x 2

Historical note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

AlexNet

AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

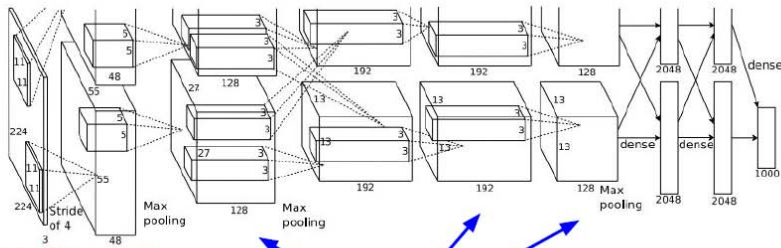
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)



CONV1, CONV2, CONV4, CONV5:
Connections only with feature maps
on same GPU

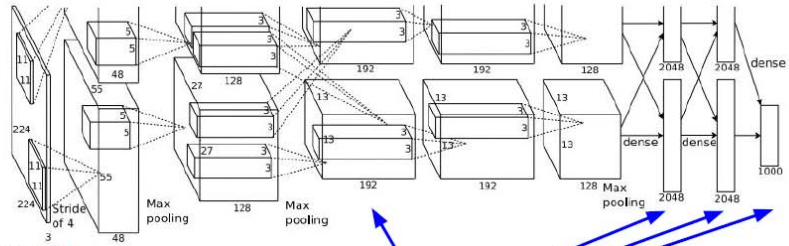
AlexNet

AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

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

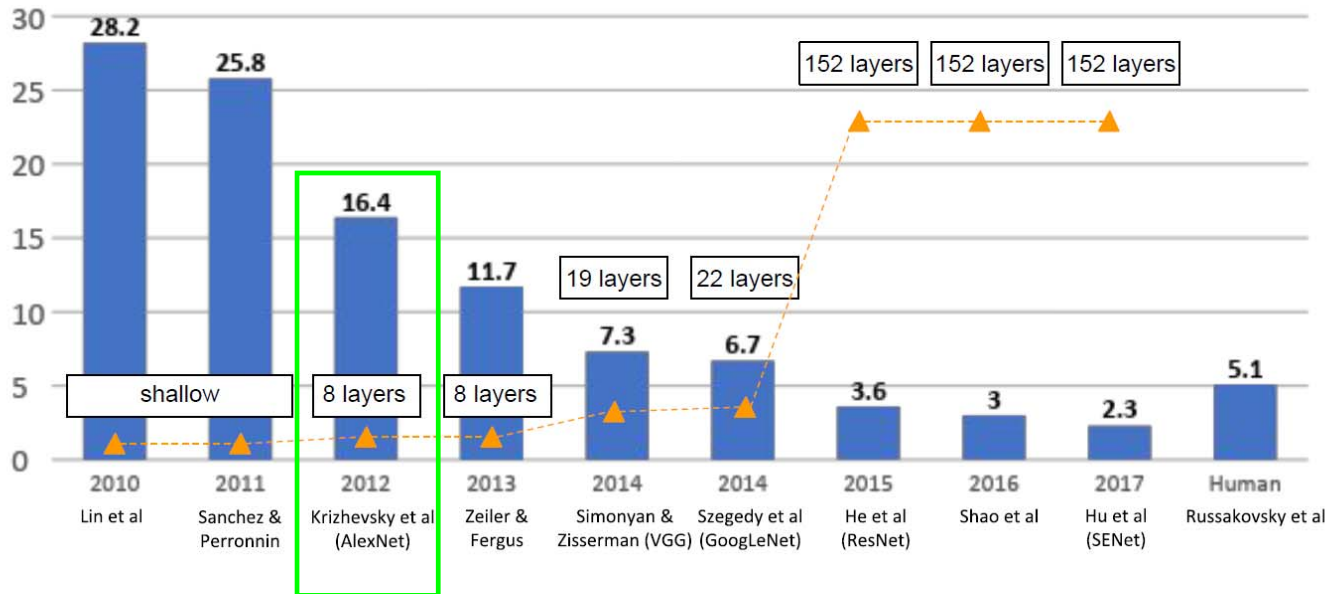
CONV3, FC6, FC7, FC8:
Connections with all feature maps in preceding layer, communication across GPUs

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

AlexNet

نتایج در چالش بازشناسی دیداری در مقیاس بالا ImageNet در مقایسه با سایر روش‌ها

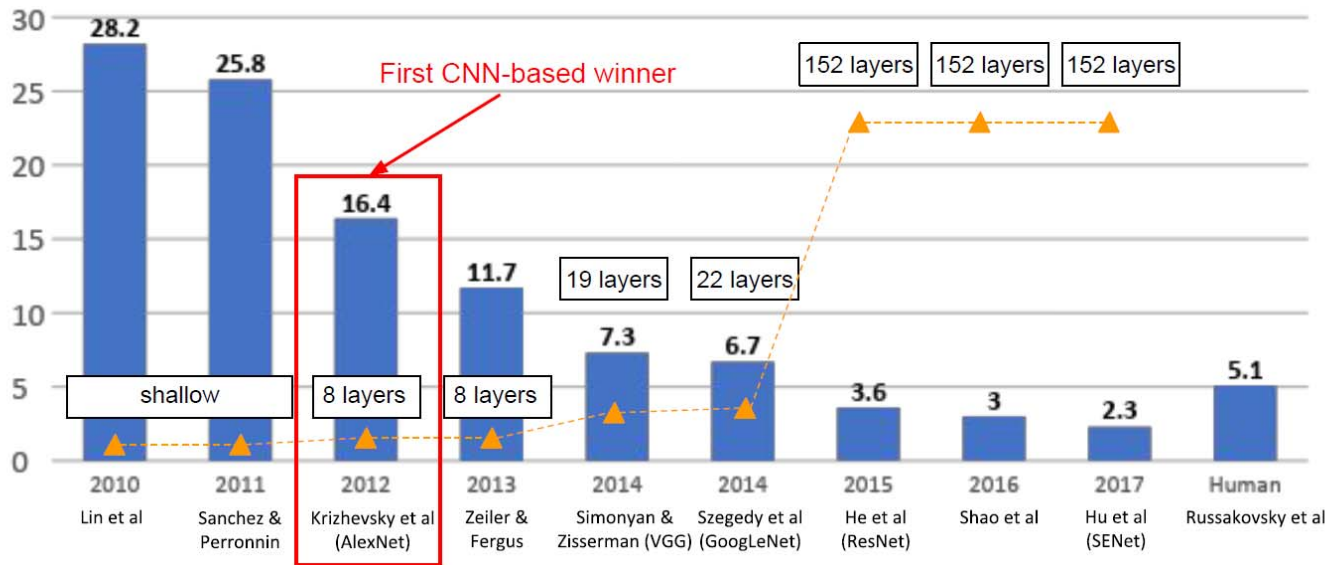
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



AlexNet

نتایج در چالش بازشناسی دیداری در مقیاس بالا ImageNet در مقایسه با سایر روش‌ها

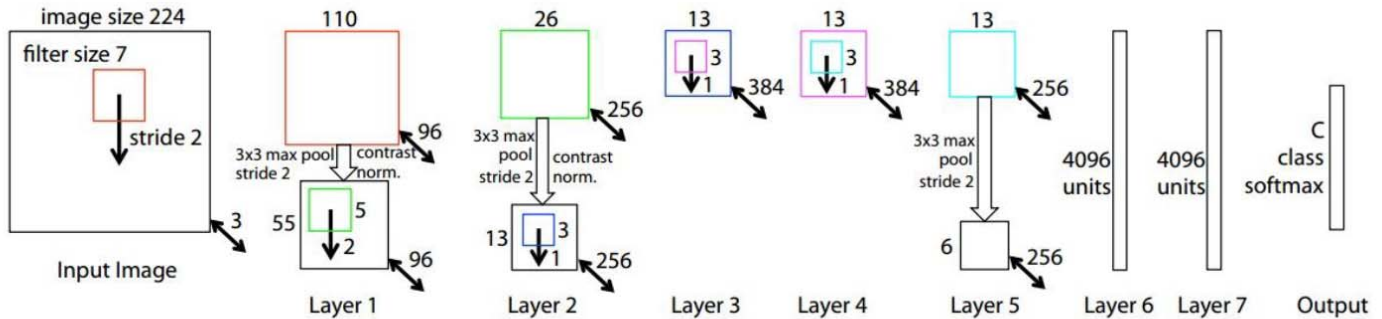
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



ZFNet

ZFNet

[Zeiler and Fergus, 2013]



AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

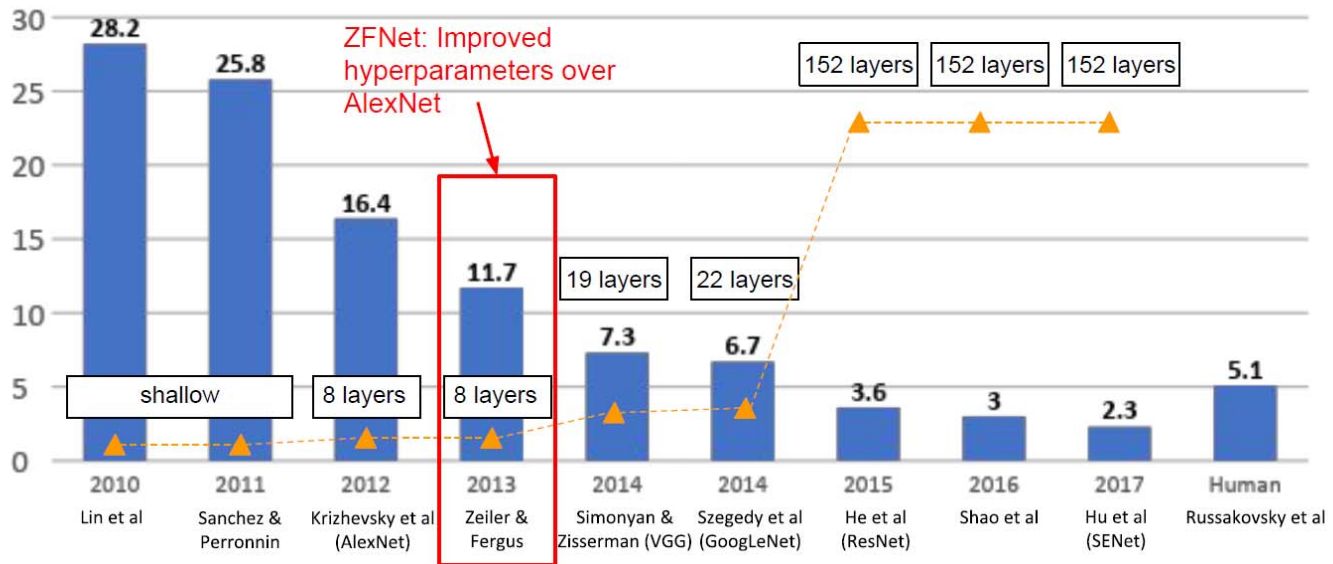
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 16.4% -> 11.7%

ZFNet

نتایج در چالش بازشناسی دیداری در مقیاس بالا ImageNet در مقایسه با سایر روش‌ها

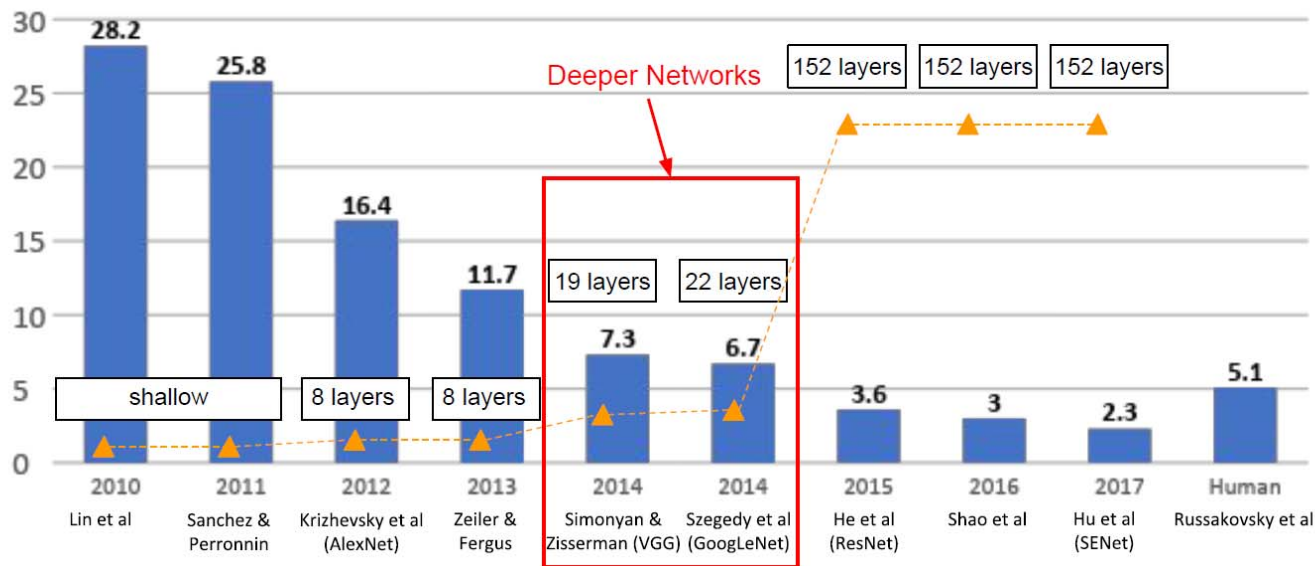
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



VGGNet - GoogLeNet

نتایج در چالش بازشناسی دیداری در مقیاس بالا ImageNet در مقایسه با سایر روش‌ها

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



VGGNet

VGGNet

[Simonyan and Zisserman, 2014]

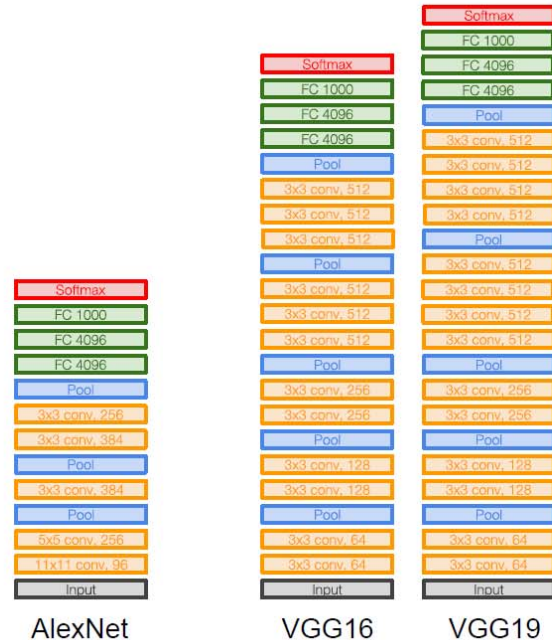
Small filters, Deeper networks

8 layers (AlexNet)

-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 211.7% top 5 error in ILSVRC'13
(ZFNet)

-> 7.3% top 5 error in ILSVRC'14



AlexNet

VGG16

VGG19

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* & Andrew Zisserman*

Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen, az}@robots.ox.ac.uk

ABSTRACT

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

1 INTRODUCTION

Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image and video recognition (Krizhevsky et al., 2012; Zeiler & Fergus, 2013; Sermanet et al., 2014; Simonyan & Zisserman, 2014) which has become possible due to the large public image repositories, such as ImageNet (Deng et al., 2009), and high-performance computing systems, such as GPUs or large-scale distributed clusters (Dean et al., 2012). In particular, an important role in the advance of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2014), which has served as a testbed for a few generations of large-scale image classification systems, from high-dimensional shallow feature encodings (Peronmin et al., 2010) (the winner of ILSVRC-2011) to deep ConvNets (Krizhevsky et al., 2012) (the winner of ILSVRC-2012).

