2024-2

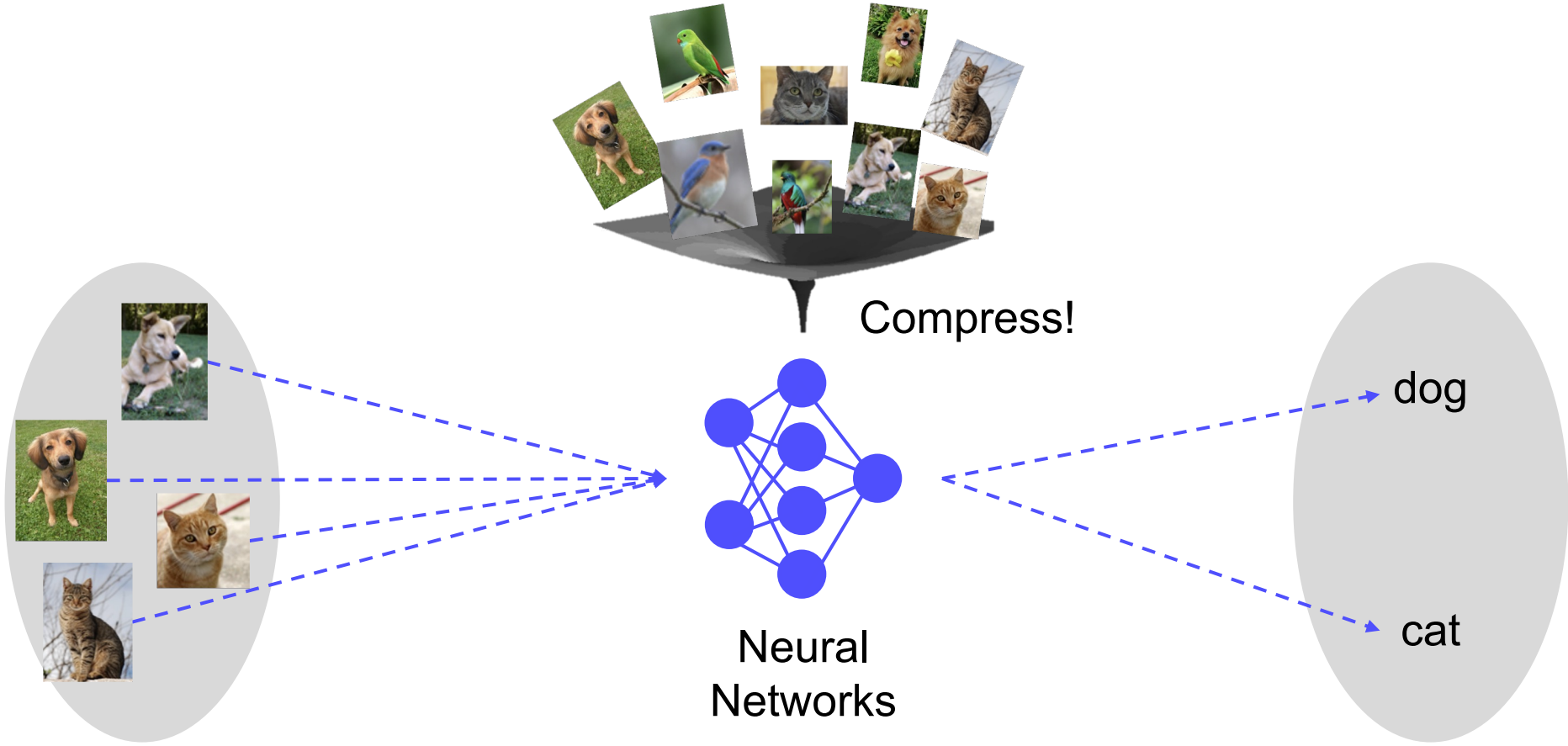
Deep Learning Programming

Lecture 3.2: Brief history of CNN

Sangryul Jeon
School of Computer Science and Engineering
srjeonn@pusan.ac.kr

1.1 Convolutional Neural Networks (CNN)

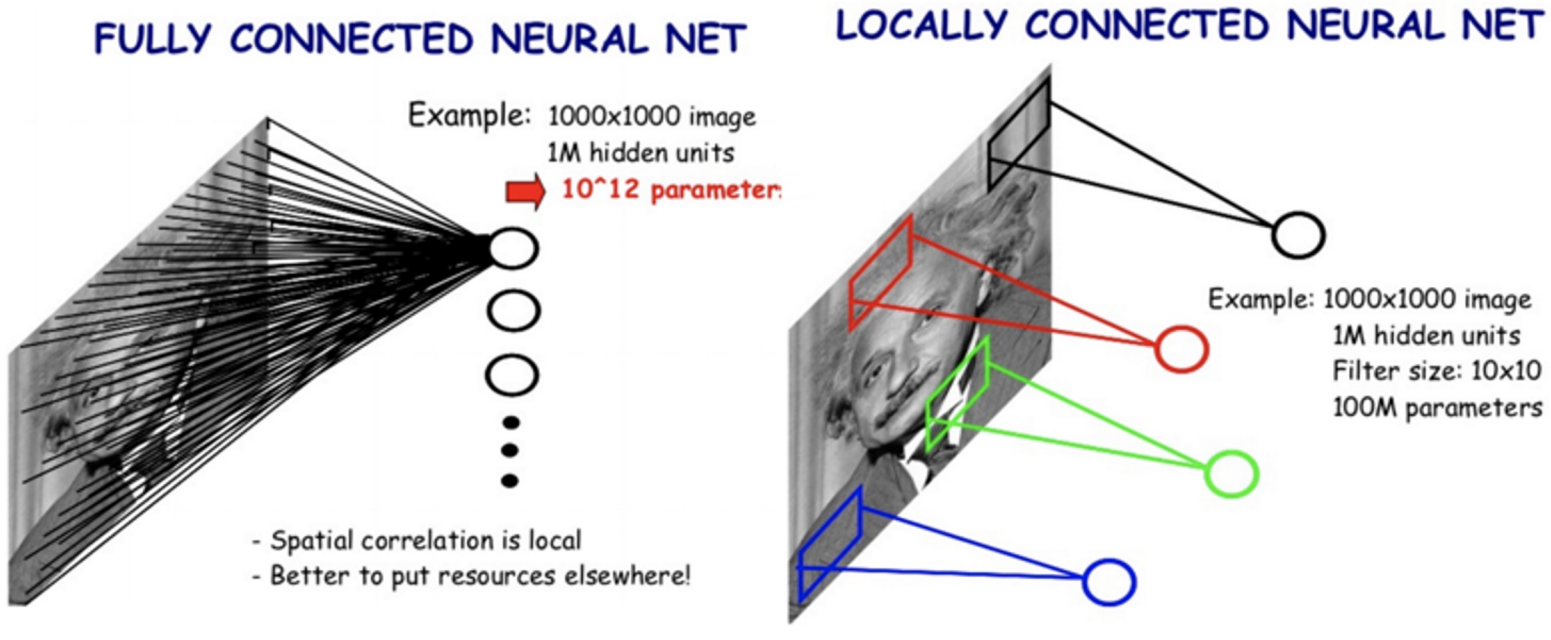
Compress all the data we have into the neural network



1.1 Convolutional Neural Networks (CNN)

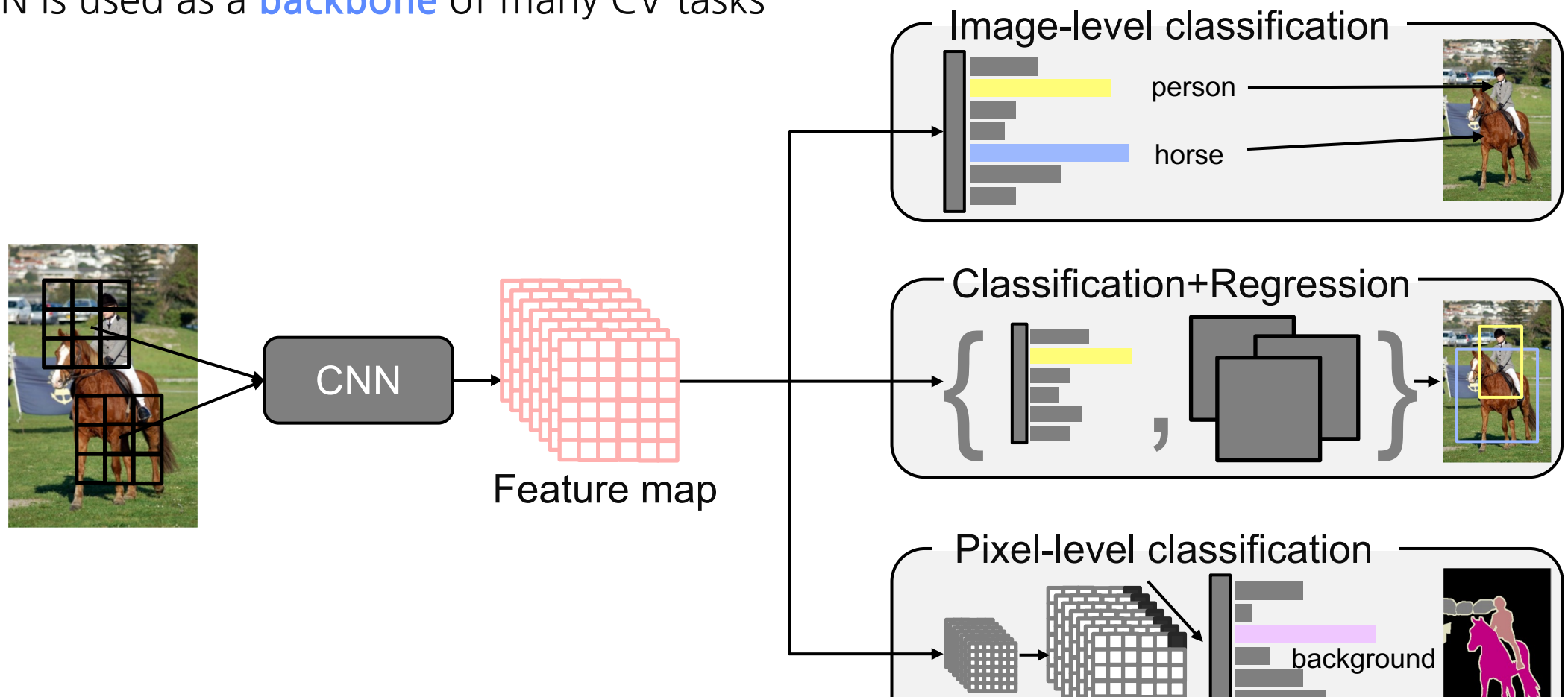
Convolution neural networks are fully locally connected neural networks

- Local feature learning
- Parameter sharing



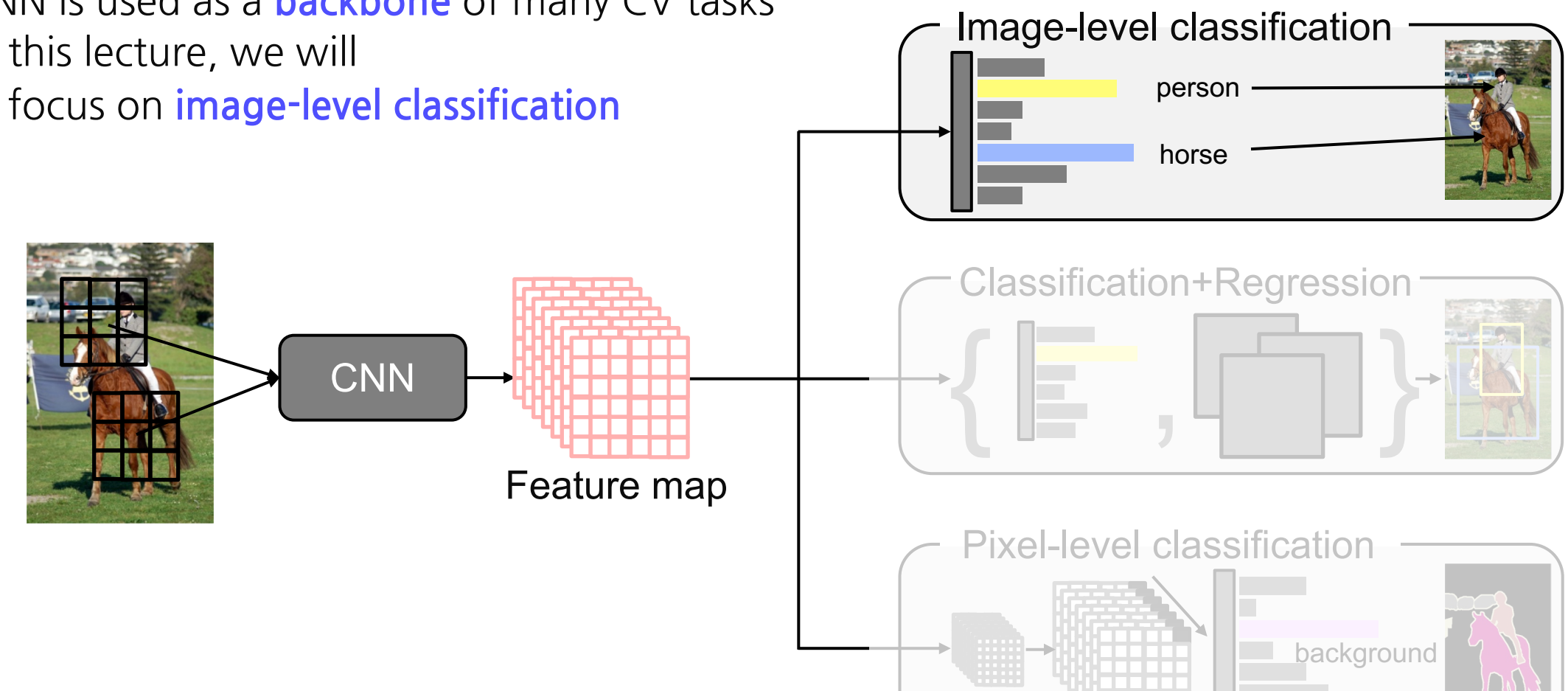
1.1 Convolutional Neural Networks (CNN)

