Convolutional Neural Networks

ITM528 Deep Learning

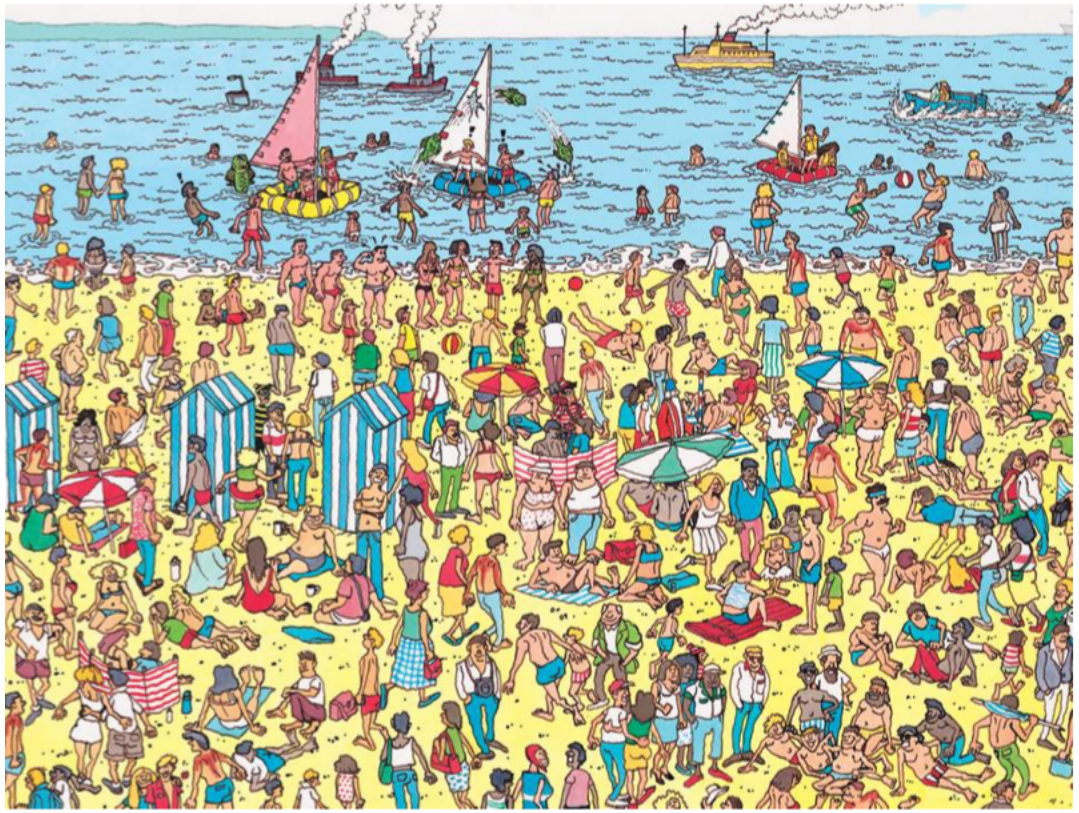
Taemoon Jeong

Convolution Operation

Fully Connected to Convolutions

Why do convolutional neural networks (CNNs) more appropriate than MLPs in image domains?

Invariance



- Imagine that we want to detect an object in an image (Wheres Waldo).
- It seems reasonable that whatever method we use, the precise location of the object in the image should not be overly concerned.

Invariance

- Despite Waldo's characteristic outfit, what Waldo looks like does not depend upon where is located.
- We could sweep the image with a Waldo detector that could assign a score to each patch, indicating the likelihood that the patch contains Waldo.
- Convolutional neural networks (CNNs) systematize this idea of spatial invariance, exploiting it to learn useful representations with fewer parameters.

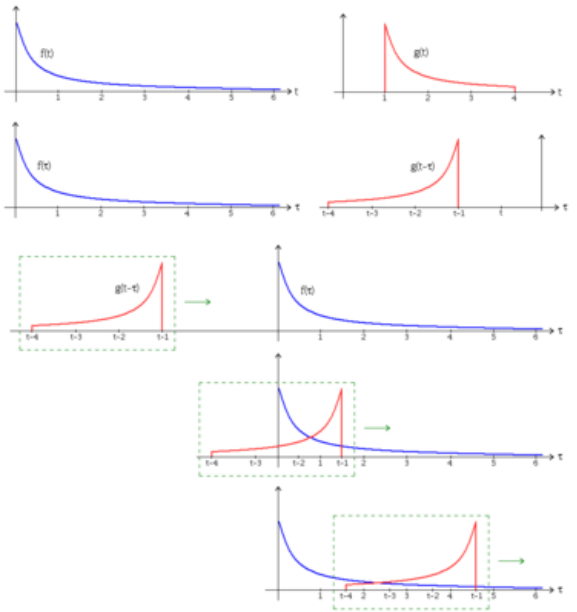
Convolutions (in signal processings)

- In mathematics, the convolution between two functions, say f and g, is defined as

$$(f * g)(\mathbf{x}) = \int f(\mathbf{z})g(\mathbf{x} - \mathbf{z})d\mathbf{z}.$$

- That is, we measure the overlap between f and g when on function if flipped and shifted by $\mathbf{x}$ (i.e., $g(\mathbf{x} - \mathbf{z})$).
- When we have discrete objects in two-dimensional spaces:

$$(f * g)(i, j) = \sum_a \sum_b f(a, b)g(i - a, j - b).$$



Convolutions (in CNNs)

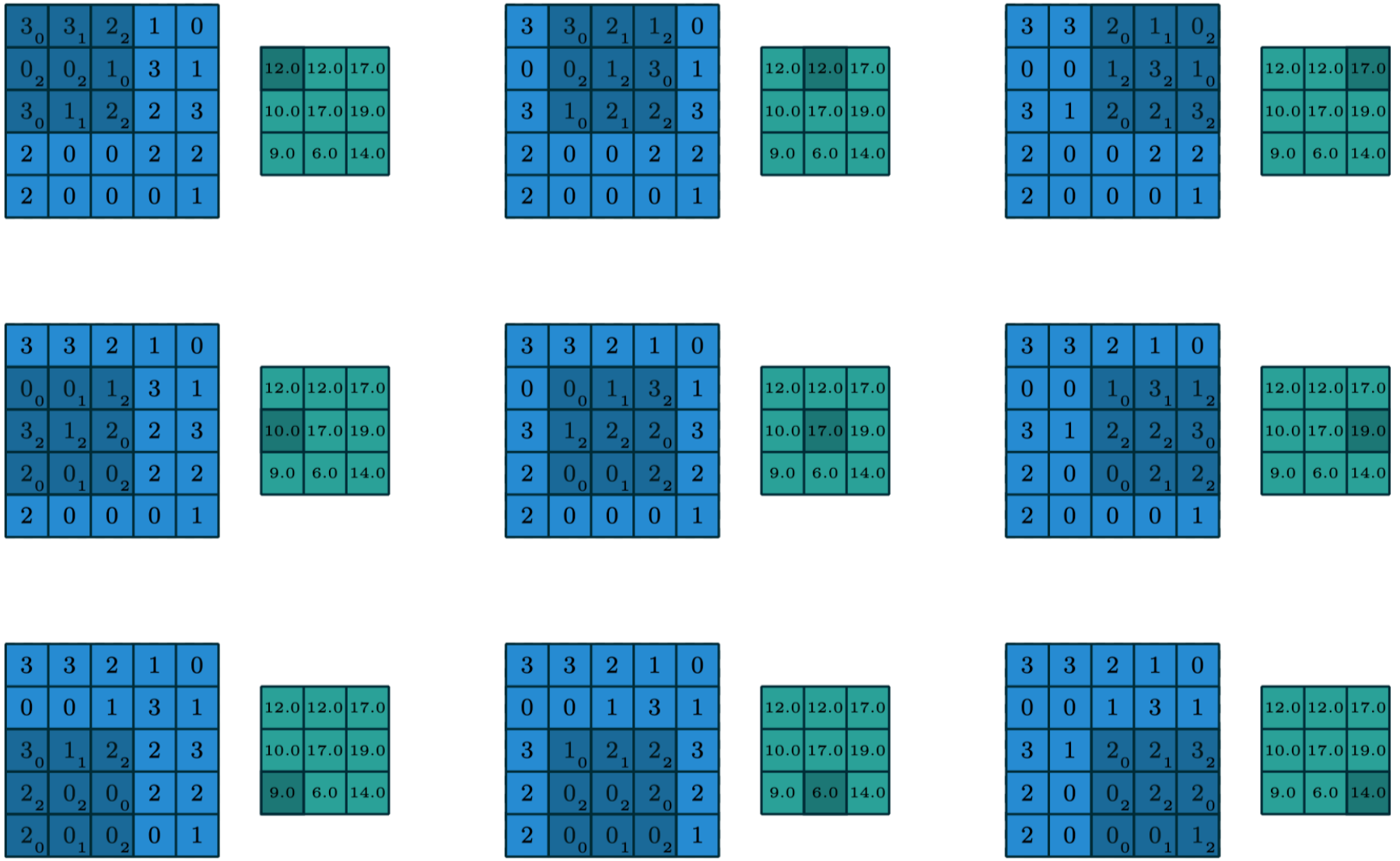
- To exploit the translation invariance, an output of a convolutional layer is computed as:

$$[\mathbf{H}]_{i,j} = u + \sum_{a=-\triangle}^{\triangle} \sum_{b=-\triangle}^{\triangle} [\mathbf{V}]_{a,b} [\mathbf{X}]_{i+a,j+b}$$

  where $\mathbf{V}$ is referred to as a convolution kernel or a filter.

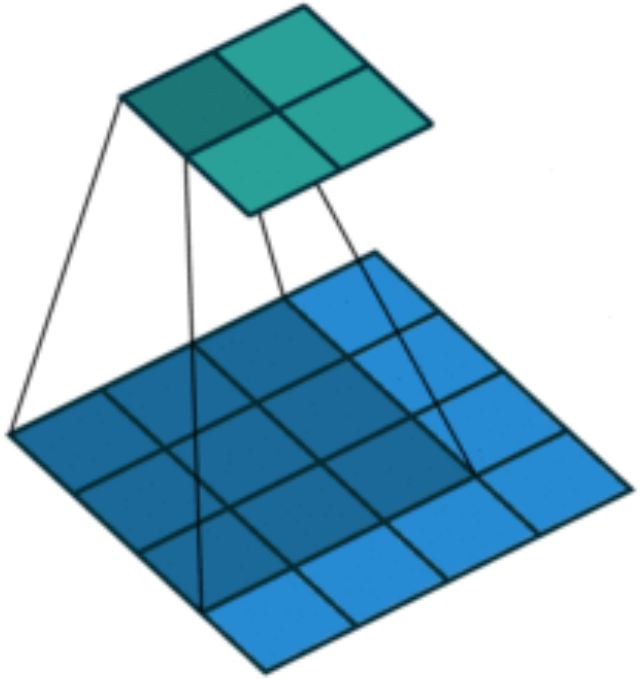
- The number of parameters for the layer is $(2\triangle + 1)^2$.
- This operation is more properly described as a cross-correlation.

Convolution

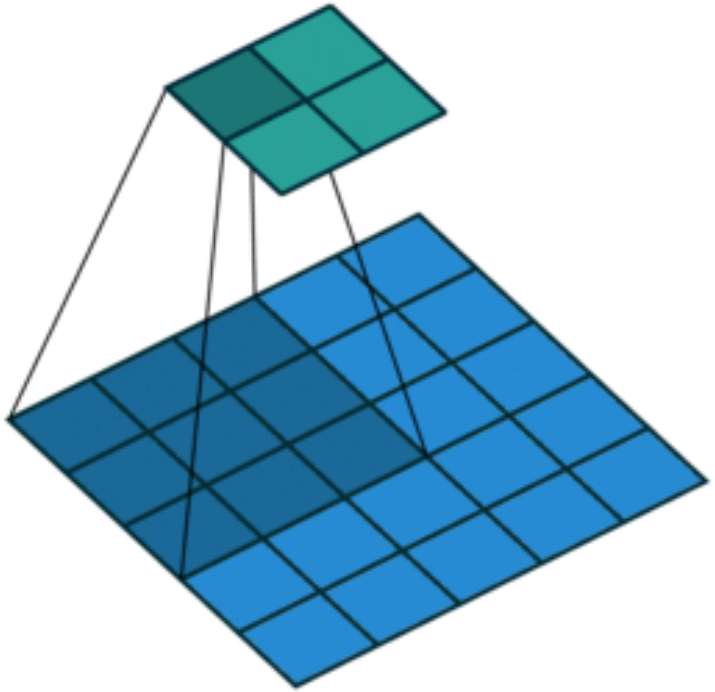


"A guide to convolution arithmetic for deep learning," 2018

Convolution (Padding and Strides)



padding=0 and stride=1

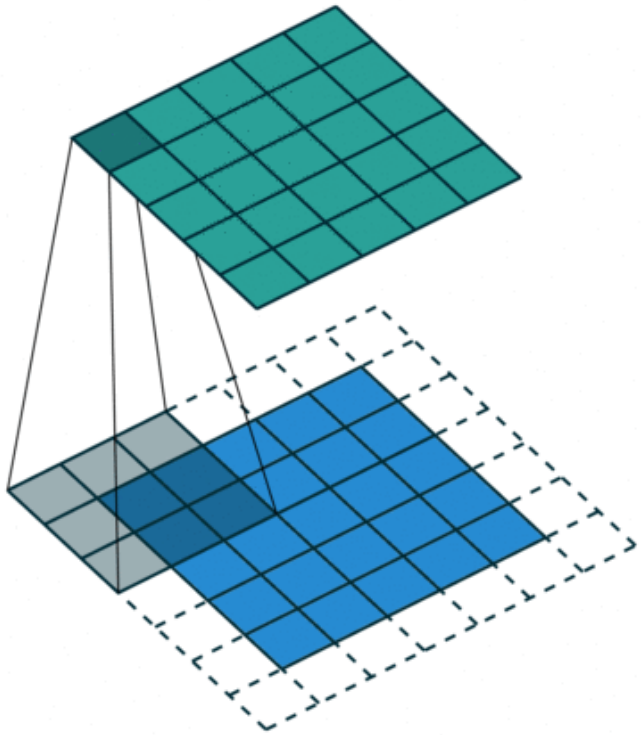


padding=0 and stride=2

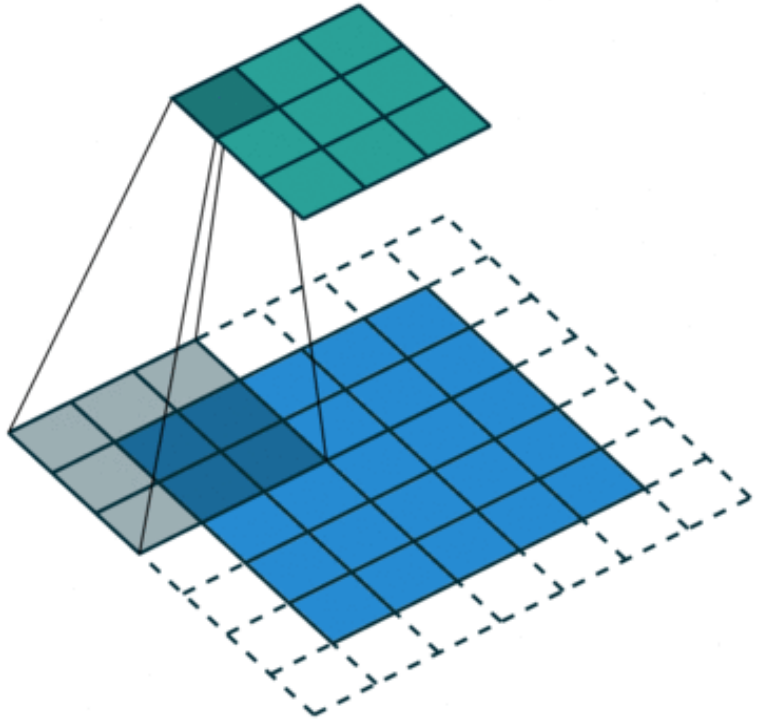
Blue maps are inputs and cyan maps are outputs.

https://github.com/vdumoulin/conv_arithmetic

Convolution (Padding and Strides)



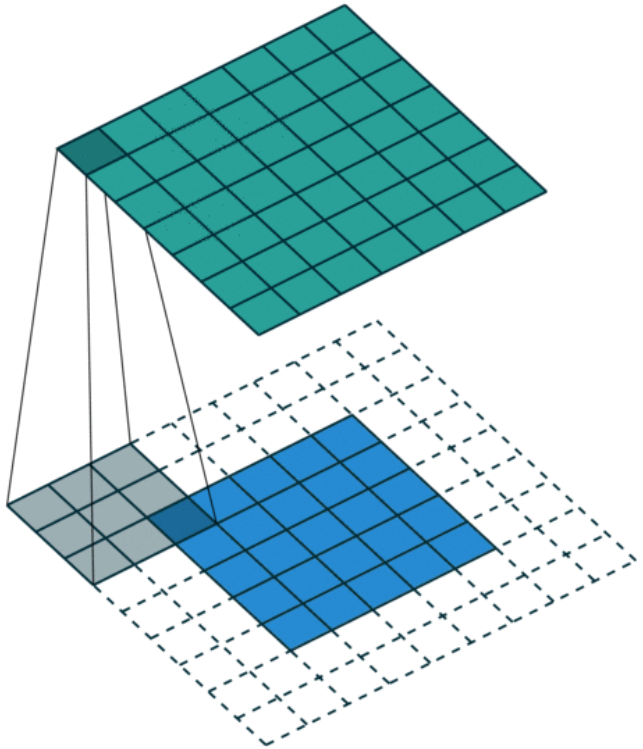
padding=1 and stride=1



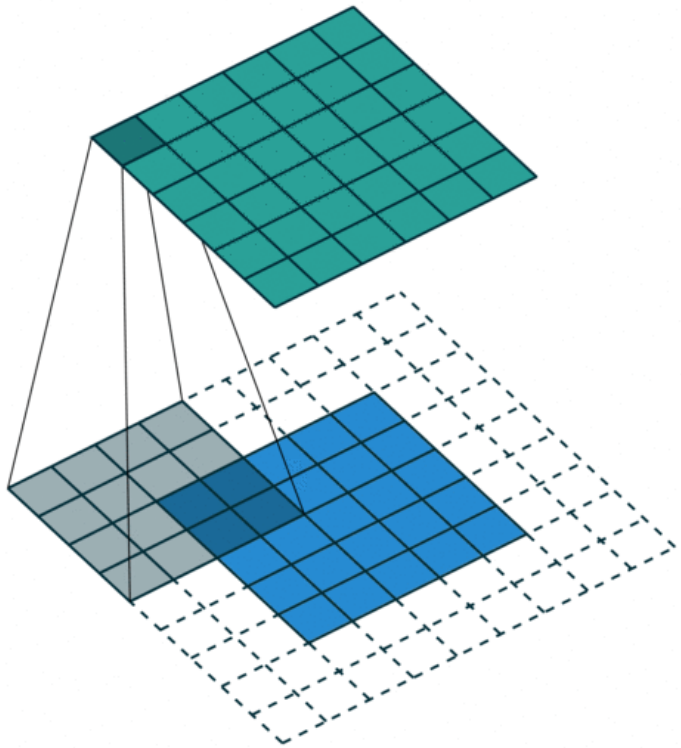
padding=1 and stride=2

https://github.com/vdumoulin/conv_arithmetic

Convolution (Padding and Strides)



padding=2 and stride=1

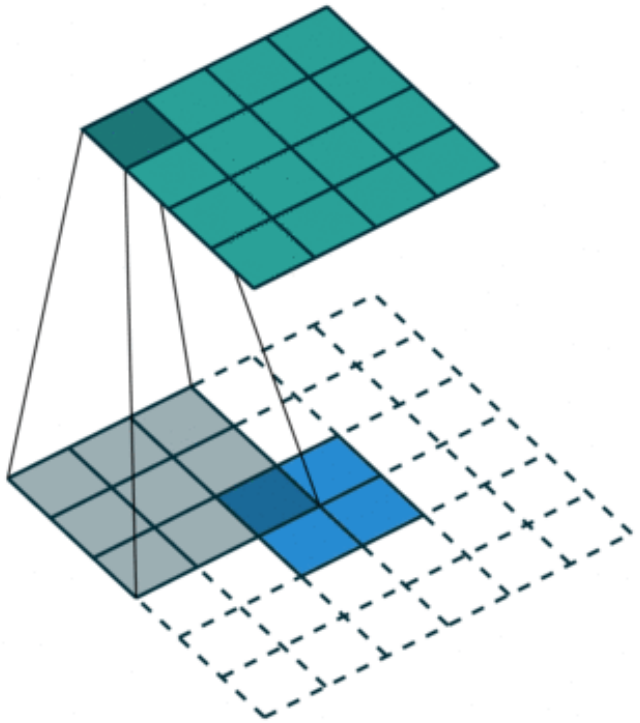


padding=2 and stride=1

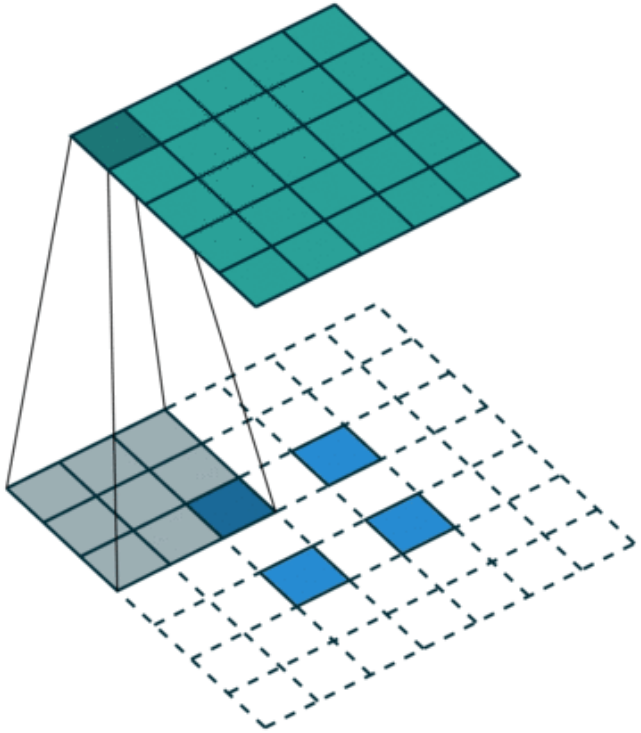
Note that the sizes of the output features vary!

