CNN Architectures (2)

محمد جواد فدائی‌اسلام

Case studies

- Modern network architectures
  - Inception (GoogLeNet)
  - ResNet
  - ResNeXt
  - DenseNet

To create better deep learning model

- You can just make a bigger model, either in the number of layers, or the number of neurons in each layer.
- But this can often create complications:
  - **Bigger the model, more vulnerable to overfitting.**
    - This is particularly noticeable when the training data is small
  - **Increasing the number of parameters means you need to increase your existing computational resources.**

ILSVRC 2014 classification Challenge ImageNet2014

	Team	Year	Place	Error (top-5)	Uses external data
AlexNet	SuperVision	2012	1st	16.4%	no
	SuperVision	2012	1st	15.3%	Imagenet 22k
ZFNet	Clarifai	2013	1st	11.7%	no
	Clarifai	2013	1st	11.2%	Imagenet 22k
SPPNet	MSRA	2014	3rd	7.35%	no
VGGNet	VGG	2014	2nd	7.32%	no
GoogLeNet	GoogLeNet	2014	1st	6.67%	no

GoogLeNet

- Dataset: ILSVRC2014 (ImageNet2014)
- 12 times fewer parameters than AlexNet, while being significantly more accurate.
  - AlexNet: 60M parameters
  - GoogLeNet: 4M parameters

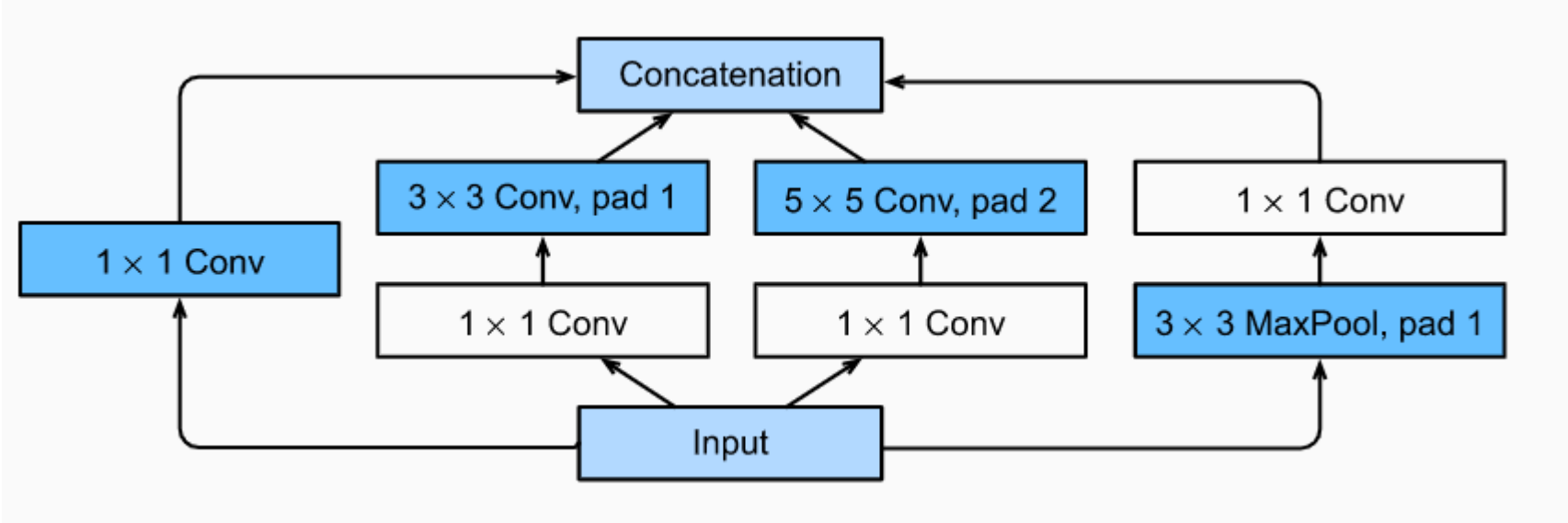
GoogLeNet (Kernel Size)

One focus of GoogLeNet was to address the question of which sized convolution kernels are best.

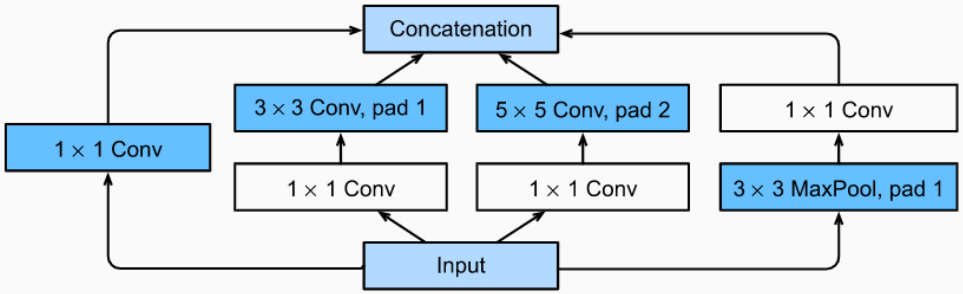
Previous popular networks employed choices as small as 1×1 and as large as 11×11.

One insight in GoogLeNet was that sometimes it can be advantageous to employ a combination of variously-sized kernels.

Inception Block
The basic convolutional block in GoogLeNet

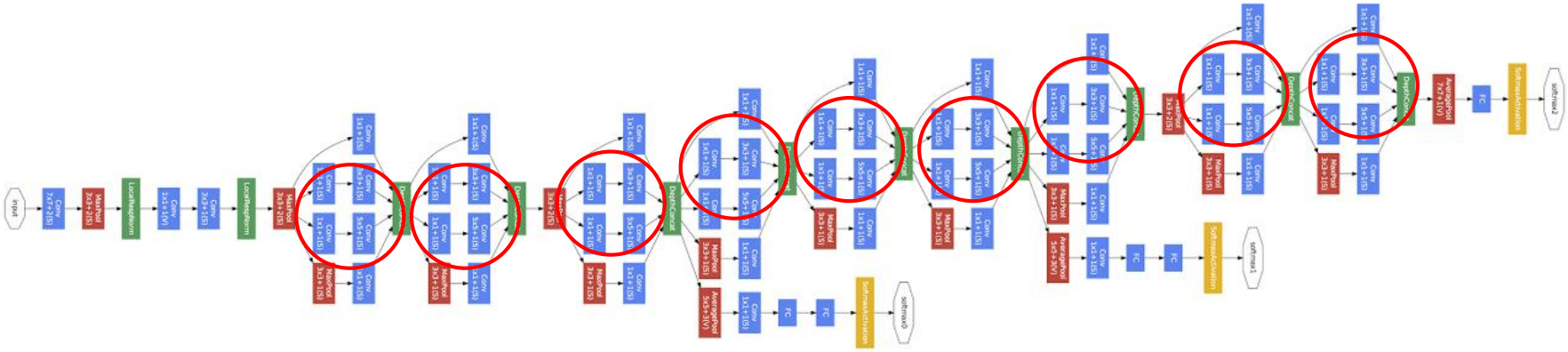


Inception Block



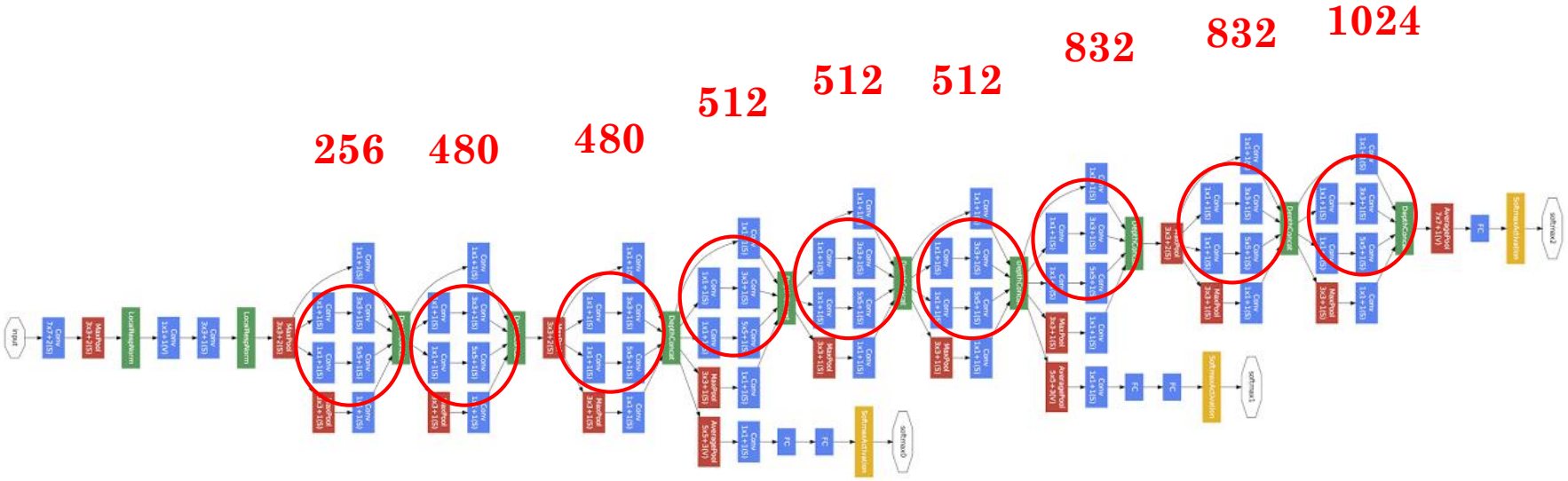
- It consists of four parallel paths.
- The first three paths use 1×1 , 3×3 , and 5×5 conv. layers to extract information from different spatial sizes.
- The middle two paths perform a 1×1 convolution on the input to reduce the number of channels.
- The fourth path uses a 3×3 maximum pooling layer, followed by a 1×1 convolutional layer to change the number of channels.
- The four paths all use appropriate padding to give the input and output the same height and width.
- Finally, the outputs along each path are concatenated