- CNN is used as a **backbone** of many CV tasks

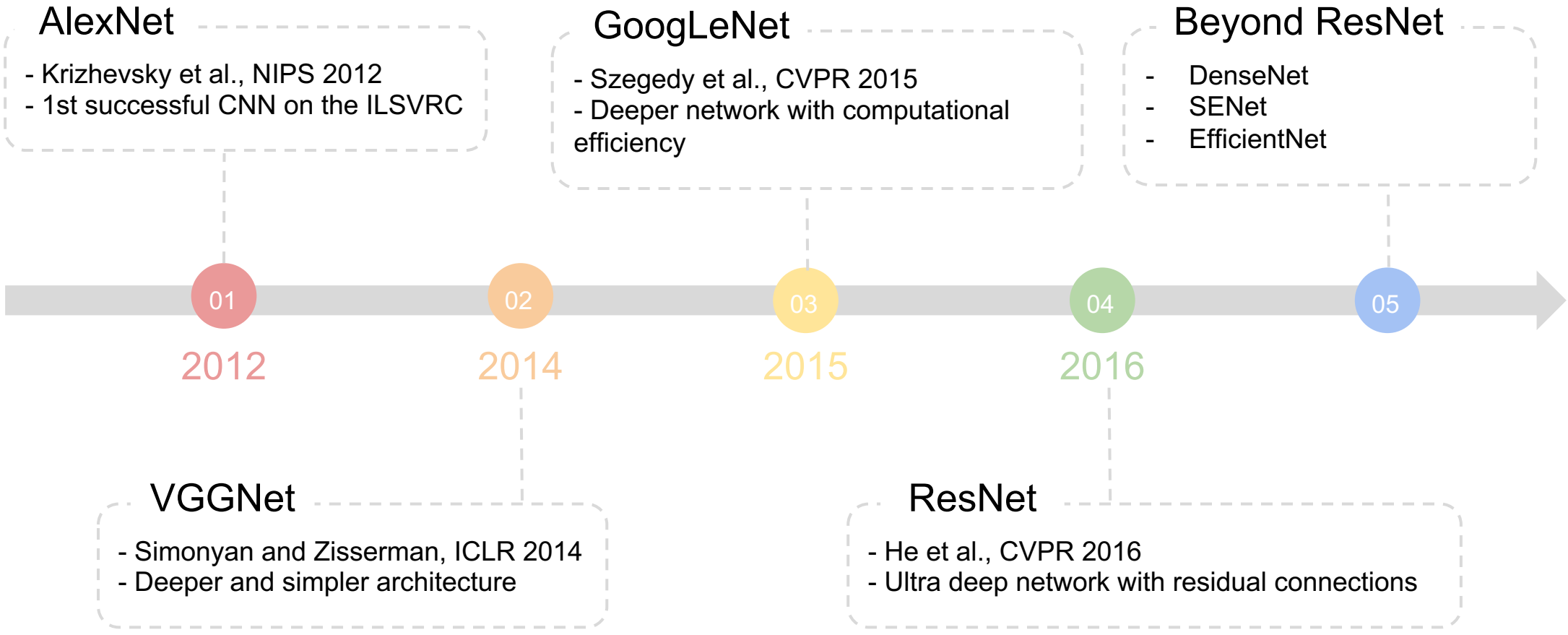


1.1 Convolutional Neural Networks (CNN)

- CNN is used as a **backbone** of many CV tasks
- In this lecture, we will
  focus on **image-level classification**



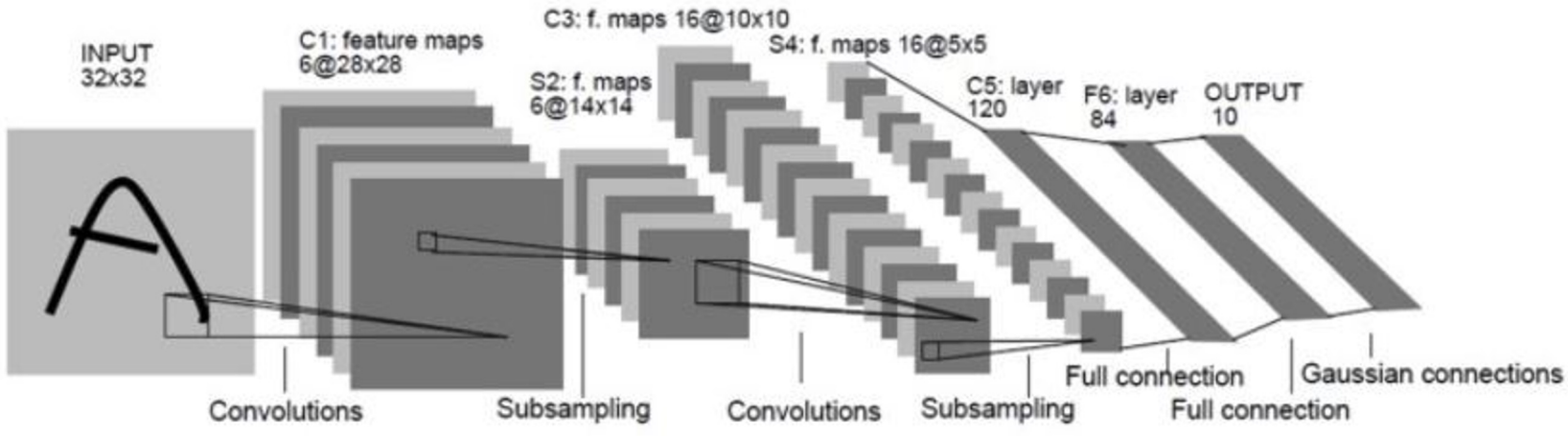
1.2 Brief history



1.2 Brief history

LeNet-5

- A very simple CNN architecture introduced by Yann LeCun in 1998
  - Overall architecture: Conv - Pool - Conv - Pool - FC - FC
  - Convolution: 5x5 filters with stride 1
  - Pooling: 2x2 max pooling with stride 2

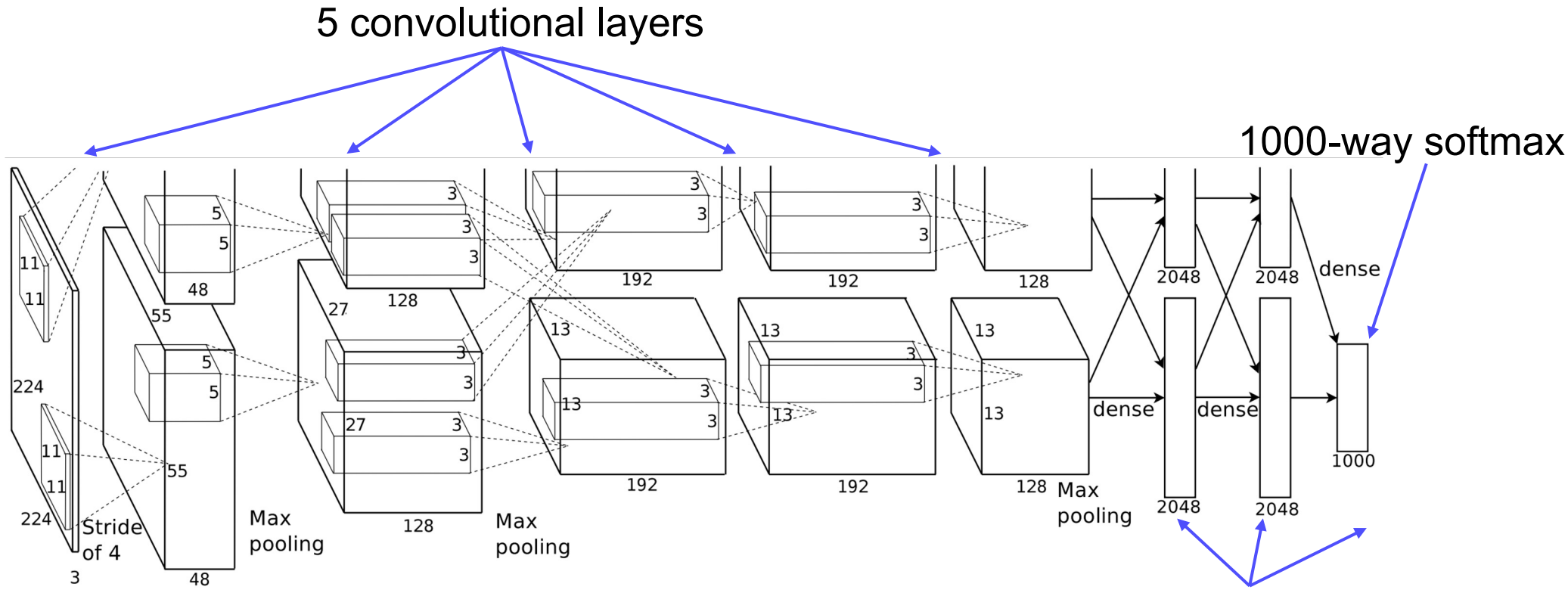


1.2 Brief history

AlexNet

* LRN = Local Response Normalization

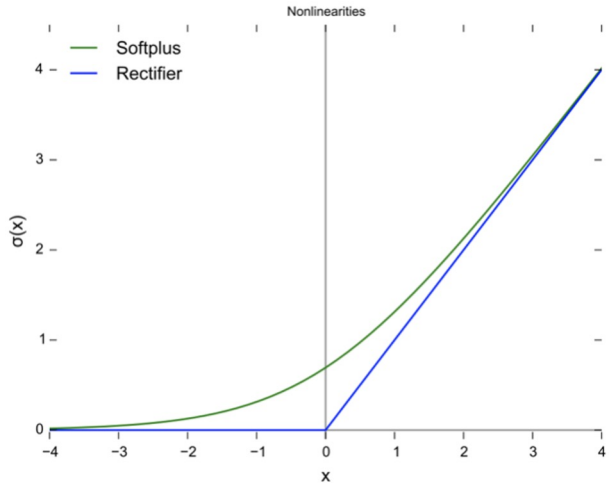
- Conv - Pool - LRN - Conv - Pool - LRN - Conv - Conv - Conv - Pool - FC - FC - FC



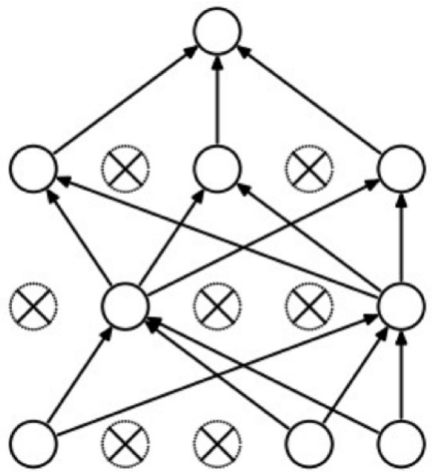
1.2 Brief history

AlexNet - Similar with LeNet-5, but

- Bigger (7 hidden layers, 605k neurons, 60 million parameters)
- Trained with ImageNet (large amount of data, 1.2 millions)
- Using better activation function (**ReLU**) and regularization technique (**dropout**)



