LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Deep Learning

Chapter 6: Important Types of Convolutions

Mina Rezaei

Department of Statistics – LMU Munich

Winter Semester 2020

LECTURE OUTLINE

1D Convolutions

2D Convolutions

3D Convolutions

Dilated Convolutions

Transposed Convolutions

Separable Convolutions

Flattened Convolutions

1D Convolutions

1D CONVOLUTIONS

Data situation: Sequential, 1-dimensional tensor data.

- Data consists of tensors with shape [depth, xdim]
- Depth 1 (single-channel):
  - Univariate time series, e.g. development of a single stock price over time
  - Functional / curve data
- Depth $>$ 1 (mutli-channel):
  - Multivariate time series, e.g.
    - Movement data measured with multiple sensors for human activity recognition [Wang et al., 2017]
    - Temperature and humidity in weather forecasting
  - Text encoded as character-level one-hot-vectors [Zhang et al., 2015]

$\rightarrow$ Convolve the data with a 1D-kernel

1D CONVOLUTIONS – OPERATION

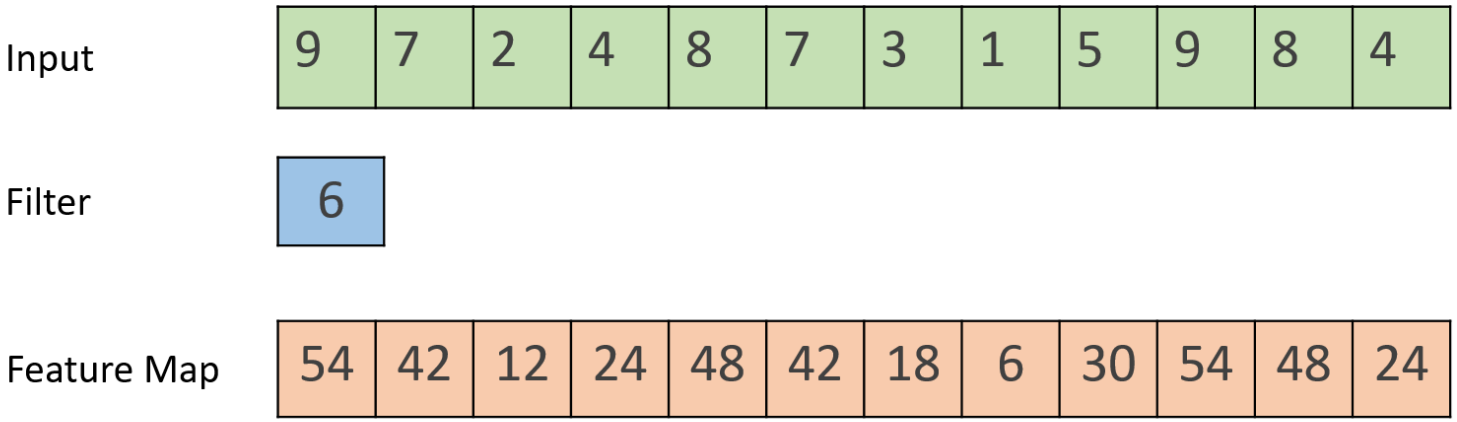


Figure: Illustration of 1D movement data with depth 1 and filter size 1.

1D CONVOLUTIONS – OPERATION

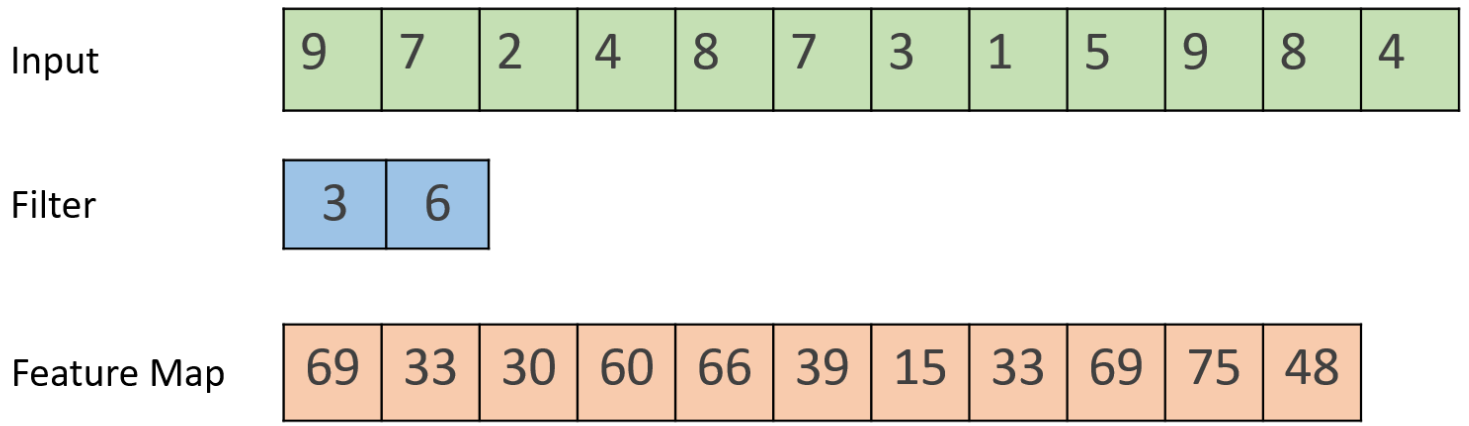


Figure: Illustration of 1D movement data with depth 1 and filter size 2.

1D CONVOLUTIONS – SENSOR DATA

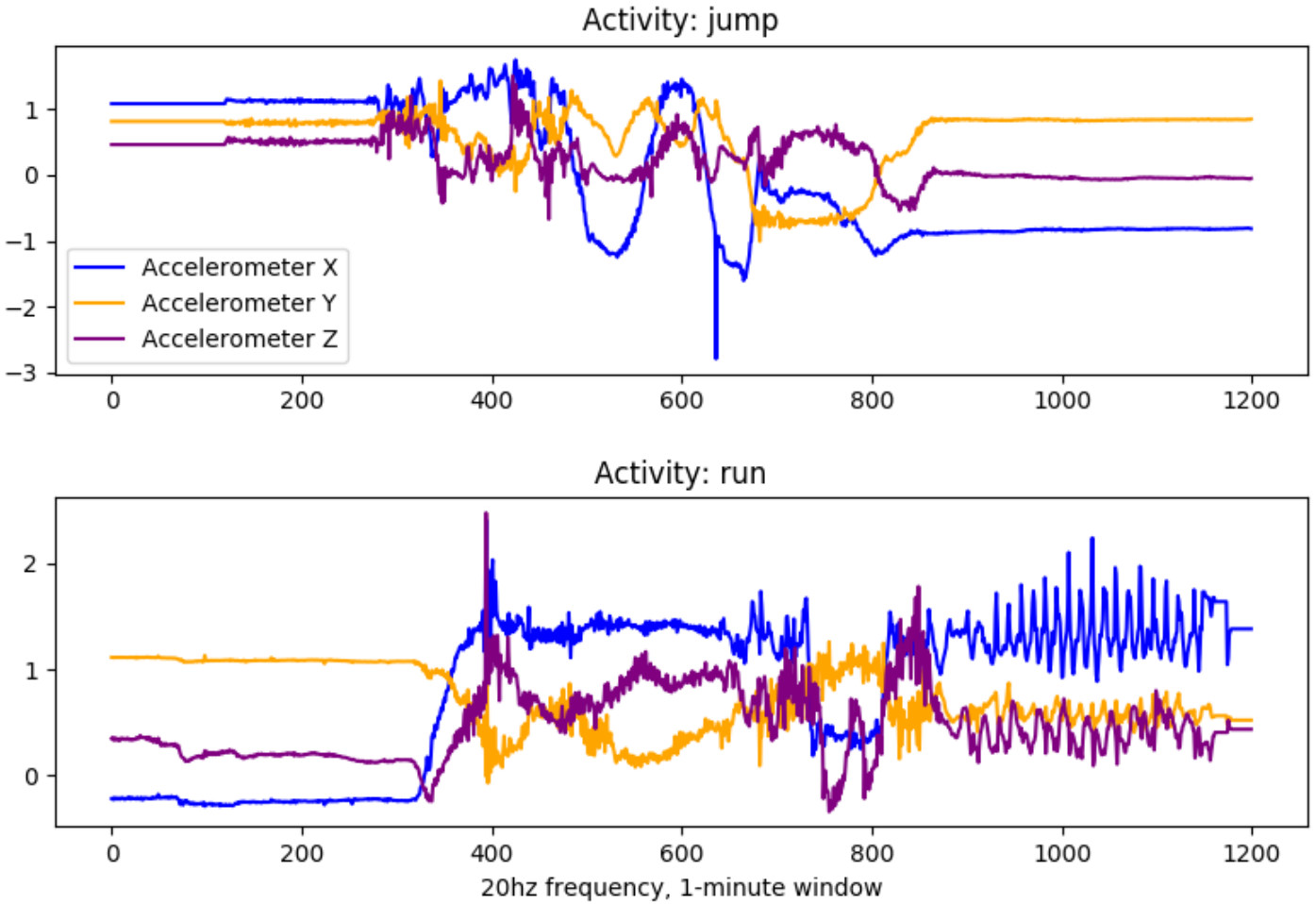


Figure: Illustration of 1D movement data with depth 3 measured with an accelerometer sensor belonging to a human activity recognition task.

1D CONVOLUTIONS – SENSOR DATA

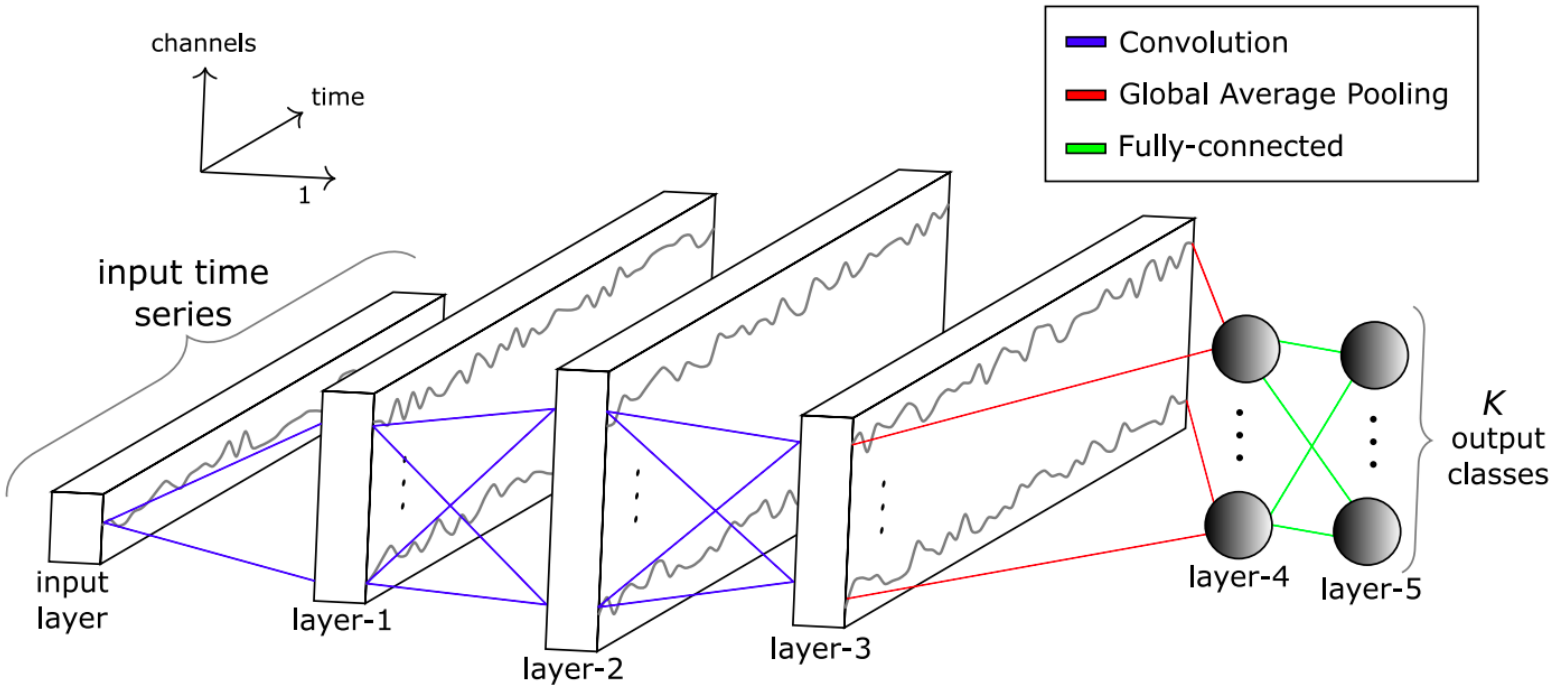


Figure: Time series classification with 1D CNNs and global average pooling (explained later). An input time series is convolved with 3 CNN layers, pooled and fed into a fully connected layer before the final softmax layer. This is one of the classic time series classification architectures [Wang et. al , 2017].