https://github.com/vdumoulin/conv_arithmetic

Transposed Convolution (Padding and Strides)



padding=0 and stride=0

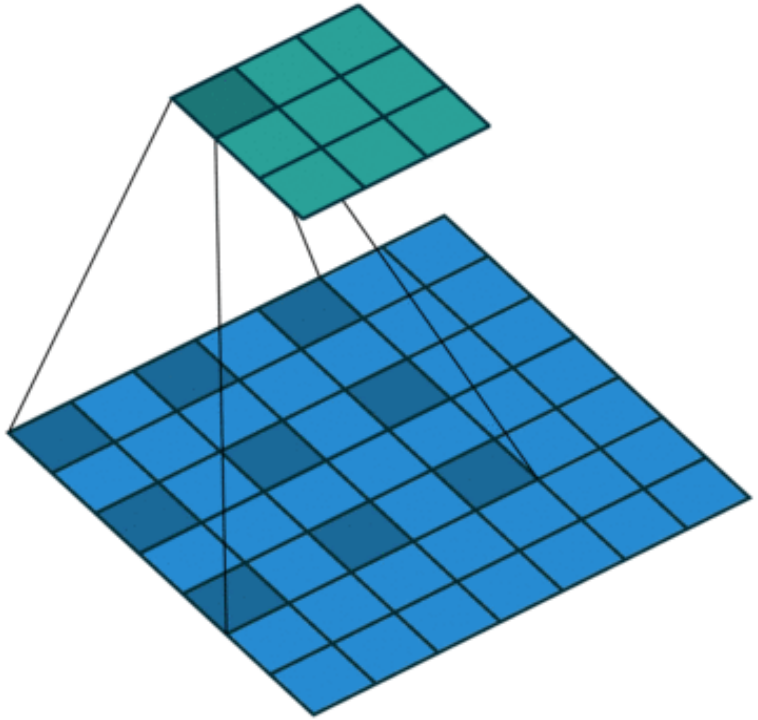


padding=0 and stride=1

Blue maps are inputs and cyan maps are outputs.

https://github.com/vdumoulin/conv_arithmetic

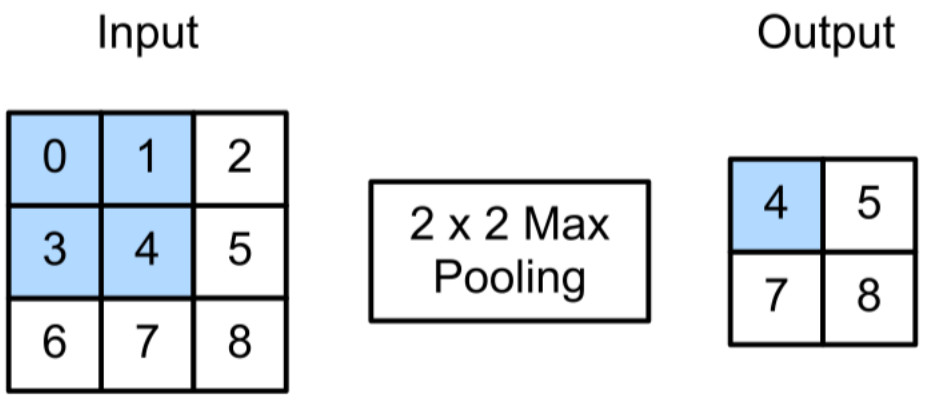
Dilated Convolution (Padding and Strides)



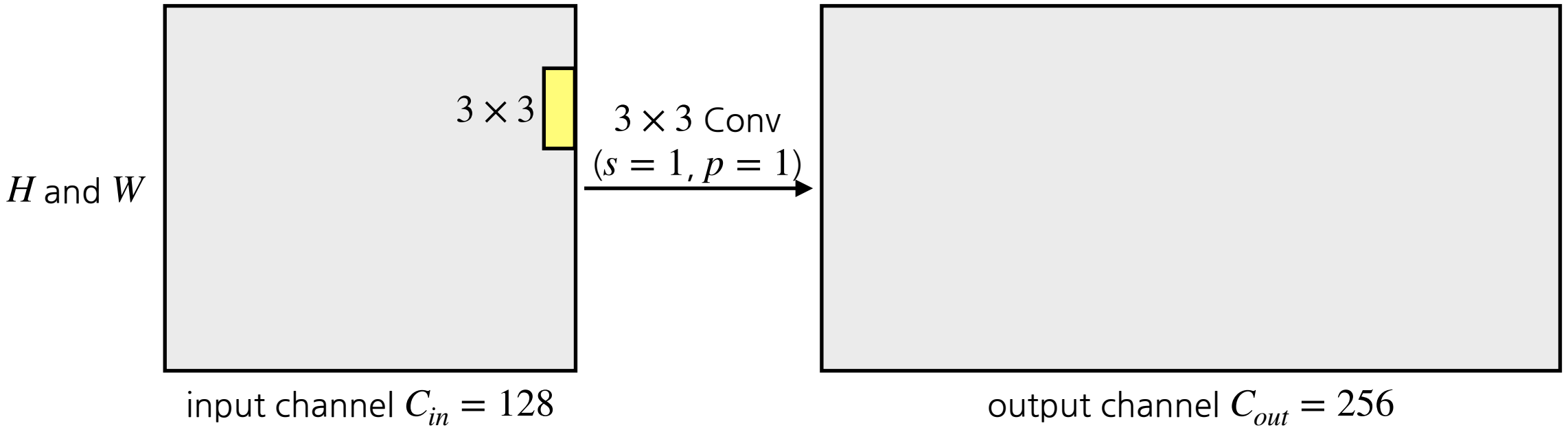
padding=0 and stride=1

https://github.com/vdumoulin/conv_arithmetic

Pooling

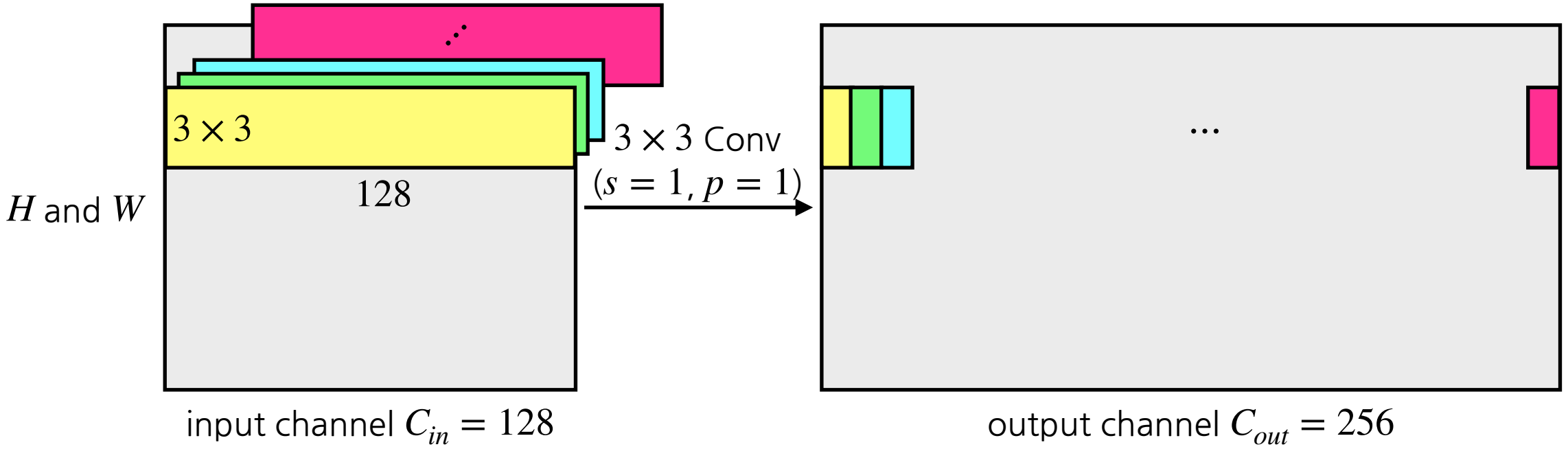


Channel



- What is the number of parameters of this 3×3 convolution layer?

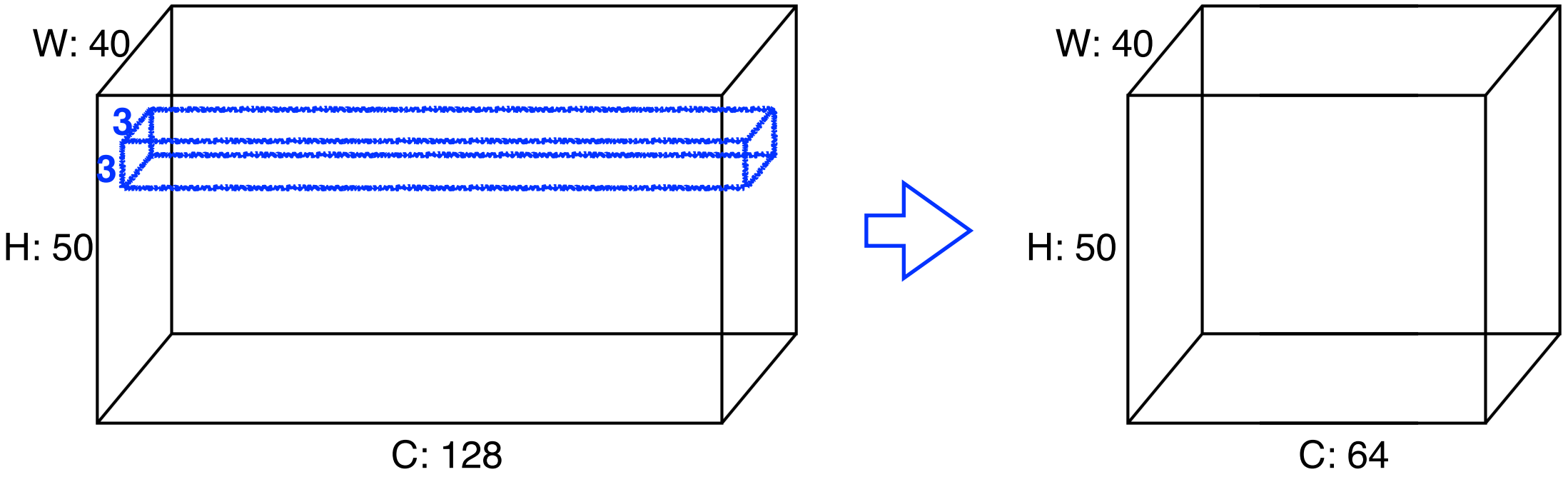
Channel



- What is the number of parameters of this 3×3 convolution layer?
  - Each kernel has a dimension of $128 \times 3 \times 3$
  - And we have 256 kernels.
  - The number of parameters is $128 \times 256 \times 3 \times 3 = 294{,}912$

Channel

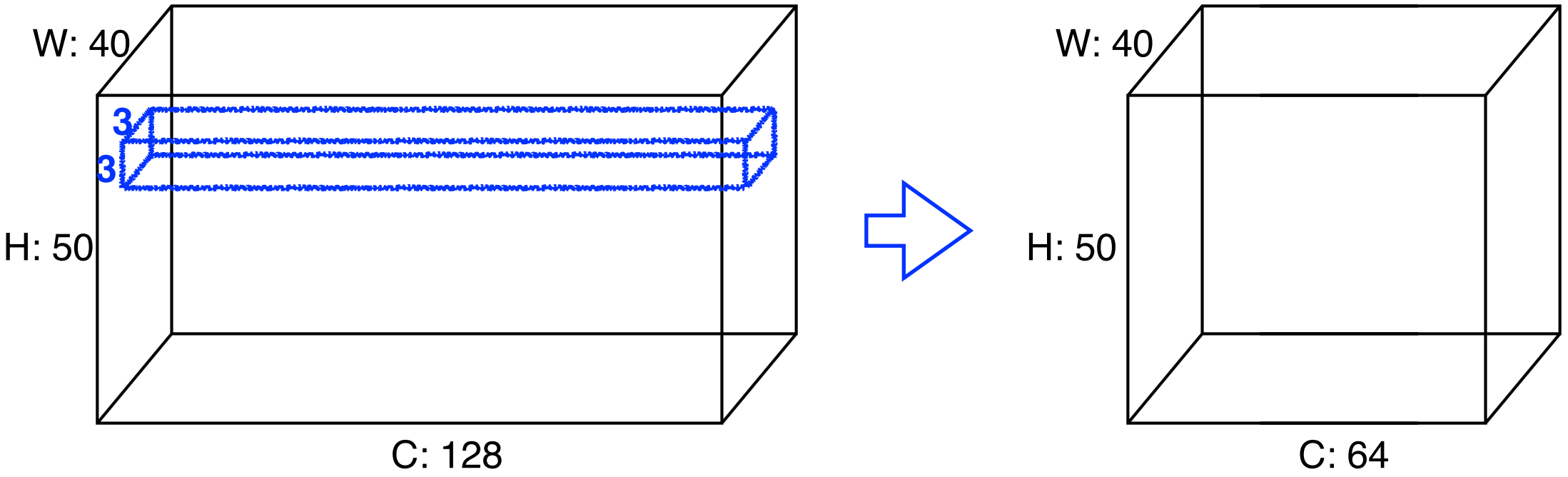
- Padding (1), Stride (1), 3×3 Kernel



What is the **number of parameters** of this model?

Channel

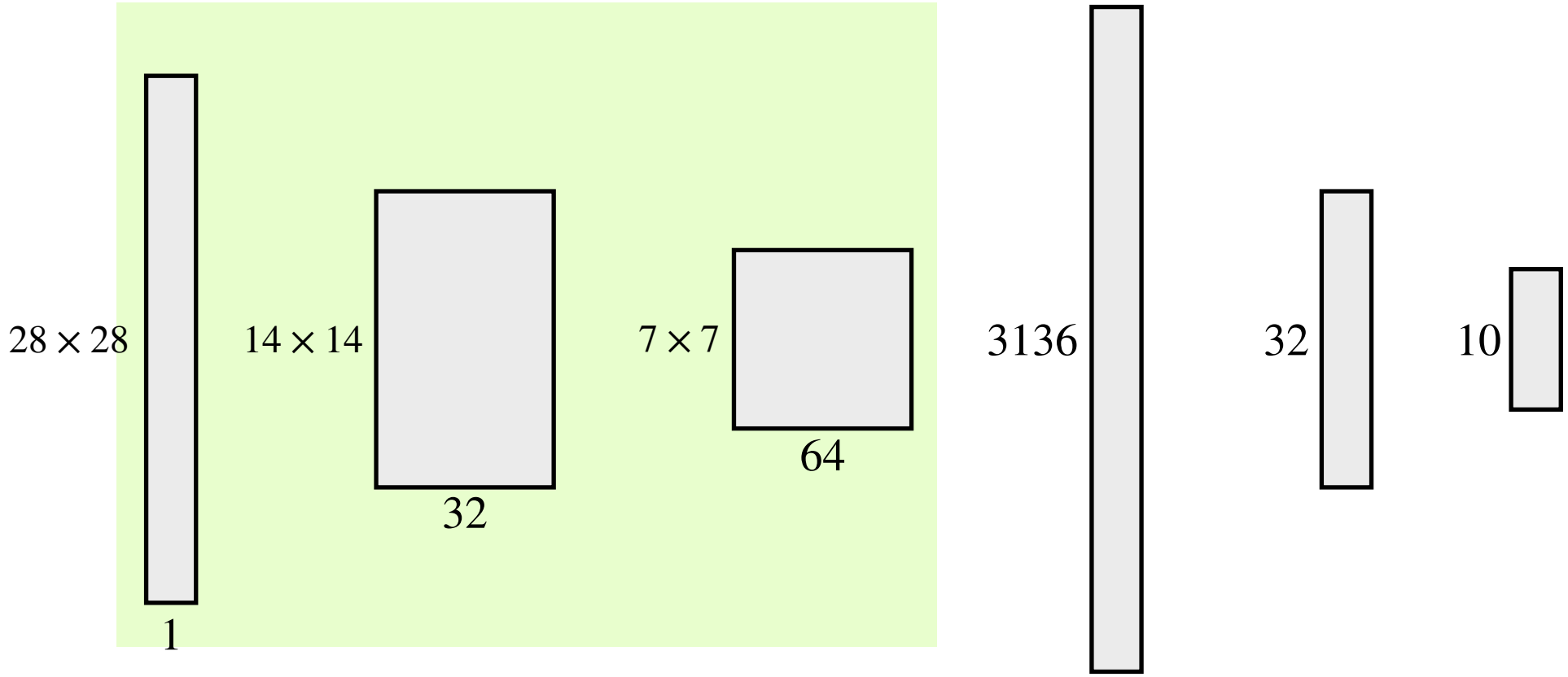
- Padding (1), Stride (1), 3×3 Kernel



What is the **number of parameters** of this model?

