Convolution in Neural Networks

Saeed Salehi

credits to Philipp Seegerer (Aignostics), Alona Fyshe (NeuroMatch), Grégoire Montavon (BIFOLD)

BIFOLD

berlin

Prelude

Prelude

Convolution and Cross-Correlation

For two functions $f(t)$ and $g(t)$ we can define the convolution denoted by $*$ (asterisk):

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(s) \cdot g(t-s) ds = (g * f)(t)$$

which can be extended to discrete signals:

$$\left[f * g\right]_t = \sum_{s=-\infty}^{+\infty} f_s \cdot g_{t-s}$$

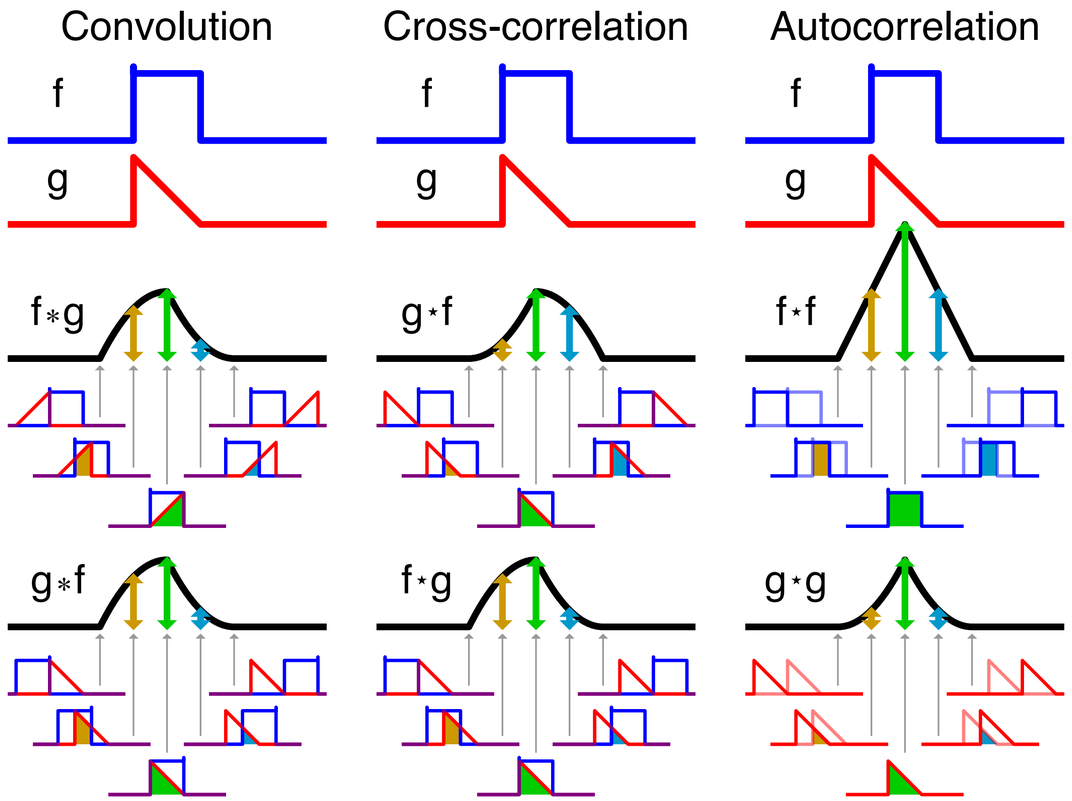
and related operator **cross-correlation** denoted with ⋆ (star):

$$\left(f \star g\right)(t) = \sum_{\tau'} f(t+\tau') \cdot g(\tau')$$

Note: Cross-Correlation is the operation commonly used in ML referred to as Convolution.

Prelude

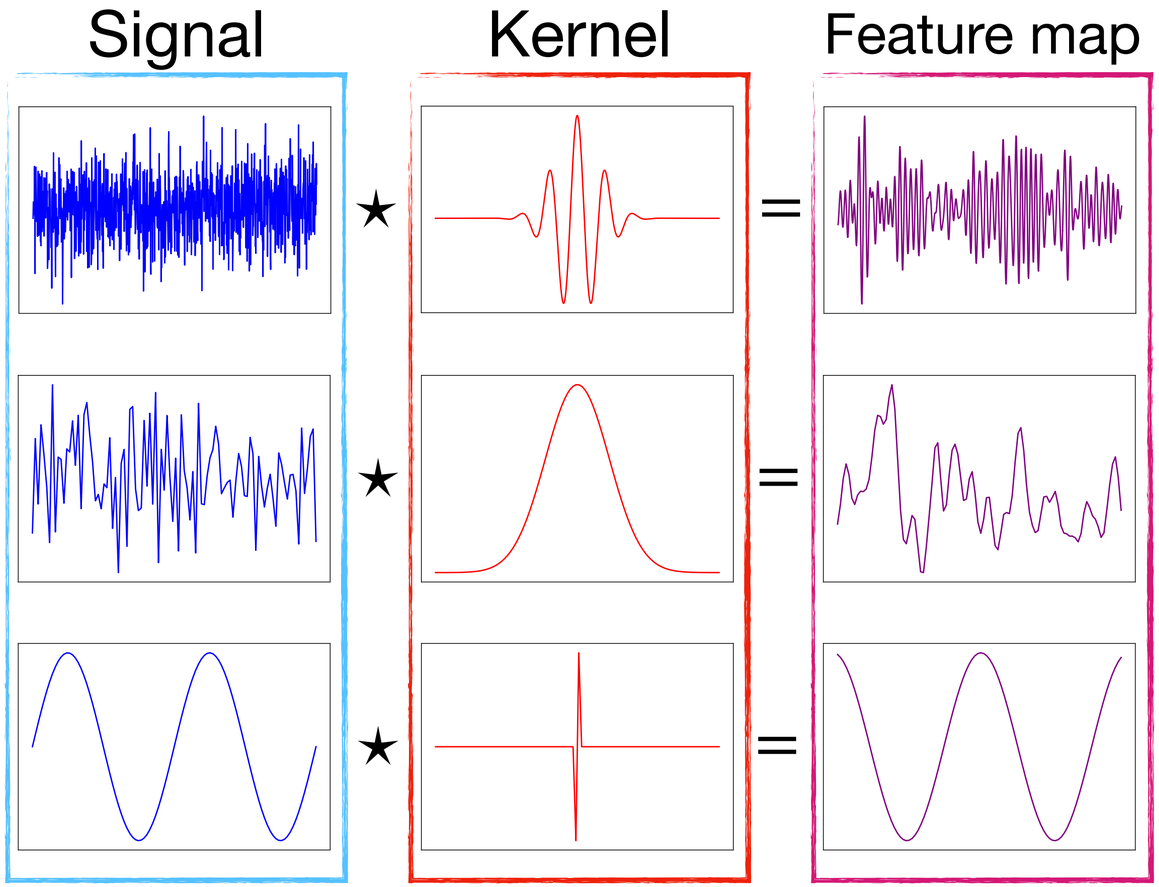
Convolution and Cross-Correlation



[Figure from wikipedia Cross-correlation]

Prelude

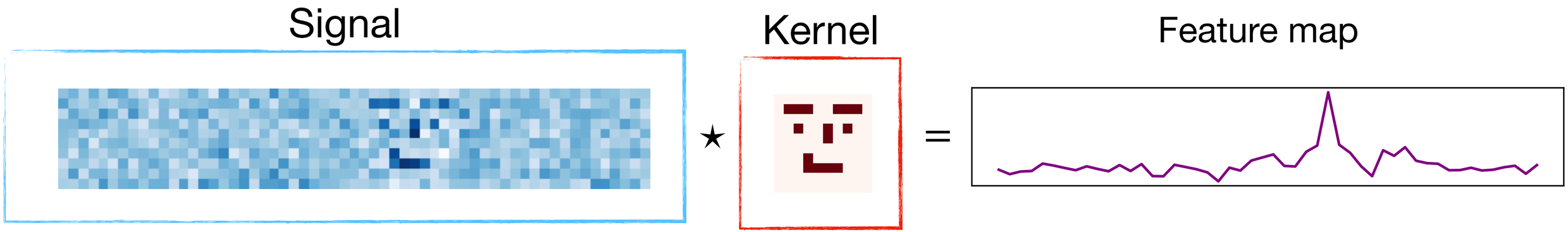
Signal vs Kernel



Note: Often convolution kernels are called!

