

# COV3 – Deep Learning in CV

---

GERALD ZWETTLER

# Topics

Overview, History and Basics

Preprocessing

Network Layers

Network Types

Application Domains

# Overview, History and Basics

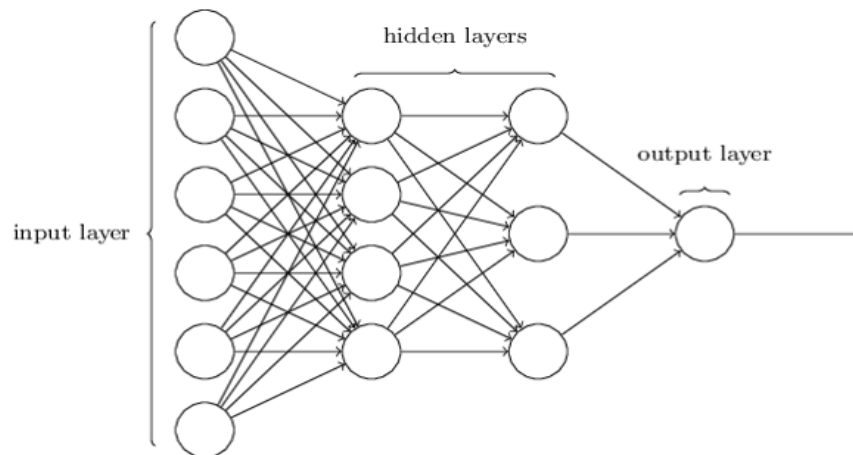
---

# Overview, History and Basics

- a child is able to reliably learn new objects (fruits, animals,...) from only **one** or at least very few reference image(s), c.f. children's book with animals.
  - thereby, the basic shape, color, texture and other object characteristics are implicitly derived and trained
- a convolutional neural network (CNN) thereby follows quite a similar learning approach, imitating the structure of human cerebral cortex.
  - although CNNs follow the concept of human learning and pattern recognition, the complexity is much lower, thus necessitating a much broader range of **labelled** reference images in domain-specific learning
- besides training deep learning CNN, *Boosting* as committee of weak classifiers performing a majority voting, is applicable too often outperforming common SVM or NN approaches [Hastie et al. 2001]
  - Boosting (e.g. ADA boost) also applicable in case of unlabeled input data in the sense of unsupervised learning, picking out the most plausible and reliable unknown samples to enrich the reference base

# Cerebral Cortex and Feed-Forward Networks

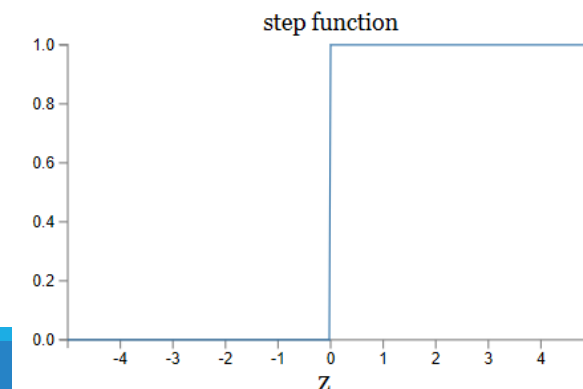
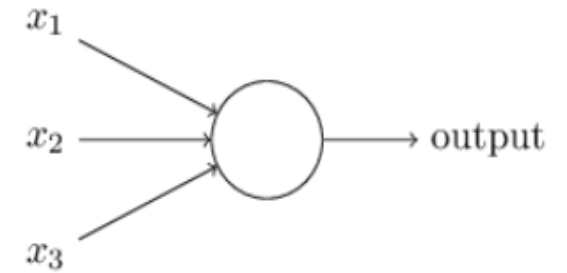
- the primary visual cortex of a human brain, also denoted as *V1*, contains more than 140 millions of neurons with tens of billions of connections between them.
- besides *V1*, human vision involves further cortices *V2*, *V3*, *V4* and *V5* progressively handling more complex image processing and recognition tasks.
  - thus, the human brain can be interpreted as super-computer perfectly adapted to the visual world and eyes as input device by evolution.
  - although the tasks and problems the human visual system has to handle every millisecond we look around to scan the environment are very, very hard, they are solved in a passive and unconscious way
    - the permanently carried out visual sensing and recognition does not lower the regular calculation power needed for active planning and thoughts.
- a common neural network (NN) is inspired by interconnected neurons.
  - *input* and *output* layer thereby adapted to the problem domain with (fully-connected) hidden layers in-between



from [Nielsen,  
<http://neuralnetworksanddeeplearning.com/chap1.html> ]

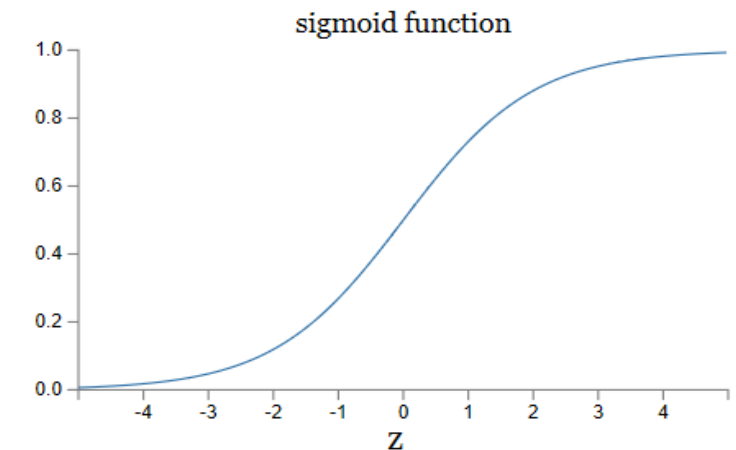
# Basic Neural Network Layout

- **Perceptron** with  $1-n$  inputs and simple arithmetic calculation of the output based on weights.
  - initially, perceptrons modelled with binary input and binary output thus allowing utilization of very trivial activation functions
  - only weights at floating point precision as the only adaptive learning paradigm of neural networks
  - several inputs for each perceptron: same as collecting various evidence before making a decision
  - [binary] activation function e.g.  $output = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq threshold \\ 1 & \text{if } \sum_j w_j x_j > threshold \end{cases}$ 
    - generally, the weighting equation is formed in a way that the threshold, then denoted as **bias**, is brought to the left side, thus only checking for a value  $> 0$ .
    - due to binary activation, output following a sharp step function, thus very sensitive to marginal value changes
  - **further layer**: more abstract, more complex decision making, while first layer uses the input data



# Basic Neural Network Layout

- Training: make small/marginal change on one weight and evaluate the corresponding change on the output quality. Small changes in perceptron might lead to huge overall changes
- thus sigmoid **neurons** introduced to overcome this instability with all inputs within [0..1] range.
  - with threshold as bias  $b$  the output function is defined as  $\sigma(w \cdot x + b)$  with sigmoid-shaped activation function  $\sigma(z) = \frac{1}{1+e^{-z}}$
- due to activation at floating point precision, marginal weight changes do not affect the remaining layers in an unpredictable way
- besides, utilization of floating-point precision implicitly introduces a confidence metric for the classification results
- [fully connected] Feed Forward Networks as basic layout type
  - applicable in many domains due to easy modelling of input and output layer
  - hidden layers as sophisticated part: determining the trade-off between training time, classification accuracy.

