CIKLUM | SPEAKER'S CORNER

SEPTEMBER, 21

18:00 (EEST)

ZOOM

ENGLISH

NEURAL NETWORK THAT CAN LEARN: CREATING FROM SCRATCH

LUCIAN GRUIA

PRINCIPAL JAVA DEVELOPER AT CIKLUM

Meet Ciklum

CIKLUM

We empower companies to meet their digital initiatives by providing end-to-end software, integration and innovation services

Our services:

- Digital Commerce
- Intelligent Automation
- ITO & Managed Service
- Data & Analytics
- Cloud
- Engineering Services

2002 founded

4000+ professionals

20+ offices

300+ clients

Leading companies choose us:

JUST EAT Takeaway.com · METRO MARKETS · eToro · ZURICH · pay · Greensill · KANTAR RETAIL · dacadoo

CIKLUM

Speaker:

Lucian Gruia

Principal Java Developer at Ciklum

- Romania, Bucharest
- 4 years @Ciklum
- 11+ years in software engineering
- Telecom, Fintech, Aerospace

🌐 luciangruia.ro

Agenda

- We will understand:
  - What is **Learning**
  - How **Neurons** work
  - How to design a **Neural Network**

- We will code:
  - A **Neuron**
  - A **Neural Network**
  - **Train** the Neural Network
  - **Test** and **Visualise** our Neural Network

Neural Networks

Human brain contains **~ 86 bln neurons**, connected by **synapses**. (source)

Synapses transmit electrochemical **signals** between neurons.

One neuron "**fires**" to the other neurons → The next neuron **fires forward** a different signal
* if it's sufficient input signal → This results into a chain reaction that we call "**cognition**"

All of our thoughts, memories, reasoning, understanding, learning are the result of a **neural network** activity.

It's fascinating that today's electronics engineering enables us to replicate this processes on machines.

Neural Networks on machines

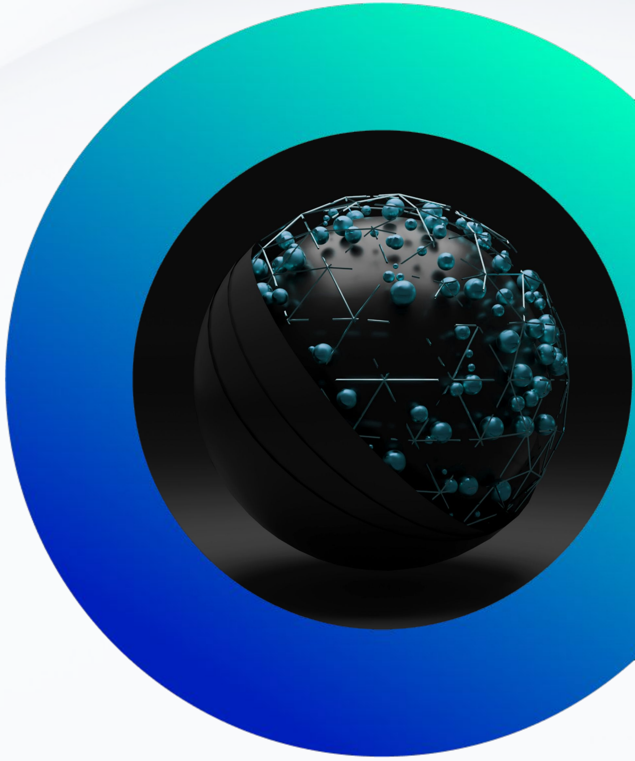
NNs are at the very heart of AI

Foundational to machine learning, neural networks enable unprecedented data processing sophistication & transform **artificial intelligence** by mimicking human brain structures.

Beyond their vast applications, as engineer we still should ask ourselves:

- **Who** develops AI?
- **How** do engineers/scientists design Neural Networks' architecture?
- **What** are the outstanding challenges in Neural Networks?

Understanding Neural Networks isn't merely academic - it's about understanding **the essence** of our digital future.