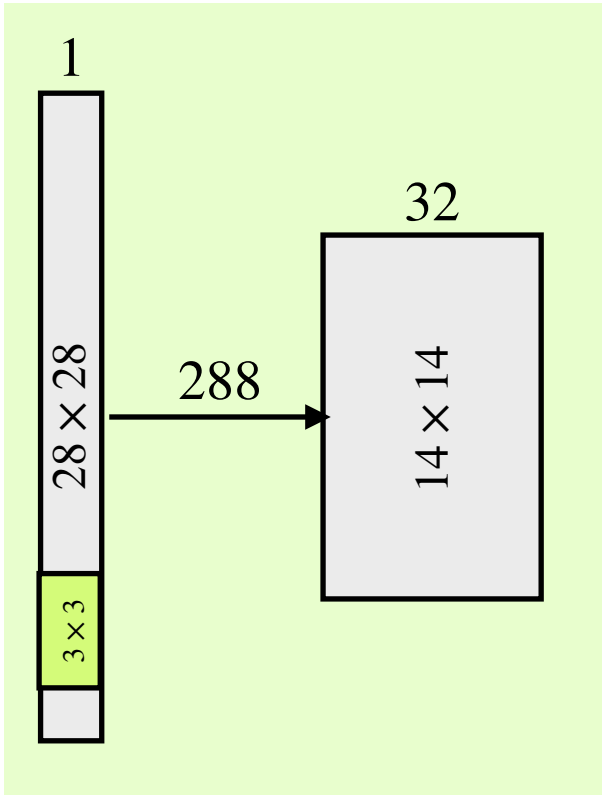
The answer is $3 \times 3 \times 128 \times 64 = 73{,}728$

Convolutional Neural Network

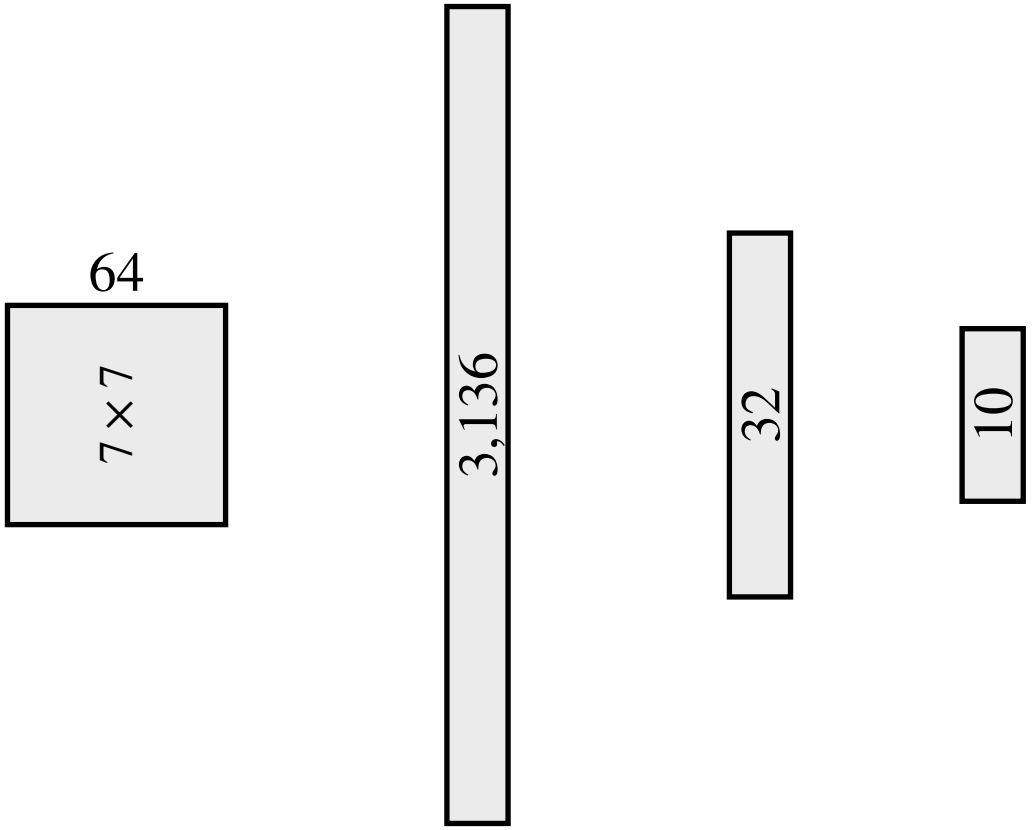


- Convolutions with 3×3 kernels, stride is two, and padding is one.
- Max-pooling with 2×2 kernels.