9 Inception modules



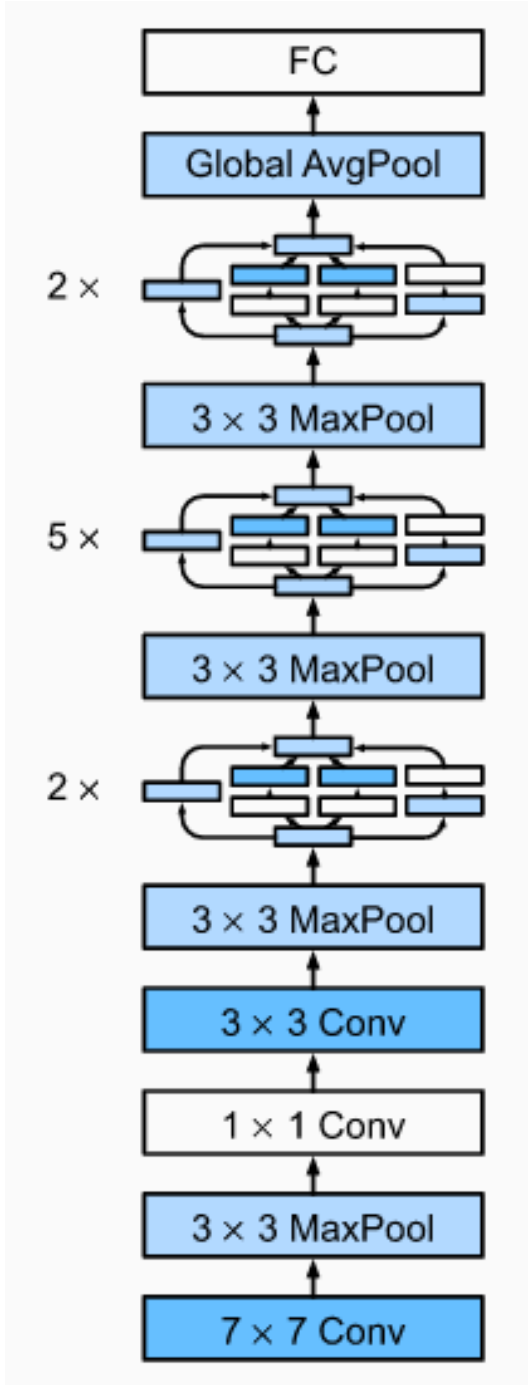
Convolution
Pooling
Softmax
Other

Inception



Width of inception modules ranges from 256 filters to 1024.
Can remove fully connected layers on top completely.

