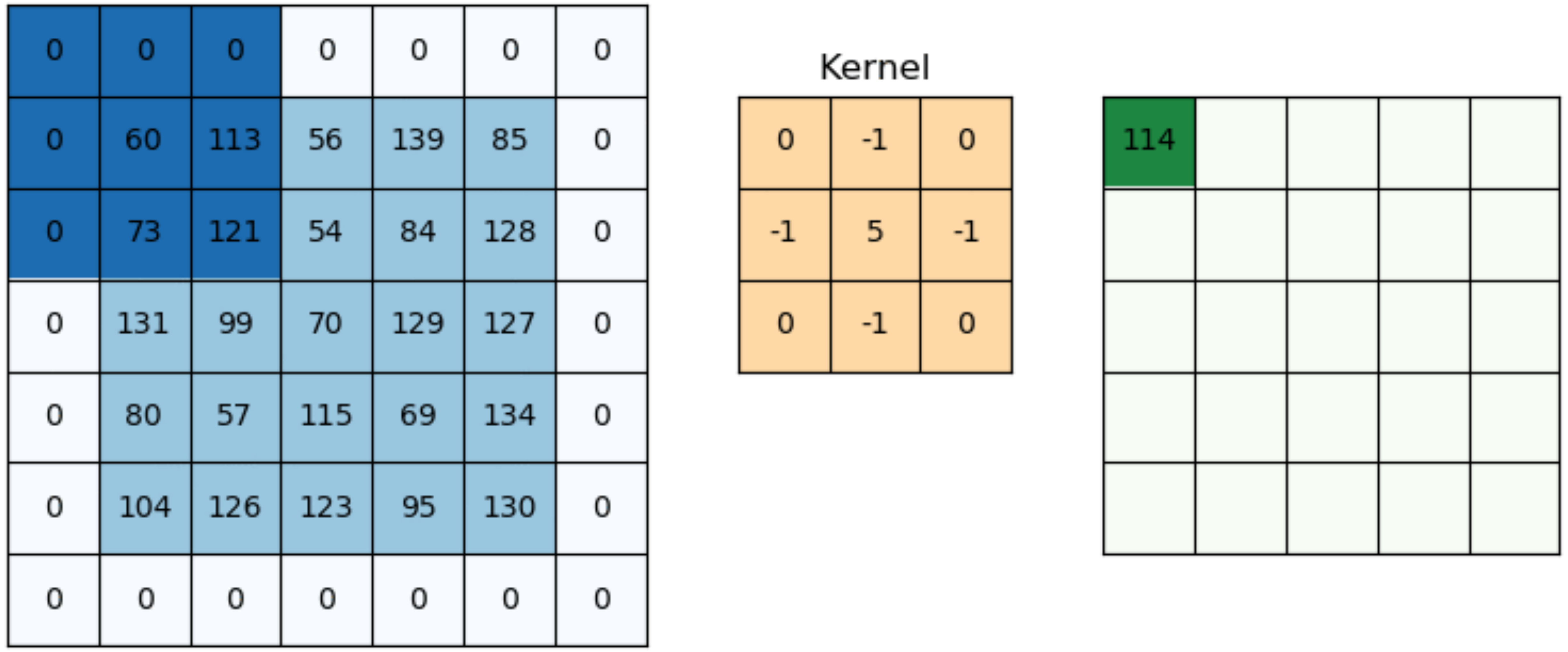
18. Popular ConvNets

EEC454 Introduction to Machine Learning Systems

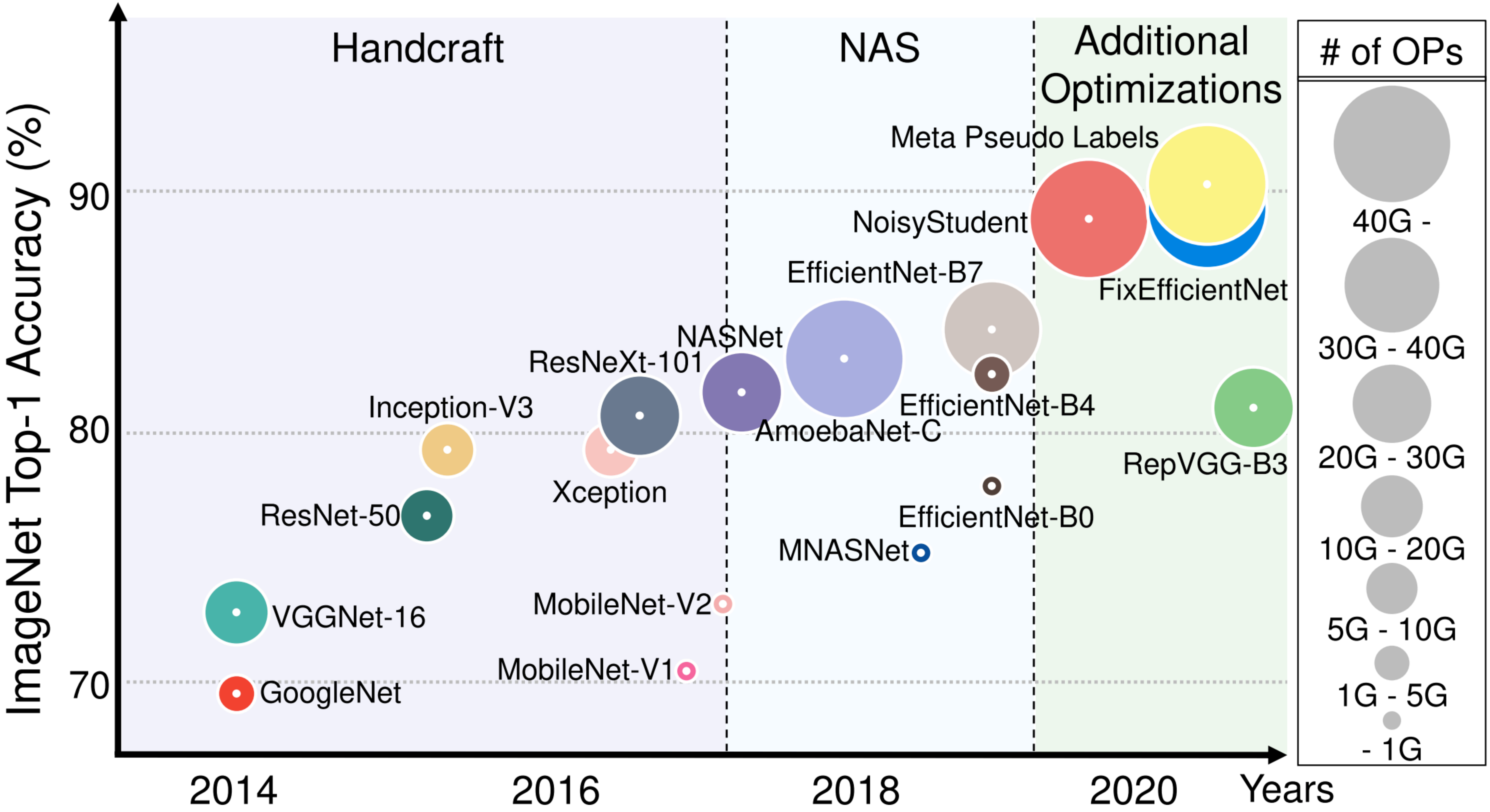
2023 Fall, Jaeho Lee

Recap

- **4 lectures ago.** Introduced the convolution layer.
  - Parameter-efficient
  - Translation-equivariant (exploits locality)
  - Can handle various resolutions



