Neural Networks:

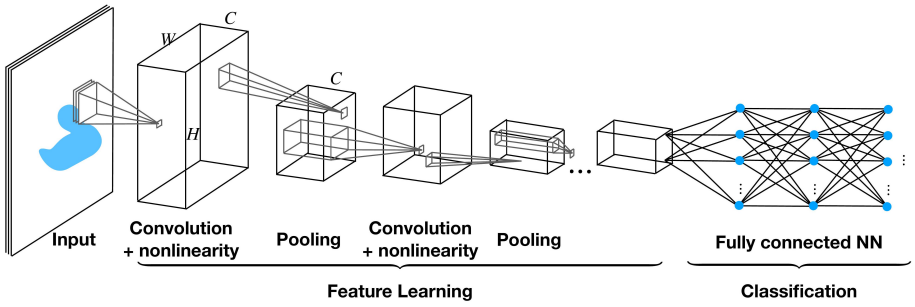
ImageNet Dimension:
$256 \times 256 \times 3 \sim 2 \cdot 10^5$

Convolutional Nets, Regularization, Data augmentation, Dropout

EPFL

Convolutional Networks

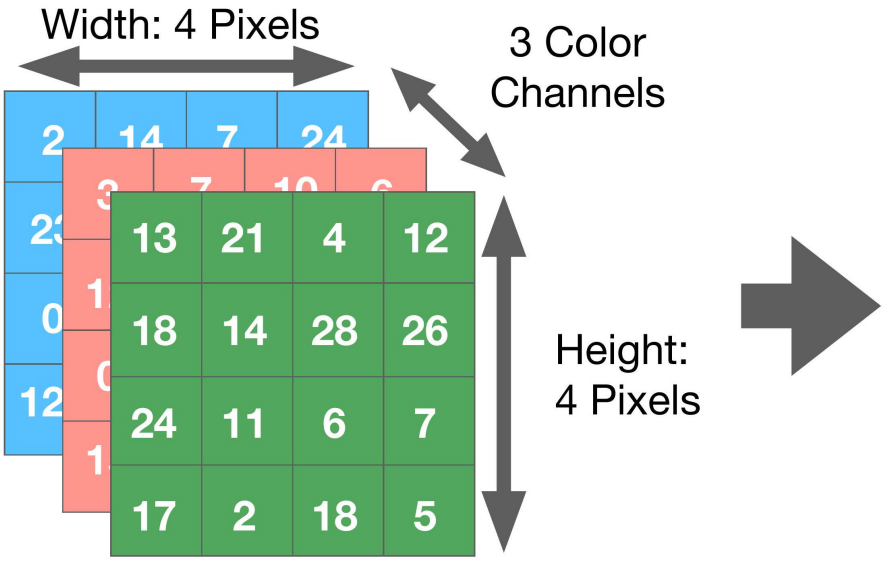
Convolutional NNs: General Structure



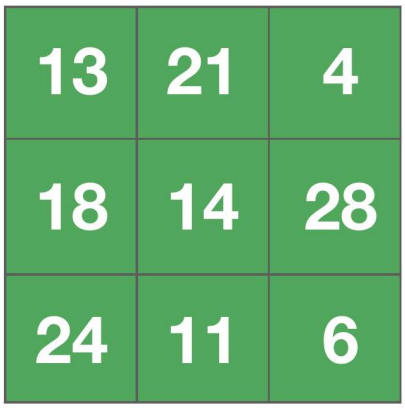
- Convolutional networks consist of sparsely connected convolutional layers in place of fully-connected linear layers
- Pooling layers perform spatial downsampling (typically typically reducing the dimensions from $H \times W$ to $H/2 \times W/2$
- A fully-connected network at the end performs classification based on the extracted features

Fully connected NNs have many parameters and do not capture spatial dependencies

- Fully connected NNs have $O\left(K^2 L\right)$ parameters: training requires a lot of data



- Fully connected neural networks interpret an image as a flattened vector,



disregarding the original spatial dependenc

13 | 21 | 4 | 18 | 14 | 28 | 24 | 11 | 6

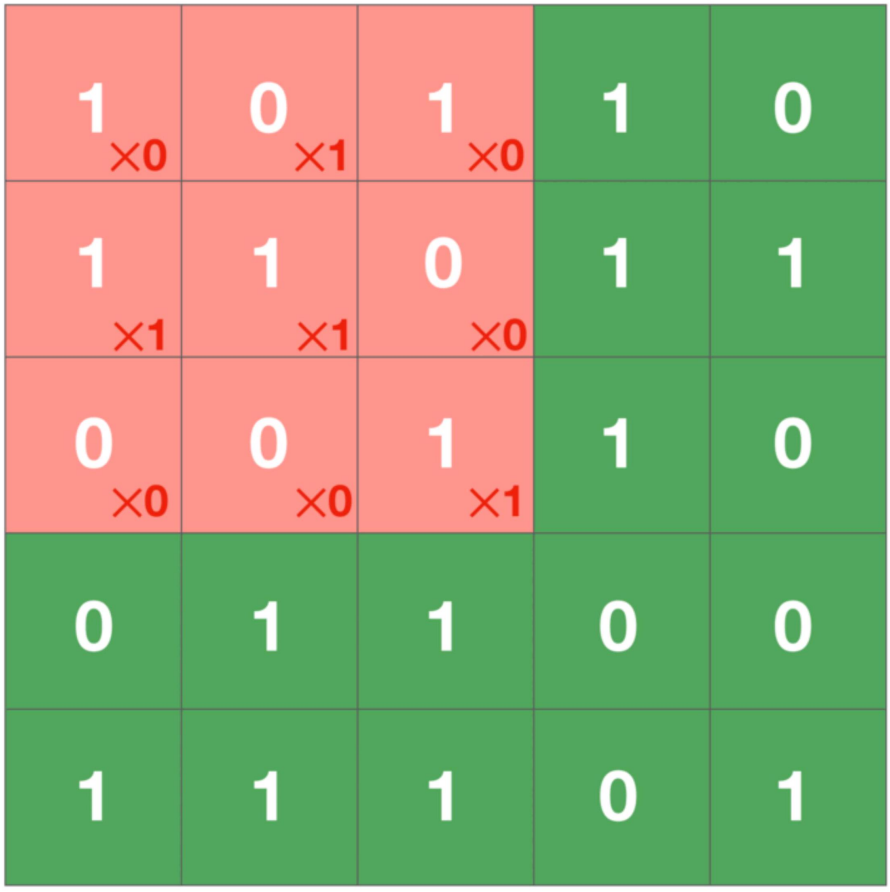
Convolution

$$x^{(1)}_{n,m} = \sum_{k,l} f_{k,l} \cdot x^{(0)}_{n-k,m-l}$$

- We consider local filters: $f_{k,l} \neq 0$ for small values of $|k|$ and $|l|$

$\Rightarrow x^{(1)}_{n,m}$ only depends on the value of $x^{(0)}$ close to (n, m)
$\Rightarrow f$ represents the learnable weights

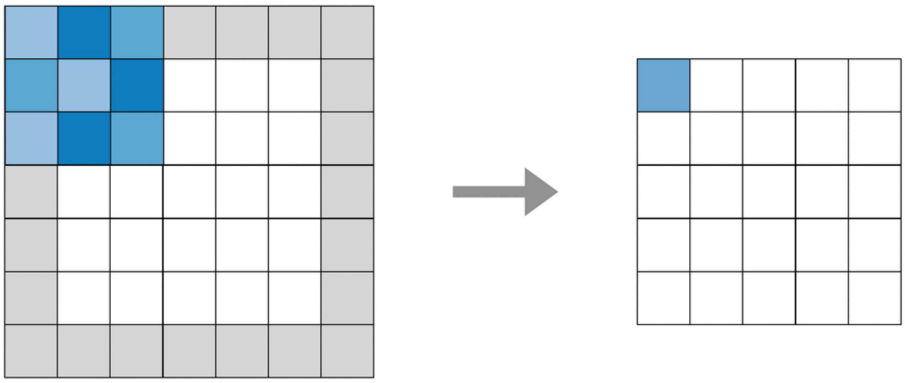
- We use the same filter at every position - weight sharing
- Translation equivariance - a shifted input results in a shifted output



Image

⇒ Convolution requires fewer parameters which are universal across different locations

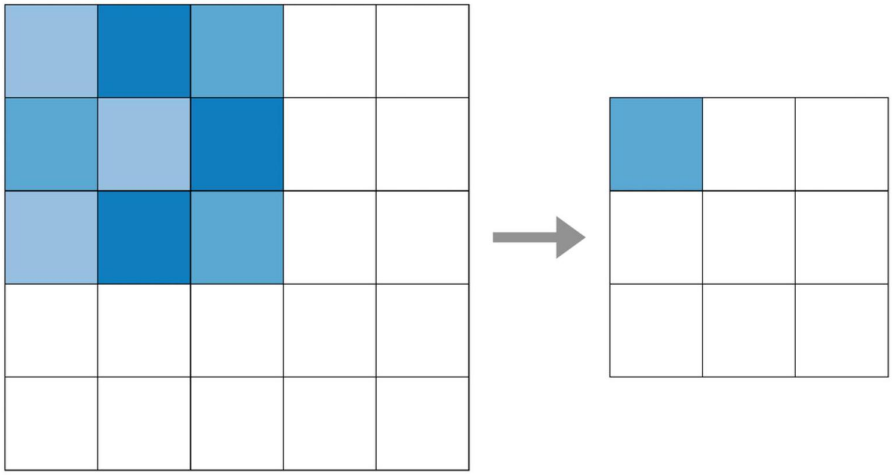
Handling of borders



Zero padding:

Add zeros to each side of the input's boundaries

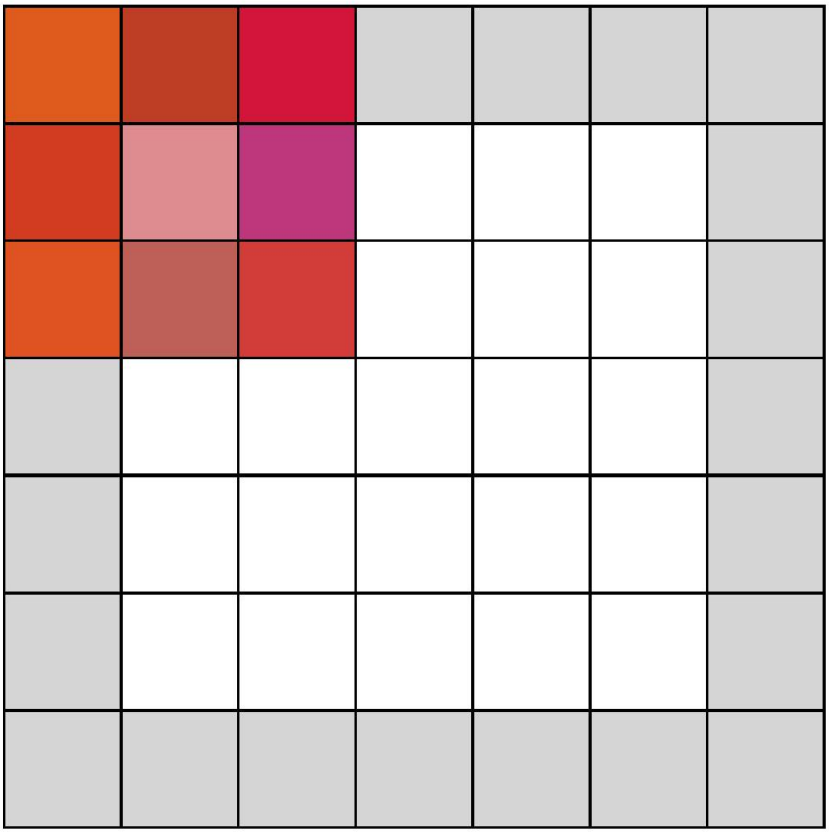
⇒ The convolved feature has the same dimension as the input Valid padding:



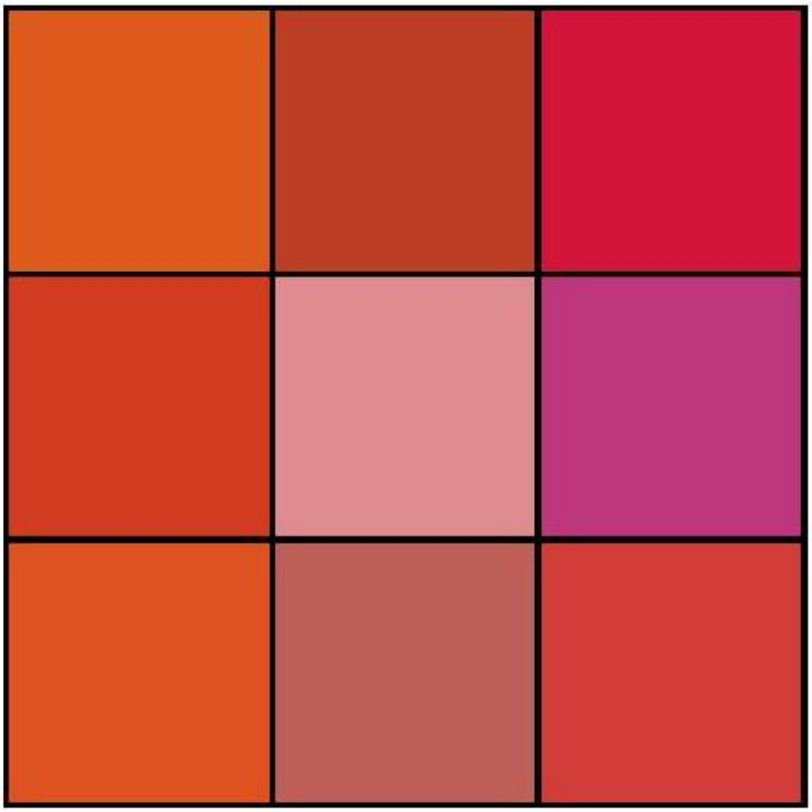
Perform the convolution only where the entire filter fits inside the original data

