INTRODUCTION TO DEEP LEARNING

BOSCH

CONTENTS

Introduction to deep learning
Contents

BOSCH

LET'S START WITH SOME EXAMPLES

Introduction
Object detection

https://www.youtube.com/watch?v=VOC3huqHrss

Introduction
Image segmentation

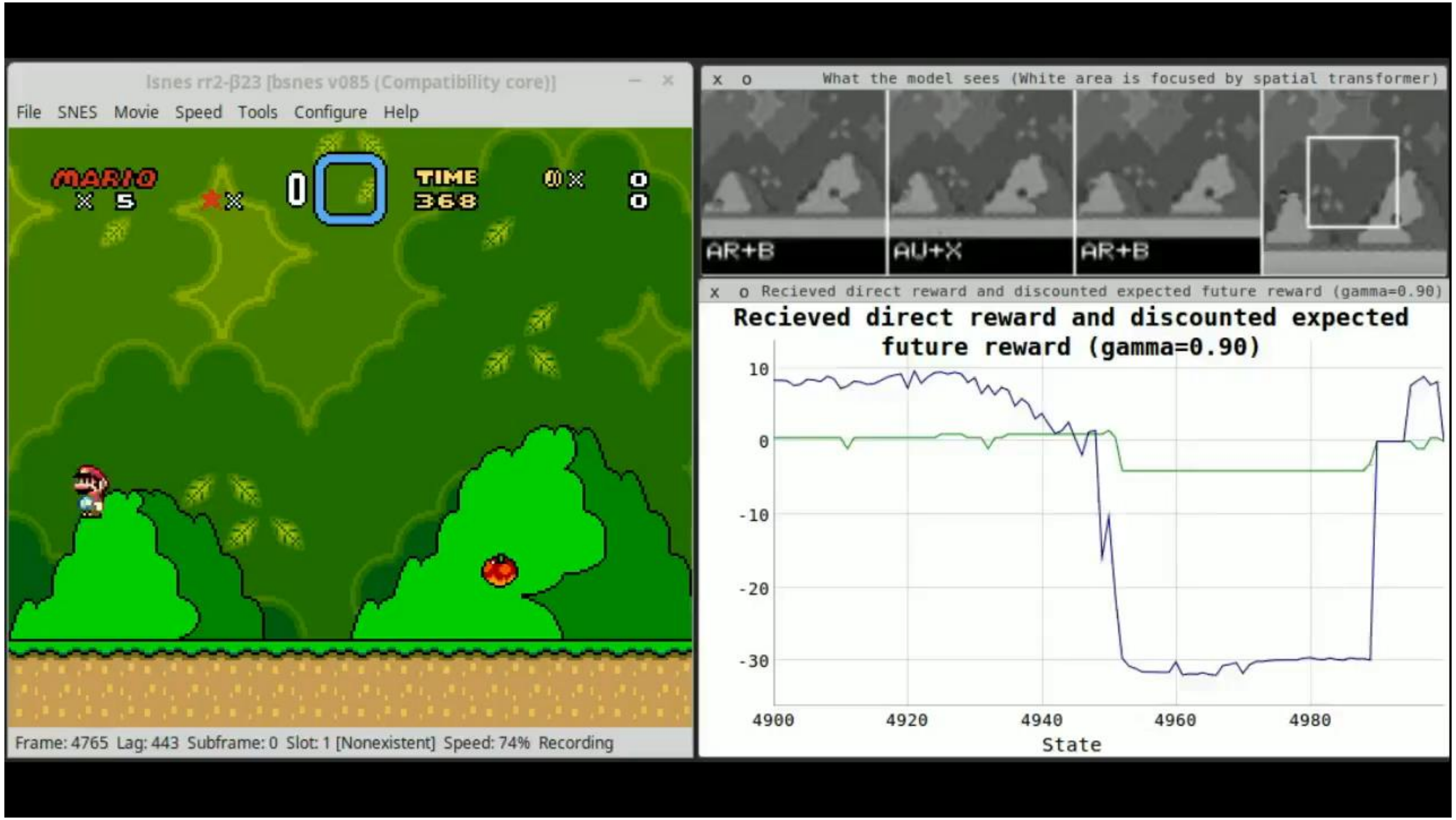
https://www.youtube.com/watch?v=1HJSMR6LW2g

Introduction
Image colorization

100 year old pictures...

https://www.youtube.com/watch?v=ys5nMO4Q0iY

Introduction
Mario

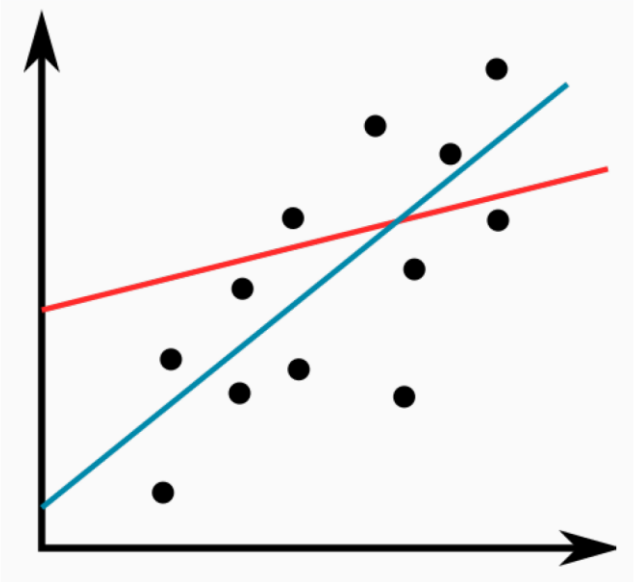


https://www.youtube.com/watch?v=L4KBBAwF_bE

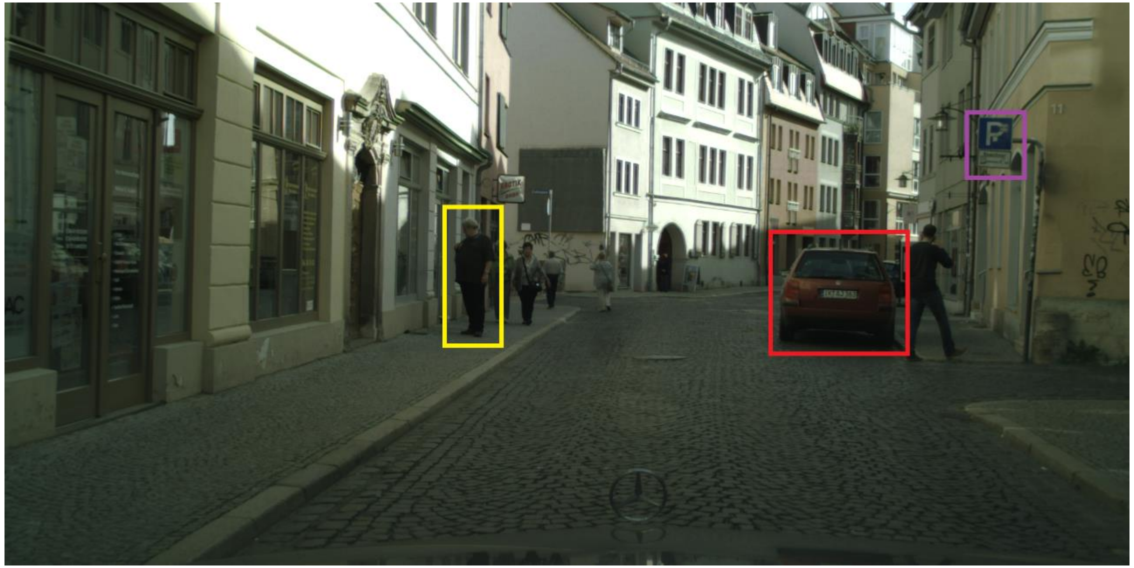
MACHINE LEARNING

Machine learning
What is machine learning?

To learn = algorithmically find the choice of parameters that best explain the data.



- Linear regression -



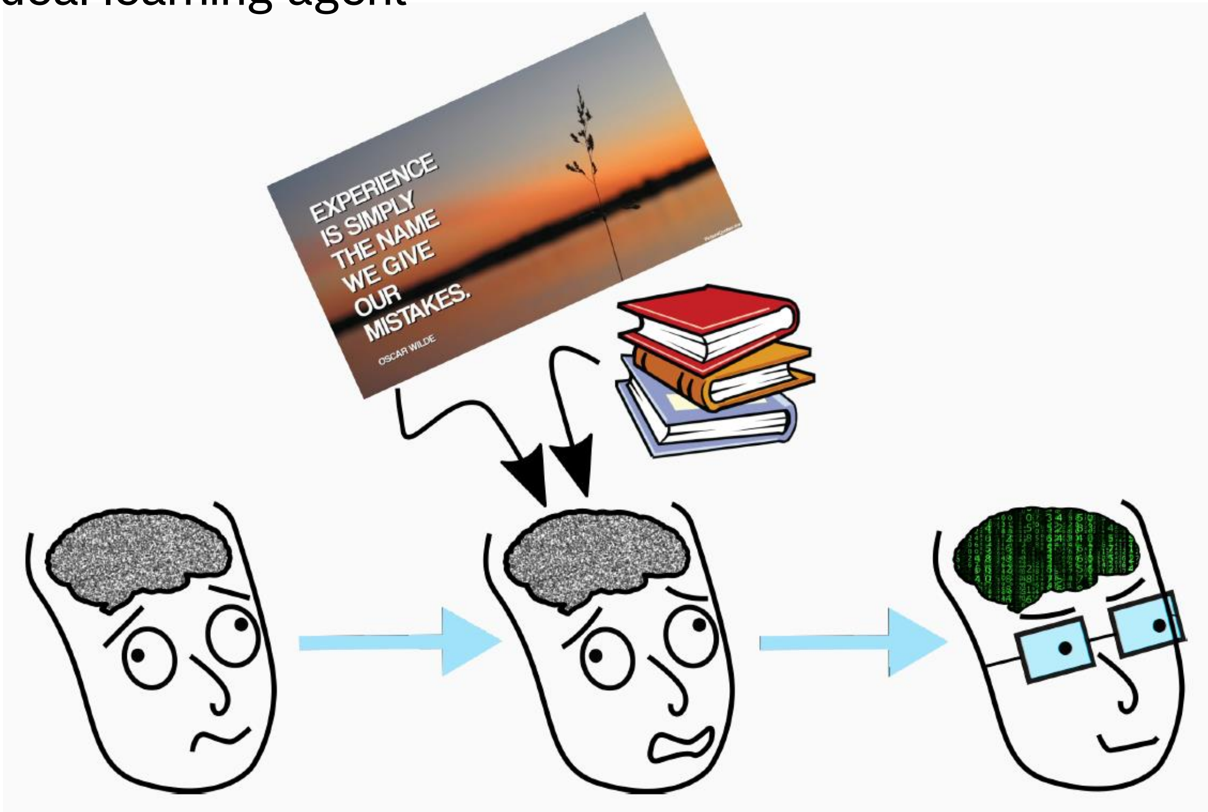
- Object detection -

Machine learning
Uses machine learning in industry

- Object detection
- Image segmentation
- Image classification
- Speech recognition
- Language understanding and translation
- Spam filters and fraud detection
- Automatic email labelling and sorting
- Personalized search results and recommendations
- Automatic image captioning
- Online advertising
- Medical diagnosis

Machine learning
Deep learning – what is it?

- A particular subset of ML algorithms a.k.a. “enhanced neural network”
- The closest to an ideal learning agent



NEURAL NETWORKS

Introduction to neural networks
Biological motivation and connections

- **Intuition** : as humans we use our brains to learn the characteristics of different objects and phenomena.
- **Brain neurons**: receive input signals from dendrites and produce output signals along axon.

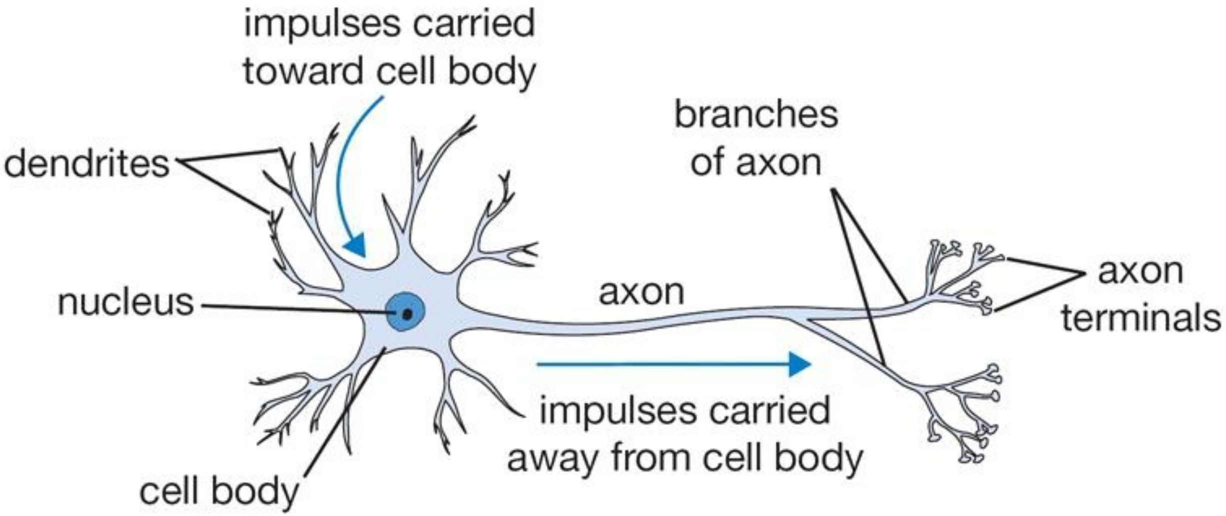
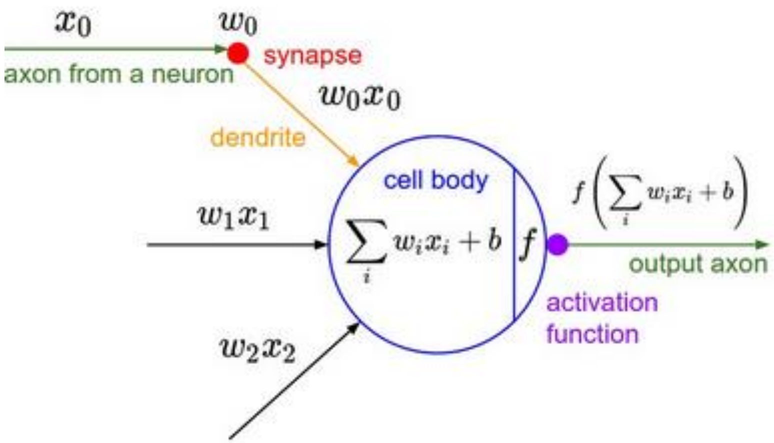


Image taken from http://cs231n.github.io/neural-networks-1

Introduction to neural networks
Biological motivation and connections

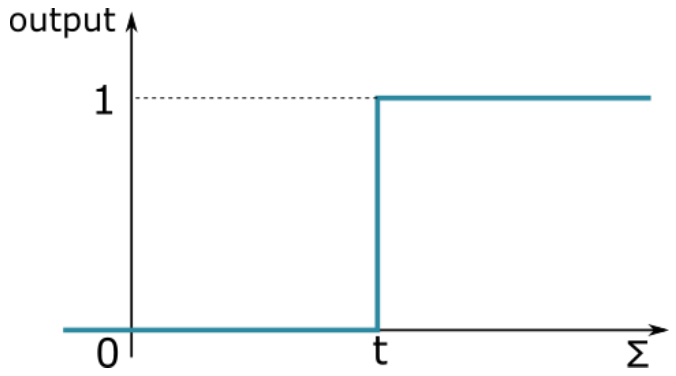
- **Computational model** : signals interact multiplicatively with dendrites of the next neuron (**linear combination of the input signals**).
- The **weights** (synaptic strengths) and **bias** (threshold) are learnable. They control the influence of one neuron over another.