Lee et al., "MVP: An Efficient CNN Accelerator with Matrix, Vector, and Processing-Near-Memory Units,"ACM DAES 2022



Lee et al., “MVP: An Efficient CNN Accelerator with Matrix, Vector, and Processing-Near-Memory Units,”ACM DAES 2022

Today

- Popular ConvNet backbones & Important ideas
    - Basic Models (LeNet / AlexNet)
    - Deeper Models (VGG / GoogLeNets / ResNets)
    - Tiny Models (MobileNets / MCUNets)
    - Efficiently Scalable Models (EfficientNet / NFNets)

Basic Models

LeCun et al., “Gradient-based learning applied to document recognition,” ACM DAES 2022

LeNet-5 (1998)

- First practically useful ConvNets
    - Convolutions, then fully-connected layers
    - Pooling after convolution

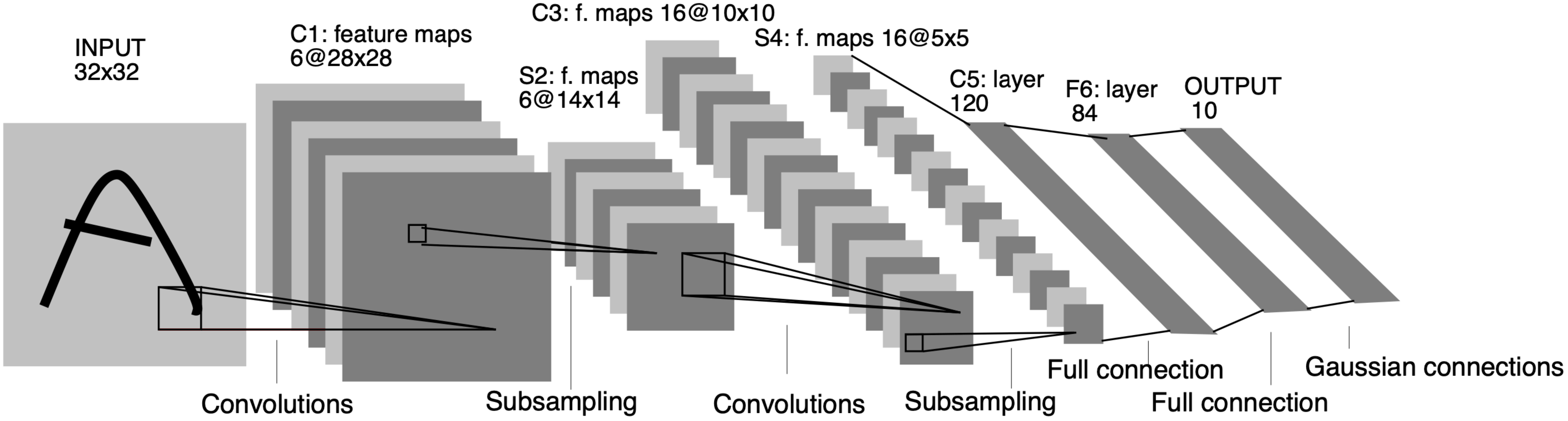


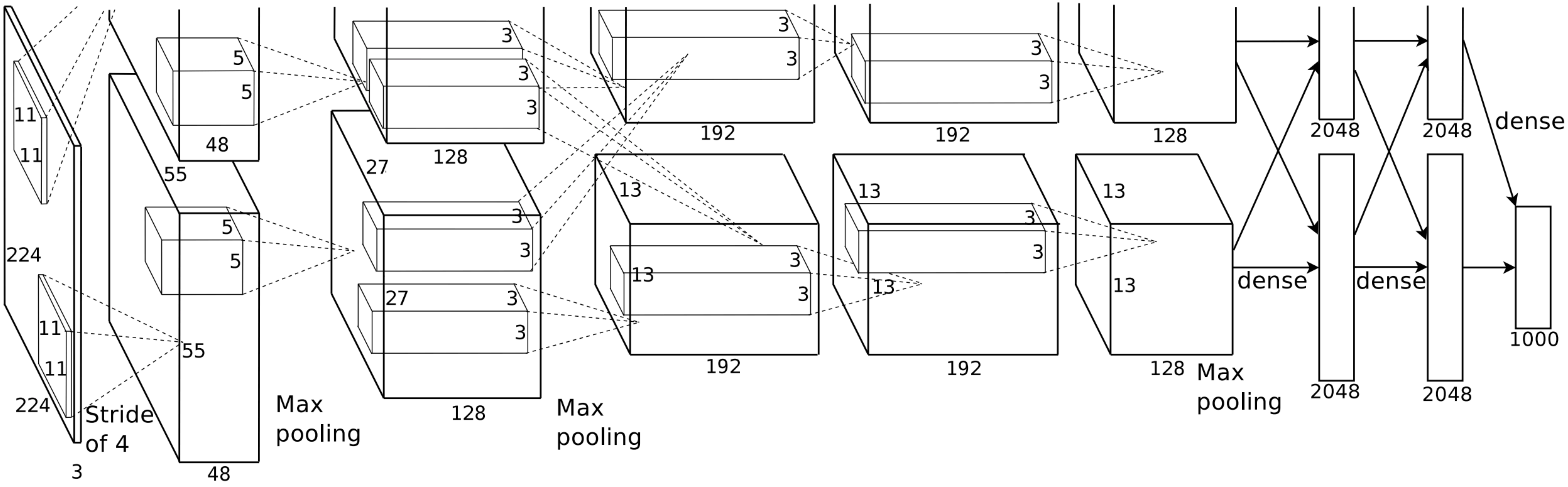
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.


4627
4 6 2 7

Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks," NeurIPS 2012

AlexNet (2012)

- Bigger and deeper version of LeNet
  - 7 hidden layers, 605k neurons, 60 million parameters

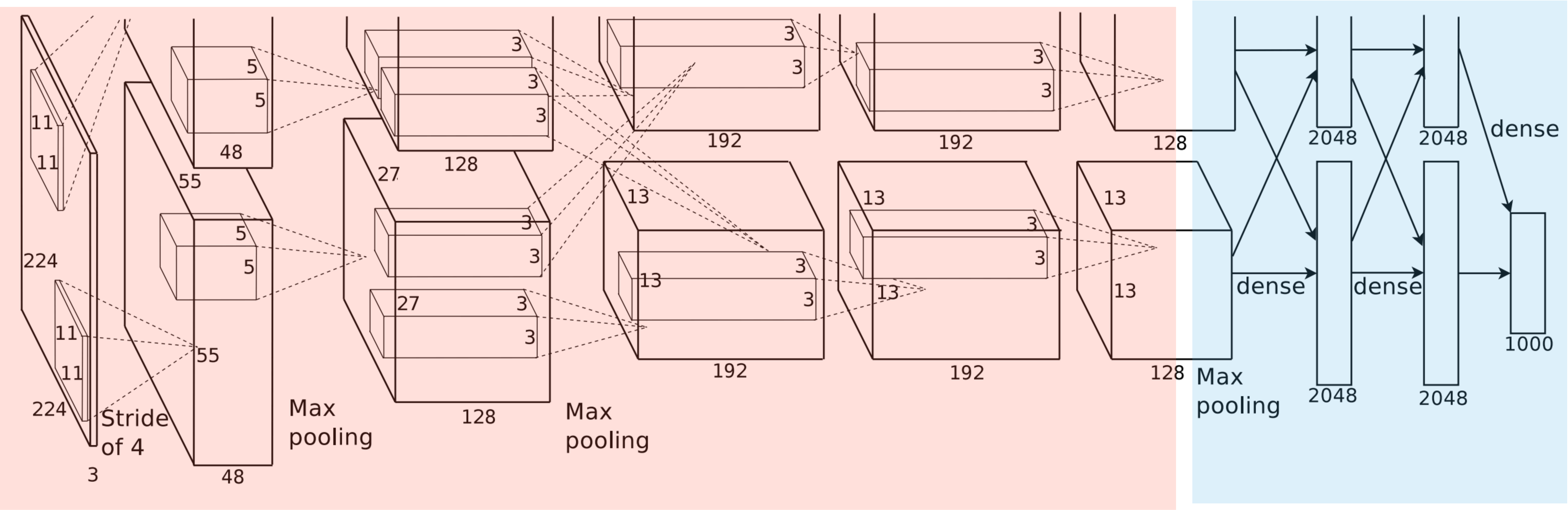


Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks,” NeurIPS 2012

AlexNet (2012)

