Neural Networks and CNNs

Paweł Gelar

Warsaw 03.2023

Contents

1. Introduction
2. Neural Networks
3. Convolutional Neural Networks (CNNs)
4. Concepts

Introduction

Classical vs Deep machine learning

Classical ML

Everything excluding Neural Networks (SVMs, trees, forests, boosting, linear models)

Deep Learning

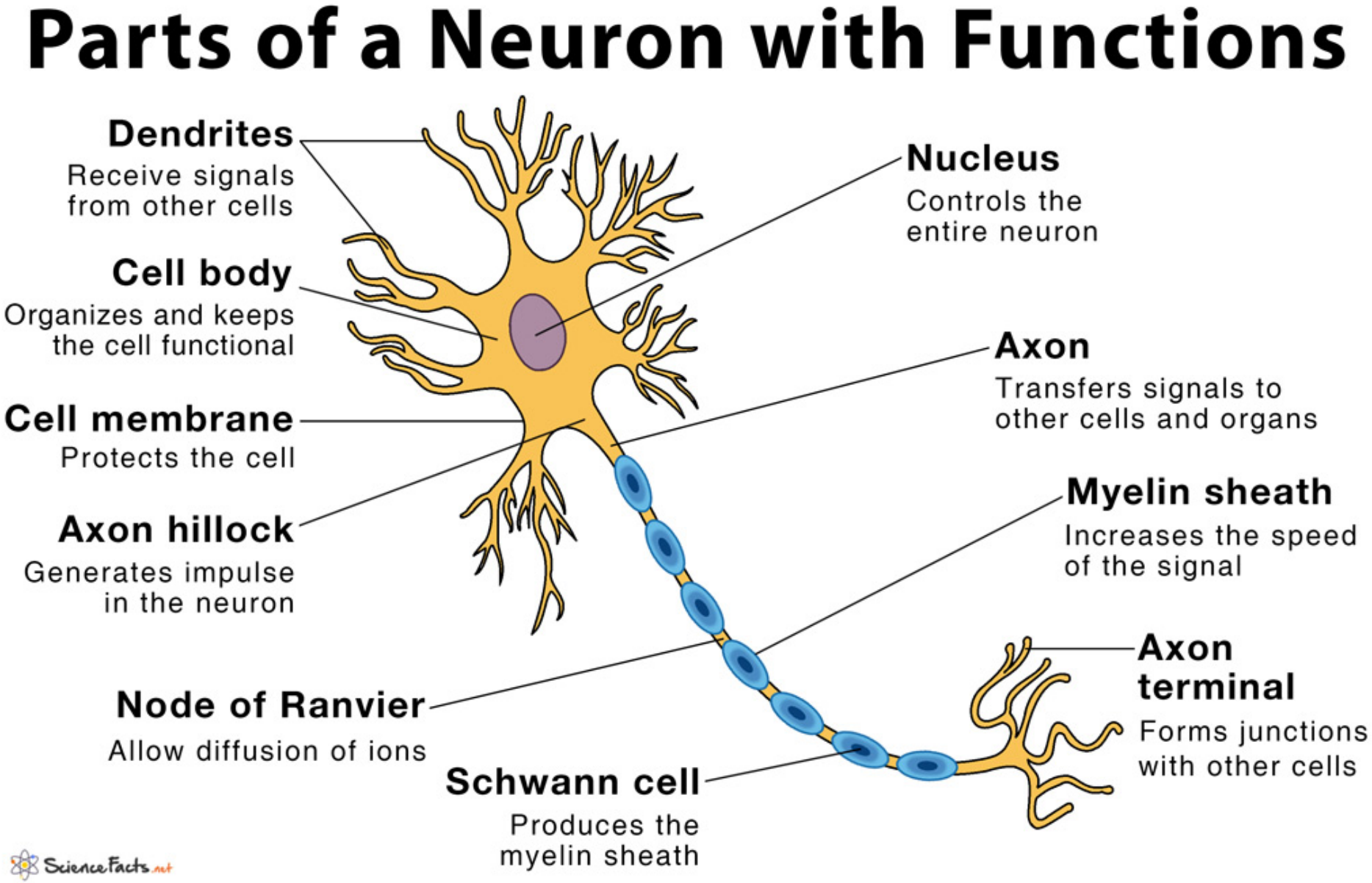
Neural Networks (regardless of complexity)

