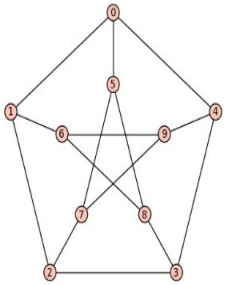
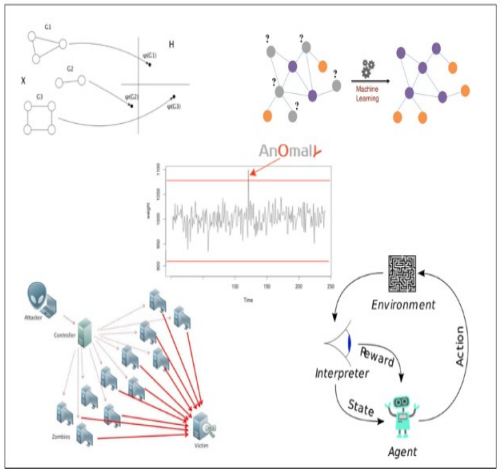
Convolutional Neural Networks

Pierre Pereira
Université Côte d'Azur / Inria Coati
pierre.pereira@inria.fr





$$\min \sum_{e \in \mathcal{E}} y_e$$

$$\text{s.t.} \sum_{a \in A_i^+(u)} f_a^i - \sum_{a \in A_i^-(u)} f_a^i = \begin{cases} |V_i| - 1 & \text{if } u = s_i \\ -1 & \text{if } u \neq s_i \end{cases} \quad \forall u \in V_i,\ V_i \in C$$

$$f_a^i \leq |V_i| \cdot x_a, \quad \forall V_i \in C, a \in A$$

$$x_{(u,v)} \leq y_{uv}, \quad \forall uv \in \mathcal{E}$$

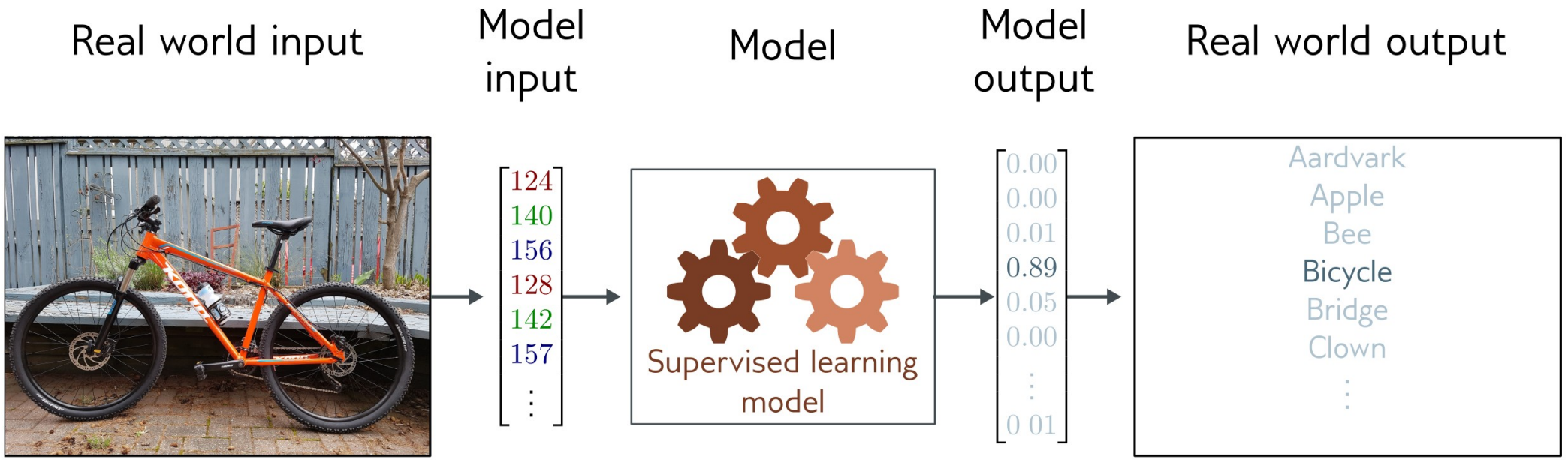
$$x_{(v,u)} \leq y_{uv}, \quad \forall uv \in \mathcal{E}$$

Sources multiples, principalement Polytechnique Montréal

Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

Image classification



- Multiclass classification problem (discrete classes, >2 possible classes)
- Convolutional network

Object detection

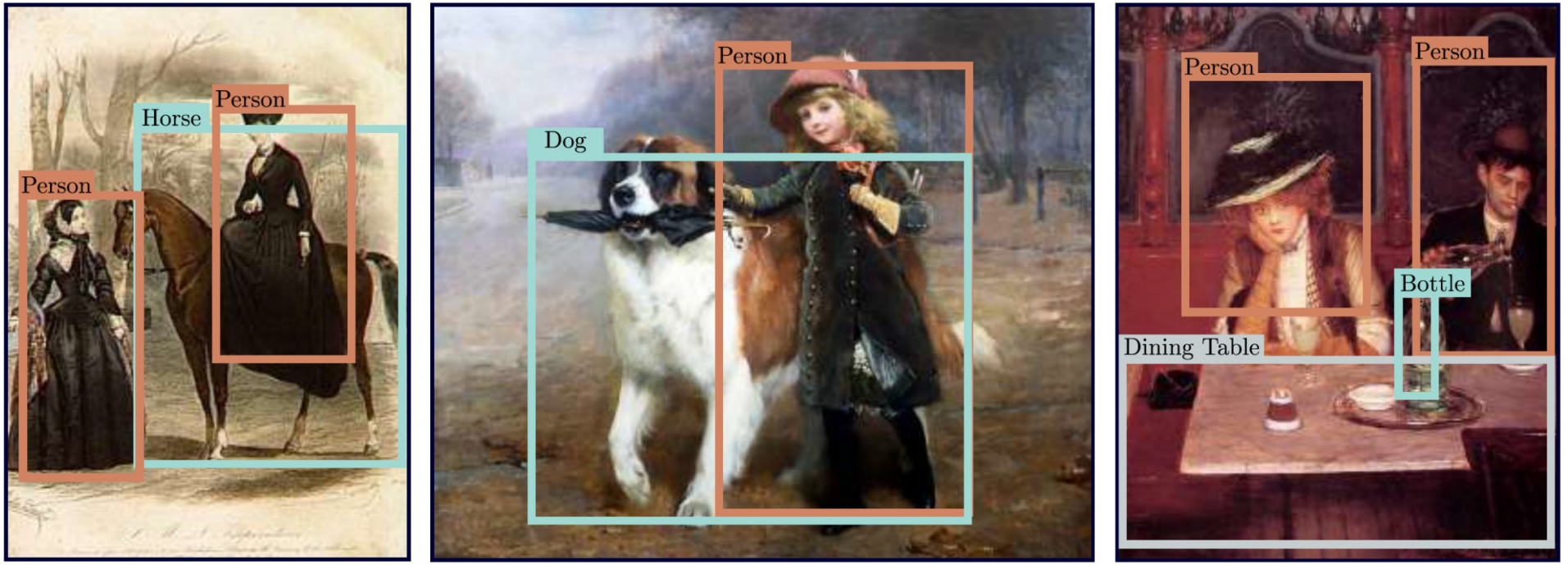
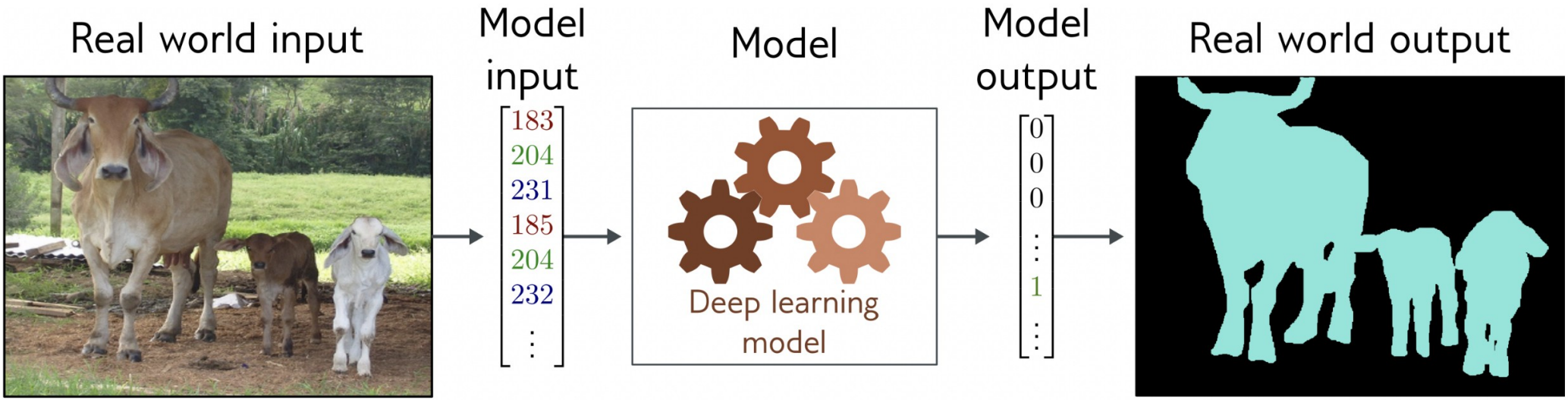


Image segmentation



- Multivariate binary classification problem (many outputs, two discrete classes)
- Convolutional encoder-decoder network

Networks for images

- Problems with fully-connected networks

1. Size
   - 224x224 RGB image = 150,528 dimensions
   - Hidden layers generally larger than inputs
   - One hidden layer = 150,520x150,528 weights -- 22 billion
2. Nearby pixels statistically related
   - But could permute pixels and relearn and get same results with FC
3. Should be stable under transformations
   - Don’t want to re-learn appearance at different parts of image

Convolutional networks

- Parameters only look at local image patches
- Share parameters across image

Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

Invariance

- A function f[x] is invariant to a transformation t[] if:

$$\mathbf{f}[\mathbf{t}[\mathbf{x}]] = \mathbf{f}[\mathbf{x}]$$

i.e., the function output is the same even after the transformation is applied.