⇒ The convolved feature has smaller dimensions than the input

Filter for Multi-Channel Convolution

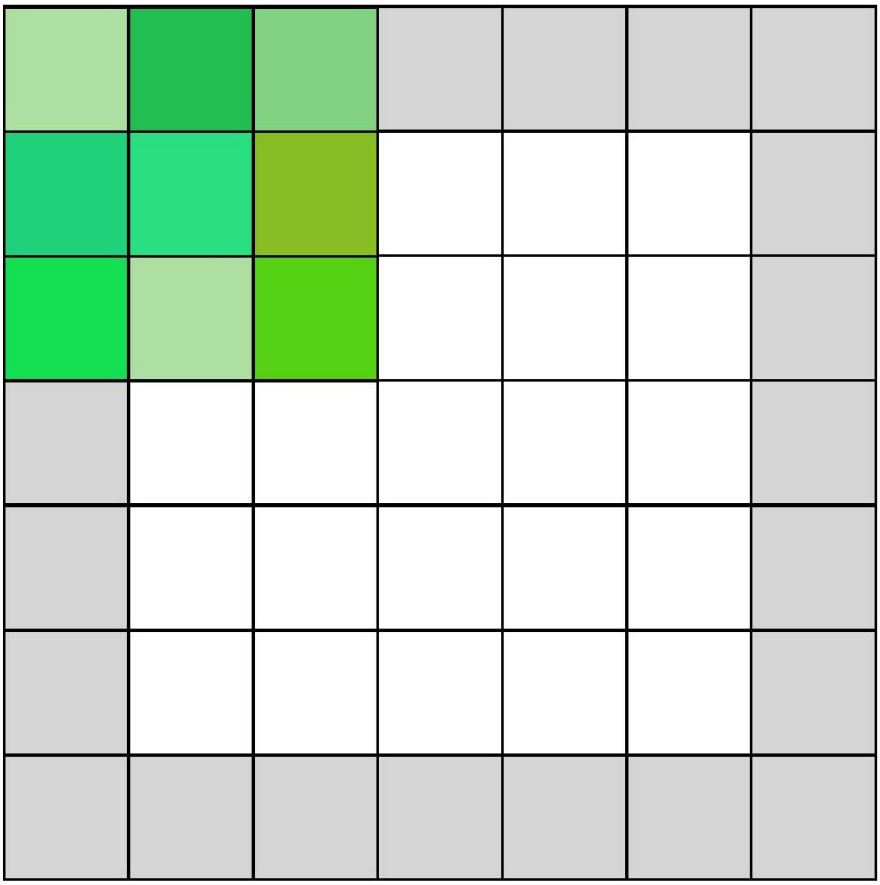


Input Channel #1

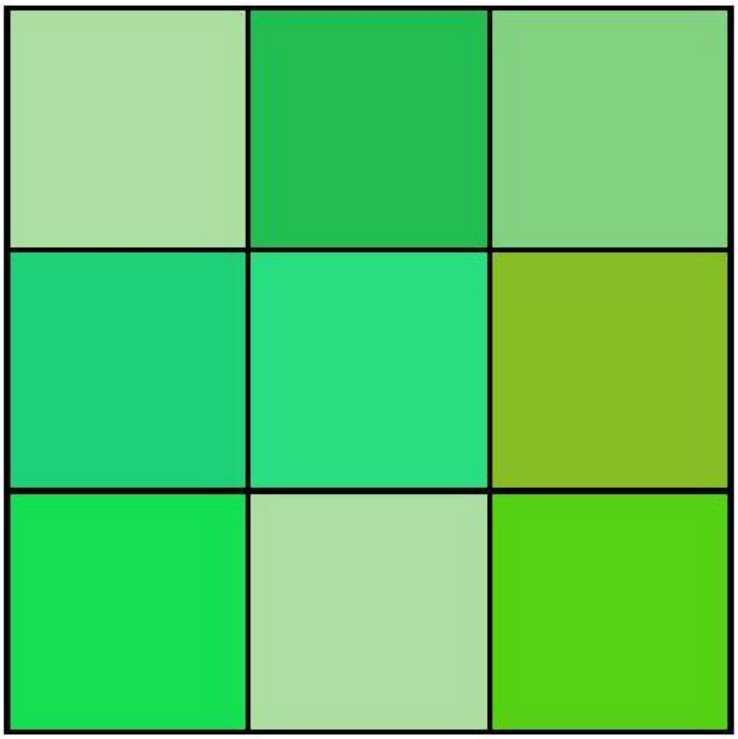


Filter Channel #1

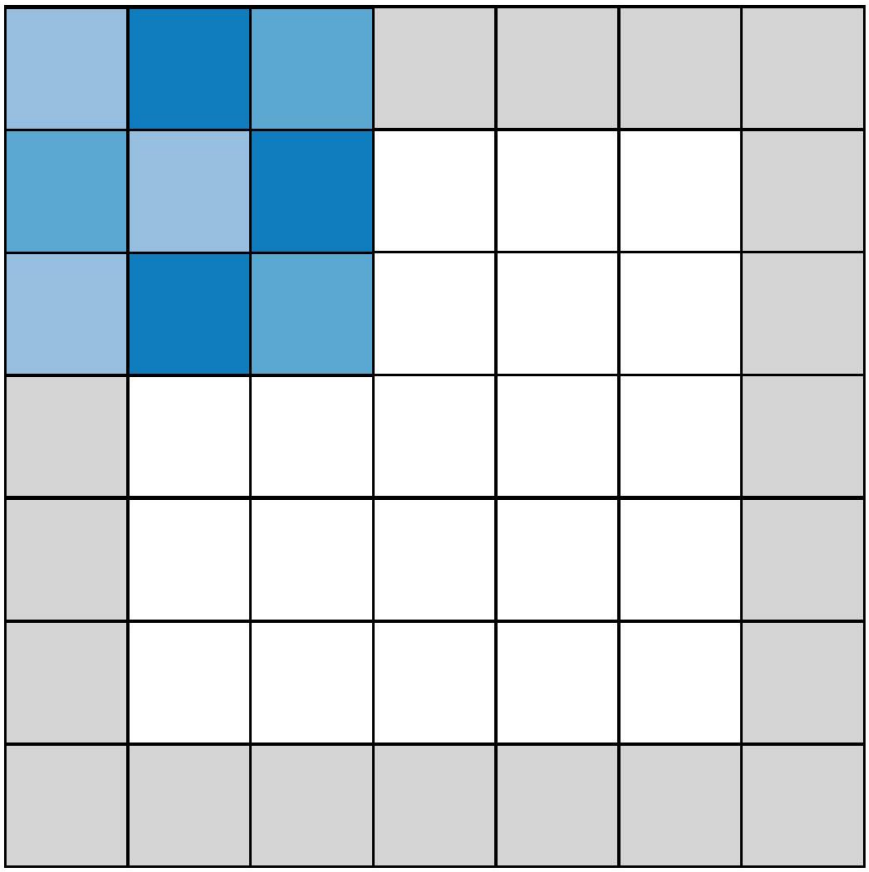
120



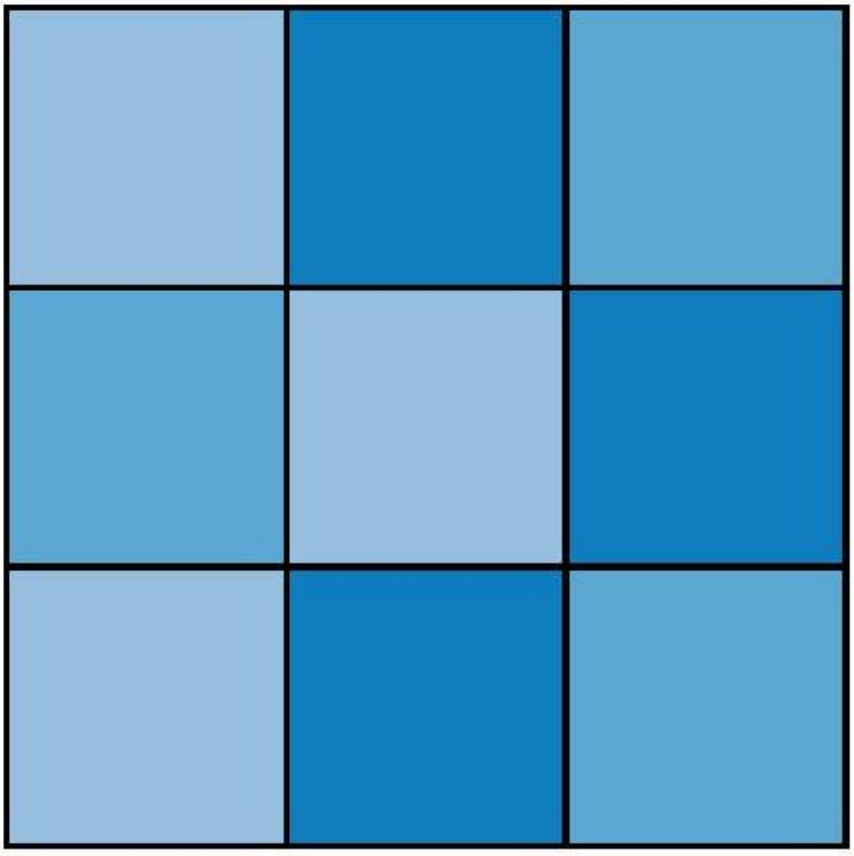
Input Channel #2



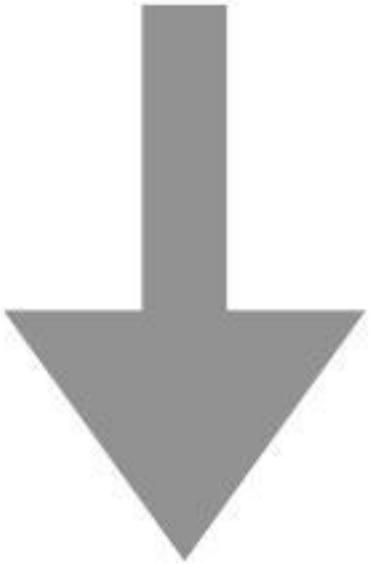
Filter Channel #2



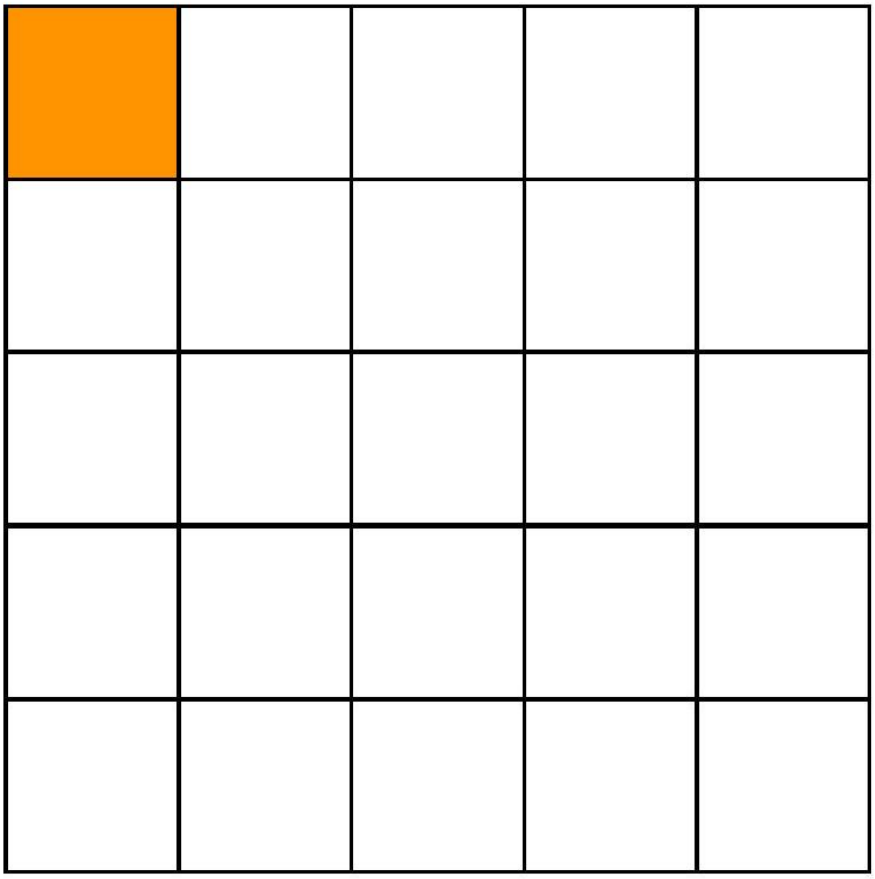
Input Channel #3



Filter Channel #3



205

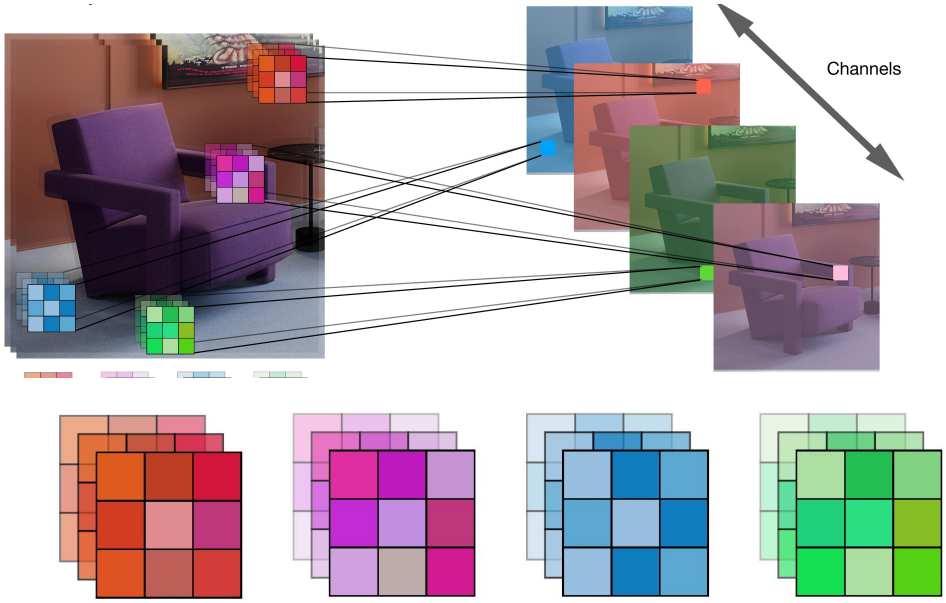


One Channel Output
260

- For multi-channel inputs, the filter has the same number of channels as the input
- The filter channels and the bias are the learnable parameters of the filter

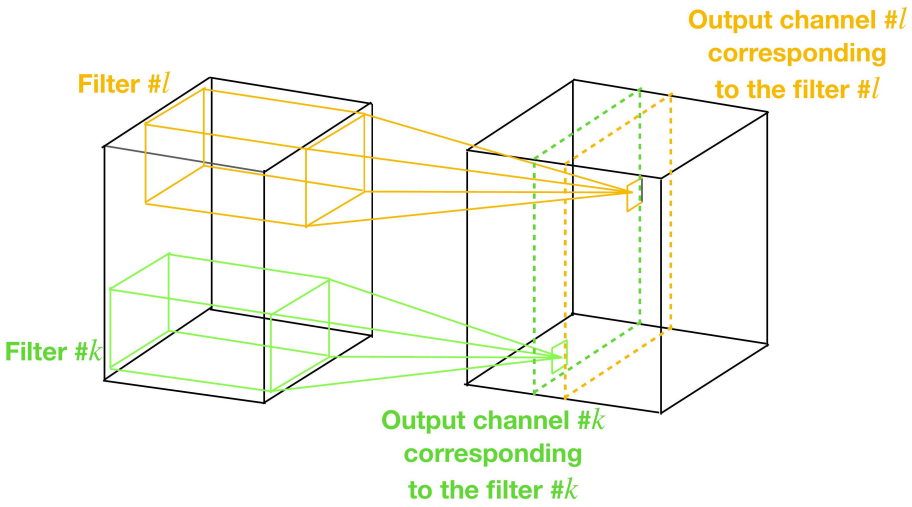
Multi-Channel Output from Multiple Filters

It is common to use multiple filters. Each filter processes the input to produce a separate channel



Filters Applying different filters to an input generates multiple output channels.

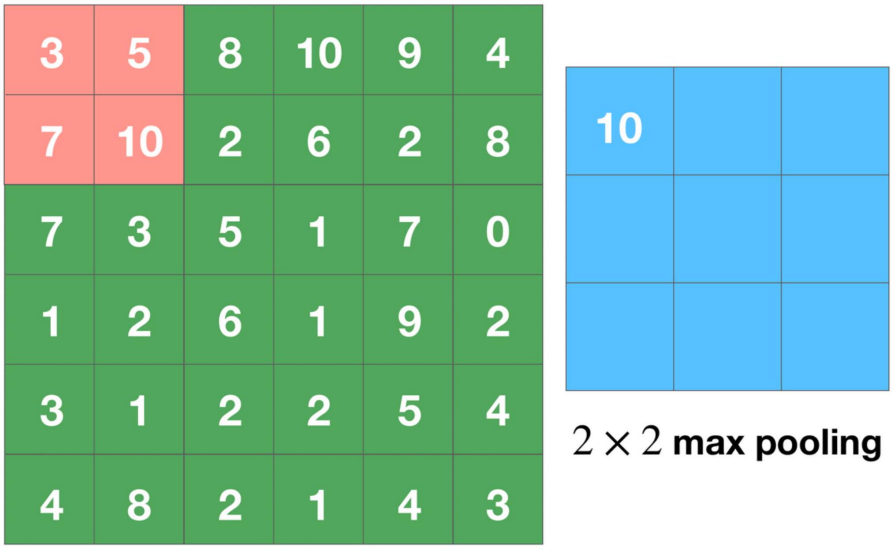
Convolutional Layer



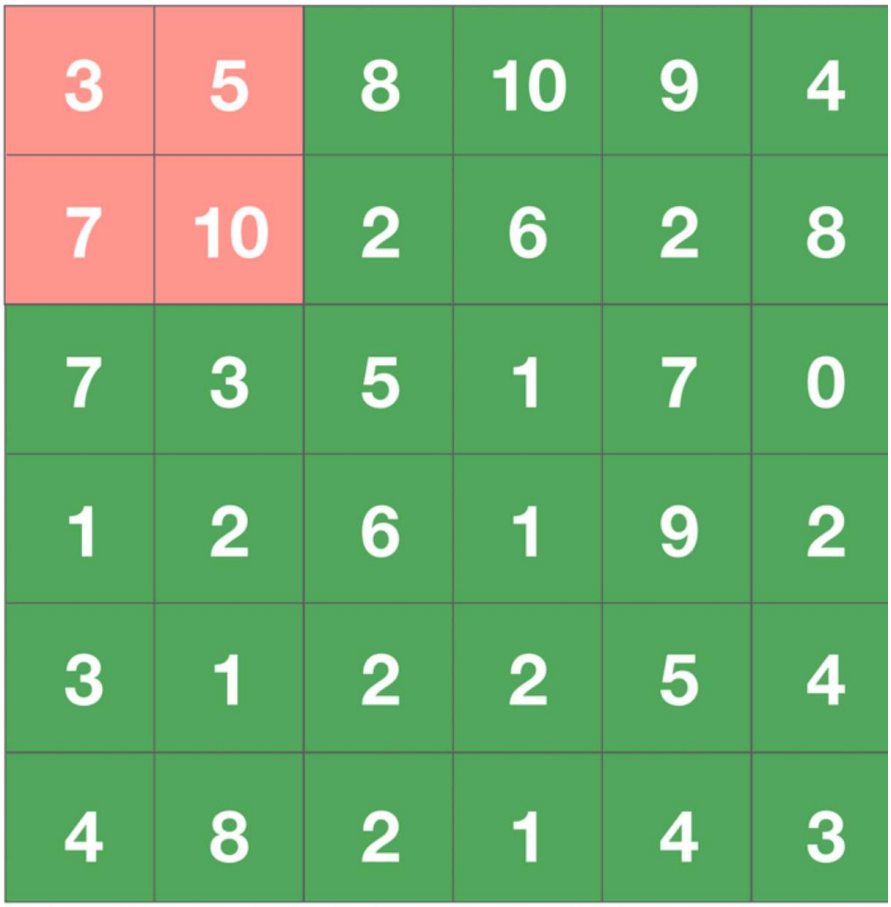
- A convolutional layer is composed of multiple filters
- Each output channel corresponds to its own independent filter
- Hyper-parameters of the convolutional layer: size, padding, stride

Pooling: often applied after the convolutional layer

Max pooling: returns the maximum value of the portion of the convolved feature that is covered by the kernel Average pooling: returns the average value of the portion of the convolved feature that is covered by the kernel



Convolved feature



6.25

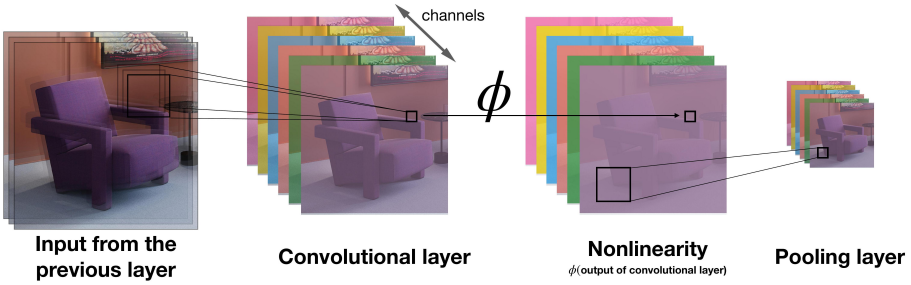
2×2 average pooling

Convolved feature

Pooling is a downsampling operation that reduces the spatial dimensions of the convolved feature