ReLU

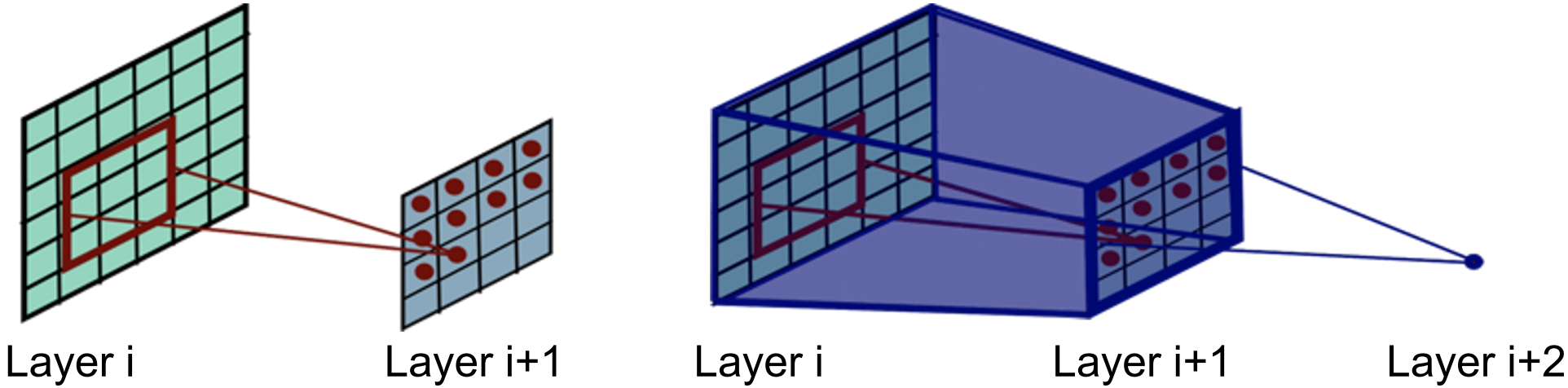


Dropout

1.2 Brief history

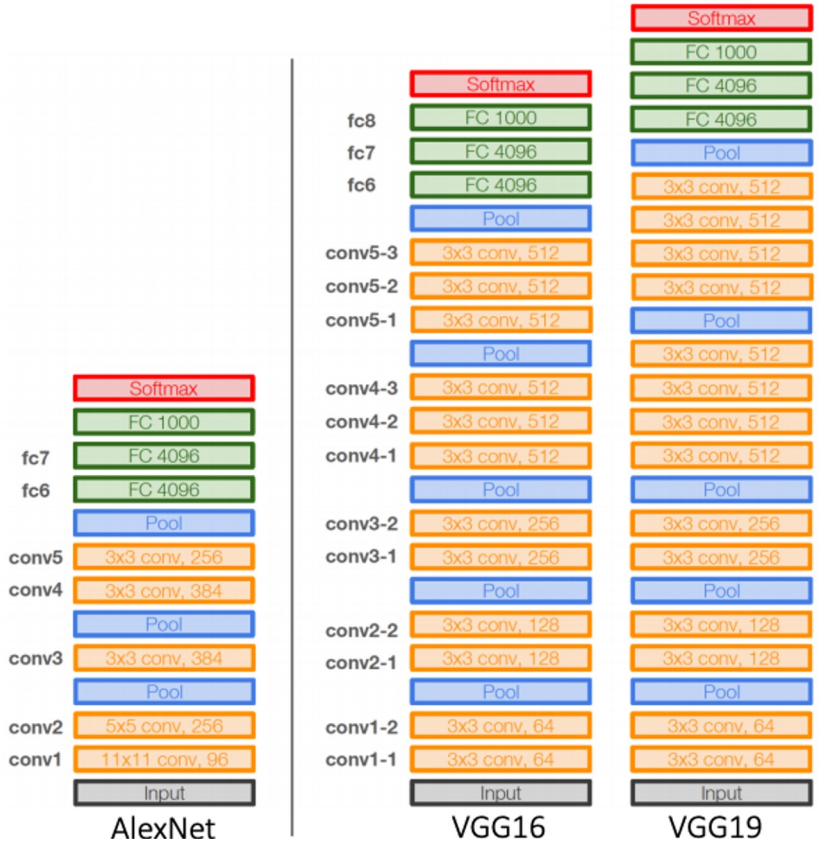
Receptive field in CNN

- The region in the input space that a particular CNN feature is looking at
- Suppose K×K conv. filters with stride 1, and a pooling layer of size P×P,
  - then a value of each unit in the pooling layer depends on an input patch of size : (P+K-1)×(P+K-1)



1.2 Brief history

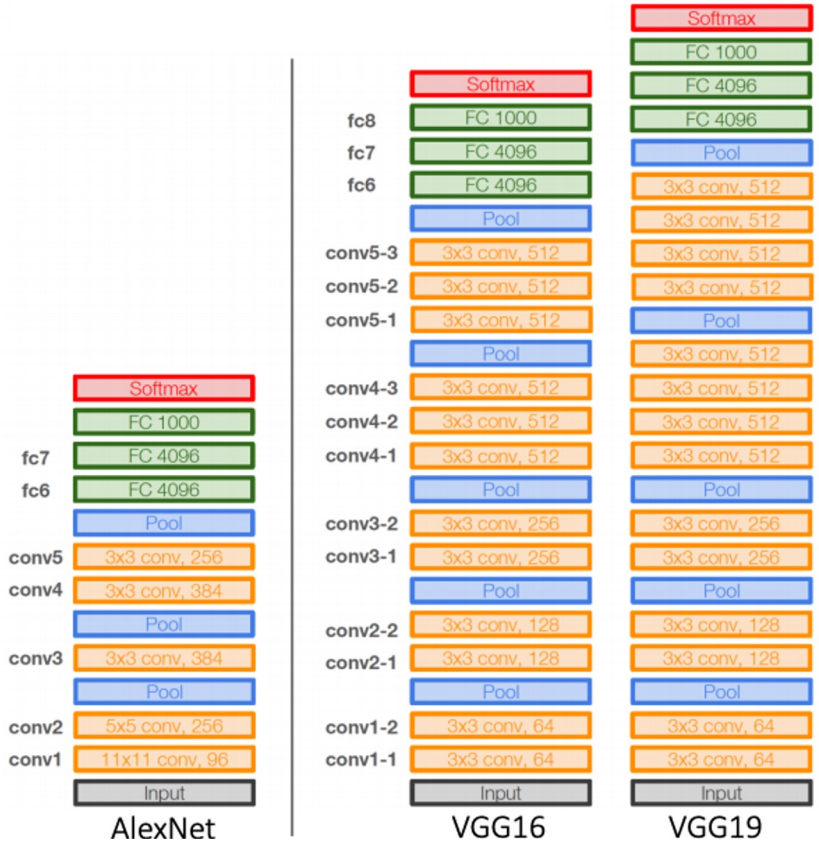
VGGNet



- Deeper architecture
  - 16 and 19 layers

1.2 Brief history

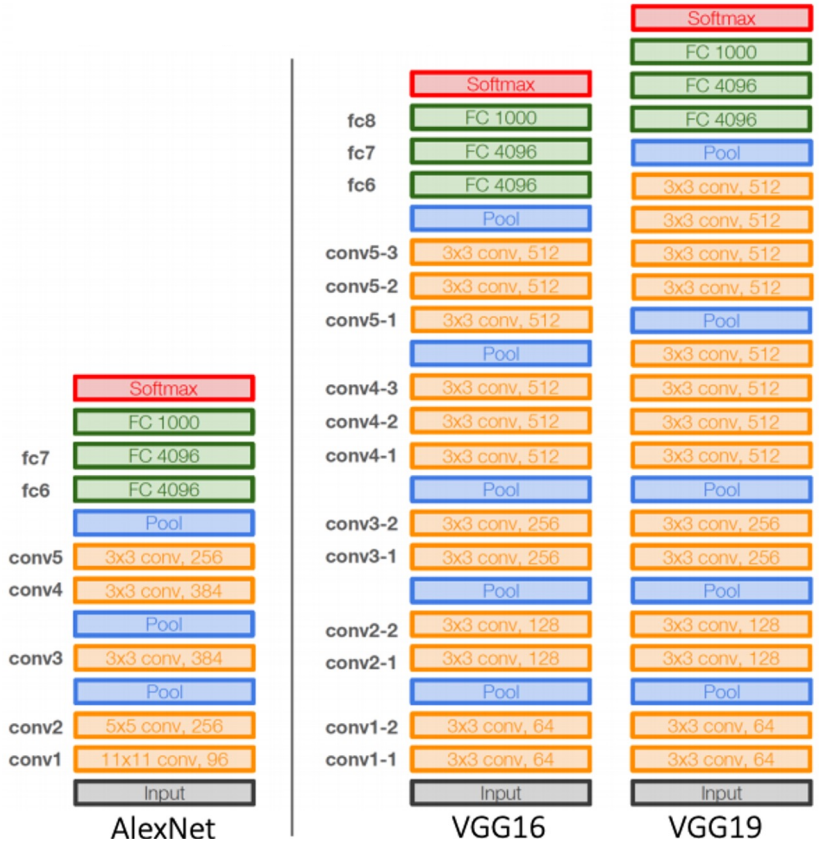
VGGNet



- Simpler architecture
  - No local response normalization
  - Only 3x3 conv filters blocks, 2x2 max pooling

1.2 Brief history

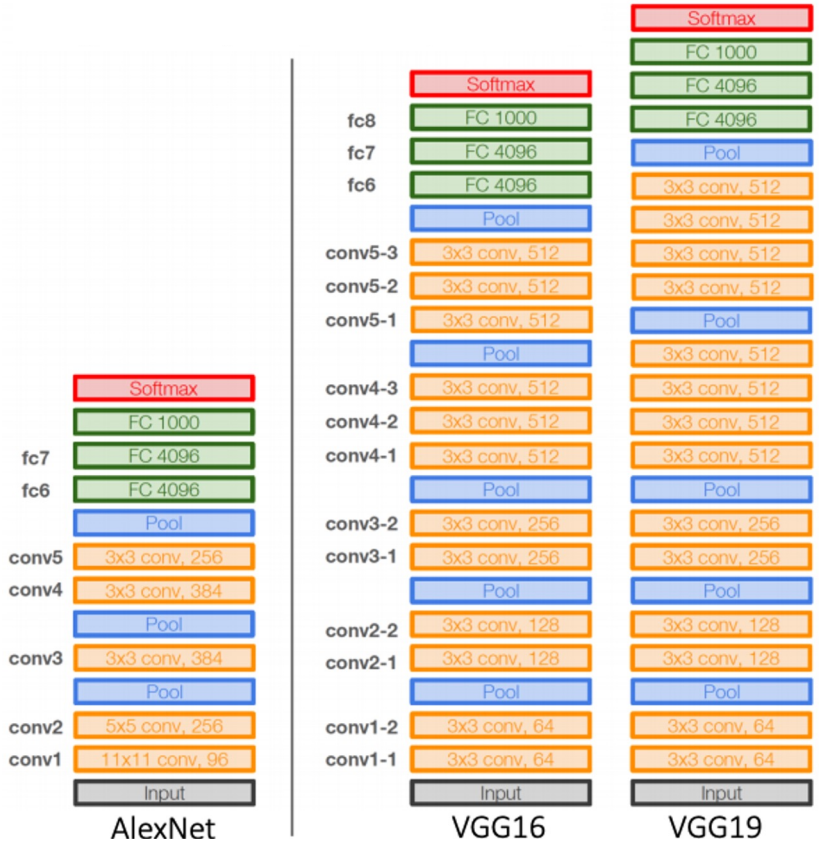
VGGNet



- Better performance
  - Significant performance improvement over AlexNet (2^{nd} in ILSVRC14)

1.2 Brief history

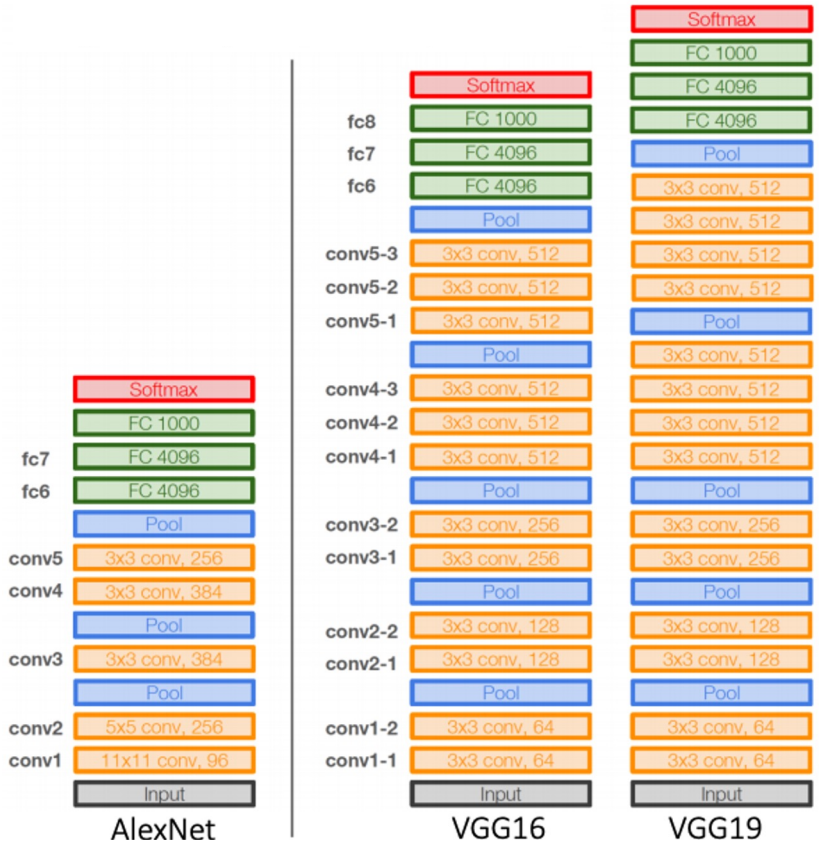
VGGNet



- Better generalization
  - Final features generalizing well to other tasks even without fine-tuning