Invariance example

e.g., Image classification

- Image has been translated, but we want our classifier to give the same result

Equivariance

- A function f[x] is equivariant to a transformation t[] if:

$$\mathbf{f}\left[\mathbf{t}[\mathbf{x}]\right] = \mathbf{t}\left[\mathbf{f}[\mathbf{x}]\right]$$

i.e., the output is transformed in the same way as the input

Equivariance example

e.g., Image segmentation

- Image has been translated and we want segmentation to translate with it

Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

Convolution* in 1D

- Input vector **x:**

$$\mathbf{x} = [x_1, x_2, \ldots, x_I]$$

- Output is weighted sum of neighbors:

$$z_i = \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}$$

- Convolutional kernel or filter:

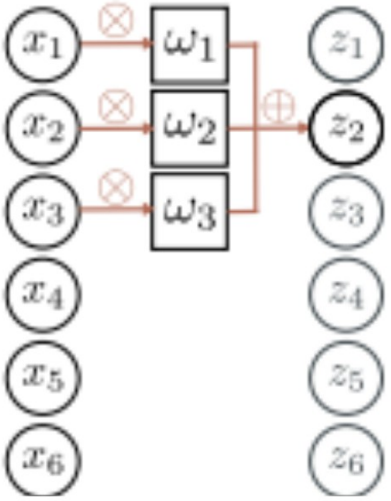
$$\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T$$

Kernel size = 3

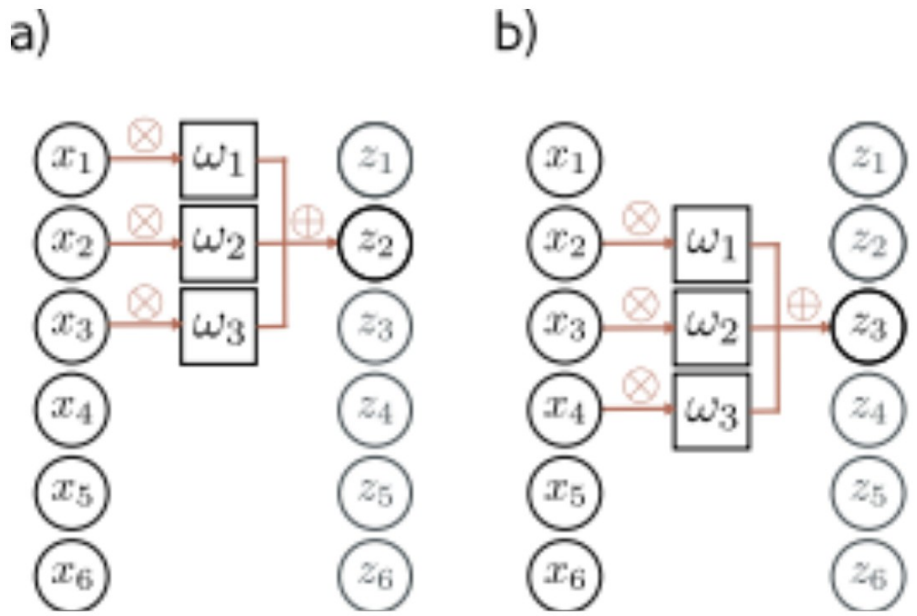
* Not really technically convolution

Convolution with kernel size 3

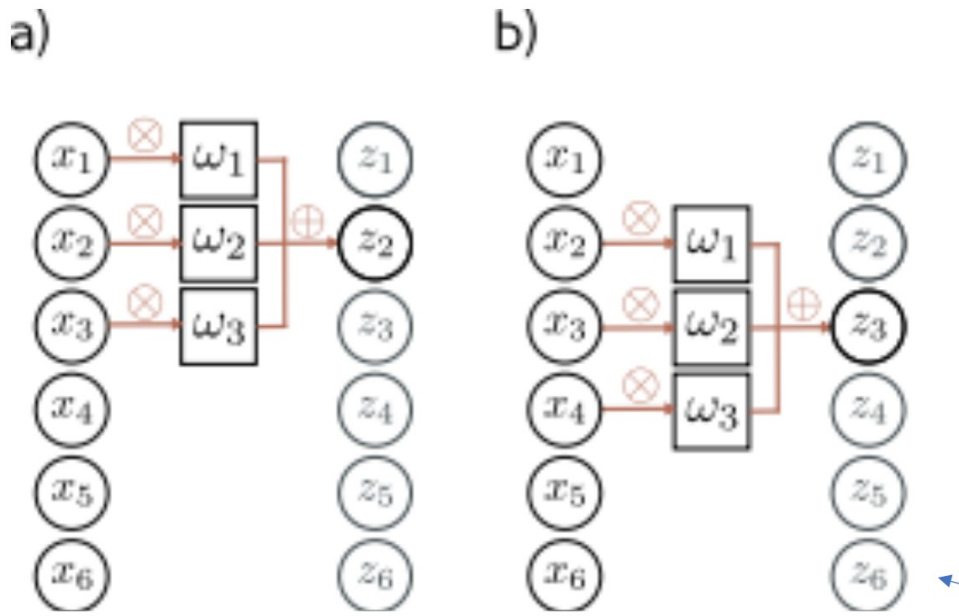
a)



Convolution with kernel size 3



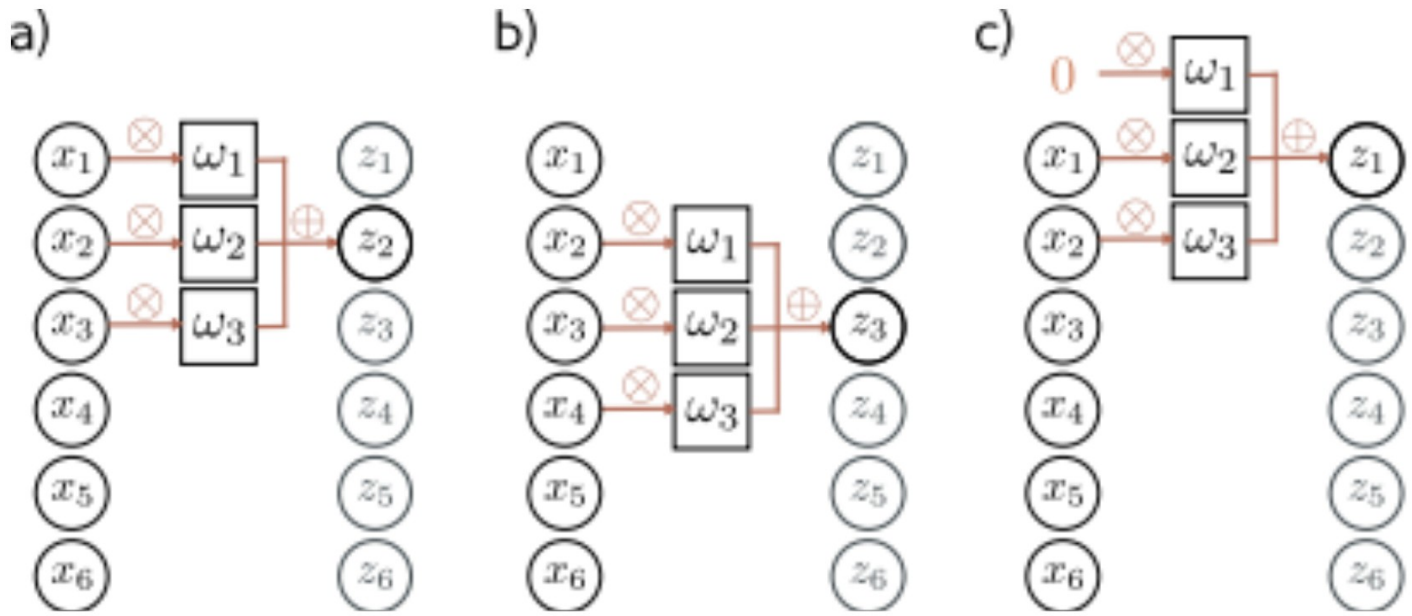
Convolution with kernel size 3



Equivariant to translation of input

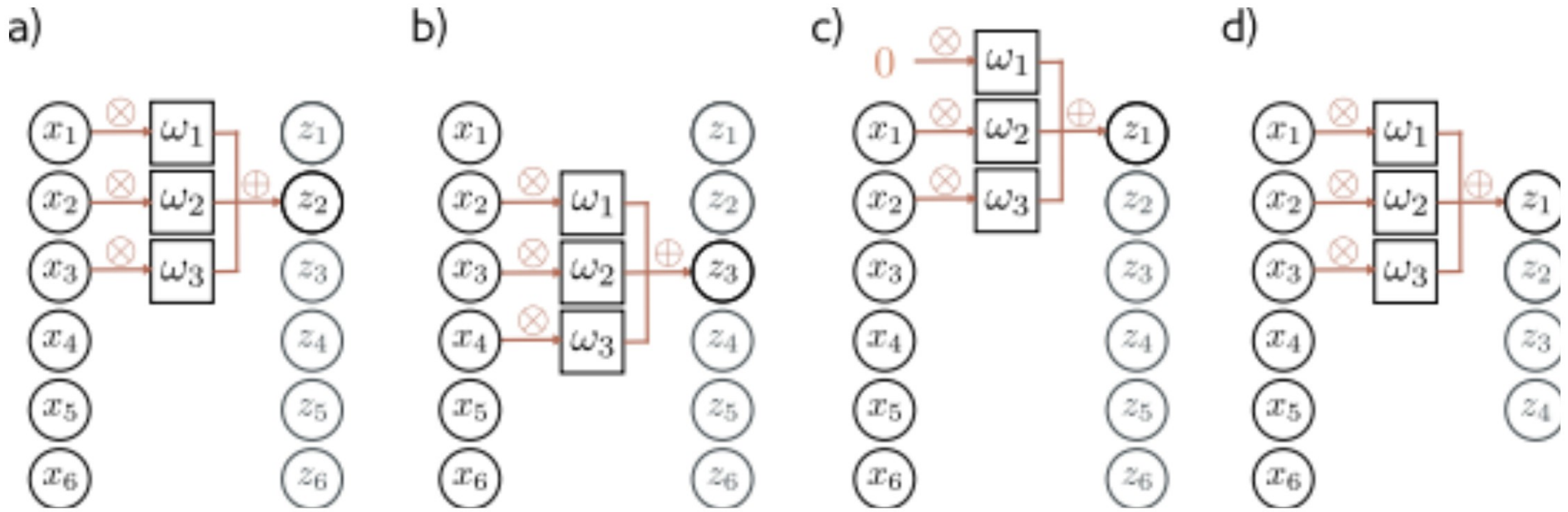
$$\mathbf{f}[\mathbf{t}[\mathbf{x}]] = \mathbf{t}[\mathbf{f}[\mathbf{x}]]$$

Zero padding



Treat positions that are beyond end of the input as zero.