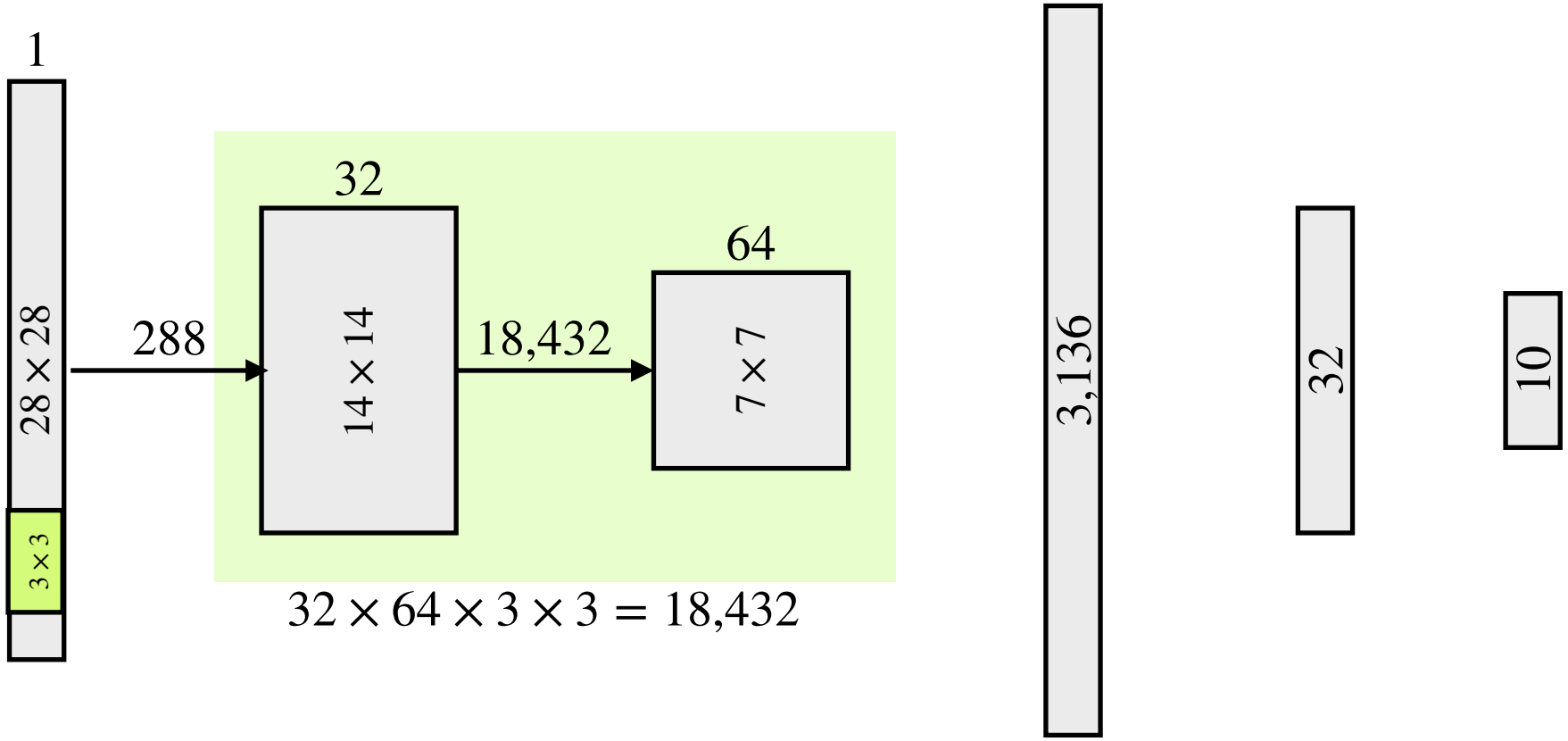
Convolutional Neural Network



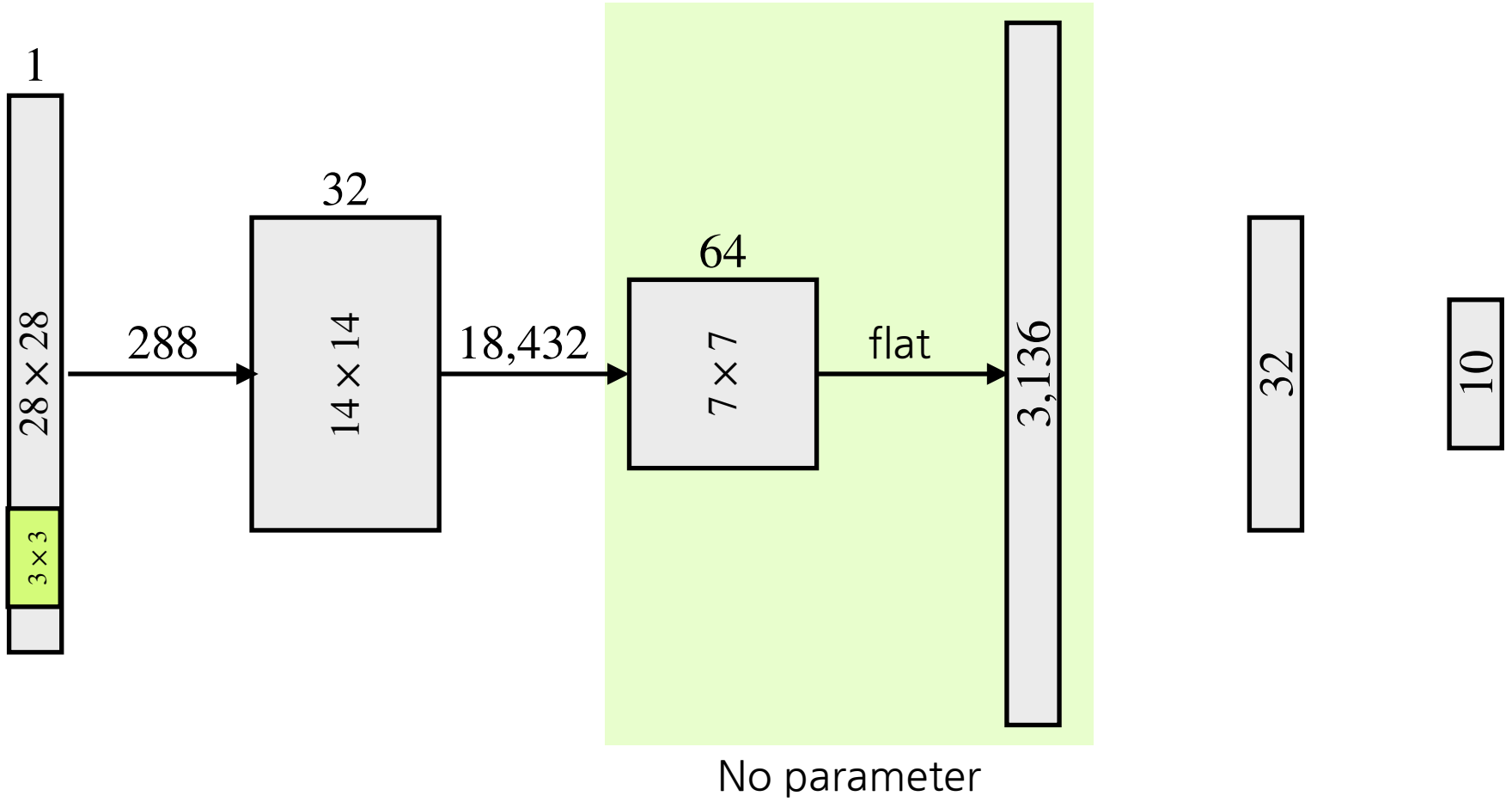
$1 \times 32 \times 3 \times 3 = 288$



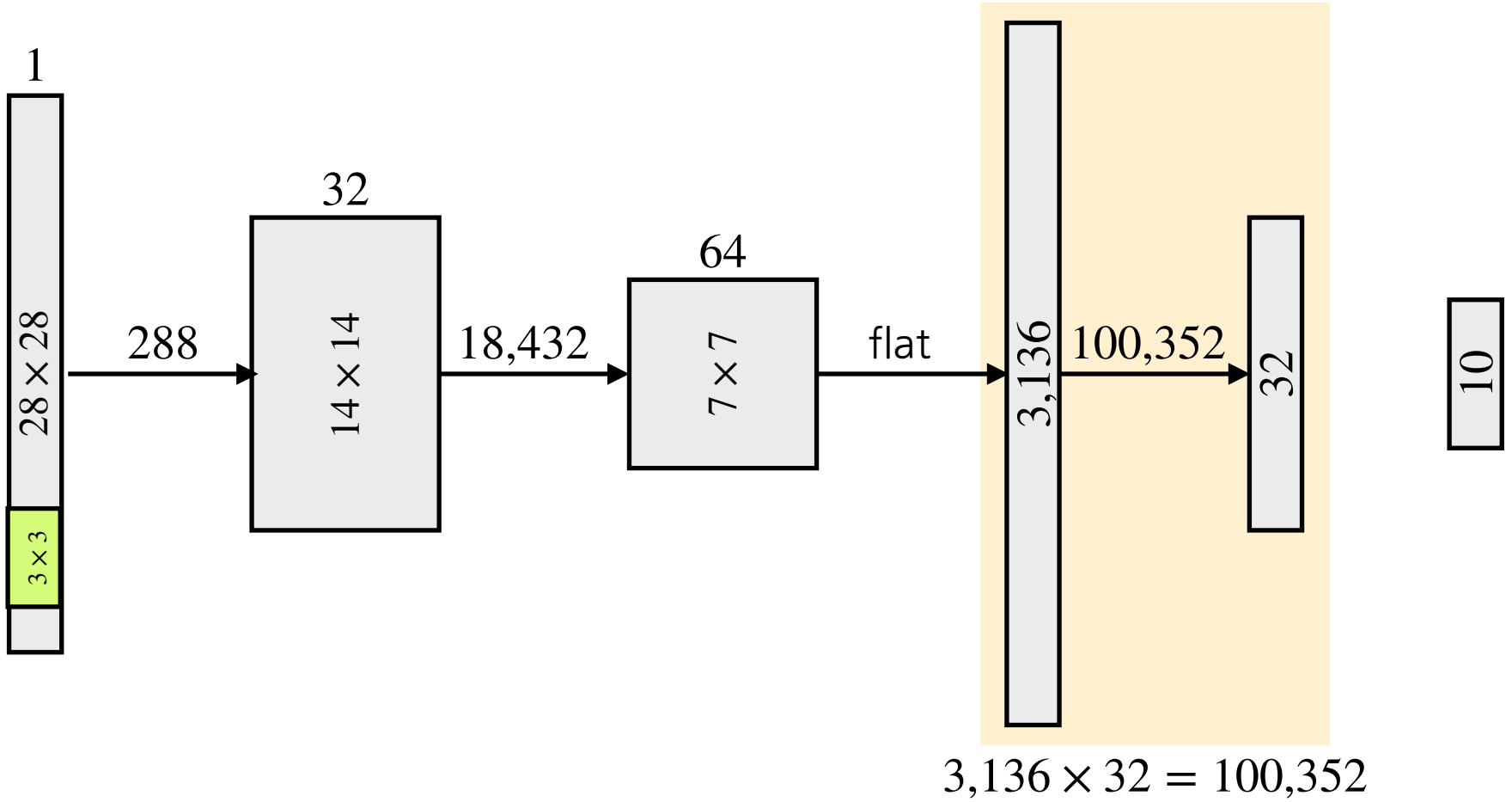
Convolutional Neural Network



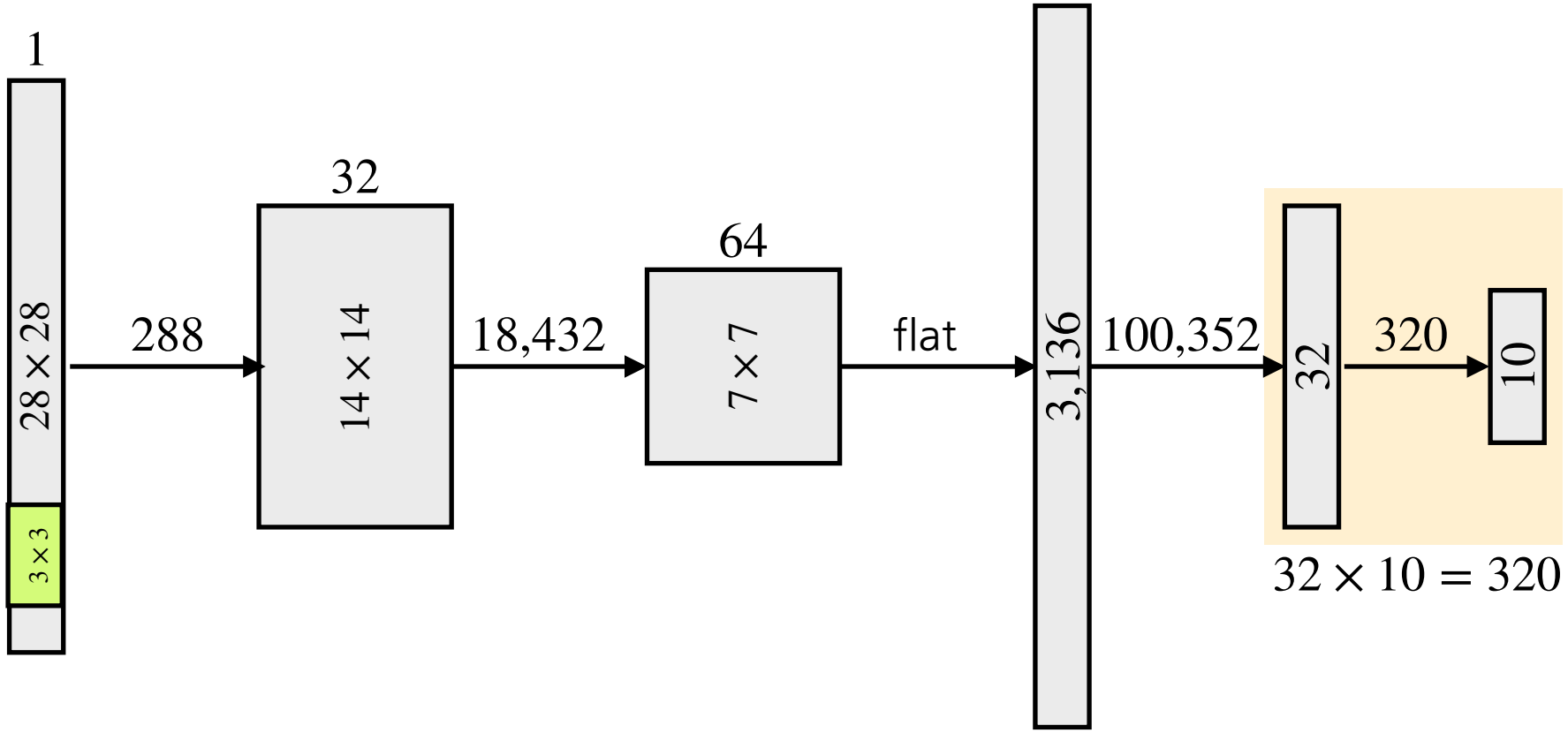
Convolutional Neural Network



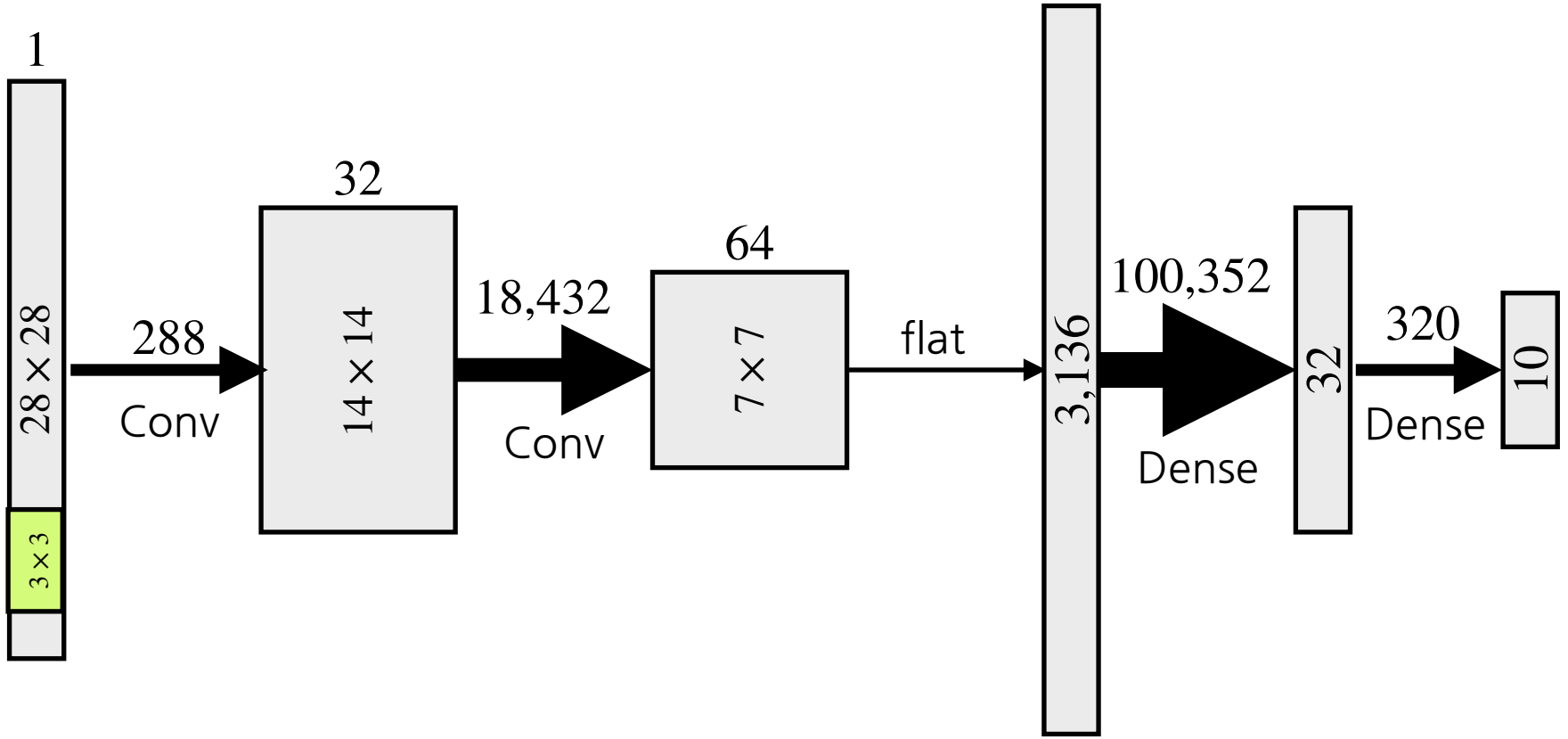
Convolutional Neural Network



Convolutional Neural Network



Convolutional Neural Network



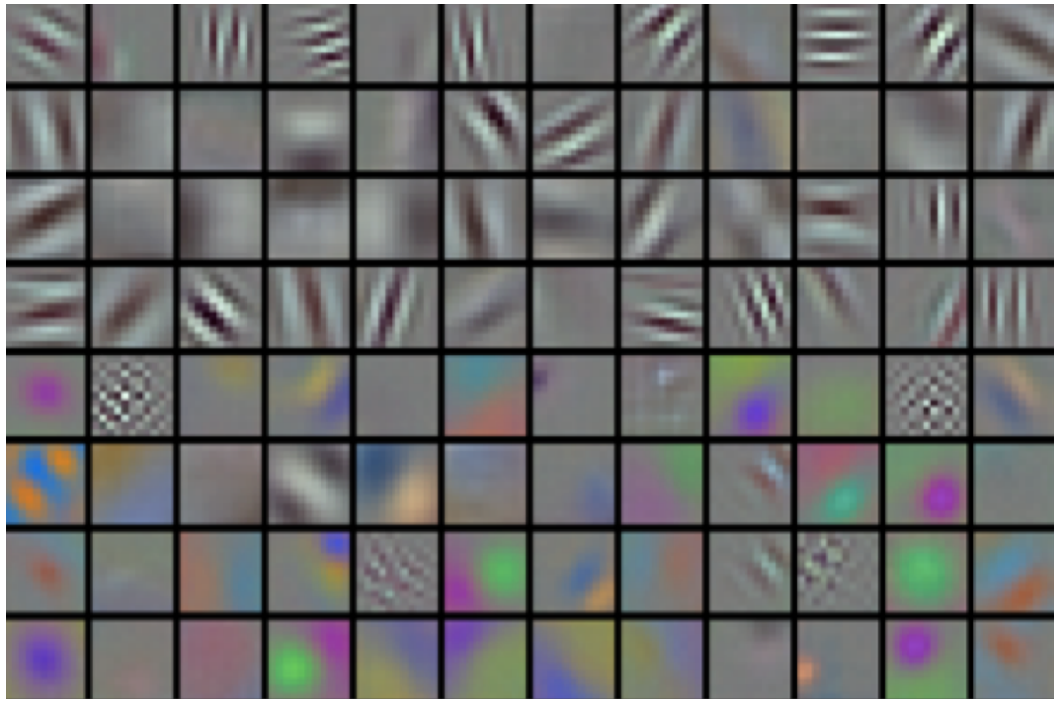
Modern CNNs

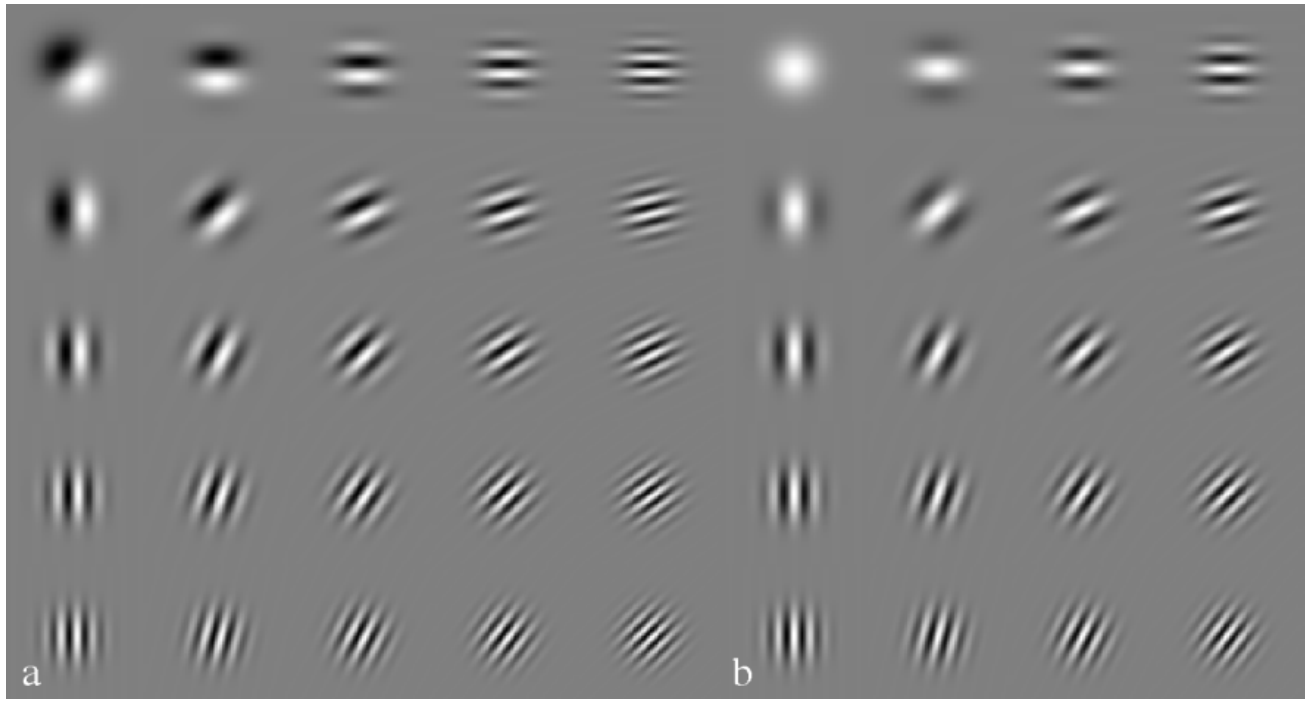
History

- The introduction of **LeNet** in 1995 introduced CNNs to computer vision and machine learning communities.
- However, CNNs did not immediately dominate the field until AlexNet in 2012.
- Back in the day, classical pipelines looked more like this:
  - Obtain an interesting dataset.
    - For instance, the Apple QuickTake 100 of 1994 supported 640x480 resolution (VGA), capable of storing up to 8 images, for the price of $1,000.
  - Preprocess the dataset with hand-crafted features based on some knowledge of optics and geometry and occasionally on the serendipitous discoveries of lucky graduate students.
    - SIFT (scale-invariant feature transform) or SURF (speeded-up robust features)
  - Dump the resulting representation into your favorite classifier
    - A linear model or kernel method

Apple QuickTake 100

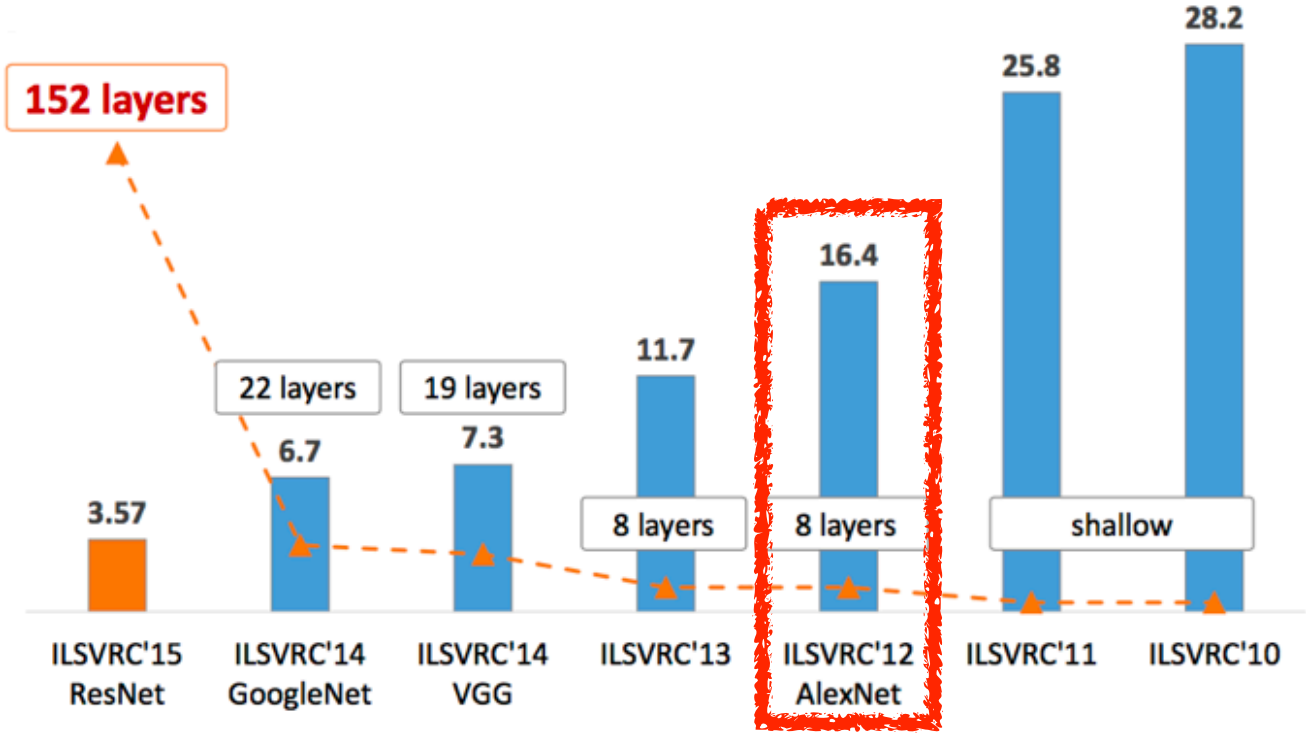
Representation Learning





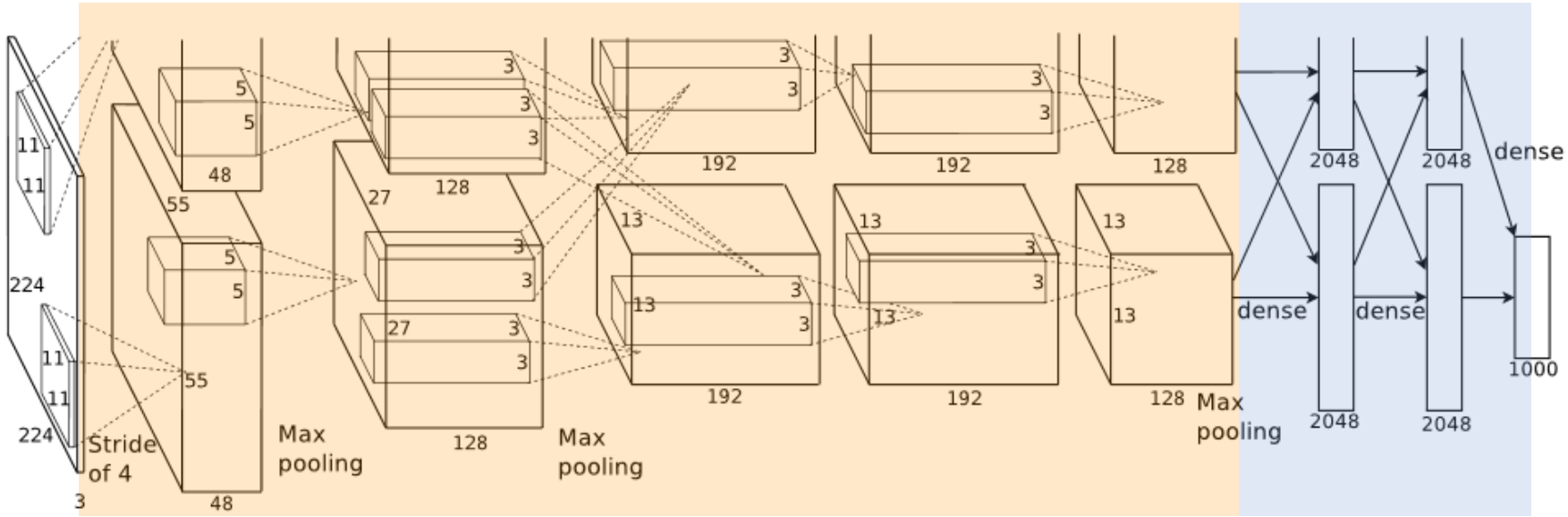
- The left figure shows image filters learned by the first layer of AlexNet.
- Interestingly, in the lowest layers of the network, the model learned feature extractors that resembled some traditional filters, as shown in the right figure.

