Processors

Ehsan Yousefzadeh-Asl-Miandoab

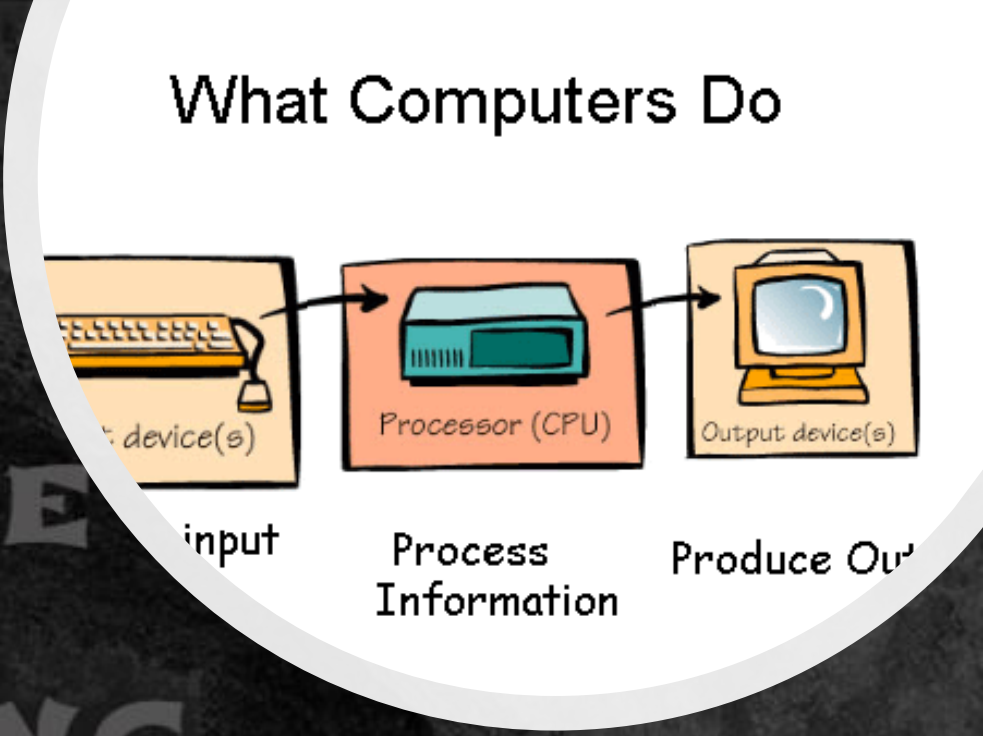
ehyo@itu.dk

April-2022

Road Map

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

Computing



USING COMPUTER TECHNOLOGY TO COMPLETE A GOAL-ORIENTED TASK.

TRANSFORMING THE INPUT DATA INTO OUTPUT DATA.

GAINING KNOWLEDGE (INSIGHT)

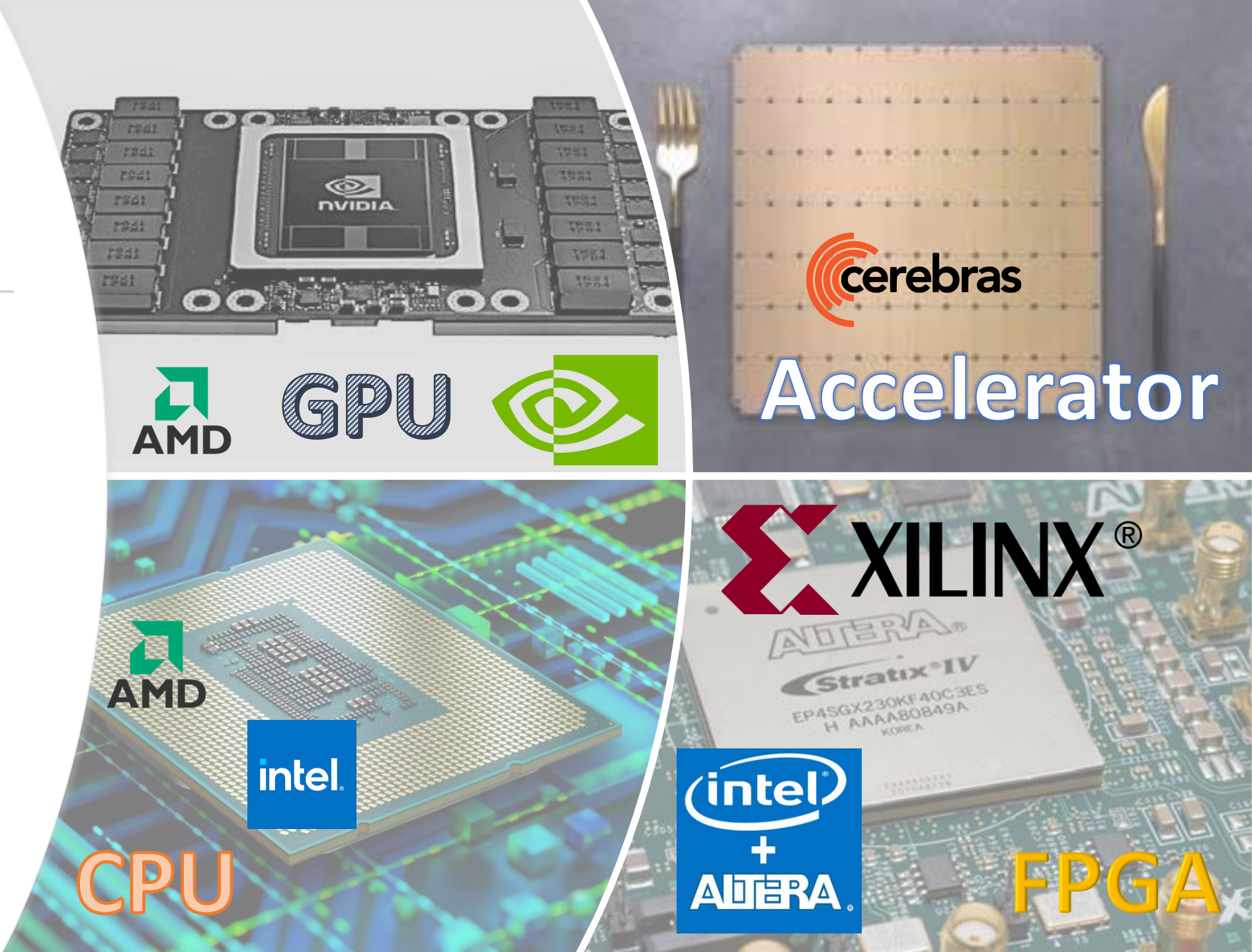
Richard Hamming (1915 - 1998)

Road Map

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

Processors

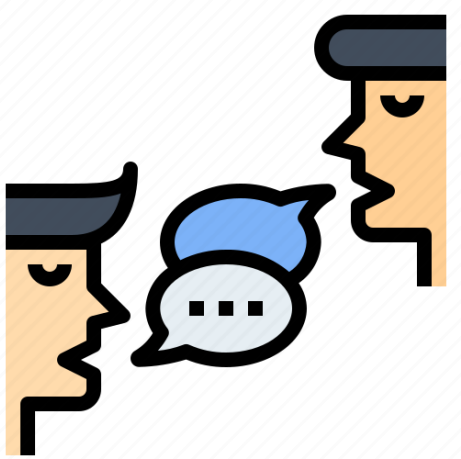
- Making Computing possible
- Making electrons to work for us to do the computation



Road Map

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

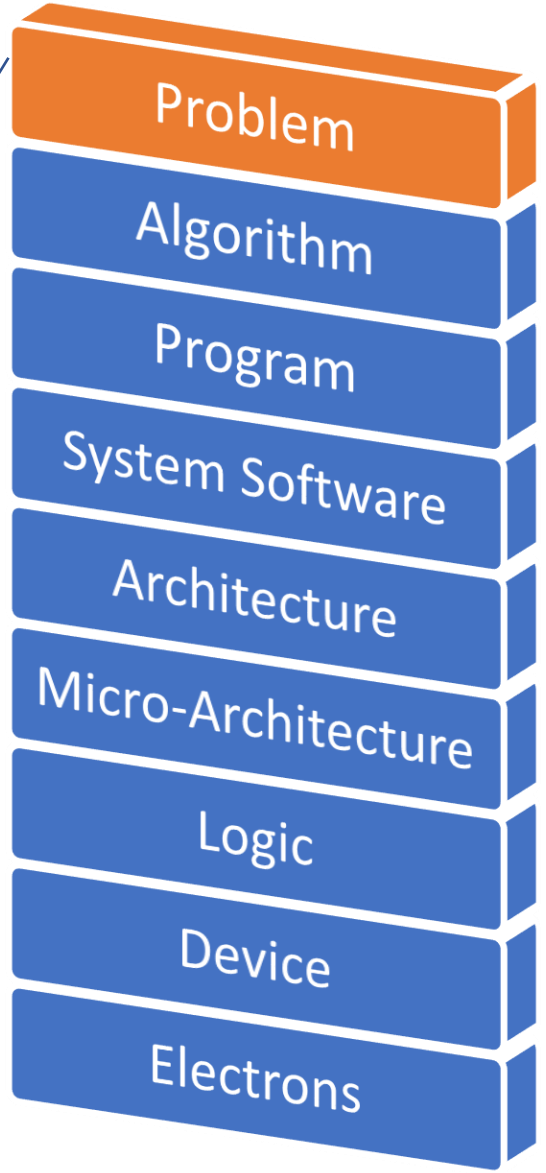
Electrons work for us?! But how?



Examples:

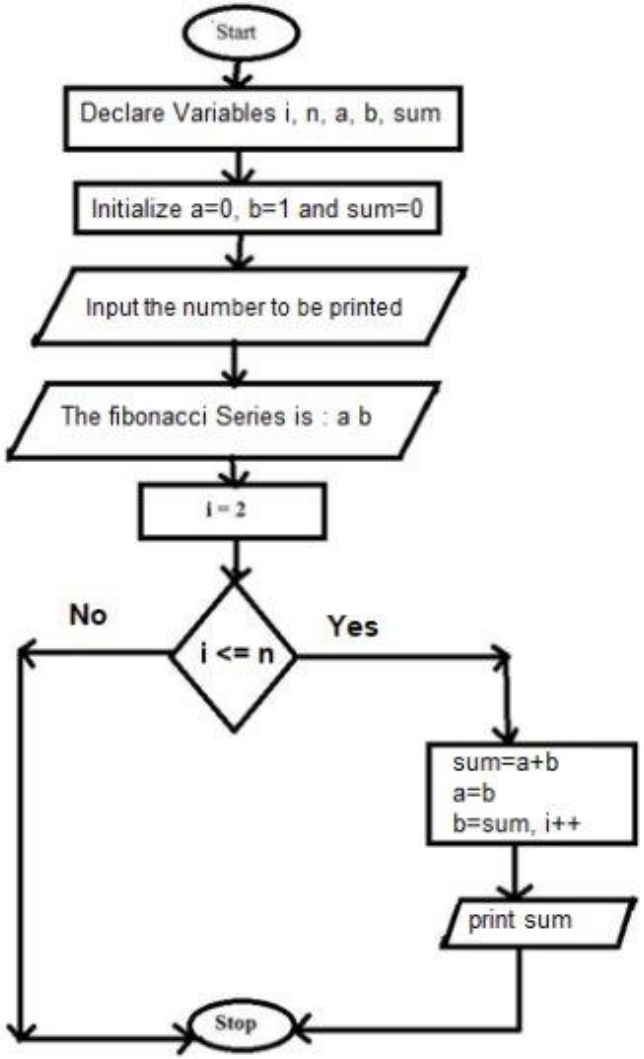
1. Calculating the First n Numbers of Fibonacci Series