“Valid” convolutions

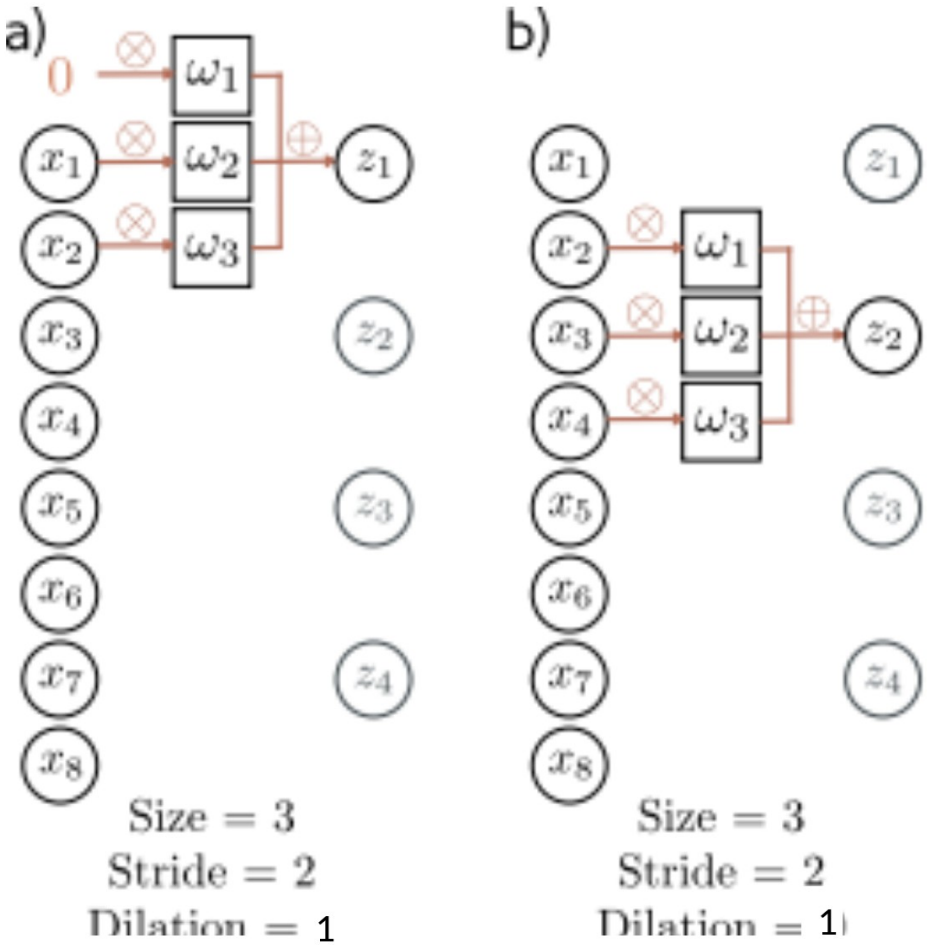


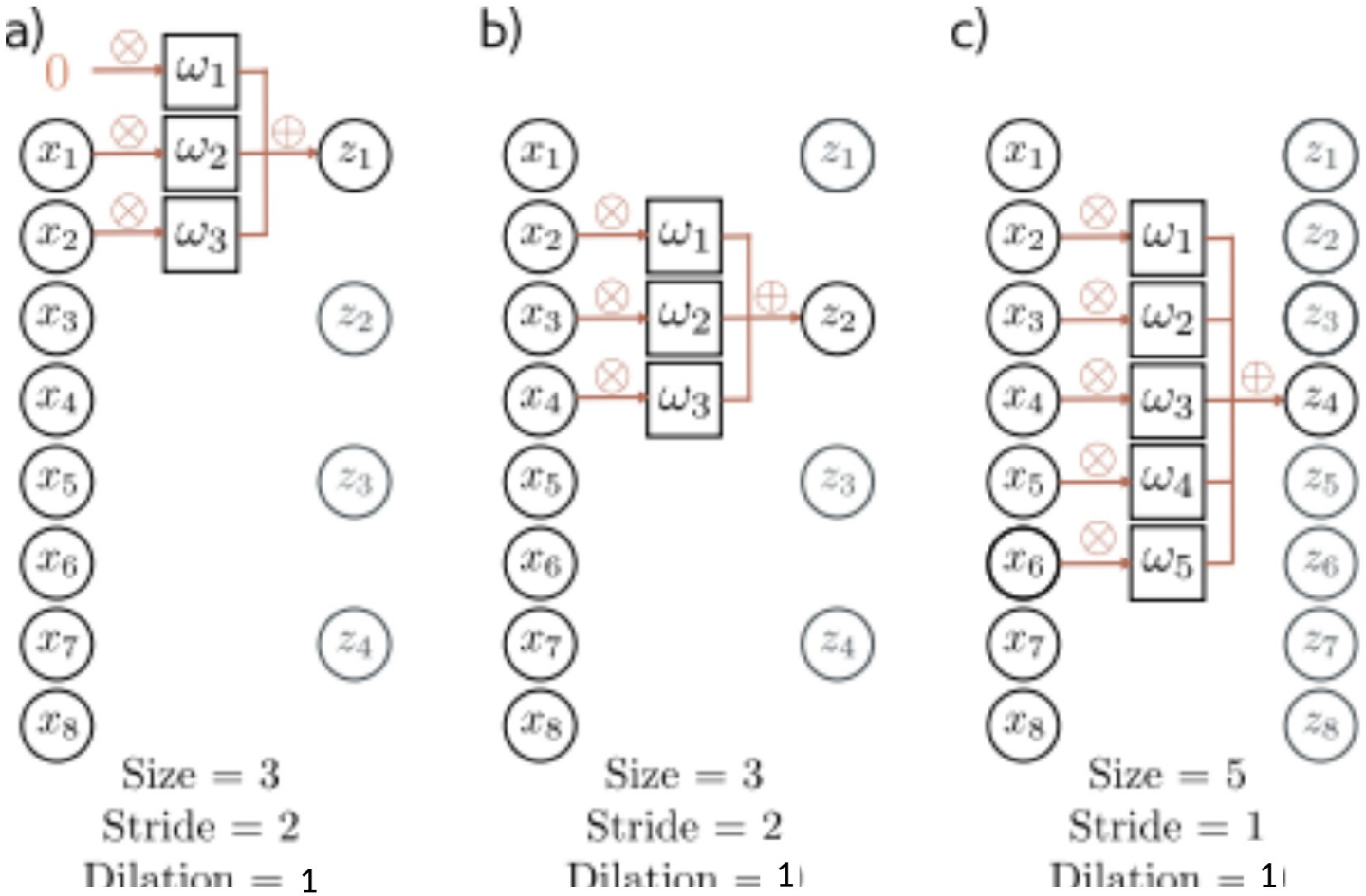
Only process positions where kernel falls in image (smaller output).

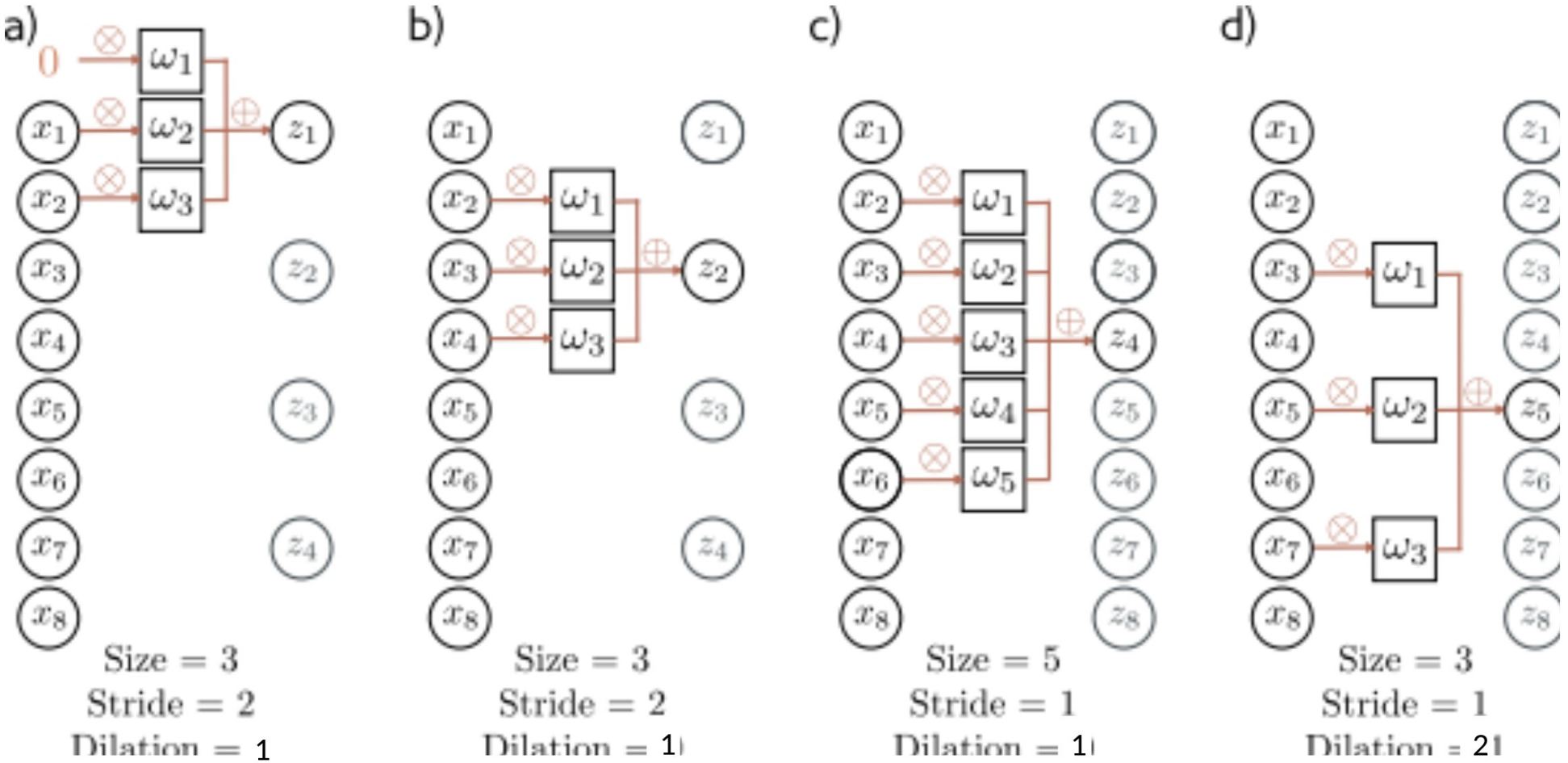
Stride, kernel size, and dilation

- Stride = shift by k positions for each output
  - Decreases size of output relative to input
- Kernel size = weight a different number of inputs for each output
  - Combine information from a larger area
  - But kernel size 5 uses 5 parameters
- Dilated or atrous convolutions = intersperse kernel values with zeros
  - Combine information from a larger area
  - Fewer parameters


a)
0
ω_1
ω_2
ω_3
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
z_1
z_2
z_3
z_4
Size = 3
Stride = 2
Dilation = 1


a)
0
ω_1
ω_2
ω_3
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
z_1
z_2
z_3
z_4
Size = 3
Stride = 2
Dilation = 1
b)
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
ω_1
ω_2
ω_3
z_1
z_2
z_3
z_4
Size = 3
Stride = 2
Dilation = 1

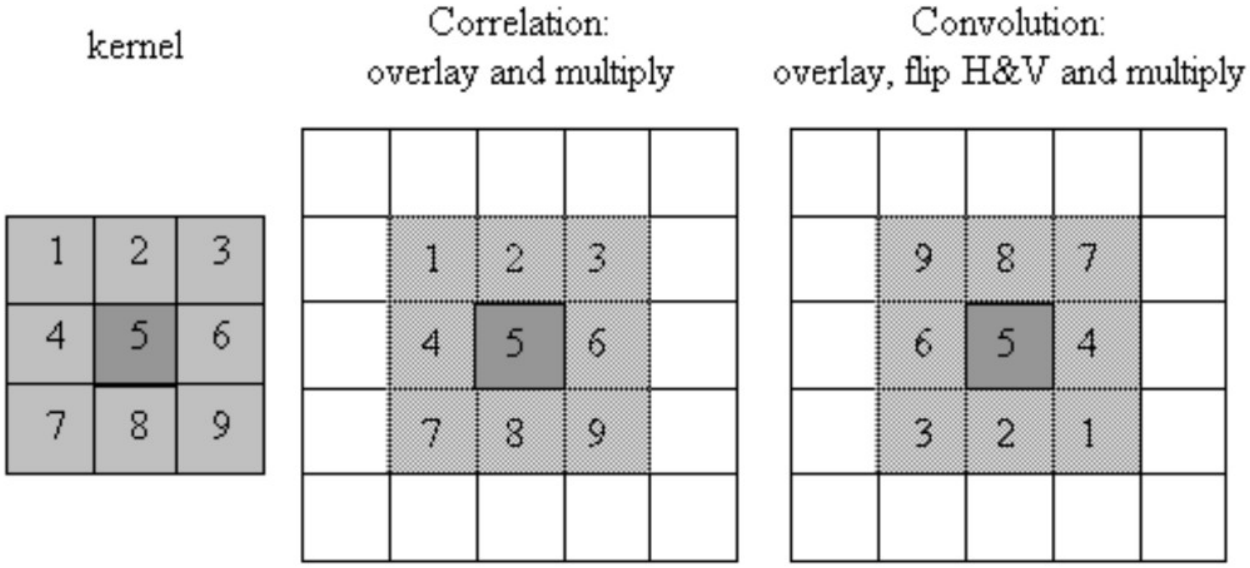

a)
0
ω_1
ω_2
ω_3
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
z_1
z_2
z_3
z_4
Size = 3
Stride = 2
Dilation = 1
b)
ω_1
ω_2
ω_3
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
z_1
z_2
z_3
z_4
Size = 3
Stride = 2
Dilation = 1
c)
ω_1
ω_2
ω_3
ω_4
ω_5
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
z_1
z_2
z_3
z_4
z_5
z_6
z_7
z_8
Size = 5
Stride = 1
Dilation = 1


a)
b)
c)
d)
Size = 3
Stride = 2
Dilation = 1
Size = 3
Stride = 2
Dilation = 1
Size = 5
Stride = 1
Dilation = 1
Size = 3
Stride = 1
Dilation = 21

Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

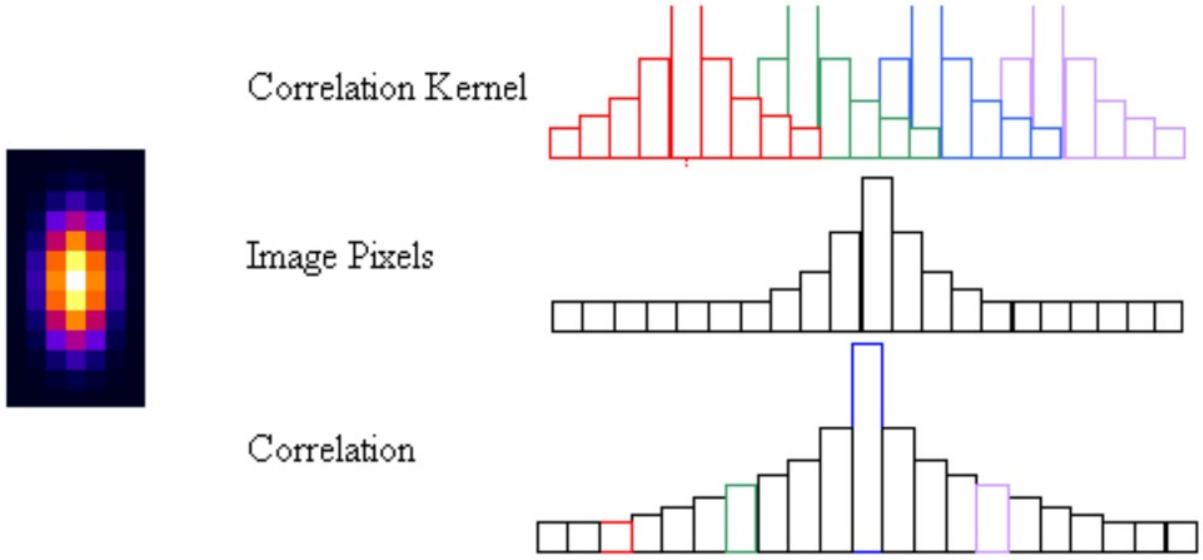
Intuition: la corrélation croisée versus le produit de convolution



Merci à U. Rochester

Quel est le résultat?

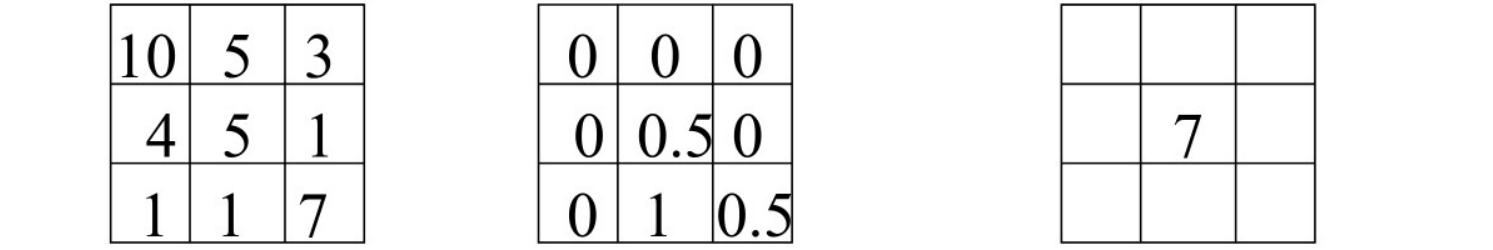
- Lorsque l'image correspond au noyau, il existe une corrélation positive élevée



- Mais, il est aussi possible de penser de l'opération comme une forme de filtrage linéaire

Linear functions

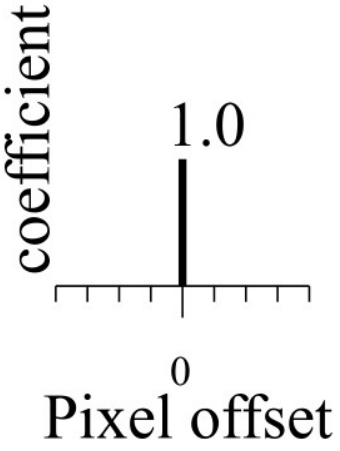
- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the "convolution kernel".



Local image data kernel Modified image data

Linear filtering (warm-up slide)

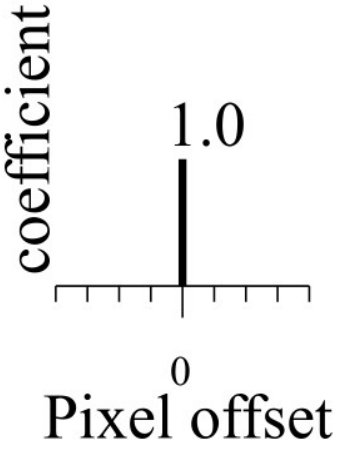
original



?

Linear filtering (warm-up slide)

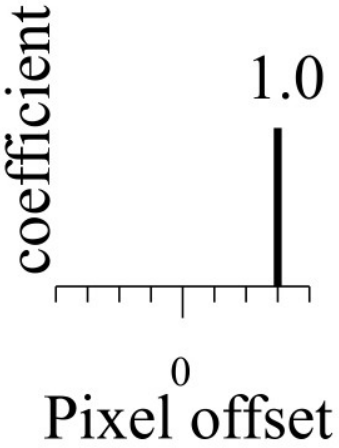
original



Filtered
(no change)

Linear filtering

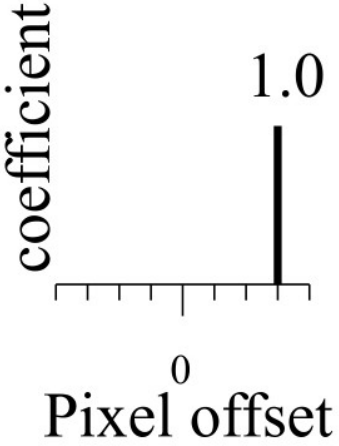
original

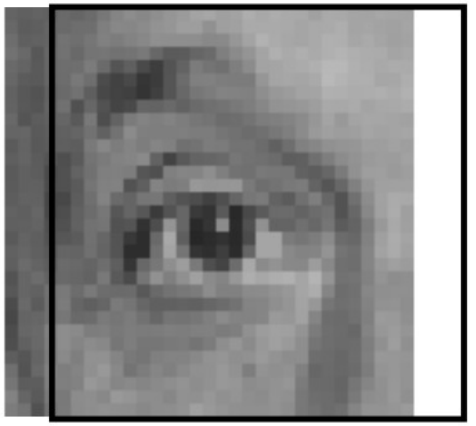


?

shift

original

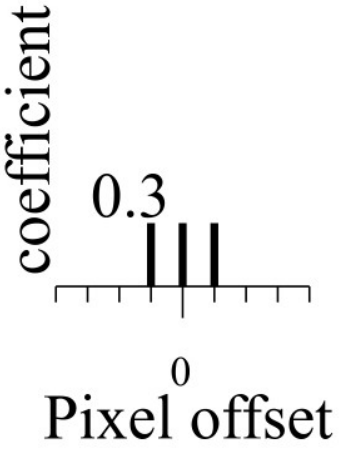




shifted

Linear filtering

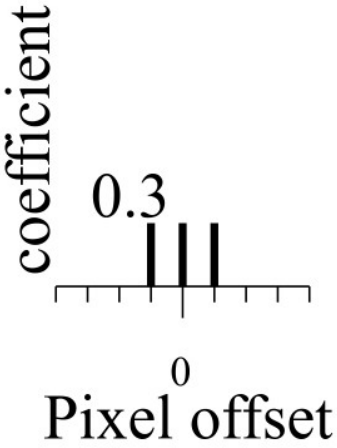
original



?

Blurring

original



Blurred (filter applied in both dimensions).

Consider now the following filters

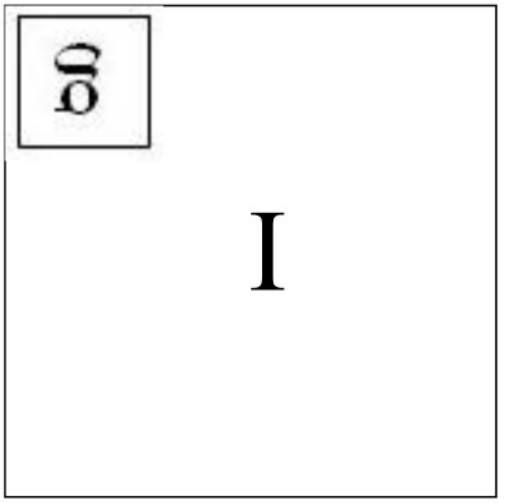
$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

From left to right: a) A photograph, b) the photograph filtered with the Sobel operator $\mathbf{G}_x$, which emphasizes vertical edges, c) the photograph filtered with the Sobel operator $\mathbf{G}_y$, which emphasizes horizontal edges, d) the magnitue of the response of the pairs of filters at each spatial location, (but with the intensity flipped so that larger values are darker).

Convolution

$$f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l] g[k,l]$$



Slides from Friedman @ MIT

Convolutional neural networks (CNNs)

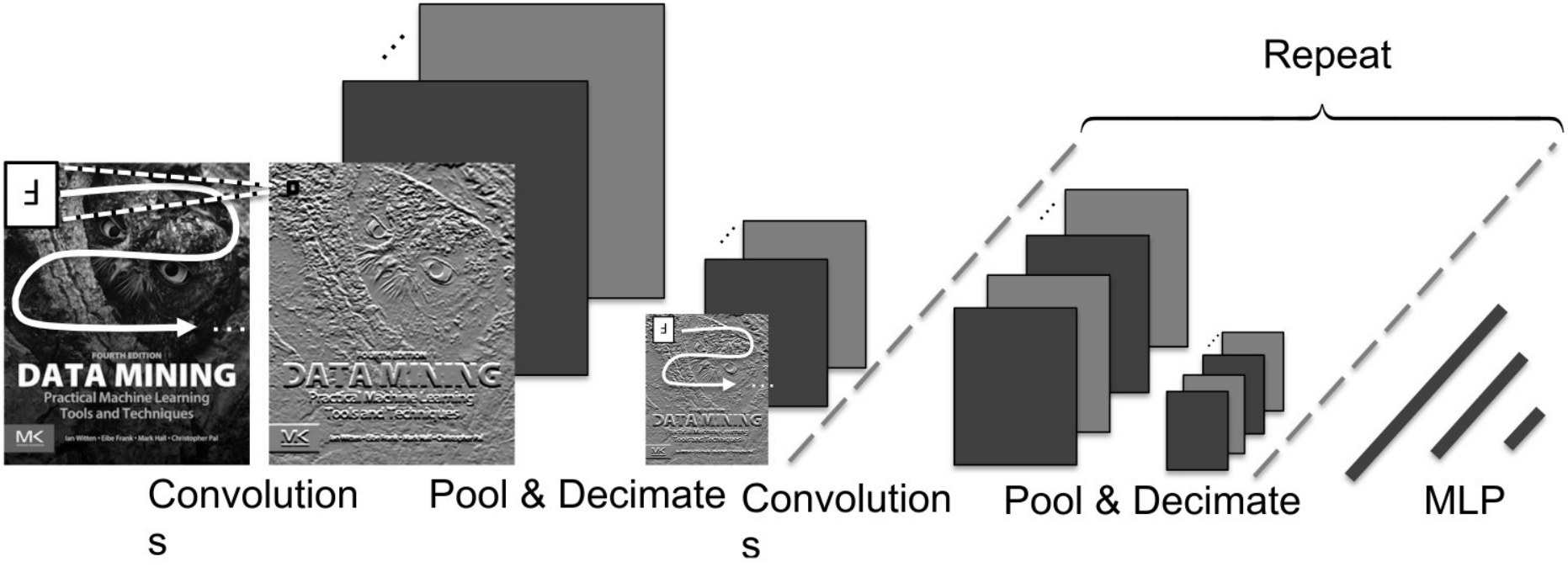
- Are a special kind of feedforward network that has proven *extremely* successful for image analysis
- Imagine filtering an image to detect edges, one could think of edges as a useful set of spatially organized 'features'
- Imagine now if one could learn many such filters jointly along with other parameters of a neural network on top
- Each filter can be implemented by multiplying a relatively small spatial zone of the image by a set of weights and feeding the result to an activation function – just like those discussed above for vanilla feedforward networks
- Because this filtering operation is simply repeated around the image using the same weights, it can be implemented using convolution operations
- The result is a CNN for which it is possible to learn both the filters and the classifier using SGD and the backpropagation algorithm