- Bigger and deeper version of LeNet
  - 7 hidden layers, 605k neurons, 60 million parameters

Conv-Pool-Norm.-Conv-Pool-Norm.-Conv-Conv-Conv-Pool-FC-FC-FC



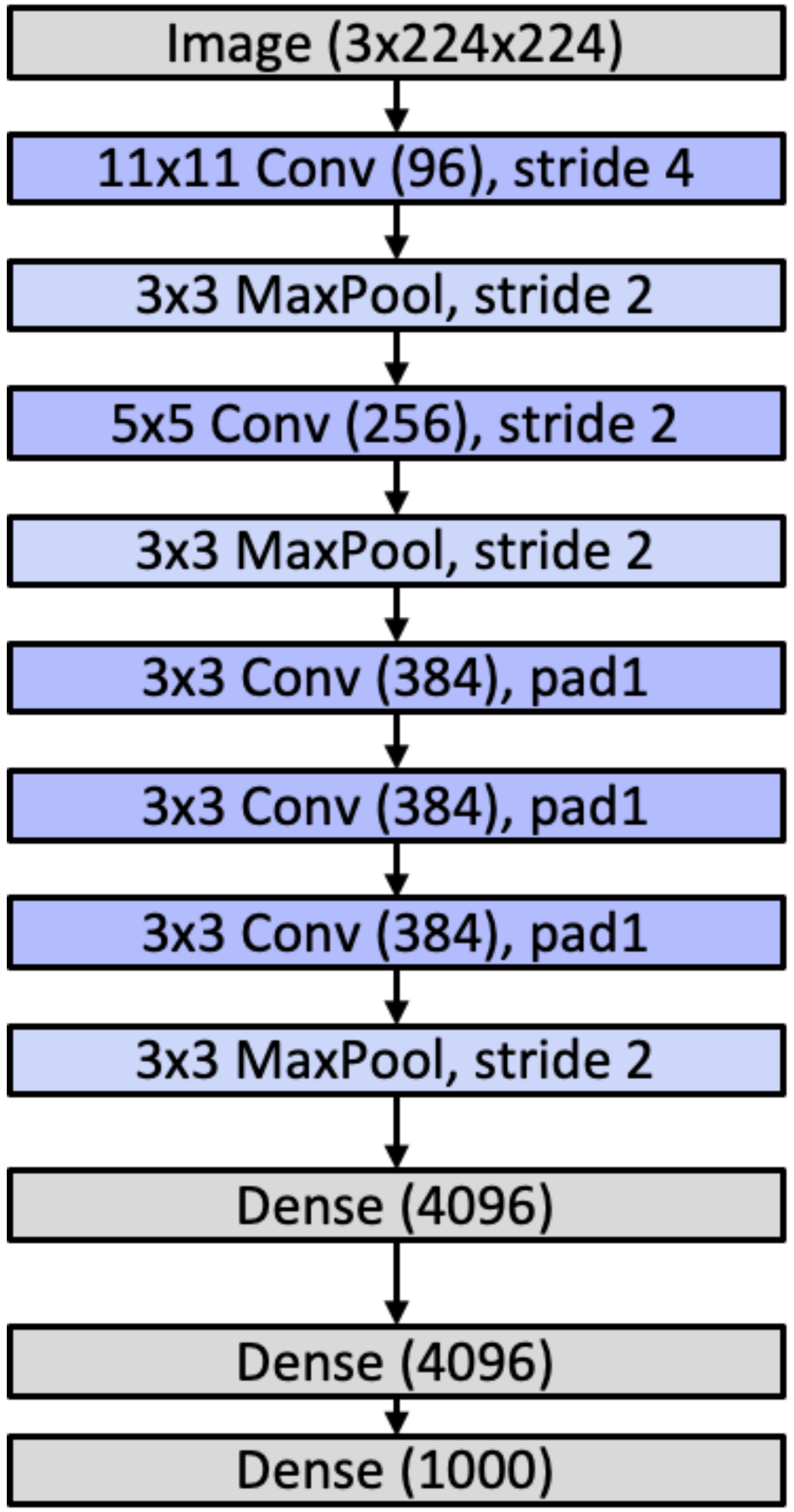
Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks,” NeurIPS 2012

AlexNet (2012)

- Bigger and deeper version of LeNet
  - 7 hidden layers, 605k neurons, 60 million parameters
- Q. Why was bigger & deeper possible?
  - **Dataset.** ImageNet Large-Scale Visual Recognition Challenge
  - **Optimization.** Better activation (ReLU)
  - **Generalization.** Better regularization (Dropout)
  - **Computation.** Distributed GPU training (two GTX 580)

… let the scale race begin!

AlexNet (2012)




```
nn.Conv2d(3, 96, kernel_size=11, stride=4, padding=2),
nn.ReLU(inplace=True),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(96, 256, kernel_size=5, padding=2),
nn.ReLU(inplace=True),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(256, 384, kernel_size=3, padding=1),
nn.ReLU(inplace=True),
nn.Conv2d(384, 384, kernel_size=3, padding=1),
nn.ReLU(inplace=True),
nn.Conv2d(384, 256, kernel_size=3, padding=1),
nn.ReLU(inplace=True),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Dropout(),
nn.Linear(256 * 6 * 6, 4096),
nn.ReLU(inplace=True),
nn.Dropout(),
nn.Linear(4096, 4096),
nn.ReLU(inplace=True),
nn.Linear(4096, 1000),
```

