

RAD

rad.itu.dk

IT UNIVERSITY OF CPH



[DASYA -- DASYA \(itu.dk\)](http://DASYA--DASYA.itu.dk)

CPU, GPU, FPGA, Accelerator

Ehsan Yousefzadeh-Asl-Miandoab

(ehyo@itu.dk)

Data-Intensive Applications and Systems (DASYA)



Road Map

- **Computing**
- Processors
- How do electrons work for us?!
- CPUs
- GPUs
- FPGAs
- Accelerators
- Tradeoff of processors

Computing

- Definition
 1. Transforming the input data into the desired output data.
 2. Gaining knowledge (insight)
 3. Using computer technology to complete a goal-oriented task.
 - Examples:
 - Weather forecast
 - Market analysis, price prediction
 - Computer-aided medical diagnosis
-

```
mirror_mod = modifier_ob.  
#set mirror object to mirror.  
mirror_mod.mirror_object =  
_operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
_operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
_operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```


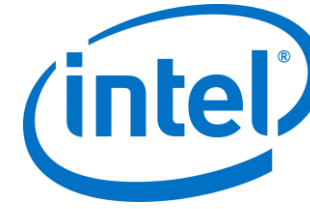


Road Map

- Computing
- **Processors**
- How do electrons work for us?!
- Tradeoff of processors
- CPU
- GPU
- FPGA
- Accelerator

Processors

- Devices that make the computing possible
- **By making electrons run (work) for us**
- Different types and vendors
 - CPUs
 - GPUs
 - FPGAs
 - Accelerators



[Products | Coral](#)

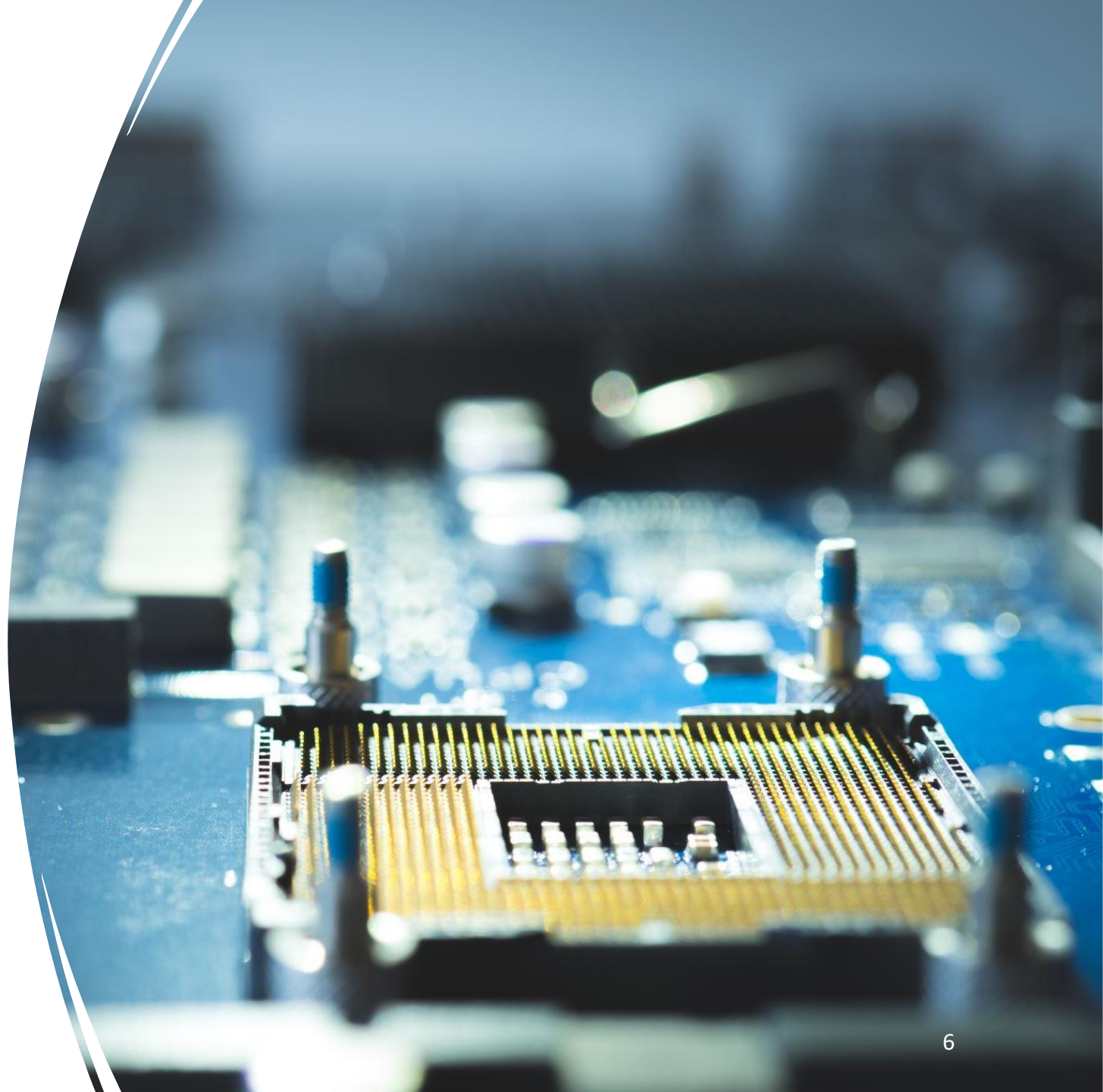


[GroqChat](#)



Road Map

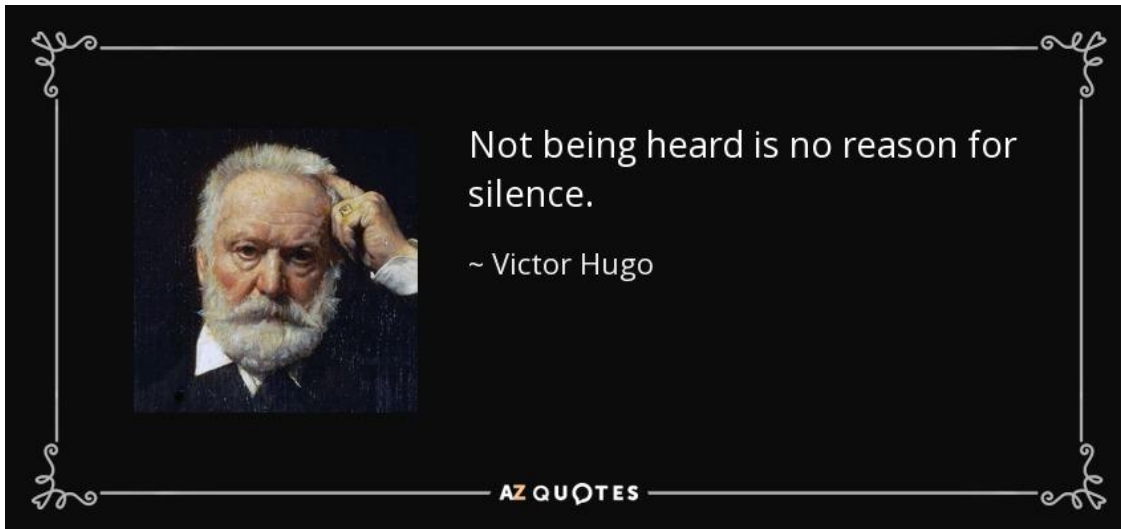
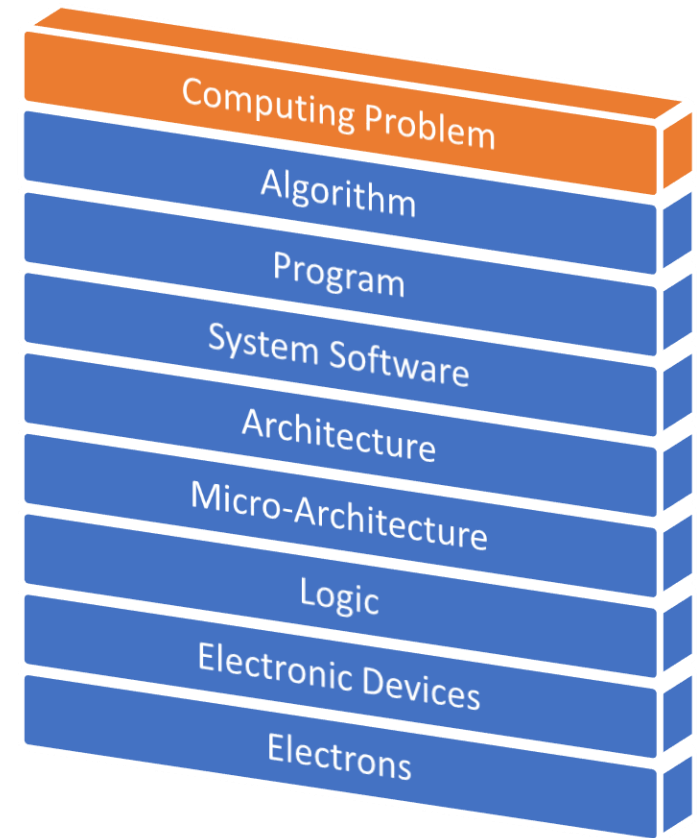
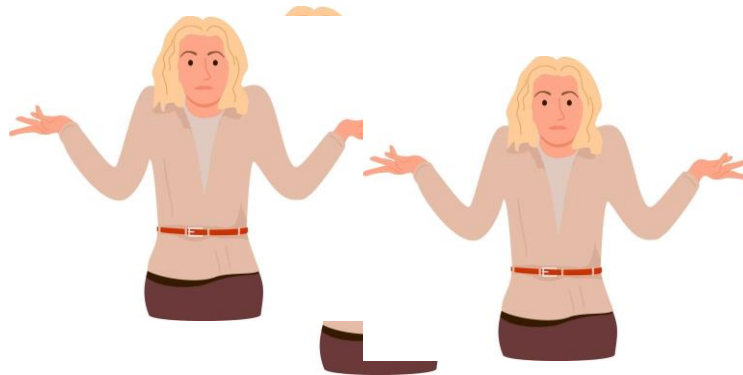
- Computing
- Processors
- **How do electrons work for us?**
- CPU
- GPU
- FPGA
- Accelerator
- Tradeoff of processors



Electrons I need your help!

Can you apply a filter to my photo?
Please 😊

Electrons' reaction!

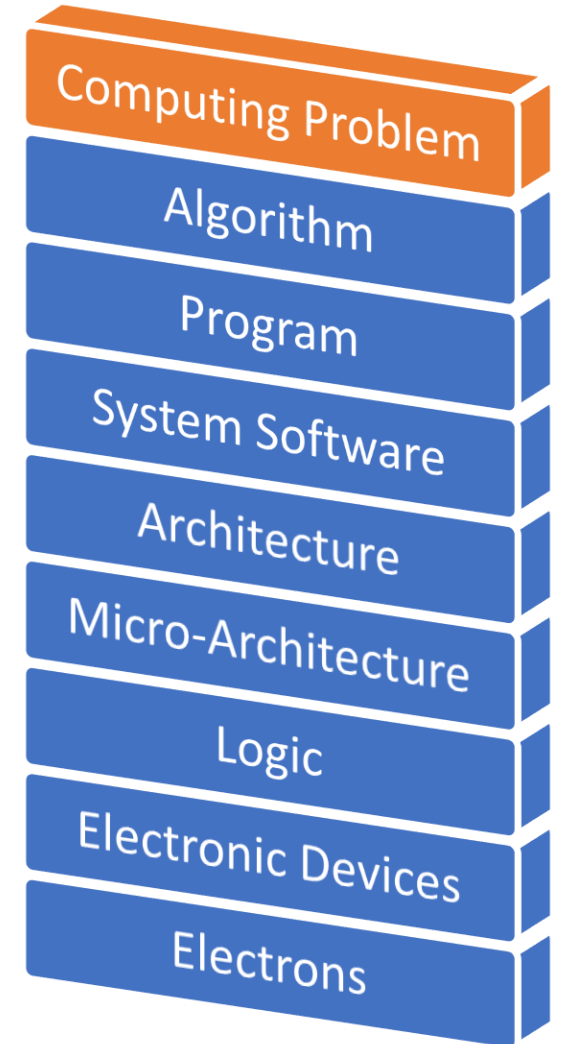


**We need a translator who can speak
Electrons' language 😊**

Talking in the language of Electrons

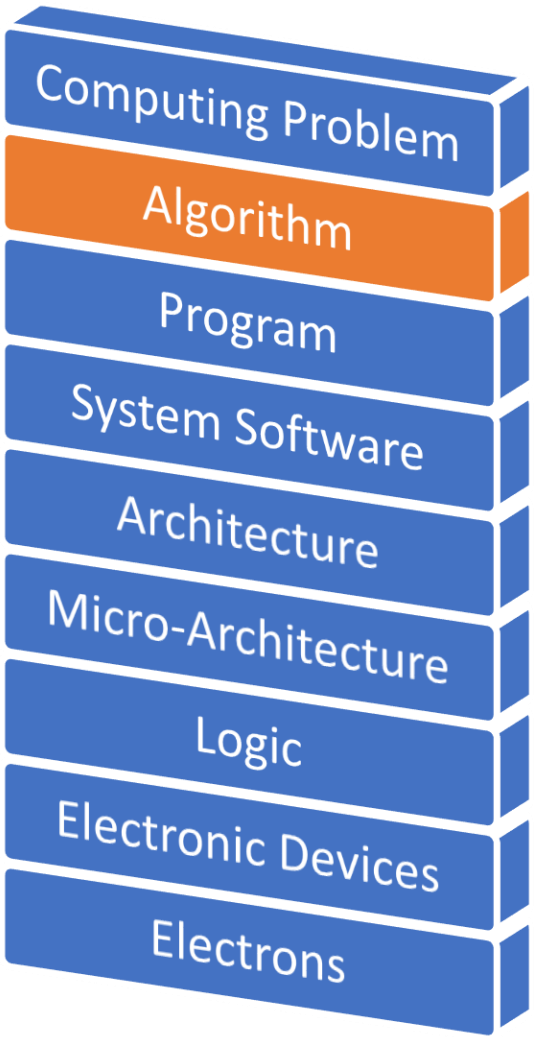
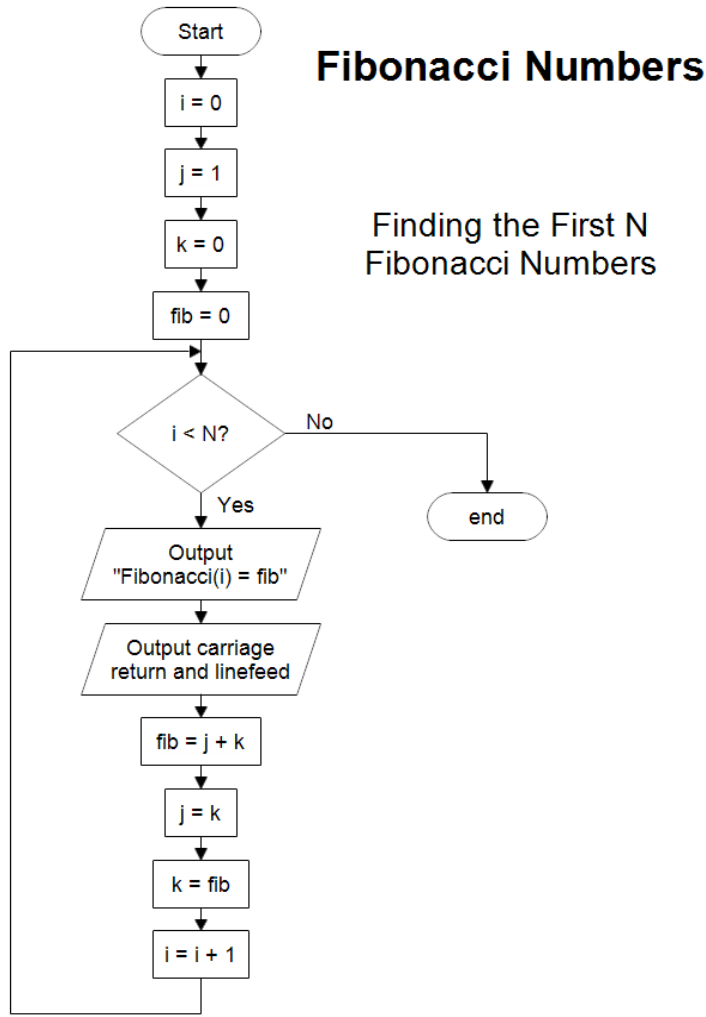
Examples:

1. Calculating the first 100 numbers of Fibonacci series
2. Training a machine learning model



Transformation Hierarchy

Talking in the language of Electrons



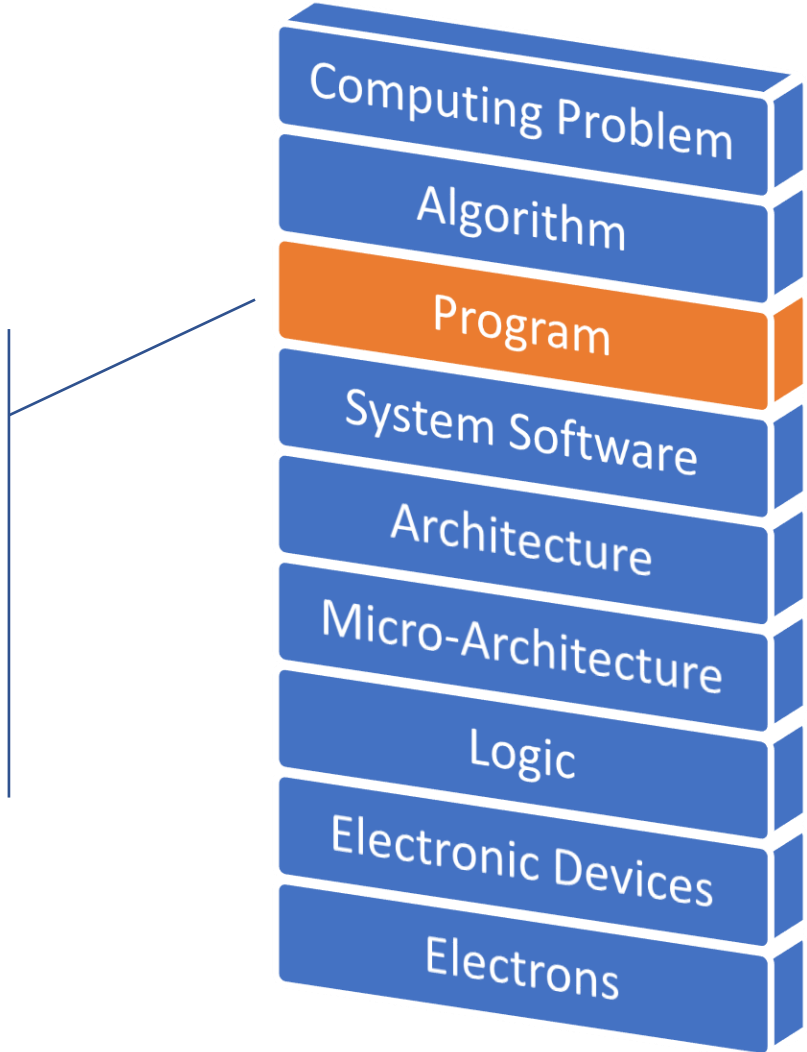
Transformation Hierarchy

Talking in the language of Electrons

```
