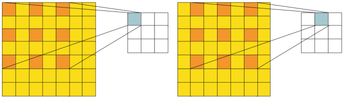
Deep Learning

Important Types of Convolutions



Learning goals

- Dilated Convolutions
- Transposed Convolutions

Dilated Convolutions

DILATED CONVOLUTIONS

- Idea : artificially increase the receptive field of the net without using more filter weights.
- The **receptive field** of a single neuron comprises all inputs that have an impact on this neuron.
- Neurons in the first layers capture less information of the input, while neurons in the last layers have huge receptive fields and can capture a lot more global information from the input.
- The size of the receptive fields depends on the filter size.

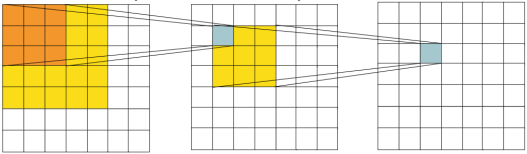


Figure: Receptive field of each convolution layer with a 3×3 kernel. The orange area marks the receptive field of one pixel in Layer 2, the yellow area marks the receptive field of one pixel in layer 3.

DILATED CONVOLUTIONS

- Intuitively, neurons in the first layers capture less information of the input (layer), while neurons in the last layers have huge receptive fields and can capture a lot more global information from the input (layer).
- The size of the receptive fields depends on the filter size.

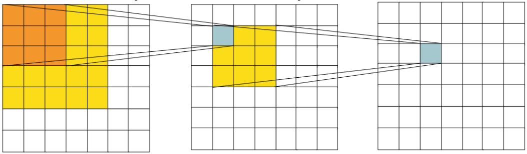


Figure: A convolutional neural network, convolved with 3 layers with 3×3 kernels. The orange area marks the receptive field of one neuron in Layer 2 w.r.t. the input layer (size 9), the yellow area marks the receptive field of one pixel in layer 3.

DILATED CONVOLUTIONS

- By increasing the filter size, the size of the receptive fields increases as well and more contextual information can be captured.
- However, increasing the filter size increases the number of parameters, which leads to increased runtime.
- Artificially increase the receptive field of the net without using more filter weights by adding a new dilation parameter to the kernel that skips pixels during convolution.
- Benefits:
  - Capture more contextual information.
  - Enable the processing of inputs in higher dimensions to detect fine details.
  - Improved run-time-performance due to less parameters.

DILATED CONVOLUTIONS

- Useful in applications where the global context is of great importance for the model decision.
- This component finds application in:
  - Generation of audio-signals and songs within the famous Wavenet developed by DeepMind.
  - Time series classification and forecasting.
  - Image segmentation.

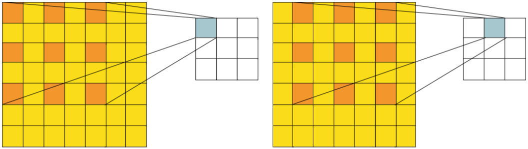


Figure: Dilated convolution on 2D data. A dilated kernel is a regular convolutional kernel interleaved with zeros.

DILATED CONVOLUTIONS

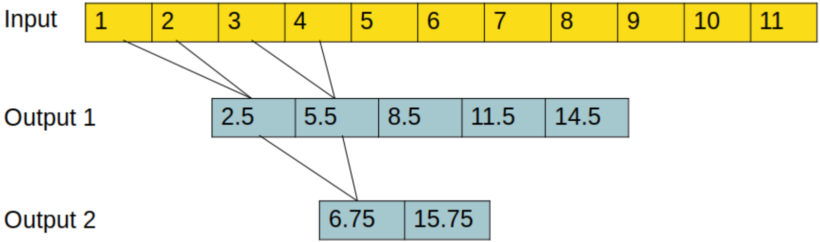


Figure: Simple 1D convolutional network with convolutional kernel of size 2, stride 2 and fixed weights $\{0.5, 1.0\}$.
The kernel is not dilated (**dilation factor 1**). One neuron in layer 2 has a receptive field of size 4 w.r.t. the input layer.