Software Development

Traditional

- Rule-based
- Explicit
- Algorithmic
- Deterministic
- Hand-coded
- Procedural
- Imperative
- Conventional

Machine Learning

- Data-Driven
- Generalistic
- Adaptive
- Probabilistic
- Implicit logic
- High dimensionality
- Feature engineering
- Black-box

Understanding Learning (_P1)

Binary Blitz

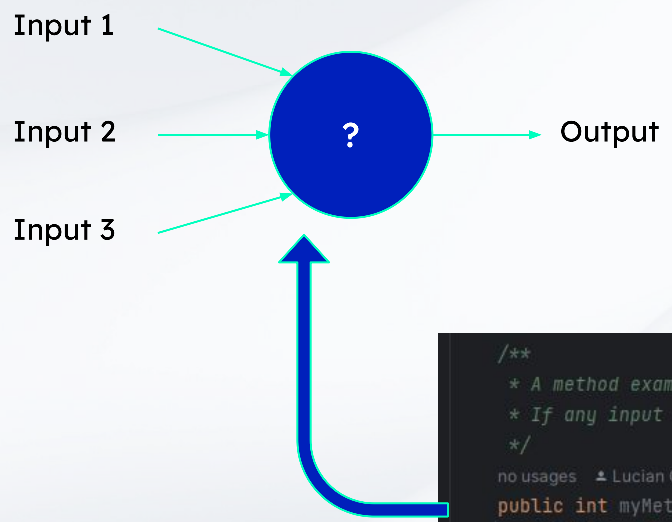
	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0
Example 5	1	0	0	?

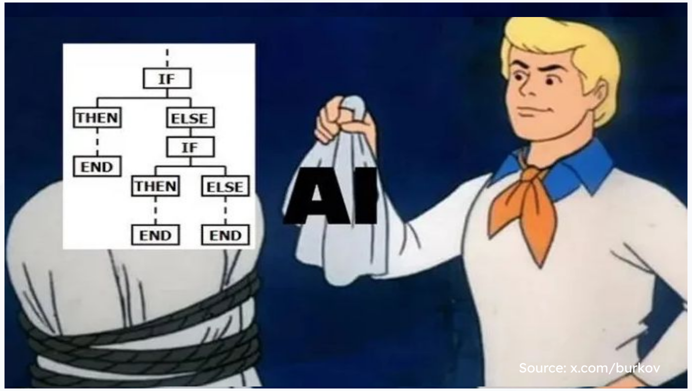
→ Supervised Learning

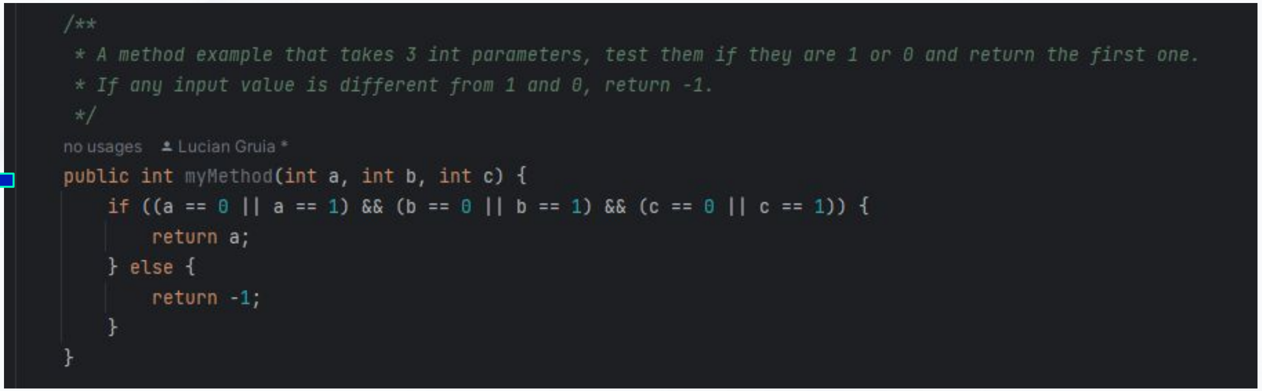
Other types of learning (in ML):

- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning
- Self-supervised Learning
- Transfer Learning
- Few-shot, One-shot, and Zero-shot Learning

Traditional software development

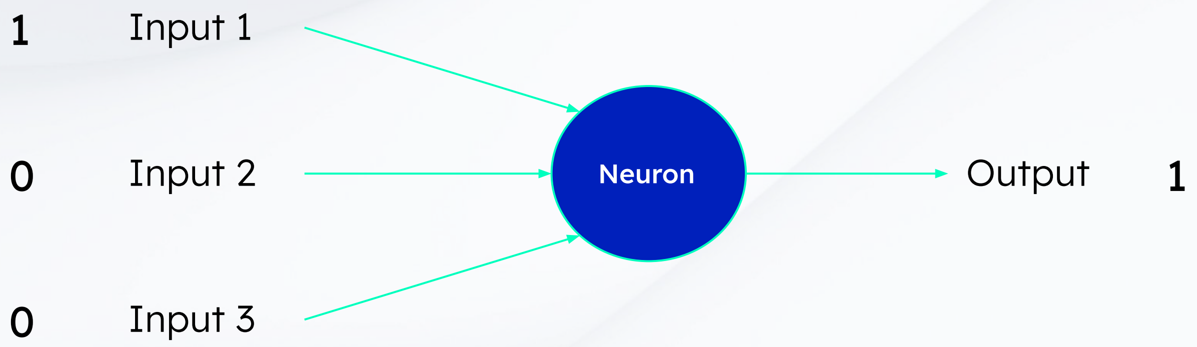




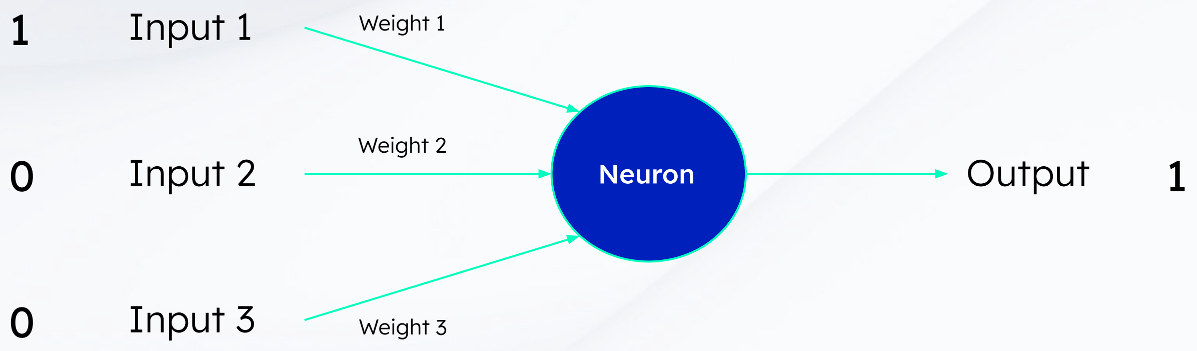


```
/**
 * A method example that takes 3 int parameters, test them if they are 1 or 0 and return the first one.
 * If any input value is different from 1 and 0, return -1.
 */
no usages   Lucian Gruia *
public int myMethod(int a, int b, int c) {
    if ((a == 0 || a == 1) && (b == 0 || b == 1) && (c == 0 || c == 1)) {
        return a;
    } else {
        return -1;
    }
}
```