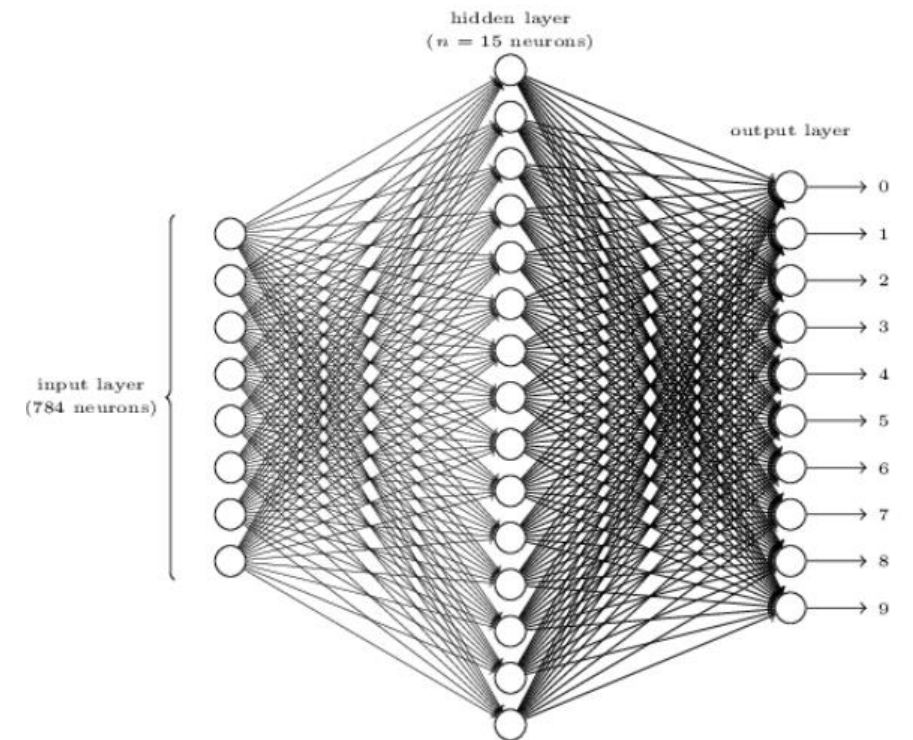


# FFNN for OCR – an example

- for some CV tasks, like OCR, a FFNN can be utilized to achieve acceptable results.
- in case of OCR to recognize digits [0-9] from single input image representing a ROI, the input layer for 28x28 image size is to be represented by 784 input neurons.
- the output layer naturally to be designed by 10 neurons representing the particular digits
  - alternatively, output could be modelled by 4 neurons only too, utilizing a binary representation with  $2^4$  code words.
  - input as single letter ROI



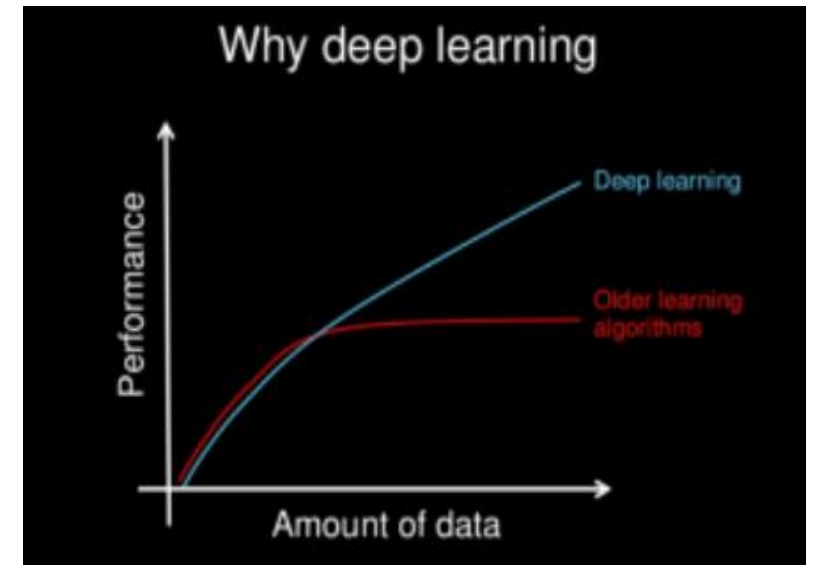
- OCR solvable with simple FFNN.
  - but how to classify faces ? pixelwise like OCR? → demand for further hidden layers and more sophisticated network layouts → CNN / Deep Learning
  - subtasks: left/right eye present, mouth in the middle,...





# Recurrent and CNN Network Layout

- Feed-Forward as basic layout design pattern (without loops).
- In case of loops: recurrent neural networks:
  - not so many training algorithms available but RNN can solve some specific domains FF would fail.
  - RNN inspired by brain.
- In the 80's: single hidden layer due to performance. More training data did rather lead to overfitting compared to better results. For deep learning, results get better with increased input data
- nowadays 5-10 hidden layers possible
- CV is an excellent domain for deep learning. In case on image classification CNNs are not only superior compared to any other machine learning strategy but are starting to be better than humans



# Frameworks cont'd

list of relevant frameworks

## ■ TensorFlow

- open source framework for numerical computations utilizing a graph-based representation. Nodes in the graph thereby represent mathematical equations and the connections data flow of n-dimensional tensors

## ■ Caffe

- framework allowing access to deep learning algorithms. Can be used on mobile devices and the Cloud.

## ■ ConvNetJS

- Deep Learning model entirely run in the browser. Thus, no installation/compilation or GPU issues to handle

## ■ YOLO (you only look once)

- pre-trained CNN system for object classification from input images

# Network Types

---

# Network Types

on a meta perspective, the following key network types exist for artificial neural networks

## ■ Feed Forward

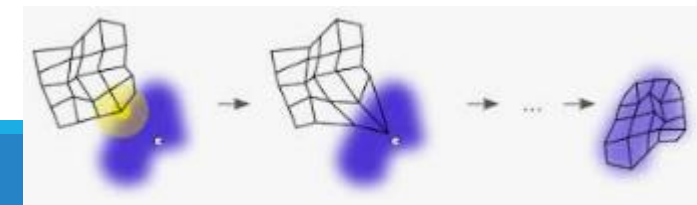
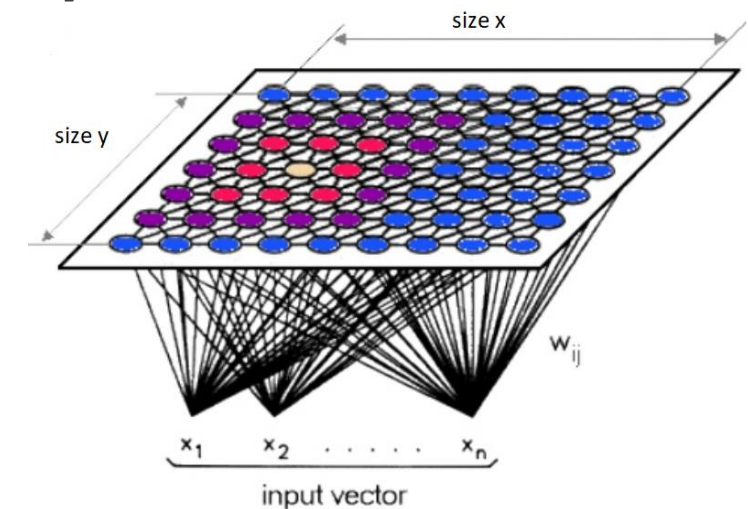
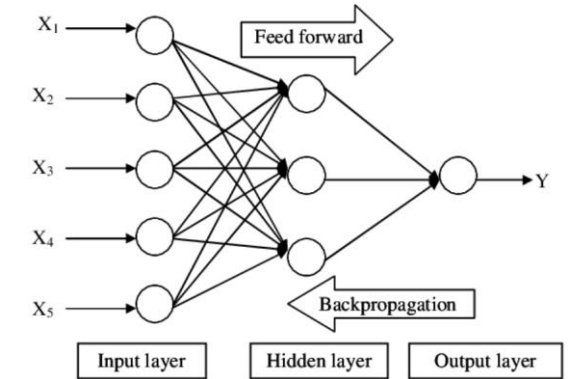
- simplest type, as data only is passed in one direction
- may have several hidden layers
- applicable in many computer vision and speech recognition domains
- can be utilized for multi-modal image fusion too, c.f. [Zhang and Wang 2011]

