VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Deep Learning

Convolution Neural Network Model

Jan Platoš, Radek Svoboda

May 6, 2024

Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB - Technical University of Ostrava

Convolution Neural Network Model

Convolution Neural Network Model

A convolution is defined as the integral of the product of the two functions after one is reversed and shifted. It is a mathematical way how to analyze behavior of the functions and the relation between the functions.

Convolution Neural Network Model

A convolution is defined as the integral of the product of the two functions after one is reversed and shifted. It is a mathematical way how to analyze behavior of the functions and the relation between the functions.

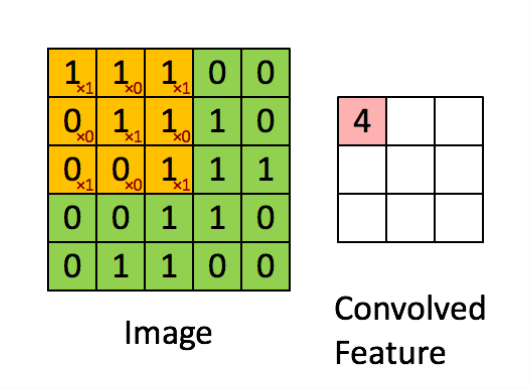
In image processing, *kernel* or *convolution matrix* or *mask* is a small matrix. In general the convolution in image processing is defined as:

$$g(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t)$$

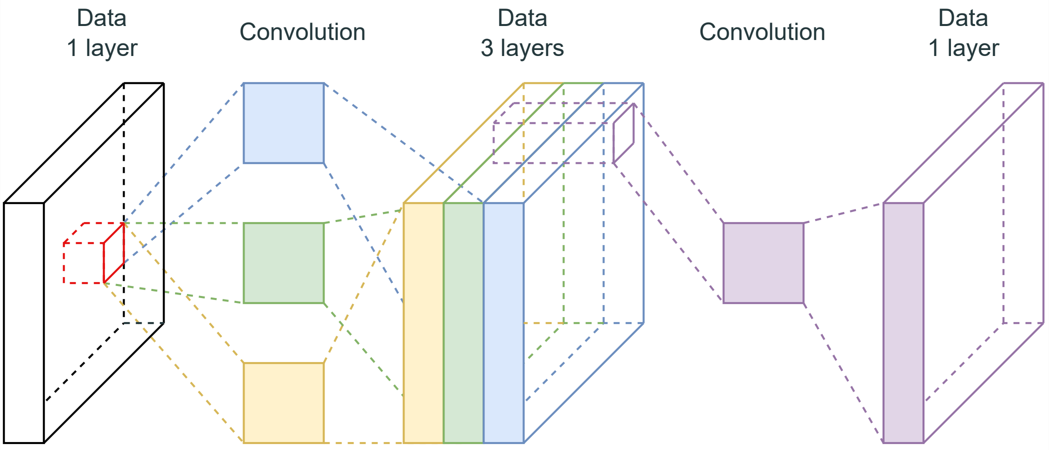
where $g(x,y)$ is filtered image, $f(x,y)$ is original image, ω if the filter kernel.

Convolution Neural Network

A kernel (also called a filter) is a smaller-sized matrix in comparison to the dimensions of the input image, that consists of real valued entries.



Convolution Neural Network



Sample Convolution Kernels

Identity

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Sobel vertical edge detection

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

Sobel horizontal edge detection

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

Edge detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Uniform blur

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian blur 3x3

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Basic properties

Size of the kernel defines the dimensions of the kernels.

Number of input channels reflects the number of channels of the image (grayscale, RGB, etc.)

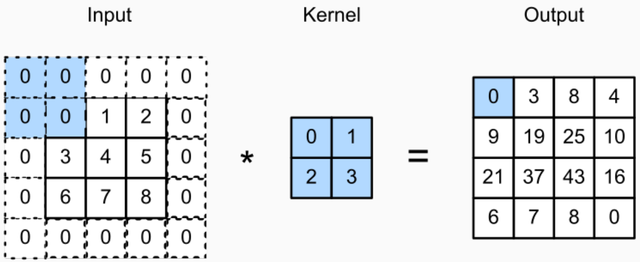
Number of output channels defines the number of kernels applied on the image, and, therefore, the output of the layer.

