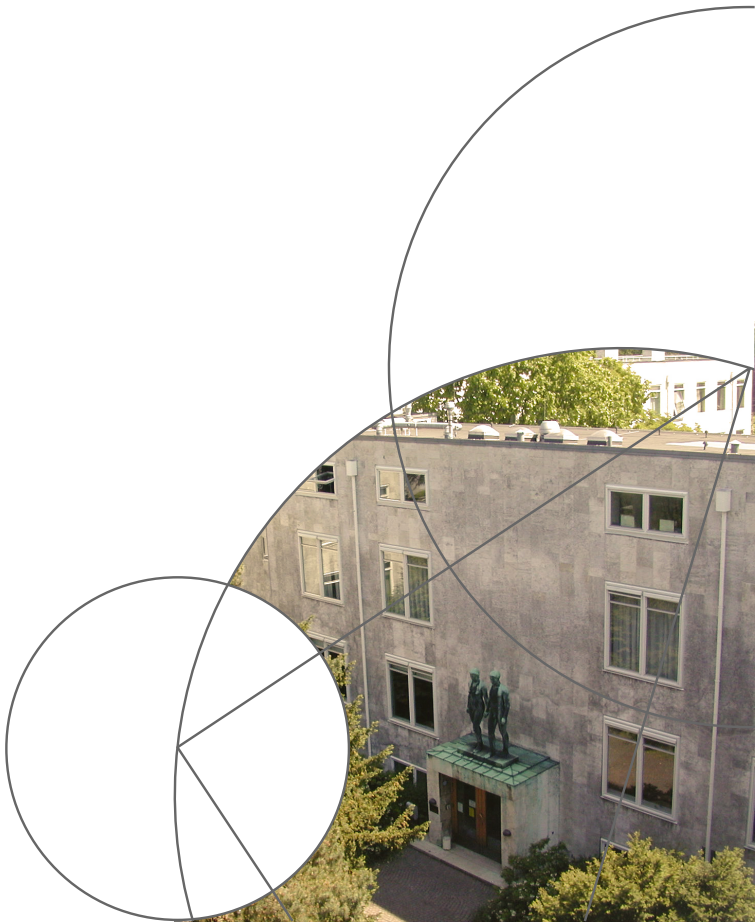
Deep Learning

Part II: Convolutional Neural Networks

Christian Igel
Department of Computer Science



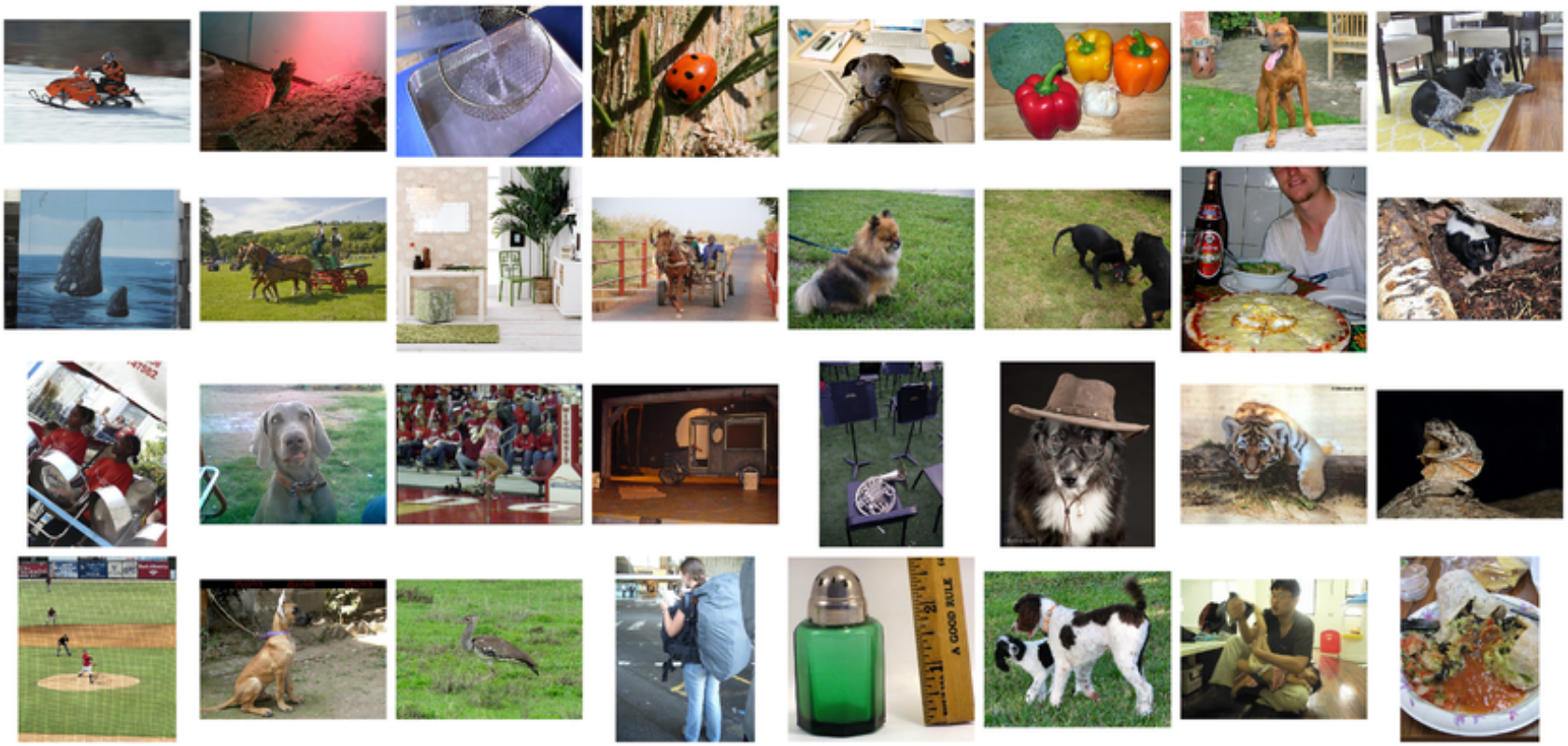
Outline

Outline

Application example: Brain segmentation

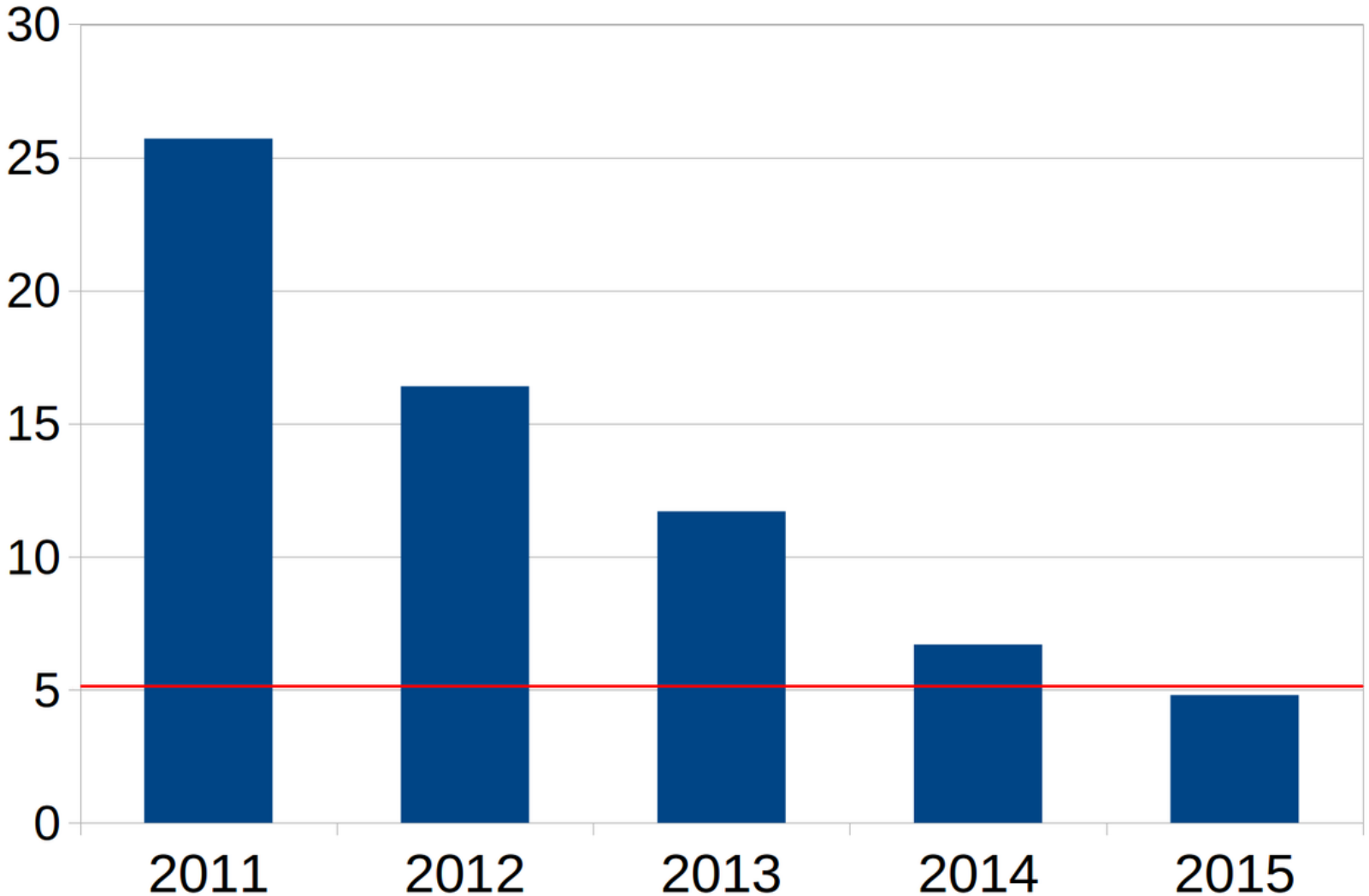
Movie of Brain Segmentation

ImageNet competition

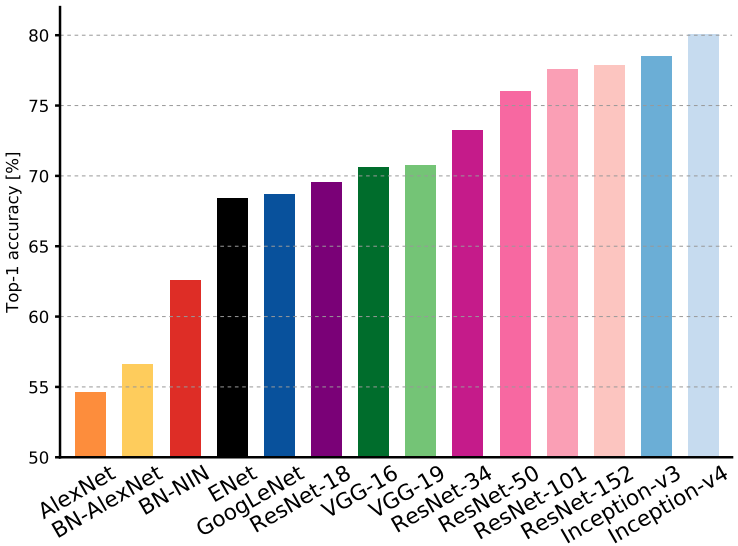


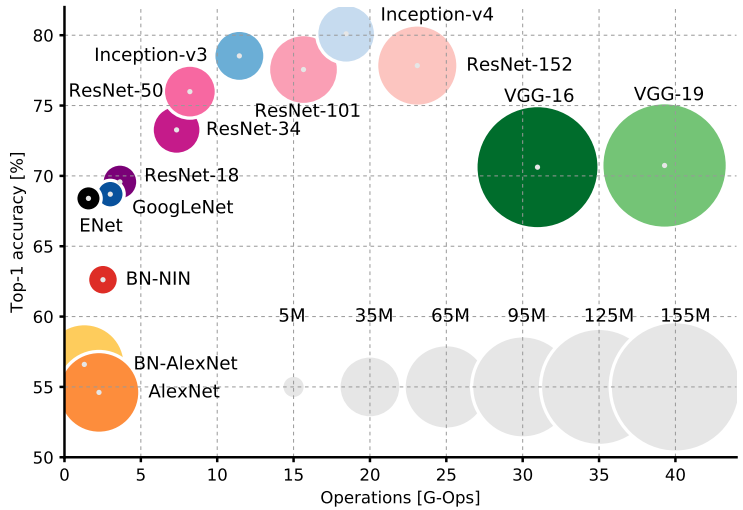
Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. *arXiv:1409.0575*

ImageNet results (top 5)



ImageNet results (top 1)





Canziani, Paszke, Culurciello. An Analysis of Deep Neural Network Models for Practical Applications. *arXiv:1605.07678*

Application example: Traffic sign recognition

Movie of Traffic Sign Recognition