Neural Networks

History

- The first ideas were proposed mid XX century trying to mimic how brain works, but weren't really successful
- Backpropagation - way to teach neural networks introduced in 1986 - beginning of neural networks as we know
- CNN introduced in 1990
- "Attention Is All You Need" - transformers introduced in 2017
- ChatGPT and recent boom

Obligatory biological neuron slide



Multi Layer Perceptron

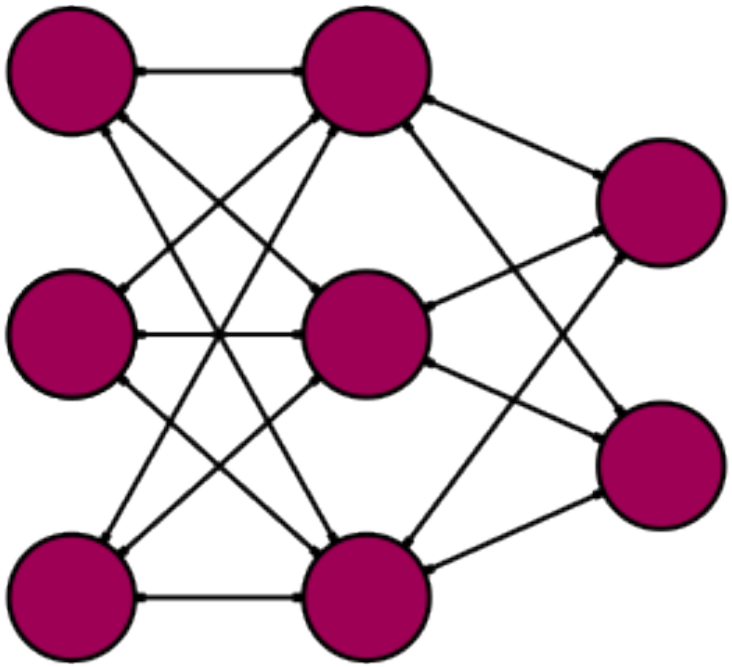


Figure: Our club logo - representation of a simple neural network

Activation Functions

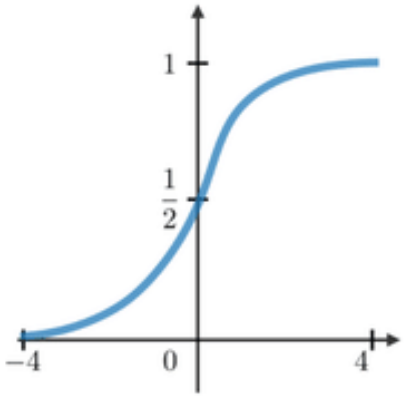
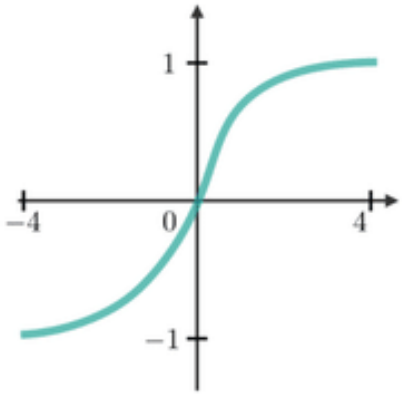
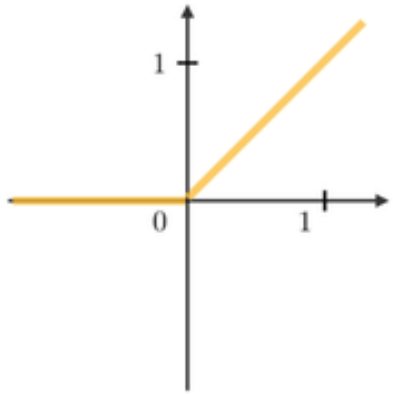
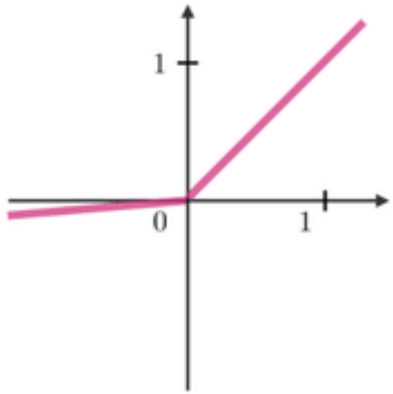
Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