DILATED CONVOLUTIONS

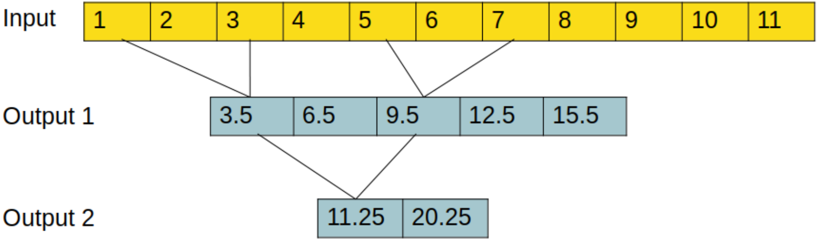


Figure: Simple 1D convolutional network with convolutional kernel of size 2, stride 2 and fixed weights $\{0.5, 1.0\}$.
The kernel is dilated with **dilation factor 2**. One neuron in layer 2 has a receptive field of size 7 w.r.t. the input layer.

DILATED CONVOLUTIONS

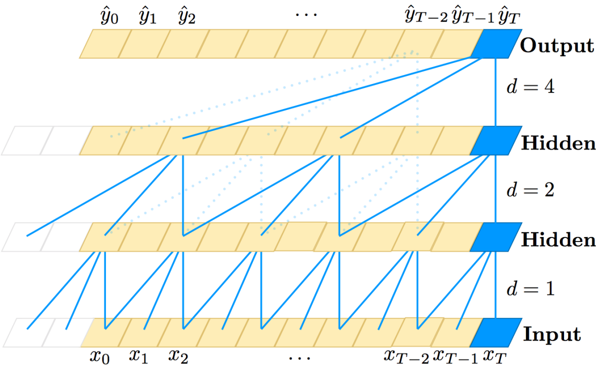


Figure: Application of (a variant of) dilated convolutions on time series for classification or seq2seq prediction (e.g. machine translation). Given an input sequence $x_0, x_1, \ldots, x_T$, the model generates an output sequence $\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_T$. Dilation factors $d = 1, 2, 4$ shown above, each with a kernel size $k = 3$. The dilations are used to drastically increase the context information for each output neuron with relatively few layers.

Transposed Convolutions

TRANSPOSED CONVOLUTIONS

- Problem setting:
  - For many applications and in many network architectures, we often want to do transformations going in the opposite direction of a normal convolution, i.e. we would like to perform up-sampling.
  - examples include generating high-resolution images and mapping low dimensional feature map to high dimensional space such as in auto-encoder or semantic segmentation.
- Instead of decreasing dimensionality as with regular convolutions, **transposed convolutions** are used to re-increase dimensionality back to the initial dimensionality.
- Note: Do not confuse this with deconvolutions (which are mathematically defined as the inverse of a convolution).

TRANSPOSED CONVOLUTIONS

- Example 1:
  - Input: yellow feature map with dim 4×4.
  - Output: blue feature map with dim 2×2.

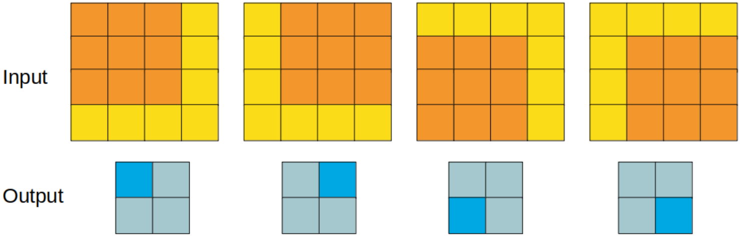


Figure: A **regular** convolution with kernel-size $k = 3$, padding $p = 0$ and stride $s = 1$.

Here, the feature map shrinks from 4×4 to 2×2.

TRANSPOSED CONVOLUTIONS

- One way to upsample is to use a regular convolution with various padding strategies.

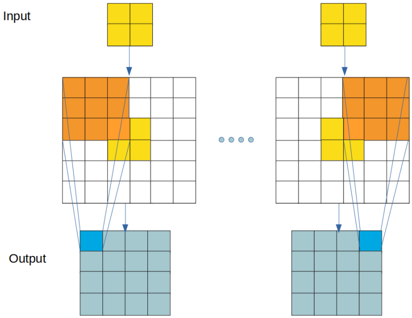


Figure: Transposed convolution can be a seen as a regular convolution. Convolution (above) with $k' = 3, s' = 1, p' = 2$ re-increases dimensionality from 2×2 to 4×4

TRANSPOSED CONVOLUTIONS

- Convolution with parameters kernel size k, stride s and padding factor p
- Associated transposed convolution has parameters $k' = k$, $s' = s$ and $p' = k - 1$

TRANSPOSED CONVOLUTIONS

Example 2 : Convolution as a matrix multiplication :

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

credit:Stanford University

Figure: A "regular" 1D convolution. stride = 1 , padding = 1. The vector *a* is the 1D input feature map.