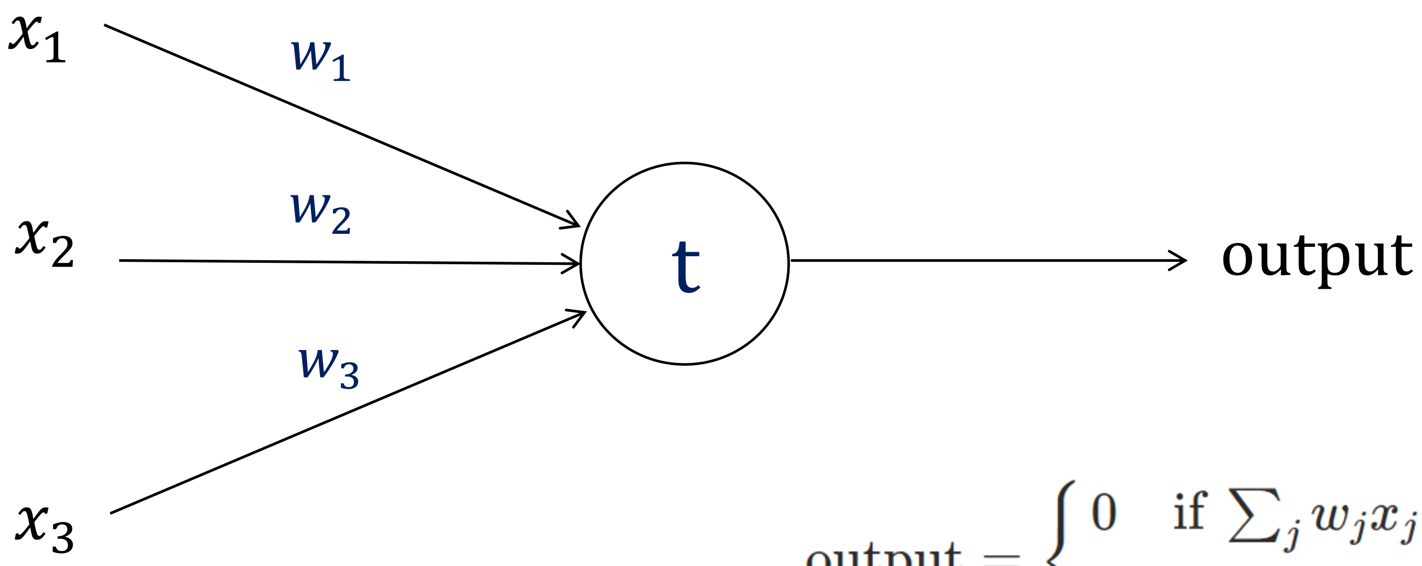
The **weights** w_i and **bias** b are parameters **learned** through **training**.

Image taken from http://cs231n.github.io/neural-networks-1

Introduction to neural network
Perceptron



- Element (neuron) that takes decisions based on evidences
- Takes several binary inputs and produces a single binary output



$x_1, x_2, x_3 \in \{0,1\}$

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Introduction to neural network
Perceptron: example

You are trying to decide whether to go to a concert or not.

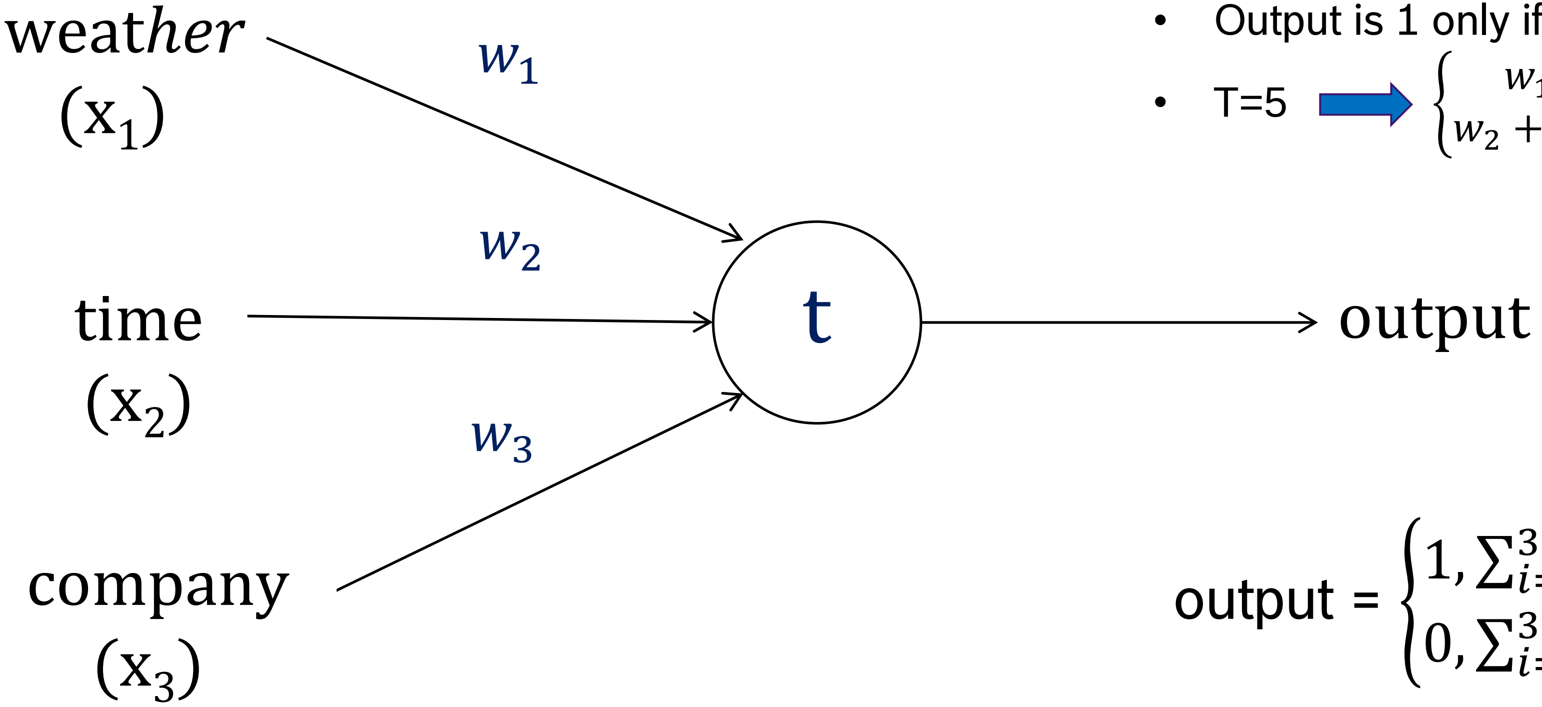
You might make your decision by weighing up three factors:

1. Is the weather good?
2. Do you have enough time to attend the concert?
3. Does your boyfriend or girlfriend want to accompany you?

Introduction to neural networks

Perceptron: example

I only go if the weather is good



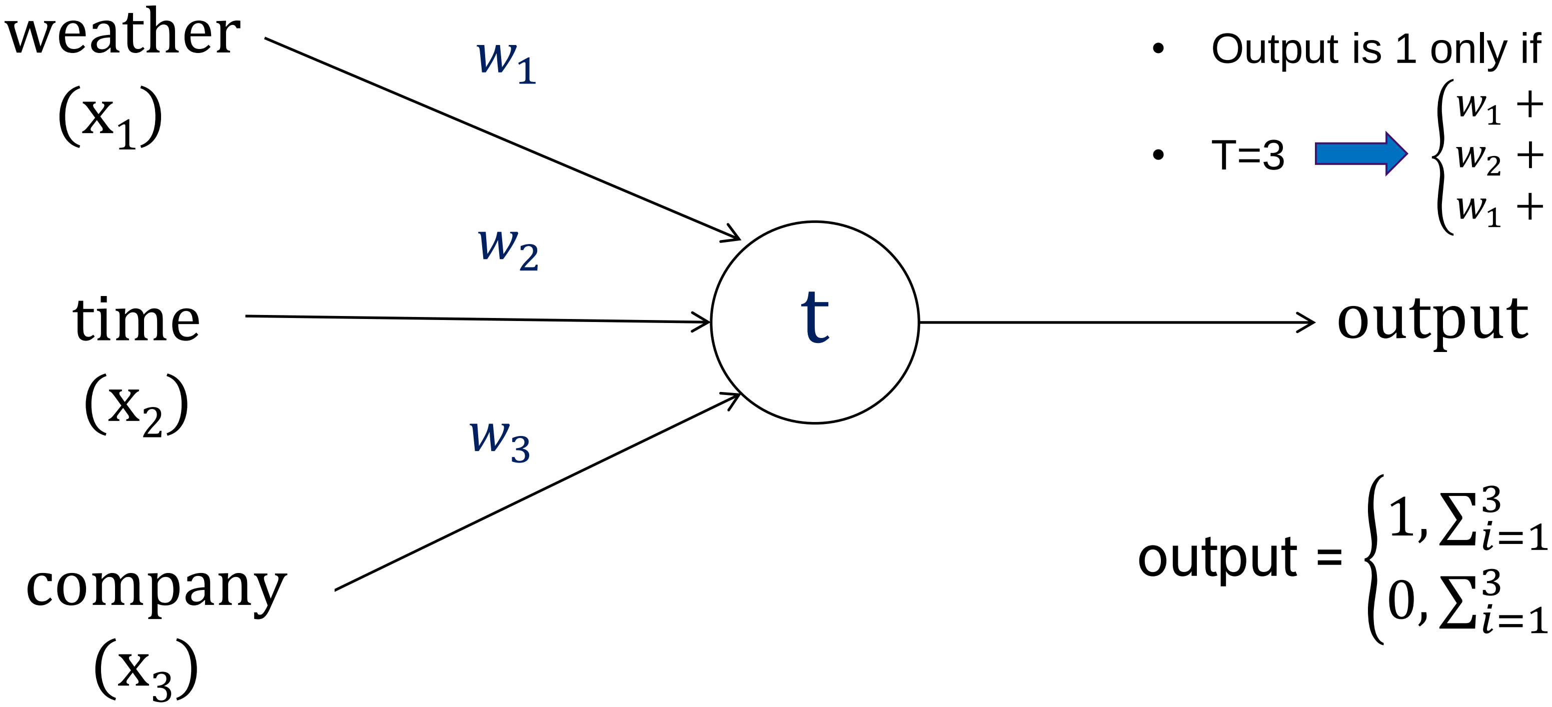
- Output is 1 only if weather's input is 1
- T=5 $\Rightarrow \begin{cases} w_1 > 5 \\ w_2 + w_3 \leq 5 \end{cases}$

$$\text{output} = \begin{cases} 1, \sum_{i=1}^{3} w_i * x_i > 5 \\ 0, \sum_{i=1}^{3} w_i * x_i \leq 5 \end{cases}$$

Introduction to neural networks

Perceptron: example

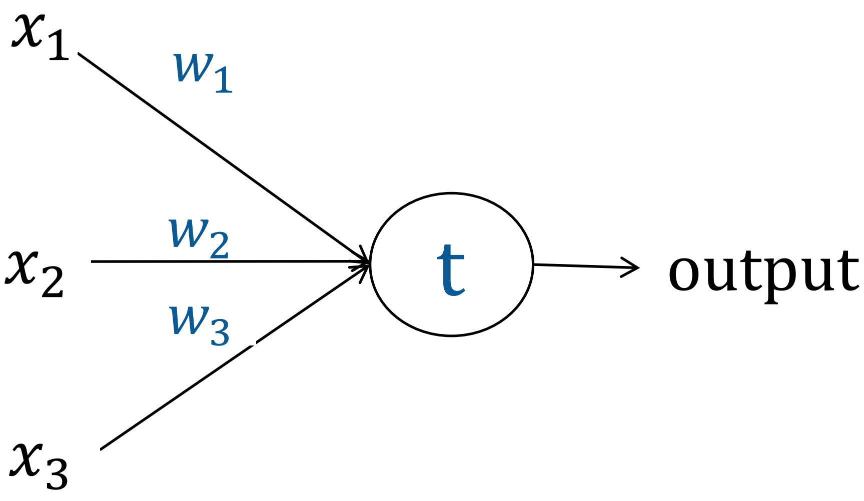
I go if the weather is good and I have enough time!