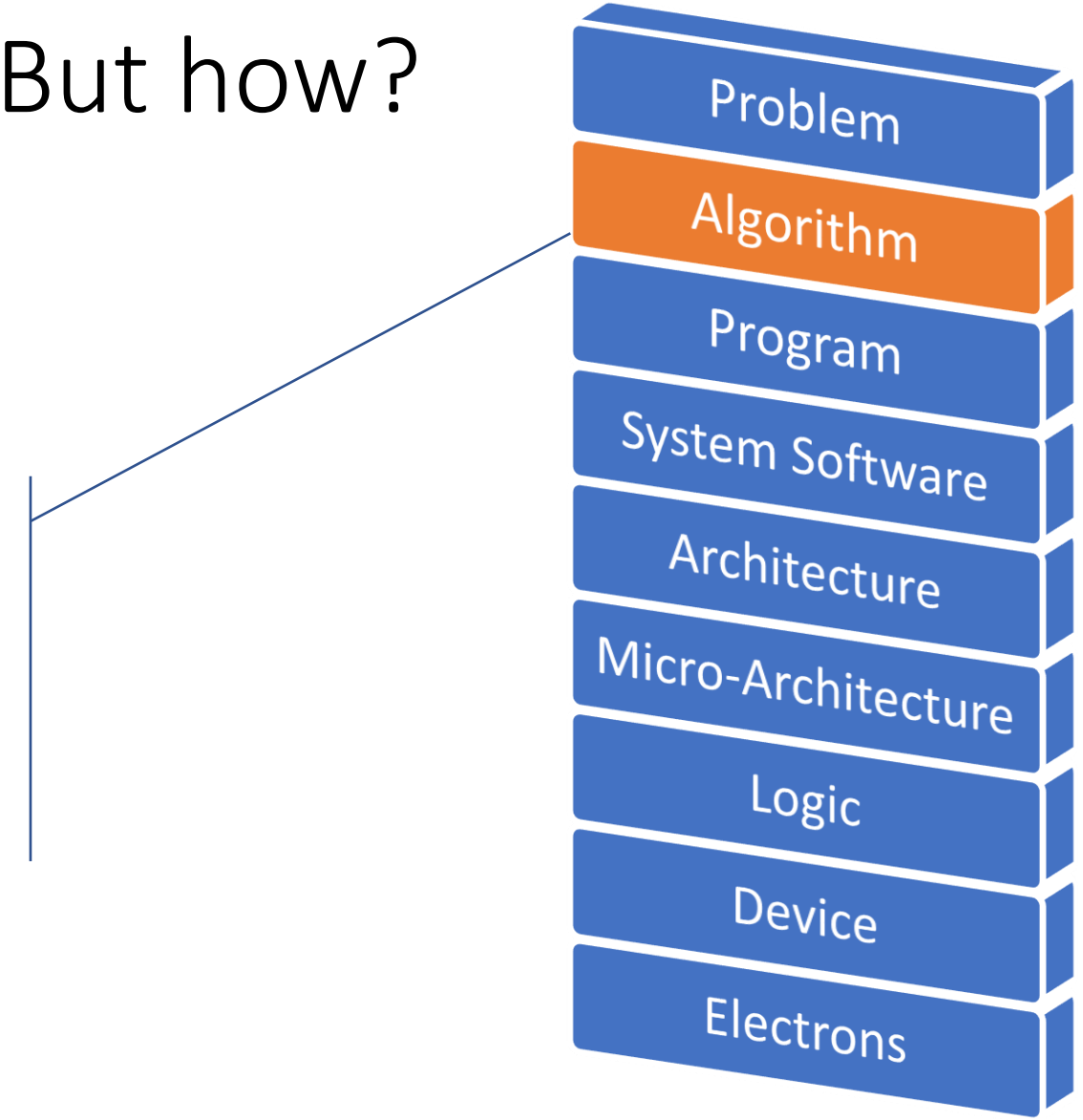
2. Train a Machine Learning Model



Transformation Hierarchy

Electrons work for us?! But how?



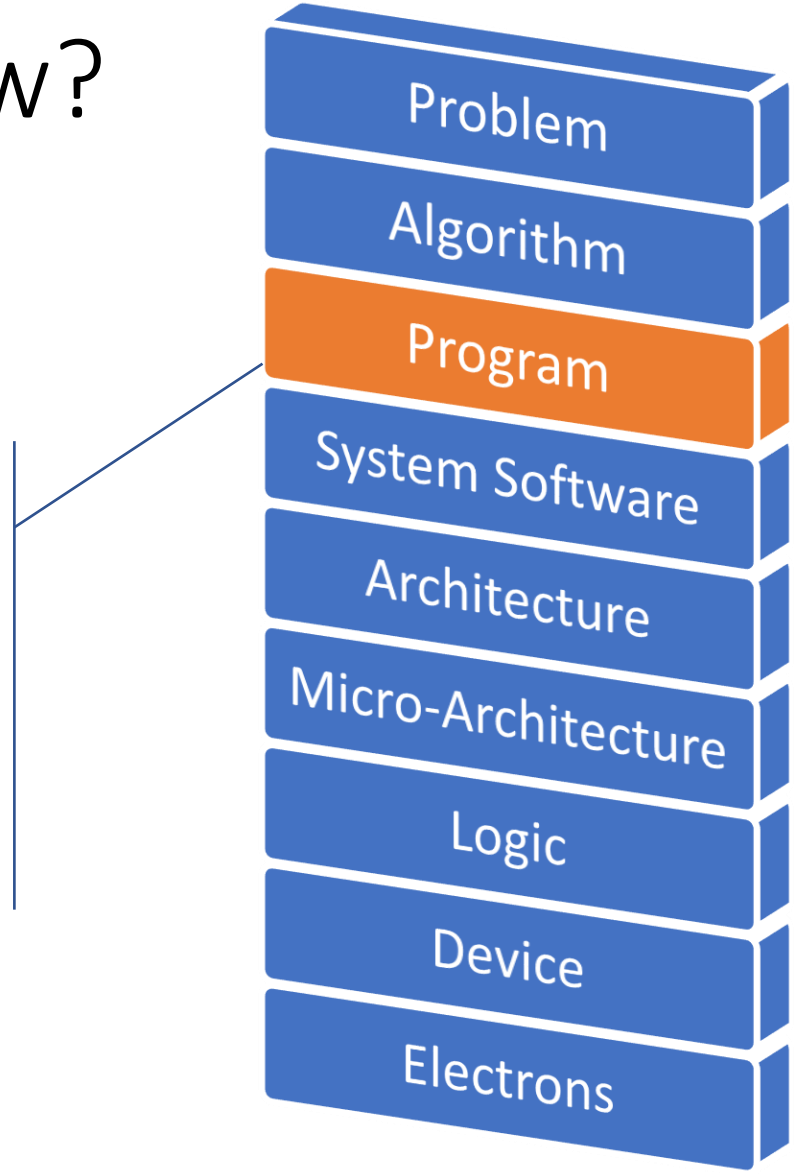


Transformation Hierarchy

Electrons work for us?! But how?

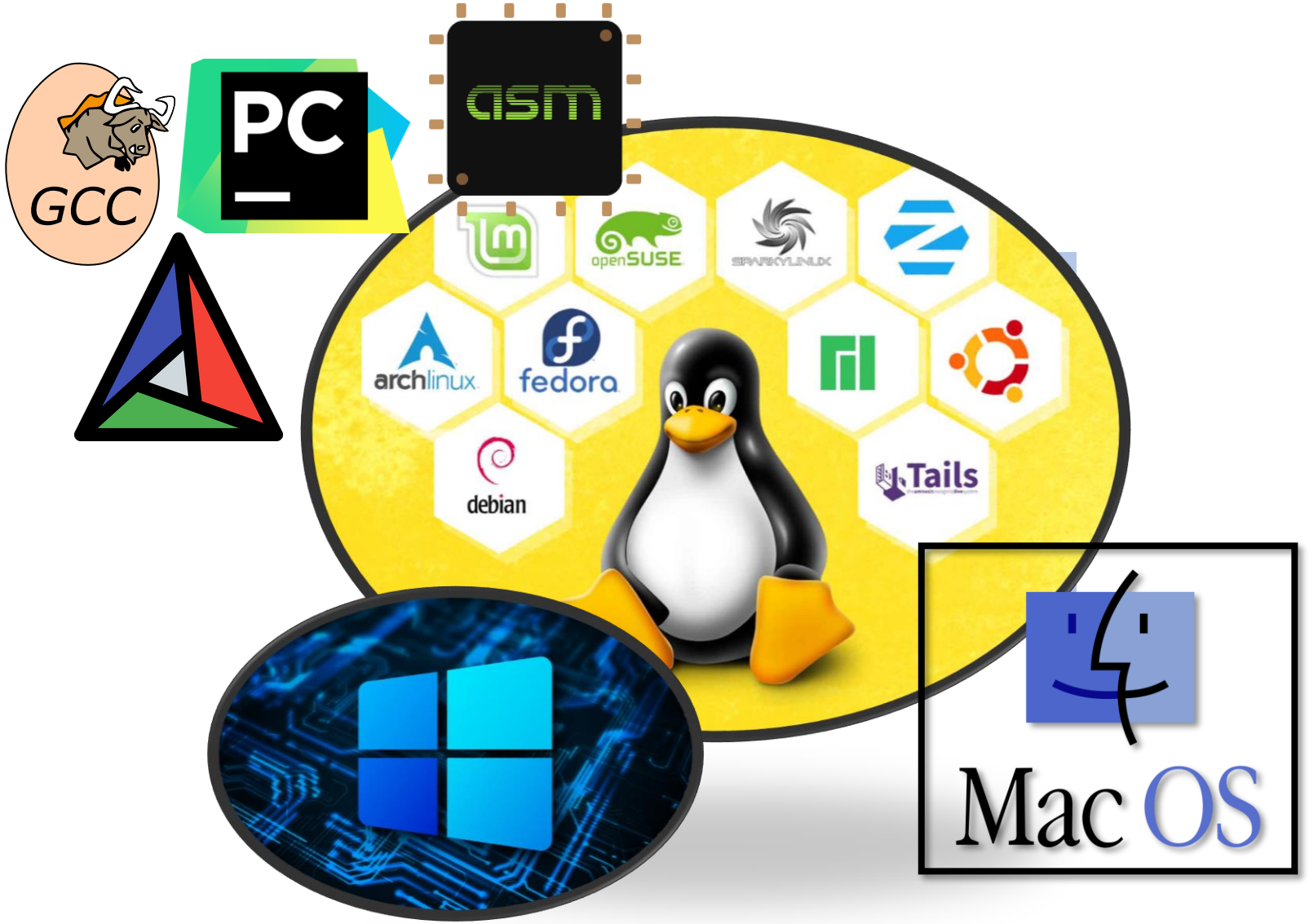
```cpp
#include <iostream>
using namespace std;

int main() {
    int n, t1 = 0, t2 = 1, nextTerm = 0;

    cout << "Enter the number of terms: ";
    cin >> n;

    cout << "Fibonacci Series: ";

    for (int i = 1; i <= n; ++i) {
        // Prints the first two terms.
        if(i == 1) {
            cout << t1 << ", ";
            continue;
        }
        if(i == 2) {
            cout << t2 << ", ";
            continue;
        }
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;

        cout << nextTerm << ", ";
    }
    return 0;
}
```