Stride is the size of the step that kernel is moved on the image.

Padding is system the kernel is placed on the image.

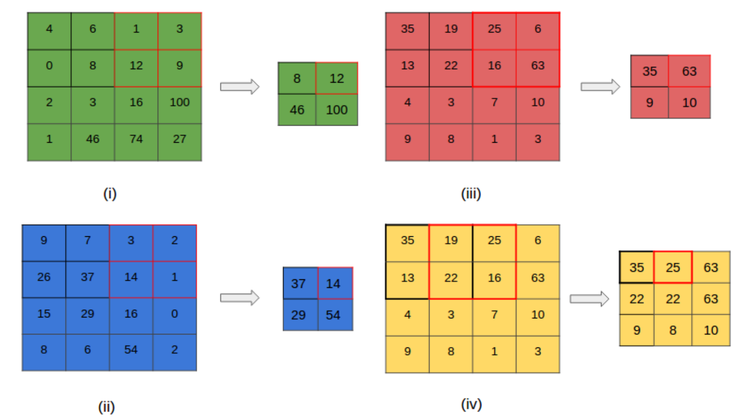
Padding

One tricky issue when applying convolution is losing pixels on the edges of our image. A straightforward solution to this problem is to add extra pixels around the boundary of our input image, which increases the effective size of the image.

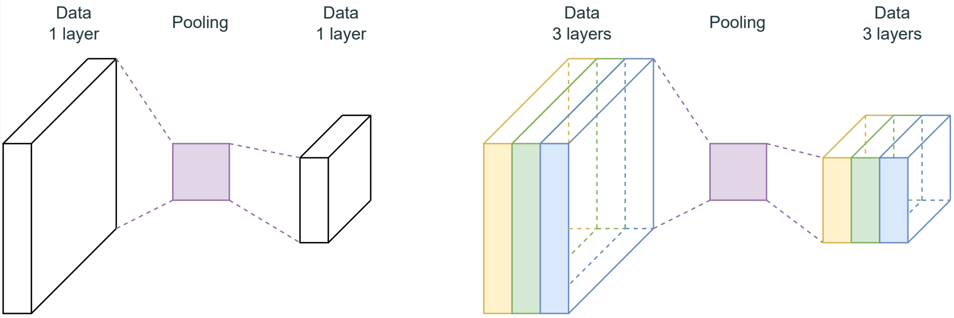


Pooling

Pooling is a way how to decrease the amount of information transfered from one layer to another. The standard way ho to do it is *Average Pooling* and *Maximum Pooling*.



