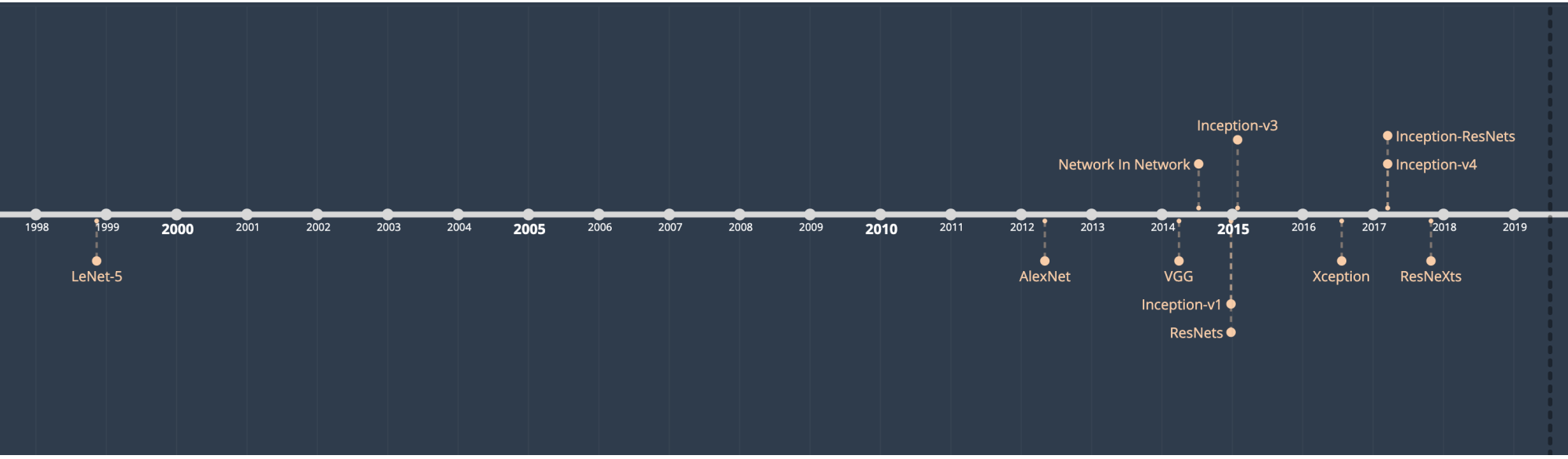
Popular CNN architectures

History


1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
LeNet-5
AlexNet
VGG
Inception-v1
ResNets
Network In Network
Inception-v3
Xception
Inception-v4
Inception-ResNets
ResNeXts

Notations

Layers

conv 3×3 — Convolutional operations, in **red**

avg-pool 2×2 — Pooling operations, in **grey**

concat — Merge operations eg. concat, add in **purple**

Dense layer, **blue**

Activation Functions

T — Tanh

R — ReLU

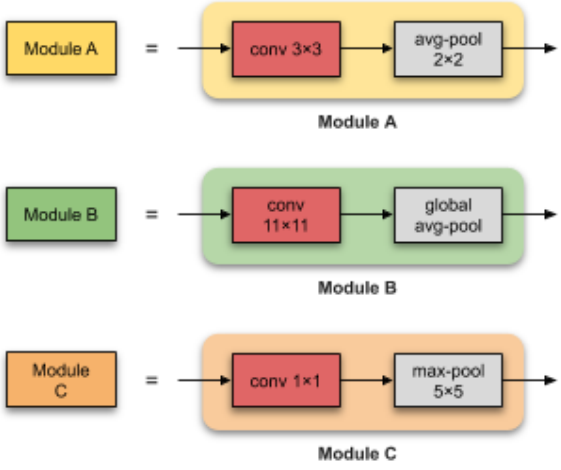
Other Functions

B — Batch normalisation

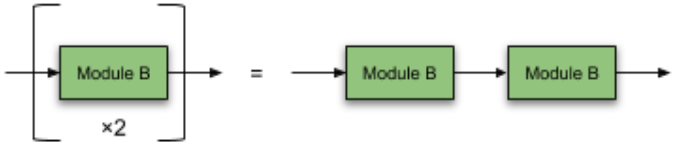
S — Softmax

Modules/Blocks

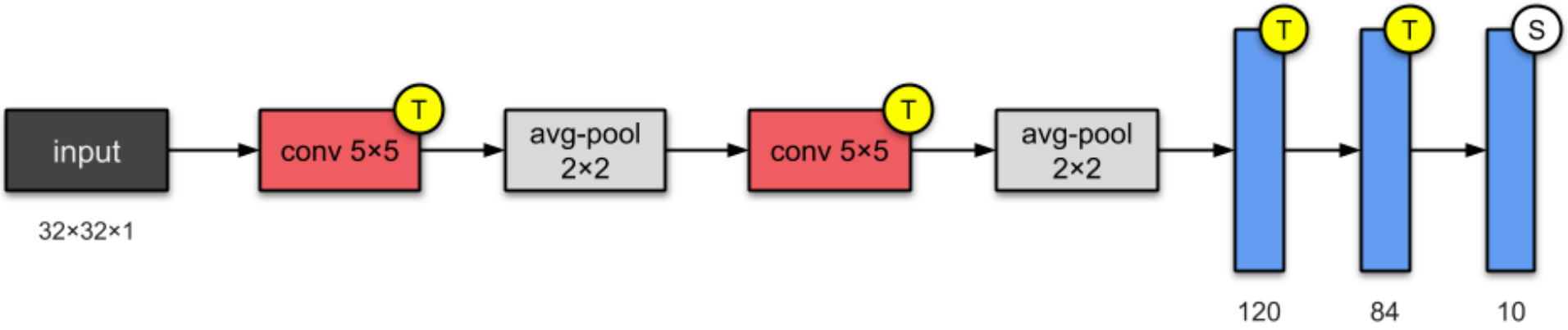
Modules (groups of convolutional, pooling and merge operations), in **yellow, green,** or **orange**.
The operations that make up these modules will also be shown.



Repeated layers or modules/blocks



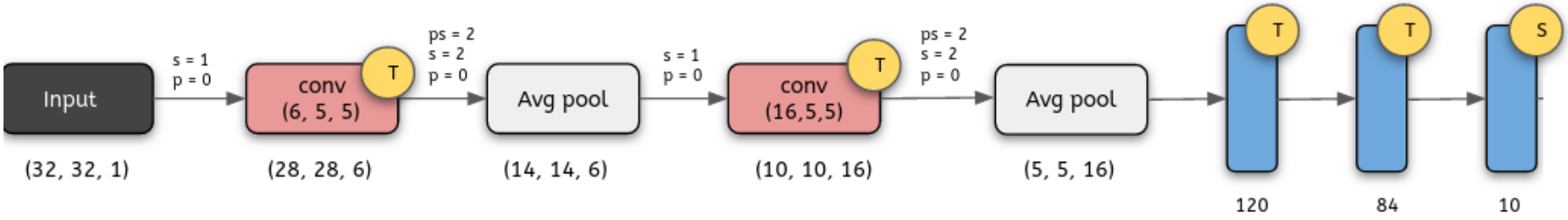
LeNet (1998)



- 2 convs + 3 FCs (hence “5”)
- 60,000 parameters.
- Stacking conv and pooling layers, ending the network with FCs.

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

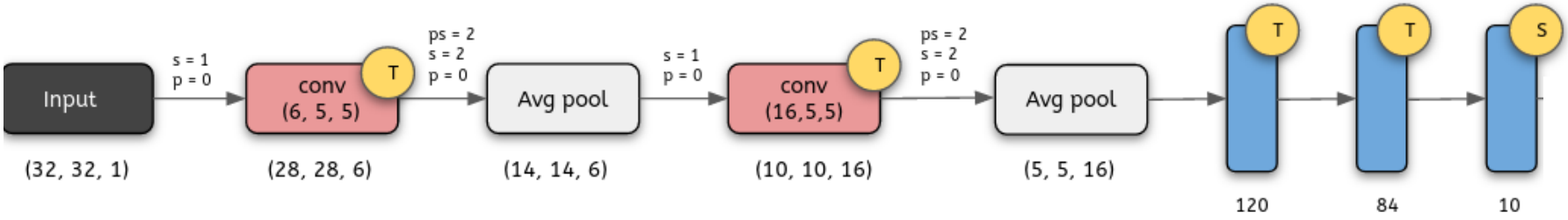
LeNet (1998)



- 2 convs + 3 FCs (hence “5”)
- 60,000 parameters.
- (New) Stacking conv and pooling layers, ending the network with FCs.

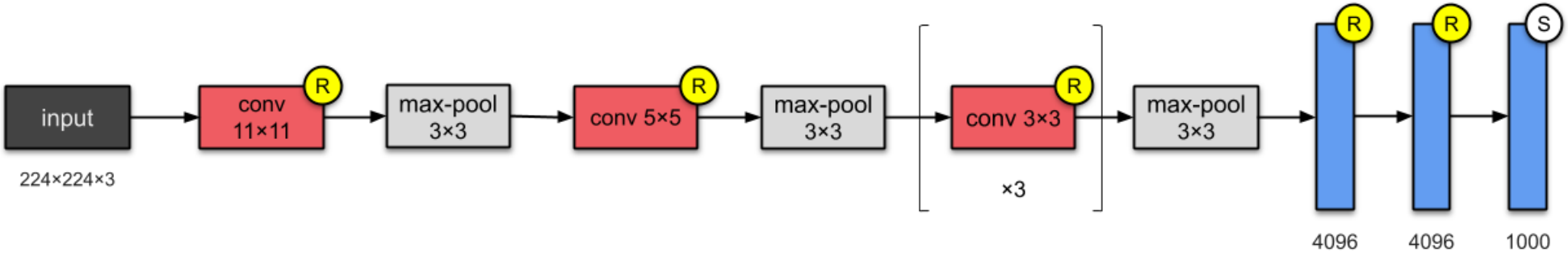
Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

LeNet (1998)



Question: How to improve LeNet?

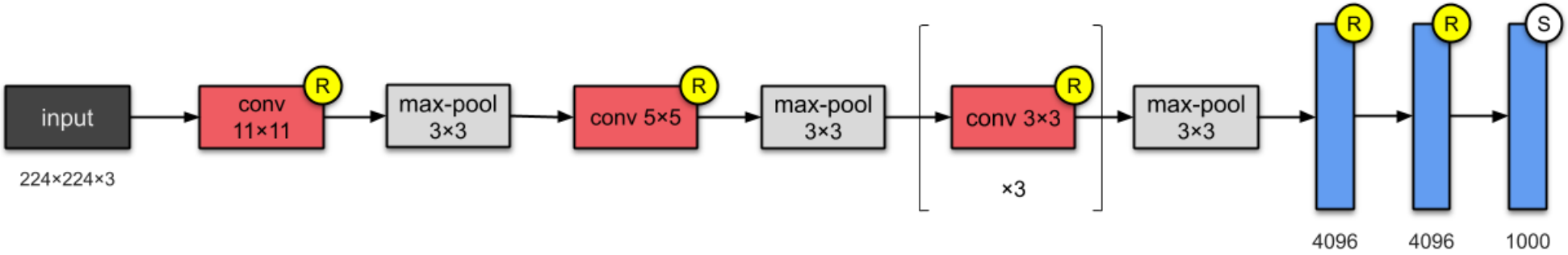
AlexNet (2012)



- 5 convs + 3 FCs.
- 60M parameters.
- First to implement Rectified Linear Units (ReLUs).
- Make use of Overlapping Pooling.

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

AlexNet (2012) – Overlapping Pooling

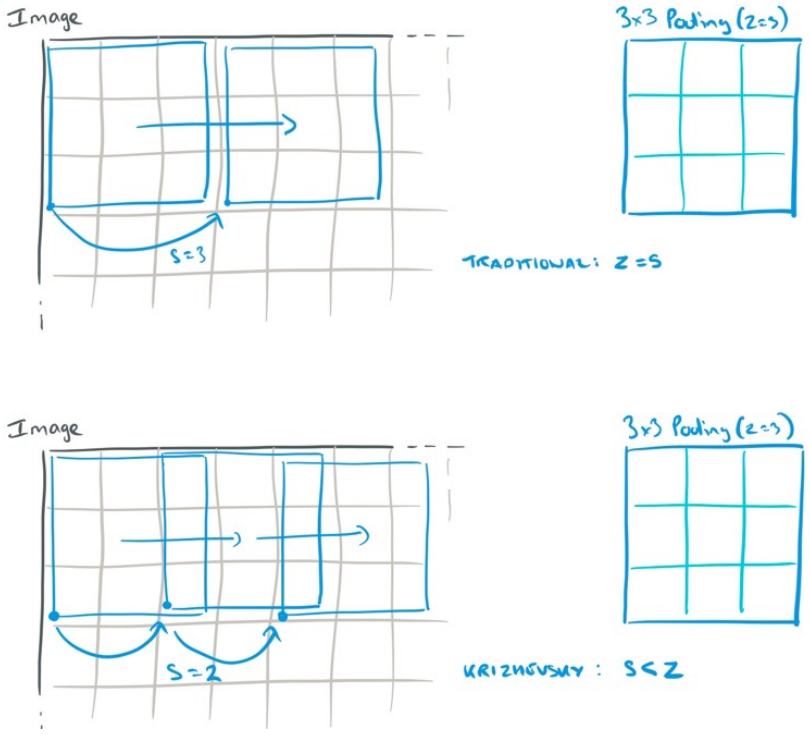


3.4 Overlapping Pooling

Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map. Traditionally, the neighborhoods summarized by adjacent pooling units do not overlap (e.g., [17, 11, 4]). To be more precise, a pooling layer can be thought of as consisting of a grid of pooling units spaced s pixels apart, each summarizing a neighborhood of size $z \times z$ centered at the location of the pooling unit. If we set $s = z$, we obtain traditional local pooling as commonly employed in CNNs. If we set $s < z$, we obtain overlapping pooling. This is what we use throughout our network, with $s = 2$ and $z = 3$. This scheme reduces the top-1 and top-5 error rates by 0.4% and 0.3%, respectively, as compared with the non-overlapping scheme $s = 2, z = 2$, which produces output of equivalent dimensions. We generally observe during training that models with overlapping pooling find it slightly more difficult to overfit.

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton: ImageNet Classification with Deep Convolutional Neural Networks, NIPS'12.

AlexNet (2012) – Overlapping Pooling



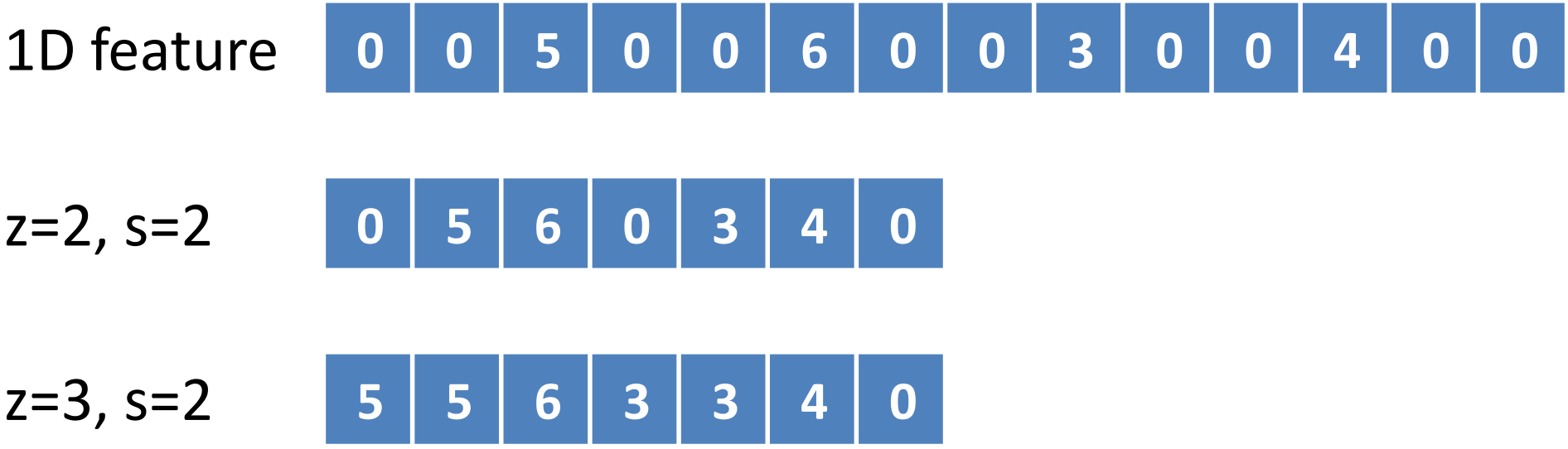
- Non-overlapping pooling: spatial information is quickly lost and the network "sees" only the dominant pixel values.
- Overlapping pooling: less loss of surrounding spatial information.