Program to display the Fibonacci sequence up to n-th term
nterms = int(input("How many terms? "))

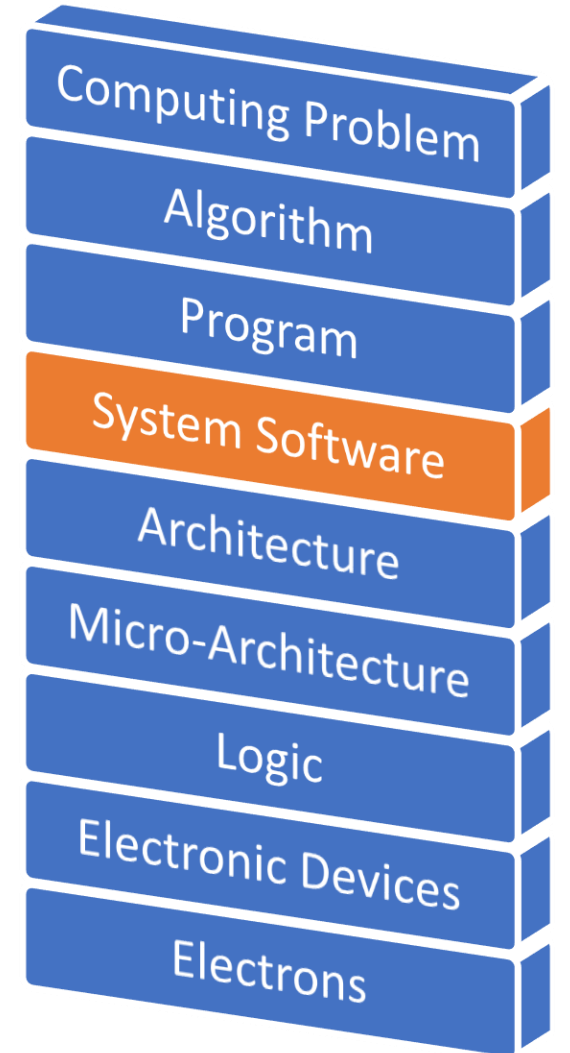
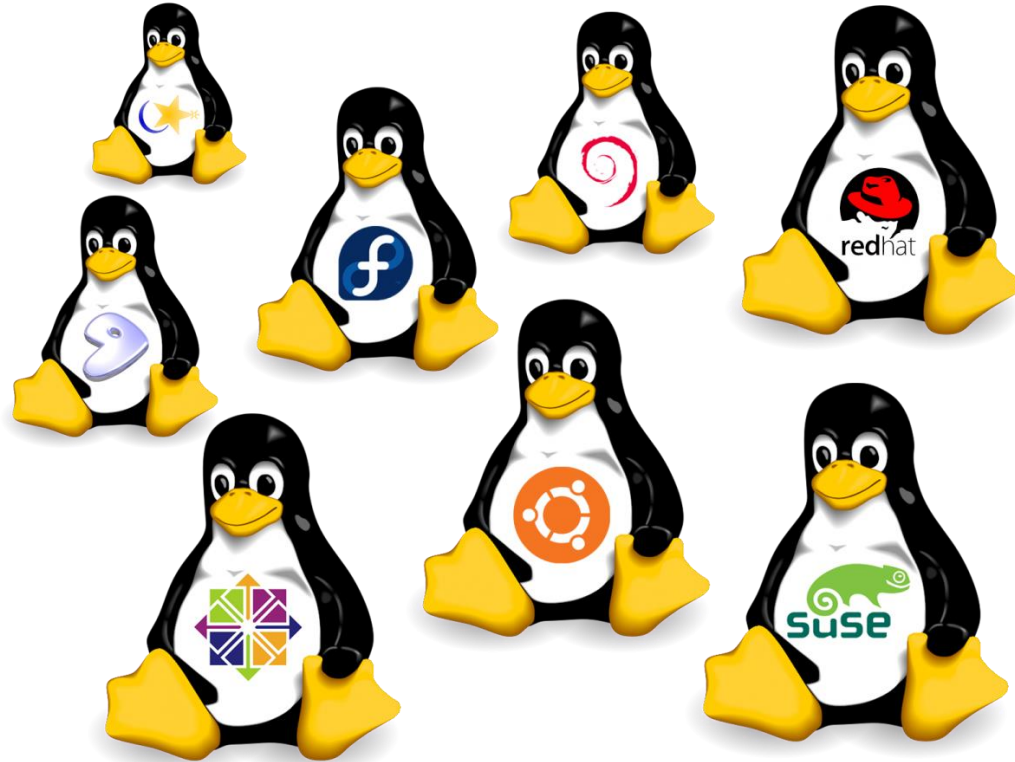
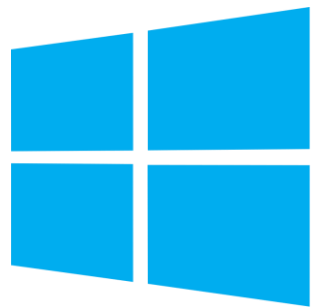
# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
    print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
    print("Fibonacci sequence upto",nterms,":")
    print(n1)
# generate fibonacci sequence
else:
    print("Fibonacci sequence:")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        # update values
        n1 = n2
        n2 = nth
        count += 1
```



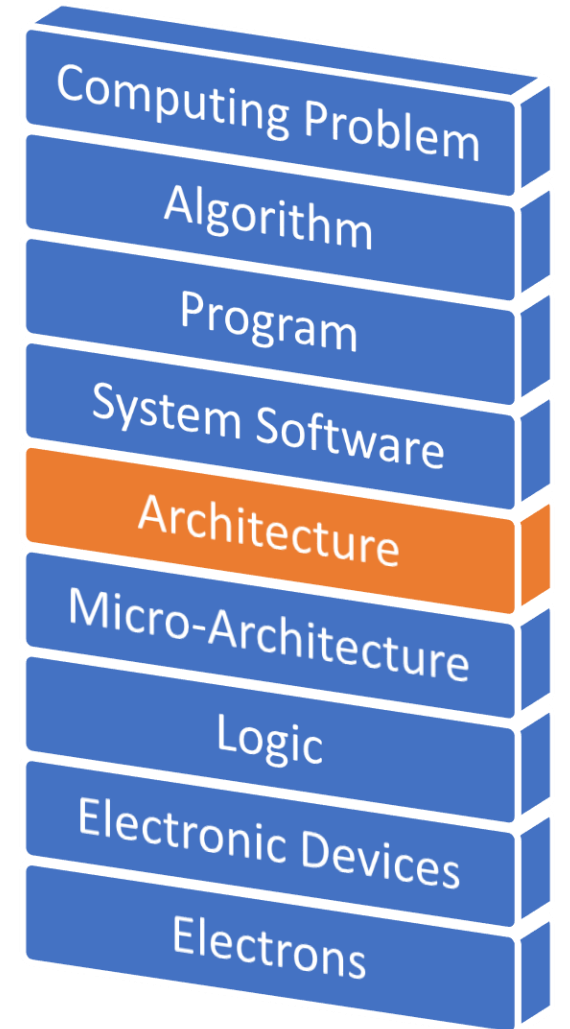
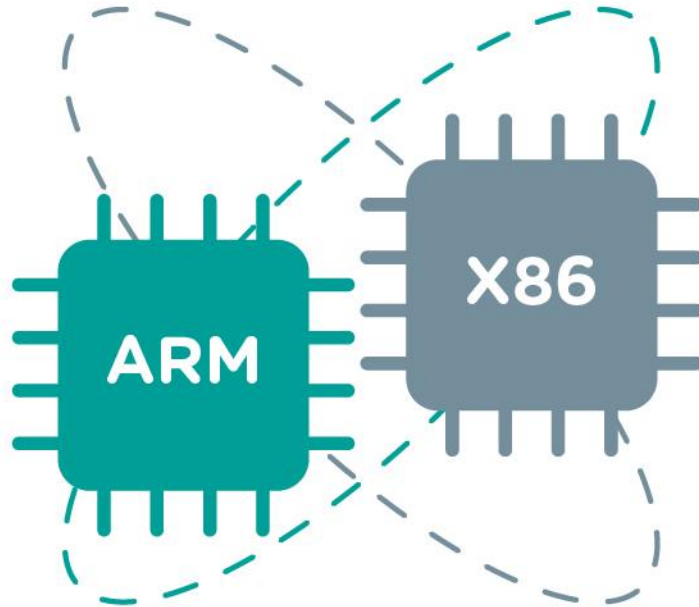
Transformation Hierarchy

Talking in the language of Electrons



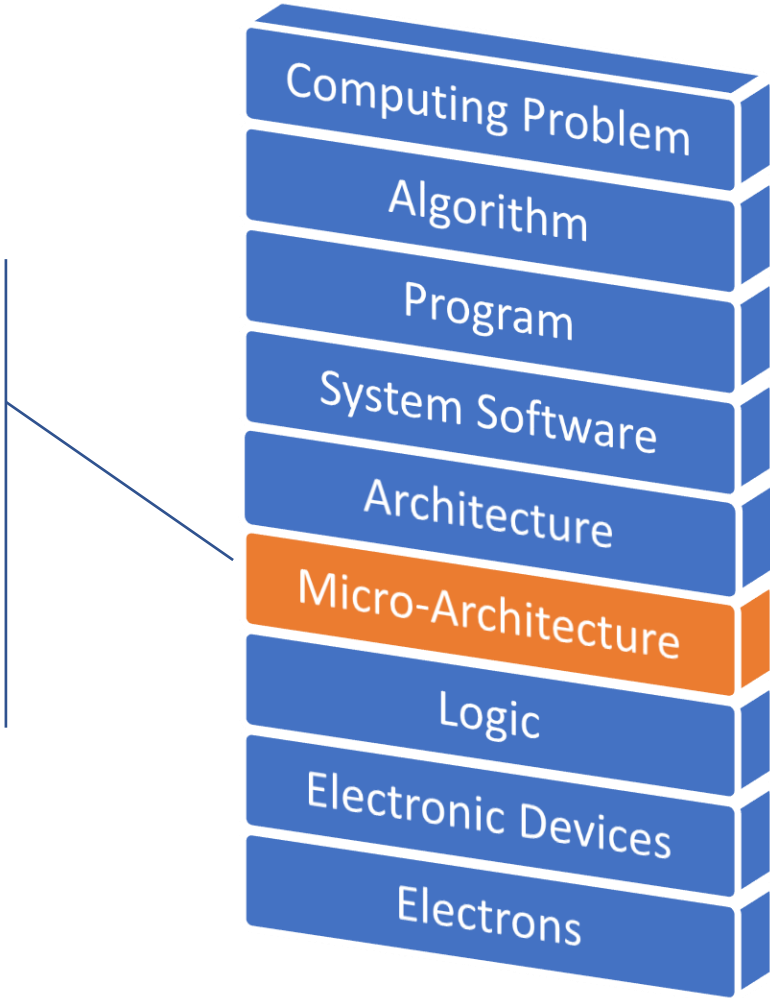
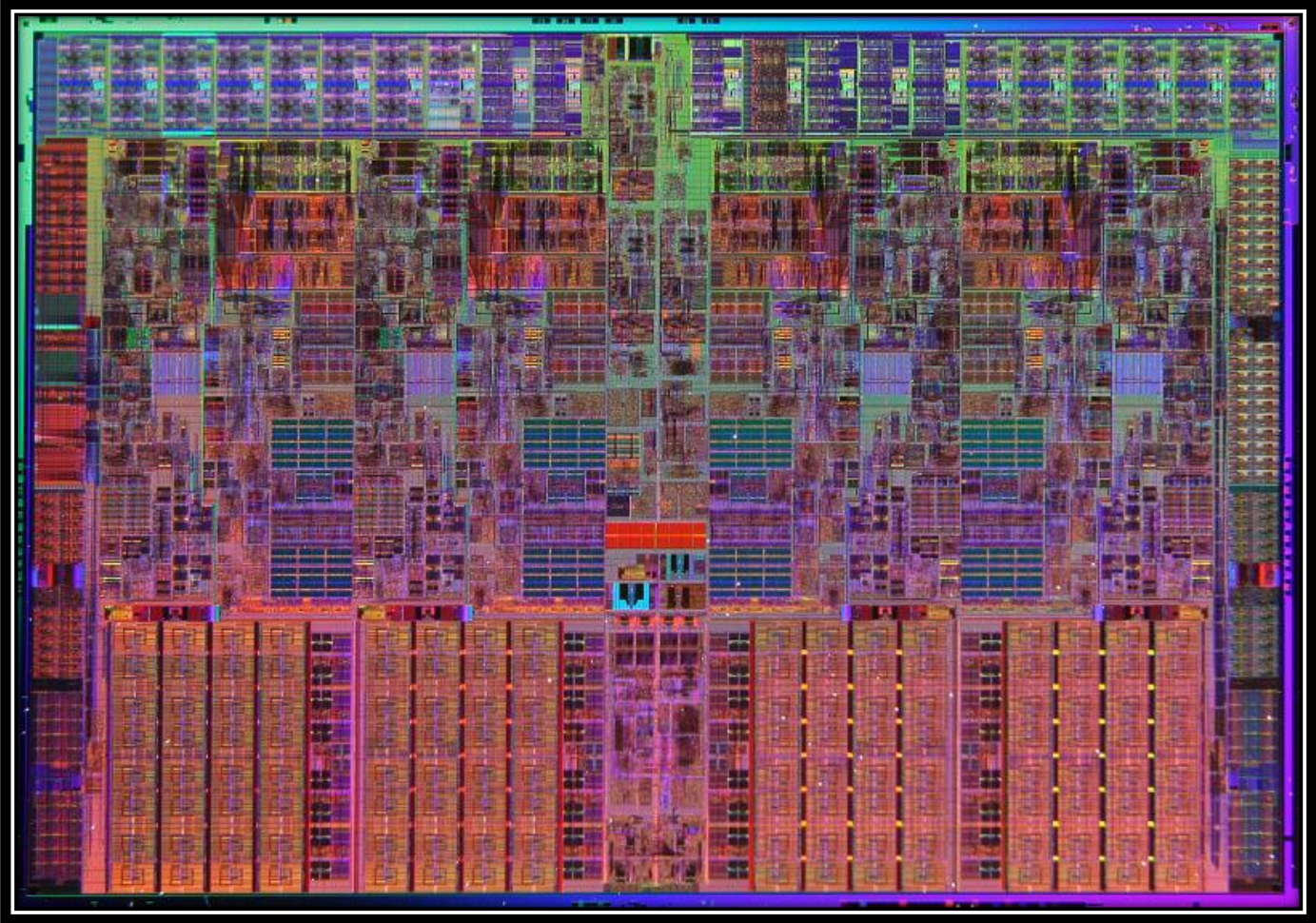
Transformation Hierarchy

Talking in the language of Electrons



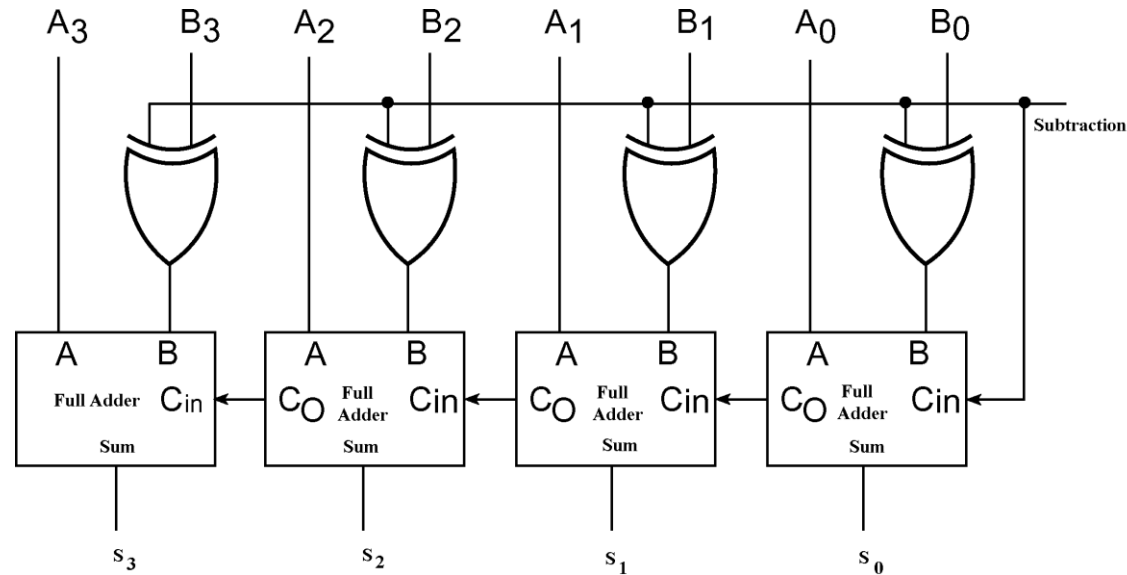
Transformation Hierarchy

Talking in the language of Electrons

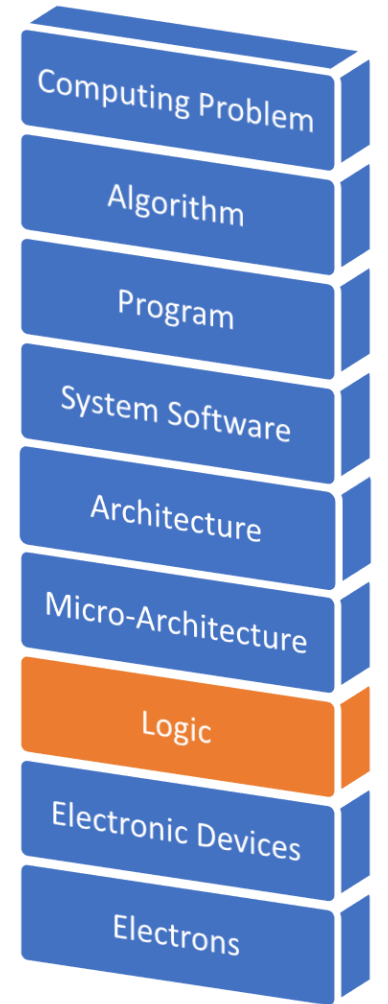


Transformation Hierarchy

Talking in the language of Electrons

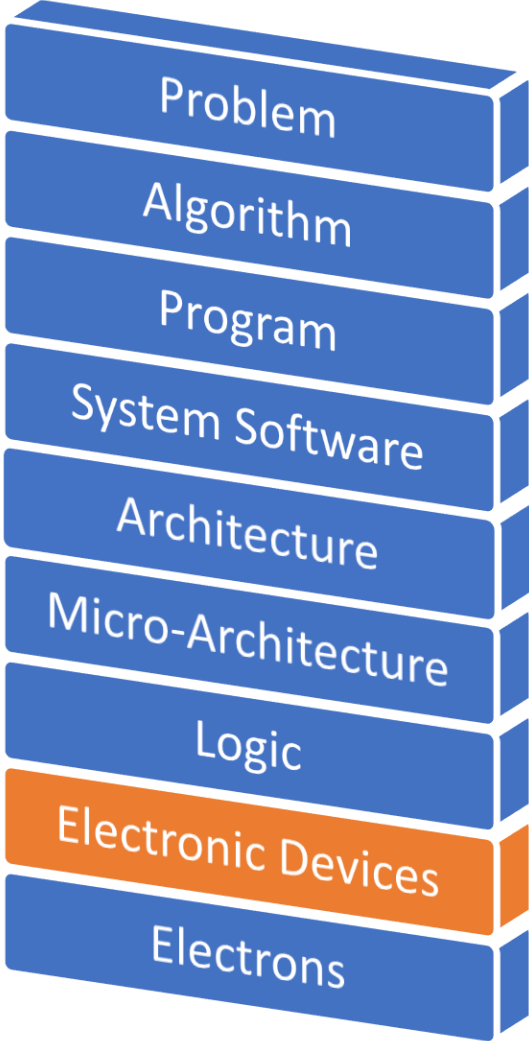
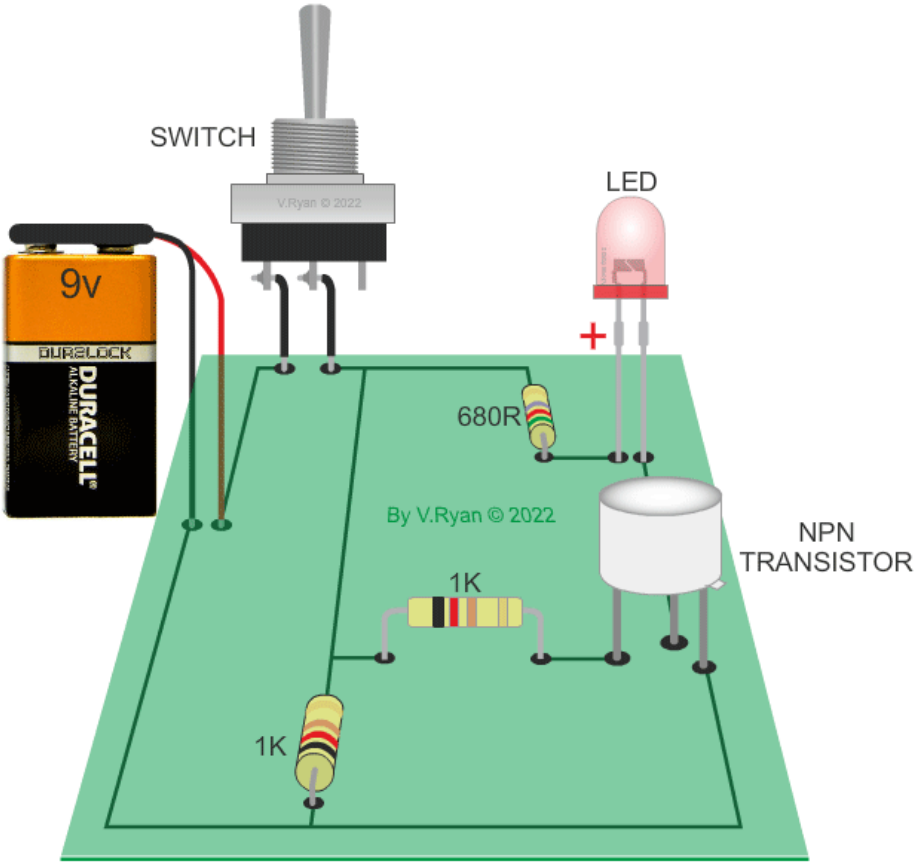


d1	d2	sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Transformation Hierarchy

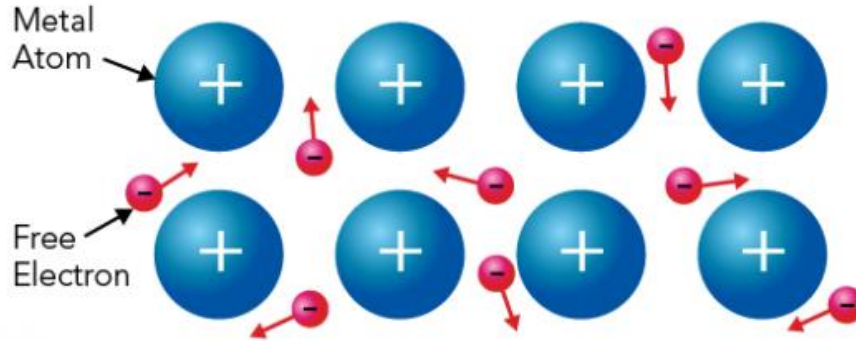
Talking in the language of Electrons



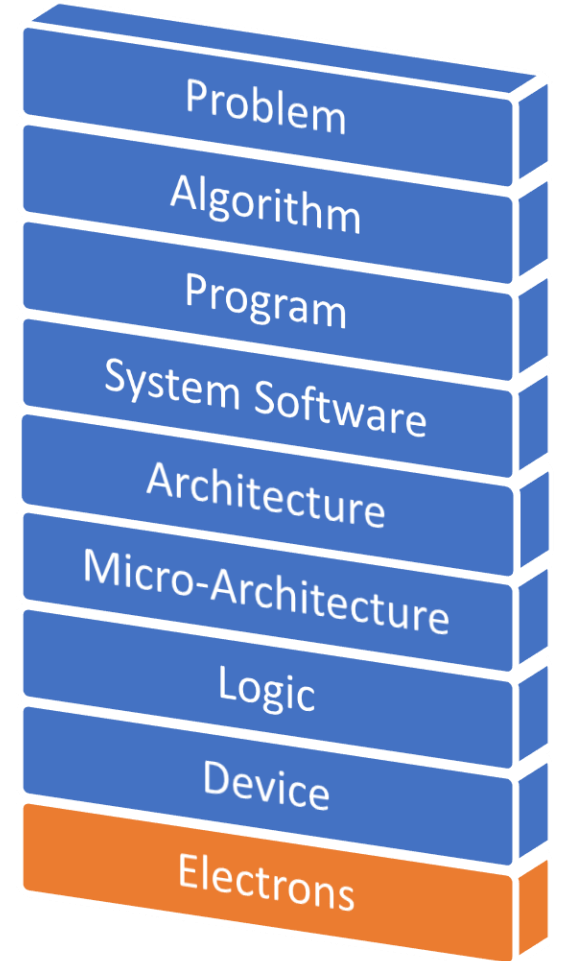
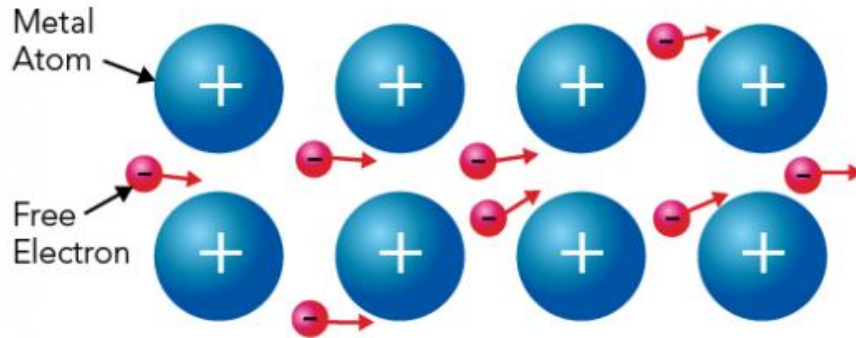
Transformation Hierarchy

Talking in the language of Electrons

Metal under normal conditions



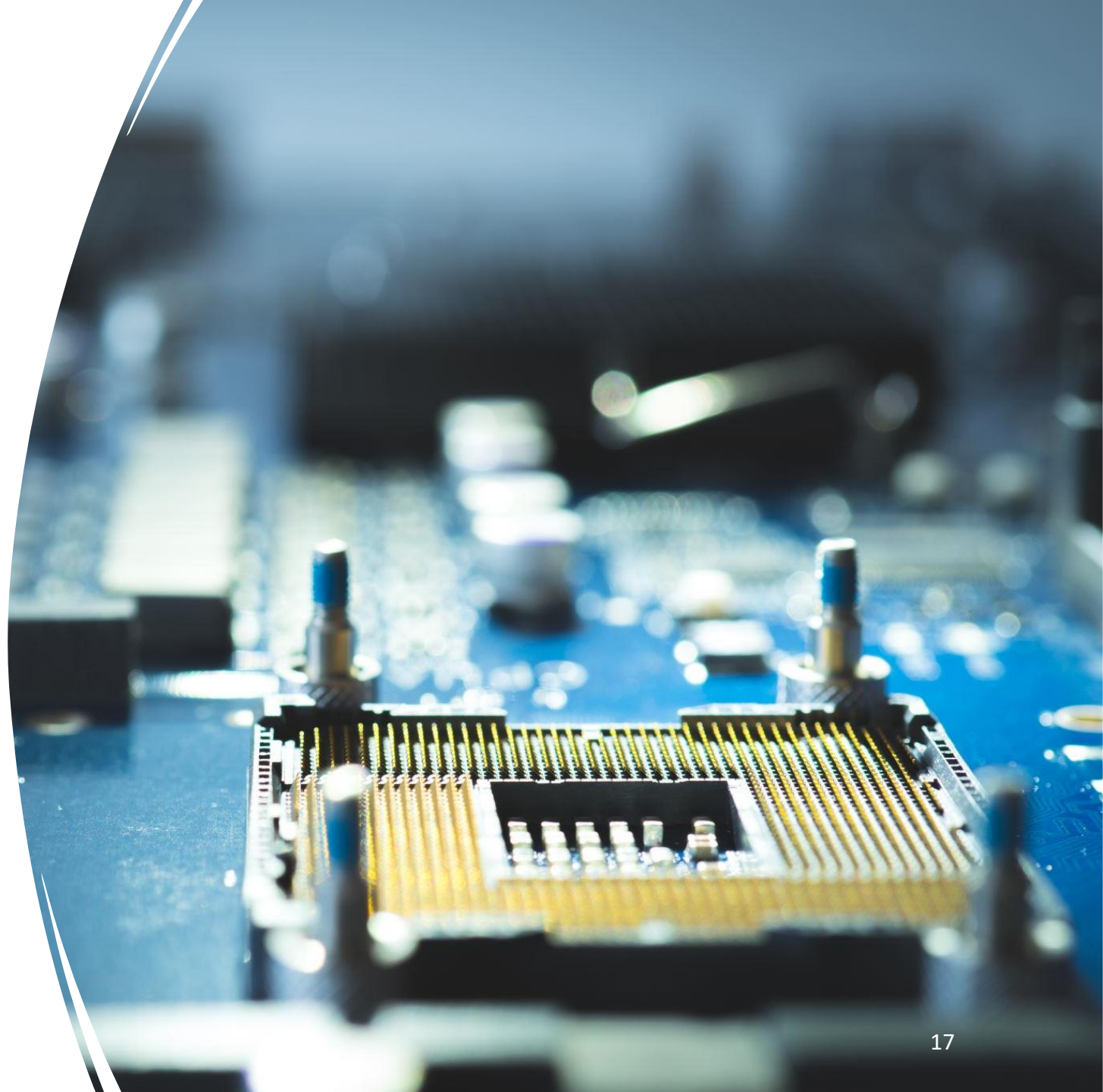
Metal in an electric field




Transformation Hierarchy


Road Map


- Computing
- Processors
- How do electrons work for us?!
- **CPU**
- GPU
- FPGA
- Accelerator
- Tradeoff of processors



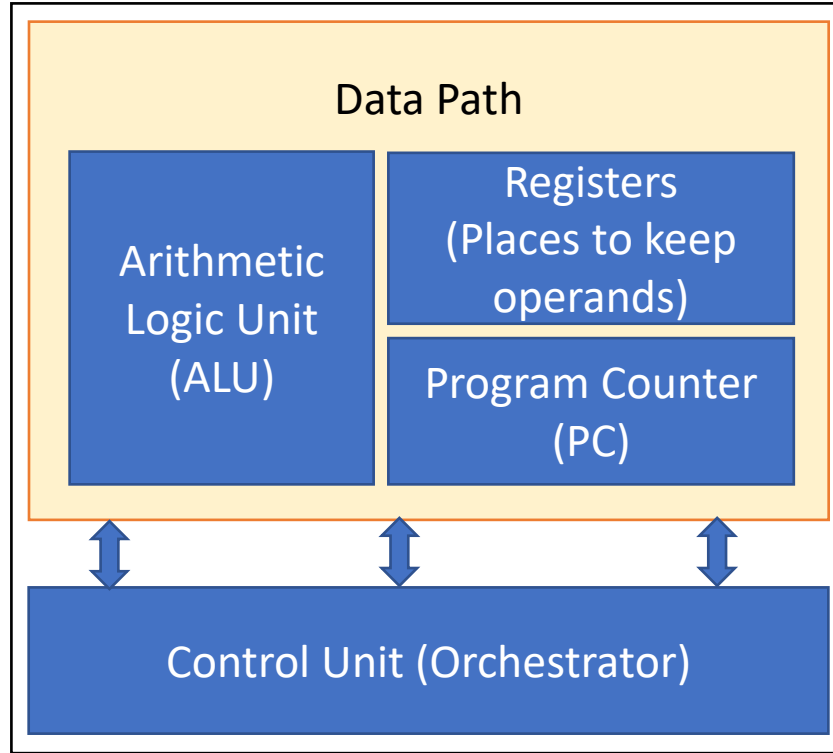
CPU Processor


Sequential Processor

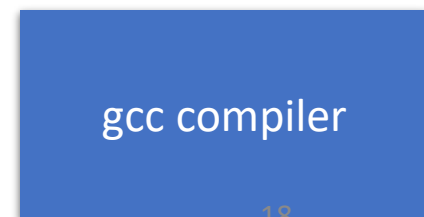
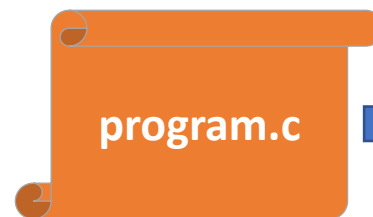
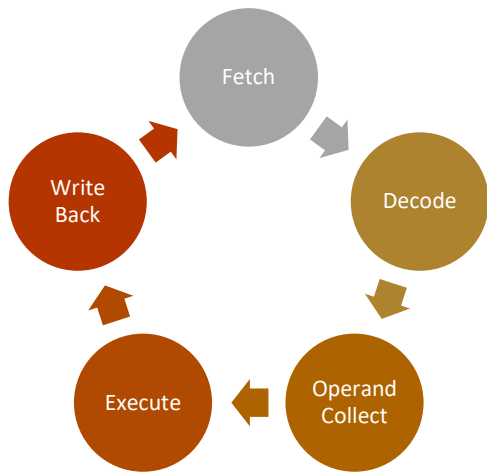
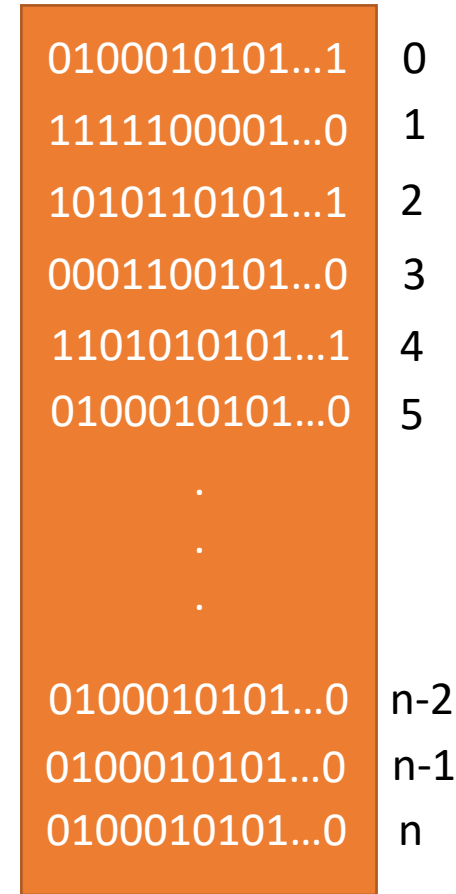

Latency-oriented
(the time taking to execute an instruction)


High Working Frequency
(example **4GHz**)

CPU



RAM



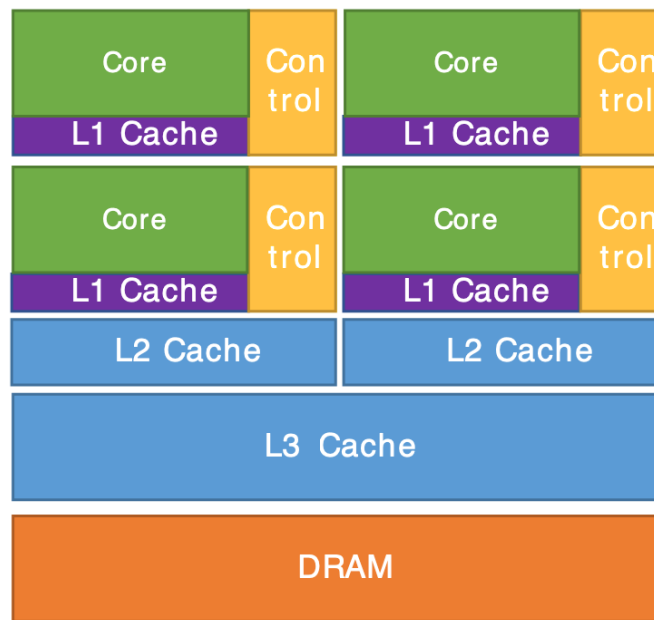
Road Map

- Computing
- Processors
- How do electrons work for us?!