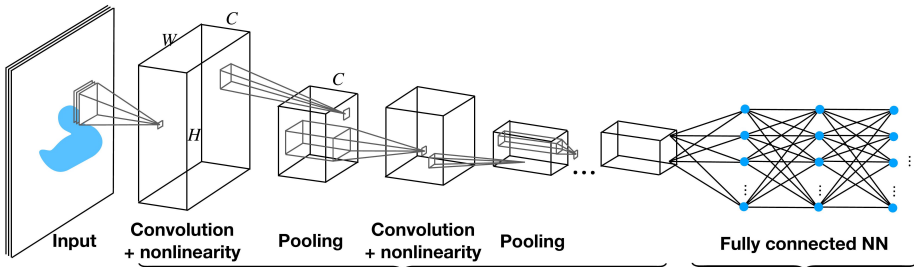
Remark: Pooling layers do not have learnable parameters Hyperparameters are the size, type, and stride of the pooling operation

Non-linearity and convolutional NNs



Important: A non-linearity such as ReLU is included after each convolutional layer to make the model non-linear

Convolutional NNs: General Structure



Feature Learning
Classification
Receptive field (area of input that affects a given neuron) increases with depth:

- First layers extract low-level features, e.g., edges, colors
- Subsequent layers extract high-level features, e.g., objects

⇒ ConvNet reduces the images to a form easier to process without losing essential features

Backpropagation with weight sharing

Weight sharing is used in CNN: many edges use the same weights

Training:

1. Run backpropagation as if the weights were not shared (treat each weight as an independent variable)
2. Once the gradient is computed, sum the gradients of all edges that share the same weight

Why: let $f(x, y, z) : \mathbb{R}^3 \to \mathbb{R}$ and $g(x, y) = f(x, y, x)$

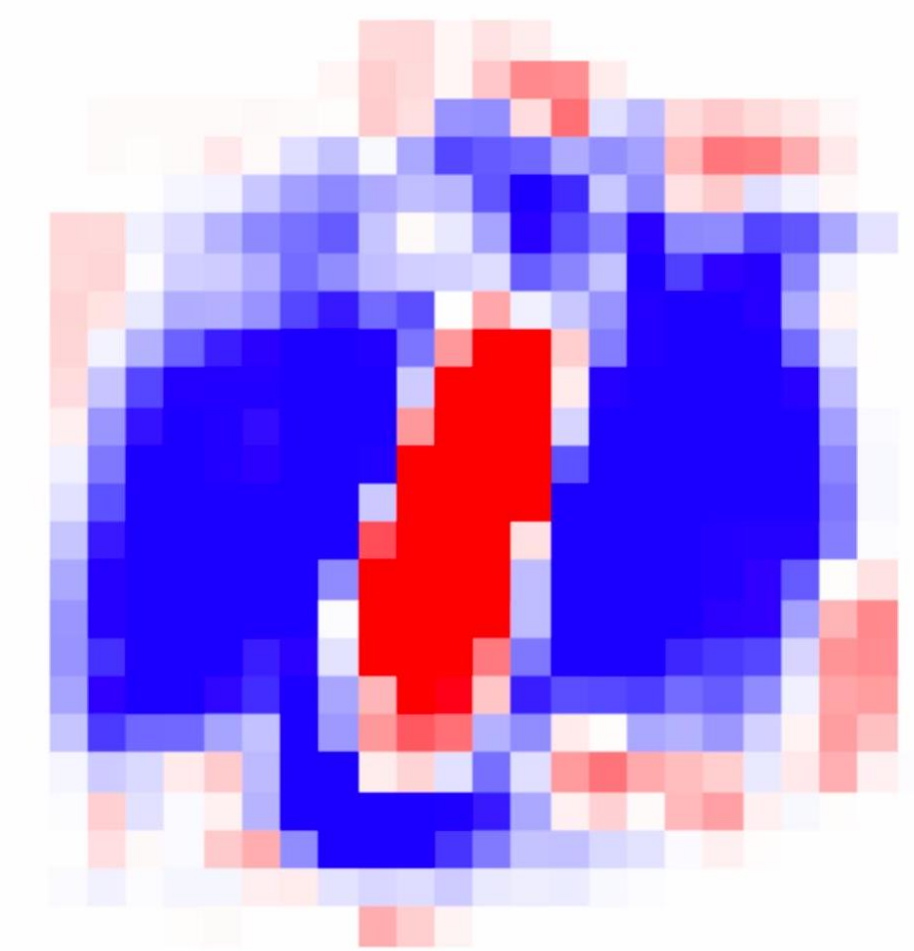
$$\left(\frac{\partial g}{\partial x}(x,y), \frac{\partial g}{\partial y}(x,y)\right) \underset{\text{Chain rule}}{=} \left(\frac{\partial f}{\partial x}(x,y,x) + \frac{\partial f}{\partial z}(x,y,x), \frac{\partial f}{\partial y}(x,y,x)\right)$$

What do ConvNets learn?