With ConvNets becoming more of a commodity in the computer vision field, a number of attempts have been made to improve the original architecture of Krizhevsky et al. (2012) in a bid to achieve better accuracy. For instance, the best-performing submissions to the ILSVRC-2013 (Zeiler & Fergus, 2013; Sermanet et al., 2014) utilised smaller receptive window size and smaller stride of the first convolutional layer. Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales (Sermanet et al., 2014; Howard, 2014). In this paper, we address another important aspect of ConvNet architecture design – its depth. To this end, we fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small (3×3) convolution filters in all layers.

As a result, we come up with significantly more accurate ConvNet architectures, which not only achieve the state-of-the-art accuracy on ILSVRC classification and localisation tasks, but are also applicable to other image recognition datasets, where they achieve excellent performance even when used as a part of a relatively simple pipelines (e.g. deep features classified by a linear SVM without fine-tuning). We have released our two best-performing models¹ to facilitate further research.

The rest of the paper is organised as follows. In Sect. 2, we describe our ConvNet configurations. The details of the image classification training and evaluation are then presented in Sect. 3, and

*current affiliation: Google DeepMind *current affiliation: University of Oxford and Google DeepMind

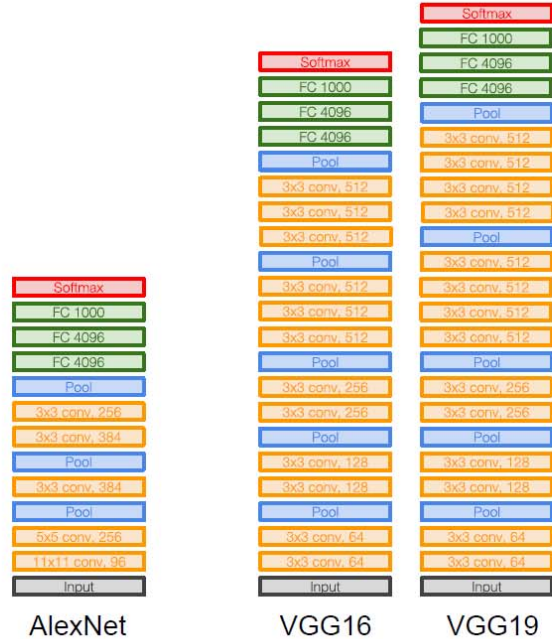
¹http://www.robots.ox.ac.uk/~vgg/research/very_deep/

VGGNet

VGGNet

[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)



VGGNet

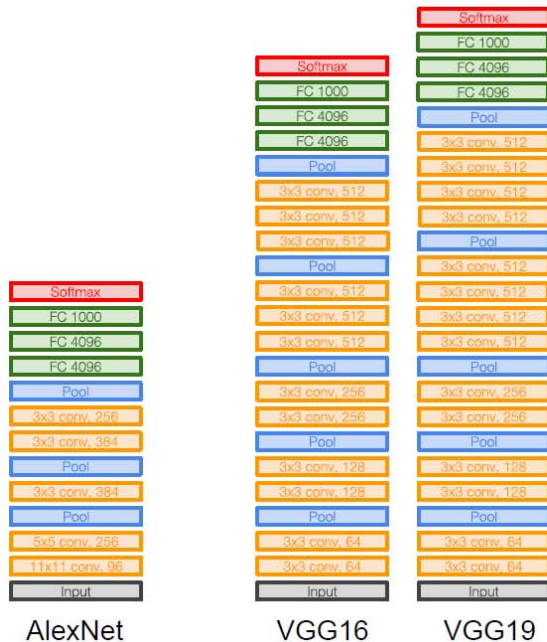
VGGNet

[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?



VGGNet

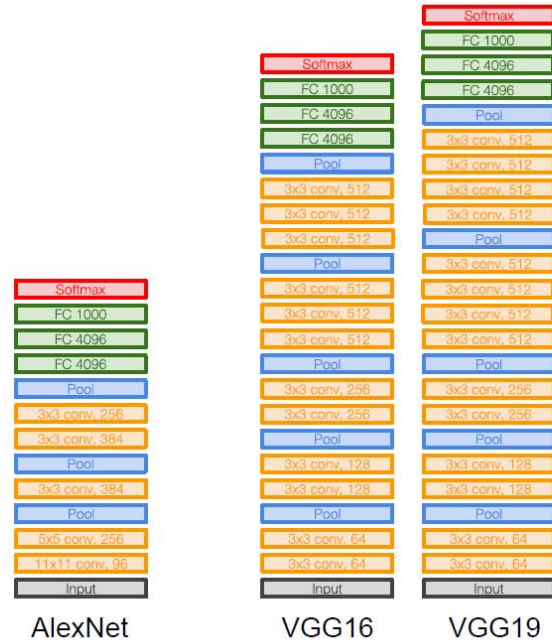
VGGNet

[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

[7x7]



AlexNet

VGG16

VGG19

VGGNet

VGGNet

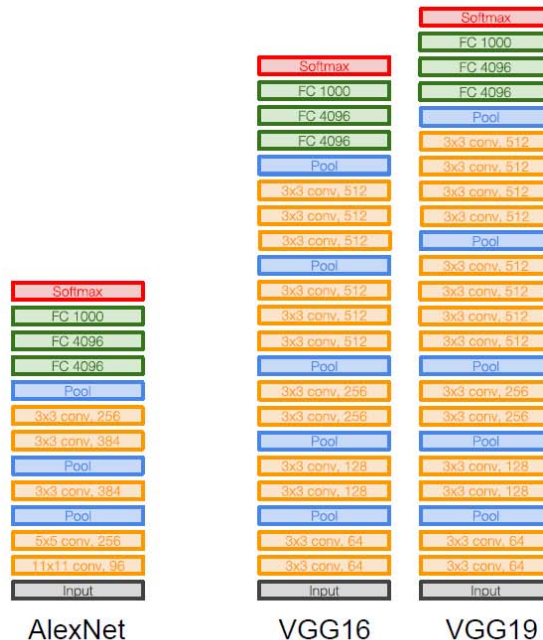
[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

But deeper, more non-linearities

And fewer parameters: $3 * (3^2C^2)$ vs. 7^2C^2 for C channels per layer



VGGNet

جزئیات معماری

INPUT: [224x224x3] memory: $224*224*3=150K$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*3)*64 = 1,728$

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*64)*64 = 36,864$

POOL2: [112x112x64] memory: $112*112*64=800K$ params: 0

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*64)*128 = 73,728$

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*128)*128 = 147,456$

POOL2: [56x56x128] memory: $56*56*128=400K$ params: 0

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*128)*256 = 294,912$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

POOL2: [28x28x256] memory: $28*28*256=200K$ params: 0

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*256)*512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [14x14x512] memory: $14*14*512=100K$ params: 0

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

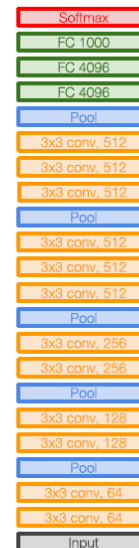
CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [7x7x512] memory: $7*7*512=25K$ params: 0

FC: [1x1x4096] memory: 4096 params: $7*7*512*4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096*4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096*1000 = 4,096,000$



VGG16

VGGNet

جزئیات معماری

INPUT: [224x224x3] memory: $224*224*3=150K$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*3)*64 = 1,728$

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*64)*64 = 36,864$

POOL2: [112x112x64] memory: $112*112*64=800K$ params: 0

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*64)*128 = 73,728$

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*128)*128 = 147,456$

POOL2: [56x56x128] memory: $56*56*128=400K$ params: 0

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*128)*256 = 294,912$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

POOL2: [28x28x256] memory: $28*28*256=200K$ params: 0

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*256)*512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [14x14x512] memory: $14*14*512=100K$ params: 0

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [7x7x512] memory: $7*7*512=25K$ params: 0

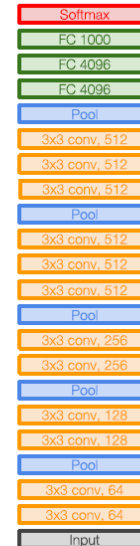
FC: [1x1x4096] memory: 4096 params: $7*7*512*4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096*4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096*1000 = 4,096,000$

TOTAL memory: 24M * 4 bytes \approx 96MB / image (for a forward pass)

TOTAL params: 138M parameters



VGG16

VGGNet

جزئیات معماری

INPUT: [224x224x3] memory: $224*224*3=150K$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*3)*64 = 1,728$

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*64)*64 = 36,864$

POOL2: [112x112x64] memory: $112*112*64=800K$ params: 0

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*64)*128 = 73,728$

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*128)*128 = 147,456$

POOL2: [56x56x128] memory: $56*56*128=400K$ params: 0

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*128)*256 = 294,912$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

POOL2: [28x28x256] memory: $28*28*256=200K$ params: 0

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*256)*512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [14x14x512] memory: $14*14*512=100K$ params: 0

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [7x7x512] memory: $7*7*512=25K$ params: 0

FC: [1x1x4096] memory: 4096 params: $7*7*512*4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096*4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096*1000 = 4,096,000$

Note:

Most memory is in early CONV

Most params are in late FC

TOTAL memory: 24M * 4 bytes \approx 96MB / image (only forward! \sim *2 for bwd)

TOTAL params: 138M parameters

VGGNet

جزئیات معماری

INPUT: [224x224x3] memory: $224*224*3=150K$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*3)*64 = 1,728$

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*64)*64 = 36,864$

POOL2: [112x112x64] memory: $112*112*64=800K$ params: 0

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*64)*128 = 73,728$

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*128)*128 = 147,456$

POOL2: [56x56x128] memory: $56*56*128=400K$ params: 0

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*128)*256 = 294,912$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

POOL2: [28x28x256] memory: $28*28*256=200K$ params: 0

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*256)*512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [14x14x512] memory: $14*14*512=100K$ params: 0

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [7x7x512] memory: $7*7*512=25K$ params: 0

FC: [1x1x4096] memory: 4096 params: $7*7*512*4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096*4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096*1000 = 4,096,000$

TOTAL memory: 24M * 4 bytes \sim 96MB / image (only forward! \sim *2 for bwd)

TOTAL params: 138M parameters



VGG16

Common names

VGGNet

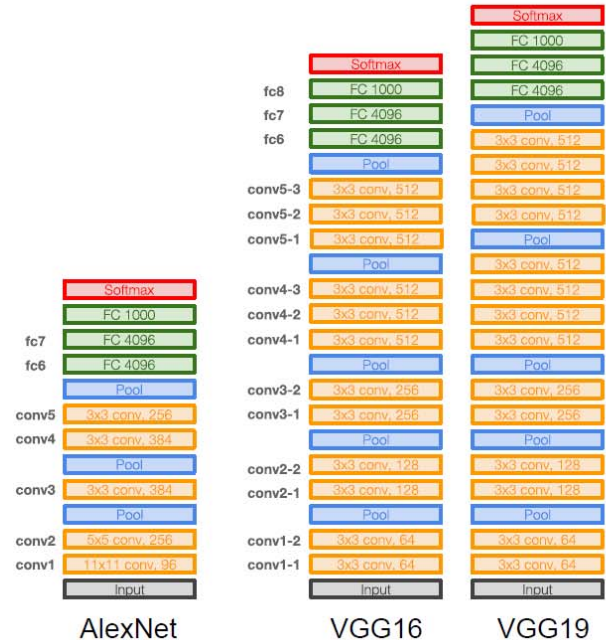
ملاحظات

VGGNet

[Simonyan and Zisserman, 2014]

Details:

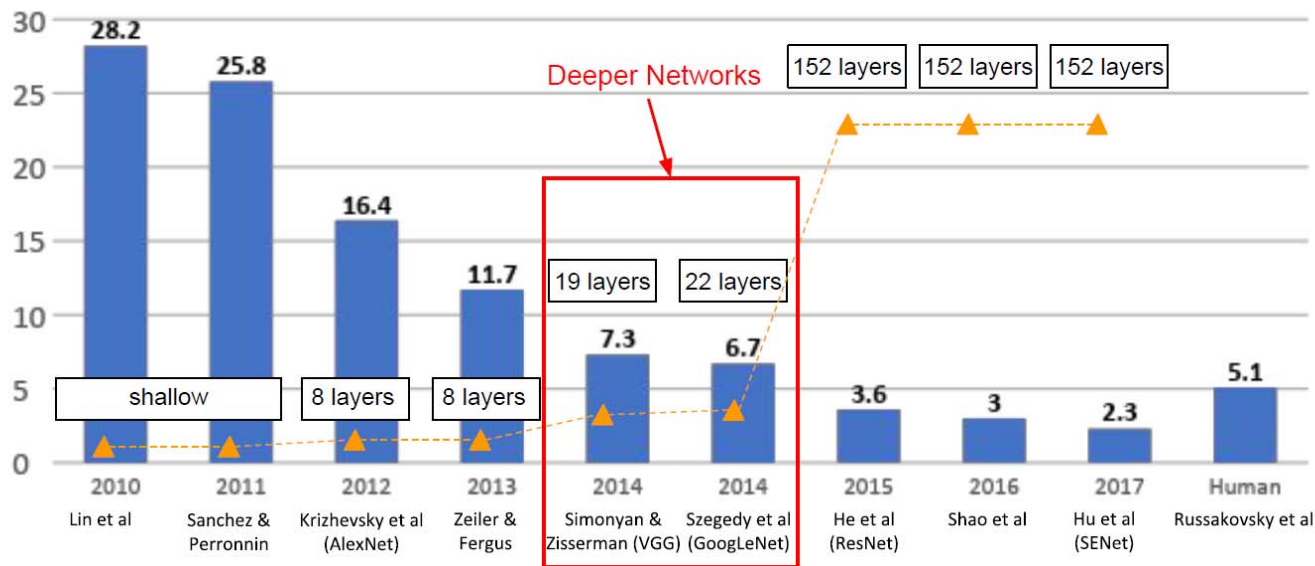
- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks



VGGNet - GoogLeNet

نتایج در چالش بازشناسی دیداری در مقیاس بالا ImageNet در مقایسه با سایر روش‌ها

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



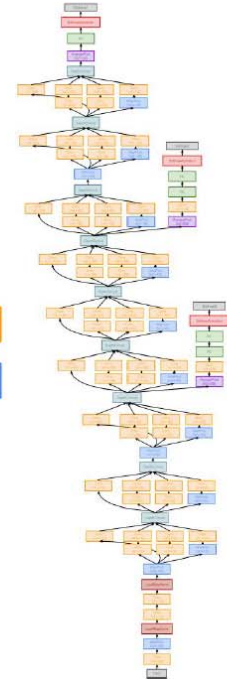
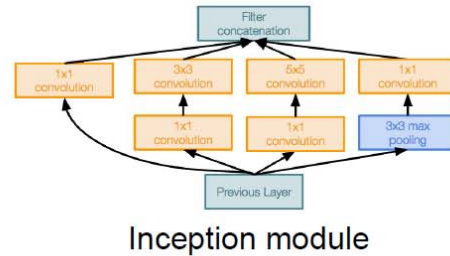
GoogLeNet

GoogLeNet

[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!
12x less than AlexNet
- ILSVRC’14 classification winner
(6.7% top 5 error)



Going deeper with convolutions

Christian Szegedy

Google Inc.

