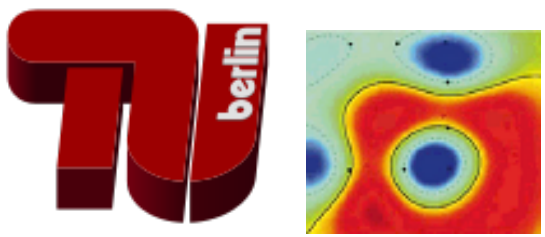
Convolutional Neural Networks

Deep Neural Networks



Kristof T. Schütt – TU Berlin – SoSe 2018

Communicated by Dana Ballard

Backpropagation Applied to Handwritten Zip Code Recognition

Y. LeCun
B. Boser
J. S. Denker
D. Henderson
R. E. Howard
W. Hubbard
L. D. Jackel
AT&T Bell Laboratories Holmdel, NJ 07733 USA

The ability of learning networks to generalize can be greatly enhanced by providing constraints from the task domain. This paper demonstrates how such constraints can be integrated into a backpropagation network through the architecture of the network. This approach has been successfully applied to the recognition of handwritten zip code digits provided by the U.S. Postal Service. A single network learns the entire recognition operation, going from the normalized image of the character to the final classification.

Learning with Sequential Data

Consider two gene sequences:

$$x = (A, T, C, A, G, \underbrace{A, C, A}_{s}, A, T, A)$$

$$x' = (C, \underbrace{G, T, A}_{s'}, C, A, A)$$

Example of a string kernel that compares two strings of variable length:

$$k_{\text{struct}}(x, x') = \sum_{(s,s') \in \mathcal{S}_d(x) \times \mathcal{S}_d(x')} k(s, s')$$

where $S_d(x)$ is the list of all substrings of x of length d.

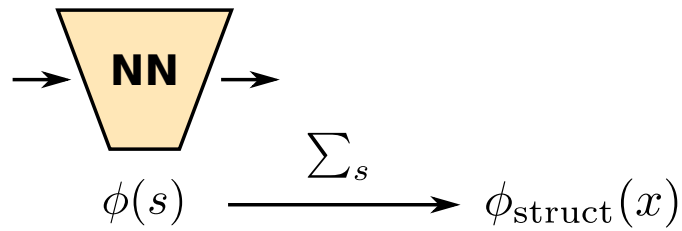
Learning with Sequential Data

- **Observation:** $k_{\text{struct}}(x, x')$ has an associated feature map:

$$\phi_{\text{struct}}(x) = \sum_{s \in \mathcal{S}_d(x)} \phi(s)$$

  where $\phi(s)$ is the feature map associated to $k(s, s')$.
- **Question:** Can a neural network represent this feature map?
- **Answer:** Apply a sliding window

$$x = (A, T, C, A, G, \underbrace{A, C, A}_{s}, A, T, A)$$



The sliding window principle is a basic idea of the convolutional neural network.

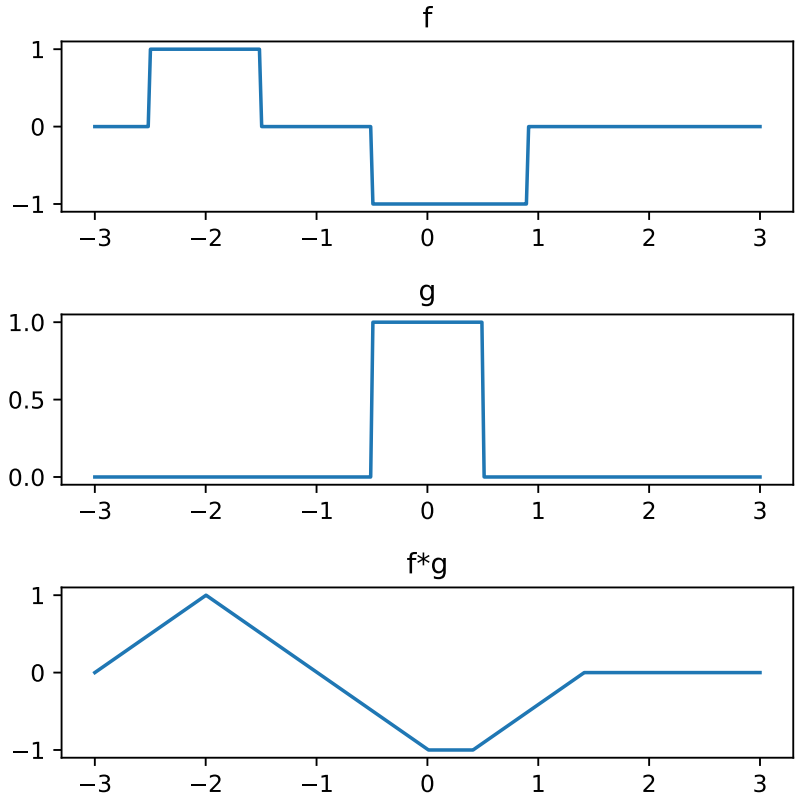
Convolution

Convolution:

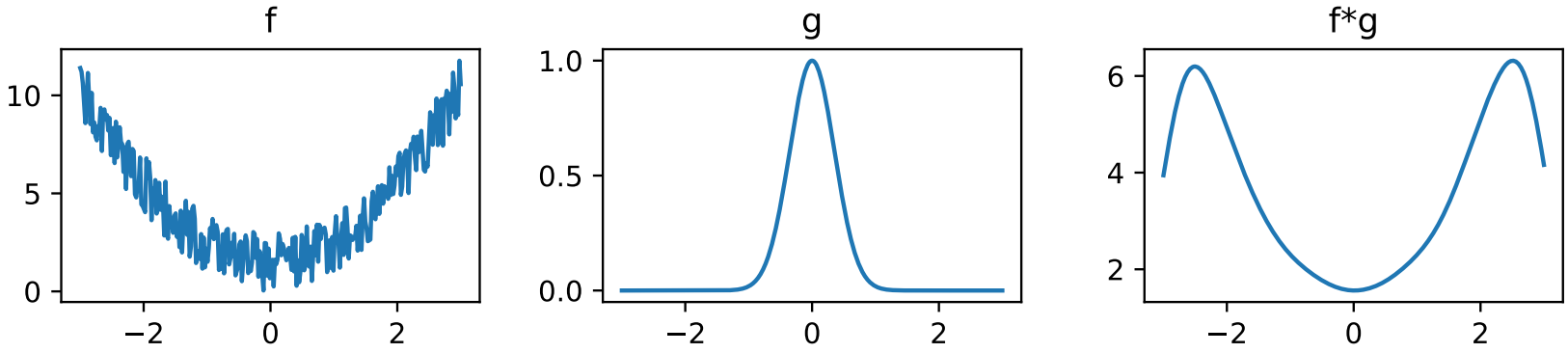
$$(f * g)(t) = \int f(s)g(t-s)ds$$

Discrete convolution:

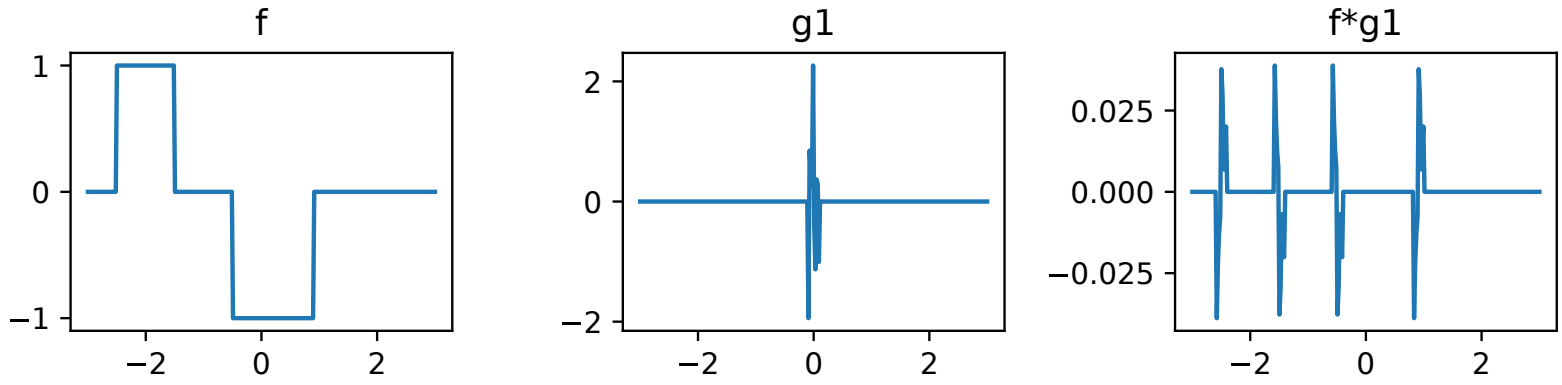
$$[f * g]_t = \sum_{s=-\infty}^{\infty} f_s \cdot g_{t-s}$$