## ■ Kohonen **Self Organizing Neural Network** (Maps) [Kohonen 1995]

- a rigid raster of neurons is implicitly deformed in an elastic way due to input data
- allows for clustering in complex domains
- classify renal (kidneys) diseases of patients, c.f. [Van Biesen et al. 1998]

## ■ Modular Neural Networks

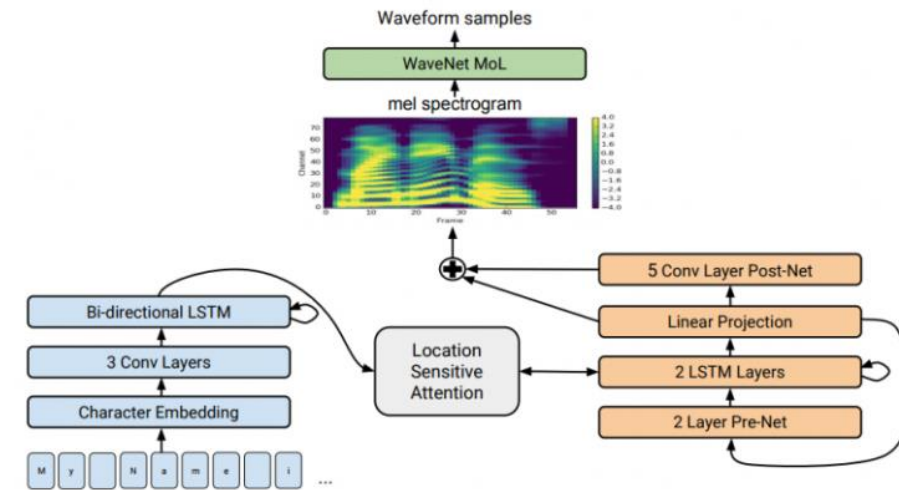
- hybrid approaches, mainly Convolutional with LSTM
- several network types combined in a larger net



# Network Types

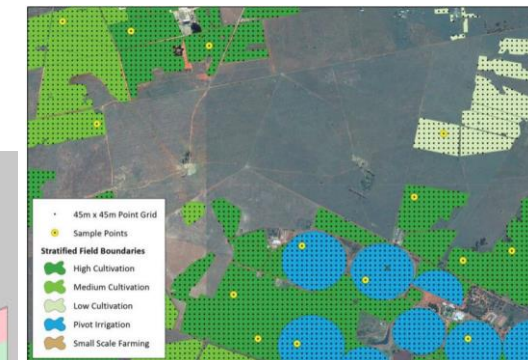
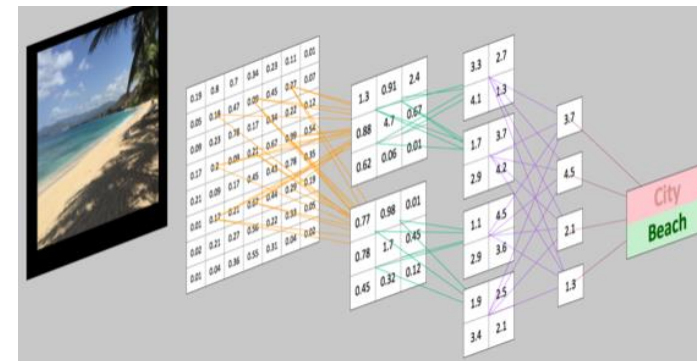
## ■ Recurrent Neural Network, LSTM

- output of a layer is stored to be incorporated for calculation of consecutive layers again
- helps stabilizing the prediction of the output layer
- contextual and semantic value of input signal is incorporated, thus. highly applicable in the domain of speech recognition, natural language processing or OCR.
- applicable for text-to-speech synthesis too, c.f. *DeepVoice* [Arik et al. 2017]



## ■ Convolutional Neural Networks

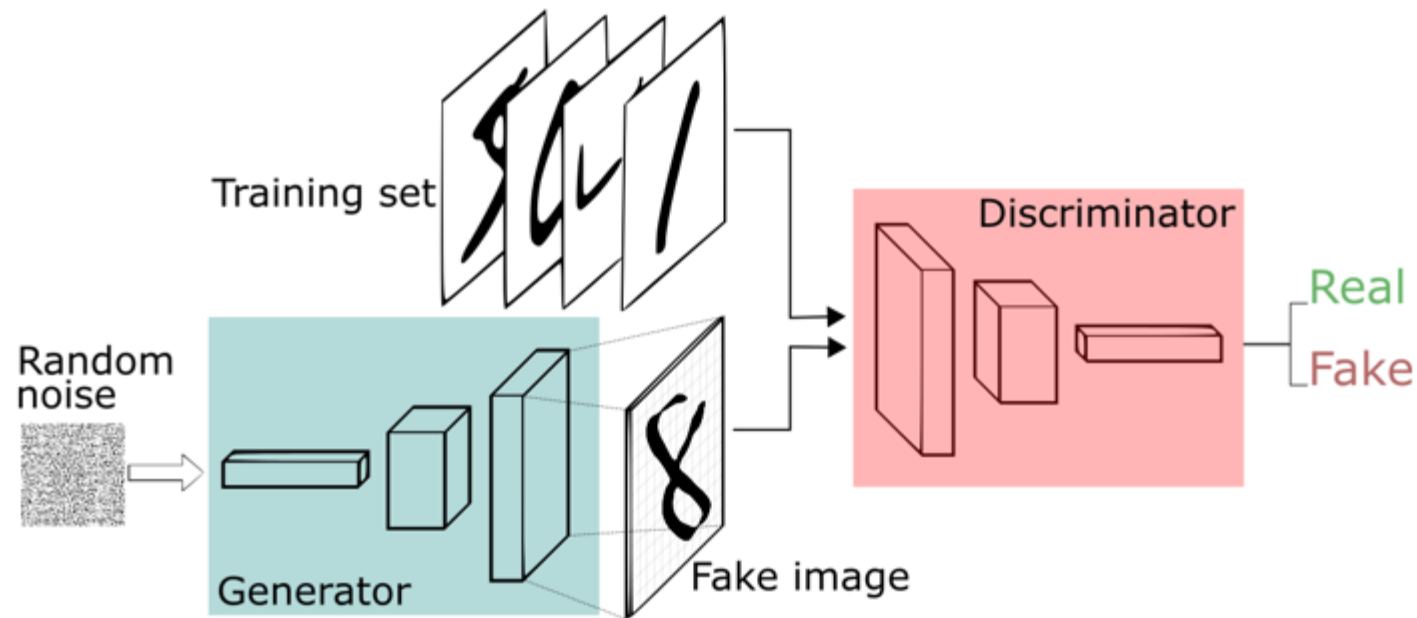
- input data processed in batches similar to common convolution filtering
- applicable for hard CV tasks like image classification, segmentation or recognition
- features implicitly present in the convolution layers, c.f. Haar Cascade, DoG pyramid
- images processed at decreasing size (pyramid)
  - for classification: on the output layer only class labels present (pyramid)
  - for segmentation: after reducing size, the image data is scaled up for final resolution (2 pyramids touching at the top)
- noteworthy CNN architectures: *LeNet*, *AlexNet*, *GoogLeNet*, *ResNet* (>1,200 layers)



satellite image classification

# Network Types

- **GAN** – generative adversarial network [Goodfellow et al. 2014]
  - perfect for generating data, e.g. artificial music, artificial paintings looking similar as human-made
  - GANs incorporate two types of networks, one (*Generator*) to generate a very large number of hypothesis and one (*Discriminator*) to decide if the generated content does look similar to a given training set
  - both networks continuously improving, the Generator in passing the “reality check” and the Discriminator in the ability to detect “bad” artificial content