Wei Liu

University of North Carolina, Chapel Hill

Yangqing Jia

Google Inc.

Pierre Sermanet

Google Inc.

Scott Reed

University of Michigan

Dragomir Anguelov

Google Inc.

Dimitru Erhan

Google Inc.

Vincent Vanhoucke

Google Inc.

Andrew Rabinovich

Google Inc.

Abstract

We propose a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

1 Introduction

In the last three years, mainly due to the advances of deep learning, more concretely convolutional networks [10], the quality of image recognition and object detection has been progressing at a dramatic pace. One encouraging news is that most of this progress is not just the result of more powerful hardware, larger datasets and bigger models, but mainly a consequence of new ideas, algorithms and improved network architectures. No new data sources were used, for example, by the top entries in the ILSVRC 2014 competition besides the classification dataset of the same competition for detection purposes. Our GoogLeNet submission to ILSVRC 2014 actually uses $12\times$ fewer parameters than the winning architecture of Krizhevsky et al [9] from two years ago, while being significantly more accurate. The biggest gains in object-detection have not come from the utilization of deep networks alone or bigger models, but from the synergy of deep architectures and classical computer vision, like the R-CNN algorithm by Girshick et al [6].

Another notable factor is that with the ongoing traction of mobile and embedded computing, the efficiency of our algorithms – especially their power and memory use – gains importance. It is noteworthy that the considerations leading to the design of the deep architecture presented in this paper included this factor rather than having a sheer fixation on accuracy numbers. For most of the experiments, the models were designed to keep a computational budget of 1.5 billion multiply-adds at inference time, so that they do not end up to be a purely academic curiosity, but could be put to real world use, even on large datasets, at a reasonable cost.

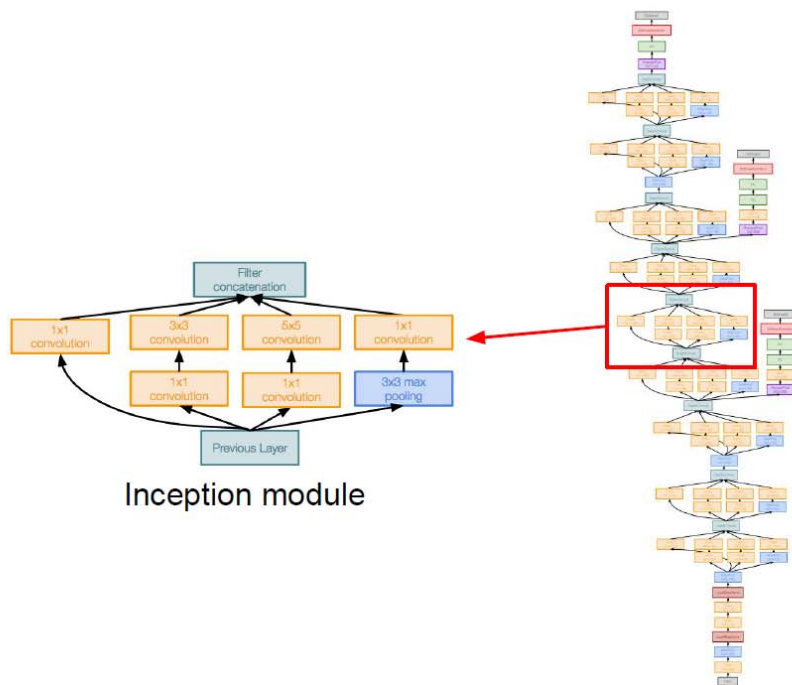
GoogLeNet

ماژول آغازش

GoogLeNet

[Szegedy et al., 2014]

“Inception module”: design a good local network topology (network within a network) and then stack these modules on top of each other



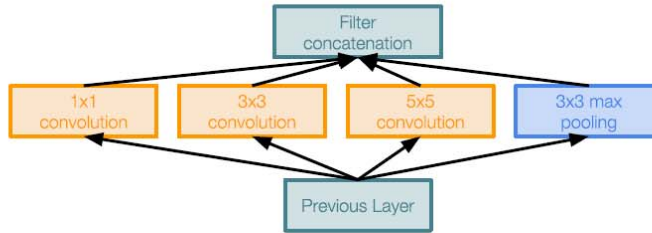
Inception module

GoogLeNet

ماژول آغازش خام

GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

Q: What is the problem with this?
[Hint: Computational complexity]

GoogLeNet

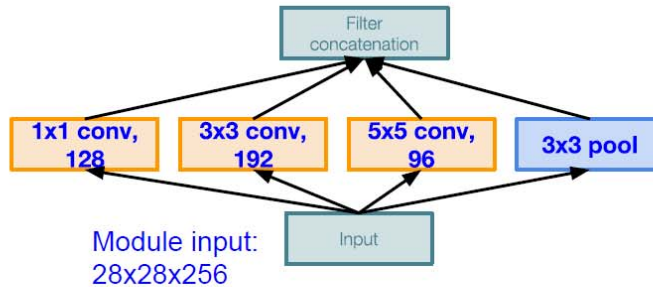
ماژول آغازش خام: مشکل پیچیدگی محاسباتی

GoogLeNet

[Szegedy et al., 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example:



Naive Inception module

GoogLeNet

ماژول آغازش خام: مشکل پیچیدگی محاسباتی

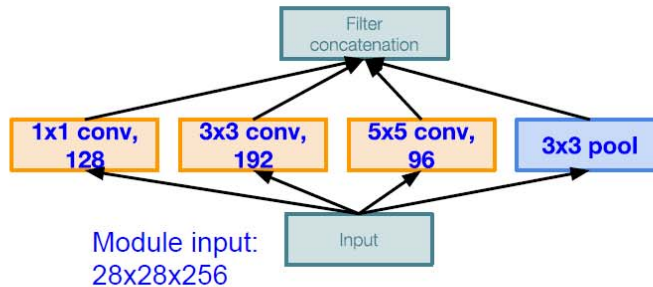
GoogLeNet

[Szegedy et al., 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example:

Q1: What is the output size of the
1x1 conv, with 128 filters?



Naive Inception module

GoogLeNet

ماژول آغازش خام: مشکل پیچیدگی محاسباتی

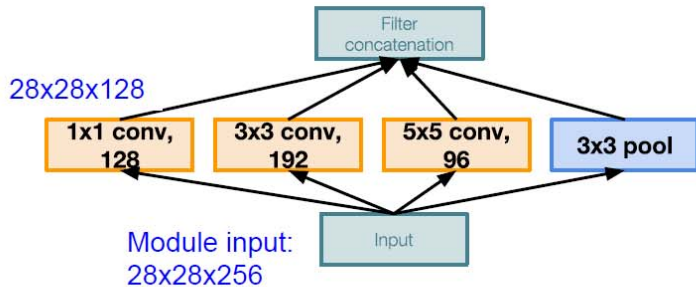
GoogLeNet

[Szegedy et al., 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example:

Q2: What are the output sizes of all different filter operations?



Naive Inception module

GoogLeNet

ماژول آغازش خام: مشکل پیچیدگی محاسباتی

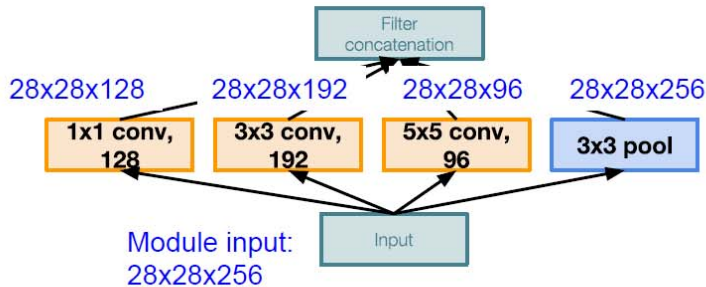
GoogLeNet

[Szegedy et al., 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example:

Q2: What are the output sizes of all different filter operations?



Naive Inception module

GoogLeNet

ماژول آغازش خام: مشکل پیچیدگی محاسباتی

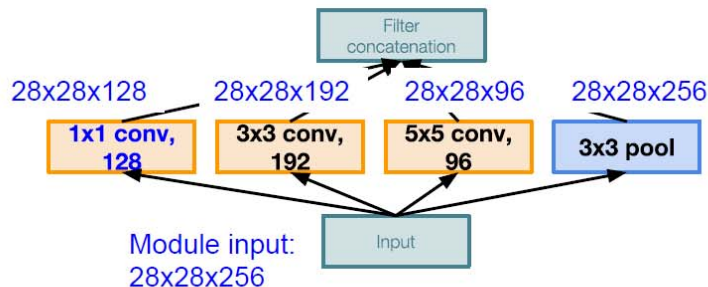
GoogLeNet

[Szegedy et al., 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example:

Q3: What is output size after filter concatenation?



Naive Inception module

GoogLeNet

ماژول آغازش خام: مشکل پیچیدگی محاسباتی

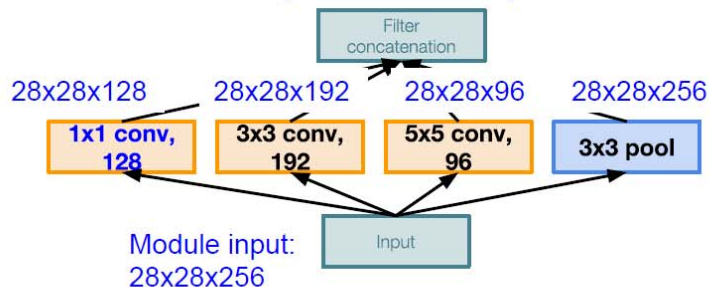
GoogLeNet

[Szegedy et al., 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example: Q3: What is output size after filter concatenation?

$$28 \times 28 \times (128 + 192 + 96 + 256) = 28 \times 28 \times 672$$



Naive Inception module

GoogLeNet

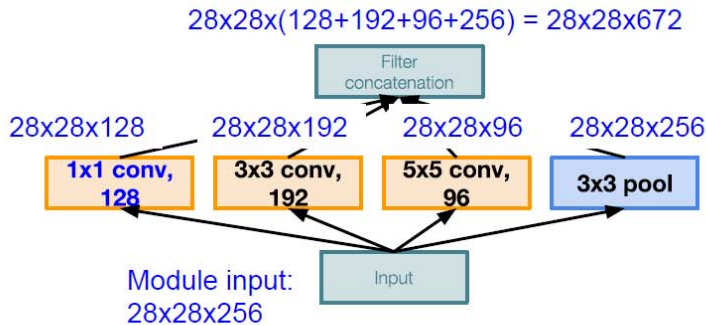
ماژول آغازش خام: مشکل پیچیدگی محاسباتی

GoogLeNet

[Szegedy et al., 2014]

Example:

Q3: What is output size after filter concatenation?



Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

Conv Ops:

[1x1 conv, 128] $28 \times 28 \times 128 \times 1 \times 1 \times 256$

[3x3 conv, 192] $28 \times 28 \times 192 \times 3 \times 3 \times 256$

[5x5 conv, 96] $28 \times 28 \times 96 \times 5 \times 5 \times 256$

Total: 854M ops

GoogLeNet

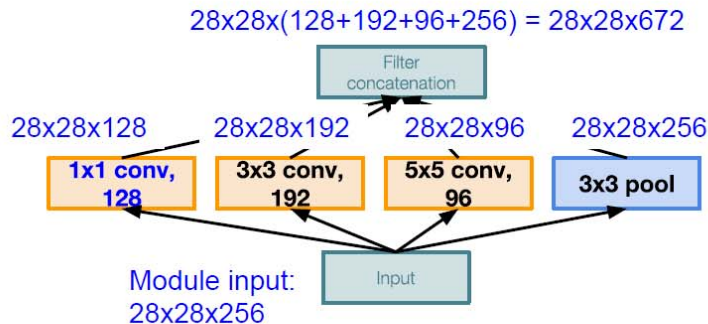
ماژول آغازش خام: مشکل پیچیدگی محاسباتی

GoogLeNet

[Szegedy et al., 2014]

Example:

Q3: What is output size after filter concatenation?



Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

Conv Ops:

[1x1 conv, 128] $28 \times 28 \times 128 \times 1 \times 1 \times 256$

[3x3 conv, 192] $28 \times 28 \times 192 \times 3 \times 3 \times 256$

[5x5 conv, 96] $28 \times 28 \times 96 \times 5 \times 5 \times 256$

Total: 854M ops

Very expensive compute

Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

GoogLeNet

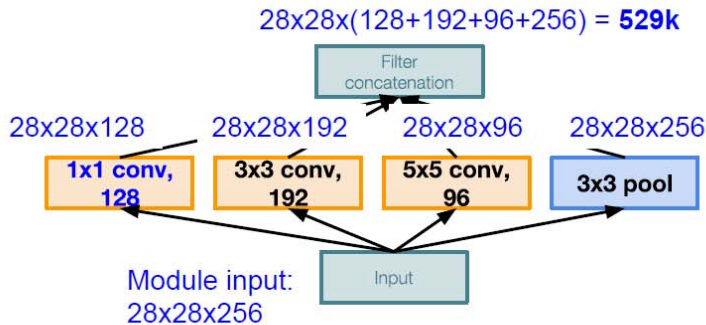
ماژول آغازش خام: مشکل پیچیدگی محاسباتی؛ راه حل: استفاده از لایه‌های «گلوگاهی» برای کاهش عمق نقشه‌ی ویژگی

GoogLeNet

[Szegedy et al., 2014]

Example:

Q3: What is output size after filter concatenation?

