PRACE Workshop: Deep Learning and GPU programming workshop

7 – 10 September 2020

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Preliminaries on Convolutional and Recurrent Neural Networks

07.09.2020 | PD Dr. Juan J. Durillo

Agenda

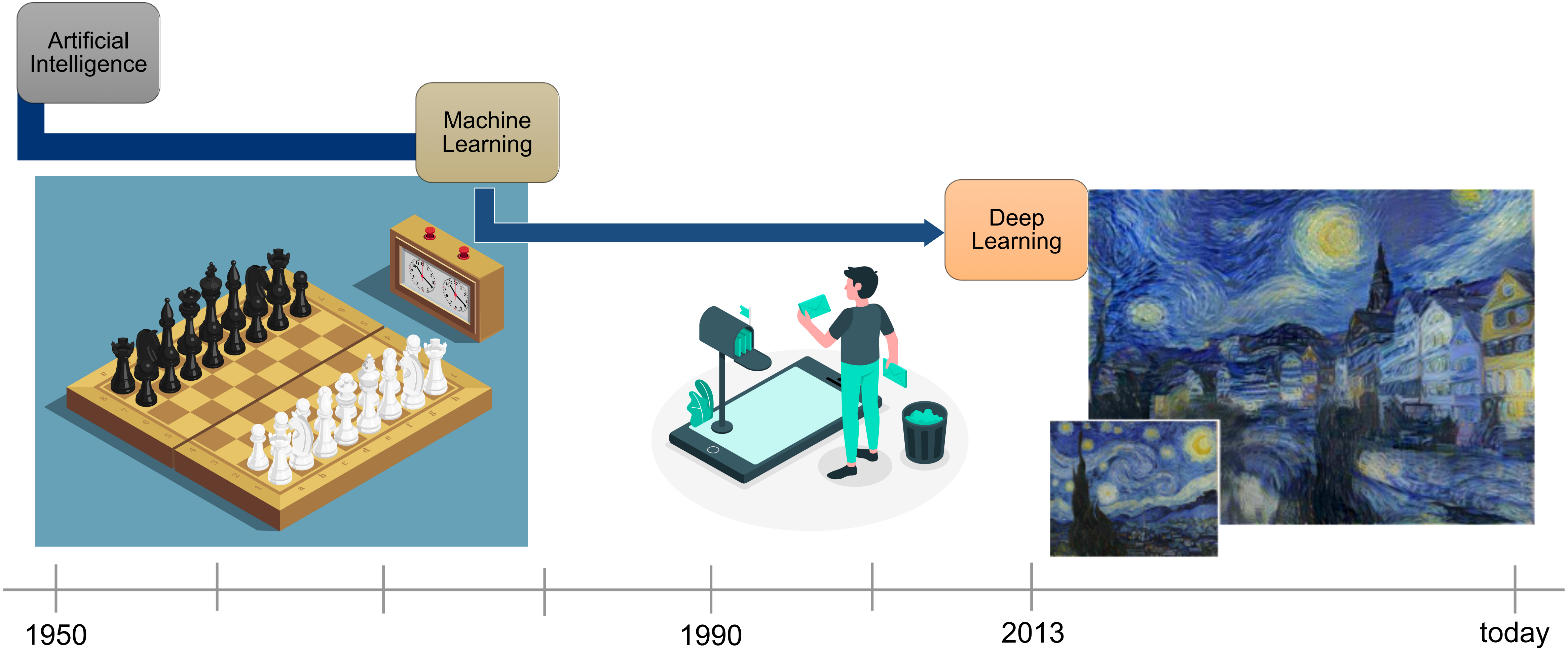
- Part 1
  - Introduction
  - Introduction to (Deep) Neural Networks for Machine Learning
  - Computer Vision as working example
  - Introduction to Convolutional Neural Networks
  - Deep Neural Network Architecture

- Part 2
  - More than Images
  - Representing Languages
  - Recurrent Neural Networks

Part 1

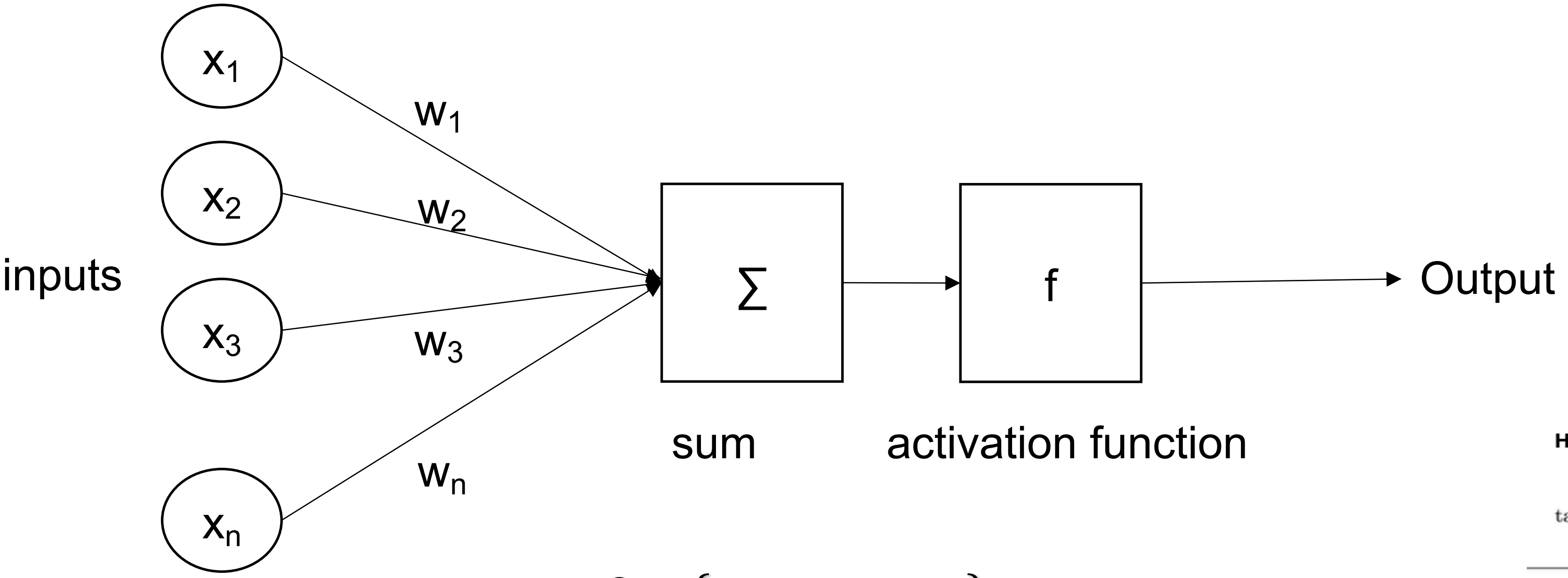
Artificial + Intelligence

PRACE NVIDIA DEEP LEARNING INSTITUTE lrz

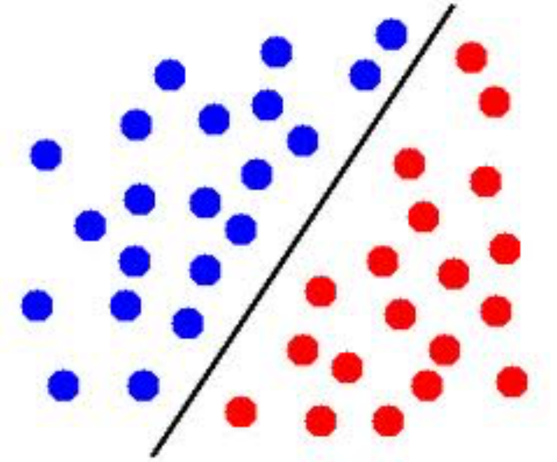


CSC VSB TECHNICAL UNIVERSITY OF OSTRAVA IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