- CPU
- **GPU**
- FPGA
- Accelerator
- Tradeoff of processors

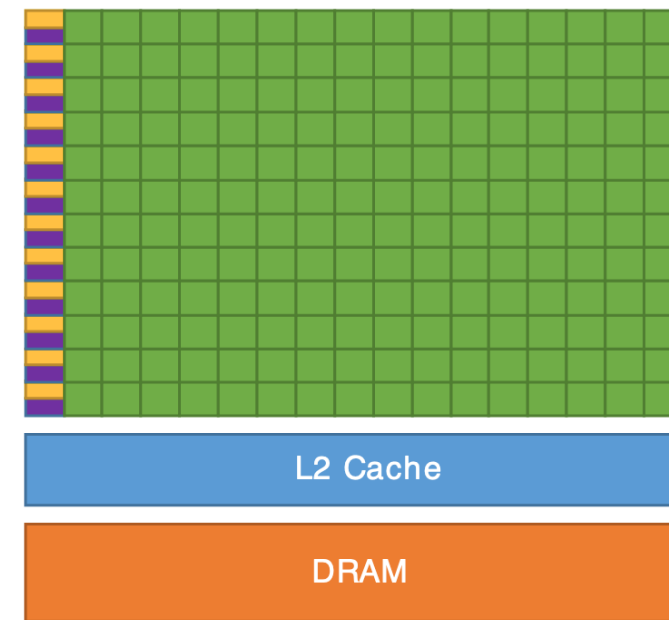


GPU

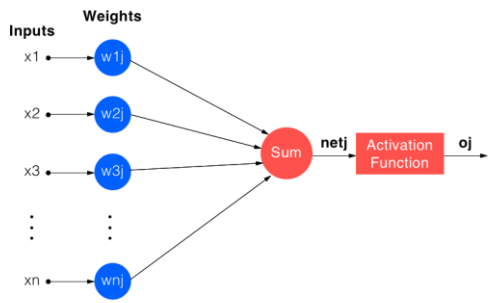
- Parallel Processor
- Throughput-oriented
- Low Working Frequency



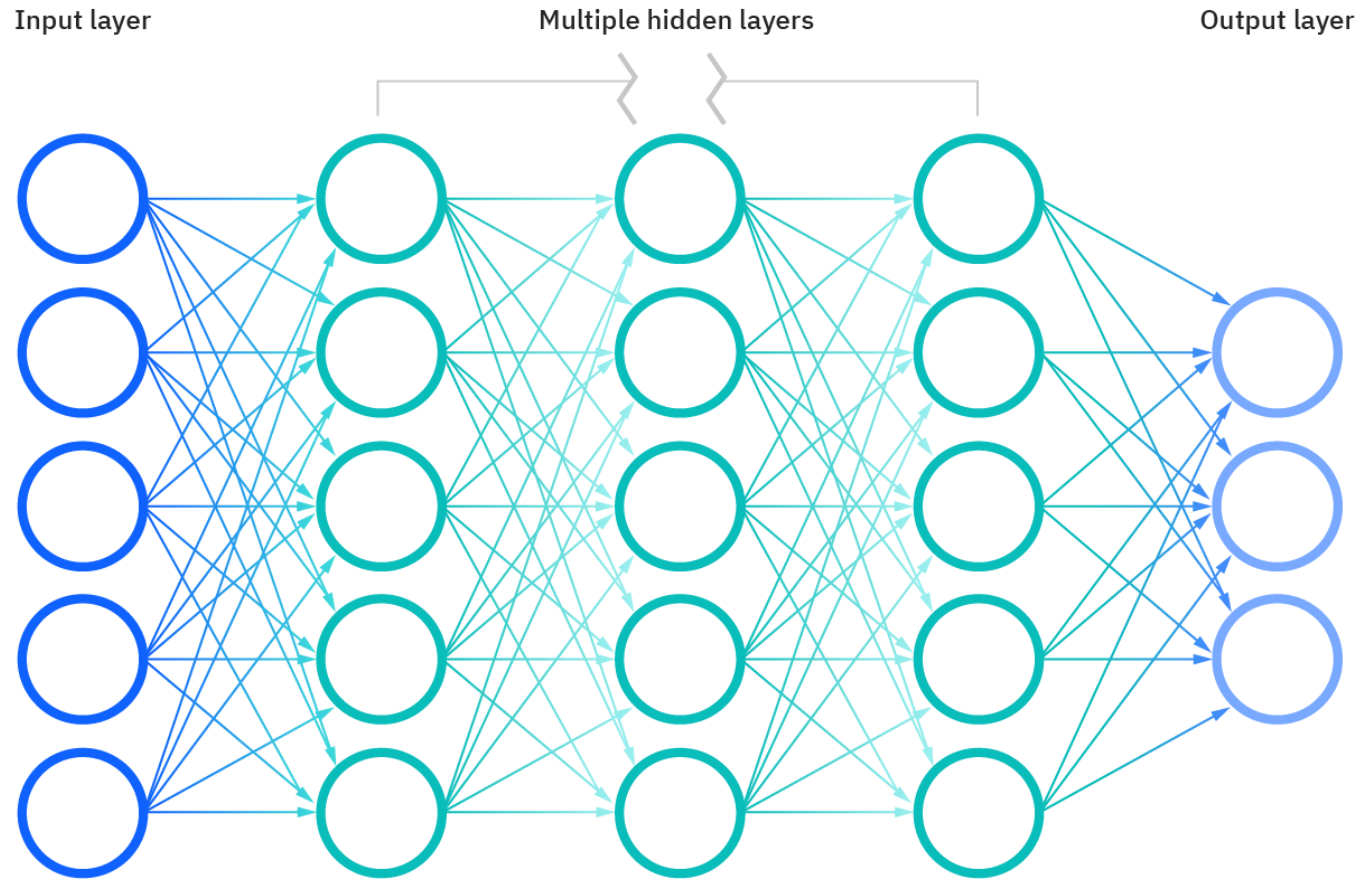
CPU

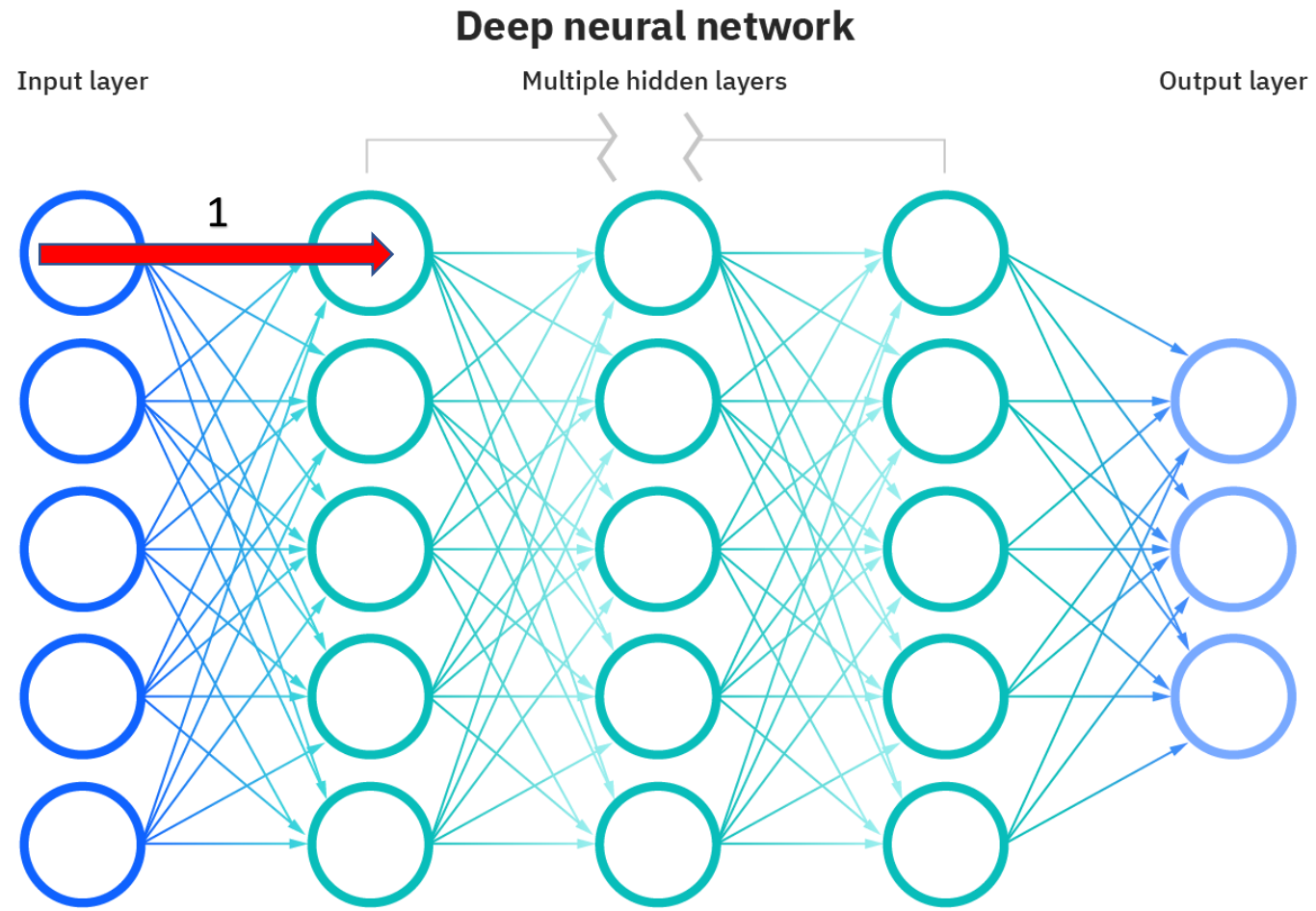
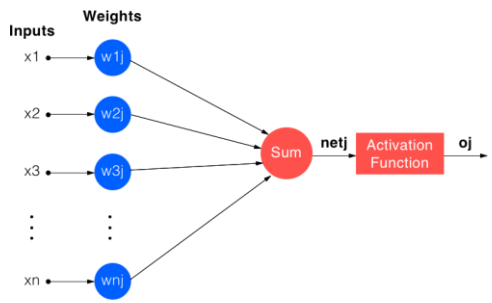


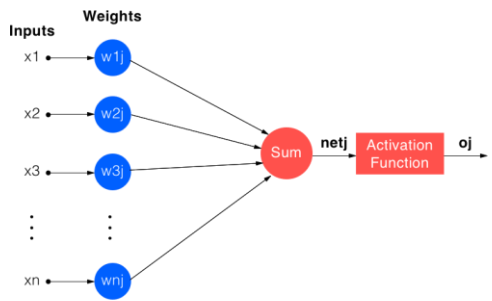
GPU



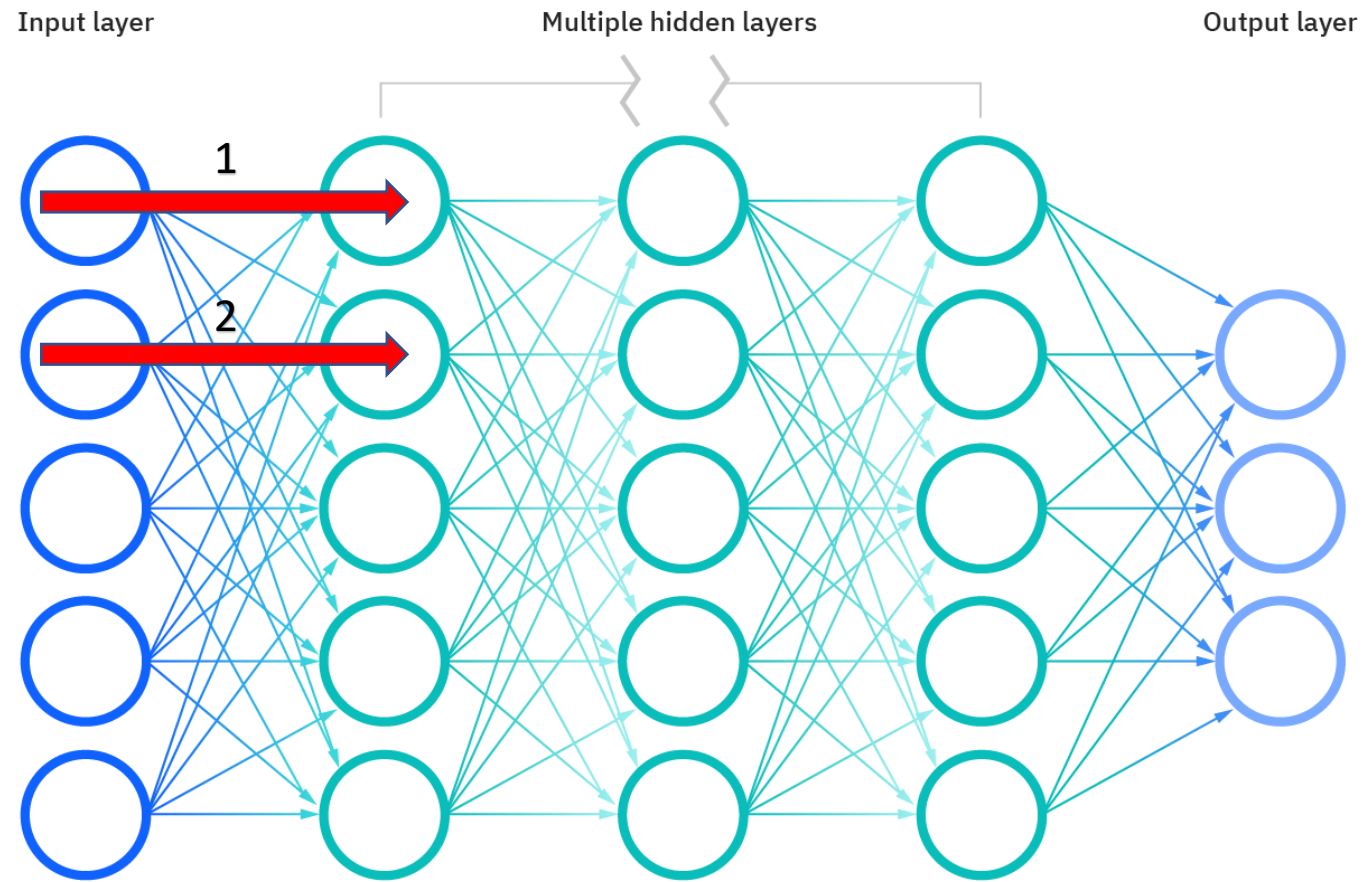
Deep neural network

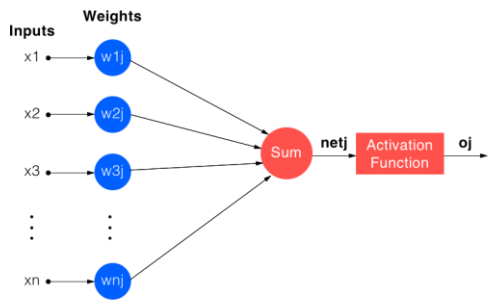




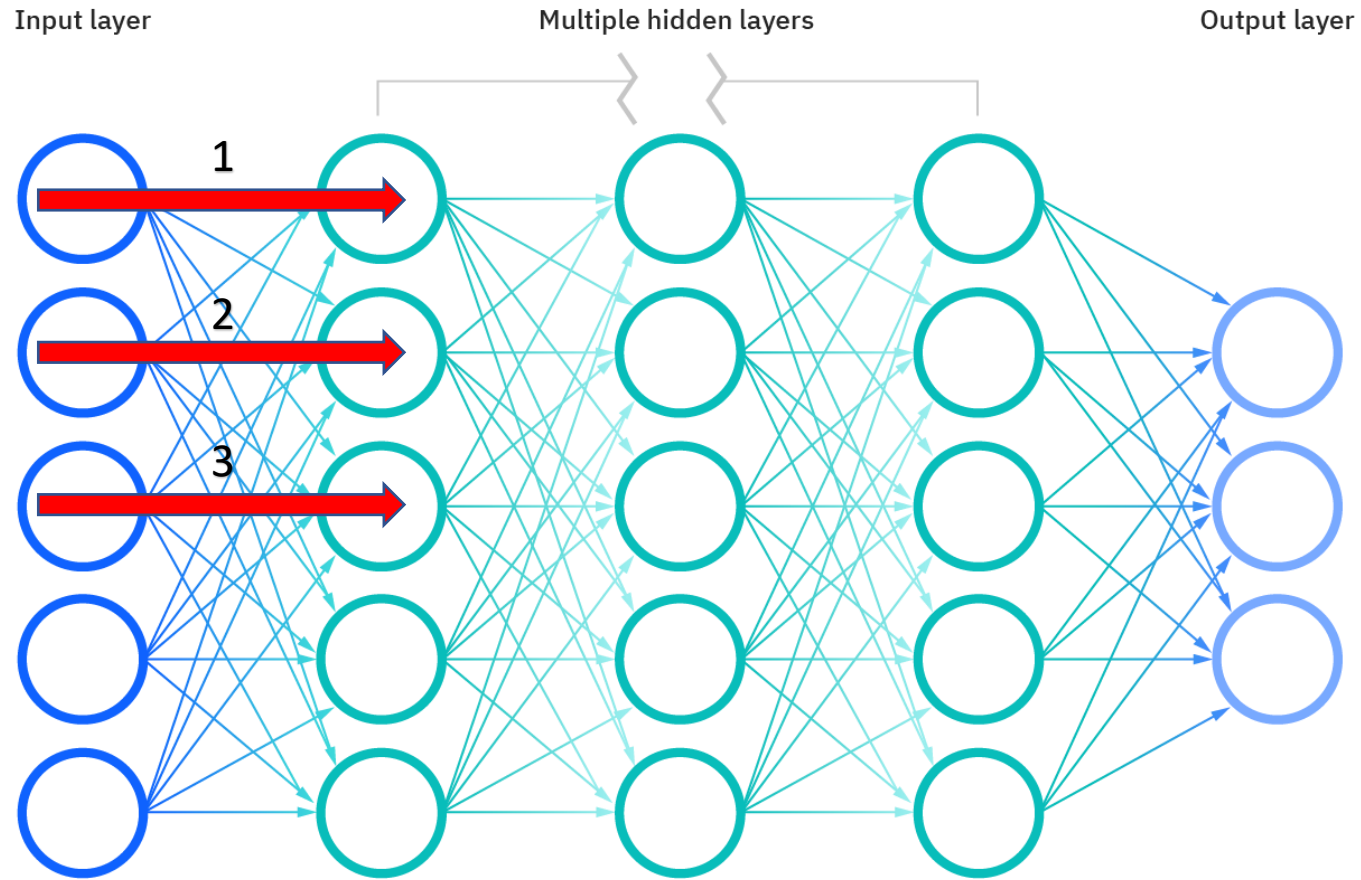


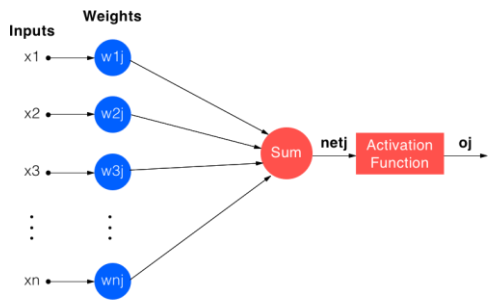
Deep neural network



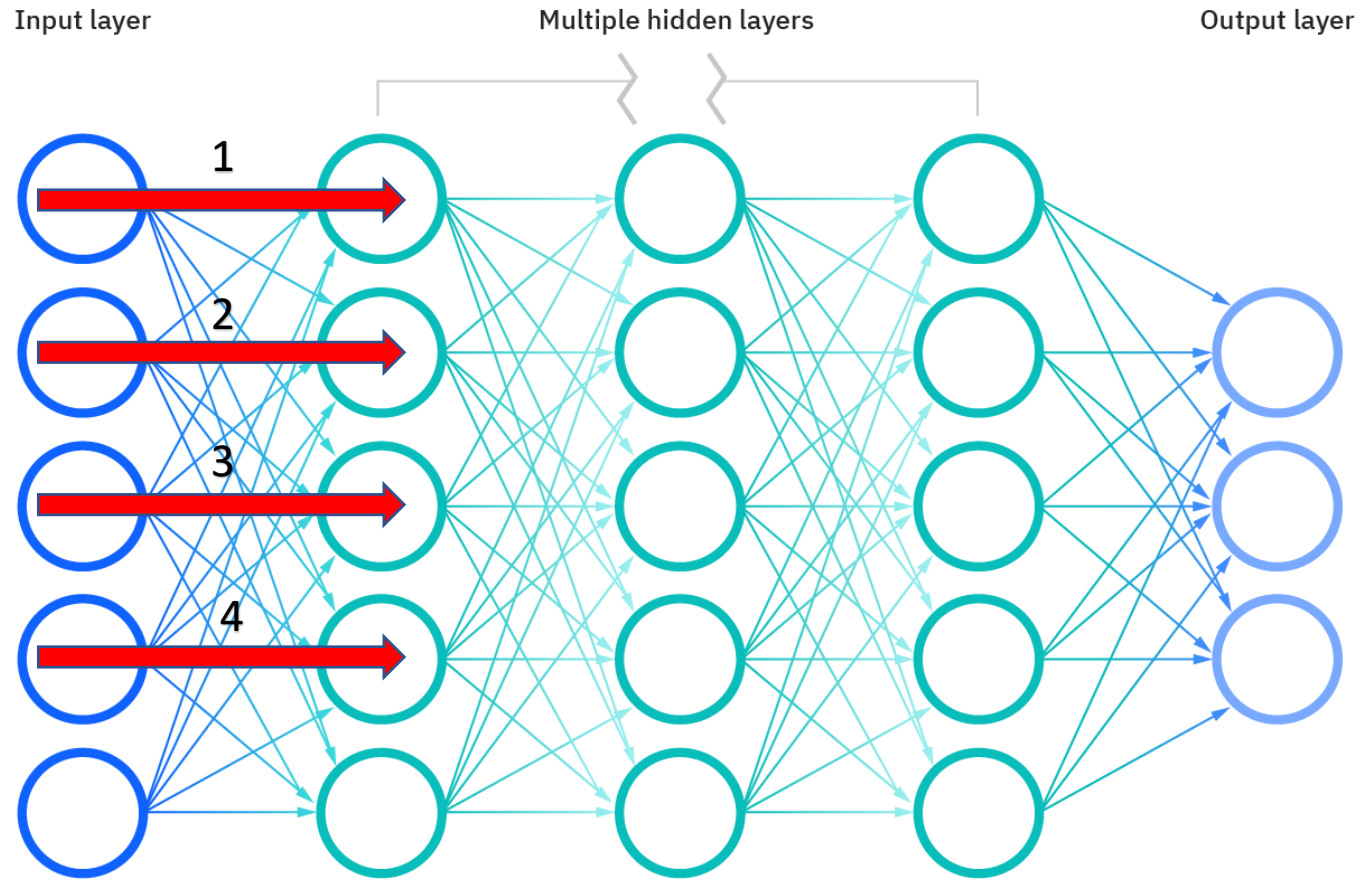


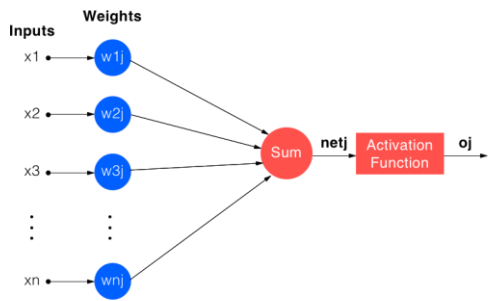
Deep neural network



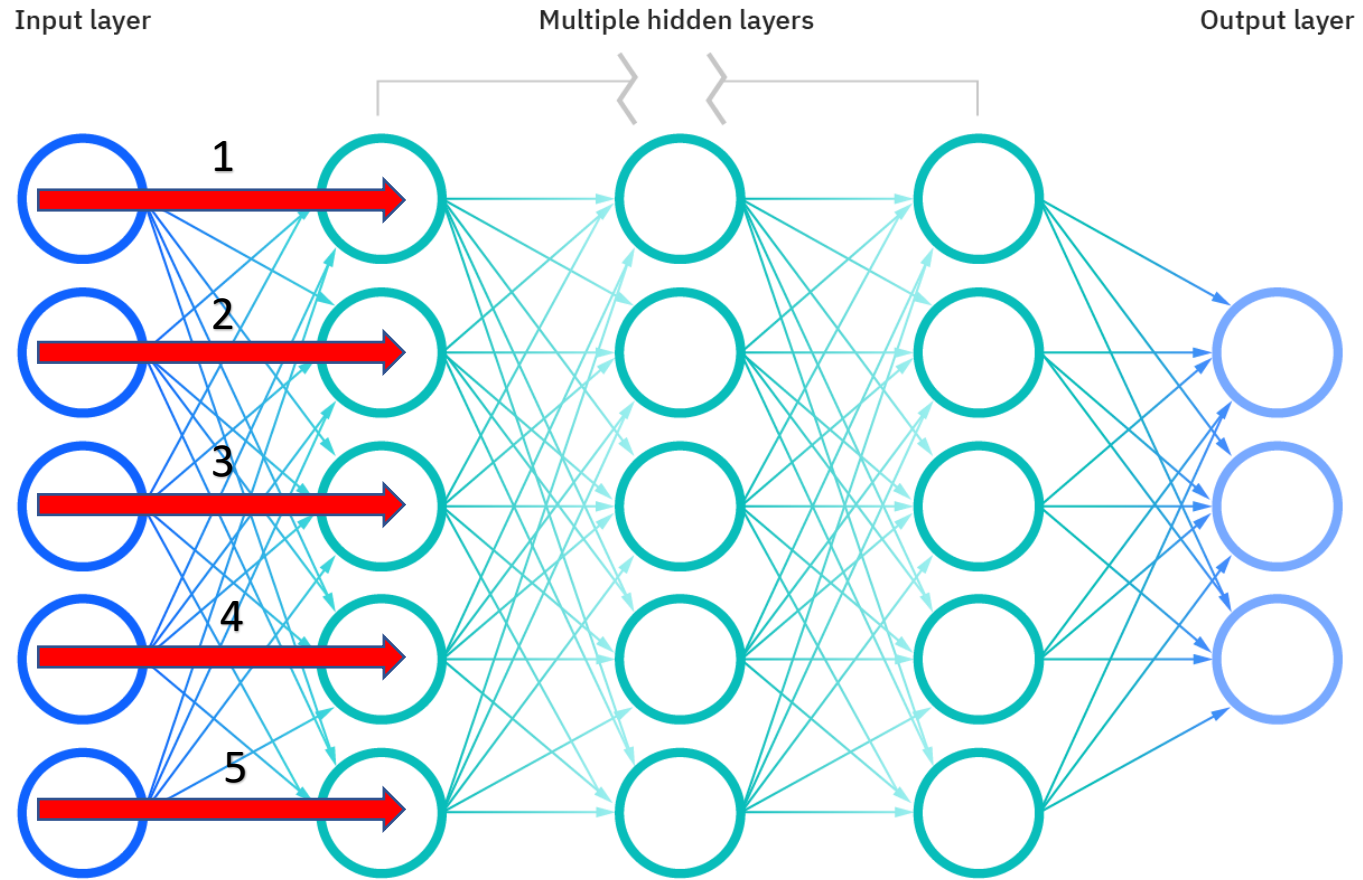