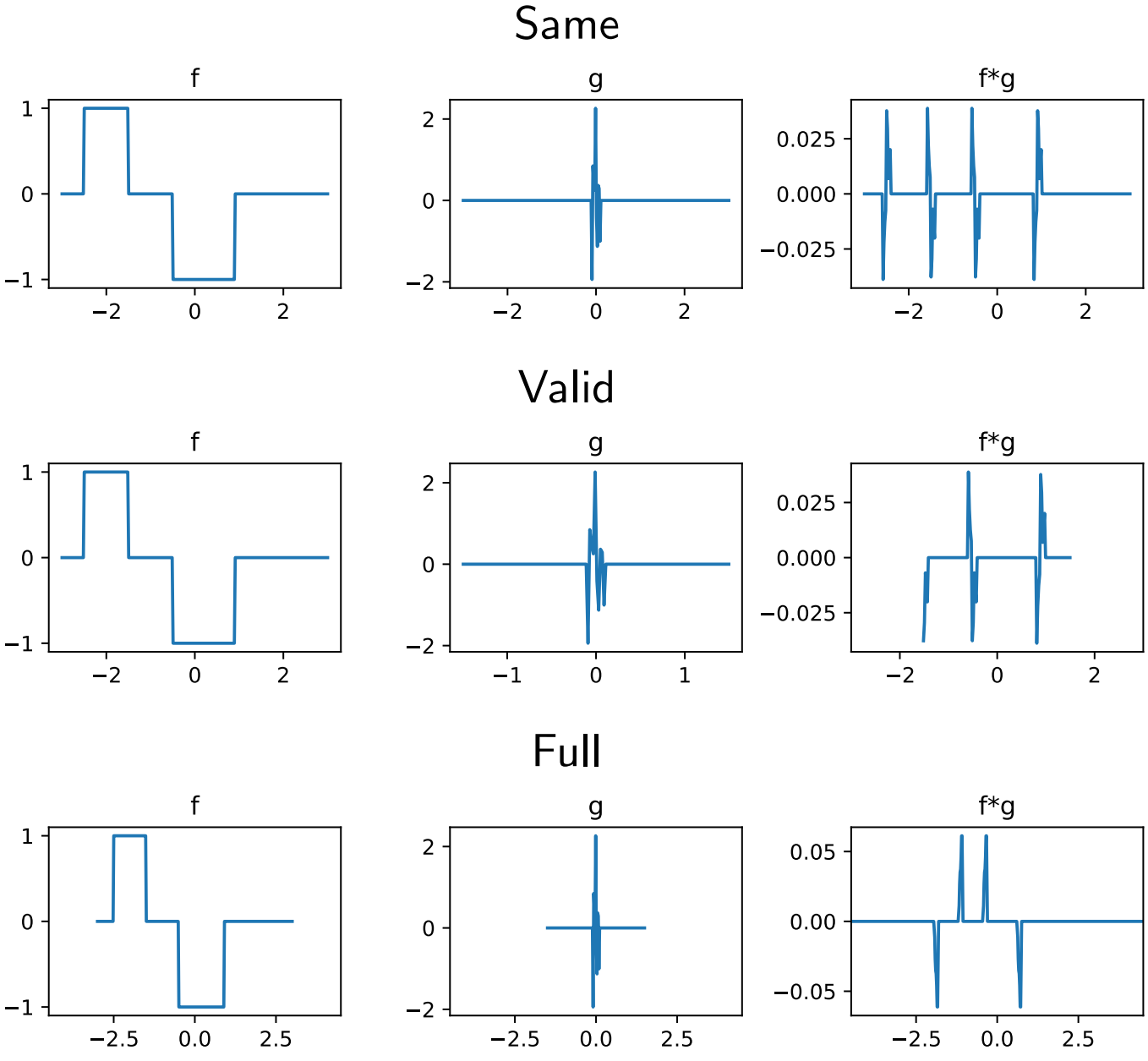
Convolution: Smoothing



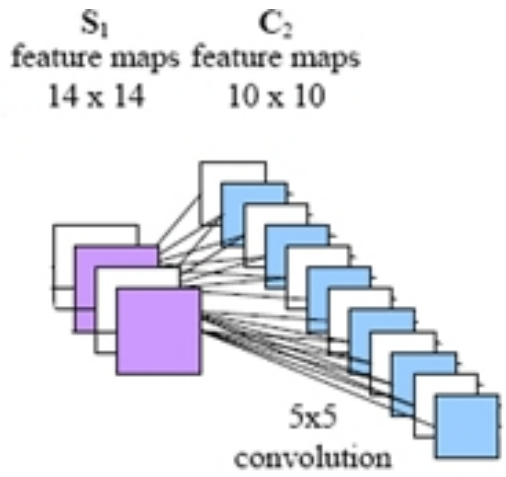
Convolution: Differentiation



Convolution: Border mode



Convolutional Layer



Let subscripts i, j denote the indices of feature maps, and superscripts a, b, α, β denote the pixel locations. The relation between input x and output y of a convolutional layer is given by:

$$\forall_{j=1}^{12}, \forall_{a,b=1}^{10} : \; y_j^{ab} = \sum_{i=1}^{4} [w_{ij} * x_i]^{ab} = \sum_{i=1}^{4} \sum_{\alpha=1}^{5} \sum_{\beta=1}^{5} w_{ij}^{\alpha\beta} \cdot x_i^{a+\alpha-1, b+\beta-1}$$

Image taken and adapted from the website of *Parallel Architecture Research Eindhoven*

Error Backpropagation in a Convolution

Forward pass:

$$z_t = [w * x]_t = \sum_{s=-\infty}^{\infty} w_s \cdot x_{t+s}$$

Backward pass:

$$\begin{aligned}
\frac{\partial E}{\partial x_u} &= \sum_{t=-\infty}^{\infty} \frac{\partial E}{\partial z_t} \cdot \frac{\partial z_t}{\partial x_u} \\
&= \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} \frac{\partial E}{\partial z_t} \cdot w_s \cdot 1_{\{t+s=u\}} \\
&= \sum_{t=-\infty}^{\infty} \frac{\partial E}{\partial z_t} \cdot w_{u-t} \\
&= \left[\frac{\partial E}{\partial z} * w\right]_u
\end{aligned}$$

Parameter gradient:

$$\begin{aligned}
\frac{\partial E}{\partial w_u} &= \sum_{t=-\infty}^{\infty} \frac{\partial E}{\partial z_t} \cdot \frac{\partial z_t}{\partial w_u} \\
&= \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} \frac{\partial E}{\partial z_t} \cdot x_{t+s} \cdot 1_{\{s=u\}} \\
&= \sum_{t=-\infty}^{\infty} \frac{\partial E}{\partial z_t} \cdot x_{u+t} \\
&= \left[\frac{\partial E}{\partial z} \star x\right]_u
\end{aligned}$$

Two Views of a Convolutional Layer (1D)

View 1: Sum of convolutions

$$\forall j, t: \; z_j(t) = \sum_{i=1}^{n} \sum_{s=-\infty}^{\infty} w_{ij}^{s} \cdot x_i^{t+s}$$

View 2: Swipe of weighted sums

$$\forall j, t: \; z_j(t) = \sum_{s=-\infty}^{\infty} \sum_{i=1}^{n} w_{ij}^{s} \cdot x_i^{t+s}$$

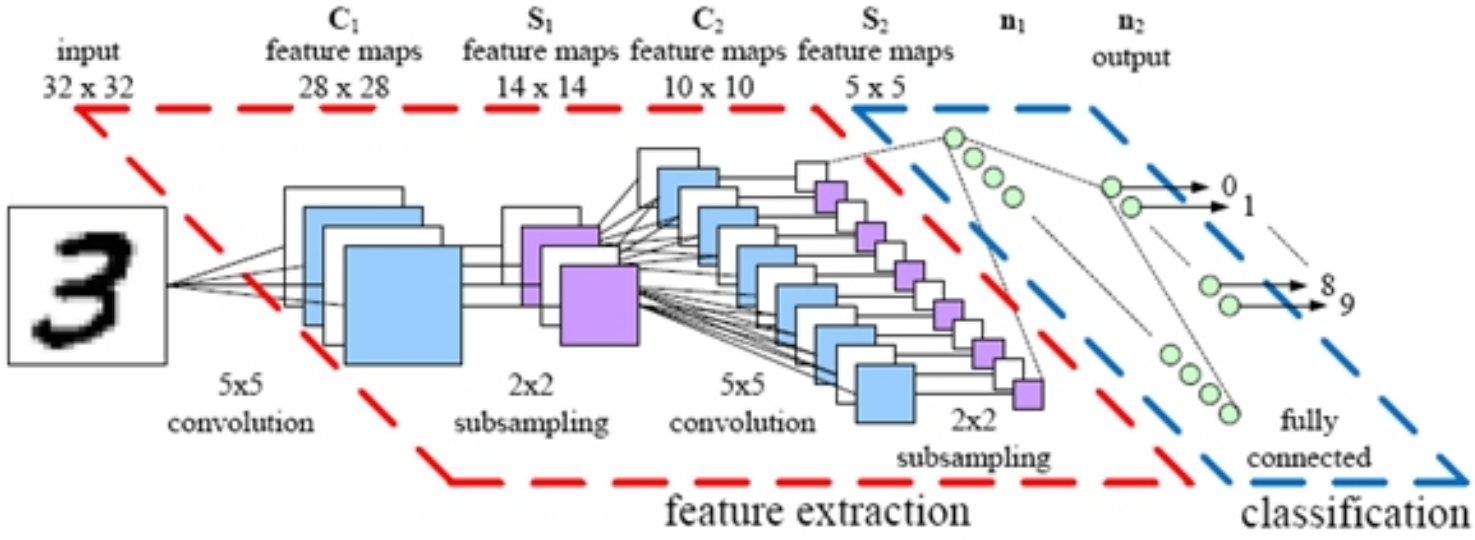
Convolution Layers as $y = W^\top x$

- $x = 2$ feature maps of 6 time steps
- $y = 3$ feature maps of 4 time steps
- $v = 2 \times 3$ convolution kernels of width 3