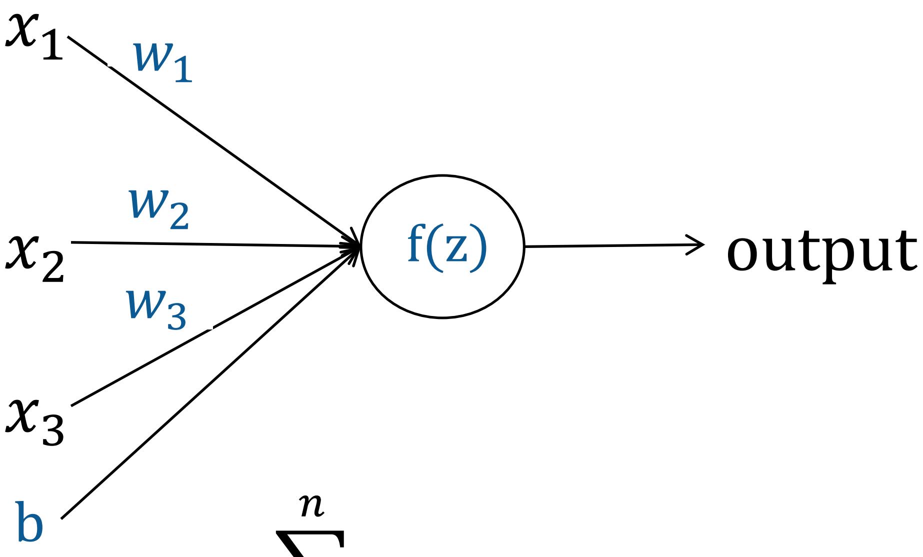
- Output is 1 only if weather’s input is 1
- T=3 ➡ $\begin{cases} w_1 + w_2 > 3 \\ w_2 + w_3 \leq 3 \\ w_1 + w_3 \leq 3 \end{cases}$

$$\text{output} = \begin{cases} 1, \sum_{i=1}^{3} w_i * x_i > 3 \\ 0, \sum_{i=1}^{3} w_i * x_i \leq 3 \end{cases}$$

Introduction to neural networks
Perceptron vs Artificial neuron



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$



$$z = \sum_{i=1}^{n} w_i * x_i + b$$

$$output = f(z)$$

activation function

RBRO/ESA1 | 30/08/2017

ACTIVATION FUNCTIONS AND ARCHITECTURES

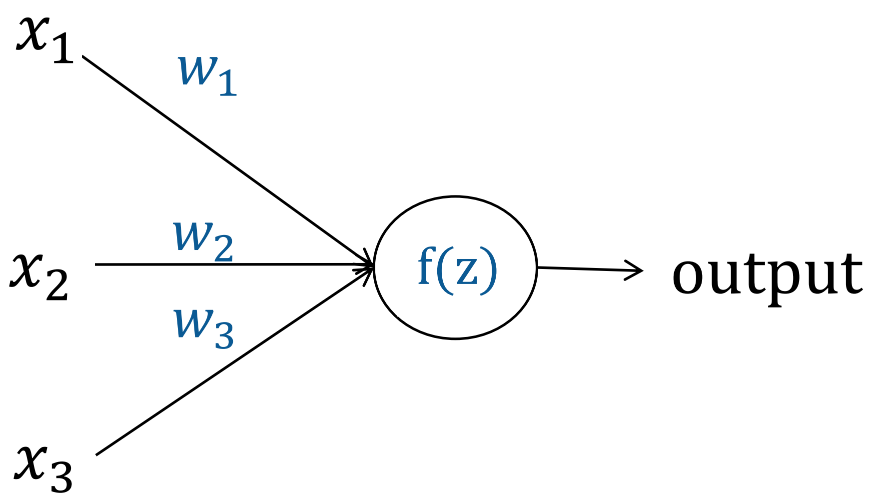
Architectures and activation functions
Activation functions

- Properties:
  - Nonlinear function: makes possible to solve more complex problems -> **artificial neural networks are universal function approximators** [Cybenko 1989, Hornik 1991]
  - Differentiable function: necessary for learning parameters

- The **activation function** is applied after computing the linear value **z** of the neuron

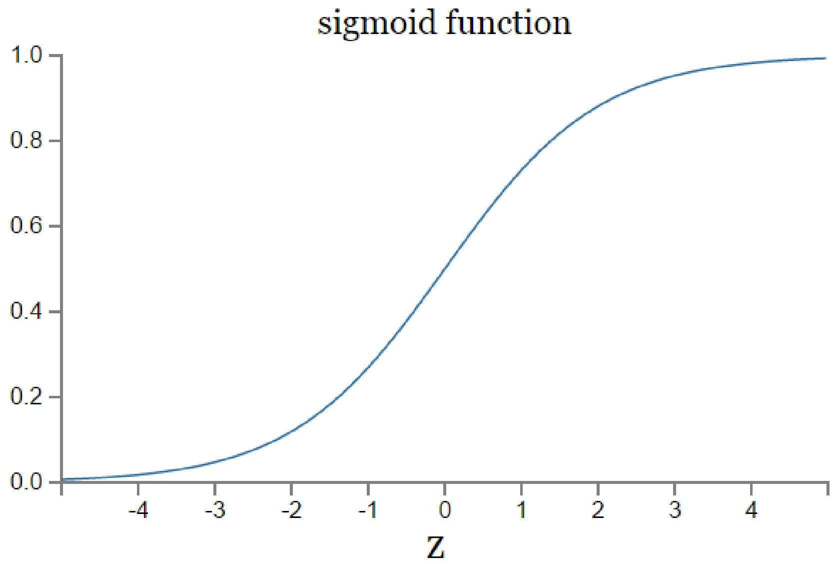
$$z = \sum_{i=1}^{n} w_i * x_i + b$$

$$output = f(z)$$



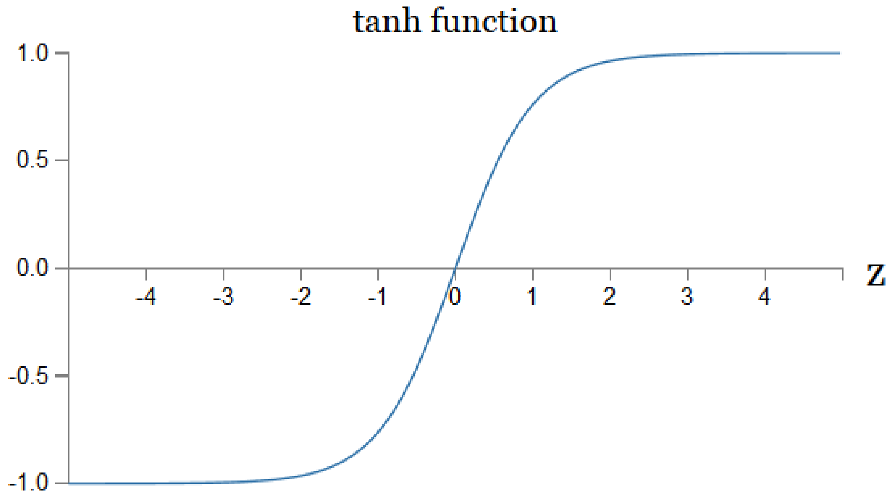
Architectures and activation functions
Activation functions

- **Sigmoid function**



$$\sigma(z) = \frac{1}{1+e^{-z}}$$

- **Tanh function**

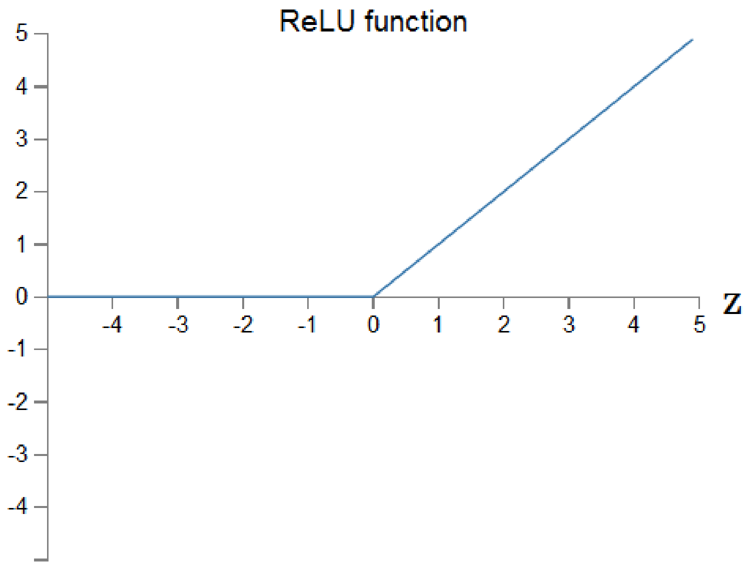


$$\tan(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

Images taken from http://neuralnetworksanddeeplearning.com"”

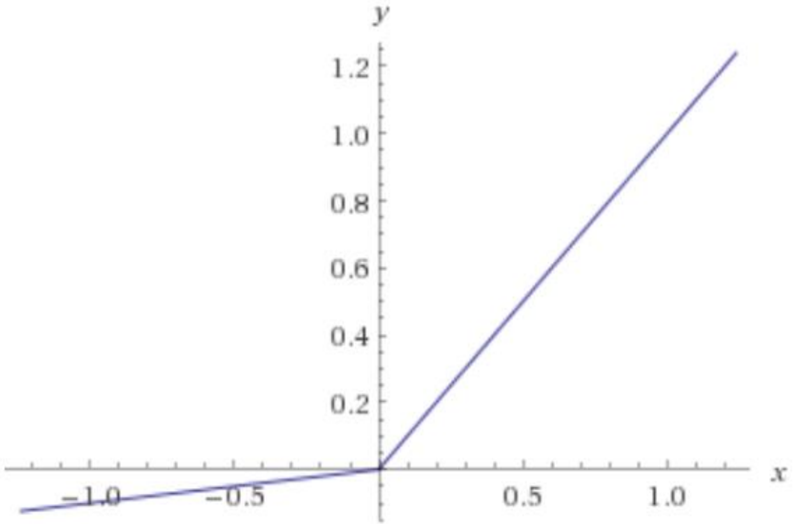
Architectures and activation functions
Activation functions

- ReLU function



$$\text{relu}(z) = \max(0, z)$$

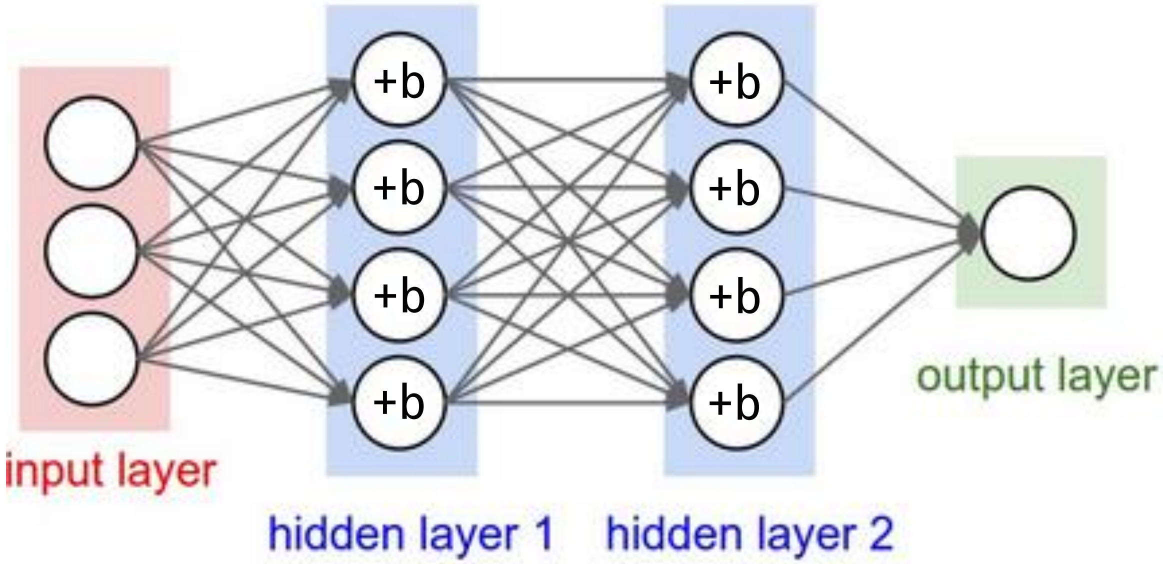
- Leaky ReLU function



$$\text{Lrelu}(z) = \max(0.01 * z, z)$$

Images taken from https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks""

Architectures and activation functions
Architectures



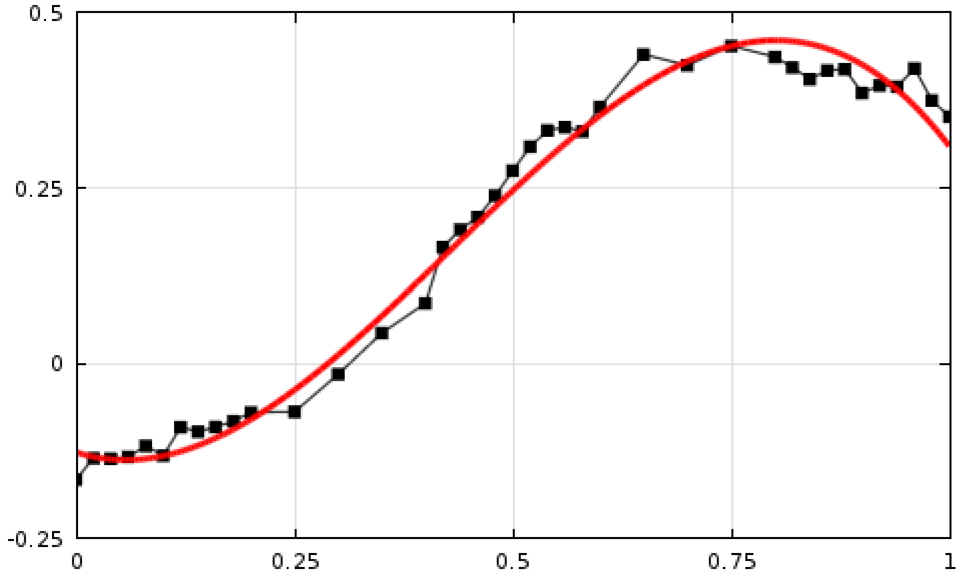
Images taken from http://cs231n.github.io/neural-networks-1””

APPLICATIONS

Neural networks

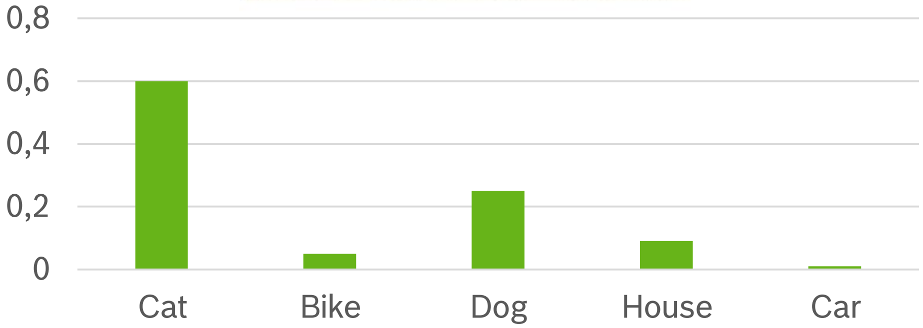
Regression vs Classification

- Fit some real-valued function
- $y_i = f(x_i, W), y_i \in \mathbb{R}$



https://cdn.comsol.com/wordpress/2015/03/Experimental-data-and-fitted-function.png