1D CONVOLUTIONS – TEXT MINING

- 1D convolutions also have an interesting application in text mining [Zhang et al., 2015].
- For example, they can be used to classify the sentiment of text snippets such as yelp reviews.

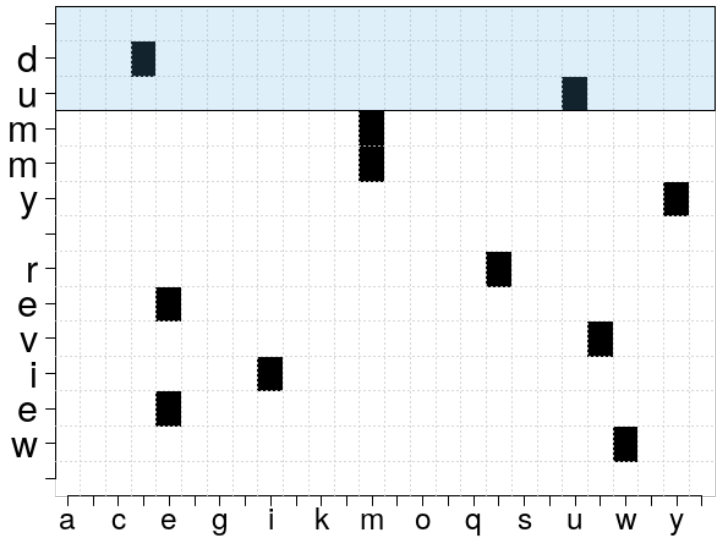
Miriam L.
Munich, Germany
57 friends
437 reviews
450 photos

7/18/2010

The LMU is main building one of the most beautiful buildings in München...nicht only in relation to the architecture just great, but above all also if the history here has taken place, is a conscious.

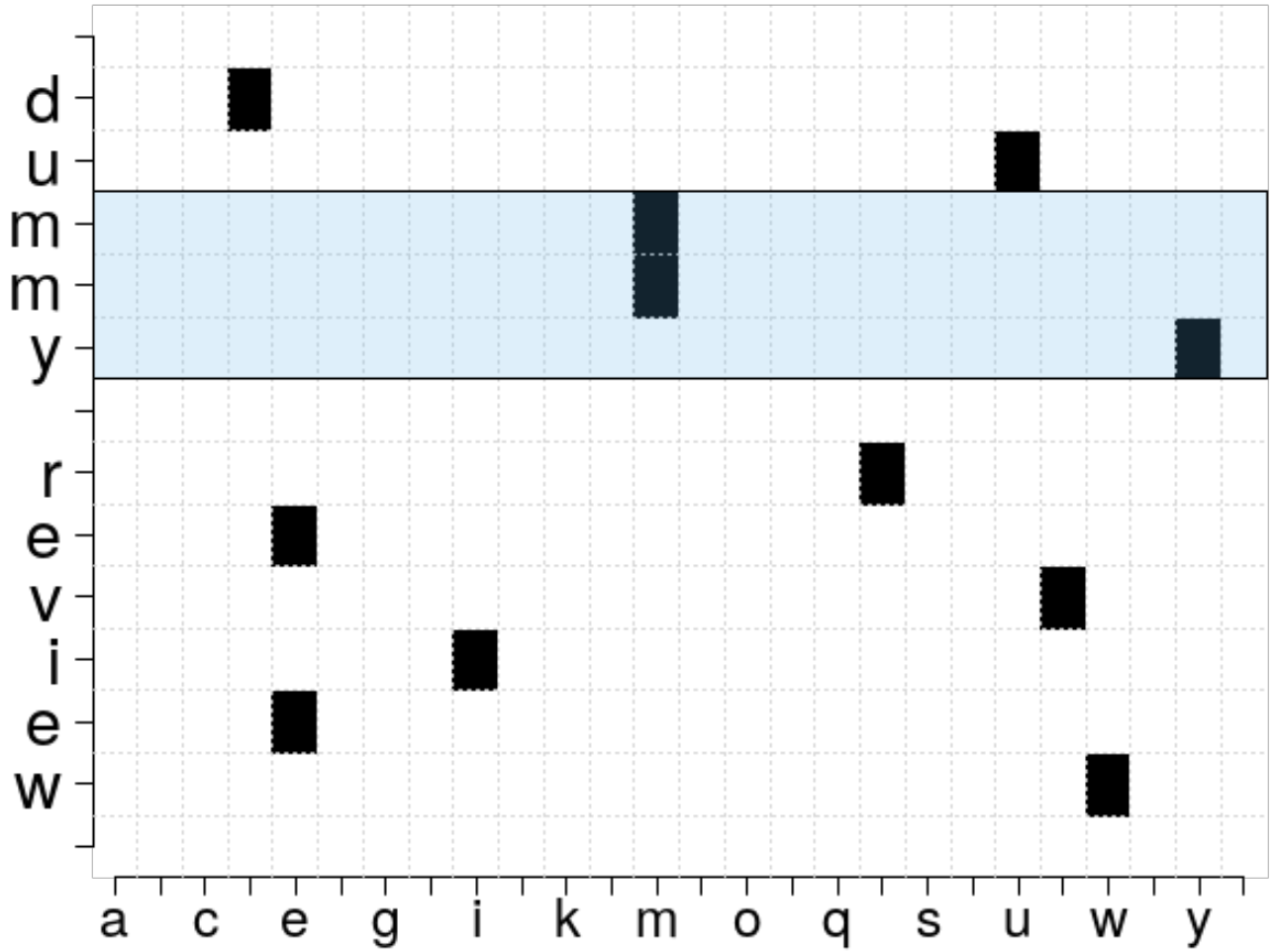
Figure: Sentiment classification: can we teach the net that this a positive review?

1D CONVOLUTIONS – TEXT MINING



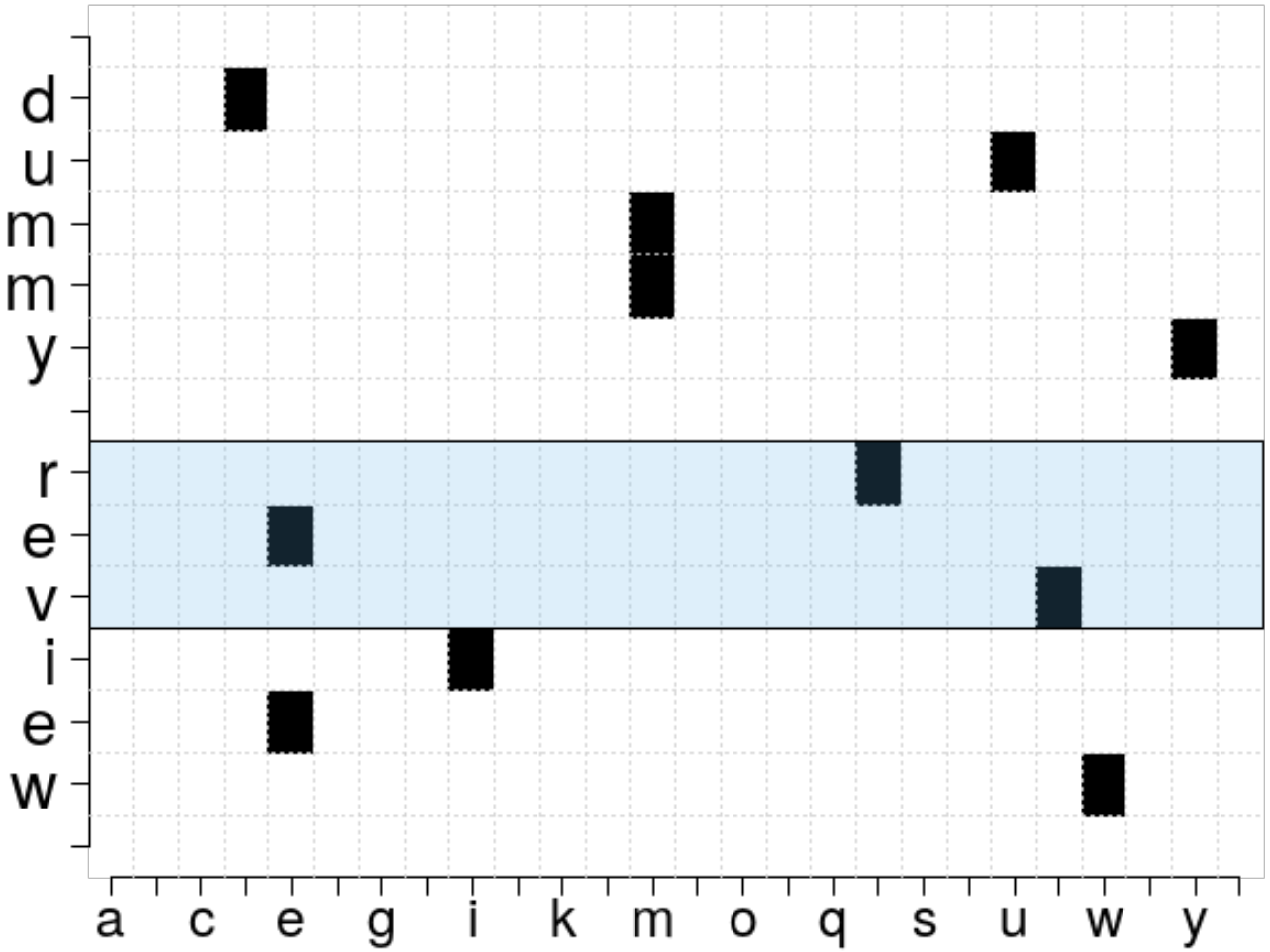
- We use a given alphabet to encode the text reviews (here: *“dummy review”*).
- Each character is transformed into a one-hot vector. The vector for character *d* contains only 0’s at all positions except for the 4th position.
- The maximum length of each review is set to 1014: shorter texts are padded with spaces (zero-vectors), longer texts are simply cut.

1D CONVOLUTIONS – TEXT MINING



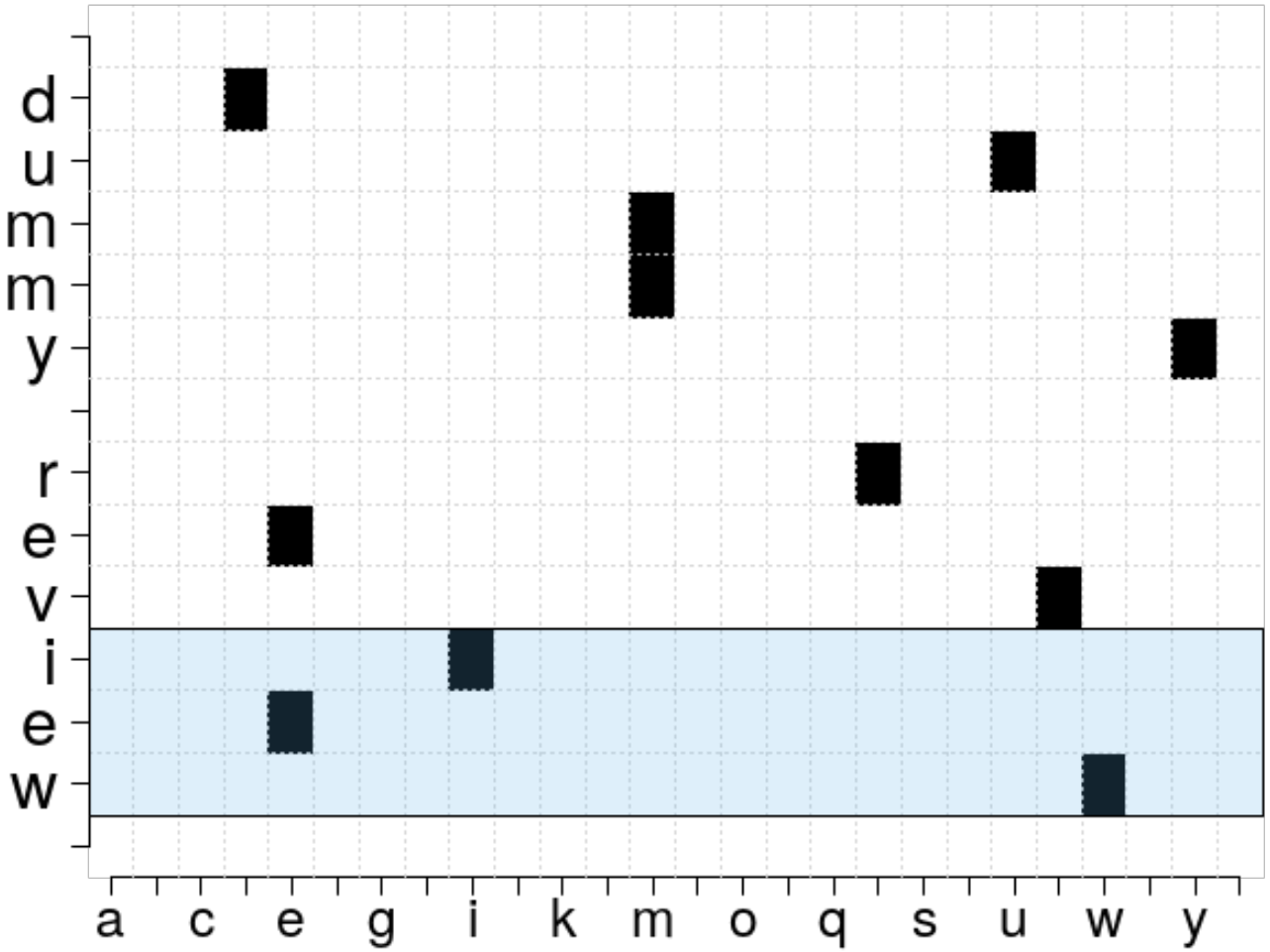
- The data is represented as 1D signal with *depth = size of the alphabet* .