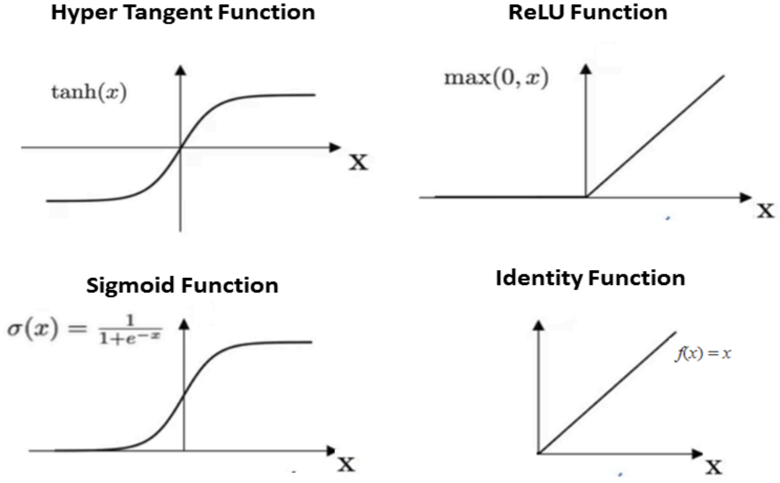
Perceptron – Artificial Neuron



$$\Theta = \{w_1, w_2 \dots, w_n\}$$

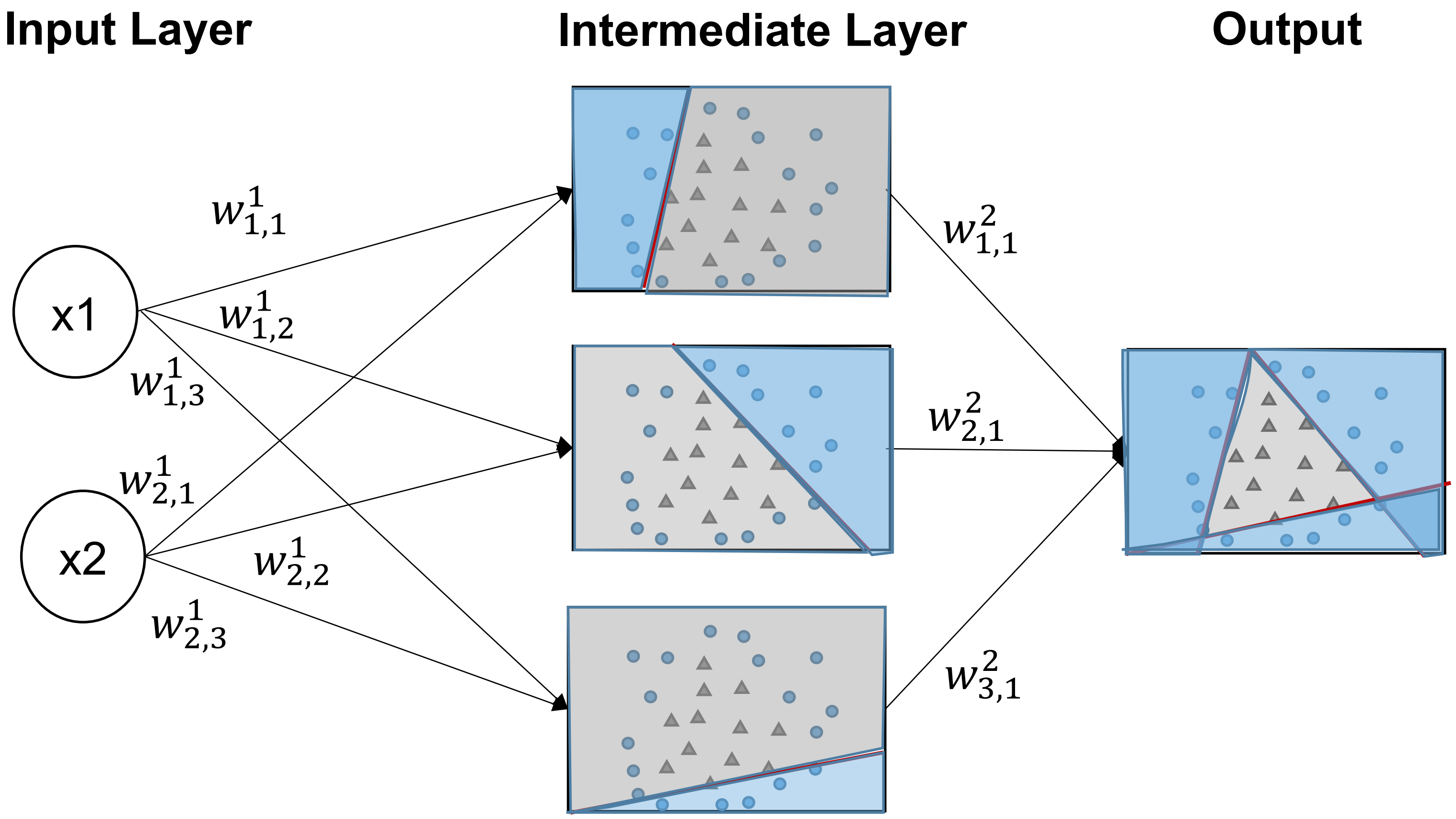


most popular activation functions



Single artificial neurons work well for linearly separable datasets (indeed output is the activation effect on a linear combination of the input)

Neural Network



$$\Theta = \{w^1_{1,1}, w^1_{1,2}, w^1_{1,3}, w^1_{2,1}, w^1_{2,2}, w^1_{2,3}, w^2_{1,1}, w^2_{2,1}, w^2_{2,3}\}$$

- Even when the data is not linearly separable

(Supervised) Learning

- Data domain Z: $X \times \Upsilon$

  X → domain of the input data

  Υ → set of labels (knowledge)

X: 32 x 32 color images

Υ : labels

{truck, car, horse, bird, boat}

Example (CIFAR10 dataset)

- Data Distribution is a probability distribution over a data domain
- Training set $z_1, \ldots, z_n$ from Z assumed to be drawn from the Data Distribution D
- Validation set $v_1, \ldots, v_m$ from Z also assumed to be drawn from D
- A machine learning model is a function that given a set of parameters Θ and z from Z produces a prediction
- The prediction quality is measured by a differentiable non-negative scalar-valued loss function, that we denote $\ell(\Theta; z)$

(Supervised) Learning

- Given Θ we can define the expected loss as: $L(\Theta) = \mathbb{E}_{z \sim D}[\ell(\Theta; z)]$
- Given D, ℓ, and a model with parameter set Θ, we can define learning as:
  "The task of finding parameters Θ that achieve low values of the expected loss, while we are given access to only n training examples"
- The mentioned task before is commonly referred to as *training*
- Empirical average loss given a subset of the training data set S(z_1, …, z_n) as:

$$\hat{L}(\Theta) = \frac{1}{n}\sum_{t=1}^{n}[\ell(\Theta; z_t)]$$

- Usually a proxy function, easier to understand by humans, is used for describing how well the training is performed (e.g., accuracy)

(Supervised) Learning

- The dominant algorithms for training neural networks are based on mini-batch stochastic gradient descent (SGD)

- Given an initial point Θ_0 SGD attempt to decrease $\hat{L}$ via the sequence of iterates

$$\Theta_t \longleftarrow \Theta_{t-1} - n_t g(\Theta_{t-1}; B_t)$$

$$g(\Theta; B) = \frac{1}{|B|} \sum_{z \epsilon B} \nabla \ell(\Theta; z)$$

Definitions

B_t: random subset of training examples

n_t: positive scalar (learning rate)

epoch: update the weights after going over all training set

Computer Vision

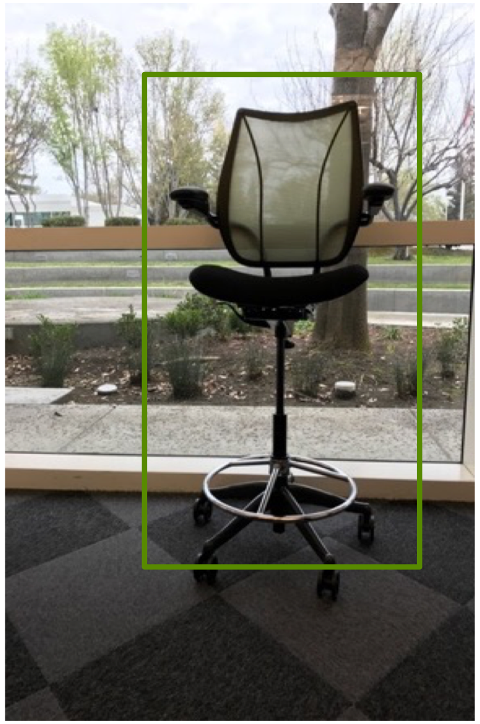
- Why? Focus on a kind of Deep Neural Network called Convolutional Neural Network (CNN)
- CNNs ability to extract multi-scale localized spatial features and compose them to construct highly expressive representations led to breakthroughs in almost all machine learning areas

COMPUTER VISION TASKS

Image Classification

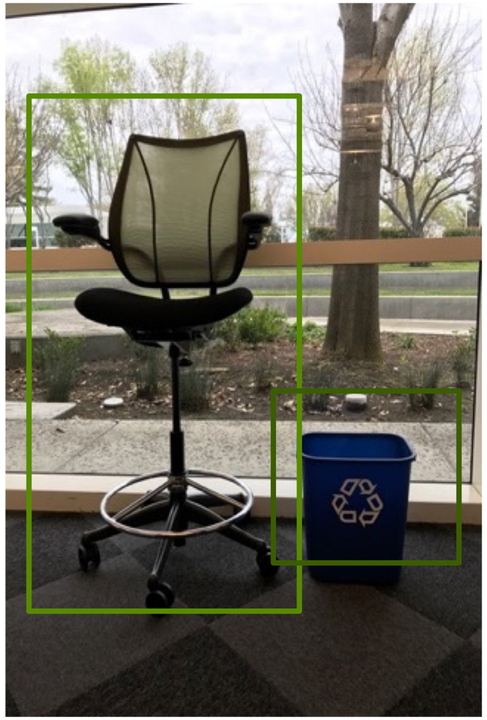
predicting the type or class of an object in an image

Image Classification + Localization



predicting the type or class on an object in an image and draw a bounding box around it

Object Detection



predicting the location of objects in an image via bounding boxes and the classes of the located objects