Learned Convolutional Filters on MNIST

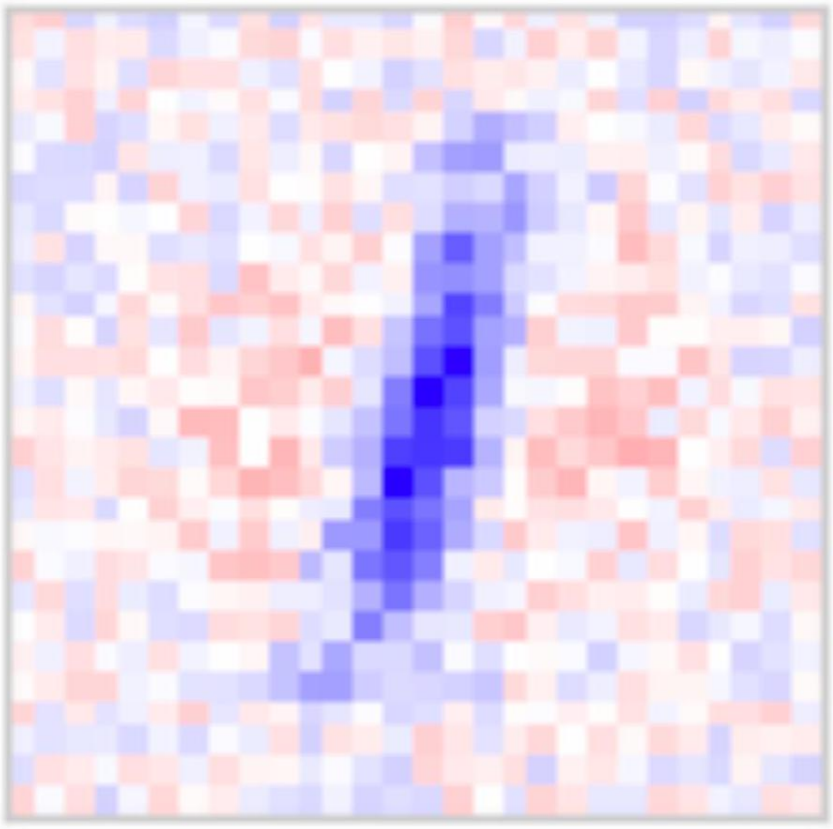
Linear model

parameter vector w

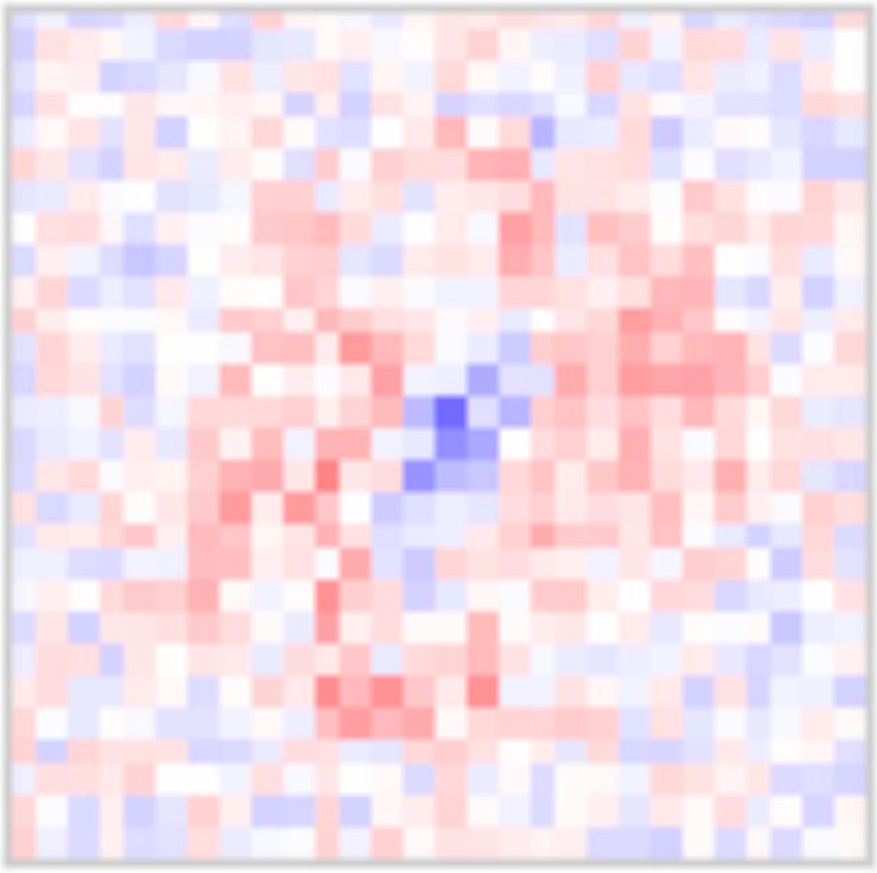


Fully-connected network

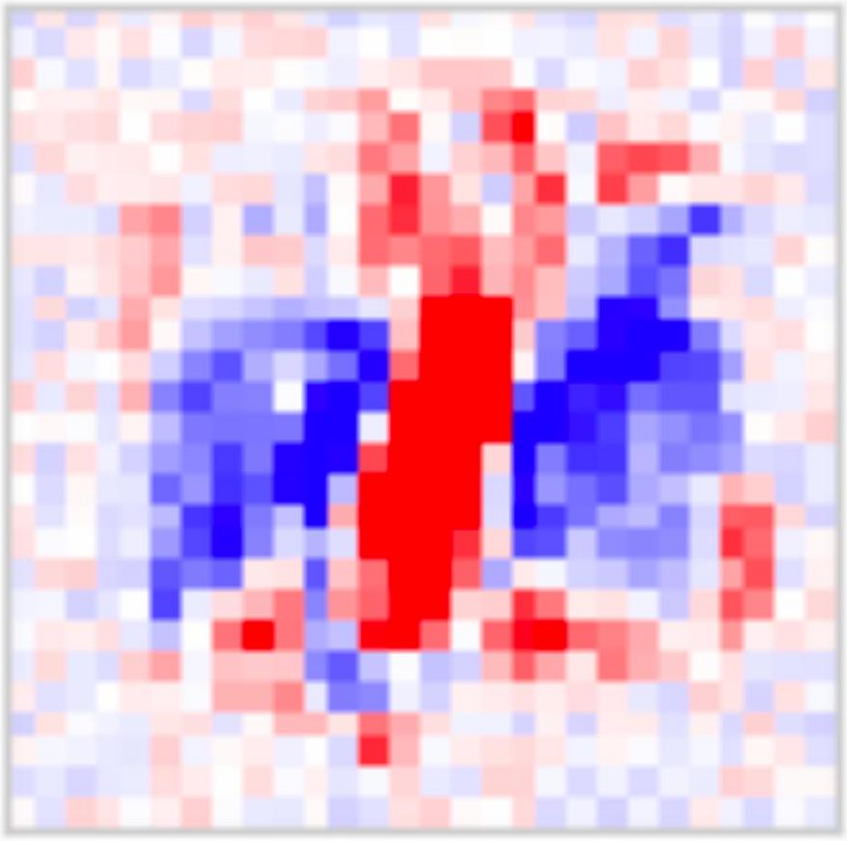
first-layer vector $w_{:,1}$



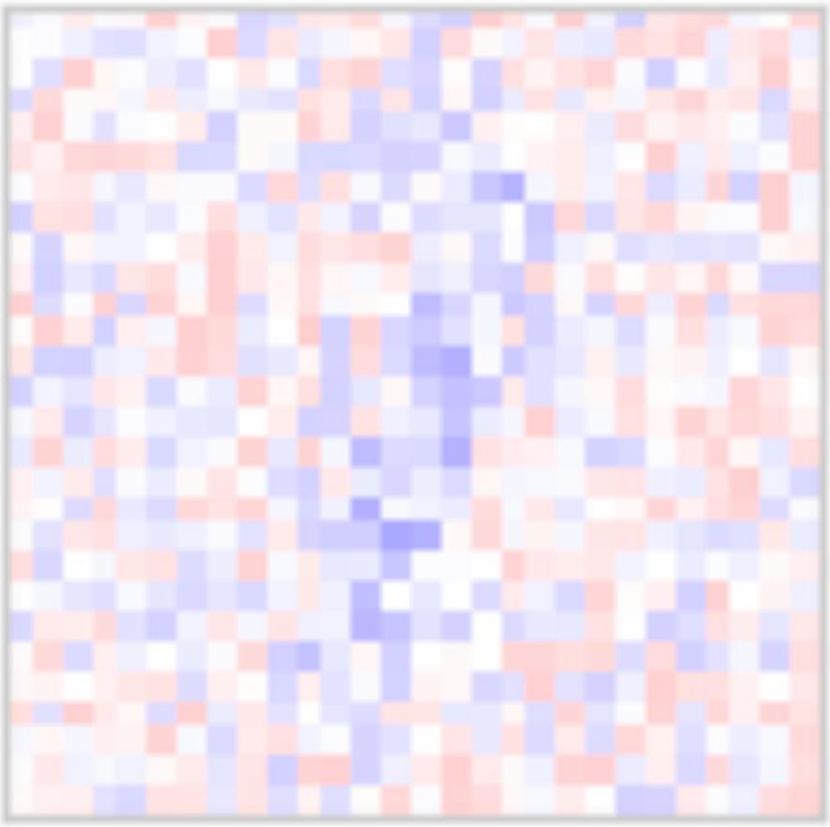
first-layer vector $w_{:,2}$



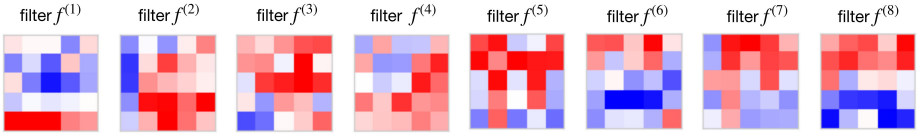
first-layer vector $w_{:,3}$



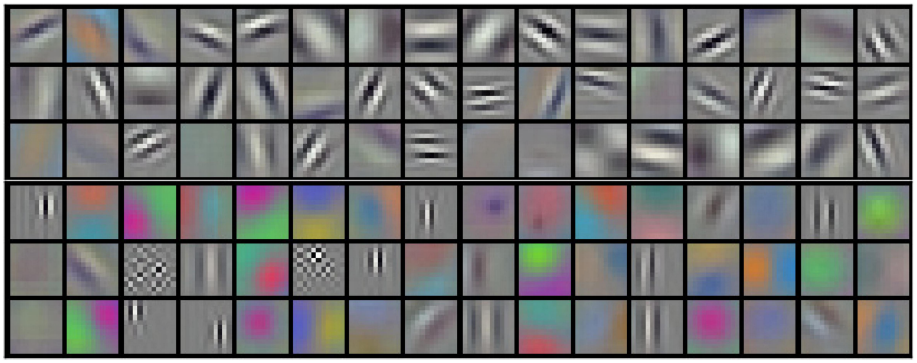
first-layer vector $w_{:,4}$



Convolutional networks



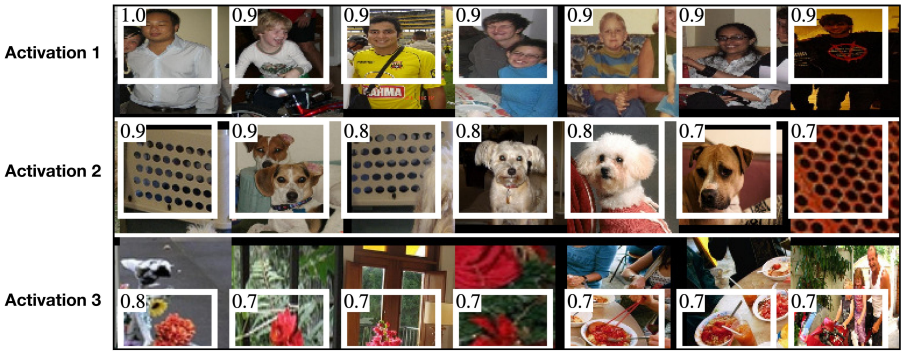
Learned Convolutional Filters on ImageNet



Source: ImageNet Classification with Deep Convolutional Neural Networks (NeurIPS 2012)

- Filters of the first layer can be interpretable
- Edge and color detectors typically emerge when trained on large datasets

Individual Activations Can Be Interpretable



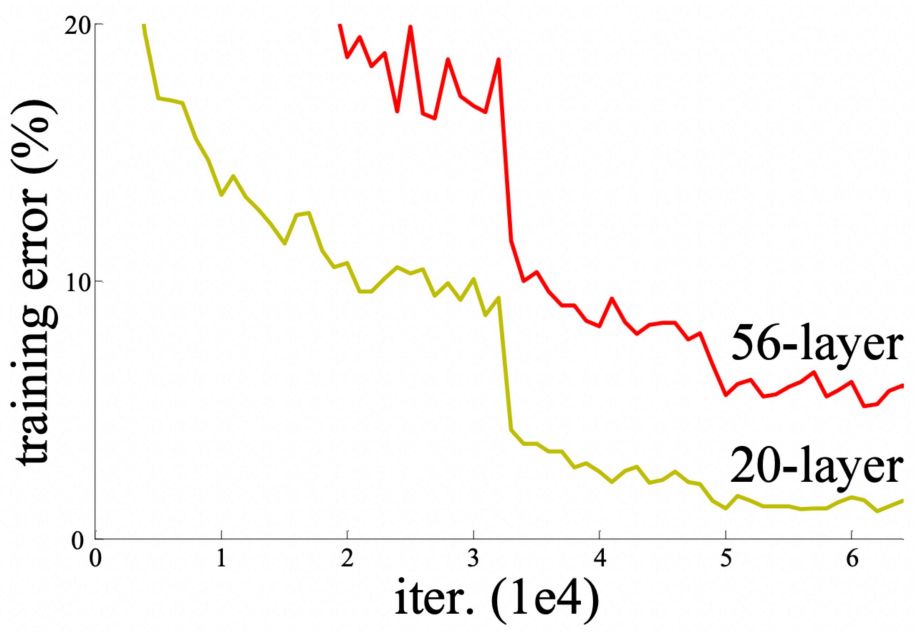
Source: Rich feature hierarchies for accurate object detection and semantic segmentation (CVPR 2014)

- Receptive fields and activation values are drawn in white
- Each activation detects some pattern or object (not always interpretable)
- Activations in later layers detect more complex patterns

Residual Networks

Skip Connections and Residuals

- Starting point: Adding more layers should lead to the same or lower training loss as they can learn the identity function
- ResNet paper indicates that this is not always the case
- Solution: add a skip connection around some layers $F(\mathbf{X})$
- Standard network: $\mathbf{Y} = F(\mathbf{X})$
- Residual network: $\mathbf{Y} = R(\mathbf{X}) + \mathbf{X}$ where $R(\mathbf{X})$ is called a residual branch

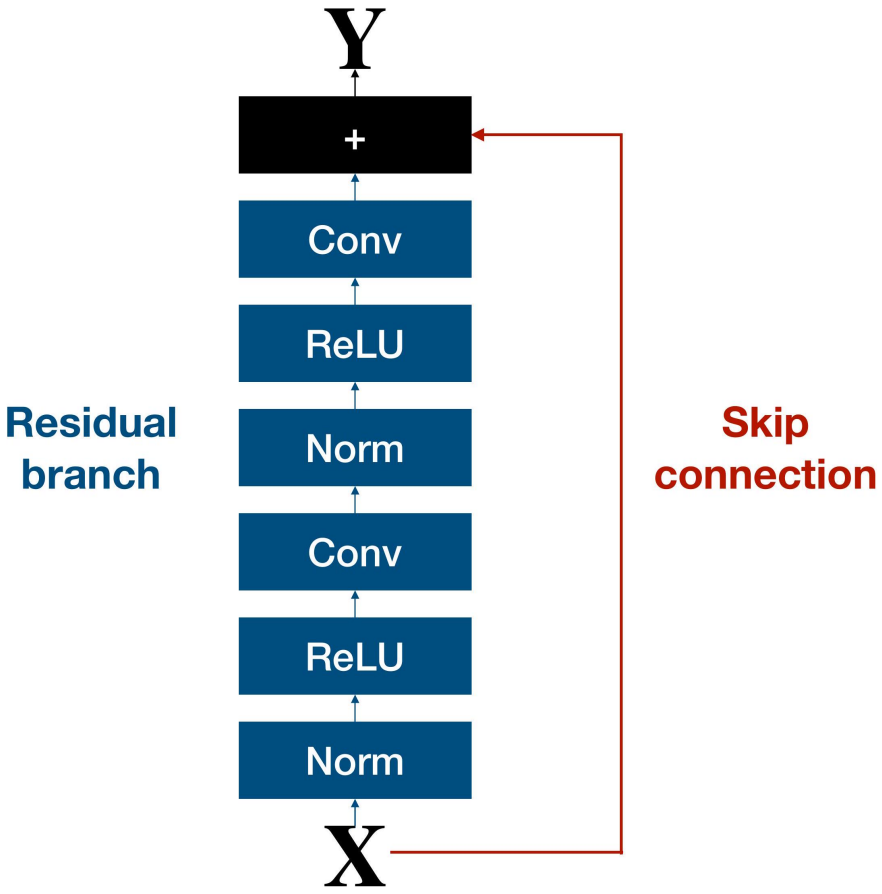


Observation from the ResNet paper: deeper CNNs are harder to train

Skip Connections and Residuals

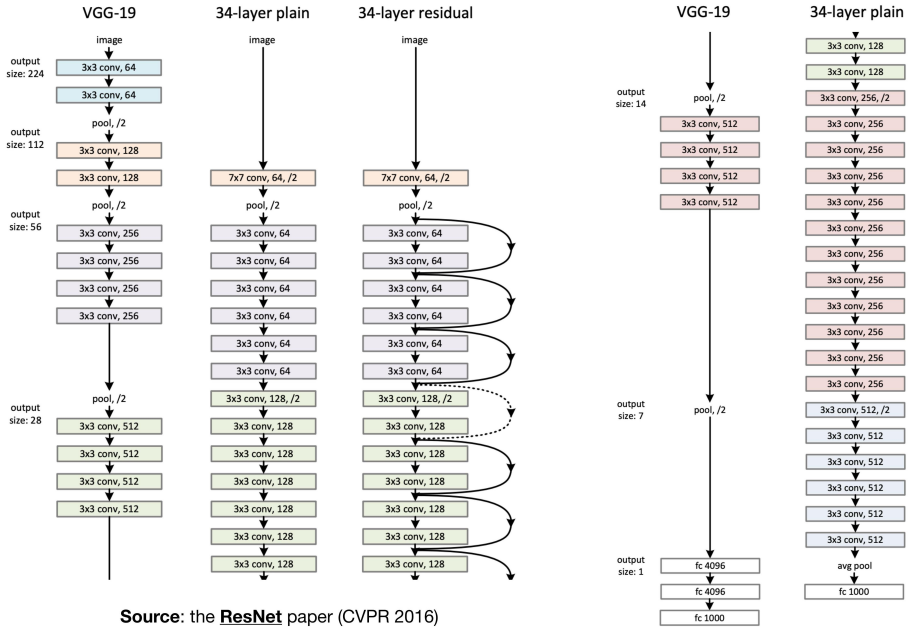
- Example of $R(\mathbf{X})$: see on the right
- Technical detail: If size($\mathbf{Y}$) $\neq$ size($\mathbf{X}$), additional operations are needed on the skip connection to match the dimensions
- Skip connections address the observed convergence issue, making the training of very deep networks (with hundreds of layers) feasible