1D CONVOLUTIONS – TEXT MINING



- The temporal dimension is shown as the y dimension for illustrative purposes.

1D CONVOLUTIONS – TEXT MINING



- The 1D-kernel (blue) convolves the input in the temporal y-dimension yielding a 1D feature vector.

ADVANTAGES OF 1D CONVOLUTIONS

A few advantages of 1 x 1 convolutions are:

- Dimensionality reduction for efficient computations
- Efficient low dimensional embedding, or feature pooling
- Applying nonlinearity again after convolution

The first two advantages can be observed in the previous examples. After 1×1 convolution, we significantly reduce the dimension depth-wise. Say if the original input has 200 channels, the 1×1 convolution will embed these channels (features) into a single channel. The third advantage comes in as after the 1×1 convolution, non-linear activation such as ReLU can be added. The non-linearity allows the network to learn more complex function.

2D Convolutions

2D CONVOLUTIONS

The basic idea behind a 2D convolution is sliding a small window (called a "kernel/filter") over a larger 2D array, and performing a dot product between the filter elements and the corresponding input array elements at every position.

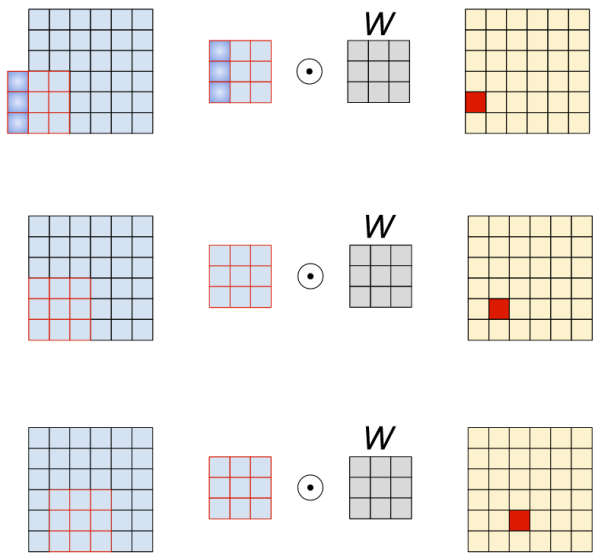
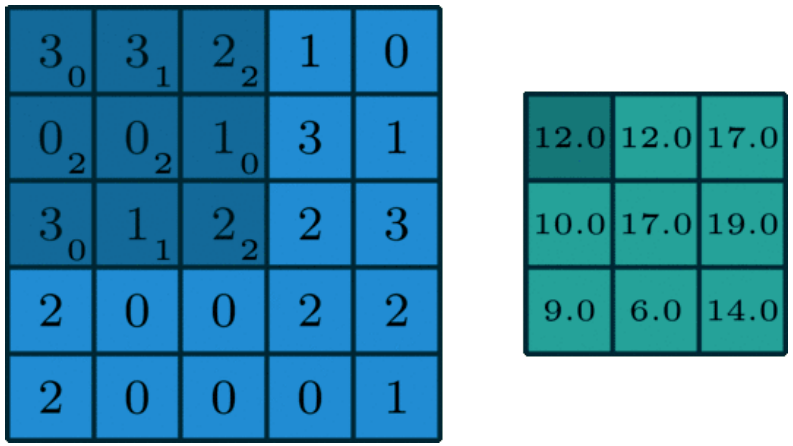


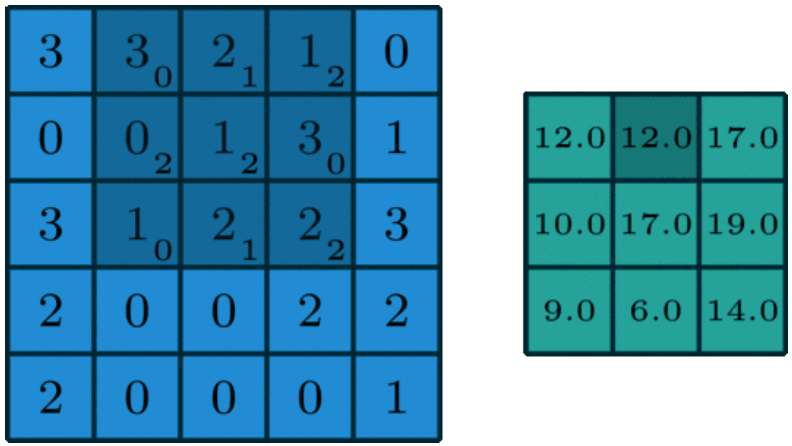
Figure: Here's a diagram demonstrating the application of a 3×3 convolution filter to a 6×6 array, in 3 different positions. W is the filter, and the yellow-ish array on the right is the result; the red square shows which element in the result array is being computed.here.

2D CONVOLUTIONS – EXAMPLE



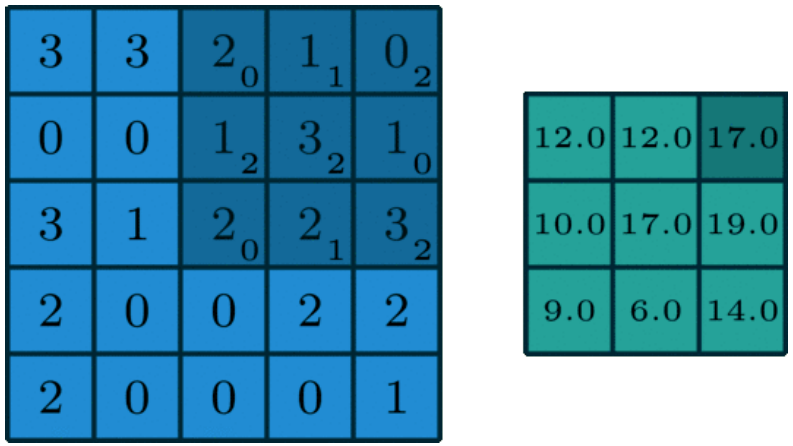
- In Deep Learning, convolution is the element-wise multiplication and addition.
- For an image with 1 channel, the convolution is demonstrated in the figure below. Here the filter is a 3×3 matrix with element [[0, 1, 2], [2, 2, 0], [0, 1, 2]].

2D CONVOLUTIONS – EXAMPLE



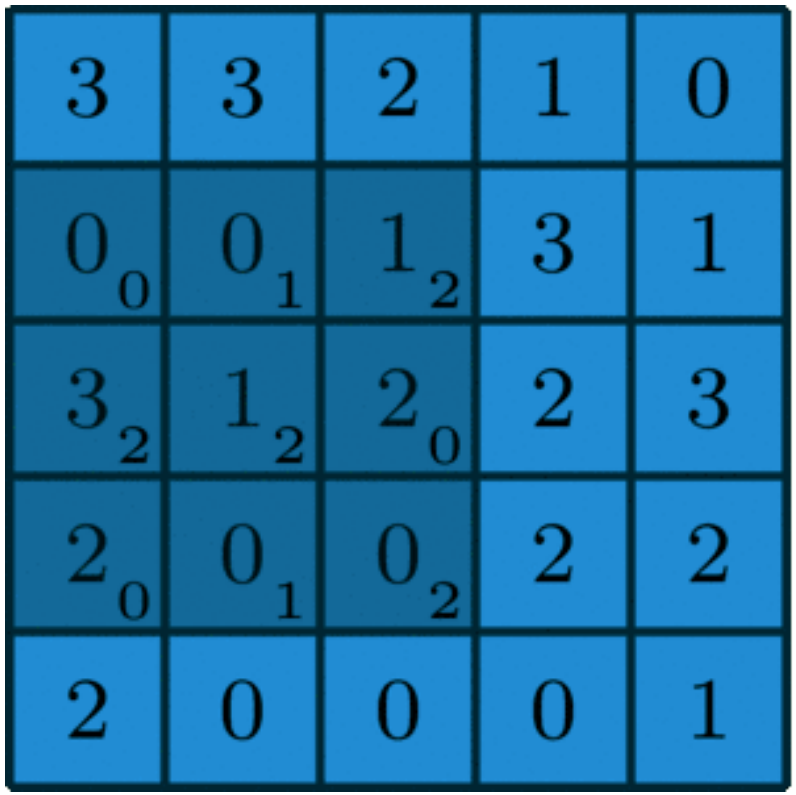
- The filter is sliding through the input.

2D CONVOLUTIONS – EXAMPLE



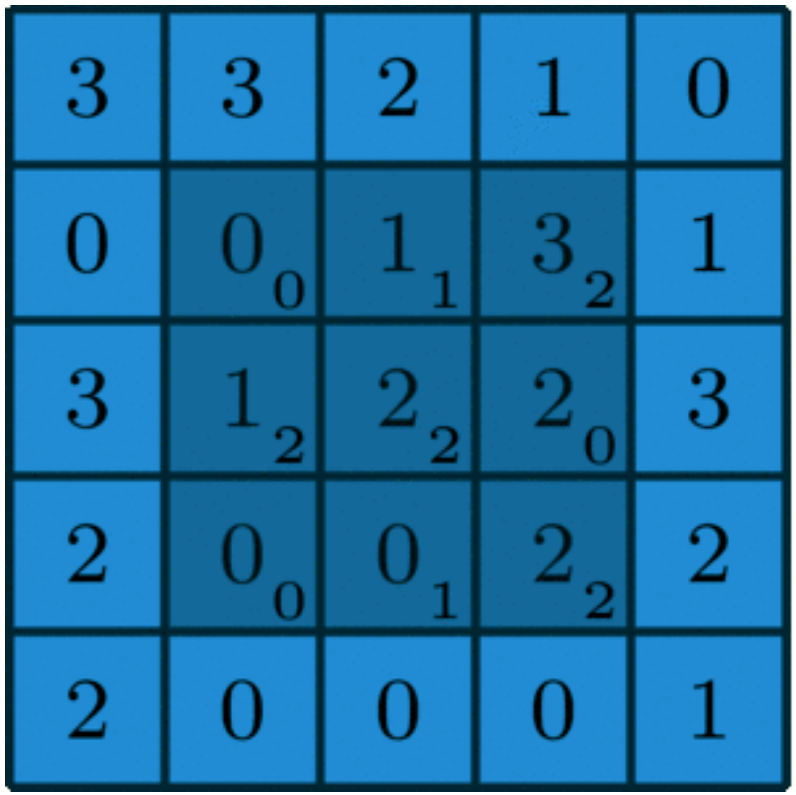
- Each sliding position ends up with one number. The final output is then a 3×3 matrix.

2D CONVOLUTIONS – EXAMPLE



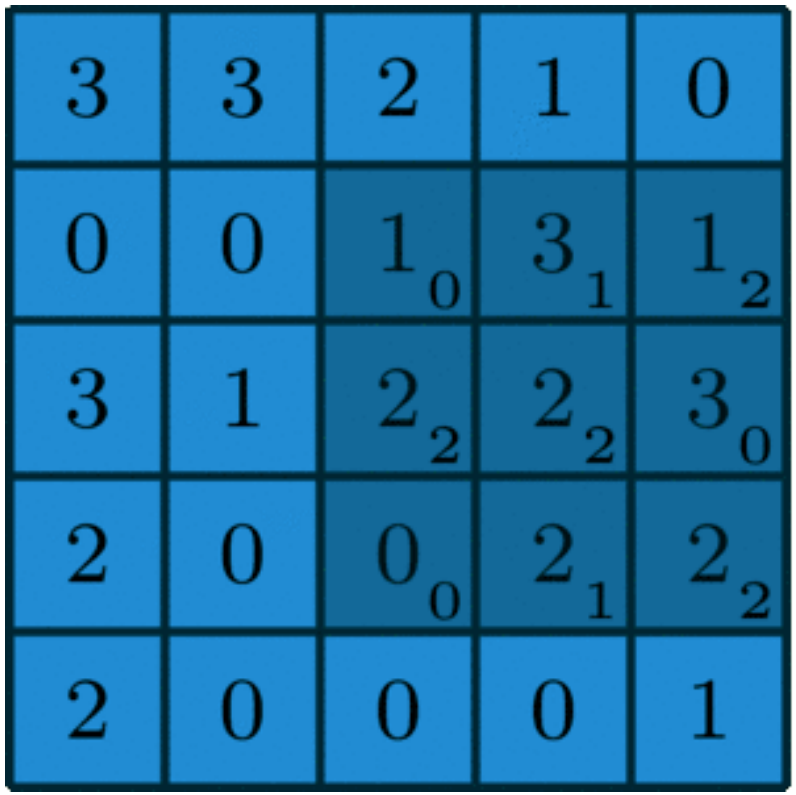
- Notice that stride is 1 and padding is 0 in this example.