Prelude

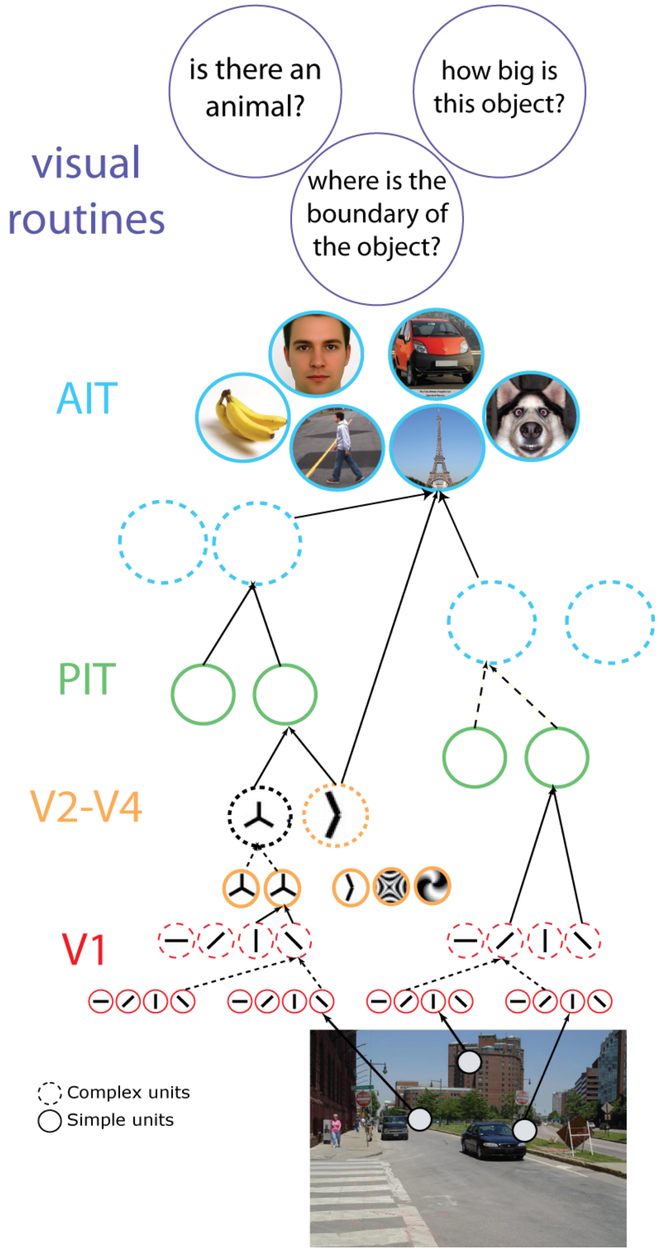
Feature detection



Motivation

Motivation

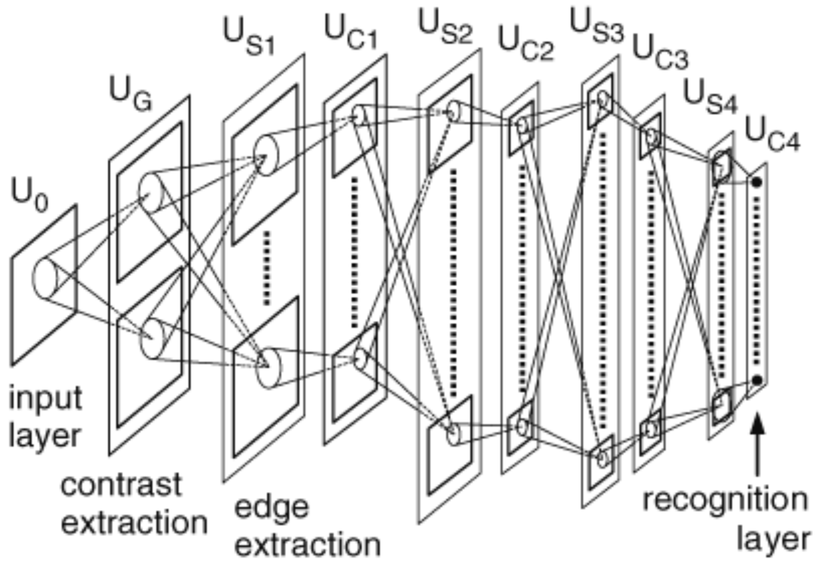
Neuroscience



[Figure from Scholarpedia, Poggio, Serre 2013

Motivation

Neocognitron



[Figure from Scholarpedia, Fukushima 2007

Motivation

Machine Learning

$1024 \cdot 1024 \cdot RGB \approx \pi 10^6$

down-size

$224 \cdot 224 \cdot RGB = 150528$

Flatten

$\mathbf{x} \in \mathbb{R}^{150528}$

$\mathbf{W}_1 \in \mathbb{R}^{150528 \times L_1}$

$\mathbf{W}_1 \in \mathbb{R}^{L_n \times 1000}$

$\mathbf{y} \in \mathbb{R}^{1000}$

Number of learnable parameters excluding bias:

$$150528 \times L_1 + L_1 \times L_2 + \ldots + L_n \times 1000$$

Motivation

Machine Learning cont.

(naively) using Fully Connected layers on images leads to

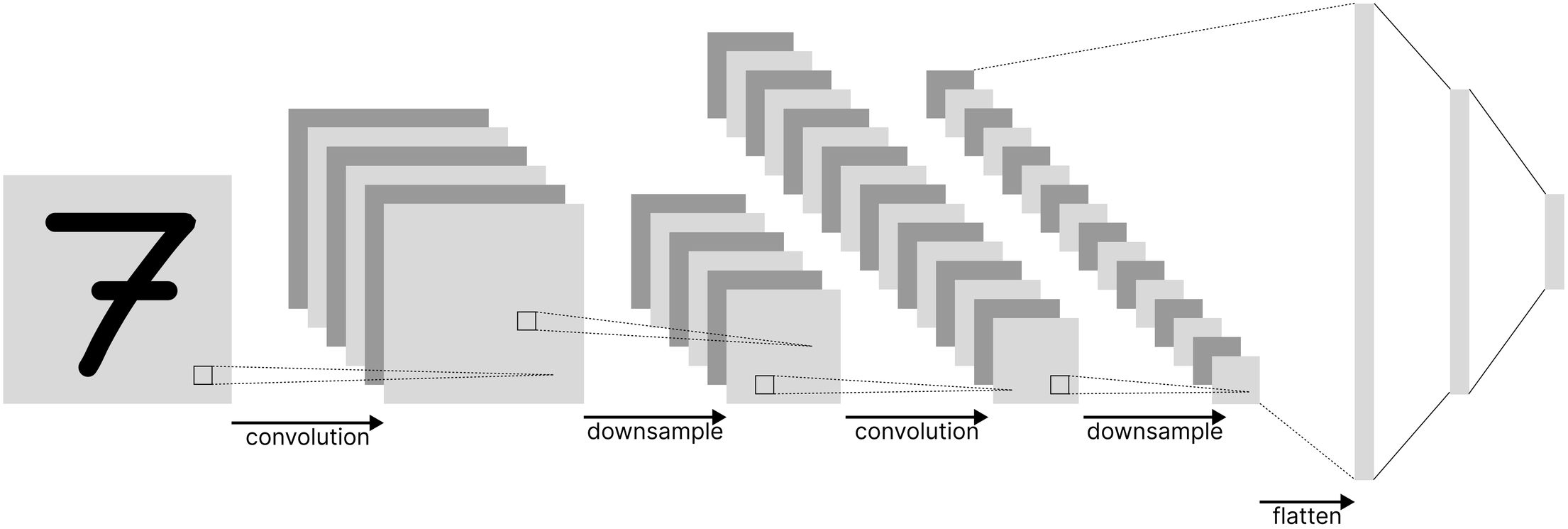
- Exploding number of learnable parameters (weights)
- Over-fitting
- loss of local geometry

Recalling **Regularization**

- Reducing number of effective parameters

LeNet-5

Gradient Based Learning Applied to Document Recognition



[Figure adopted from Yann LeCun et al. 1998]

intuition

intuition of Why

Feature Learning (extraction)

Using backprop to **learn** to extract **relevant features** from the data

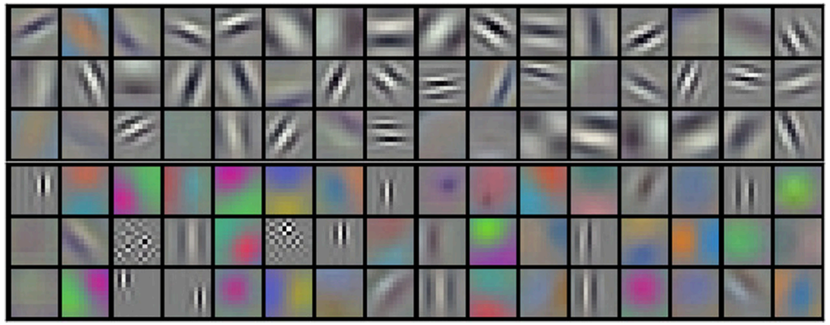
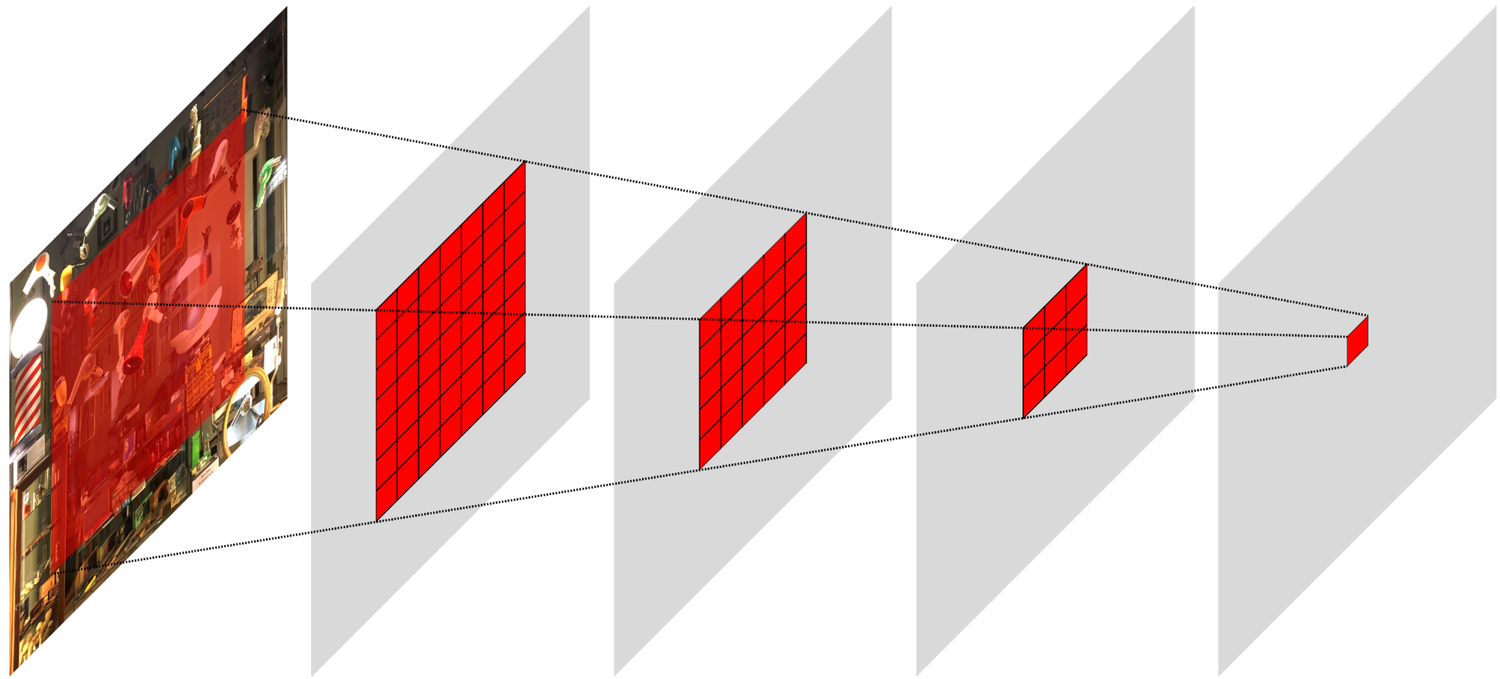