- Skip connections are used in almost all modern neural networks (including CNNs and transformers)



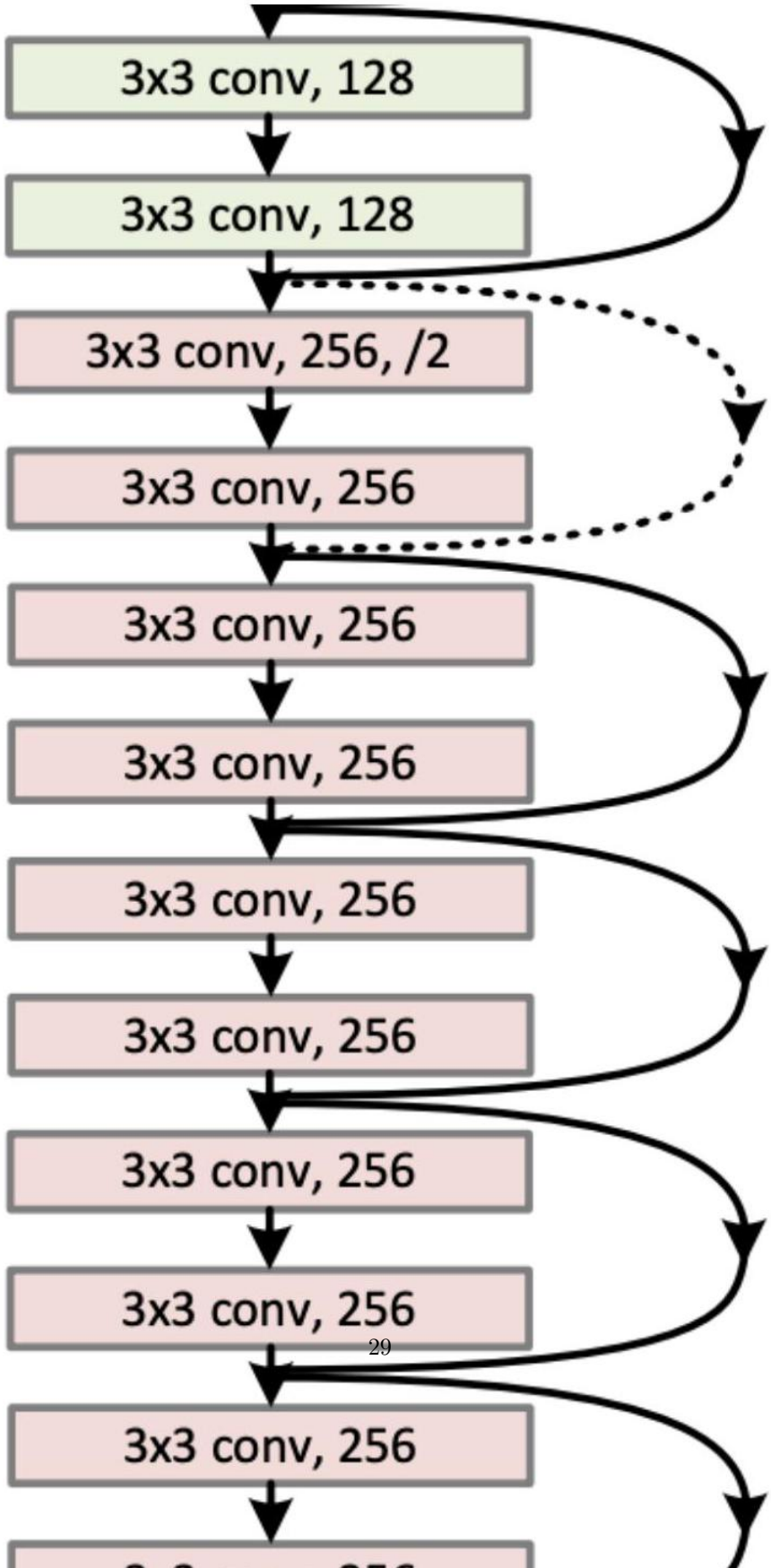
Example residual block

Popular architectures: VGG vs. ResNet



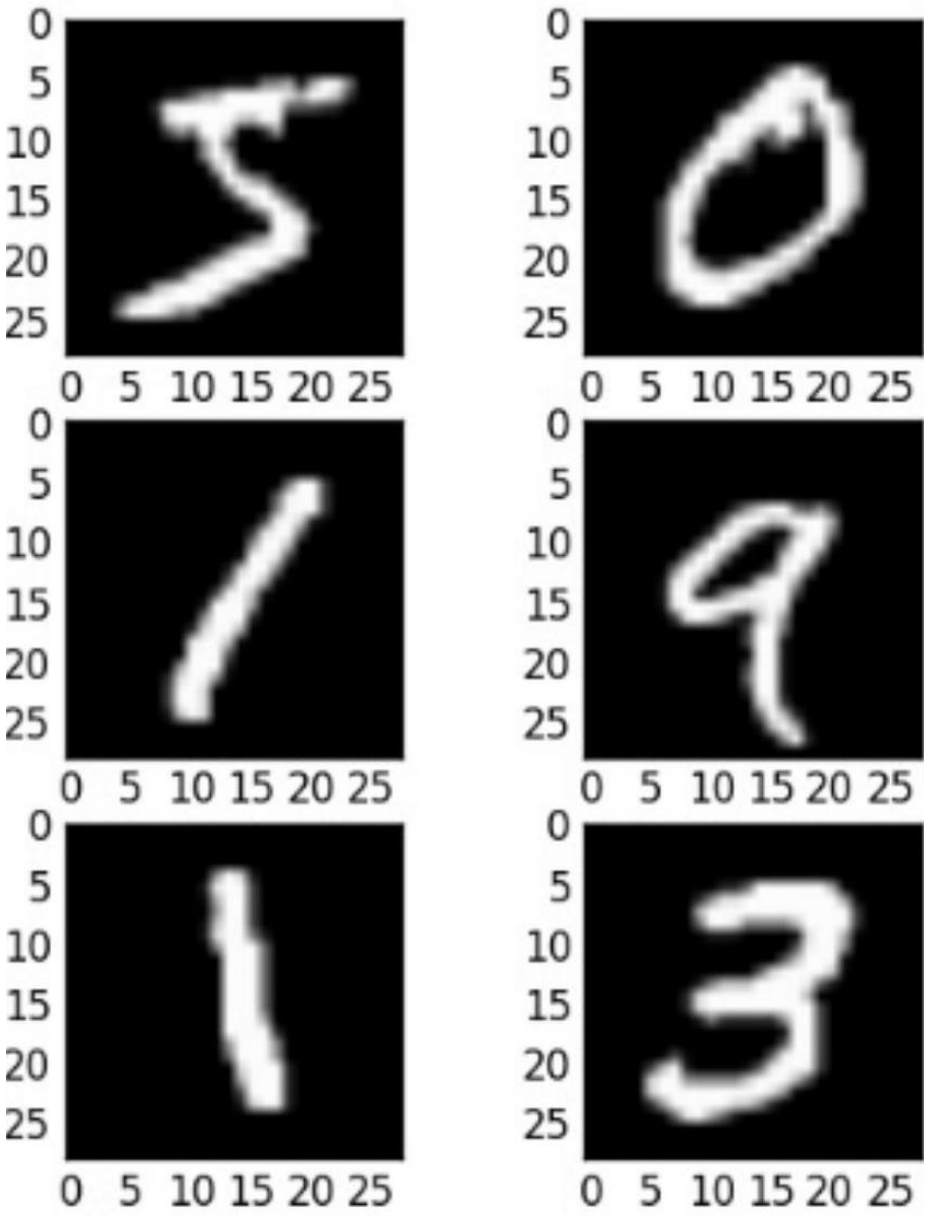
Source: the **ResNet** paper (CVPR 2016)

