The Good, The Bad and the Ugly:

A Primer on Deep Learning

Pio Raffaele Fina

March 5, 2021

Università degli Studi di Torino

Who am I?

- MSc student in AI and Informative Systems (CS dept. University of Turin)
- Co-founder of Machine Learning Journal Club (unito): non-profit organization interested in ML and AI research topics (*info@mljc.it*).
- Just drop me a message:
  pioraffaelefina *pio.fina@edu.unito.it*

ML JC

Overview

Introduction

Deep Learning 101

Demo I

Convolutional Neural Networks

Demo II

Introduction

The (not) so Bad

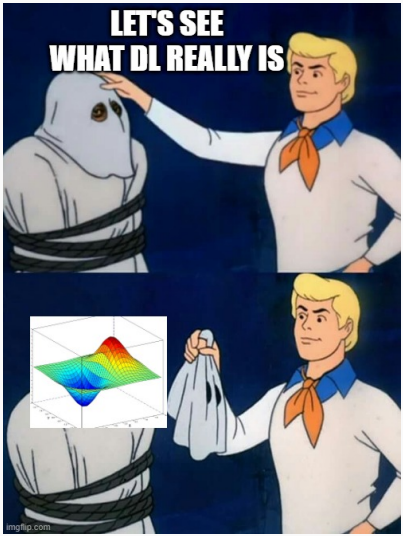
The marketing way: re-branding the same stuff over and over again.

- Why it's bad? sensationalism, misinformation, huge hype... all feelings that took us to the "*AI Winters*": expectations that can't be met with actual technology [1, 2]
- Why not so bad? Research as well as technology needs funding and trust by the society. People should know, but in the **right** way!

The (not) so Ugly

The real way: deep learning in essence is just a clever **pattern recognition** and **function approximation** machinery.

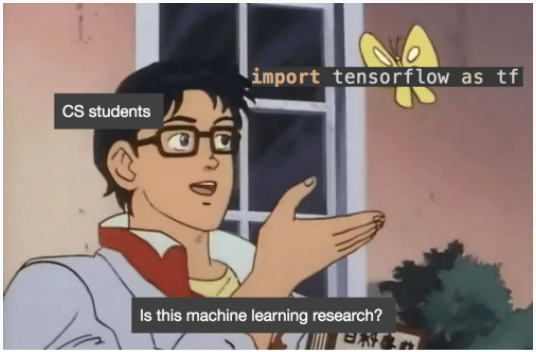
- Linear Algebra
- Mathematical Optimization
- Probability and Statistics
- Numerical Methods
- Parallel & Distributed Computing



The Good

The pragmatic way: deep learning frameworks and platforms.

- Lots of software engineering in the last decade [3].
- Off-the-shelf Support to GPU, parallel and distributed computing.
- Tensorflow, PyTorch, MXNet, JAX, etc.



Programmers Nowadays

Successful Applications

- NLP, CV, Speech Recognition $\longrightarrow$ just pattern recognition/generation.
- Understanding, Thinking, Reasoning, Explaining $\longrightarrow$ high order cognitive tasks, not so good for now [4, p. 167].

DL isn't useful? Not at all, **patterns are everywhere**!

Why ”Deep”?

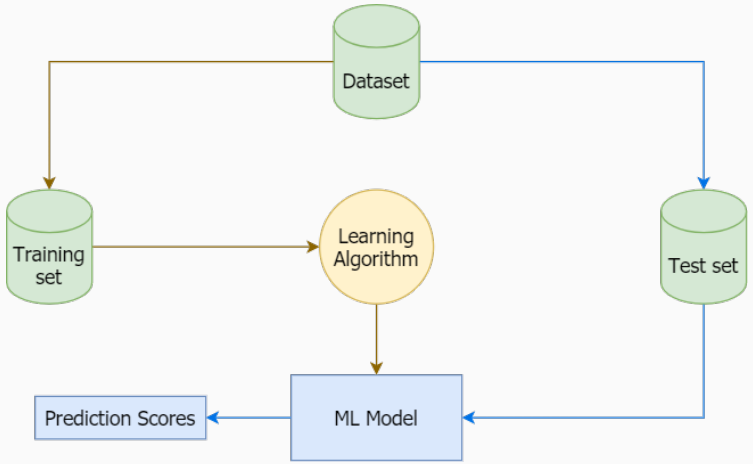
- Representations matter [5]! Learn how to **extract** information.
- Nature often exhibit **hierarchical structures** of abstractions
- Feature Engineering takes time and expertise, can we automatize this stage?

Key Idea

Can we fed raw data and let the network learn how to represent patterns?

Deep Learning 101

Framing the supervised learning problem:



From Perceptron to MLP and Beyond

Let's see how we come so far

McCullock-Pitts Neuron $\rightarrow$ Rosenblatt's Perceptron $\rightarrow$
Multilayer Perceptron $\rightarrow$ Backpropagation: going deep

Disclosure

The overview is over simplistic, to grasp in a systematic way the working details of every algorithm and model presented herein, please refer to [4, 6].

McCullock-Pitts Neuron (1943)

Seminal Paper where they proposed a first mathematical model of the biological neuron

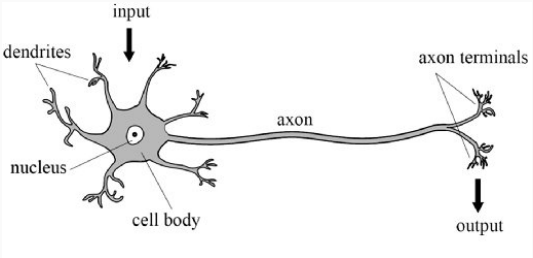
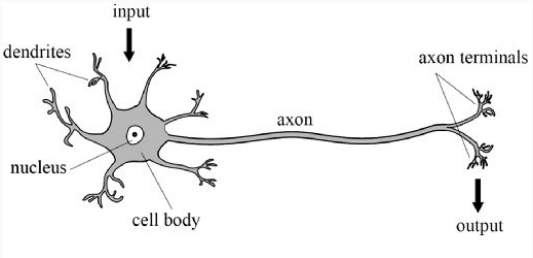


Figure 1: Image taken from A.C. Neves et. al

McCullock-Pitts Neuron (1943)

Seminal Paper where they proposed a first mathematical model of the biological neuron