- Assign a label to an input vector
- $y_i = f(x_i, W),\ y_i \in \{cat, bike, dog, house, car\}$



Applications
Applications of shallow neural networks

Handwritten digit/character recognition

https://knowm.org/wp-content/uploads/Screen-Shot-2015-08-14-at-2.44.57-PM.png

Stock market (time series) prediction

http://milenia-finance.com/wp-content/uploads/635963392980931659203580943З_stock-market.jpg

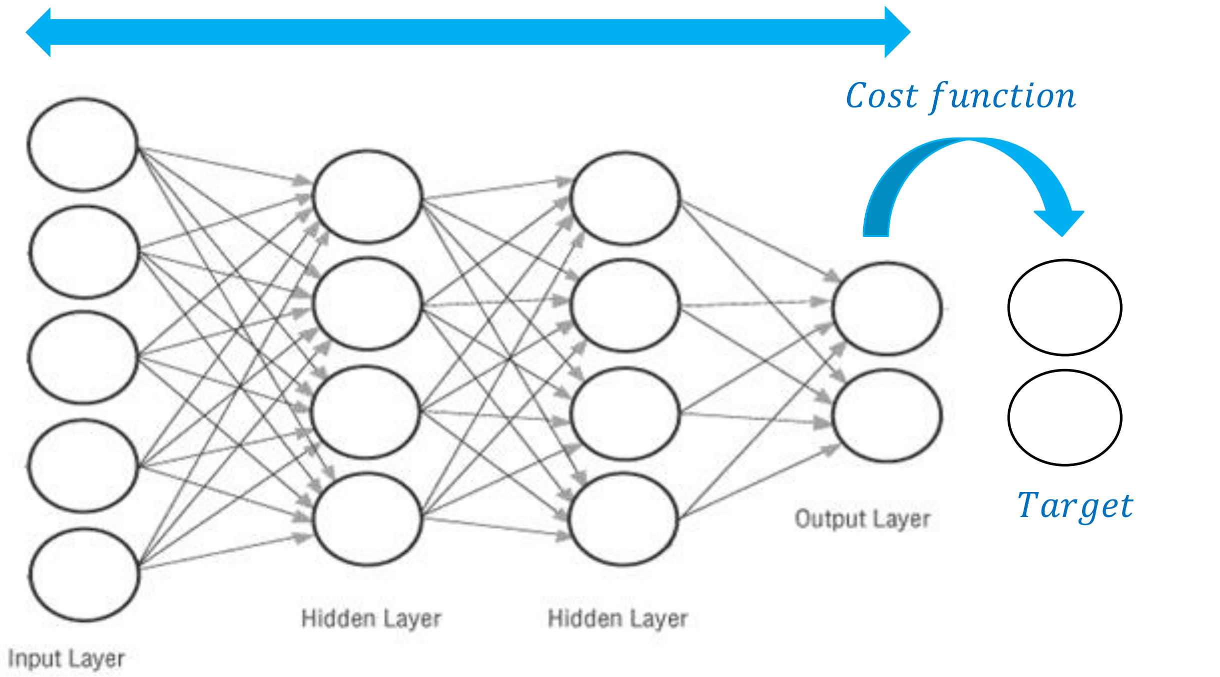
Image compression

https://ai2-s2-public.s3.amazonaws.com/figures/2016-11-08/1e50094bcaf81dac5ea44cea87fd84b25ceb9090/2-Figure3-1.png

TRAINING A NEURAL NETWORK
BACKPROPAGATION

Training a neural network - Backpropagation
General aspects

- Common method for training a neural network
- Goal: optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs (learn to generalize a problem)
- Steps:
  - ➢ Forward step
  - ➢ Compute the error
  - ➢ Backward step
  - ➢ Update parameters



*example valid for supervised learning

Image taken from “” https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example””

Training a neural network - Backpropagation
Cost function

The Squared Error: $$C(w,b) = \frac{1}{2}\sum_{i=1}^{m}\|y(x_i) - \hat{y}(x_i,w,b)\|^2$$

- x_i: network input set
- $y(x_i)$: labeled output set (expected, true outputs)
- $\hat{y}(x_i,w,b)$: network outputs

- “How good" a neural network did w.r.t. it's given *m* training samples and the expected output
- The set of weights and biases have done a great job if $C(w,b) \approx 0$
- Our aim is to minimize it such that $\hat{y}(x_i,w,b)$ **becomes identical to** $y(x_i)$

How ?

Training a neural network - Backpropagation
Gradient descent

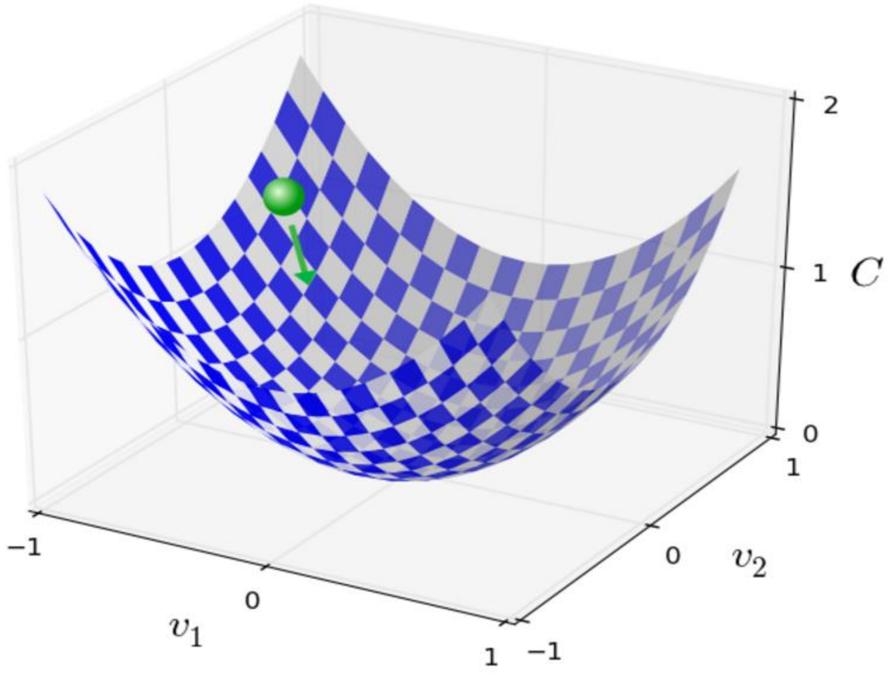
- Optimization algorithm used for finding the minimum of a cost function
- Cost function depends on weights and biases
- Gradient finds how much a weight or bias

causes the cost function's value

- Update weights and biases to minimize the cost

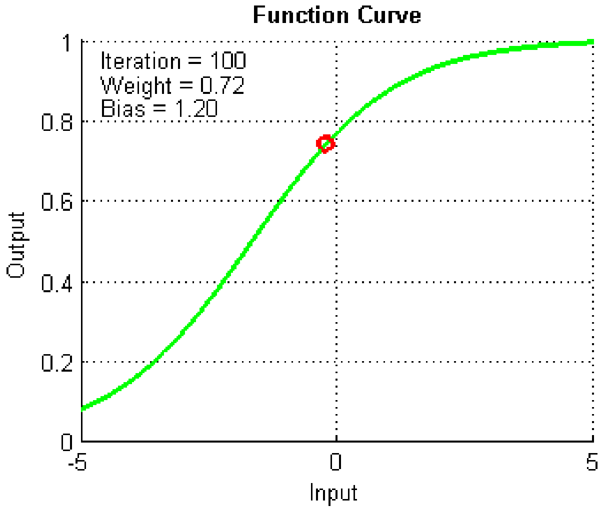
function

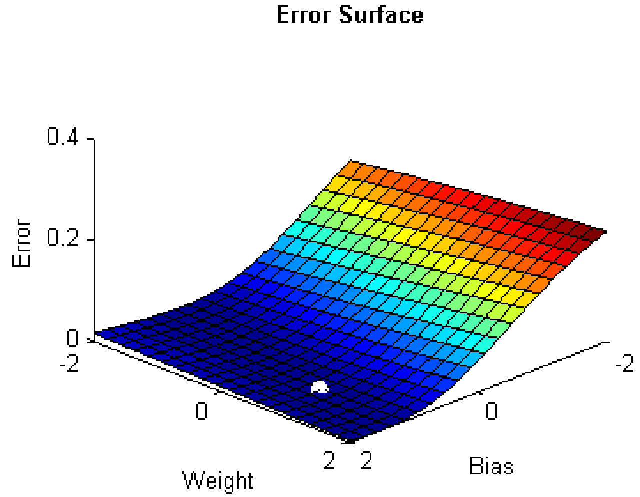
- **Learning rate:**
  - ➢ used for weights and bias updates
  - ➢ small, positive parameter
  - ➢ fixed or dynamic



Images taken from http://neuralnetworksanddeeplearning.com""

Training a neural network - Backpropagation
Gradient descent





Weight and bias update after a training sample:

learning rate

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}$$

Images taken from http://neuralnetworksanddeeplearning.com""

DEEP LEARNING

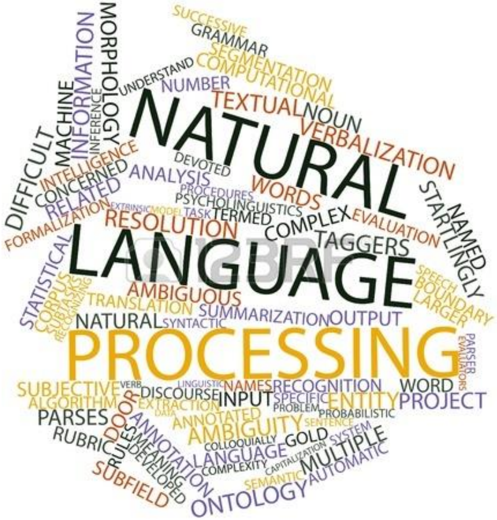
NAÏVE DEEP LEARNING

Naïve deep learning
Applications

- Exceptional effective at learning patterns
- Solves complex problems
- Applications:

Speech recognition

Computer vision

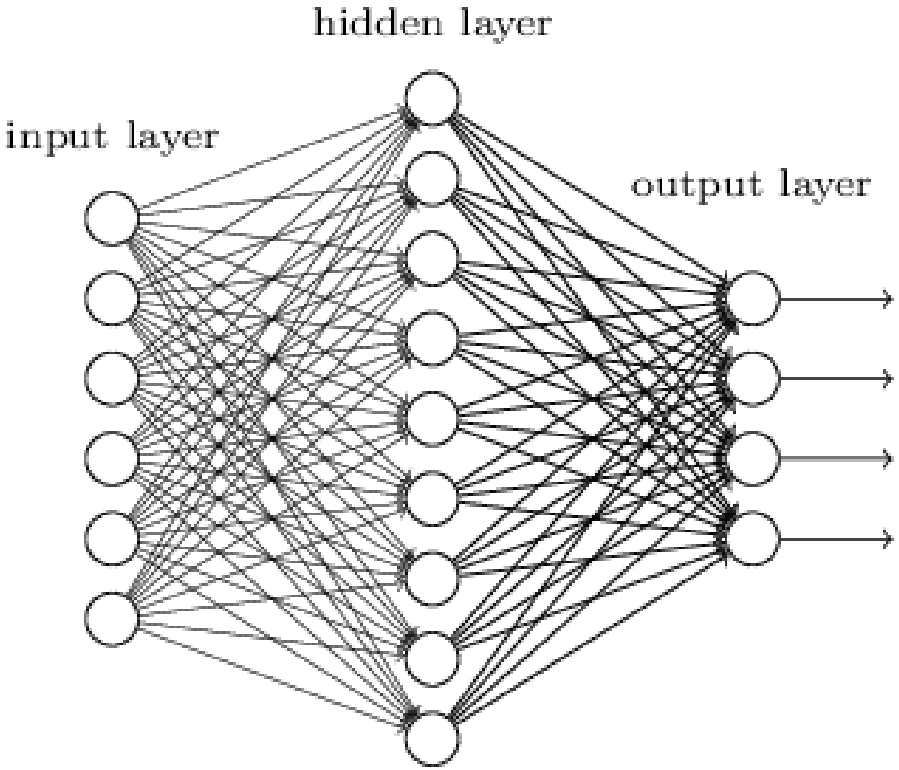


Natural language processing

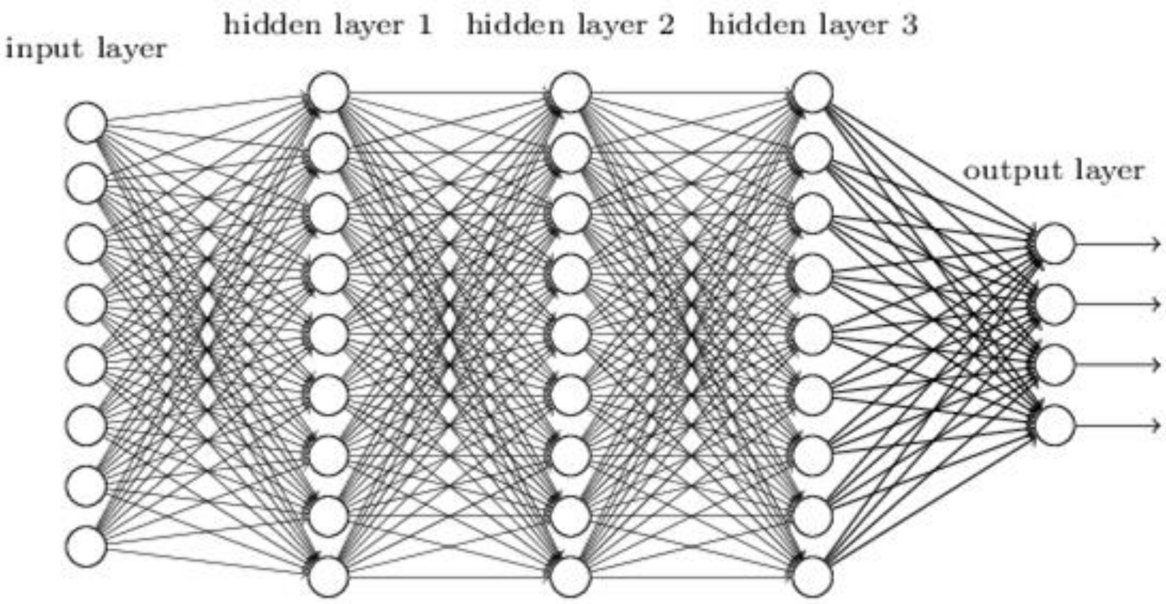
RBRO/ESA1 | 30/08/2017

BOSCH

Naïve deep learning
From shallow to deep



Shallow Neural Network



Deep Neural Network

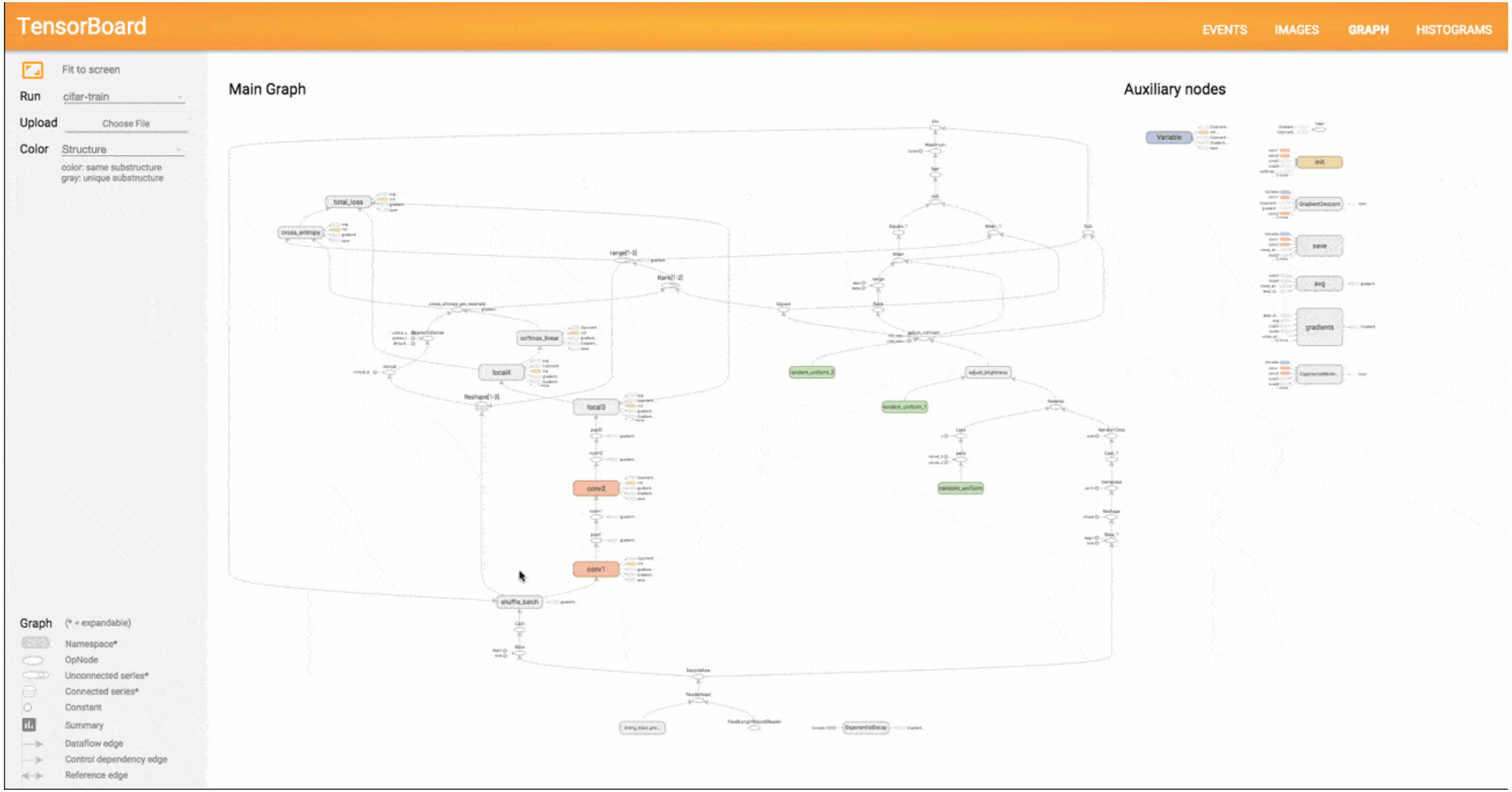
Source: http://neuralnetworksanddeeplearning.com

Naïve deep learning
Deep neural networks challenges

- Inputs are vectors => spatial relationships are not preserved (input scrambling)
- The number of parameters increases exponentially with the number of layers
- Huge number of parameters would quickly lead to overfitting
- Networks with many layers have an unstable gradient problem

DEEP LEARNING

Deep learning
Neural networks as computational graphs

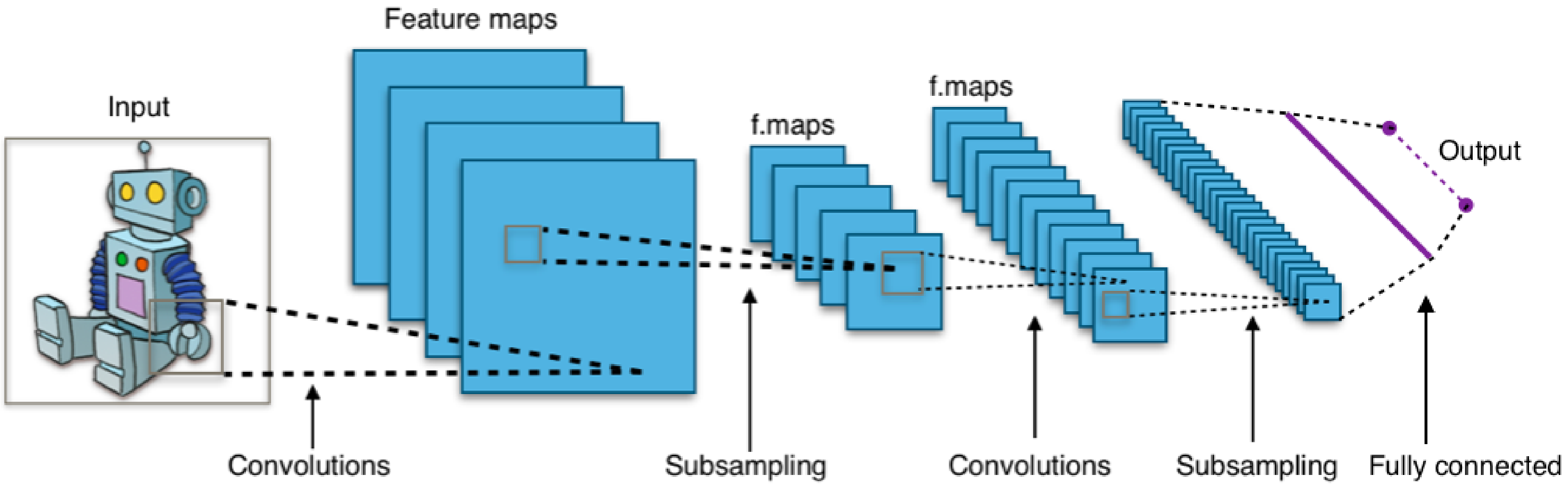


BOSCH

CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Network
Layers of Convolutional Neural Network

1. Input layer
2. Convolutional layer
3. Subsampling layer
4. Fully connected layer

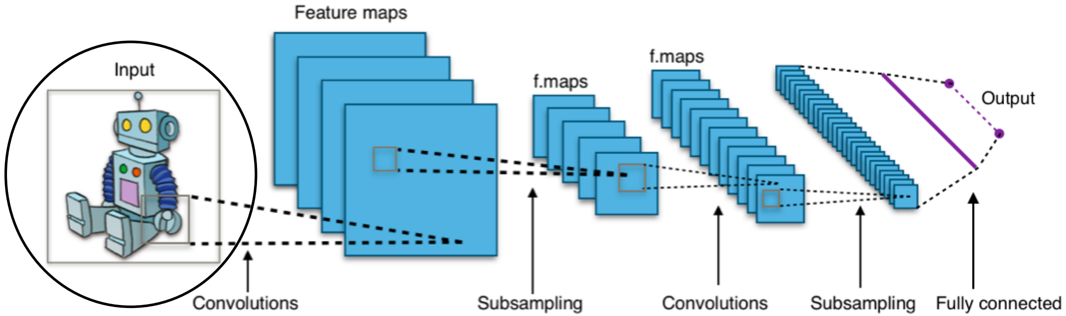


Source: https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

BOSCH

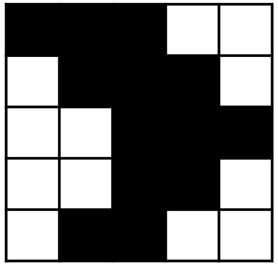
Convolutional Neural Network

Input layer: What is an image?



Binary image:

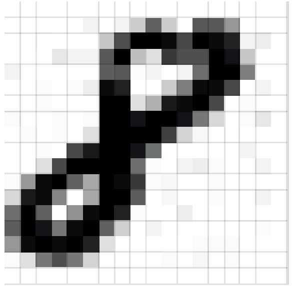
- A matrix of pixel values – each pixel is either 0 or 1



Binary

Grayscale image:

- A matrix of pixel values – each pixel is a natural number between 0 and 255
- Pixel value – intensity of light



Grayscale

RGB image:

- 3 matrices of pixel values – each pixel is a natural number between 0 and 255
- Pixel value – intensity of the color (red, green or blue)

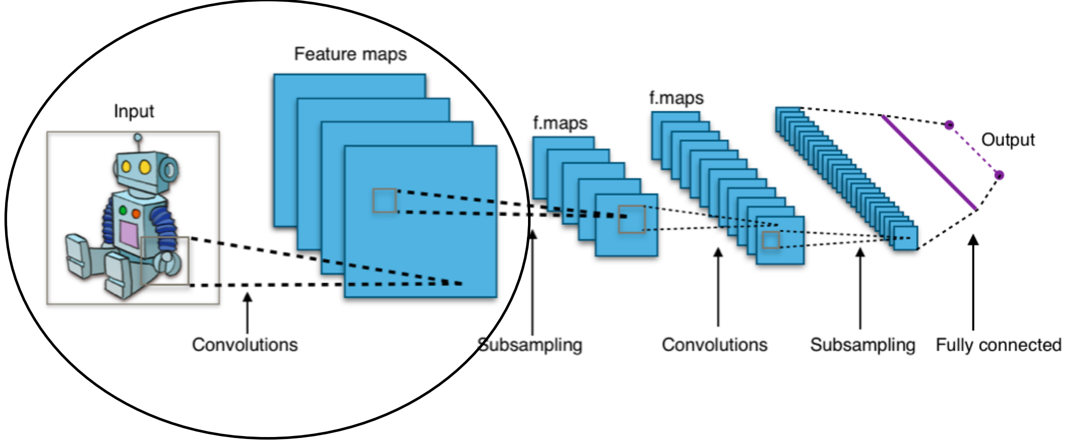
RGB image

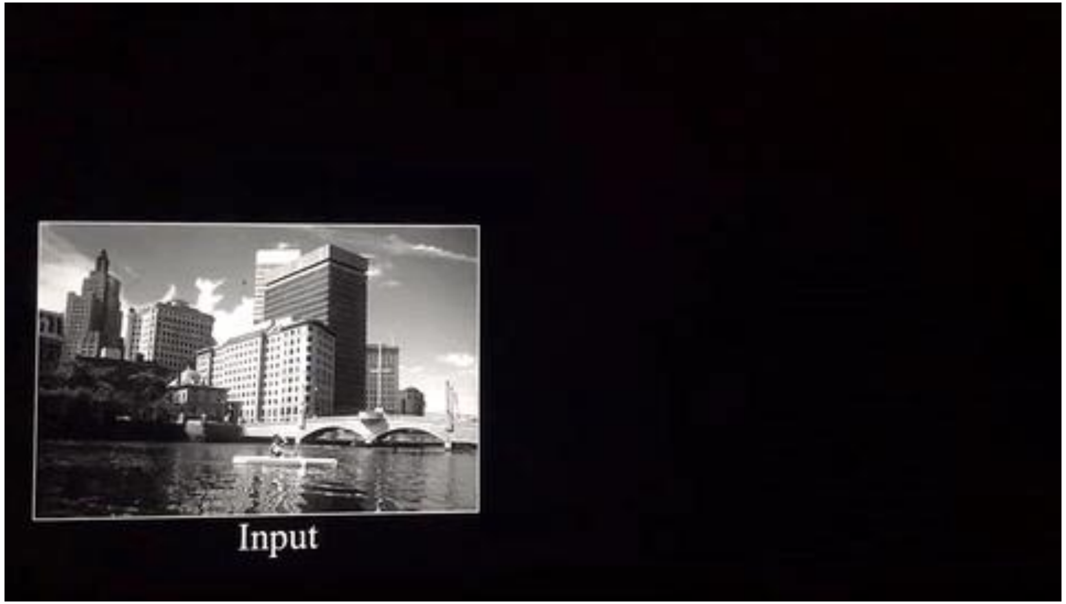
Source: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

Convolutional Neural Network
Convolutional layer

The purpose of convolution is to **extract features** from the input:

1. Gets a 3D matrix as an input
   e.g.: an RGB image with depth 3
2. “Convolves” multiple kernels on the input 3D matrix
3. Creates the output 3D matrix: the feature maps – also called activation maps





Source: http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

Convolutional Neural Network
Convolution on a single matrix

Input:

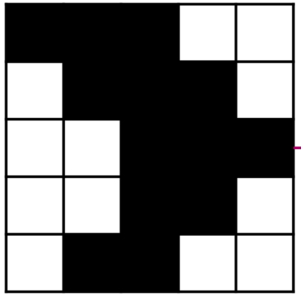
- W_1 x H_1 matrix (e.g. binary image)
- Kernel (filter): W_2 x H_2 matrix

Convolution operation:

- Slide the kernel over the W_1 x H_1 matrix
- At each position calculate element wise multiplication
- Calculate the sum of multiplications

Output:

- W_3 x H_3 matrix –> feature map (activation map)