TRANSPOSED CONVOLUTIONS

Example 2 : Transposed Convolution as a matrix multiplication :

$$\vec{x} *^{T} \vec{a} = X^{T}\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

credit:Stanford University

Figure: "Transposed" convolution upsamples a vector of length 4 to a vector of length 6. Stride is 1. Note the change in padding.

Important : Even though the "structure" of the matrix here is the transpose of the original matrix, the non-zero elements are, in general, different from the correponding elements in the original matrix. These (non-zero) elements/weights are tuned by backpropagation.

TRANSPOSED CONVOLUTIONS

Example 3: Convolution as matrix multiplication:

$$\underbrace{\begin{bmatrix} w_1 & w_2 & w_3 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & w_3 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{bmatrix}}_{z} = \begin{bmatrix} w_1 z_1 + w_2 z_2 + w_3 z_3 \\ w_1 z_2 + w_2 z_3 + w_3 z_4 \\ w_1 z_3 + w_2 z_4 + w_3 z_5 \\ w_1 z_4 + w_2 z_5 + w_3 z_6 \end{bmatrix} = \begin{bmatrix} z_7 \\ z_8 \\ z_9 \\ z_{10} \end{bmatrix}$$

Figure: A regular 1D convolution with stride = 1 ,and padding = 0. The vector z is in the input feature map. The matrix K represents the convolution operation.

A regular convolution decreases the dimensionality of the feature map from 6 to 4.

TRANSPOSED CONVOLUTIONS

Example 3: Transposed Convolution as matrix multiplication:

$$\underbrace{\begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ w_3 & w_2 & w_1 & 0 \\ 0 & w_3 & w_2 & w_1 \\ 0 & 0 & w_3 & w_2 \\ 0 & 0 & 0 & w_3 \end{bmatrix}}_{K^\top} \begin{bmatrix} z_7 \\ z_8 \\ z_9 \\ z_{10} \end{bmatrix} = \begin{bmatrix} w_1 z_7 \\ w_2 z_7 + w_1 z_8 \\ w_3 z_7 + w_2 z_8 + w_1 z_9 \\ w_3 z_8 + w_2 z_9 + w_1 z_{10} \\ w_3 z_9 + w_2 z_{10} \\ w_3 z_{10} \end{bmatrix} = \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \\ \tilde{z}_4 \\ \tilde{z}_5 \\ \tilde{z}_6 \end{bmatrix}$$

Figure: A transposed convolution can be used to upsample the feature vector of length 4 back to a feature vector of length 6.

Note:

- Even though the transpose of the original matrix is shown in this example, the actual values of the weights are different from the original matrix (and optimized by backpropagation).
- The goal of the transposed convolution here is simply to get back the original dimensionality. It is *not* necessarily to get back the original feature map itself.

TRANSPOSED CONVOLUTIONS

Example 3: Transposed Convolution as matrix multiplication:

$$\underbrace{\begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ w_3 & w_2 & w_1 & 0 \\ 0 & w_3 & w_2 & w_1 \\ 0 & 0 & w_3 & w_2 \\ 0 & 0 & 0 & w_3 \end{bmatrix}}_{K^\top} \begin{bmatrix} z_7 \\ z_8 \\ z_9 \\ z_{10} \end{bmatrix} = \begin{bmatrix} w_1 z_7 \\ w_2 z_7 + w_1 z_8 \\ w_3 z_7 + w_2 z_8 + w_1 z_9 \\ w_3 z_8 + w_2 z_9 + w_1 z_{10} \\ w_3 z_9 + w_2 z_{10} \\ w_3 z_{10} \end{bmatrix} = \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \\ \tilde{z}_4 \\ \tilde{z}_5 \\ \tilde{z}_6 \end{bmatrix}$$

Figure: A transposed convolution can be used to upsample the feature vector of length 4 back to a feature vector of length 6.