Question: to have overlapping pooling, should we reuse traditional (non-overlapping) pooling with a *larger pooling kernel*?

AlexNet (2012)

Question: to have overlapping pooling, should we reuse traditional (non-overlapping) pooling with a *larger pooling kernel*?

No, large pooling kernel leads to information destruction.

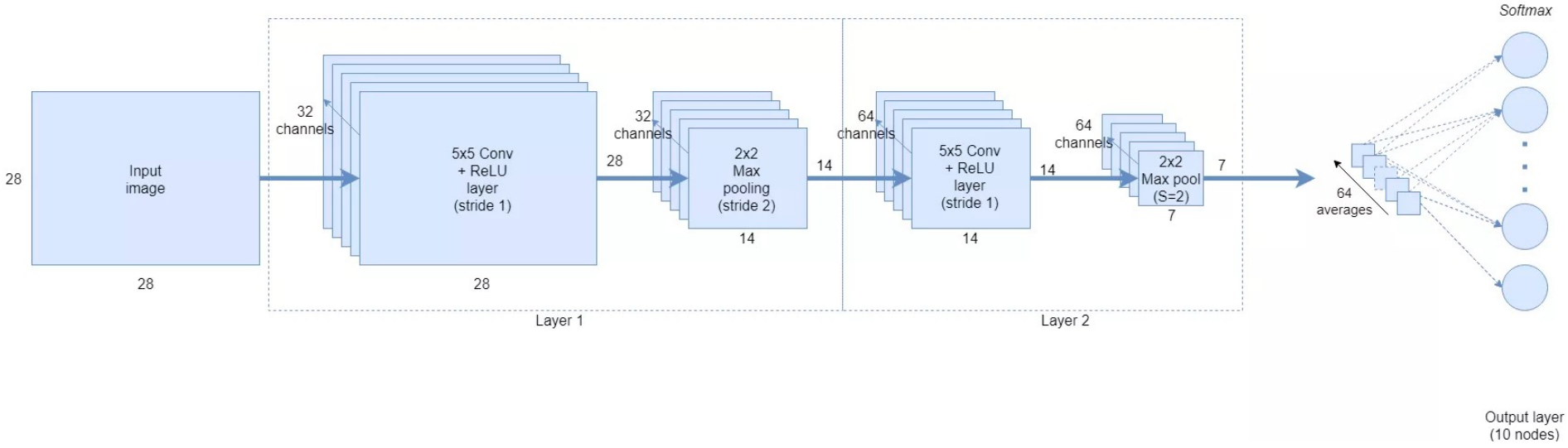


Grid-like pattern disappears!

Global Average Pooling (GAP)

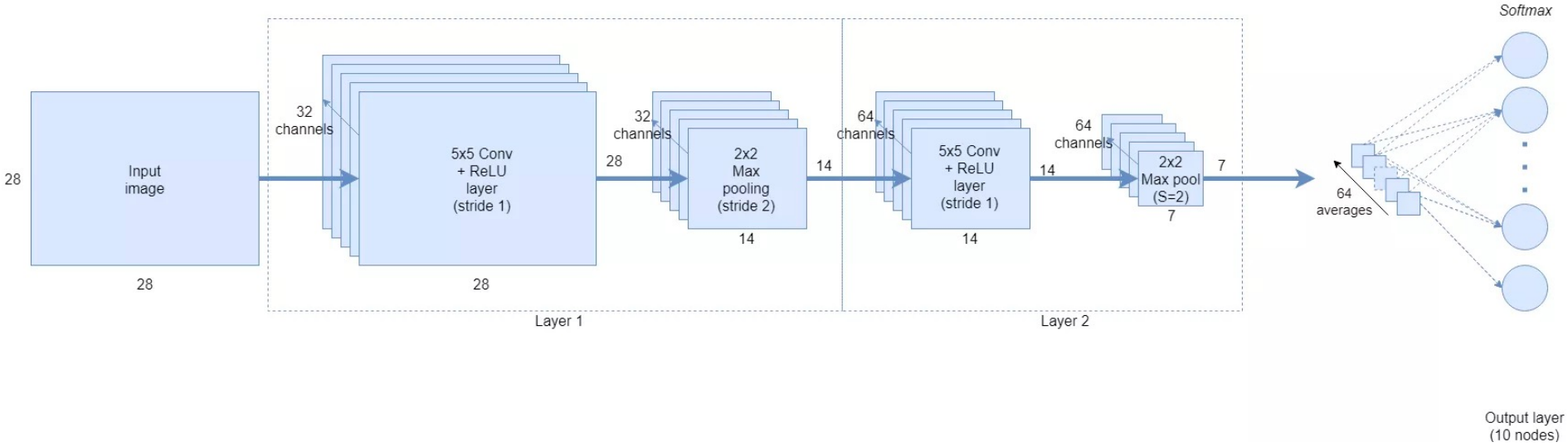
GAP is an operation that calculates the **average output of each feature map** in the previous layer:

- reduces the data significantly and prepares the model for the final classification layer.
- no trainable parameters.



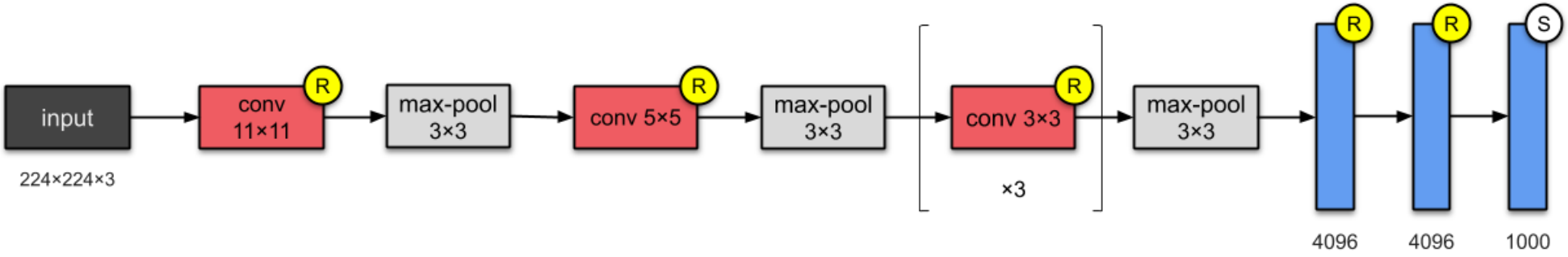
GAP in a CNN architecture

Why GAP?



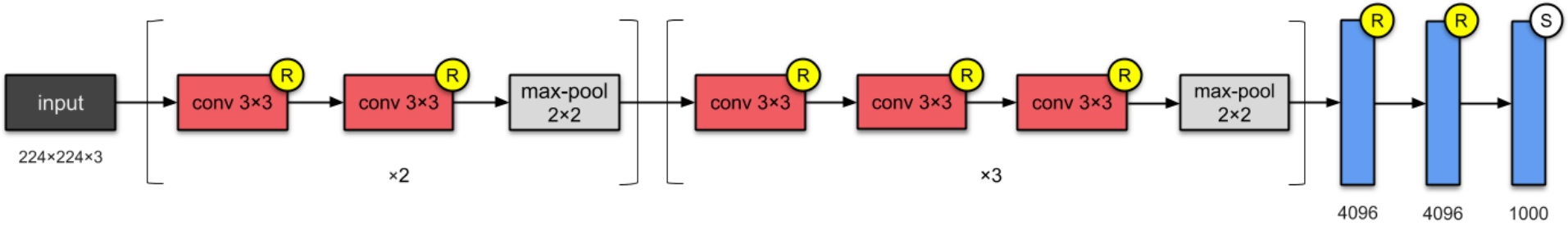
- Less number of trainable parameters: 64 (non-trainable) vs. 7x7x64xN (FC, trainable) → speed up factor.
- Replace FC → the feature maps are forced to be more closely related to the classification categories.
- Draw back: lack of complexity compared to FC.

AlexNet (2012)



Question: How to improve AlexNet?

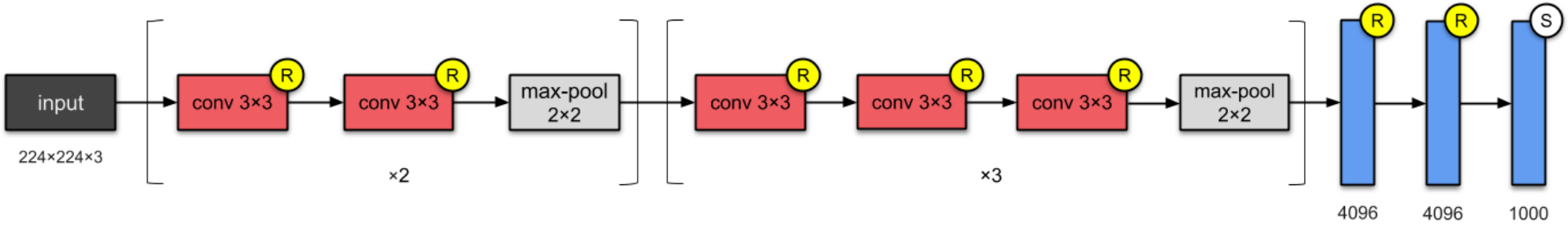
VGG16 (2014)



- 13 convs + 3 FCs.
- 138M parameters.
- ReLU activation functions as in AlexNet.
- Deeper network (twice as deep as AlexNet).

Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition.

VGG16 (2014)

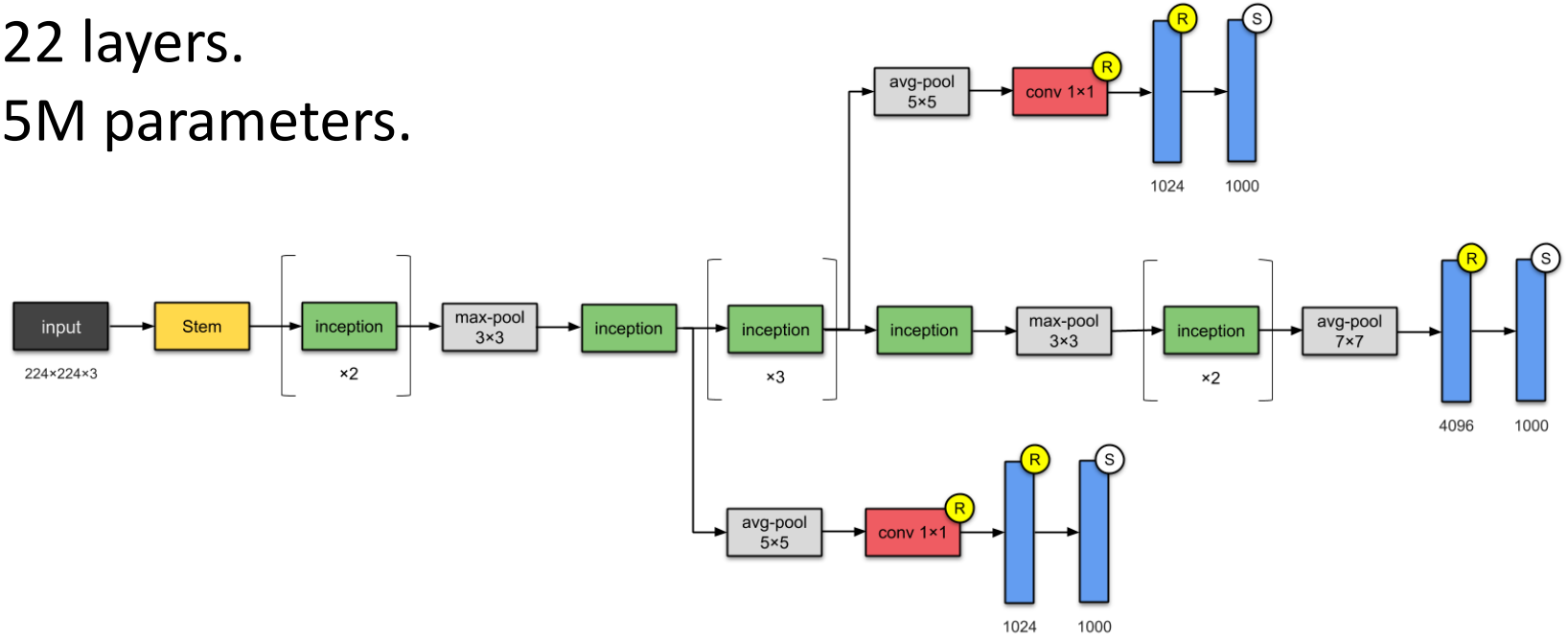


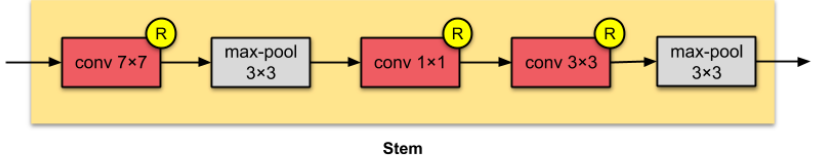
Question: How to improve VGG16?

Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition.

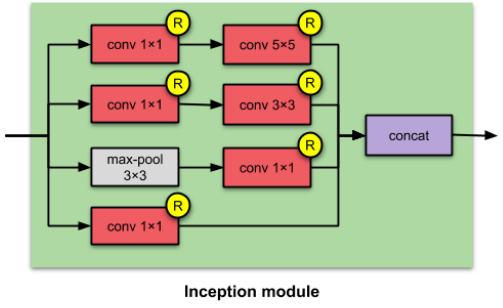
Inception-v1 (2014)

- 22 layers.
- 5M parameters.





Stem



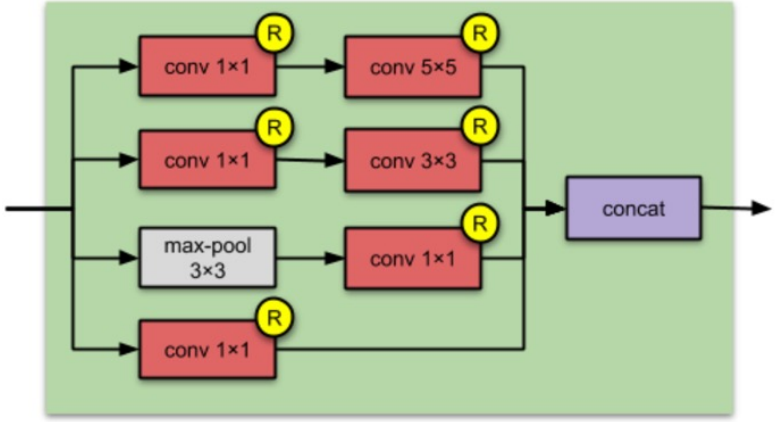
Inception module

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich: Going Deeper with Convolutions.

Inception-v1 (2014) - Inception Module

- Huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough.
- Naively stacking large convolution operations is computationally expensive.
- Very deep networks are prone to overfitting. It also hard to pass gradient updates through the entire network.

→ Creation of **Inception** module.