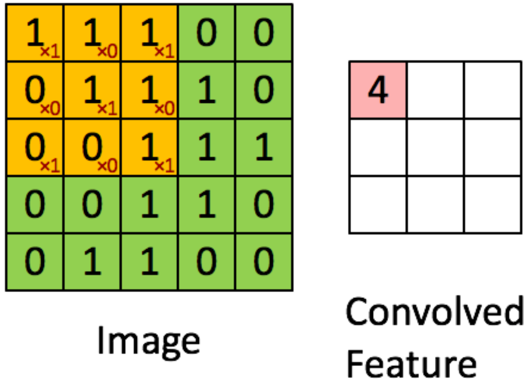
Input image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input matrix

1	0	1
0	1	0
1	0	1

Kernel

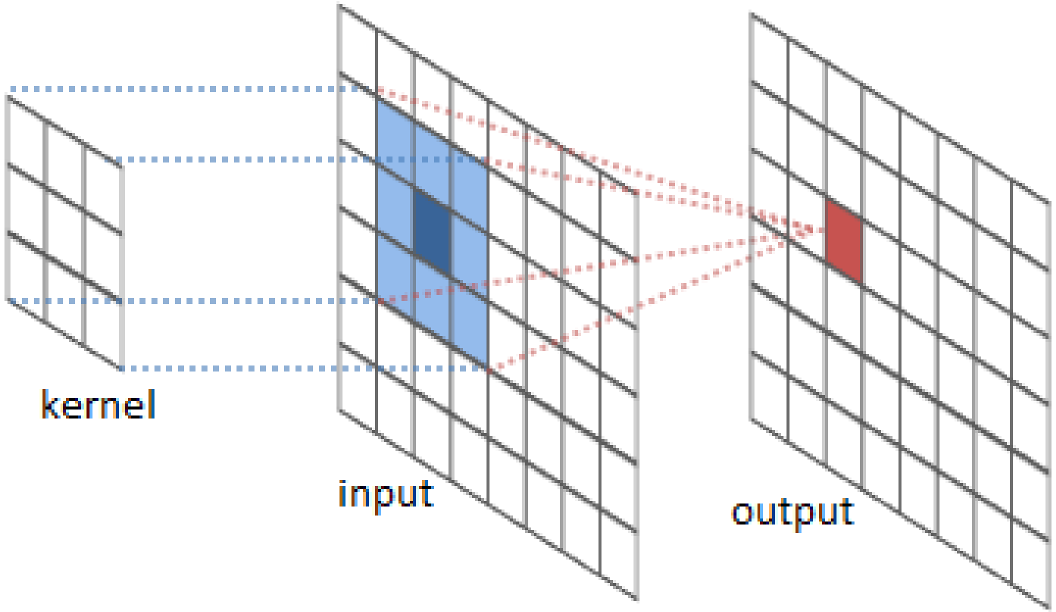


Source: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

BOSCH

Convolutional Neural Network
Convolution on a single matrix

- Another view of the convolution operation



W_1 x H_1 matrix

W_2 x H_2 matrix

W_3 x H_3 matrix (feature map)

http://intellabs.github.io/RiverTrail/tutorial/

BOSCH

Convolutional Neural Network
What is a kernel?

- **It is a learnable filter**
- The values (weights) of the kernel will be learned during the training process
- The trained filter will activate on the image (during the forward pass) when it sees some type of visual feature on the image (e.g.: edges)

- The **size of the kernel** is a **hypermarameter** of the convolutional layer
- Typical kernel sizes: 3x3, 5x5

-5	8	-1
-2	6	3
7	-1	4

Random kernel

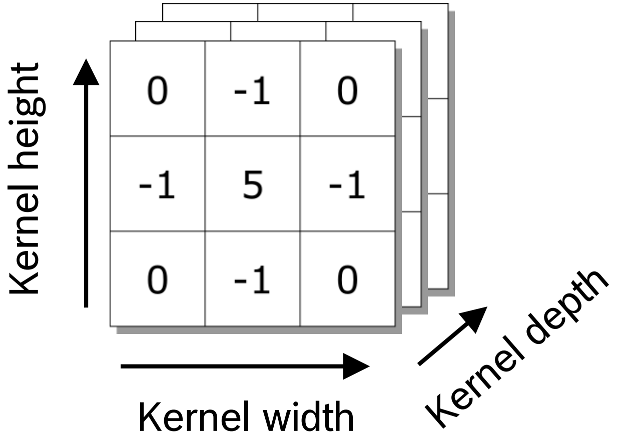
→

0	-1	0
-1	5	-1
0	-1	0

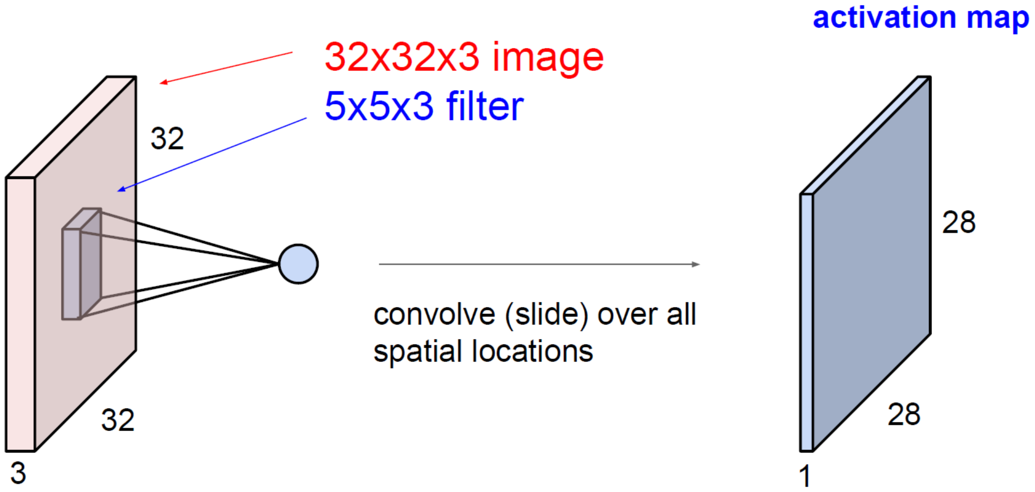
Trained kernel

Convolutional Neural Network
Convolution on a 3D matrix

- The input is an W_1 x H_1 x $\mathbf{D_1}$ 3D matrix
  - W_1 x H_1 is the width and height of the 3D matrix
  - $\mathbf{D_1}$ is the depth of the 3D matrix
  - E.g.: an RGB image (with 3 channels)
- To convolve a kernel on the 3D matrix, the depth of the kernel should be the same: W_2 x H_2 x $\mathbf{D_1}$



- The convolution operation is still the same:
  - Slide (convolve) the kernel across the width and height of the input 3D matrix
  - At each position calculate element wise multiplication
  - Calculate the sum of multiplications
  - (It produces 1 value at each position)
- The output of the convolution is an W_3 x H_3 x 1 feature map



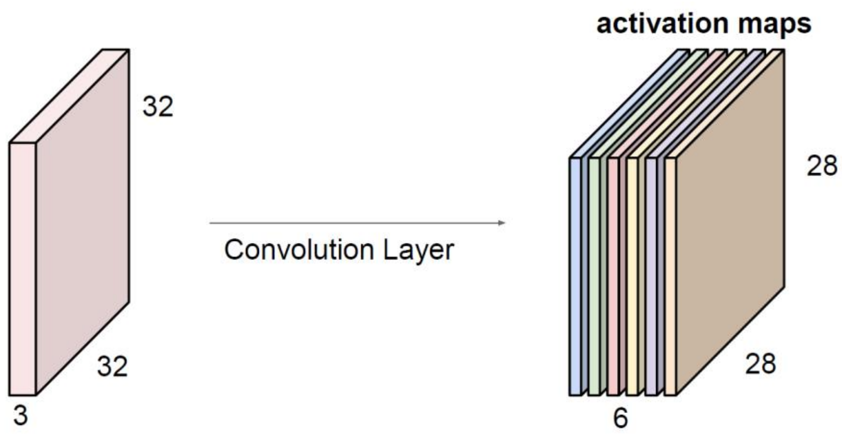
Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolutional_neural_networks.html

Convolutional Neural Network
Multiple kernels

- A convolution layer usually contains **multiple kernels**
- Each kernel produces a separate 2 dimensional feature map
- Stacking these feature maps, one can get the output volume of the convolution layer

- The number of kernels (the depth of the output volume) is defined by the **depth hyperparameter**

Convolution with 6 different kernels:



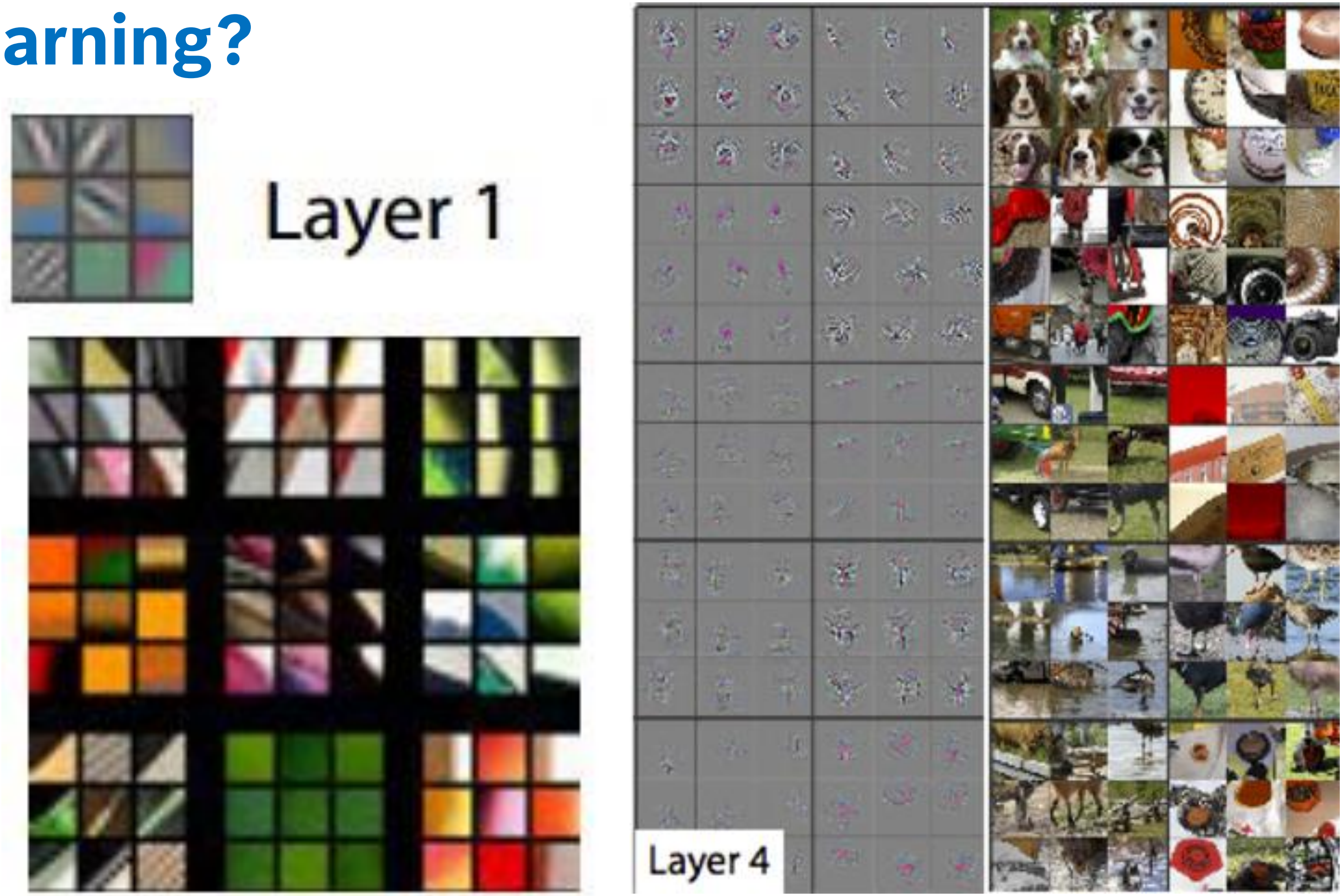
Input 3D matrix

Output - 3D matrix
6 different feature maps

Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolutional_neural_networks.html

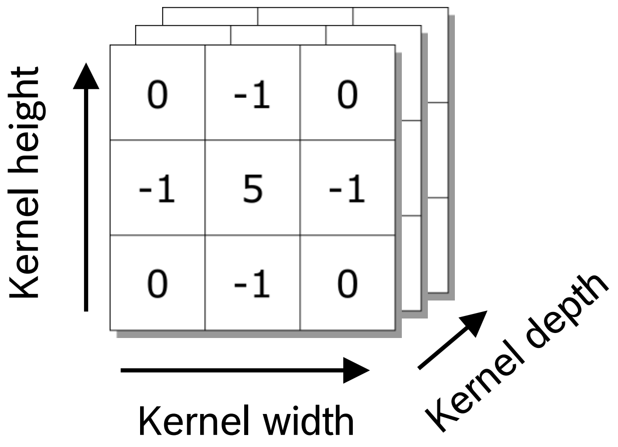
Convolutional Neural Network
What are the filters learning?