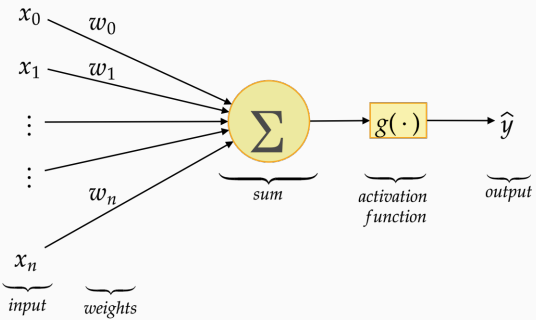
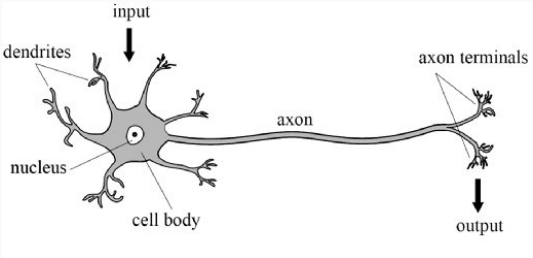


Figure 1: Image taken from A.C. Neves et. al

McCullock-Pitts Neuron (1943)

Seminal Paper where they proposed a first mathematical model of the biological neuron



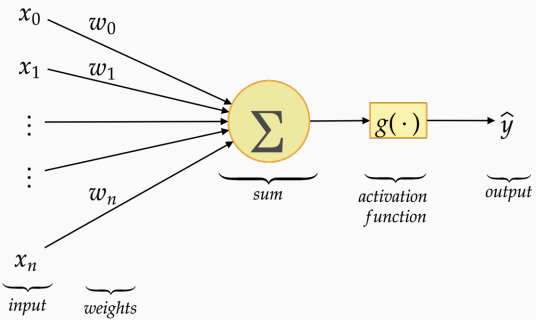


Figure 1: Image taken from A.C. Neves et. al

$$\hat{y} = g(\mathbf{w} \cdot \mathbf{x} + w_0)$$

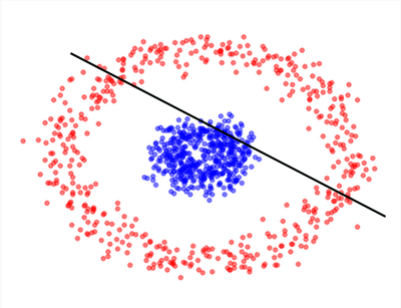
Rosenblatt's Perceptron (1958)

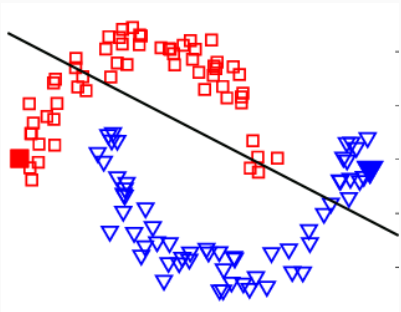
Simplest form of a neural network (NN) used for binary classification of **linearly separable patterns**.

- Just one artificial neuron (aka unit)
- Iterative Learning algorithm: given an input pair $(\mathbf{x}, y)$, compute the neuron output $\hat{y}$, update $\mathbf{w}$ according to a specified rule when $y \neq \hat{y}$.
- Extension to multi-class is straightforward

Linear Separability

What about real cases with arbitrarily complex data?



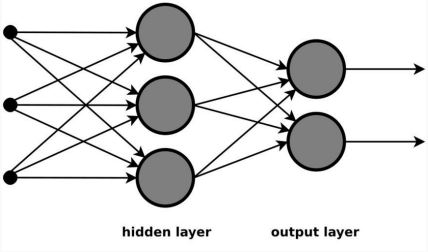


Splitting Hyperplane equation:

$$\mathbf{w} \cdot \mathbf{x} + w_0 = 0$$

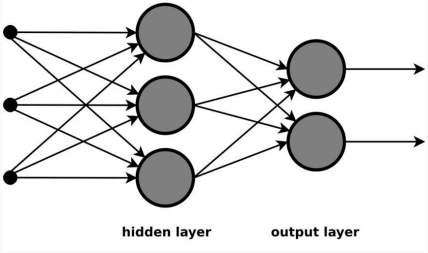
Multi Layer Perceptron

- Combine **multiple** units in hidden layers
- Each unit is **fully connected** with those of previous level

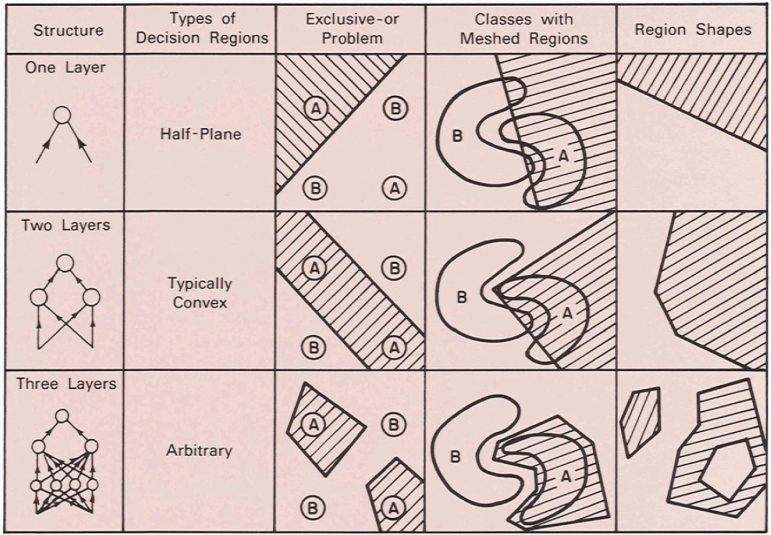


Multi Layer Perceptron

- Combine **multiple** units in hidden layers
- Each unit is **fully connected** with those of previous level
- Use **non linear** activation function (eg. ReLU, Sigmoid, Tanh, etc.)



MLP Decision Regions



What about deep learning?

A deep feedforward NN is just a MLP with more than 1 hidden layer.