<https://skymind.ai/wiki/generative-adversarial-network-gan>

# Network Layers

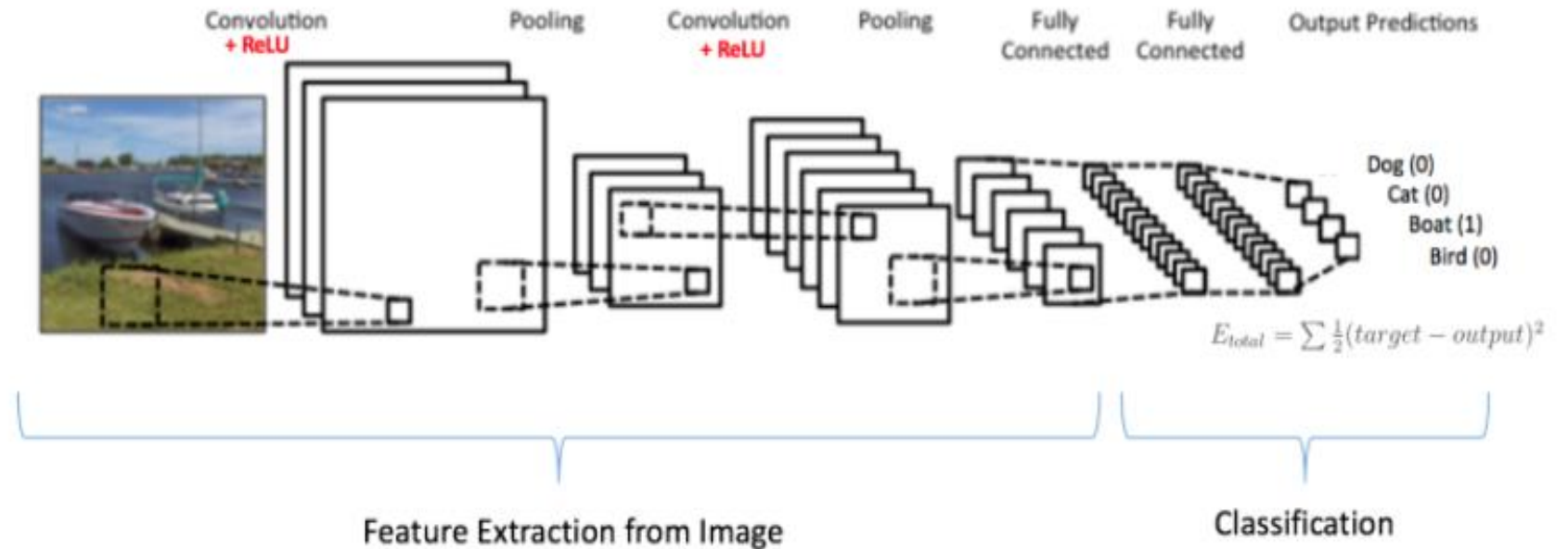
---

FOR CNN AND RNN

# Network Layers

- input images as BLOB are interpreted as multi-dimensional structure, e.g. 1x3x200x200 for RGB images of size 200x200 pixels.
- a common CNN includes various network layers
- relevant layers overview:

- **Convolution**
- **ReLU**
- **BN**
- **Pooling**
- **Reshape**
- **Concat**
- **Deconvolution**
- **LSTM**
- **Recurrent**
- **RNN**
- and many more....



[from <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>]










# Network Layers cont'd

## network layers in detail

### ■ Convolution

- input images are modified by a convolution filter (matrix)
- in course of this step, a reduction in size might be included too (down-sampling)
- output of this process often called Feature Map, as information level is higher compared to initial pixel values

Operation	Filter	Convolved Image	Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$		Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$		Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$		Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$				

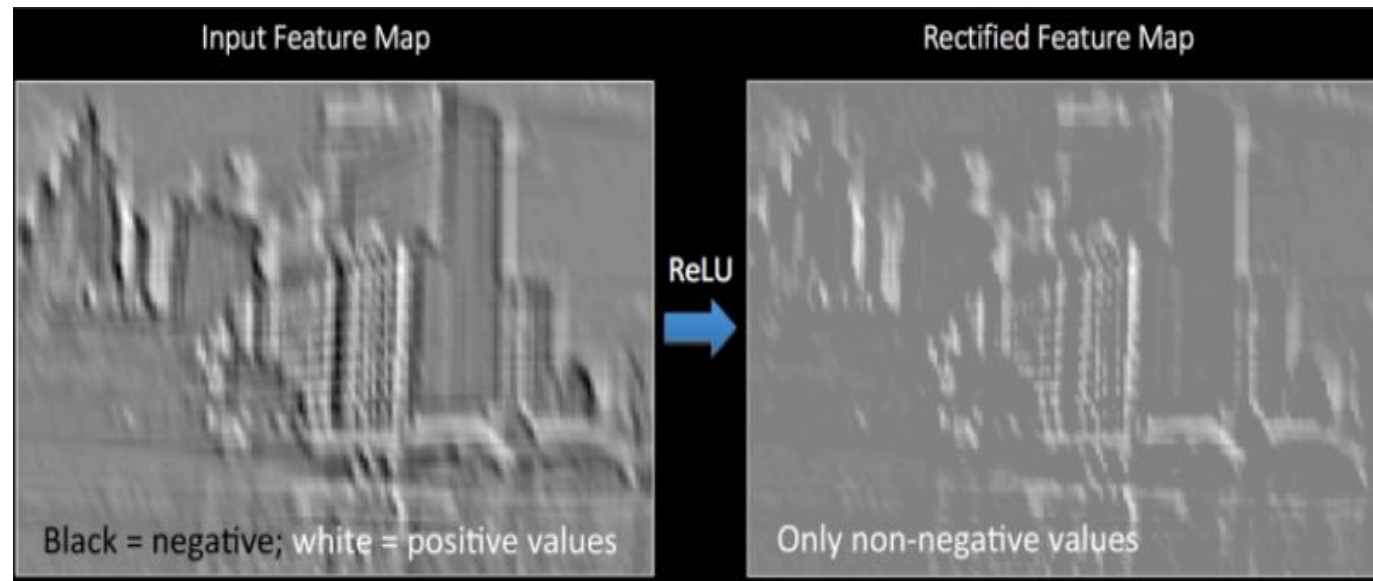


convolution masks to apply

# Network Layers cont'd

- **ReLU** (rectified linear unit)