Stallkamp et al. Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Neural Networks 32*, 2012

Medical application: Knee segmentation

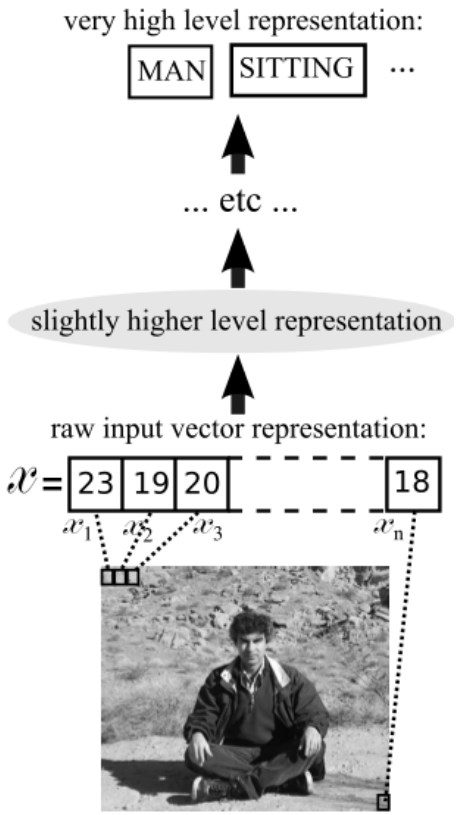
Movie of Knee Segmentation

Perslev, Pai, Igel, Dam. Knee Segmentation by Multiplanar Deep Learning Network – with data from OAI. *International Workshop on Osteoarthritis Imaging*, 2018

Deep learning

- Deep learning refers to ML architectures composed of multiple levels of non-linear transformations.
- Idea: Extracting more and more abstract features from input data, learning more abstract representations.
- Representations can be learned in a supervised and/or unsupervised manner.
- Example: Convolutional neural networks (CNNs) are popular deep learning architectures.