1.2 Brief history

VGGNet



- Deeper architecture
- Simpler architecture
- Better performance
- Better generalization

VGGNet - Overall architecture

- Input

- 224x224 RGB images (same with AlexNet)
- Subtracting mean RGB values of training images

VGGNet - Overall architecture

- Key design choices

- 3x3 convolution filters with stride 1
- 2x2 max pooling operations

VGGNet - Overall architecture

- Key design choices

- 3x3 convolution filters with stride 1
- 2x2 max pooling operations

Using many 3x3 conv layers instead of a small number of larger conv filters

- Keeping receptive field sizes large enough
- Deeper with more non-linearities
- Fewer parameters

VGGNet - Overall architecture

- 3 fully-connected (FC) layers

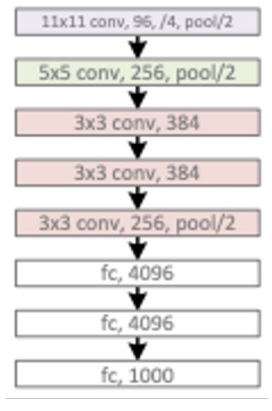
- Other details

- ReLU for non-linearity
- No local response normalization

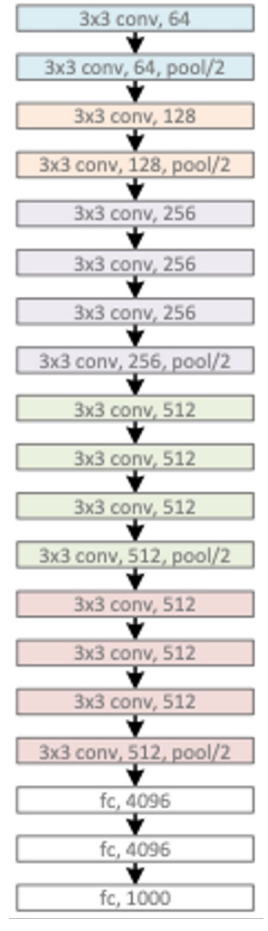
1.3 Going deeper with convolutions

The neural network is getting deeper and wider

- Deeper networks learn more powerful features, because of
  - Larger receptive fields
  - More capacity and non-linearity



AlexNet (8 layers)



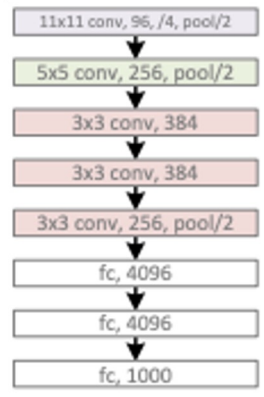
VGGNet (19 layers)

1.3 Going deeper with convolutions

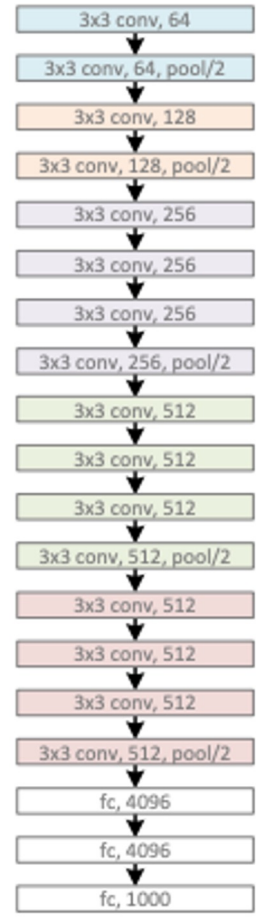
The neural network is getting deeper and wider

- Deeper networks learn more powerful features, because of
  - Larger receptive fields
  - More capacity and non-linearity

- But, getting deeper and deeper always works better?



AlexNet (8 layers)

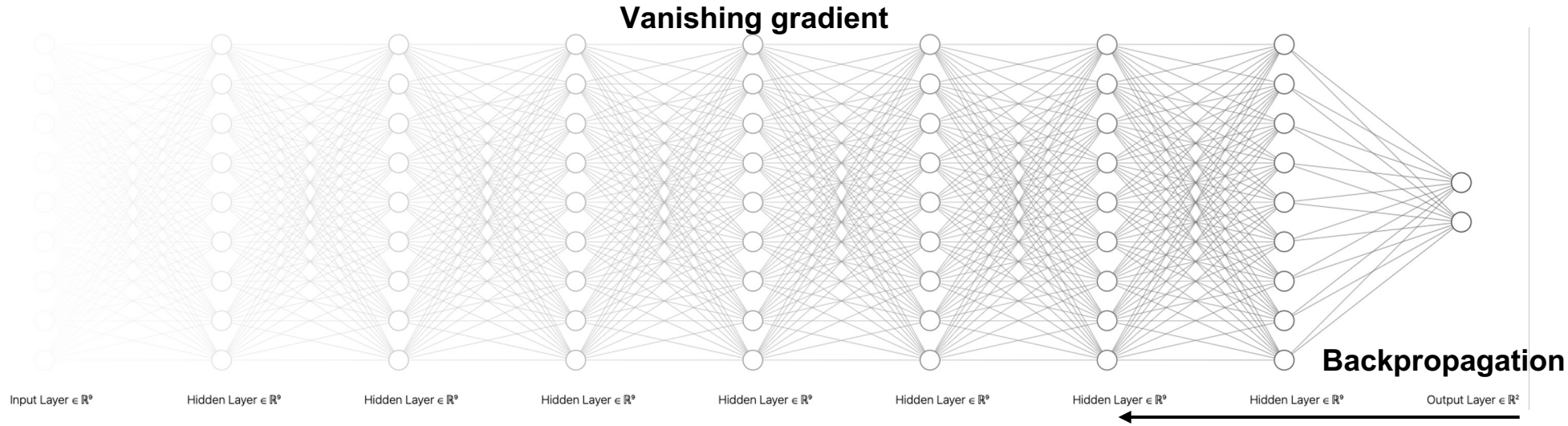


VGGNet (19 layers)

1.3 Going deeper with convolutions

Deeper networks are harder to optimize

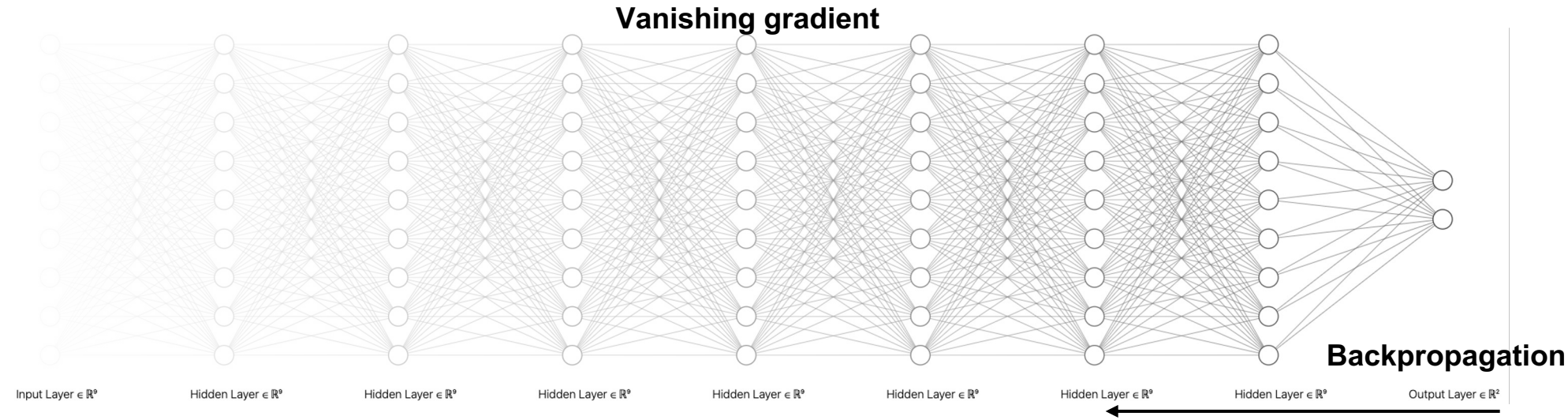
- Gradient vanishing / exploding



1.3 Going deeper with convolutions

Deeper networks are harder to optimize

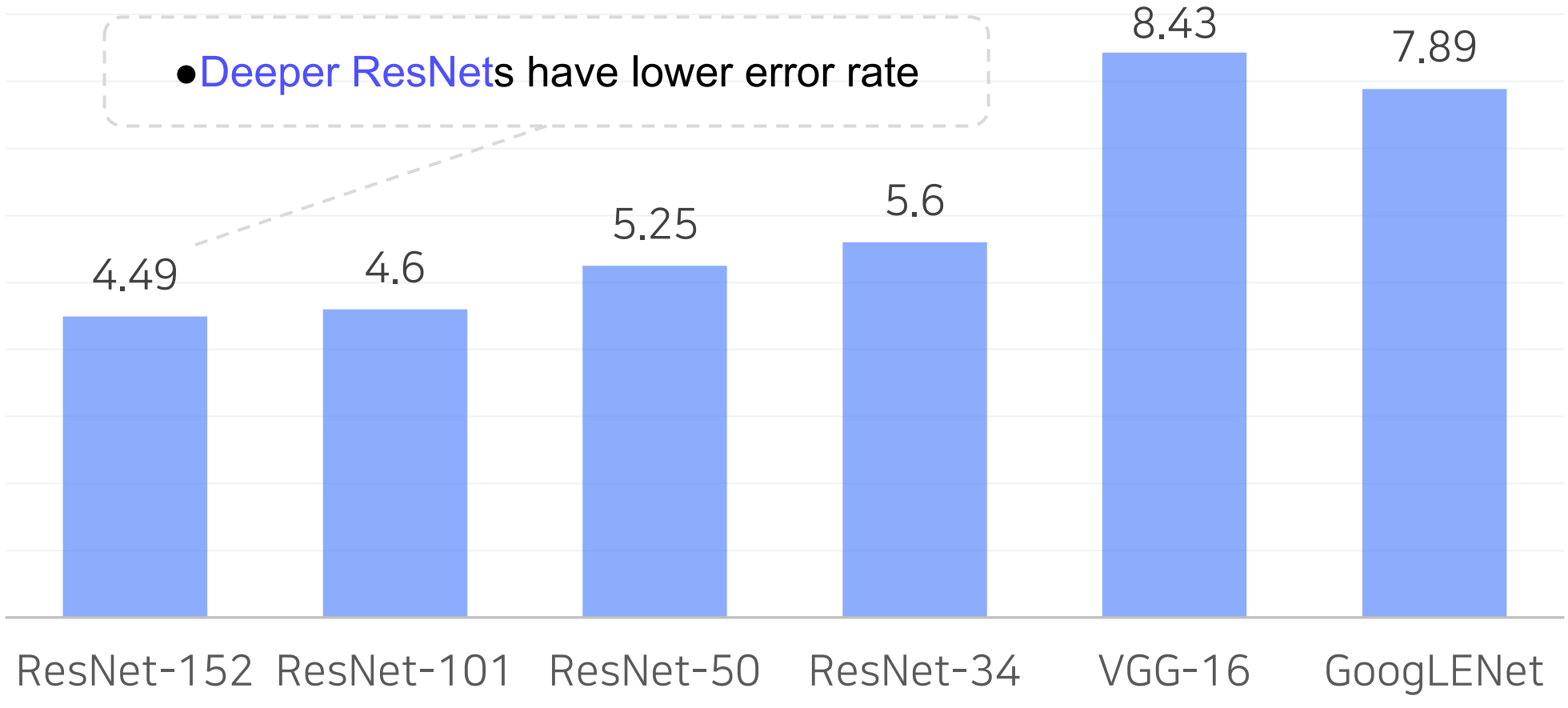
- Gradient vanishing / exploding
- ~~Overfitting problem~~ Degradation problem



1.3 Going deeper with convolutions

ResNet

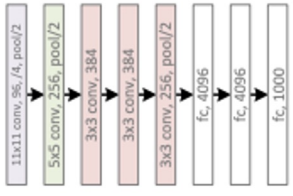
- Top-5 validation error on the ImageNet



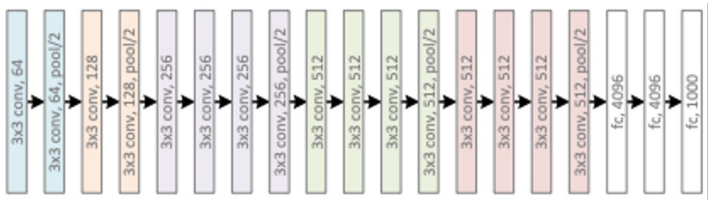
1.3 Going deeper with convolutions

Revolutions of depth

- Building ultra-deeper than any other networks
- What makes it hard to build a very deep architecture?