Figure: Common Activation functions

Forward Pass

Example

https://playground.tensorflow.org

Mathematical formulation

$$y = f(X, \theta) \tag{1}$$

$$\min_{\theta} \mathcal{L}(f(X, \theta)) \tag{2}$$

$\mathcal{L}$ - loss function

$$\theta_{n+1} = \theta_n + \frac{\partial \mathcal{L}(\theta_n)}{\partial \theta} \tag{3}$$

Calculating derivatives

Chain Rule

$$\frac{d}{dx}f(g(x)) = \frac{d}{dg(x)}f(g(x)) \times \frac{d}{dx}g(x) \tag{4}$$

Automatic Differentiation (Backward pass)

Example on the blackboard:

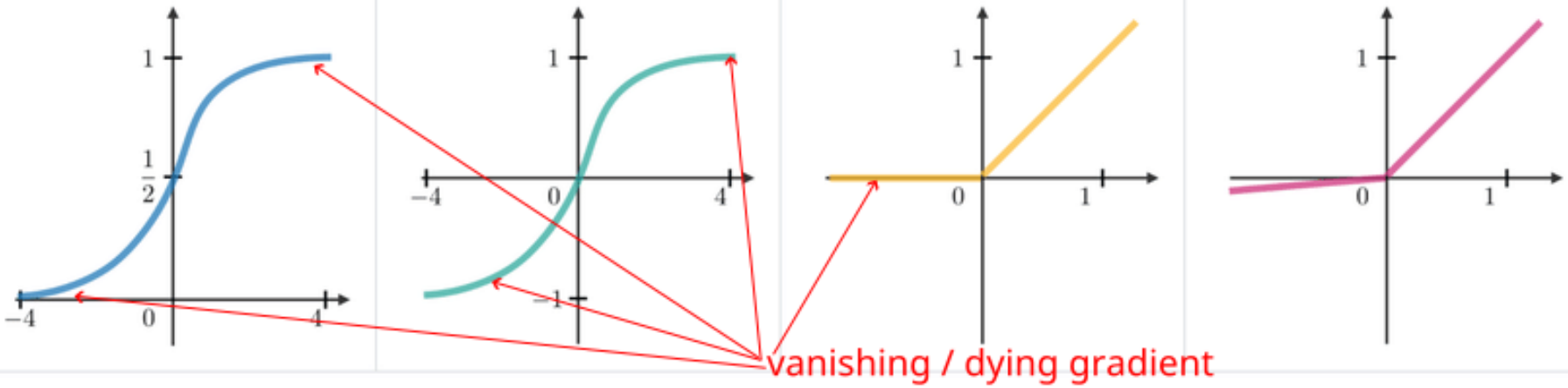
$f(\mathbf{x}) = (x_1 x_2 sin(x_3) + e^{x_1 x_2})/x_3$