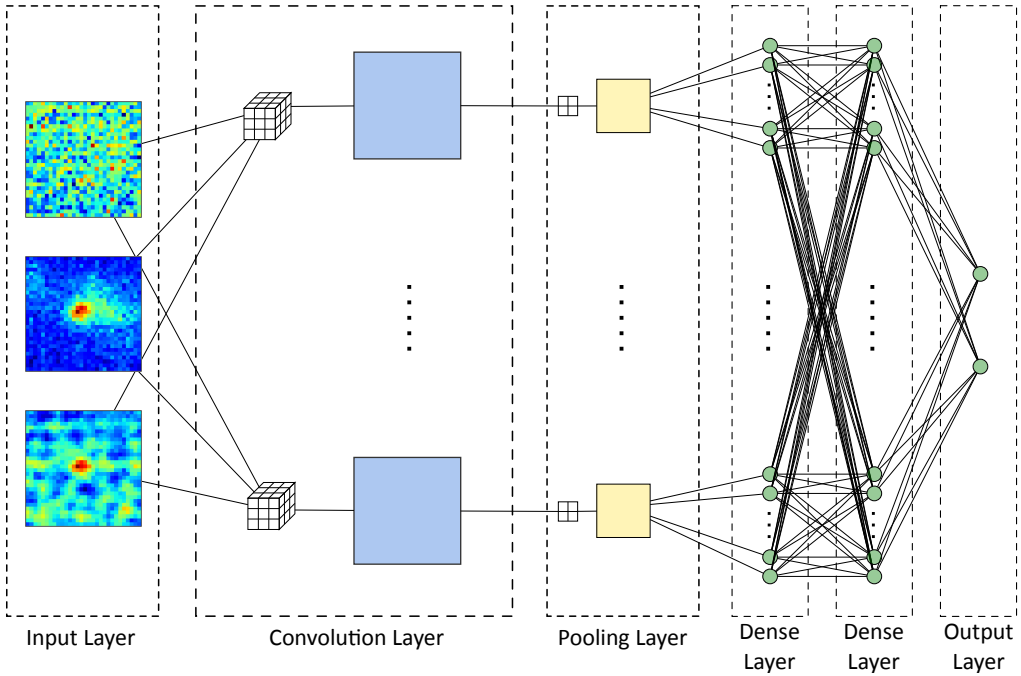
LeCun, Bottou, Bengio, Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998



Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* 2(1): 1–127, 2009

Outline

Convolutional neural network (CNN)



Gieseke et al.: Convolutional Neural Networks for Transient Candidate Vetting in Large-Scale Surveys. *Monthly Notices of the Royal Astronomical Society*, 2017

A canonical CNN consists of

- convolutional layers, each of which producing several *feature maps*; interleaved with
- pooling layers; and
- a standard neural network on top.

Convolution

1D convolution with filter kernel w:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\mathrm{d}a$$

2D convolution of a discrete image I with filter kernel K:

$$\begin{aligned} S(i,j) = (I * K)(i,j) &= \sum_m \sum_n I(m,n)K(i-m, j-n) \\ &= \sum_m \sum_n I(i-m, j-n)K(m,n) \end{aligned}$$

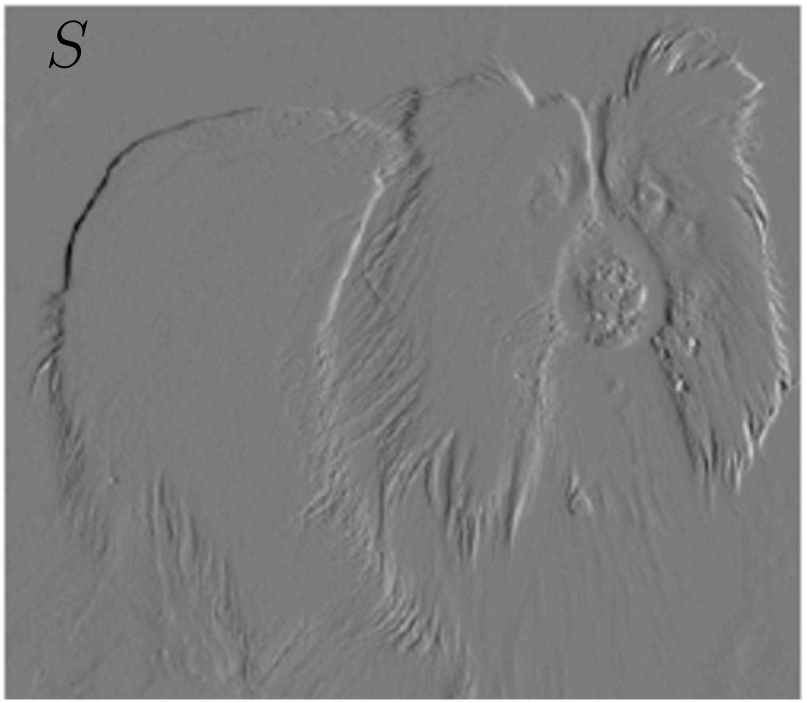
Similar to (and often mistaken with) cross-correlation:

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

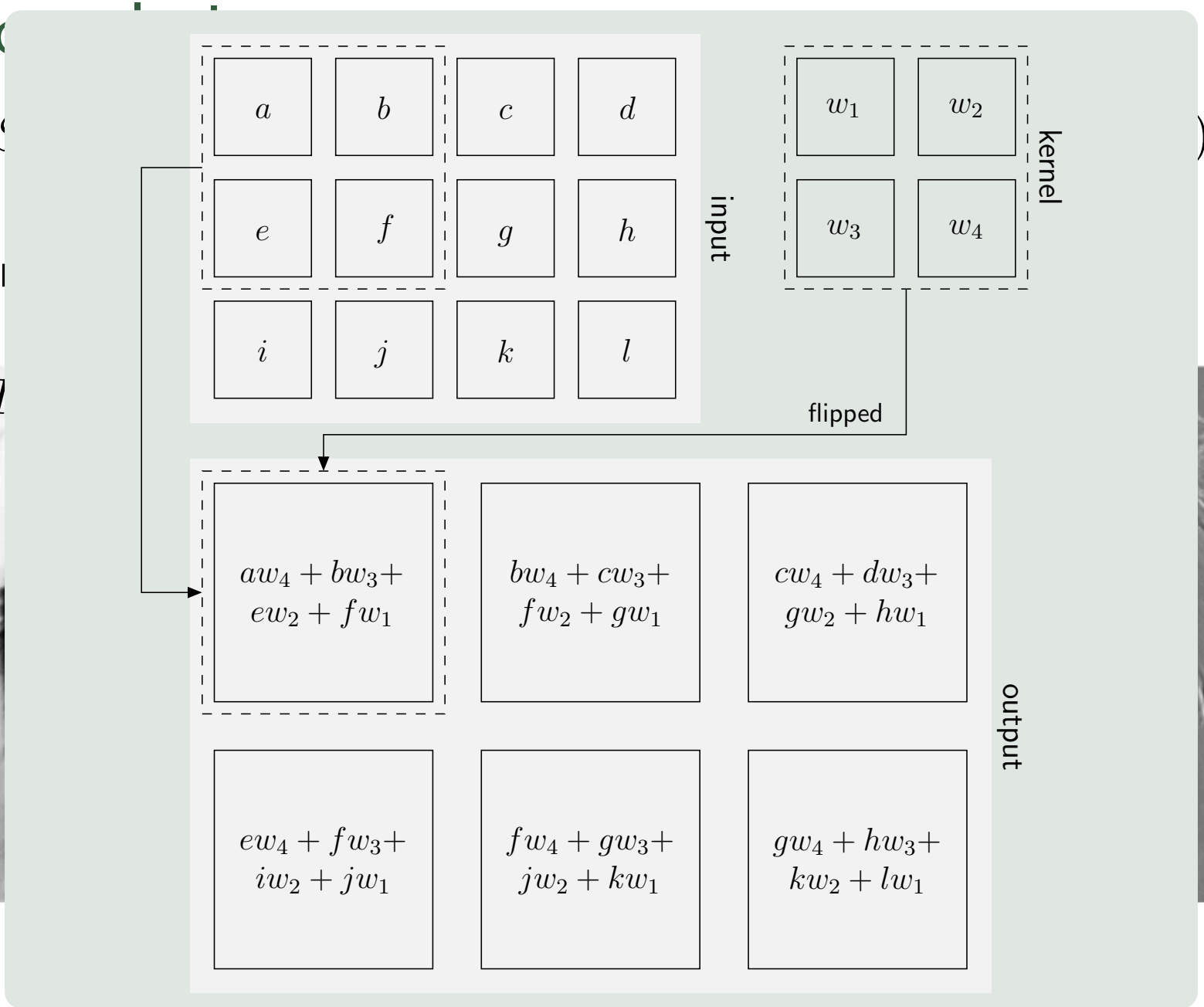
Convolution

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

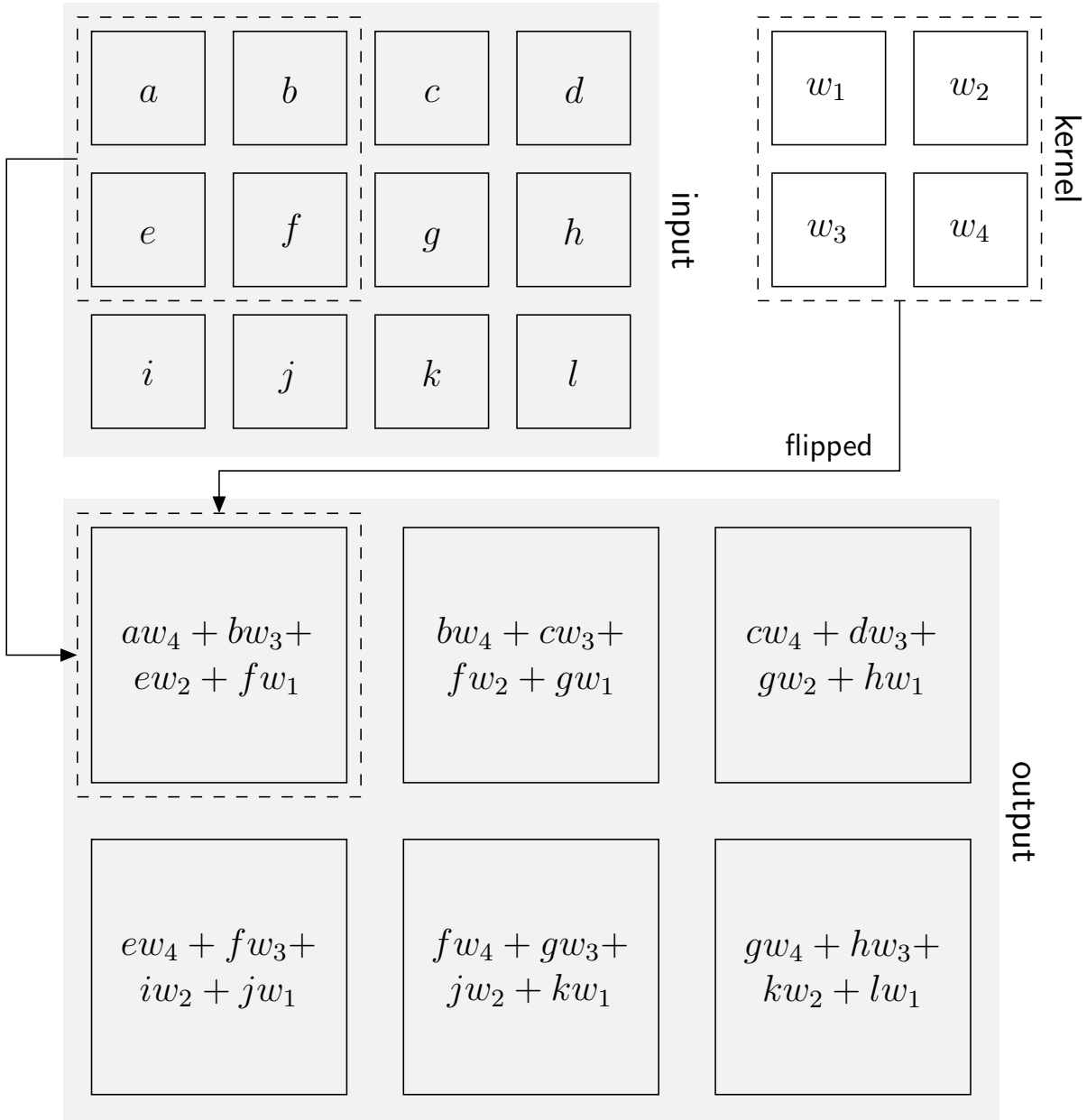
horizontal derivative filter $K = (w_1, w_2) = (1, -1)$:



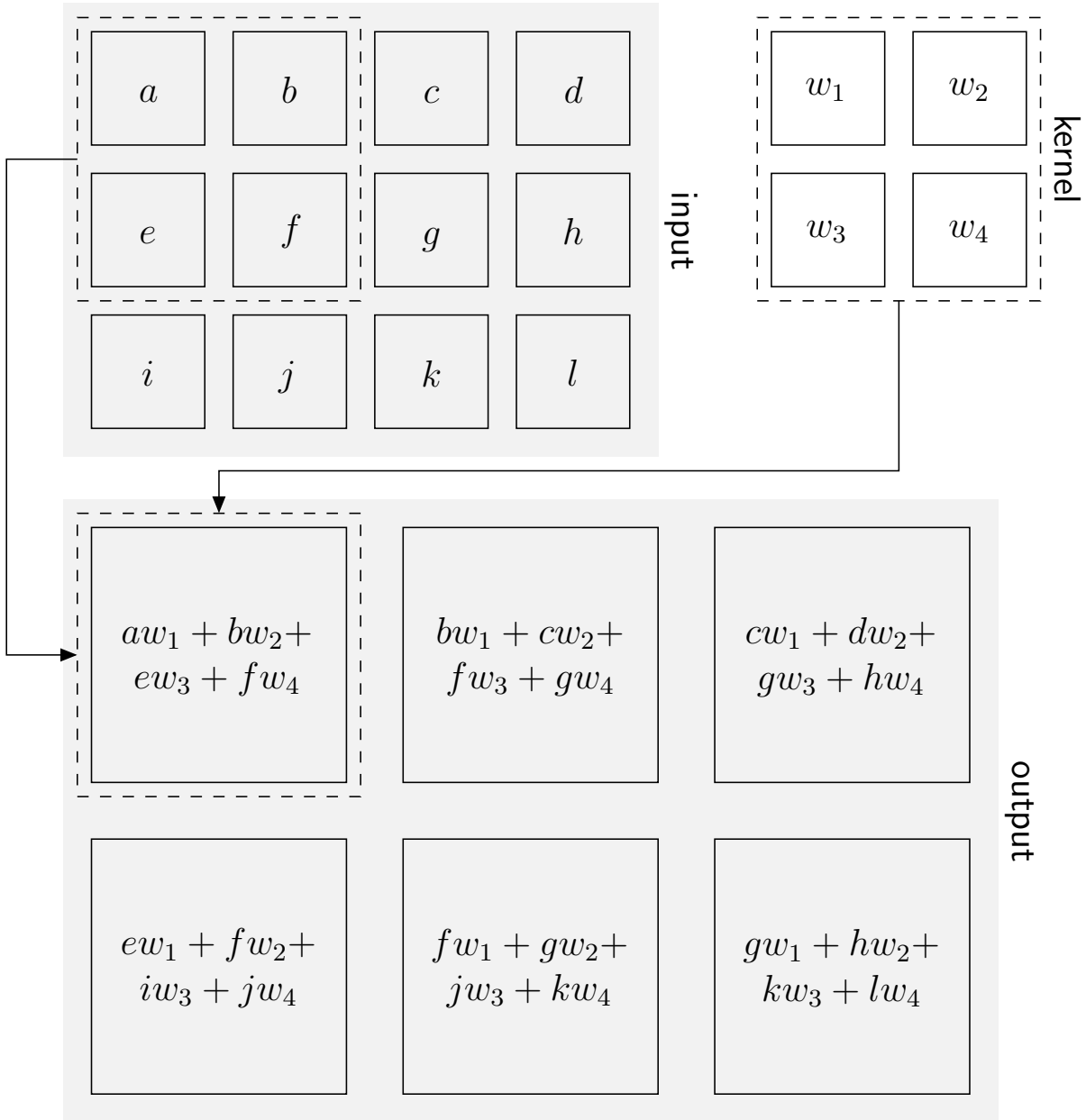
Goodfellow, Bengio, Courville. *Deep Learning*. MIT Press


a
b
c
d
e
f
g
h
i
j
k
l
input
w_1
w_2
w_3
w_4
kernel
flipped
$aw_4 + bw_3 + ew_2 + fw_1$
$bw_4 + cw_3 + fw_2 + gw_1$
$cw_4 + dw_3 + gw_2 + hw_1$
$ew_4 + fw_3 + iw_2 + jw_1$
$fw_4 + gw_3 + jw_2 + kw_1$
$gw_4 + hw_3 + kw_2 + lw_1$
output