AlexNet (8 layers)



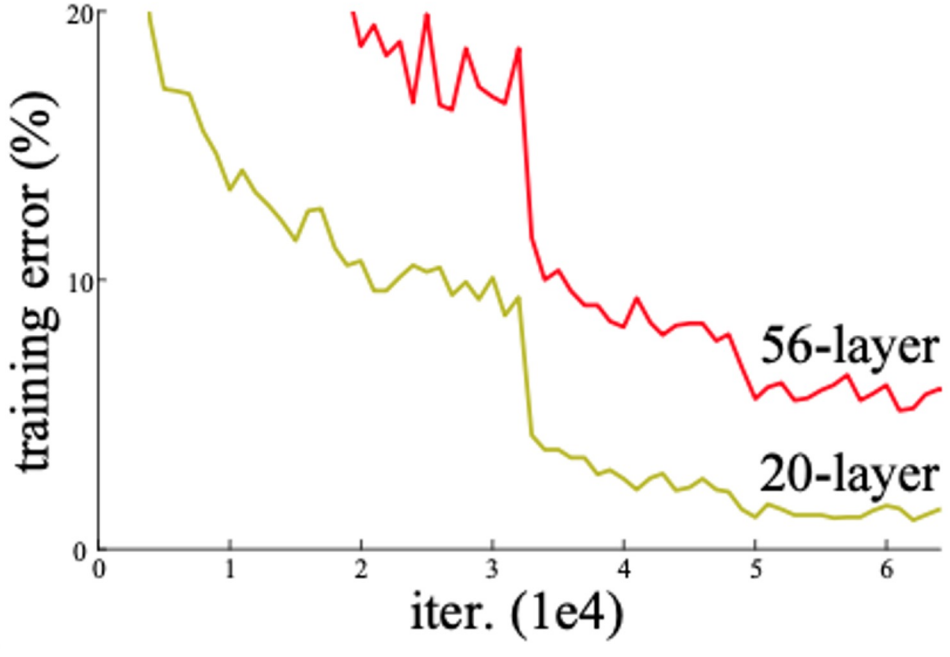
VGGNet (19 layers)

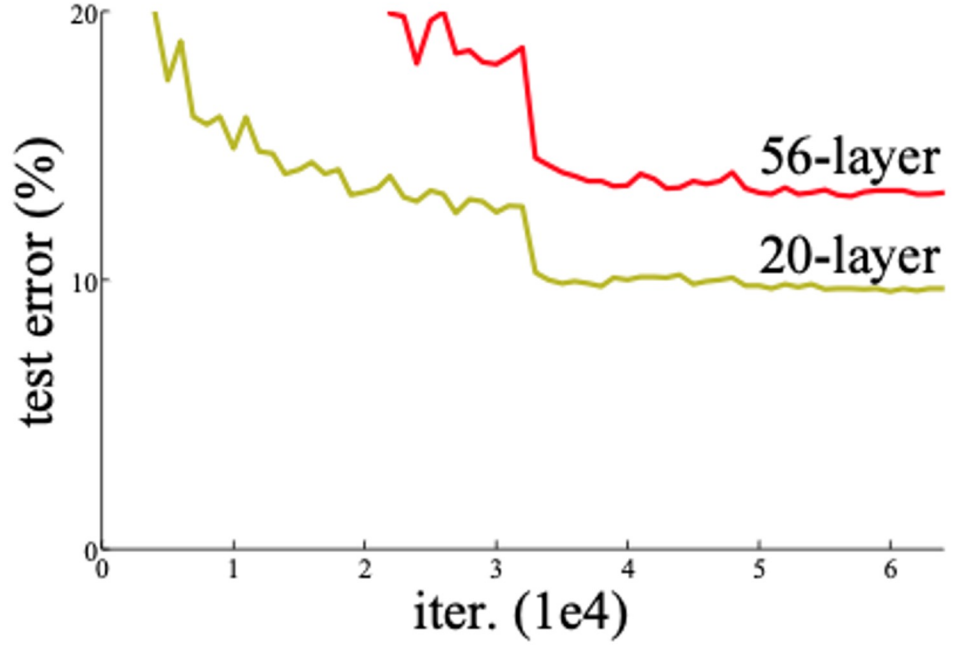
ResNet (152 layers)

1.3 Going deeper with convolutions

Degradation problem

- As the network depth increases, accuracy gets saturated ⇒ **degrade rapidly**

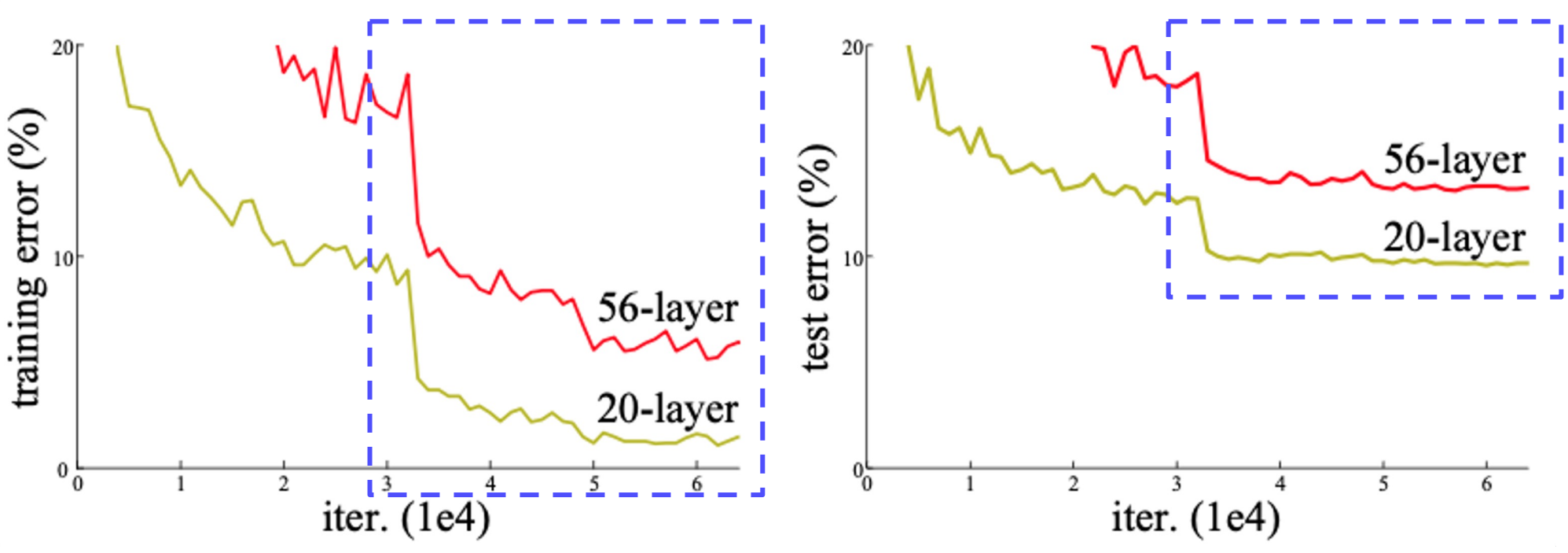




1.3 Going deeper with convolutions

Degradation problem

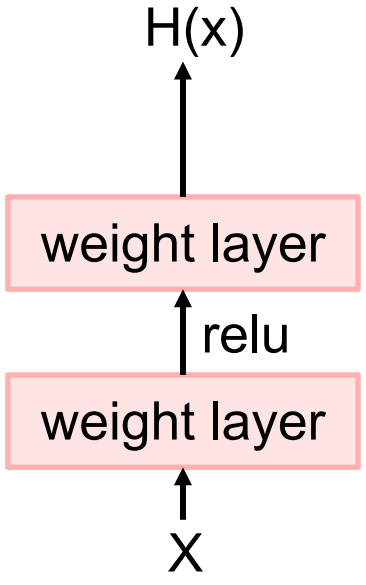
- As the network depth increases, accuracy gets saturated ⇒ **degrade rapidly**
- This is not caused by overfitting. The problem is **optimization!**



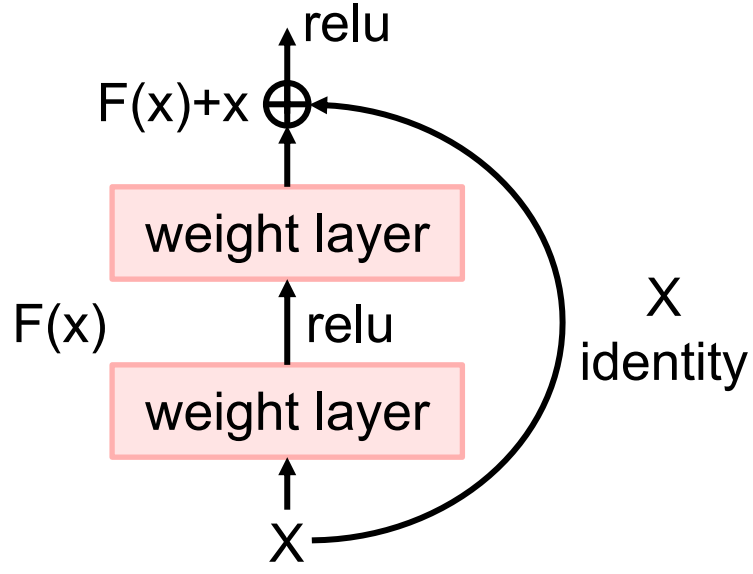
ResNet - Hypothesis

- Plain layer: As the layers get deeper, it is hard to learn good $\boldsymbol{H}(x)$ directly