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

- Blocks = pairs of feature maps
- Band matrices = convolutions
- W has 144 entries, but only 18 effective parameters v_{ij}^t.

Convolutional Neural Network



(Image taken from the website of *Parallel Architecture Research Eindhoven*)

A convolutional neural network alternates several stages of

- Convolutions (convolve the image with different filters)
- Pooling (pool features at neighboring locations)

Why CNNs?

1. Translational invariance: Meaning of an input is unchanged with respect to a particular transformation of it.

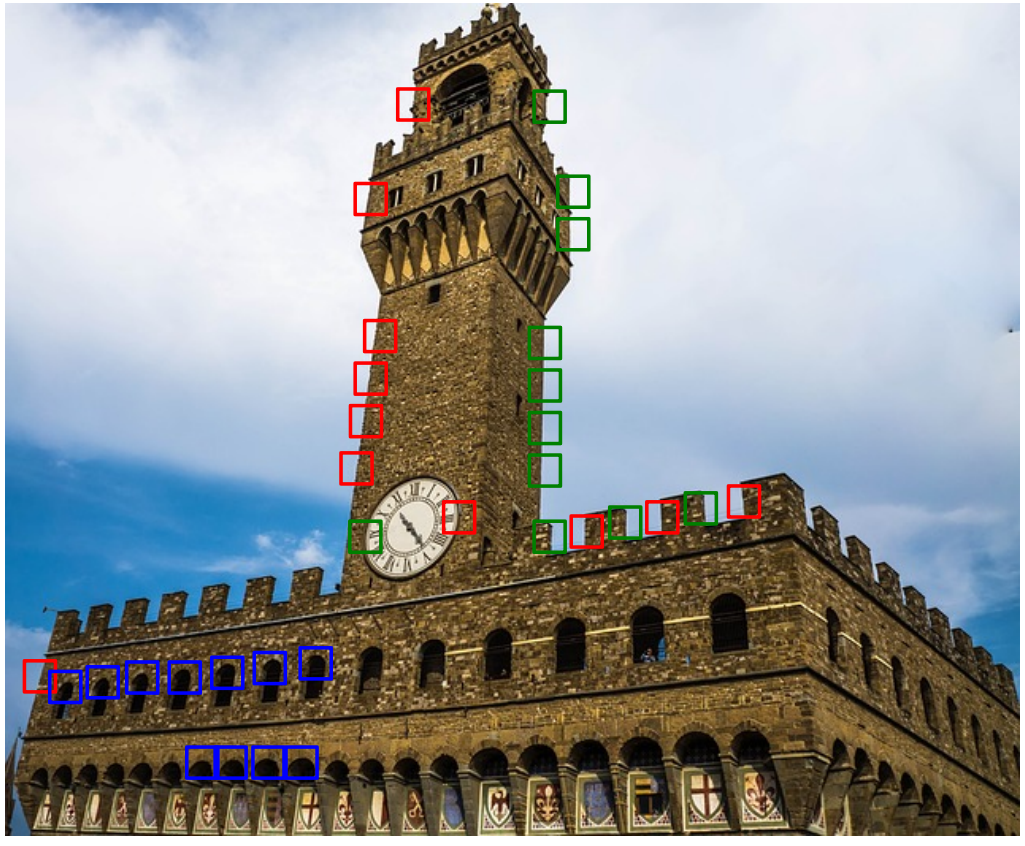
lighthouse

lighthouse

lighthouse

Why CNNs?

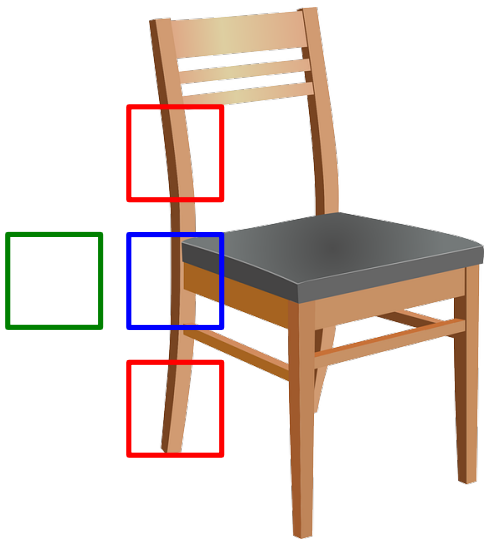
2. Homogeneity: Same feature detectors are needed for different subsets of input dimensions (e.g. image patches).



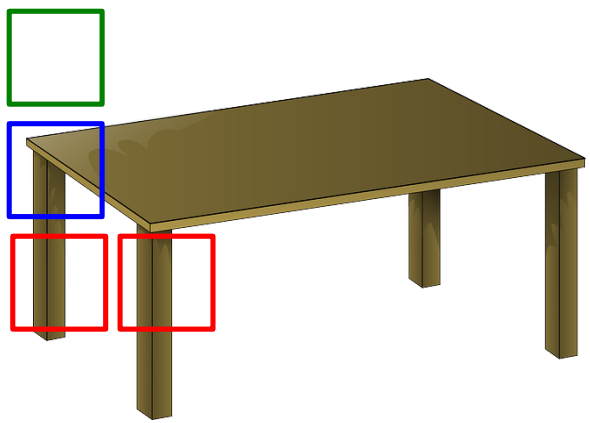
Why CNNs?