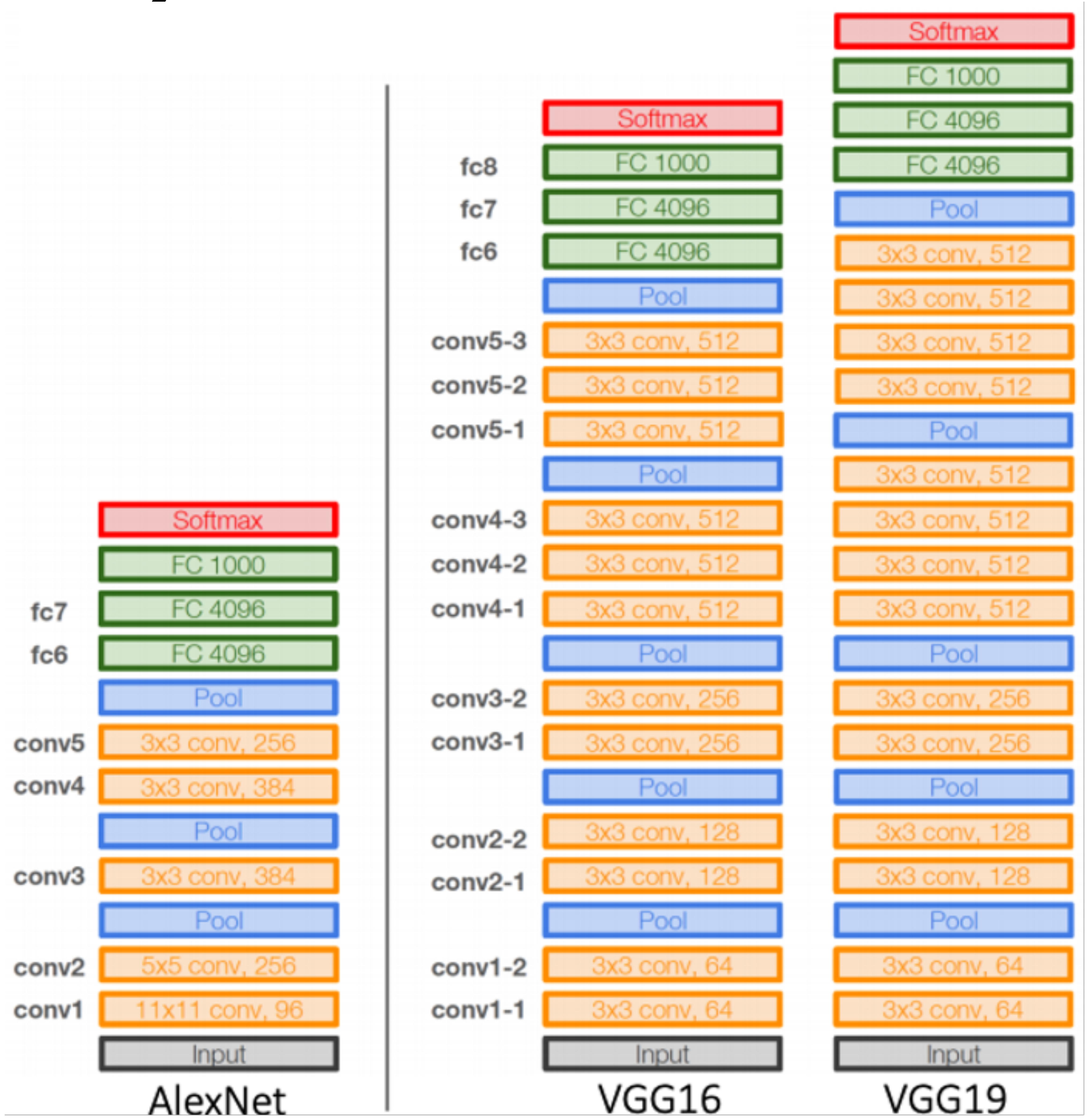
```
nn.AdaptiveAvgPool2d((6, 6))
torch.flatten(x, 1)
```

Deeper Models

Simonyan and Zisserman, “Very deep convolutional networks for large-scale image recognition,” ICLR 2015

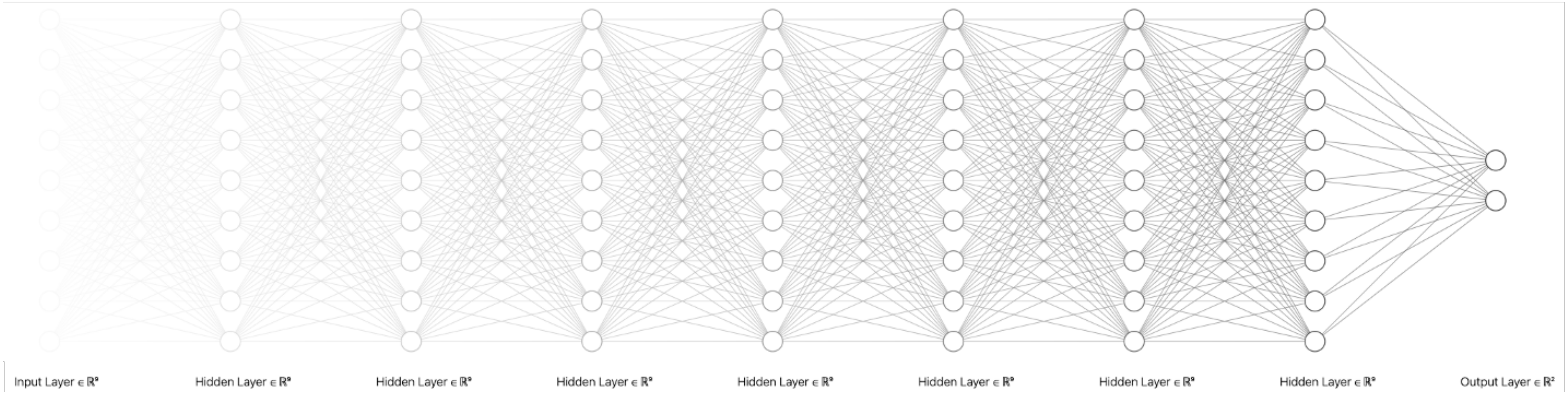
VGG (2014)

- **Deeper networks**
  - up to 19 layers
- **Simpler architecture**
  - Only 3x3 convolution
    - *vs 5x5?* less parameters but deeper
  - Only 2x2 pooling
  - No “local response normalization”



Stacking Deeper

- **Key obstacles.**
  - Gradient vanishing / exploding (no batchnorm back then)
  - Too many parameters



Stacking Deeper

- **Key obstacles.**
  - Gradient vanishing / exploding (no batchnorm back then)
  - Too many parameters