AlexNet



"ImageNet Classification with Deep Convolutional Neural Networks," 2012

Architecture

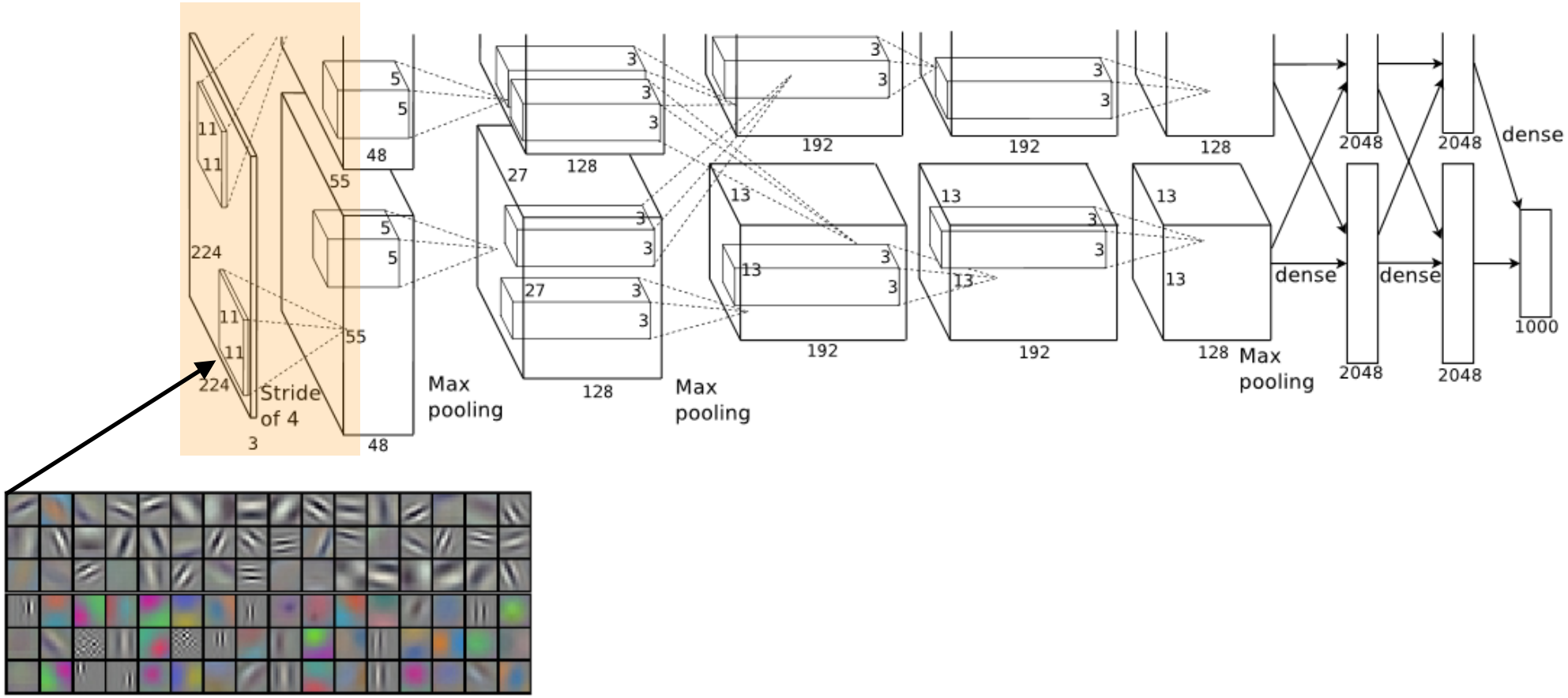


224x224 RGB image

5 convolutional layers

3 dense layers

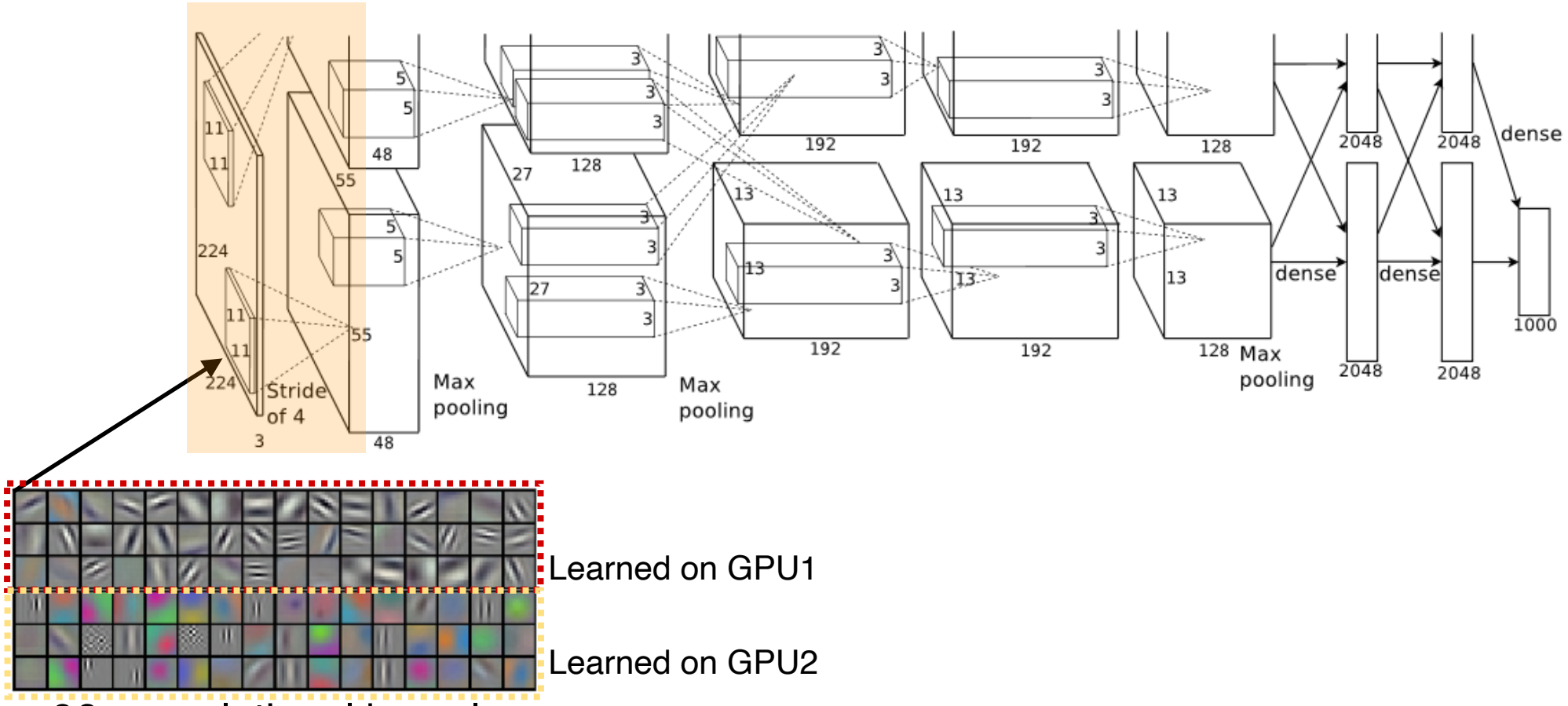
Architecture



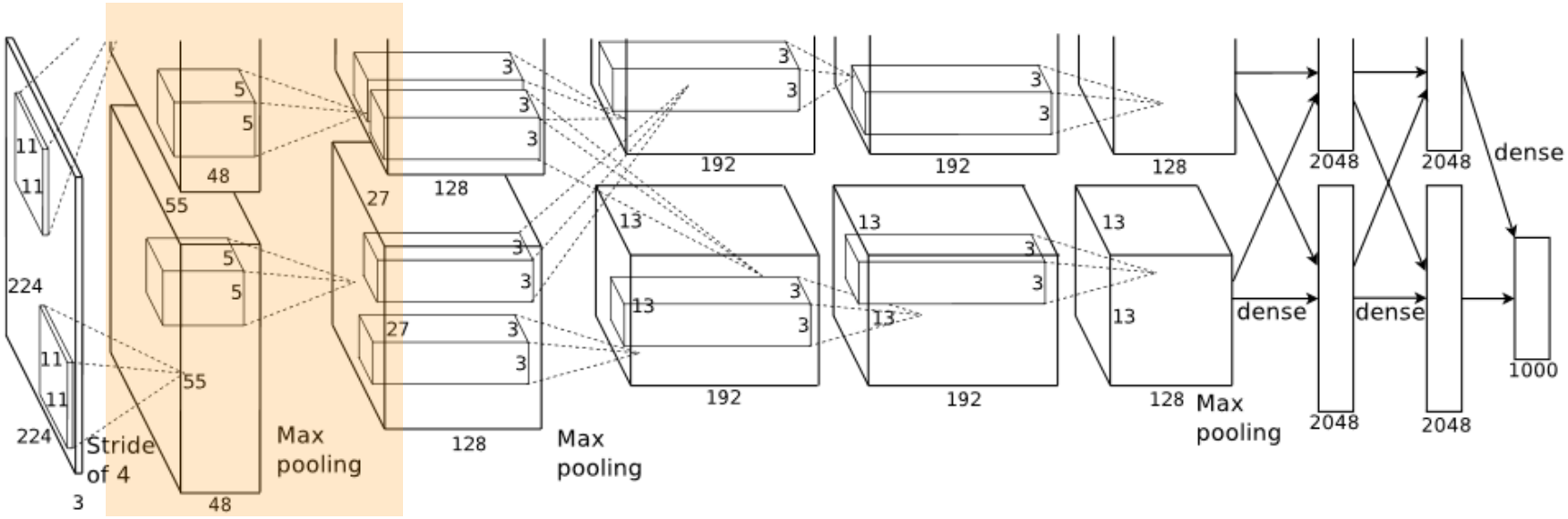
96 convolutional kernels

of size 11x11x3 filters

Architecture



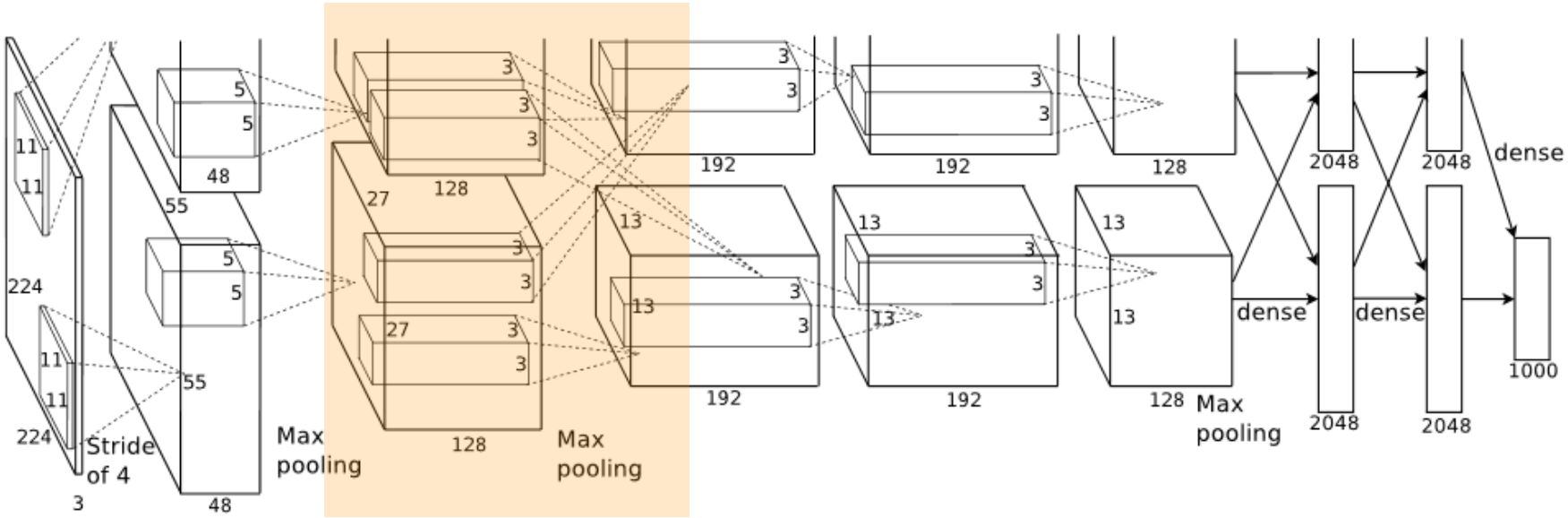
Architecture



256 convolutional kernels

of size 5x5x48 filters

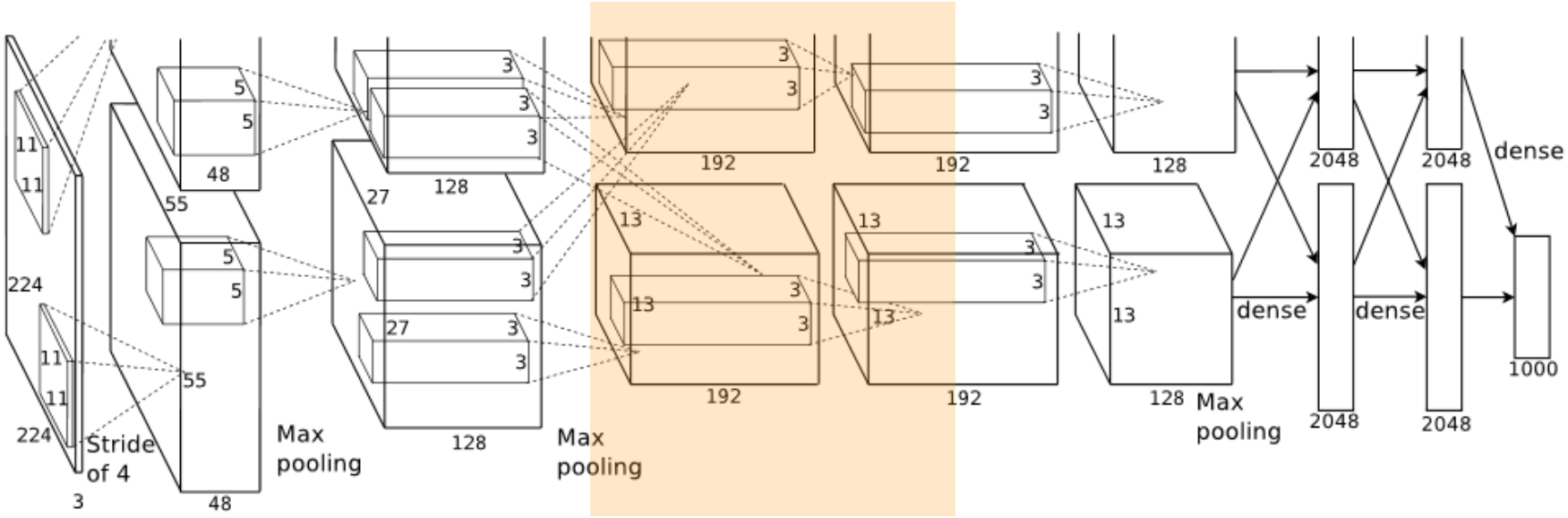
Architecture



384 convolutional kernels

of size 3x3x128 filters

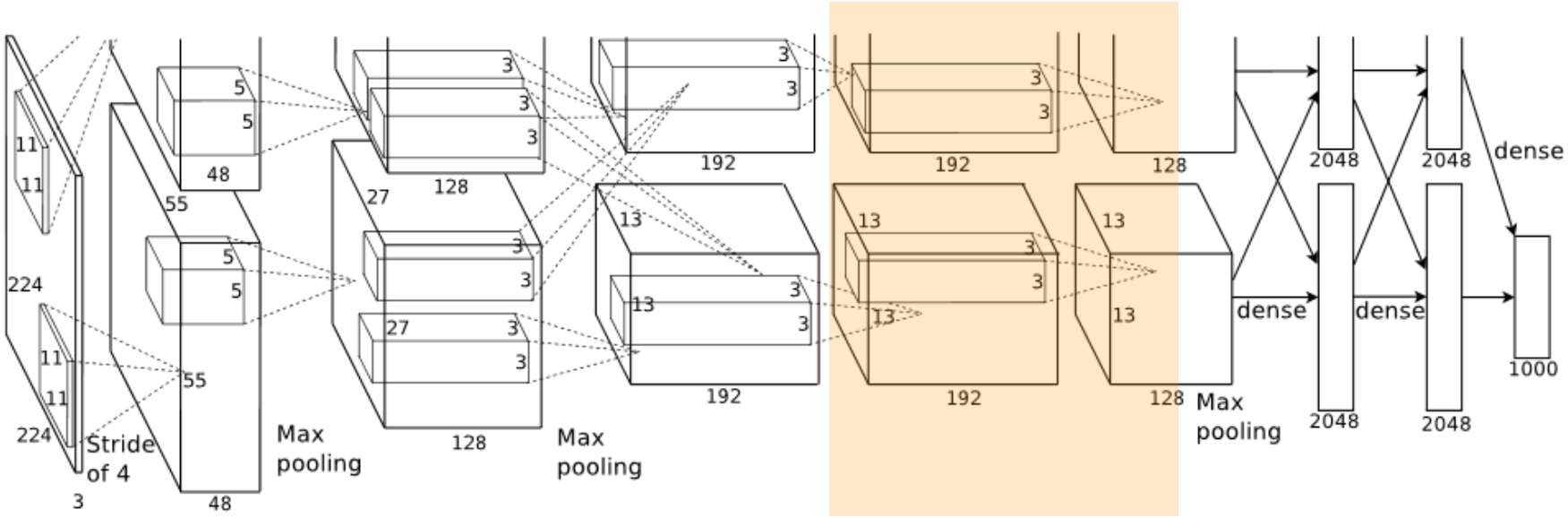
Architecture



384 convolutional kernels

of size 3x3x192 filters

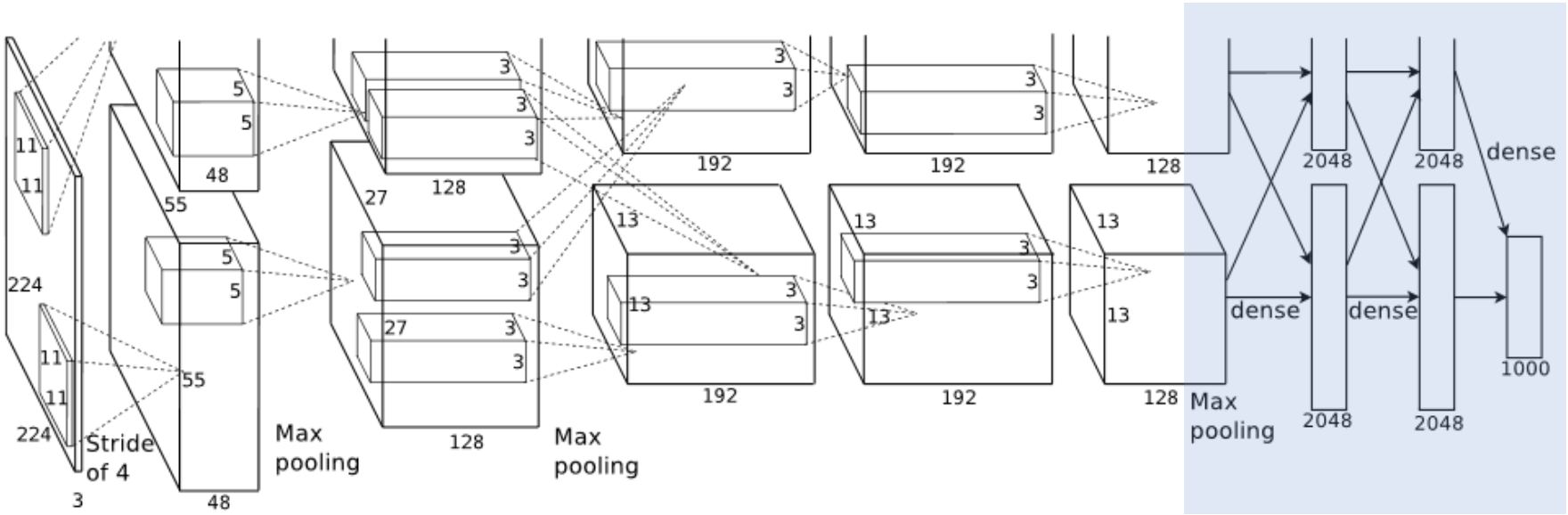
Architecture



256 convolutional kernels

of size 3x3x192 filters

Architecture

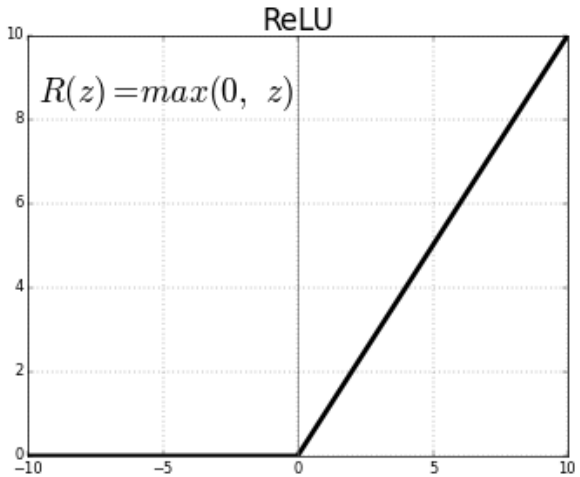


Implementation Details

- Rectified Linear Unit (ReLU) activation
- Training on Multiple GPUs (2 GPUs)
- Local Response Normalization
- Overlapping Pooling
- Reducing Overfitting
    - Data Augmentation
    - Dropout

Implementation Details

- **Rectified Linear Unit (ReLU) activation**
  - They apply this activation function to each layer.
  - ReLU can preserves properties of linear models.
  - Easy to optimize with gradient descent.
  - ReLUs can train several times faster than their equivalents with tanh units.



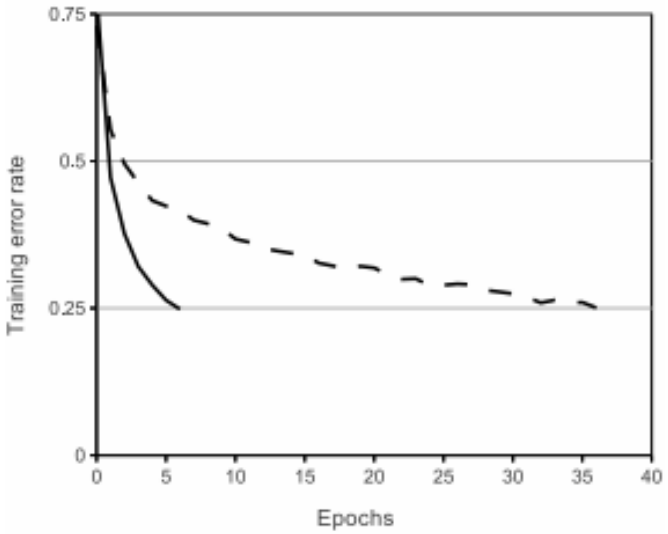
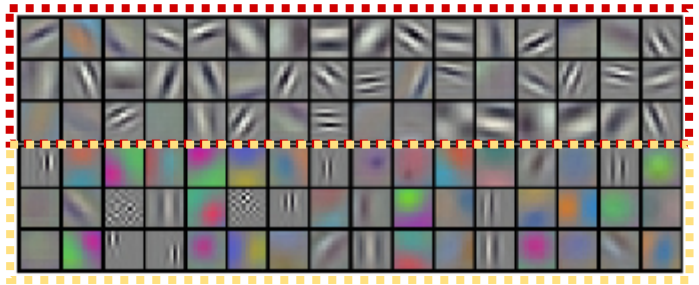


Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons **(dashed line)**. The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.

Implementation Details

- **Training on Multiple GPUs (2 GPUs)**
  - A single GTX 580 GPU has only 3GB of memory.
  - So, they spread the network and train across two GPUs.
  - They put half of the kernels (or neurons) on each GPU.



96 convolutional kernels

of size 11x11x3 filters

Implementation Details

- **Local Response Normalization (LRN)**
  - A technique used to enhance the generalization ability of a neural network.
  - This normalization apply lateral inhibition, which encourages competition among neurons within the same layer.
  - Where strong responses suppress weaker ones in nearby neurons, leading to more effective feature extraction.
  - Helps reduce overfitting by normalizing activations, leading to better performance on unseen data.

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

Implementation Details

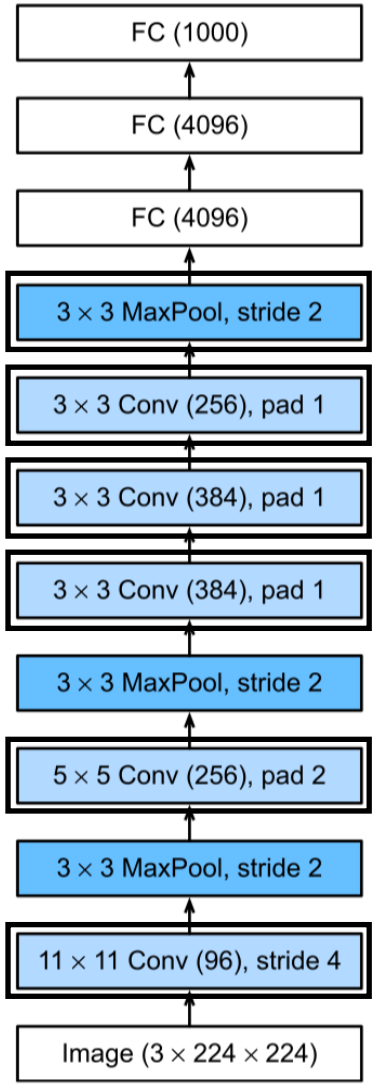
- **Overlapping Pooling**
  - The pooling windows overlap with each other.
  - In traditional pooling, the pooling window slides over the image without any overlap, meaning the step size (stride) is equal to the pooling window size.
  - In AlexNet, the pooling window is set to 3x3 and the stride is set to 2, meaning the pooling windows overlap by one pixel.
  - This approach reduces the spatial dimensions of the feature maps, such as regular pooling.
  - This allows for more information to be retained since neighboring regions contribute to the pooled value.

Implementation Details

- **Reducing Overfitting**
  - Data Augmentation
    - Image Translations and Reflections
    - PCA-based Color Alteration
  - Dropout
    - In the fully connected layers, some neurons are randomly "dropped out" (i.e., set to zero) during training.