Design the Architecture _P1



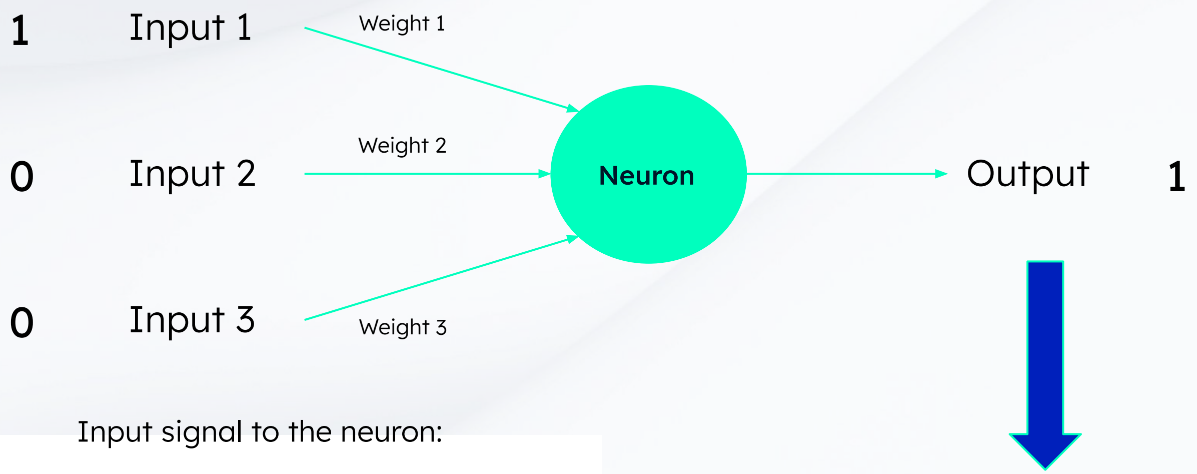
Weights



Input signal to the neuron:
$Input_1 * Weight_1 + Input_2 * Weight_2 + Input_3 * Weight_3$

$$\sum_{i=1}^{n} (Input_i \cdot Weight_i) + \underbrace{b}_{\text{bias}}$$

Activation Function



Input signal to the neuron:

$$\sum_{i=1}^{n}(Input_i \cdot Weight_i) + b$$

ActivationFunction(IncomingSignal)

Activation Function

Neuron

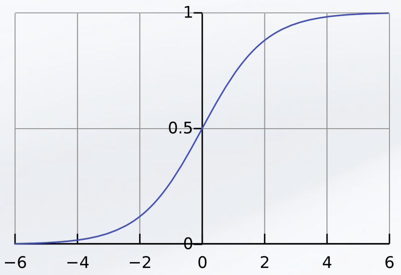
CIKLUM

Activation function **translates** the **input** signal **to** the **output**.

For this problem, we will use **Sigmoid** function.

- It has codomain in **(0;1)** -> asymptotic.
- It helps us to estimate the **probability** of an event.

$$\frac{1}{1+e^{-x}} \rightarrow \frac{1}{1+e^{-(NeuronInput)}}$$



Neuron Input:

$$\sum_{i=1}^{n}(Input_i \cdot Weight_i) + b$$

Neuron Output:

$$\frac{1}{1+e^{-(\sum_{i=1}^{n}(Input_i \cdot Weight_i)+b)}}$$

Activation Functions

Common Activation functions in NN:

- **Sigmoid**
- Hyperbolic Tangent (tanh)
- Rectified Linear Unit (ReLU)
- Exponential Linear Unit (ELU)
- **Softmax**
- Swish
- Mish

Loss function

Loss (also known as error/cost) function measures the deviation between the **Predicted** Output and the **Expected** (Correct) Output values.

In other words, it measures the **accuracy** of neural network's reasoning.

Mean Squared Error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n}(\mathrm{CorrectOutput}_i - \mathit{PredictedOutput}_i)^2$$

Common Loss functions in NN:

- **MSE**
- Cross-Entropy Loss (Log Loss)
- Hinge Loss
- Huber Loss

Training

➢ Assign **random** weights. (It is important to be random.)

Loop:

1. **Predict** the Output for a Training Set example.
   [Pass The Inputs to the Activation Function to get the Output.]

2. **Calculate the loss**.
   [The difference between the Predicted Output and the Training Set Output.]

3. **Adjust the weights** to reduce the loss.
   [This is where ***the magic*** happens!]