2D CONVOLUTIONS – EXAMPLE



- We move/convolve filter on input neurons to create a feature maps.

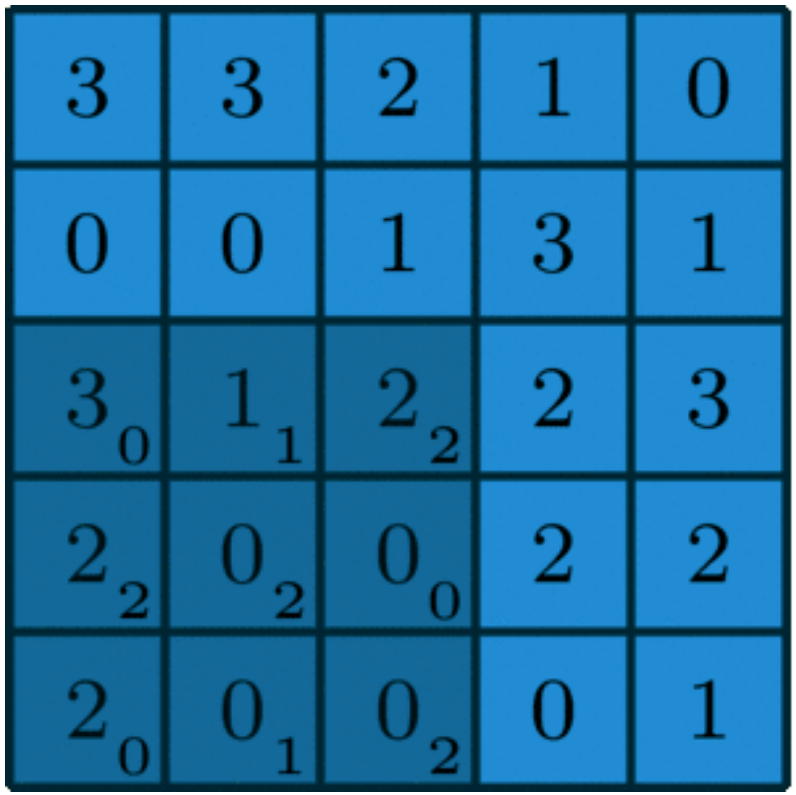
2D CONVOLUTIONS – EXAMPLE



12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

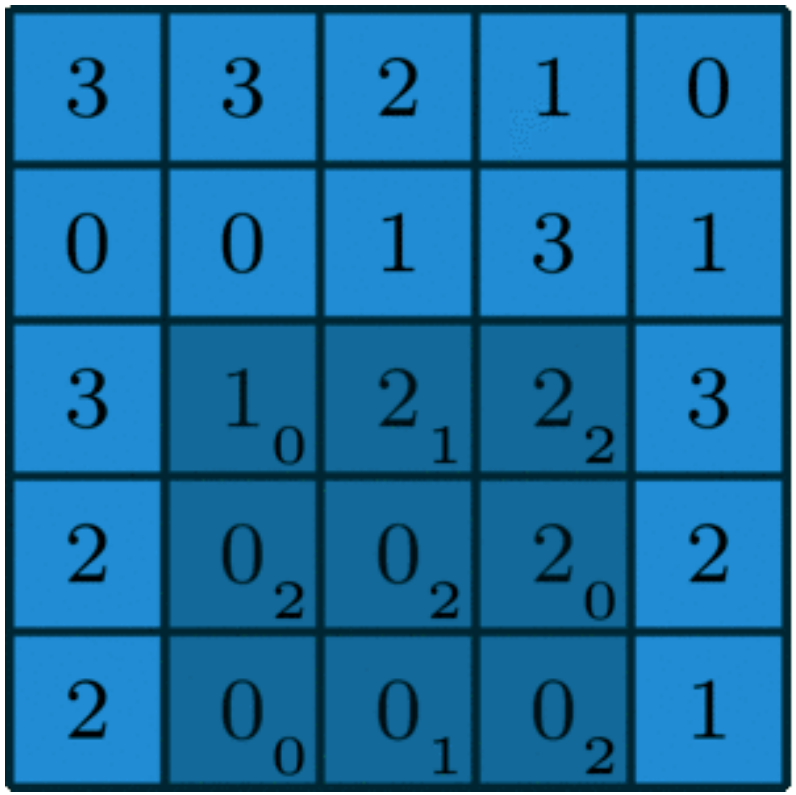
- and ...

2D CONVOLUTIONS – EXAMPLE



- and ...

2D CONVOLUTIONS – EXAMPLE



12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

- and ...

2D CONVOLUTIONS – EXAMPLE

- and at the end we have a 3×3 matrix of feature map!

3D Convolutions

3D CONVOLUTIONS

Data situation: 3-dimensional tensor data.

- Data consists of tensors with shape [depth, xdim, ydim, zdim].
- Dimensions can be both temporal (e.g. video frames) or spatial (e.g. MRI)
- Examples:
  - Human activity recognition in video data [Tran et al., 2015]
  - Disease classification or tumor segmentation on MRI scans [Milletari et al., 2016]

Solution: Move a 3D-kernel in x, y and z direction to capture all important information.

3D CONVOLUTIONS – DATA

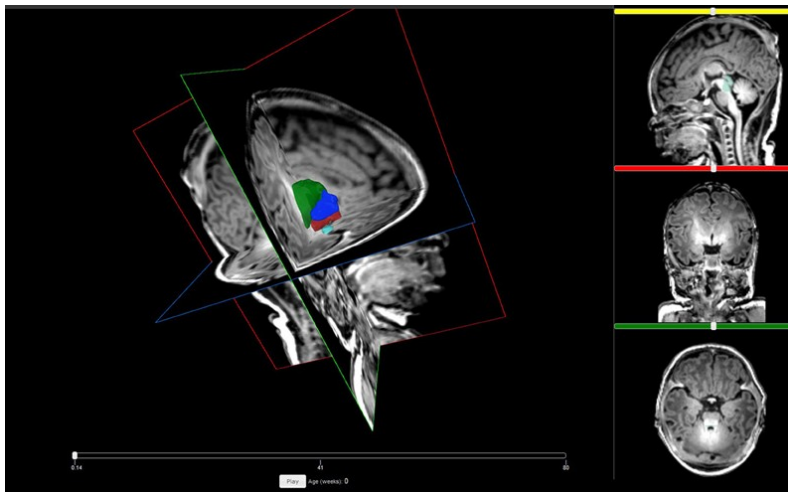


Figure: Illustration of depth 1 volumetric data: MRI scan [Gennart et al., 1996]. Each slice of the stack has depth 1, as the frames are black-white.

3D CONVOLUTIONS – DATA

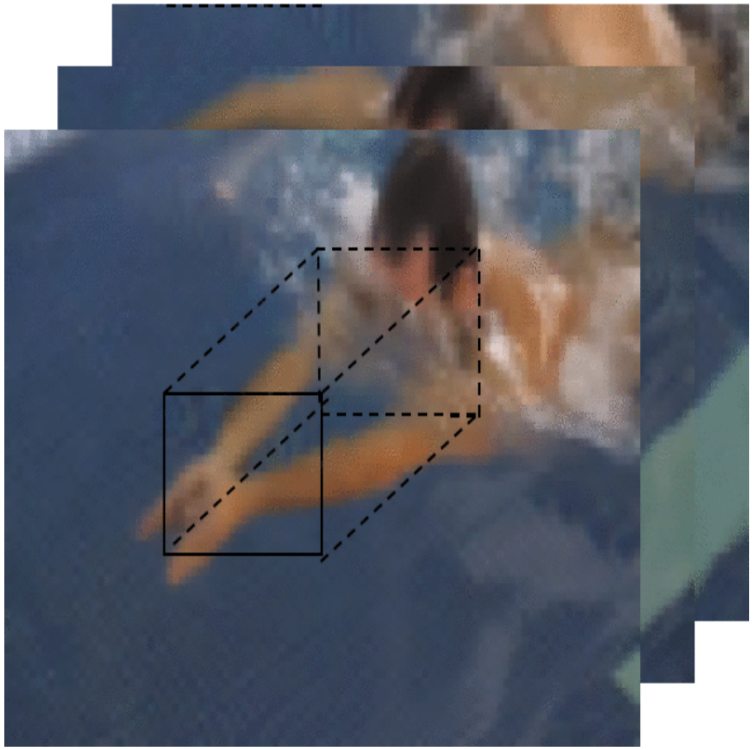
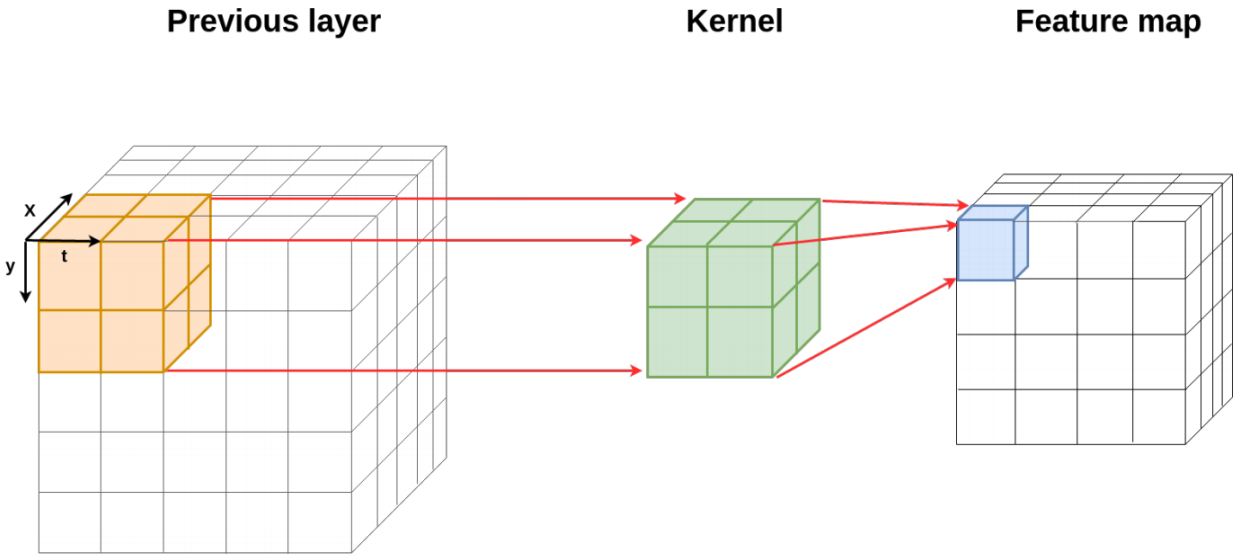


Figure: Illustration of volumetric data with depth > 1: video snippet of an action detection task. The video consists of several slices, stacked in temporal order. Frames have depth 3, as they are RGB.

3D CONVOLUTIONS



- Note: 3D convolutions yield a 3D output.