"Plain" layers



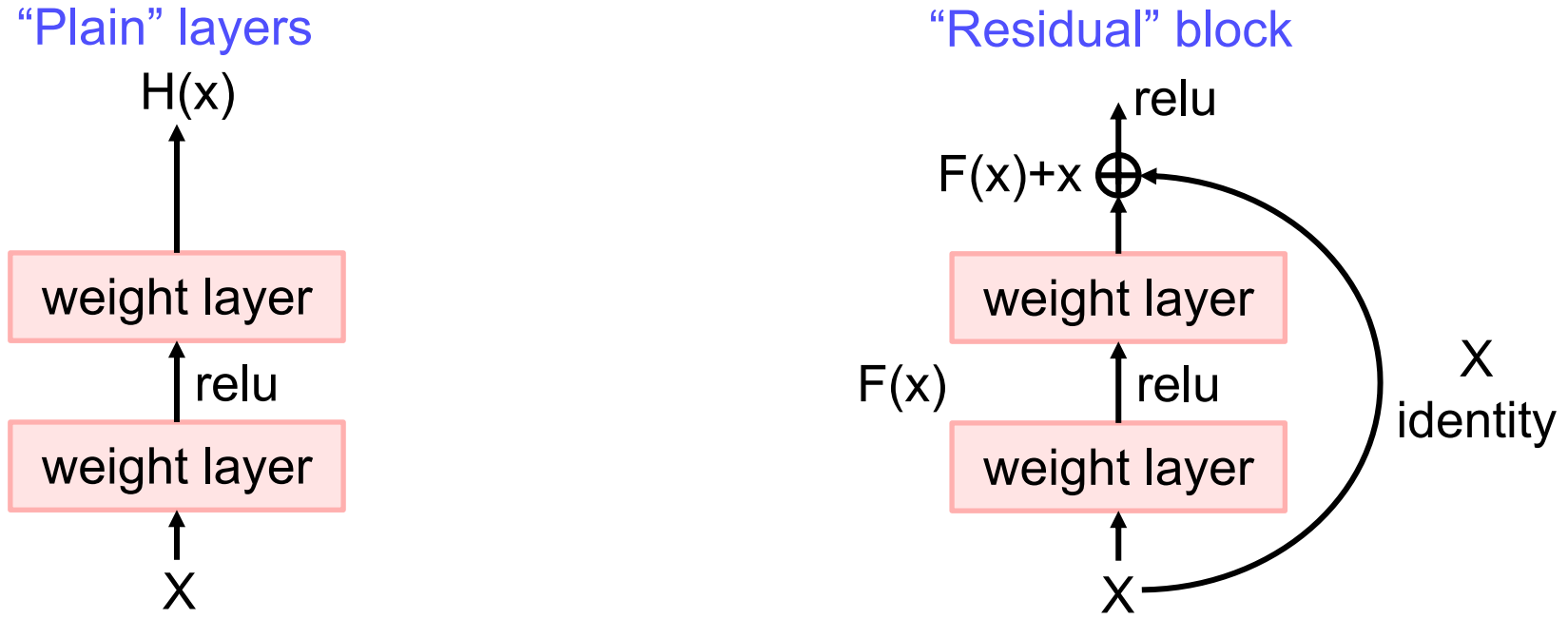
"Residual" block



1.3 Going deeper with convolutions

ResNet - Hypothesis

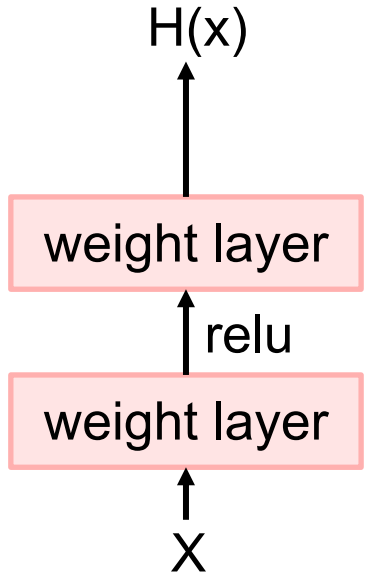
- Plain layer: As the layers get deeper, it is hard to learn good $\boldsymbol{H}(x)$ directly
- Residual block: Instead, we learn residual
  - Target function : $H(x) = F(x) + x$
  - Residual function : $F(x) = H(x) - x$

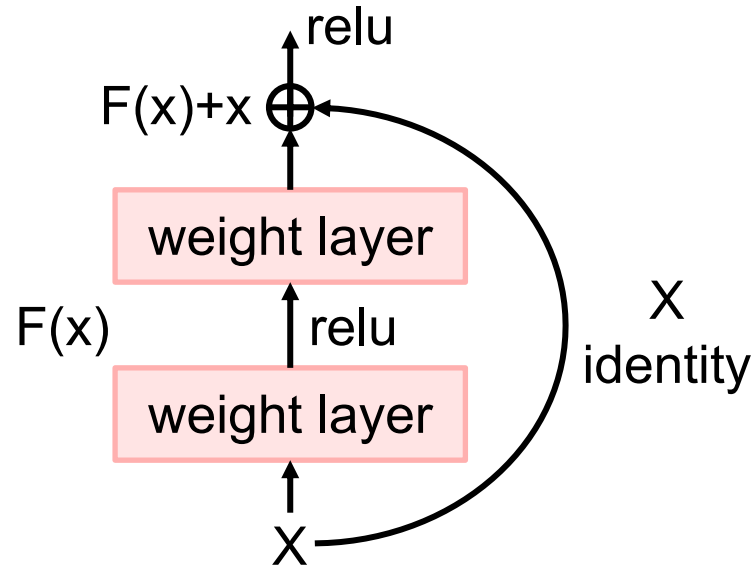


1.3 Going deeper with convolutions

ResNet - A solution: Shortcut connection

- Use layers to fit a residual mapping instead of directly fitting a desired underlying mapping
- The vanishing gradient problem is solved by shortcut connection
- Don’t just stack layers up, instead use shortcut connection!

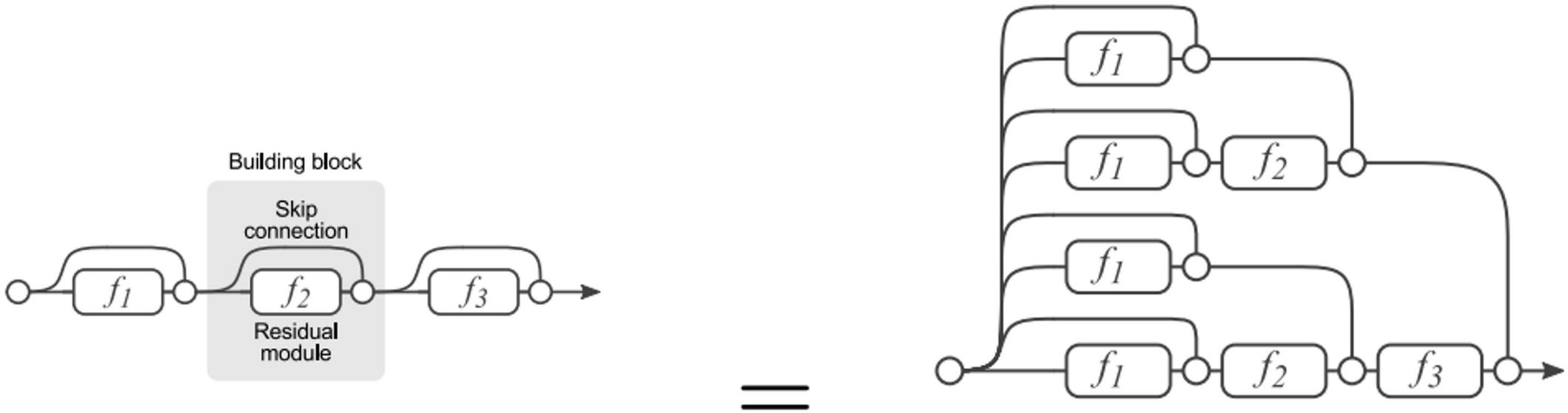




1.3 Going deeper with convolutions

ResNet - Analysis of residual connection

- During training, gradients are mainly from relatively shorter paths
- Residual networks have $O(2^n)$ implicit paths connecting input and output, and adding a block doubles the number of paths

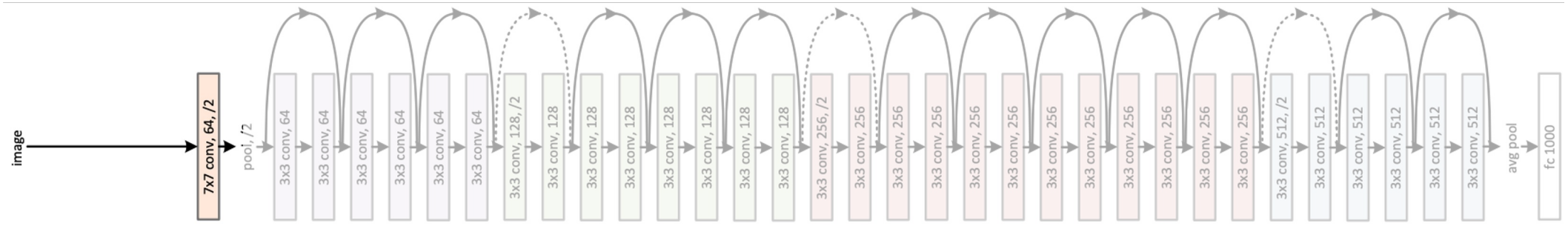


(a) Conventional 3-block residual network

(b) Unraveled view of (a)

ResNet - Overall architecture

- He initialization
- Additional conv layer at the beginning

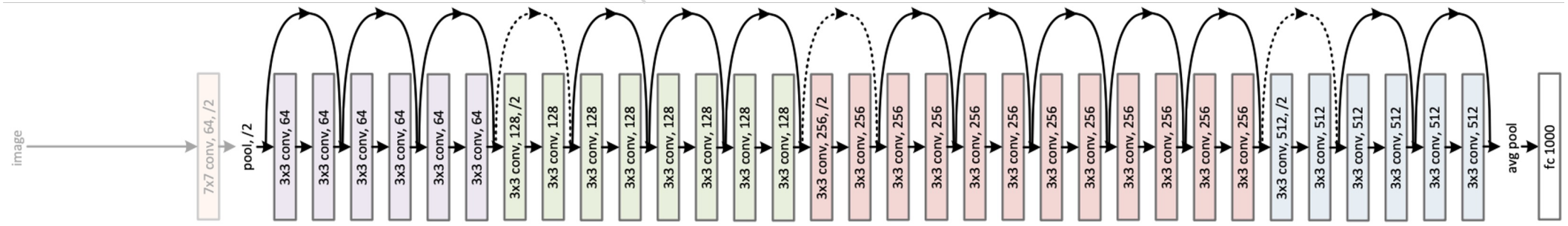


1.3 Going deeper with convolutions

ResNet - Overall architecture

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Batch norm after every conv layer

- He initialization
- Additional conv layer at the beginning



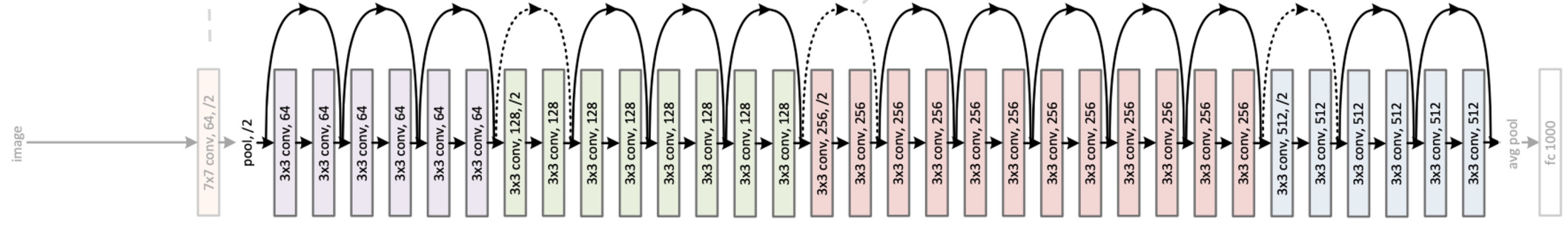
1.3 Going deeper with convolutions

ResNet - Overall architecture

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Batch norm after every conv layer

- Doubling the number of filters and spatially down-sampling by stride 2 instead of spatial pooling

- He initialization
- Additional conv layer at the beginning



1.3 Going deeper with convolutions

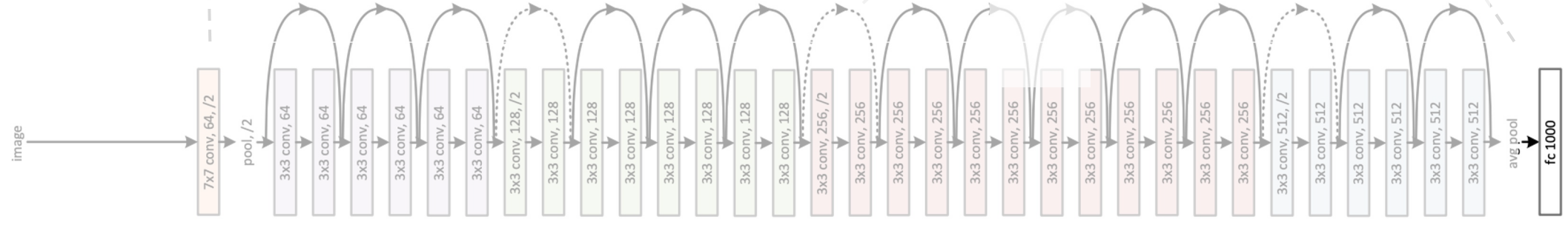
ResNet - Overall architecture

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Batch norm after every conv layer

- Doubling the number of filters and spatially down-sampling by stride 2 instead of spatial pooling

- Only a single FC layer for output classes

- He initialization
- Additional conv layer at the beginning



ResNet - Overall architecture

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, \\ 3\times3, \\ 1\times1, \end{bmatrix}$	
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, \\ 3\times3, \end{bmatrix}$	
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times3, 2 \\ 1\times1, 1 \end{bmatrix}$	
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, \\ 3\times3, \\ 1\times1, 2 \end{bmatrix}$	
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	$7.6\times$	

```
self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
self.fc = nn.Linear(512 * block.expansion, num_classes)
```