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

[LeCun 1998], [Figure from AlexKrizhevsky, Ilya Sutskever, Geoffrey Hinton 2012]

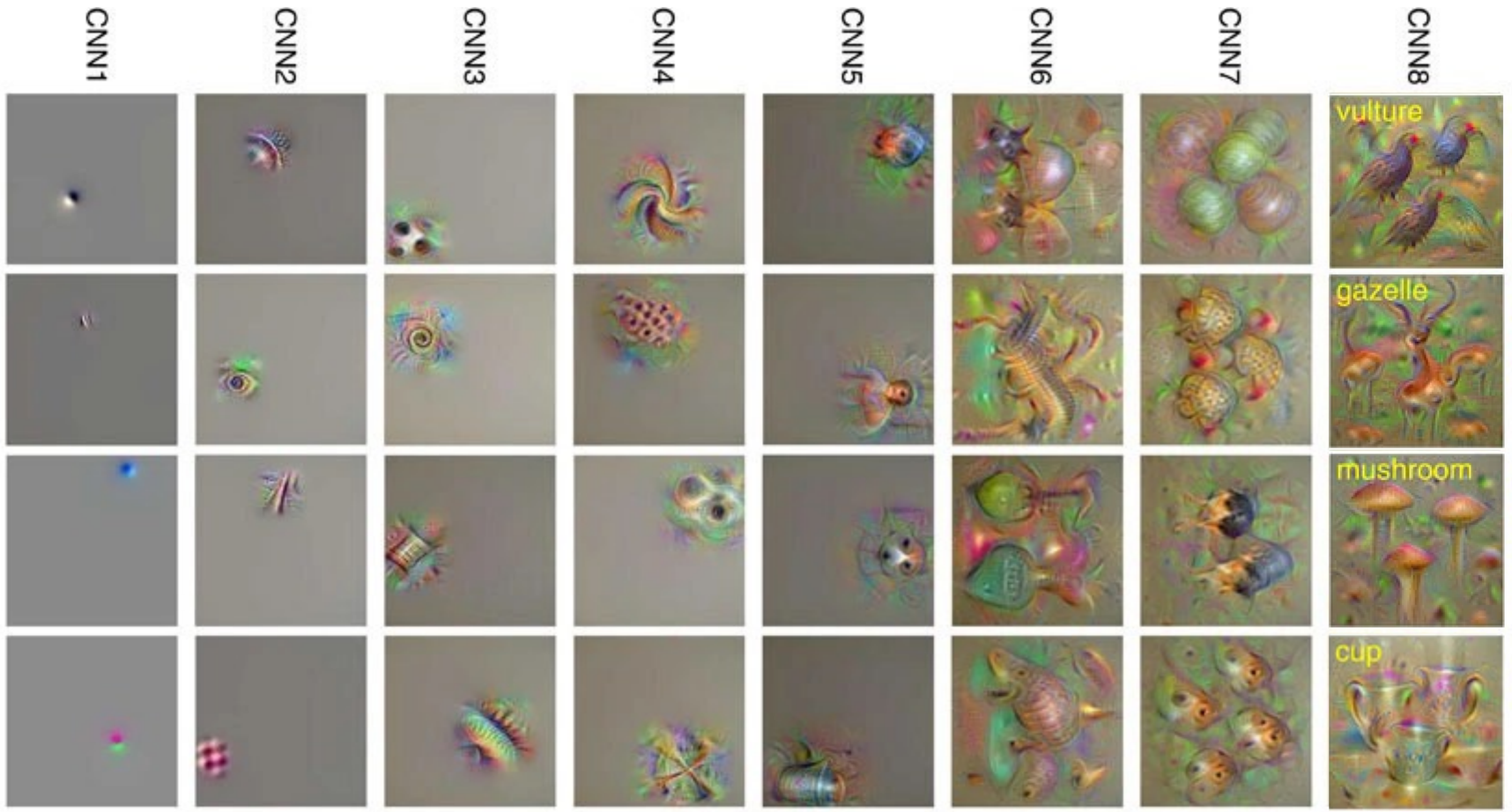
intuition of Why

Receptive Field



intuition of Why

Hierarchical Feature composition and learning

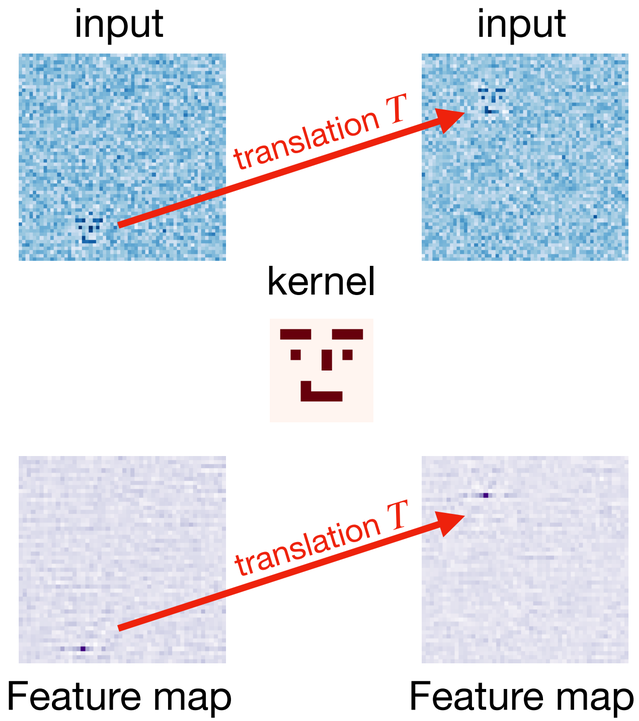


[Figure from Horikawa & Kamitani 2017]

intuition of Why

invariance and equivariance

- Using the same feature detector at different position
- Translation equivariance
- after flattening: Location invariant



intuition of Why

Weight Sharing

rewriting convolution with a 3×3 kernel as a matrix multiplication (as in fully connected layer)

$$\mathbf{y} = \mathbf{x} * \mathbf{k} = \mathbf{W}^T \mathbf{x}$$

$$w^\top = \left(\begin{array}{cccccc|cccccc}
v_{11}^1 & v_{11}^2 & v_{11}^3 & 0 & 0 & 0 & v_{12}^1 & v_{12}^2 & v_{12}^3 & 0 & 0 & 0 \\
0 & v_{11}^1 & v_{11}^2 & v_{11}^3 & 0 & 0 & 0 & v_{12}^1 & v_{12}^2 & v_{12}^3 & 0 & 0 \\
0 & 0 & v_{11}^1 & v_{11}^2 & v_{11}^3 & 0 & 0 & 0 & v_{12}^1 & v_{12}^2 & v_{12}^3 & 0 \\
0 & 0 & 0 & v_{11}^1 & v_{11}^2 & v_{11}^3 & 0 & 0 & 0 & v_{12}^1 & v_{12}^2 & v_{12}^3 \\
\hline
v_{21}^1 & v_{21}^2 & v_{21}^3 & 0 & 0 & 0 & v_{22}^1 & v_{22}^2 & v_{22}^3 & 0 & 0 & 0 \\
0 & v_{21}^1 & v_{21}^2 & v_{21}^3 & 0 & 0 & 0 & v_{22}^1 & v_{22}^2 & v_{22}^3 & 0 & 0 \\
0 & 0 & v_{21}^1 & v_{21}^2 & v_{21}^3 & 0 & 0 & 0 & v_{22}^1 & v_{22}^2 & v_{22}^3 & 0 \\
0 & 0 & 0 & v_{21}^1 & v_{21}^2 & v_{21}^3 & 0 & 0 & 0 & v_{22}^1 & v_{22}^2 & v_{22}^3 \\
\hline
v_{31}^1 & v_{31}^2 & v_{31}^3 & 0 & 0 & 0 & v_{32}^1 & v_{32}^2 & v_{32}^3 & 0 & 0 & 0 \\
0 & v_{31}^1 & v_{31}^2 & v_{31}^3 & 0 & 0 & 0 & v_{32}^1 & v_{32}^2 & v_{32}^3 & 0 & 0 \\
0 & 0 & v_{31}^1 & v_{31}^2 & v_{31}^3 & 0 & 0 & 0 & v_{32}^1 & v_{32}^2 & v_{32}^3 & 0 \\
0 & 0 & 0 & v_{31}^1 & v_{31}^2 & v_{31}^3 & 0 & 0 & 0 & v_{32}^1 & v_{32}^2 & v_{32}^3
\end{array}\right)$$

$\mathbf{W}$ has 144 entries, but only 18 effective learnable parameters!