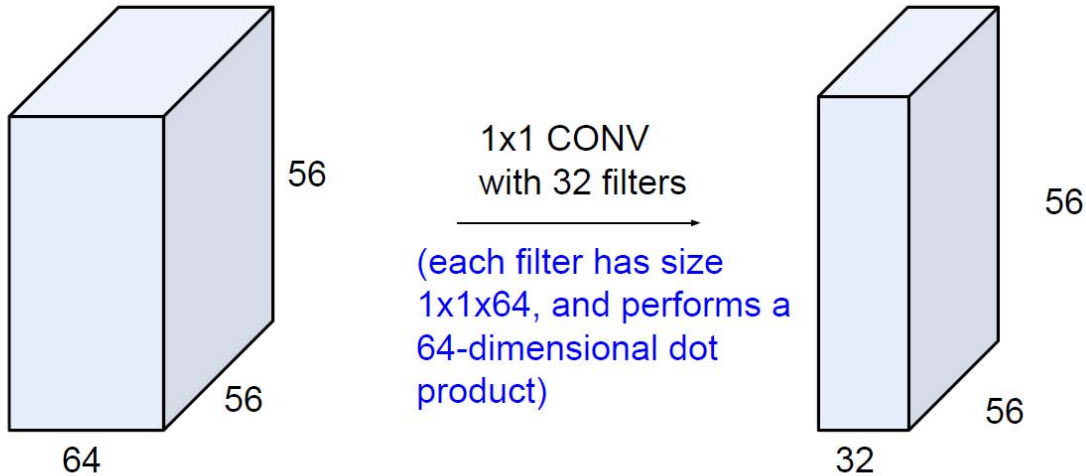


Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

Solution: “bottleneck” layers that use 1×1 convolutions to reduce feature depth

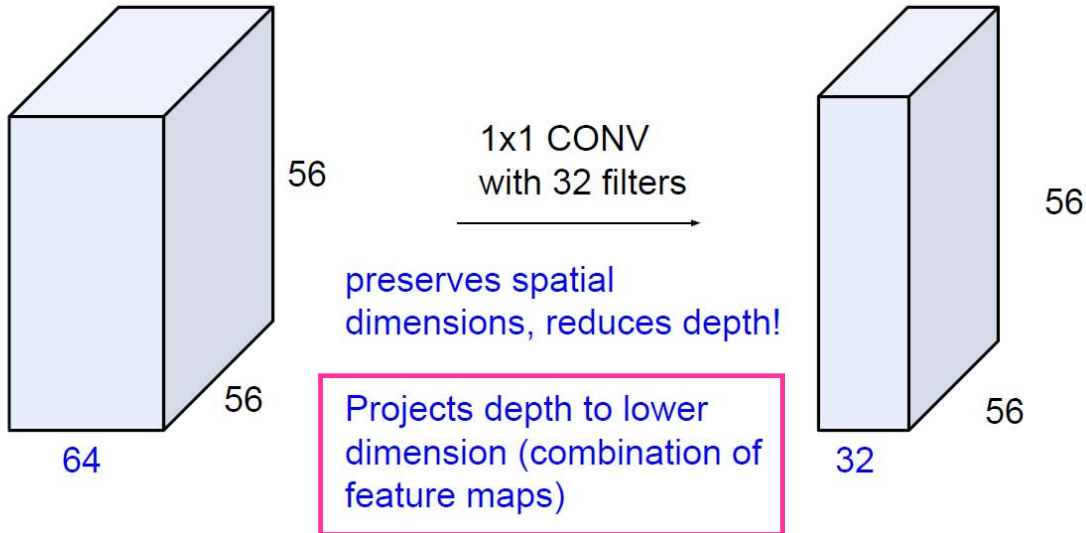
GoogLeNet

کانولوشن 1×1 Reminder: 1×1 convolutions

GoogLeNet

کانولوشن ۱×۱

Reminder: 1x1 convolutions

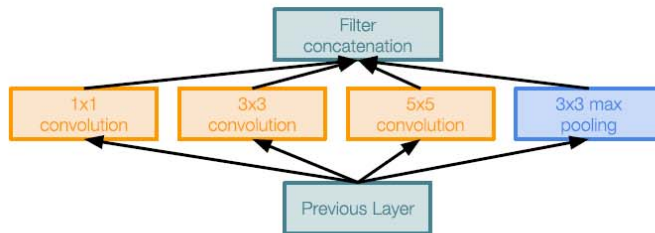


GoogLeNet

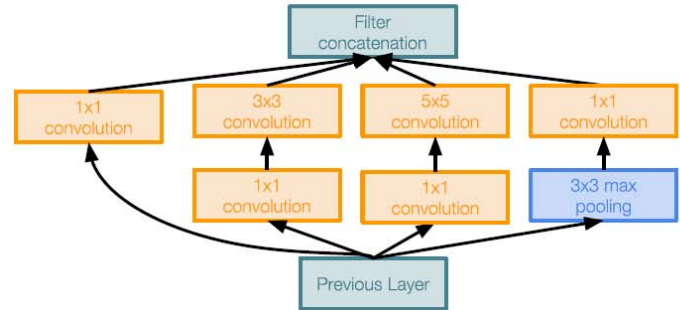
ماژول آغازش با کاهش بعد

GoogLeNet

[Szegedy et al., 2014]



Naive Inception module



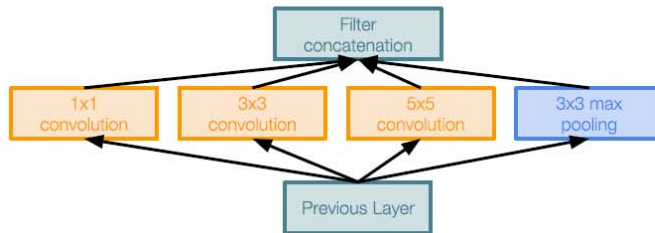
Inception module with dimension reduction

GoogLeNet

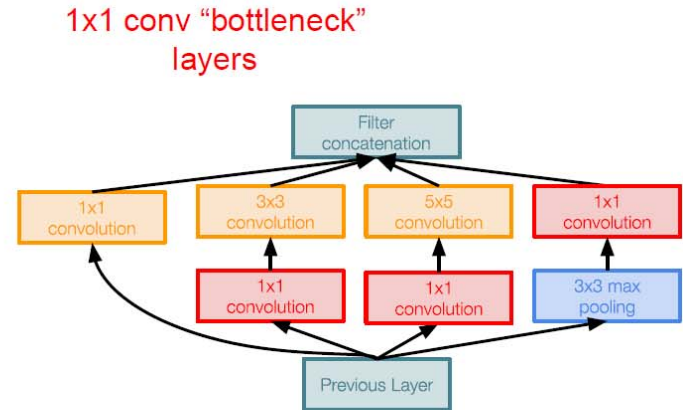
ماژول آغازش با کاهش بعد: لایه‌های گلوگاهی 1×1

GoogLeNet

[Szegedy et al., 2014]



Naive Inception module



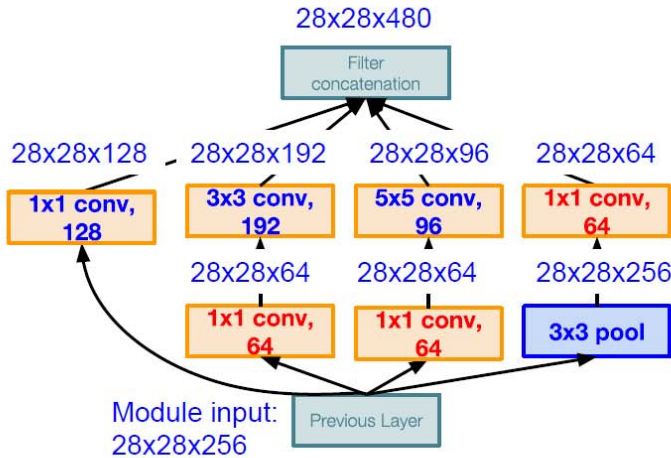
Inception module with dimension reduction

GoogLeNet

ماژول آغازش با کاهش بعد: معماری

GoogLeNet

[Szegedy et al., 2014]



Inception module with dimension reduction

Using same parallel layers as naive example, and adding “1x1 conv, 64 filter” bottlenecks:

Conv Ops:

[1x1 conv, 64] 28x28x64x1x1x256
 [1x1 conv, 64] 28x28x64x1x1x256
 [1x1 conv, 128] 28x28x128x1x1x256
 [3x3 conv, 192] 28x28x192x3x3x64
 [5x5 conv, 96] 28x28x96x5x5x64
 [1x1 conv, 64] 28x28x64x1x1x256

Total: 358M ops

Compared to 854M ops for naive version
 Bottleneck can also reduce depth after pooling layer

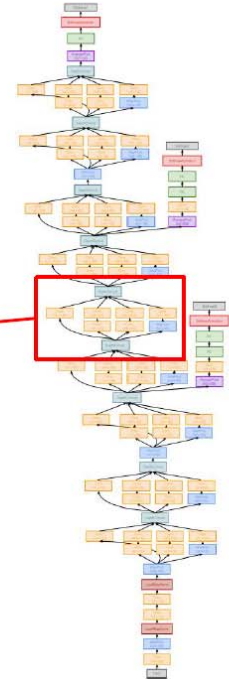
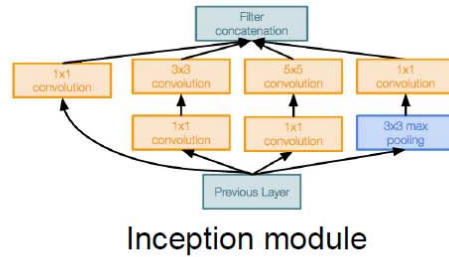
GoogLeNet

پشته‌سازی با ماژول‌های آغازش با کاهش بعد

GoogLeNet

[Szegedy et al., 2014]

Stack Inception modules
with dimension reduction
on top of each other



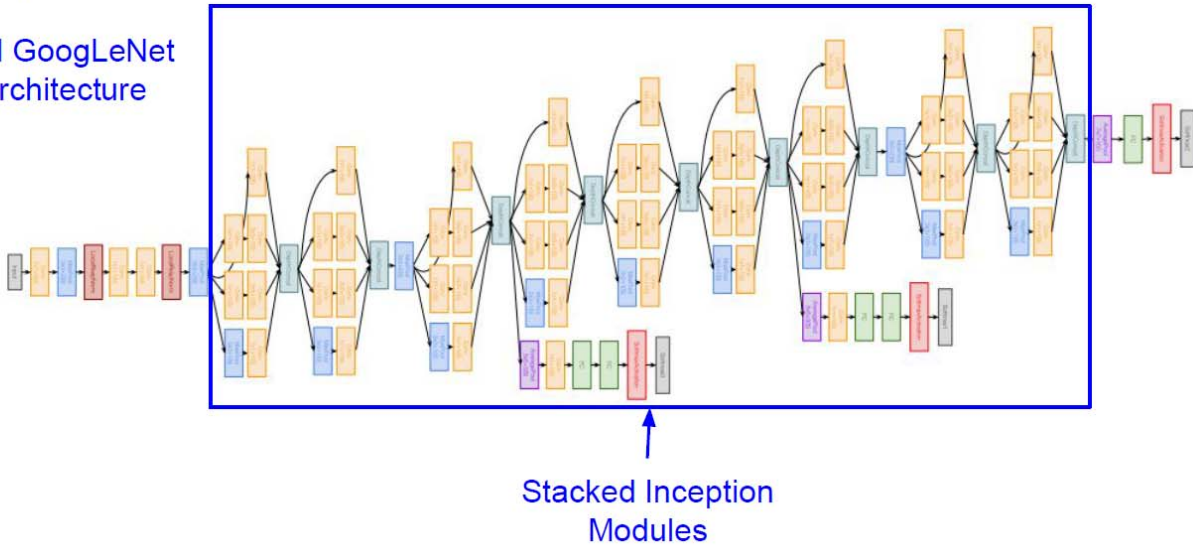
GoogLeNet

معماری کامل

GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet
architecture



Stacked Inception
Modules

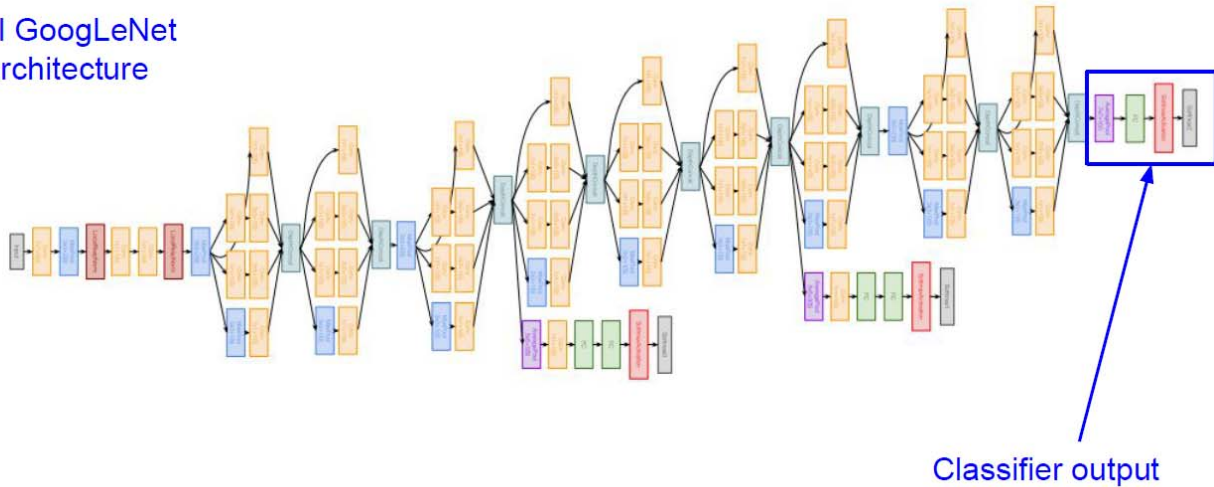
GoogLeNet

معماری کامل: خروجی طبقه‌بندی کننده

GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet
architecture



Classifier output

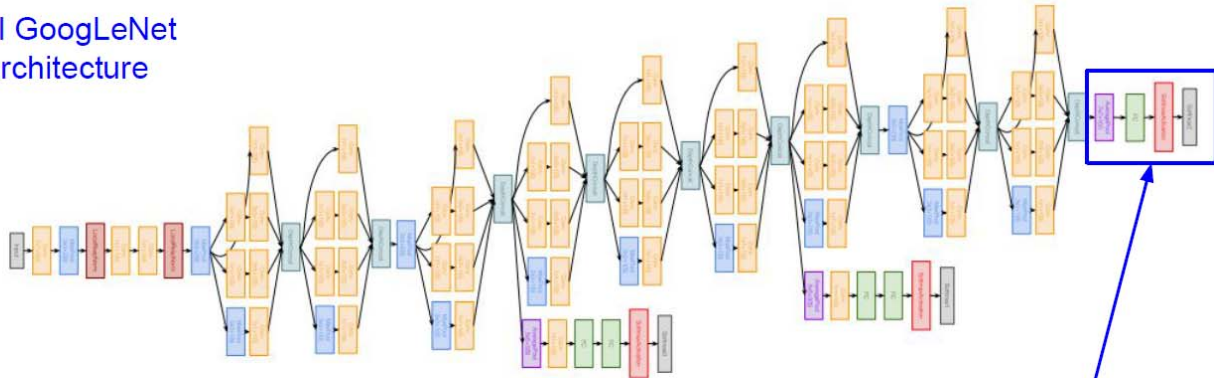
GoogLeNet

معماری کامل: خروجی طبقه‌بندی کننده (حذف لایه‌های تماماً متصل گران)

GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet
architecture



Classifier output
(removed expensive FC layers!)