The GoogLeNet Architecture

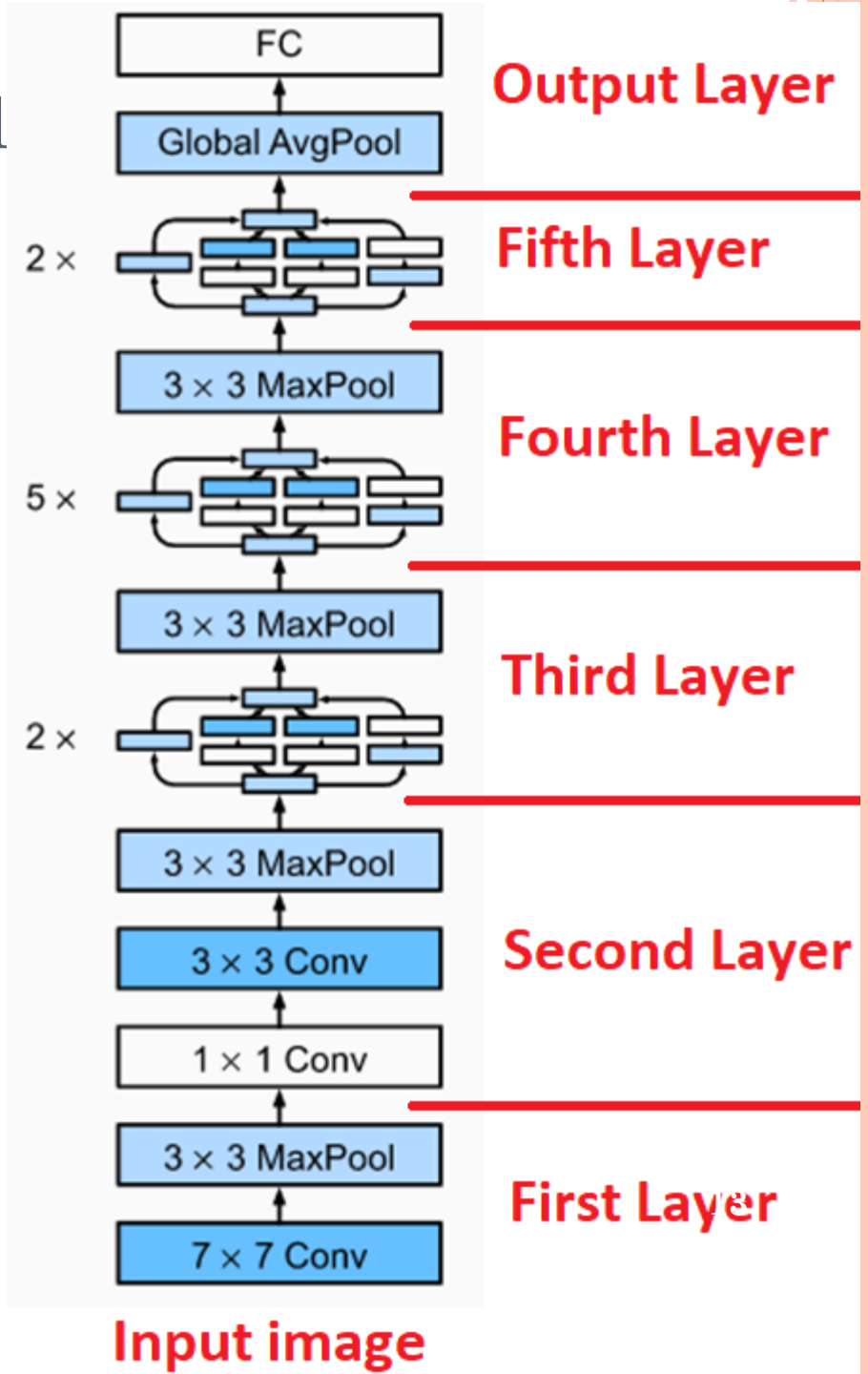


The GoogLeNet Architecture

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

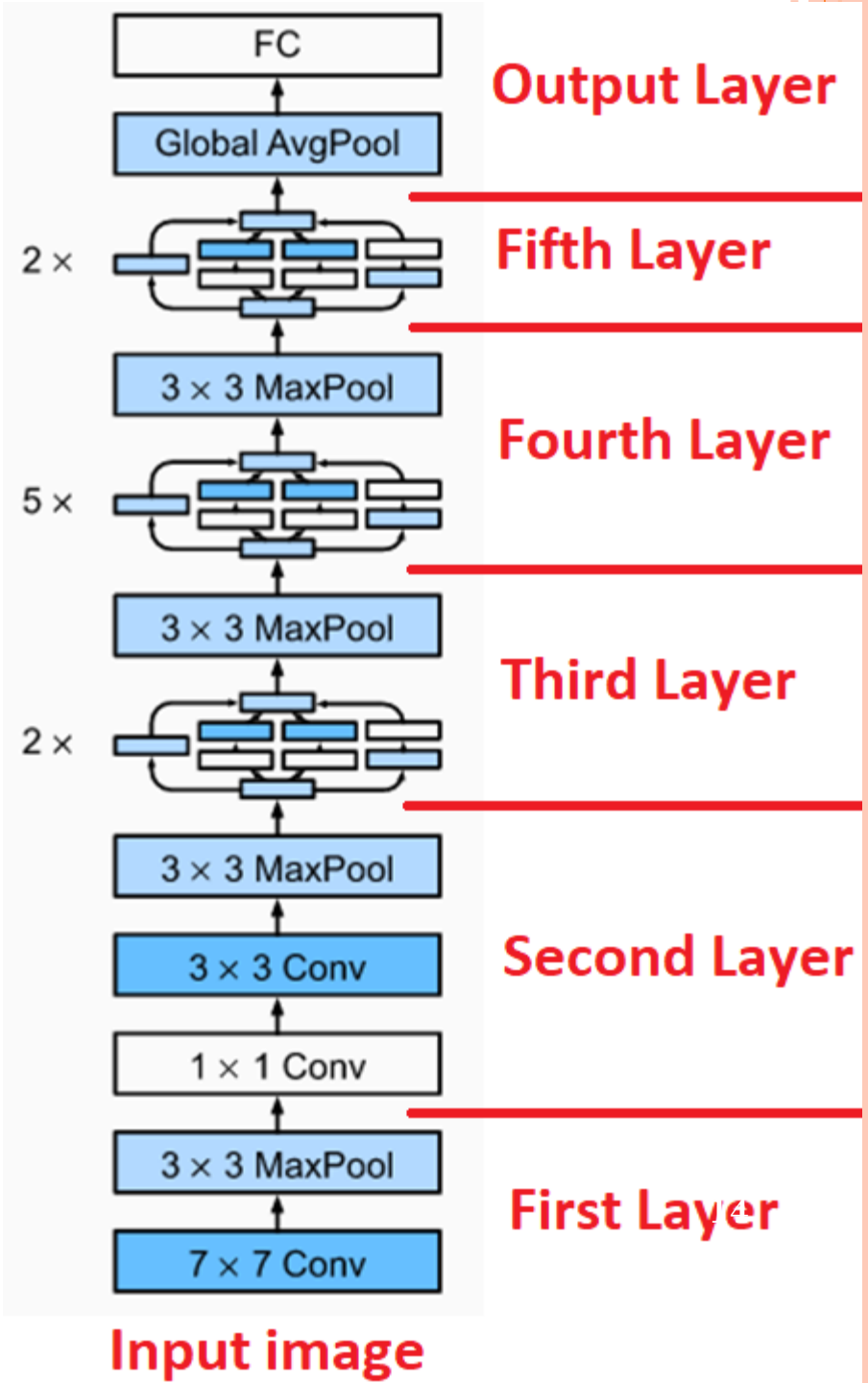
GoogLeNet Arch.(in,1

- The size of input is 224*224 in the RGB color space with zero mean.
- The first module uses a 64-channel 7×7 conv. layer.
- The second module uses two conv. layers:
  - first, a 64-channel 1×1 conv. layer,
  - Second 3×3 conv. layer
- This layer triples the number of channels.



GoogLeNet Arch.(3)

- The third module connects two complete Inception blocks in series.
- The number of output channels of the first Inception block is 64+128+32+32=256.
- The number of output channels of the second Inception block is increased to 128+192+96+64=480.



GoogLeNet Arch.(4)

- The fourth module is more complicated.
- It connects five Inception blocks in series, and they have