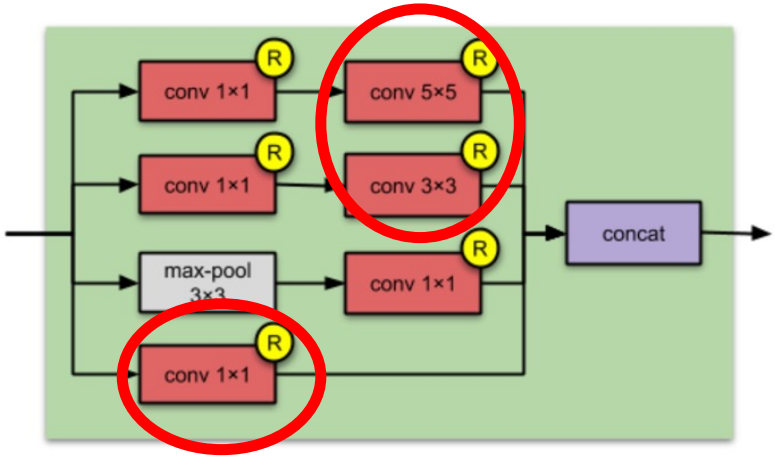


Inception module

Inception-v1 (2014) - Inception Module

- Huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough.
- Naively stacking large convolution operations is computationally expensive.
- Very deep networks are prone to overfitting. It also hard to pass gradient updates through the entire network.

→ Convolutional filters with **various kernel sizes** operating on the **same feature map**. The network is **wider** instead of deeper (which often leads to vanishing gradient).

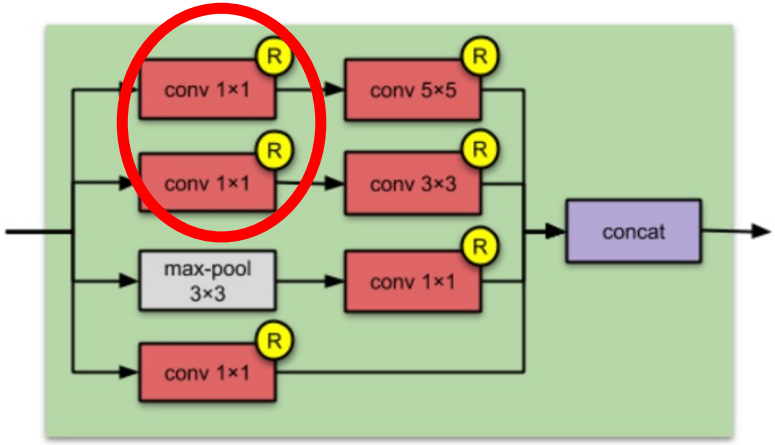


Inception module

Inception-v1 (2014) - Inception Module

- Huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough.
- Naively stacking large convolution operations is computationally expensive.
- Very deep networks are prone to overfitting. It also hard to pass gradient updates through the entire network.

Use **1x1** convolutional filter to reduce the computational cost (limiting the number of input channels before 3x3 and 5x5 conv).



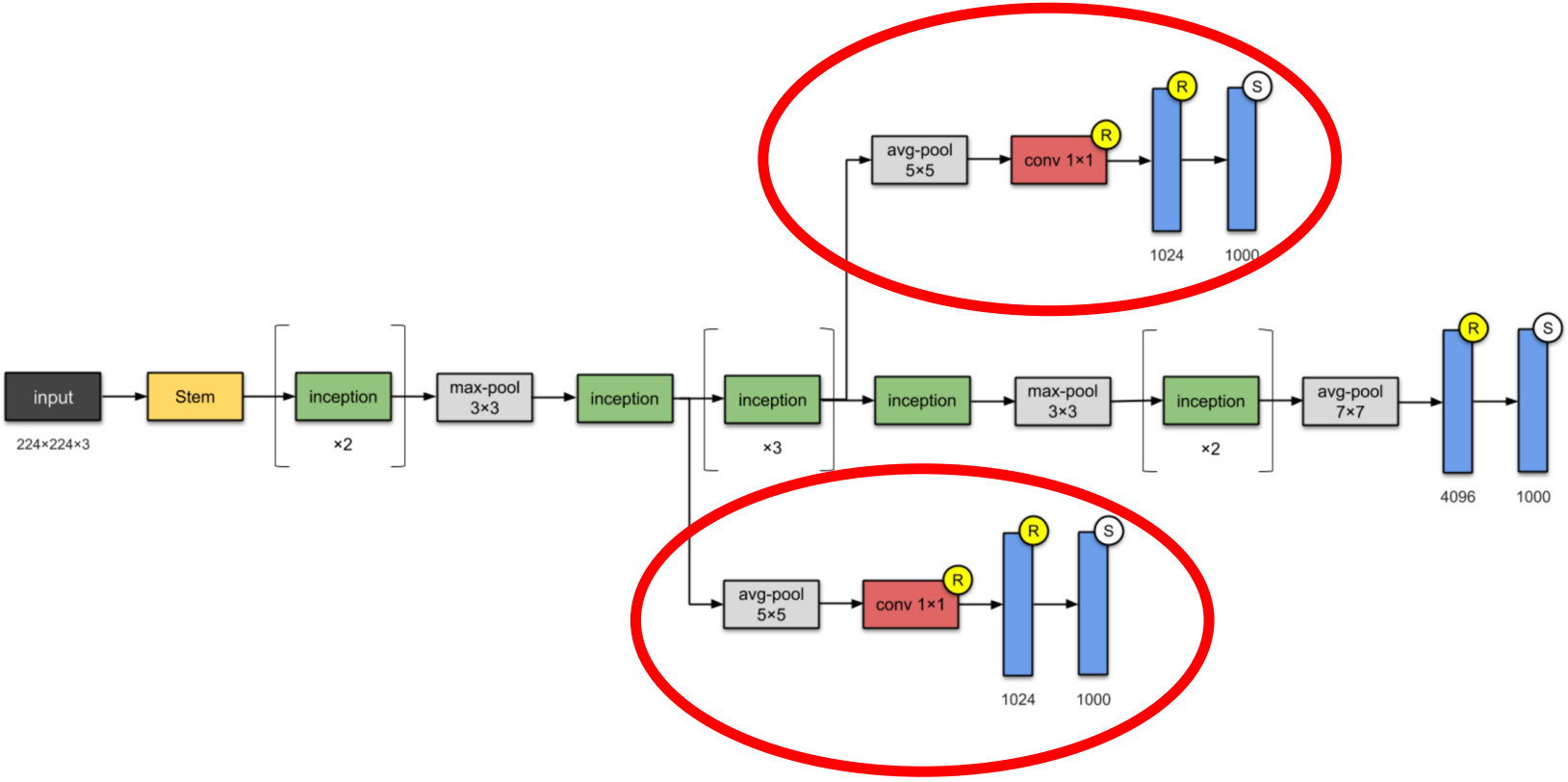
Inception module

Inception-v1 (2014) – Auxiliary classifiers

- Very deep networks are prone to overfitting. It also hard to pass gradient updates through the entire network.

→ Using **auxiliary classifiers** over the same labels to avoid vanishing gradient.

$$total\ loss = real\_loss + 0.3 * aux\_loss\_1 + 0.3 * aux\_loss\_2$$



Vanishing gradient

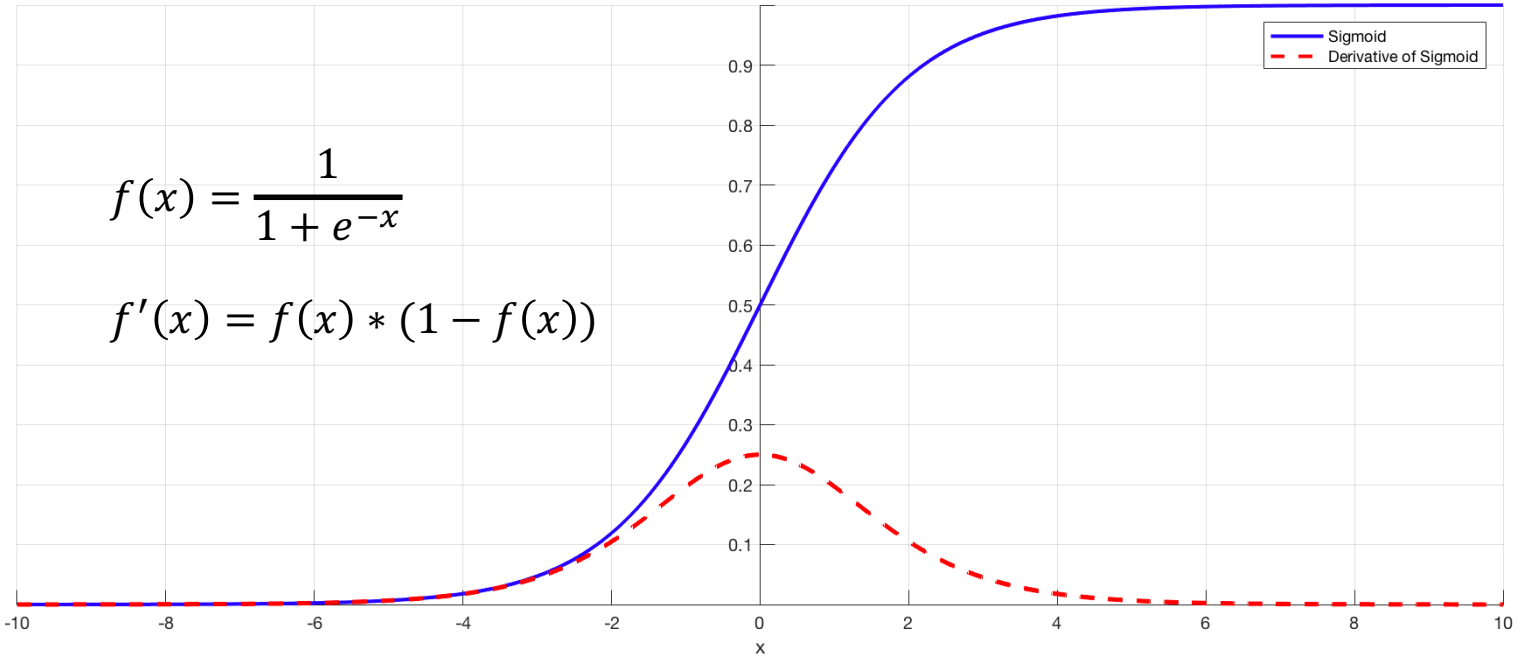
As more layers using certain activation functions (e.g., sigmoid) are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train, i.e., weights cannot be updated. This issue is called **vanishing gradient**.

Question: how to solve?

Sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}}$$

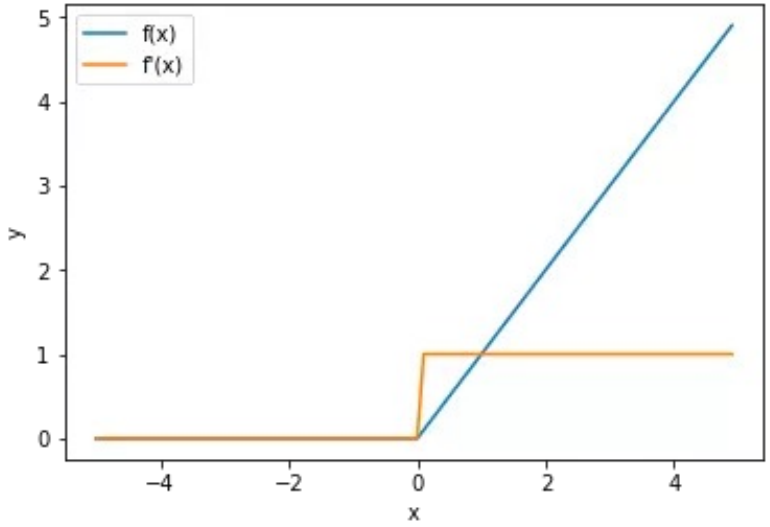
$$f'(x) = f(x) * (1 - f(x))$$



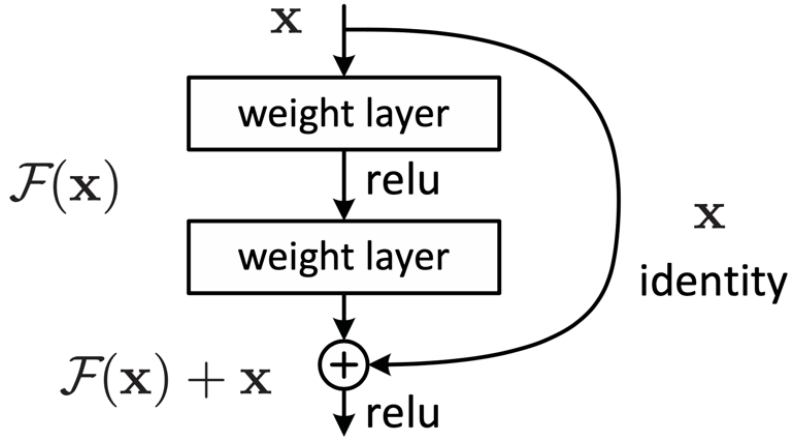
Vanishing gradient solutions

To solve the vanishing gradient issue:

- Use other activation functions, such as ReLU, which doesn't cause a small derivative.
- Get rid of the activation functions by **residual connection** → motivation of **ResNet**.
- Some network architectures are designed to prevent the vanishing gradient such as LSTM, GRU.



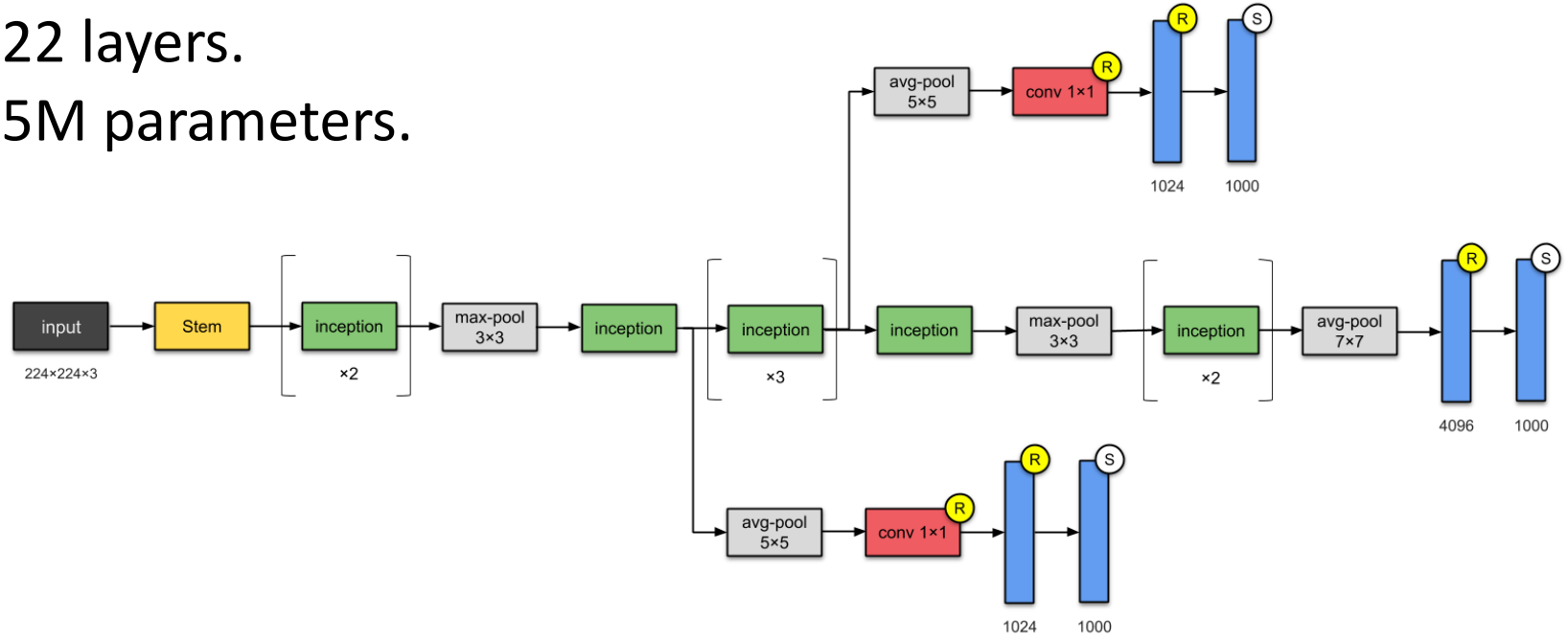
Derivative of ReLU



Residual block

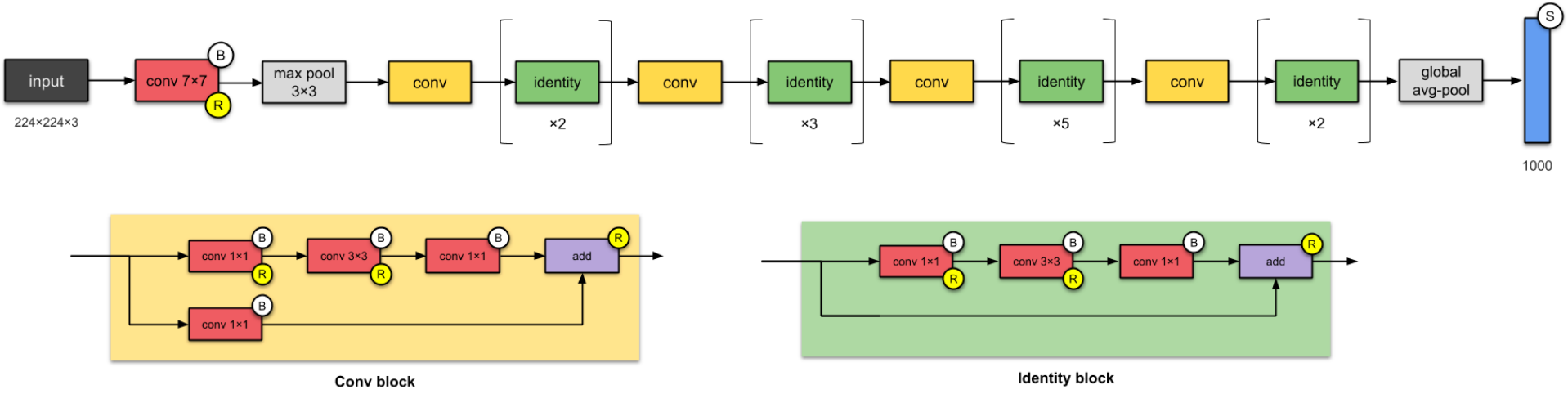
Inception-v1 (2014)

- 22 layers.
- 5M parameters.