3. Depth: Meaning of an input is determined by how particular subparts of the input are interrelated, and not by the presence or absence of the subparts themselves.

chair



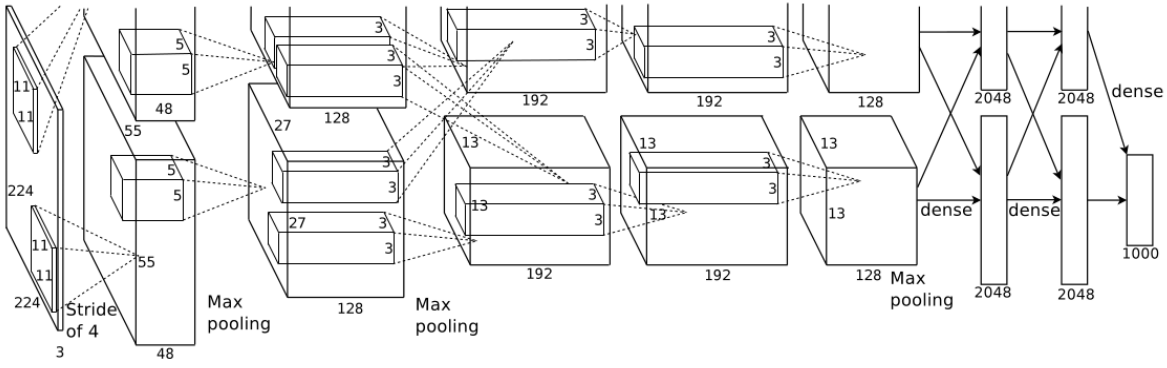
table



Application of CNNs to Image Classification

ImageNet (12 millions of 224×224 images, 1000 classes):

CNN Architecture by Krizhevsky et al. 2012 (SuperVision)

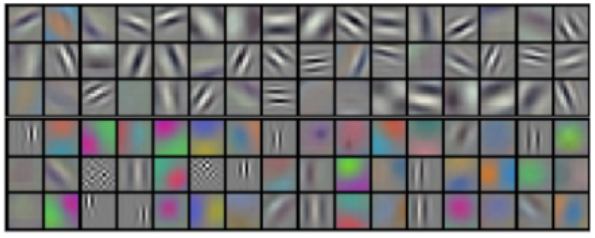


Images taken from the paper Krizhevsky et al. 2012 ImageNet Classification with Deep Convolutional Neural Networks

Application of CNNs to Image Classification

Predictions of the SuperVision Network (Krizhevsky et al. 2012)

SuperVision conv. kernels

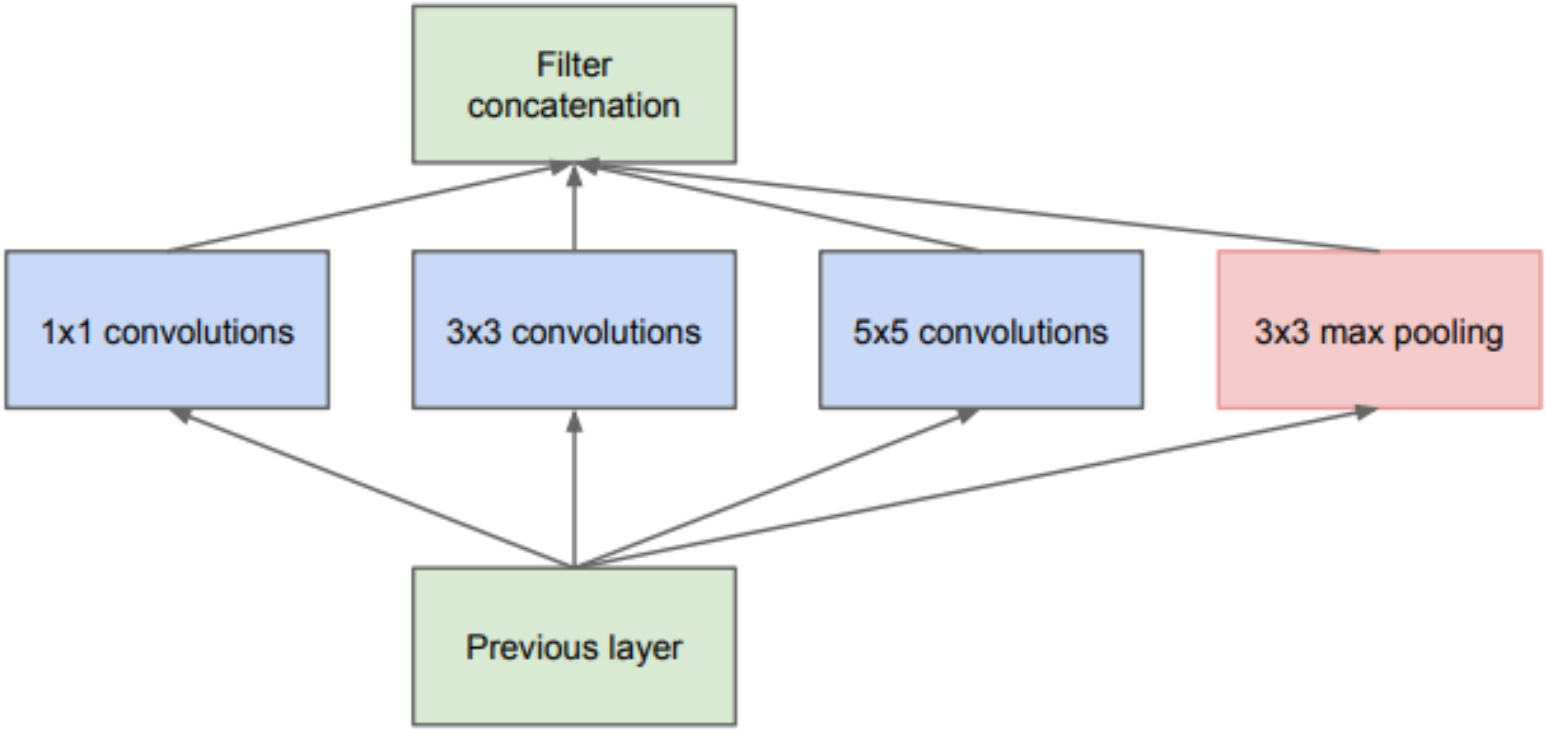


ILSVRC2012 Results

Team name	Error (5 guesses)
SuperVision	0.15315
SuperVision	0.16422
ISI	0.26172
ISI	0.26602
ISI	0.26646
ISI	0.26952
OXFORD_VGG	0.26979
XRCE/INRIA	0.27058
OXFORD_VGG	0.27079
OXFORD_VGG	0.27302
...	...