Number of parameters

- Kernel: [out_channels x in_channels x kernel_height x kernel_width]



$4{,}096 \times 1{,}000 = 4{,}096{,}000$

$4{,}096 \times 4{,}096 = 16{,}777{,}216$

$4{,}096 \times 6{,}400 = 26{,}214{,}400$

$256 \times 384 \times 3 \times 3 = 884{,}736$

$256 \times 384 \times 3 \times 3 = 884{,}736$

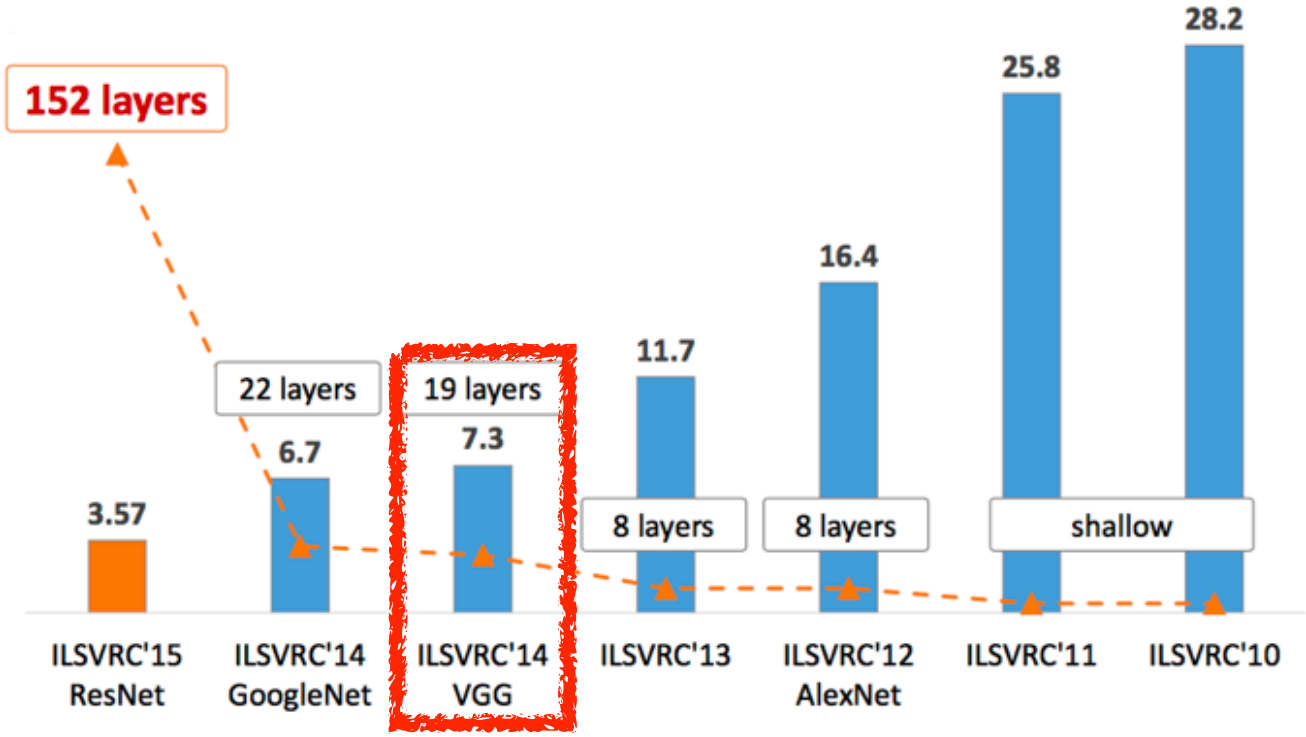
$384 \times 384 \times 3 \times 3 = 1{,}327{,}104$

$384 \times 256 \times 3 \times 3 = 884{,}736$

$256 \times 96 \times 5 \times 5 = 614{,}400$

$96 \times 3 \times 11 \times 11 = 34{,}848$

VGGNet



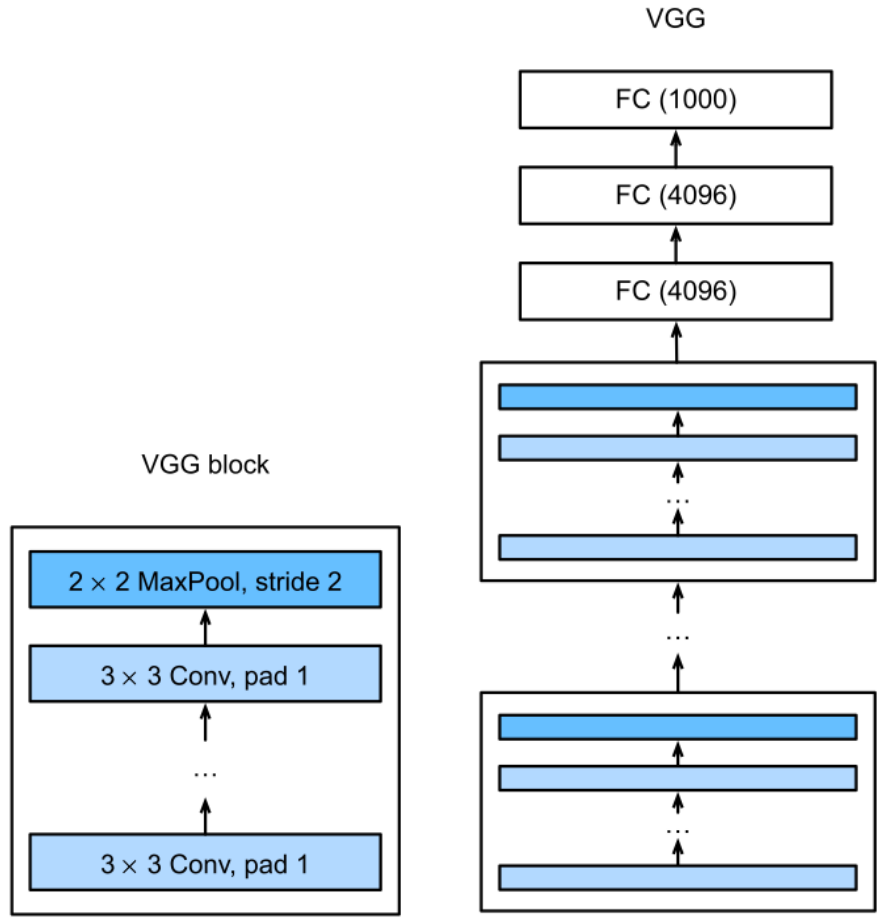
"Very Deep Convolutional Networks for Large-Scale Image Recognition," 2015

Contribution

- The main goal of the VGGNet is to demonstrate that increasing the depth of CNNs significantly improves performance on large-scale image recognition tasks.
- To enable deeper networks, architectures are with 16-19 layers, using very small convolutional filters (3x3).
- While AlexNet introduced many of the components of what makes deep learning effective at scale, it is VGG that arguably introduced key properties such as blocks of multiple convolutions and a preference for deep and narrow networks.

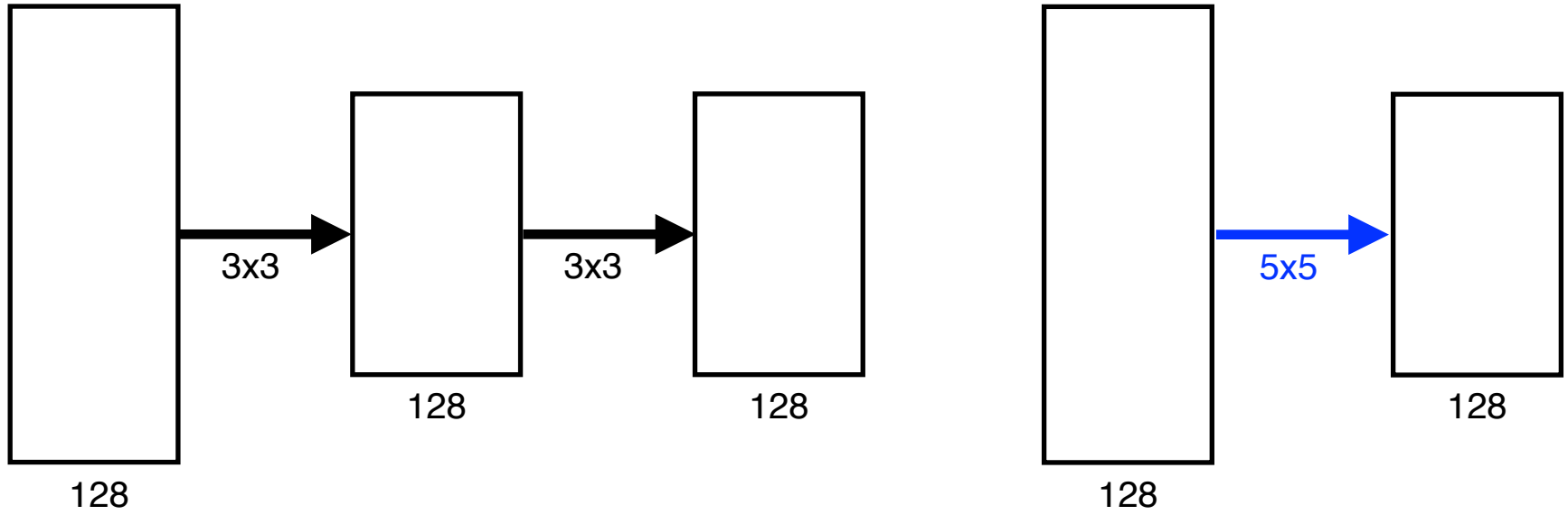
Architecture

- Basic building blocks
  - a convolutional layer with padding to maint
  - a nonlinearity such as a ReLU
  - a pooling layer such as max-pooling to redu
- The key idea of VGG was to use multiple convc downsampling via max-pooling in the form of a



Depth of the network

- Why use 3x3 convolution



Receptive field	5x5	5x5
# of params	3x3x128x128+3x3x128x128 = **294,912**	5*5*128*128 = **409,600**

Performance

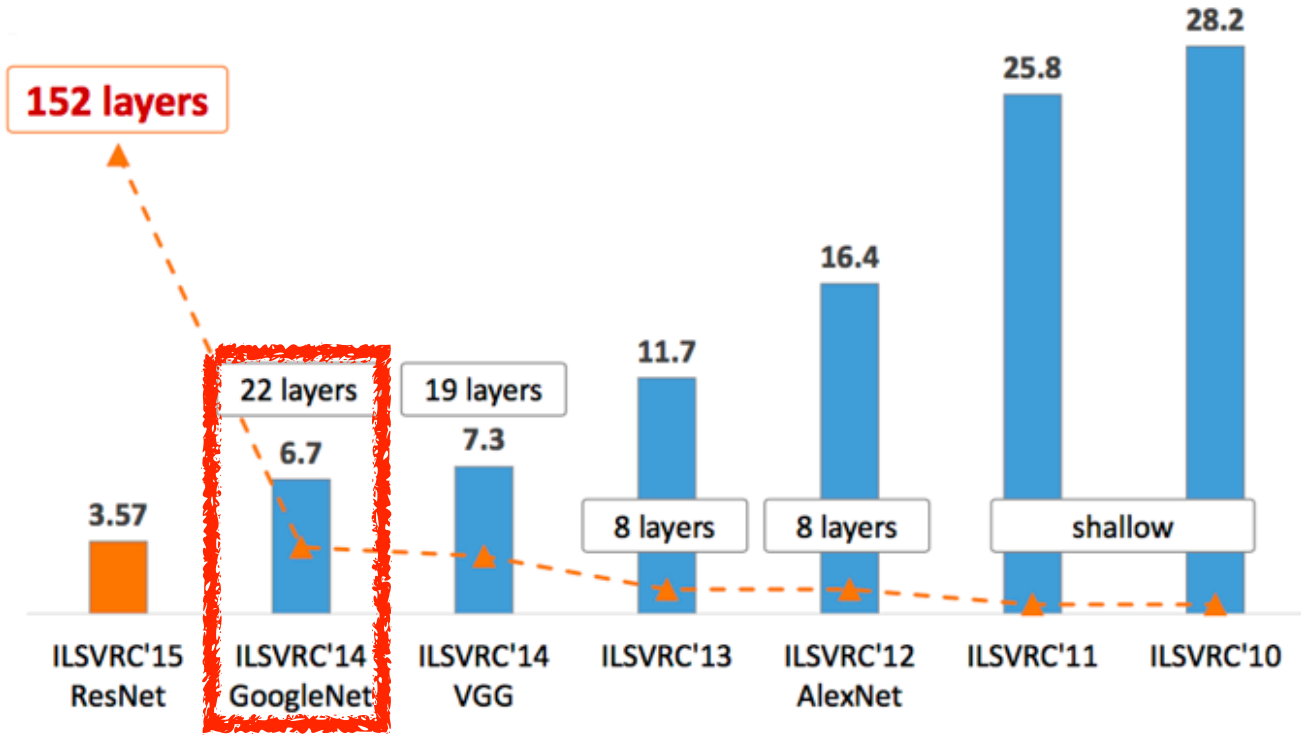
- The VGGNet achieves better accuracy as the depth increases, with the 19-layer model (VGG-19) providing the best performance.
- The architecture outperformed earlier models, such as AlexNet (which used 11x11 and 5x5 filters)

Table 7: **Comparison with the state of the art in ILSVRC classification**. Our method is denoted as "VGG". Only the results obtained without outside training data are reported.

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

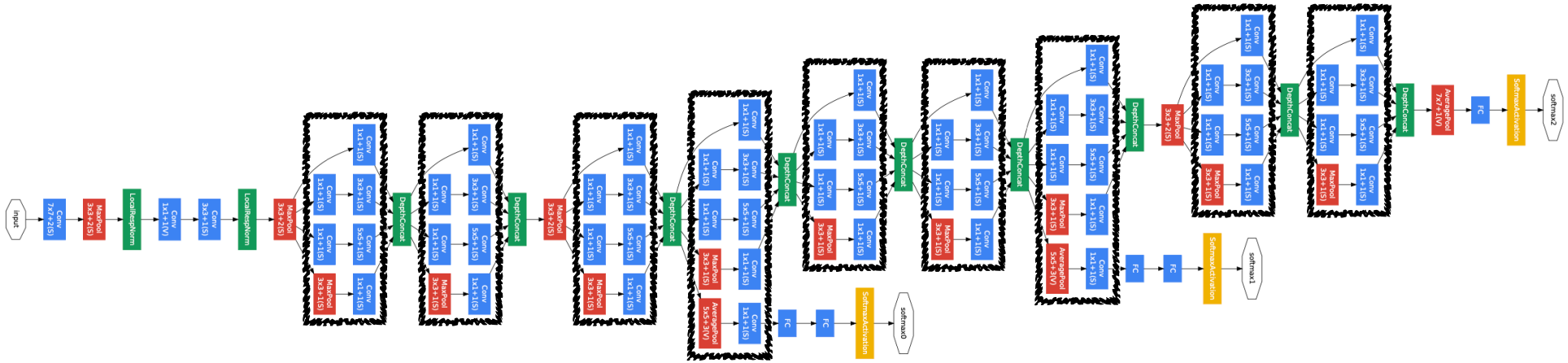
GoogLeNet

"Going Deeper with Convolutions," 2015

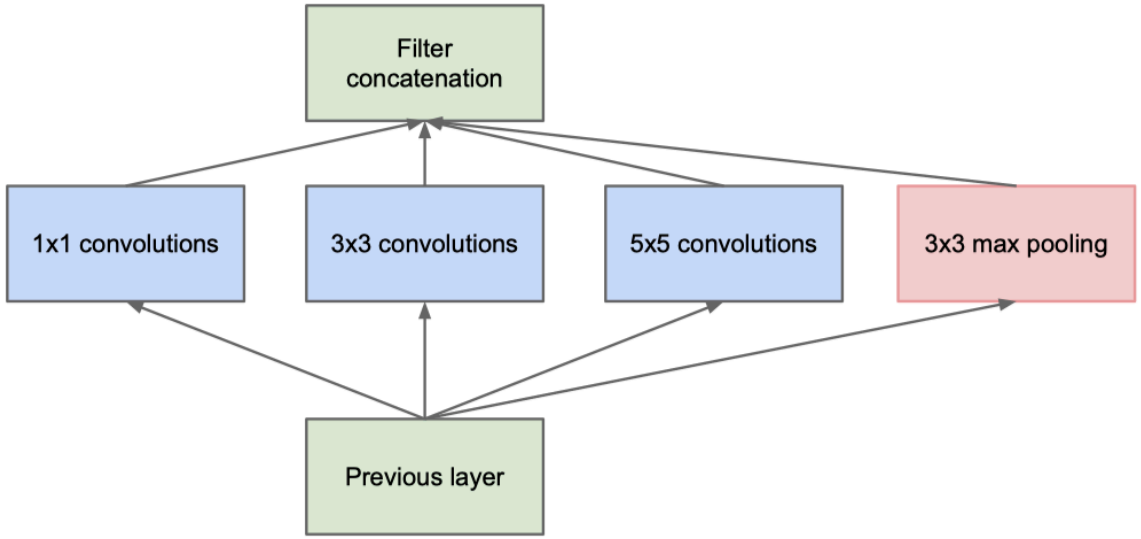


Multi-Branch Networks

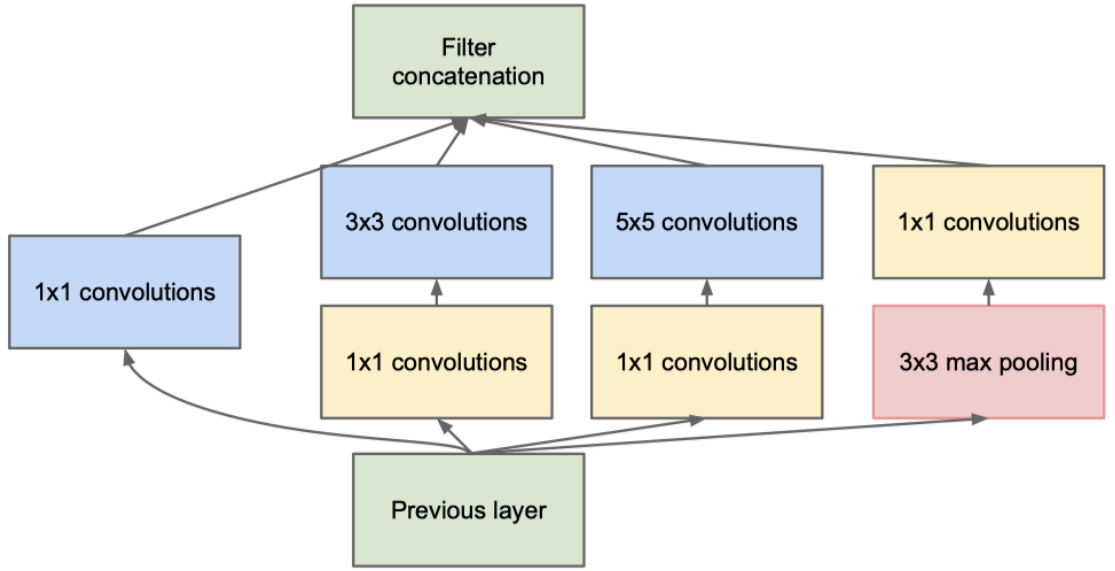
- It is the first network that exhibits a clear distinction among the stem (data ingest), body (data processing), and head (prediction) in a CNN.
  - The **stem** is given by the first 2-3 convolutions that operate on the image.
  - This is followed by a **body** of convolutional blocks.
  - Finally, the **head** maps the feature obtained so far to the required problem (e.g., classification or detection).
- The key contribution in GoogLeNet was the design of the **network body**.



Inception Blocks



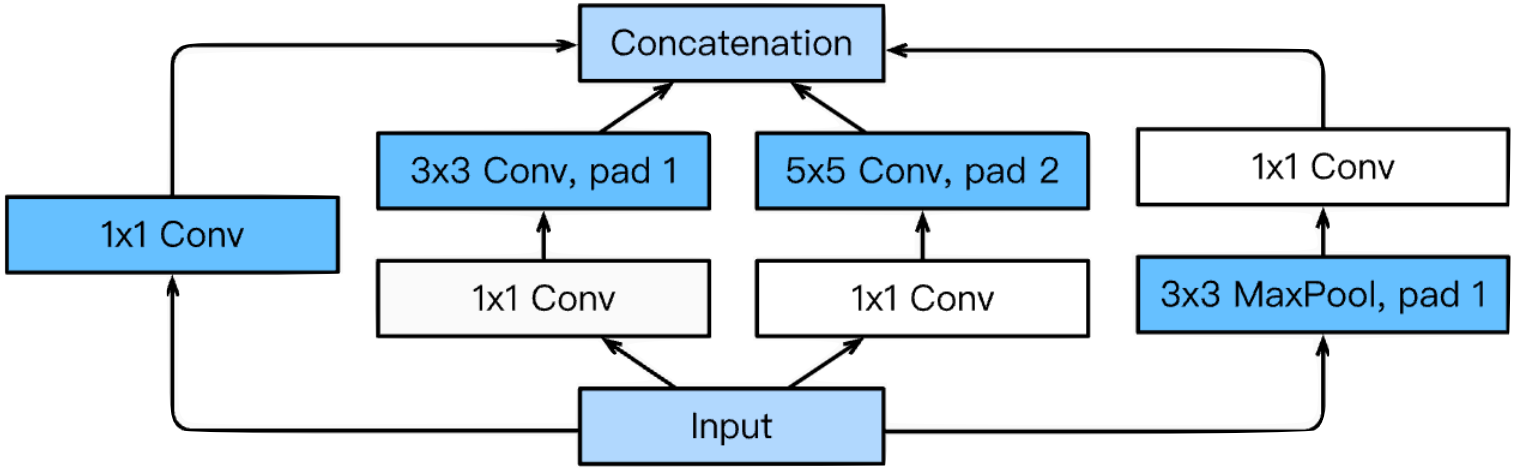
(a) Inception module, naïve version



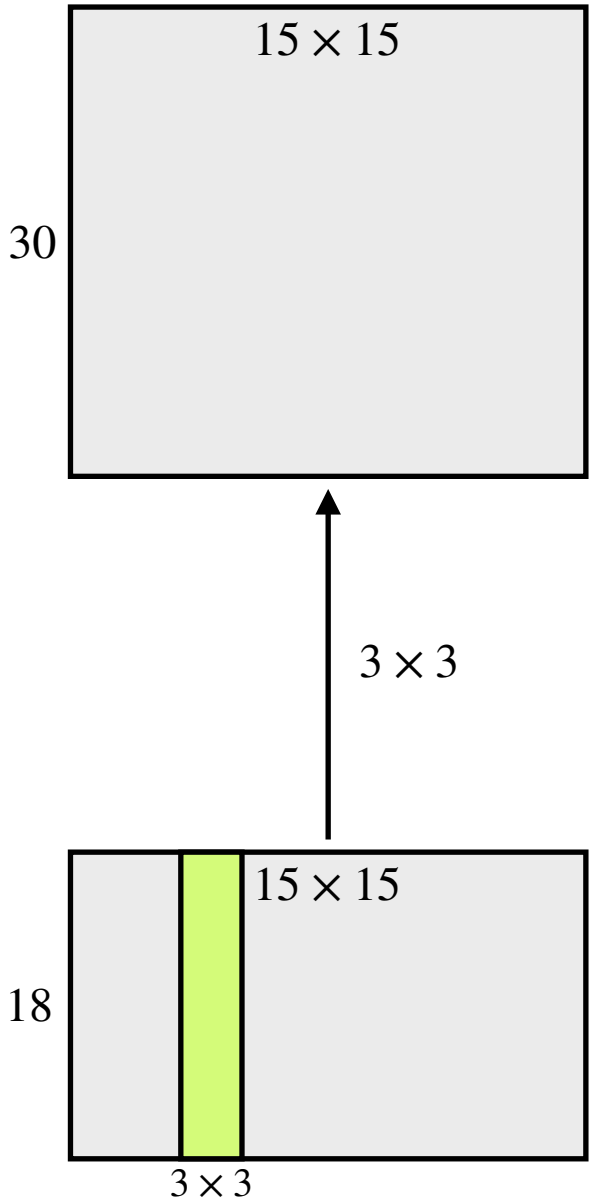
(b) Inception module with dimension reductions