Deep CNNs

- In a convolutional neural network, once an image has been filtered by several learnable filters, each filter bank's output is often aggregated across a small spatial region, using the average or maximum value.
- Aggregation can be performed within non-overlapping regions, or using subsampling, yielding a lower-resolution layer of spatially organized features—a process that is sometimes referred to as "decimation"
- This gives the model a degree of invariance to small differences as to exactly where a feature has been detected.
- If aggregation uses the max operation, a feature is activated if it is detected anywhere in the pooling zone
- The result can be filtered and aggregated again

A typical CNN architecture

- Many feature maps are obtained from convolving learnable filters across an image
- Results are aggregated or pooled & decimated
- Process repeats until last set of feature maps are given to an MLP for final prediction



Starting simply: image filtering

- When an image is filtered, the output can be thought of as another image that contains the filter's response at each spatial location
- Consider filtering a 1D vector **x** by multiplication with a matrix **W** that has a special structure, such as

$$\mathbf{y} = \mathbf{W}\mathbf{x} = \begin{bmatrix} w_1 & w_2 & w_3 & & & \\ & w_1 & w_2 & w_3 & & \\ & & & \ddots & & \\ & & & w_1 & w_2 & w_3 \end{bmatrix}$$

where the elements left blank in the matrix above are zero and we have used a simple filter having only three non-zero coefficients and a "stride" of one

Understanding Convolution versus Deconvolution

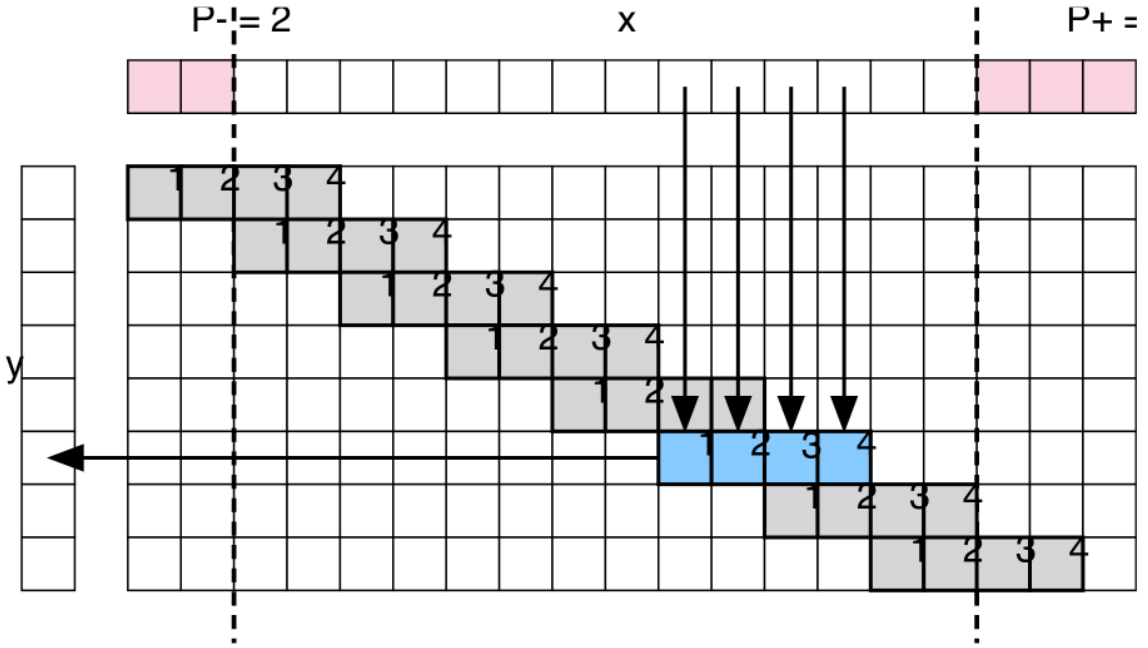


Figure 4.1: **Convolution.** The figure illustrates the process of filtering a 1D signal $\mathbf{x}$ by a filter f to obtain a signal $\mathbf{y}$. The filter has $H' = 4$ elements and is applied with a stride of $S_h = 2$ samples. The purple areas represented padding $P_- = 2$ and $P_+ = 3$ which is zero-filled. Filters are applied in a sliding-window manner across the input signal. The samples of $\mathbf{x}$ involved in the calculation of a sample of $\mathbf{y}$ are shown with arrow. Note that the rightmost sample of $\mathbf{x}$ is never processed by any filter application due to the sampling step. While in this case the sample is in the padded region, this can happen also without padding.

Figure from the MatConvNet documentation.

Convolution Transpose

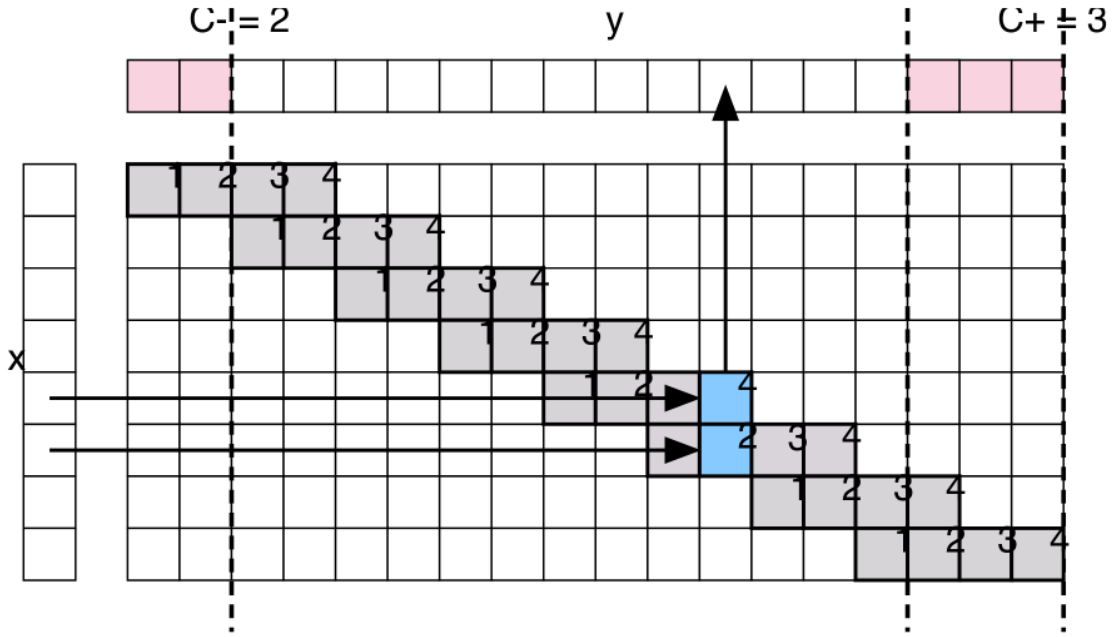


Figure 4.2: **Convolution transpose.** The figure illustrates the process of filtering a 1D signal x by a filter f to obtain a signal y. The filter is applied as a sliding-window, forming a pattern which is the transpose of the one of fig. 4.1. The filter has $H' = 4$ samples in total, although each filter application uses two of them (blue squares) in a circulant manner. The purple areas represent crops with $C_- = 2$ and $C_+ = 3$ which are discarded. The arrows exemplify which samples of x are involved in the calculation of a particular sample of y. Note that, differently from the forward convolution fig. 4.1, there is no need to add padding to the input array; instead, the convolution transpose filters can be seen as being applied with maximum input padding (more would result in zero output values), and the latter can be reduced by cropping the output instead.

Figure from the MatConvNet documentation.

Visualizing the filters learned by a CNN

- Learned edge-like filters and texture-like filters are frequently observed in the early layers of CNNs trained using natural images
- Since each layer in a CNN involves filtering the feature map below, so as one moves up the receptive fields become larger
- Higher- level layers learn to detect larger features, which often correspond to textures, then small pieces of objects

First Layer

Second Layer

Third Layer

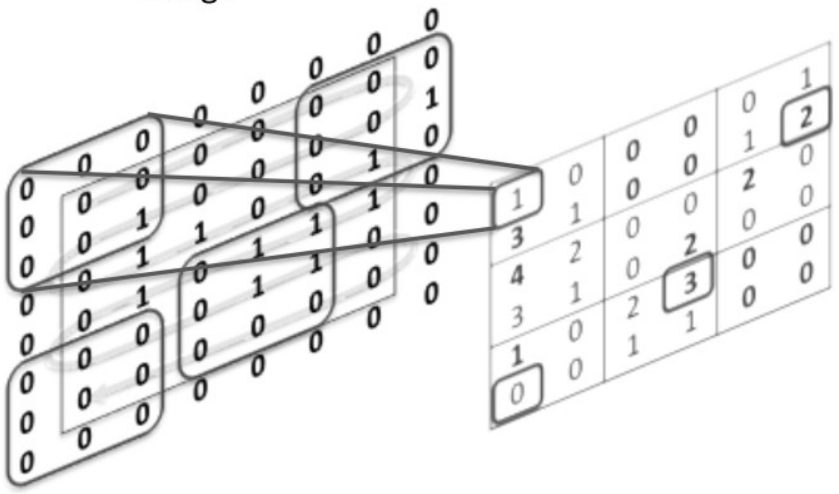
(Imagery kindly provided by Matthew Zeiler)

- Above are the strongest activations of random neurons projecting the activation back into image space using the deconvolution approach of Zeiler and Fergus

Simple example of: convolution, pooling, and decimation operations

Filter

-1	0	1
-2	0	2
-1	0	1



Max Pooling

3	3	0	0	2	2
3	3	0	0	2	2
4	4	2	2	2	2
4	4	2	2	2	2
1	1	3	3	0	0
1	1	3	3	0	0

Decimation

3	0	2
4	2	2
1	3	0

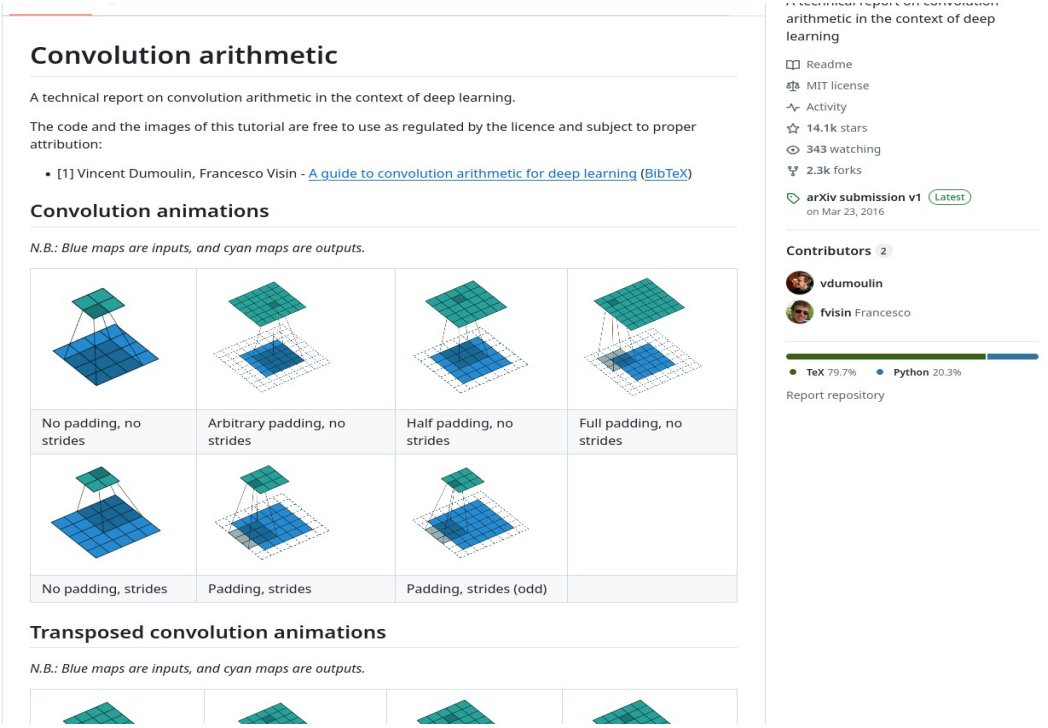
- An image is convolved with a filter; curved rectangular regions in the first large matrix depict a random set of image locations
- Maximum values within small 2×2 regions are indicated in bold in the central matrix
- The results are pooled, using max-pooling then decimated by a factor of two, to yield the final matrix

Pooling and subsampling layers

- What are the consequences of backpropagating gradients through max or average pooling layers?
- In the former case, the units that are responsible for the maximum within each zone *j*, *k* —the "winning units"— get the backpropagated gradient
- For average pooling, the averaging is simply a special type of convolution with a fixed kernel that computes the (possibly weighted) average of pixels in a zone
  - the required gradients are therefore like std conv. layers
- The subsampling step either samples every n^{th} output, or avoids needless computation by only evaluating every n^{th} pooling computation

Convolutions with padding and dilations

See the animations there:
https://github.com/vdumoulin/conv_arithmetic



Convolution arithmetic

A technical report on convolution arithmetic in the context of deep learning.