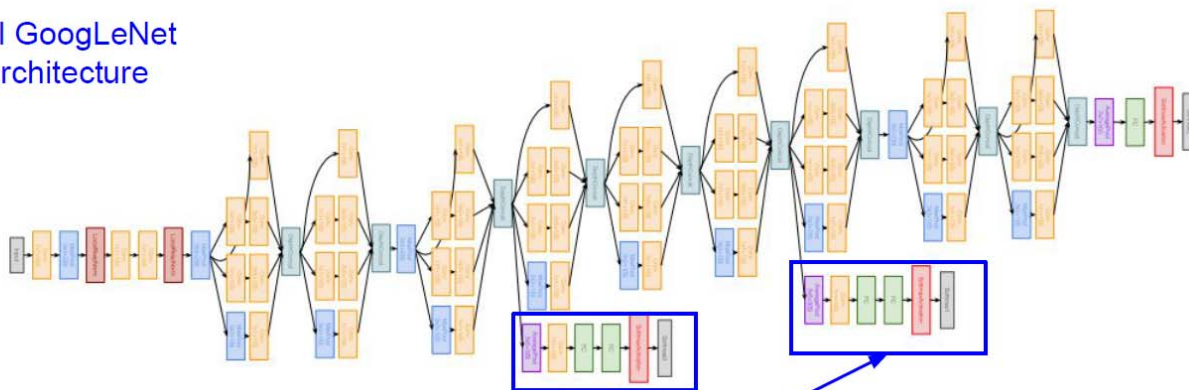
GoogLeNet

معماری کامل: خروجی‌های کمکی طبقه‌بندی کننده

GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

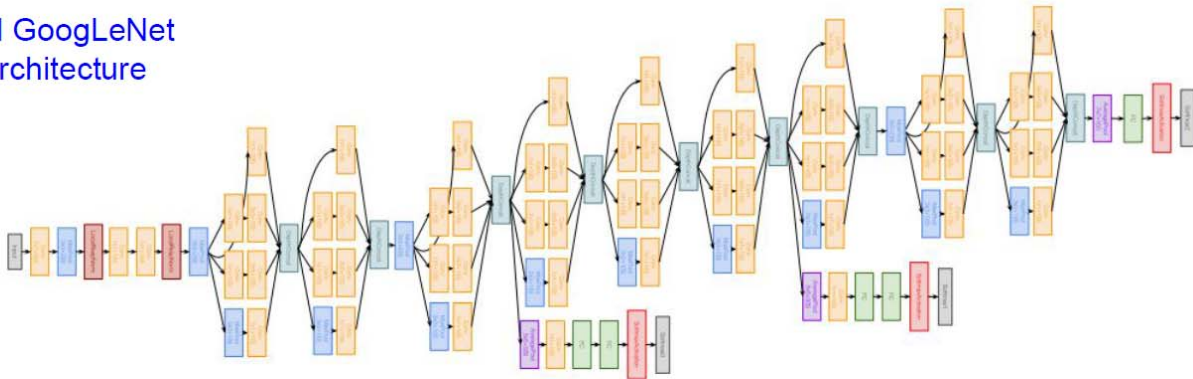
GoogLeNet

معماری کامل

GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet
architecture



22 total layers with weights

(parallel layers count as 1 layer => 2 layers per Inception module. Don't count auxiliary output layers)

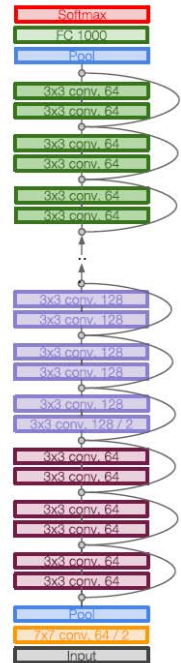
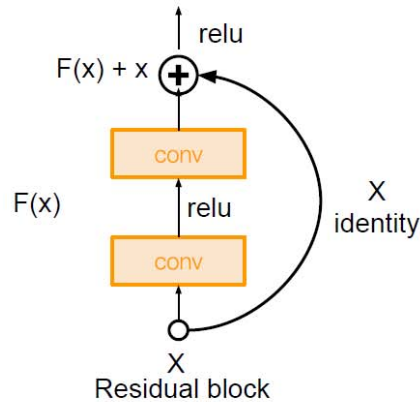
ResNet

ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



ResNet

ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

با ادامه دادن روی هم قرار دهی لایه‌های عمیق‌تر بر روی یک شبکه‌ی عصبی کانولوشنال «ساده» چه اتفاقی می‌افتد؟

ResNet

ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?



Q: What's strange about these training and test curves?
[Hint: look at the order of the curves]

ResNet

ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?



56-layer model performs worse on both training and test error
 -> The deeper model performs worse, but it's not caused by overfitting!

مدل عمیق‌تر بدتر عمل کرده است، اما این به سبب بیش‌برازش نیست!

ResNet

ResNet

[He et al., 2015]

Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

فرضیه: مشکل، مسئله‌ی بهینه‌سازی است: بهینه‌سازی مدل‌های عمیق‌تر سخت‌تر است.

ResNet

ResNet

[He et al., 2015]

Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

The deeper model should be able to perform at least as well as the shallower model.

مدل عمیق‌تر باید بتواند حداقل به خوبی مدل کم‌عمق‌تر عمل کند.

A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

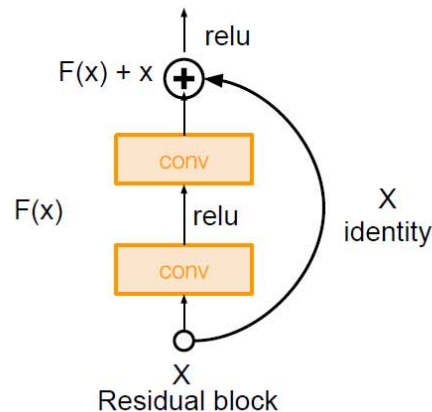
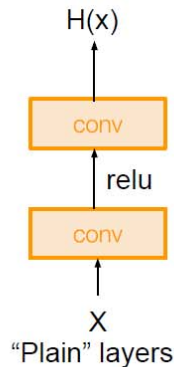
یک راه‌حل از طریق ساختن: کپی کردن لایه‌های یادگیری شده از مدل کم‌عمق‌تر و قرار دادن لایه‌های بیشتر برای نگاشت همانی

ResNet

ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



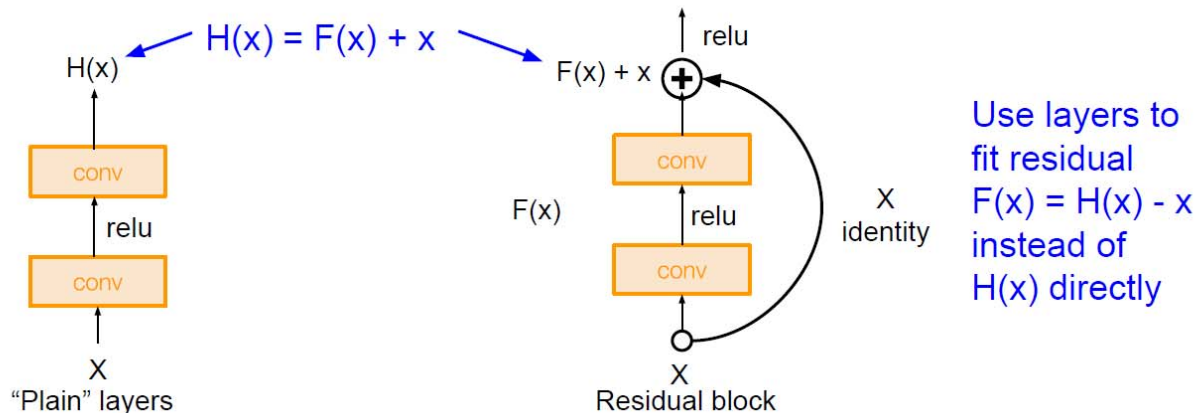
راه حل: استفاده از لایه‌های شبکه برای برازش یک نگاشت باقیمانده‌ای به جای تلاش مستقیم برای برازش مورد نظر مطلوب

ResNet

ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



از لایه‌ها برای برازش باقیمانده $F(x) = H(x) - x$ به جای $H(x)$ به طور مستقیم استفاده می‌کنیم.

ResNet

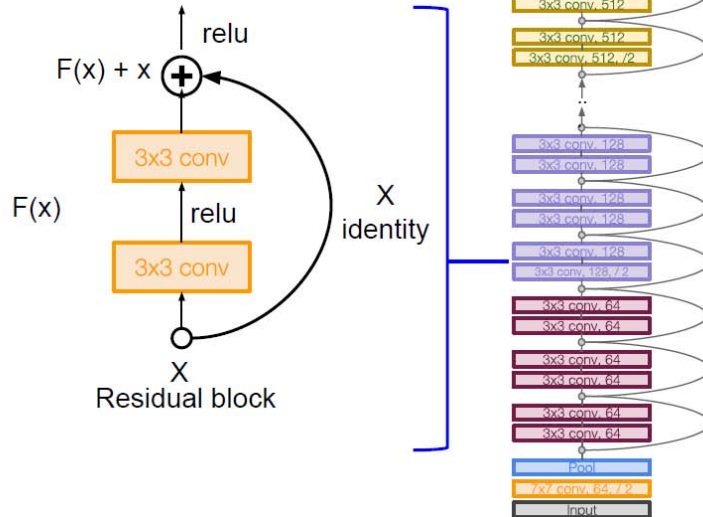
جزئیات معماری

ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers



ResNet

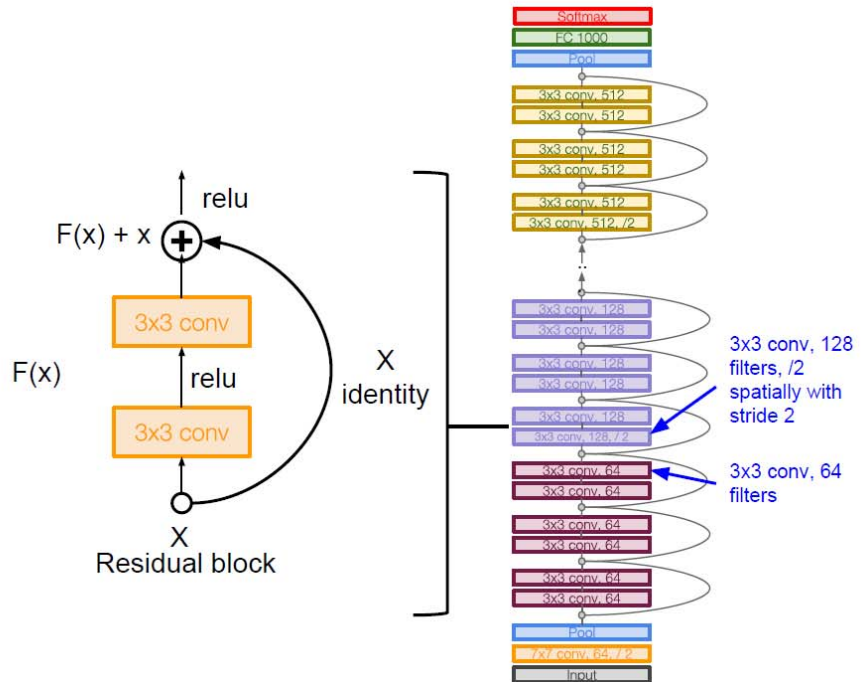
جزئیات معماری

ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)



ResNet

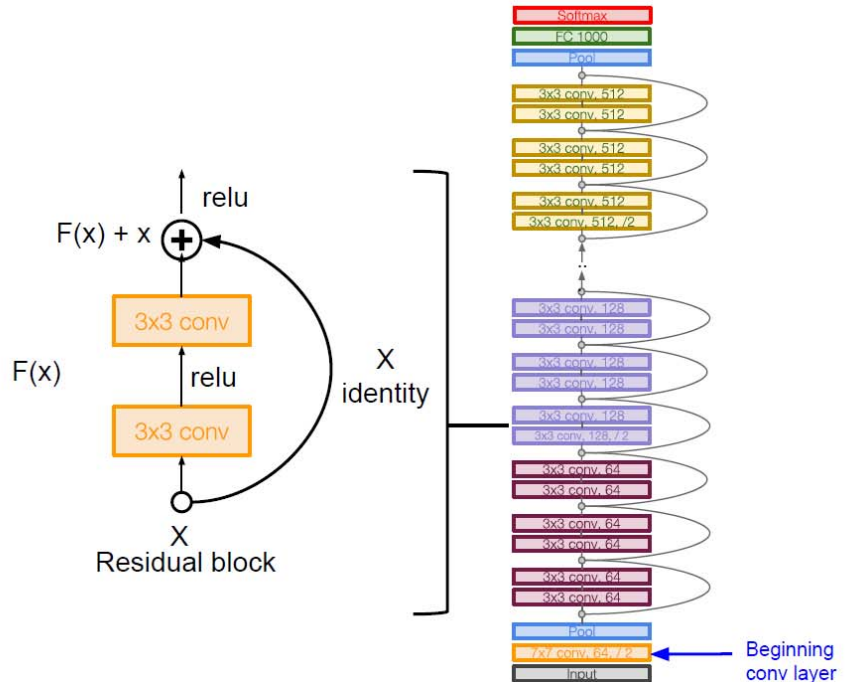
جزئیات معماری

ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning



ResNet

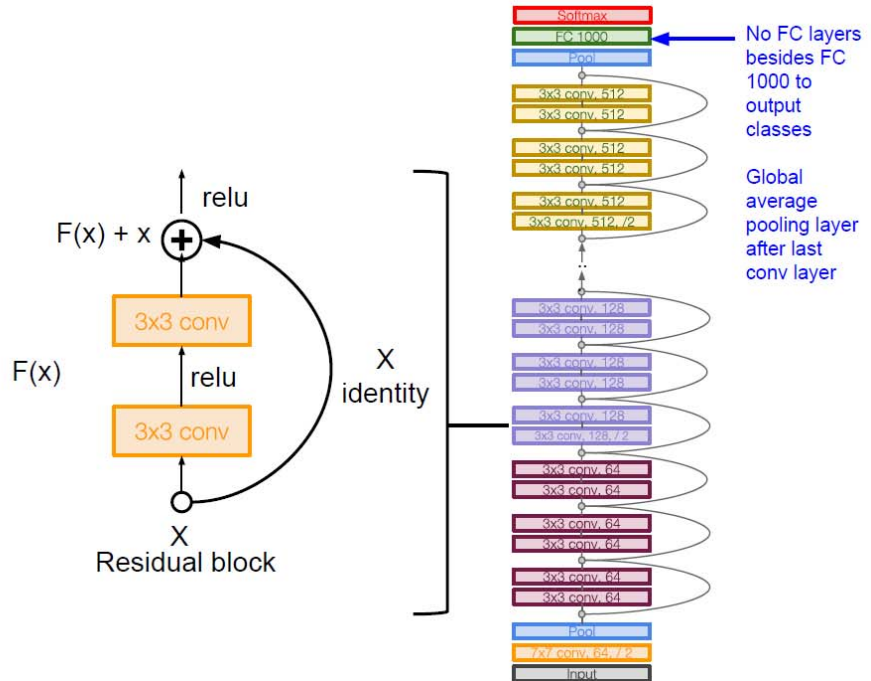
جزئیات معماری

ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)