Image Segmentation

predicting the class to which each pixel in the image belongs to

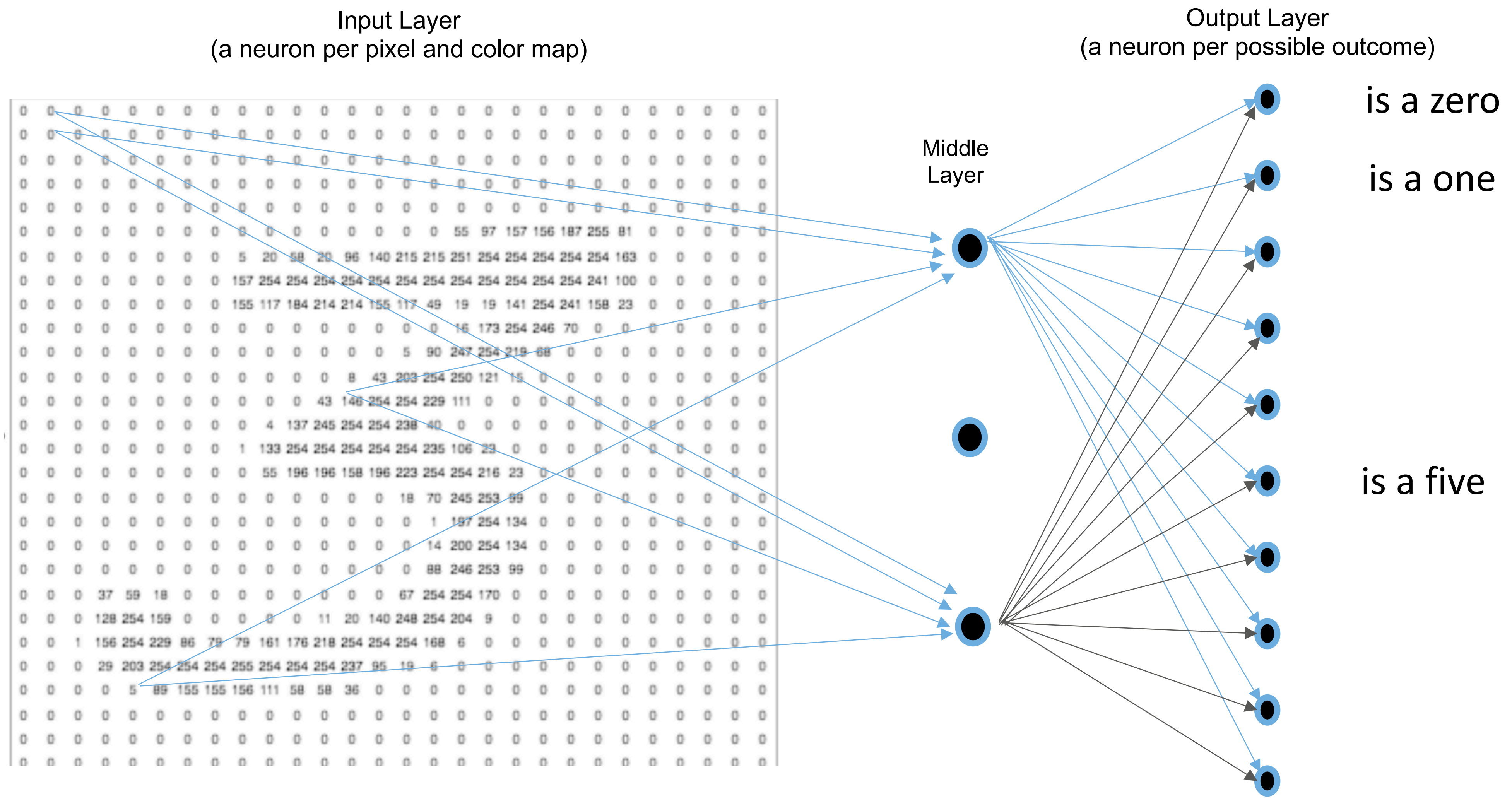
On Input Representation

28 x 28
= 784 pixels

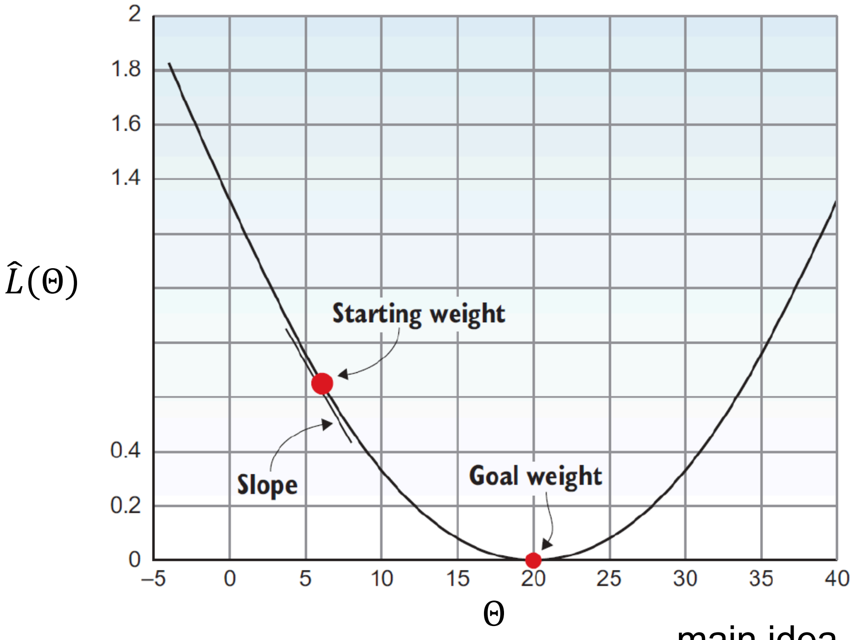
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	97	157	156	187	255	81	0	0	0	0	0
0	0	0	0	0	0	0	0	5	20	58	20	96	140	215	215	251	254	254	254	254	254	163	0	0	0	0	0
0	0	0	0	0	0	0	0	157	254	254	254	254	254	254	254	254	254	254	254	254	241	100	0	0	0	0	0
0	0	0	0	0	0	0	0	155	117	184	214	214	155	117	49	19	19	141	254	241	158	23	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	173	254	246	70	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	90	247	254	219	68	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	8	43	203	254	250	121	15	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	43	146	254	254	229	111	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	137	245	254	254	238	40	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	133	254	254	254	254	254	235	106	23	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	55	196	196	158	196	223	254	254	216	23	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	70	245	253	99	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	197	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	200	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	246	253	99	0	0	0	0	0	0	0	0	0
0	0	0	37	59	18	0	0	0	0	0	0	0	0	67	254	254	170	0	0	0	0	0	0	0	0	0	0
0	0	0	128	254	159	0	0	0	0	0	11	20	140	248	254	204	9	0	0	0	0	0	0	0	0	0	0
0	0	1	156	254	229	86	79	79	161	176	218	254	254	254	168	6	0	0	0	0	0	0	0	0	0	0	0
0	0	0	29	203	254	254	254	255	254	254	254	237	95	19	6	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	89	155	155	156	111	58	58	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

image

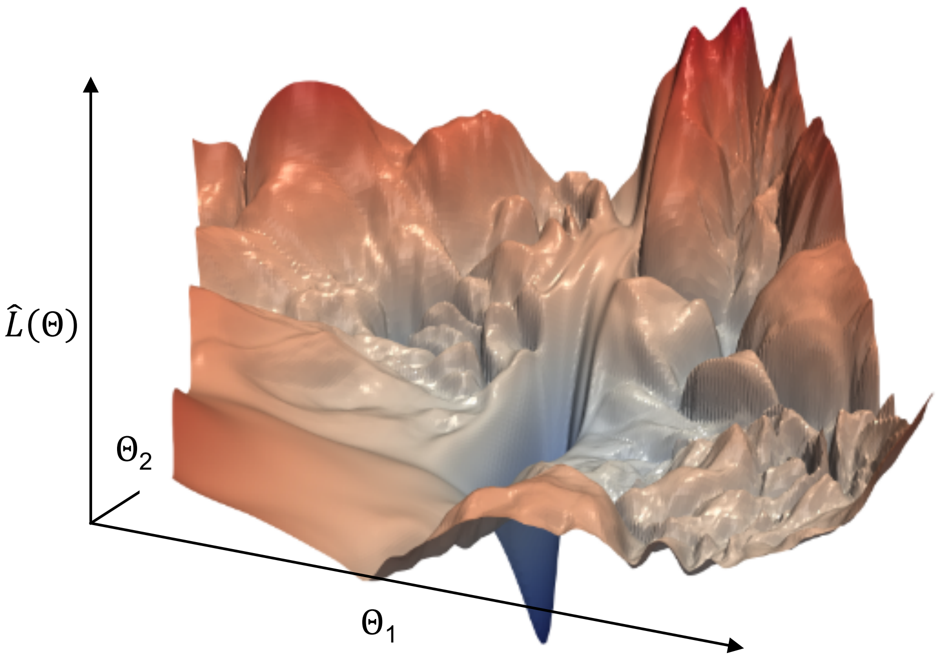
Neural Networks for Image Classification