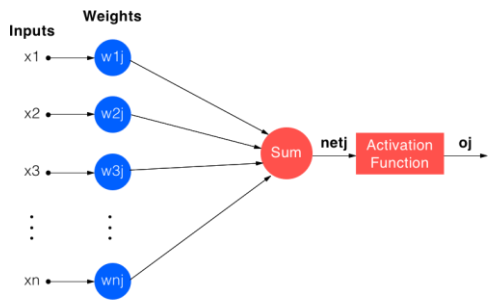
Deep neural network



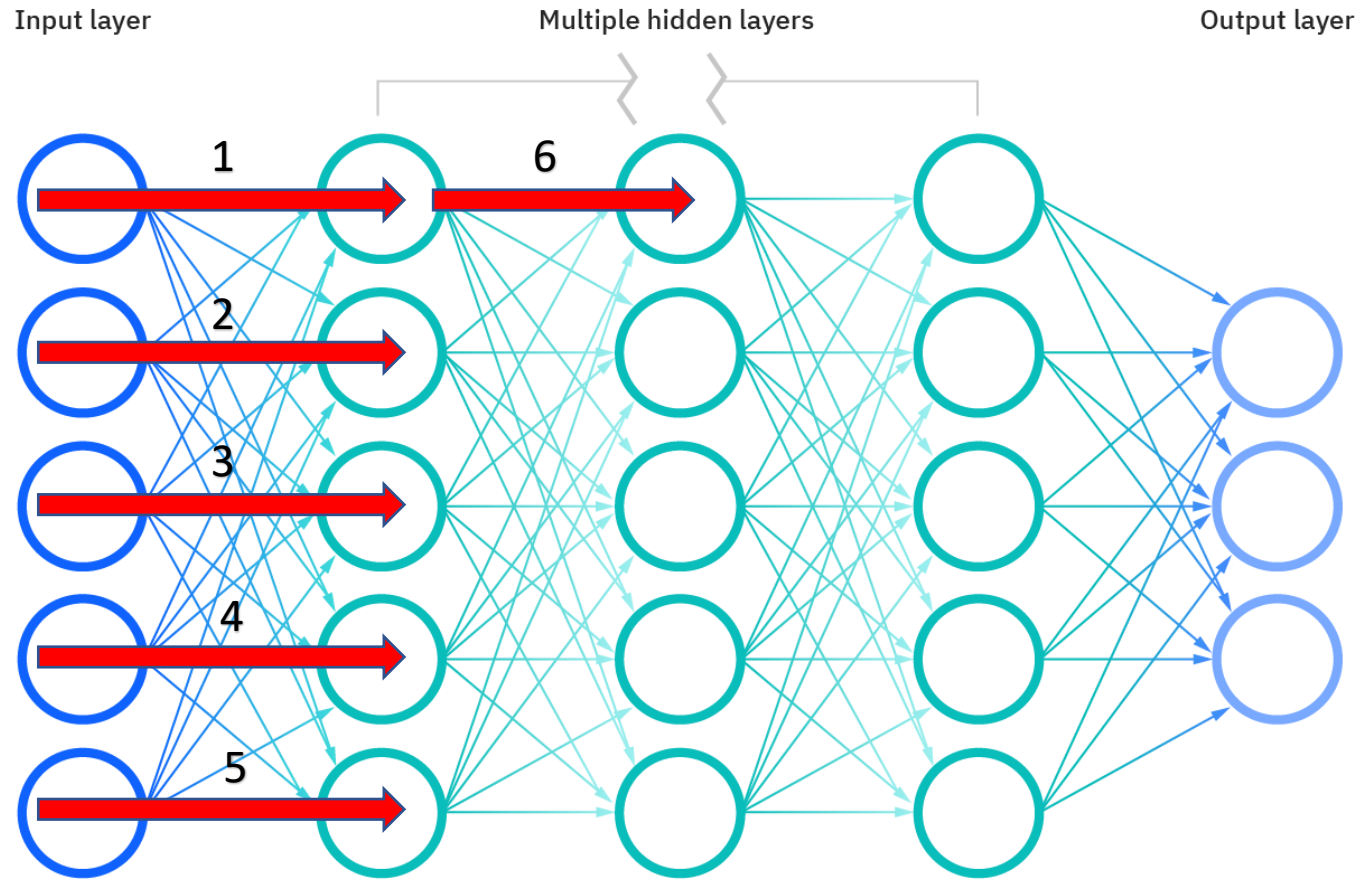


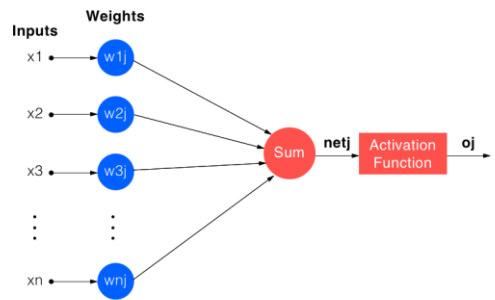
Deep neural network



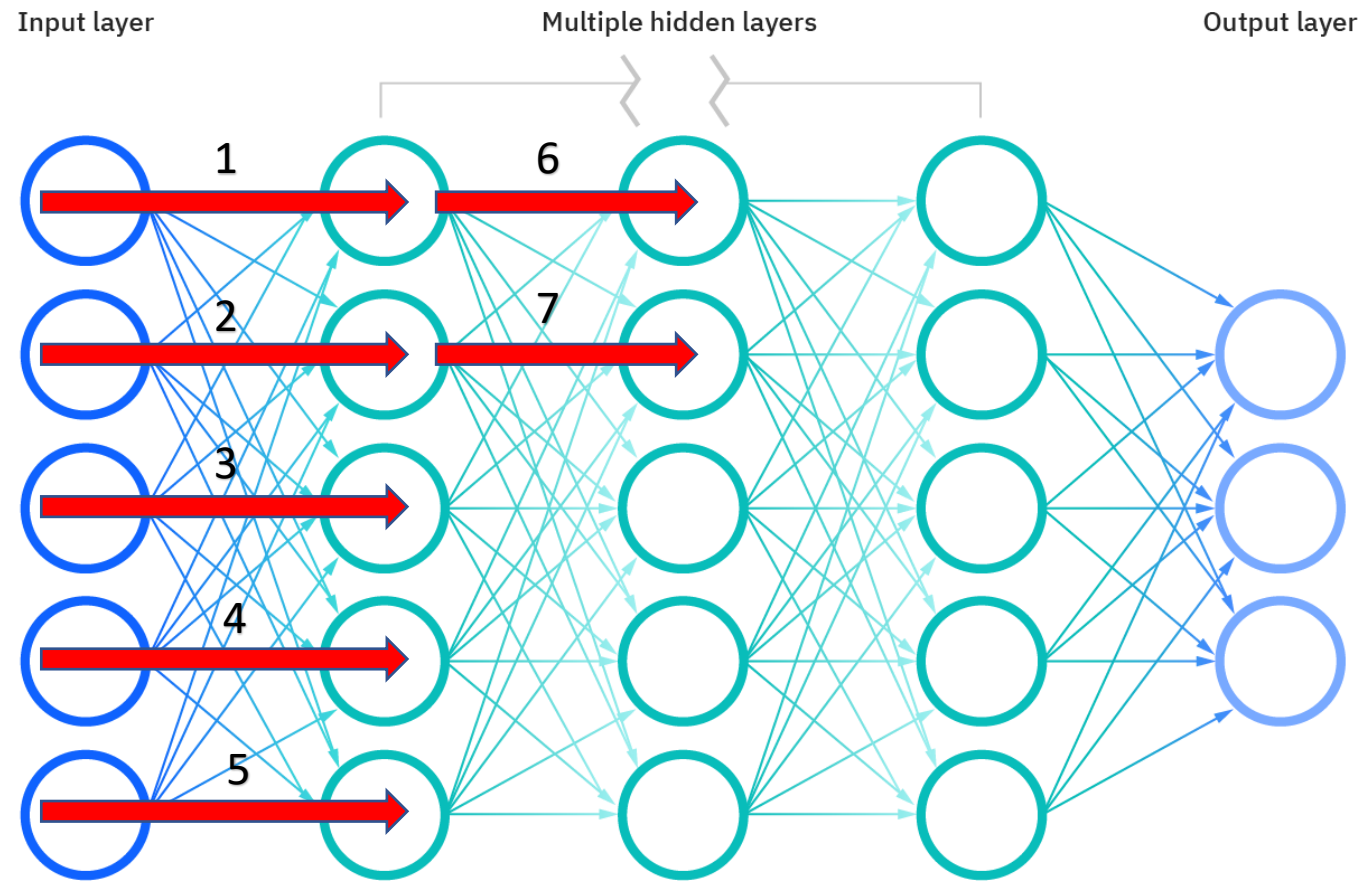


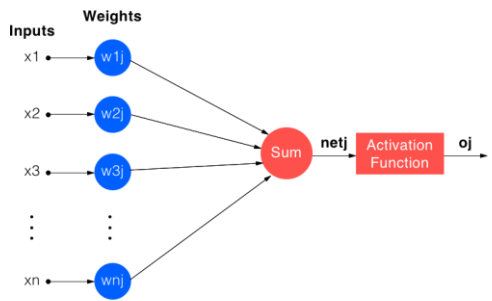
Deep neural network



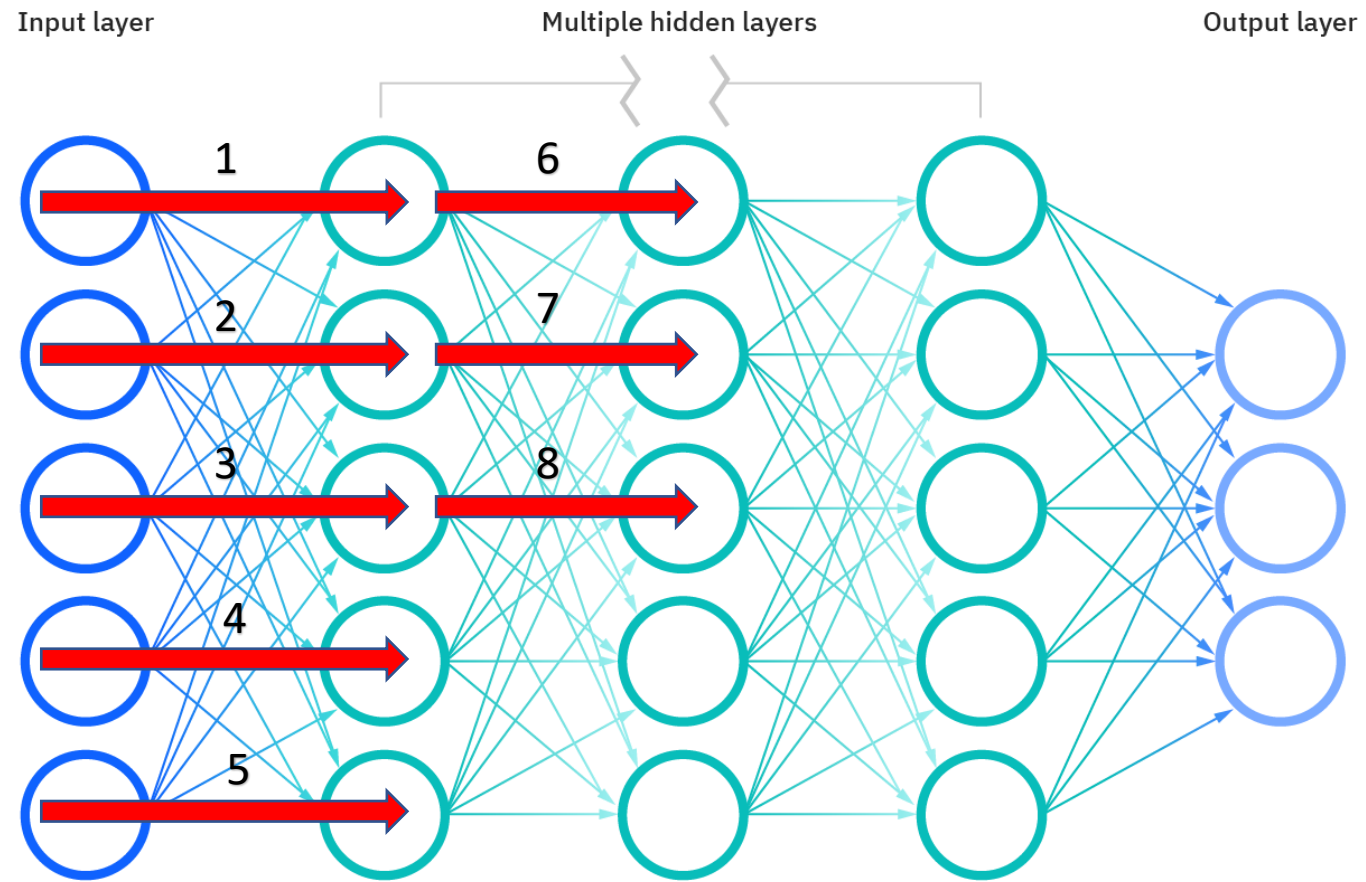


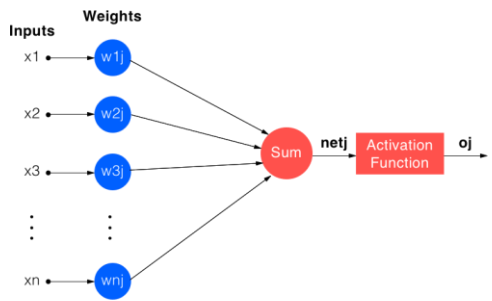
Deep neural network



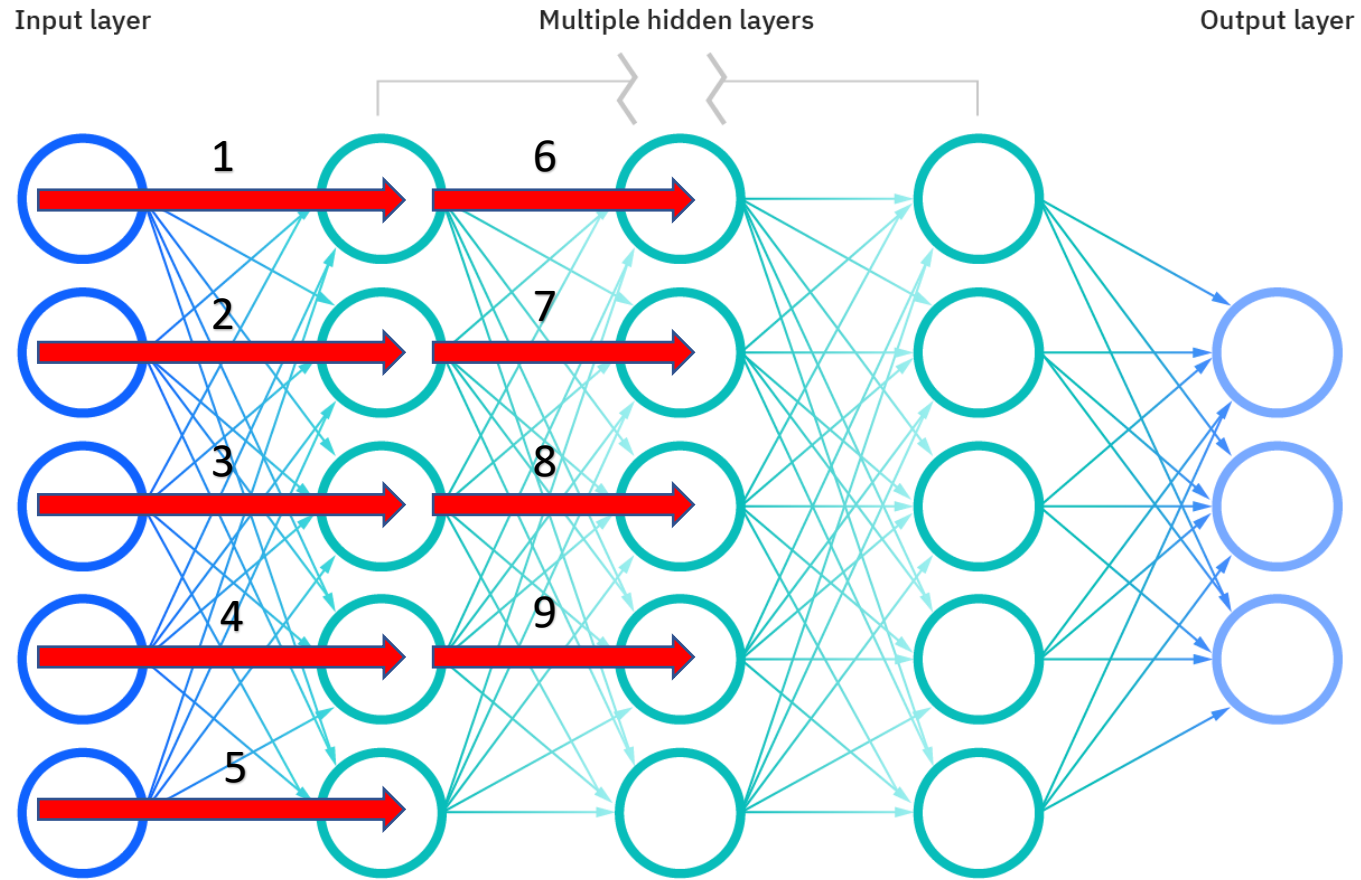


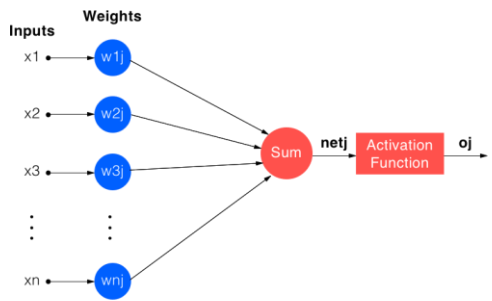
Deep neural network



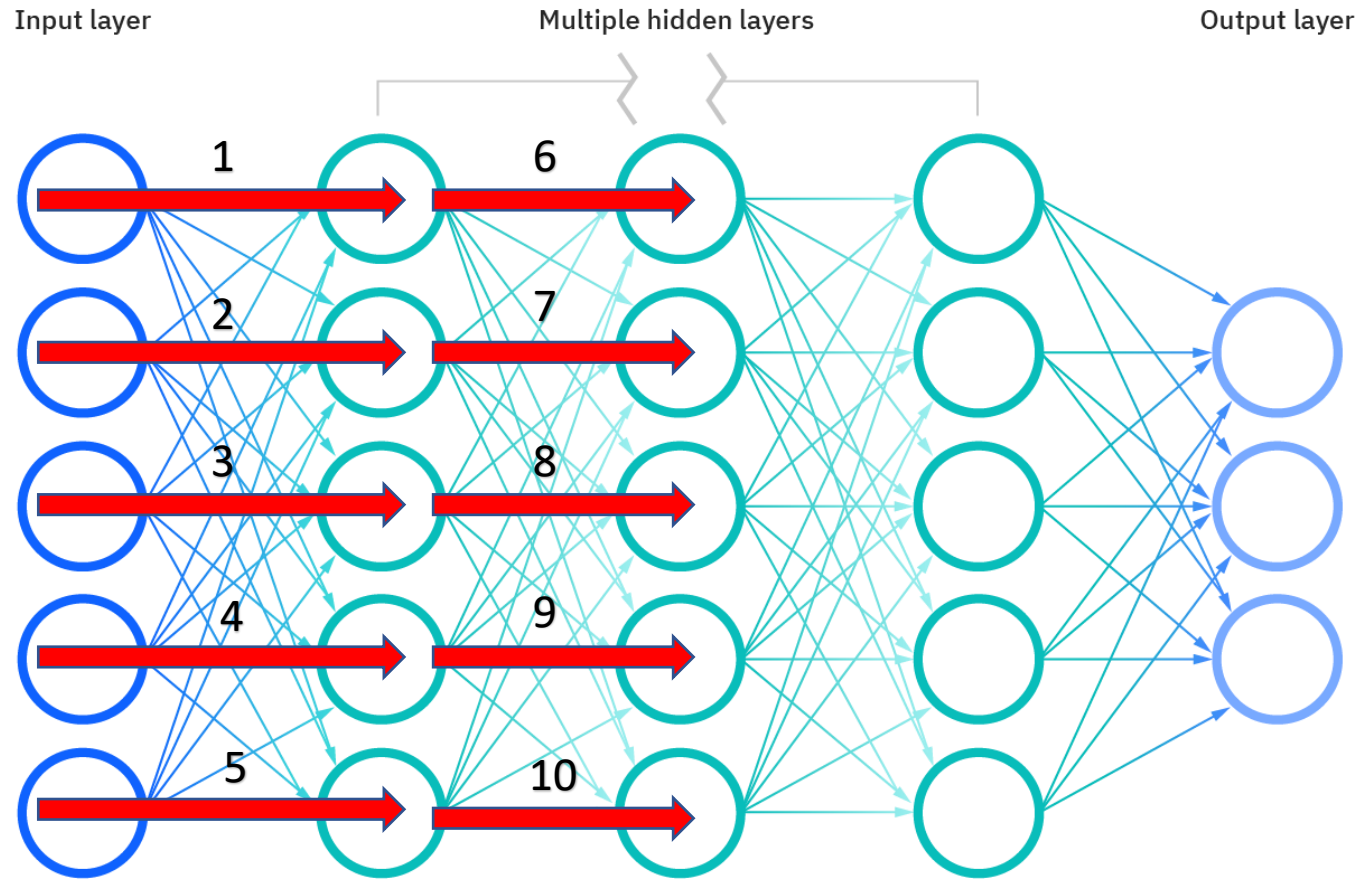


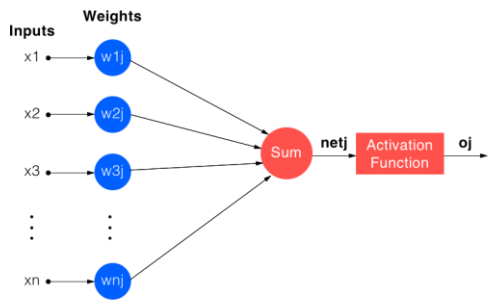
Deep neural network



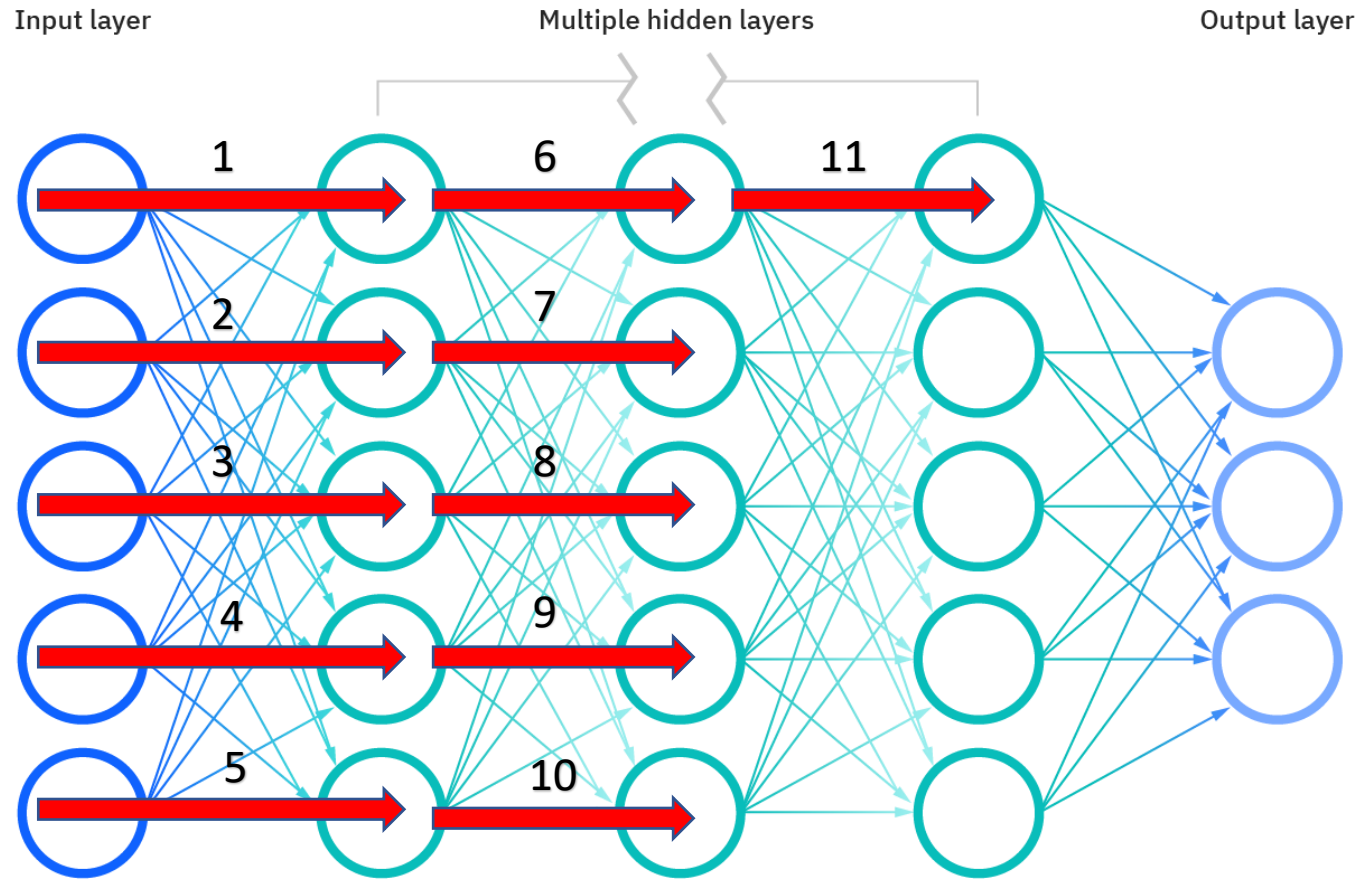


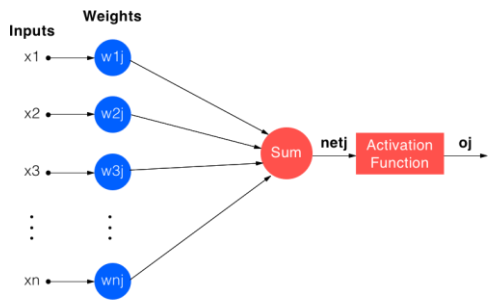
Deep neural network



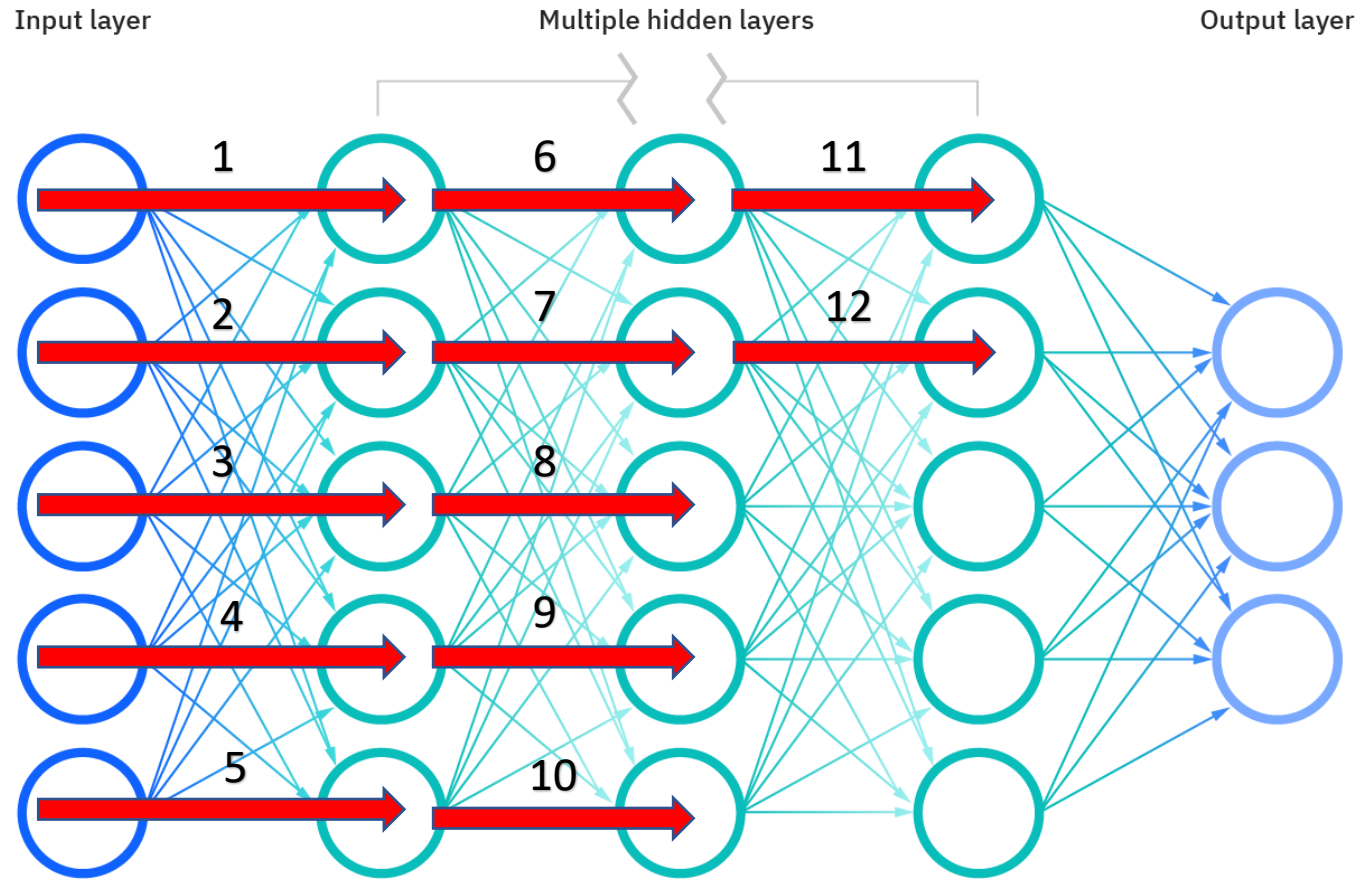