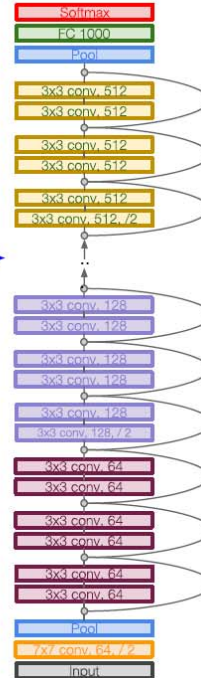
ResNet

جزئیات معماری

ResNet

[He et al., 2015]

Total depths of 34, 50, 101, or
152 layers for ImageNet



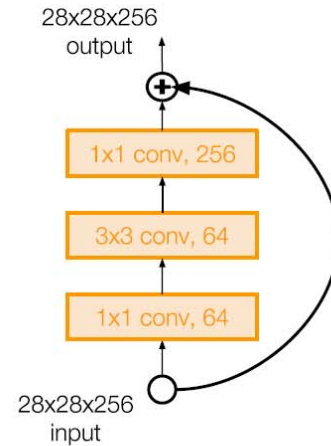
ResNet

جزئیات معماری

ResNet

[He et al., 2015]

For deeper networks
(ResNet-50+), use “bottleneck”
layer to improve efficiency
(similar to GoogLeNet)

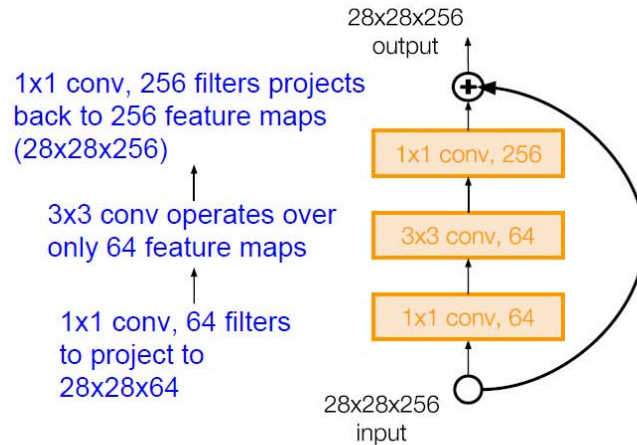


ResNet

ResNet

[He et al., 2015]

For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to GoogLeNet)



ResNet

آموزش شبکه در عمل

ResNet

[He et al., 2015]

Training ResNet in practice:

- Batch Normalization after every CONV layer
- Xavier 2/ initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of $1e-5$
- No dropout used

ResNet

نتایج تجربی

ResNet

[He et al., 2015]

Experimental Results

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowering training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

MSRA @ ILSVRC & COCO 2015 Competitions

• **1st places in all five main tracks**

- ImageNet Classification: “*Ultra-deep*” (quote Yann) **152-layer** nets
- ImageNet Detection: **16%** better than 2nd
- ImageNet Localization: **27%** better than 2nd
- COCO Detection: **11%** better than 2nd
- COCO Segmentation: **12%** better than 2nd

ResNet

نتایج تجربی

ResNet

[He et al., 2015]

Experimental Results

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowering training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**

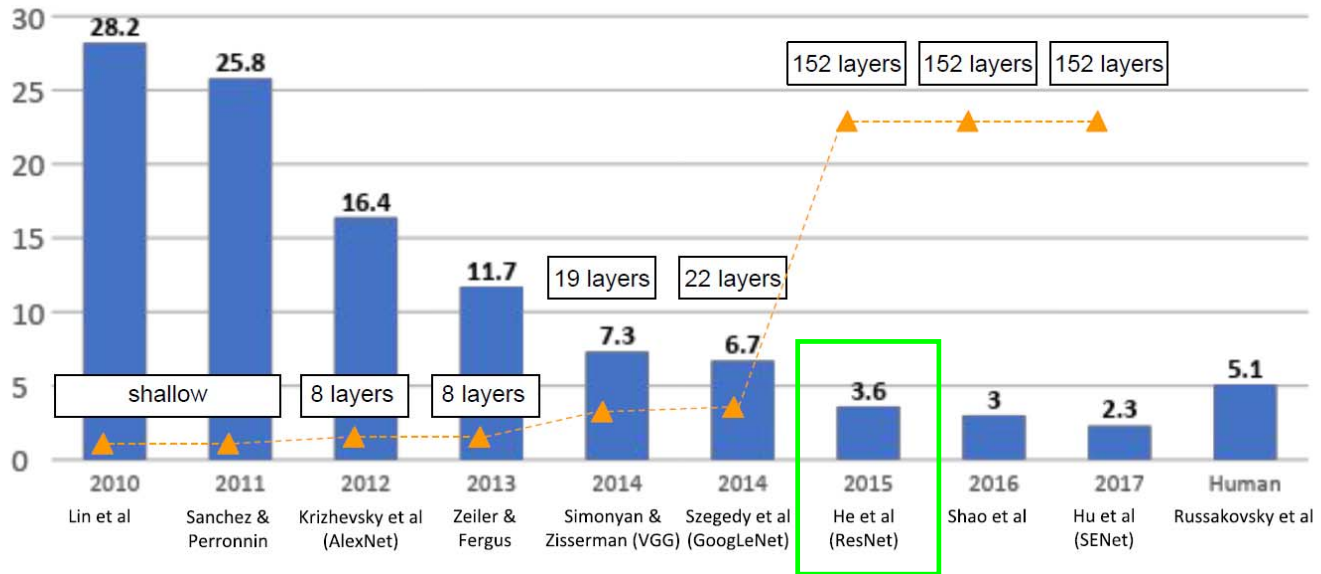
- ImageNet Classification: “*Ultra-deep*” (quote Yann) **152-layer** nets
- ImageNet Detection: **16%** better than 2nd
- ImageNet Localization: **27%** better than 2nd
- COCO Detection: **11%** better than 2nd
- COCO Segmentation: **12%** better than 2nd

ILSVRC 2015 classification winner (3.6% top 5 error) -- better than “human performance”! (Russakovsky 2014)

ResNet

نتایج در چالش بازشناسی دیداری در مقیاس بالا ImageNet در مقایسه با سایر روش‌ها

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



معماری‌های گوناگون
شبکه‌های عصبی کانوولوشنال

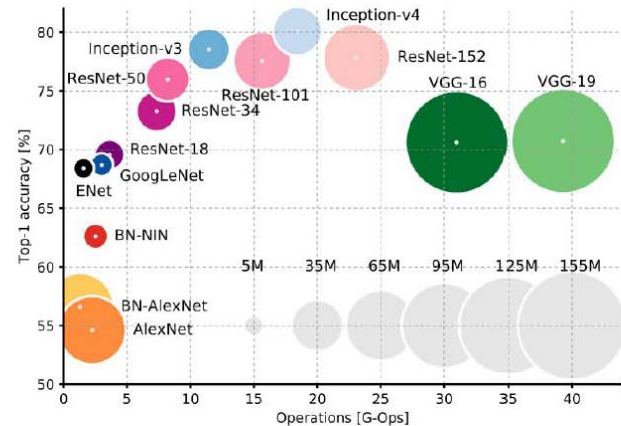
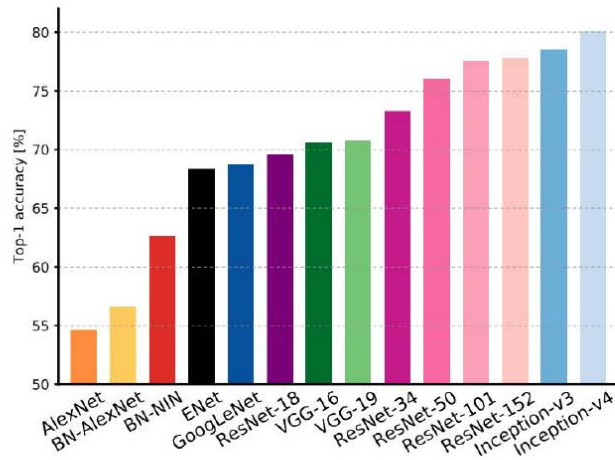
۲

مقایسه‌ی معماری‌ها

معماری‌های گوناگون شبکه‌های عصبی کانولوشنال

مقایسه‌ی پیچیدگی

Comparing complexity...



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

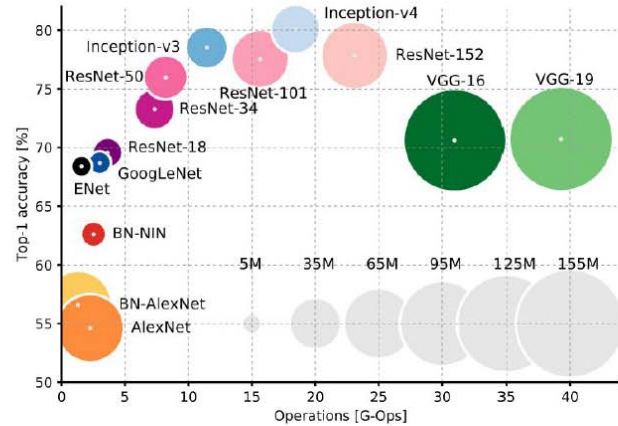
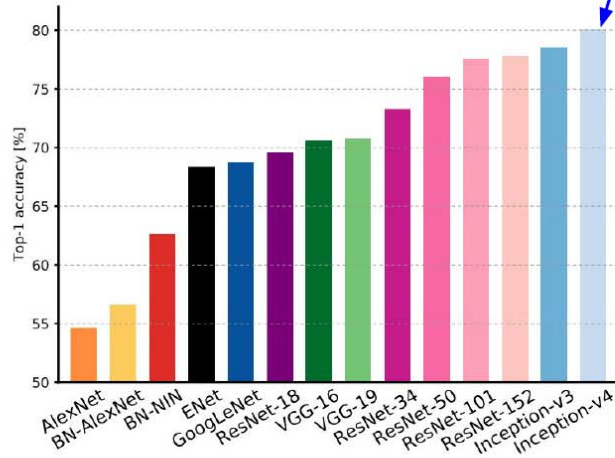
Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

معماری‌های گوناگون شبکه‌های عصبی کانولوشنال

مقایسه‌ی پیچیدگی: Inception-v4

Comparing complexity...

Inception-v4: Resnet + Inception!



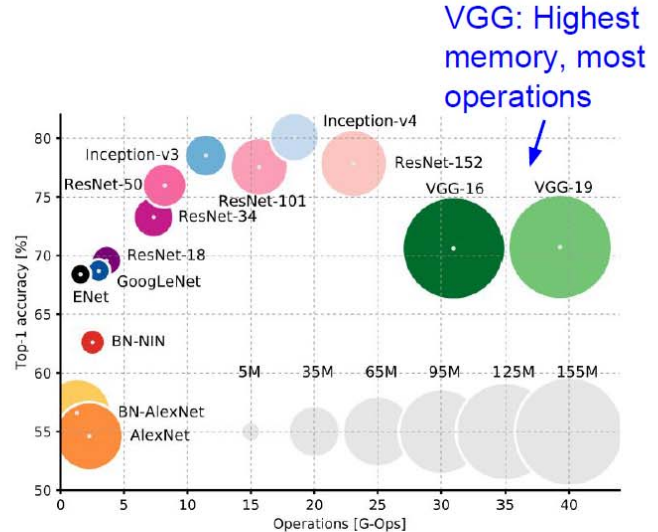
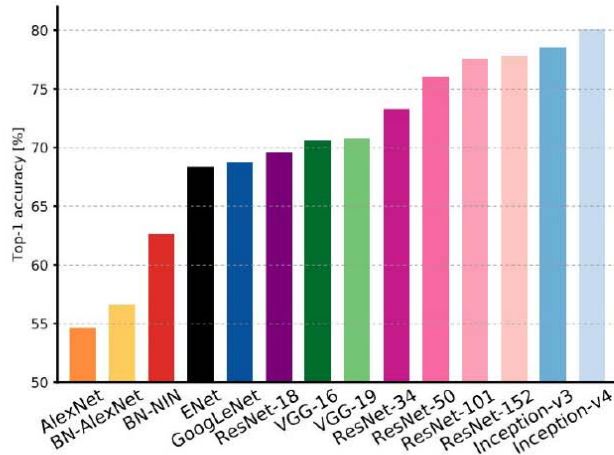
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

معماری‌های گوناگون شبکه‌های عصبی کانولوشنال

مقایسه‌ی پیچیدگی: VGG

Comparing complexity...



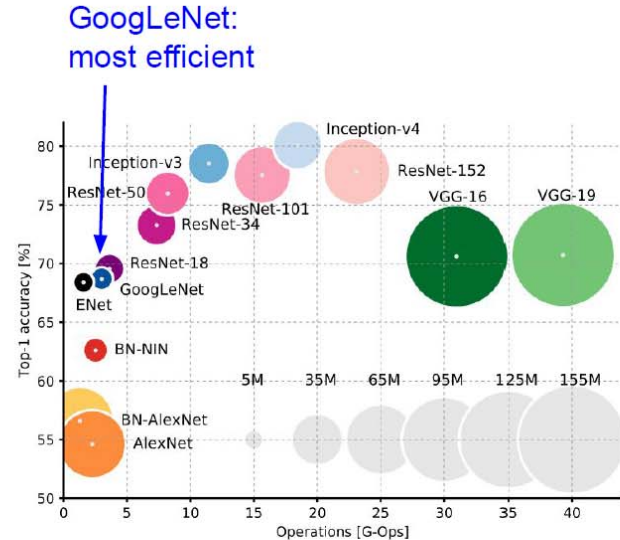
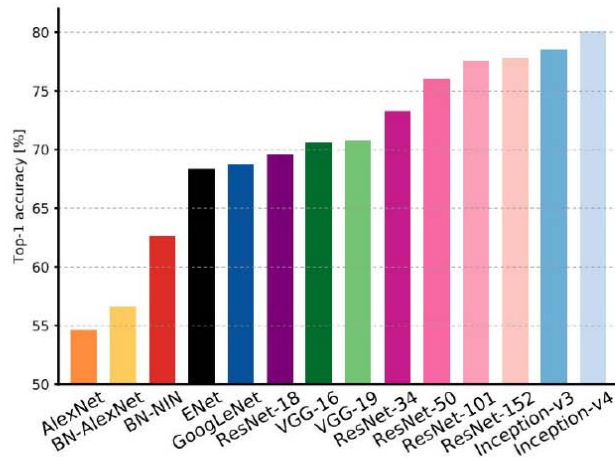
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

معماری‌های گوناگون شبکه‌های عصبی کانولوشنال

مقایسه‌ی پیچیدگی: GoogLeNet

Comparing complexity...