- ▶ 1st layer: edges
- ▶ 2nd layer: corners, local textures
- ▶ 3rd layer: simple shapes
- ▶ ...
- ▶ nth layer: complex shapes, objects



Source: Zeiler & Fergus, 2014

Convolutional Neural Network
Convolutional Layer - Summary

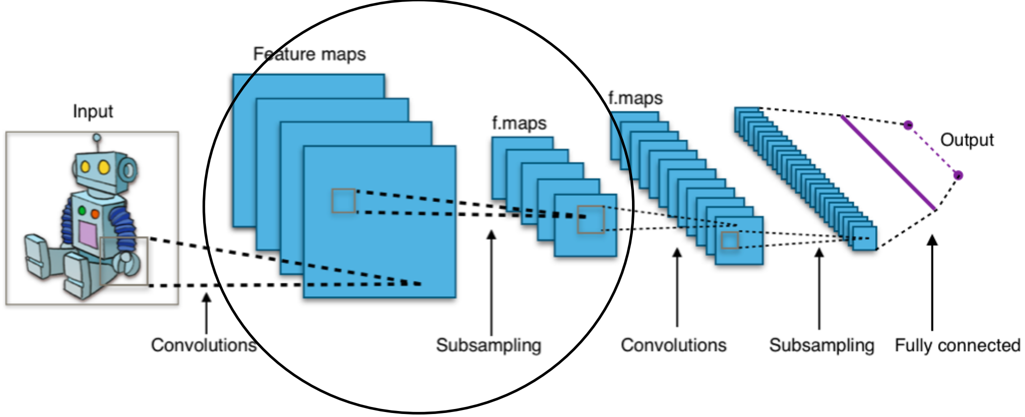


Convolution:

- Input: W_1 x H_1 x D_1 3D matrix
- Output: W_3 x H_3 x D_3 3D matrix

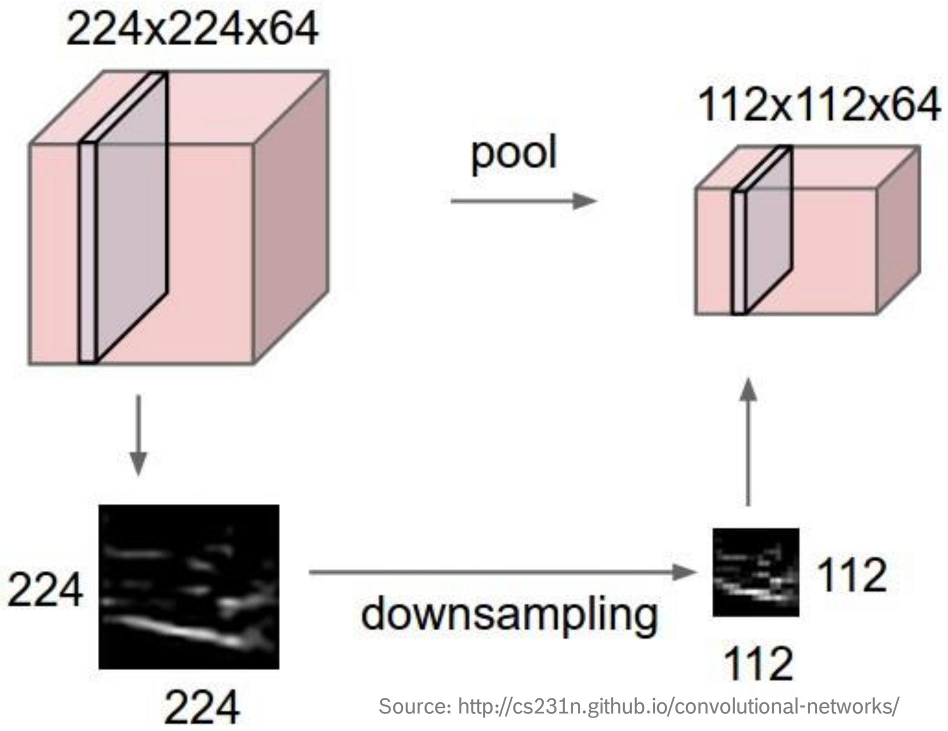
- Hyperparameters of the convolutional layer:
  - Size of the kernel: $\mathbf{W_2}$ **x** $\mathbf{H_2}$ (the depth of the kernel is equal with the input depth)
  - Number of kernels (depth of the output): $\mathbf{D_3}$
  - Stride: **S**
  - Padding: **P**

Convolutional Neural Network
Subsampling (Pooling) Layer



- Perform a downsampling operation along the spatial dimensions – width, height
- It reduces the spatial size of the representation

- It operates independently on every depth slice of the representation
- Most common subsampling operation: **max pooling**

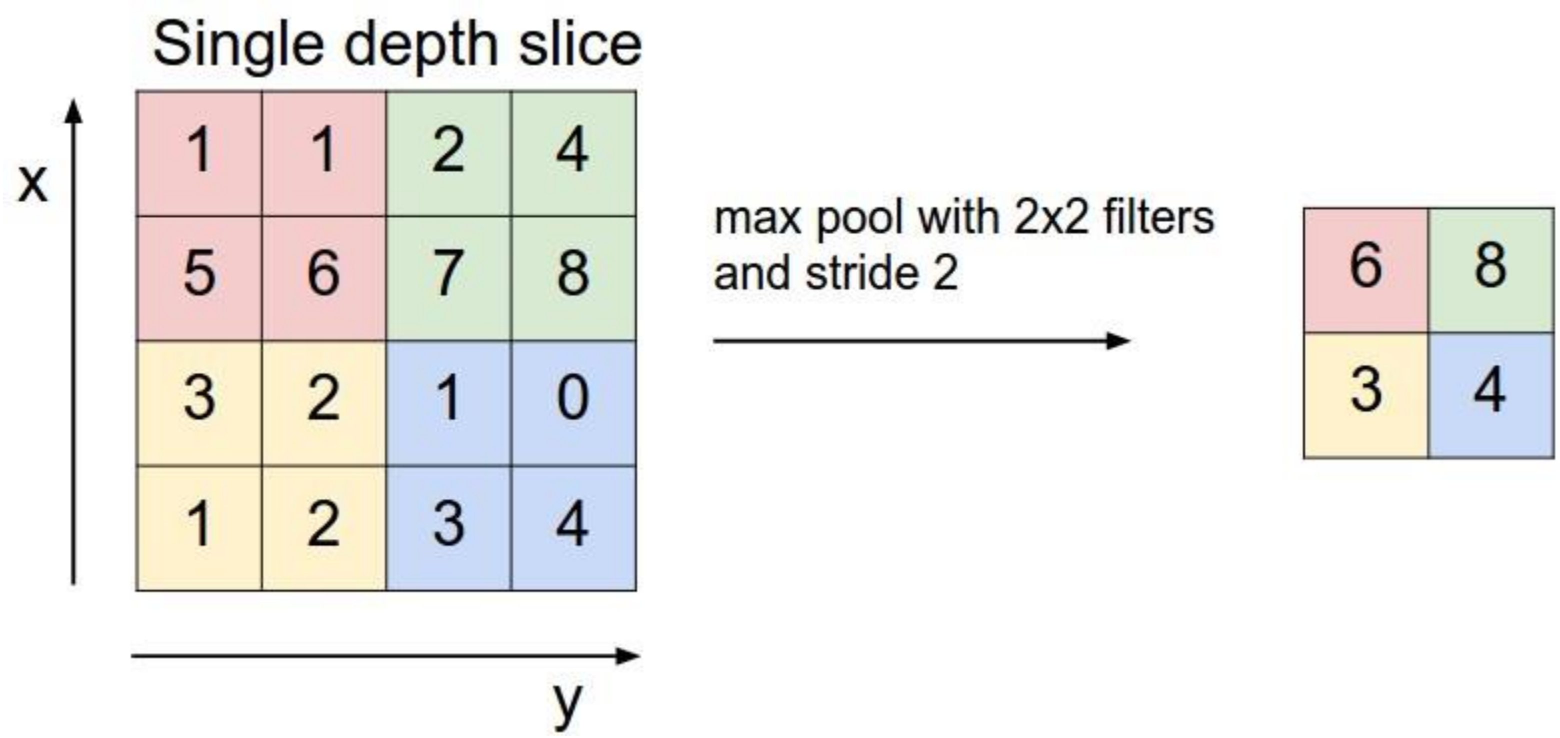


Source: http://cs231n.github.io/convolutional-networks/

Convolutional Neural Network
Example: max pooling

Hyperparameters:

- Size of stride
- Size of the filter

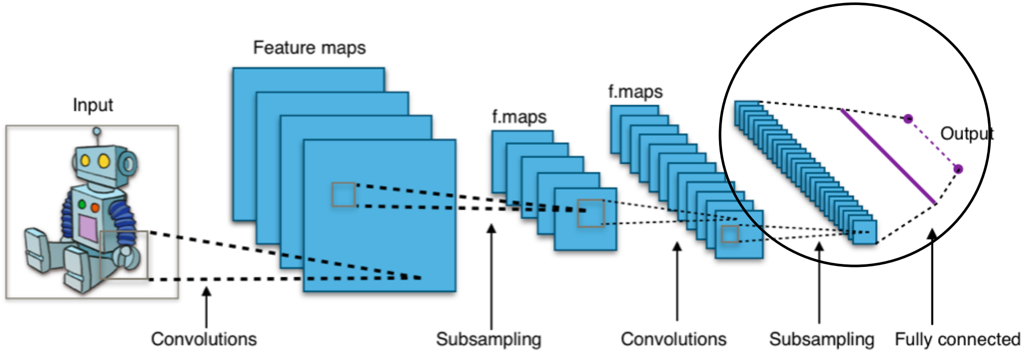


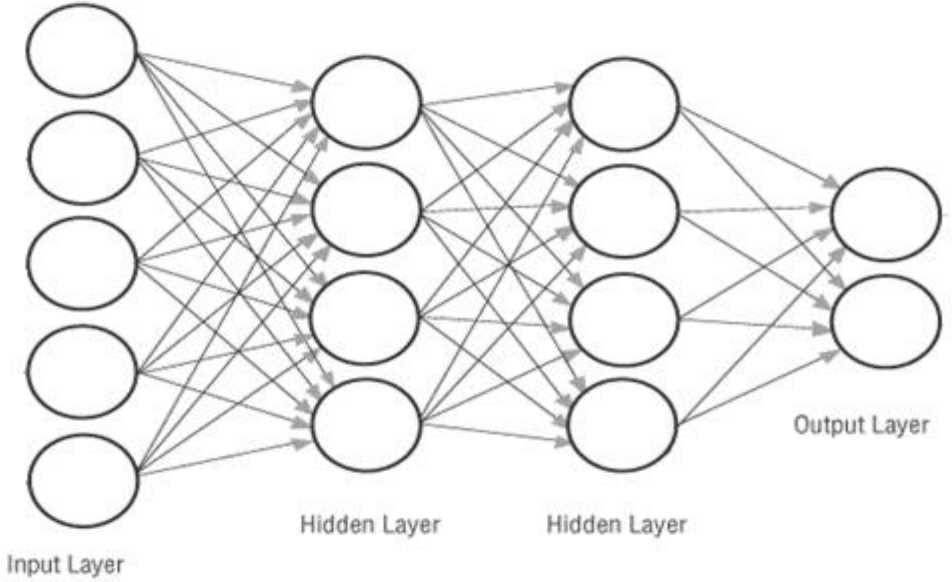
Source: http://cs231n.github.io/convolutional-networks/

BOSCH

Convolutional Neural Network
Fully Connected Layer

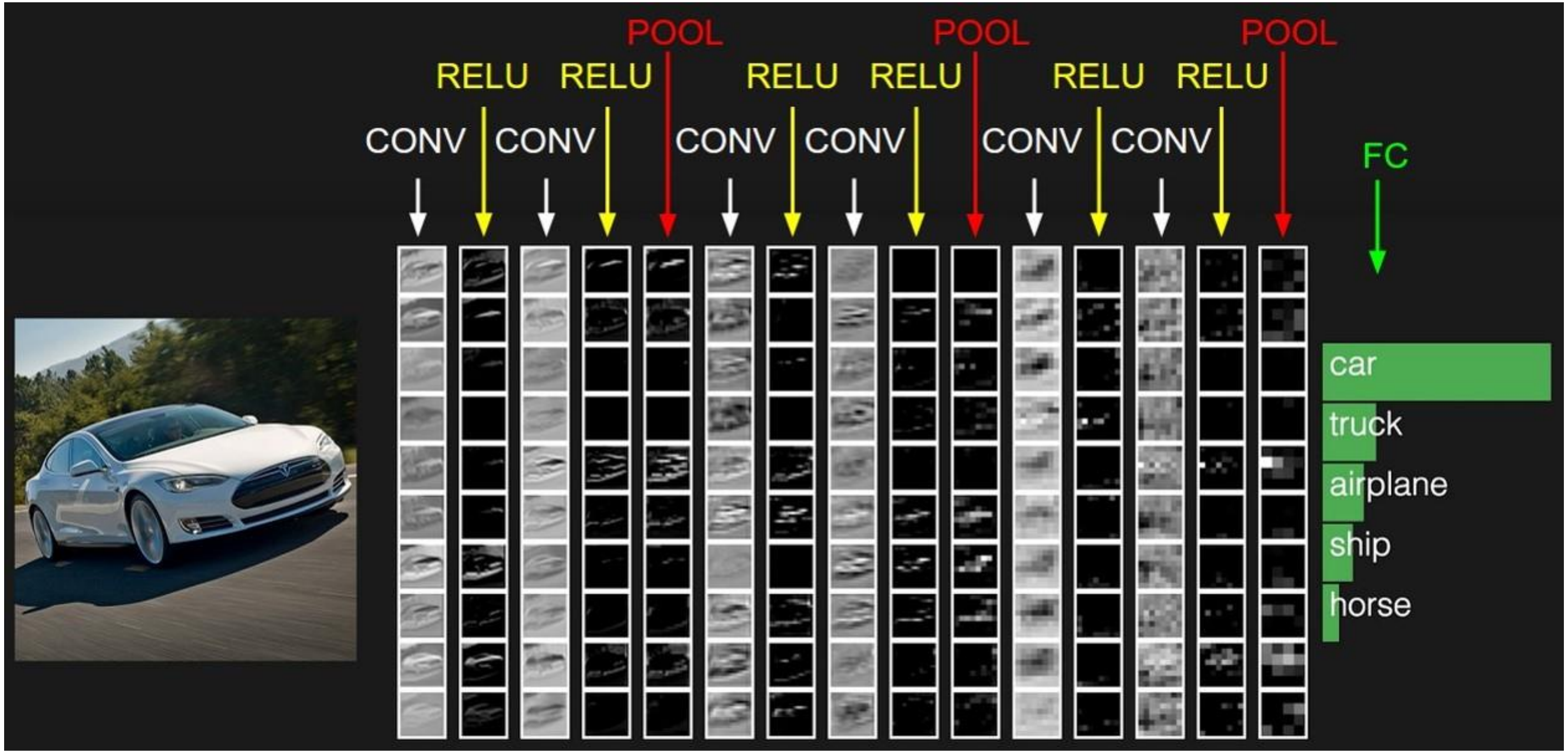
- It is a neural network (similar as in the previous course)
- The **input** of the layer are the features extracted by the convolution and subsampling layers
- The **output** of the layer is the output of the full CNN
(for example class probabilities)





Source: http://cs231n.github.io/convolutional-networks/

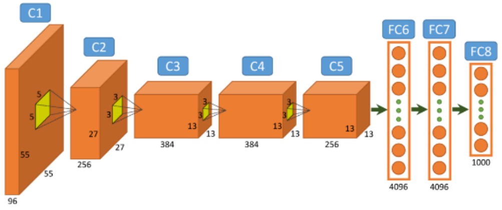
Convolutional Neural Network
Information flow in a CNN