Elements

Anatomy of a Conv-Layer

Case of torch.nn.Conv2d

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, ...)`

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} \text{weight}(C_{out_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid 2D Cross-Correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W width in pixels.

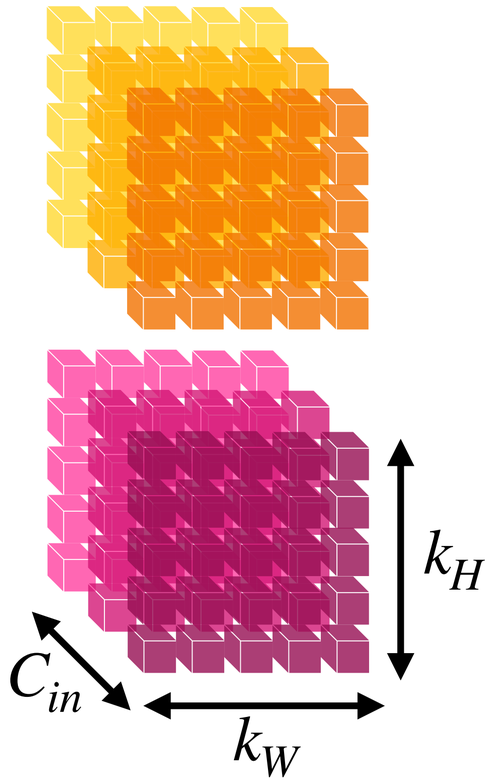
Note: in Conv2d layer, kernels are the weights (learnable parameters): $\mathbf{W} \in \mathbb{R}^{C_{out} \times C_{in} \times k_H \times k_W}$

Anatomy of a Conv-Layer

Case of torch.nn.Conv2d

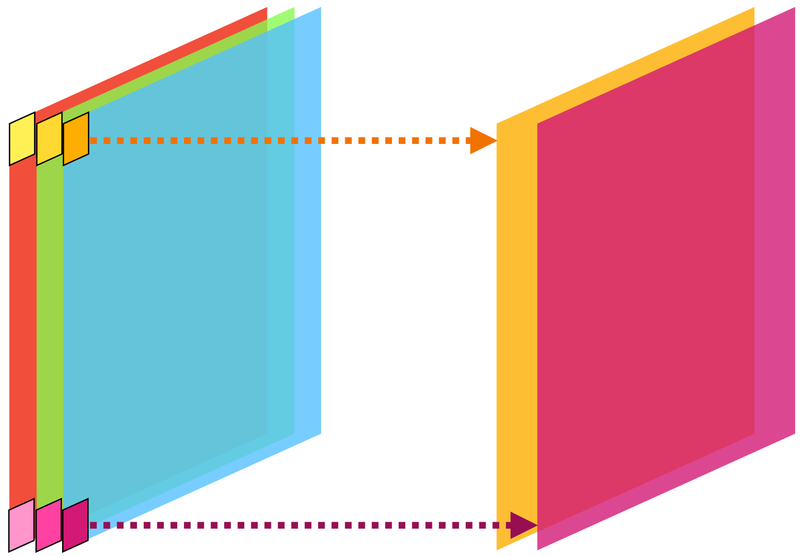
torch.nn.Conv2d(**in_channels=3, out_channels=2, kernel_size=(5, 5),** ...)

kernel_size = 5x5



n_channels = 3

out_channels = 2



number of learnable parameters:

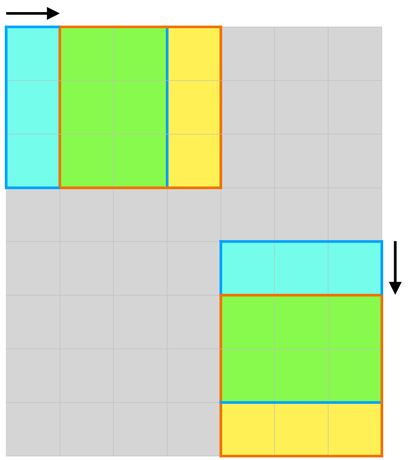
$$C_{out} \times (C_{in} \times k_H \times k_w)$$

Anatomy of a Conv-Layer

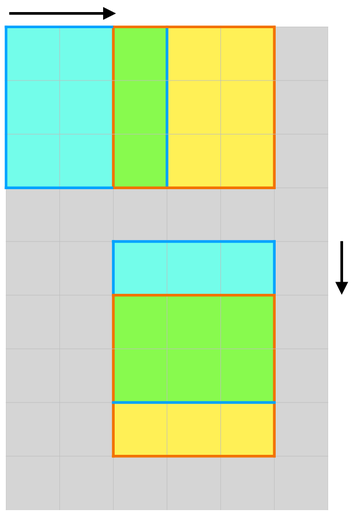
Case of torch.nn.Conv2d

`torch.nn.Conv2d(in_channels=1, out_channels=1, kernel_size=(3, 3), **stride, padding**)`

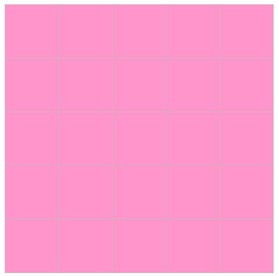
stride=1



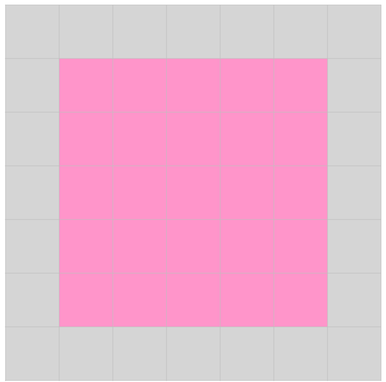
stride=(1, 2)



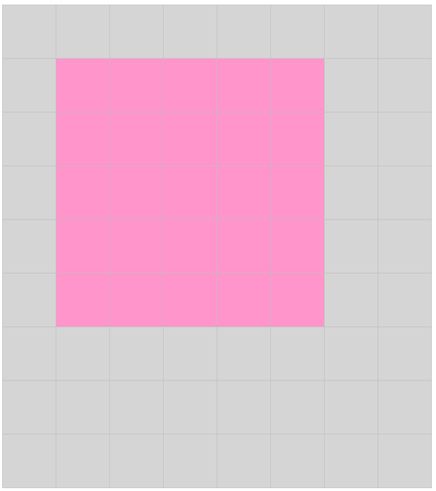
padding=0



padding=1



padding=(1, 2, 1, 3)



Exercise: Calculate the output dimensions (height and width) for input of size (h=9, w=6), kernel=(3, 3) stride=(1, 2) and padding=0 . Can you also find the padding that would results in output of the same dimension as the input?

All the Conv-Layers

- 1D-Convolution: Commonly used for Time-series
- 2D-Convolution: Popular for vision tasks
- 3D-Convolution: Suitable for volume (3-dimensional data) e.g. MRI
- Separable Convolution
- De-Convolution (ConvTranspose2d)

Pooling layer

input

2	3	6	7	6	7
2	5	3	4	3	2
1	7	3	1	4	9
9	5	4	8	1	2

torch.nn.**MaxPool2d**(kernel_size=2, stride=2)

5	7	7
9	8	9

torch.nn.**AvgPool2d**(kernel_size=2, stride=2)

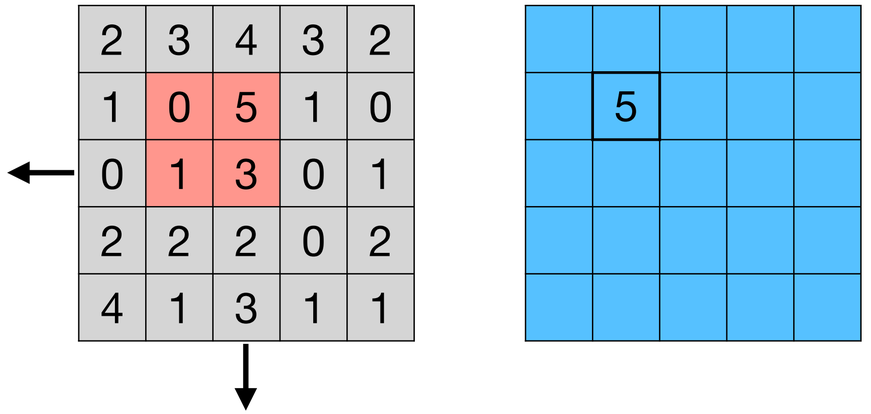
3	5	4.5
5.5	4	4

torch.nn.**AdaptiveAvgPool2d**(output_size=(2, 2))