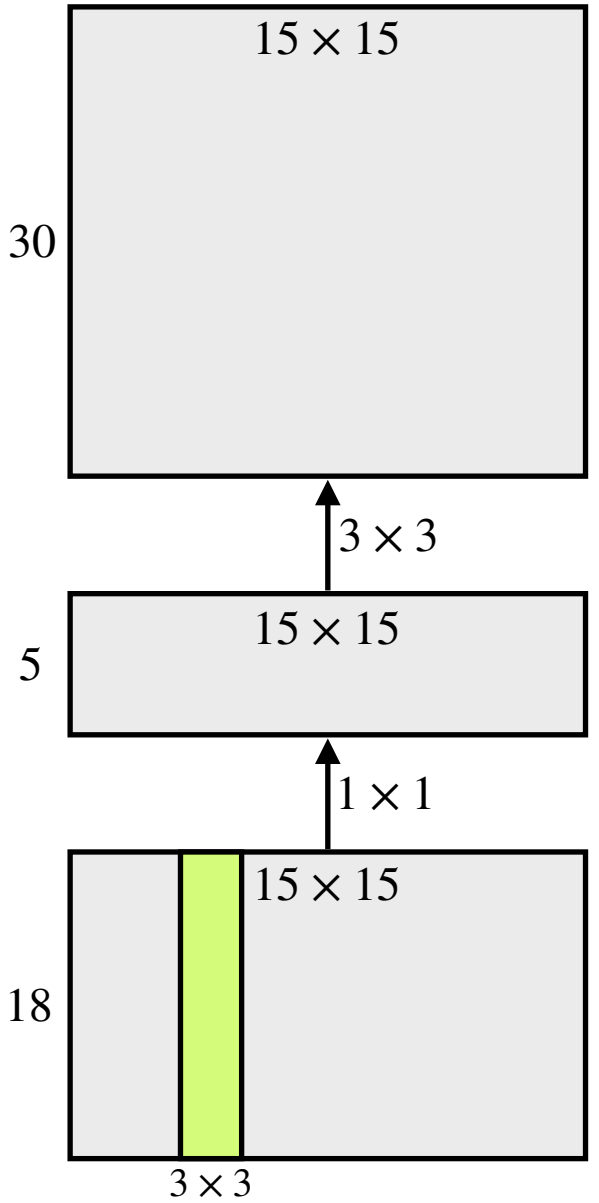
Inception Blocks

- What are the benefits of the inception block?
  - Reduce the number of parameter.
- How?
  - Recall how the number of parameters is computed.
  - 1x1 convolution can be seen as channel-wise dimension reduction.

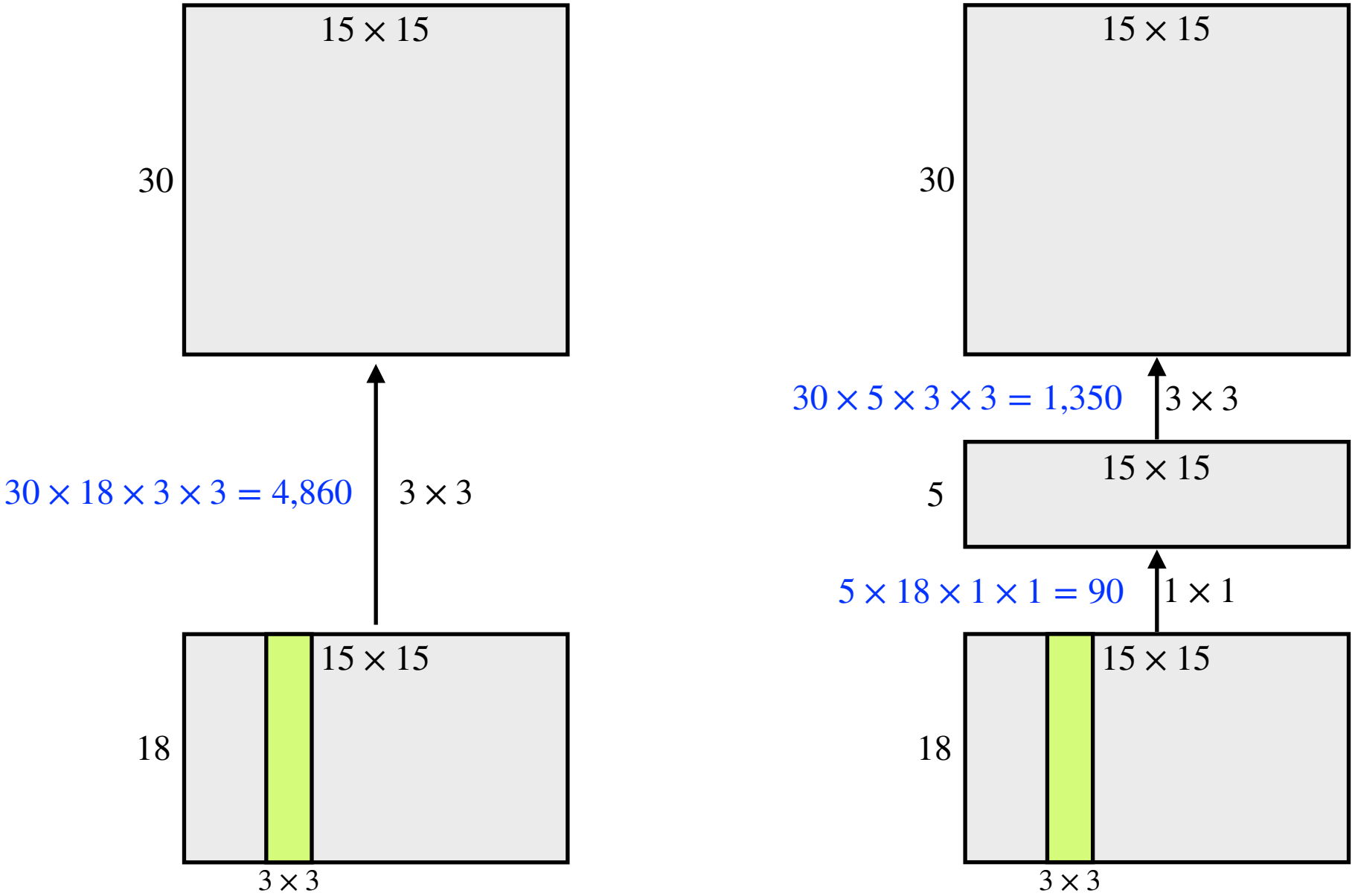


Why adding 1x1 Convolution?

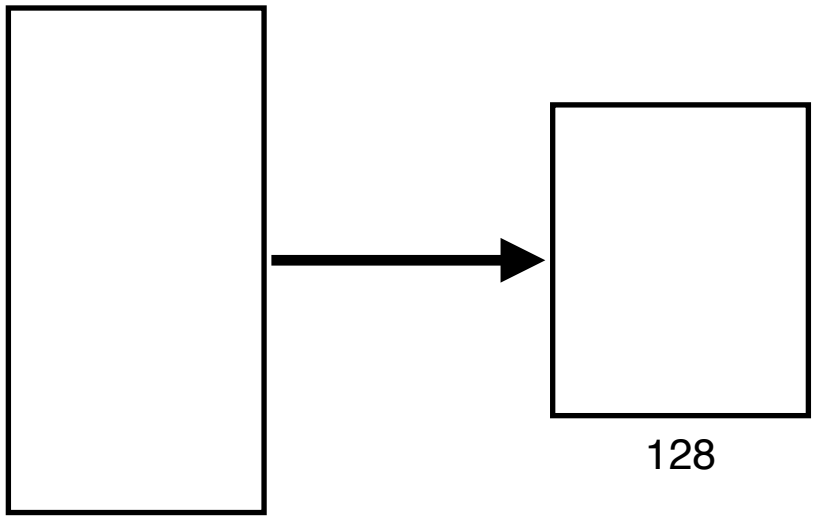




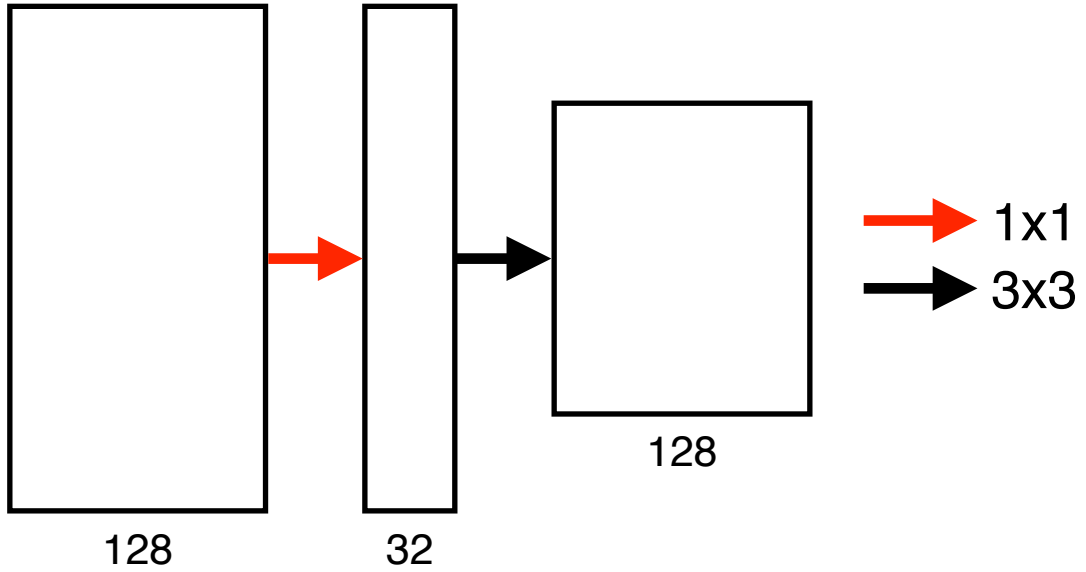
Why adding 1x1 Convolution?



Why adding 1x1 Convolution?



$$3 \times 3 \times 128 \times 128 = 147,456$$



$$1 \times 1 \times 128 \times 32 = 4,096$$

$$3 \times 3 \times 32 \times 128 = 36,864$$

$$4,096 + 36,864 = 40,960$$

1x1 convolution enables about 30% reduce of the number of parameters!

Quiz

- Which CNN architecture has the least number of parameters?
  - AlexNet (8-layers)
  - VGGNet (19-layers)
  - GoogLeNet (22-layers)

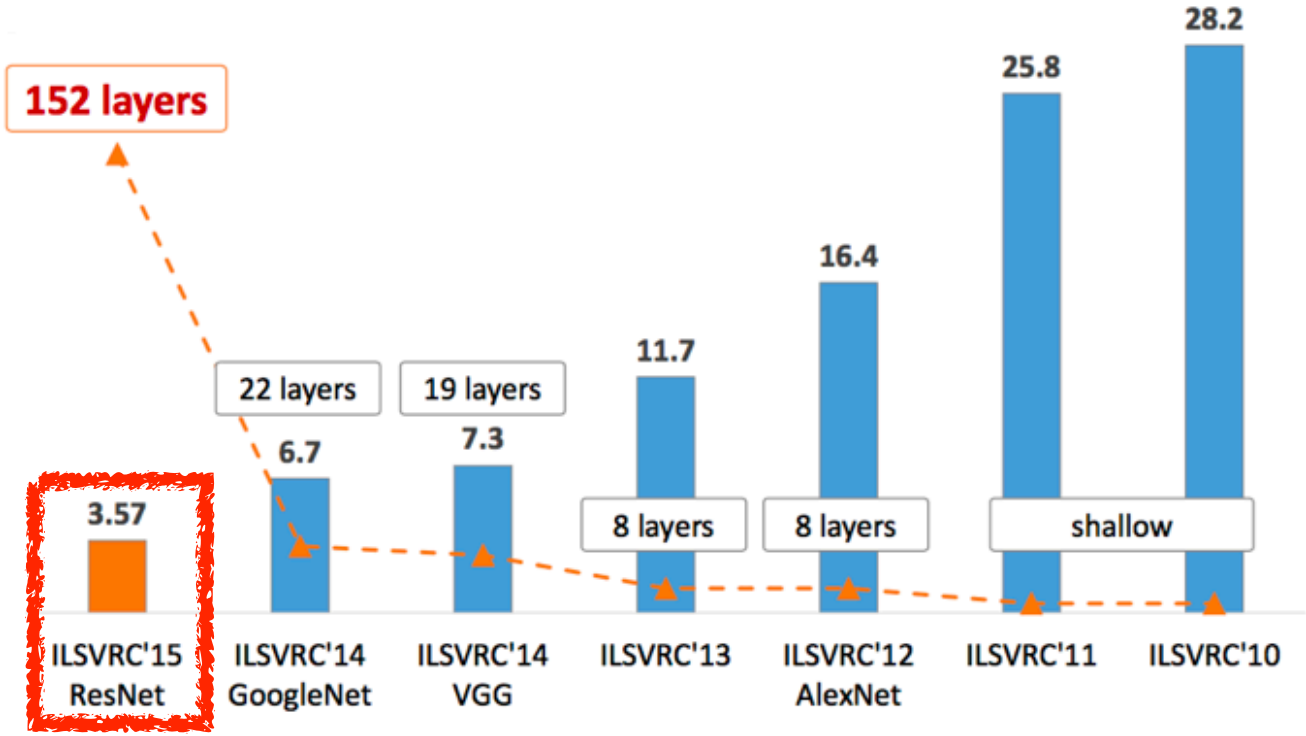
Quiz

- Which CNN architecture has the least number of parameters?
  - AlexNet (8-layers) (60M)
  - VGGNet (19-layers) (110M)
  - GoogLeNet (22-layers) (4M)

- The answer is GoogLeNet
  - 1x1 convolution can be seen as channel-wise dimension reduction.

ResNet

"Deep Residual Learning for Image Recognition," 2015



A degradation problem

- Deeper neural networks are hard to train.
  - Overfitting is usually caused by an excessive number of parameters.
  - But, the **degradation** is not caused by overfitting, and adding more layers leads to higher training errors (and also higher test errors).

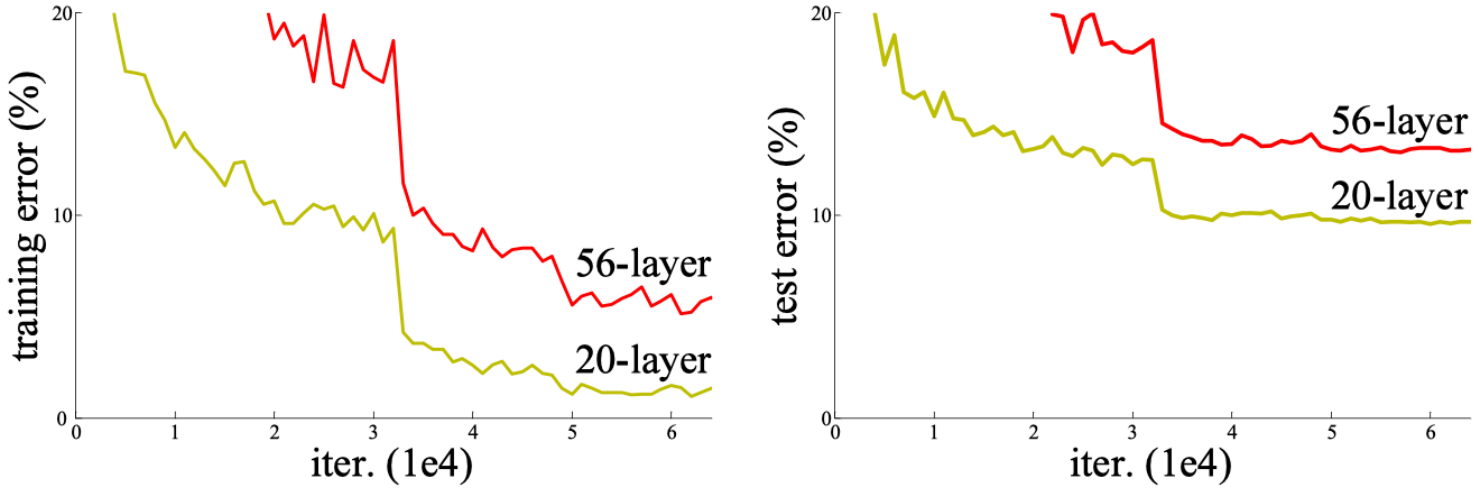
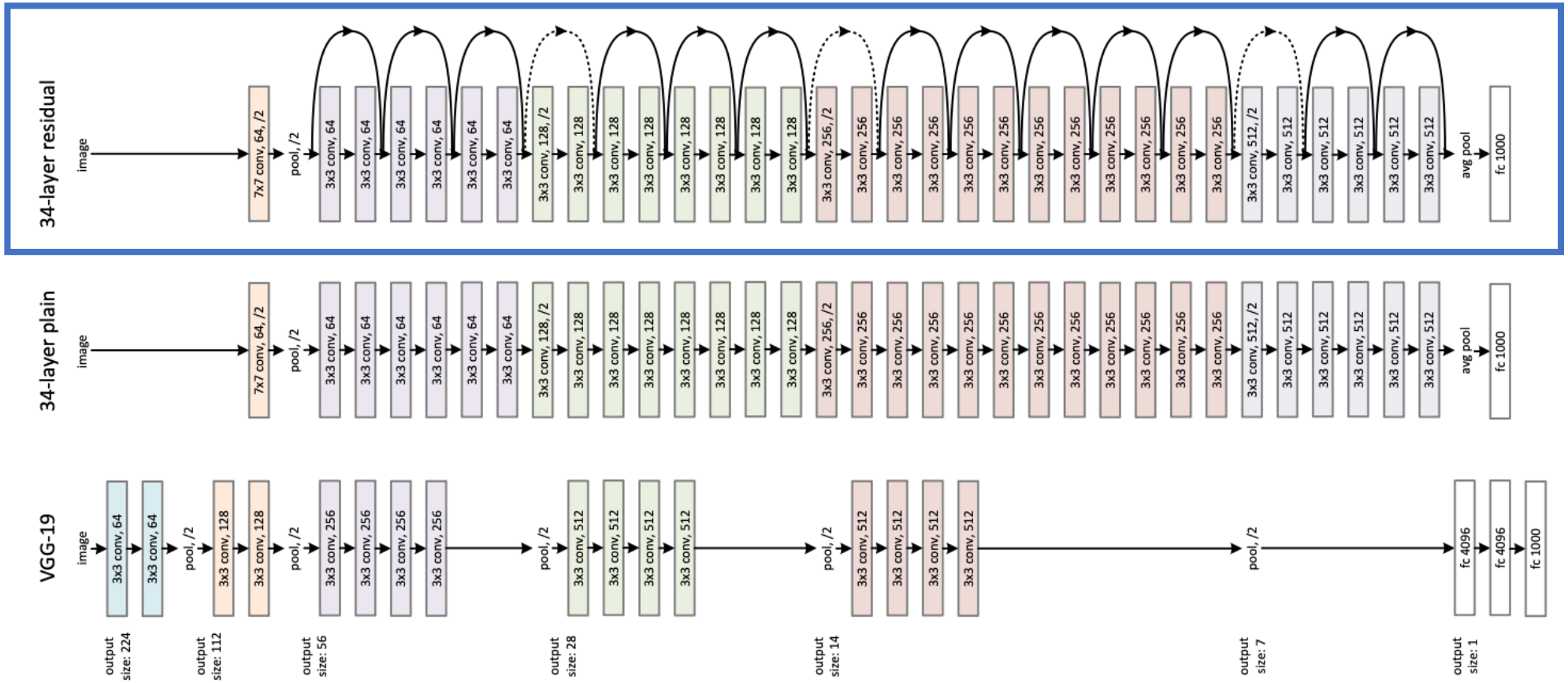


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

ResNet

- ResNet won the ILSVRC in 2015.
- The key idea is that **every additional layer** should contain the **identity function** as one of its elements.