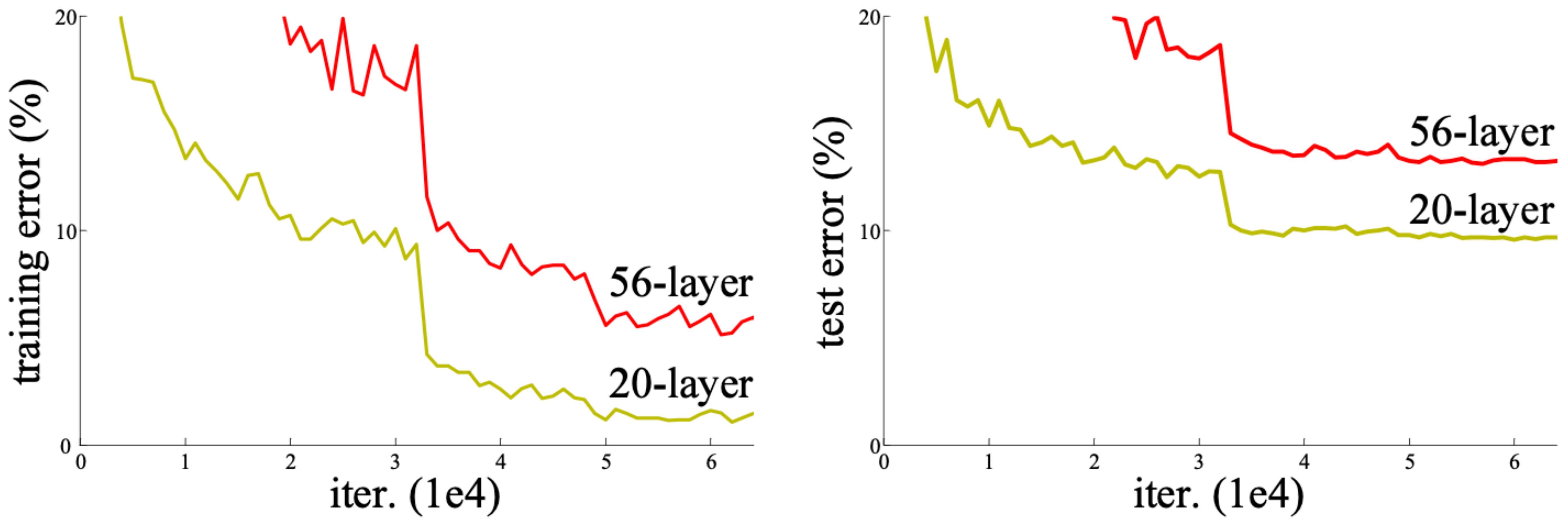
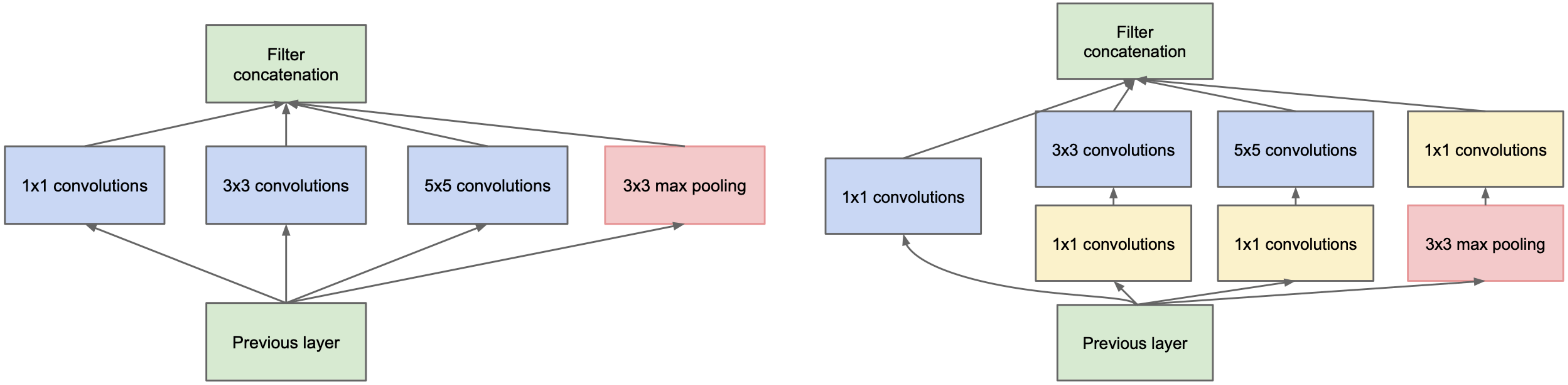


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Szegedy et al., “Going deeper with convolutions,” arXiv 2014

GoogLeNet (2014)

- Two notable differences
  - **Inception module.** Works as a ***bottleneck*** to reduce #channels.



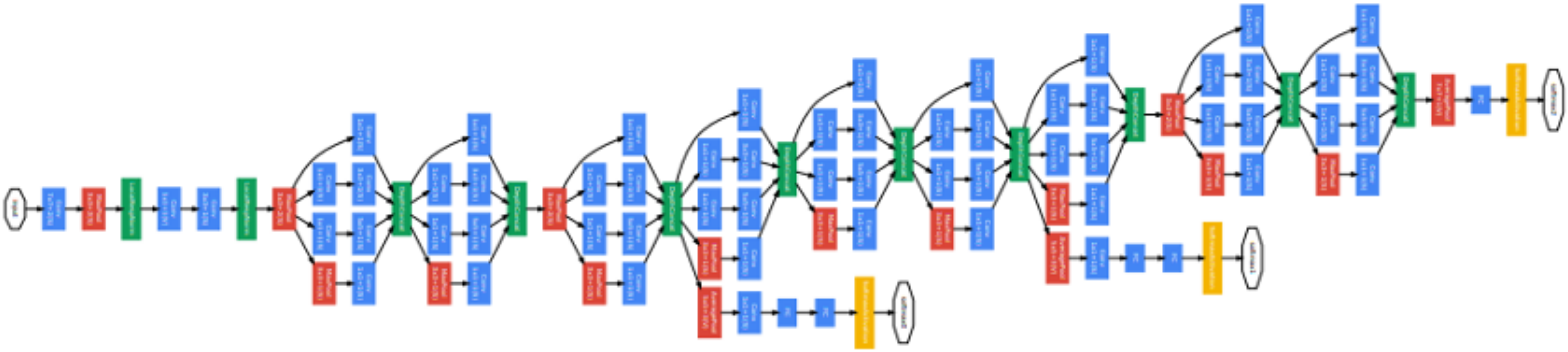
(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

Szegedy et al., “Going deeper with convolutions,” arXiv 2014

GoogLeNet (2014)

- Two notable differences
  - **Inception module.**
  - **Auxiliary classifier.** Resolves vanishing gradient



Szegedy et al., “Going deeper with convolutions,” arXiv 2014

GoogLeNet (2014)

- Two notable differences
  - **Inception module.**
  - **Auxiliary classifier.** Resolves vanishing gradient

  Note. Only one FC layer (reduces #params)

ResNet (2016)

- A more elegant solution to the vanishing gradient.