Question: How to improve Inception?

ResNet (2015)

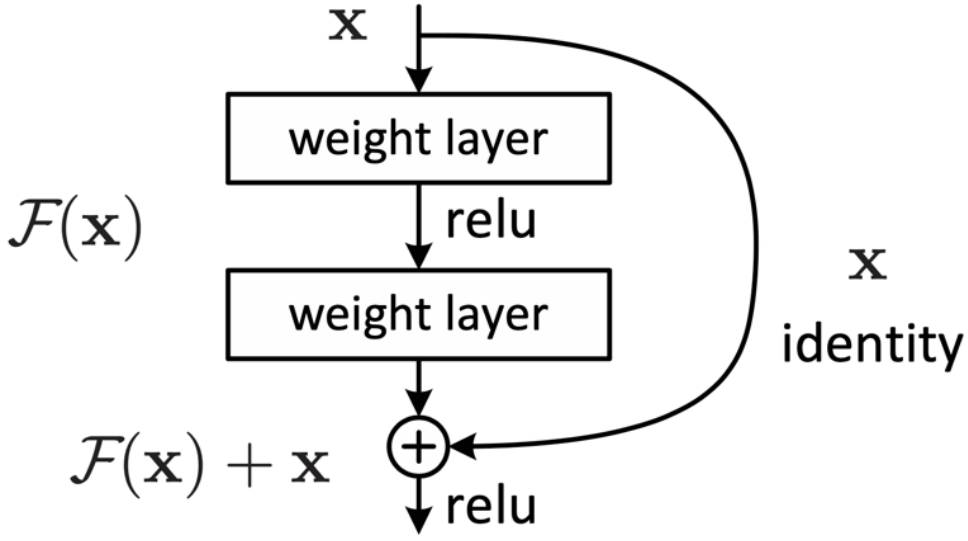


- A deeper plain network tends to perform bad because of the **vanishing** gradient and **exploding** gradient.
- Many variants of ResNet architectures, same concept but with a different number of layers: ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152, ResNet-164, ResNet-1202, etc.
- ResNet-50: 26M parameters.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Microsoft: Deep Residual Learning for Image Recognition, CVPR'16.

ResNet (2015)

Residual block

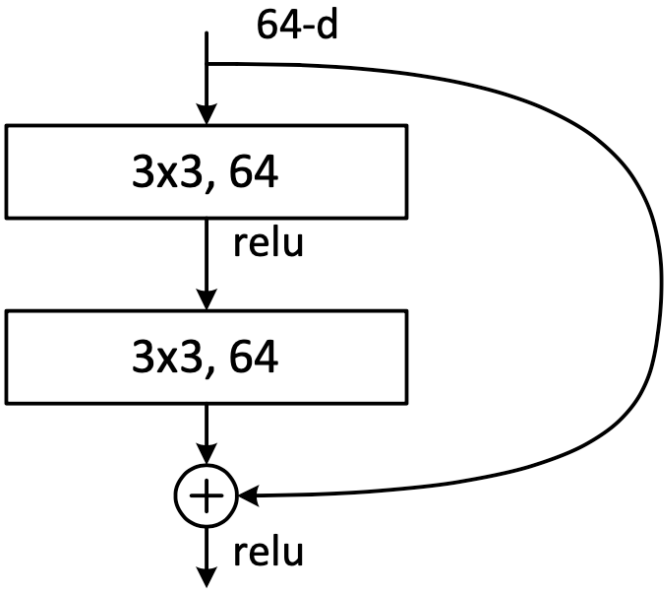


- Use skip/residual connection to avoid vanishing gradient → the network is able to go really deep.
- Basic residual block adds little additional computational load to the network.

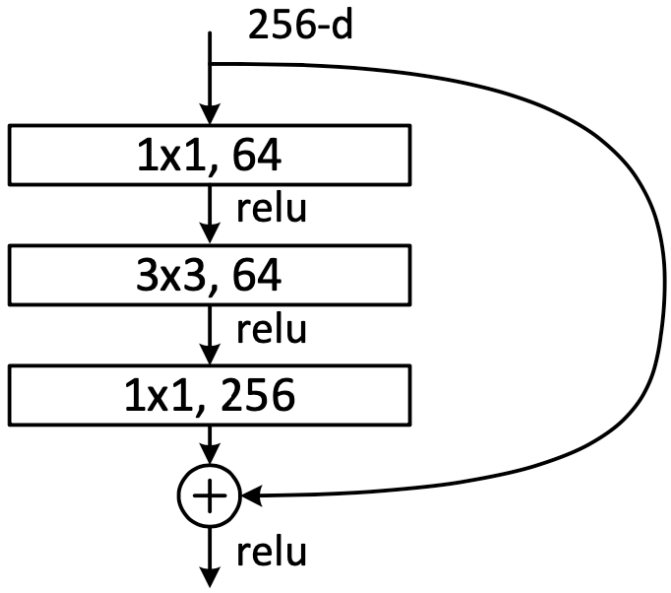
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Microsoft: Deep Residual Learning for Image Recognition, CVPR'16.

ResNet (2015)

Basic residual block
(56x56 feature map)



Bottleneck residual block
(56x56 feature map)



A deeper residual function:

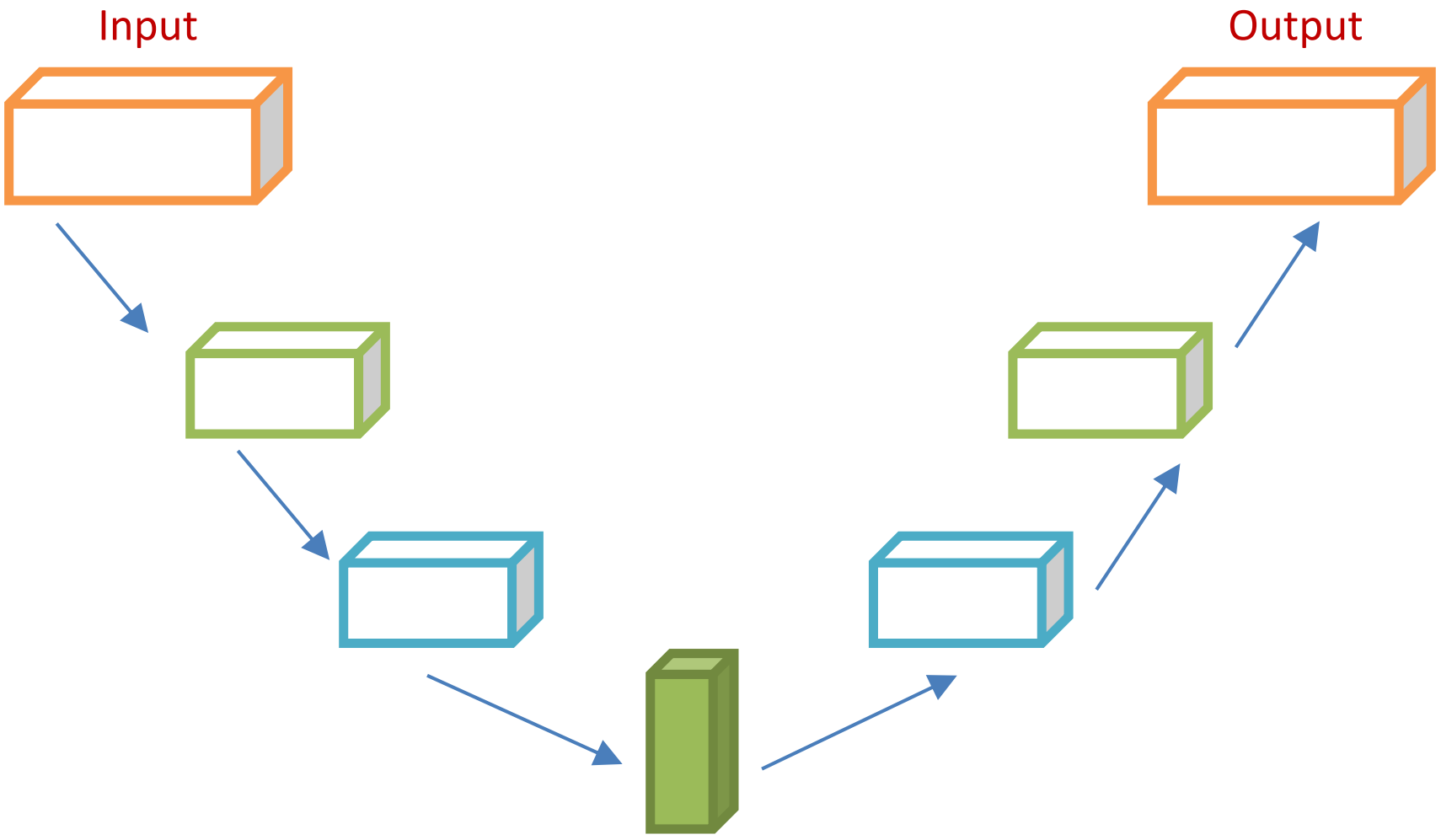
- Left (basic): a building block (on 56×56 feature maps) for ResNet- 34.
- Right (bottleneck): a building block for ResNet-50/101/152. The **1×1** layers are responsible for **reducing** and then **increasing** (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Microsoft: Deep Residual Learning for Image Recognition, CVPR'16.

U-Net (2015)

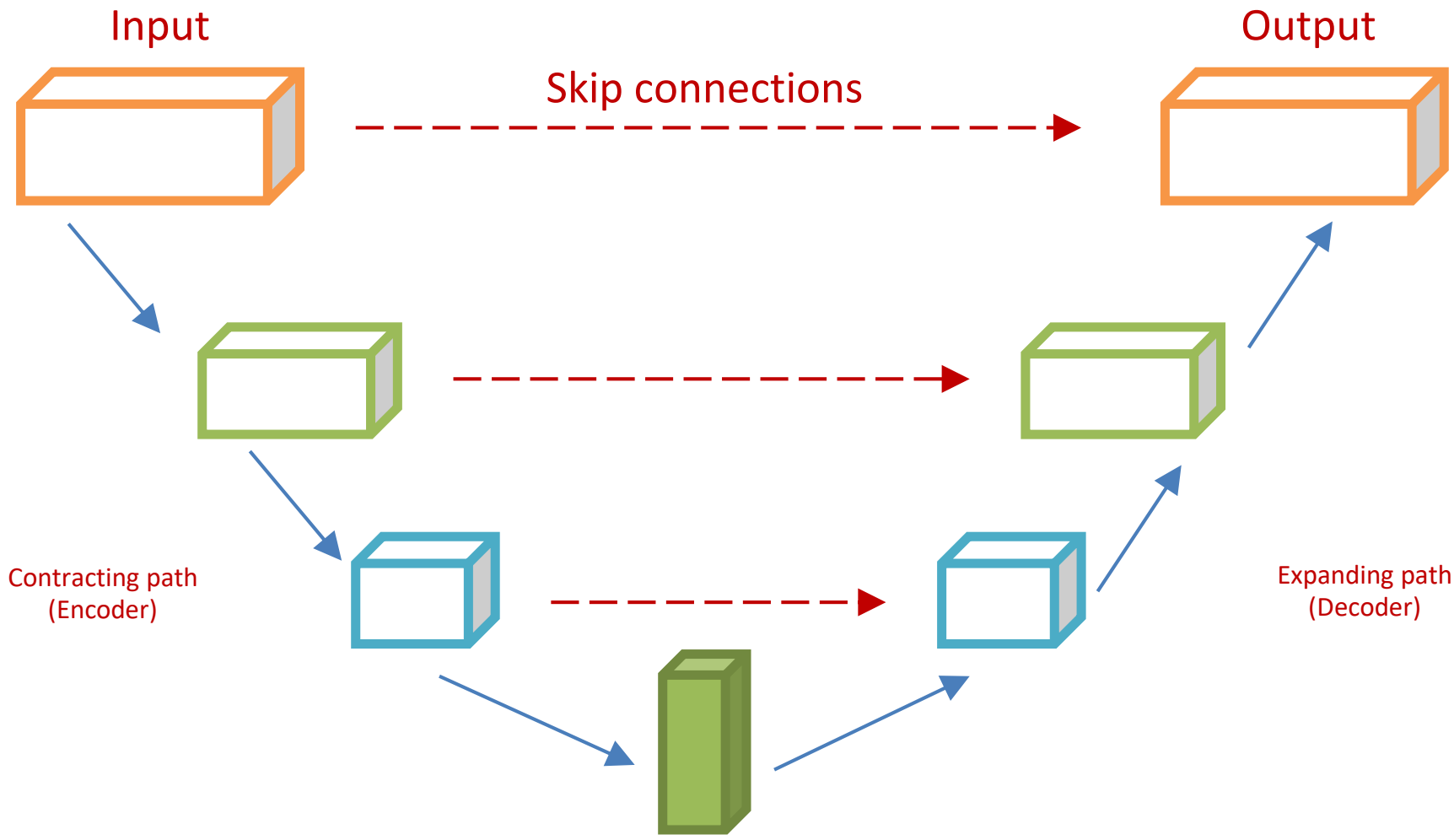
Question: Skip/residual connections are exploited avoid vanishing gradient to build really deep neural network. What is the **other benefit** of using the skip/residual connection?