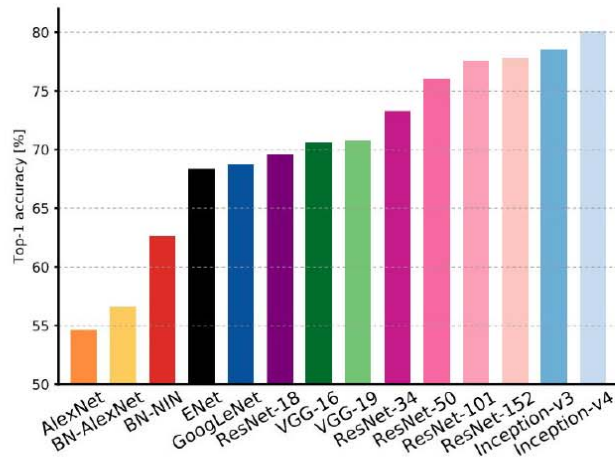
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

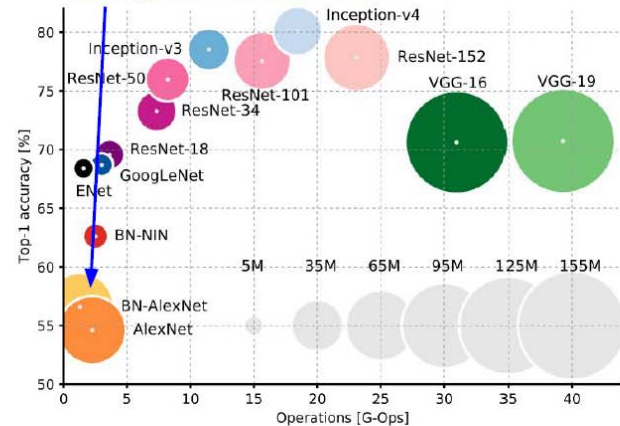
معماری‌های گوناگون شبکه‌های عصبی کانولوشنال

مقایسه‌ی پیچیدگی: AlexNet

Comparing complexity...



AlexNet:
Smaller compute, still memory heavy, lower accuracy



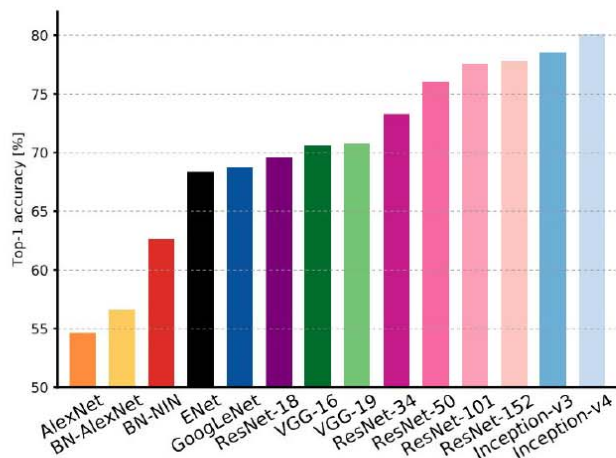
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

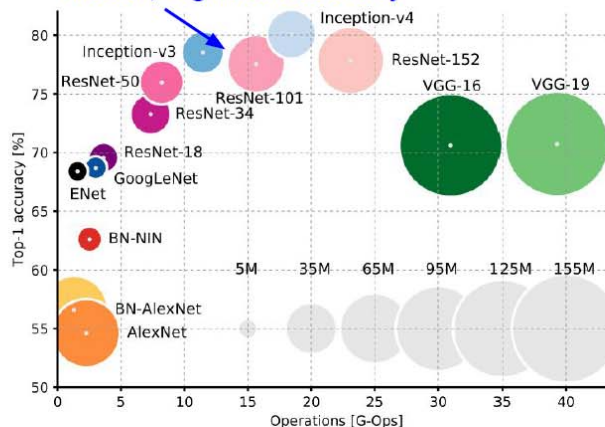
معماری‌های گوناگون شبکه‌های عصبی کانولوشنال

مقایسه‌ی پیچیدگی: ResNet

Comparing complexity...



ResNet:
Moderate efficiency depending on
model, highest accuracy



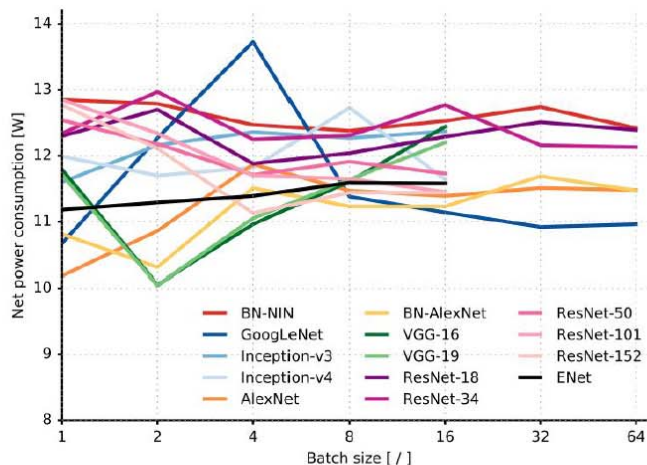
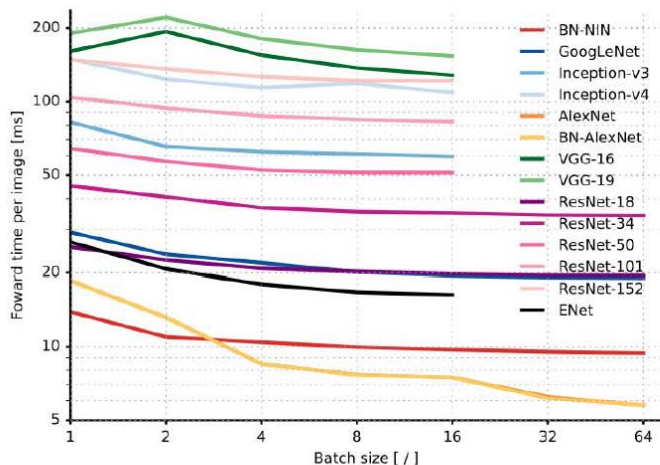
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

معماری‌های گوناگون شبکه‌های عصبی کانولوشنال

مقایسه‌ی زمان اجرای گذر پیشرو و توان مصرفی

Forward pass time and power consumption



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

معماری‌های گوناگون
شبکه‌های عصبی کانوولوشنال

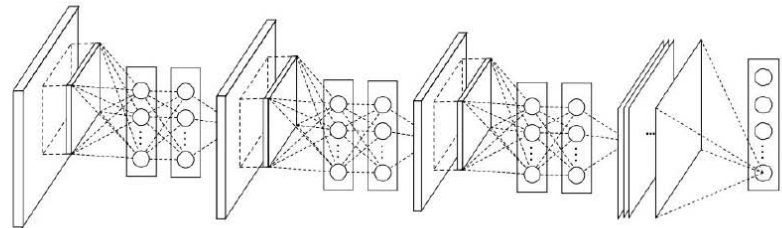
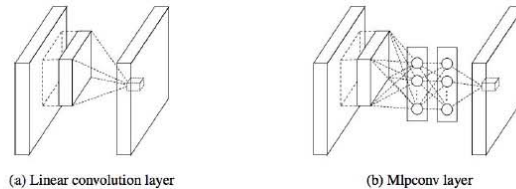
۲

سایر
معماری‌ها

Network in Network (NiN)

[Lin et al. 2014]

- Mlpconv layer with “micronetwork” within each conv layer to compute more abstract features for local patches
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)
- Precursor to GoogLeNet and ResNet “bottleneck” layers
- Philosophical inspiration for GoogLeNet



Figures copyright Lin et al., 2014. Reproduced with permission.

ResNet های در حال بهبود

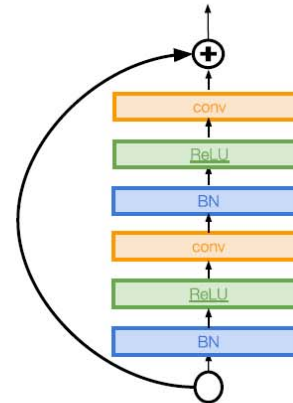
نگاشت های همانی در شبکه های باقیمانده ای عمیق

Improving ResNets...

Identity Mappings in Deep Residual Networks

[He et al. 2016]

- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout network (moves activation to residual mapping pathway)
- Gives better performance



ResNet های در حال بهبود

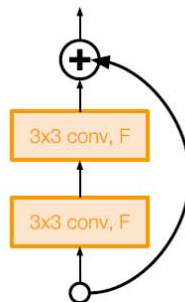
شبکه های باقیمانده ای عریض

Improving ResNets...

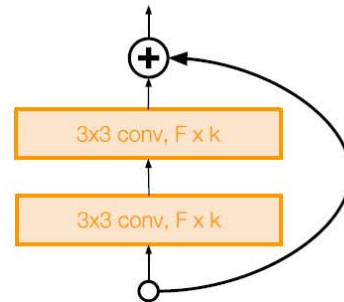
Wide Residual Networks

[Zagoruyko et al. 2016]

- Argues that residuals are the important factor, not depth
- User wider residual blocks ($F \times k$ filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)



Basic residual block



Wide residual block

ResNet های در حال بهبود

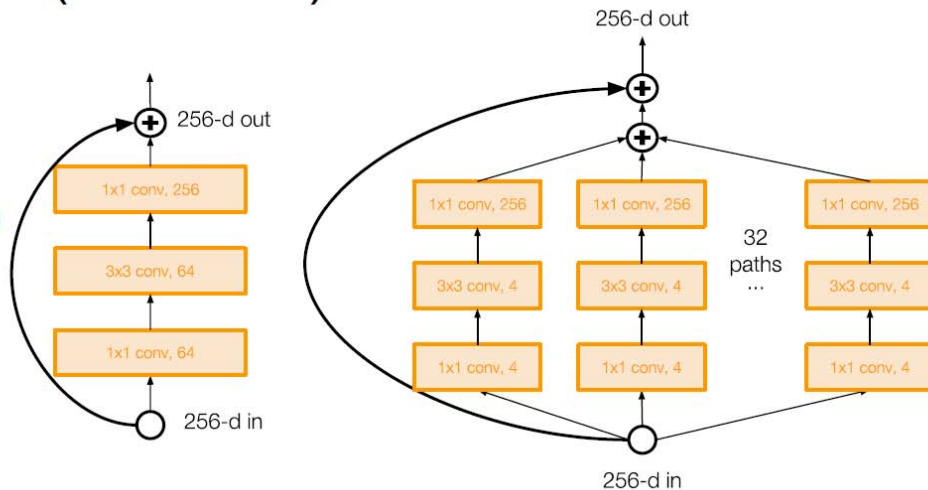
تبدیل های باقیمانده ای تجمیع شده برای شبکه های عصبی عمیق (ResNeXt)

Improving ResNets...

Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)

[Xie et al. 2016]

- Also from creators of ResNet
- Increases width of residual block through multiple parallel pathways (“cardinality”)
- Parallel pathways similar in spirit to Inception module



ResNet های در حال بهبود

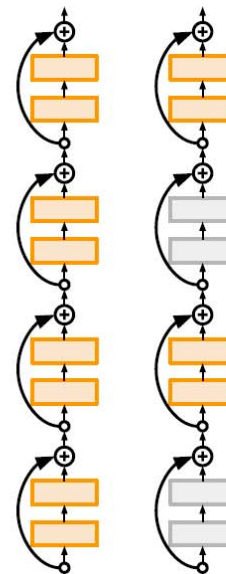
شبکه های عمیق تر با عمق اتفاقی

Improving ResNets...

Deep Networks with Stochastic Depth

[Huang et al. 2016]

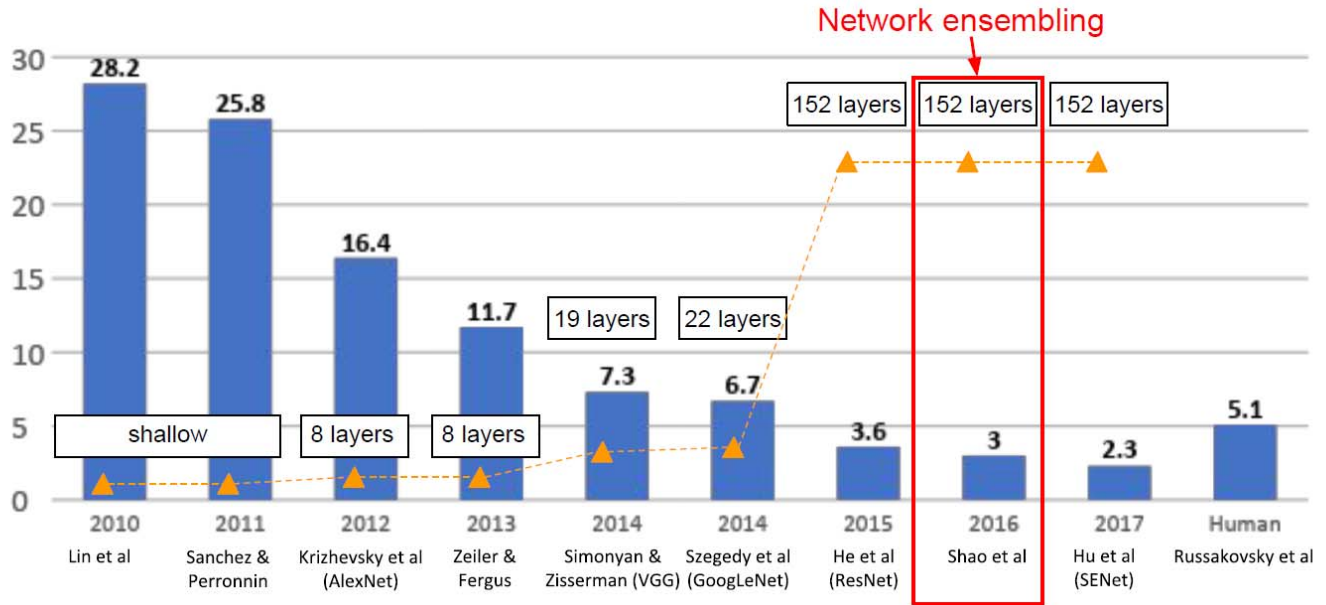
- Motivation: reduce vanishing gradients and training time through short networks during training
- Randomly drop a subset of layers during each training pass
- Bypass with identity function
- Use full deep network at test time



ResNet های در حال بهبود

نتایج در چالش بازشناسی دیداری در مقیاس بالا ImageNet در مقایسه با سایر روش‌ها

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



ResNet های در حال بهبود

تجربه های عملی خوب برای هم جوشی عمیق ویژگی ها (تکنیک یادگیری دسته جمعی)

Improving ResNets...