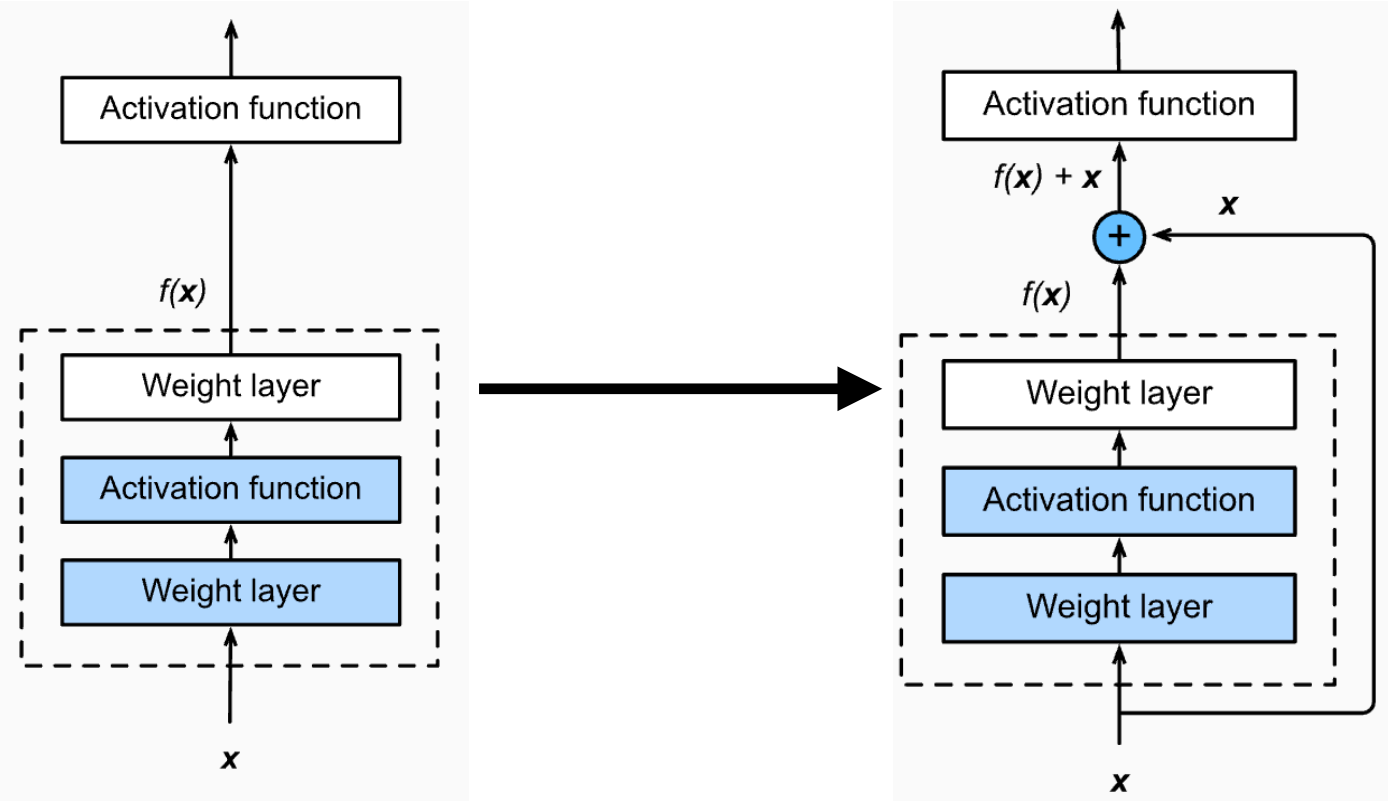


ResNet

We hypothesize that it is easier to optimize the **residual mapping** than to optimize the original mapping.

Residual Blocks

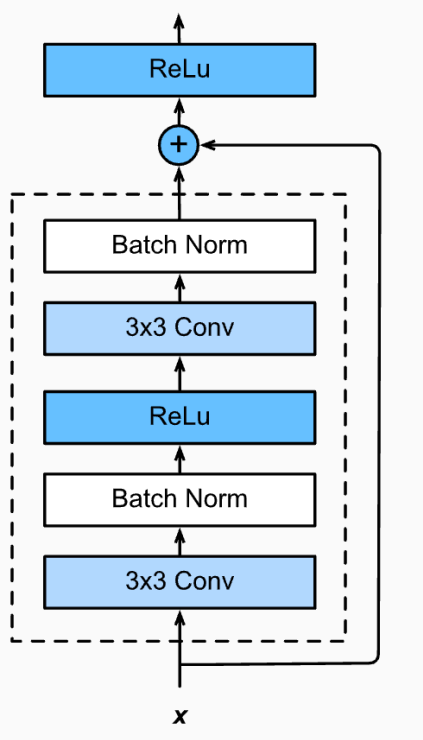
- Add an identity map (Shortcut connection)
- Shortcut allows the model to pass the input, and only learn the difference.



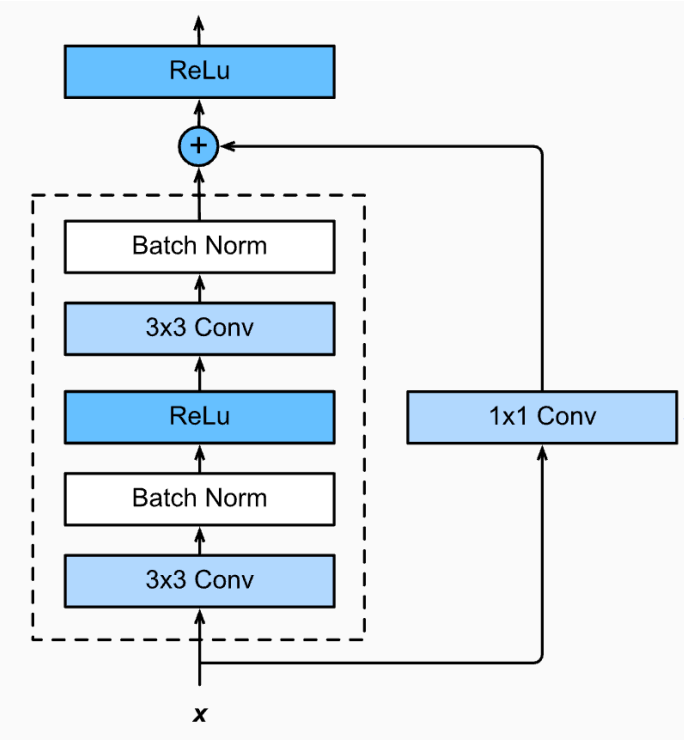
Shortcut connection:
$f(x) \rightarrow x + f(x)$

Residual Blocks

- Add an identity map after nonlinear activations.
- The projected shortcut uses a 1x1 convolution to match the number of channels.



Simple Shortcut

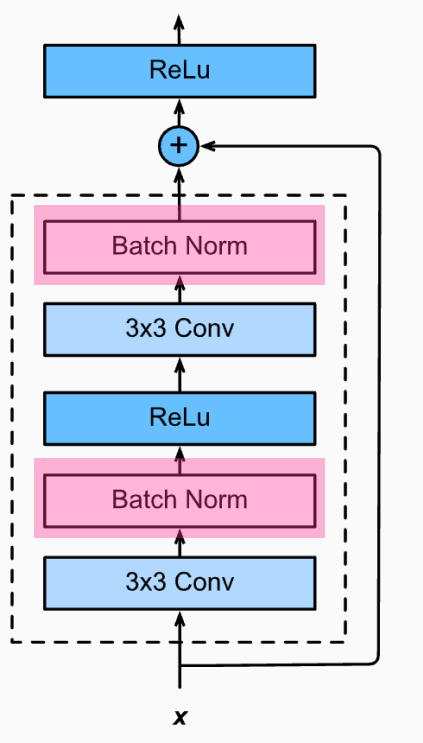


1x1 convolution to match the channel depth

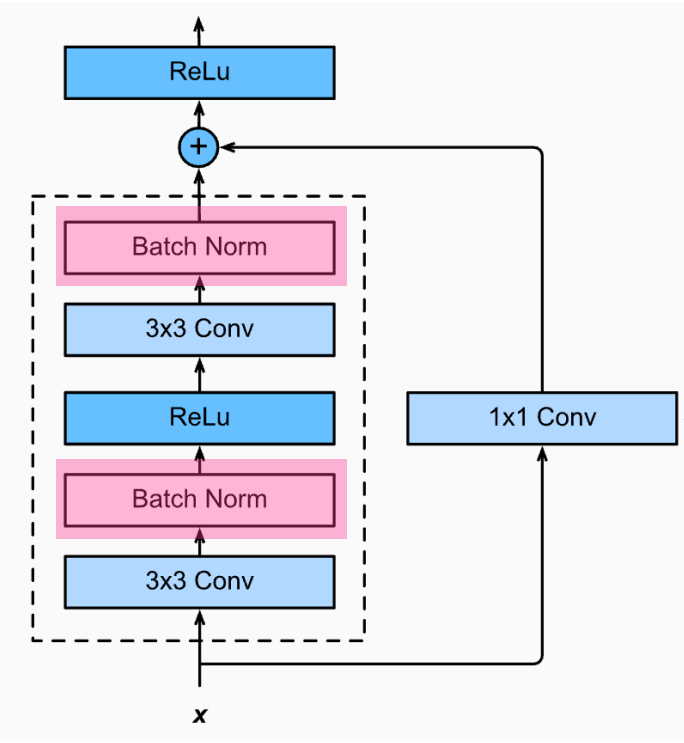
Projected Shortcut

Residual Blocks

- Batch normalization after convolutions.



Simple Shortcut

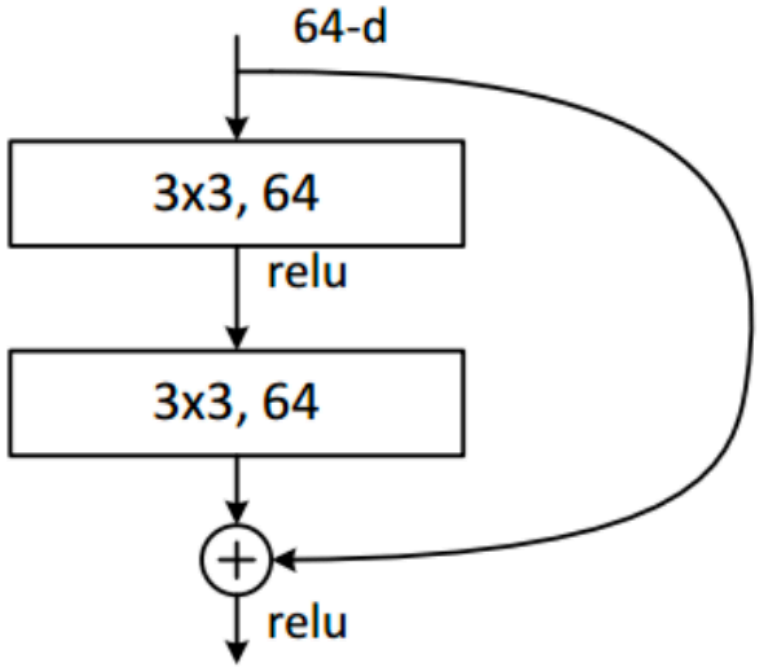


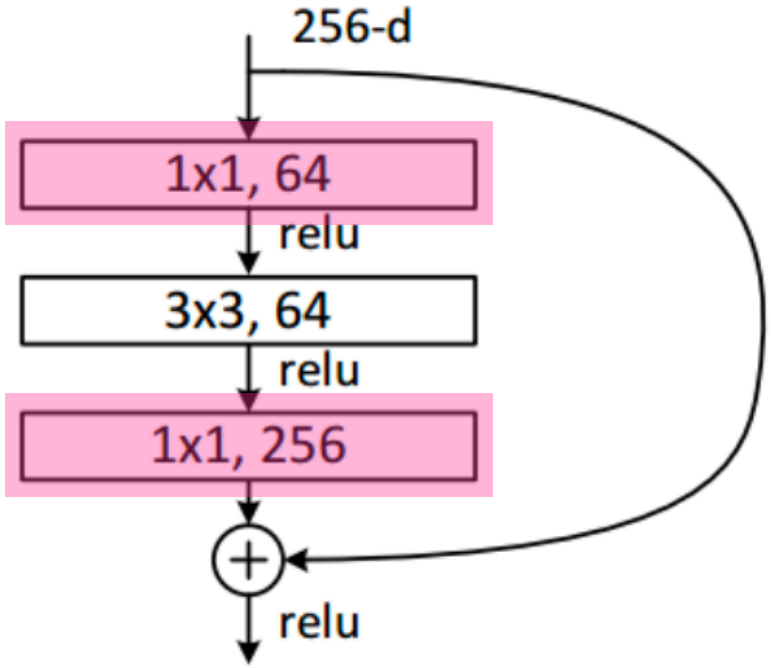
Projected Shortcut

1x1 convolution to match the channel depth

Bottleneck Blocks

- Similar to GoogLeNet's Inception architecture.
- 1x1 convolution can be seen as channel-wise dimension reduction.





ImageNet Results

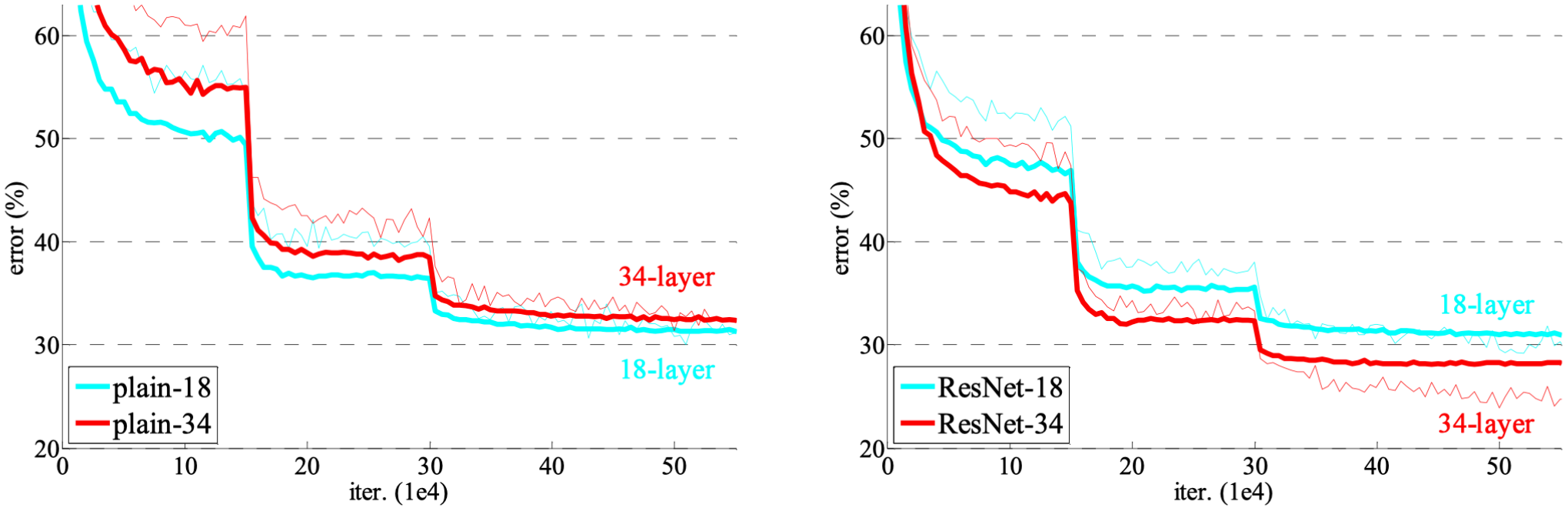
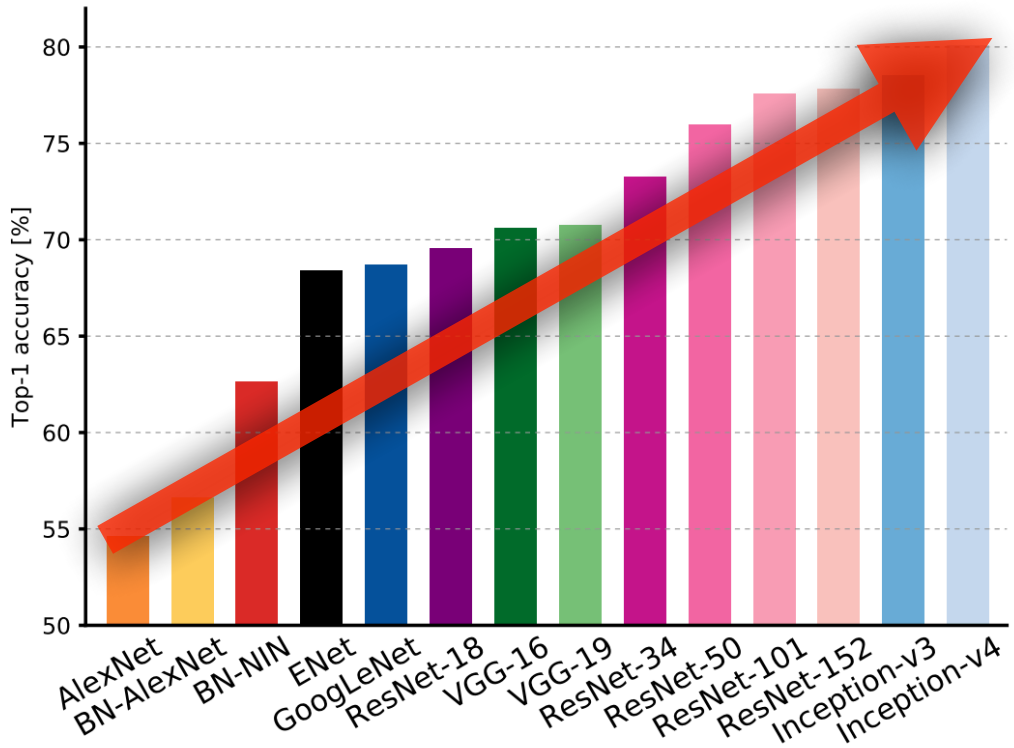
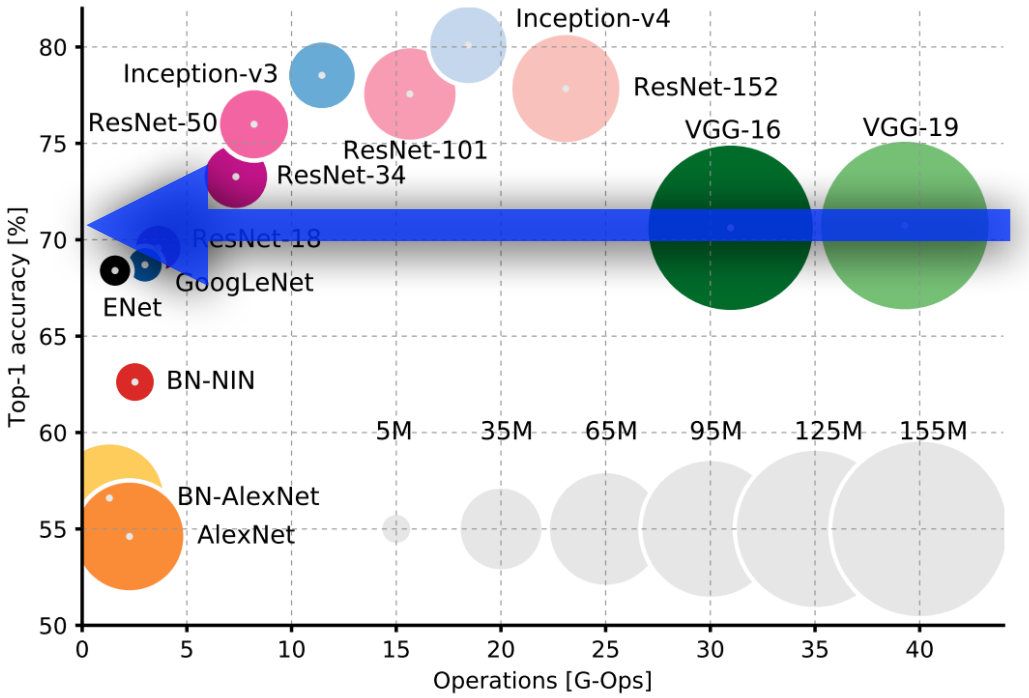


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Conclusion

- **Performance** increases while **parameter size** decreases.






서울과학기술대학교
SEOUL NATIONAL UNIV. OF SCIENCE & TECHNOLOGY
1910