U-Net (2015)



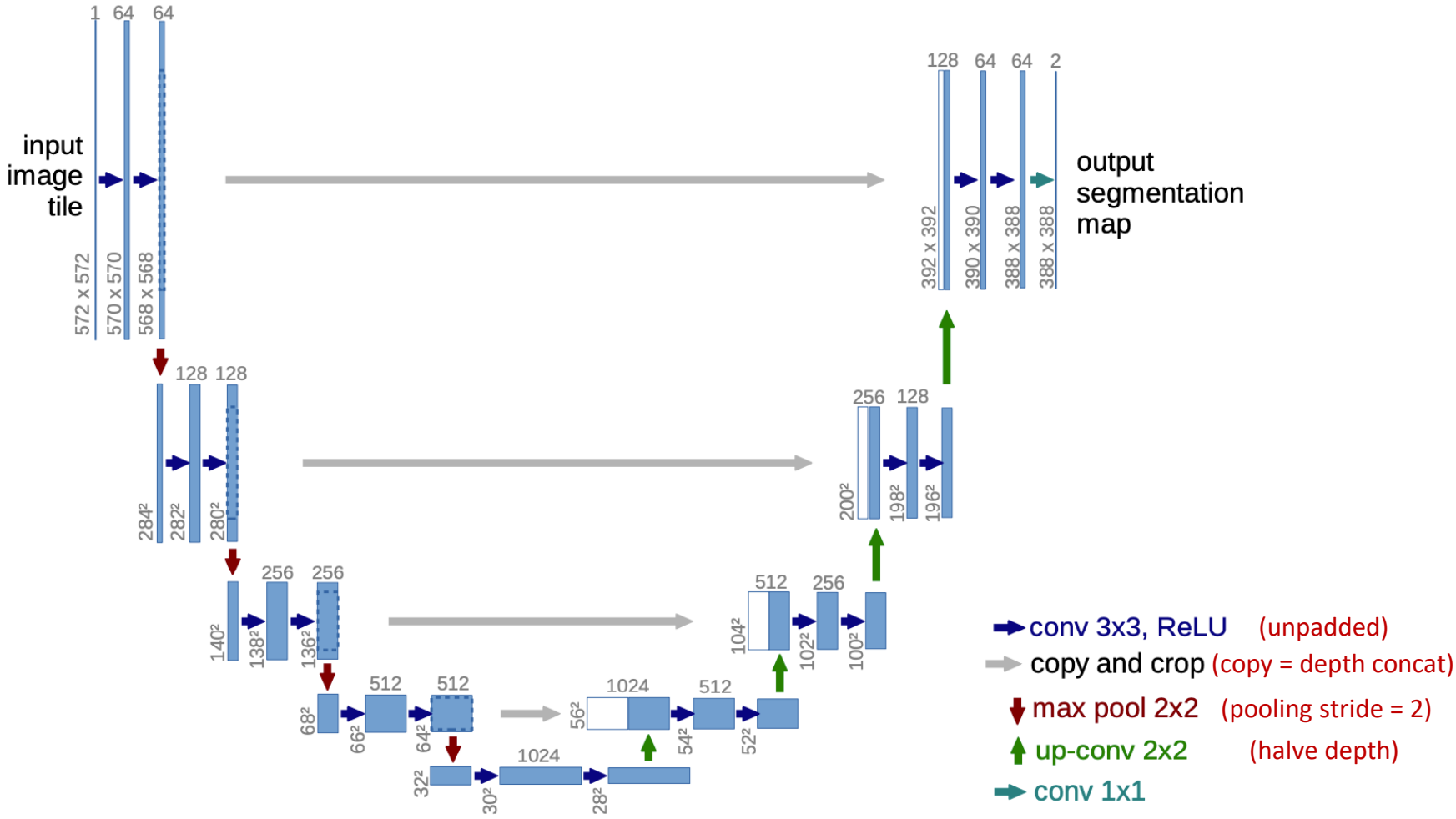
Question: what issue is with this network?

U-Net (2015)



Without skip connections, there may be a **loss of information** from layer to layer.
→ Skip connections enable **feature reusability** and stabilize training and convergence.

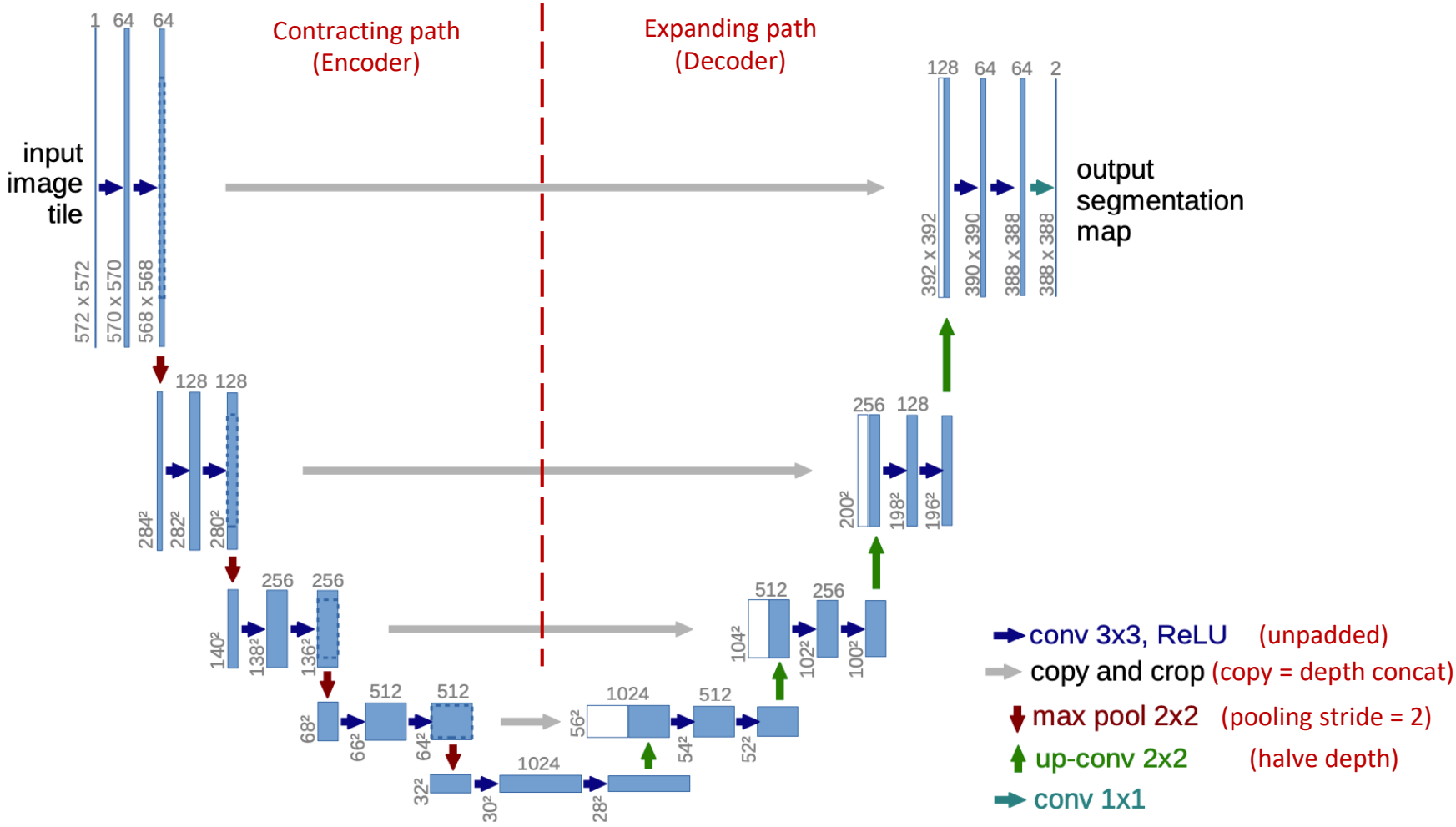
U-Net (2015)



U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", MICCAI'15.

U-Net (2015)



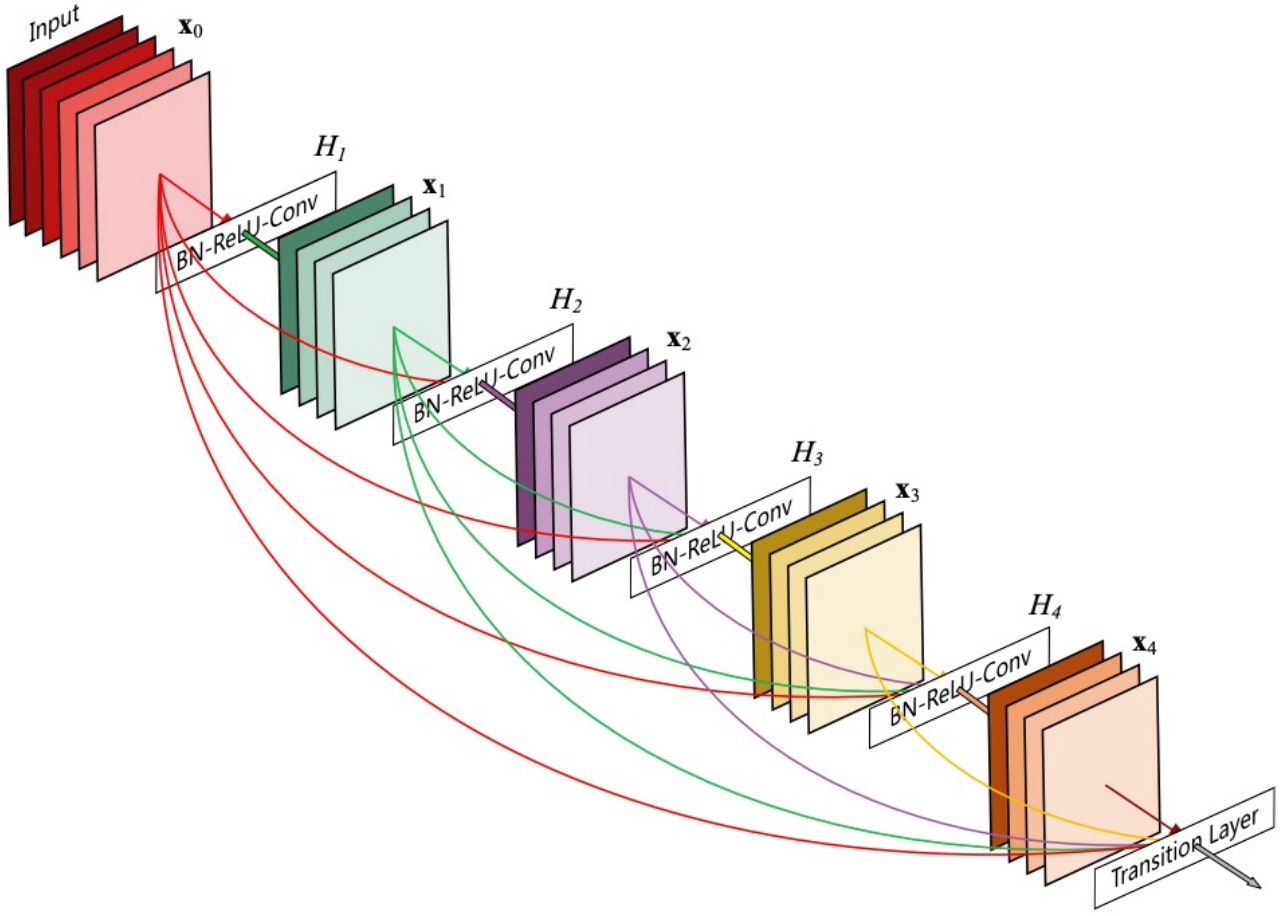
- **Long skip connections** are used to **pass features** from the encoder path to the decoder path in order to recover spatial information lost during down-sampling.
- **Short skip connections** appear to **stabilize gradient updates** in deep architectures.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", MICCAI'15.

DenseNet (2017)

Question: How can we choose the starting and end point of a skip connection?

DenseNet (2017)



Each layer takes all preceding feature maps as input.

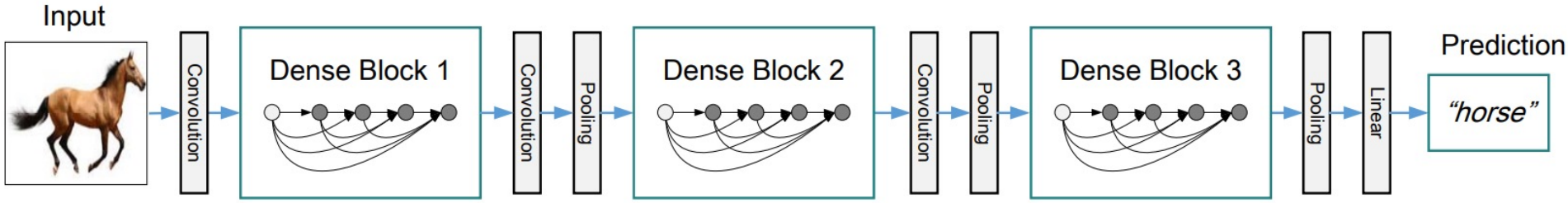
Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, "Densely Connected Convolutional Networks", CVPR'17.

DenseNet (2017)

Question: How many connections are there in a N-layer **dense** network?

DenseNet (2017)

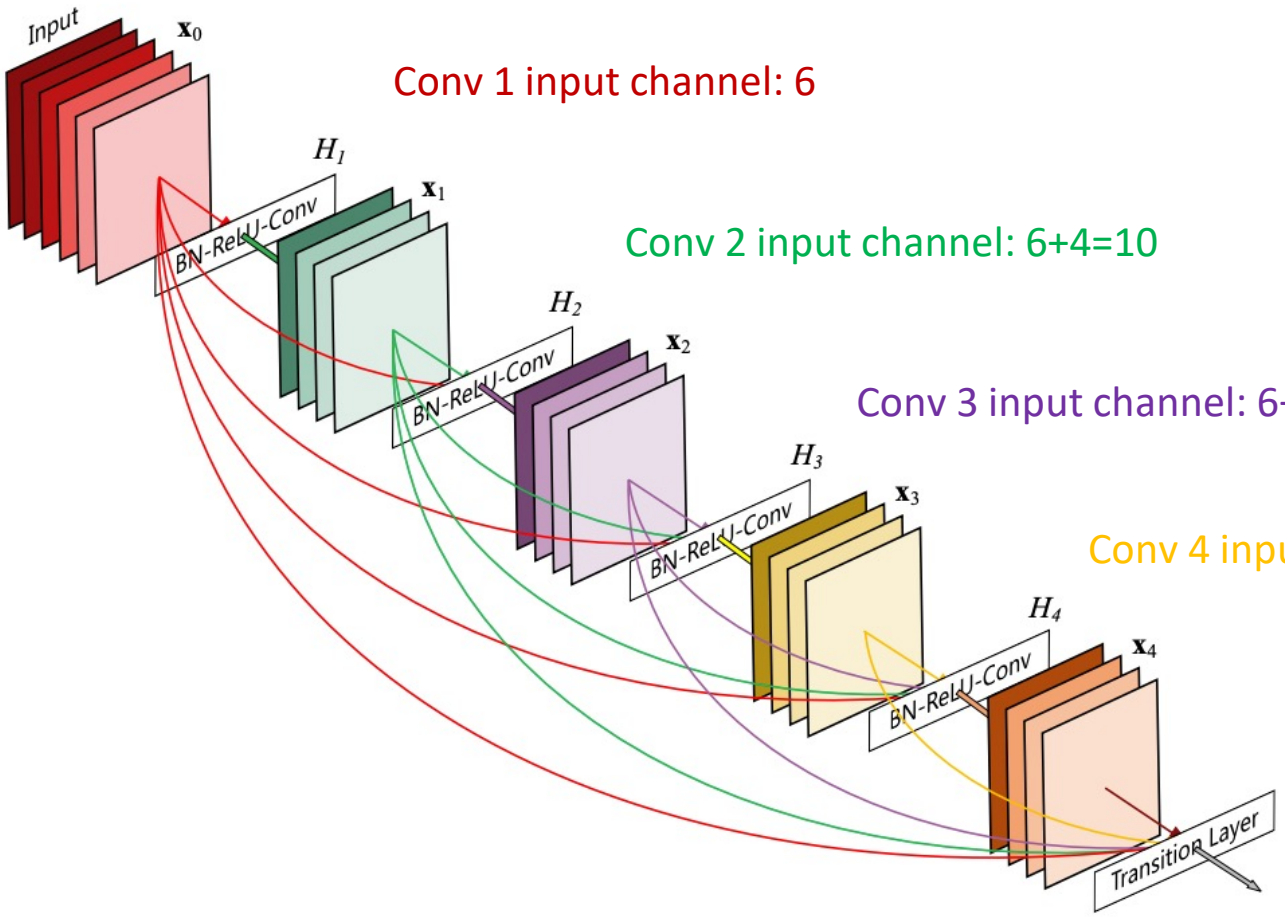
DenseNet is a type of convolutional neural network that utilises dense connections between layers, through Dense Blocks, where we connect all layers (with matching feature-map sizes) directly with each other.



A 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.

Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, "Densely Connected Convolutional Networks", CVPR'17.