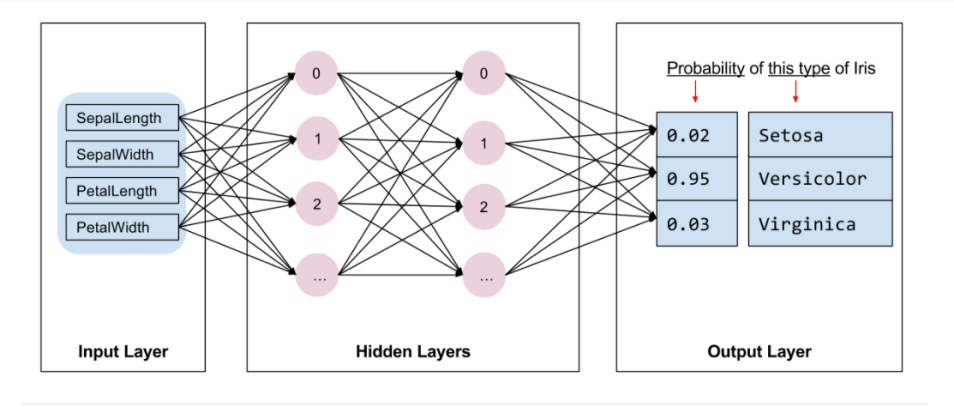


Figure 3: taken from Tensorflow guide

Recap

So far:

- Historical evolution.
- With MLP we can approximate an arbitrarily complex function given a sample of data
- MLP is one of the simplest architecture of a broader class called Feedforward NN.

Now:

How can we learn weights $\mathbf{W}$ from data D?

Gradient Based Learning

- Define some kind of measure that quantify the network error $\longrightarrow$ **Loss Function**
- Adjust weights to **minimize** the loss function $\longrightarrow$ Gradient Descent Optimization

Loss Function

"How much the network is wrong on a single input **x**?"

$$\mathcal{L}(\hat{y}, y)$$

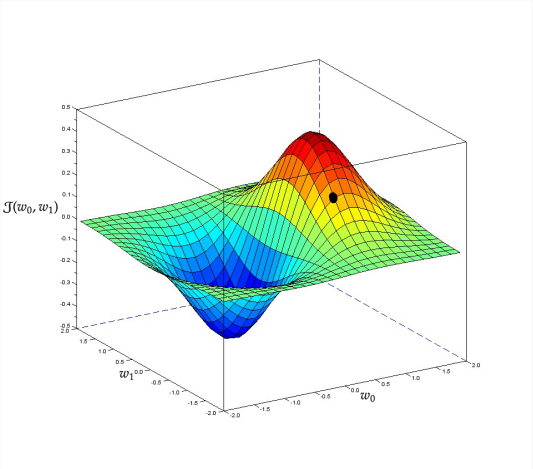
"How much the network is wrong on the entire training set D?"

$$\mathcal{J}(W) = \frac{1}{|D|} \sum_{x \in D} \mathcal{L}(\hat{y}, y)$$

Common choices: *cross entropy*, *MSE*, *MAE*, *KL Divergence*, etc.

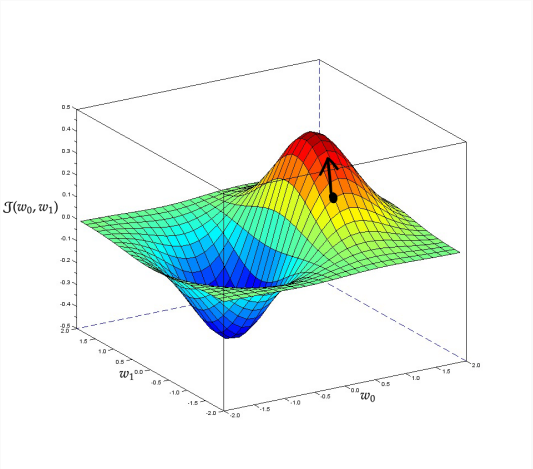
Gradient Descent (GD)

1. Initialize W according to some scheme



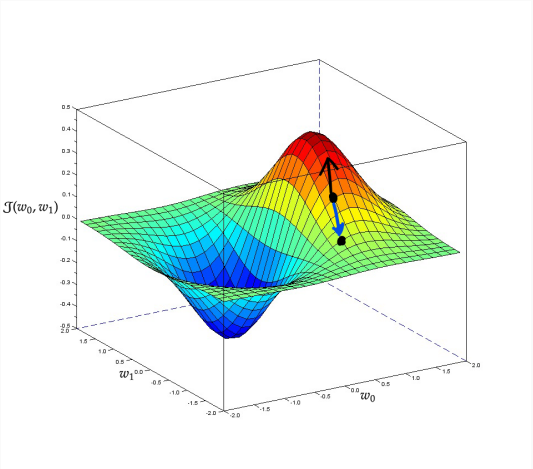
Gradient Descent (GD)

1. Initialize W according to some scheme
2. Compute the gradient



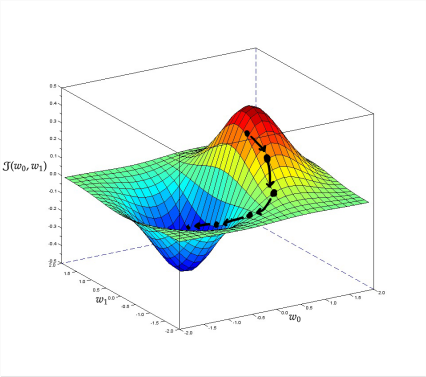
Gradient Descent (GD)

1. Initialize W according to some scheme
2. Compute the gradient
3. Update W taking a step of size η into the opposite direction



Gradient Descent (GD)

1. Initialize *W* according to some scheme
2. Compute the gradient
3. Update *W* taking a step of size η into the opposite direction
4. Repeat until some convergence condition



Backpropagation

How to compute a gradient of an extremely complex function? $\longrightarrow$ recursive application of chain rule of derivation a.k.a **Backpropagation** (BP)

Key Idea

How much a weight **w** should be **changed** given the input **x** and the output $\hat{y}$ to **reduce** the loss $\mathcal{L}(\hat{y}, y)$?

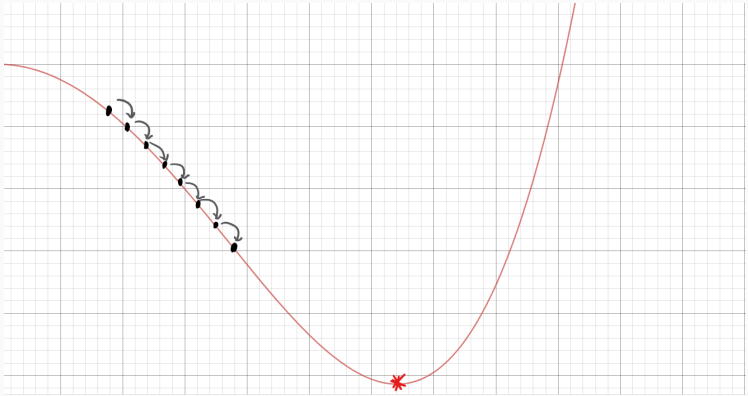
Main Issues

So far we have almost a complete picture of how to train a NN.