  192+208+48+64=512

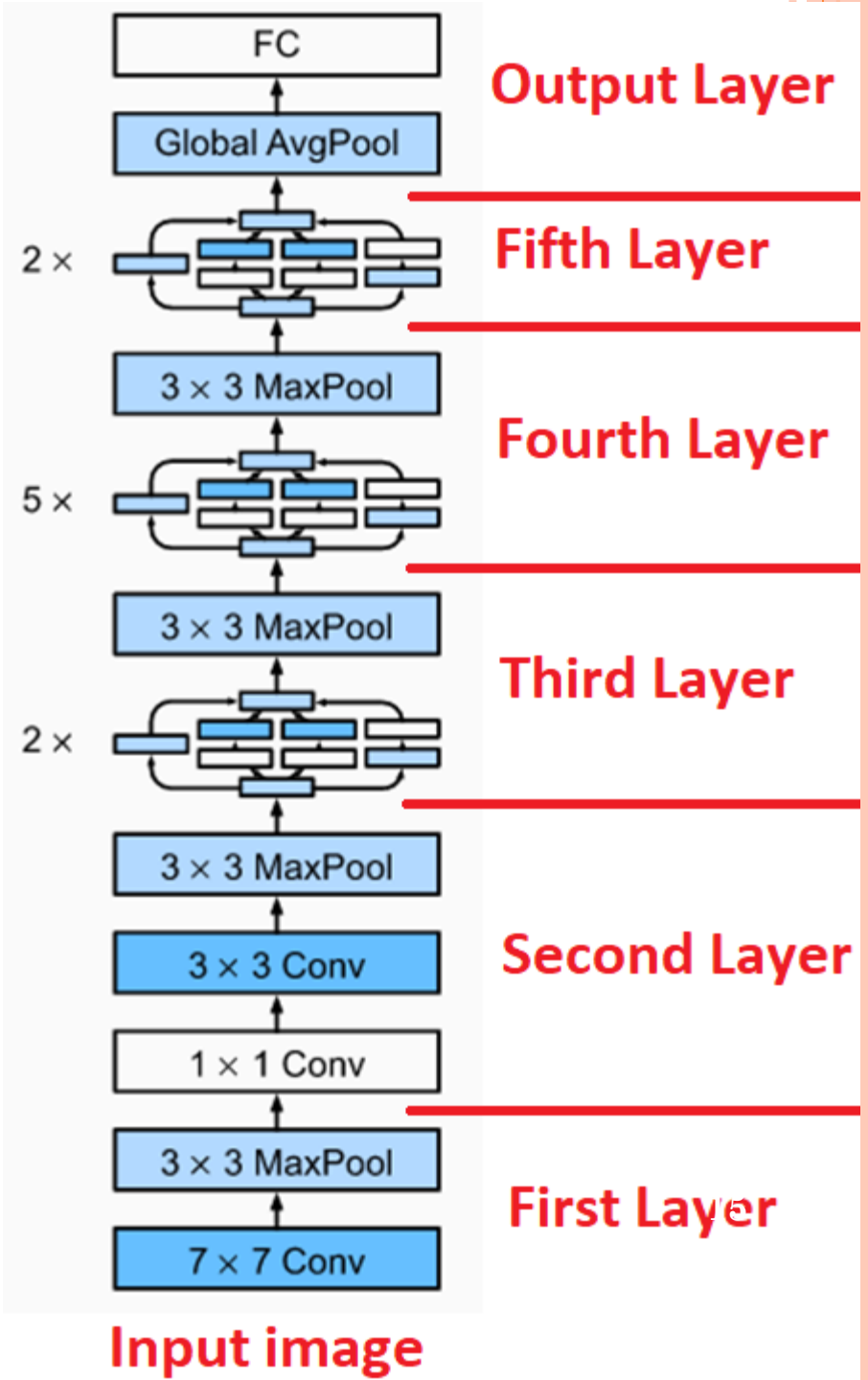
  160+224+64+64=512

  128+256+64+64=512

  112+288+64+64=528

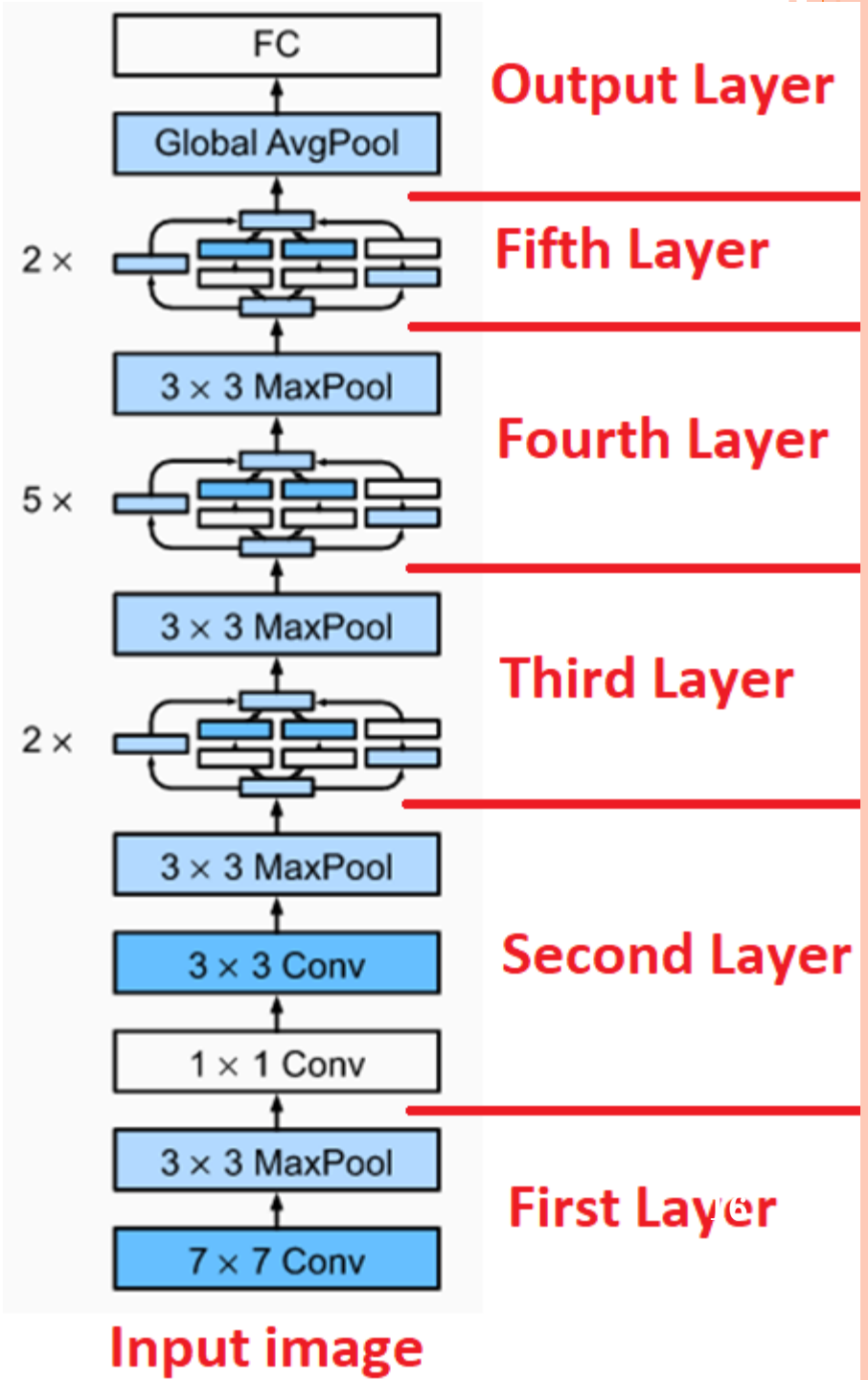
  256+320+128+128=832

output channels, respectively.



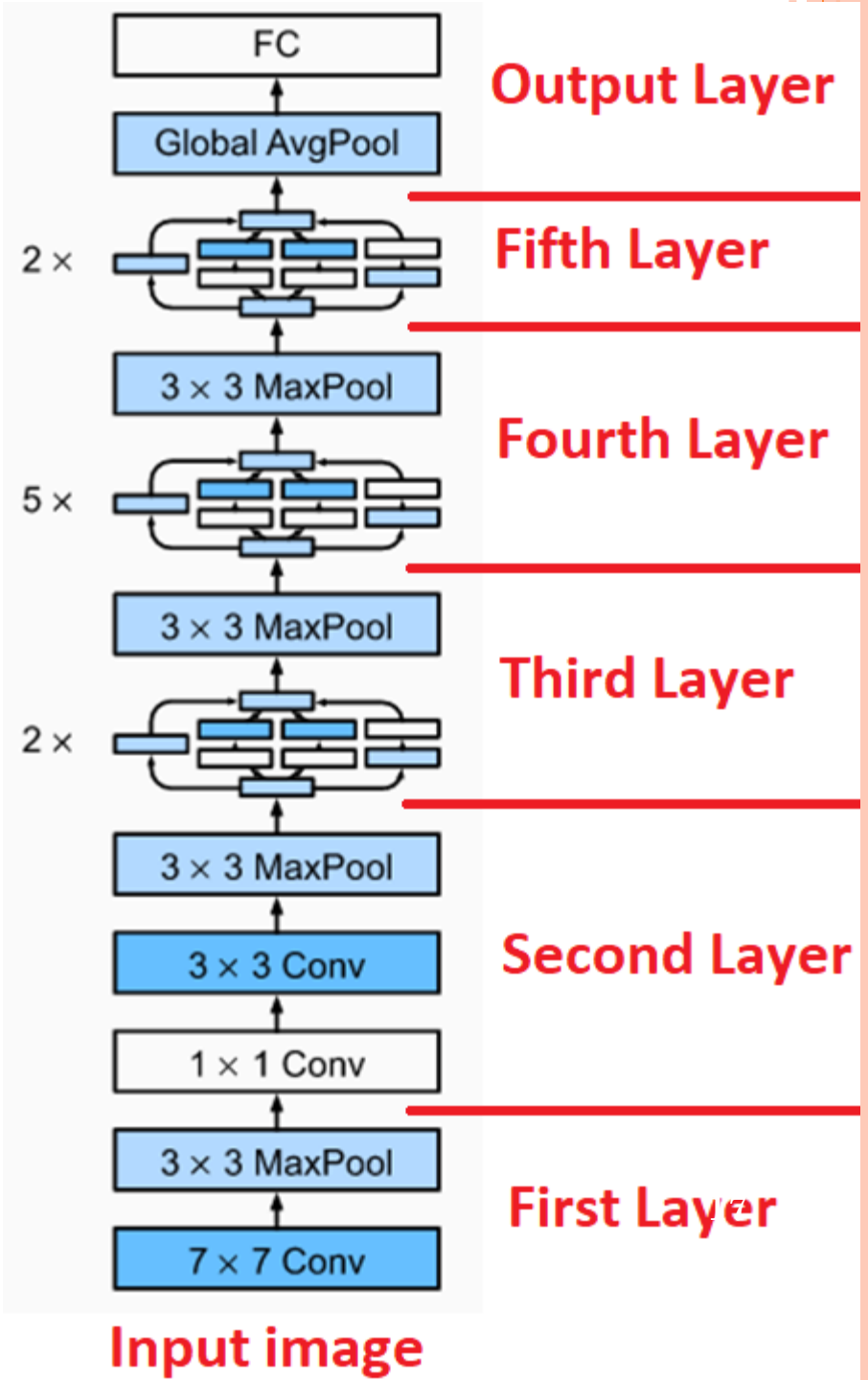
GoogLeNet Arch.(5)

- The fifth module has two Inception blocks with 256+320+128+128=832 and 384+384+128+128=1024 output channels.
- It should be noted that the fifth block is followed by the output layer.
- This block uses the global average pooling layer to change the height and width of each channel to 1.
- Finally, we turn the output into a two-dimensional array followed by a FC layer whose number of outputs is the number of label classes.



GoogLeNet
The Number of Layer

- The network is 22 layers deep when counting only layers with parameters (or 27 layers if we also count pooling).
- The overall number of layers (independent building blocks) used for the construction of the network is about 100



ResNet

- Deep CNNs have led to a series of breakthroughs for image classification.
- Deep networks naturally integrate low/mid/high level features in an end-to-end multilayer fashion,
- The "levels" of features can be enriched by the number of stacked layers (depth).

ResNet

- Driven by the significance of depth, a question arises:

  Is learning better networks as easy as stacking more layers?

- An obstacle to answering this question was the notorious problem of vanishing/exploding gradients, which prevent convergence from the beginning.
- This problem, however, has been largely addressed by normalized initialization and intermediate normalization layers, which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with backpropagation.