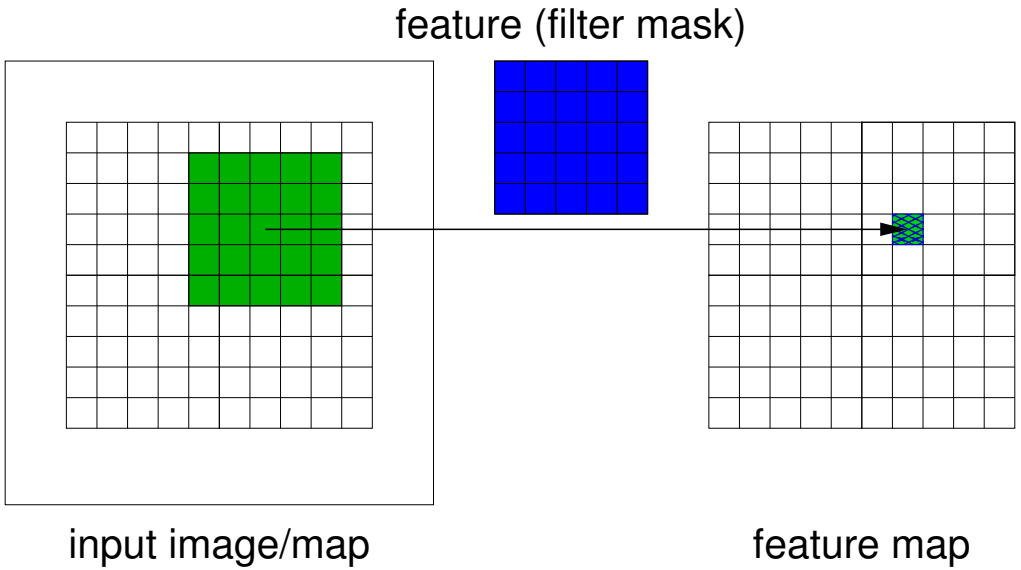
2D convolution



2D cross-correlation

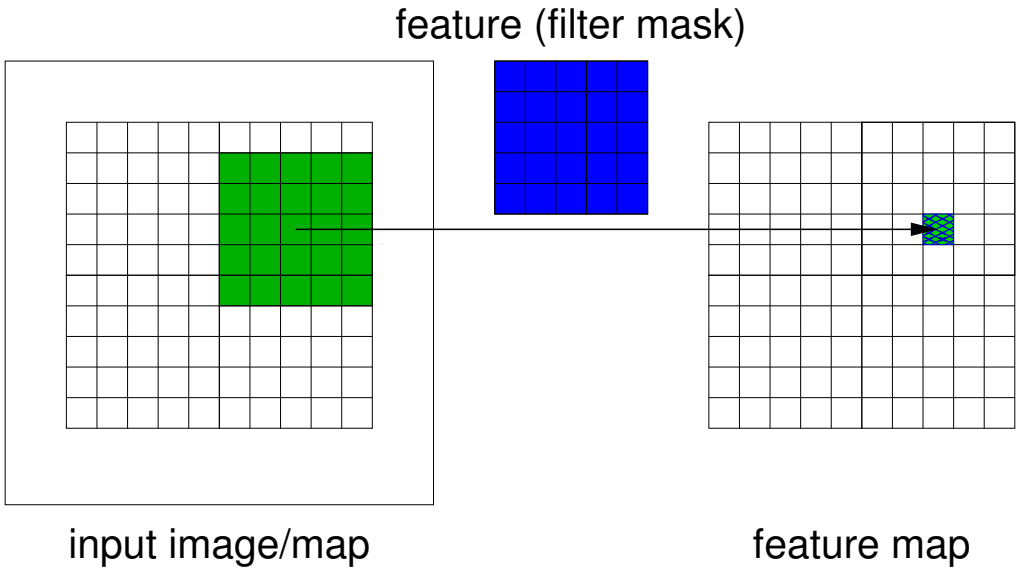


Convolutional layer



- A convolutional layer computes a feature map by convolution of the input with a linear filter.
- The coefficients of each convolution kernel are learned as weights in a neural network.
- A non-linear function is applied to the feature map elements, which can be viewed as neurons with shared weights.

Convolutional layer



- A convolutional layer computes a feature map by convolution of the input with a linear filter.
- The coefficients of each convolution kernel are learned as weights in a neural network.
- A non-linear function is applied to the feature map elements, which can be viewed as neurons with shared weights.

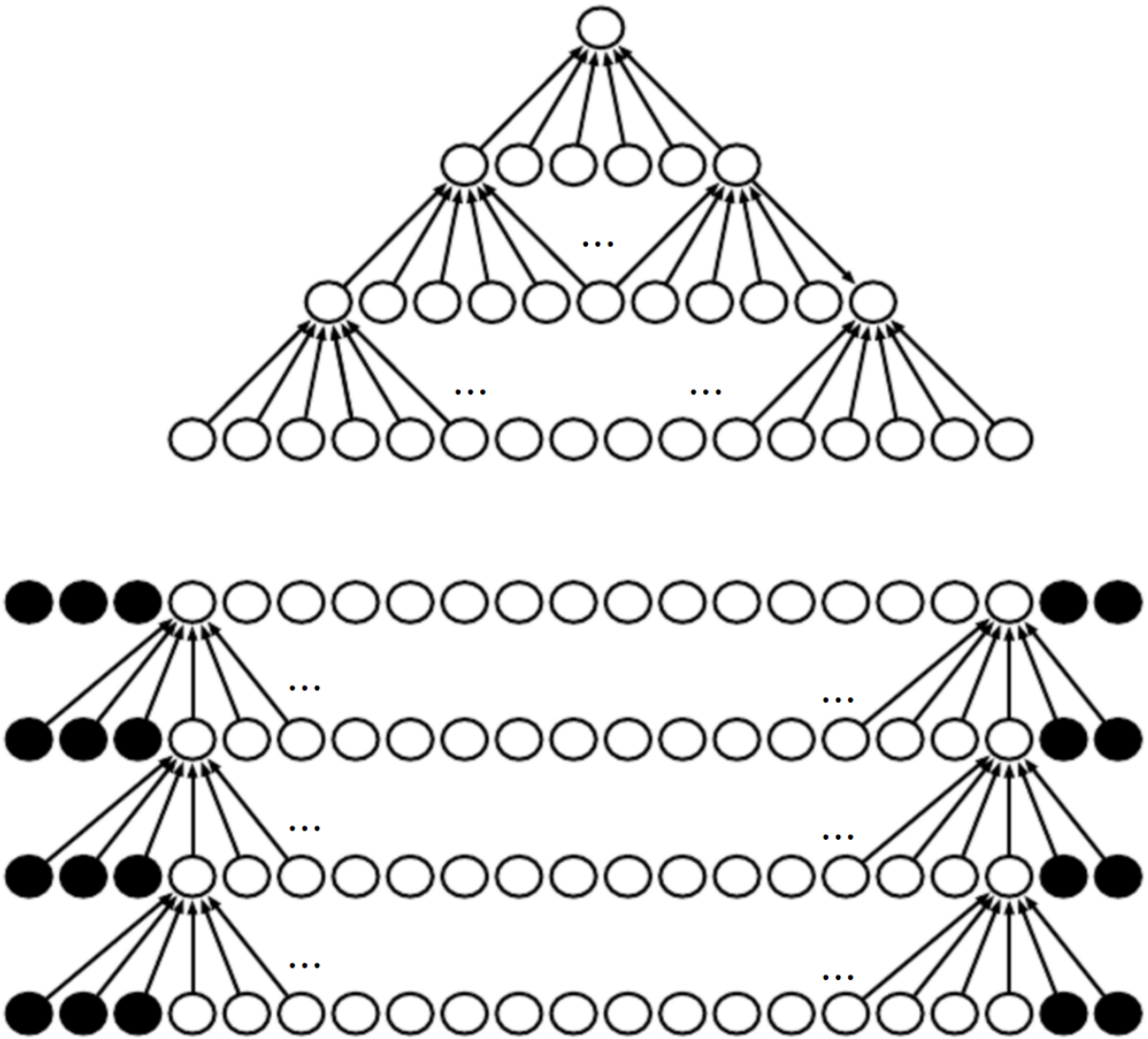
Convolutional layer: 3 ideas

Sparse interactions: Kernel is typically smaller than input, i.e., each neuron is not fully connected; increased memory and statistical efficiency