However we left unspecified some details, that not by chance are the main issues in real use case.

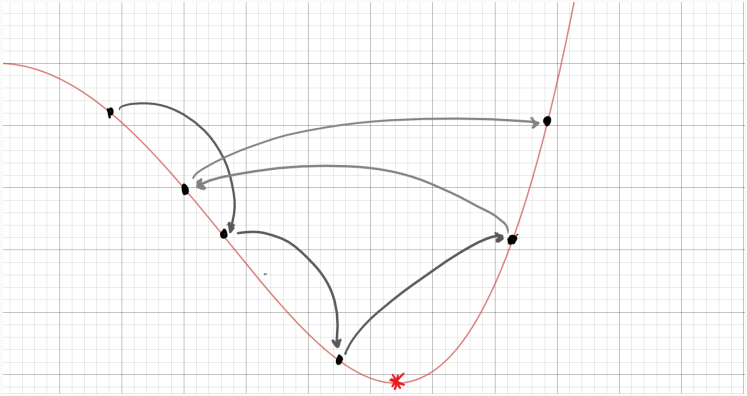
Main Issues I

How long should be the gradient step? $\longrightarrow$ learning rate **adaptive** schemes (time decay, Ada-*, etc.)



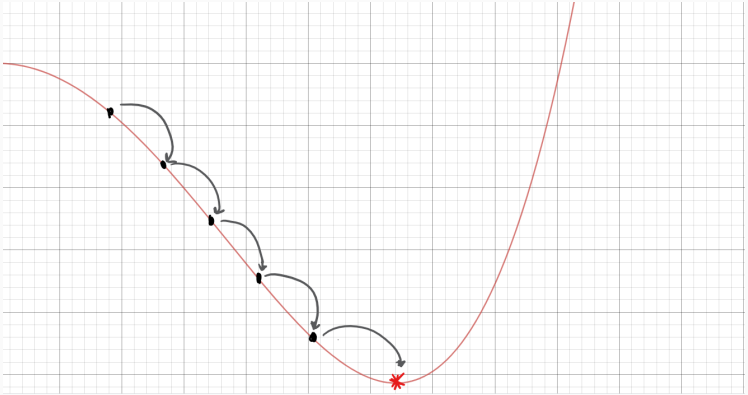
Main Issues I

How long should be the gradient step? $\longrightarrow$ learning rate **adaptive** schemes (time decay, Ada-*, etc.)



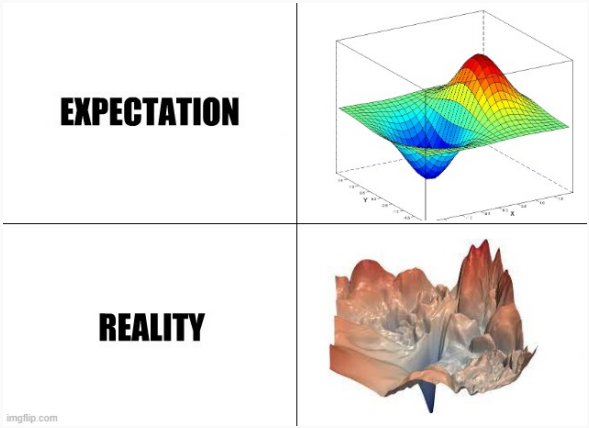
Main Issues I

How long should be the gradient step? $\longrightarrow$ learning rate **adaptive** schemes (time decay, Ada-*, etc.)



Non Convex Optimization

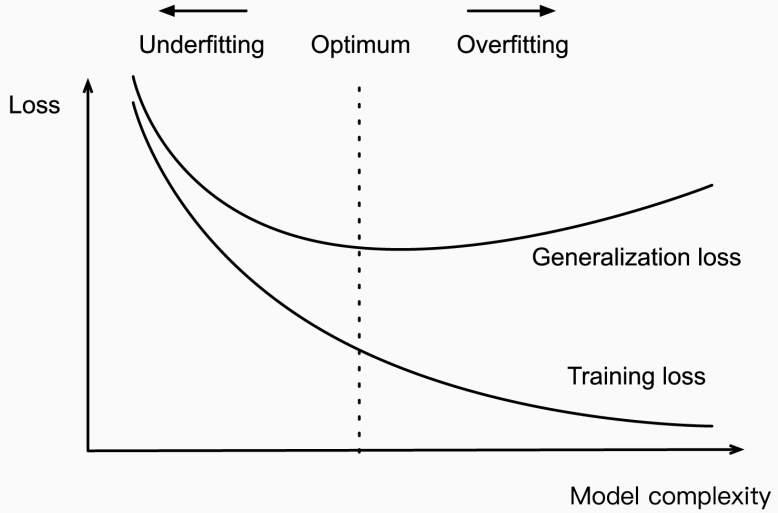
No global minimum $\longrightarrow$ **local minima** traps $\longrightarrow$ no optimal solution!



Main Issues II

- Computing $\mathcal{J}(\mathbf{W})$ for large training set is a memory intensive task $\longrightarrow$ Stochastic Gradient Descent (SGD) and Mini-Batch SGD.
- Low error on training set $\neq$ low **generalization error** $\longrightarrow$ underfitting & overfitting $\longrightarrow$ **regularization** (dropout, early stopping, L1/L2 loss, etc.)

Generalization Issues



Some references

We barely scratch the surface of GD and BP. Some good resources if you're really into this:

1. MIT 6.S191: Introduction to Deep Learning (easy to grasp)
2. 3Blue1Brown: Neural Networks (wonderful visualizations)
3. Deep Learning: An Introduction for Applied Mathematicians (formal)
4. The Matrix Calculus you Need for Deep Learning (complete but pretty wild)

Is this what happen in our brain when we learn something? Amongs many, gradient based learning and BP is considered not a biological plausible procedure. Other types of learning mechanism have been proposed in literature and starts to emerging [7].

Remember

NN are very loosely inspired by our brain, they are a **function approximation** machinery.

Demo I

MLP in Tensorflow and Keras

Convolutional Neural Networks

Recap

So far:

- Historical evolution.
- With MLP we can approximate an arbitrarily complex function given a sample of data.
- Train a NN can be casted as a **non convex** minimization problem of the loss function.