DEGRADATION PROBLEM

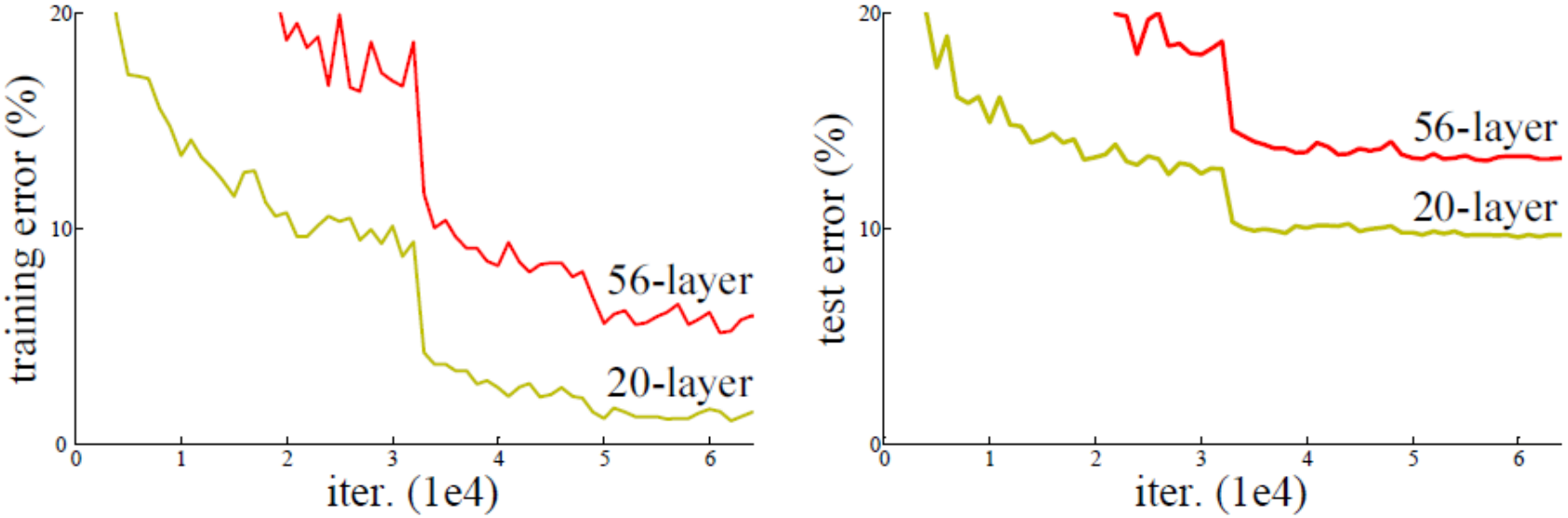
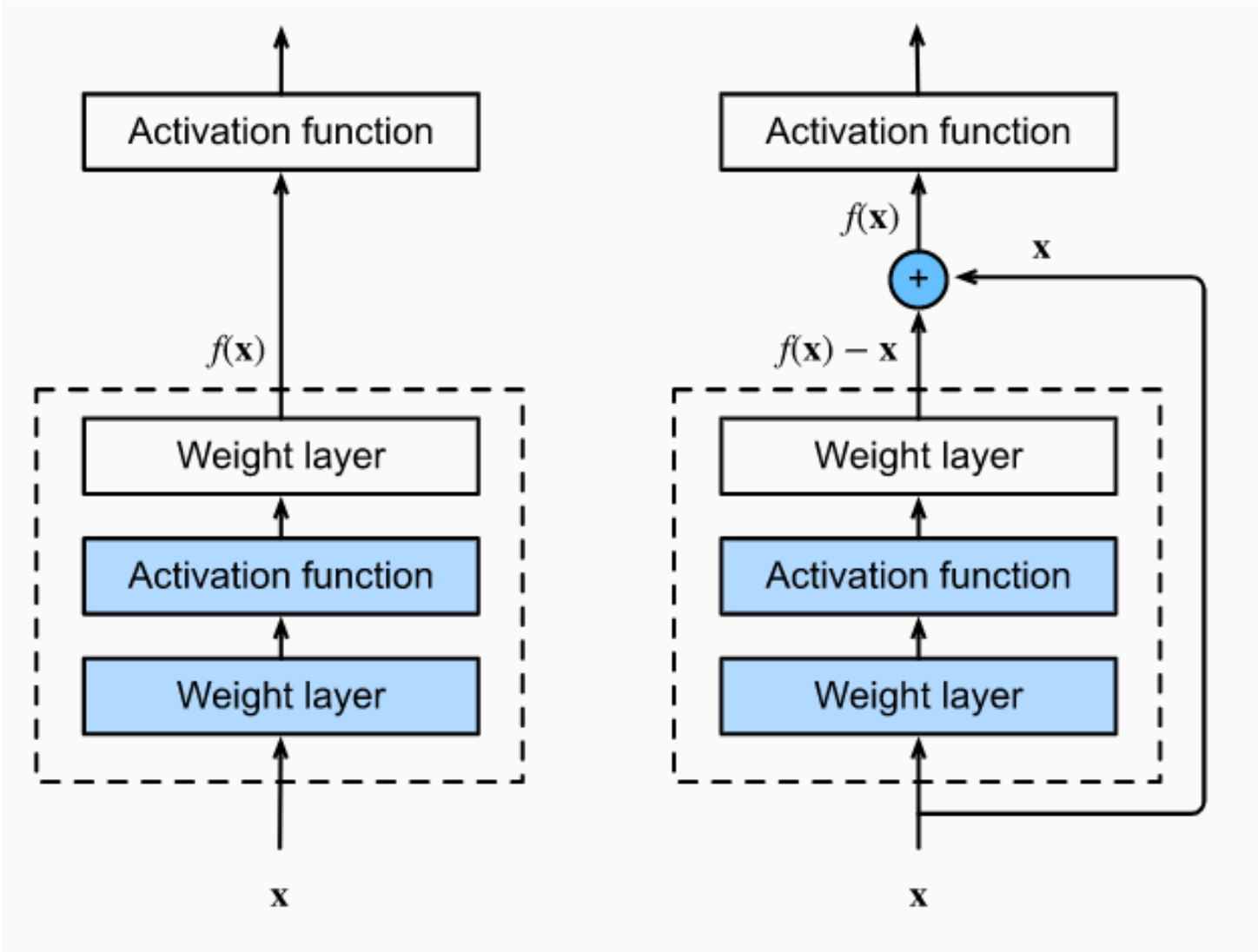


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented

ResNet

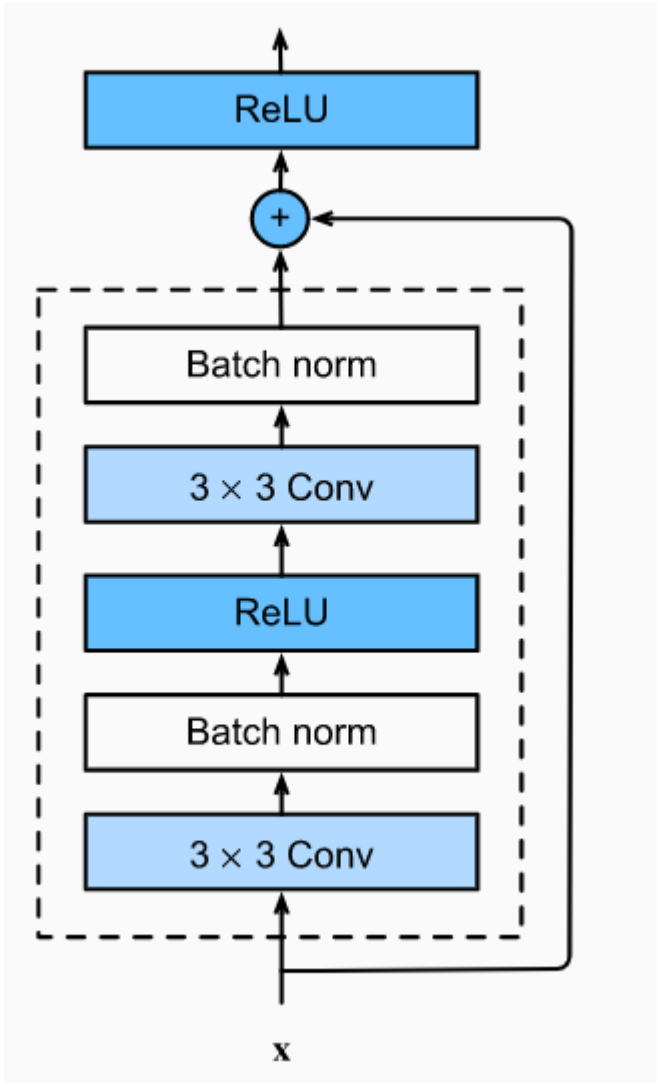
- A widely observed phenomenon in deep learning is the degradation problem: increasing the depth of a network leads to a decrease in performance on both test and training data
- Unexpectedly, such degradation is not caused by overfitting.

A REGULAR BLOCK (LEFT) AND A RESIDUAL BLOCK (RIGHT)