Parameter sharing: Neurons in a layer have same weights; increased memory and statistical efficiency

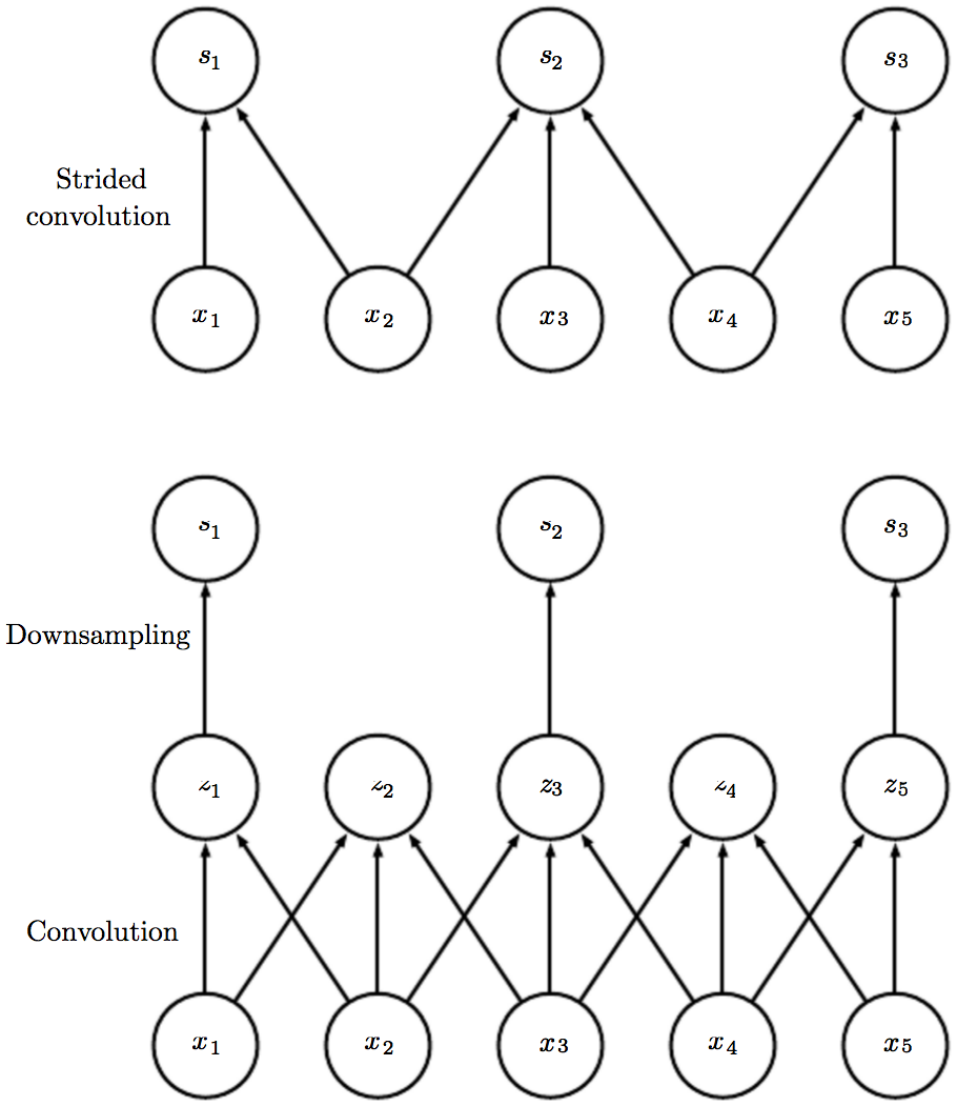
Equivariance: If the input is translated (an object in an image or an event in a time series is shifted), the output is translated in the same way

Zero padding: Dealing with the boundary



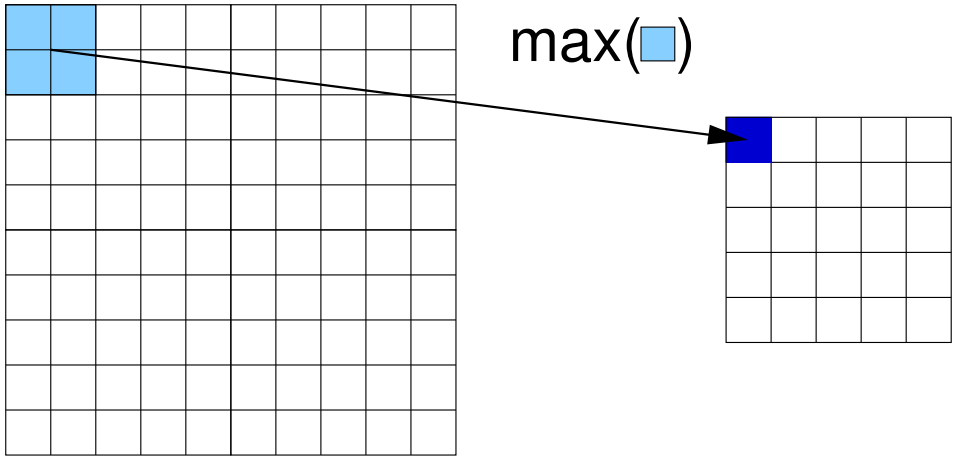
Goodfellow, Bengio, Courville. *Deep Learning*. MIT Press

Convolution with stride



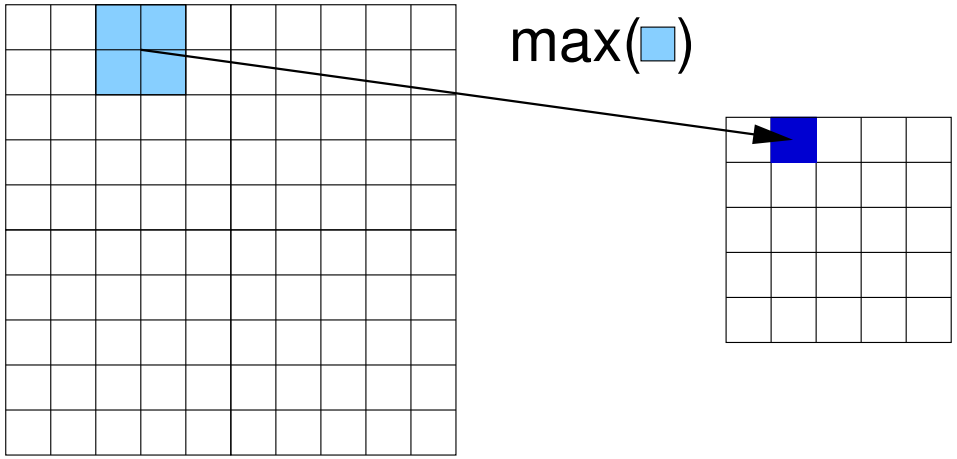
Goodfellow, Bengio, Courville. *Deep Learning*. MIT Press

Pooling layer



- Pooling layers compute the maximum or average over a region of a feature map.
- They are used to reduce the dimensionality, increase the scale, and to support translation invariance.

Pooling layer



- Pooling layers compute the maximum or average over a region of a feature map.
- They are used to reduce the dimensionality, increase the scale, and to support translation invariance.