Note:

- The elements in the downsampled vector only affect those elements in the upsampled vector that they were originally "derived" from. For example, z_7 was computed using z_1 , z_2 and z_3 and it is only used to compute $\tilde{z}_1$, $\tilde{z}_2$ and $\tilde{z}_3$.
- In general, transposing the original matrix does not result in a convolution. But a transposed convolution can always be implemented as a regular convolution by using various padding strategies (this would not be very efficient, however).

TRANSPOSED CONVOLUTIONS

Example 4: Let us now view transposed convolutions from a different perspective.

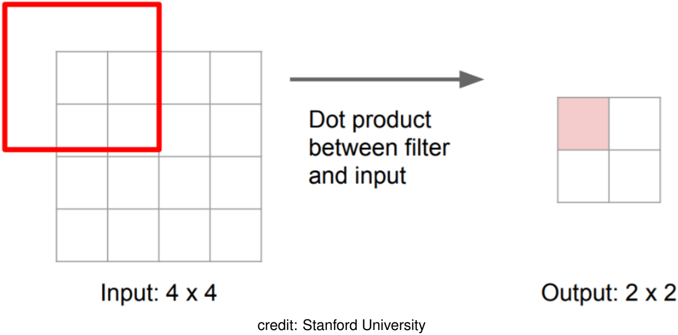


Figure: Regular 3 $\times$ 3 convolution, stride 2, padding 1.

TRANSPOSED CONVOLUTIONS

Example 4: Let us now view transposed convolutions from a different perspective.

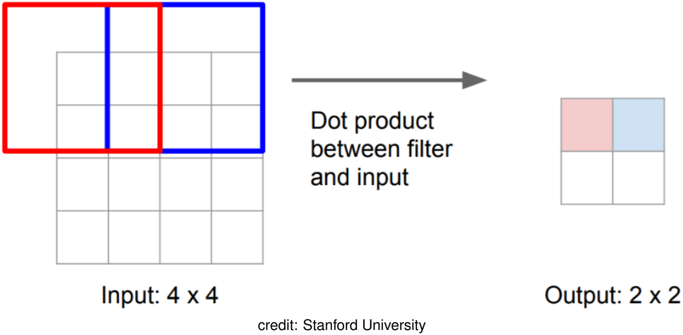
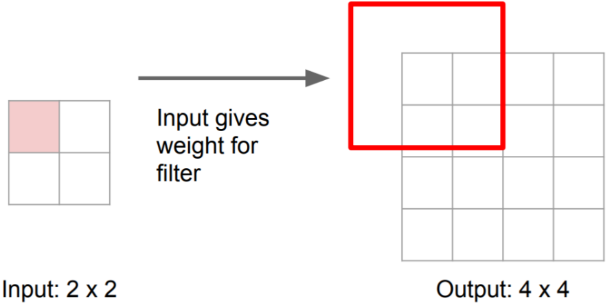


Figure: Regular 3×3 convolution, stride 2, padding 1.

TRANSPOSED CONVOLUTIONS

Example 4: Let us now view transposed convolutions from a different perspective.



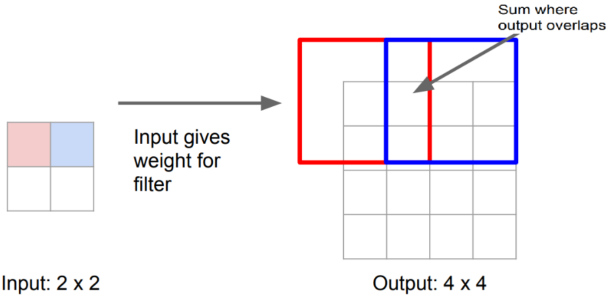
credit: Stanford University

Figure: *Transposed* 3×3 convolution, stride 2, padding 1. Note: stride now refers to the "stride" in the *output*.

Here, the filter is *scaled* by the input.

TRANSPOSED CONVOLUTIONS

Example 4: Let us now view transposed convolutions from a different perspective.



credit: Stanford University

Figure: *Transposed* 3×3 convolution, stride 2, padding 1. Note: stride now refers to the "stride" in the *output*.

Here, the filter is *scaled* by the input.

TRANSPOSED CONVOLUTIONS – DRAWBACK

Figure: Artifacts produced by transposed convolutions.