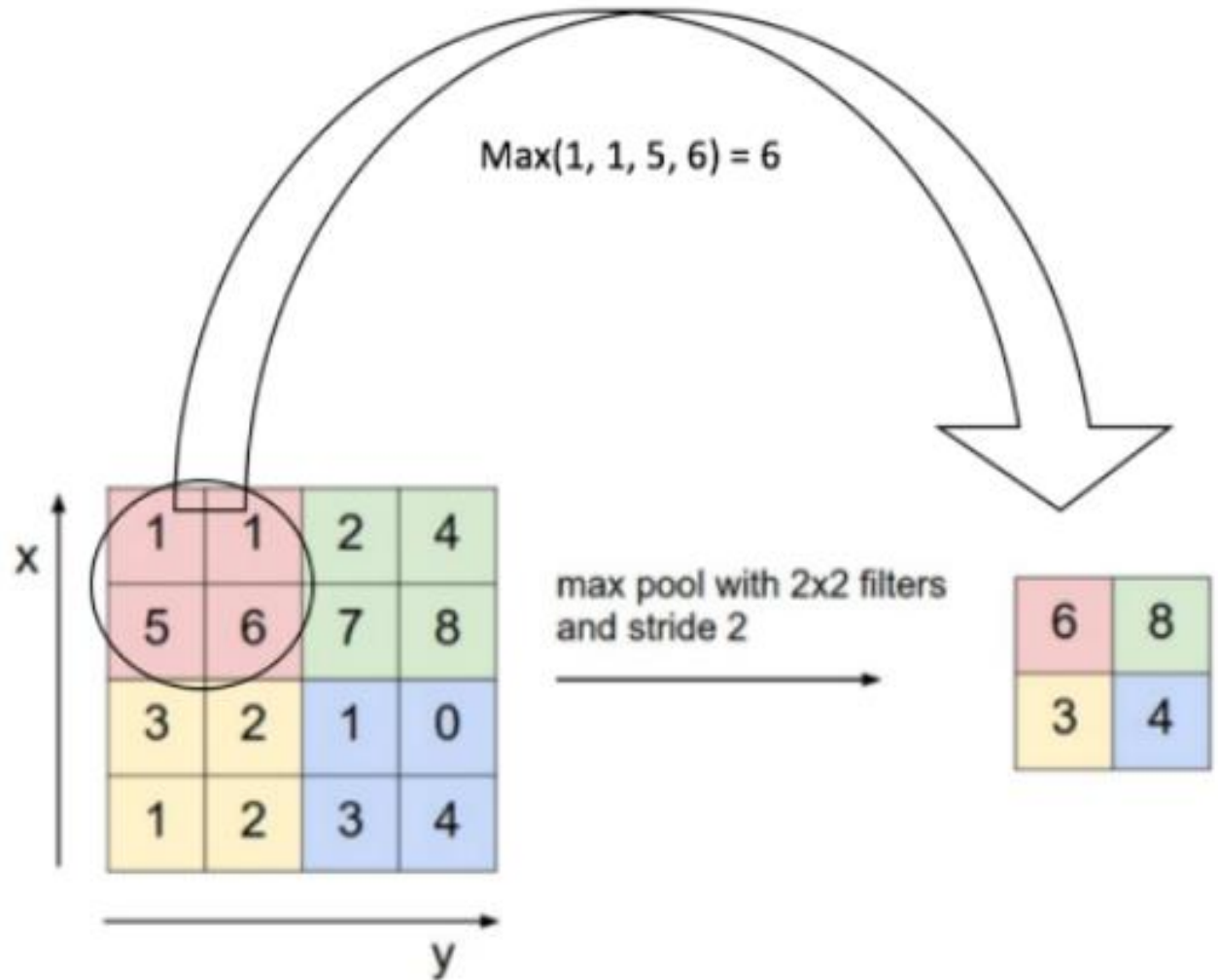
- corrects negative values arising during the previous layer processing by simply adapting them to zero.
- thus, a function  $x' = \max(0, x)$  is applied.



# Network Layers cont'd

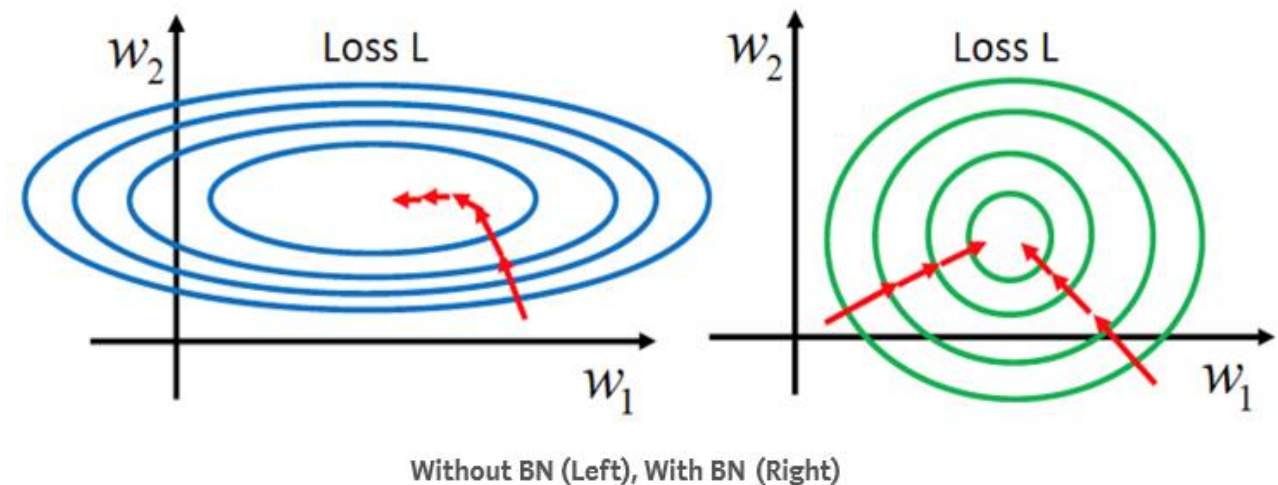
## ■ Pooling

- pixels are aggregated, by applying an arithmetic function
- interpolation strategy to reduce size of the input data
- relevant operations:
  - *max*
  - *sum*
  - *average*



# Network Layers cont'd

- **BN** (batch normalization) [Bjorck et al. 2018]
  - normalization of the activation weights in the intermediate layers
  - it is advantageous for the distribution of  $X$  to remain fixed over time because a small change will be amplified when network goes deeper
  - BN can reduce the dependence of gradients on the scale of the parameters or their initial values. As a result:
    - higher learning rate can be used
    - speed-up for the training
    - the need for *dropout* (prevent from overfitting by randomly excluding neurons) can be reduced
  - *method*: the input is normalized by subtracting the mean  $\mu$  and dividing it by the standard deviation  $\sigma$ 
    - usually inputs to neural networks are normalized to either the range of  $[0, 1]$  or  $[-1, 1]$  or to mean=0 and variance=1. The latter is called *Whitening*.
    - BN essentially performs Whitening to the intermediate layers of the networks
  - with BN all activation / weights approximately normal-deviated → thus no aggregating effect for the consecutive layers



# Network Layers cont'd

## network layers in detail cont'd

### ■ Reshape

- change output dimension of the layer. E.g. transferring from 8x8 to 1x64 and so on.
- re-interpret byte content without copying the data – only manipulate the tensor dimensions (header) itself

### ■ Concat

- concat or interweave multiple BLOBs to one BLOB (c.f. colors)
- e.g.: RGB width \* 3 and order R1 G1 B1 R2 G2 B2 and so on instead of 3 separate BLOBS

### ■ Deconvolution

- transposed convolution
- a deconvolutional network allows to examine different feature activations and their relation to the input space

# Network Layers cont'd

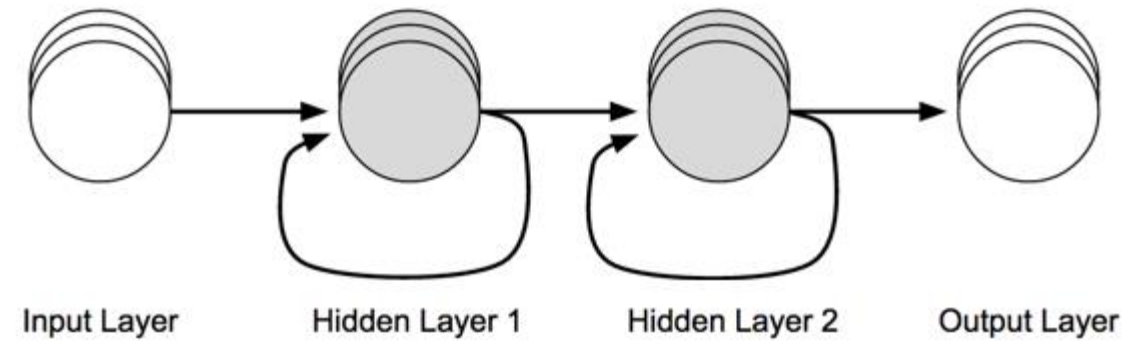
## network layers in detail cont'd

### ■ RNN (recurrent neural networks)

- are able to cover the long-time dependency within the input data
- hidden state stores long-time contextual information on the data
- number of context states modelled by the number of nodes in the recurrent layer
- at each time-step, nodes receiving the current input signal also incorporate information from previous input data through recurrent edge connection. In that sense, the output layer is somehow recursively calculated from the last input signal sequence

### ■ LSTM (long-short term memory)

- variation of RNN first published by [Hochreiter and Schmidhuber 1997]