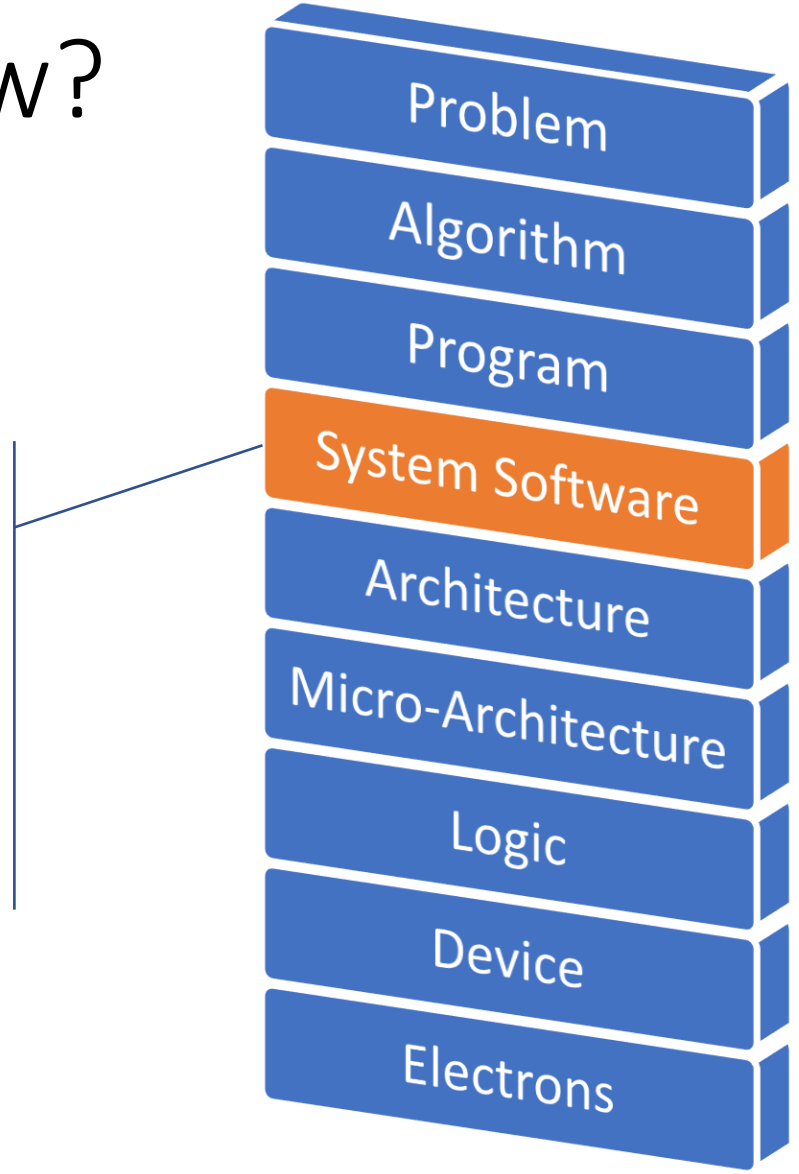
```python
# Program to display the Fibonacci sequence up to n-th term

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

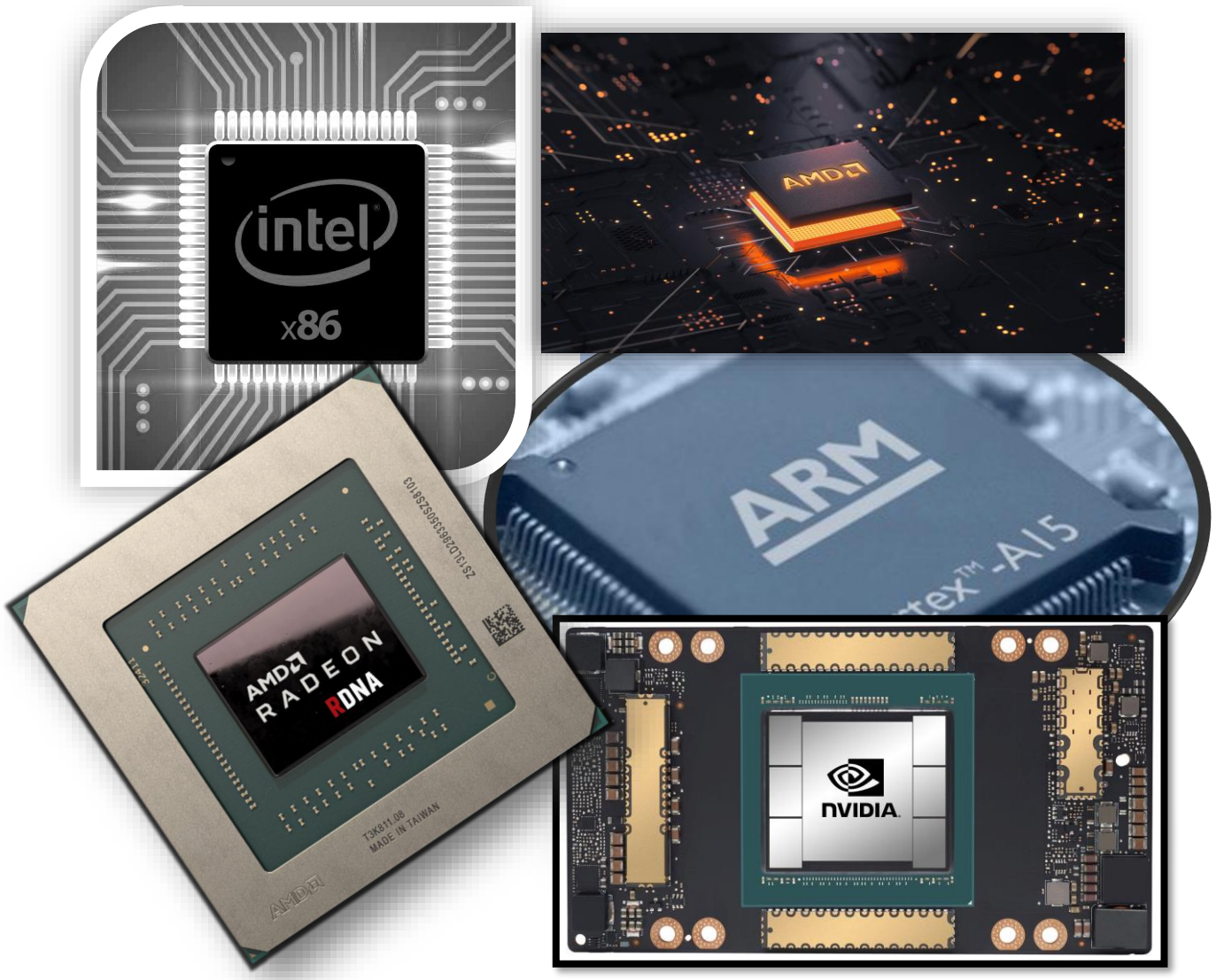
Electrons work for us?! But how?

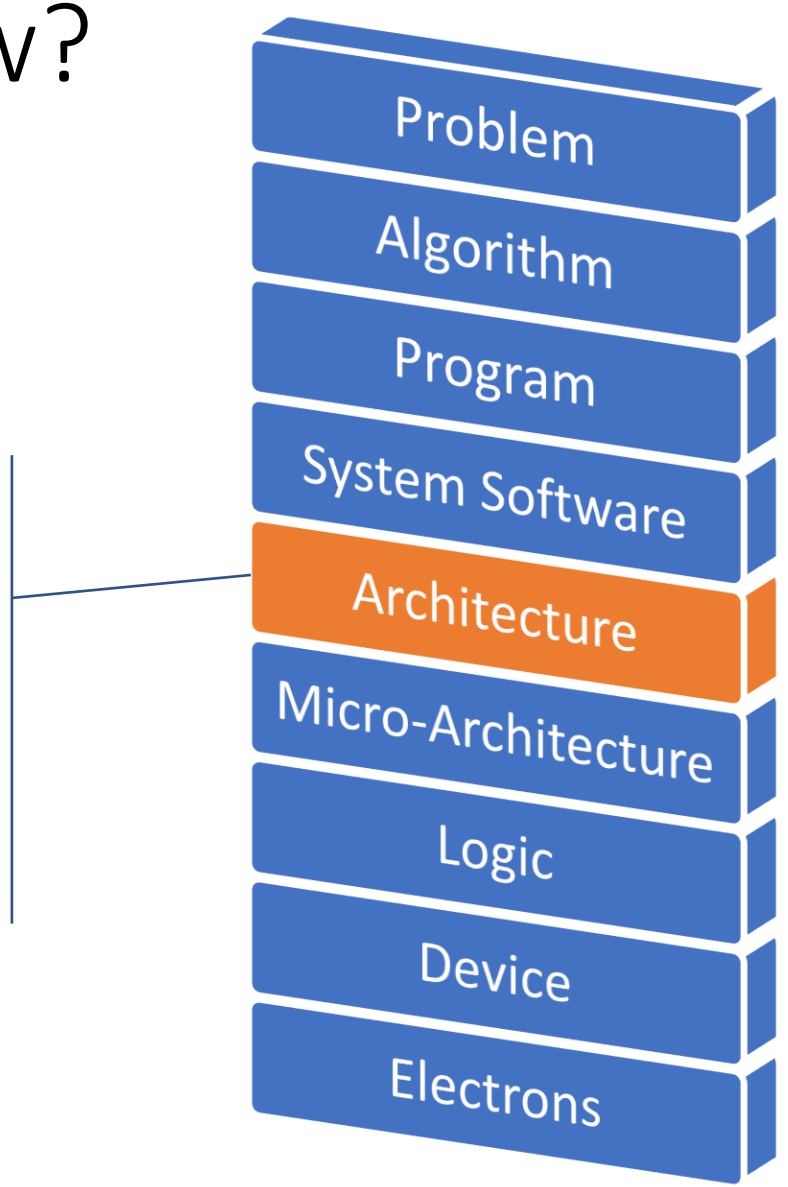




Transformation Hierarchy

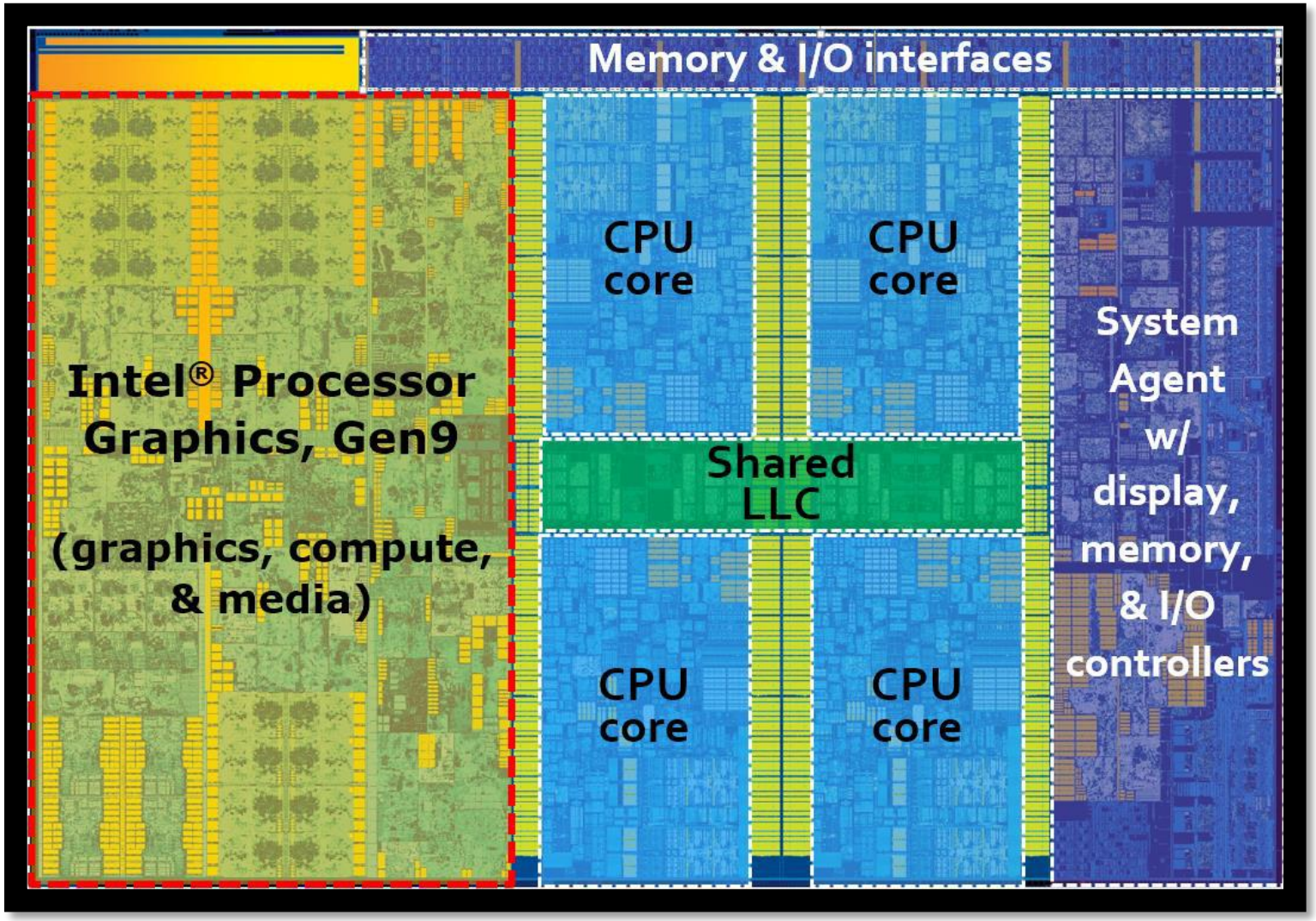
Electrons work for us?! But how?

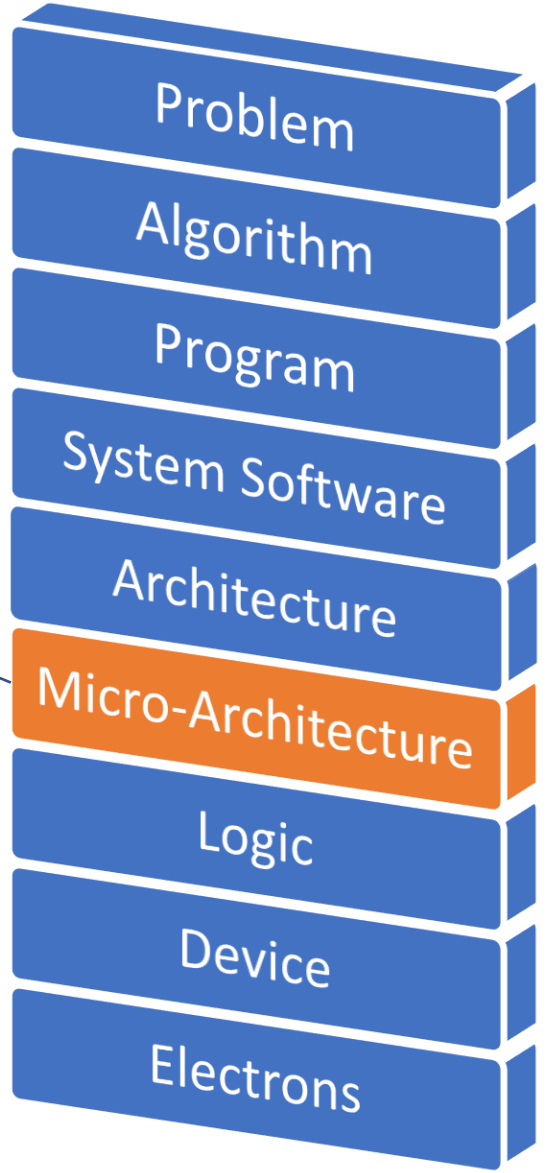




Transformation Hierarchy

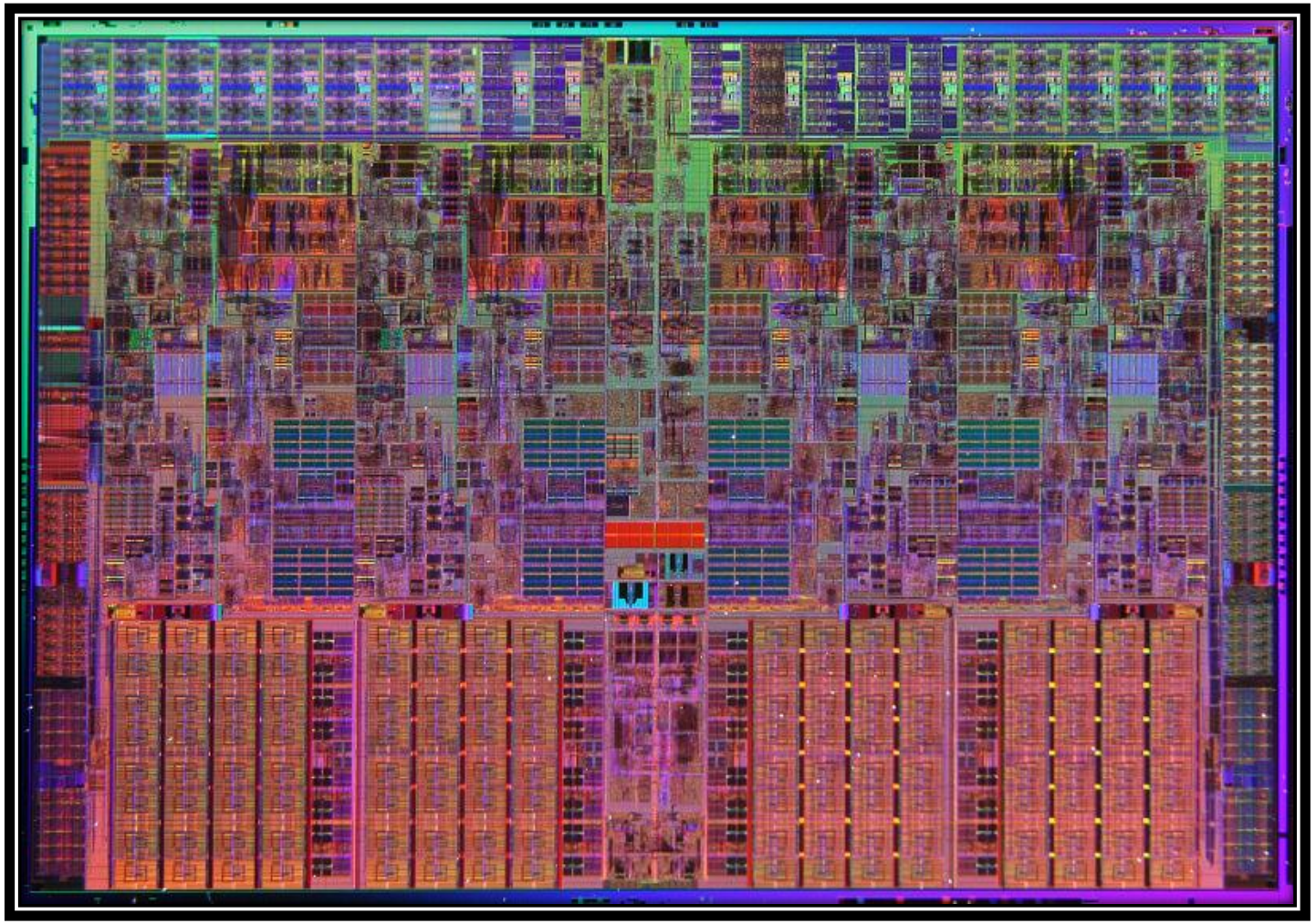
Electrons work for us?! But how?

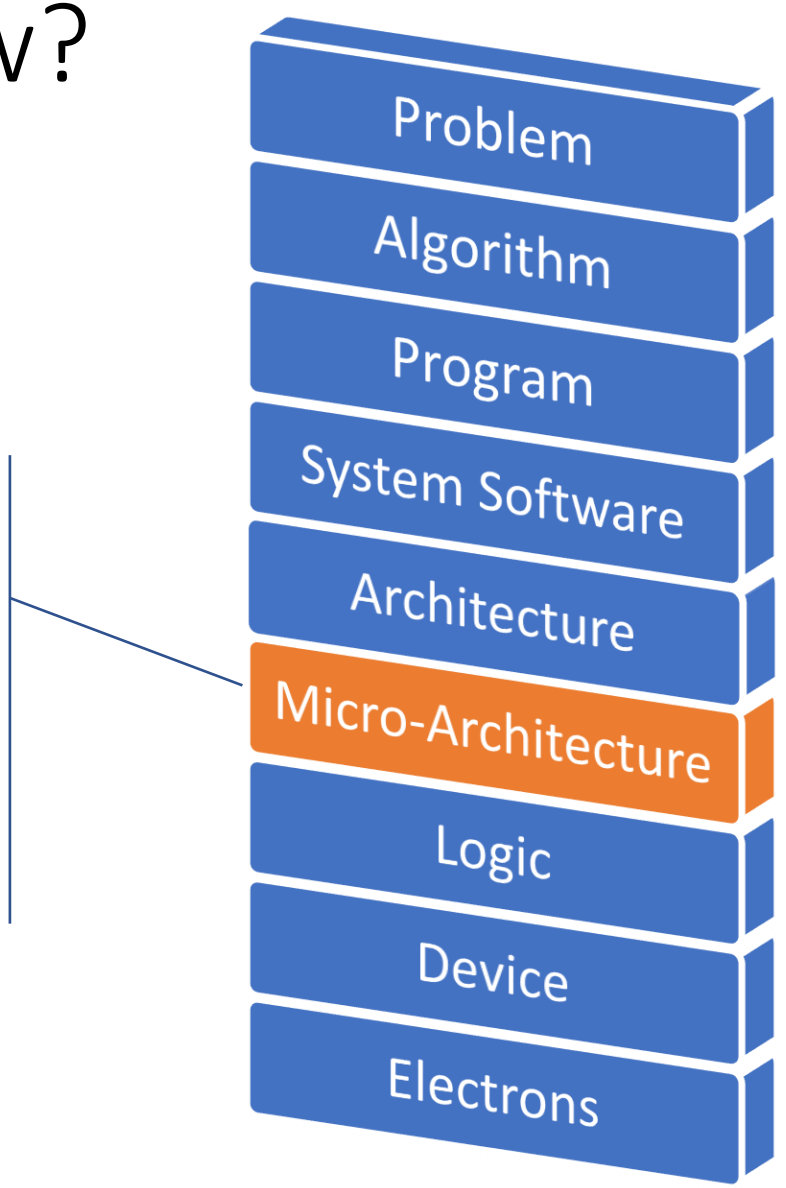