Training Neural Networks



main idea



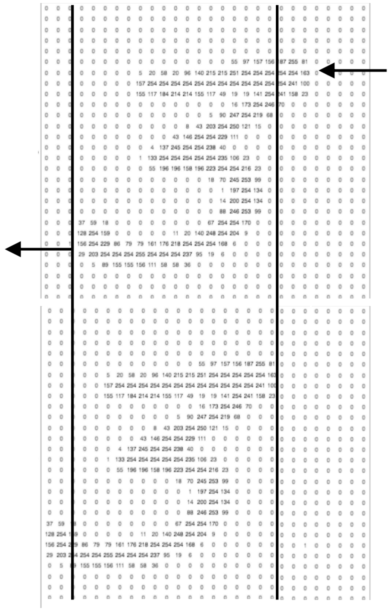
how the surface looks like in reality

Stochastic Gradient Descent

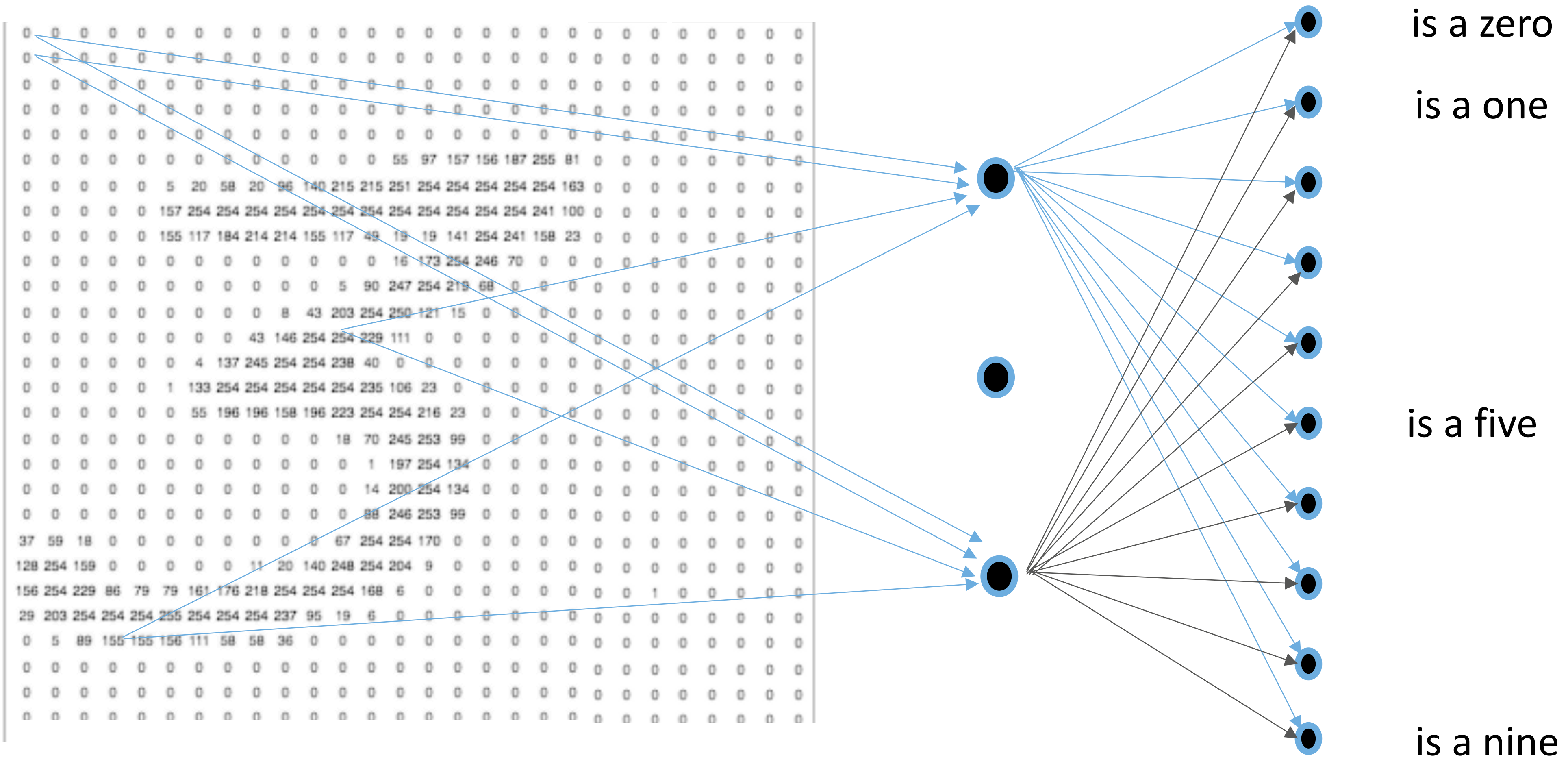
$$\Theta_t \leftarrow \Theta_{t-1} - n_t g(\Theta_{t-1}; B_t)$$

$$g(\Theta; B) = \frac{1}{|B|} \sum_{z \epsilon B} \nabla \ell(\Theta; z)$$

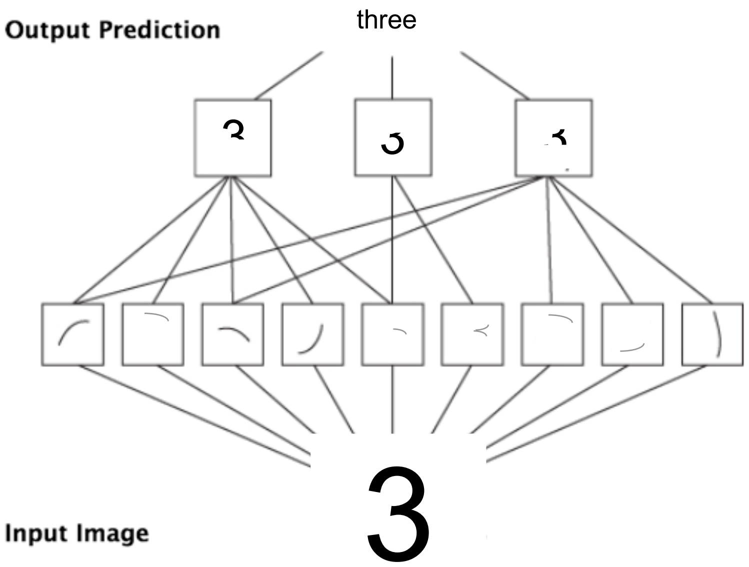
Neural Networks for Image Classification



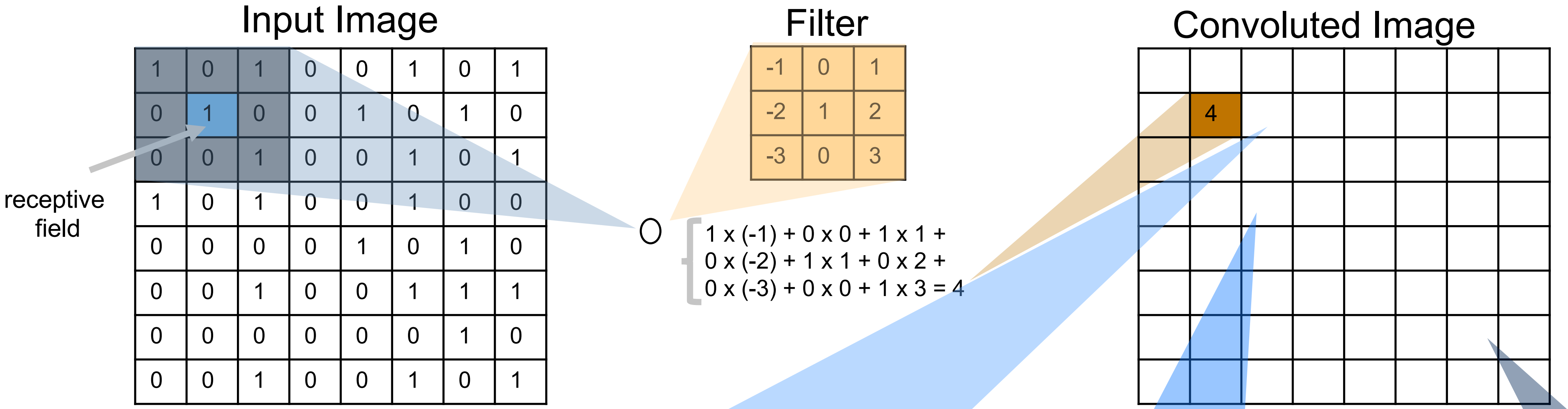
shift to the left



No More Feature Engineering



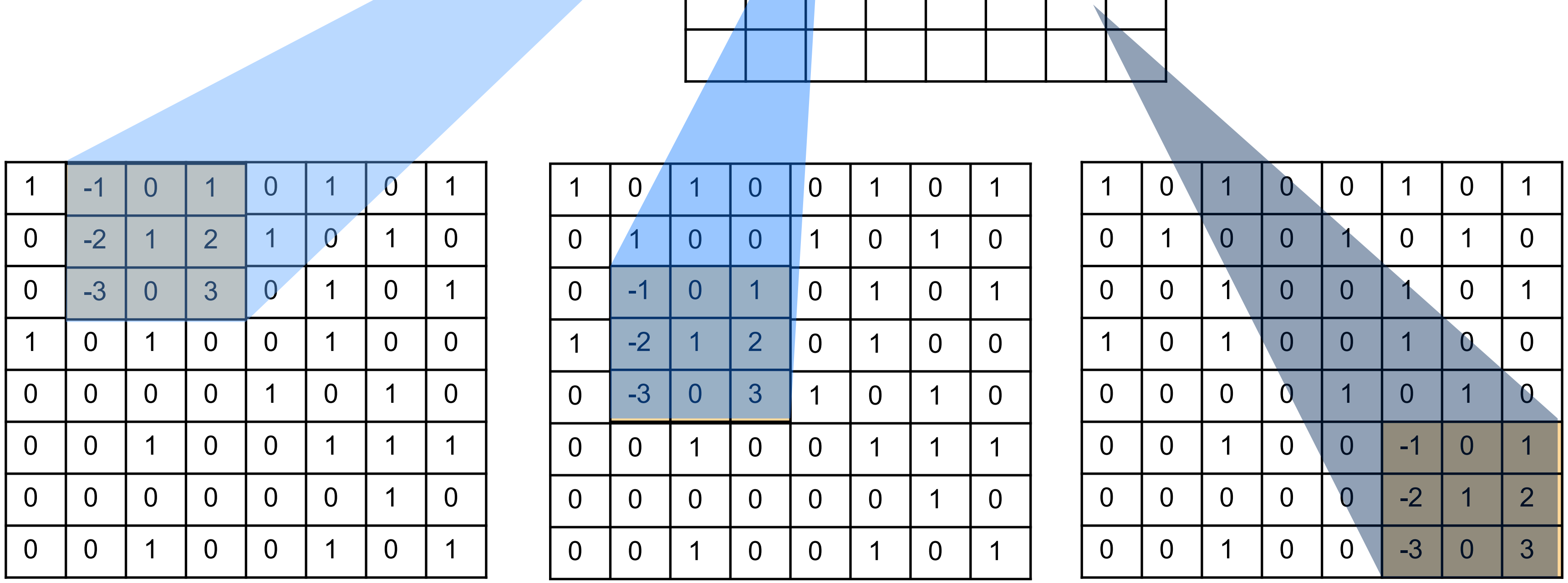
Learning features from data: Convolutions



Filter is convoluted with all the pixels of the image

How many units the filter moves horizontally or vertically is called **stride** and can be different in both dimensions

The stride defines the size of the convoluted image



Filters

Input Image:

Can we get only vertical lines out of this picture?