“Good Practices for Deep Feature Fusion”

[Shao et al. 2016]

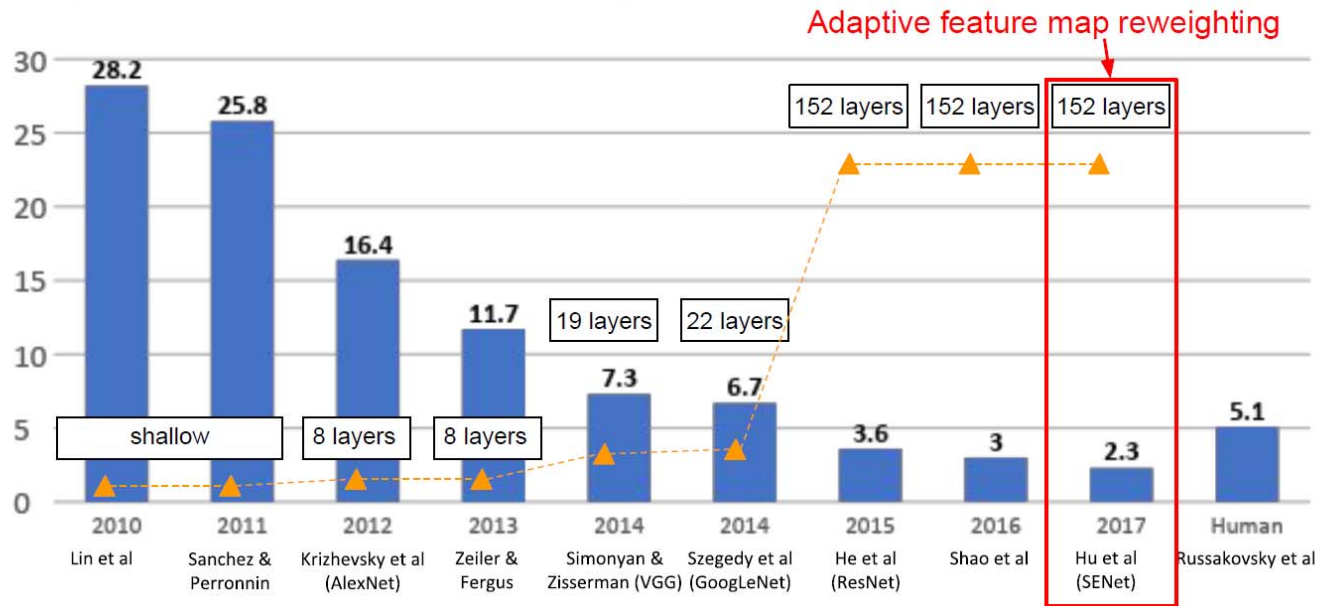
- Multi-scale ensembling of Inception, Inception-Resnet, Resnet, Wide Resnet models
- ILSVRC'16 classification winner

	Inception-v3	Inception-v4	Inception-Resnet-v2	Resnet-200	Wrn-68-3	Fusion (Val.)	Fusion (Test)
Err. (%)	4.20	4.01	3.52	4.26	4.65	2.92 (-0.6)	2.99

ResNet های در حال بهبود

نتایج در چالش بازشناسی دیداری در مقیاس بالا ImageNet در مقایسه با سایر روش‌ها

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



ResNet‌های در حال بهبود

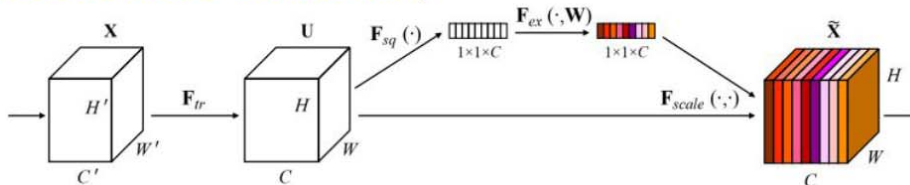
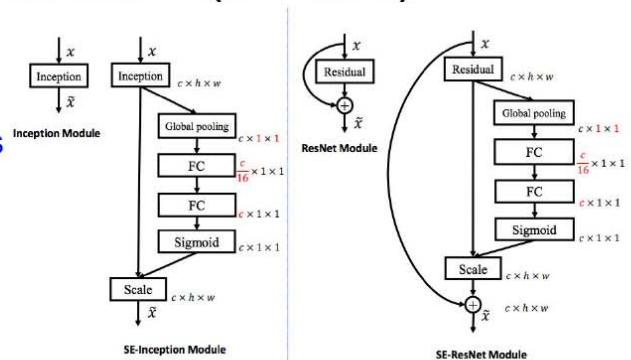
شبکه‌های Squeeze-and-Excitation (SENet)

Improving ResNets...

Squeeze-and-Excitation Networks (SENet)

[Hu et al. 2017]

- Add a “feature recalibration” module that learns to adaptively reweight feature maps
- Global information (global avg. pooling layer) + 2 FC layers used to determine feature map weights
- ILSVRC'17 classification winner (using ResNeXt-152 as a base architecture)



فرا تر از ResNet ها

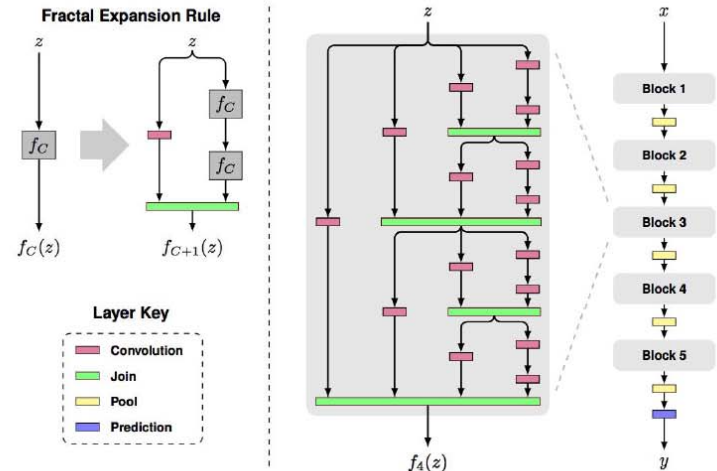
شبکه‌ی فراکتالی: شبکه‌های عصبی ماورای عمیق بدون استفاده از باقیمانده‌ای‌ها

Beyond ResNets...

FractalNet: Ultra-Deep Neural Networks without Residuals

[Larsson et al. 2017]

- Argues that key is transitioning effectively from shallow to deep and residual representations are not necessary
- Fractal architecture with both shallow and deep paths to output
- Trained with dropping out sub-paths
- Full network at test time



Figures copyright Larsson et al., 2017. Reproduced with permission.

فرا تر از ResNet ها

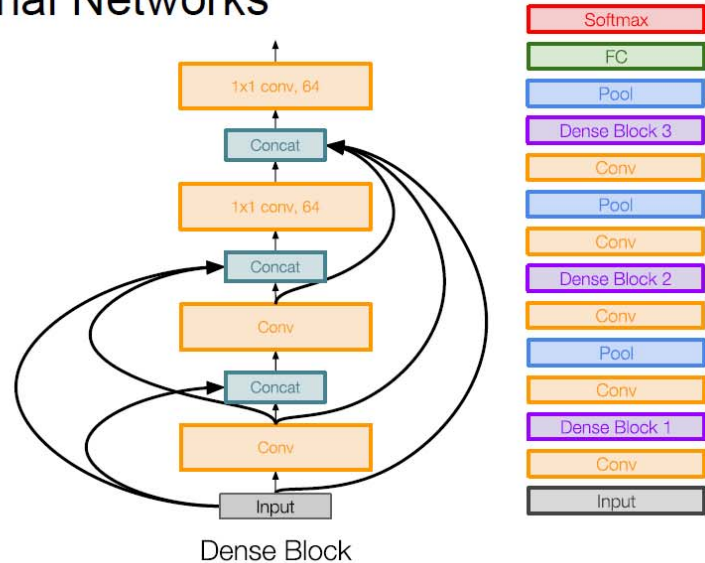
شبکه‌های کانولوشنال متصل متراکم

Beyond ResNets...

Densely Connected Convolutional Networks

[Huang et al. 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse



شبکه‌های کارآمد

SqueezeNet

Efficient networks...

SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size

[Iandola et al. 2017]

- Fire modules consisting of a 'squeeze' layer with 1x1 filters feeding an 'expand' layer with 1x1 and 3x3 filters
- AlexNet level accuracy on ImageNet with 50x fewer parameters
- Can compress to 510x smaller than AlexNet (0.5Mb)

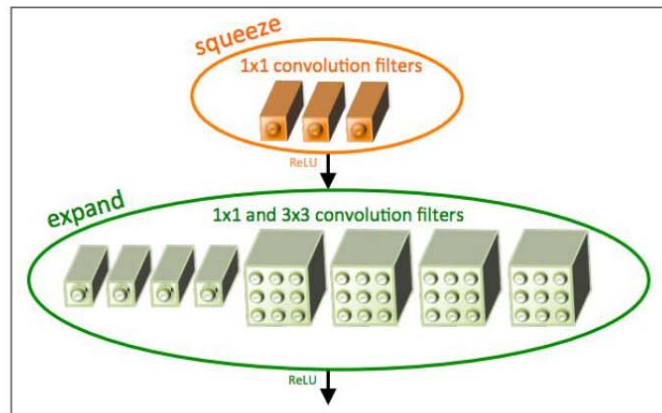


Figure copyright Iandola, Han, Moskewicz, Ashraf, Dally, Keutzer, 2017. Reproduced with permission.

فرا-یادگیری

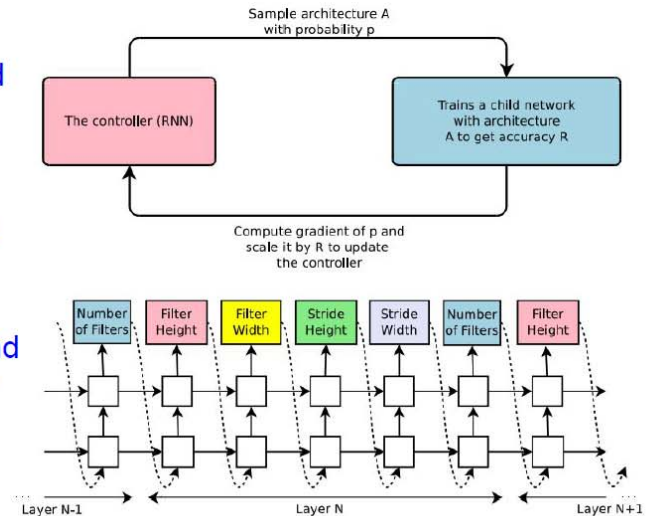
یادگرفتن یادگیری معماری‌های شبکه: جستجوی معماری عصبی با یادگیری تقویتی (NAS)

Meta-learning: Learning to learn network architectures...

Neural Architecture Search with Reinforcement Learning (NAS)

[Zoph et al. 2016]

- “Controller” network that learns to design a good network architecture (output a string corresponding to network design)
- Iterate:
 - 1) Sample an architecture from search space
 - 2) Train the architecture to get a “reward” R corresponding to accuracy
 - 3) Compute gradient of sample probability, and scale by R to perform controller parameter update (i.e. increase likelihood of good architecture being sampled, decrease likelihood of bad architecture)



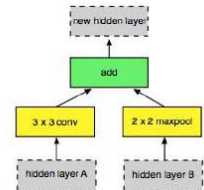
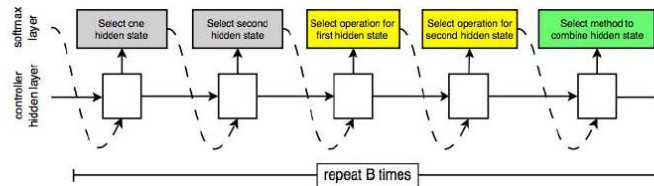
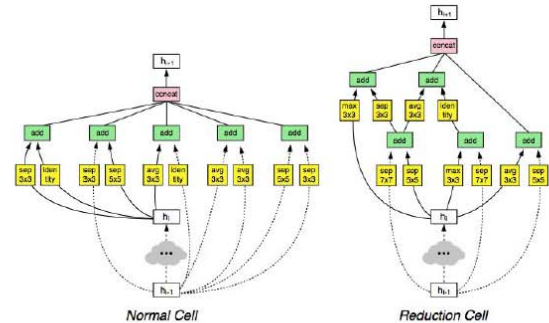
فرا-یادگیری

یادگرفتن یادگیری معماری‌های شبکه: یادگیری معماری‌های انتقال‌پذیر برای بازشناسی مقیاس‌پذیر تصاویر

Meta-learning: Learning to learn network architectures... Learning Transferable Architectures for Scalable Image Recognition

[Zoph et al. 2017]

- Applying neural architecture search (NAS) to a large dataset like ImageNet is expensive
- Design a search space of building blocks (“cells”) that can be flexibly stacked
- NASNet: Use NAS to find best cell structure on smaller CIFAR-10 dataset, then transfer architecture to ImageNet



Summary: CNN Architectures

Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

Also....

- NiN (Network in Network)
- Wide ResNet
- ResNeXT
- Stochastic Depth
- Squeeze-and-Excitation Network
- DenseNet
- FractalNet
- SqueezeNet
- NASNet

معماری‌های شبکه‌های عصبی کانولوشنال

جمع‌بندی


- ❖ VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- ❖ ResNet current best default, also consider SENet when available
- ❖ Trend towards **extremely deep** networks
- ❖ Significant research centers around **design of layer / skip connections** and **improving gradient flow**
- ❖ Efforts to investigate necessity of depth vs. width and residual connections
- ❖ Even more recent trend towards **meta-learning**

معماری‌های گوناگون
شبکه‌های عصبی کانوولوشنال

۳

منابع


منبع اصلی

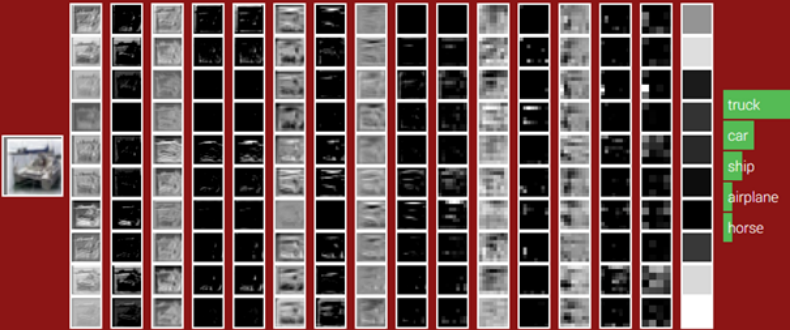


CS231n: Convolutional Neural Networks for Visual Recognition

Spring 2019

Previous Years: [\[Winter 2015\]](#) [\[Winter 2016\]](#) [\[Spring 2017\]](#) [\[Spring 2018\]](#)





*This network is running live in your browser

Course Description

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification, localization and detection. Recent developments in neural network (aka "deep learning") approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into details of the deep learning architectures with a focus on learning end-to-end models for these tasks, particularly image classification. During the 10-week course, students will learn to implement, train and debug their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. The final assignment will involve training a multi-million parameter convolutional neural network and applying it on the largest image classification dataset (ImageNet). We will focus on teaching how to set up the problem of image recognition, the learning algorithms (e.g. backpropagation), practical engineering tricks for training and fine-tuning the networks and guide the students through hands-on assignments and a final course project. Much of the background and materials of this course will be drawn from the [ImageNet Challenge](#).

<http://cs231n.stanford.edu>

<http://cs231n.github.io/convolutional-networks/>