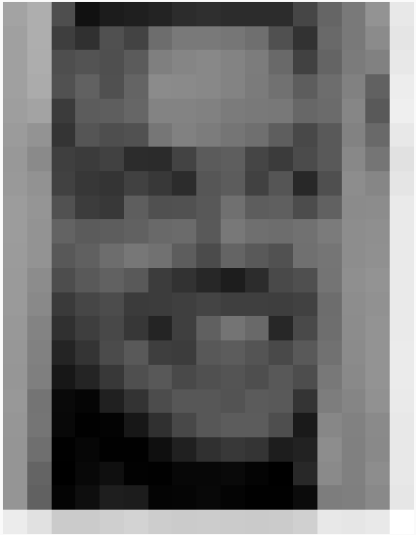
Now:

How can we **detect patterns** in high dimensional spatial data like images?

Vision

- The most information rich sense.
- We don't just "see" we also "perceive" a scene. It seems that human vision is deeply entangled with cognition and semantic representations of objects.
- In Computer Vision, machines just see **patterns** of numbers.

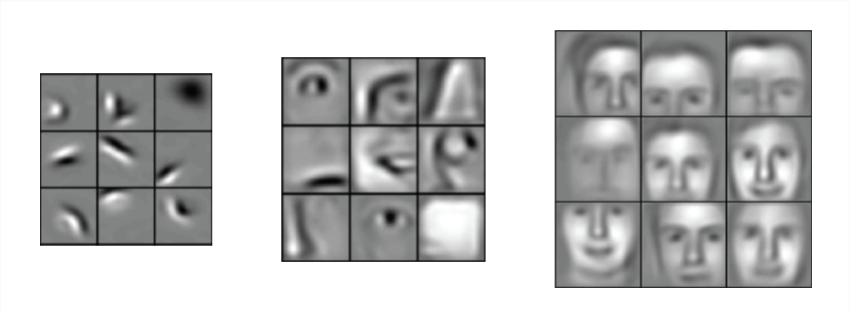
How Machines ”See”?



160	171	81	23	33	36	40	49	51	50	48	49	73	102	121	146	225
157	171	66	52	82	70	110	121	122	118	112	82	54	107	121	137	222
157	170	81	85	81	92	126	132	135	131	123	106	68	101	122	128	225
157	167	84	98	82	101	137	137	135	130	123	113	95	109	128	98	229
155	161	62	94	96	102	130	130	129	123	121	120	104	107	133	92	232
153	146	57	87	81	84	118	128	124	117	111	96	74	91	129	107	225
149	137	66	62	68	48	47	69	93	95	72	62	78	92	135	117	219
151	144	68	59	56	68	60	48	88	93	69	87	44	81	139	133	222
153	147	84	63	60	95	82	84	90	112	94	96	78	99	137	139	227
153	146	100	92	94	99	107	108	90	120	112	109	114	123	139	142	227
153	146	90	97	109	119	113	94	77	101	92	83	112	117	140	143	227
151	138	79	91	99	90	63	54	44	34	49	76	81	116	142	145	227
151	135	62	72	78	63	63	66	70	67	66	66	68	109	139	143	227
149	131	52	65	75	59	40	66	97	116	107	43	76	110	139	145	227
149	128	42	56	77	92	65	64	89	93	86	77	91	114	137	145	228
148	126	29	41	61	81	90	76	82	86	81	90	79	120	136	146	228
150	116	16	11	38	55	78	89	88	85	92	92	56	124	131	143	227
148	116	15	8	10	28	53	75	89	93	93	80	32	134	128	136	225
147	111	8	12	8	7	21	37	53	60	56	34	39	139	127	136	225
147	101	6	14	21	8	8	13	14	16	11	7	46	137	128	140	225
147	98	11	21	35	34	10	12	15	11	8	8	18	125	127	135	223
229	218	197	197	202	205	199	198	199	197	196	197	202	225	223	226	247

Hierarchy of Visual Features

We don't just see flat images, our visual system build **part-whole hierarchies** of increasing complex and abstract levels [8].



Nice properties

Human vision is really good because is **robust to scene variations** :

- Translation Invariance

Nice properties

Human vision is really good because is **robust to scene variations** :

- Translation Invariance
- Rotation Invariance

Nice properties

Human vision is really good because is **robust to scene variations** :

- Translation Invariance
- Rotation Invariance
- Scale Invariance

Nice properties

Human vision is really good because is **robust to scene variations** :

- Translation Invariance
- Rotation Invariance
- Scale Invariance
- Viewpoint Invariance

Nice properties

Human vision is really good because is **robust to scene variations** :

- Translation Invariance
- Rotation Invariance
- Scale Invariance
- Viewpoint Invariance
- Illumination Invariance

Nice properties

Human vision is really good because is **robust to scene variations** :

- Translation Invariance
- Rotation Invariance
- Scale Invariance
- Viewpoint Invariance
- Illumination Invariance
- Shape Deformation

Nice properties

Human vision is really good because is **robust to scene variations** :

- Translation Invariance
- Rotation Invariance
- Scale Invariance
- Viewpoint Invariance
- Illumination Invariance
- Shape Deformation
- Occlusion

Spatial Relations

Spatial relations matters! Usually, near pixels are semantically correlated.

Spatial Relations

Spatial relations matters! Usually, near pixels are semantically correlated.

These images represents the same object!

Convolution I

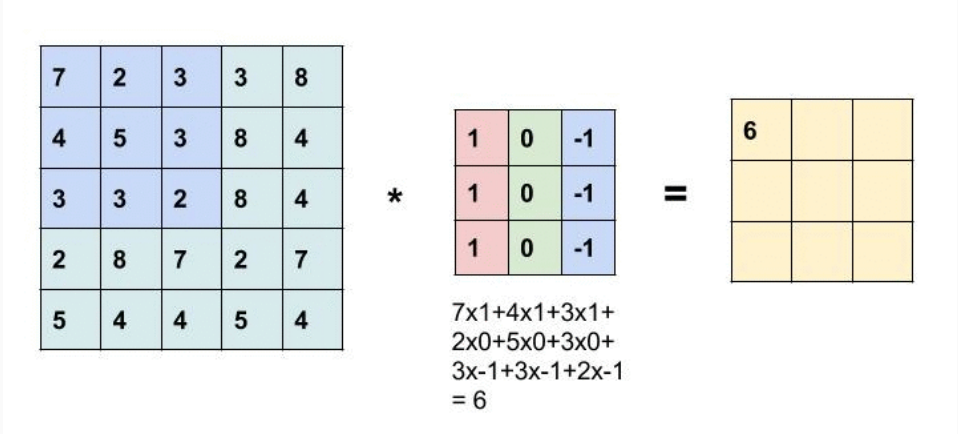
Key Idea

We need an operation that detect pattern and is spatial aware.