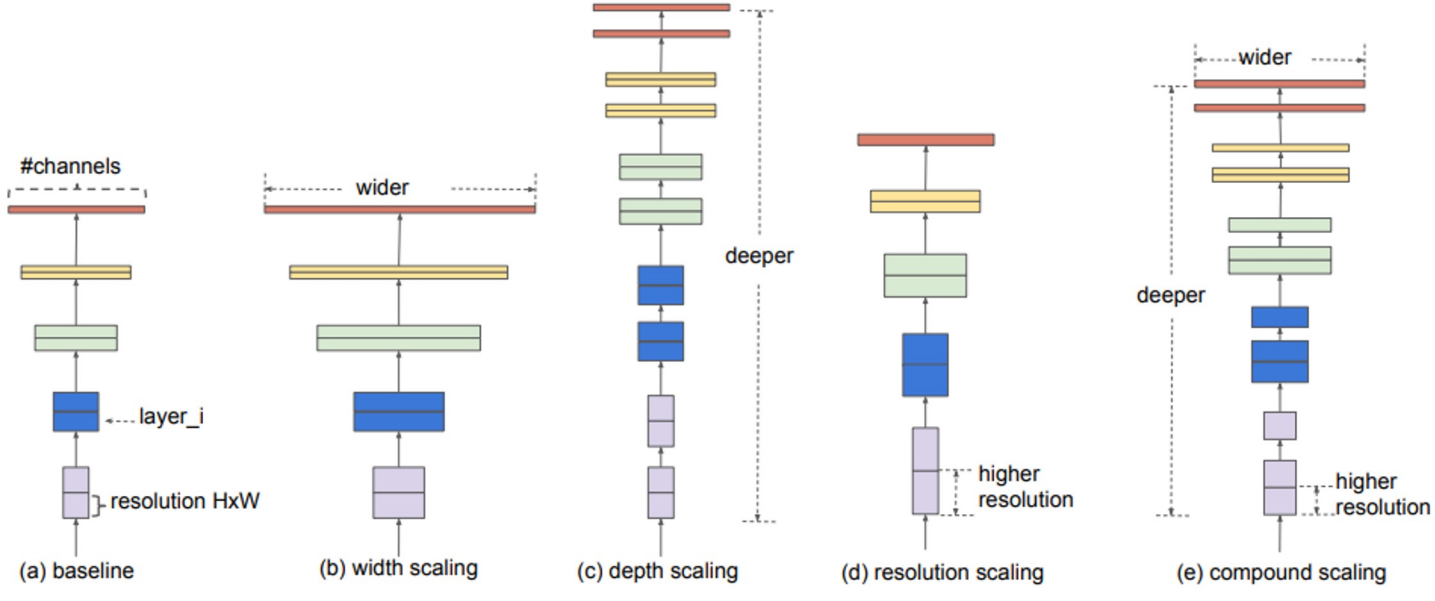
1

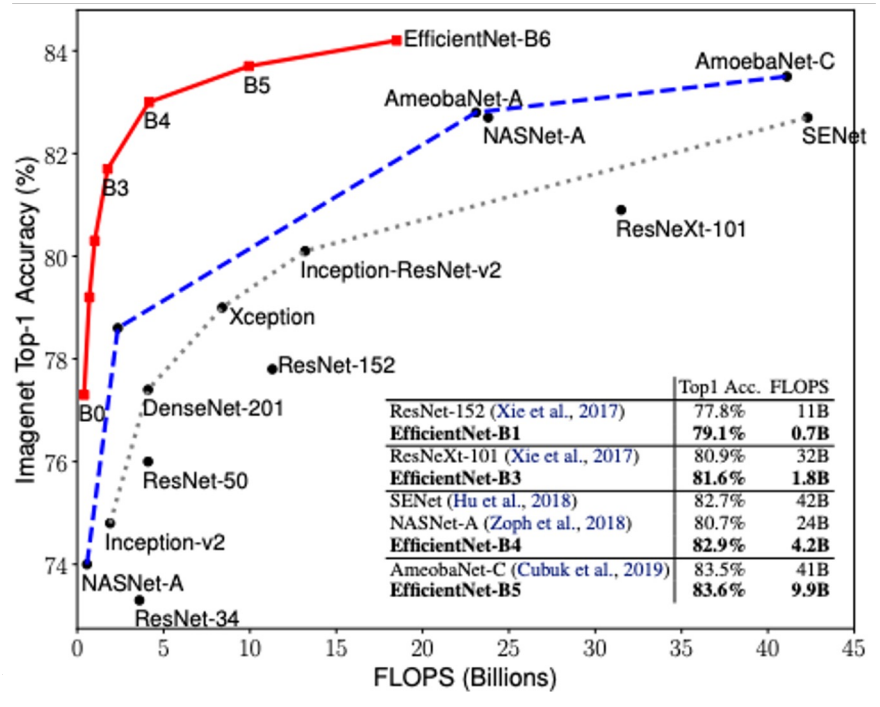
```
def _forward_impl(self, x: Tensor):
    # See note [TorchScript super()]
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)

    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)

    x = self.avgpool(x)
    x = torch.flatten(x, 1)
    x = self.fc(x)

    return x
```

1.3 Going deeper with convolutions

Beyond ResNets - EfficientNet

- Building **deep**, **wide**, and **high resolution** networks in an efficient way

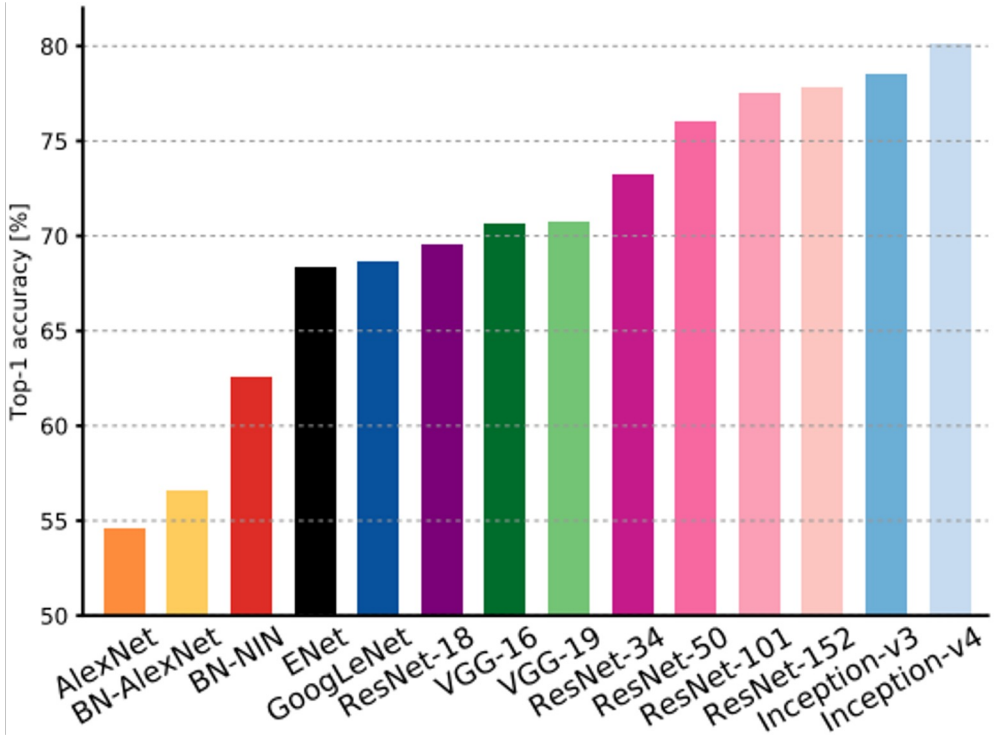


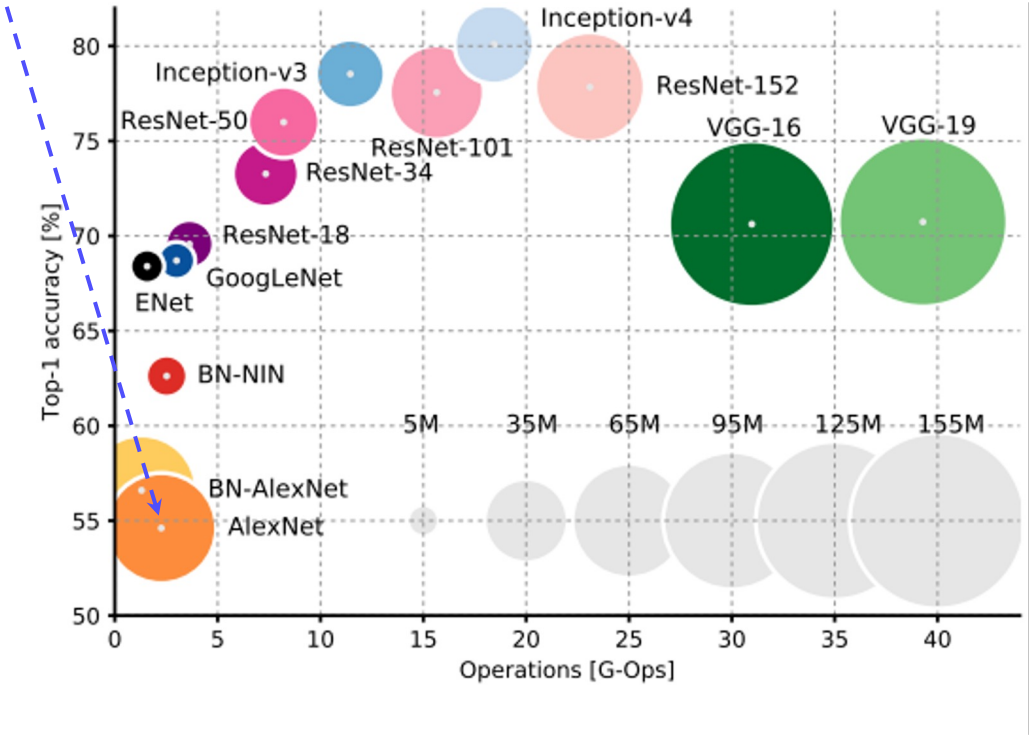


	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.1%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.6%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.9%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.6%**	**9.9B**

1.4 CNN backbones

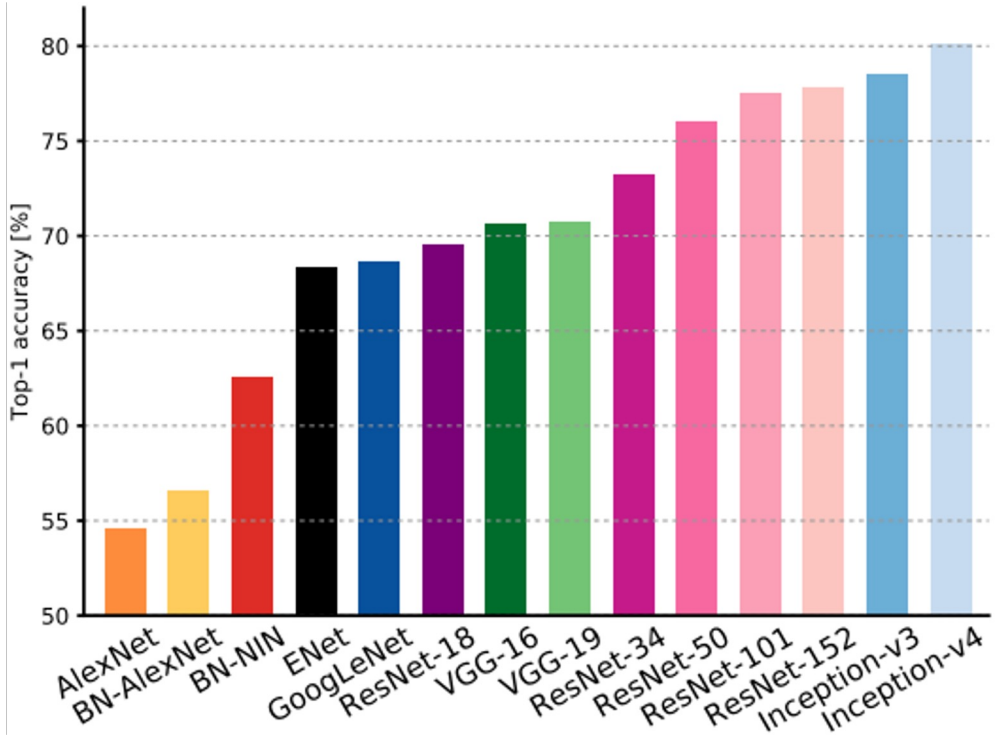
- **AlexNet**: simple CNN architecture
- Simple computation, but heavy memory size
- Low accuracy