3D CONVOLUTIONS

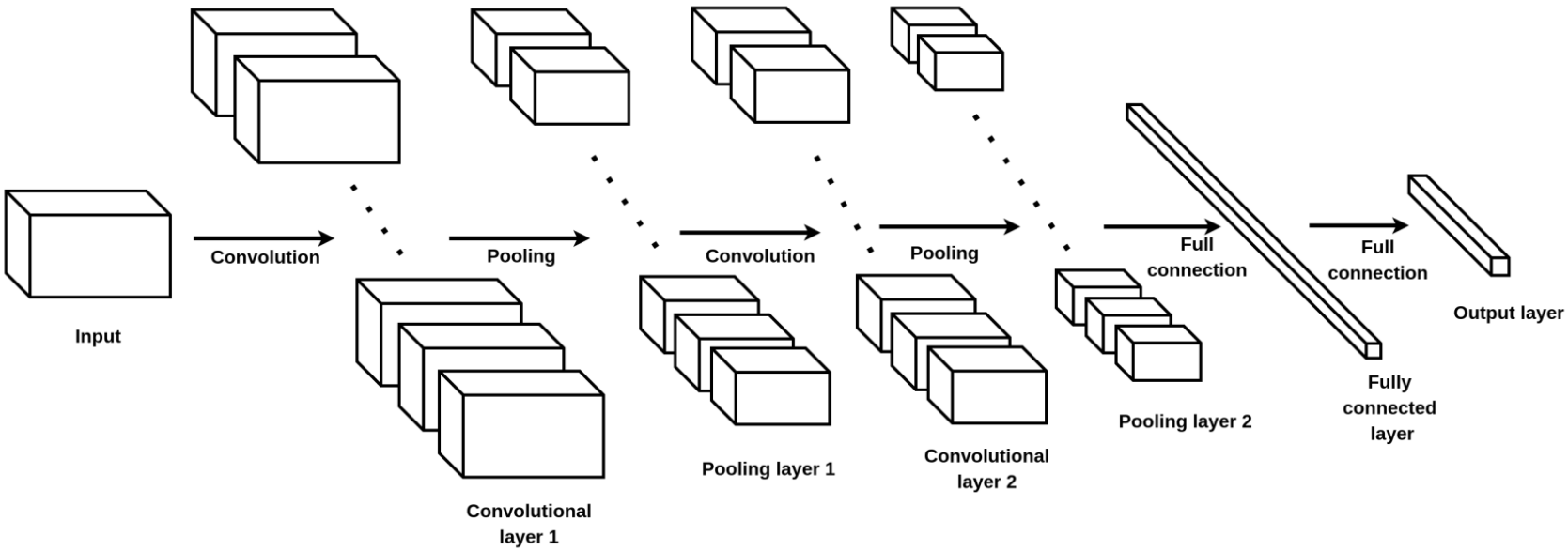


Figure: Basic 3D-CNN architecture.

- Basic architecture of the CNN stays the same.
- 3D convolutions output 3D feature maps which are element-wise activated and then (eventually) pooled in 3 dimensions.

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.

DILATED CONVOLUTIONS

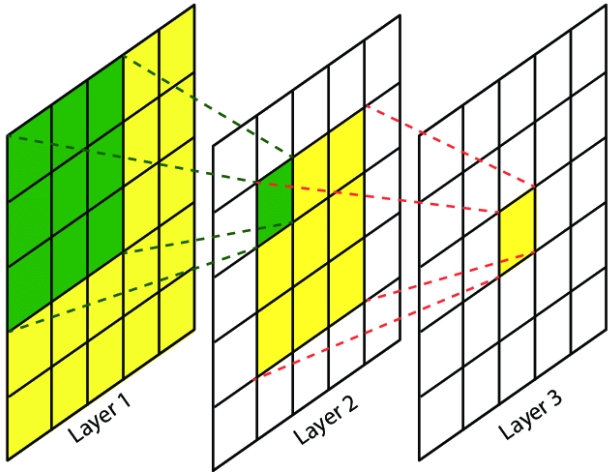


Figure: Receptive field of each convolution layer with a 3×3 kernel. The green area marks the receptive field of one pixel in Layer 2, the yellow area marks the receptive field of one pixel in layer 3 [Lin et al., 2017].

- 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.

DILATED CONVOLUTIONS

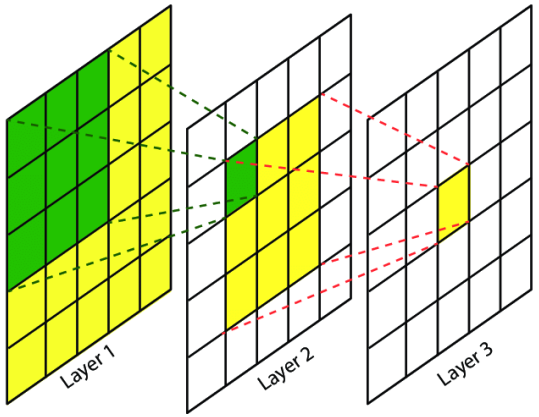


Figure: A convolutional neural network, convolved with 3 layers with 3×3 kernels. The green area marks the receptive field of one neuron in Layer 2 w.r.t. the input layer (size 9), the yellow area marks the receptive field of one pixel in layer 3 [Lin et al., 2017].

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 [van den Oord et al., 2016].
  - Time series classification and forecasting [Bai et. al, 2018].
  - Image segmentation [Yu et. al, 2015].

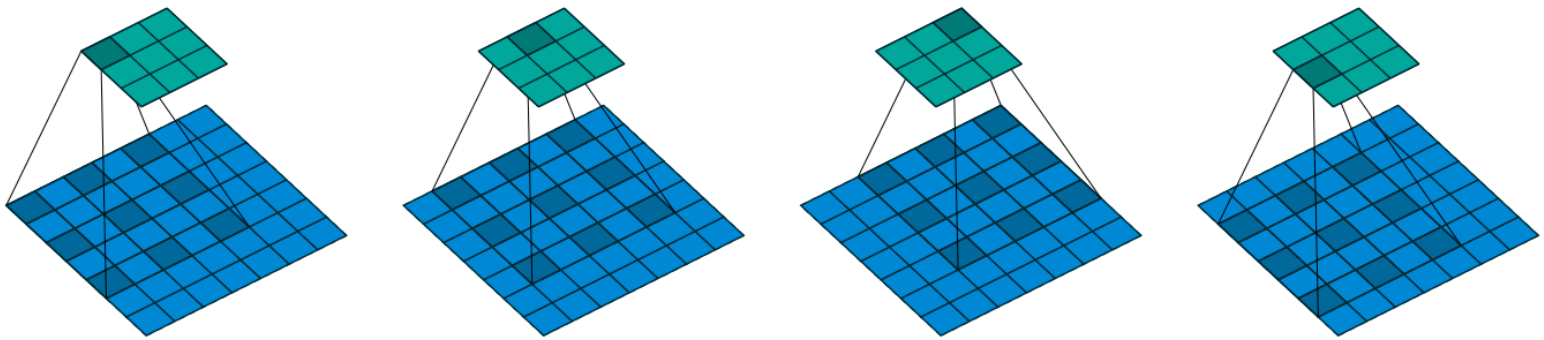


Figure: Dilated convolution on 2D data [Dumoulin et al., 2016]. A dilated kernel is a regular convolutional kernel interleaved with zeros.

DILATED CONVOLUTIONS

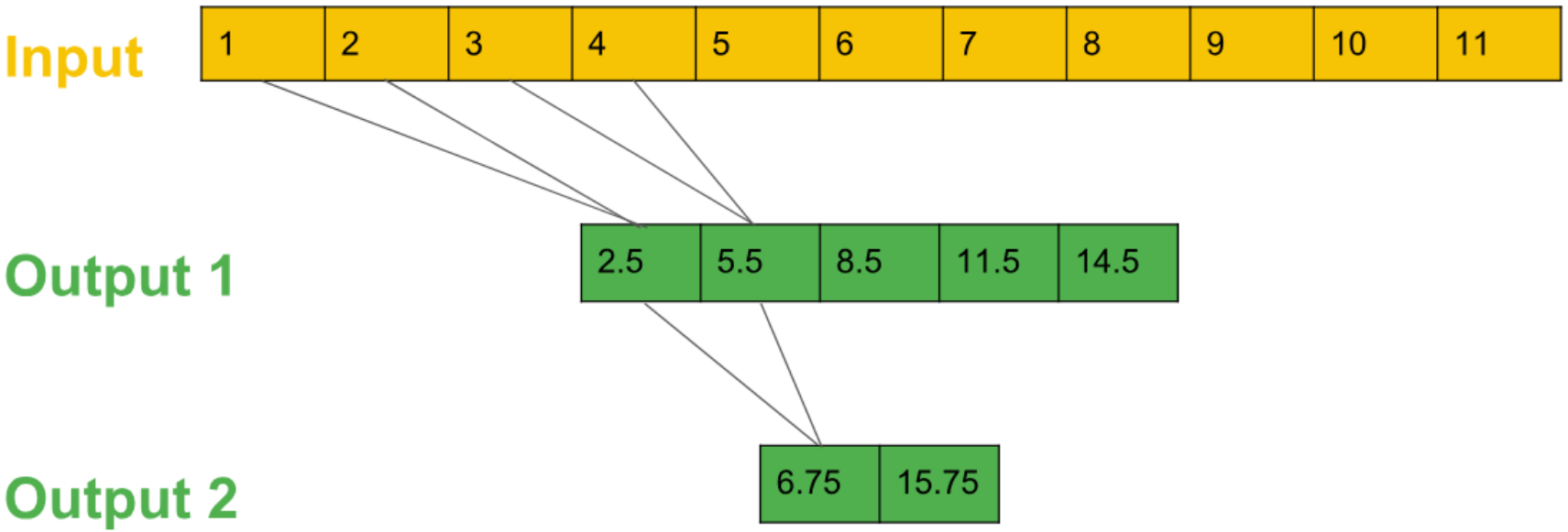


Figure: Simple 1D convolutional network with convolutional kernel of size 2, stride 2 and fixed weights $\{0.5, 1.0\}$.
The kernel is not dilated (**dilation factor 1**). One neuron in layer 2 has a receptive field of size 4 w.r.t. the input layer.

DILATED CONVOLUTIONS

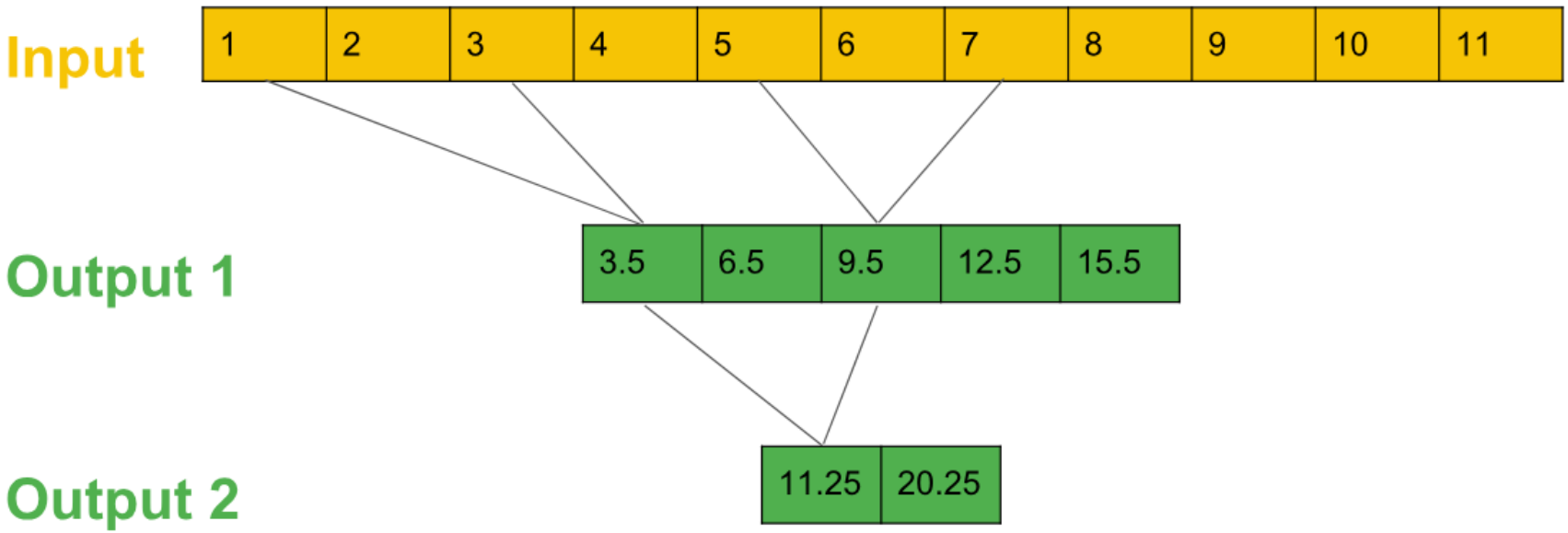


Figure: Simple 1D convolutional network with convolutional kernel of size 2, stride 2 and fixed weights $\{0.5, 1.0\}$.
The kernel is dilated with **dilation factor 2**. One neuron in layer 2 has a receptive field of size 7 w.r.t. the input layer.