3.5	3.8
3.8	3.2

Pooling layer

- increases the effective receptive field by subsampling (i.e. pooling)
- (Local) translation invariance of Max-Pooling



data-Augmentation

Beyond translation invariance

Random Perspective

Random Rotation

Color Jitter

Elastic Transform

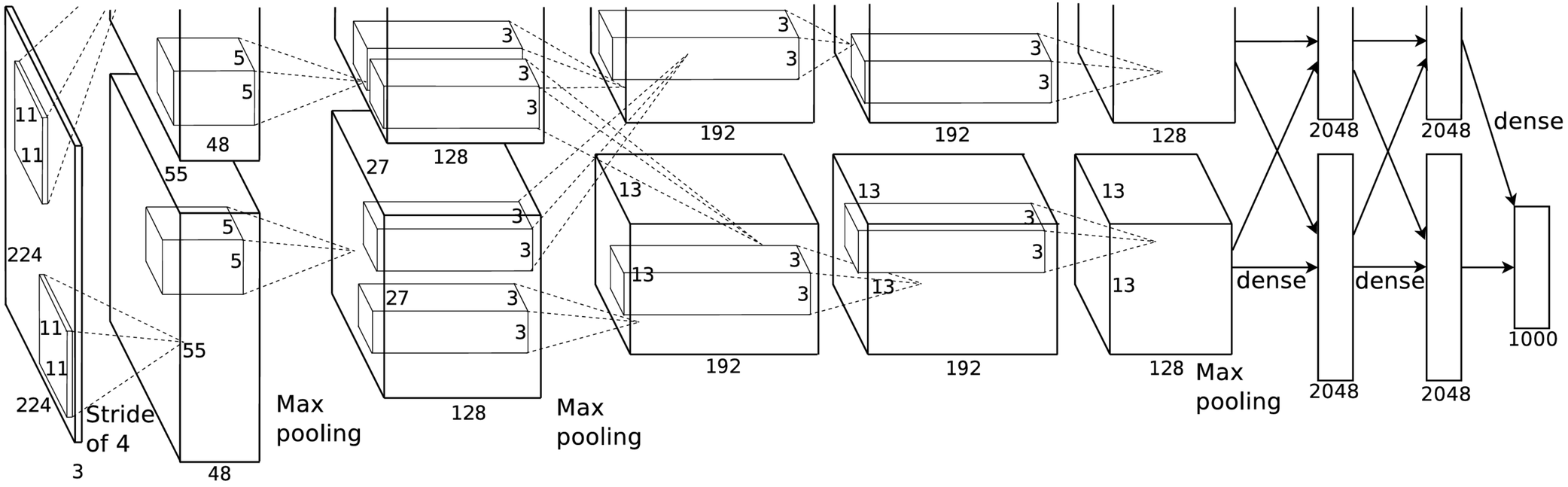
[Figures from PyTorch's illustration of transforms]

Notable Architectures

notable architectures

AlexNet

Key Contribution: First CNN winning ImageNet Challenge (2012); Making use of GPUs in training

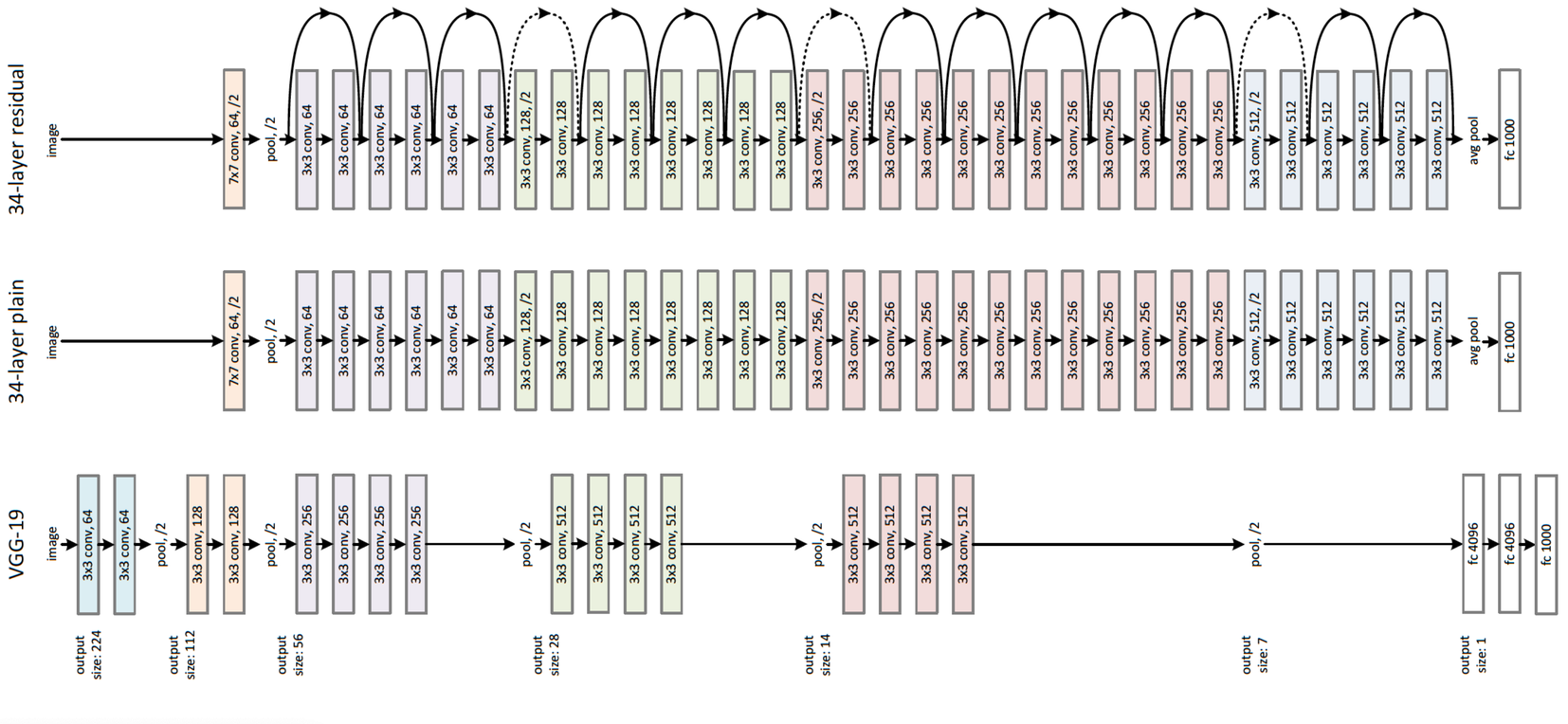


[Figure from AlexKrizhevsky, Ilya Sutskever, Geoffrey Hinton 2012]

notable architectures

ResNet

Key Contribution: Winner of ImageNet Challenge (2015) and many other challenges; Inventing the residue connection that allow training CNNs with many more layers (e.g., $8\times$ deeper than VGG).



[Figure from ResNet paper, Kaiming He et al. 2015]

notable architectures

ResNet

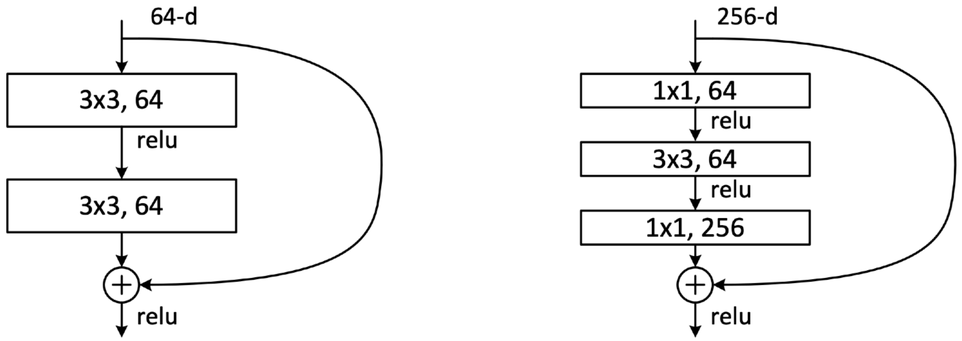
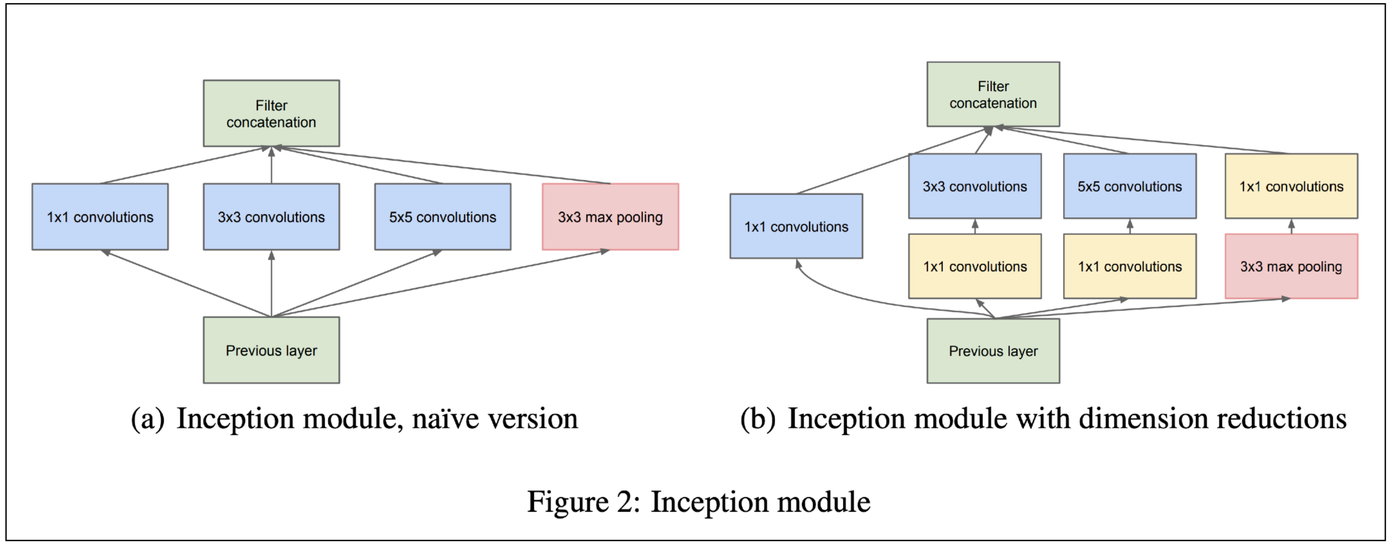


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

[Figure from ResNet paper, Kaiming He et al. 2015]

notable architectures

more..