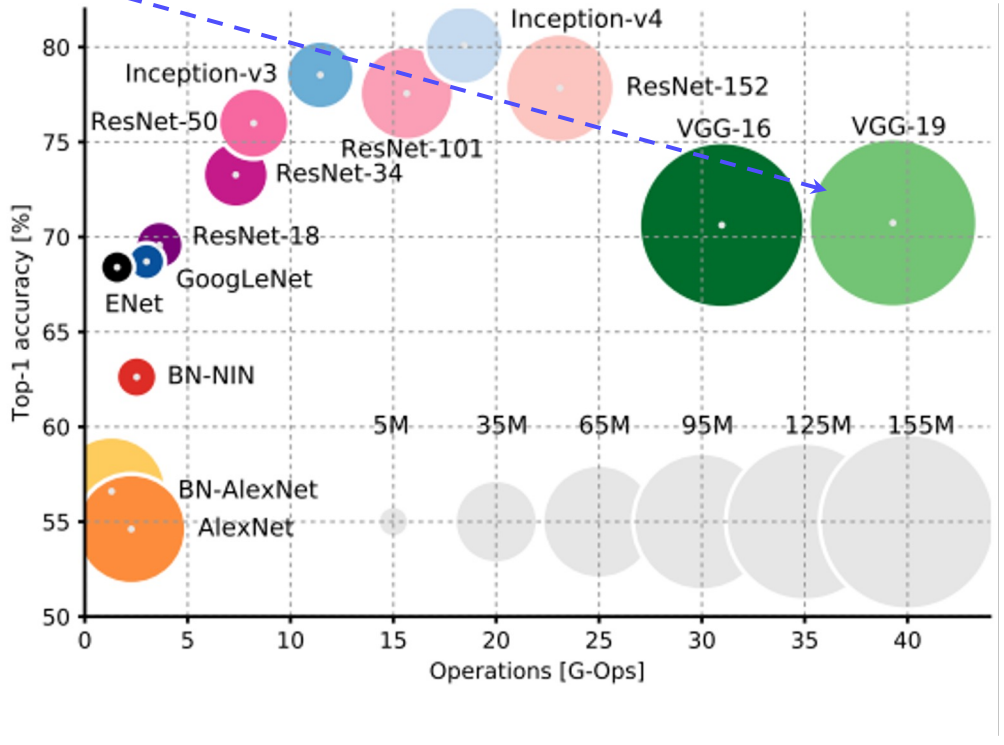




1.4 CNN backbones

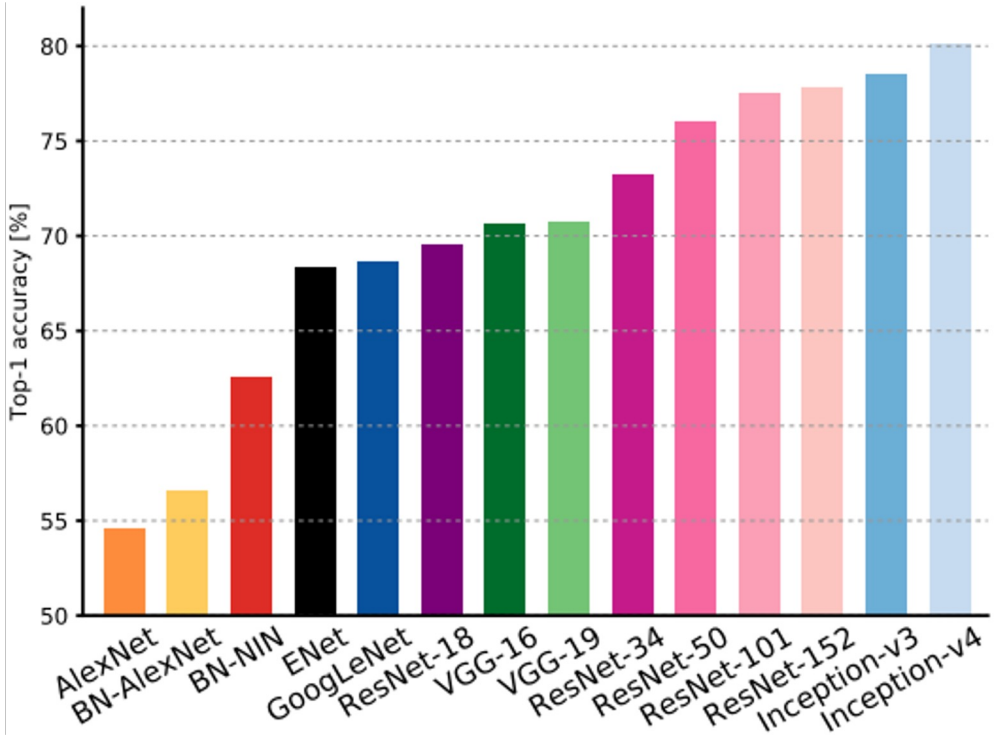
- **VGGNet**: simple with 3x3 convolutions
- Highest memory, the heaviest computation

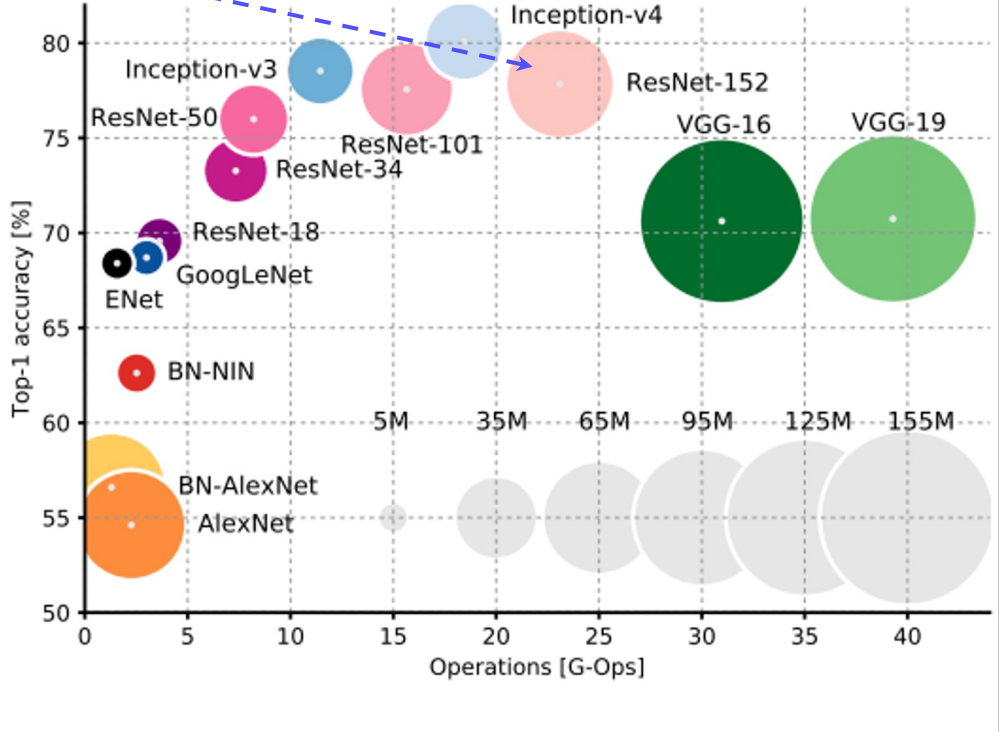




1.4 CNN backbones

- **ResNet**: deeper layers with residual blocks
- Moderate efficiency (depending on the model)

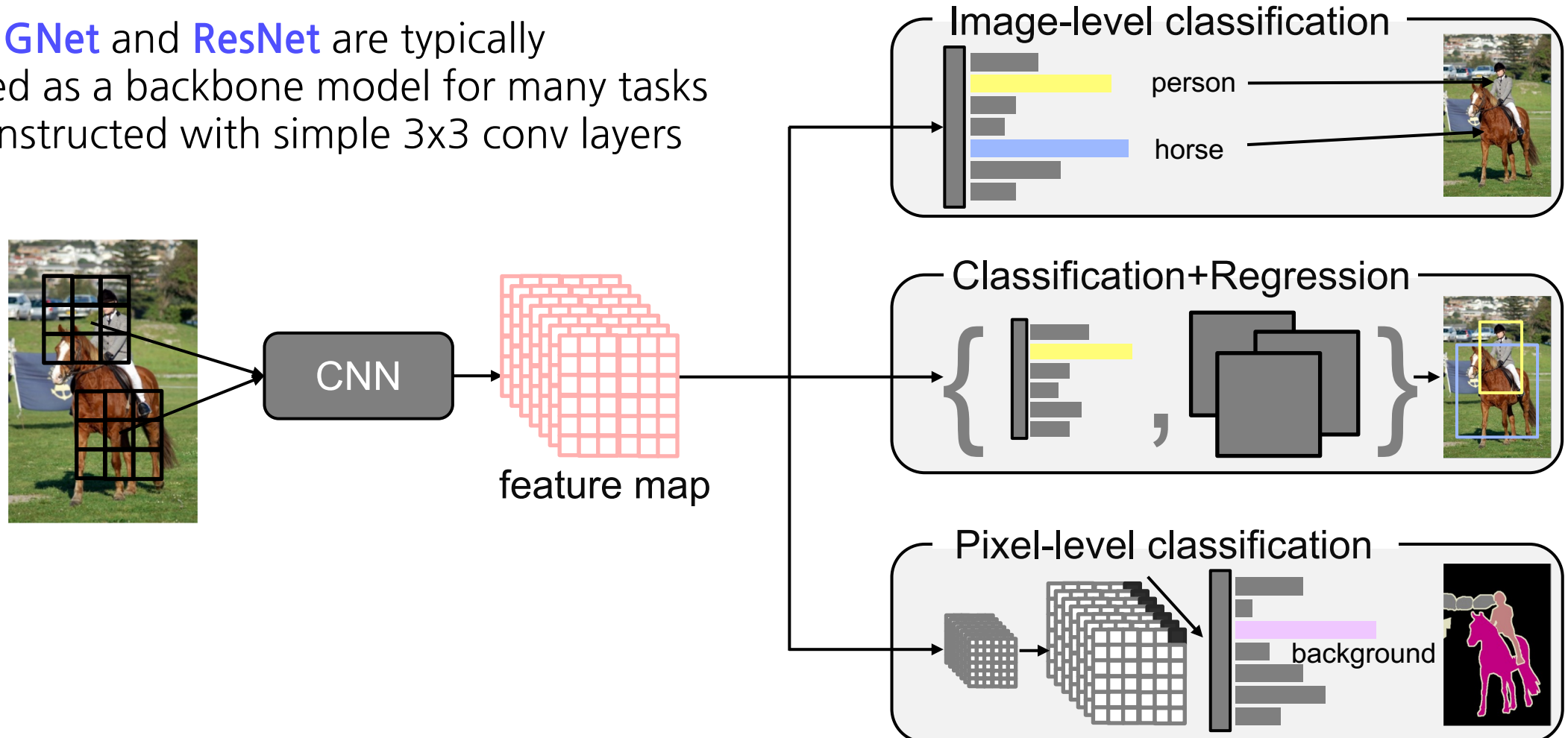




1.4 CNN backbones

Simple but powerful backbone model

- **VGGNet** and **ResNet** are typically used as a backbone model for many tasks
- Constructed with simple 3x3 conv layers



Reference

1. CNN architectures

- Lecun et al., Gradient-based Learning Applied to Document Recognition, Proceedings of the IEEE 1998
- Krizhevsky et al., ImageNet Classification with Deep Convolutional Neural Networks, NeurIPS 2012
- Simonyan and Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
- Szegedy et al., Going Deeper with Convolution, CVPR 2015
- He et al., Deep Residual Learning for Image Recognition, CVPR 2015
- Veit et al., Residual Networks Behave Like Ensembles of Relatively Shallow Networks, NIPS 2016
- Huang et al., Densely Connected Convolutional Networks, CVPR 2017
- Hu et al., Squeeze-and-Excitation Networks, CVPR 2018
- Tan and Le, EfficientNet: Rethinking Model Scalinng for Convolutional Neural Networks, ICML 2019
- Dai et al., Deformable Convolutional Networks, ICCV 2017
- Canziani et al., An Analysis of Deep Neural Network Models for Practical Applications, CVPR 2016

Reference

2. Vision Transformers

- Jeffery L., Finding structure in time, Cognitive science, 1990
- Mike et al., Bidirectional Recurrent Neural Networks, Transactions on signal processing, 1997
- Vaswani et al., Attention is all you need, NeurIPS, 2017
- Dosovitskiy et al., An image is worth 16x16 words: Transformers for image recognition at scale, ICLR, 2021
- Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL, 2019
- Kaplan et al. "Scaling laws for neural language models." arXiv preprint arXiv 2020
- Zhai et al. "Scaling vision transformers." CVPR 2022
- Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows." ICCV 2021
- He et al. "Masked autoencoders are scalable vision learners." CVPR 2022
- Caron et al. "Emerging properties in self-supervised vision transformers." ICCV 2021

Thank you

Prof. Jeon, Sangryul

Computer Vision Lab.

Pusan National University, Korea

Tel: +82-51-510-2423

Web: http://sr-jeon.github.io/

E-mail: srjeonn@pusan.ac.kr