The code and the images of this tutorial are free to use as regulated by the licence and subject to proper attribution:

- [1] Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning (BibTeX)

Convolution animations

N.B.: Blue maps are inputs, and cyan maps are outputs.

No padding, no strides	Arbitrary padding, no strides	Half padding, no strides	Full padding, no strides
No padding, strides	Padding, strides	Padding, strides (odd)	

Transposed convolution animations

N.B.: Blue maps are inputs, and cyan maps are outputs.

arithmetic in the context of deep learning

Readme
MIT license
Activity
14.1k stars
343 watching
2.3k forks

arXiv submission v1 Latest
on Mar 23, 2016

Contributors 2

vdumoulin
fvisin Francesco

TeX 79.7% Python 20.3%

Report repository

Implementing CNNs

- Convolutions are very well suited for acceleration using GPUs
- Since graphics hardware can accelerate convolutions by an *order of magnitude* or more over CPU implementations, they play an important often *critical* role in training CNNs
- An experimental turn-around time of days rather than weeks makes a huge difference to model development times!
- Can also be challenging to construct software for learning a convolutional neural network in such a way that alternative architectures can be explored
- Early GPU implementations were hard to extend, newer tools allow for both fast computation and flexible high-level programming primitives
- Many software tools allow gradient computations and the backpropagation algorithm for large networks to be almost completely automated.

CNNs in practice

- LeNet and AlexNet architectures are canonical models
- While CNNs are designed to have a certain degree of translational invariance, augmenting data through global synthetic transformations like the cropping trick can increase performance significantly
- CNNs are usually optimized using mini-batch-based stochastic gradient descent, so practical discussions above about learning deep networks apply
- The use of GPU computing is typically *essential* to accelerate convolution operations significantly
- Resource issues related to the amount of CPU vs. GPU memory available are often important to consider

The ImageNet challenge

- Crucial in demonstrating the effectiveness of deep CNNs
- Problem: recognize object categories in Internet imagery
- The 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification task - classify image from Flickr and other search engines into 1 of 1000 possible object categories
- Serves as a standard benchmark for deep learning
- The imagery was hand-labeled based on the presence or absence of an object belonging to these categories
- There are 1.2 million images in the training set with 732-1300 training images available per class
- A random subset of 50,000 images was used as the validation set, and 100,000 images were used for the test set where there are 50 and 100 images per class respectively

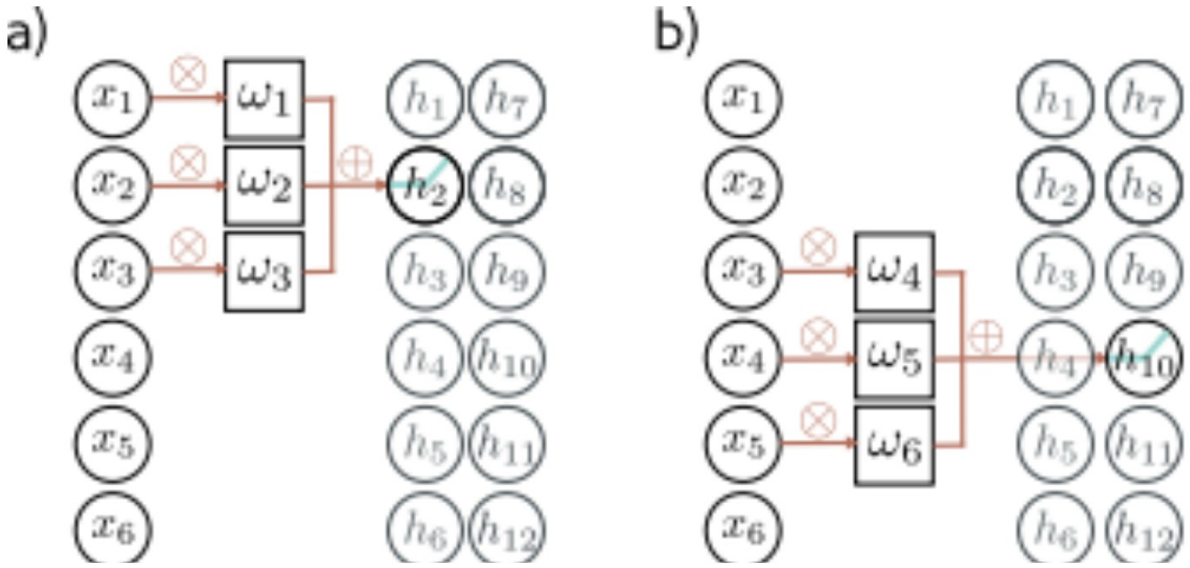
Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

Channels

- The convolutional operation averages together the inputs
- Plus passes through ReLU function
- Has to lose information
- Solution:
  - apply several convolutions and stack them in channels
  - Sometimes also called feature maps

Two output channels, one input channel



Two input channels, one output channel

How many parameters?

- If there are C_i input channels and kernel size K

$$\boldsymbol{\Omega} \in \mathbb{R}^{C_i \times K} \qquad \boldsymbol{\beta} \in \mathbb{R}$$

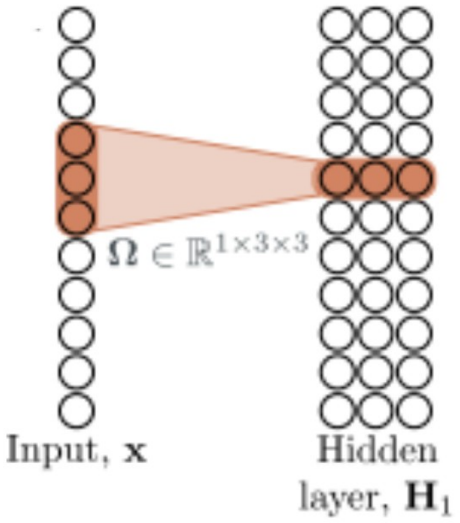
- If there are C_i input channels and C_o output channels

$$\boldsymbol{\Omega} \in \mathbb{R}^{C_i \times C_o \times K} \qquad \boldsymbol{\beta} \in \mathbb{R}^{C_o}$$

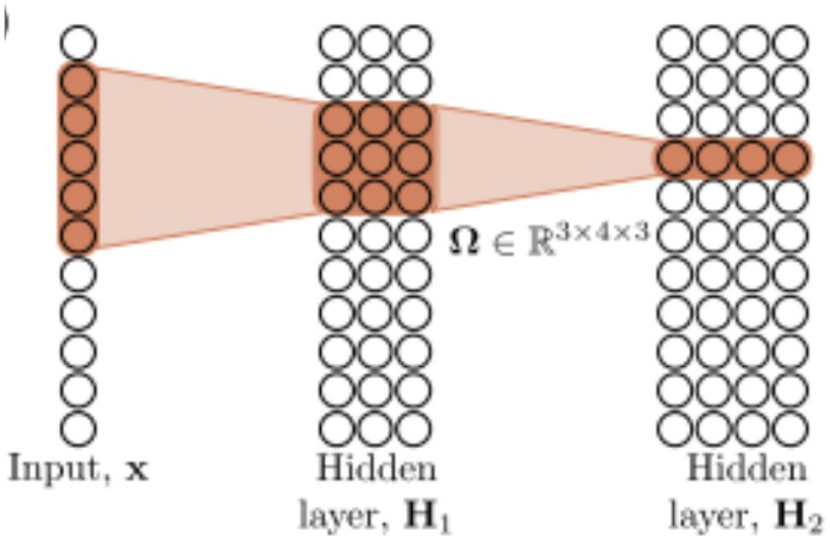
Convolutional networks

- Networks for images
- Invariance and equivariance
- 1D convolution
- Convolutional layers
- Channels
- Receptive fields
- Convolutional network for MNIST 1D