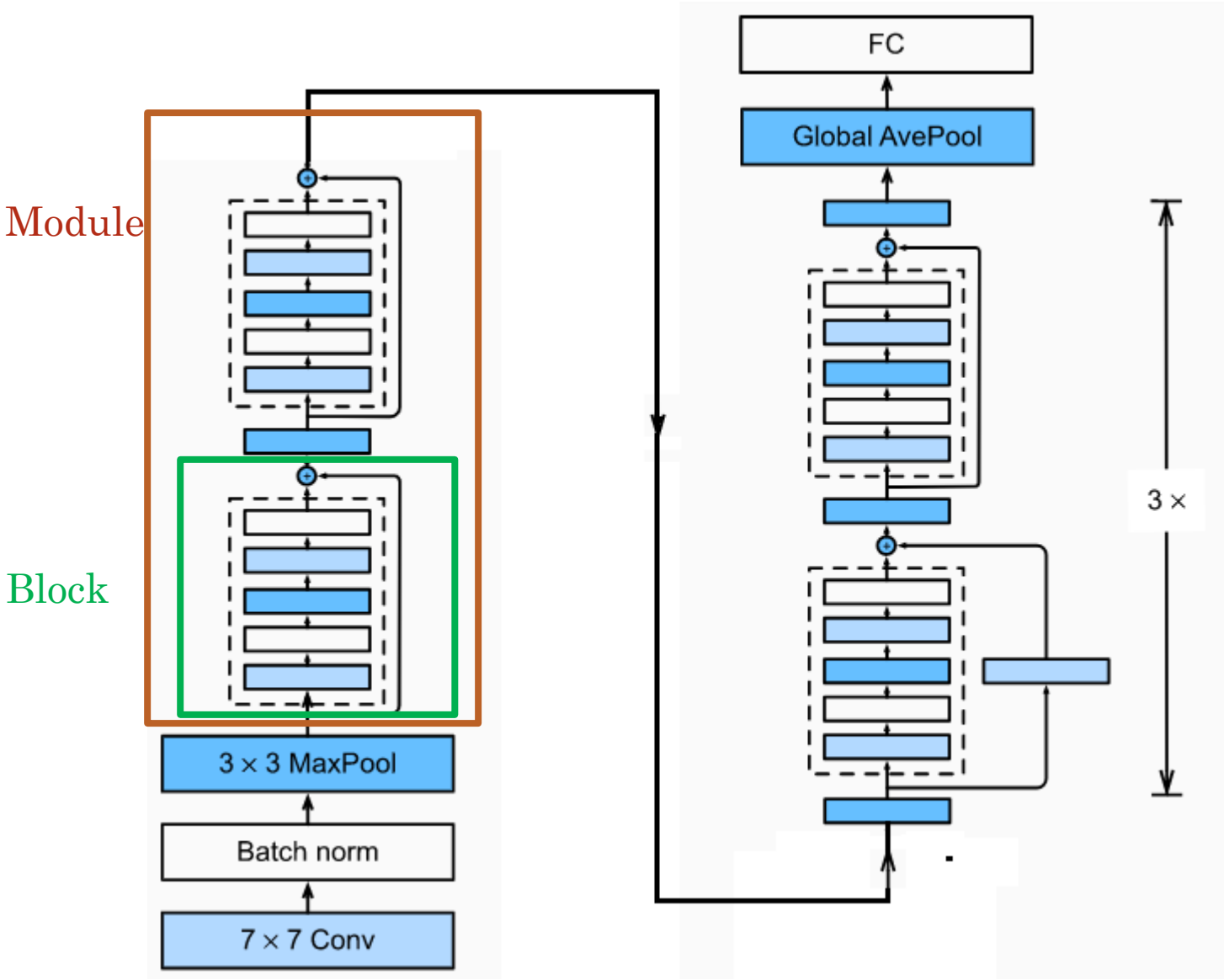


ResNet block

- ResNet follows VGG's full 3×3 convolutional layer design.
- The residual block has two 3×3 convolutional layers.
- This kind of design requires that the output of the two convolutional layers has to be of the same shape as the input, so that they can be added together.



The ResNet-18 architecture

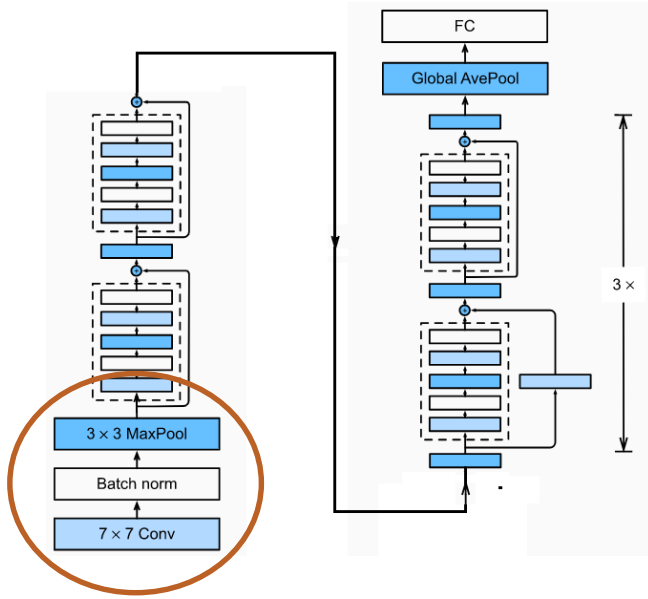


ResNet Model (first two layers)

They are the same as those of the GoogLeNet:

the 7×7 convolutional layer with 64 output channels and a stride of 2 is followed by the 3×3 maximum pooling layer with a stride of 2.

The difference is the batch normalization layer added after each convolutional layer in ResNet.



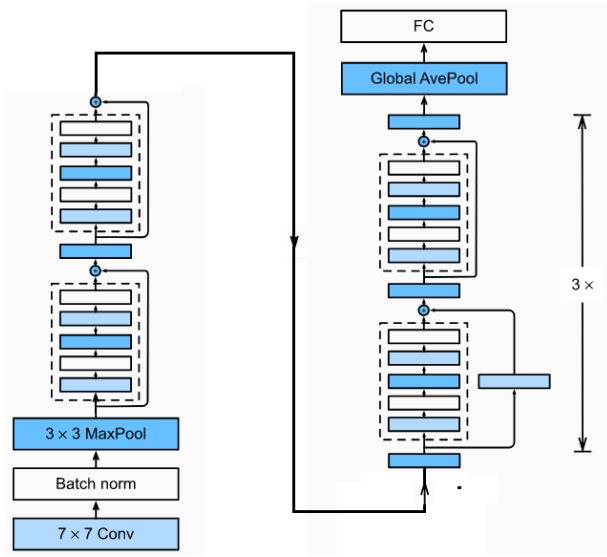
ResNet Model

ResNet uses four modules.

Two residual blocks are used for each module.

The number of channels in the first module is the same as the number of input channels.

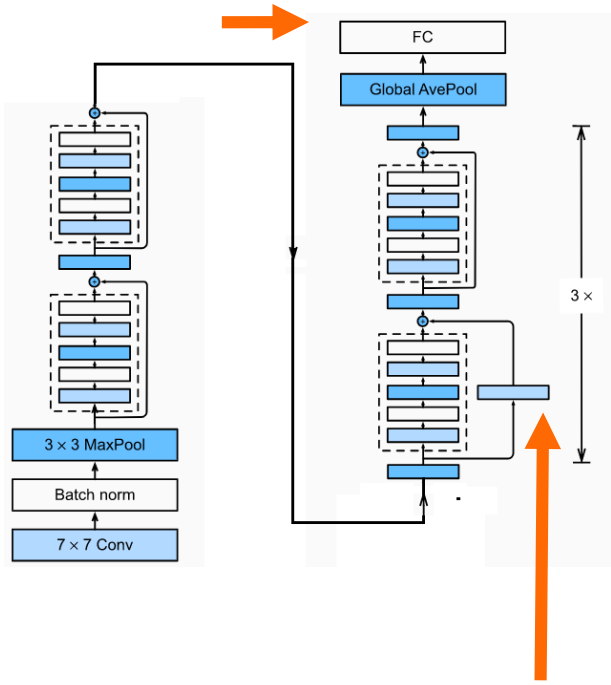
A max pooling layer with a stride of 2 has already been used between two blocks in each module.



ResNet Model

- In the first residual block for each of the subsequent modules, the number of channels is doubled compared with that of the previous module.

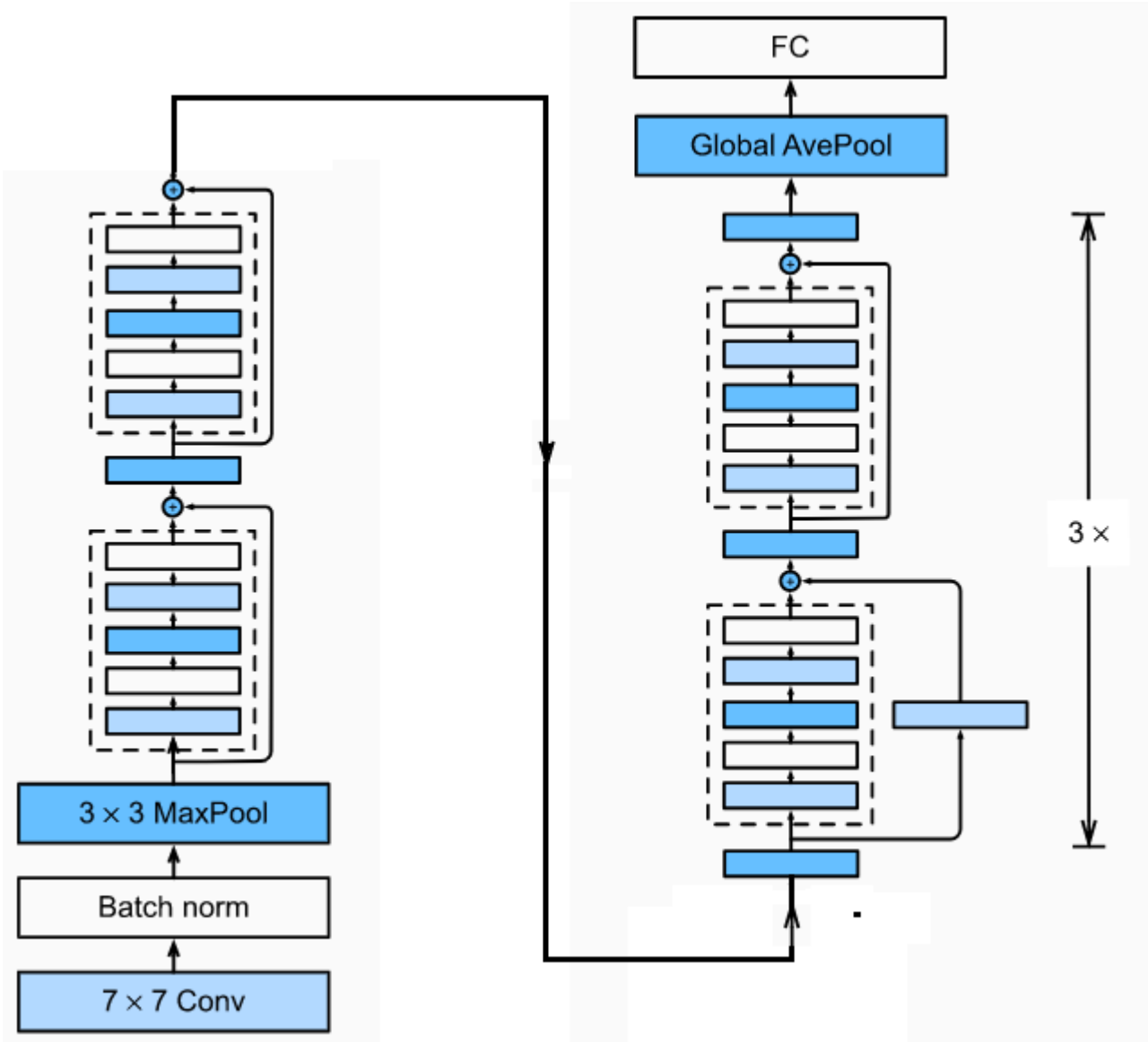
- Finally, just like GoogLeNet, we add a global average pooling layer, followed by the fully-connected layer output.