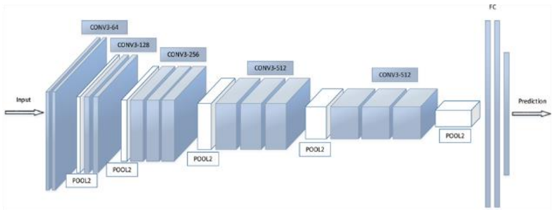


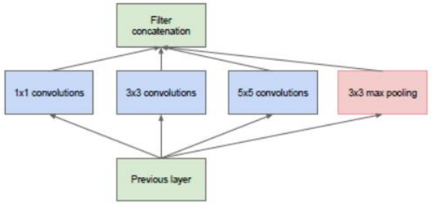
Source: http://cs231n.github.io/convolutional-networks/

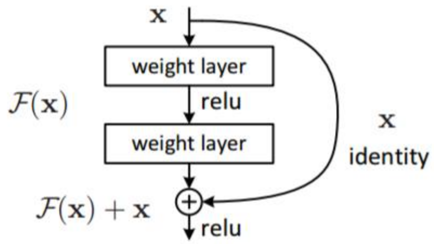
Convolutional Neural Network
History and state of the art

- **AlexNet**
  - Initial architecture that had good performance (2012)
  - 7 layers
- **VGGNet**
  - Better performance using deeper network with less parameters (2014)
  - 16 layers
- **GoogLeNet**
  - Better performance using processing done in parallel on same input (2014)
  - Over 100 layers
- **ResNet (Microsoft)**
  - Better performance using residual information (2015)
  - 152 layers





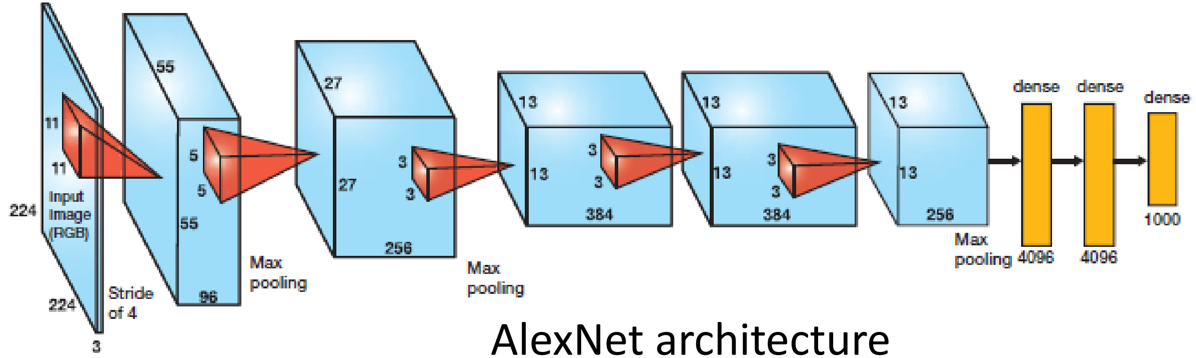




- Models can be trained to perform many different tasks, with few modifications

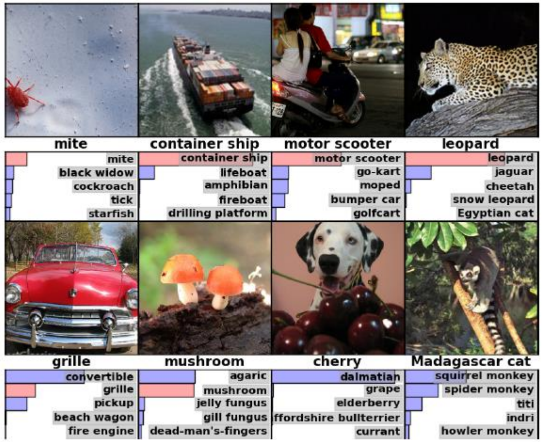
Sources: http://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/
https://www.saagie.com/blog/object-detection-part1
http://file.scirp.org/Html/4-7800353_65406.htm

Convolutional Neural Network
Case study: AlexNet



AlexNet architecture

Train images



Results on test images

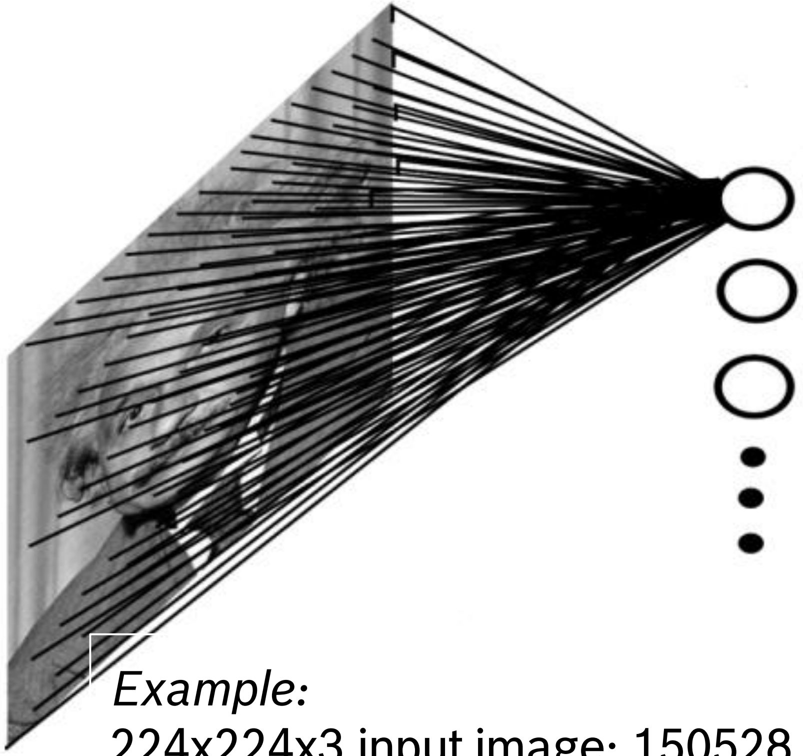
- ImageNet Large Scale Visual Recognition Challenge winner in 2012
- Task: images classification (each image is associated with a class)
- Dataset:
  - ImageNet 2012
  - Training set: 1.2 million images containing 1000 categories (classes)
  - Testing set: 200.000 images
- Training details:
  - 90 full training cycle on the training set
  - The **training took 6 days** on two GeForce 580

Sources: http://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

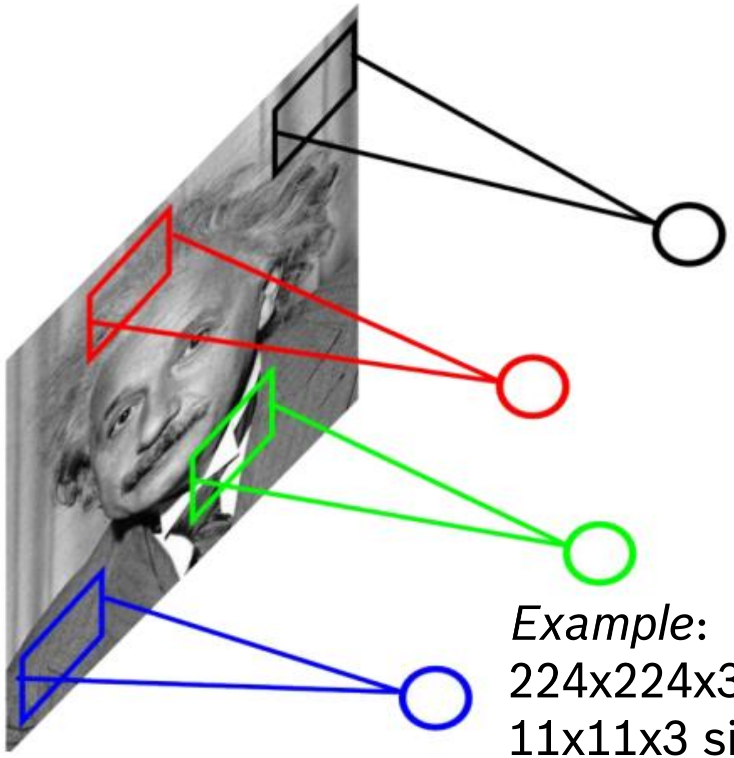
MOTIVATION OF CONVOLUTIONAL NEURAL NETWORK

BOSCH

Motivation of Convolutional Neural Network
Parameter sharing



Example:
224x224x3 input image: 150528 long input
=> 150528 weights per neuron
150528*3 (451584) parameters (for 3 neurons)

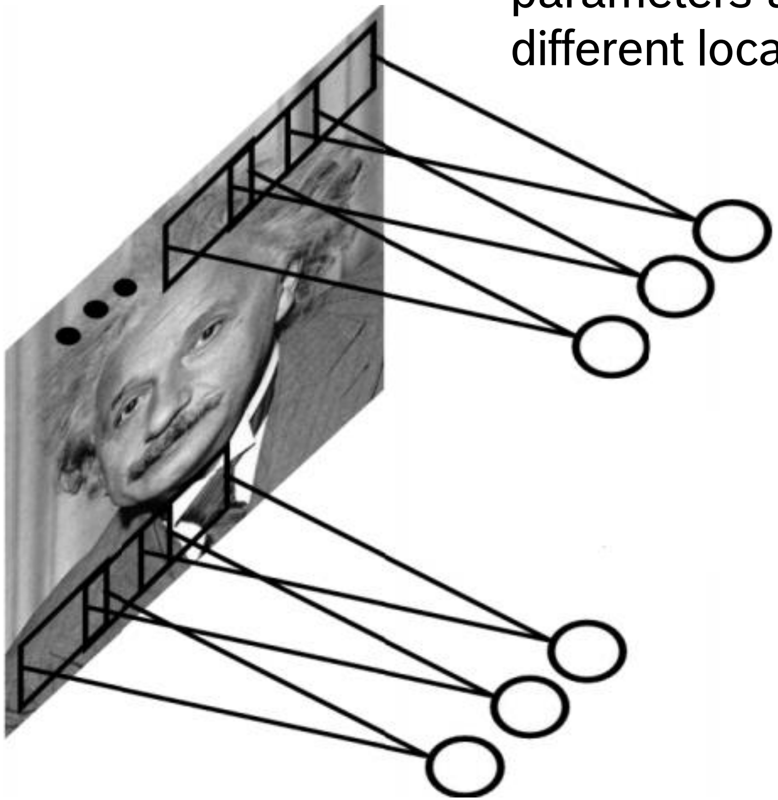


Example:
224x224x3 input image
11x11x3 size of a filter
3 filters
11*11*3*3 (1089) parameters (for 3 filters)

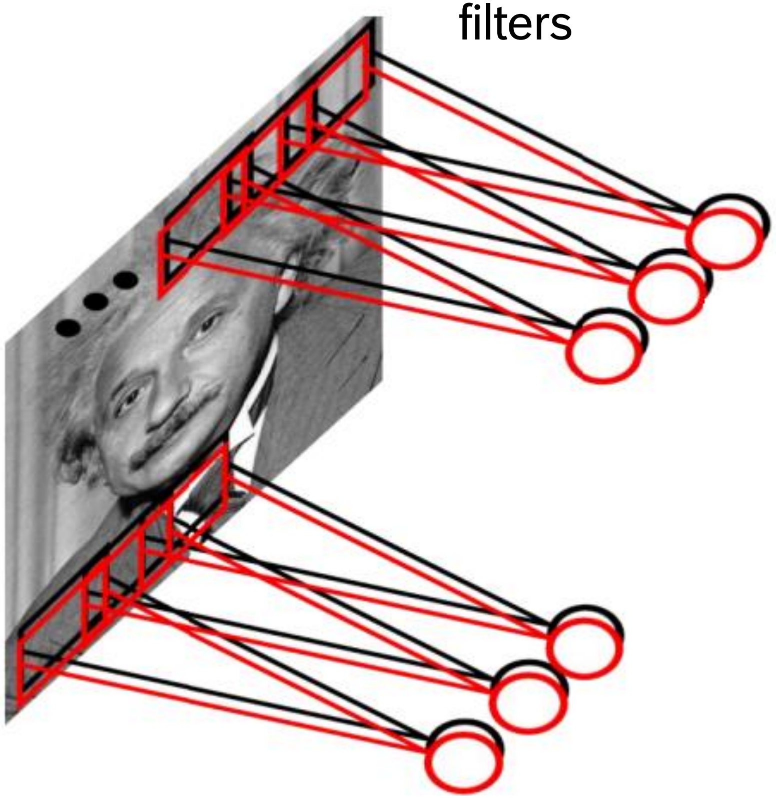
Source: https://noppa.oulu.fi/noppa/kurssi/521010j/luennot/521010J_convolutional_neural_network.pdf

Motivation of Convolutional Neural Network
Local connectivity and spatial invariance

Share the same parameters across different locations



Apply and learn multiple filters



Source: https://noppa.oulu.fi/noppa/kurssi/521010j/luennot/521010J_convolutional_neural_network.pdf

Motivation of Convolutional Neural Network
Benefits

- The same small set of weights (parameters) is applied on an entire image
- Better training. Better generalization
- Preserves spatial relationships in the receptive field
- Spatial invariance

Parameter sharing

Local connectivity

BOSCH

CONCLUSION

Conclusion
And some tips

Strengths of deep learning:

- It's a general framework
- Efficient graph structure
- Well performing given the right circumstances

You should consider deep learning if:

- you have access to quite large amounts of data
- the problem is reasonably complex, in a high dimensional space
- no hard constraints or hard logic (smooth, differentiable)

RESOURCES

Resources
Libraries and networks

- C++ fans for easy prototyping:
  - OpenCV – both Neural Networks and Adaboost
  - FANN – Fast Artificial Neural Network Library
  - OpenNN – Open Neural Network Library
  - Nnabla – Neural network libraries by Sony

- Python:
  - **Tensorflow**
  - Keras
  - Theano
  - **Caffe(also for C++)**
  - **Torch7**
  - **PyTorch**
  - **DeepLearning4J**
  - **MXNet**
  - Deepy
  - Lasagne
  - Nolearn
  - NeuPy

Resources

Learning resources

- Stanford CS231n course (Karpathy et. al.)
- Deep learning book (Goodfellow)
- various introductory YouTube videos
- For beginners: www.reddit.com/r/LearnMachineLearning
- After you grasp the concepts: www.reddit.com/r/MachineLearning

THANK YOU

QUESTIONS?