Transformation Hierarchy

Electrons work for us?! But how?

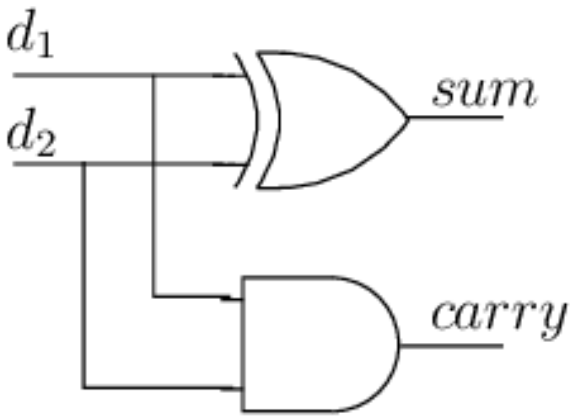


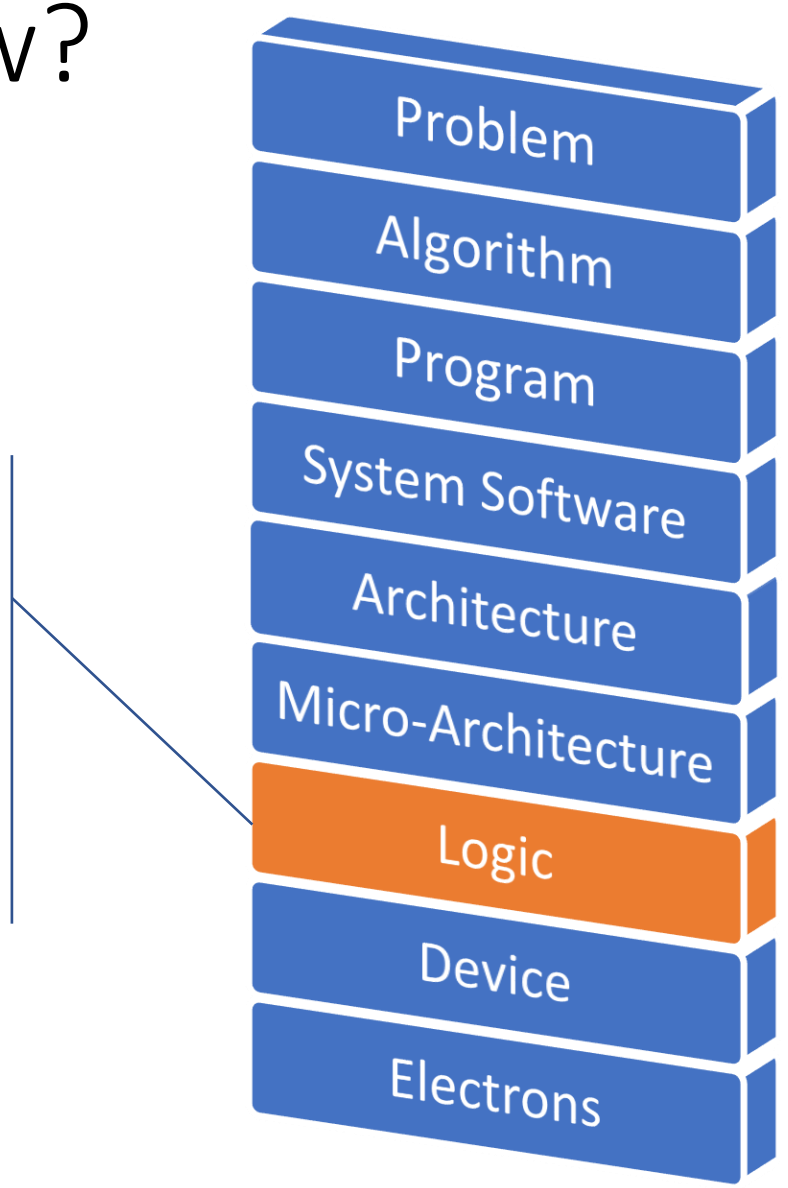


Transformation Hierarchy

Electrons work for us?! But how?

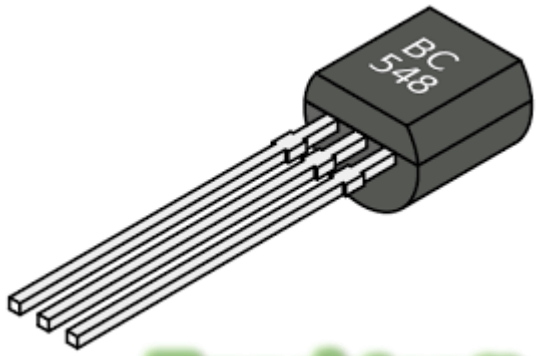
d1	d2	sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



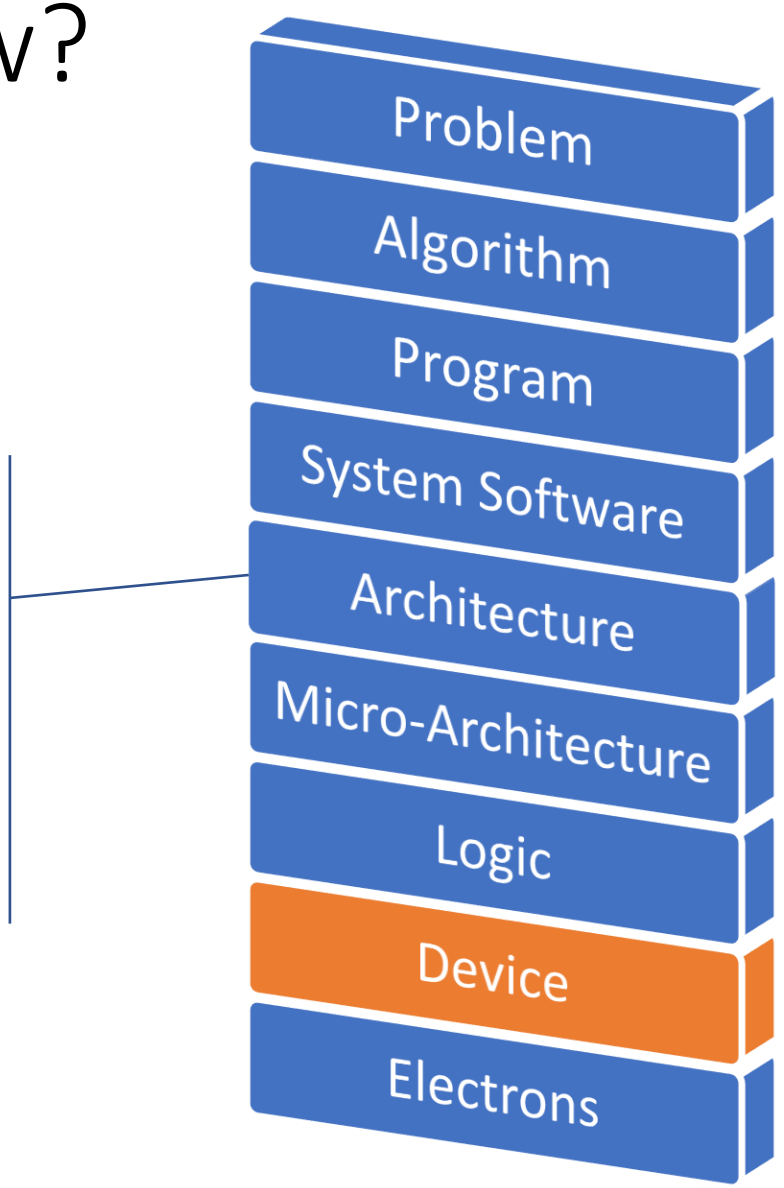


Transformation Hierarchy

Electrons work for us?! But how?

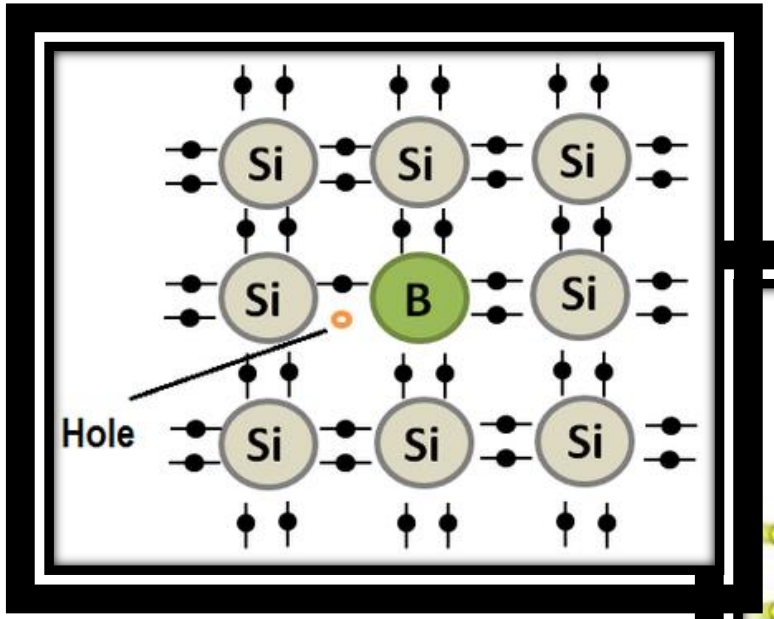


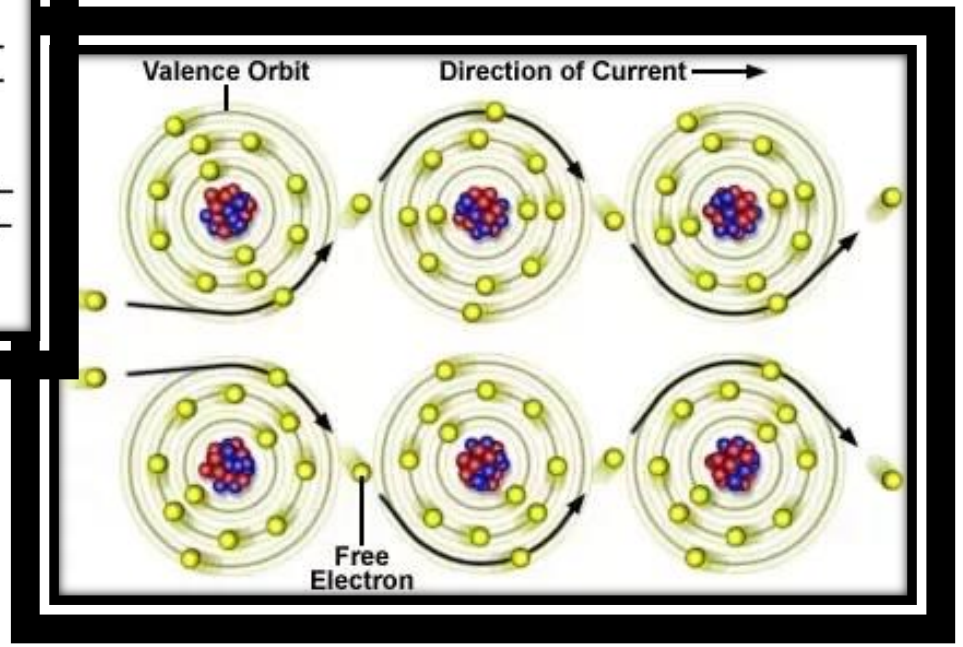
Transistor ~= Switch

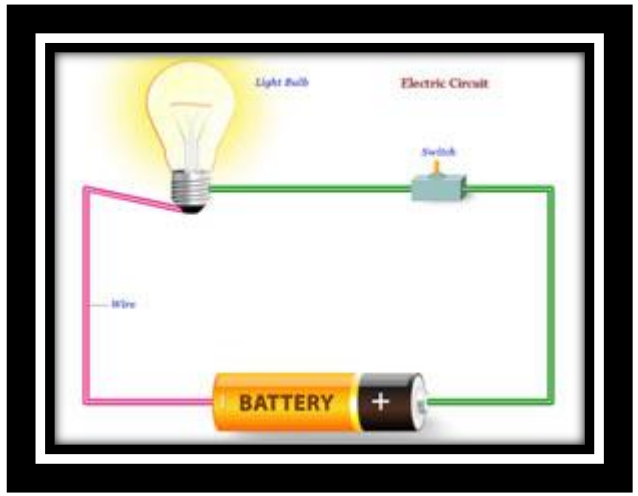


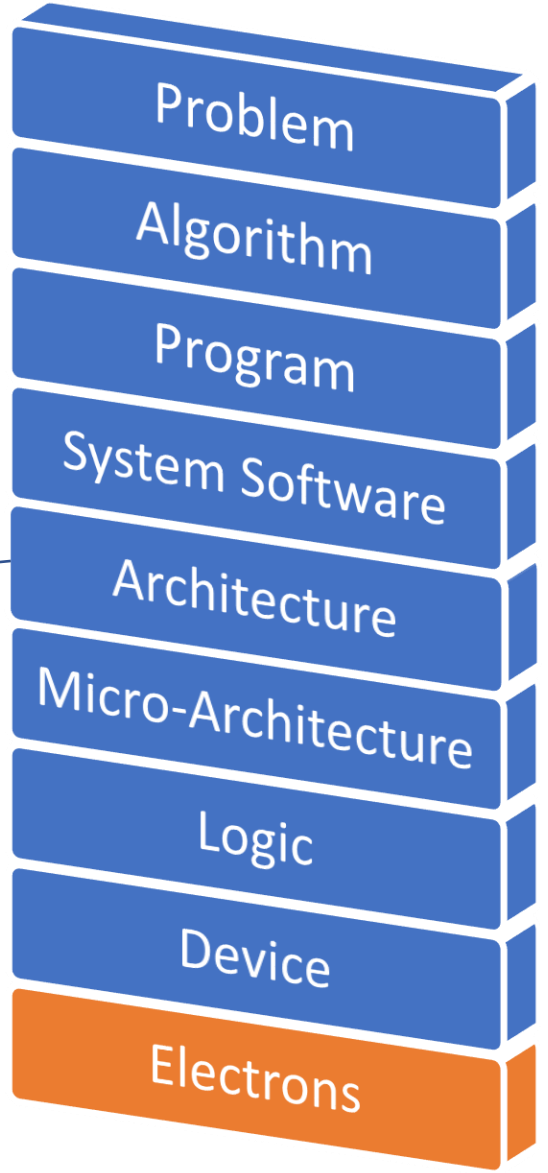