- Transposed convolutions lead to checkerboard-style artifacts in resulting images.

TRANSPOSED CONVOLUTIONS – DRAWBACK

- Explanation: transposed convolution yields an overlap in some feature map values.
- This leads to higher magnitude for some feature map elements than for others, resulting in the checkerboard pattern.
- One solution is to ensure that the kernel size is divisible by the stride.

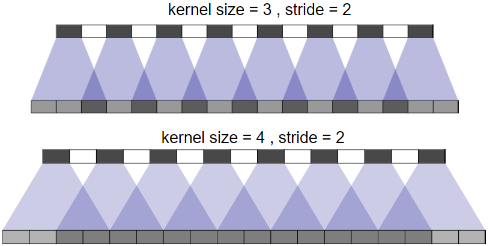


Figure: 1D example. In both images, top row = input and bottom row = output. *Top*: Here, kernel weights overlap unevenly which results in a checkerboard pattern. *Bottom*: There is no checkerboard pattern as the kernel size is divisible by the stride.

TRANSPOSED CONVOLUTIONS – DRAWBACK

- Solutions:
  - Increase dimensionality via upsampling (bilinear, nearest neighbor) and then convolve this output with regular convolution.
  - Make sure that the kernel size *k* is divisible by the stride *s*.

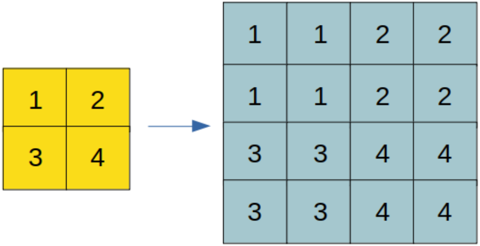


Figure: Nearest neighbor upsampling and subsequent same convolution to avoid checkerboard patterns.

REFERENCES

Dumoulin, Vincent and Visin, Francesco (2016)
A guide to convolution arithmetic for deep learning
https://arxiv.org/abs/1603.07285v1

Van den Oord, Aaron, Sander Dielman, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, and Koray Kavukocuoglu (2016)
WaveNet: A Generative Model for Raw Audio
https://arxiv.org/abs/1609.03499

Benoit A., Gennart, Bernard Krummenacher, Roger D. Hersch, Bernard Saugy, J.C. Hadorn and D. Mueller (1996)
The Giga View Multiprocessor Multidisk Image Server
https://www.researchgate.net/publication/220060811_The_Giga_View_Multiprocessor_Multidisk_Image_Server

Tran, Du, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani and Paluri Manohar (2015)
Learning Spatiotemporal Features with 3D Convolutional Networks
https://arxiv.org/pdf/1412.0767.pdf

REFERENCES

Milletari, Fausto, Nassir Navab and Seyed-Ahmad Ahmadi (2016)
V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation
https://arxiv.org/pdf/1606.04797.pdf

Zhang, Xiang, Junbo Zhao and Yann LeCun (2015)
Character-level Convolutional Networks for Text Classification
http://arxiv.org/abs/1509.01626

Wang, Zhiguang, Weizhong Yan and Tim Oates (2017)
Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline
http://arxiv.org/abs/1509.01626

Fisher Yu and Vladlen Koltun (2015)
Multi-Scale Context Aggregation by Dilated Convolutions
https://arxiv.org/abs/1511.07122

REFERENCES

Bai, Shaojie, Zico J. Kolter and Vladlen Koltun (2018)
An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling
http://arxiv.org/abs/1509.01626

Augustus Odena, Vincent Dumoulin and Chris Olah (2016)
Deconvolution and Checkerboard Artifacts
https://distill.pub/2016/deconv-checkerboard/ *https://distill.pub/2016/deconv-checkerboard/*

Andre Araujo, Wade Norris and Jack Sim (2019)
Computing Receptive Fields of Convolutional Neural Networks
https://distill.pub/2019/computing-receptive-fields/

Zhiguang Wang, Yan, Weizhong and Tim Oates (2017)
Time series classification from scratch with deep neural networks: A strong baseline
https://arxiv.org/1611.06455

REFERENCES

Lin, Haoning and Shi, Zhenwei and Zou, Zhengxia (2017)
Maritime Semantic Labeling of Optical Remote Sensing Images with Multi-Scale Fully Convolutional Network