CIKLUM

Adjusting the weights

Adjusting the weights is the process of **fine-tuning the parameters** of a neural network to **minimize the loss**.

Weights are updated iteratively using optimization algorithms that rely on calculated **gradients** - in our case, gradient descent.

We need to compute the **gradient** of the loss function with respect to each weight. This indicates the direction and magnitude of change needed.

Learning rate determines the size of the steps we take based on the gradient, balancing between fast convergence and the risk of overshooting the minimum.

Gradient descent moves in the direction opposite to the gradient to minimize the loss, with the step size determined by the learning rate.

CIKLUM

Recap

Recap

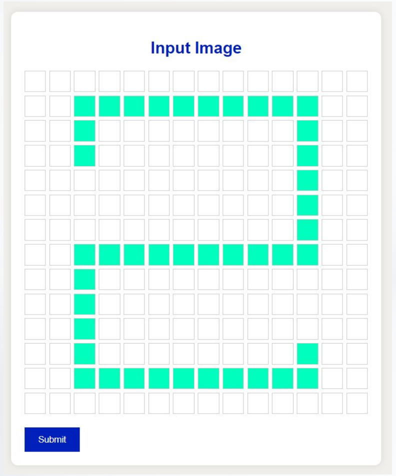
1.	Understand the problem -> Design the Architecture	(there is no one fits all here)
2.	Initialize Weights with random values	(important)
3.	Choose and implement the Activation Function	(decides if the neuron *fires* forward)
4.	Choose and implement the Loss Function	(checking how wrong is the prediction)
5.	Make a Prediction	(this returns *output(input)*)
6.	Calculate the Loss	(evaluate the current state of the NN)
7.	Calculate the Gradient	(derivative of the loss)
8.	Adjust the Weights to minimize the Loss	(here the learning emerges)
9.	Make another Prediction	. . .

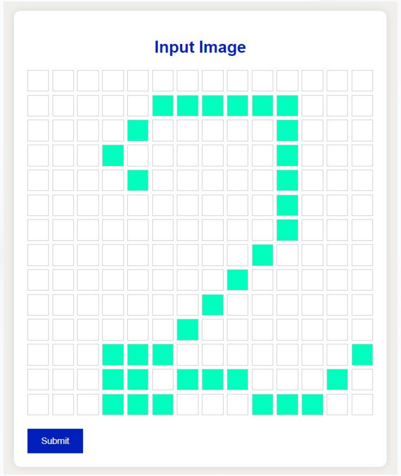
Recognizing complex patterns (_P2)

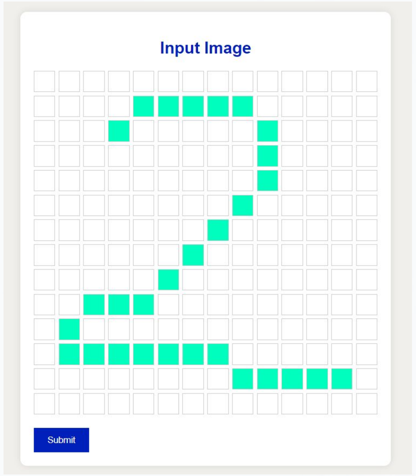
Handwriting
Handwriting

Handwriting
Handwriting

Handwriting
Handwriting







NN Architectures

Common architectures:

- Feedforward Neural Networks (**FNN**) or Multi-Layer Perceptrons (MLP)
- Convolutional Neural Networks (CNN)
- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM) Networks
- Gated Recurrent Units (GRU)
- Radial Basis Function Neural Networks (RBFNN)
- Self Organizing Maps (SOM)
- Deep Belief Networks (DBN)
- Autoencoders
- Generative Adversarial Networks (GAN)
- **Transformer Architectures**
- Neural Turing Machines (NTM) and Differentiable Neural Computers (DNC)
- Capsule Networks
- Sequence-to-Sequence Models
- Hybrid Models