Transformation Hierarchy

Electrons work for us?! But how?









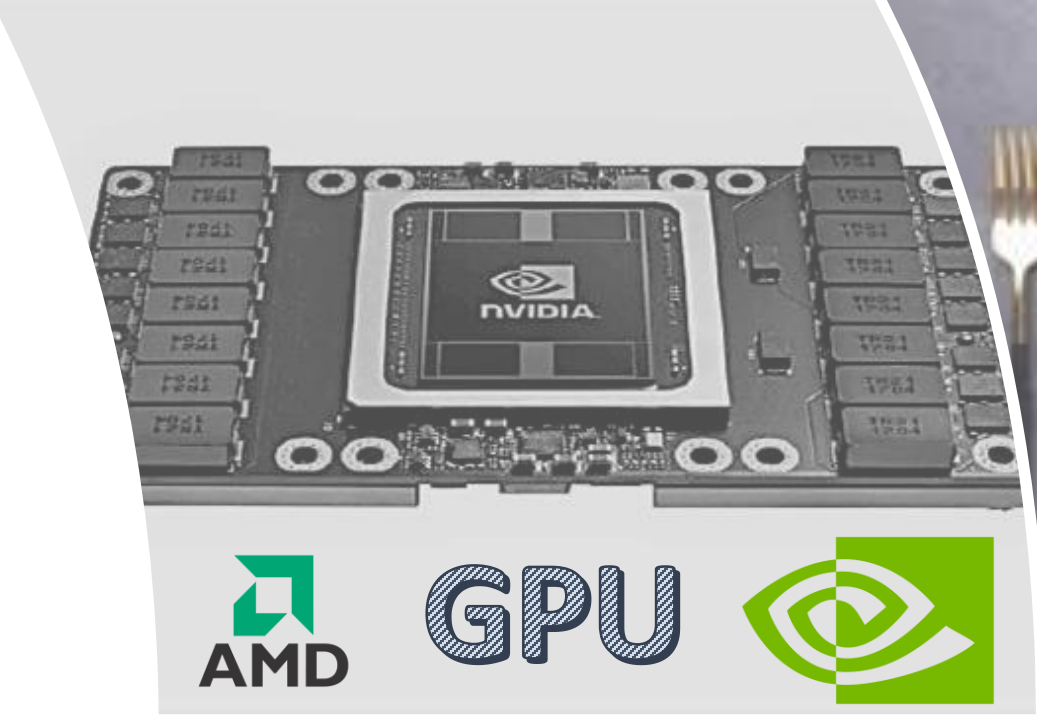
Transformation Hierarchy

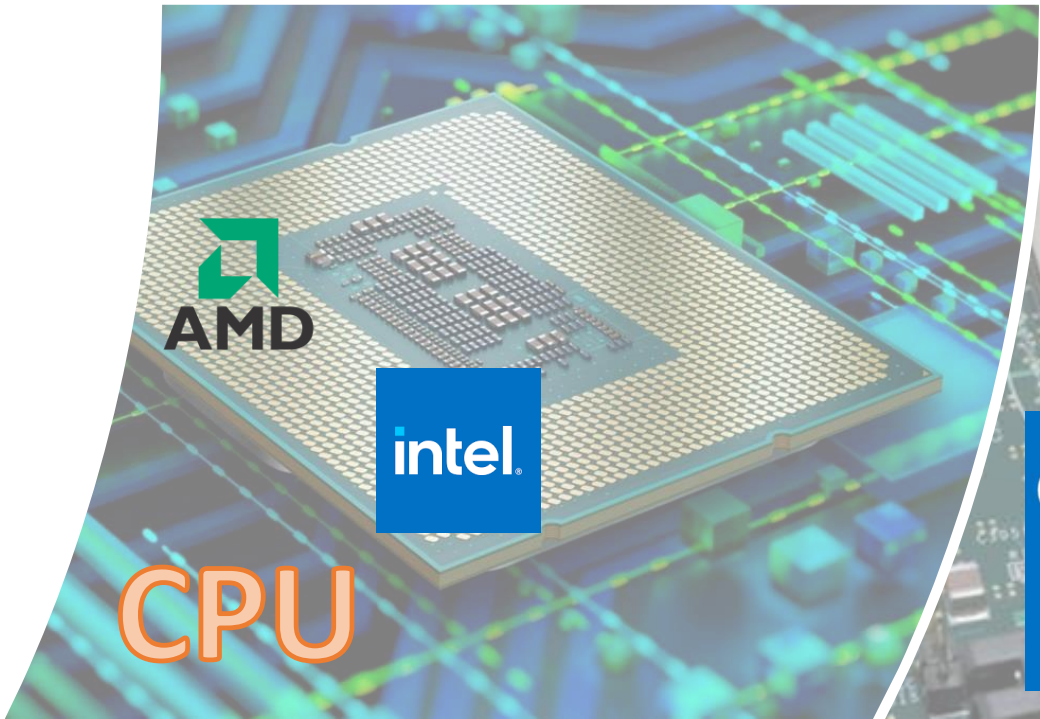
Road Map

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

Processors

- Making Computing possible
- Making electrons to work for us to do the computation





Tradeoff

Processor	Programmability	Goal Program	Flexibility (Different Programs)	Promised Performance
Central Processing Unit (CPU)	Easy	Latency Oriented	Super High	Fair
Graphical Processing Unit (GPU)	Medium	Throughput Oriented	High	Medium