1	0	-1

filter 1

1	0	-1
1	0	-1
1	0	-1

filter 2

1	0	0	0	-1
1	0	0	0	-1
1	0	0	0	-1
1	0	0	0	-1
1	0	0	0	-1

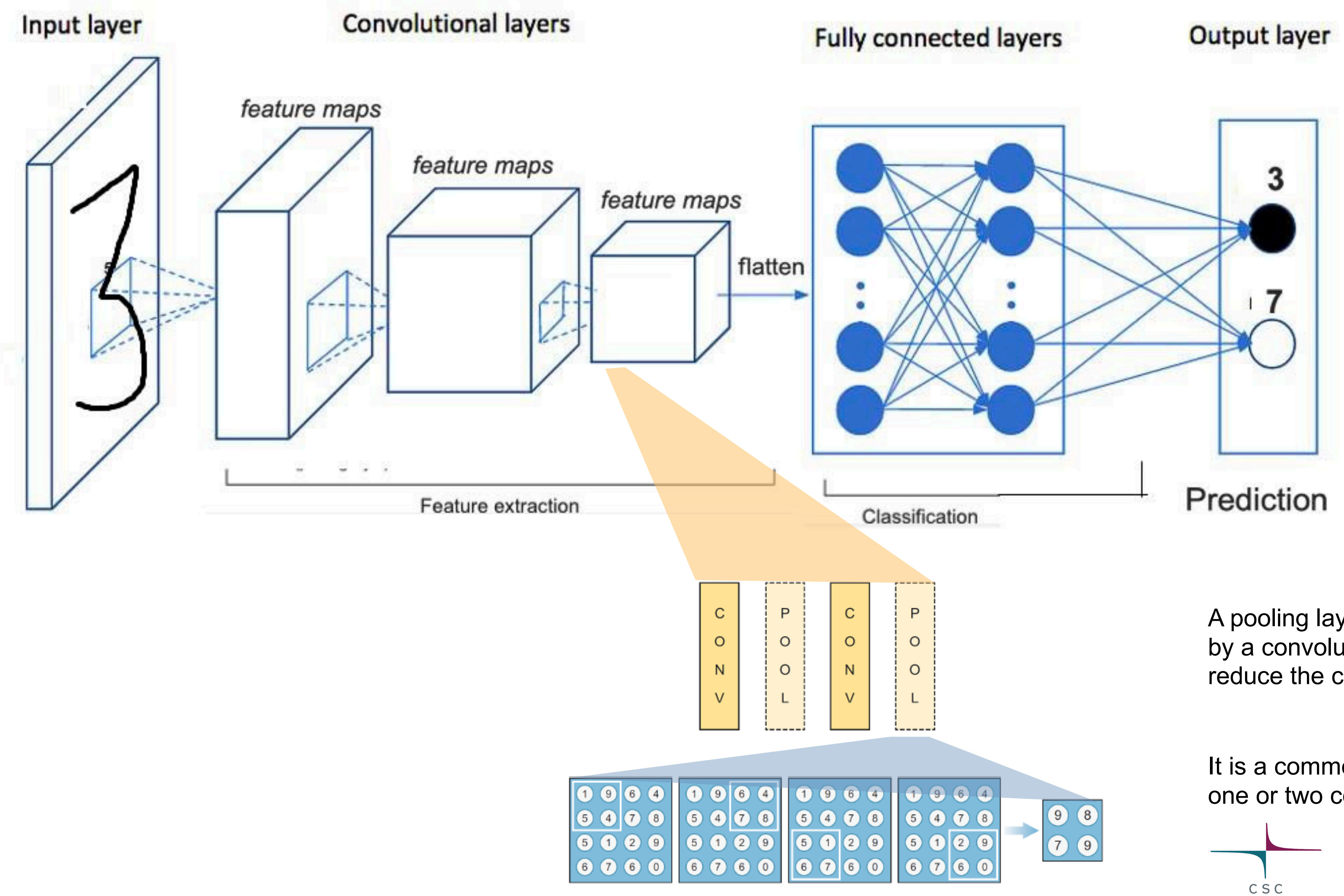
filter 3

try the code yourself (in octave)!

```
I=imread(<path-to-image>);
GRAY=rgb2gray(I)
FILTER=[ 1 0 -1; 1 0 -1; 1 0 -1]; % filter 2
CONVOLUTED=conv2(GREY,FILTER);
Imwrite(CONVOLUTED, <path-to-result>);
```

*The London skyline image is designed by Freepik

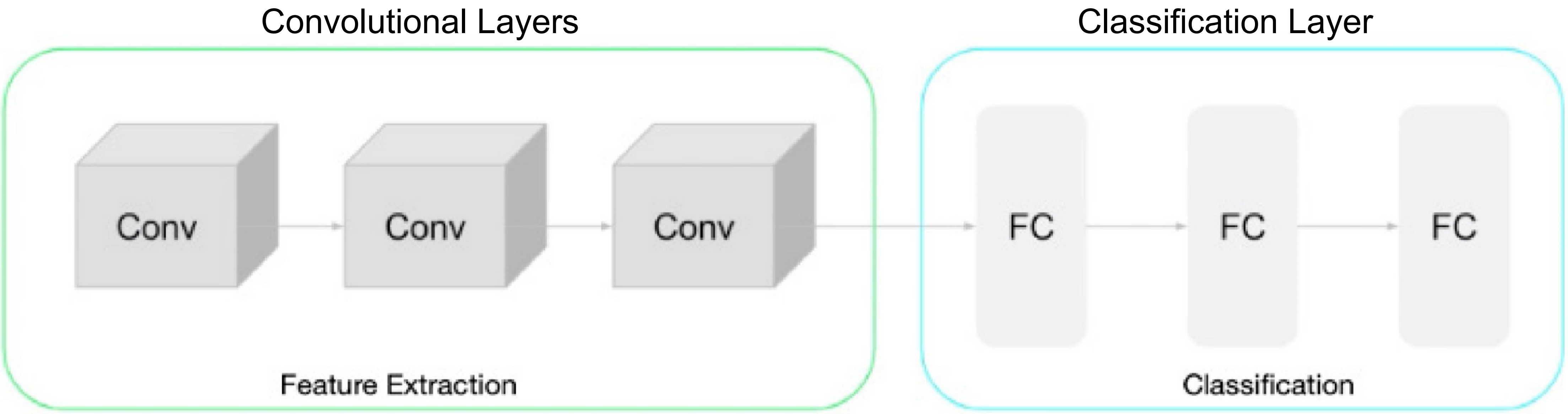
Convolutional Neural Networks (CNN)



A pooling layer down sample the feature maps produced by a convolution into smaller number of parameters to reduce the computational complexity.

It is a common practice to add pooling layers after each one or two convolutions layers in the CNN architecture.

CNN Architecture: A Common Pattern and its Influence

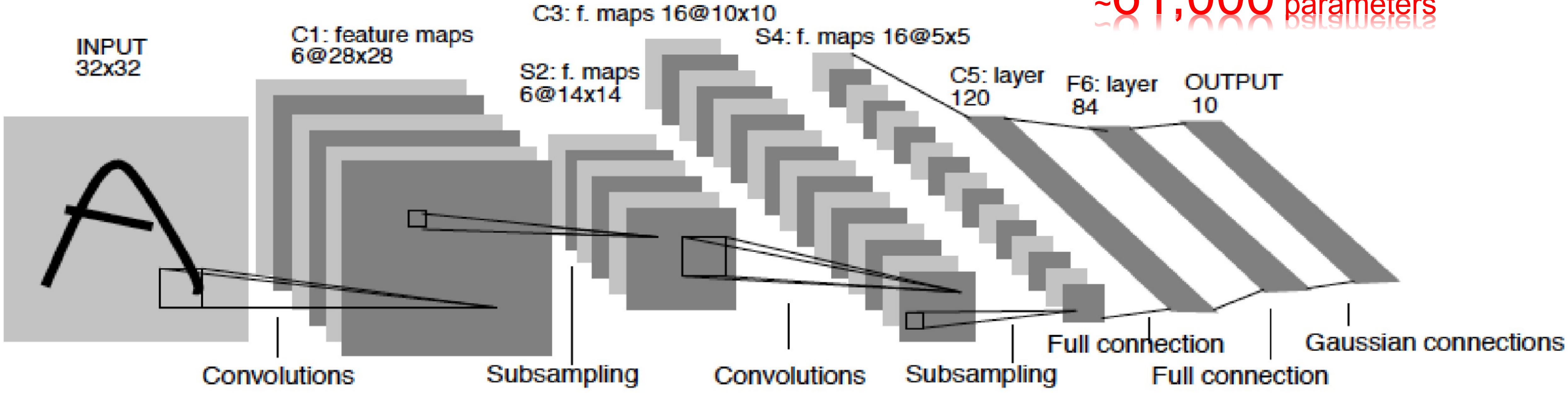


The execution time required during a forward pass through a neural network is bounded from below by the number of floating point operations (FLOPs).

This FLOP count depends on the deep neural network architecture and the amount of data.