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

Figure 2: Inception module

[Figure from inception paper, Szegedy et al. 2014]

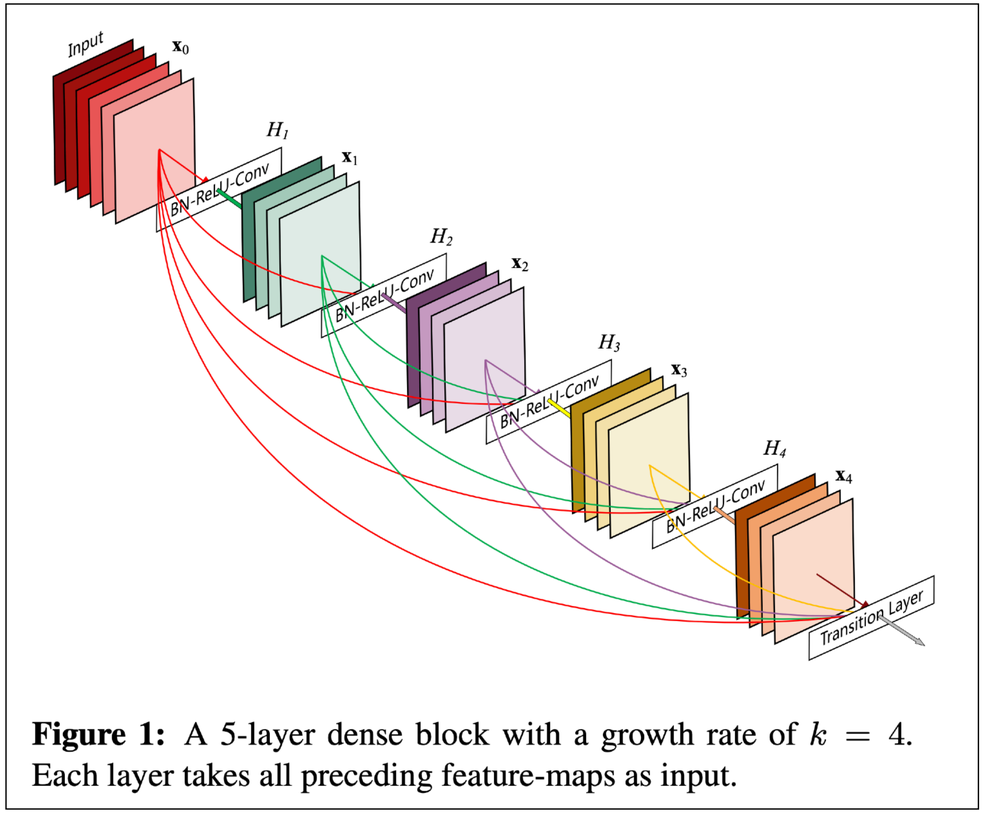


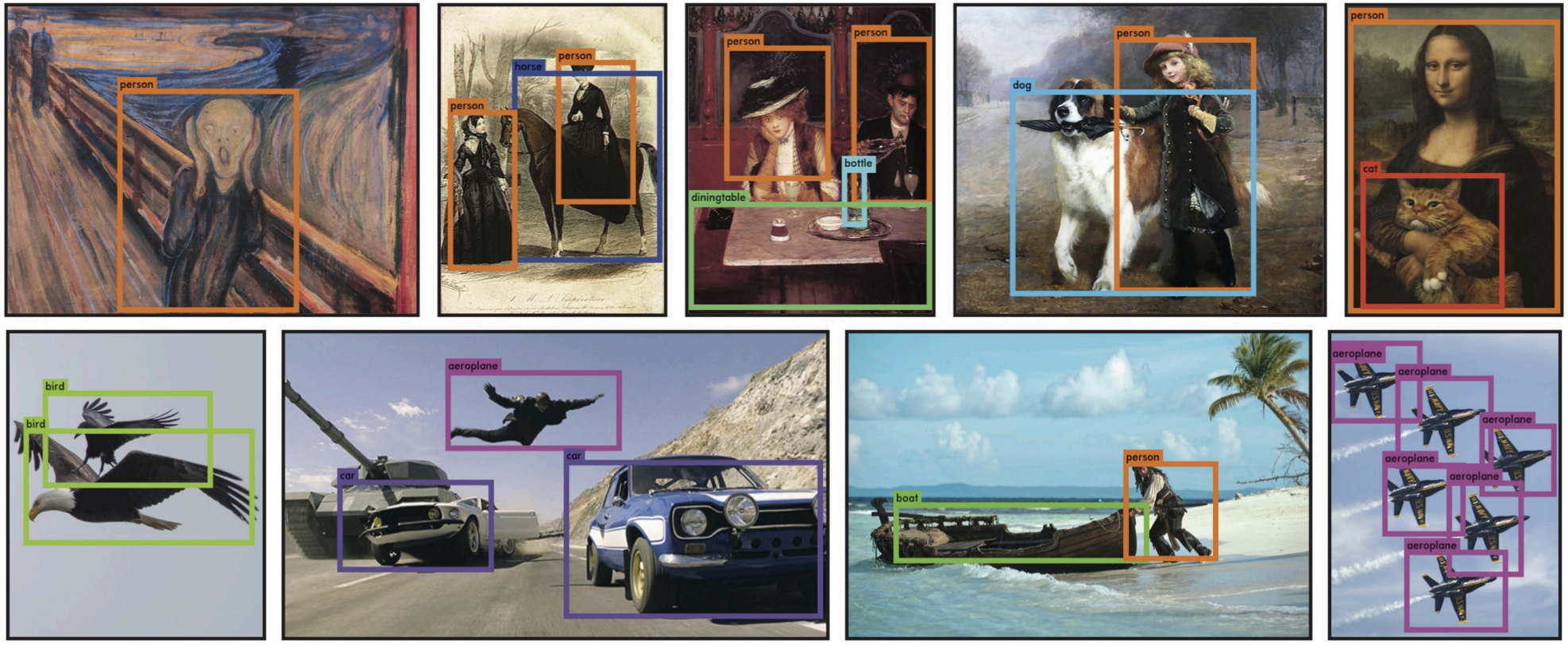
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

[Figure from DenseNet paper, Gao Huang et al. 2016]