DenseNet (2017)



Conv 1 input channel: 6

Conv 2 input channel: 6+4=10

Conv 3 input channel: 6+4+4=14

Conv 4 input channel: 6+4+4+4=18

Transition layer input channel: 6+4+4+4+4=22

A 5-layer dense block with a **growth rate** of k = 4, with channel wise concatenation. (Note: Skip connection in ResNet is element-wise addition).

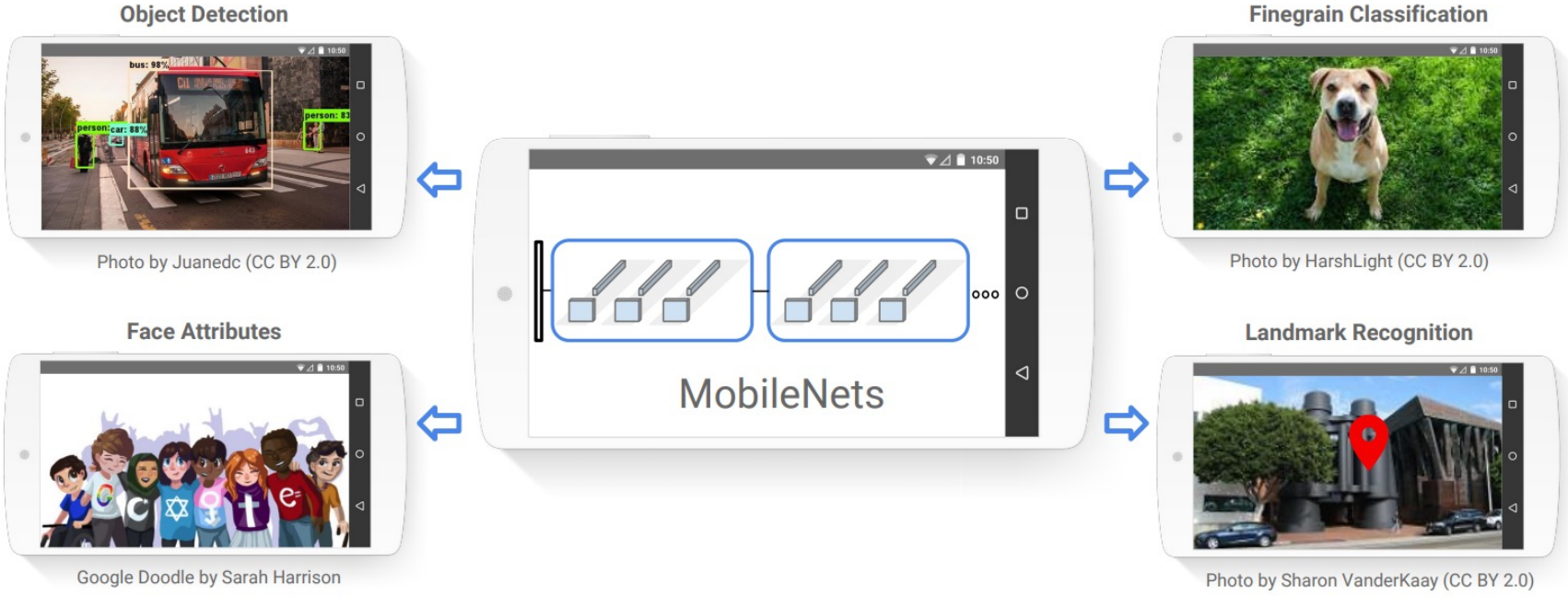
MobileNet-v1 (2015)

Question: How to build a compact and efficient neural network that can run on an embedded device?

MobileNet-v1 (2015)

Limitation of embedded devices:

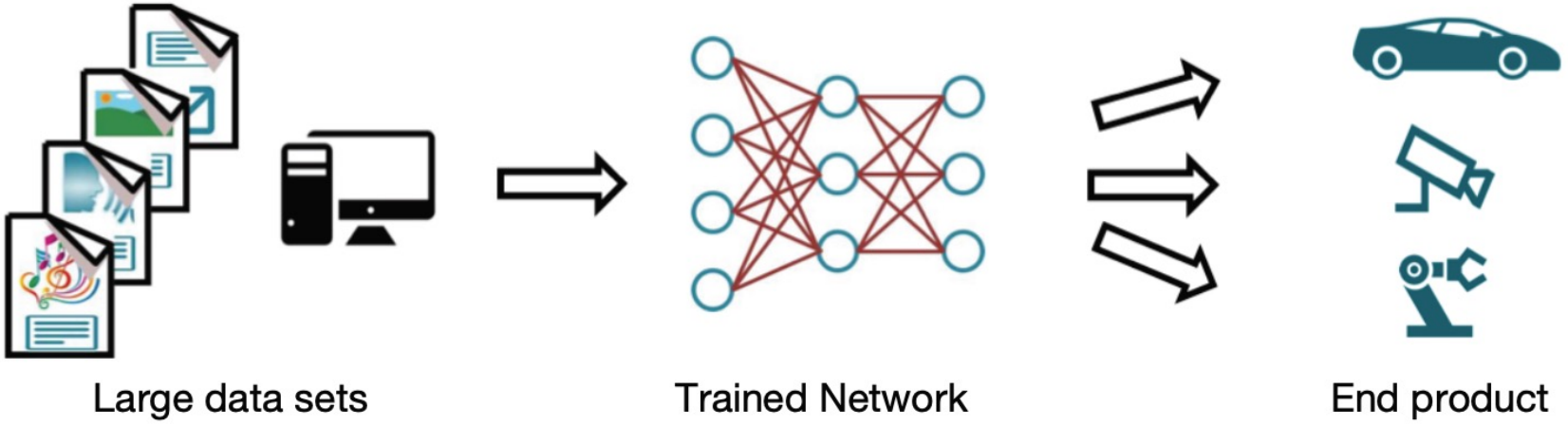
- Low computational resource and capacity.
- Low battery.



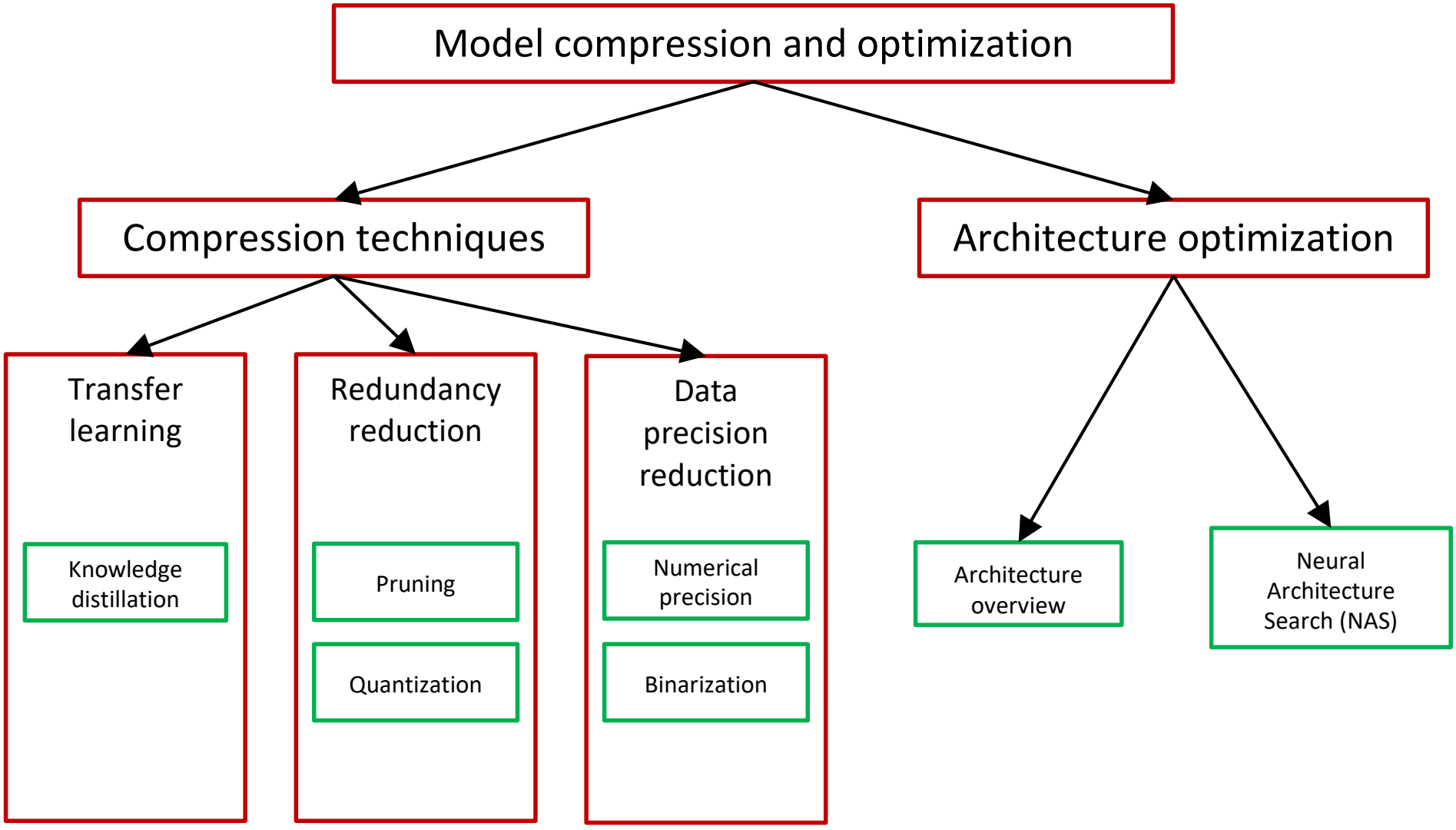
MobileNet-v1 (2015)

MobileNet is **light-weight** deep neural networks that can have **low latency** for mobile and embedded devices. It is:

- designed for mobile and embedded applications.
- based on an architecture that uses depth-wise separable convolutions.

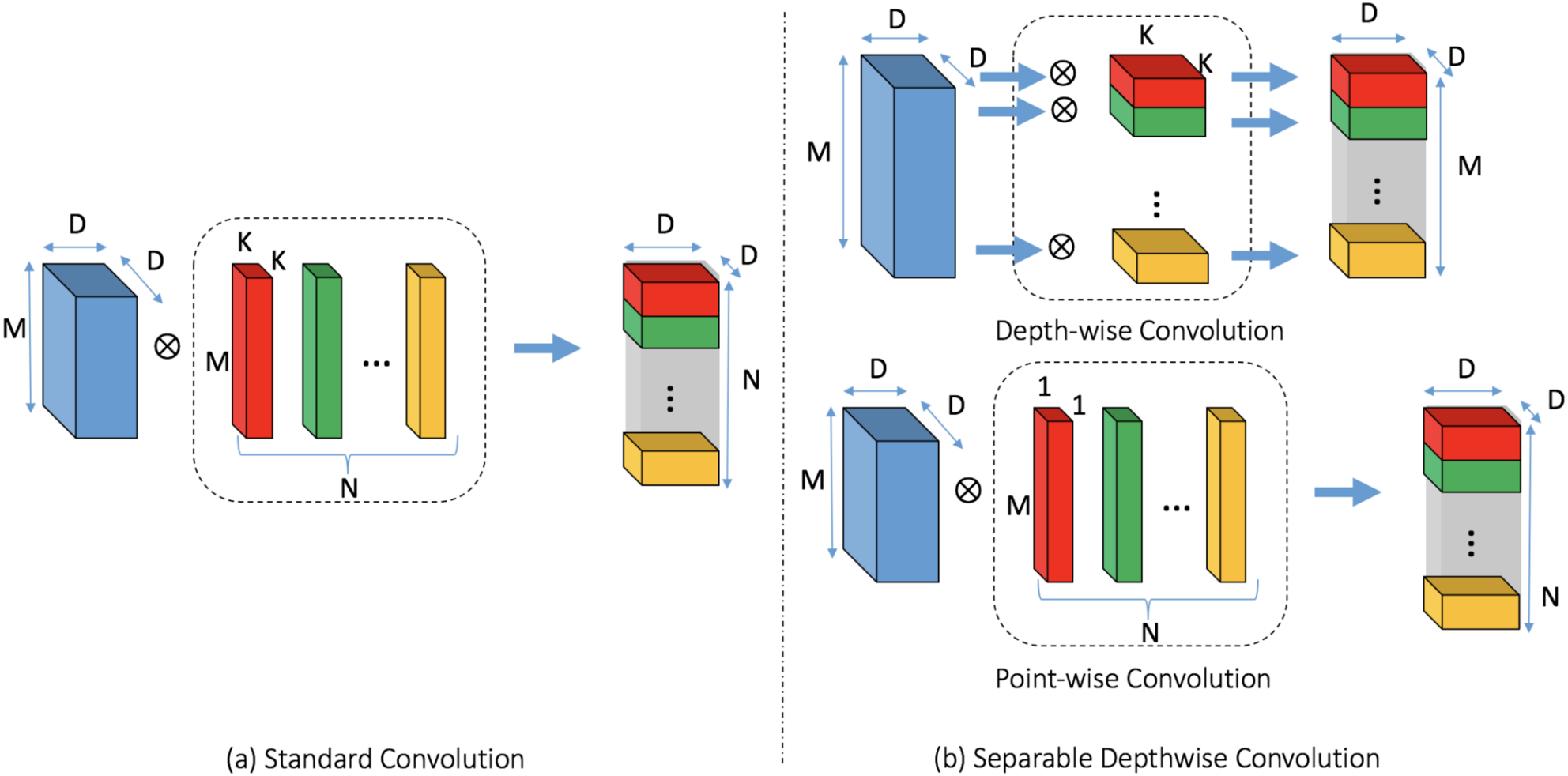


MobileNet-v1 (2015)



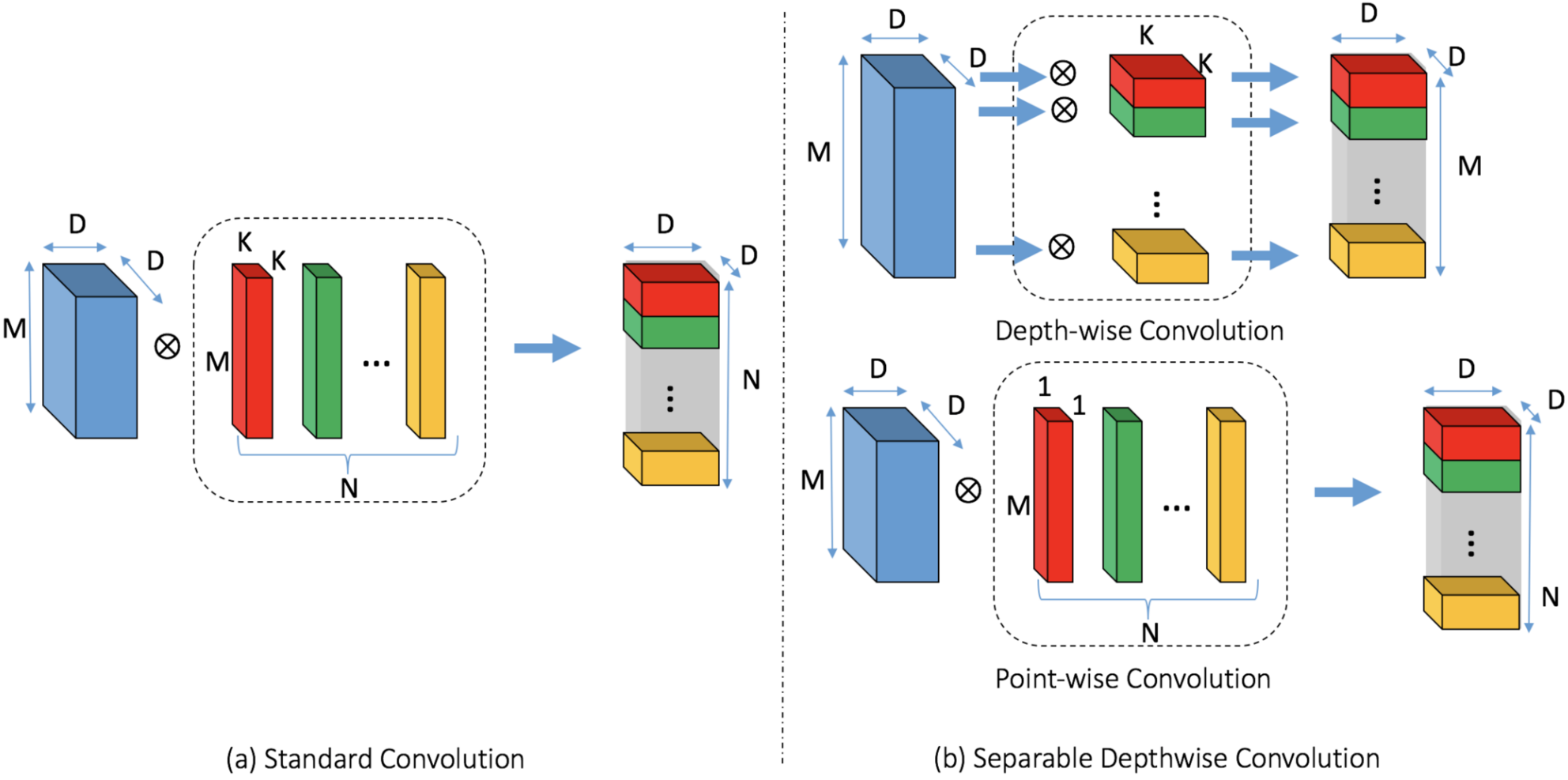
Deep neural network compression and optimization