LeNet Architecture

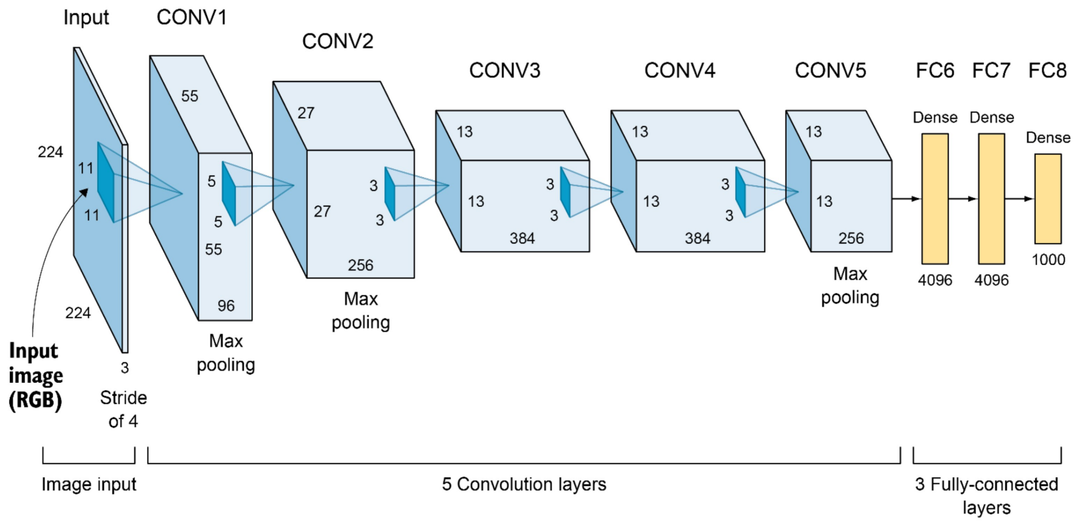
~61,000 parameters



Architecture summary :

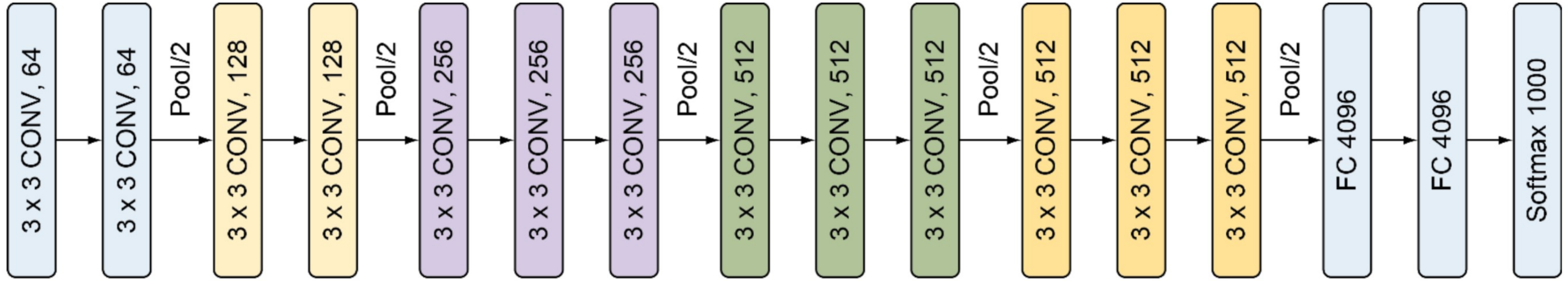
- 3 convolutional layers filters in all the layers equal to 5x5
  (layer 1 depth = 6, layer 2 depth = 16, layer 3 depth = 120)
- As activation function the tanh function is used