Beyond Classification

More than classification

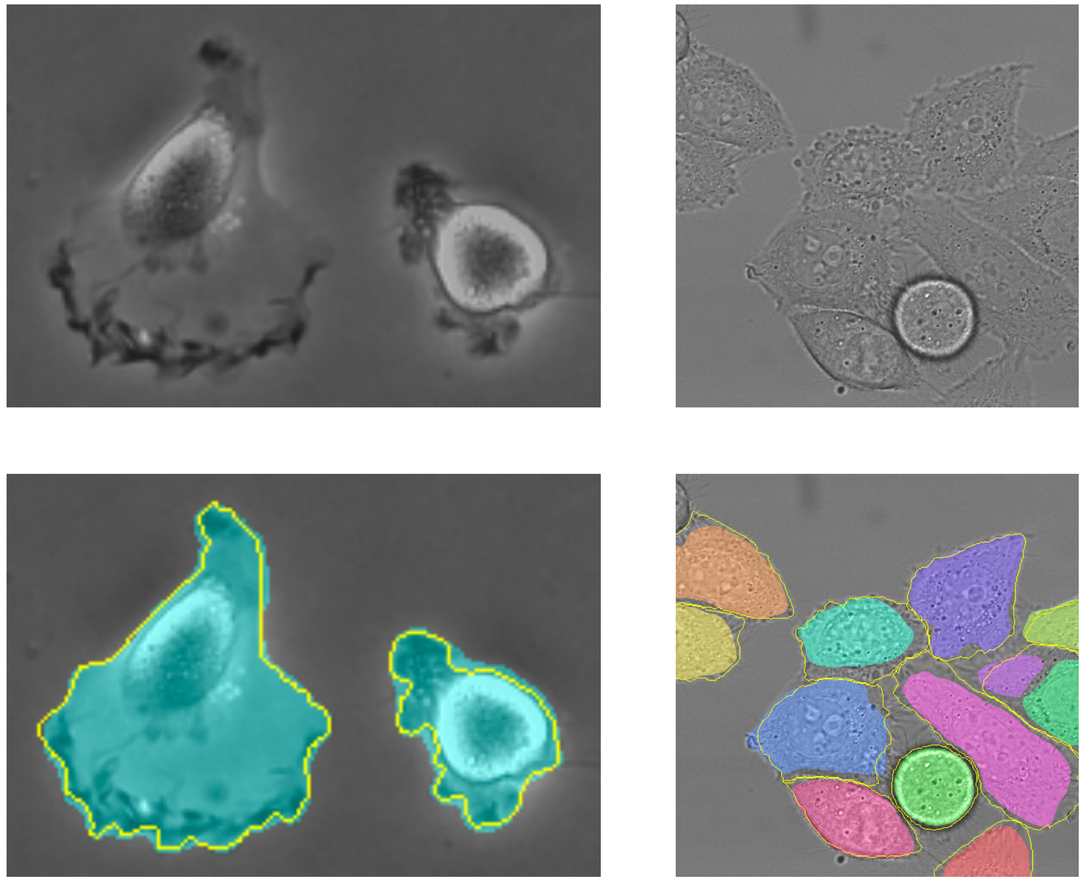
Object Detection

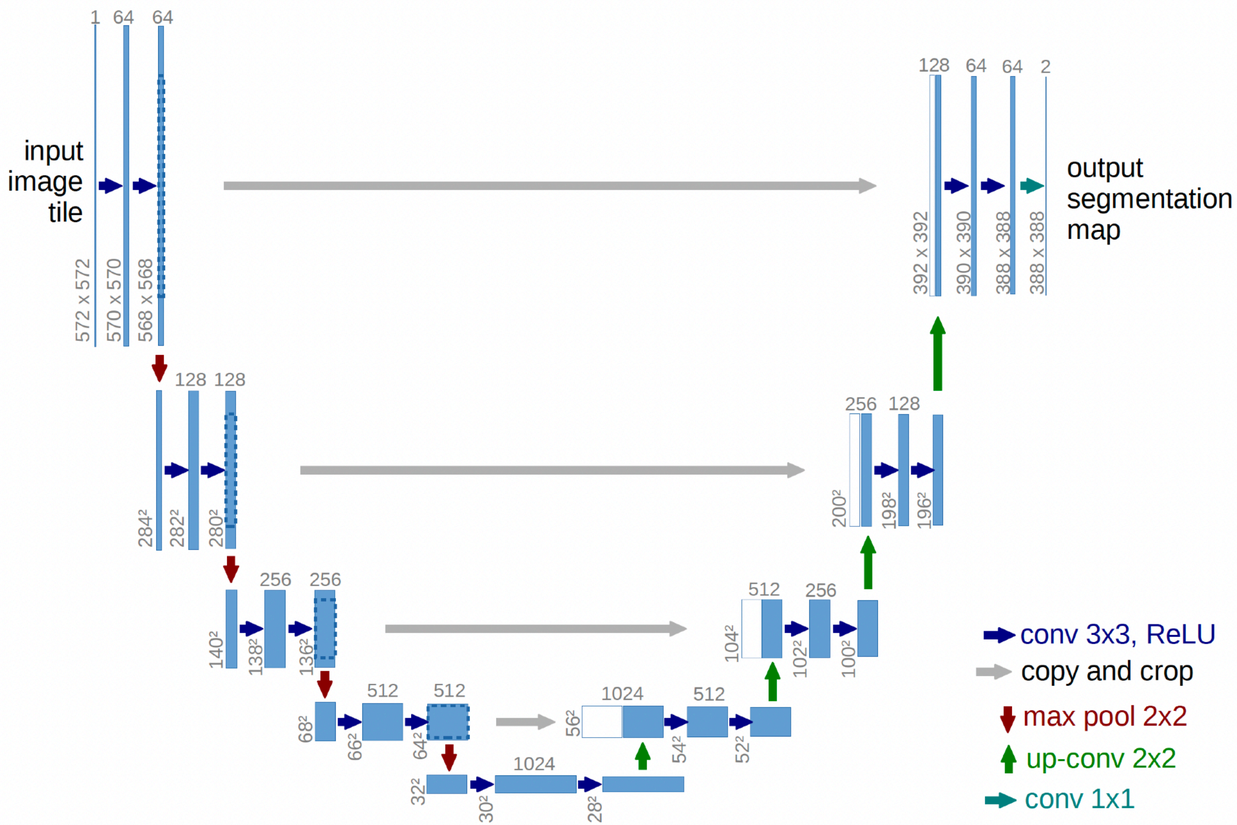


[Figure from YOLO paper, Redmon et al. 2015]

More than classification

Object Segmentation





[Figure from U-Net paper, Ronneberger et al. 2015]

More than classification

image Captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

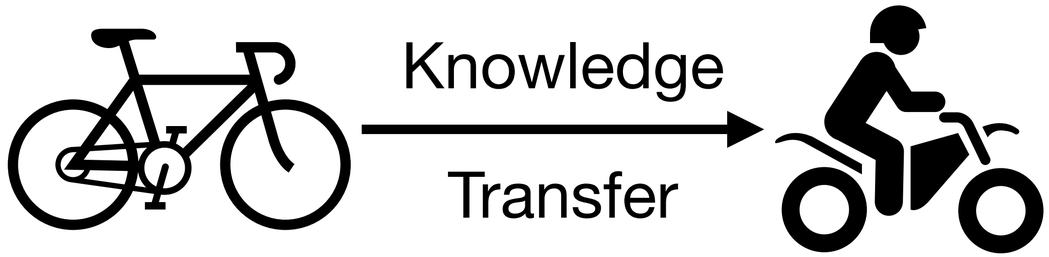
A group of people sitting on a boat in the water.

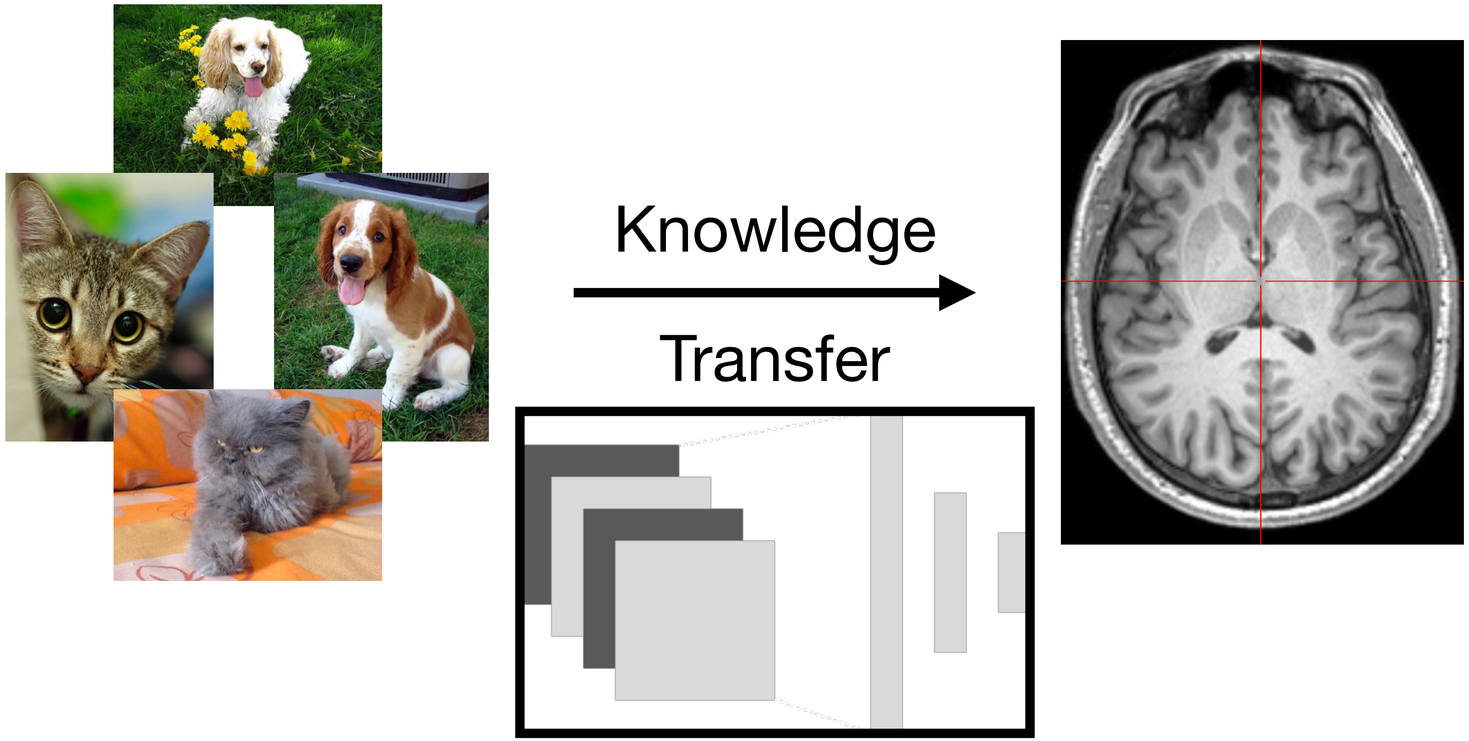
A giraffe standing in a forest with trees in the background.

[Figure from Kelvin Xu et al. 2015]