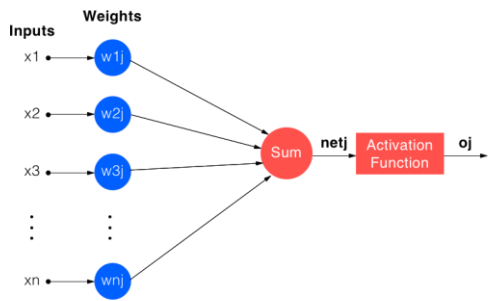
Deep neural network



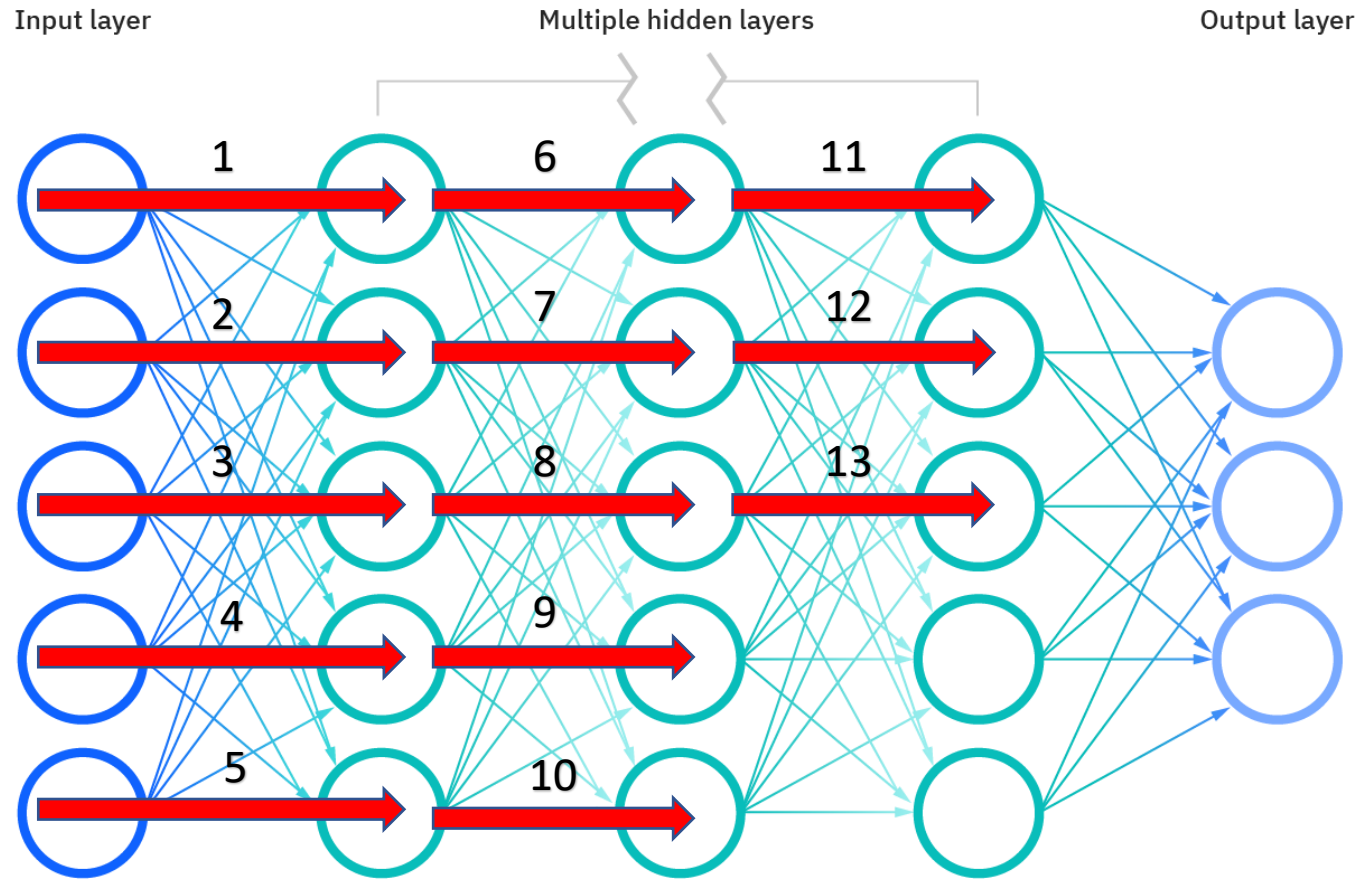


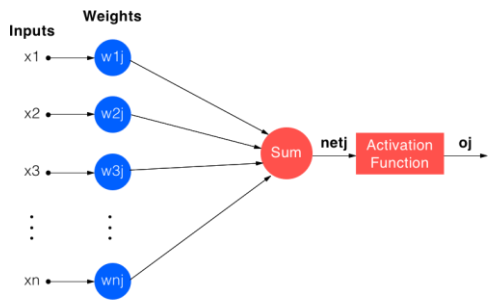
Deep neural network



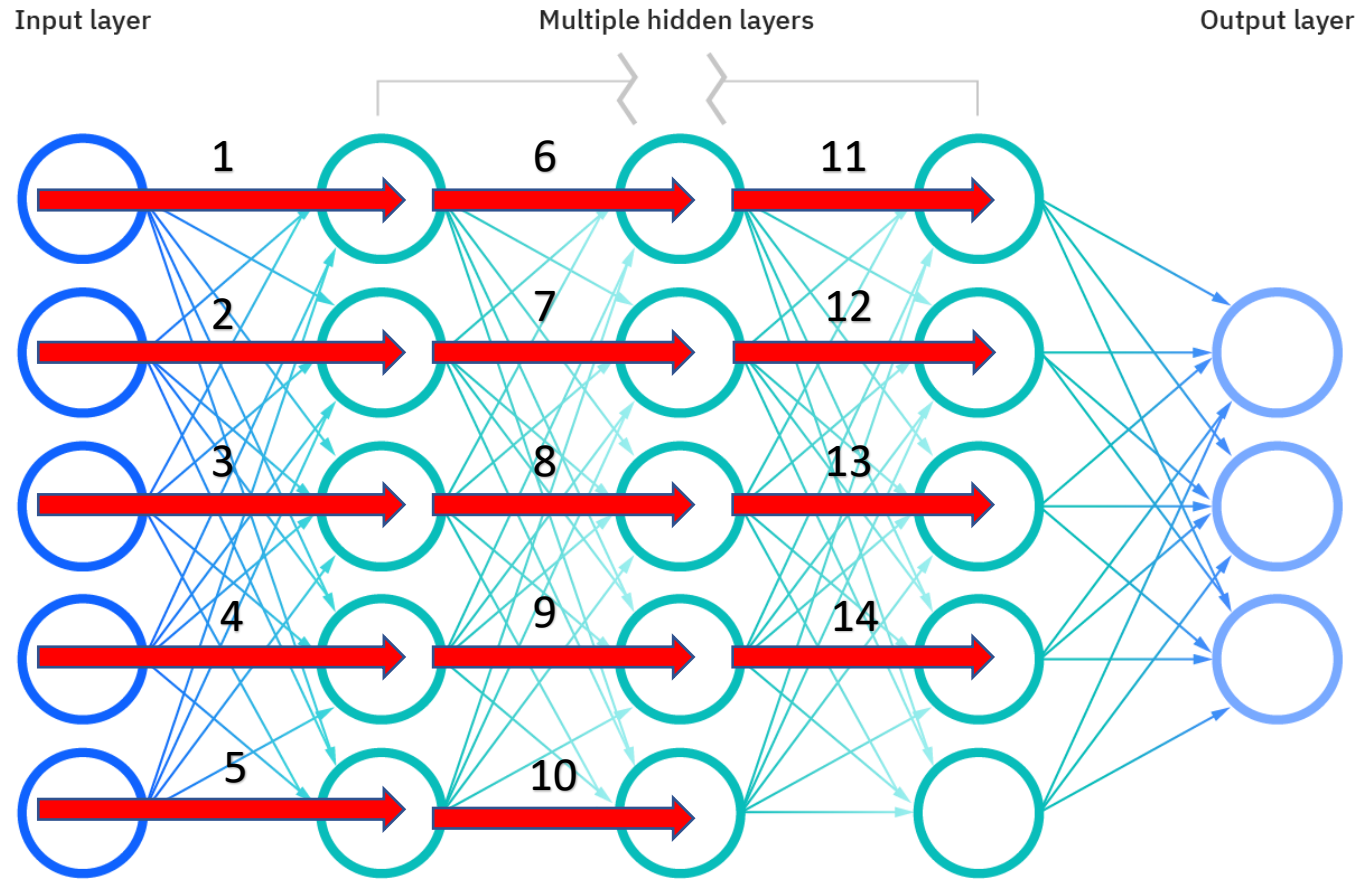


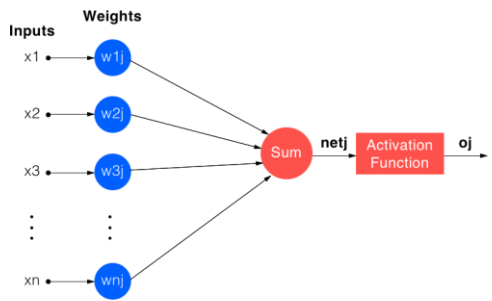
Deep neural network



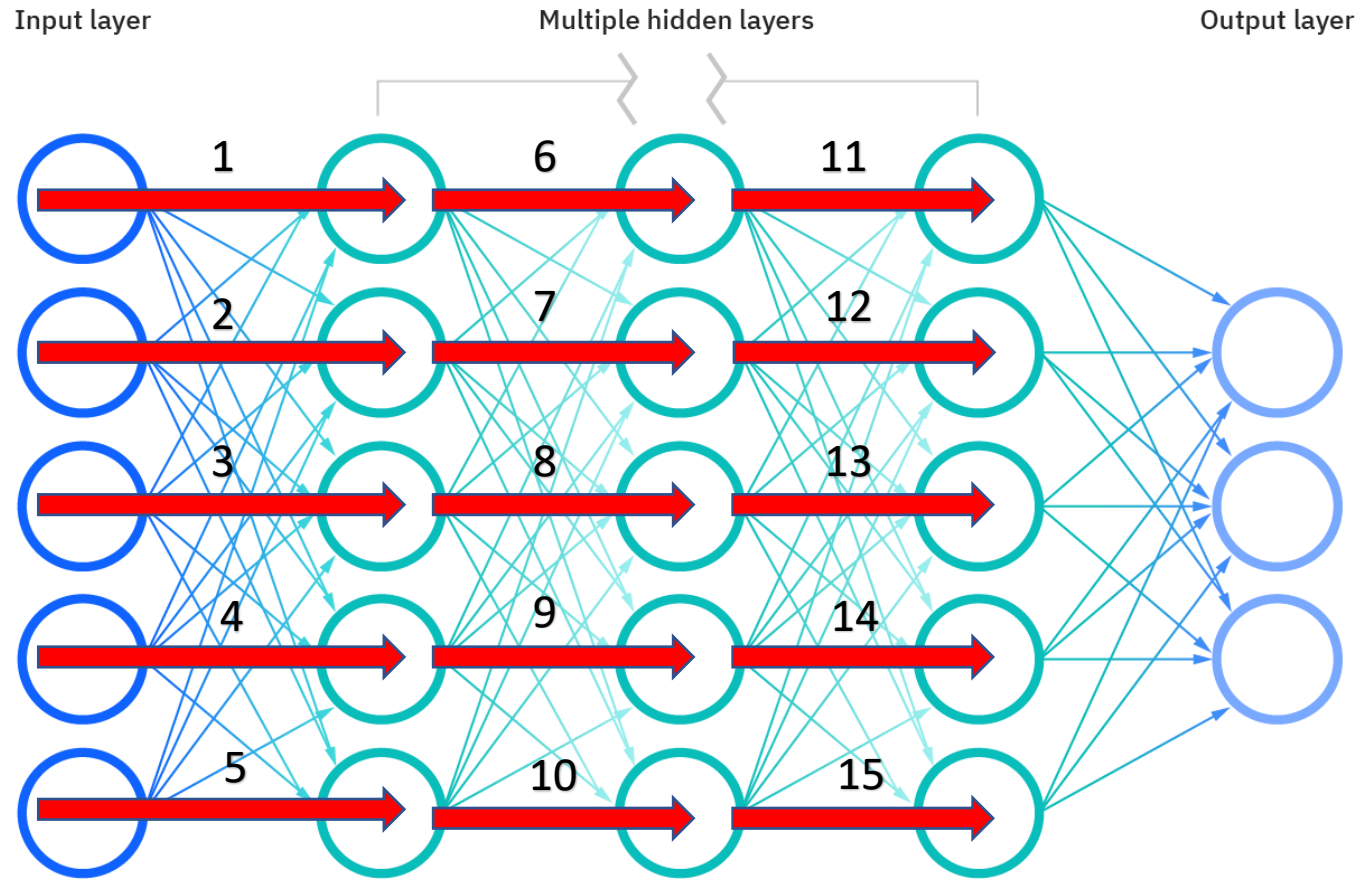


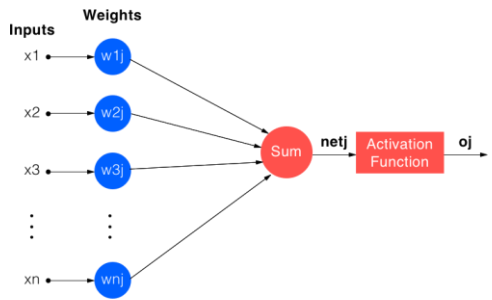
Deep neural network



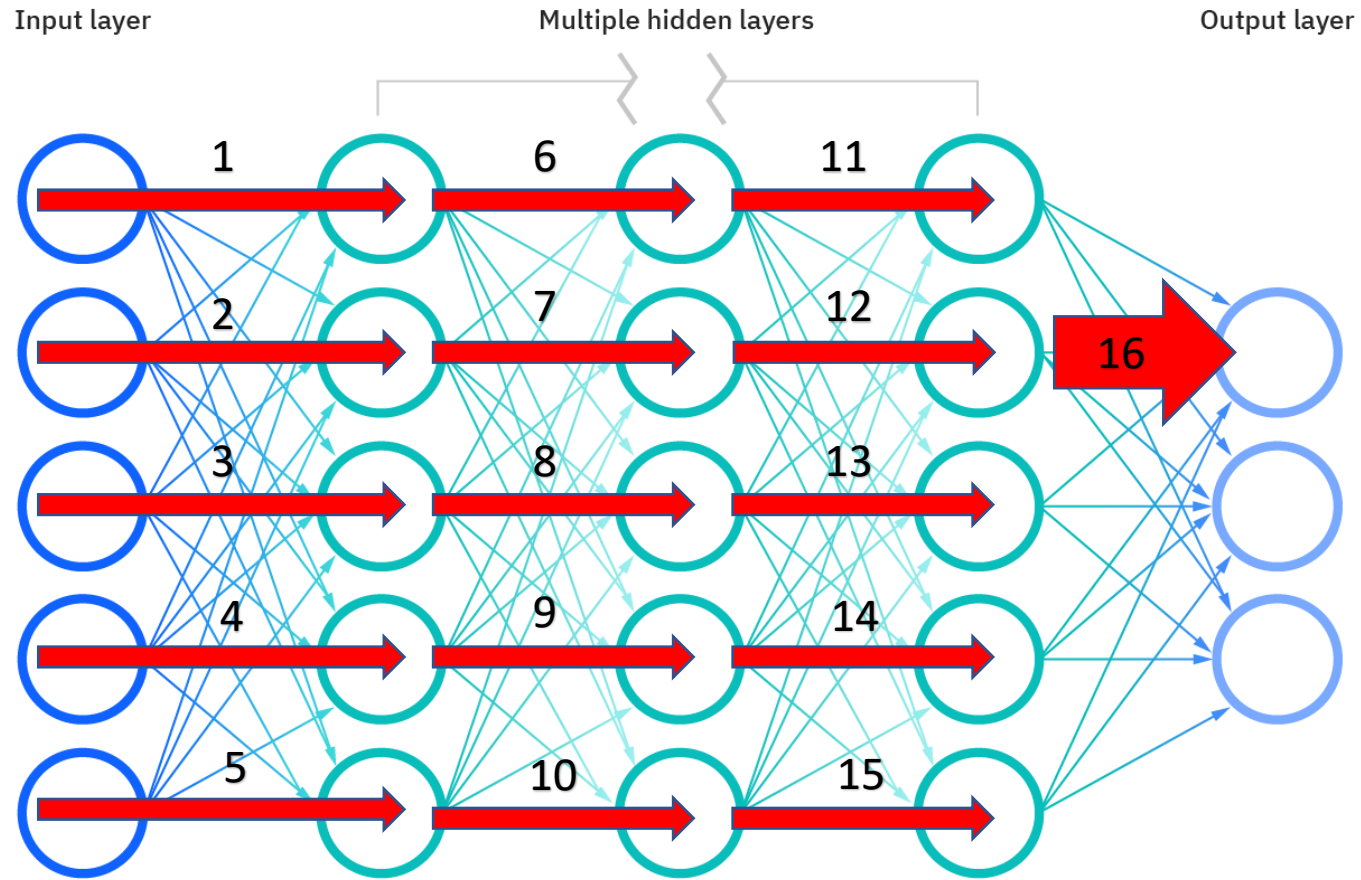


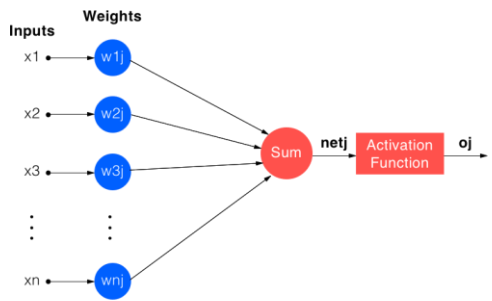
Deep neural network



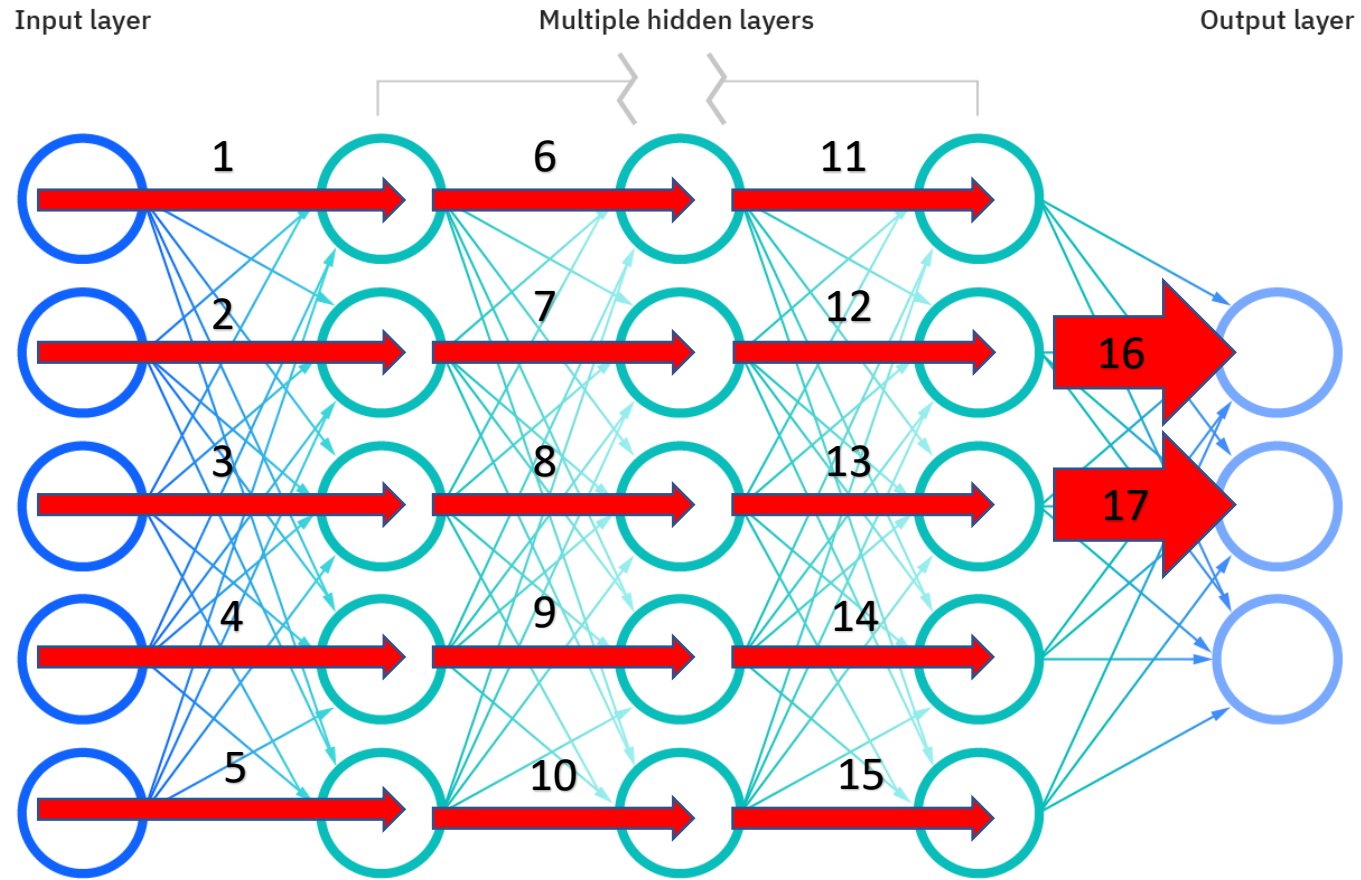


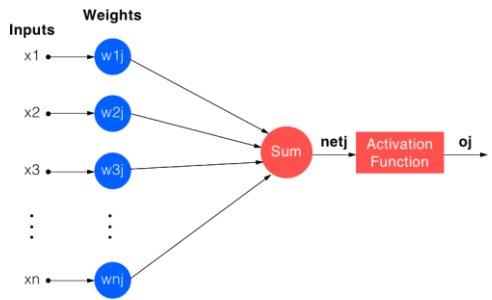
Deep neural network



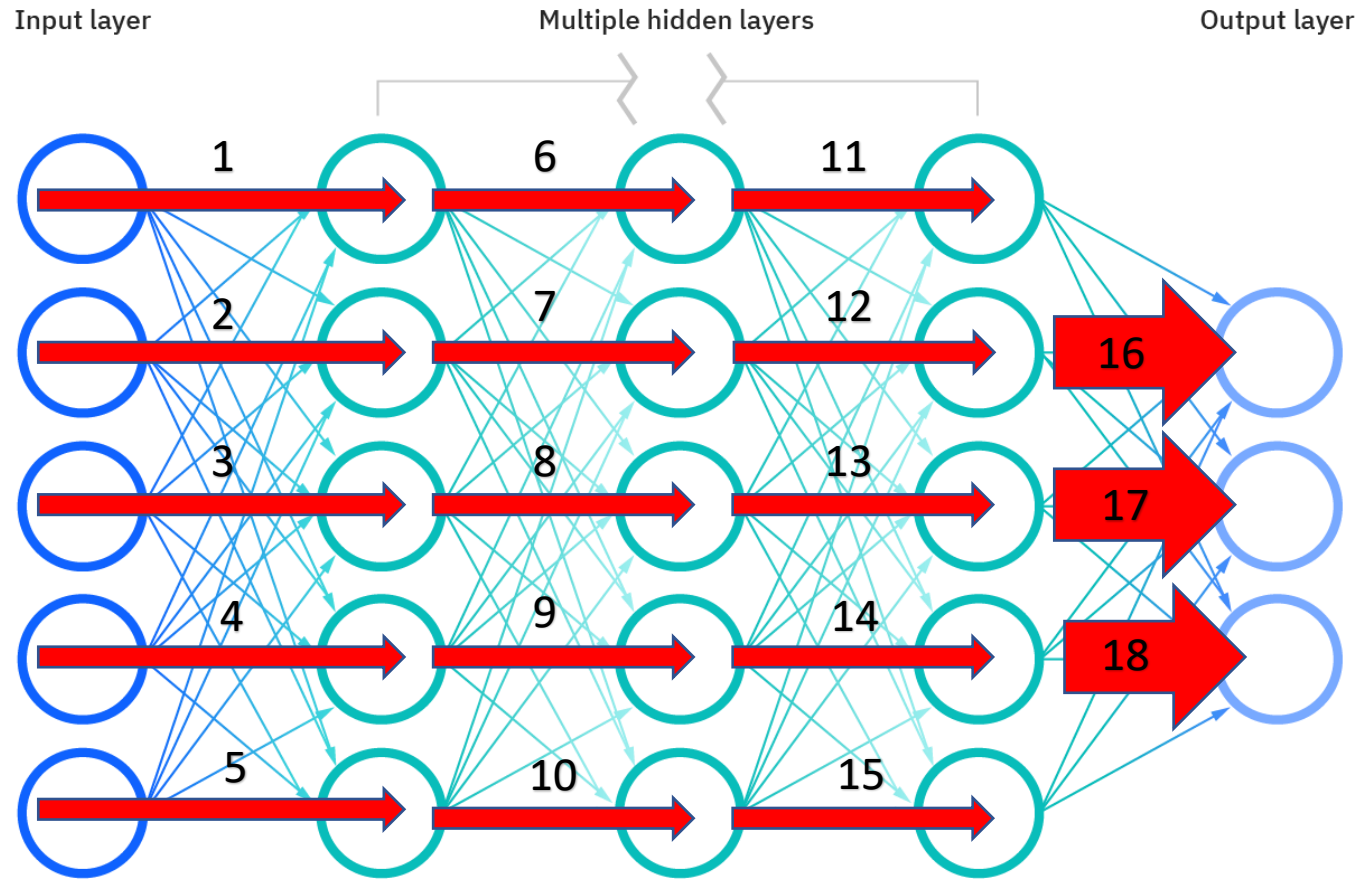


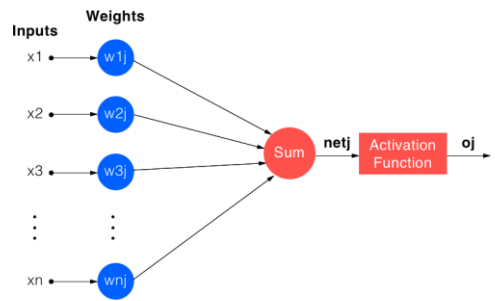
Deep neural network



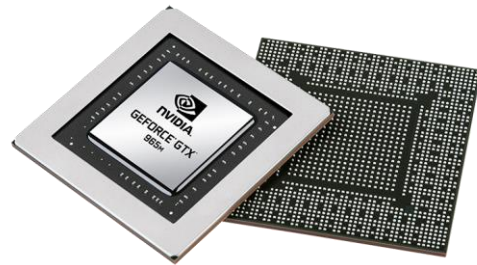
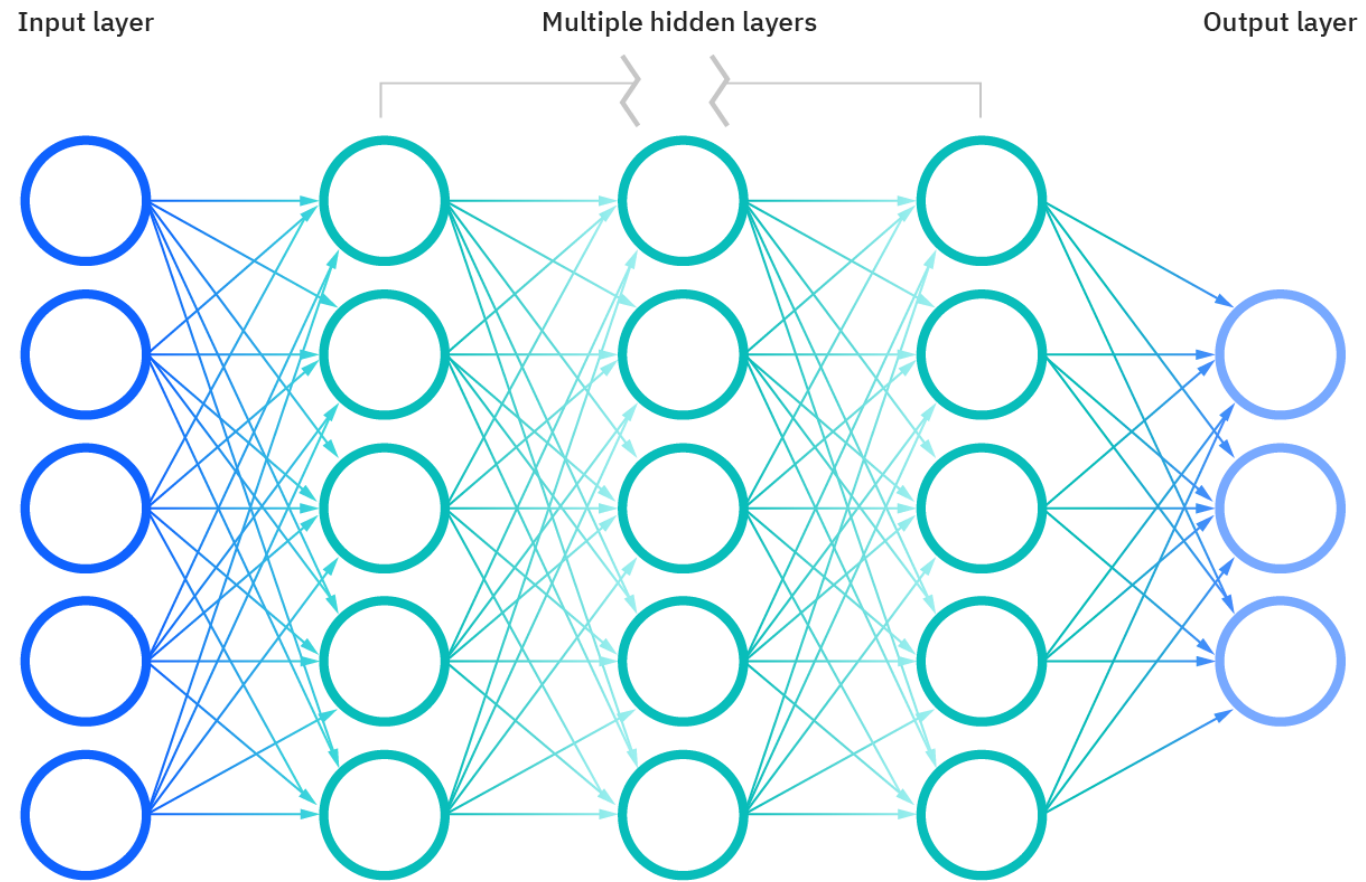