# Custom Network Layers

- frameworks like OpenCV with *dnn-module* allow design and implementation of generic custom network layers as basic building blocks of deep learning CNNs
  - thereby, brand new ideas as well as adaptations of existing layers are feasible
  - a network layer thereby is a common filter, transferring between specified *input* and *output* data
  - furthermore, the *weights* and so called *hyperparameters* need to be addressed.
    - hyperparameters thereby specific for the used framework (Caffe, Yolo, TensorFlow,...)
  - examples for custom layers
    - resize via interpolation, edge detection (multi-resolution), feature extraction,...

```
class MyLayer : public cv::dnn::Layer
{
public:
    MyLayer(const cv::dnn::LayerParams &params);

    static cv::Ptr<cv::dnn::Layer> create(cv::dnn::LayerParams& params);

    virtual bool getMemoryShapes(const std::vector<std::vector<int> > &inputs,
                                const int requiredOutputs,
                                std::vector<std::vector<int> > &outputs,
                                std::vector<std::vector<int> > &internals) const CV_OVERRIDE;

    virtual void forward(std::vector<cv::Mat*> &inputs, std::vector<cv::Mat> &outputs, std::vector<cv::Mat> &internals) CV_OVERRIDE;

    virtual void finalize(const std::vector<cv::Mat*> &inputs, std::vector<cv::Mat> &outputs) CV_OVERRIDE;

    virtual void forward(cv::InputArrayOfArrays inputs, cv::OutputArrayOfArrays outputs, cv::OutputArrayOfArrays internals)
        CV_OVERRIDE;
};
```

# Network Architectures

---

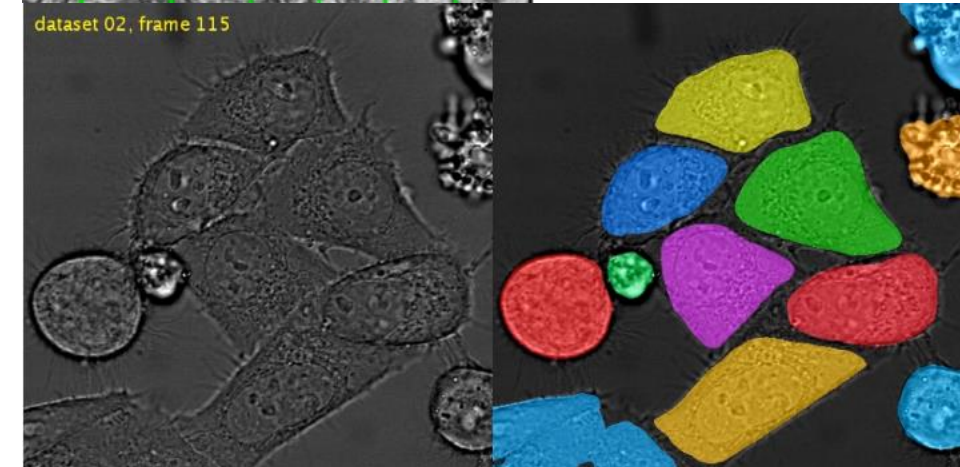
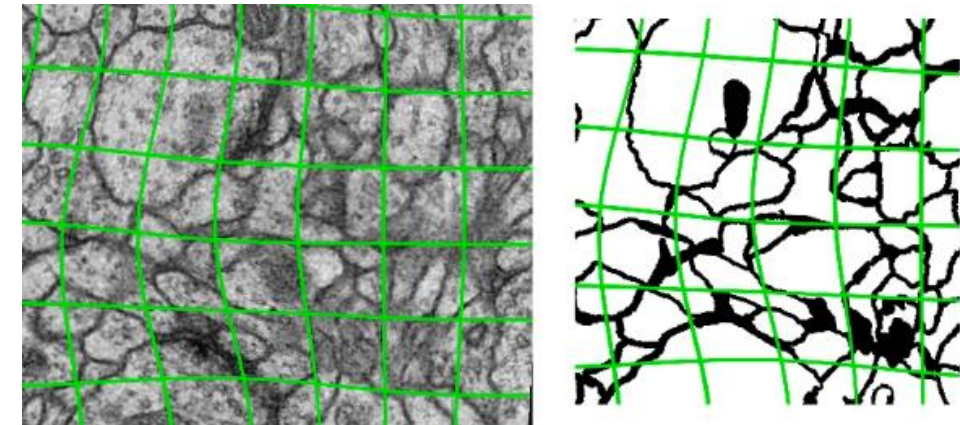
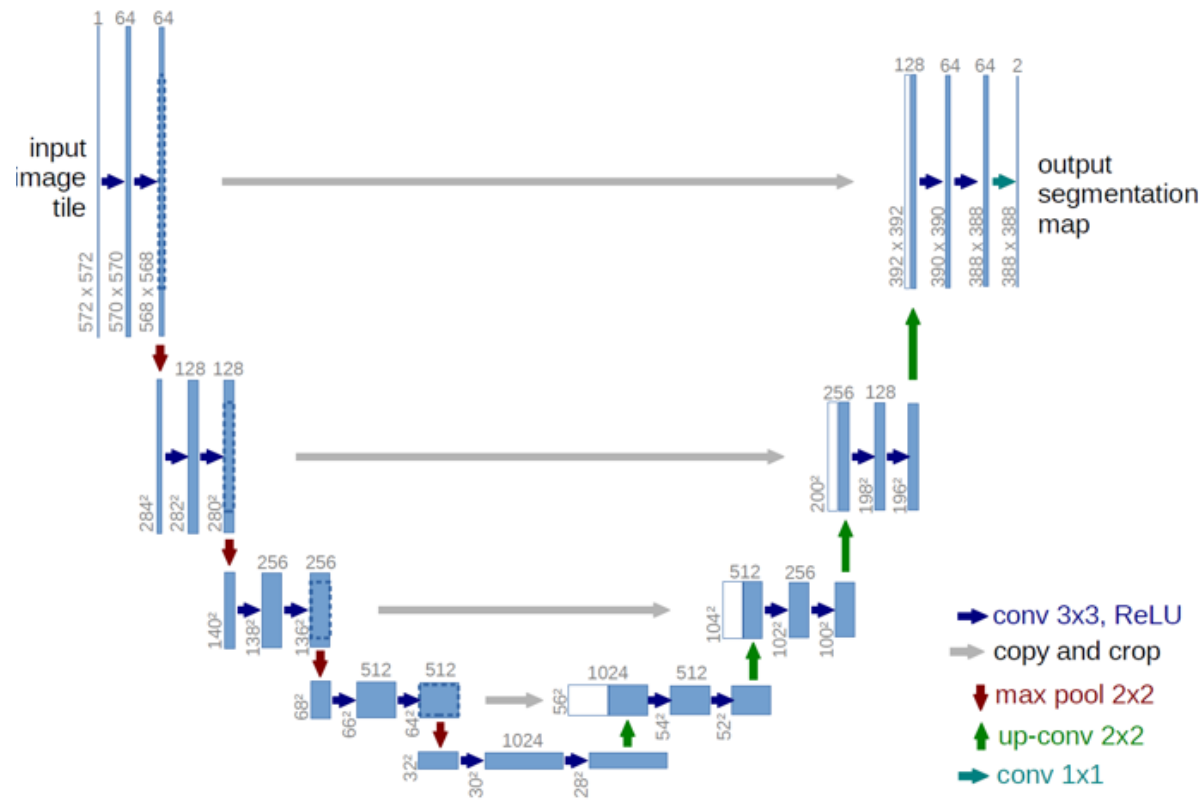
FOR CNN



# Network Architectures

## ■ U-Net [Ronneberger et al. 2015]

- perfectly applicable for fast and precise image segmentation tasks
- input image is convolved at different scale
- layers of matching size in the down- and up-sample process exchange their results (similar to LSTM)



applicable to detect cell borders, deformation and cell segmentation even in case of very weak borders. Cell segmentation challenge 2015: top-place with >75% accuracy, second rank below 50%