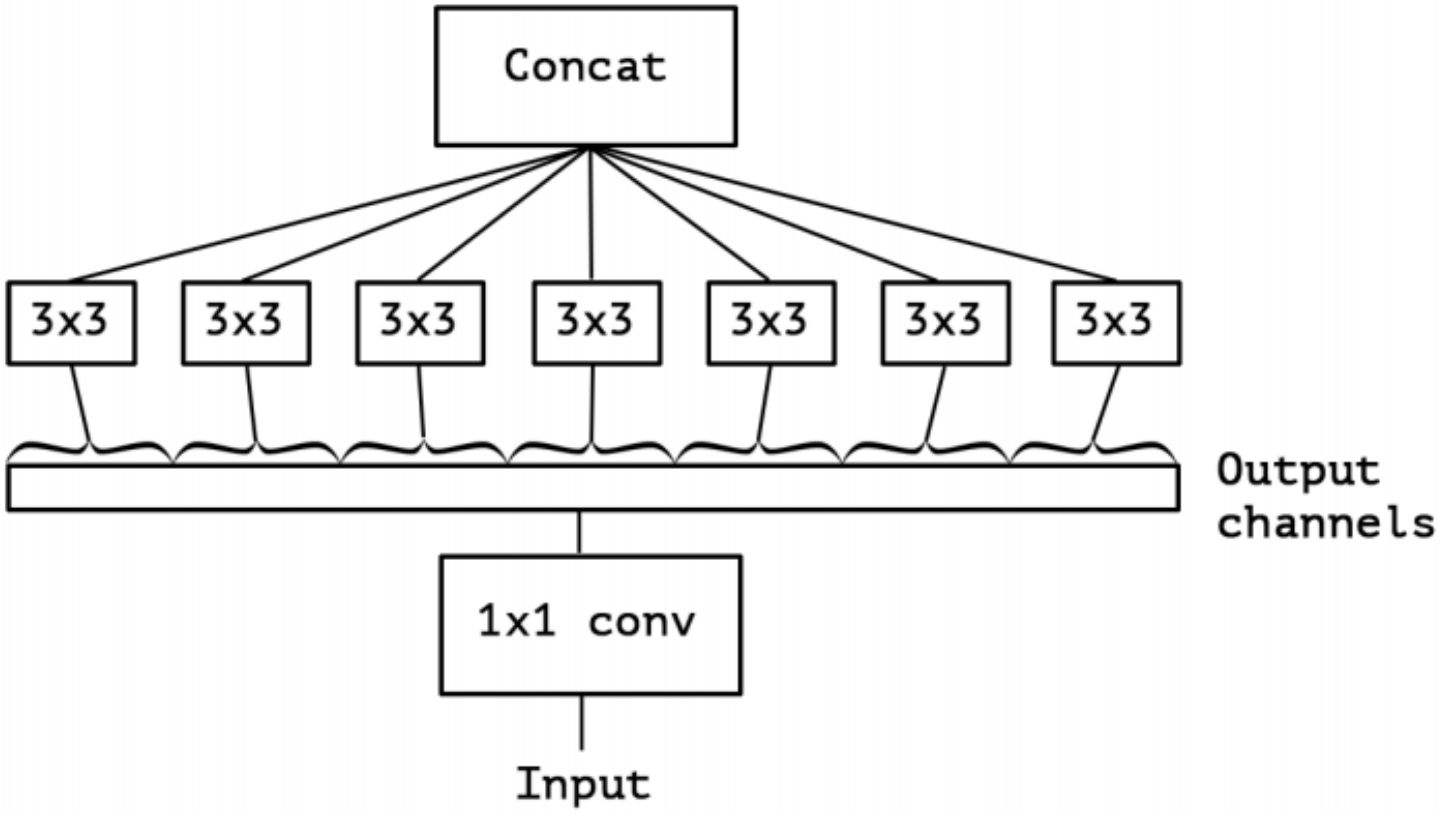
Images taken from the paper Krizhevsky et al. 2012 ImageNet Classification with Deep Convolutional Neural Networks. List of results taken from the website Large Scale Visual Recognition Challenge 2012

Advanced architectures: Inception



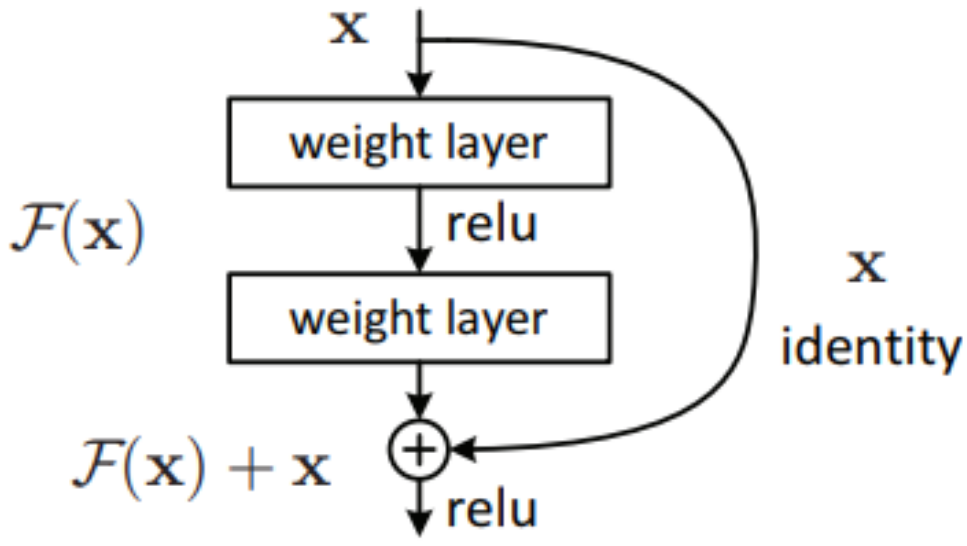
Szegedy, Christian, et al. "Going deeper with convolutions." CVPR, 2015.