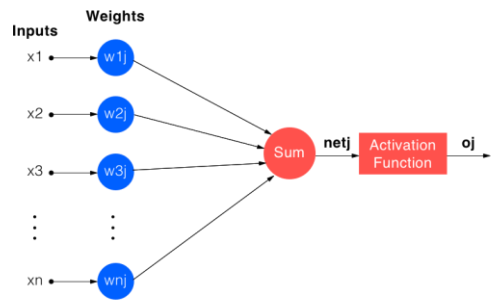
Deep neural network



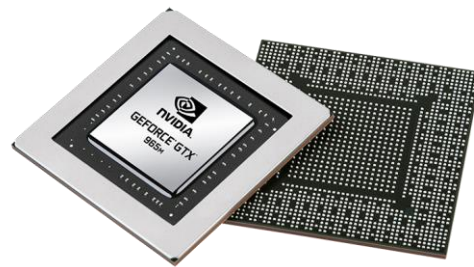
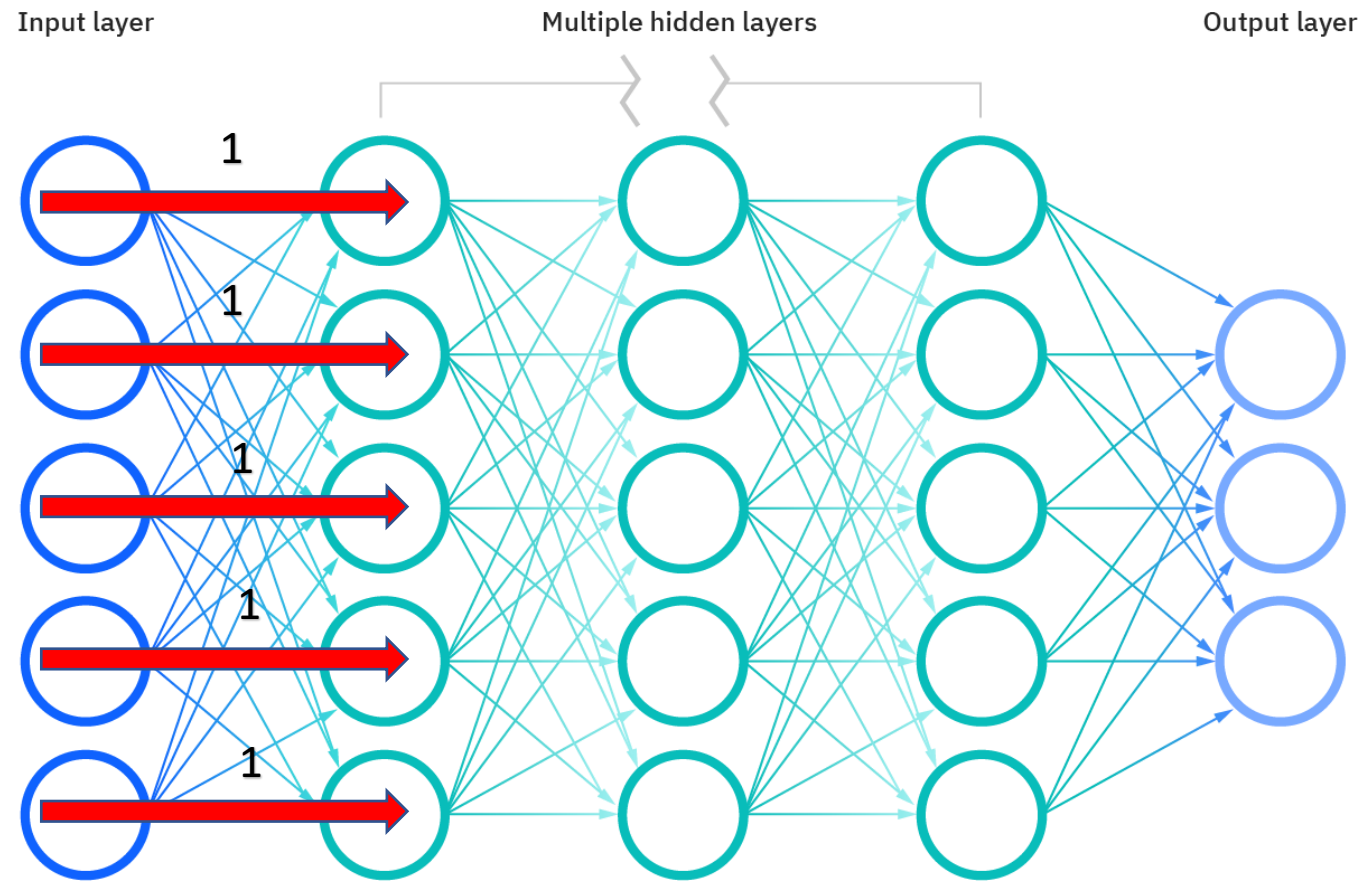


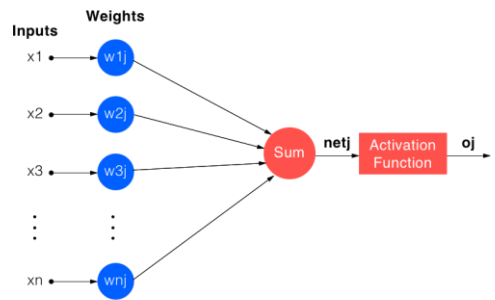
Deep neural network



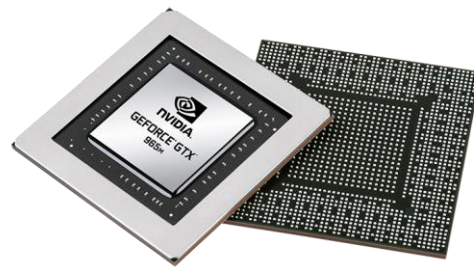
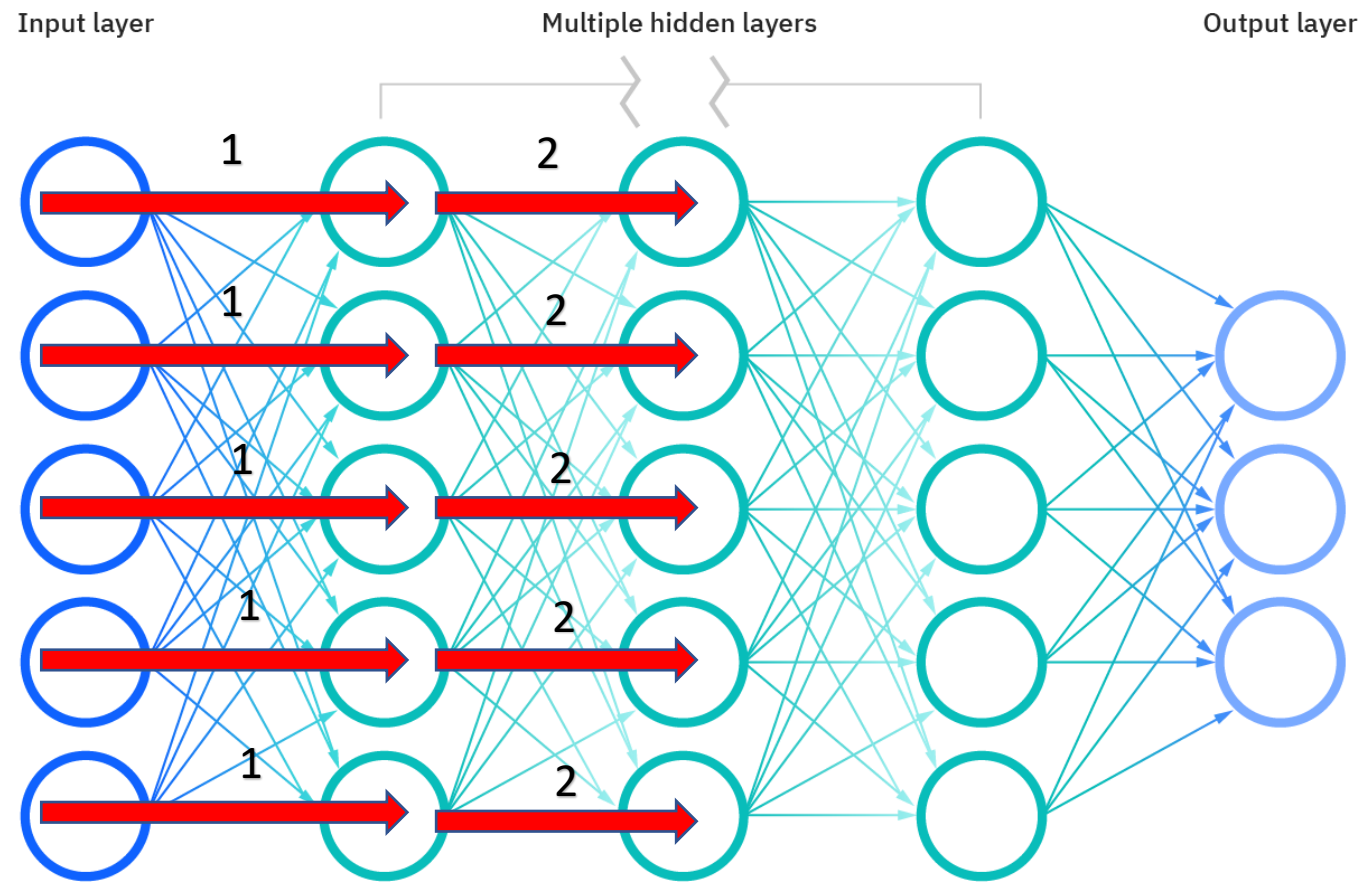


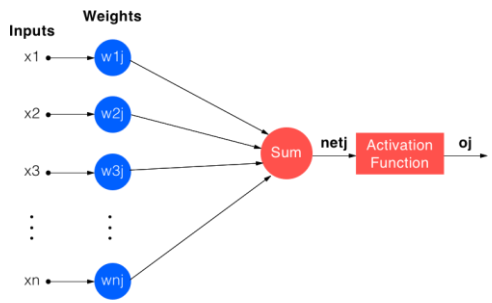
Deep neural network



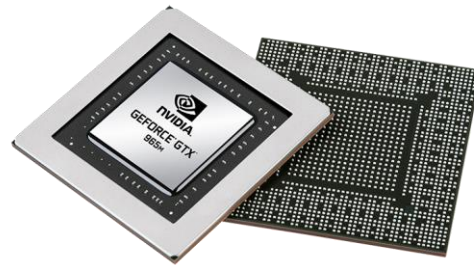
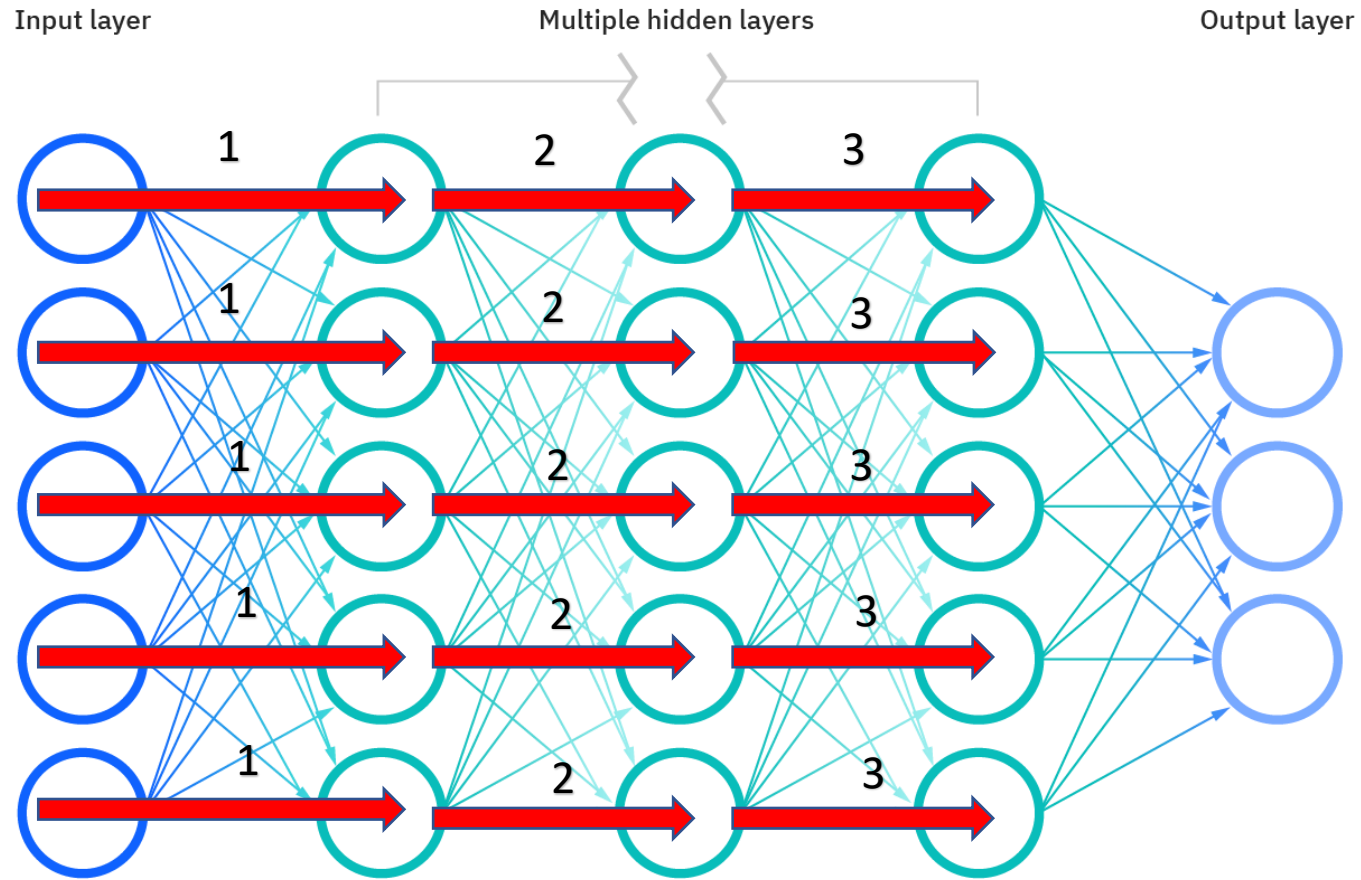


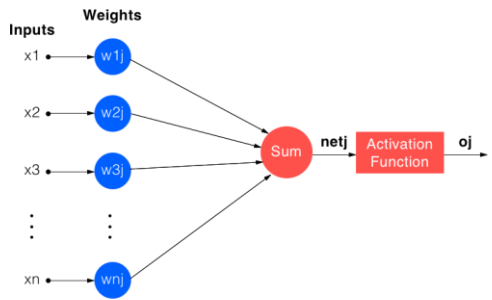
Deep neural network



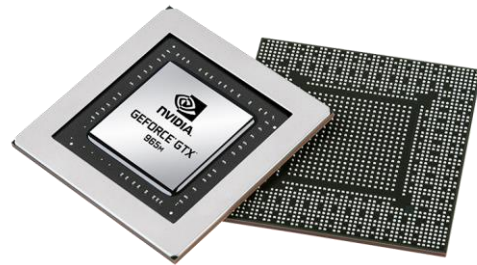
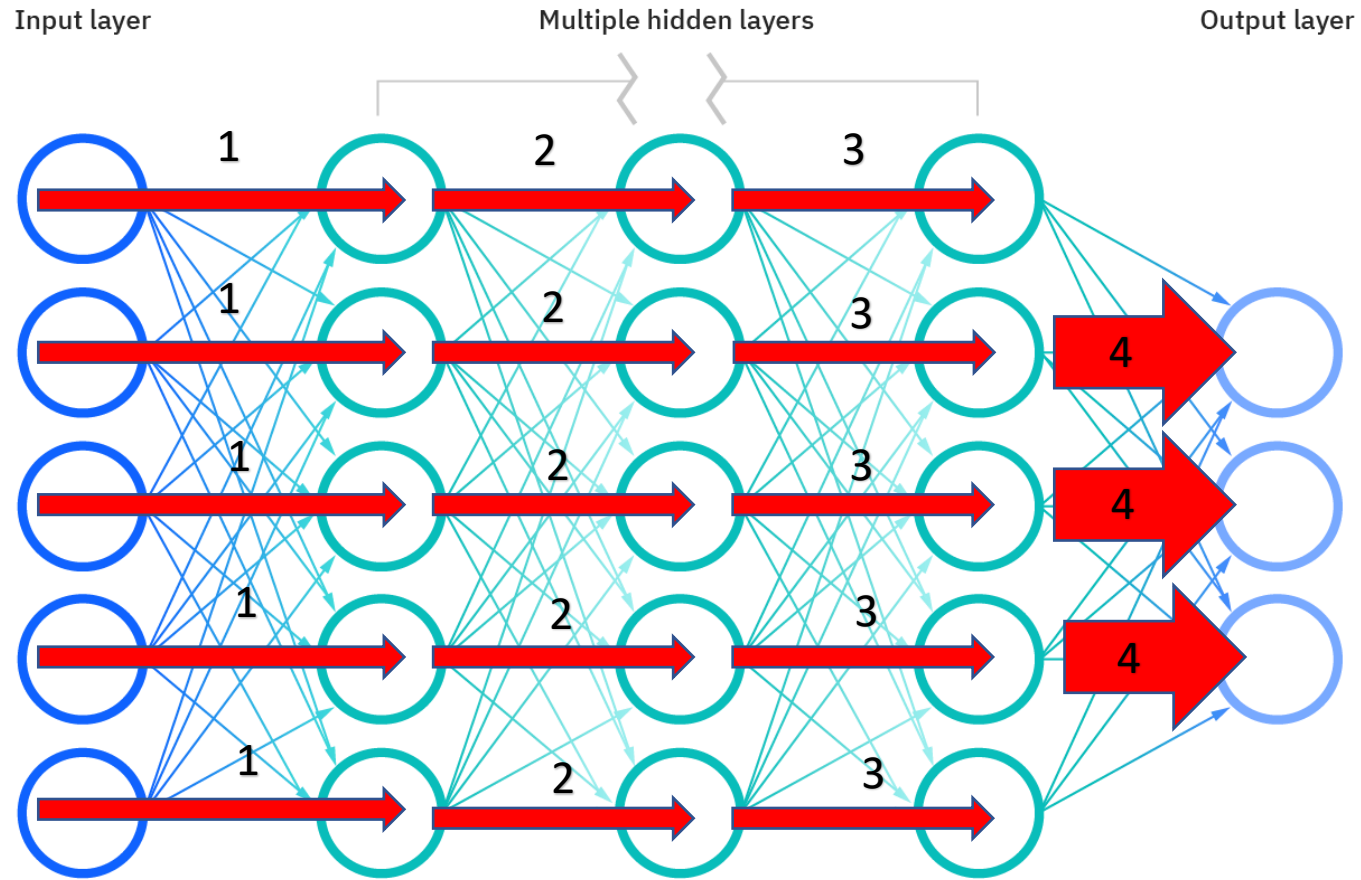


Deep neural network



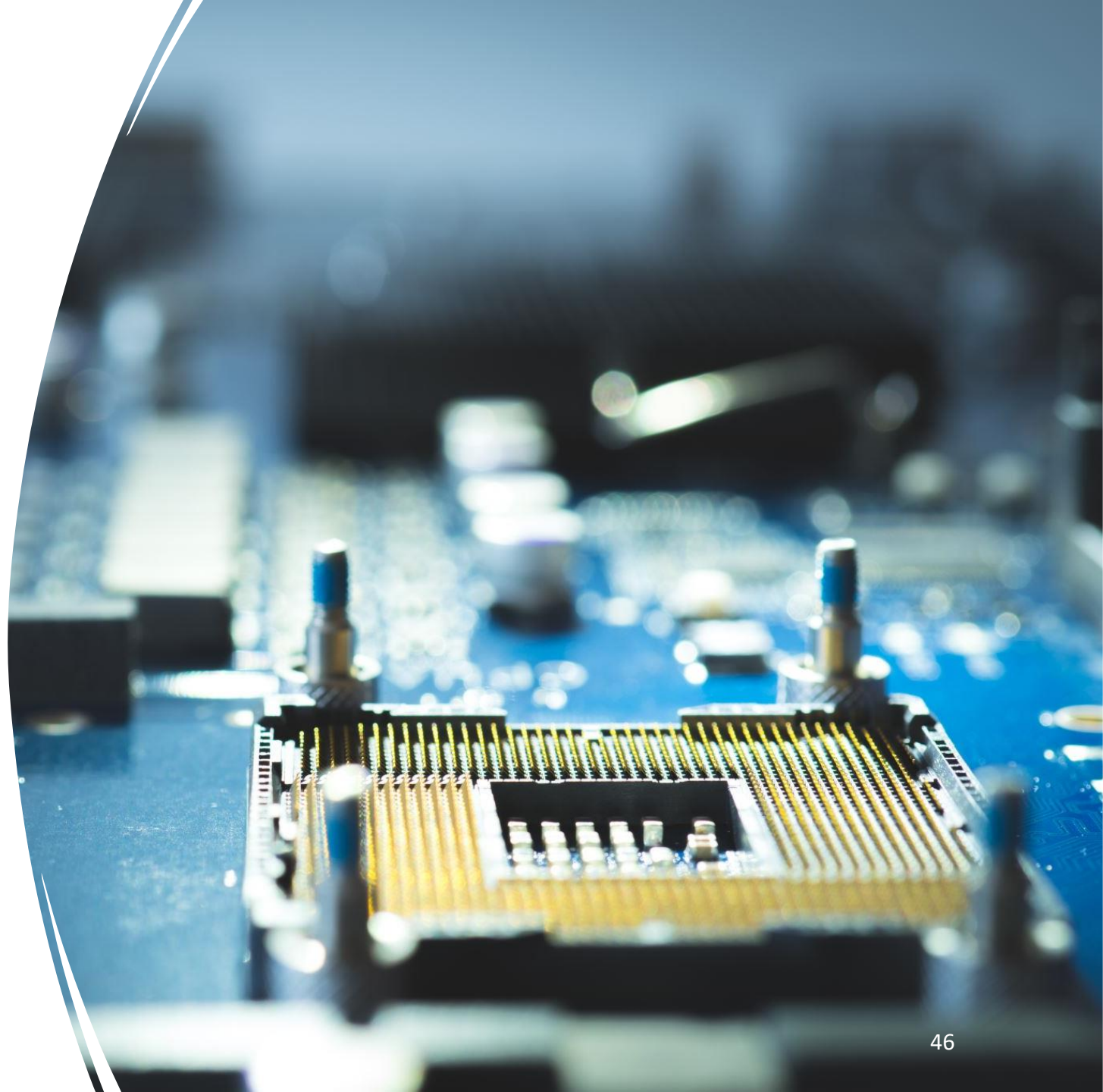


Deep neural network



Road Map

- Computing
- Processors
- How do electrons work for us?!