AlexNet and VGG Architectures



AlexNet

~60,000,000 parameters

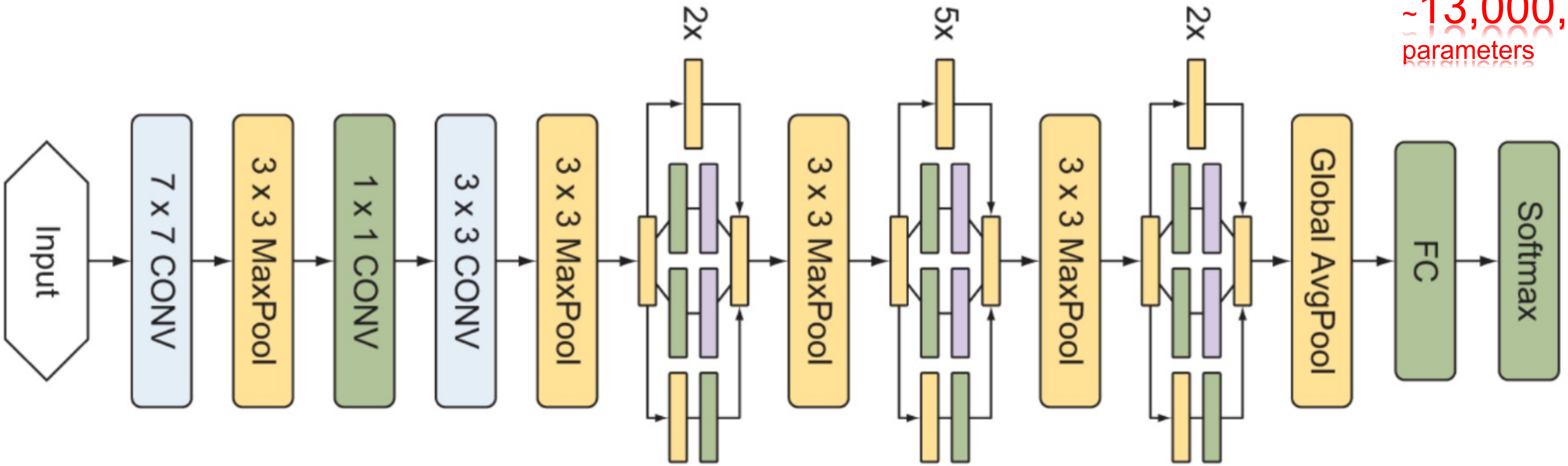


VGG16

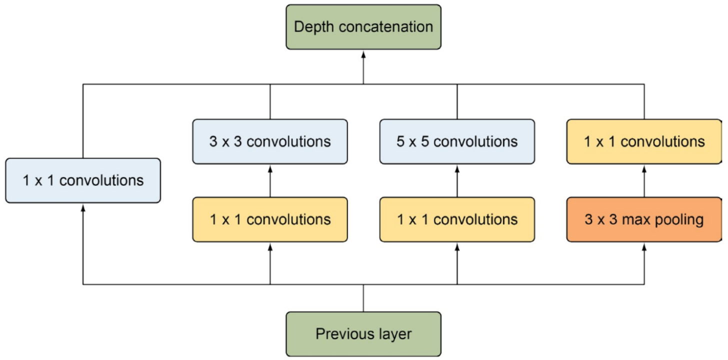
~138,000,000 parameters

GoogleNet

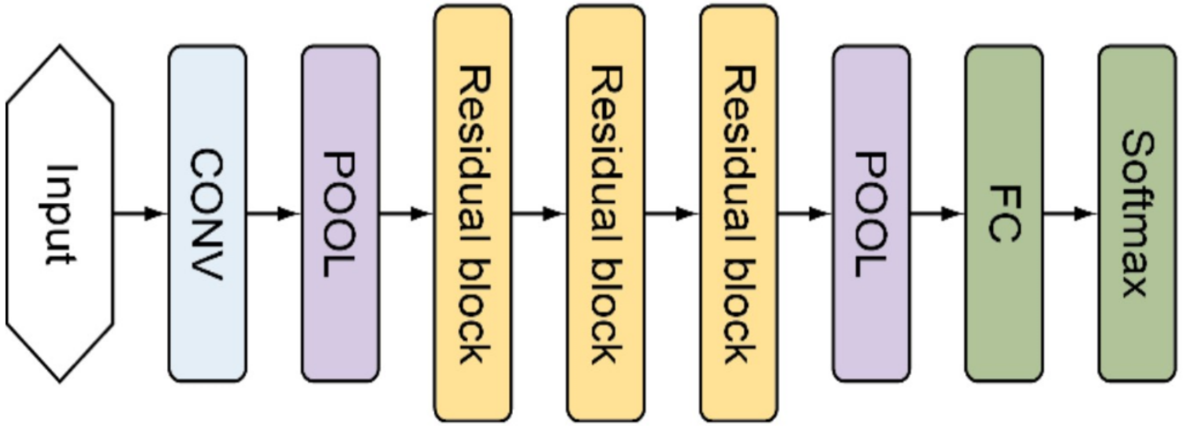
~13,000,000 parameters

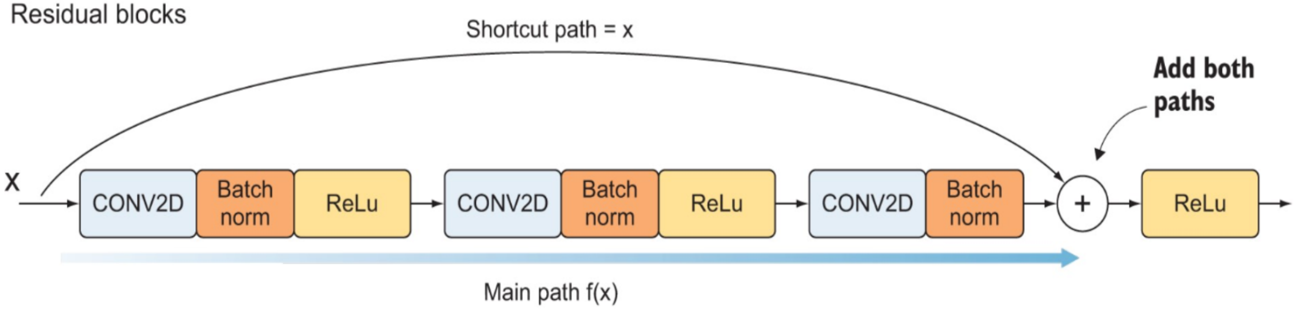


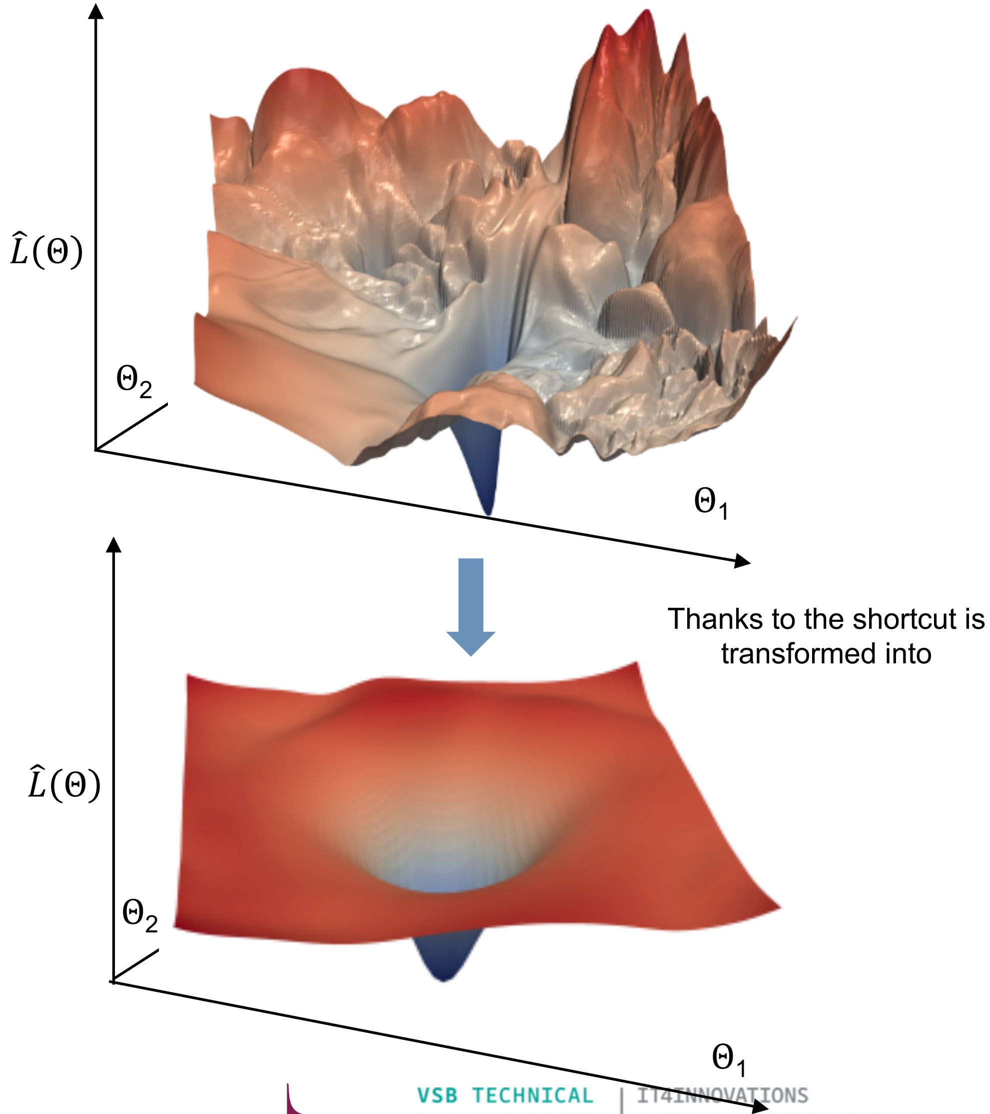
- What is the best kernel size for each layer?
- Concatenating filters instead of stacking them for reducing computational expenses



RestNet


Input
CONV
POOL
Residual block
Residual block
Residual block
POOL
FC
Softmax

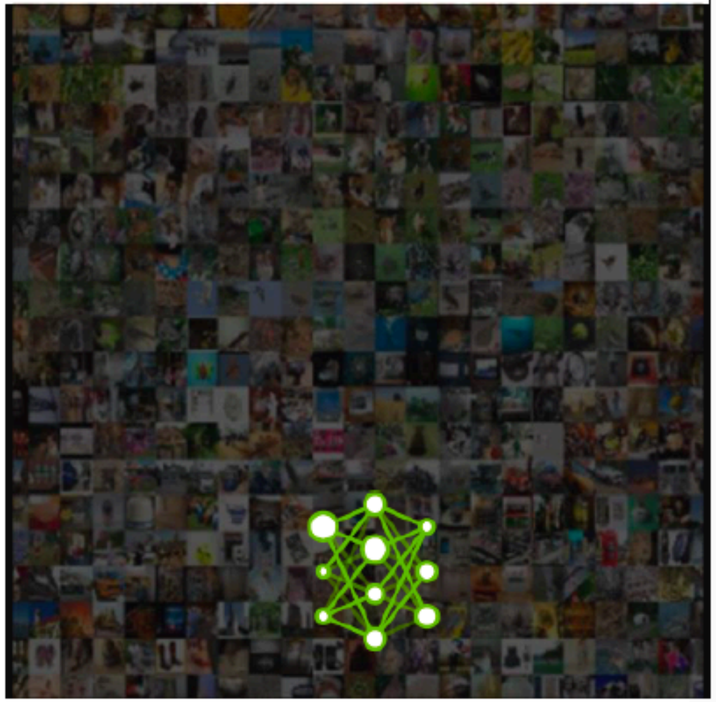

Residual blocks
Shortcut path = x
Add both paths
x
CONV2D
Batch norm
ReLu
CONV2D
Batch norm
ReLu
CONV2D
Batch norm
+
ReLu
Main path f(x)


$\hat{L}(\Theta)$
Θ_2
Θ_1
Thanks to the shortcut is transformed into
$\hat{L}(\Theta)$
Θ_2
Θ_1

Increasing complexity

PRACE NVIDIA DEEP LEARNING INSTITUTE lrz

7 Exaflops
60 Million Parameters



2015 - Microsoft ResNet
Superhuman Image Recognition

20 Exaflops
300 Million Parameters

2016 - Baidu Deep Speech 2
Superhuman Voice Recognition

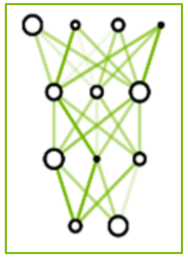
100 Exaflops
8700 Million Parameters

2017 - Google Neural Machine Translation
Near Human Language Translation

Part 2

Images – Input and Output

100	37	59	87	55	29	13	44
62	79	54	62	23	93	93	26
50	57	93	17	67	53	60	75
3	54	70	37	17	20	69	7
86	42	2	55	90	45	74	77
59	39	100	52	10	8	20	37
61	2	62	92	83	18	12	82
11	7	87	20	5	13	4	34



Deep Neural Network

0.04	0	0.02	0.01	0.92	0.01
Kites	Harrier	Vulture	Hawk	Eagle	Buzzards

Eagle

One-Hot: Turning words into Numbers

- Numerical vector representation for each word
- Dictionary of N words
- Each word is a vector with N-1 zeros and one 1, at the position of the word in the dictionary
- A document can be represented as a sequence of these one-hot vectors
- One interesting property of this representation is that no information is lost

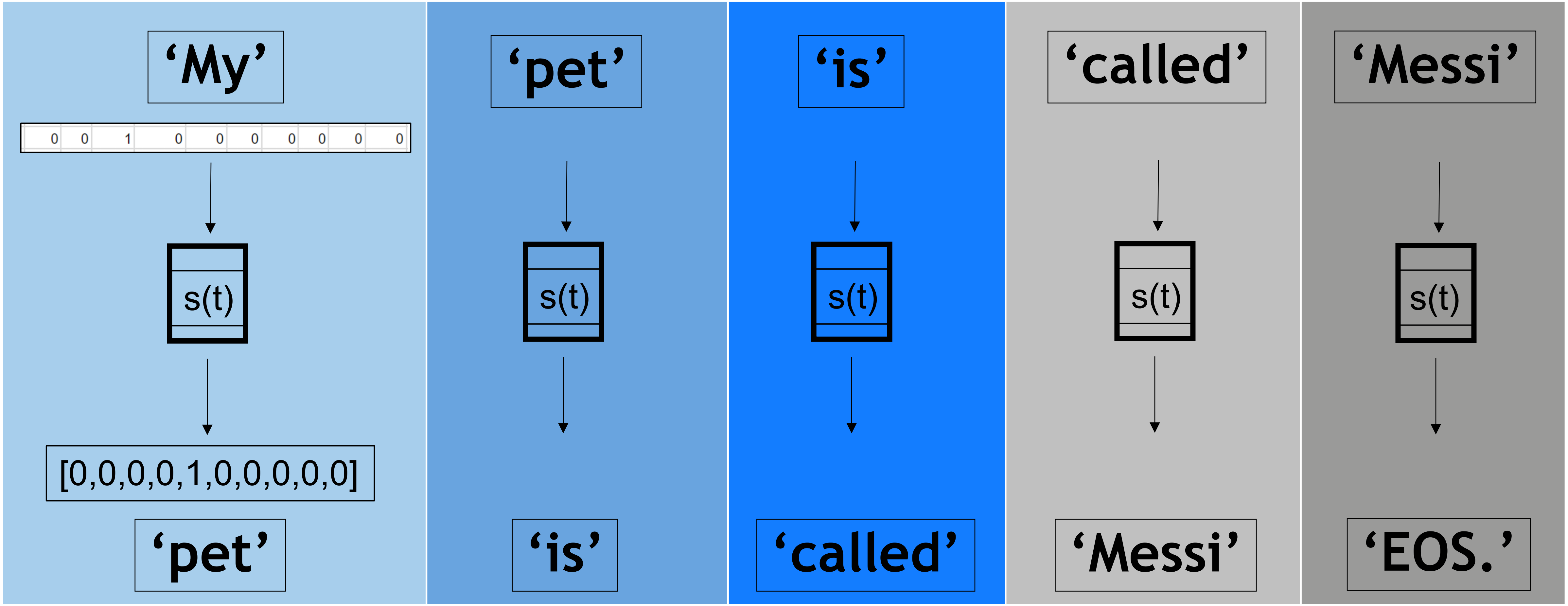
ONE-HOT ENCODING

```
small_dict=['EOS','a','my','sleeps','on','dog','cat','the','bed','floor'] #'EOS' means end of sentence.
```