Advanced architectures: Xception



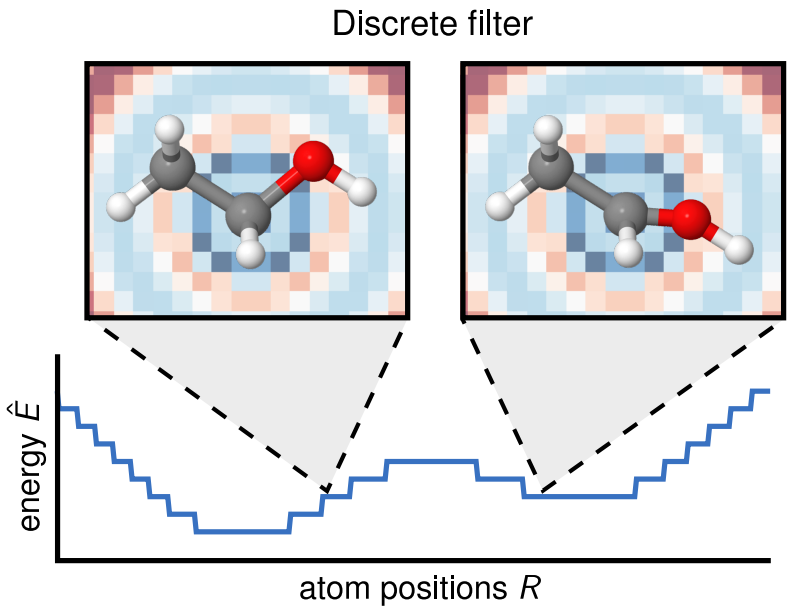
Chollet, François. "Xception: Deep learning with depthwise separable convolutions." arXiv preprint. 2016.

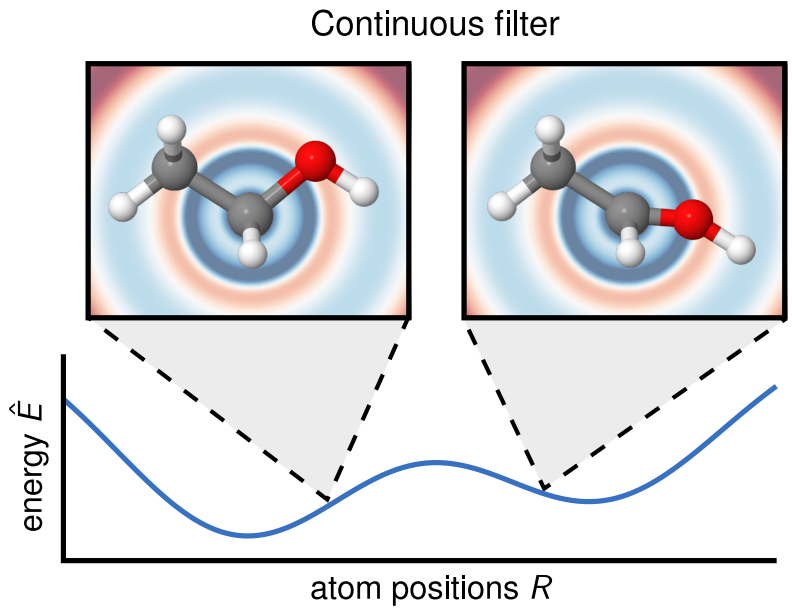
Advanced architectures: ResNet



He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Advanced architectures: Continuous-filter convolutions