- CPU
- GPU
- **FPGA**
- Accelerator
- Tradeoff of processors



Field Programmable Gate Array (FPGA)

Initially, used for Prototyping by Electronic people

High-Level Language

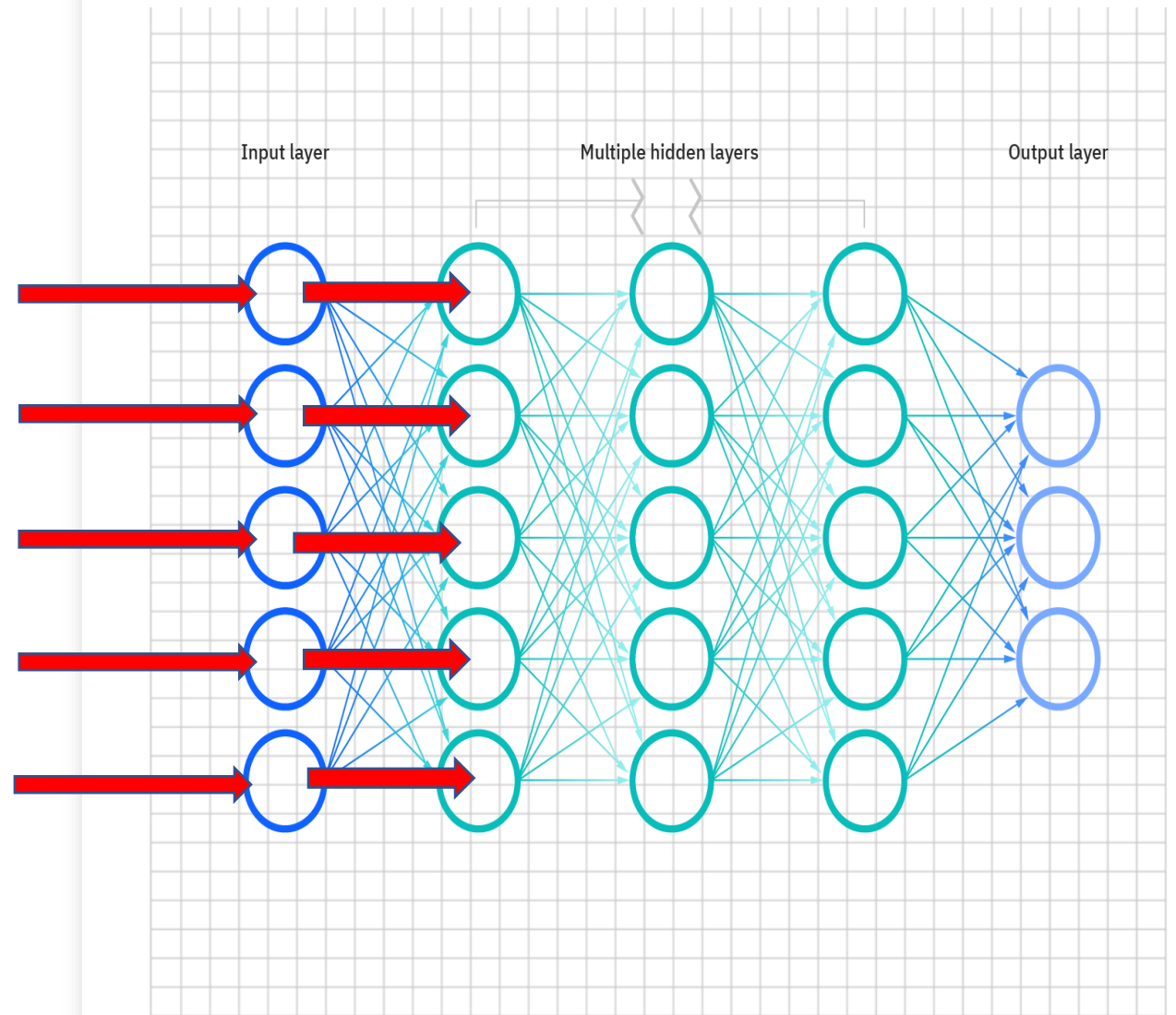


Very Low-Level Language
(requires Digital Electronics basics knowledge)



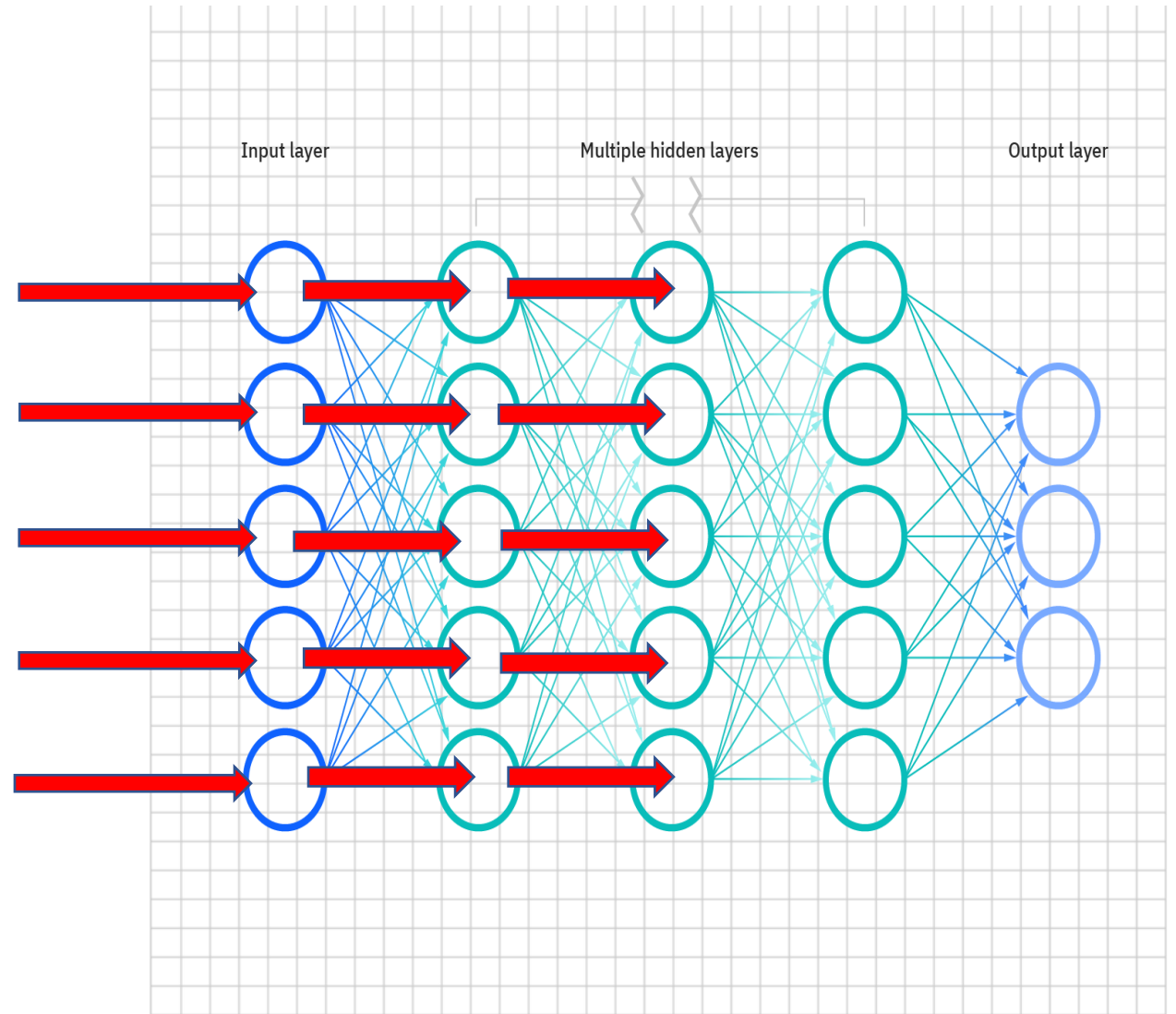
FPGAs in the Age of AI

- Supply the practitioners with the highest parallelism
- Built a specific Neural Network on Hardware
- Change it whenever you decide



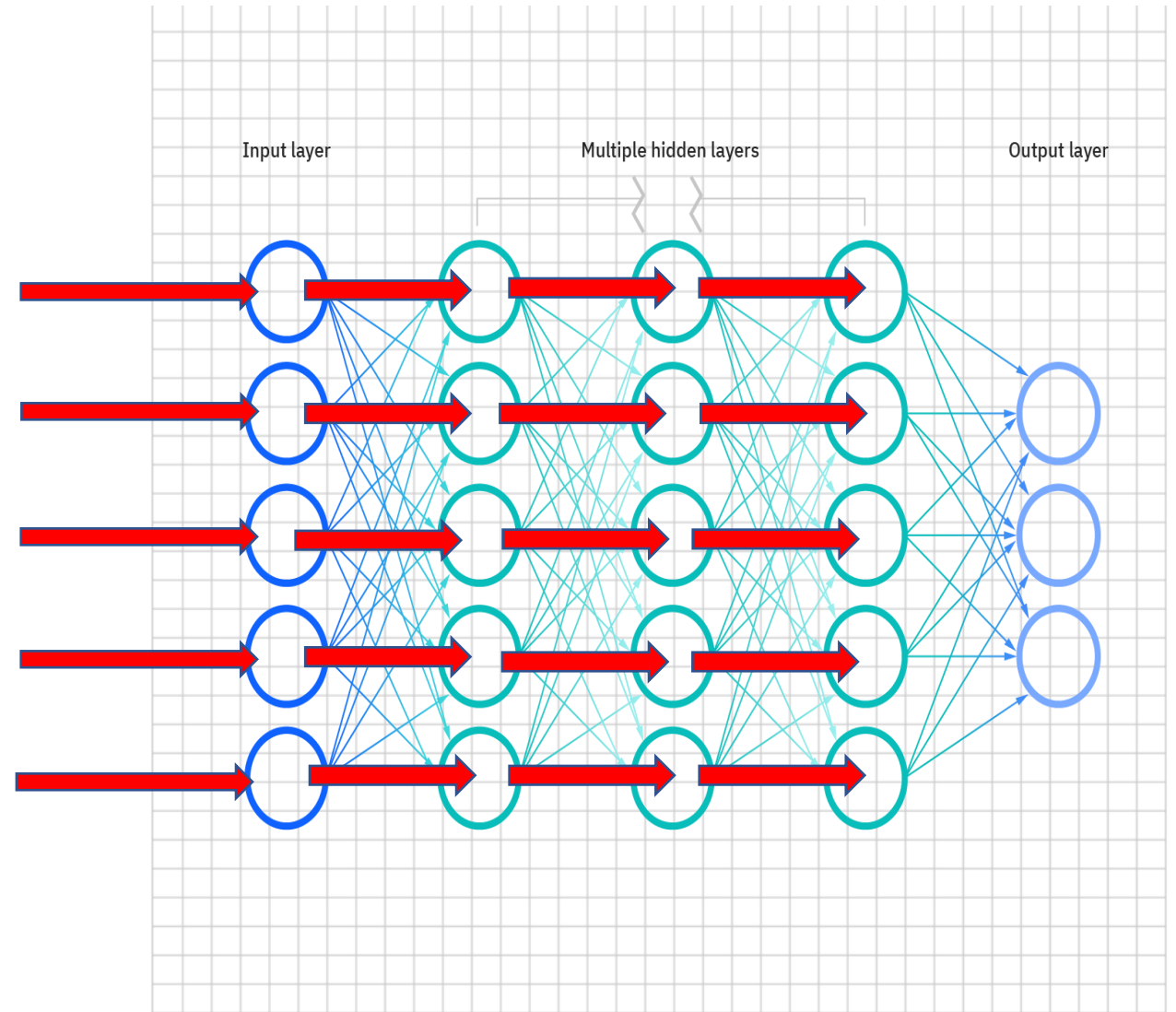
FPGAs in the Age of AI

- Supply the practitioners with the highest parallelism
- Built a specific Neural Network on Hardware
- Change it whenever you decide



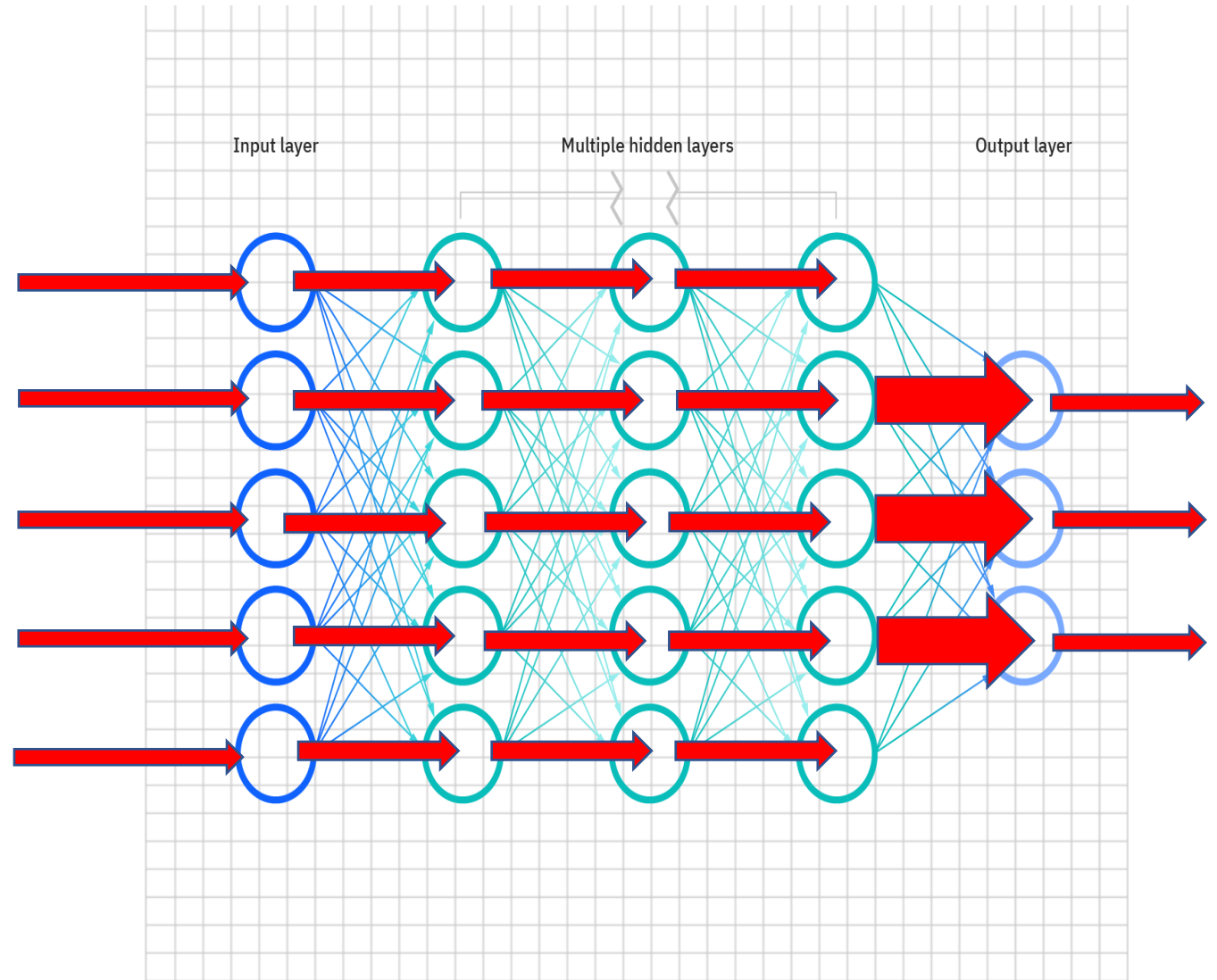
FPGAs in the Age of AI

- Supply the practitioners with the highest parallelism
- Built a specific Neural Network on Hardware
- Change it whenever you decide



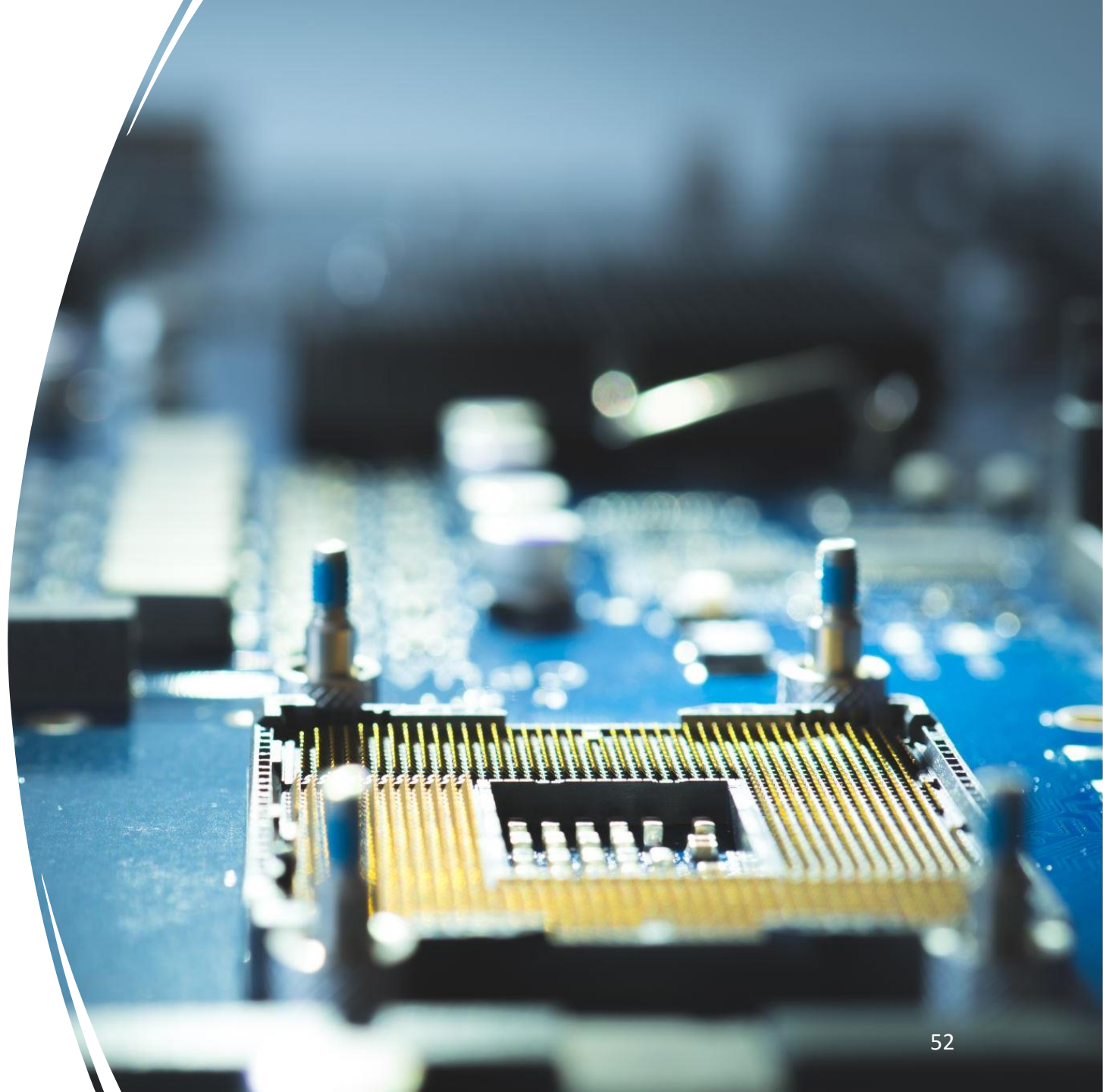
FPGAs in the Age of AI

- Supply the practitioners with the highest parallelism
- Built a specific Neural Network on Hardware
- Change it whenever you decide



Road Map

- Computing
- Processors
- How do electrons work for us?!