# Preprocessing

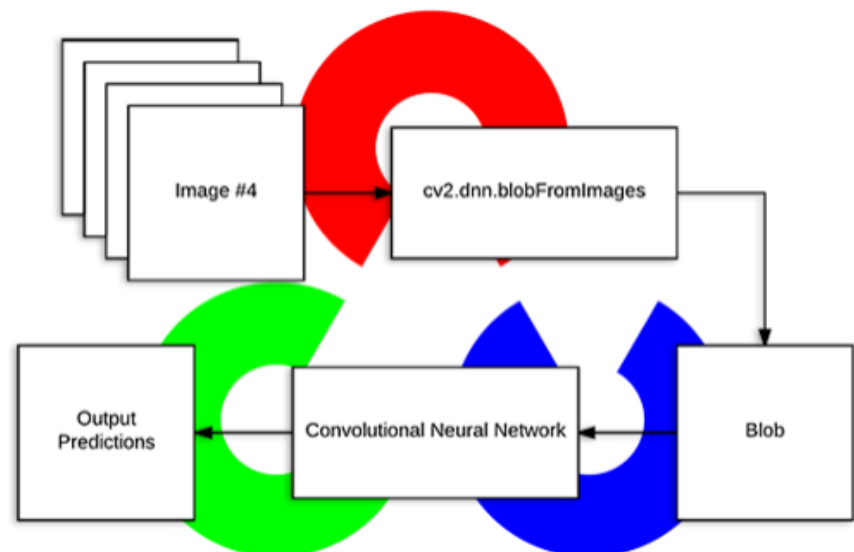
---

FOR CNN

# Preprocessing

## Overview

- **ATTENTION:** the pre-processing strategies highly depend on the CNN layout and the way the input image ROIs are fed into the net
  - if inadequate strategy is chosen, weak results might be the consequence
  - thus, documentation needs to be read in detail
  - pre-processing highly relevant to get plausible, robust and accurate results from CNNs
- **STEP 1 BLOB-conversion:** matching size, pixel-depth and scalar range of all input images
  - choice of interpolation strategy might have significant influence on the achievable output quality. CNN are trained for specific input image size



in OpenCV, the cumulated pre-processing is called conversion to a BLOB.

# Preprocessing cont'd

- **STEP 2 Mean-subtraction:** balances color information from background
  - mean of the image (or the input images of all reference images) is subtracted leading to a normalized scale.
  - will balance illumination changes in the datasets

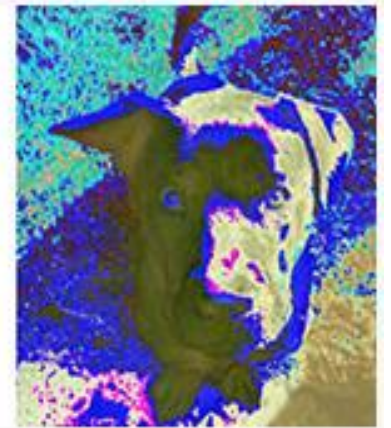
$$R = R - \mu_R$$

$$G = G - \mu_G$$

$$B = B - \mu_B$$



$$\begin{array}{r} R=124.96 \\ - G=115.97 = \\ B=106.13 \end{array}$$



- additionally, scaling by sigma (standard deviation across the training set) is applicable, leading to equal value range per channel

$$R = (R - \mu_R) / \sigma$$

$$G = (G - \mu_G) / \sigma$$

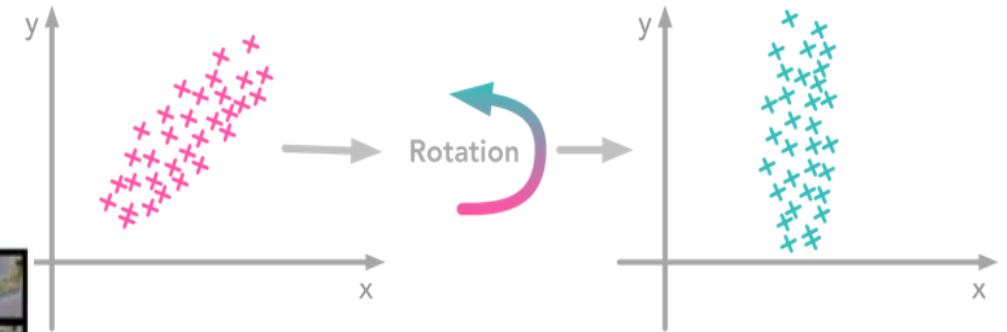
$$B = (B - \mu_B) / \sigma$$



# Preprocessing cont'd

## ■ STEP3 Image Whitening [Pal and Sudeep 2016]

- also called zero component analysis
- transform data in a way that the covariance matrix is the same as the identity matrix (1 in diagonal, rest as zero)
  - zero-center the data
  - uncorrelated the data: rotate, until there is no more correlation
  - rescale the data → dividing by square-root of eigenvectors



after ZCA, the edges are more prominent  
as key features for CNN layers

# Training the CNN

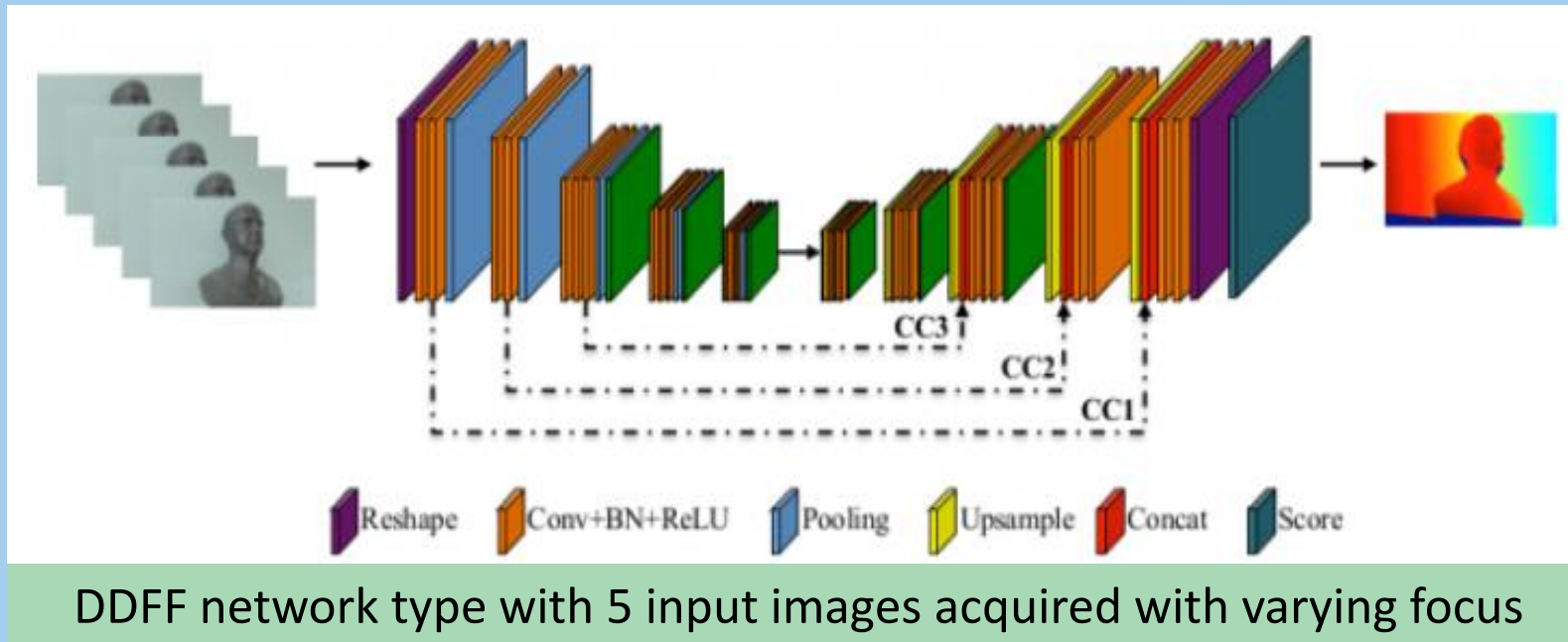
- Backpropagation as key training concept for FFNN
  - **Step1:** initialize all filters and parameters / weights with random values
  - **Step2:** use a training image as input and apply forward propagation for the network (*Convolution, ReLU, Pooling, FCN,...*) to determine the output probability for each class. At step2 the classification (as random value based) most likely expected to be totally wrong
  - **Step3:** calculate the total error from the output layer for all classes to discriminate
    - for  $c$  classes, the total error is calculated as difference between measured and expected per-class probability with  $Err_{total} = \sum_{i=1}^c (prob_{tgt,i} - prob_{out,i})^2$
  - **Step4:** backpropagation is applied to calculate the gradients of the error w.r.t. all weights of the network architecture.
    - gradient descent optimization applicable to step-wise improve the overall quality
  - **Step5:** repeat steps 2-4 until convergence criterion is reached