Our NN Architecture

- 28 x 28, input image => 784 neurons
  [0..9] ∩ ℕ, 10 possible results => 10 neurons
- 2 hidden layers => 2 x 16 neurons
  They increase depth of the neural network.
  That’s why we call it “*deep learning*”!
- Fully-connected FNN, total 826 neurons
  (only 42 have activation function)
- Input Layer -> First Hidden
  784 × 16 = 12 544 weights + 16 biases = **12 560 parameters**
- First -> Second Hidden Layer
  16 x 16 = 256 weights + 16 biases = **272 parameters**
- Second Hidden Layer -> Output Layer
  16 x 10 = 160 weights + 10 biases = **170 parameters**

Total trainable parameters

13 002

Facts about NNs

- LLMs' ability to learn how to learn to use tools emerges at ~ **775 M** parameters and not before. (source)
- Popular LLM models by number of parameters (source):

BERT	340 M	(Google, 2018)
GPT-3	175 B	(OpenAI, 2020)
Claude	50 B	(Anthropic, Dec 2021)
Bloom	176 B	(Hugging Face, July 2022)
GPT-4	1.76 T	(OpenAI, March 2023)
Falcon	180 B	(TII, March 2023)
PanGu-Σ	1,085 T	(Huawei, March 2023)
Llama2	70 B	(Meta, July 2023)

- Human brain, **very rough** estimate. 86 B neurons, 7000 synapses each => > 600 T
- Just to emphasize, the number of parameters is not the only measure of a neural network. Its architecture is actually very important.

Future

- **Energy-Efficient Models**: Reducing the power consumption of training and inference.
- **Improved Generalization**: Models that can perform well across diverse tasks.
- **General AI**: Deep learning and neural networks will likely play a significant role in the pursuit of an AI that can perform any intellectual task that a human can do.
- **Neuromorphic Computing**: Researchers are developing neuromorphic chips that imitate the brain's architecture.
- **Continual Learning**: Instead of static models, future networks might continuously learn and adapt throughout their lifecycle, much like humans do.
- **Brain-Computer Interfaces**: Several companies now aim to directly interface the brain with computers. Such technology, combined with advanced neural networks, could lead to direct communication between the human brains and machines.
- **Quantum Neural Networks**: Computational NN models which are based on the principles of quantum mechanics.

History

1748

Leonhard Euler's work on "*Introductio in analysin infinitorum*"

1943

McCulloch & Pitts: Warren McCulloch and Walter Pitts introduced a simplified computational model of a **neuron** that was binary and deterministic.

1949

Hebbian Learning: Donald Hebb proposed a learning mechanism based on the idea that when two **neurons fire together**, the connection between them strengthens.

1957

Perceptron: Frank Rosenblatt introduced the perceptron, a type of artificial neuron, and an associated learning algorithm.

1970s-1980s

Backpropagation algorithm, was introduced by researchers like Rumelhart, Hinton, and Williams

2000s - present

Deep Learning: With the advent of more powerful computing hardware & the availability of large datasets, neural networks have seen a resurgence.

CIKLUM

Learn more

Some nice books

- Sequence to Sequence Learning with Neural Networks
  [Google 2014]
- Understanding Machine Learning: From Theory to Algorithms
  [Cambridge University Press 2014]
- Attention is all you need
  [Google 2017]
- Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm
  [DeepMind 2017]

Online resources:

- https://github.com/mrsaeeddev/free-ai-resources
- https://www.deeplearning.ai/courses/

Let’s talk about NN and AI

Why am I interested in NN

- They exhibit superior abilities, such as **Learning**. These abilities emerge only when neural networks' components are combined.

- Then more complex philosophical questions arise:
    - Can neural networks truly **understand**?
    - Are neural networks **simulating** or **replicating** human cognition?
    - Can machines become **conscious** or sentient one day?
    - Can we build **free will**?
    - **How do we know** what a neural network knows?
    - How do we handle the problems of **ethics, accountability, morality, rights** in AI?

CIKLUM

Any questions?

Share your feedback!

CIKLUM

Thank you!

luciangruia.ro

lugr@ciklum.com | www.ciklum.com