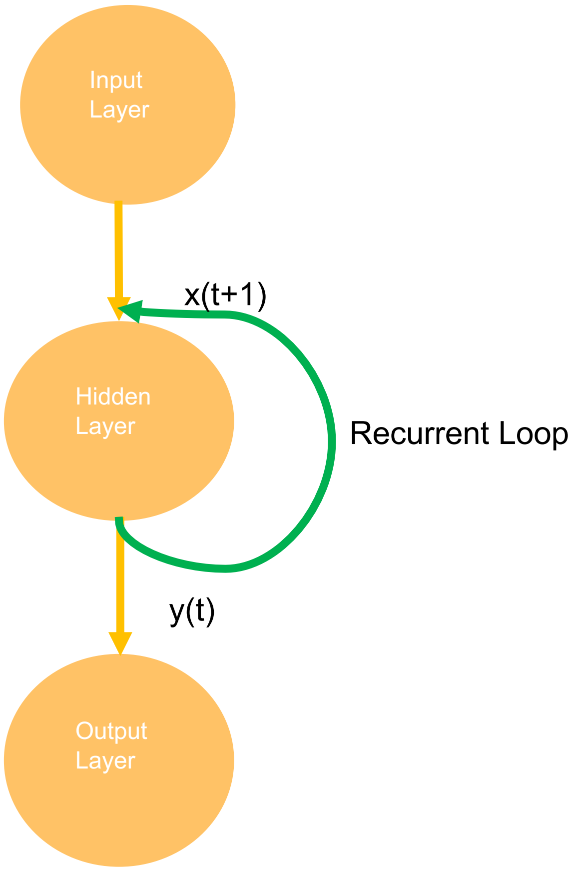
```
import numpy as np #numpy is "numerical python" and is used in deep learning mostly for its n-dimensional array
X=np.array([[2,6,3,4,2,8,0],[1,5,3,4,7,9,0]],dtype=np.int32)
print([small_dict[ind] for ind in X[1,:]]) #Feel free to change 1 to 0 to see the other sentence.
```

```
['a', 'dog', 'sleeps', 'on', 'the', 'floor', 'EOS']

one-hot encoded inputs
[[[ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
  [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
  [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
  [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]

 [[ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
  [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
  [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]]
shape of the input
(2, 7, 10)
```

Generating Language

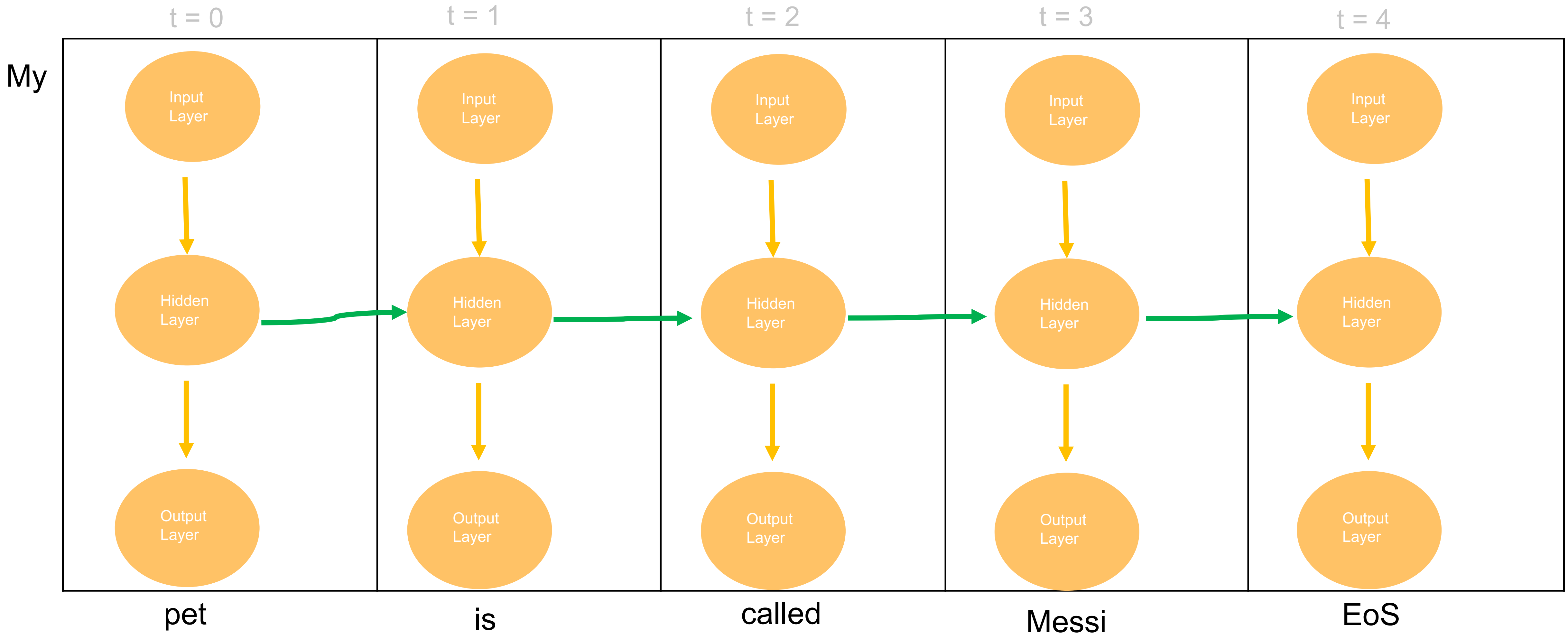


Recurrent Neural Network (RNN)

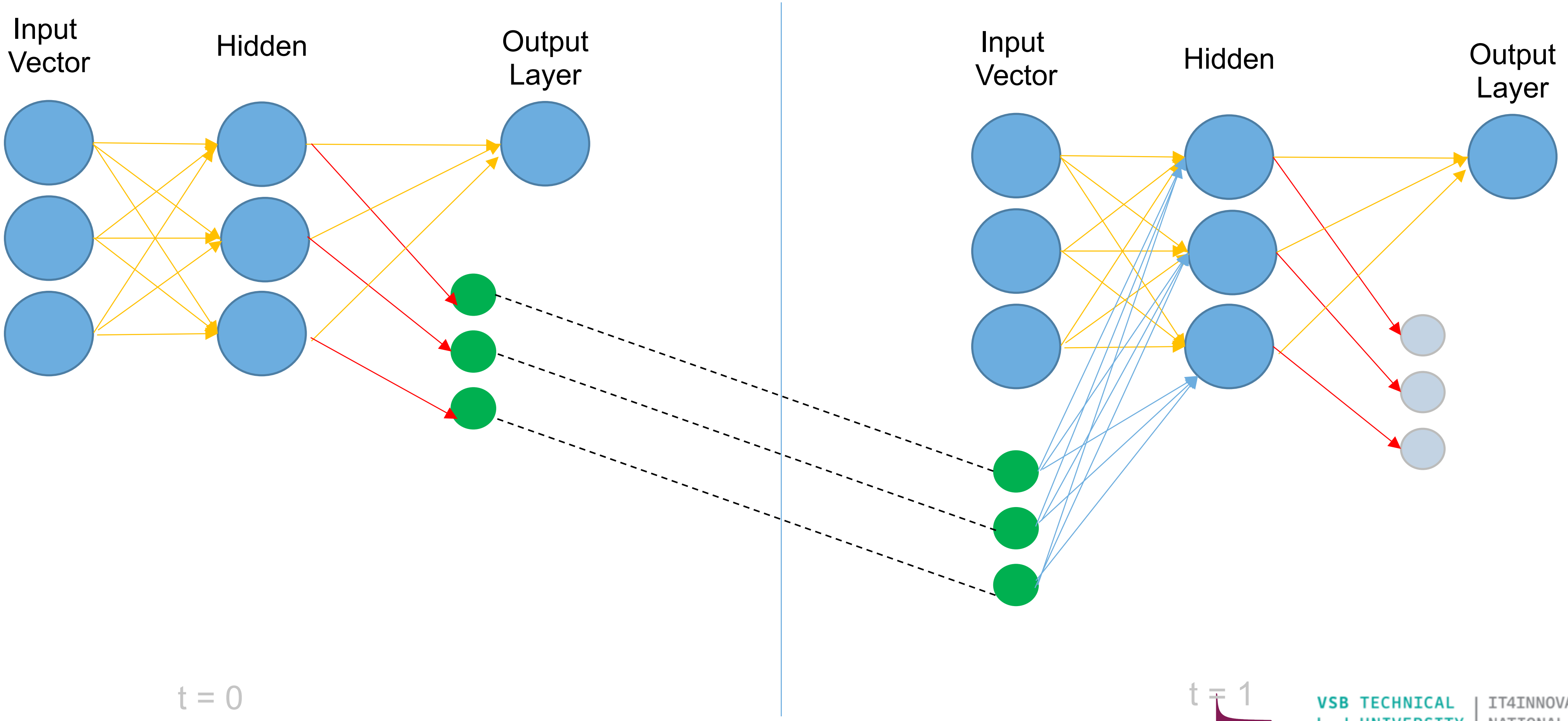


- Enable neural nets to remember past words within a sentence
- Recycle the output of the hidden layer at time *t* by adding as next input at time *t+1*
- Easier way of understanding its working behavior unrolling the net

RNN Unrolled



Understanding RNNs



Shortcomings of RNNs

1. Expensive training procedures
   1. A back propagation iteration updates each of the unrolled steps
2. Relationships between a word and the words that have appeared before
   1. Some words in some languages depends on what comes afterwards
   2. Bidirectional Recurrent Neural Networks
3. How many words in the past (or the future) influence the next word
   1. e.g., “The young woman, having found a free ticket on the ground, went to the movies.”
   2. Need of remembering the past across the entire input (young woman -> went)
   3. LSTM (Long Short Term Memory) Cells

Summary

- Brief introduction to Deep Learning with emphasis in Deep Convolutional Neural Networks
- Review of basic concepts: from perceptron to the learning task
- Debrief of most important concepts of neural network architectures
- Introduction to language modeling and RNNs