DILATED CONVOLUTIONS

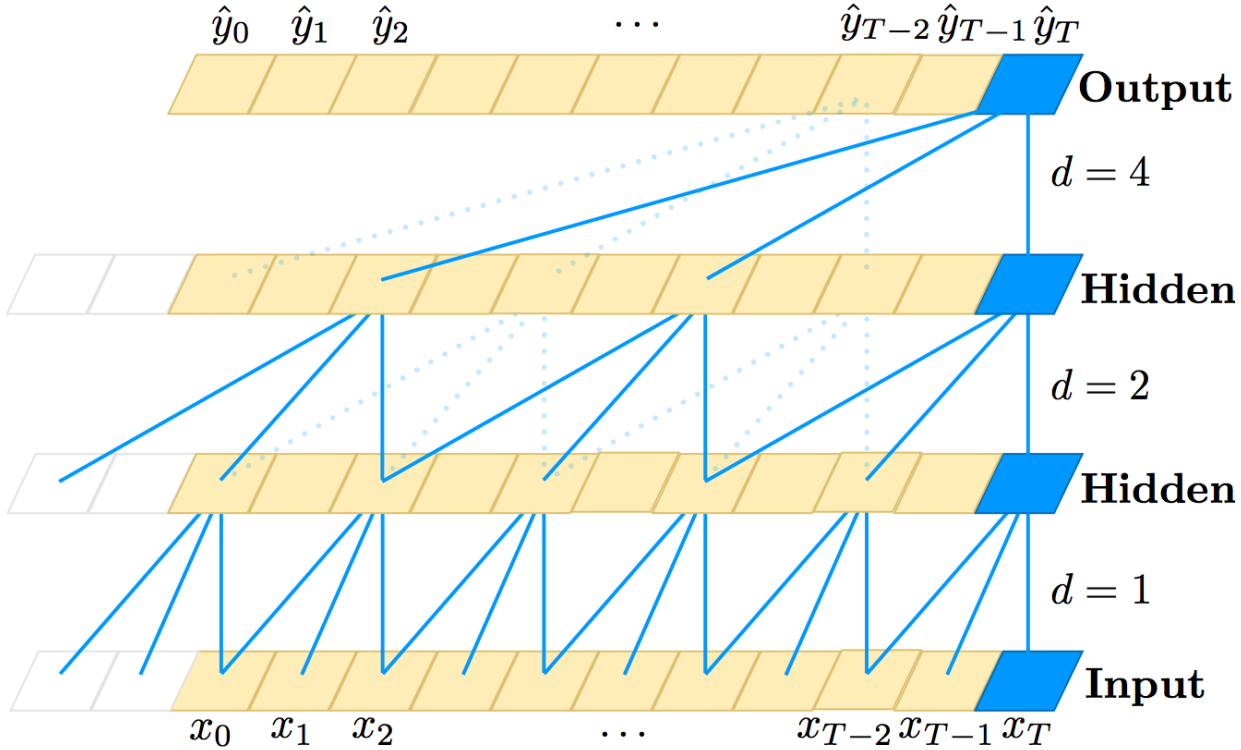


Figure: Application of (a variant of) dilated convolutions on time series for classification or seq2seq prediction (e.g. machine translation)
[Bai et. al, 2018]. Given an input sequence $x_0, x_1, \ldots, x_T$, the model generates an output sequence $\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_T$. Dilation 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: blue feature map with dim 4×4.
  - Output: turquoise feature map with dim 2×2.

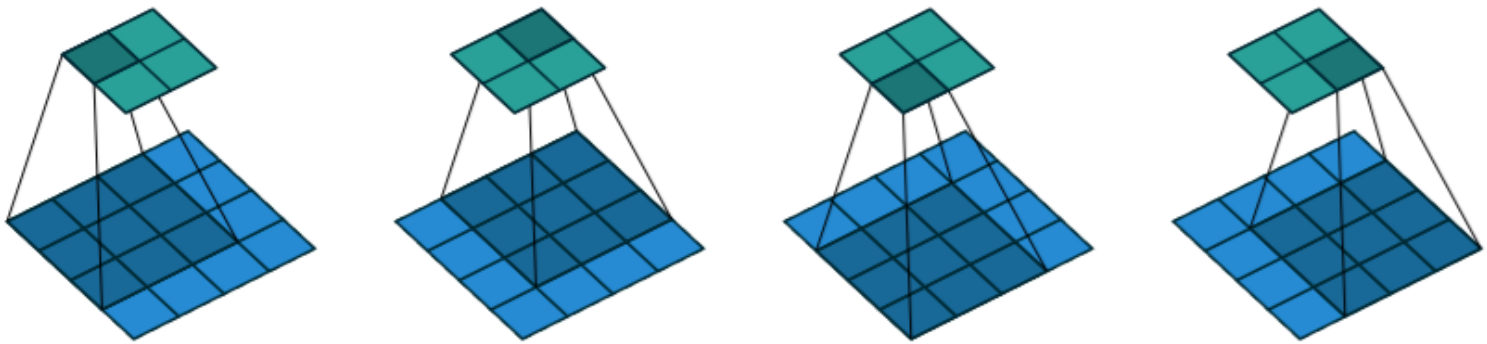


Figure: A **regular** convolution with kernel-size $k = 3$, padding $p = 0$ and stride $s = 1$.

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

TRANSPOSED CONVOLUTIONS

- Example 1:
  - Now, let us upsample the 2×2 feature map back to a 4×4 feature map.
  - Input: 2×2 (blue). Output: 4×4 (turquoise).
- One way to upsample is to use a regular convolution with various padding strategies.

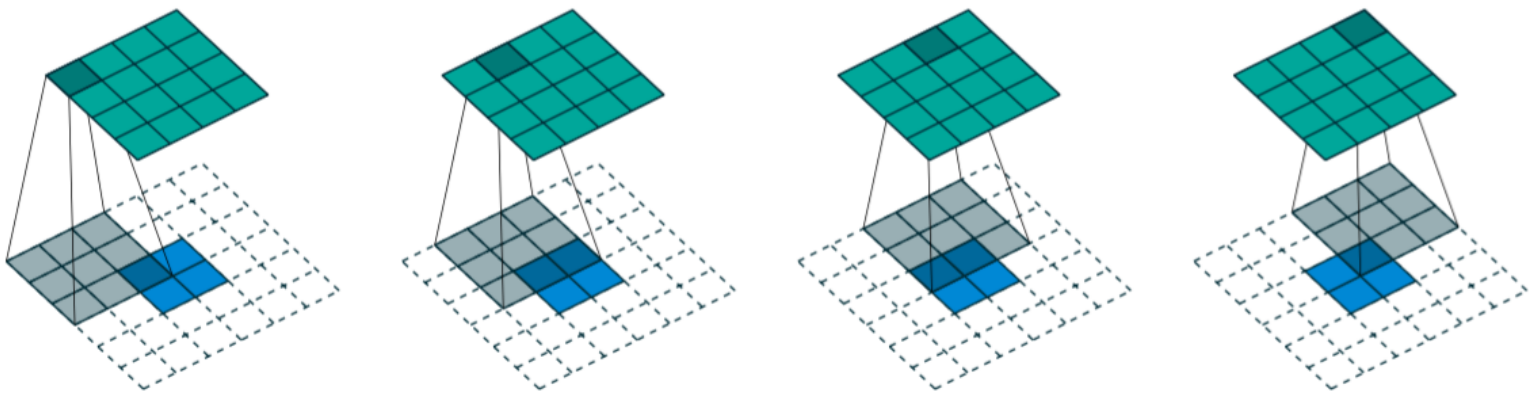


Figure: Transposed convolution can be a seen as a regular convolution. Convolution (above) with $k' = 3, s' = 1, p' = 2$ re-increases dimensionality from 2×2 to 4×4 as shown in [Dumoulin et al., 2016]

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: Transposed 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:

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

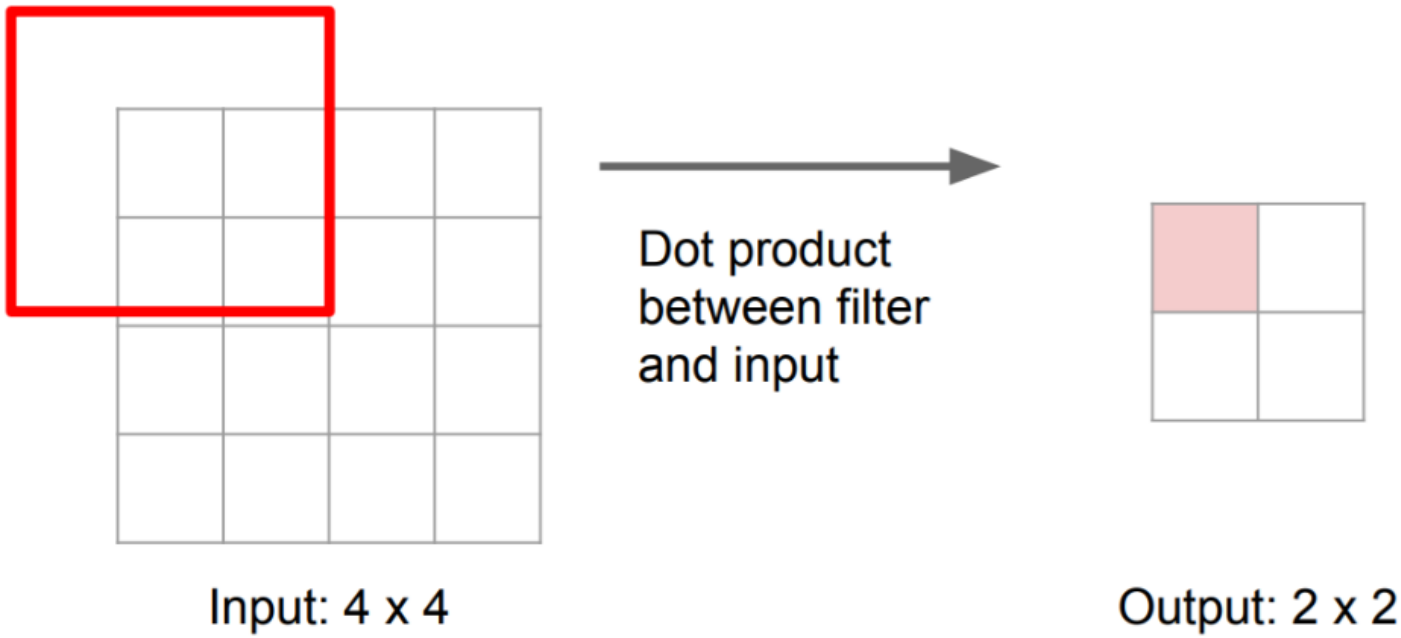
Figure: A transposed convolution can be used to upsample the feature vector of length 4 back to a feature vector of length 6.

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.