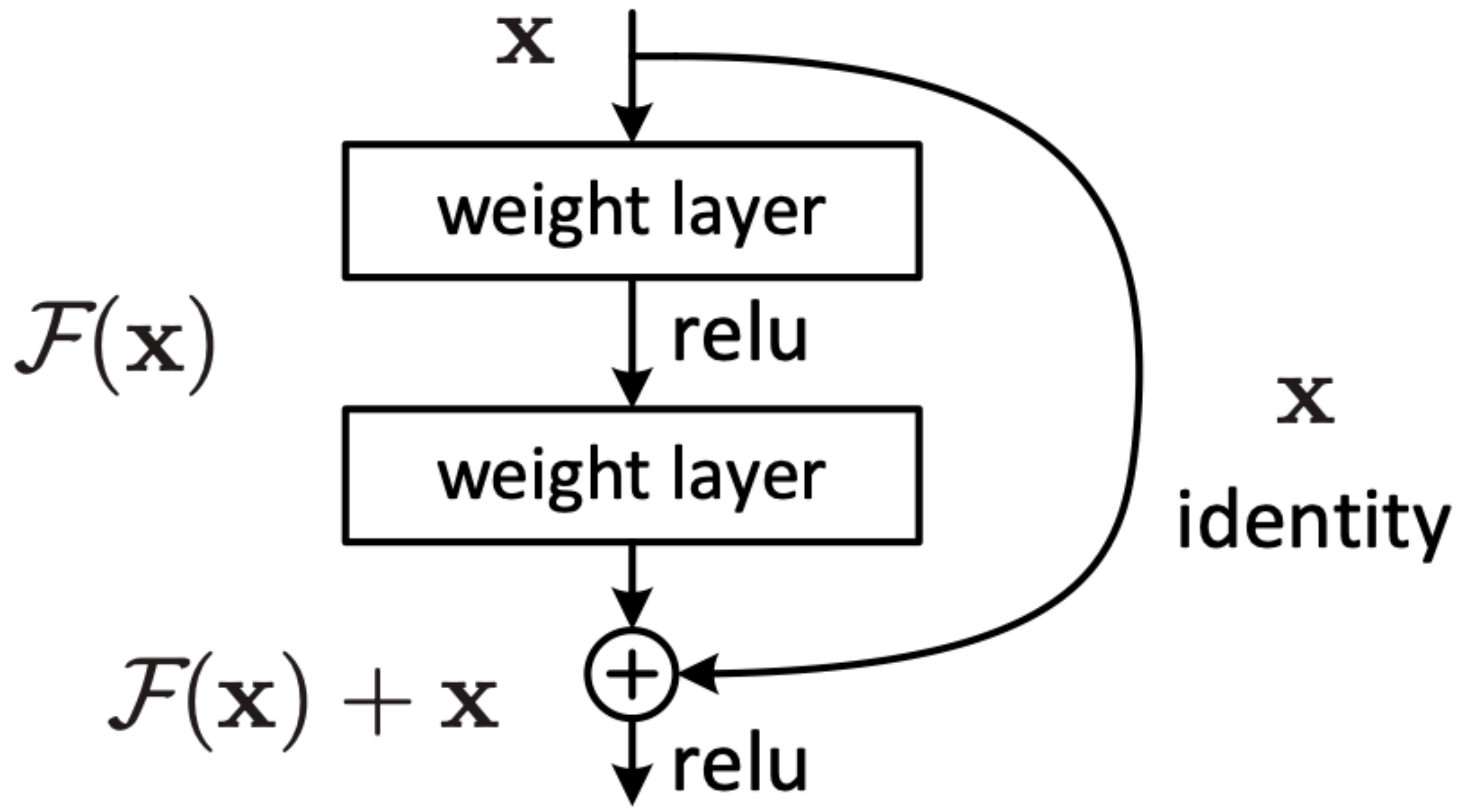
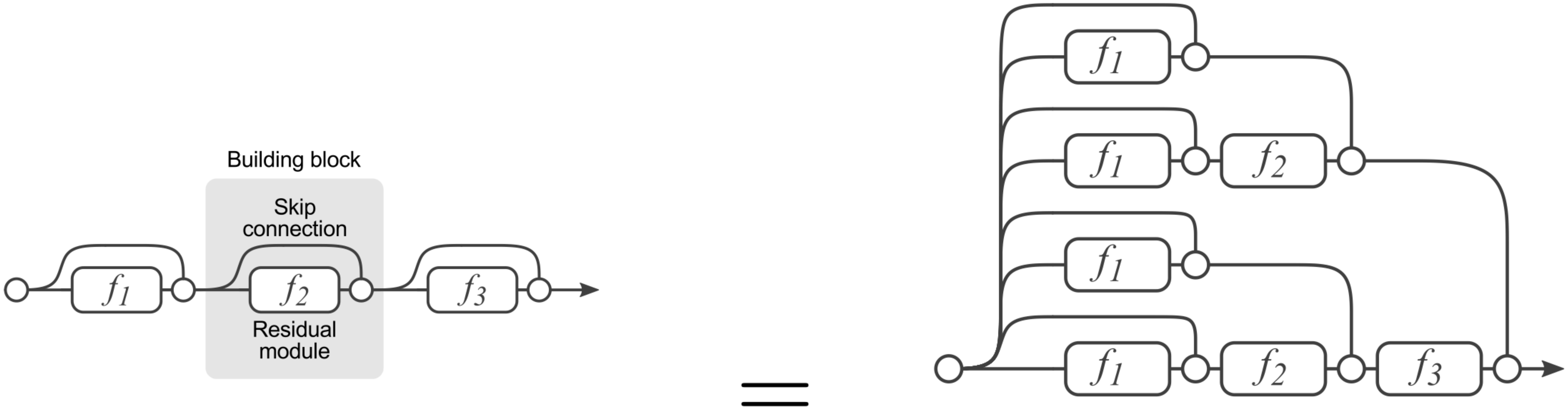


Figure 2. Residual learning: a building block.

Veit et al., “Residual Networks Behave Like Ensembles of Relatively Shallow Networks,” NeurIPS 2016

ResNet (2016)

- A more elegant solution to the vanishing gradient.
  - The gradients come from shorter paths



(a) Conventional 3-block residual network

(b) Unraveled view of (a)

ResNet (2016)

- “Bottleneck” blocks
  - Known to accelerate the training.

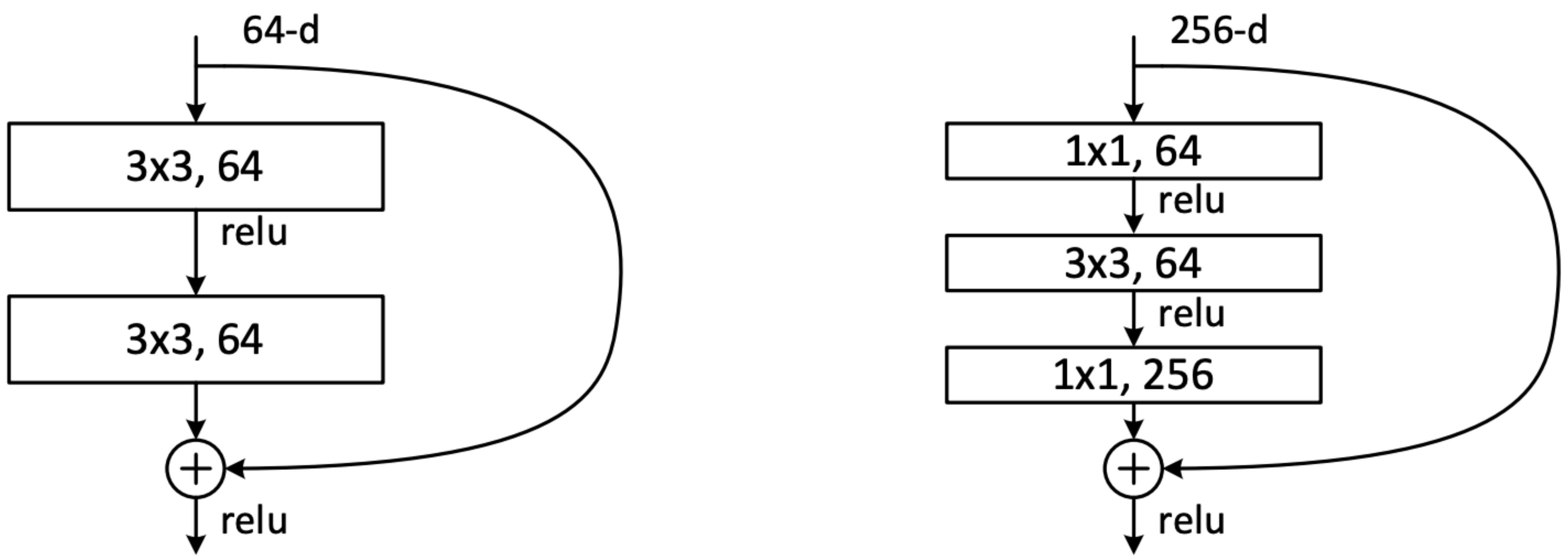
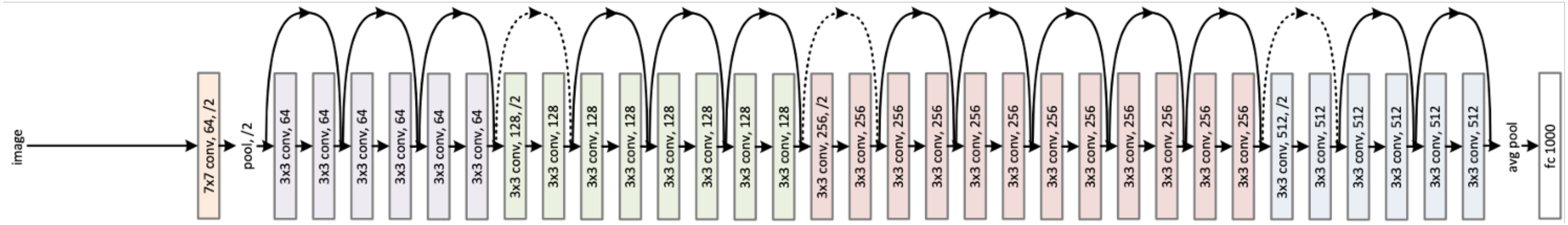