# Application Domains

---

# Deep Depth from Focus (DDFF)

- With deep depth from focus, static images of the same scene at *varying focus* settings are utilized for precise depth approximation.
- Utilization of deep learning thereby allows for superior results compared to classic solutions for the depth-from-focus task [Hazirbas et al. 2018].
- Idea behind the concept of depth from focus is unsharpening if objects are out of focus (too near or too far). With full knowledge of intrinsic camera parameters, one can deduce the extrinsic camera parameters.

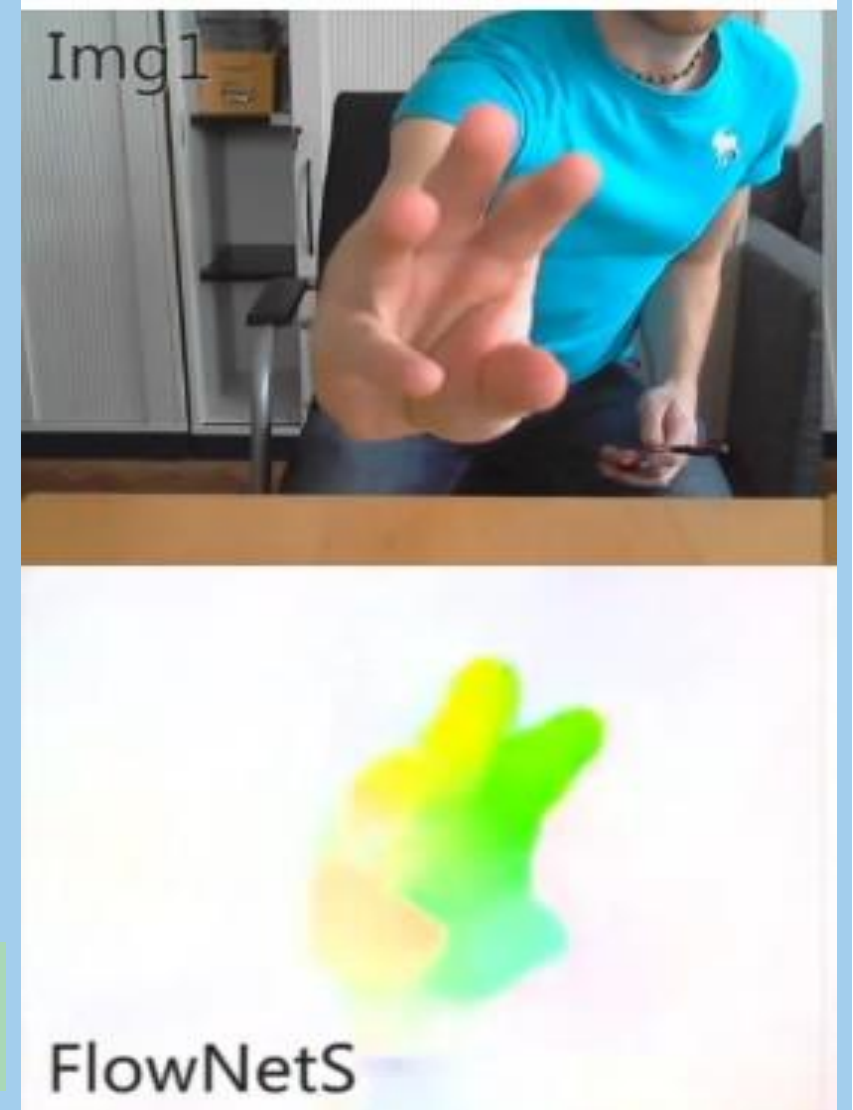




# Optical Flow

- Highly relevant in many domains.
- Required for autonomous driving too to detect moving cars.
- Weather conditions thereby need to be addressed in the training data as they represent the key reason for these approaches to fail in reality. [Fischer et al. 2015]

moving objects detected by  
optical flow



# Video Segmentation

- fully-automated segmentation of object masks (previous classification and tracking required) in videos a hard task
- Utilizing a CNN pre-trained on ImageNet and DAVIS training set, a precisely segmented initial frame (or several frames) is required for fine-tuning of the network.
- As long as the object shows a comparable orientation compared to the input frame, it will be precisely detected.



with one semi-automatically labelled input frame (red), the consecutive frames can be segmented in an automated way

# References

- [Arik et al. 2017] Arik, S.O., Chrzanowski, M., Coates, A., Damos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Raiman, J., Sengupta, S., and Shoeybi, M. 2017. Deep voice: Real-time neural text-to-speech. In *ICML*, 2017.
- [Bjorck et al. 2018] Bjorck, J., Gomes, C., Selman, B., and Weinberger, K.Q. 2018. *Understanding Batch Normalization*. In: Proc. of the Neural Information Processing Systems Conference.
- [Caelles et al. 2017] Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-Taixe, L., Cremers, D., and Van Gool, L. 2017. One-Shot Video Object Segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Available from: <https://vision.in.tum.de/research/deeplearning>, last visited 27.12.2018.
- [Fischer et al. 2015] Fischer, P., Dosovitskiy, A., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., and Brox, T. 2015. FlowNet: Learning Optical Flow with Convolutional Networks. In *CoRR*.
- [Goodfellow et al. 2014] Goodfellow, Ian J. and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, 2014. Generative Adversarial Nets. In Proceedings of the 27th International Conference on Neural Information Processing Systems vol. 2
- [Hastie et al. 2001] Hastie, T., Tibshirani, R., Friedman, J.H., 2001. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. In: *Springer Series in Statistics*.
- [Hazirbas et al. 2018] Hazirbas, C., Soyer, S.G., Staab, M.C., Leal-Taixe L., and Cremers, D. 2018. *Deep Depth From Focus*, In Proc. of the Asian Conference on Computer Vision (ACCV).

# References

- [Hochreiter and Schmidhuber 1997] Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. In: *Neural Computation* 9(8), pp. 1735-1780.
- [Kohonen 1995] Kohonen, T. 1995. Self-Organizing Maps. In: *Springer Series in Information Sciences* 30, Springer, Berlin.
- [Pal and Sudeep 2016] Pal, K.K. and Sudeep, K.S. 2016. *Preprocessing for image classification by convolutional neural networks*. In: Proc. of the 2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), pp.1778-1781.
- [Ronneberger et al. 2015] Ronneberger, O., Fischer, P., and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS (9351), pp. 234-241.
- [Van Biesen et al. 1998] Van Biesen, W., Sieben, G., Lameire, N., and Vanholder, R. 1998. Application of Kohonen neural networks for the non-morphological distinction between glomerular and tubular renal disease. In: *Nephrol Dial Transplant* 13(1), pp. 59-66.
- [Zhang and Wang 2011] Zhang, J., and Wang, X.W. 2011. The application of feed-forward neural network for the X-ray image fusion. In: *J. Phys.*