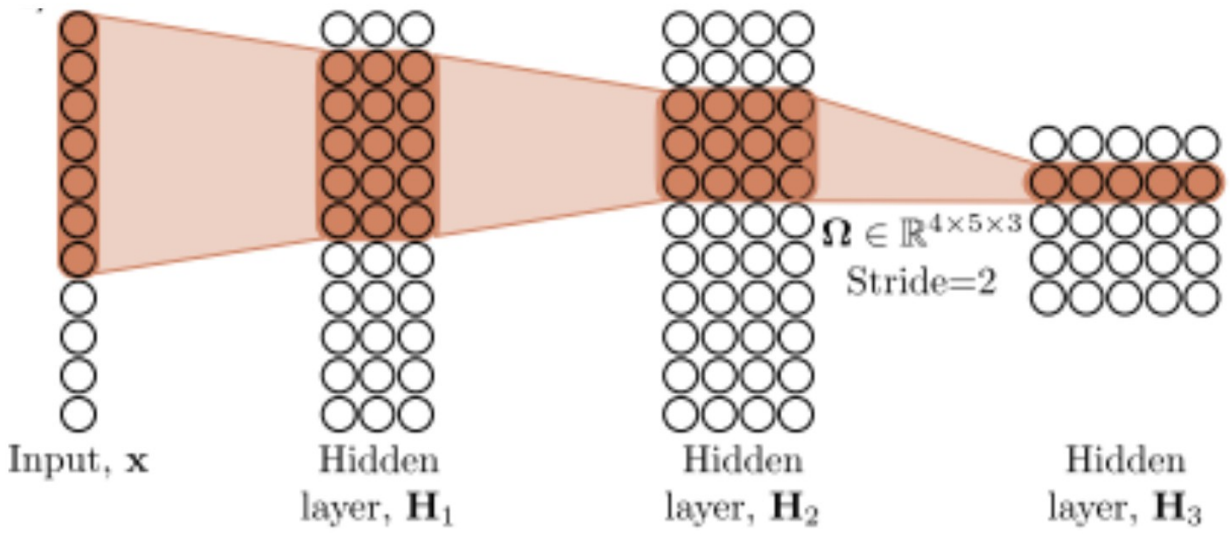
Receptive fields



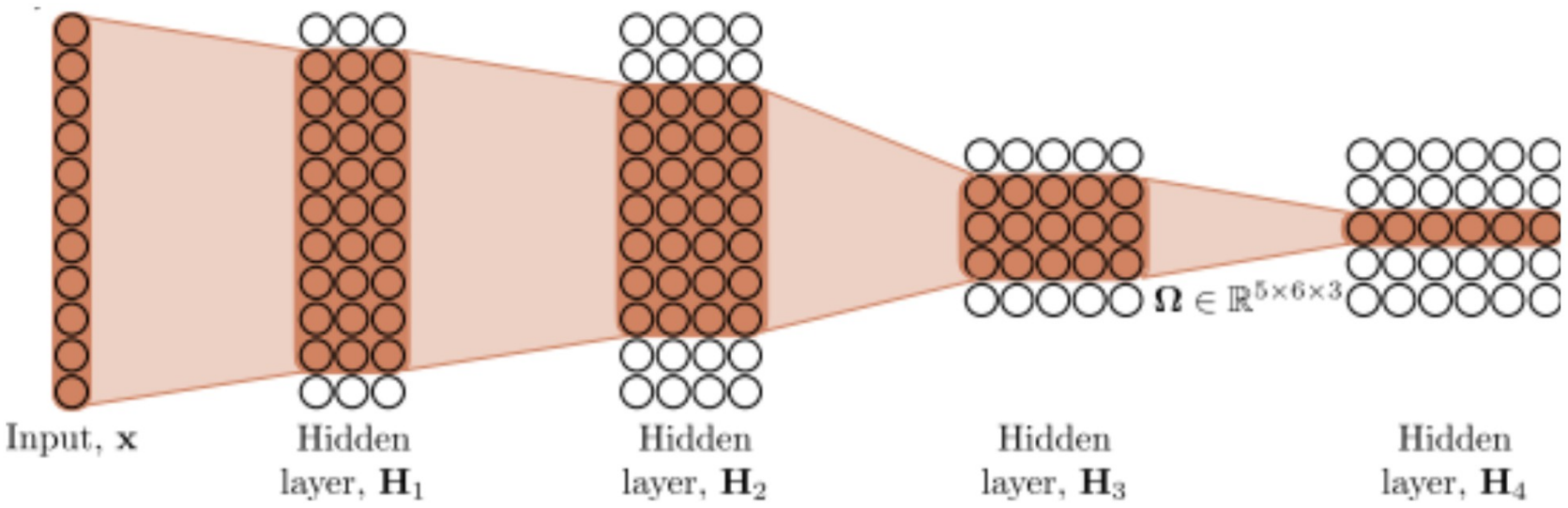
Receptive fields



Receptive fields



Receptive fields



A plateau, then rapid advances

- "Top-5 error" is the % of times that the target label does not appear among the 5 highest-probability predictions
- Visual recognition methods not based on deep CNNs hit a plateau in performance at 25%

Name	Layers	Top-5 Error (%)	References
AlexNet	8	15.3	Krizhevsky et al. (2012)
VGG Net	19	7.3	Simonyan and Zisserman (2014)
ResNet	152	3.6	He et al. (2016)

- Note: the performance for human agreement has been measured at 5.1% top-5 error
- Smaller filters have been found to lead to superior results in deep networks: the methods with 19 and 152 layers use filters of size 3×3

Bibliographic Notes & Further Reading

Convolutional Networks

- Modern convolutional neural networks are widely acknowledged as having their roots with the "neocognitron" proposed by Fukushima (1980); however
- The work of LeCun et al. (1998) on the LeNet convolutional network architecture has been extremely influential.
- The MNIST dataset containing 28×28 pixel images of handwritten digits has been popular in deep learning research community since 1998
- However, it was the ImageNet challenge (Russakovsky et al., 2015), with a variety of much higher resolutions, that catapulted deep learning into the spotlight in 2012.
  - The winning entry from the University of Toronto (Krizhevsky et al. , 2012) processed the images at a resolution of 256×256 pixels.
  - Up till then, CNNs were simply incapable of processing such large volumes of imagery at such high resolutions in a reasonable amount of time.

Bibliographic Notes & Further Reading

Convolutional Networks

- Krizhevsky et al. (2012)'s dramatic ImageNet win used a GPU accelerated convolutional neural networks.
  - This spurred a great deal of development, reflected in rapid subsequent advances in visual recognition performance and on the ImageNet benchmark.
- In the 2014 challenge, the Oxford Visual Geometry Group and a team from Google pushed performance even further using much deeper architectures: 16-19 weight layers for the Oxford group, using tiny 3×3 convolutional filters (Simonyan and Zisserman, 2014); 22 layers, with filters up to 5×5 for the Google team (Szegedy et al., 2015).
- The 2015 ImageNet challenge was won by a team from Microsoft Research Asia (MSRA) using an architecture with 152 layers (He et al., 2015), using tiny 3×3 filters combined with "shortcut" connections that skip over layers, and pooling and decimating the result of multiple layers of small convolution operations

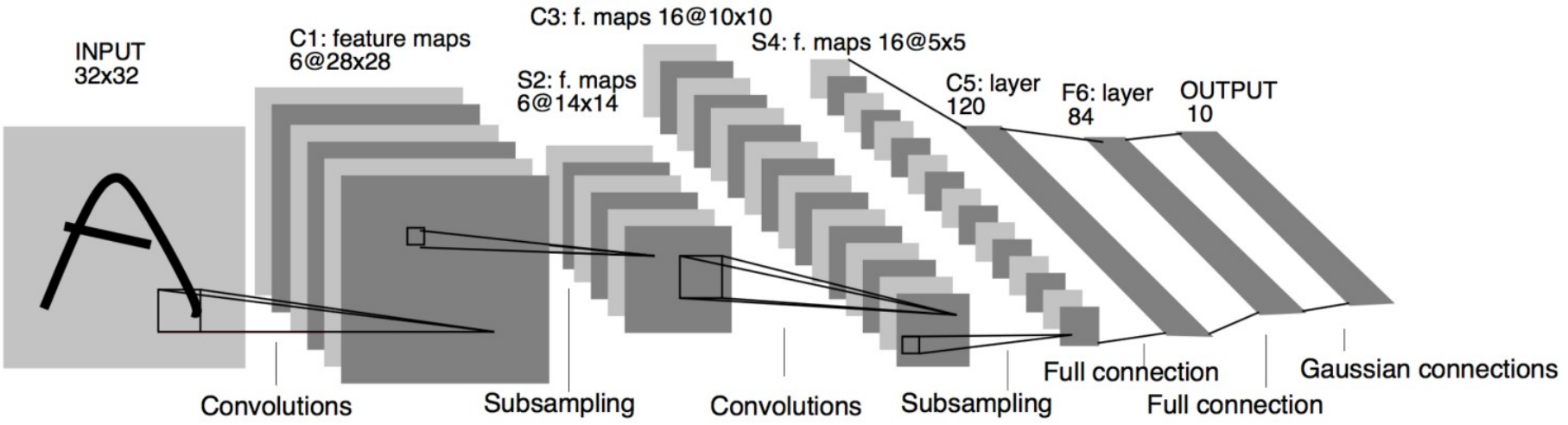
Bibliographic Notes & Further Reading

Convolutional Networks

- Good parameter initialization can be critical for the success of neural networks, as discussed in LeCun et al. (1998)'s classic work and the more recent work of Glorot and Bengio (2010).
- Krizhevsky et al. (2012)'s convolutional network of rectified linear units (ReLUs) initialized weights using 0-mean isotropic Gaussian distributions with a standard deviation of 0.01, and initialized the biases to 1 for most hidden convolutional layers as well as their model's hidden fully connected layers.
- They observed that this initialization accelerated the early phase of learning by providing ReLUs with positive inputs.

ConvNet Canonical Model Zoo

LeNet-5



LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86, no. 11 (1998): 2278-2324. (Note: 18k citations)

Alex Net

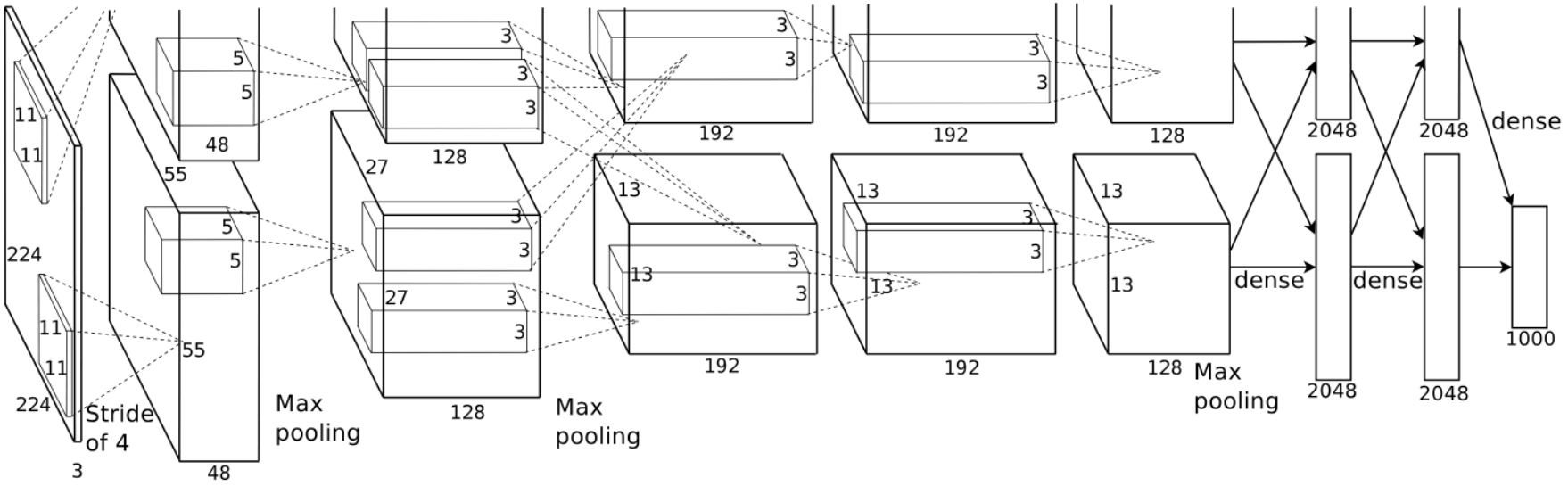
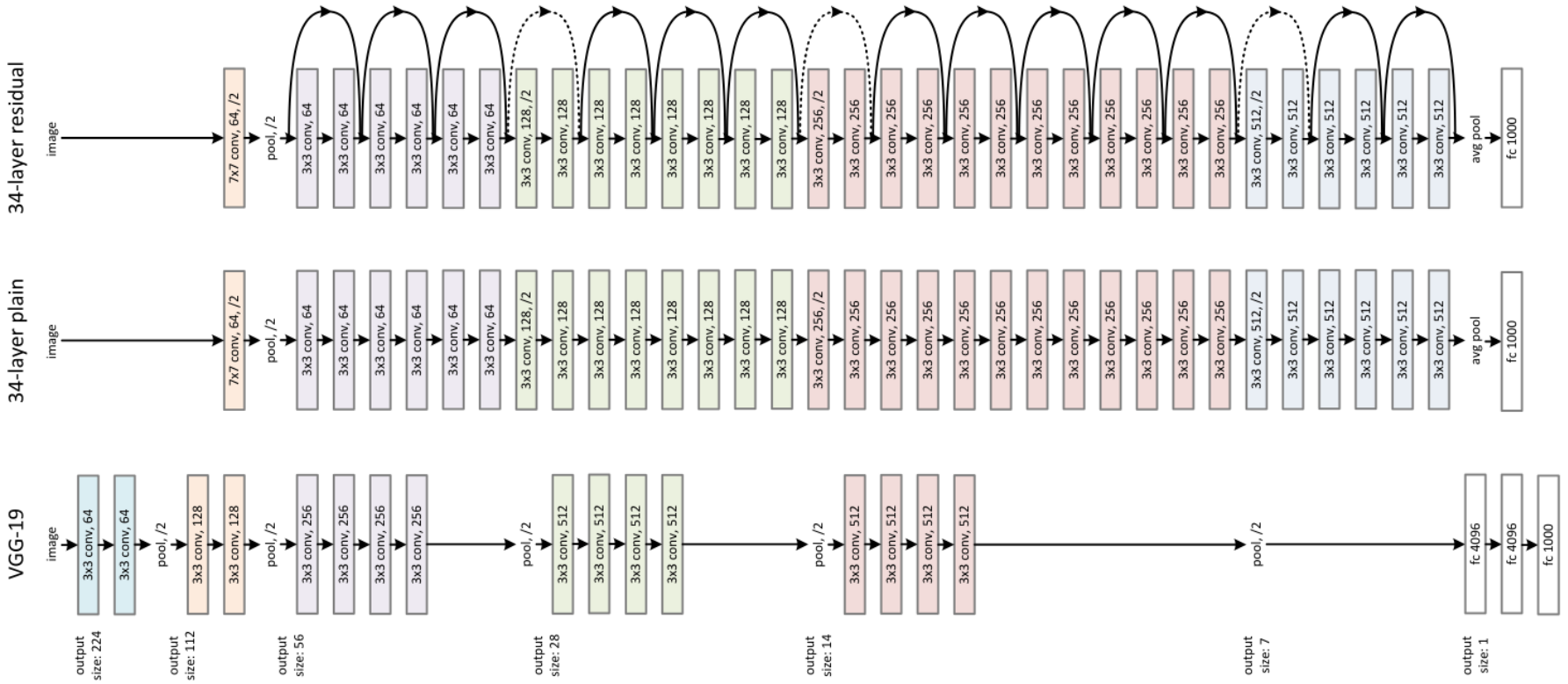