Field Programmable Gate Array	Hard	Both	Low	High
Accelerator	Super Hard	Both	Super Low	Super High

Road Map

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

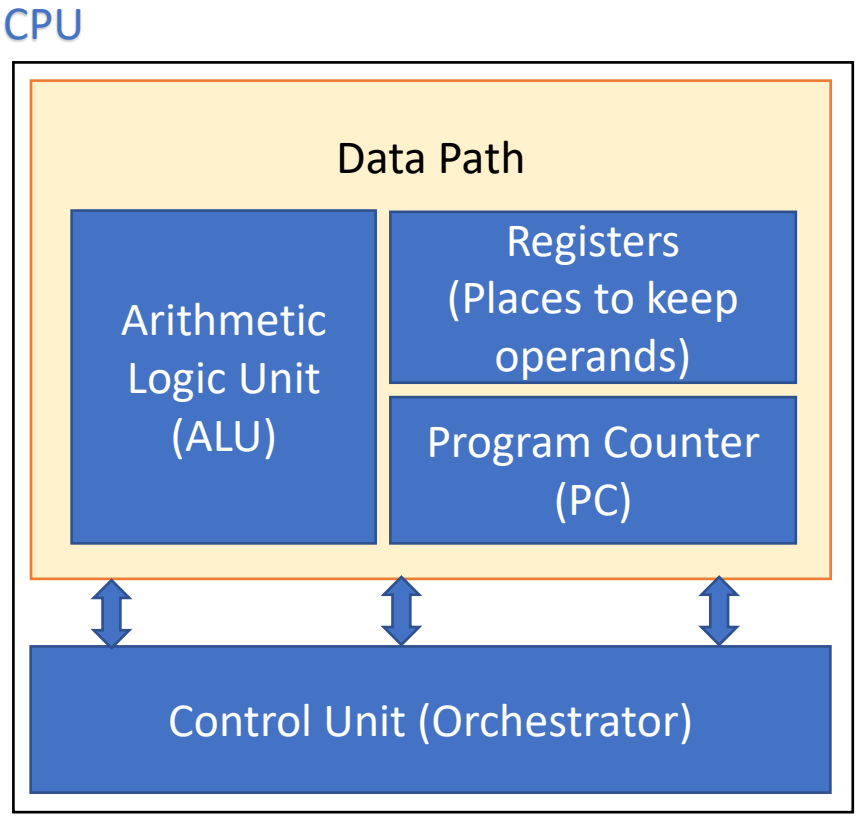
CPU

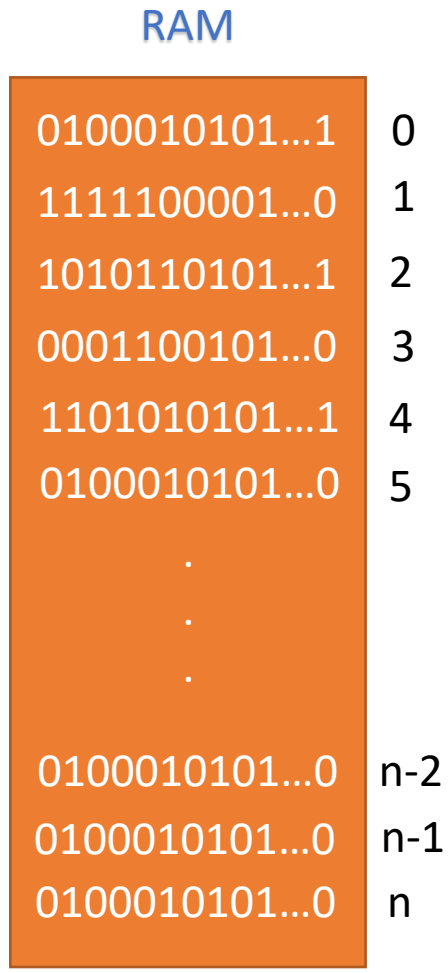
Sequential Processor

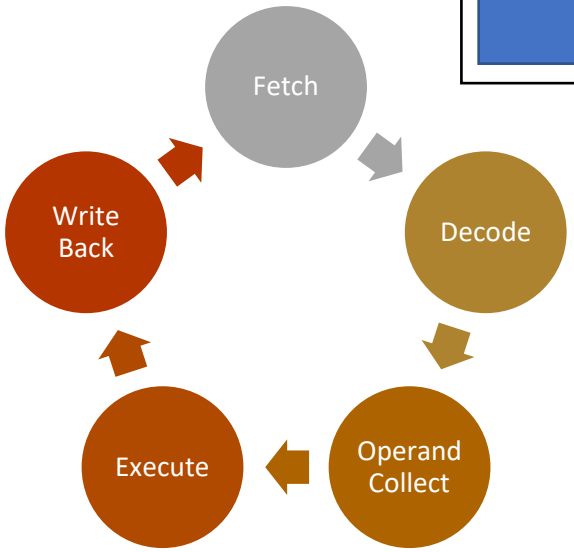
Latency

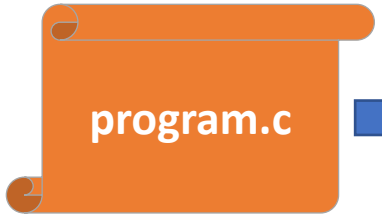
(the time taking to execute an instruction)

High Working Frequency (4GHz)









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

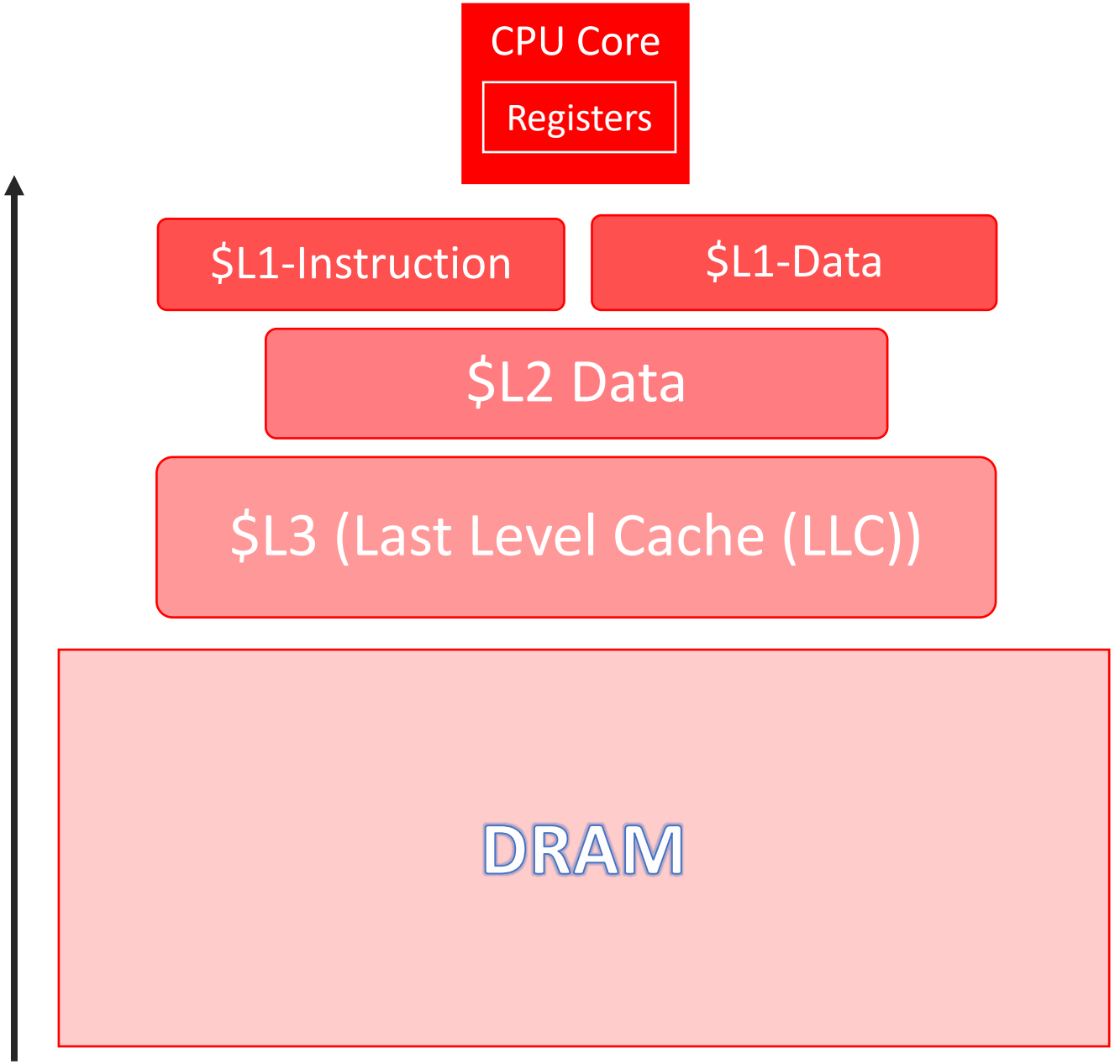
Cache Hierarchy

Less Access latency