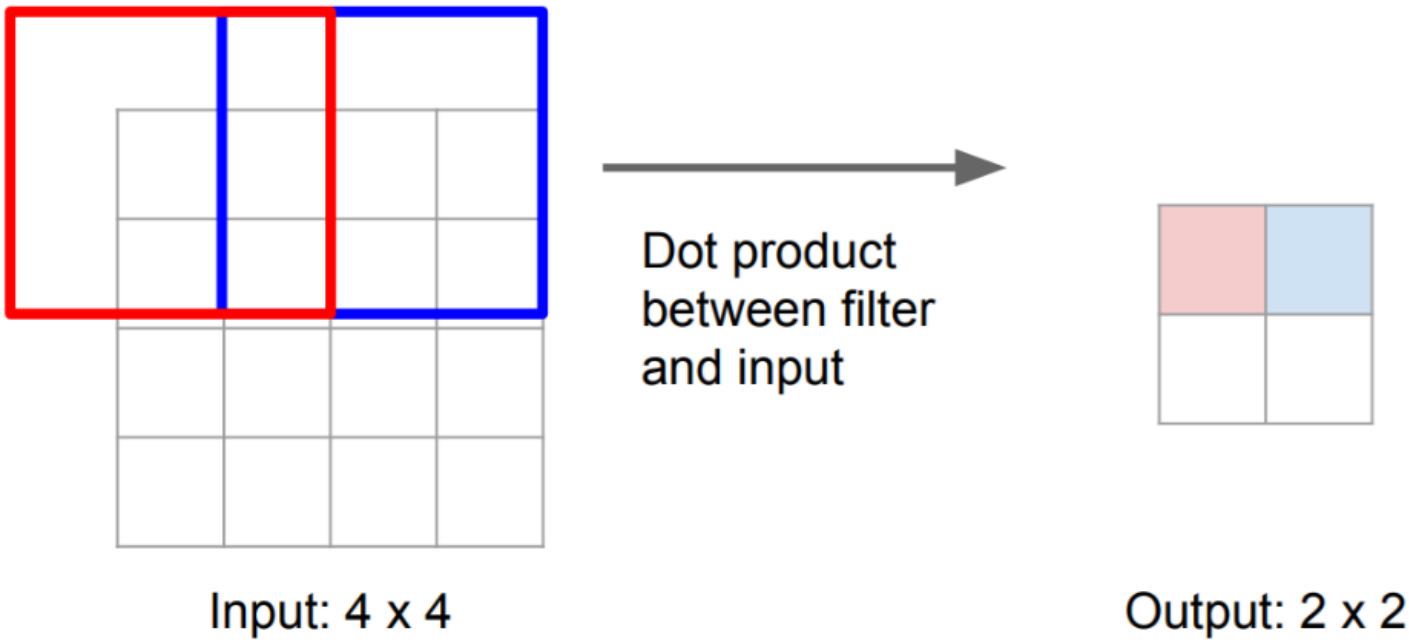


credit: Stanford University

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

TRANSPOSED CONVOLUTIONS

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

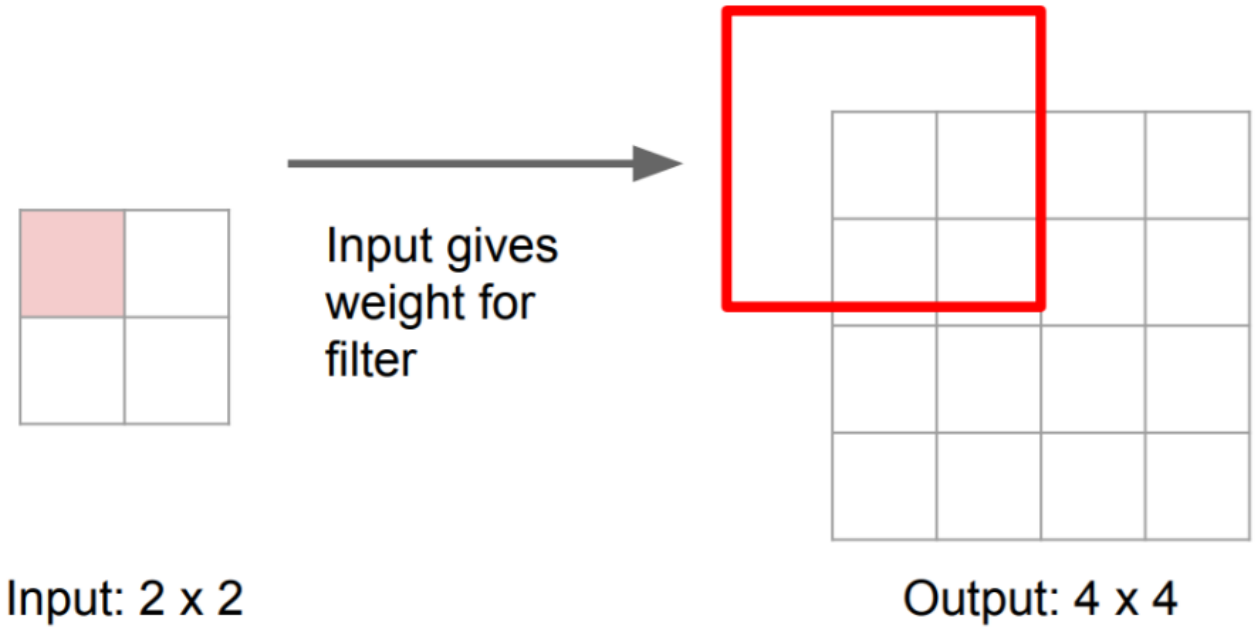


credit: Stanford University

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

TRANSPOSED CONVOLUTIONS

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



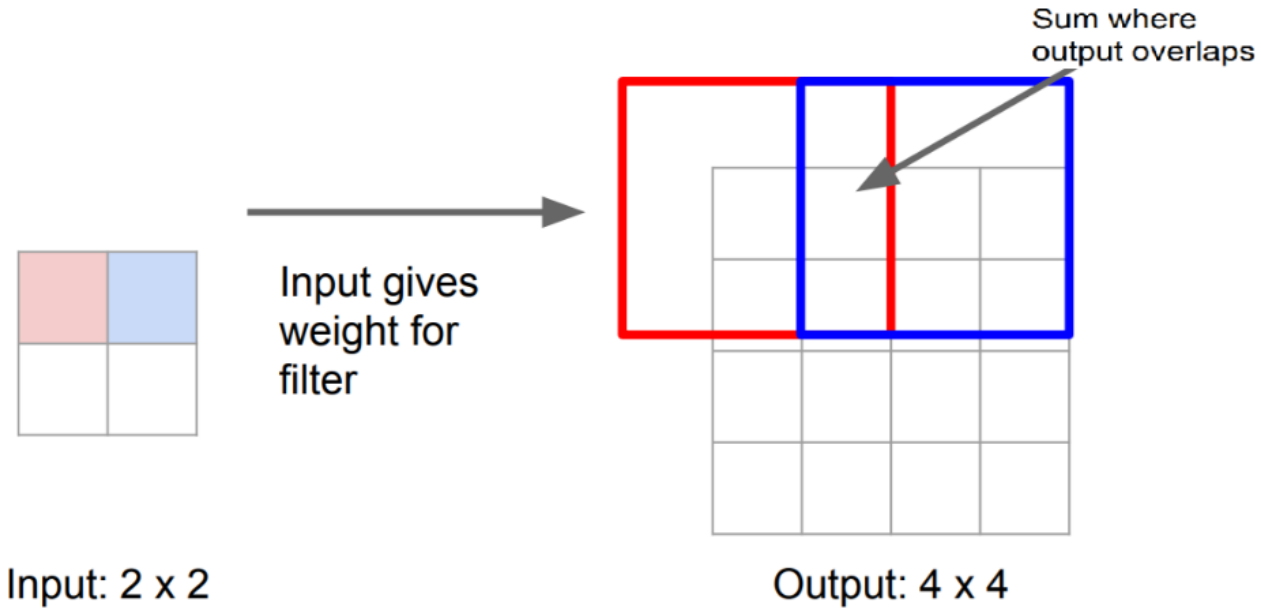
credit: Stanford University

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 [Odena et. al , 2017].

- 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.

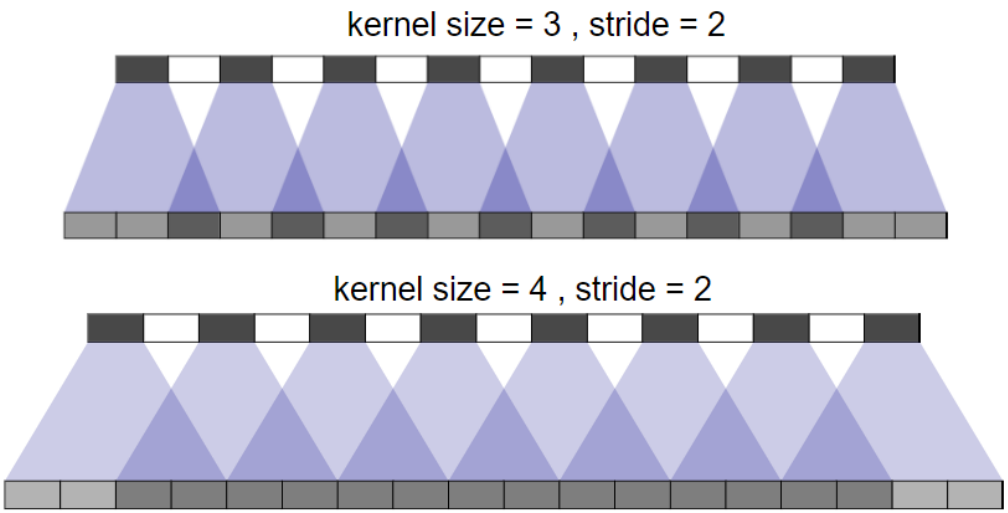


Figure: 1D example. In both images, top row = input and bottom row = output. *Top*: Here, kernel weights overlap unevenly which results in a checkerboard pattern. *Bottom*: There is no checkerboard pattern as the kernel size is divisible by the stride.

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.

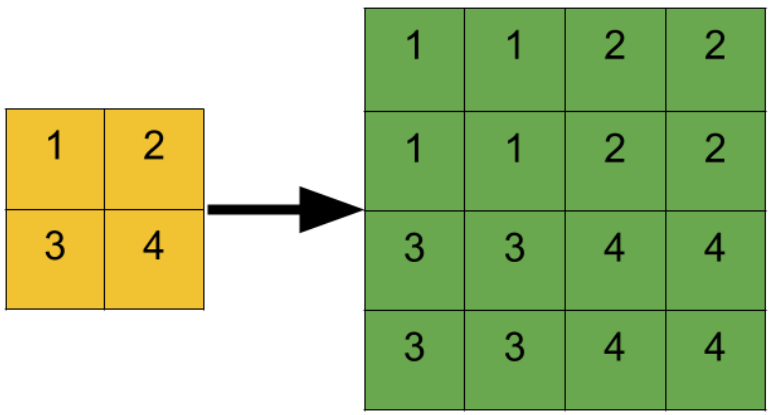


Figure: Nearest neighbor upsampling and subsequent same convolution to avoid checkerboard patterns.

Separable Convolutions

SEPARABLE CONVOLUTIONS

- Separable Convolutions are used in some neural net architectures, such as the MobileNet.
- Motivation: make convolution computationally more efficient.
- One can perform:
  - spatially separable convolution
  - depthwise separable convolution.

spatially separable convolution: The spatially separable convolution operates on the 2D spatial dimensions of images, i.e. height and width. Conceptually, spatially separable convolution decomposes a convolution into two separate operations.

- Consider the sobel kernel from the previous lecture:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

SEPARABLE CONVOLUTIONS

- this 3x3 dimensional kernel can be replaced by the outer product of two 3x1 and 1x3 dimensional kernels:

$$\begin{bmatrix} +1 \\ +2 \\ +1 \end{bmatrix} * \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

- Convolving with both filters subsequently has a similar effect, reduces the amount of parameters to be stored and thus improves speed:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Figure: In convolution, the 3x3 kernel directly convolves with the image. In spatially separable convolution, the 3x1 kernel first convolves with the image. Then the 1x3 kernel is applied. This would require 6 instead of 9 parameters while doing the same operations.

SPATIALLY SEPARABLE CONVOLUTION

Example 1: A convolution on a 5 $\times$ 5 image with a 3 $\times$ 3 kernel (stride=1, padding=0) requires scanning the kernel at 3 positions horizontally and 3 vertically. That is 9 positions in total, indicated as the dots in the image below. At each position, 9 element-wise multiplications are applied. Overall, that is 9 x 9 = 81 multiplications.

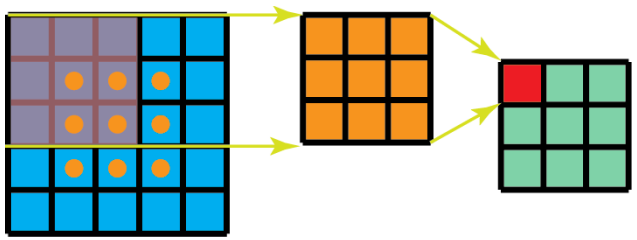


Figure: Standard convolution with 1 channel.

SPATIALLY SEPARABLE CONVOLUTION

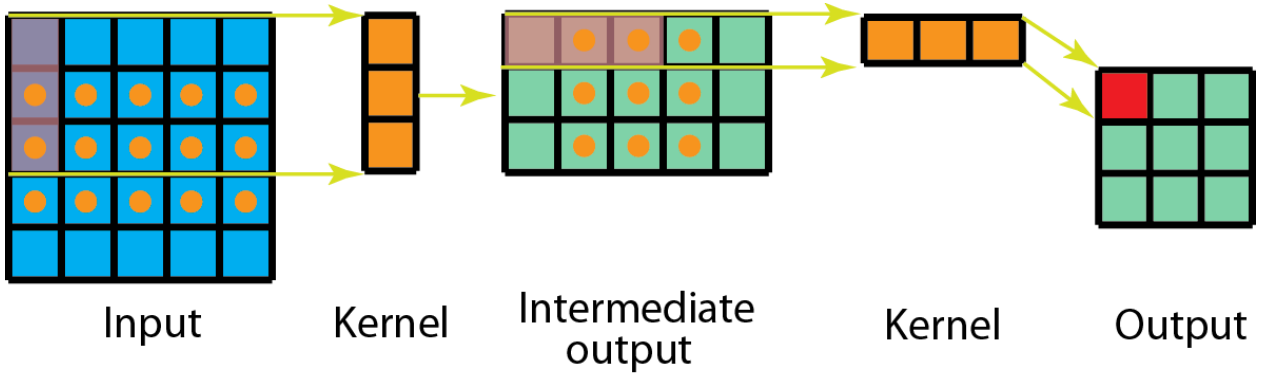


Figure: Spatially separable convolution with 1 channel. Overall, the spatially separable convolution takes 45 + 27 = 72 multiplications. (Image source: Bai (2019))

Note: However, despite their advantages, spatial separable convolutions are seldom applied in deep learning. This is mainly due to not all kernels being able to get divided into two smaller ones. Replacing all standard convolutions by spatial separable would also introduce a limit in searching for all possible kernels in the training process, implying worse training results.