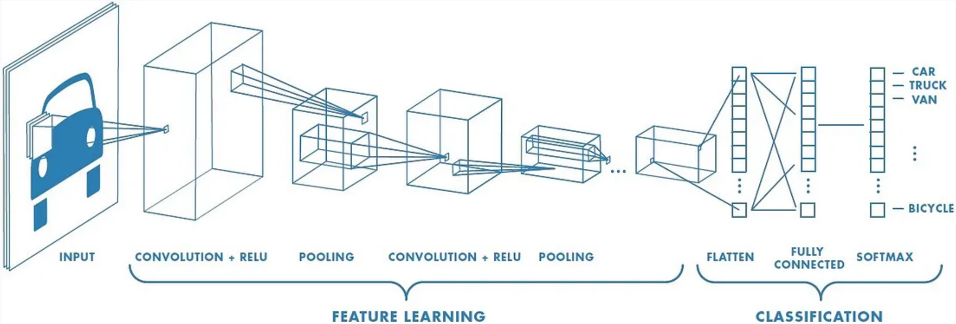
Pooling


Data
1 layer
Pooling
Data
1 layer
Data
3 layers
Pooling
Data
3 layers

Main advantage of the CNN

Weights sharing

Basic architecture of the CNN



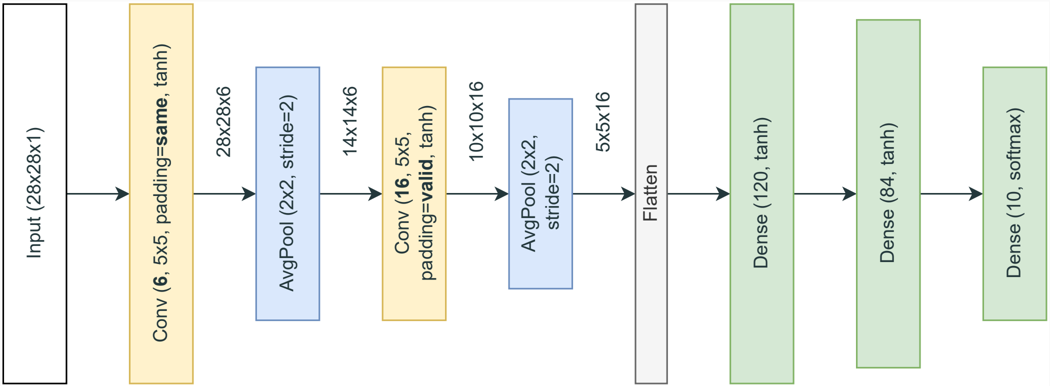
Most important CNN Architectures

- **1998** LeNet-5 [1]: One of the first CNN architectures designed for handwritten digit recognition.
- **2012** AlexNet [2] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with a significant margin. It has eight layers, including five convolutional layers and three fully connected layers.
- **2014** VGG [3] (Visual Geometry Group) Network has a deeper architectures than AlexNet, with up to 19 layers. It uses a smaller kernel size (3x3) and the same padding for all layers.
- **2014** GoogLeNet [4] (Inception Network) has a unique architecture of using multiple Inception modules, which allow it to use both deep and wide networks while keeping the computation cost low.

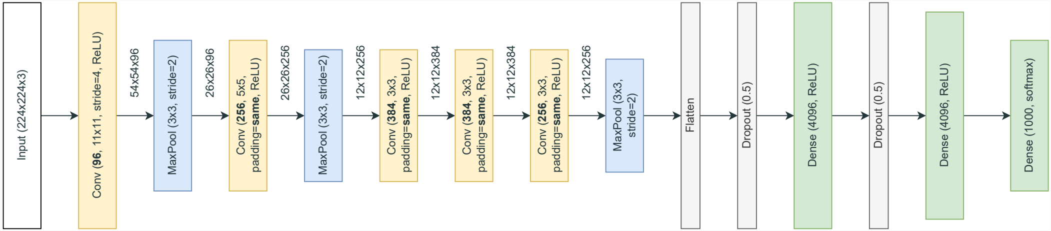
Most important CNN Architectures

- **2015** ResNet [5] (Residual Network) uses a shortcut connection between the input and output of a layer, allowing the gradient signal to propagate more easily through deep networks.
- **2016** DenseNet [6] (Dense Convolutional Network) connects all layers to each other in a dense block and reuses features from all previous layers, making it more efficient in parameter usage.
- **2017** MobileNet [7] is designed to run efficiently on mobile and embedded devices by using depthwise separable convolutions, which separate the spatial and channel-wise convolutions.
- **2019** EfficientNet [8] uses a compound scaling method to scale up all dimensions of the CNN architecture (depth, width, resolution) in a balanced way, leading to better performance and efficiency.