ResNet Models - layers

- There are 4 convolutional layers in each module (excluding the 1×1 convolutional layer). Together with the first 7×7 convolutional layer and the final fully-connected layer, there are 18 layers in total. Therefore, this model is commonly known as ResNet-18.
- By configuring different numbers of channels and residual blocks in the module, we can create different ResNet models, such as the deeper 152-layer ResNet-152.
- Although the main architecture of ResNet is similar to that of GoogLeNet, ResNet's structure is simpler and easier to modify.

The ResNet-18 architecture



The Structure of ResNet

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

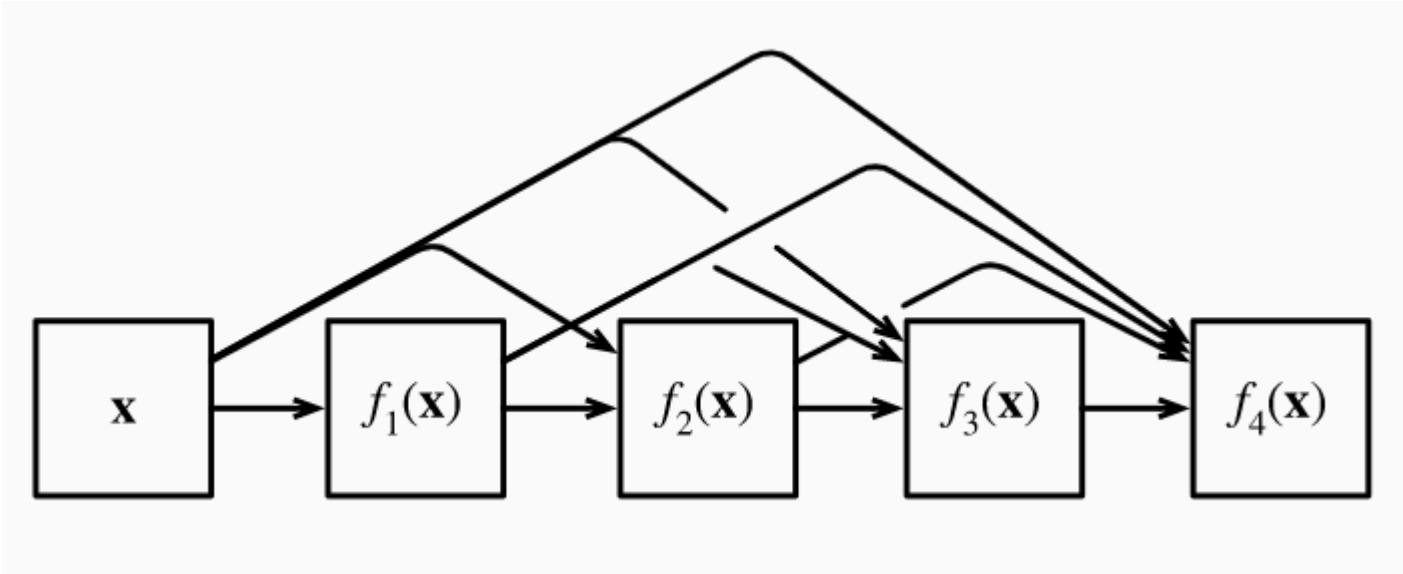
Batch Normalization

- Batch normalization is one of the reasons why deep learning has made such outstanding progress in recent years.
- It enables the use of higher learning rates, greatly accelerating the learning process.
- "covariate shift" refers to the change in the distribution of the input values to a learning algorithm.
- For instance, if the train and test sets come from entirely different sources (e.g. training images come from the web while test images are pictures taken on the iPhone), the distributions would differ.
- The reason covariance shift can be a problem is that the behavior of machine learning algorithms can change when the input distribution changes.

Batch Normalization

- The basic idea behind batch normalization is to limit covariate shift by normalizing the activations of each layer (transforming the inputs to be mean 0 and unit variance).
- This allows each layer to learn on a more stable distribution of inputs, and would thus accelerate the training of the network.

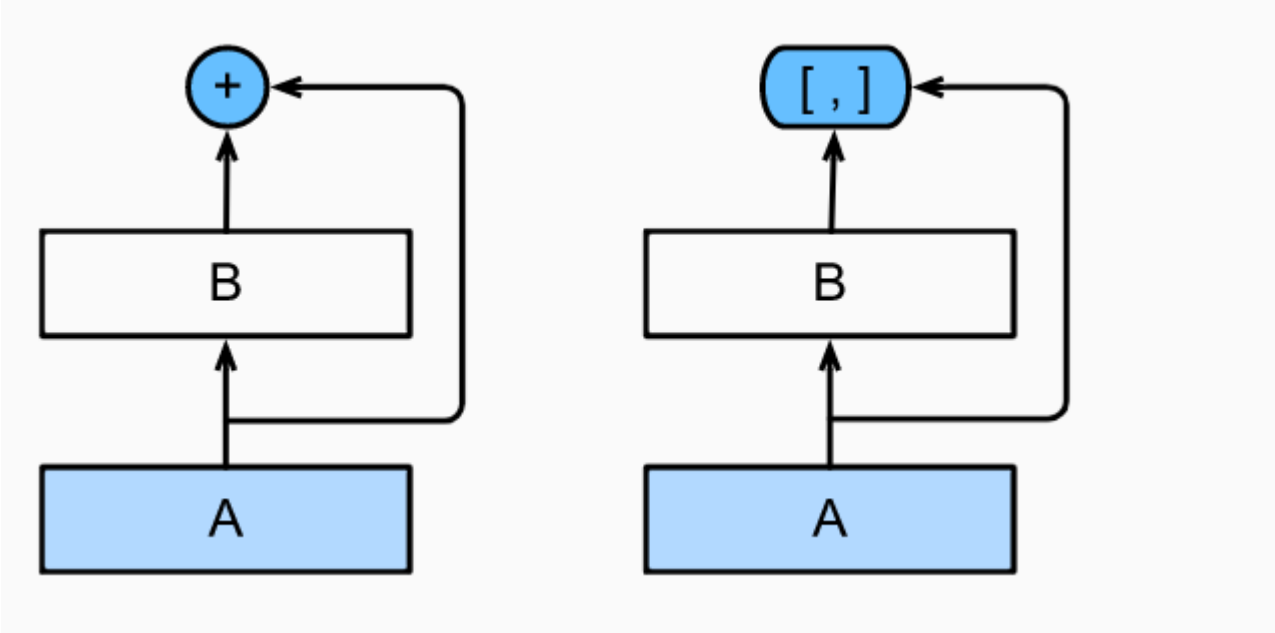
DenseNet2017



- The name DenseNet arises from the fact that the dependency graph between variables becomes quite dense.
- The last layer of such a chain is densely connected to all previous layers.

The main difference between ResNet and DenseNet

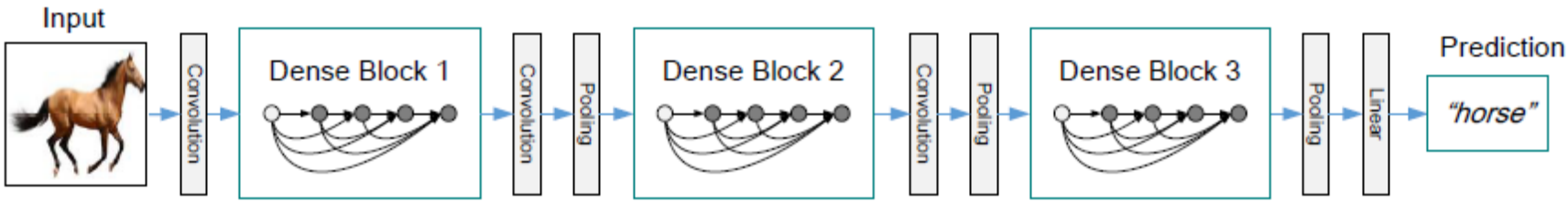
- The main difference between ResNet (left) and DenseNet (right) in cross-layer connections: use of addition and use of concatenation.