Outline

Ensemble methods

- Using an ensemble of models can avoid overfitting.
- Typically, these models are trained on different data sets
- Even trained on the same data, CNNs can be quite different because of different random initializations, different mini-batches, different hyperparameters, etc.
- Ensemble can "average out" prediction errors if these are independent of each other.
- CNN ensembles have been very successful.

Ensemble methods example

- k regression models
- For each example, model i makes error ϵ_i normally distributed around 0 with variance $\mathbb{E}[\epsilon_i^2] = v$ and covariance $\mathbb{E}[\epsilon_i \epsilon_j] = c$
- We have:

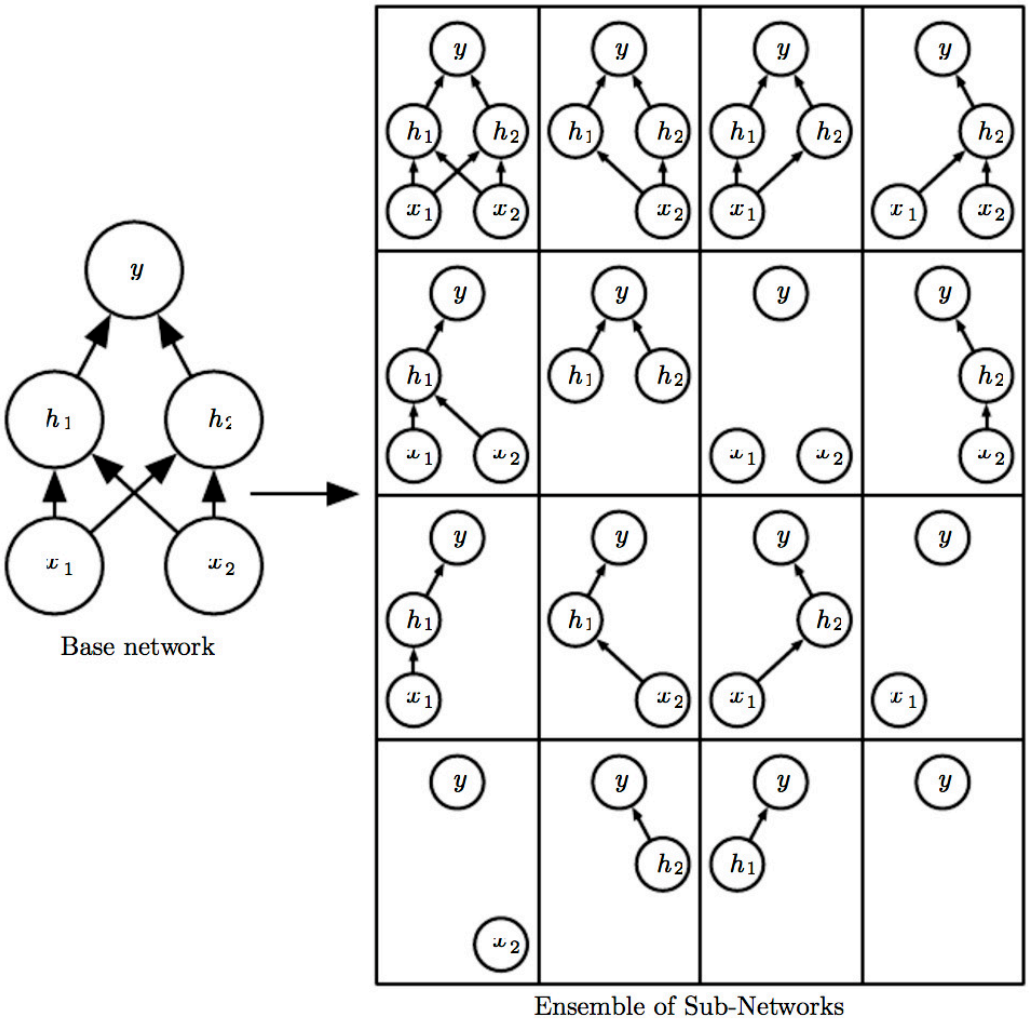
$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i \left(\epsilon_i^2 + \sum_{j \neq i} \epsilon_i \epsilon_j\right)\right]$$
$$= \frac{1}{k}v + \frac{k-1}{k}c$$

Dropout idea

- Dropout is a heuristic to avoid overfitting.
- Idea: View subnetworks of a deep neural network as members of an ensemble
- A binary mask $\boldsymbol{\mu}$ determines whether a node n is used ($\mu_n = 1$) or not ($\mu_n = 0$).

Srivastava et al. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014

Dropout subnetworks



Goodfellow, Bengio, Courville. *Deep Learning*. MIT Press

Dropout outline

- Mini-batch training
- Mask μ resampled after every batch
- Probability of hidden neuron to be include typically 0.5, for an input 0.8
- Learning in each step as normal (ignoring weights associated with nodes having mask value 0)

Dropout ensemble inference

- Output of subnetwork defined by mask $\boldsymbol{\mu}$ given input $\boldsymbol{x}$ is by $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\mu})$.
- For d units that may be dropped, the full ensemble gives:

$$2^{-d} \sum_{\boldsymbol{\mu}} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\mu})$$

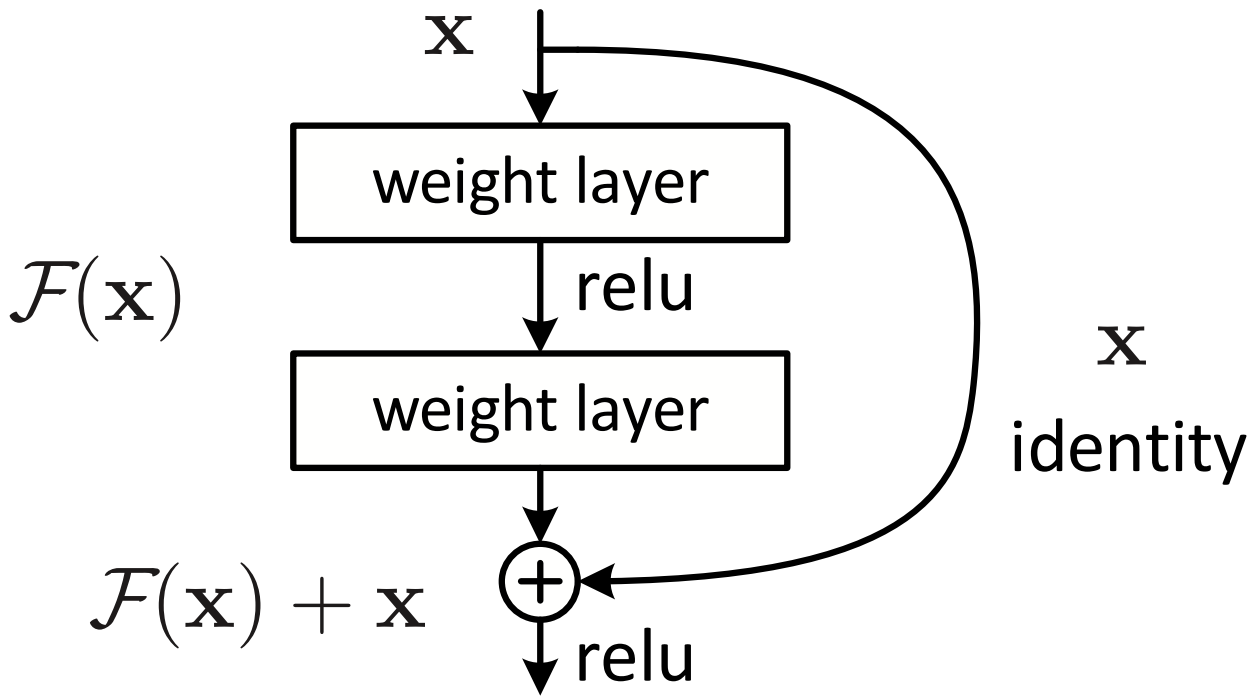
- In practice, the sum above is approximated by sampling some masks $\boldsymbol{\mu}$.