$\mathbf{x} = [1, 2, \pi/2]^T$

$\nabla f(\mathbf{x}) = [(4 + 4e^2)/\pi, (2 + 2e^2)/\pi, (-8 - 4e^2)\pi^2]^T$

Back to activations

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$

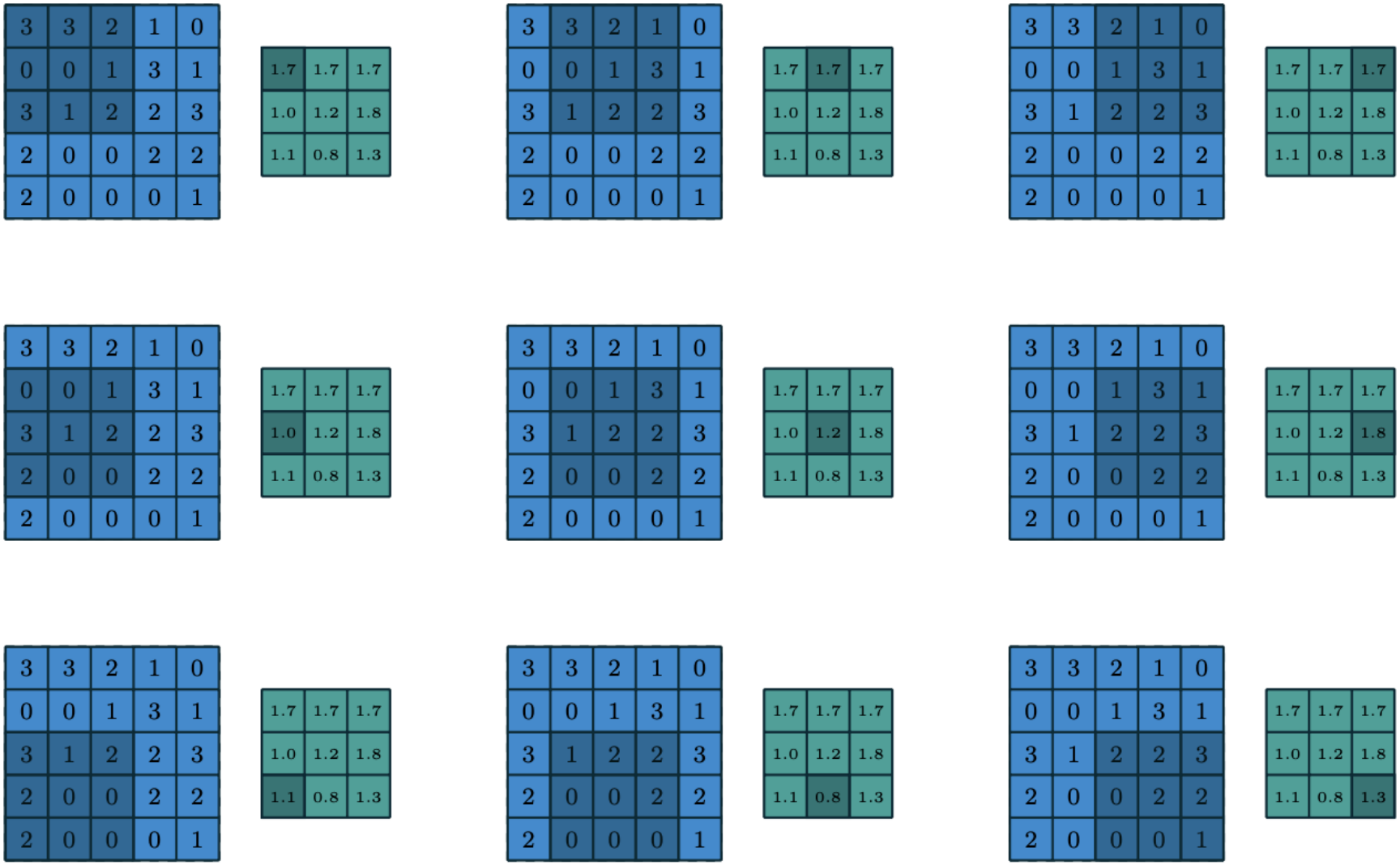


Pros Cons

- Computation heavy
- **A LOT** of parameters (10^{12} for GPT-4)
- Easily overfited
- Can approximate any function*
- Very versatile
- Sequential Learning

Convolutional Neural Networks (CNNs)

Convolution



Some Convolution Kernels

Example

https://setosa.io/ev/image-kernels/

How to choose the right kernels?