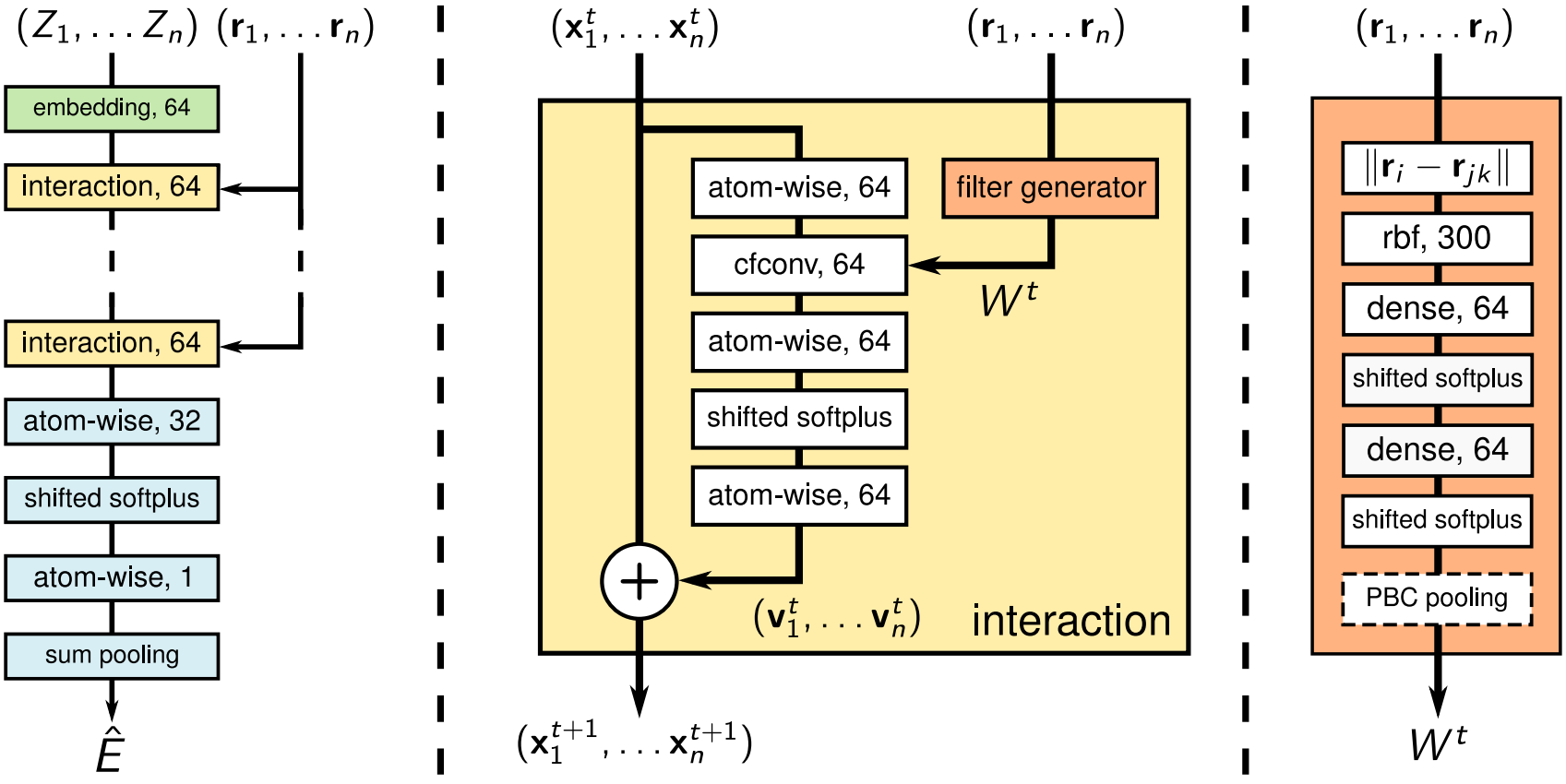


$$(\mathbf{x} * W)(\mathbf{r}_i) = \sum_{j=1}^{N_{\text{atom}}} \mathbf{x}_j^{(t)} \circ \underbrace{W^{(t)}_{[\mathbf{r}_i - \mathbf{r}_j]}}_{\text{parameter tensor}}$$

$$(\mathbf{x} * W)(\mathbf{r}_i) = \sum_{j=1}^{N_{\text{atom}}} \mathbf{x}_j^{(t)} \circ \underbrace{W^{(t)}(\mathbf{r}_i - \mathbf{r}_j)}_{\text{neural network}}$$

K.T. Schütt, P.J. Kindermans, H.E. Sauceda, S. Chmiela, A. Tkatchenko, K.-R. Müller (2017). SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. NIPS 30.

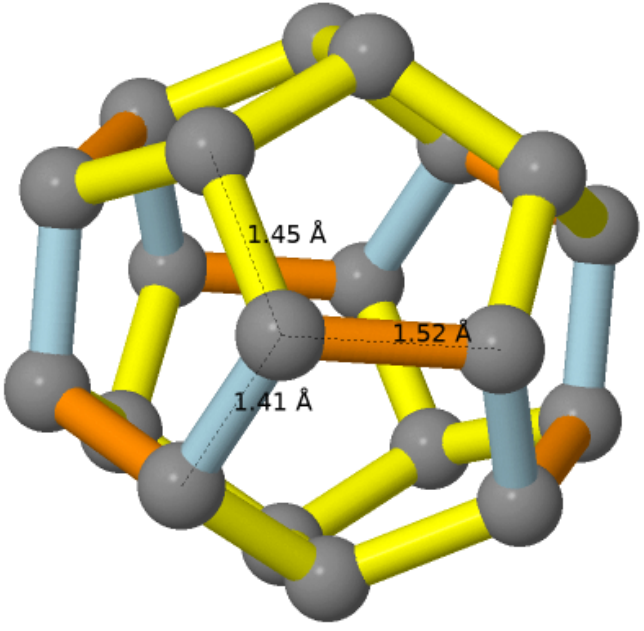
SchNet - a continuous-filter convolutional neural network



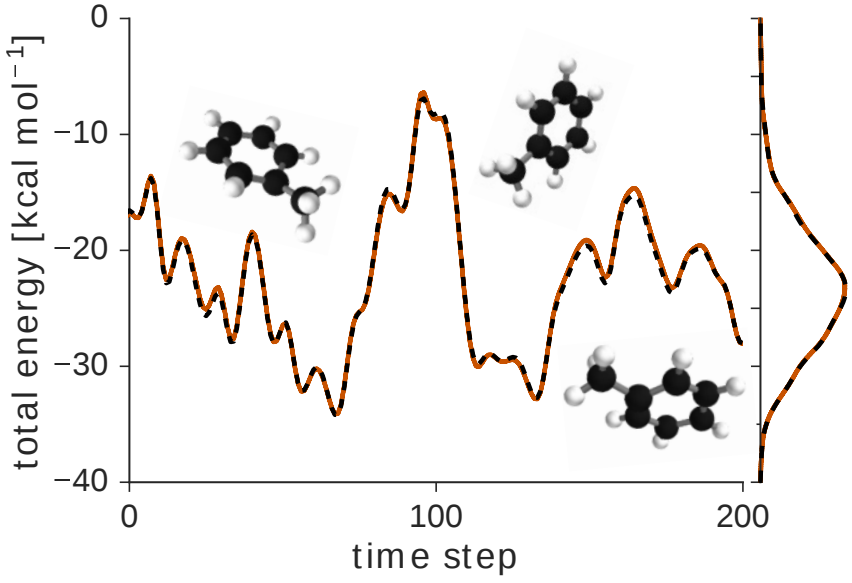
K.T. Schütt, P.J. Kindermans, H.E. Sauceda, S. Chmiela, A. Tkatchenko, K.-R. Müller (2017). SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. NIPS 30.

SchNet: Application to molecules

Structure optimization



Molecular dynamics simulation



Summary

- Convolutional layers
  - reduce number of parameters (weight sharing)
  - translational invariance
  - model locally correlated structure
- CNNs are suitable for
  - 1d: text, time series
  - 2d: images
  - 3d: videos, molecules / materials
  - nd: graphs