$$(I * K)(i, j) = \sum_m \sum_n I(m, n) \odot K(i - m, j - n)$$

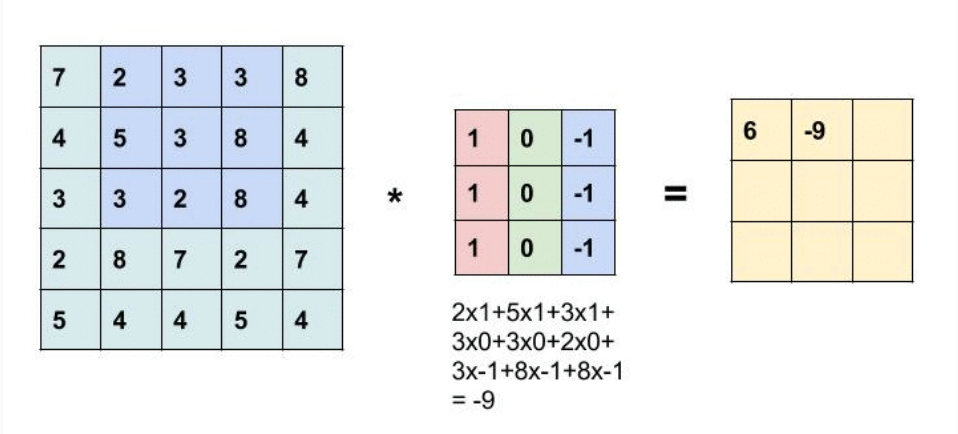
I is a two-dimensional image and K is called Kernel. Essentially, convolution is just a **linear combination** with a **sliding window**.