Use learnable parameters

Example of CNN

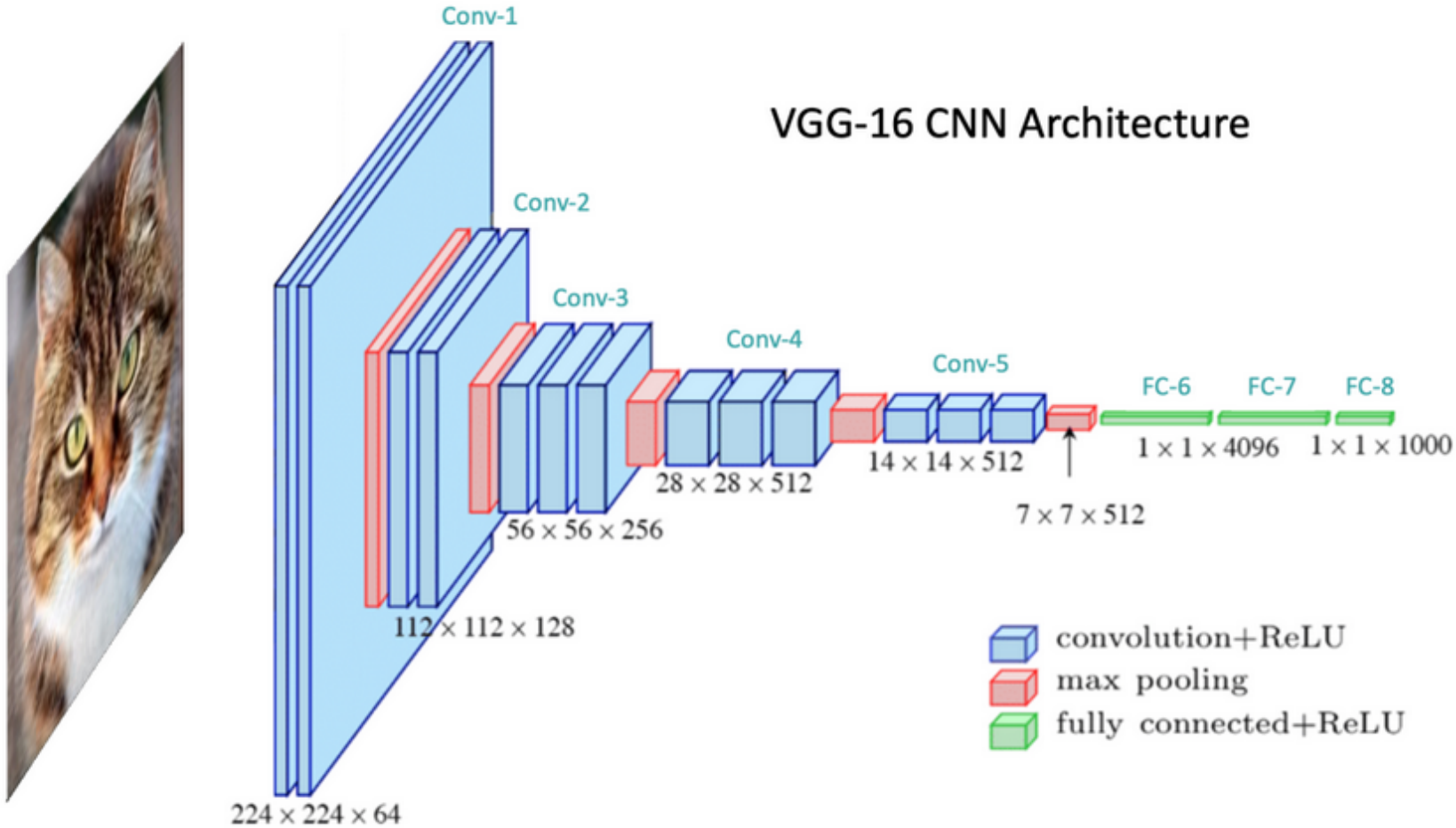


Figure: VGG-16 Architecture

Concepts

Concepts in Neural Networks

- Classification
- Transfer Learning
- Encoder-Decoder Architecture
- Siamese Nets
- Generational Networks
- Auxiliary task

Thanks for your attention

Additional sources

- https://pytorch.org/tutorials/
- https://youtu.be/aircAruvnKk
- https://developers.google.com/machine-learning/crash-course
- https://playground.tensorflow.org
- https://github.com/PawelGlr/MIOwAD_labs (self promotion ^^)