DEPTHWISE SEPARABLE CONVOLUTION

- The depthwise separable convolutions, which is much more commonly used in deep learning (e.g. in MobileNet and Xception).
- This convolution separates convolutional process into two stages of depthwise and pointwise.

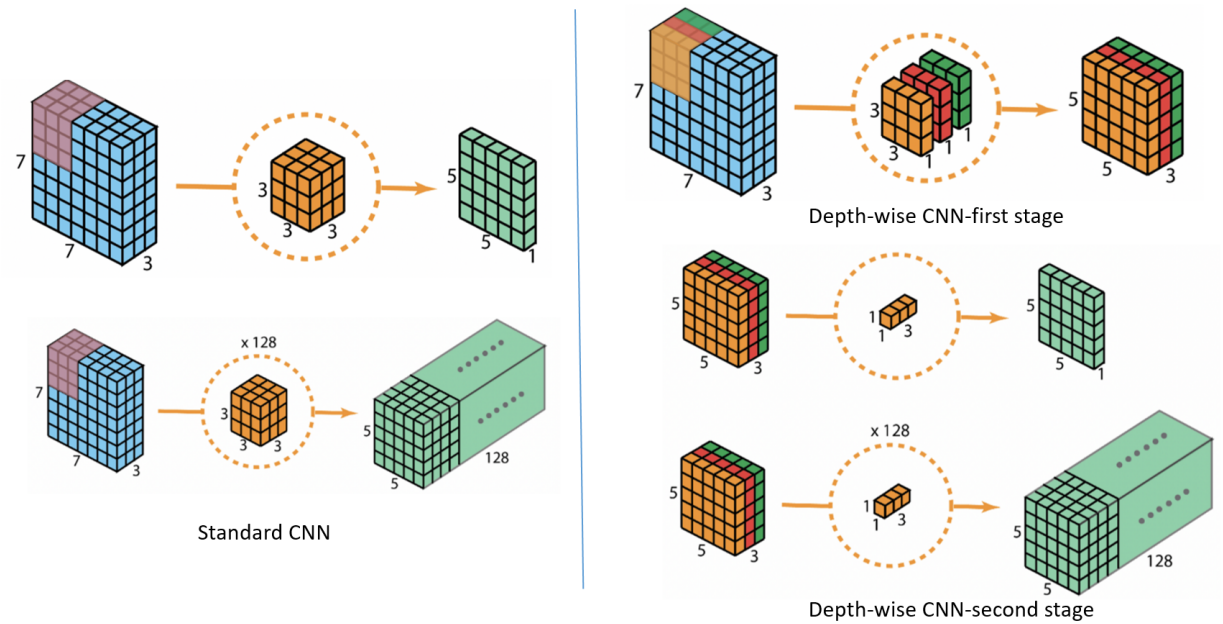


Figure: Comparision between standard cnn and separable depthwise cnn

DEPTHWISE SEPARABLE CONVOLUTION

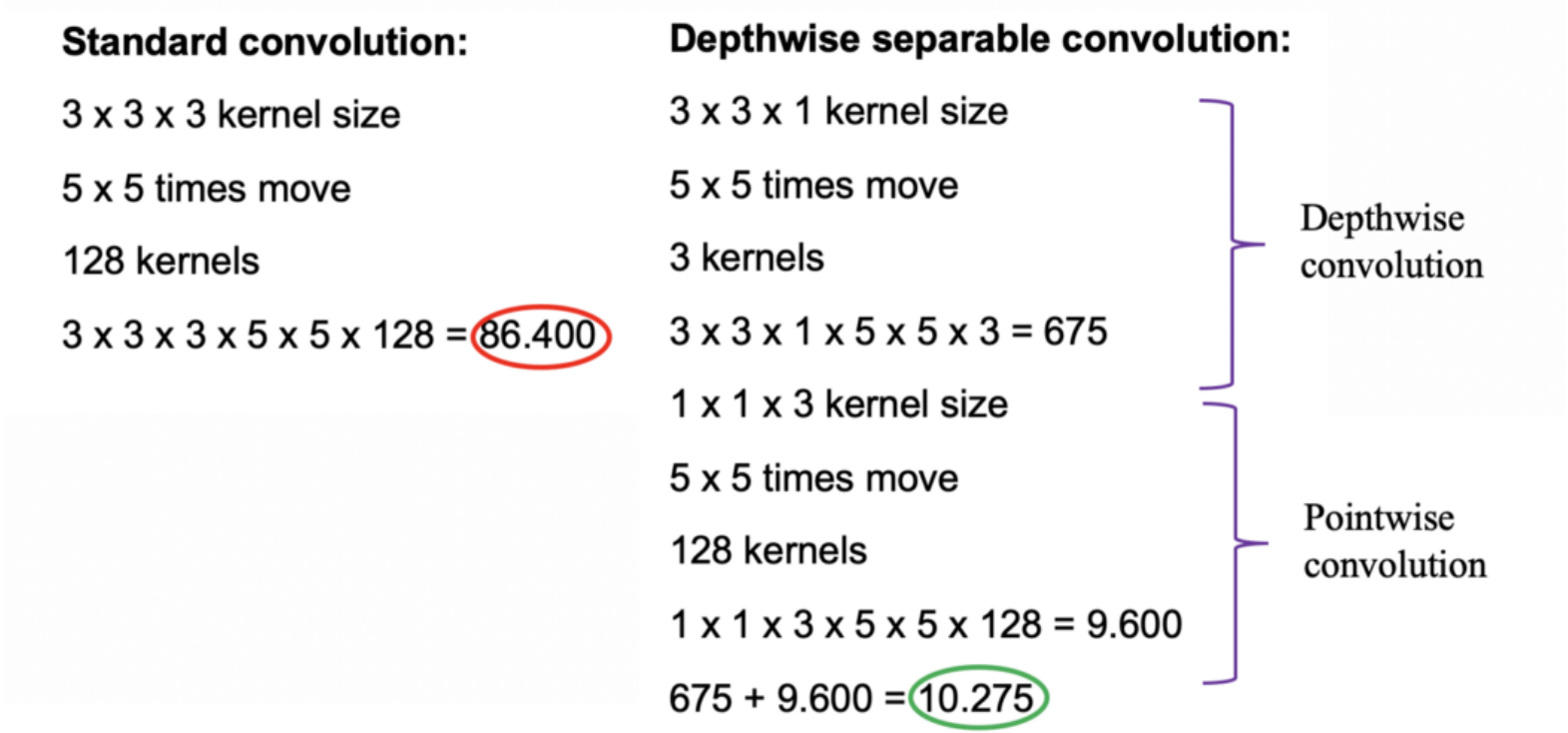


Figure: Comparision of number of multiplications in Depthwise separable cnn and standard cnn

Therefore, fewer computations leads faster network.

DEPTHWISE SEPARABLE CONVOLUTION

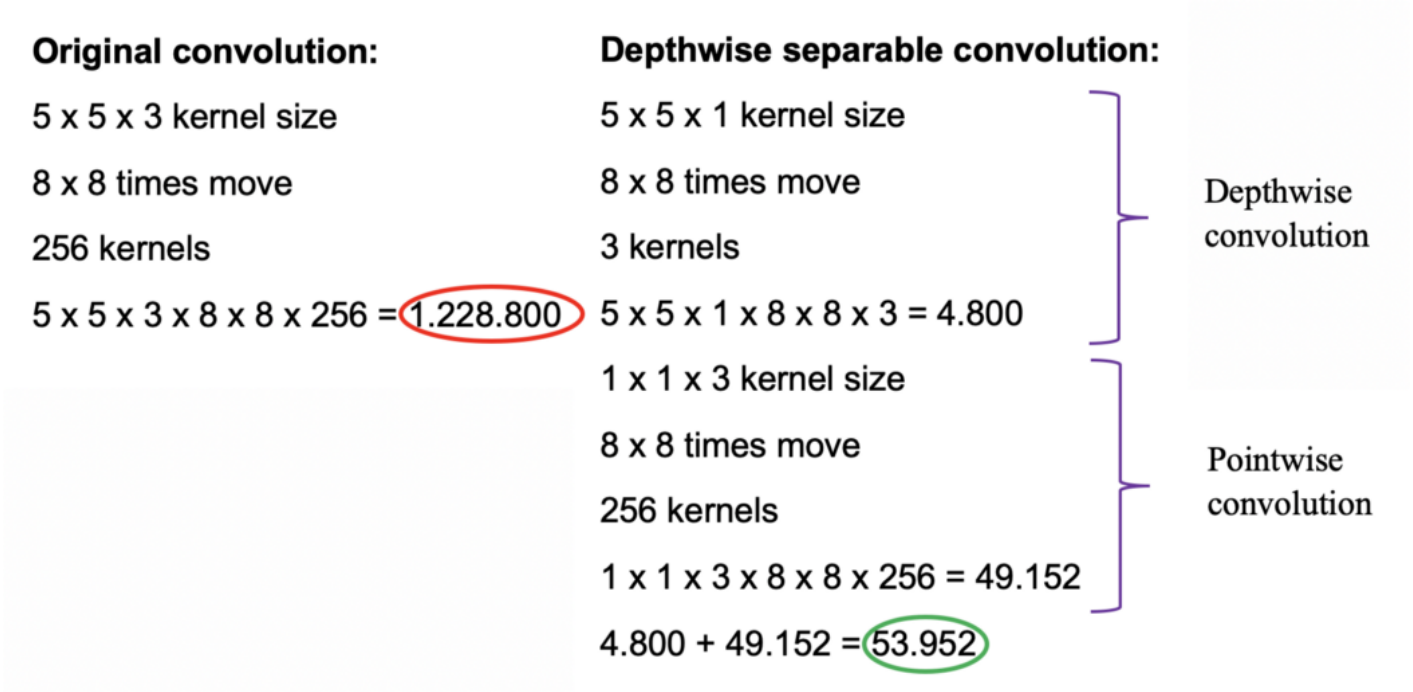


Figure: Comparision of number of multiplications in Depthwise separable cnn and standard cnn

DEPTHWISE CONVOLUTION

As the name suggests, we perform kernel on depth of the input volume (on the input channels). The steps followed in this convolution are:

- Take number of kernels equal to the number of input channels, each kernel having depth 1. Example, if we have a kernel of size 3ÃŮ3 and an input of size 6ÃŮ6 with 16 channels, then there will be 16 3ÃŮ3 kernels.
- Every channel thus has 1 kernel associated with it. This kernel is convolved over the associated channel separately resulting in 16 feature maps.
- Stack all these feature maps to get the output volume with 4ÃŮ4 output size and 16 channels.

POINTWISE CONVOLUTION

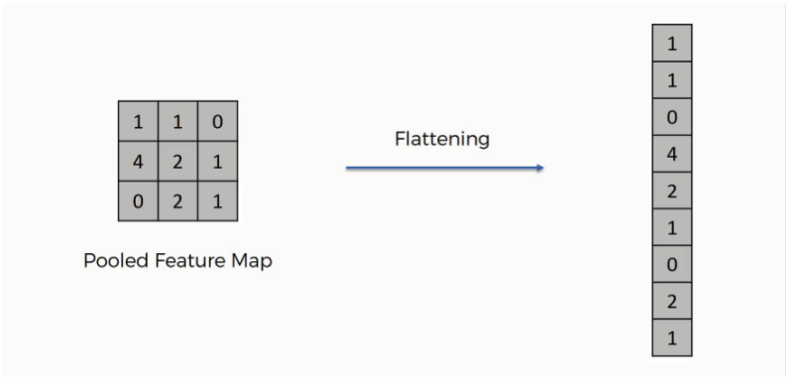
As the name suggests, this type of convolution is applied to every single point in the convolution separately (remember 1ÃŮ1 convs?). So how does this work?

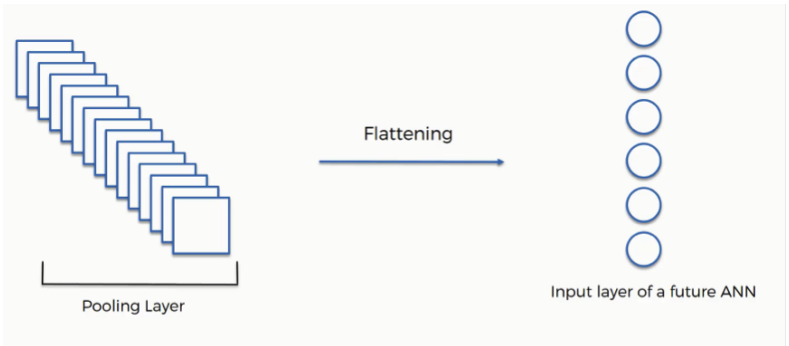
- Take a 1ÃŮ1 conv with number of filters equal to number of channels you want as output.
- Perform basic convolution applied in 1ÃŮ1 conv to the output of the Depth-wise convolution.

Flattened Convolutions

FLATTENED CONVOLUTION

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.





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