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012. (Note: 39k citations)

From a VGG-19 to Residual Net 34



- VGG Net: Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition

DenseNets

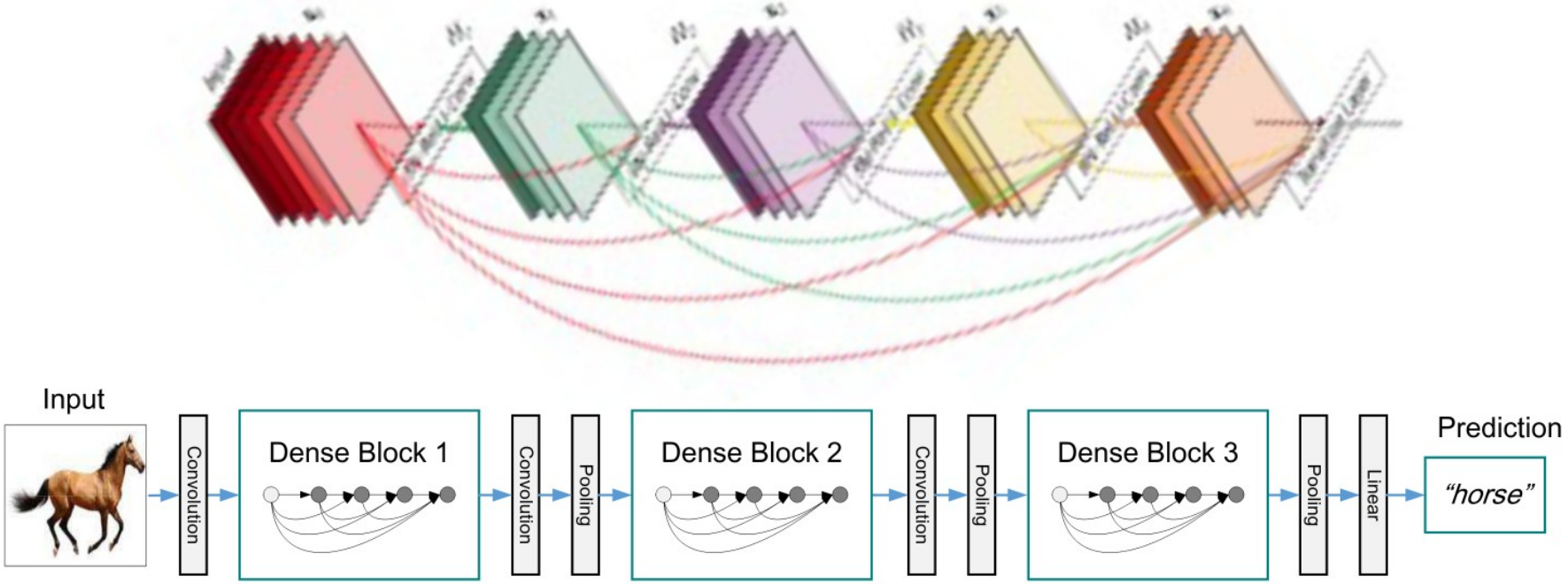
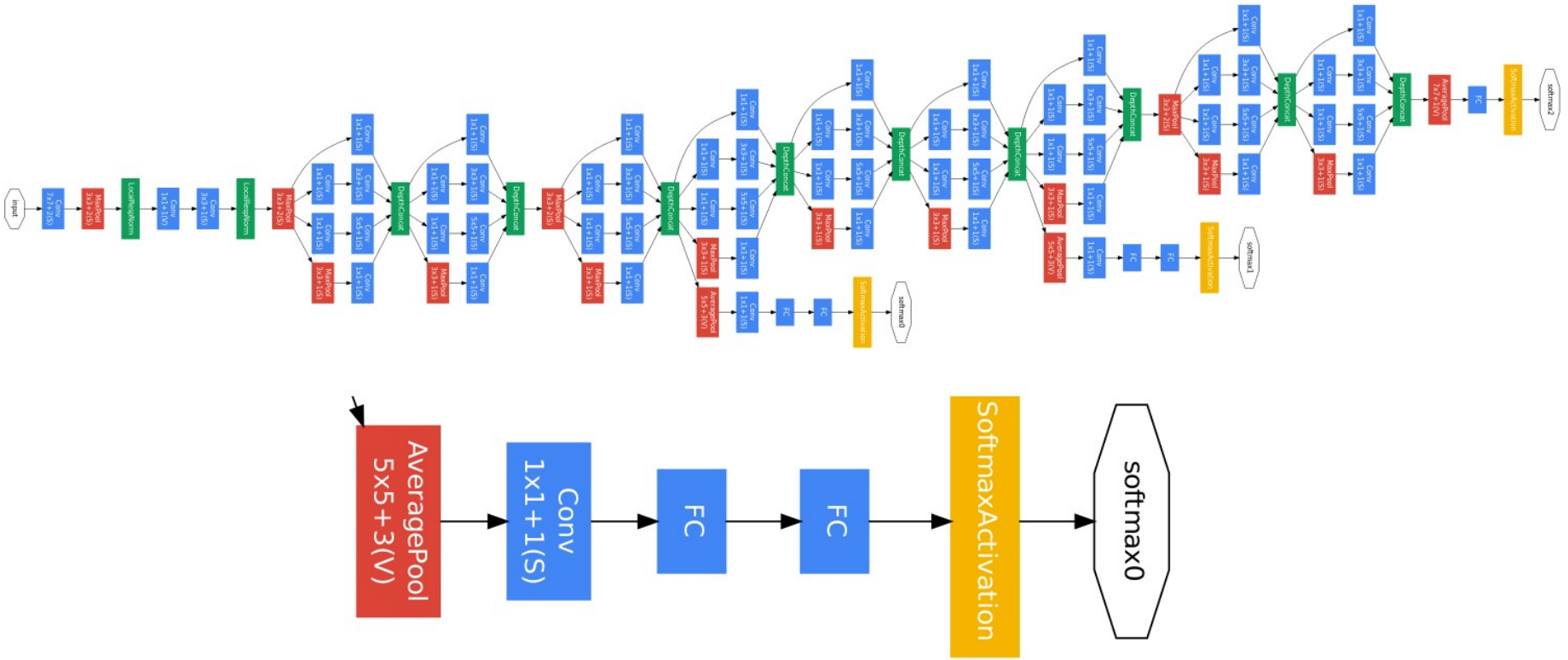


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

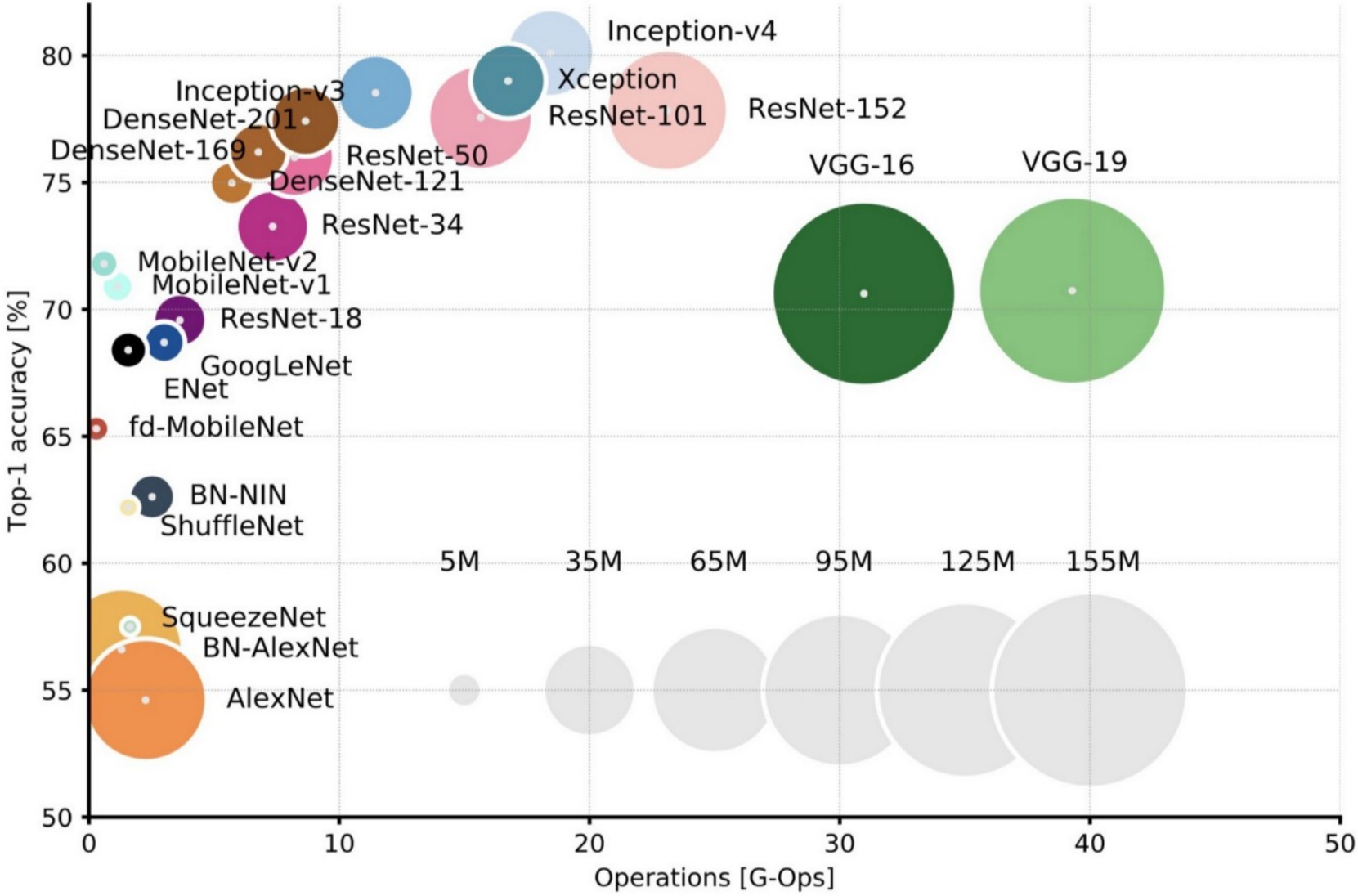
Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

The Inception Architecture



Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9. 2015.

Comparing Models



https://towardsdatascience.com/neural-network-architectures-156e5bad51ba