Dropout weight scaling

- Weight scaling inference rule is an efficient alternative to dropout ensemble inference.
- Only the full model is evaluated, however, all weights are multiplied with the probability of the corresponding weight being included in the ensemble (i.e., typically divided by 2).
- This heuristic is exact, for example, if
  - the networks are combined by the geometric (instead of arithmetic) mean, and
  - there are no non-linear hidden units.

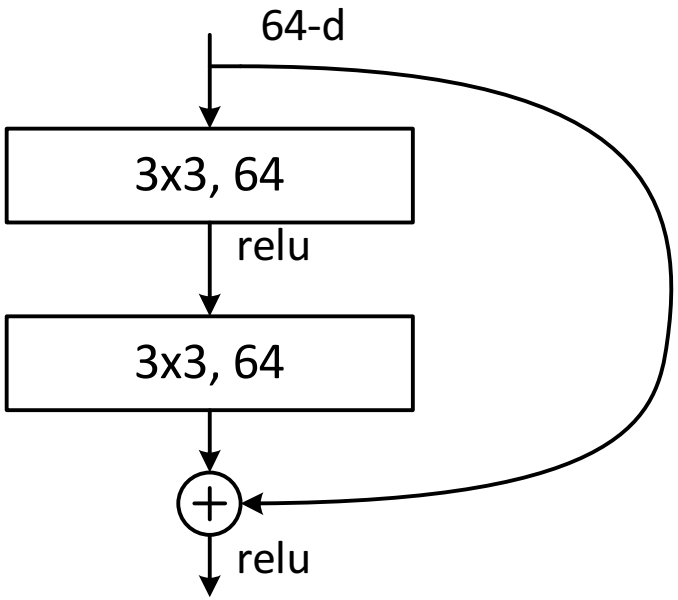
Hinton et al. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*

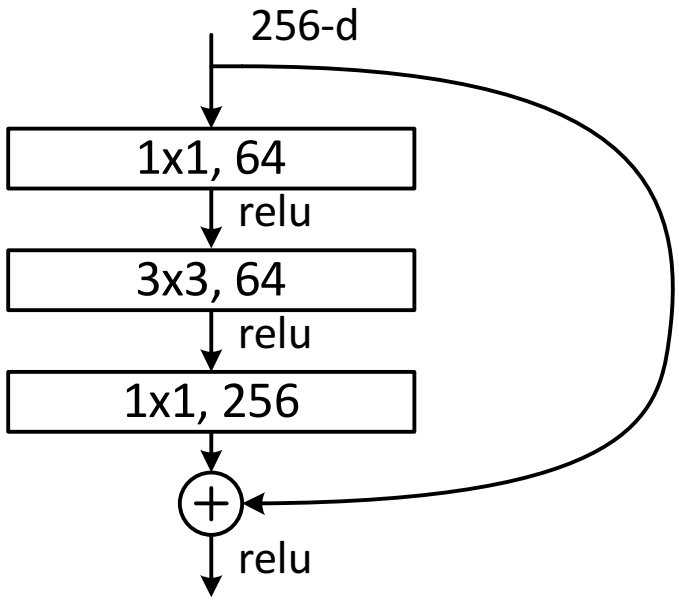
Residual learning / Shortcuts revisited



He et al. Deep Residual Learning for Image Recognition. *CVPR*, IEEE Press, 2016

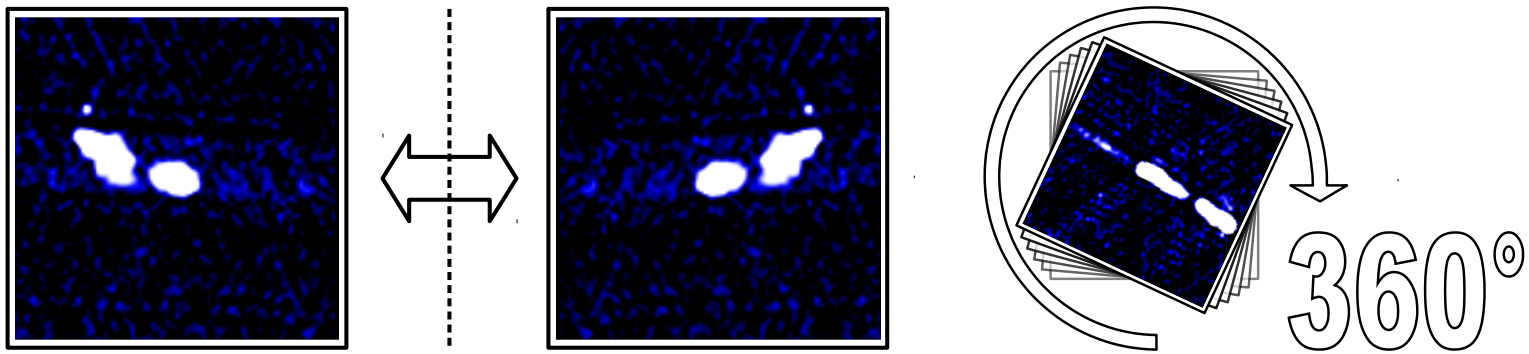
Residual learning / Shortcuts revisited





He et al. Deep Residual Learning for Image Recognition. *CVPR*, IEEE Press, 2016

Invariance & augmentation



Batch normalization

- A batch normalization (BN) layer normalizes inputs at mini-batch level.
- During training, BN can ensure that every mini-batch leads to input with zero mean and unit variance to the next layer.
- During testing, the statistics over all training mini-batches are used.
- Reduces problems due to differences between mini-batches ("internal covariant shift").
- Removes dependence of gradients on scaling of intermediate-outputs within the network, increases numerical stability and can prevent neurons from saturation ($\rightarrow$ vanishing gradients).

Ioffe, Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML*, 2015

Outline

Some thoughts on CNNs

- CNNs are well suited for image, language, and sound processing; they have won several competitions recently.

Some thoughts on CNNs

- CNNs are well suited for image, language, and sound processing; they have won several competitions recently. The best performance is usually achieved when the CNN is combined with careful preprocessing.

Some thoughts on CNNs

- CNNs are well suited for image, language, and sound processing; they have won several competitions recently. The best performance is usually achieved when the CNN is combined with careful preprocessing.
- CNNs exploit "spatial" / "temporal" structure in the input; in particular, they incorporate translation invariance.

Some thoughts on CNNs

- CNNs are well suited for image, language, and sound processing; they have won several competitions recently. The best performance is usually achieved when the CNN is combined with careful preprocessing.
- CNNs exploit "spatial" / "temporal" structure in the input; in particular, they incorporate translation invariance.
- Unsupervised and supervised learning can be combined.

Some thoughts on CNNs

- CNNs are well suited for image, language, and sound processing; they have won several competitions recently. The best performance is usually achieved when the CNN is combined with careful preprocessing.
- CNNs exploit "spatial" / "temporal" structure in the input; in particular, they incorporate translation invariance.
- Unsupervised and supervised learning can be combined.
- Training CNNs typically scales linearly in training set size.
- CNNs profit from massively parallel computing (e.g., GPUs)

Some more thoughts

- ⊕/⊖ Deep networks are highly non-linear models.
- ⊖ Deep networks typically have many hyperparameters.
- ⊖ Tuning the architecture of a deep learning system can be very difficult.

Some more thoughts

- ⊕/⊖ Deep networks are highly non-linear models.
- ⊖ Deep networks typically have many hyperparameters.
- ⊖ Tuning the architecture of a deep learning system can be very difficult.
- ⊖ Deep learning is badly understood theoretically.

Some more thoughts

- ⊕/⊖ Deep networks are highly non-linear models.
- ⊖ Deep networks typically have many hyperparameters.
- ⊖ Tuning the architecture of a deep learning system can be very difficult.
- ⊖ Deep learning is badly understood theoretically.

The revival of deep neural networks is partly due to the availability of fast (parallel) hardware and larger training data sets (which makes overfitting less of a problem).

Some deep learning resources

Goodfellow, Bengio, Courville. *Deep Learning*. MIT Press, to appear, `http://www.deeplearningbook.org/`

Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks* 61:85–117, 2015