Convolution II



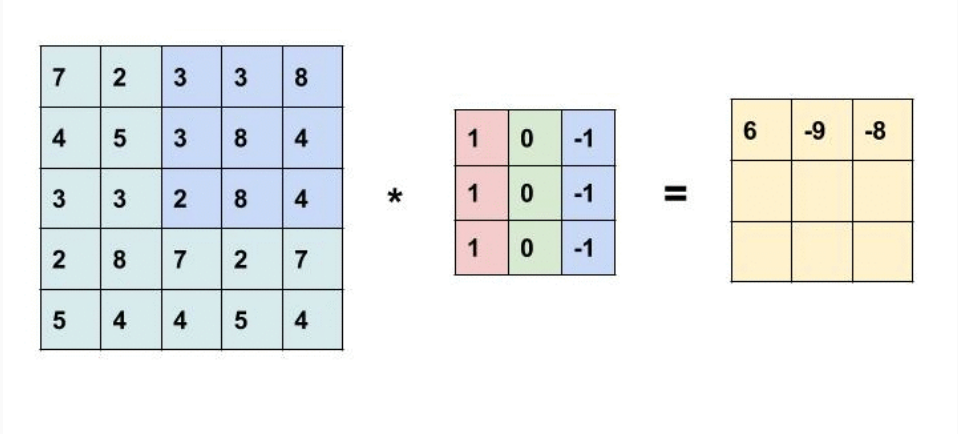
Images taken from: CNN by Rijul Vohra

Convolution II



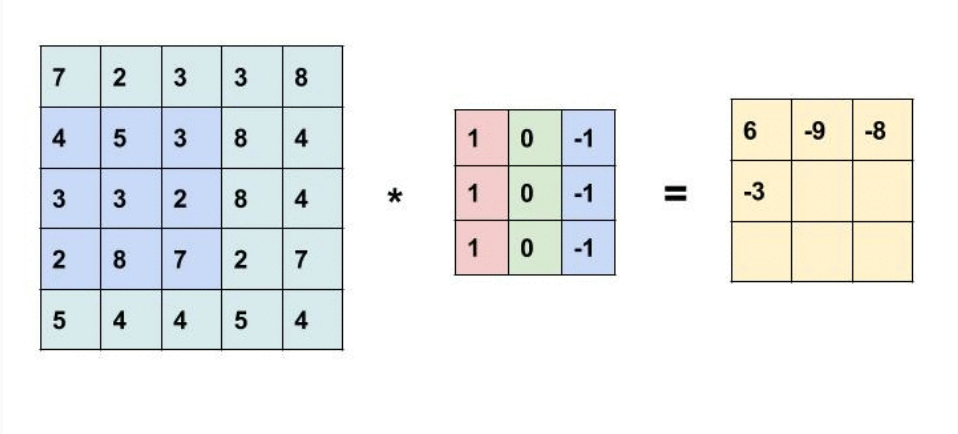
Images taken from: CNN by Rijul Vohra

Convolution II



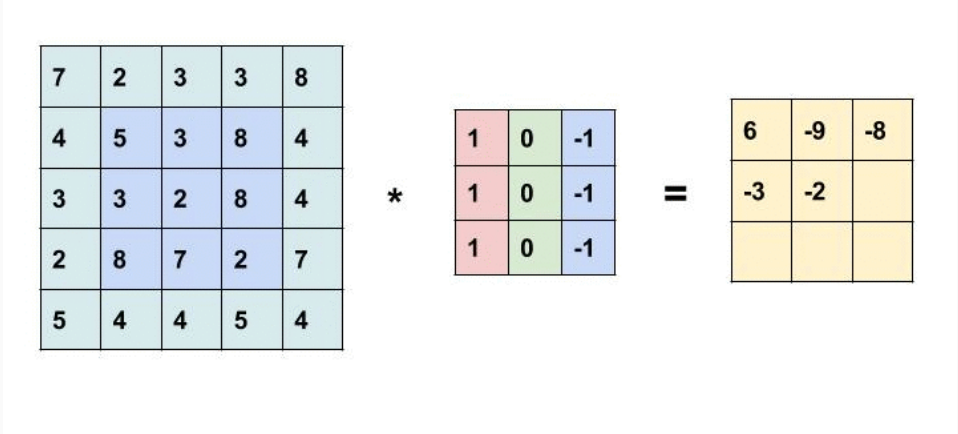
Images taken from: CNN by Rijul Vohra

Convolution II



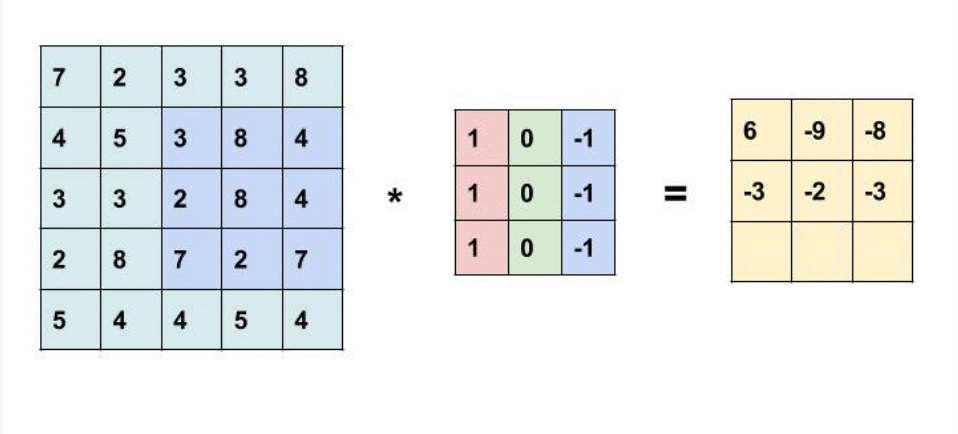
Images taken from: CNN by Rijul Vohra

Convolution II



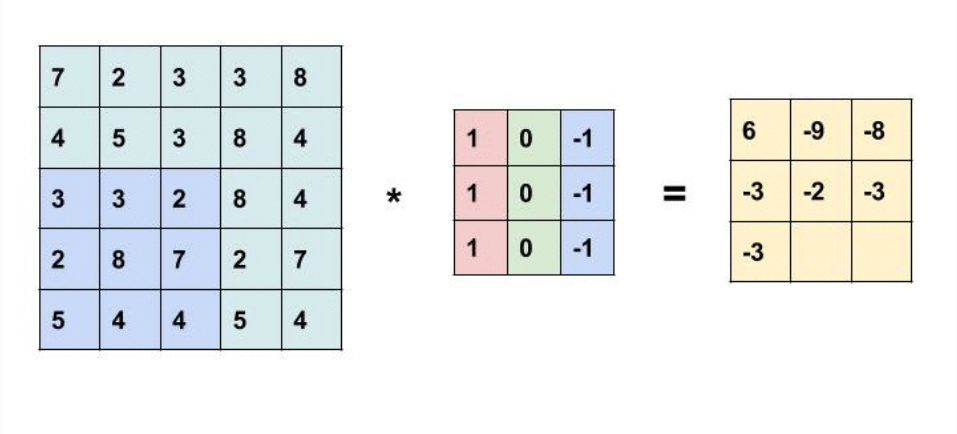
Images taken from: CNN by Rijul Vohra

Convolution II



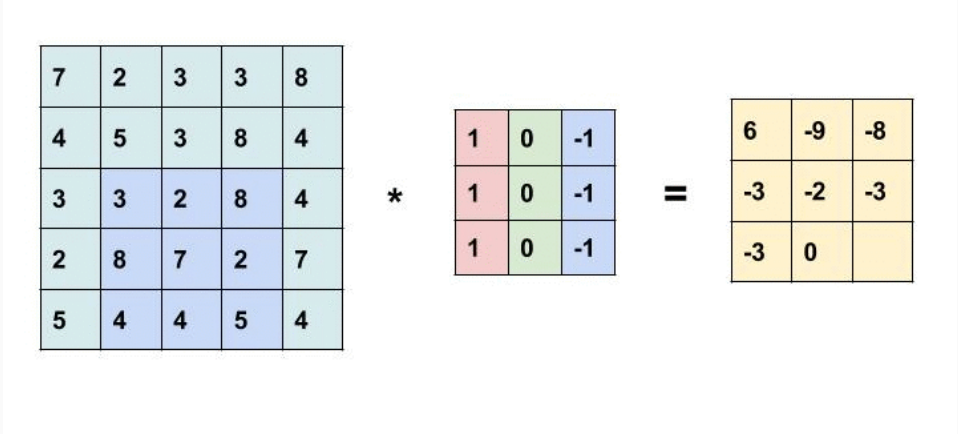
Images taken from: CNN by Rijul Vohra

Convolution II



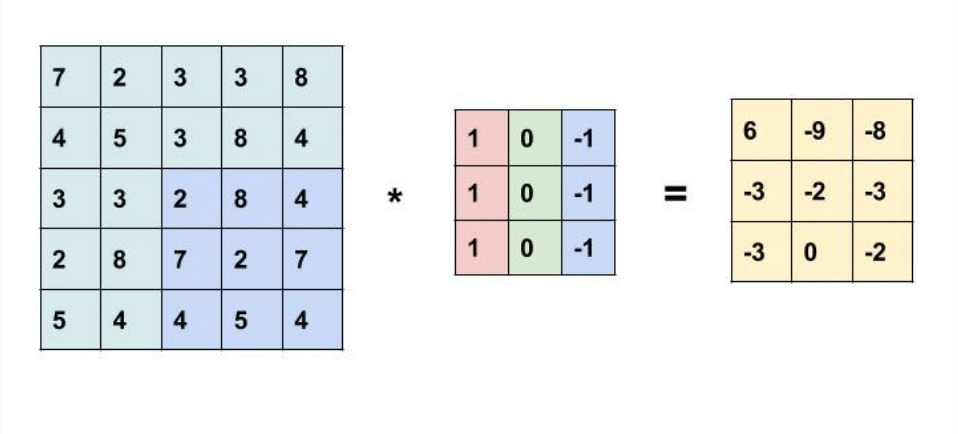
Images taken from: CNN by Rijul Vohra

Convolution II



Images taken from: CNN by Rijul Vohra

Convolution II

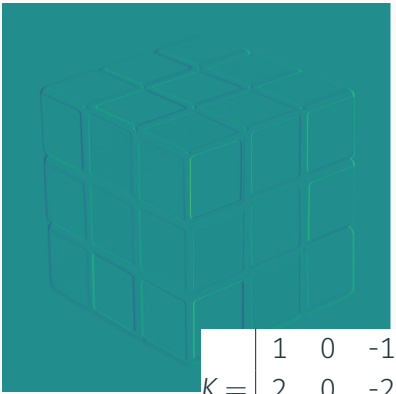


Images taken from: CNN by Rijul Vohra

Convolution Feature Map

Different kernel configurations detect different patterns.