MobileNet-v1 (2015)



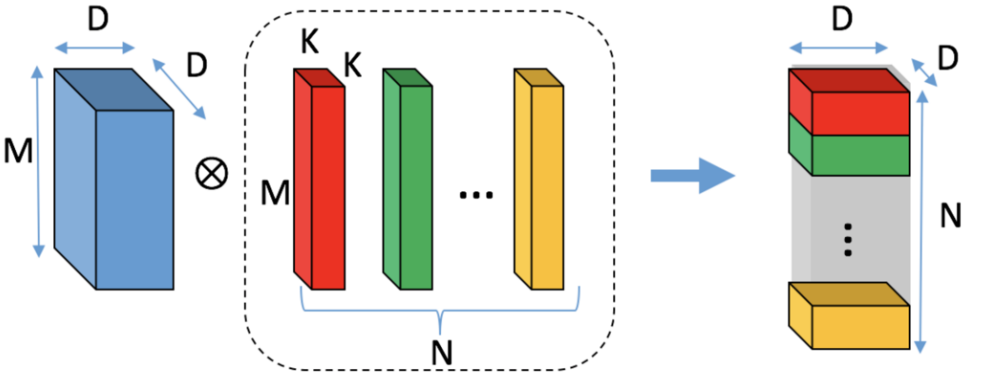
Idea: the standard convolutional filters are replaced by two layers: (1) **depthwise convolution** and (2) **pointwise convolution** to build **a depthwise separable filter**.

MobileNet-v1 (2015)



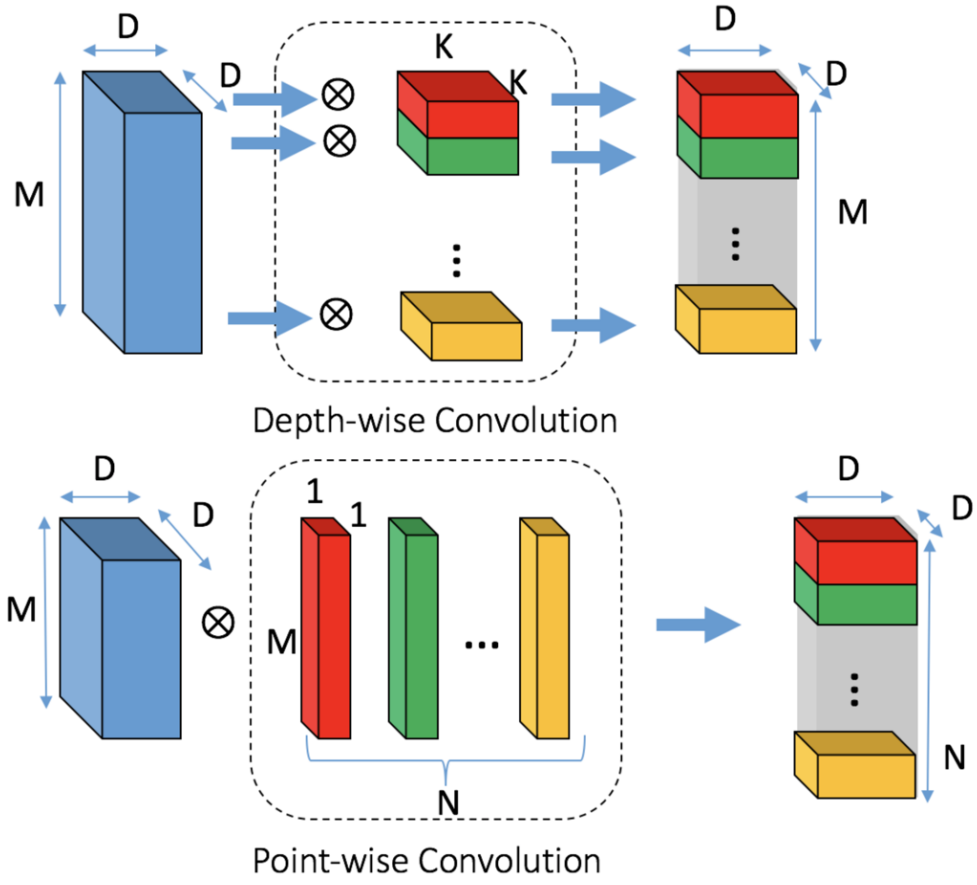
Question: How many learnable parameters in each of the convolution types?

MobileNet-v1 (2015)



(a) Standard Convolution

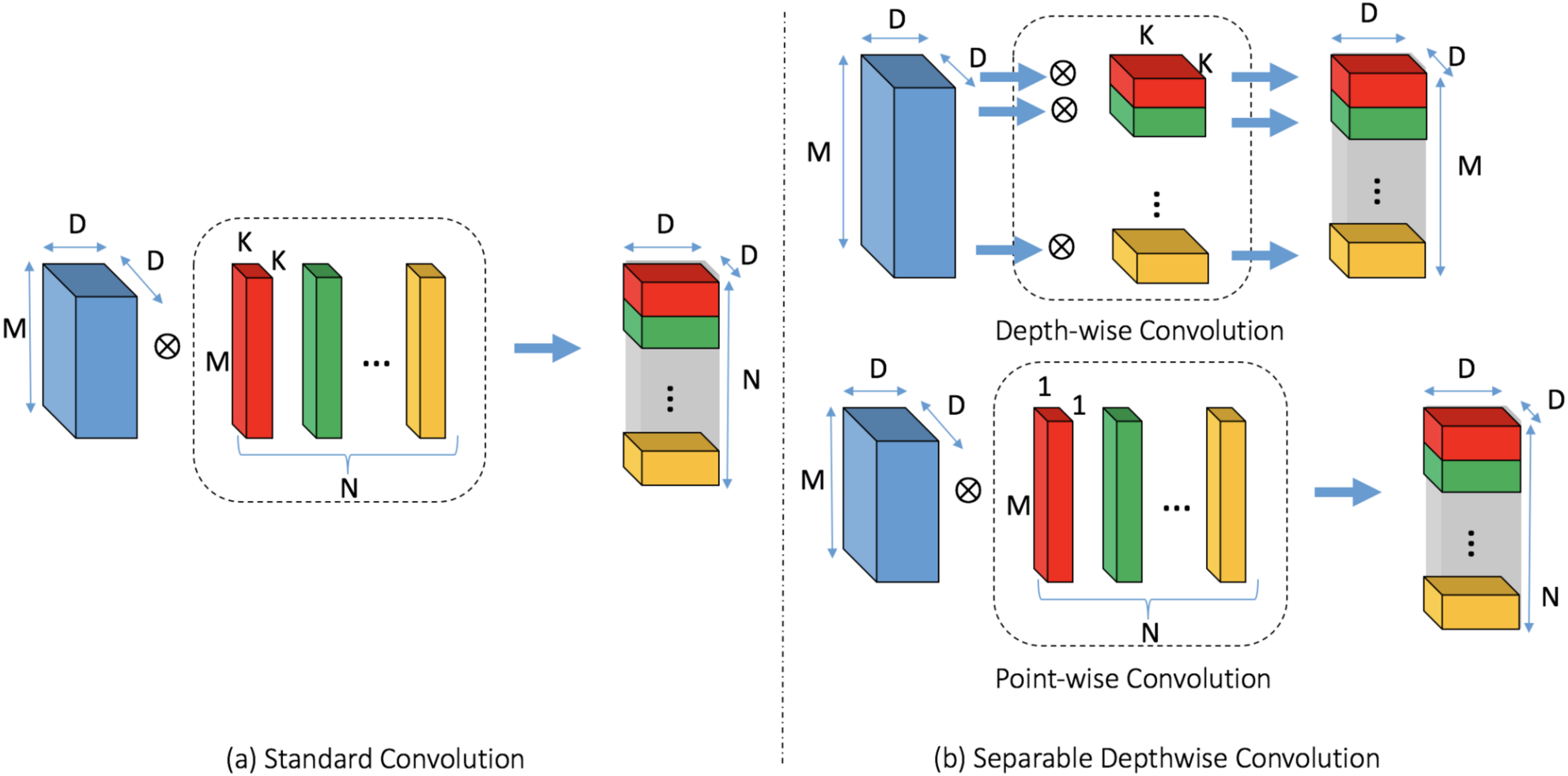
$K \times K \times M \times N$



(b) Separable Depthwise Convolution

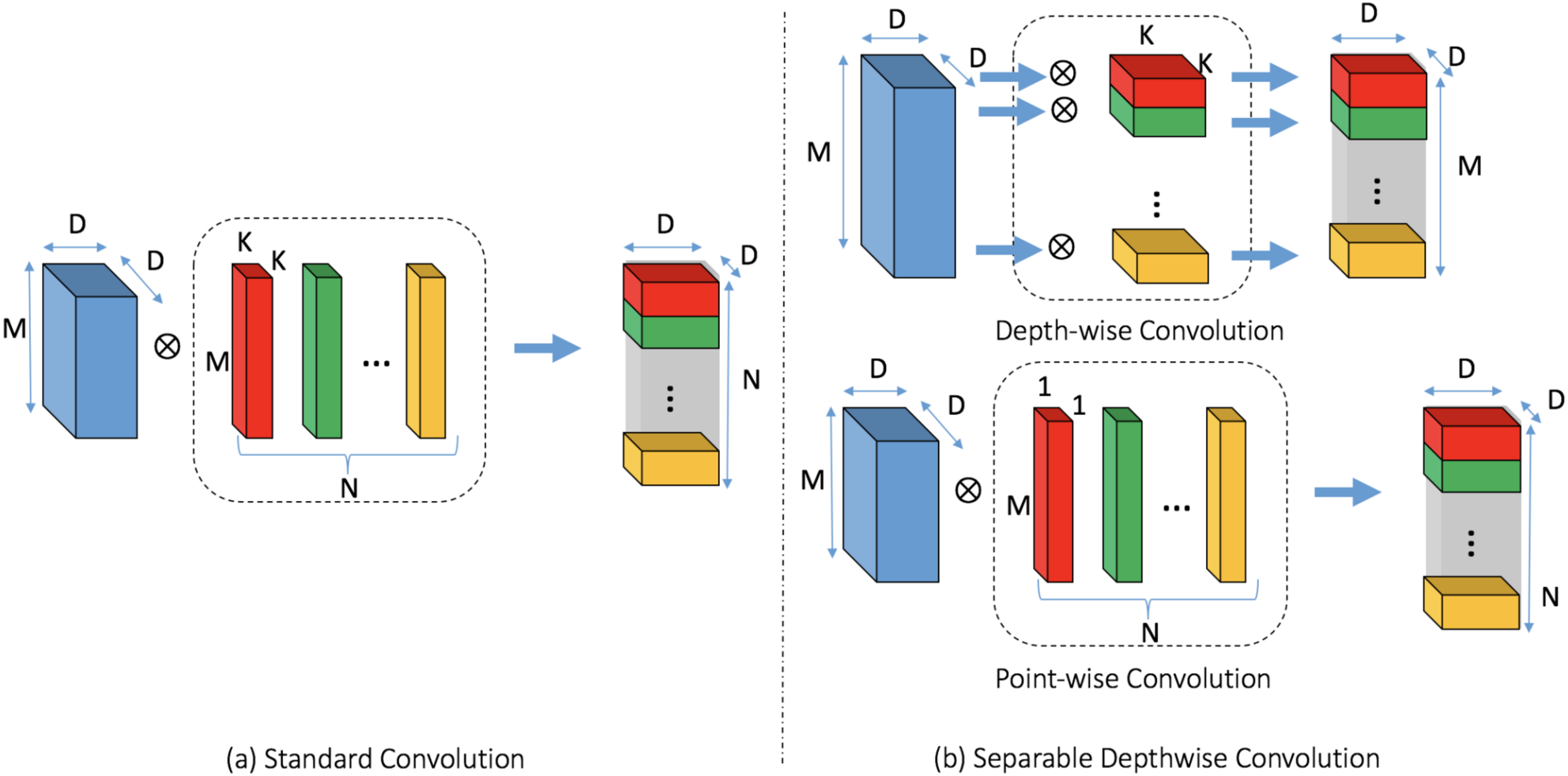
$K \times K \times M + 1 \times 1 \times M \times N$

MobileNet-v1 (2015)



Question: What is the computation cost?

MobileNet-v1 (2015)



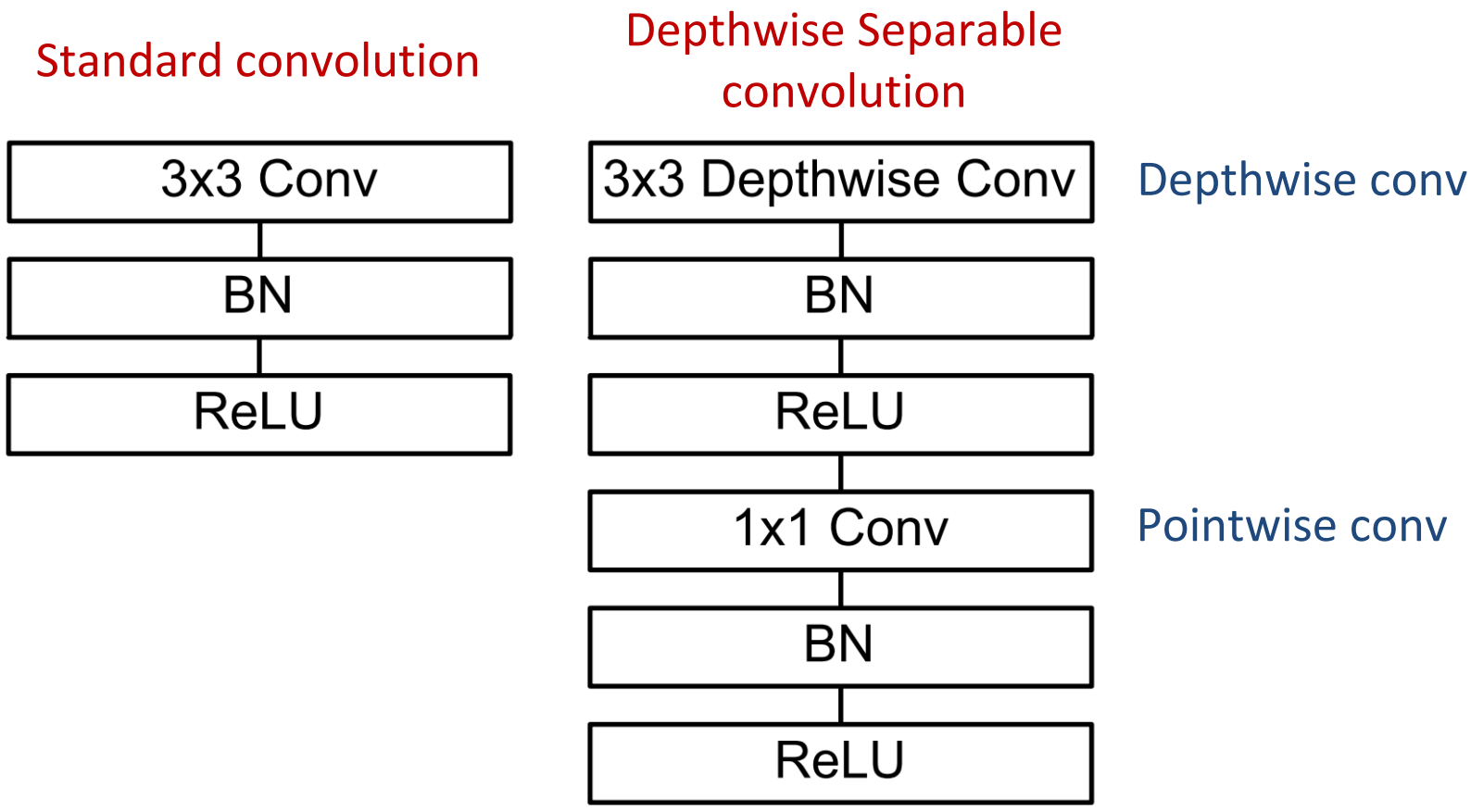
(a) Standard Convolution

(b) Separable Depthwise Convolution

Cost of one convolution: $K \times K \times M$
Cost of $D \times D$ convolutions: $K \times K \times M \times D \times D$
Total cost of N filters: $K \times K \times M \times D \times D \times N$

Cost of depthwise: $K \times K \times M \times D \times D$
Cost of pointwise: $1 \times 1 \times M \times D \times D \times N$
Total cost: $K \times K \times M \times D \times D + M \times D \times D \times N$

MobileNet-v1 (2015)



Left: **Standard convolutional** layer with batch norm and ReLU. Right: **Depthwise Separable convolutions** with Depthwise and Pointwise layers followed by batch norm and ReLU.