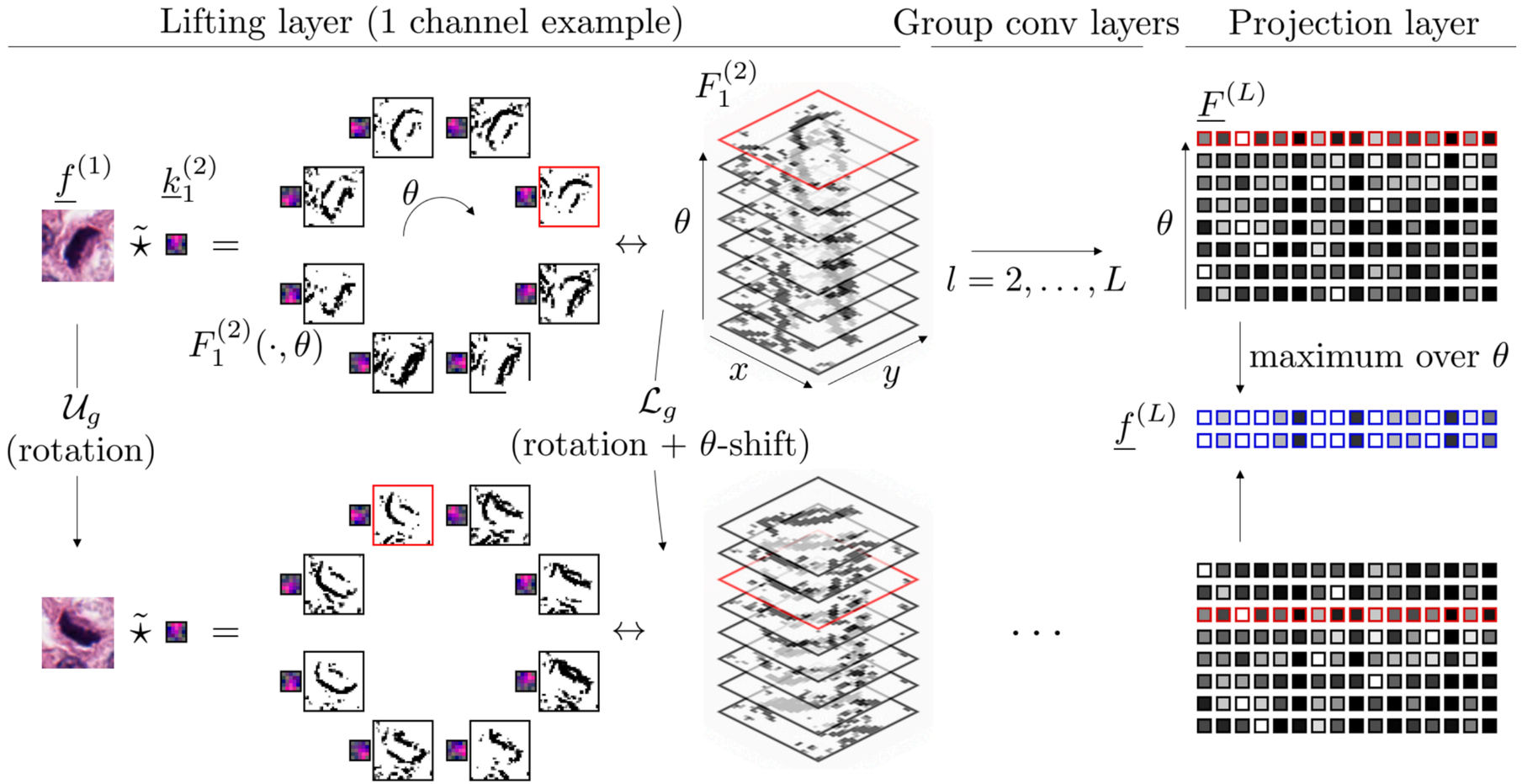
Back-up

Transfer Learning



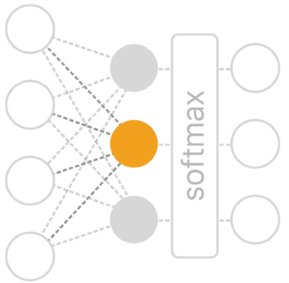


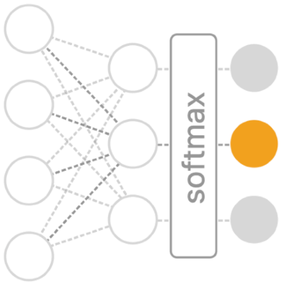
Group convolution



[Figure from Bekkers et al. 2018]

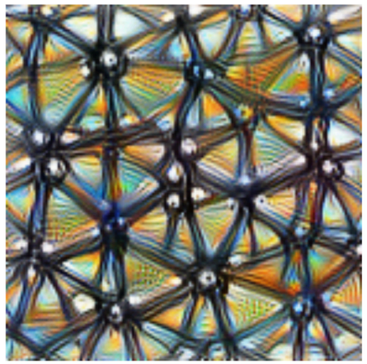
Feature-Visualization





Neuron

`layer`$_n$`[x,y,z]`



Channel

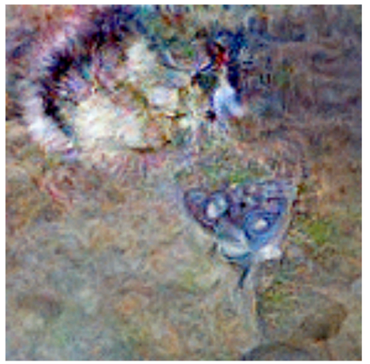
`layer`$_n$`[:,:,z]`

Layer/DeepDream

`layer`$_n$`[:,:,:]`^{2}

Class Logits

`pre_softmax[k]`

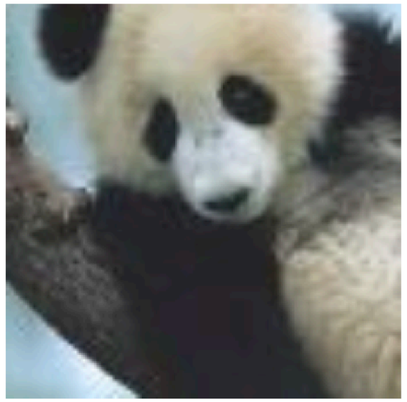


Class Probability

`softmax[k]`

[Figure from distill.pub, Olah et al. 2017]

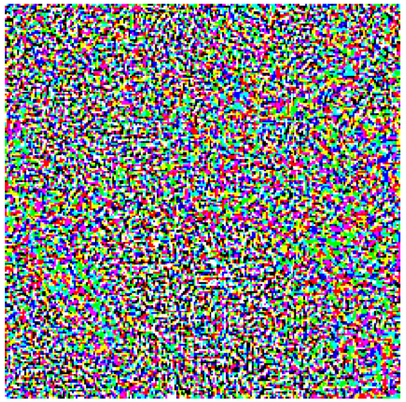
Adversarial attacks



$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

[Figure from Goodfellow et al. 2017]

Batch and Layer normalisation

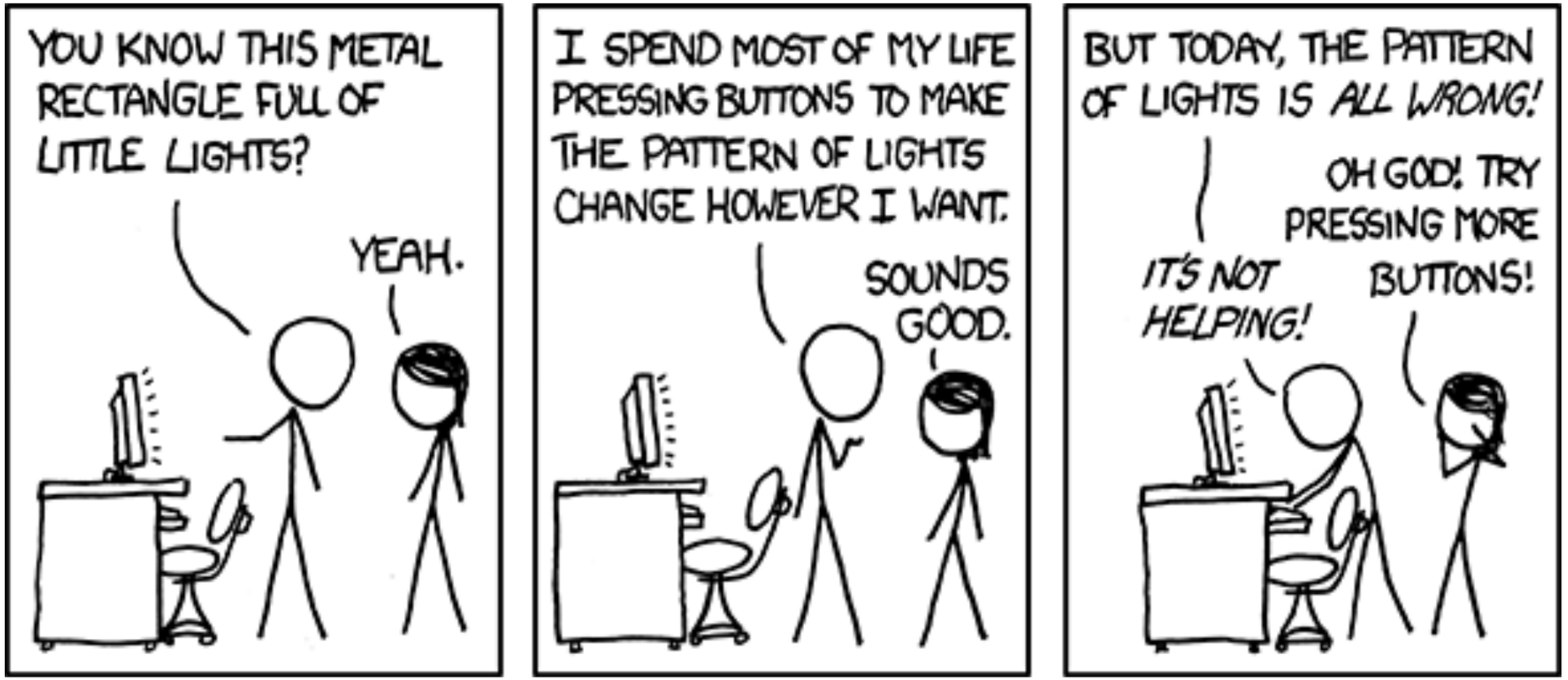
$$y = \frac{x - \mathbb{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} \cdot \gamma + \beta$$

`torch.nn.BatchNorm2d`	Based on Layer Normalization paper (Lei Ba et al. 2016)
`torch.nn.LayerNorm`	Based on Batch Normalization paper (Ioffe and Szegedy 2016)

Resources

- An Introduction to Group Equivariant Deep Learning [https://uvagedl.github.io/]
- Explanations and visualizations of machine learning topics [https://distill.pub/] and [https://colah.github.io/]
- But what is a convolution? [video by 3blue1brown https://youtu.be/KuXjwB4LzSA]
-

Thank you!



xkcd 722, Computer problems