LeNet-5

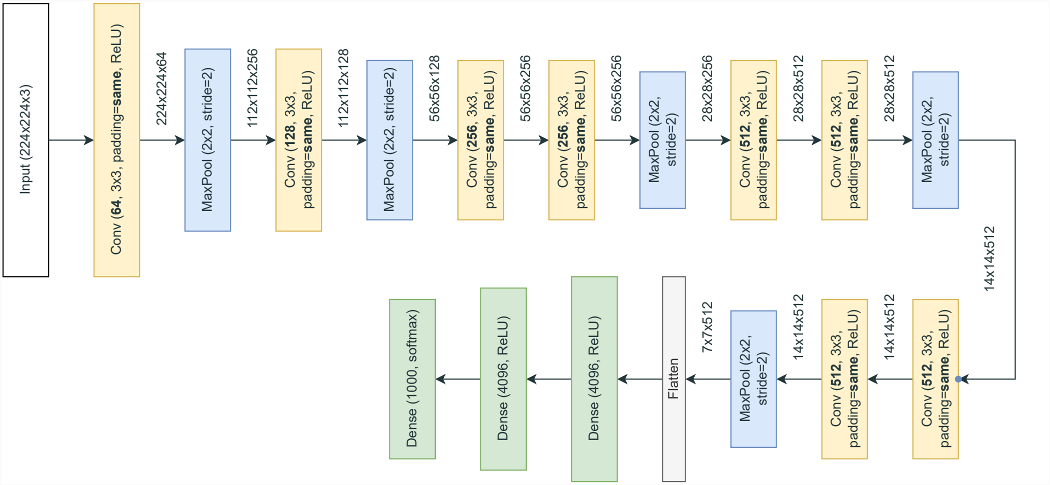

Input (28x28x1)
Conv (**6**, 5x5, padding=**same**, tanh)
28x28x6
AvgPool (2x2, stride=2)
14x14x6
Conv (**16**, 5x5, padding=**valid**, tanh)
10x10x16
AvgPool (2x2, stride=2)
5x5x16
Flatten
Dense (120, tanh)
Dense (84, tanh)
Dense (10, softmax)

AlexNet

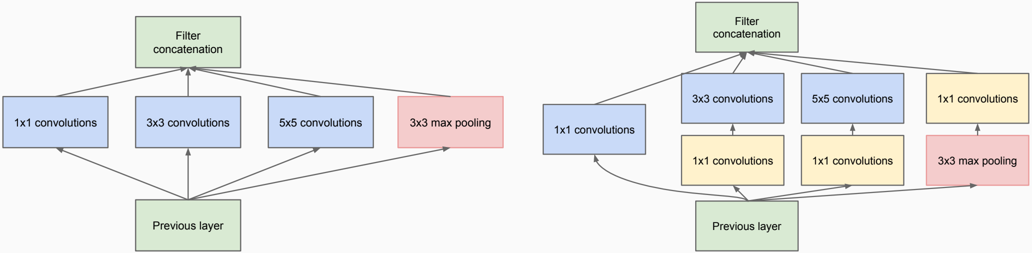


Input (224x224x3)
Conv (96, 11x11, stride=4, ReLU)
54x54x96
MaxPool (3x3, stride=2)
26x26x96
Conv (256, 5x5, padding=same, ReLU)
26x26x256
MaxPool (3x3, stride=2)
12x12x256
Conv (384, 3x3, padding=same, ReLU)
12x12x384
Conv (384, 3x3, padding=same, ReLU)
12x12x384
Conv (256, 3x3, padding=same, ReLU)
12x12x256
MaxPool (3x3, stride=2)
Flatten
Dropout (0.5)
Dense (4096, ReLU)
Dropout (0.5)
Dense (4096, ReLU)
Dense (1000, softmax)

VGG-A


Input (224x224x3)
Conv (64, 3x3, padding=same, ReLU)
224x224x64
MaxPool (2x2, stride=2)
112x112x256
Conv (128, 3x3, padding=same, ReLU)
112x112x128
MaxPool (2x2, stride=2)
56x56x128
Conv (256, 3x3, padding=same, ReLU)
56x56x256
Conv (256, 3x3, padding=same, ReLU)
56x56x256
MaxPool (2x2, stride=2)
28x28x256
Conv (512, 3x3, padding=same, ReLU)
28x28x512
Conv (512, 3x3, padding=same, ReLU)
28x28x512
MaxPool (2x2, stride=2)
14x14x512
Conv (512, 3x3, padding=same, ReLU)
14x14x512
Conv (512, 3x3, padding=same, ReLU)
14x14x512
MaxPool (2x2, stride=2)
7x7x512
Flatten
Dense (4096, ReLU)
Dense (4096, ReLU)
Dense (1000, softmax)

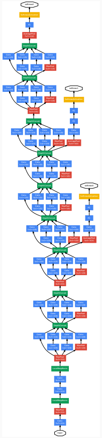
GoogLeNet - Inception network



(a) Inception module, naïve version (b) Inception module with dimension reductions