$$K = \begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix}$$



$$K = \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix}$$

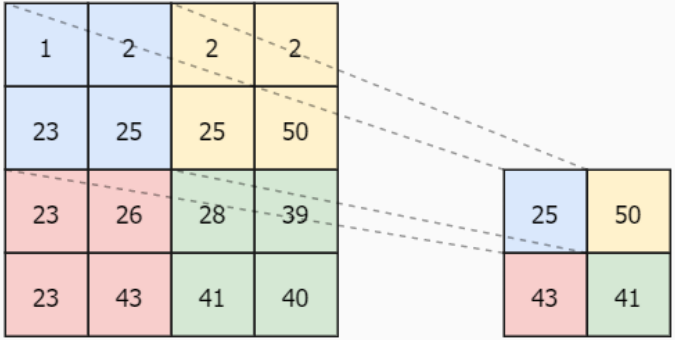
Convolutional Neural Network (CNN)

Key Idea

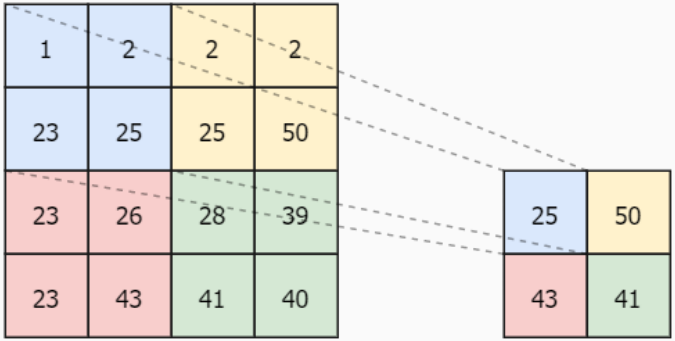
Why don't let the NN learn kernel configurations from data?

- CNN are feedforward NN that use **convolution** (on volumes) and **pooling** as main operations.
- CNN weights are kernels values (filters in this context).

Pooling



Pooling



- Reduce volumes dimensions $\longrightarrow$ Fast computations.
- Pooling makes CNN **invariant** (only) to small translations $\longrightarrow$ More robust to pattern position variations.

Vanilla CNN Architecture

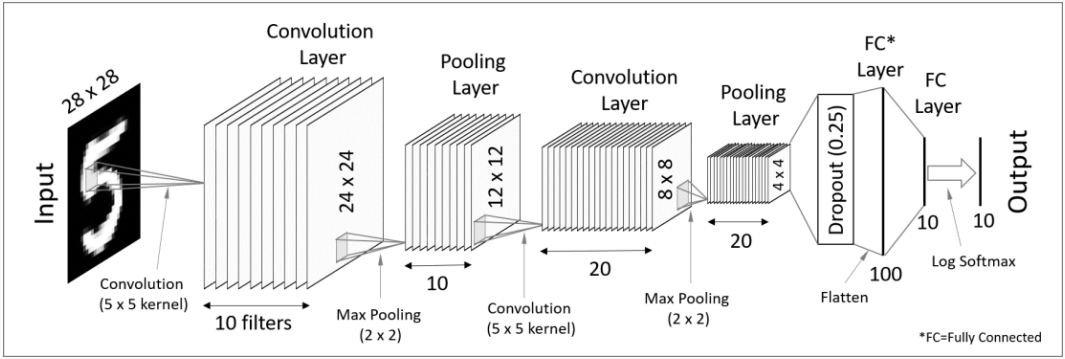


Figure 4: LeNet like CNN

CNN Motivations

- CNN **preserve spatial relationships**, input and output are **volumes** of size $w \times h \times d$.
- CNN is more **computational efficient** w.r.t fully connected NN
  - $|K| << |I| \longrightarrow$ less weights to learn and store
  - reuse same weights on multiple positions

  (see *sparse connectivity* and *parameters sharing* properties [4])
- The sliding window mechanism allows no fixed size input.

Recap

So far:

- Convolution replace matrix multiplication operations of FC layers.
- CNN act as a spatial aware feature detector.
- CNN have nice desirable properties.

Demo II

Vanilla CNN in Tensorflow and Keras

Closing Remarks

- Building blocks of a FNN and CNN.
- NN are clever pattern detectors.
- NN architectures depends on the nature of your data.
- DL frameworks are very easy to use on toy problems, but things goes pretty wild on complex stuff.

Beware that we missed a lot of stuff today, there are tons of other stuff still to learn!

if you dont' mind

https://forms.gle/5HBcjKsA2fmgZLBi9

Thank you!

References

[1] *On the sensationalism of artificial intelligence news*. en-US. URL: https://www.kdnuggets.com/on-the-sensationalism-of-artificial-intelligence-news.html/ (visited on 02/24/2021).

[2] Thomas Nield. *Is Another AI Winter Coming?* en. Sept. 2019. URL: https://medium.com/hackernoon/is-another-ai-winter-coming-ac552669e58c (visited on 02/24/2021).

[3] Lin Yuan. *A Brief History of Deep Learning Frameworks*. en. Jan. 2021. URL: https://towardsdatascience.com/a-brief-history-of-deep-learning-frameworks-8debf3ba6607 (visited on 02/24/2021).

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 9780262035613.

[5] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (Aug. 2013), pp. 1798–1828. ISSN: 0162-8828, 2160-9292. DOI: *10.1109/TPAMI.2013.50*. URL: *http://ieeexplore.ieee.org/document/6472238/* (visited on 02/27/2021).

[6] Simon S. Haykin. *Neural networks and learning machines*. 3rd ed. OCLC: ocn237325326. New York: Prentice Hall, 2009. ISBN: 9780131471399.

[7] Anil Ananthaswamy. *Artificial Neural Nets Finally Yield Clues to How Brains Learn*. en. URL: https://www.quantamagazine.org/artificial-neural-nets-finally-yield-clues-to-how-brains-learn-20210218/ (visited on 02/26/2021).

[8] Antonio J. Rodríguez-Sánchez, Mazyar Fallah, and Aleš Leonardis. "Editorial: Hierarchical Object Representations in the Visual Cortex and Computer Vision". In: *Frontiers in Computational Neuroscience* 9 (Nov. 2015). ISSN: 1662-5188. DOI: 10.3389/fncom.2015.00142. URL: http://journal.frontiersin.org/Article/10.3389/fncom.2015.00142/abstract (visited on 02/28/2021).