DenseNet Architecture

- The main components that compose a DenseNet are
  - *dense blocks* and
  - *transition layers*.
- The former define how the inputs and outputs are concatenated, while the latter control the number of channels so that it is not too large.

A DEEP DENSENET WITH THREE DENSE BLOCKS



- The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.
- A *dense block* consists of multiple convolution blocks, each using the same number of output channels.
- In the forward propagation, however, we concatenate the input and output of each convolution block on the channel dimension.

Transition Layer

- Since each dense block will increase the number of channels, adding too many of them will lead to an excessively complex model.
- A transition layer is used to control the complexity of the model.
- It reduces the number of channels by using the 1×1 convolutional layer and halves the height and width of the average pooling layer with a stride of 2, further reducing the complexity of the model.

DenseNet-121

- DenseNet-121 was used in our proposed network architecture, in which each layer was directly connected to every other layer in a feed-forward fashion.
- It consists four dense blocks, three transition layers and a total of 121 layers (117-conv, 3-transition, and 1-classification).
- Each conv layer corresponds to a composite sequence of operations consisting of batch normalization (BN-Relu-Conv).
- The Classification subnetwork includes 7 × 7 global average pooling, 1000D fully-connected layer, and softmax.

DenseNet architectures for ImageNet

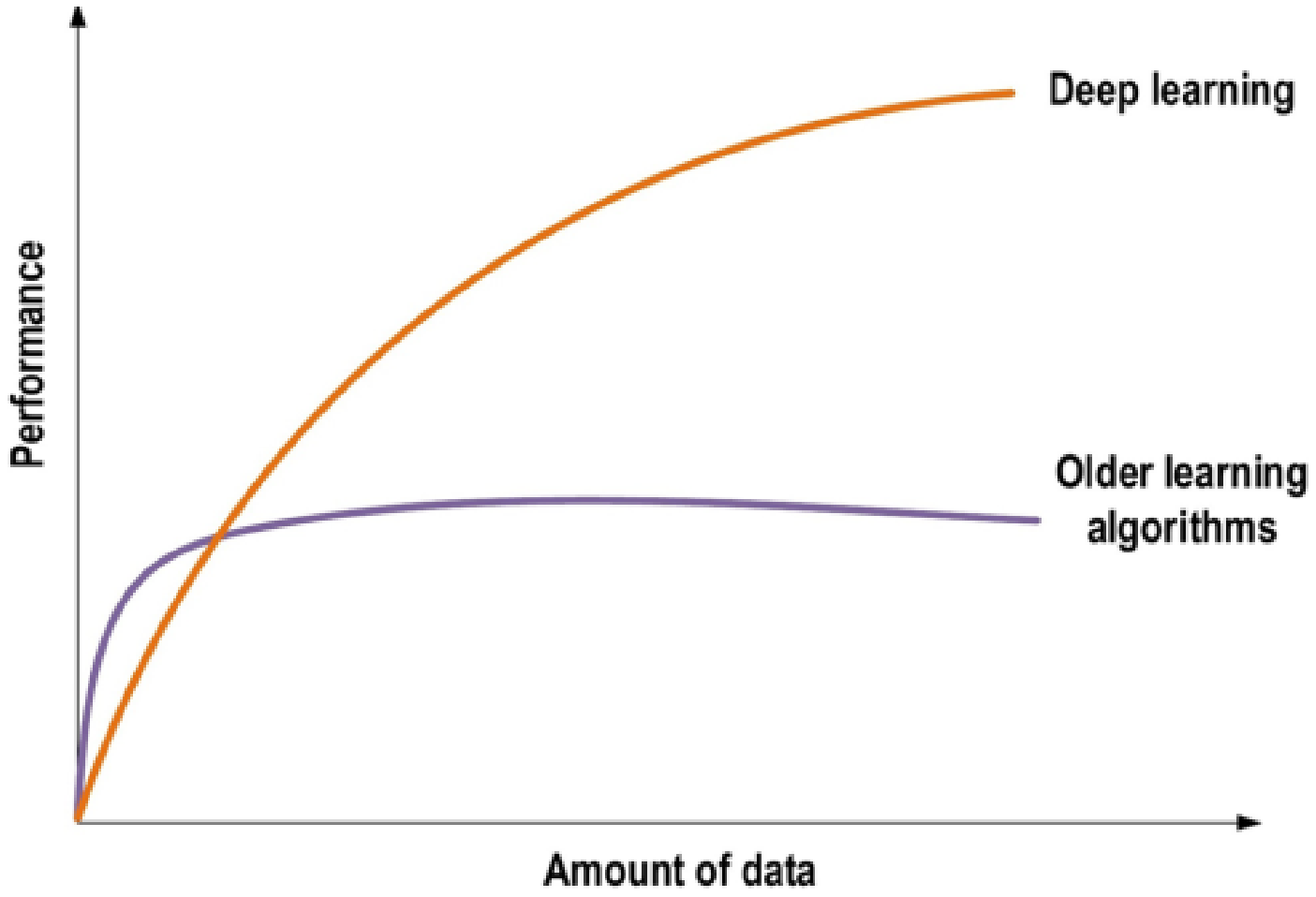
Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

(6+12+24+16)*2+1=117

Number of parameters

- LeNet > 60K
- AlexNet > 60.97M
- VGGNet16 > 138.36M
- GoogLeNet > 4M
- ResNet-18 > 11M
- DenseNet-121 > 8M

The performance of DL regarding the amount of data

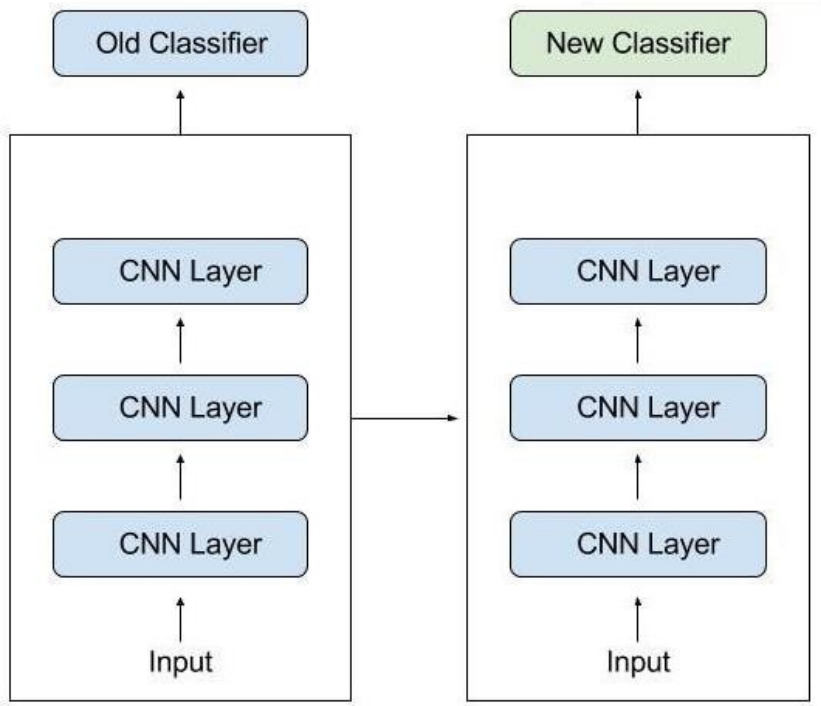


Transfer Learning

- The knowledge of an already trained machine learning model is applied to a different but related problem. We transfer the weights that a network has learned at "task A" to a new "task B.“
- The general idea is to use the knowledge a model has learned from a task with a lot of available labeled training data in a new task that doesn't have much data.
- Instead of starting the learning process from scratch, we start with patterns learned from solving a related task.
- Transfer learning is mostly used in computer vision and natural language processing tasks like sentiment analysis due to the huge amount of computational power required.

Transfer Learning in Computer Vision

- In computer vision, for example, neural networks usually try to detect edges in the earlier layers, shapes in the middle layer and some task-specific features in the later layers. In transfer learning, the early and middle layers are used and we only retrain the latter layers.



Transfer learning Advantages

- Transfer learning has several benefits, but the main advantages are saving training time, better performance of neural networks (in most cases), and not needing a lot of data.
- Another approach is to use deep learning to discover the best representation of your problem, which means finding the most important features. This approach is also known as representation learning, and can often result in a much better performance than can be obtained with hand-designed representation.

Reference

- GoogLeNet: C Szegedy et al. “Going Deeper with Convolutions”, *CVPR2015*.
- https://d2l.ai/chapter_convolutional-modern/googlenet.html
- RezNet: K. He; X. Zhang; S. Ren; J. Sun “Deep Residual Learning for Image Recognition”, CVPR 2016.
- S. Ioffe, C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift”, *arXiv preprint arXiv:1502.03167*.
- DenseNet: G Huang et. al, “Densely Connected Convolutional Networks”, CVPR 2017.