More Data Locality

Less Storage Capacity

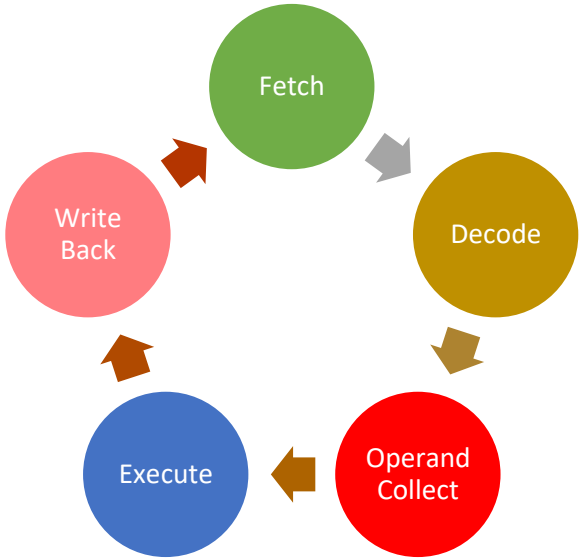
More Expensive per bit

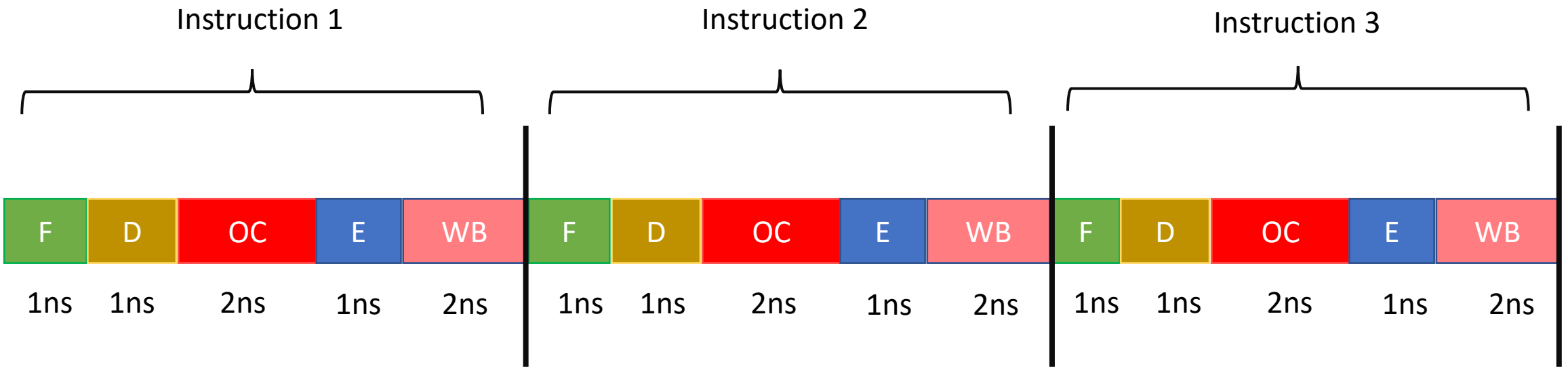


Pipelining

- Basic Processor

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

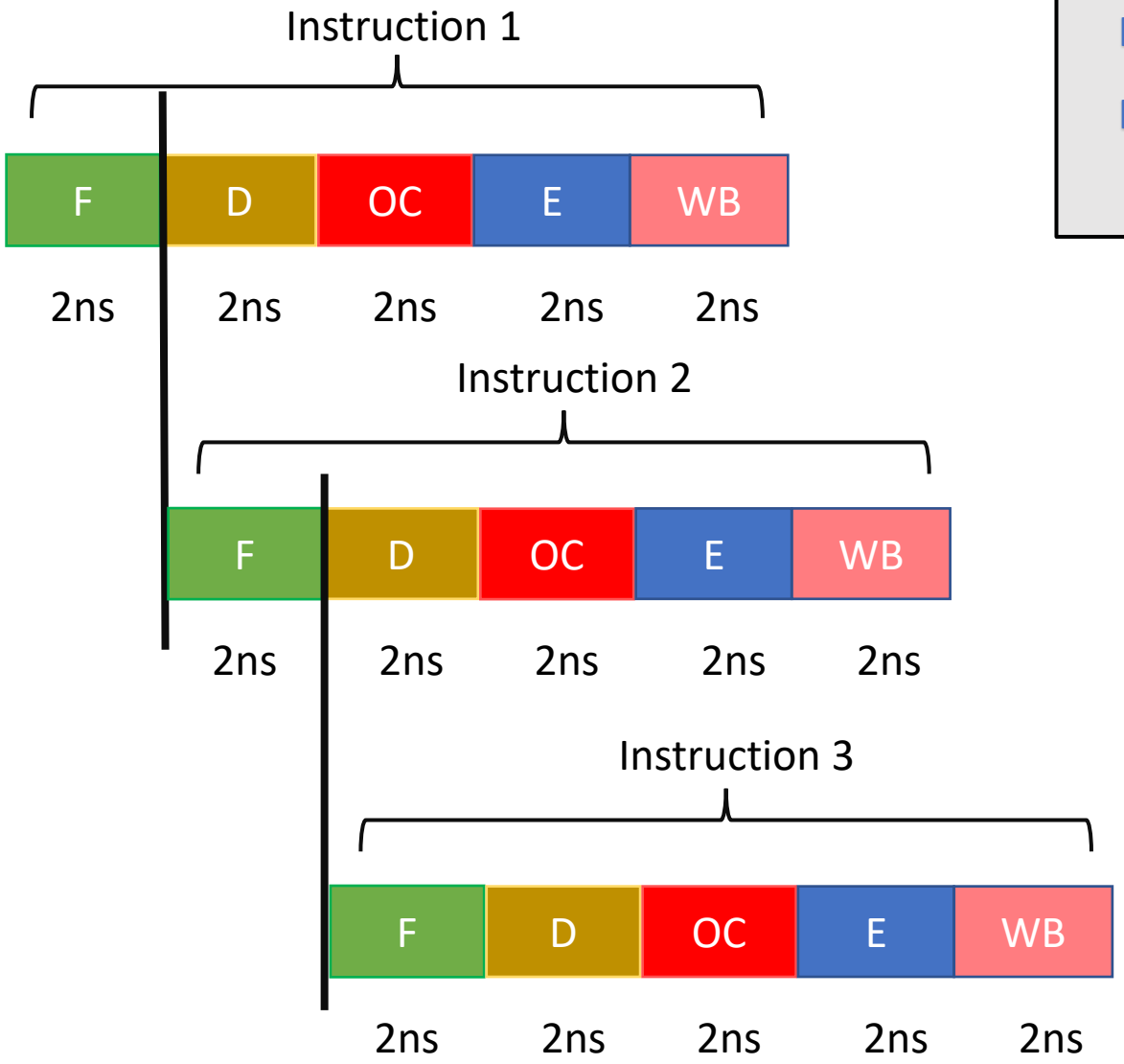




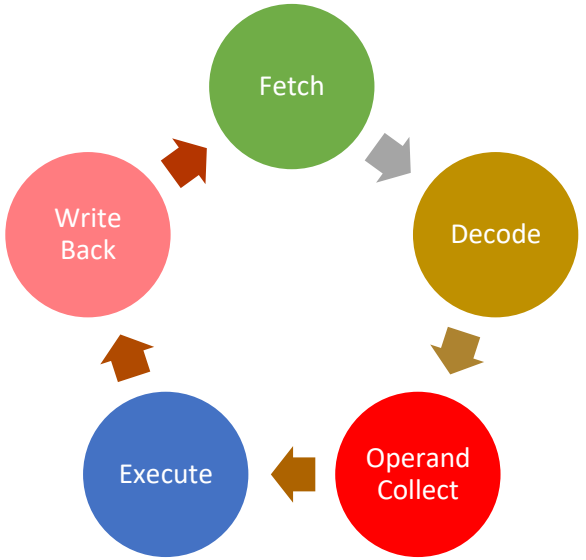
Overall Time of Executing three Instruction: 3 * (1ns + 1ns + 2ns + 1ns + 2ns) = 3 * 7ns = 21ns

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

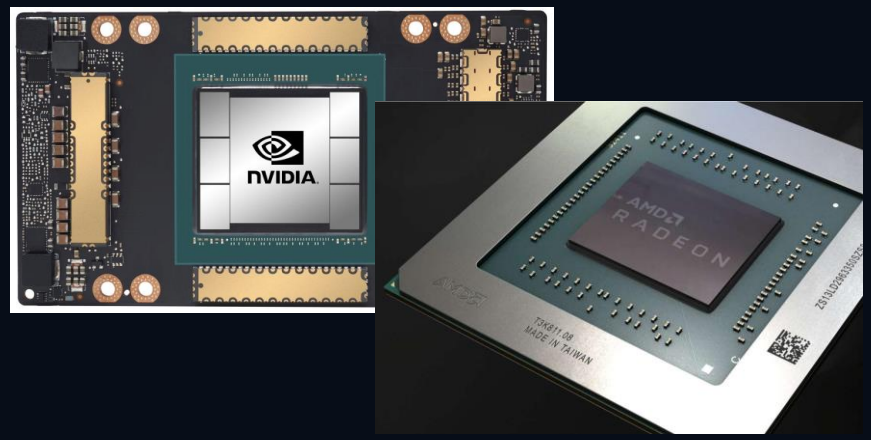
Implicit Parallelism
Instruction-Level Parallelism (ILP)