34-layer residual

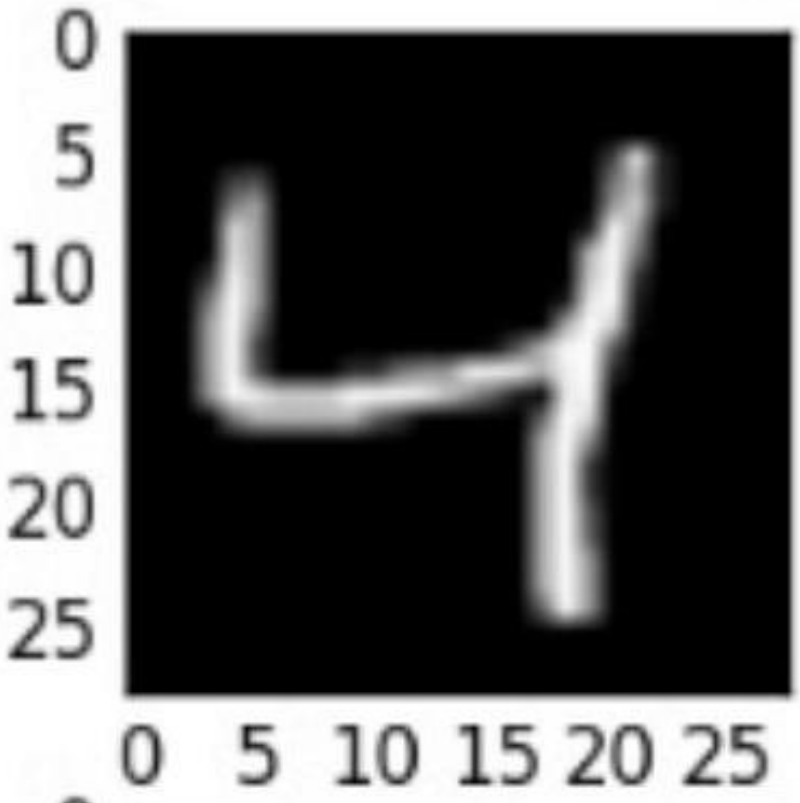

3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256

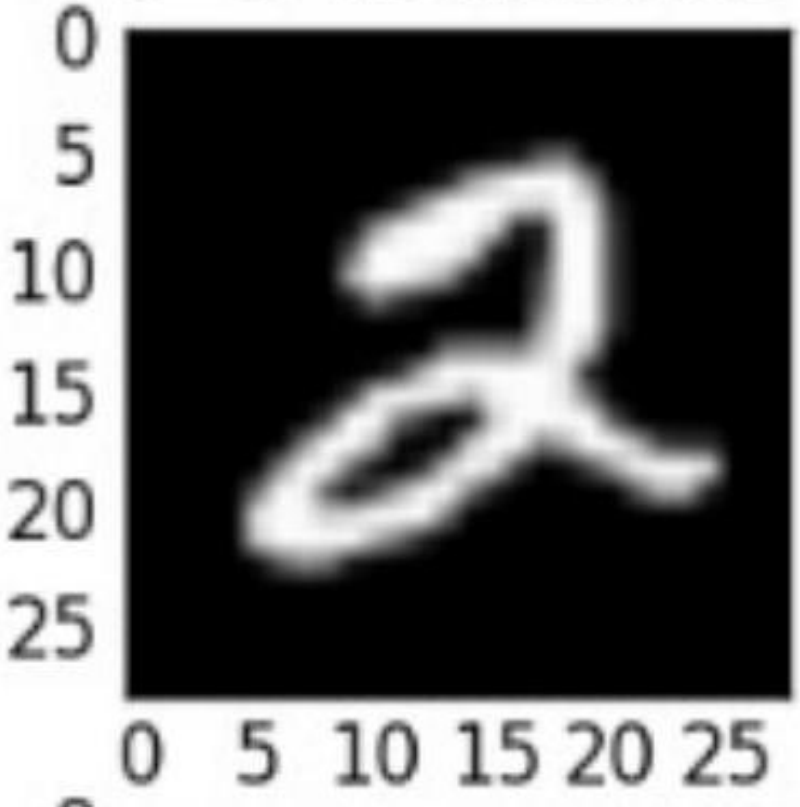
Data Augmentation

Data augmentation: generate new data from the data

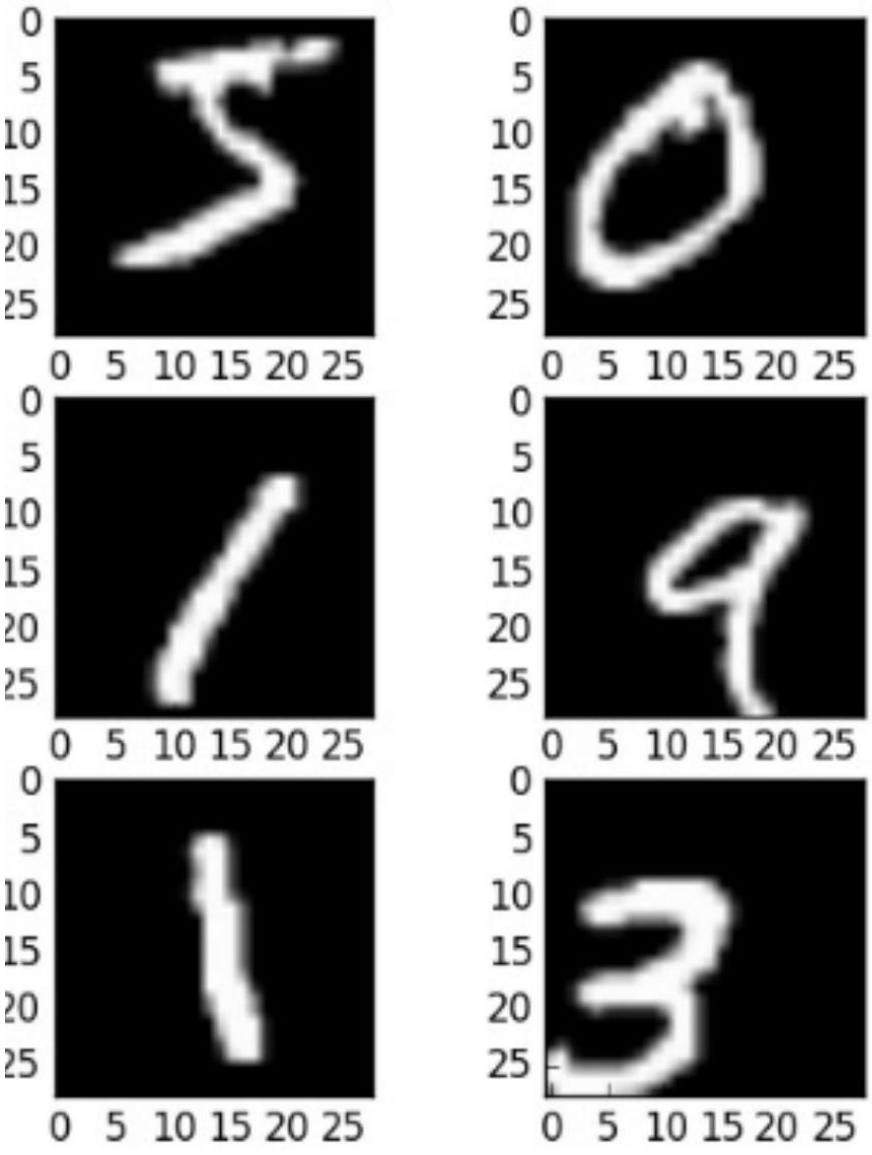


MNIST

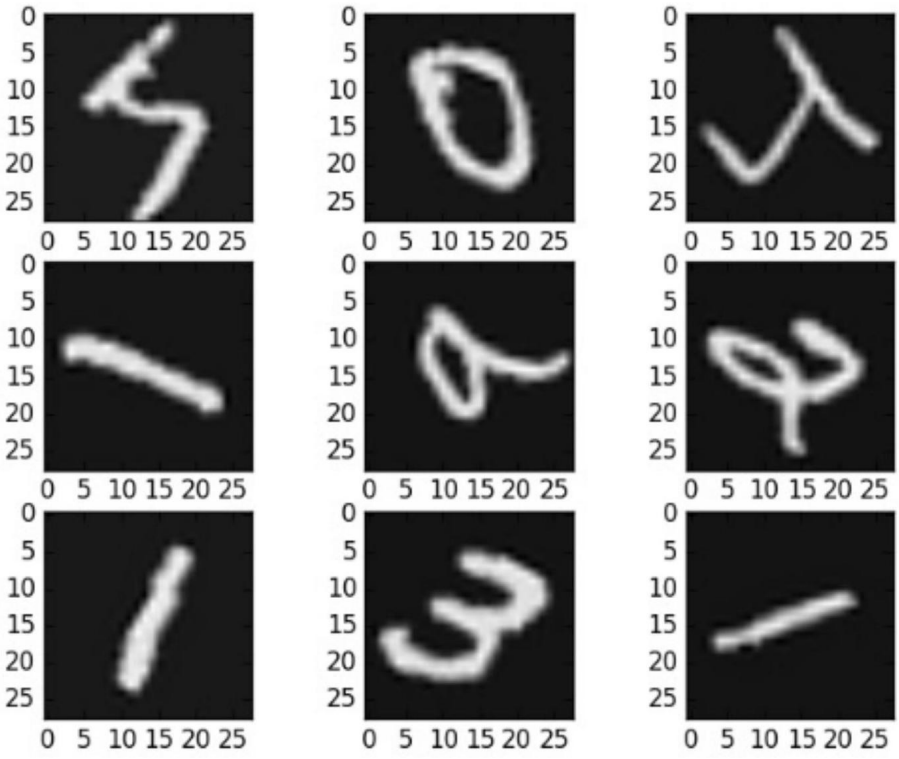

0
5
10
15
20
25
0 5 10 15 20 25


0
5
10
15
20
25
0 5 10 15 20 25


0



0510152025



Shifted MNIST

Rotated MNIST Note dangers of excessive augmentation! May eventually confuse 6 and 9

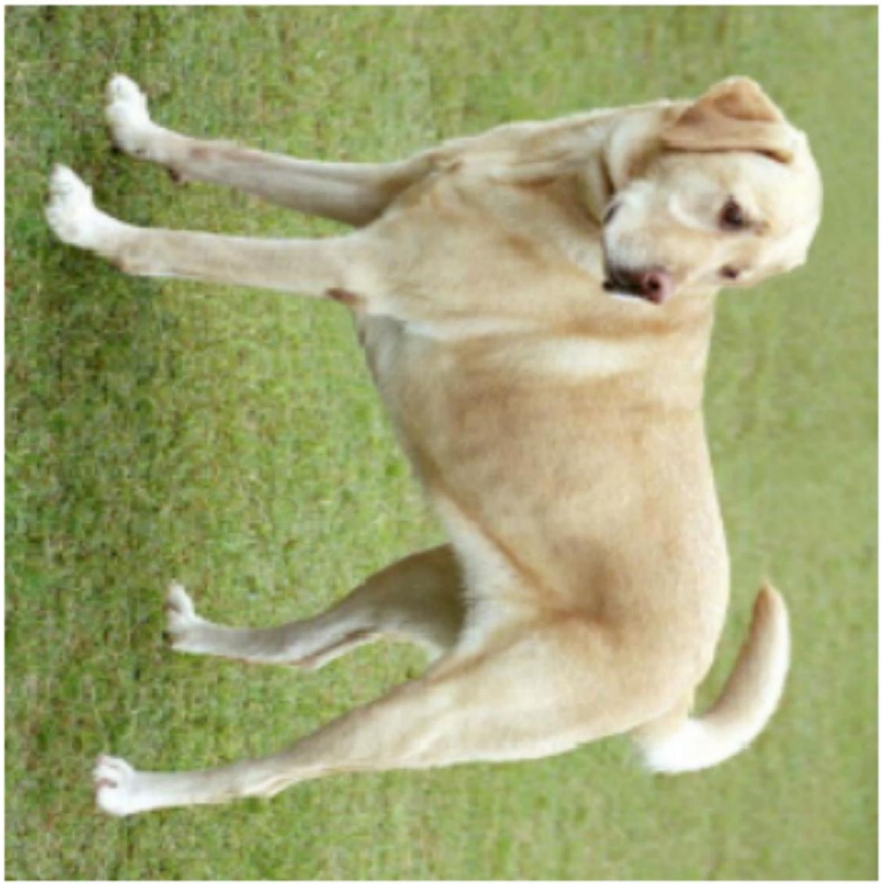
Transformation $\tau : \mathbb{R}^d \to \mathbb{R}^d$ which preserves the labels (i.e., $y_x = y_{\tau(x)}$)

$$S = S_{\text{train}} \cup \{(\tau(x_i), y_i)\}_{i=1}^{n}$$

- We train on more data
- Encourages models to be invariant to τ
- It can be seen as regularization
- These transformations are task and dataset specific

Data augmentation: pictures can also be cropped, resized, or perturbed by a small amount of noise

(a) Original



(f) Rotate $\{90°, 180°, 270°\}$

(b) Crop and resize

(g) Cutout

(c) Crop, resize (and flip)

(h) Gaussian noise

(d) Color distort. (drop)

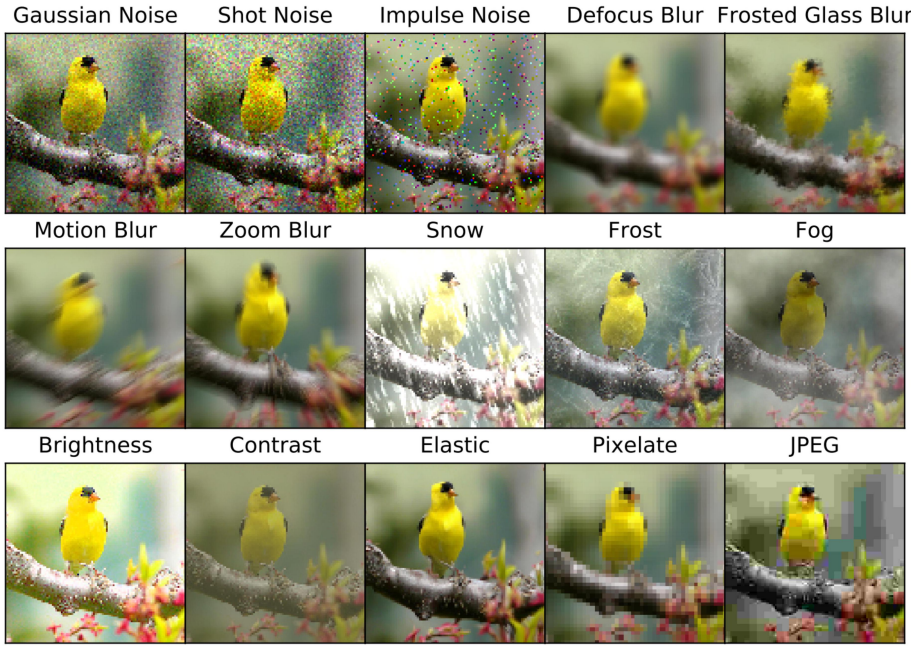
(i) Gaussian blur

(e) Color distort. (jitter)

(j) Sobel filtering

T Chen, S Kornblith, M Norouzi, G Hinton. A simple framework for contrastive learning of visual representations. ICML, 2020

Data augmentation: generated corruptions



D. Hendrycks and T. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. ICLR, 2019

Weight Decay

Weight decay: l_2-regularization for NNs

It is standard practice to regularize weights without regularizing bias terms:

$$\min \mathscr{L} + \frac{\lambda}{2} \sum_l \left\| \mathbf{W}^{(l)} \right\|_F^2$$