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

He et al., “Deep residual learning for image recognition,” CVPR 2016

ResNet (2016)

- Up to over 150 layers in total
  - A better initialization (He init.)
  - Batch normalization after every convolution
  - Dotted line: doubles #channel & downsample by 2



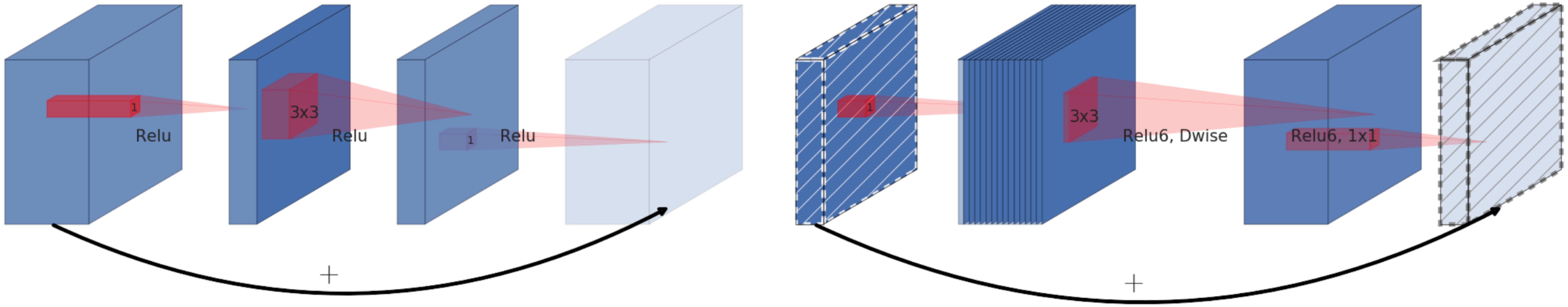
Tiny Models

MobileNets (2017, 2018,2019)

- Less number of parameters for inference on mobile devices
- **Key component.**
    - Depthwise convolution (v1)
        - Used for Inverted Residual (v2)

v3 optimized architectures with NAS

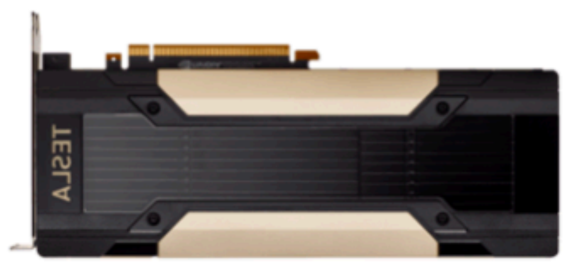
(a) Residual block (b) Inverted residual block



Lin et al., “MCUNet: Tiny Deep Learning on IoT Devices,” NeurIPS 2020

MCUNets (2020, 2021,2022)

- Less memory requirement for training on microcontrollers
- **Tools**
  - Algorithm/system co-design
  - Neural Architecture Search
  - Better quantization-awareness



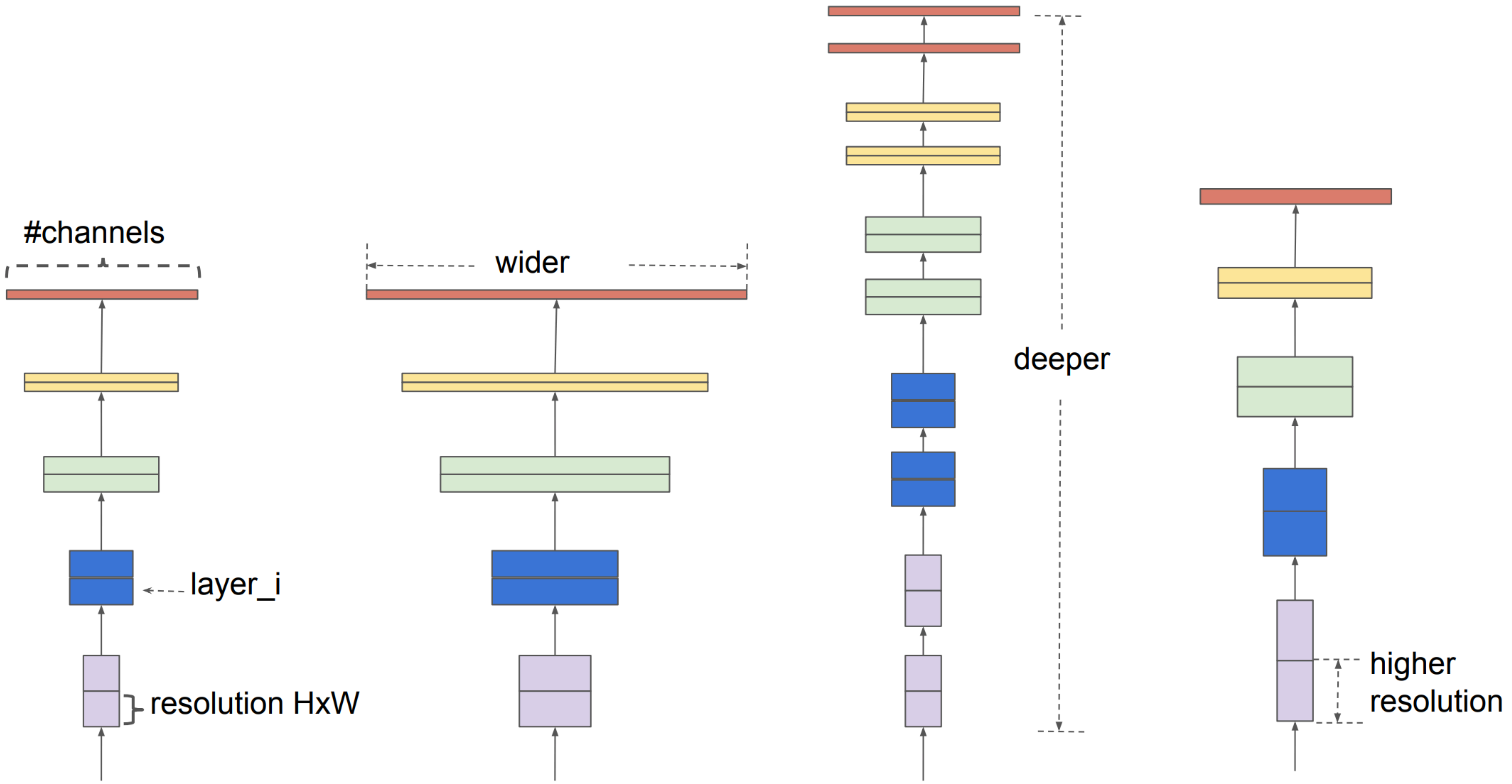
	ResNet **Cloud AI**	MobileNet **Mobile AI**		MCUNet **Tiny AI**
Memory (Activation)	16GB	4GB	**13,000x smaller** →	320kB
Storage (Weights)	~TB/PB	256GB		1MB

Efficiently Scalable Models

Tan and Le, “Rethinking Model Scaling for Convolutional Neural Networks,” ICML 2019