MobileNet-v1 (2015) Depthwise conv Pointwise conv

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Table 10. MobileNet for Stanford Dogs

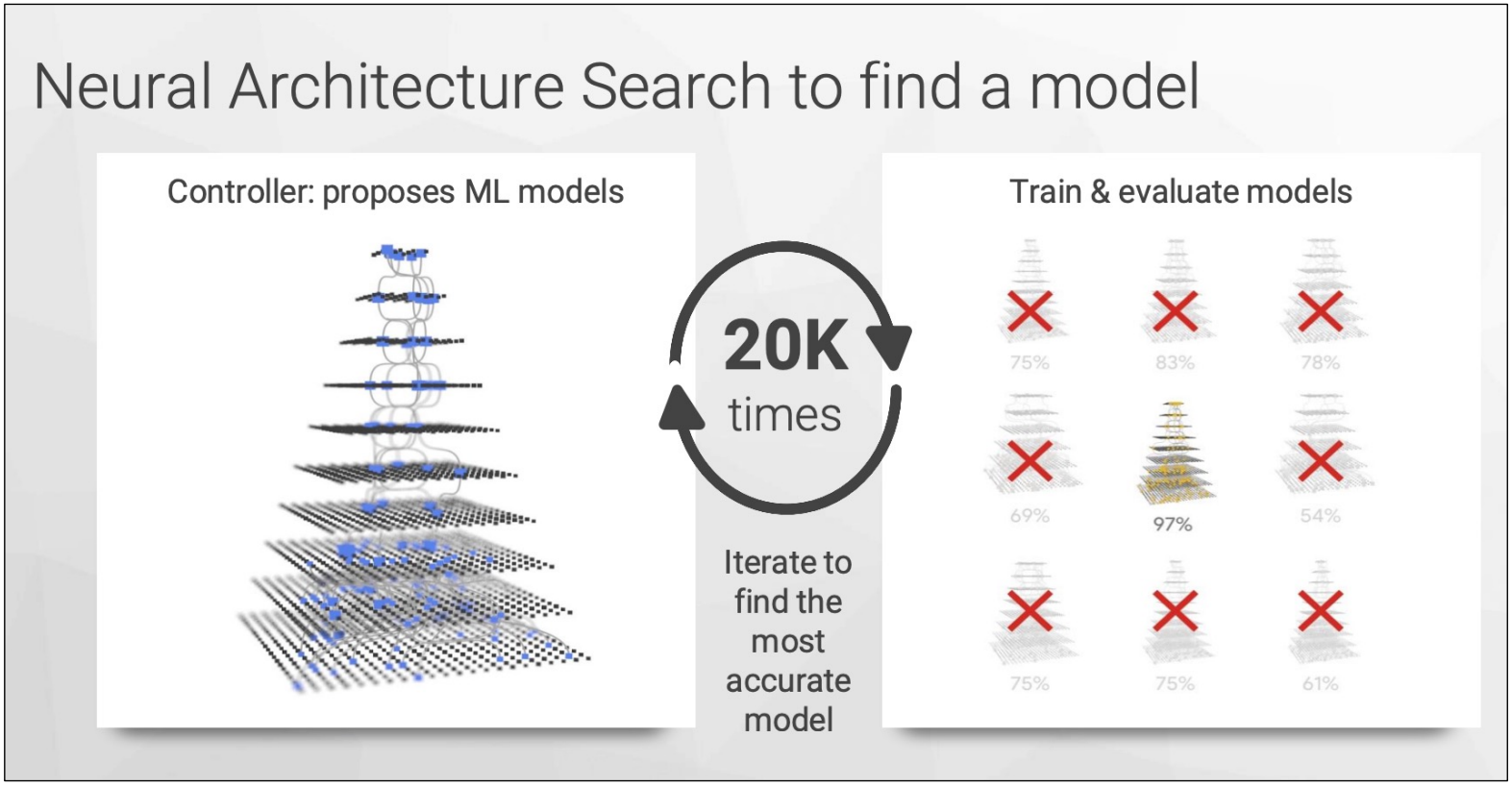
Model	Top-1 Accuracy	Million Mult-Adds	Million Parameters
Inception V3 [18]	84%	5000	23.2
1.0 MobileNet-224	83.3%	569	3.3
0.75 MobileNet-224	81.9%	325	1.9
1.0 MobileNet-192	81.9%	418	3.3
0.75 MobileNet-192	80.5%	239	1.9

- GoogleNet: Inception-v1
- 1.0 MobileNet-224: baseline MobileNet with 224x224 input.
- 0.75, 0.5 MobileNet: thin MobileNet (reduce input/output channels thanks to dw)

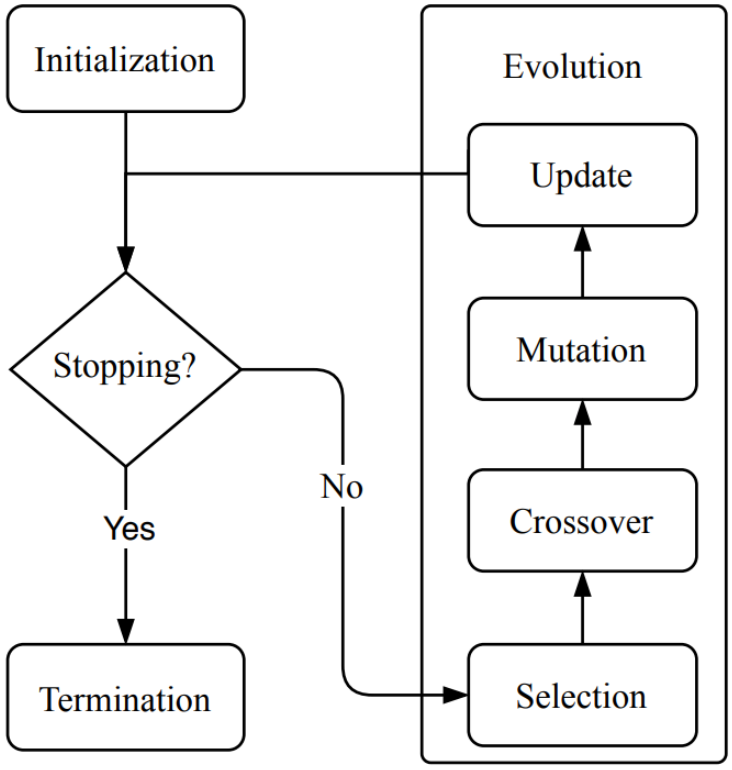
Network Architecture Search (NAS)

Neural architecture search (NAS) is a technique for **automating** the design of Deep Neural Network. NAS has been used to design networks that **outperform** hand-designed architectures. Some approaches of NAS:

- Evolution-based NAS.
- Reinforcement learning based NAS.
- Gradient-based NAS.



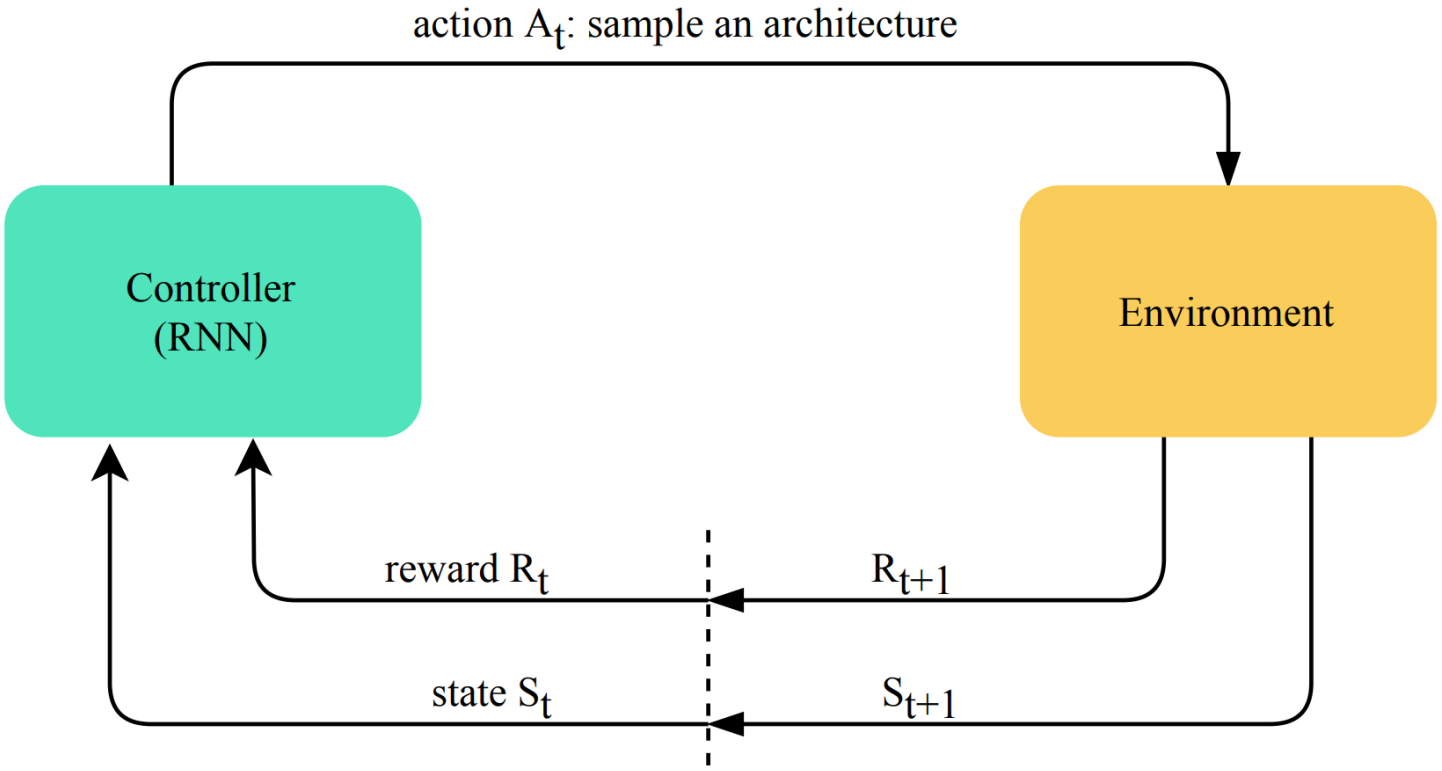
Evolution-based NAS



Evolution-based NAS (inspired by biological evolution).
The search space is assumed to be discrete.

- **Selection** & **Update**: initialize the population and update it with best candidates.
- **Crossover**: two networks are randomly selected to generate a new offspring network, through one- or multiple-point crossover.
- **Mutation**: an offspring network is copied and inherited from a parent network with mutation, e.g., adds or removes skip connections between two nodes or layers in the parent network.

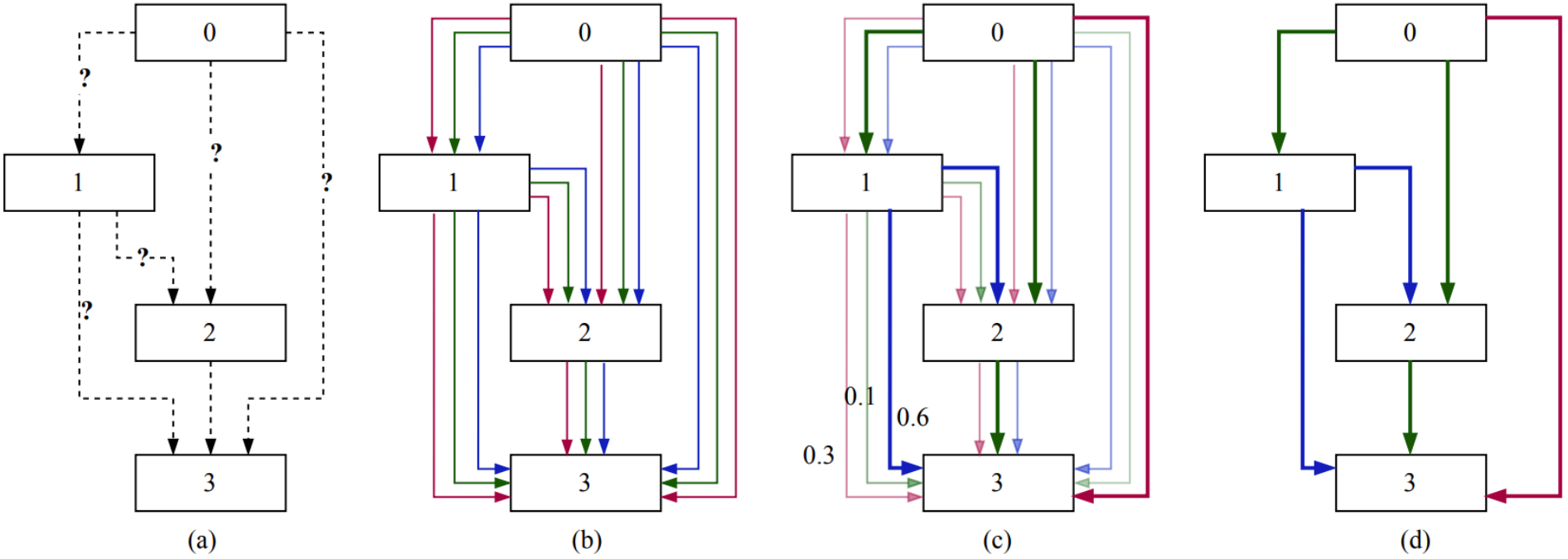
Reinforcement learning based NAS



Reinforcement learning based NAS.
The search space is assumed to be discrete.

- **Environment**: refers to the use of a standard neural network training procedure to train and evaluate the network generated by the controller.
- **Controller**: executes an action A_t at each step t to sample a new architecture from the search space and receives an a reward R_t from the environment to update the controller's sampling strategy.

Gradient-based NAS



Gradient-based NAS (Differentiable Architecture Search, DARTS)
The search space is assumed to be continuous and differentiable.

(a) The data can only flow from lower-level nodes to higher-level nodes, and the operations on edges are initially unknown. **(b)** The initial operation on each edge is a mixture of candidate operations, each having equal weight. **(c)** The weight of each operation is learnable and ranges from 0 to 1. **(d)** The final neural architecture is constructed by preserving the maximum weight-value operation on each edge.

Summary

Popular network architectures

- LeNet
- AlexNet
- VGG16
- Inception v1
- ResNet
- DenseNet
- UNet
- MobileNet v1

Network Architecture Search (NAS)

Q&A

Thank you