Weight decay [1] favors small weights which can aid in generalization and optimization Optimization with gradient descent:

$$\left(w_{i,j}^{(l)}\right)_{t+1} = \left(w_{i,j}^{(l)}\right)_t - \eta \nabla \mathscr{L} - \eta\lambda \left(w_{i,j}^{(l)}\right)_t = \underbrace{(1-\eta\lambda)}_{\text{weight decay}} \left(w_{i,j}^{(l)}\right)_t - \eta \nabla \mathscr{L}$$

Interaction with BatchNorm:

- $\mathrm{BN}(\mathbf{WX}) = \mathrm{BN}(\alpha \mathbf{WX})$ for $\alpha \in \mathbb{R}_{>0}$ (assuming $\varepsilon \approx 0$)
- BN is scale invariant in $\mathbf{W}$, hence there is no direct regularization effect from WD

- However, the training dynamics differ [2]

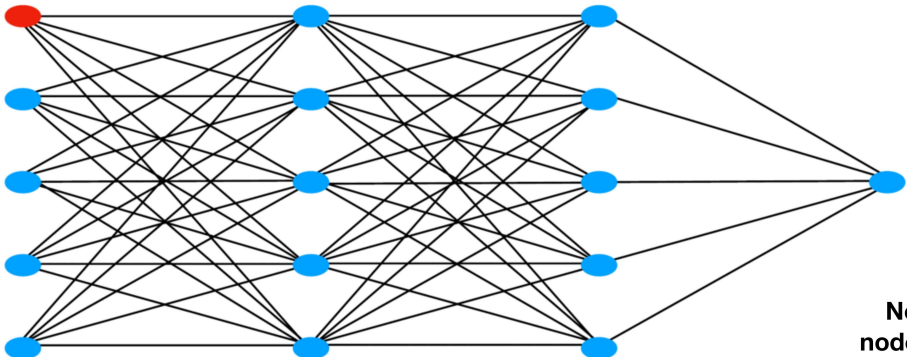
[1] A. Krogh and J. A Hertz. A simple weight decay can improve generalization. NeurIPS, 1992

[2] R. Wan, et al. Spherical Motion Dynamics: Learning Dynamics of Normalized Neural Network using SGD and Weight Decay. NeurlPS, 2021

Dropout

Dropout: randomly drop nodes

Def: At each training step, retain with probability $p^{(l)}$ each node in layer (l) :

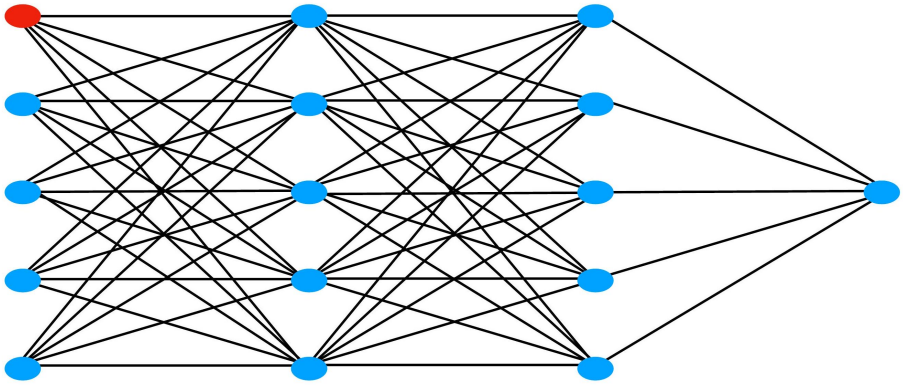


Note: In practice we drop nodes independently for each element of a mini-batch

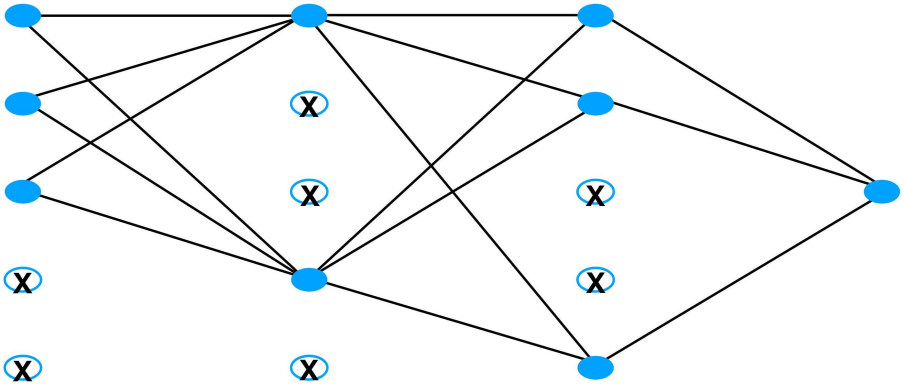
N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, JMLR, 2014

Dropout: training phase

Def: At each training step, retain with probability $p^{(l)}$ each node in layer (l) :

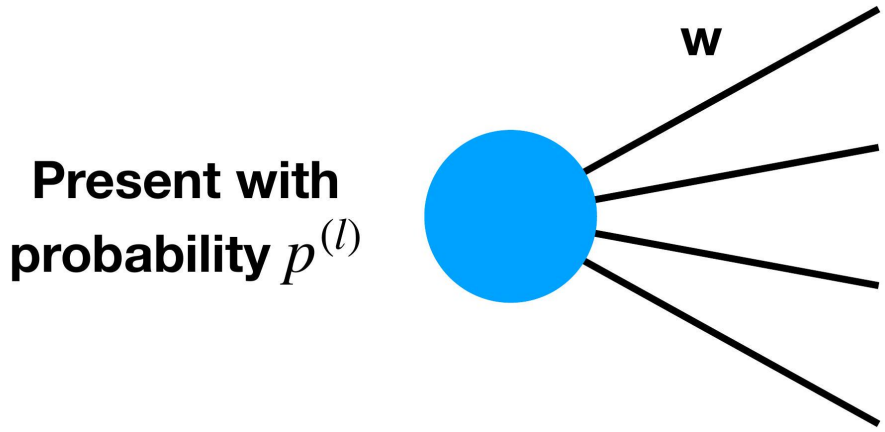


Original network



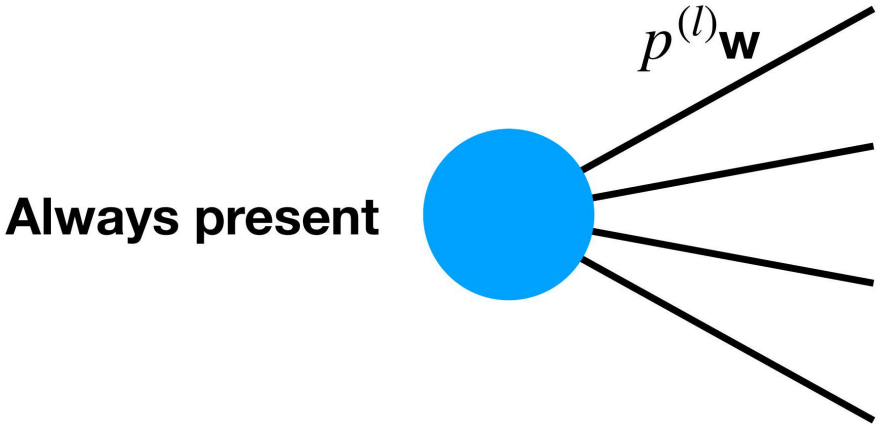
Random subnetwork
Run one step of SGD on the subnetwork and update the weights

Dropout: testing phase



At training time
When testing:

- Use all nodes



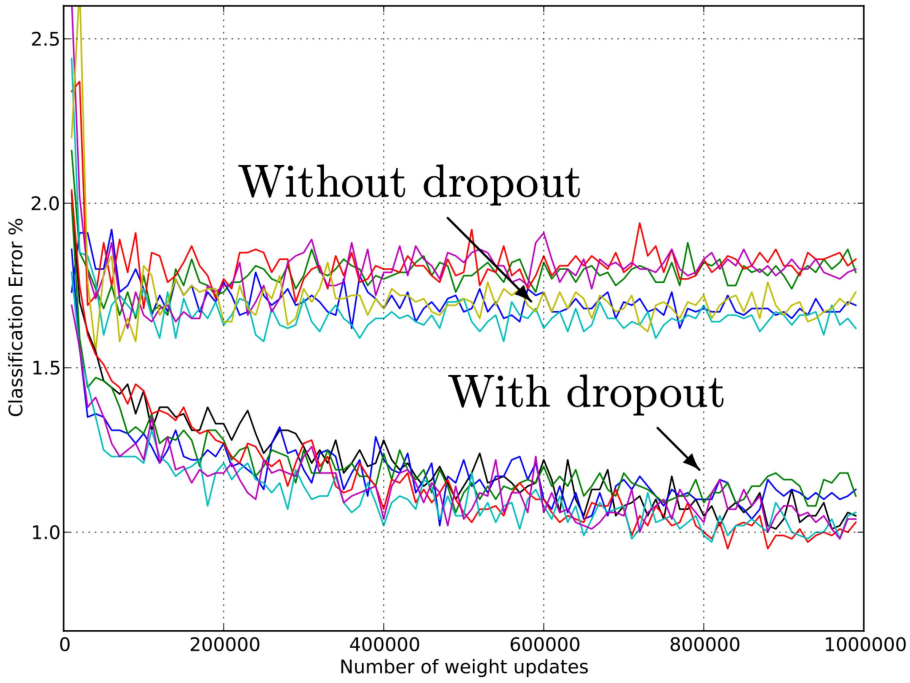
At test time
Note: Variance is generally not preserved and as a result Dropout often works
poorly with normalization

- Scale each of them by the factor $p^{(l)}$ to ensure that the expected output (when considering the probability of dropping nodes during training) matches the actual output at test time

Remark: Weight rescaling can be implemented during training time by scaling the weights by $1/p^{(l)}$ after each weight update - this is how it is implemented in practice

Dropout: results

- Setting: Fully-connected networks of different width and depth on MNIST
- Dropout results in lower test error
- However, dropout typically requires more iterations to converge due to increased stochastic noise



Conclusion

Entangled effects of various methods: CIFAR10

model	# parameters	Random crop	Weight decay	Train accuracy	Test accuracy
		Yes	Yes	100.0	89.05
~~Inception~~	~~1′649′402~~	Yes	No	100.0	89.31
		No	Yes	100.0	86.03
		No	No	100.0	85.75
Inception w/0 BatchNorm	1'649'402	No	Yes	100.0	83.00
		No	No	100.0	82.00
		Yes	Yes	99.90	81.22
~~Alexnet~~	~~1'387'786~~	Yes	No	99.82	79.66
		No	Yes	100.0	77.36
		No	No	100.0	76.07
~~MLP 3×512~~	~~1'735'178~~	No	Yes	100.0	53.35
		No	No	100.0	52.39
~~MLP 1×512~~	~~1'209'866~~	No	Yes	99.80	50.39
		No	No	100.0	50.51

Entangled effects of various methods: ImageNet

model	Data aug	Dropout	Weight decay	Batch Norm	Skip Connections	Top-1 train	Top-1 test
ResNet200 (v2)	Yes	No	Yes	Yes	Yes	?	79.9
Inception (v3)	Yes	Yes	Yes	Yes	No	92.18	77.84
	Yes	No	No	Yes	No	92.33	72.95
	No	No	Yes	Yes	No	90.60	67.18(72.57)
	No	No	No	Yes	No	99.53	59.80(63.16)
VGG19	Yes	Yes	Yes	No	No	?	72.7

() : best test accuracy during training, i.e., with early stopping C Zhang, S Bengio, M Hardt, B Recht, O Vinyals. Understanding deep learning requires rethinking generalization. ICLR, 2017

Recap

- Convolutional networks are composed of sparsely connected convolutional layers instead of fully-connected linear layers
- The same convolution is applied as a sliding window across all spatial locations
- Data augmentation usually results in a significant improvement in the model's generalization performance

- Weight decay and dropout can further enhance the performance, though typically to a lesser extent