EfficientNets (2019,2021)

- **Q.** If we have more budget, how should we increase #param?
  - Increase depth
  - Increase width
  - Less downsampling



(a) baseline (b) width scaling (c) depth scaling (d) resolution scaling

EfficientNets (2019,2021)

- **A.** All of them, by some ratio (discovered empirically)   v2: + NAS

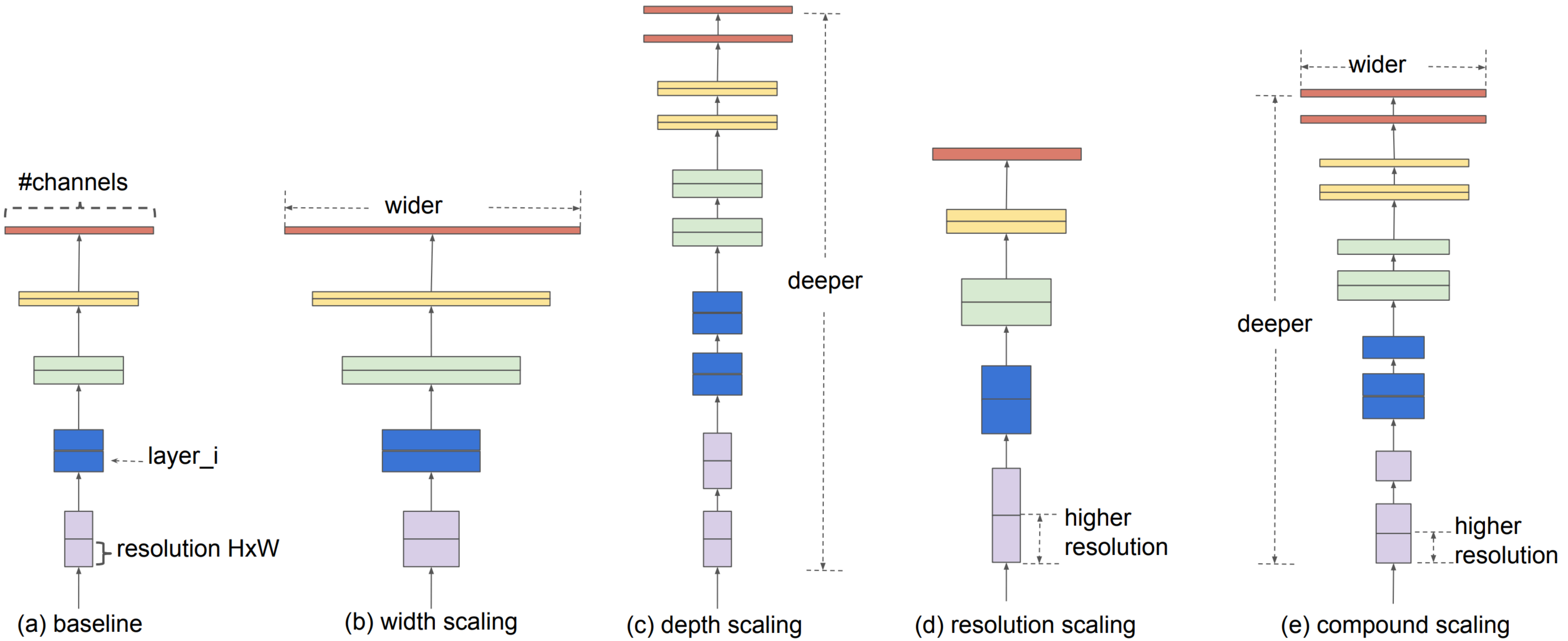
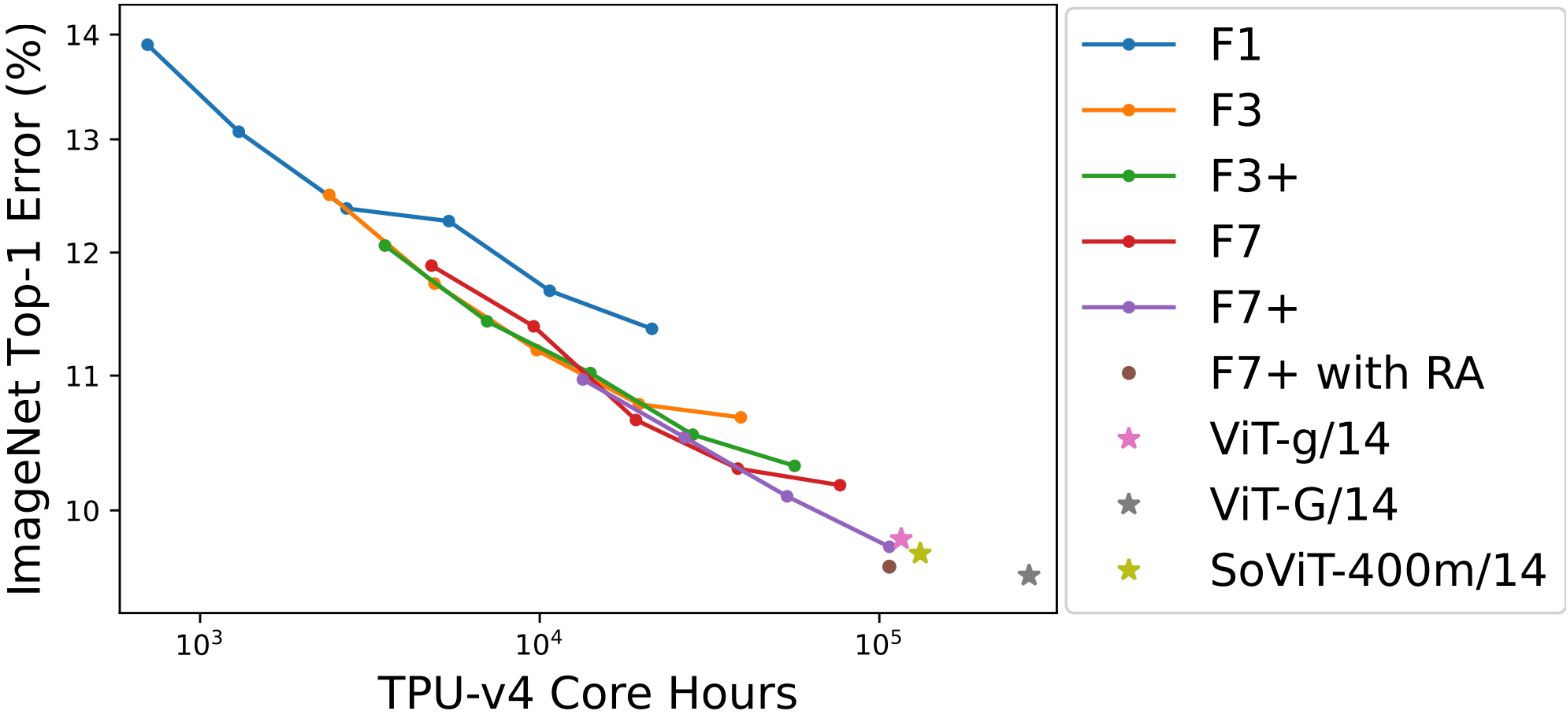


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Brock et al., "High-performance large-scale image recognition without normalization," ICML 2021

NFNets (2021,2023)

- As of 2023, it seems like ***removing batch norms*** is a key to build very big convolutional networks
  - Before this, people thought large convnets cannot work as good as transformers…



Cheers

- *Next up.* Generative models