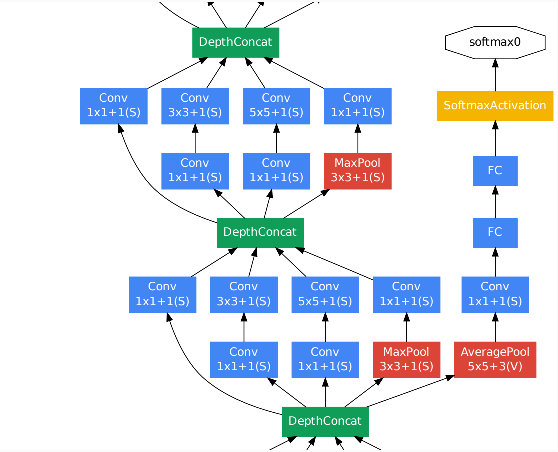
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2014). Going Deeper with Convolutions. arXiv. https://doi.org/10.48550/ARXIV.1409.4842

GoogLeNet - Inception network



Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2014). Going Deeper with Convolutions. arXiv. https://doi.org/10.48550/ARXIV.1409.4842

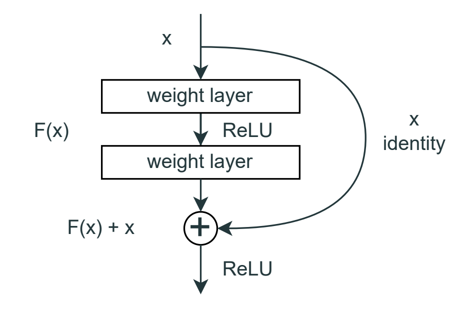
GoogLeNet - Inception network



Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2014). Going Deeper with Convolutions. arXiv. https://doi.org/10.48550/ARXIV.1409.4842

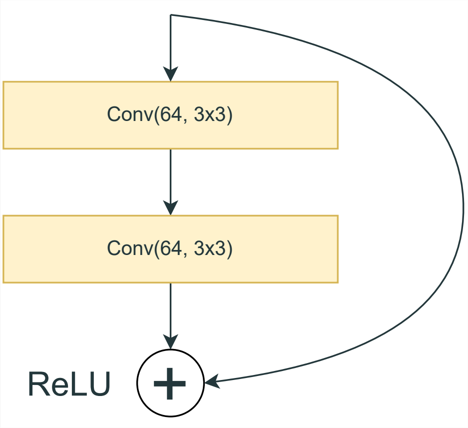
ResNet

- Residual block reformulates the problem of learning a mapping $H(x)$ by learning the residual mapping $F(x) = H(x) - x$.
- The original function is therefore $F(x) + x$.
- This (counter-intuitively) leads into faster learning and allows more deeper topology.



ResNet

- ResNet introduces *skip connections* that solves the problem of the vanishing gradient.
- ResNet stacks multiple identity mappings (convolutional layers that do nothing at first), skips those layers, and reuses the activations of the previous layer.
- Skipping speeds up initial training by compressing the network into fewer layers.



DenseNet

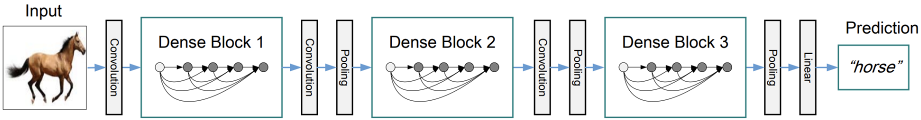


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

1. 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.
2. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Communications of the ACM 60, no. 6 (2017): 84-90.
3. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

References ii

4. 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.
5. 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, pp. 770-778. 2016.
6. Huang, G., Z. Liu, L. Maaten, and K. Weinberger. ”Densely Connected Convolutional Networks. Computer Vision and Pattern Recognition.” arXiv preprint arXiv:1608.06993 (2016).

7. Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. ”Mobilenets: Efficient convolutional neural networks for mobile vision applications.” arXiv preprint arXiv:1704.04861 (2017).
8. Tan, Mingxing, and Quoc Le. ”Efficientnet: Rethinking model scaling for convolutional neural networks.” In International conference on machine learning, pp. 6105-6114. PMLR, 2019.

Questions?