- CPU
- GPU
- FPGA
- **Accelerator**
- Tradeoff of processors



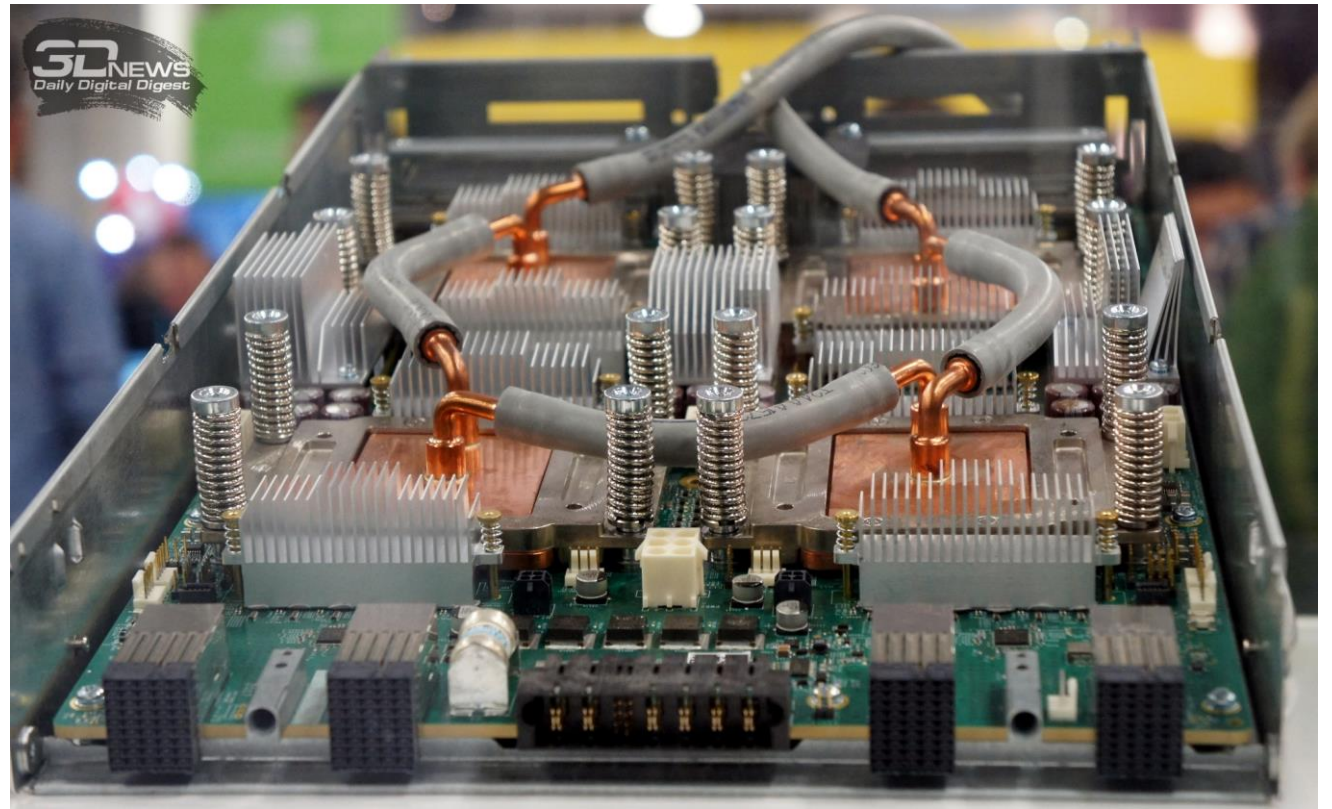
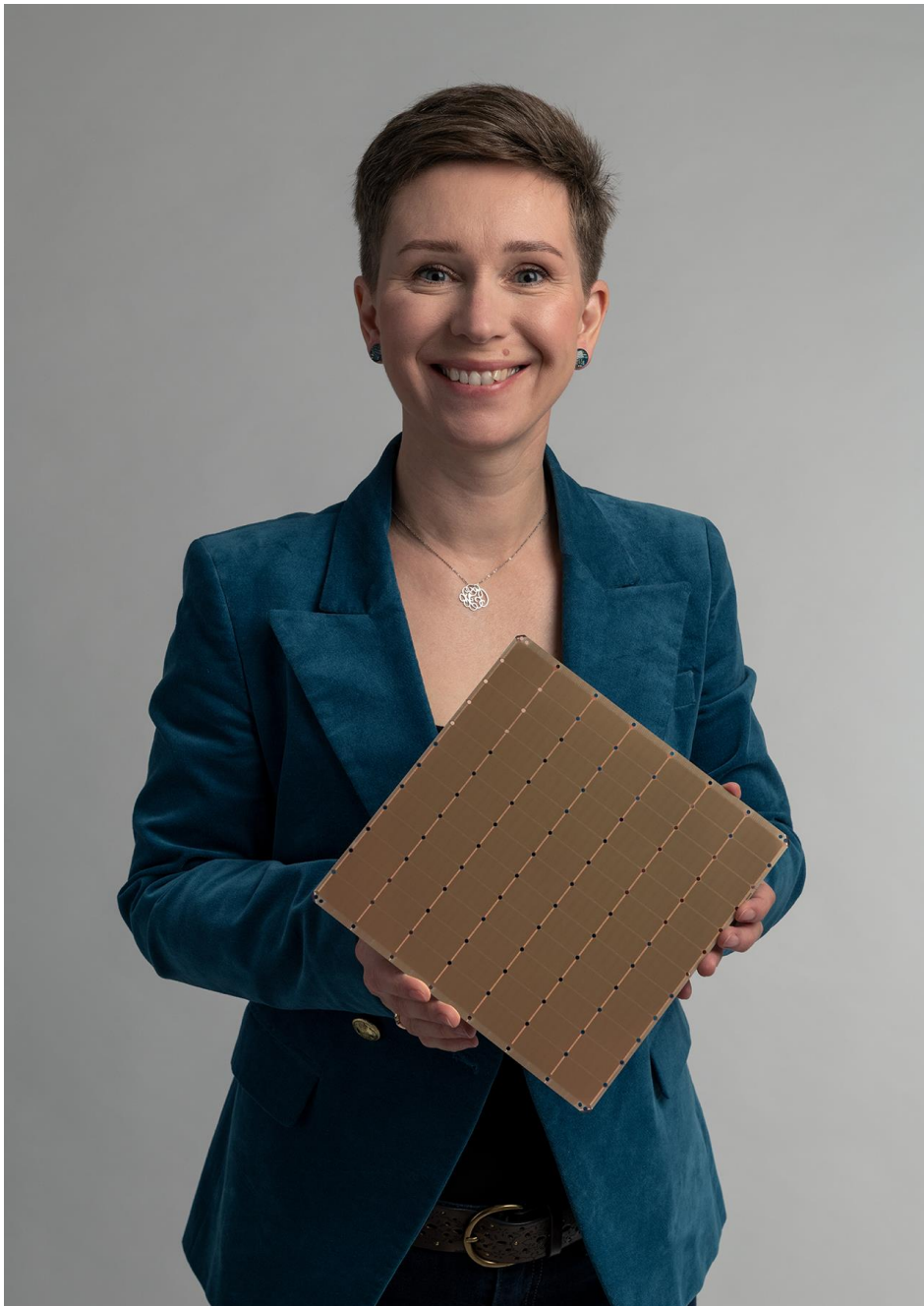
Accelerator

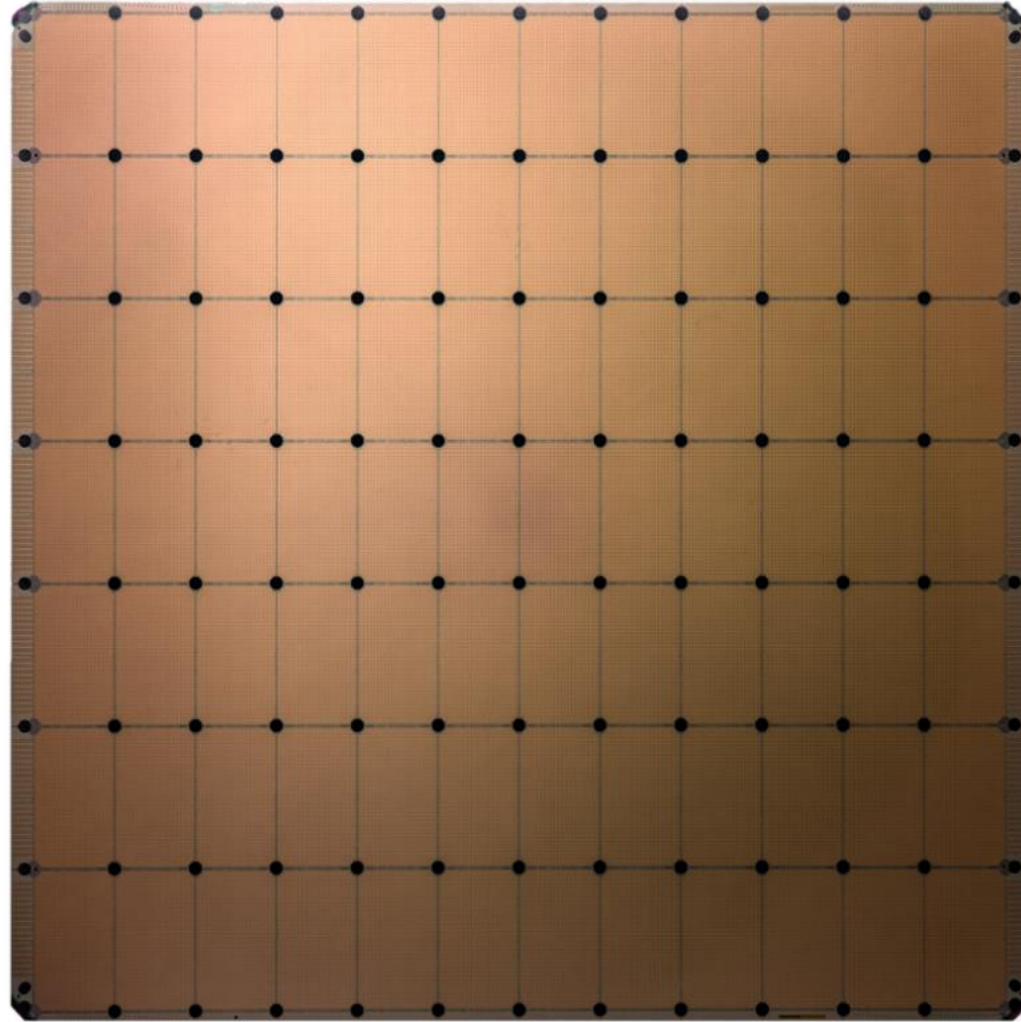
- Application Specific Integrated Circuit (ASIC)
- Everything is specified by the designers
- No flexibility is expected but depends on the designers!



GRAPHCORE







CEREBRAS WSE-3

46,225mm² Silicon
4 Trillion transistors

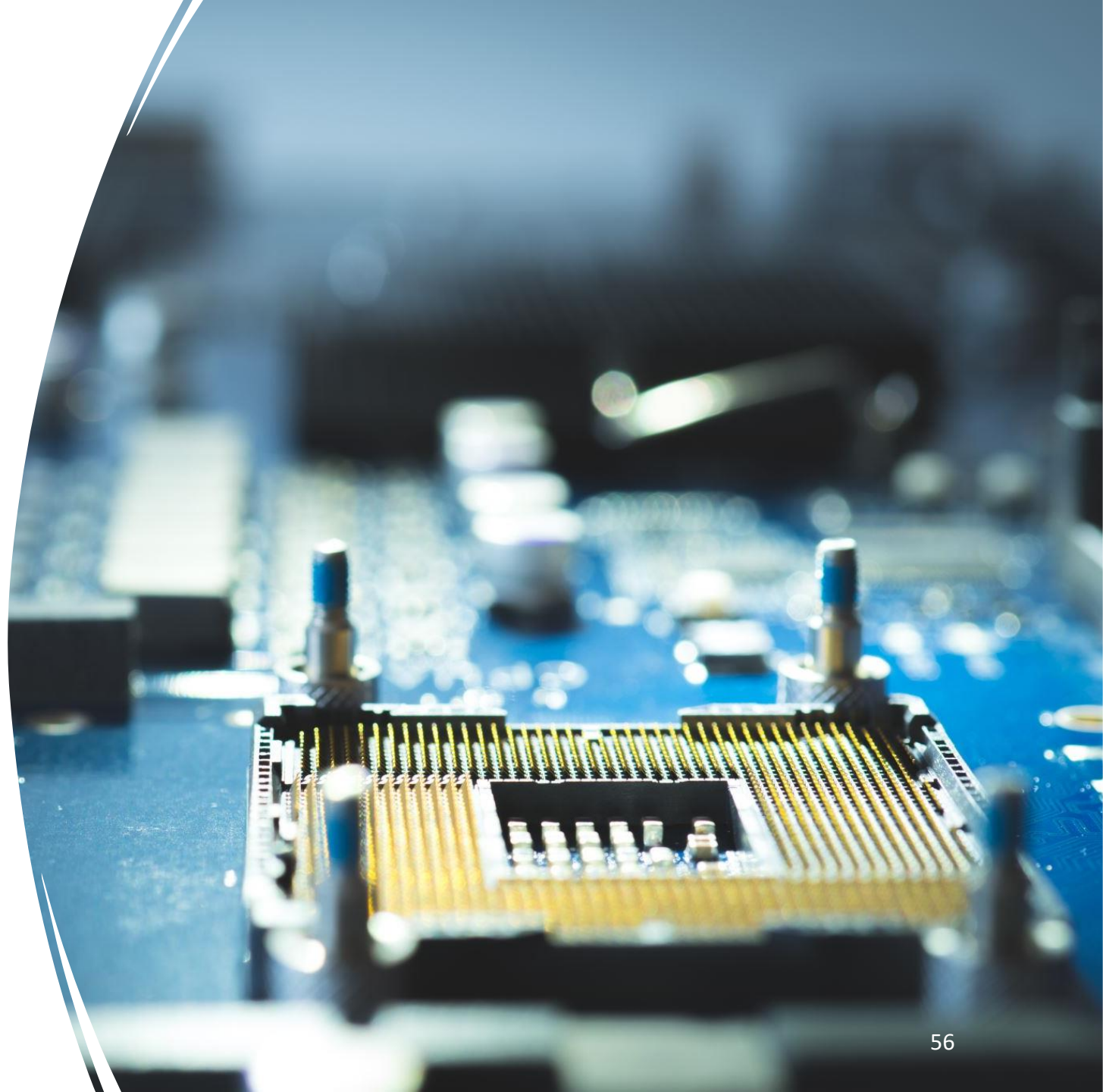


LARGEST GPU

826mm² Silicon
80 Billion transistors

Road Map

- Computing
- Processors
- How do electrons work for us?!
- CPU
- GPU
- FPGA
- Accelerator
- **Tradeoff of processors**



Wrap-up

Processor	Programmability	Goal Program	Flexibility (Different Programs)	Promised Performance
Central Processing Unit (CPU)	Easy (use any programming language you know)	Latency Oriented	Super High	Fair
Graphical Processing Unit (GPU)	Medium (learning CUDA!)	Throughput Oriented	High	Medium
Field Programmable Gate Array	Hard (Learn Verilog + Digital electronics basics)	Both	Low	High
Accelerator	Very Hard (read manuals)	Both	Super Low	Super High

Some useful links for you

[CUDA C++ Programming Guide \(nvidia.com\)](#)

[CUDA Toolkit - Free Tools and Training | NVIDIA Developer](#)

[NVIDIA Blog](#)

[Deep Learning Institute and Training Solutions | NVIDIA](#)

[DGX Platform | NVIDIA](#)

[Intel Field Programmable Gate Arrays \(FPGA\) Technical Training | Intel](#)

[Product - Chip - Cerebras](#)

[Products | Coral](#)

[Altera® FPGAs and Programmable Devices \(intel.com\)](#)

[Reimagining the Data Center \(amd.com\)](#)

[FPGAs & 3D ICs \(xilinx.com\)](#)

[Intel® Core™ Processors - View Latest Generation Core Processors](#)

[AMD Processors | AMD](#)

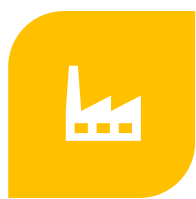
Conclusion



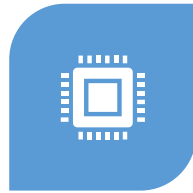
COMPUTATION



**PROCESSORS
MAKE THE
COMPUTATION
POSSIBLE**



**TRADEOFF OF
DIFFERENT
PROCESSORS**



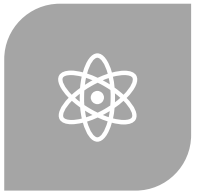
CPU



GPU



FPGA



ACCELERATOR



Questions?

Thanks for your attention!

BackUp if sb was curious!

Modern CPUs

- They fetch and execute more than one instruction (a windows of instruction)
 - Higher throughput
- Advanced Hardware Execution Mechanisms to execute faster
- Employ Cache Hierarchy to fill the Memory-Processor performance gap
 - Temporal/ Spatial Locality
- They have several cores (parallel computing)



Hennessy and Patterson



Tomasulo

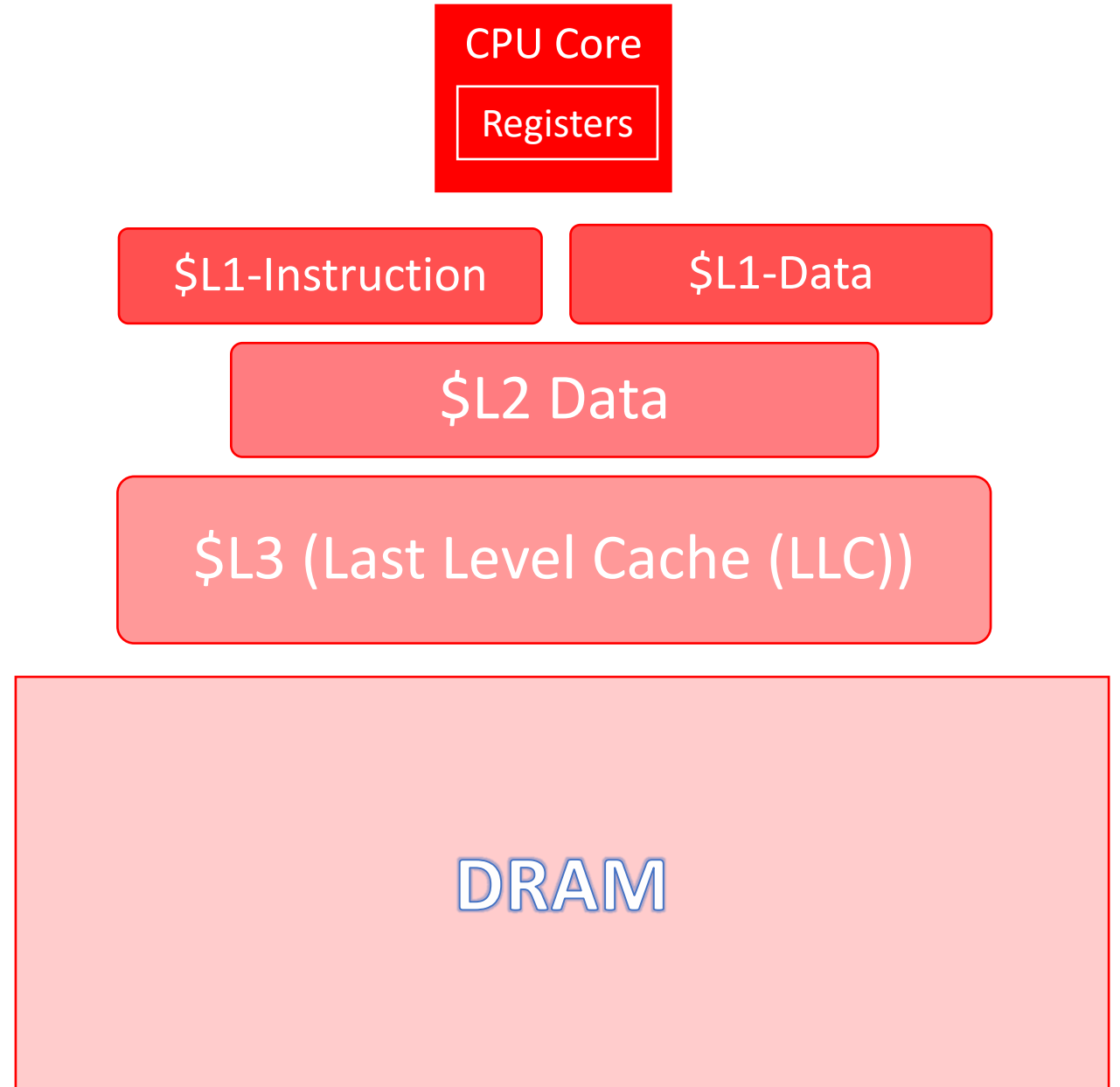


Yale Patt

Cache Hierarchy

Less Access latency
More Data Locality

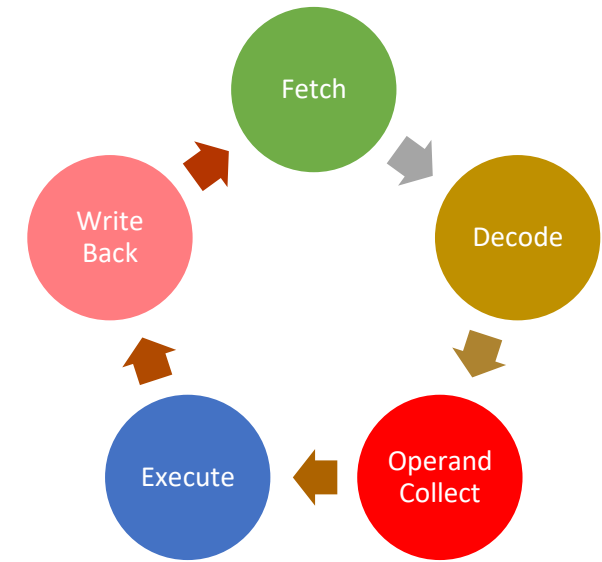
Less Storage Capacity
More Expensive per bit



Pipelining

- Basic Processor

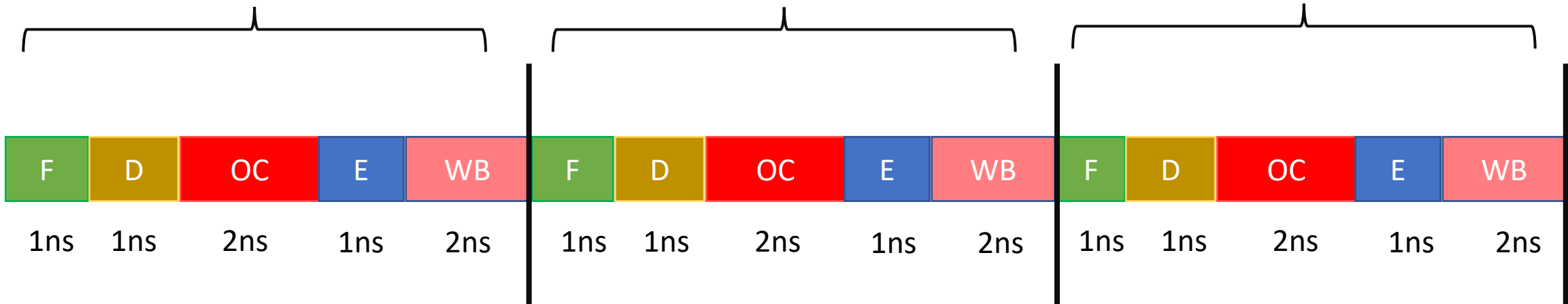
Inst1	O11, O12
Inst2	O21, O22
Inst3	O31, O32



Instruction 1

Instruction 2

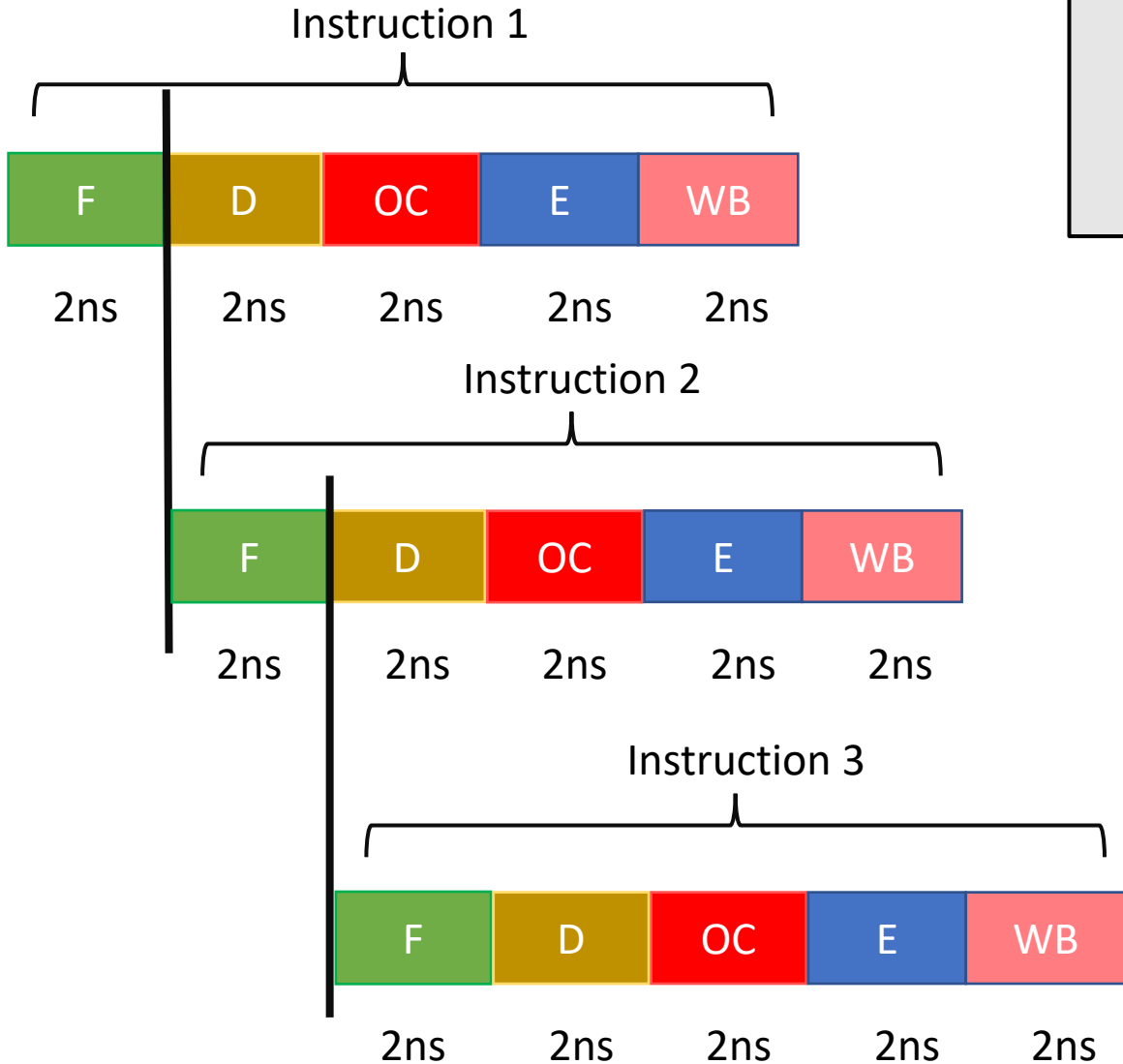
Instruction 3



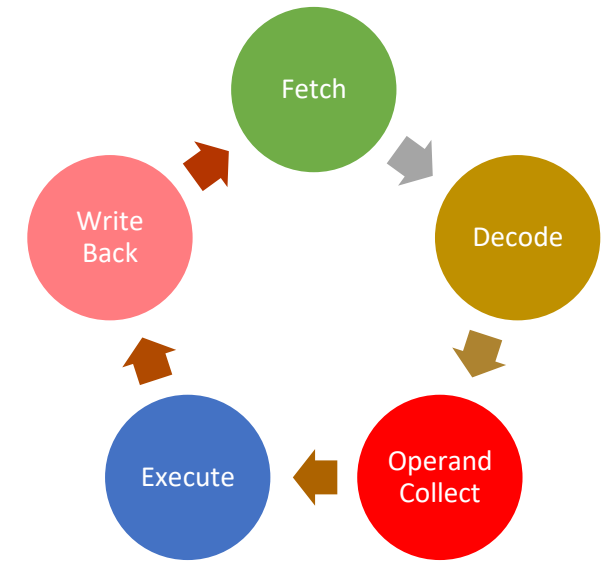
Overall Time of Executing three Instruction: $3 * (1ns + 1ns + 2ns + 1ns + 2ns) = 3 * 7ns = 21ns$

Pipelining

- Pipelined Basic Processor



Inst1	O11, O12
Inst2	O21, O22
Inst3	O31, O32



Overall Time of Executing three Instruction:

$$= (2ns + 2ns + 2ns + 2ns + 2ns) + 2ns + 2ns$$

$$= 10ns + 2ns + 2ns = 14ns$$

Implicit Parallelism

Instruction-Level Parallelism (ILP)

Instruction-Level Parallelism (ILP)