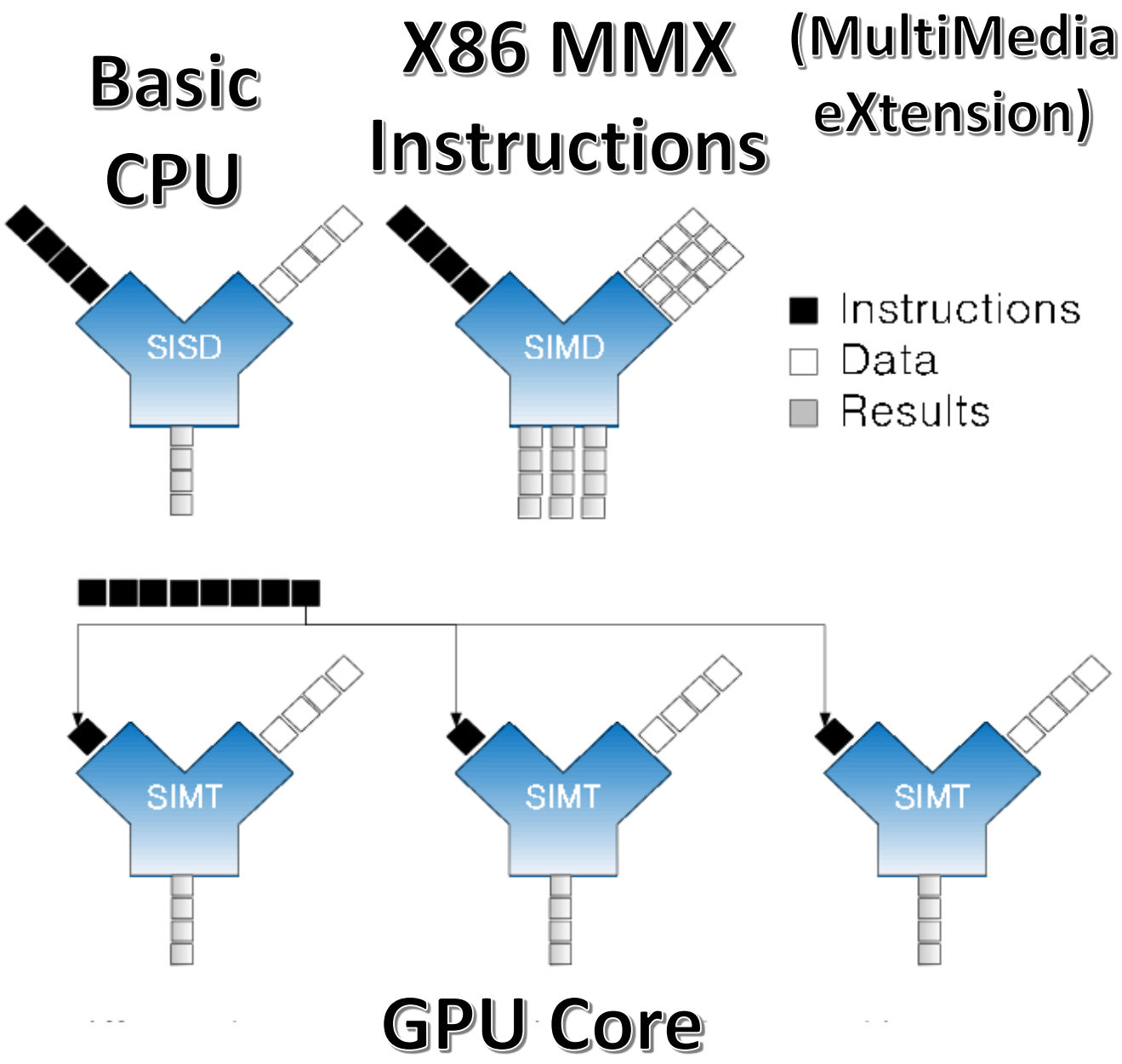
Road Map

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

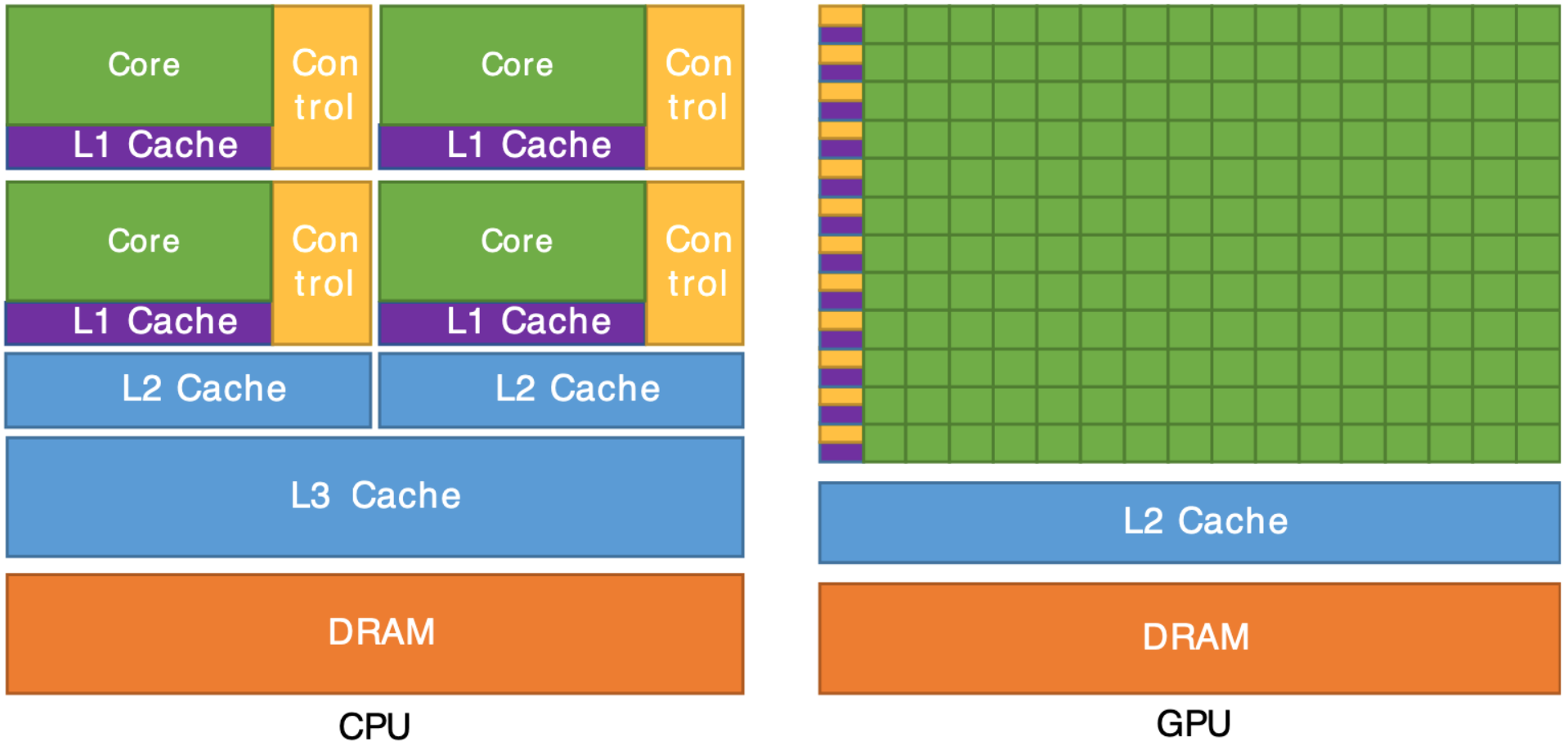
GPU

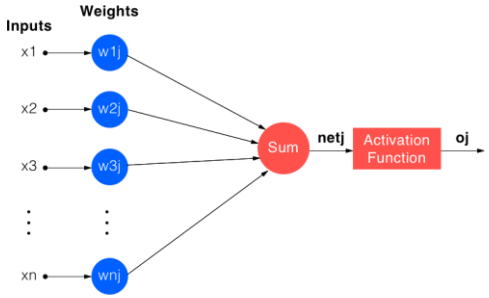
- Parallel Processor
- Throughput
- Low Working Frequency



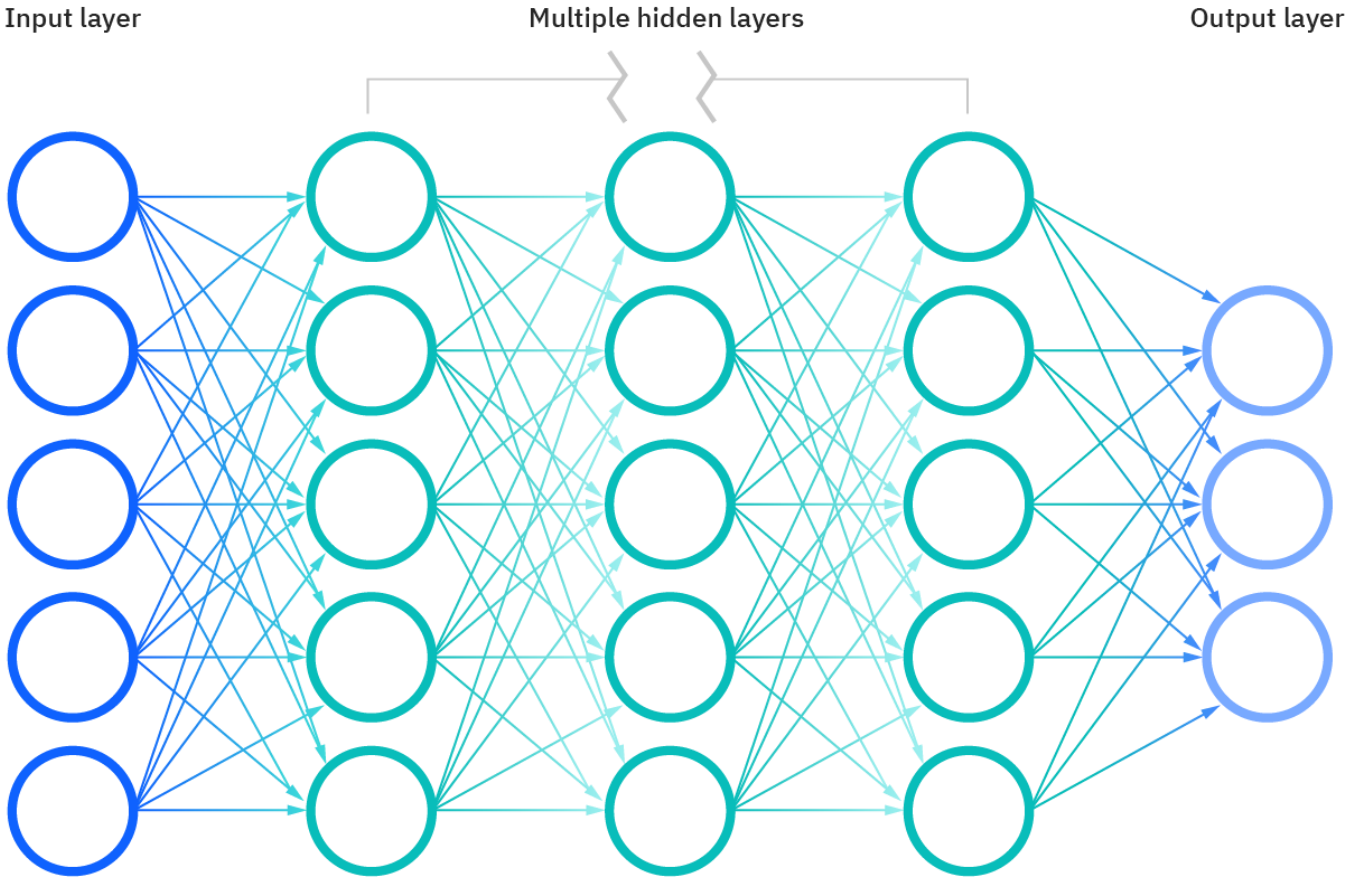


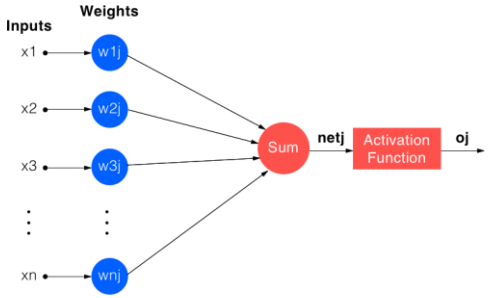
Kyung, Gyutaek et al. "An implementation of a SIMT architecture-based stream processor." *TENCON 2014 - 2014 IEEE Region 10 Conference* (2014): 1-5.


Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
L2 Cache
L2 Cache
L3 Cache
DRAM
CPU
L2 Cache
DRAM
GPU

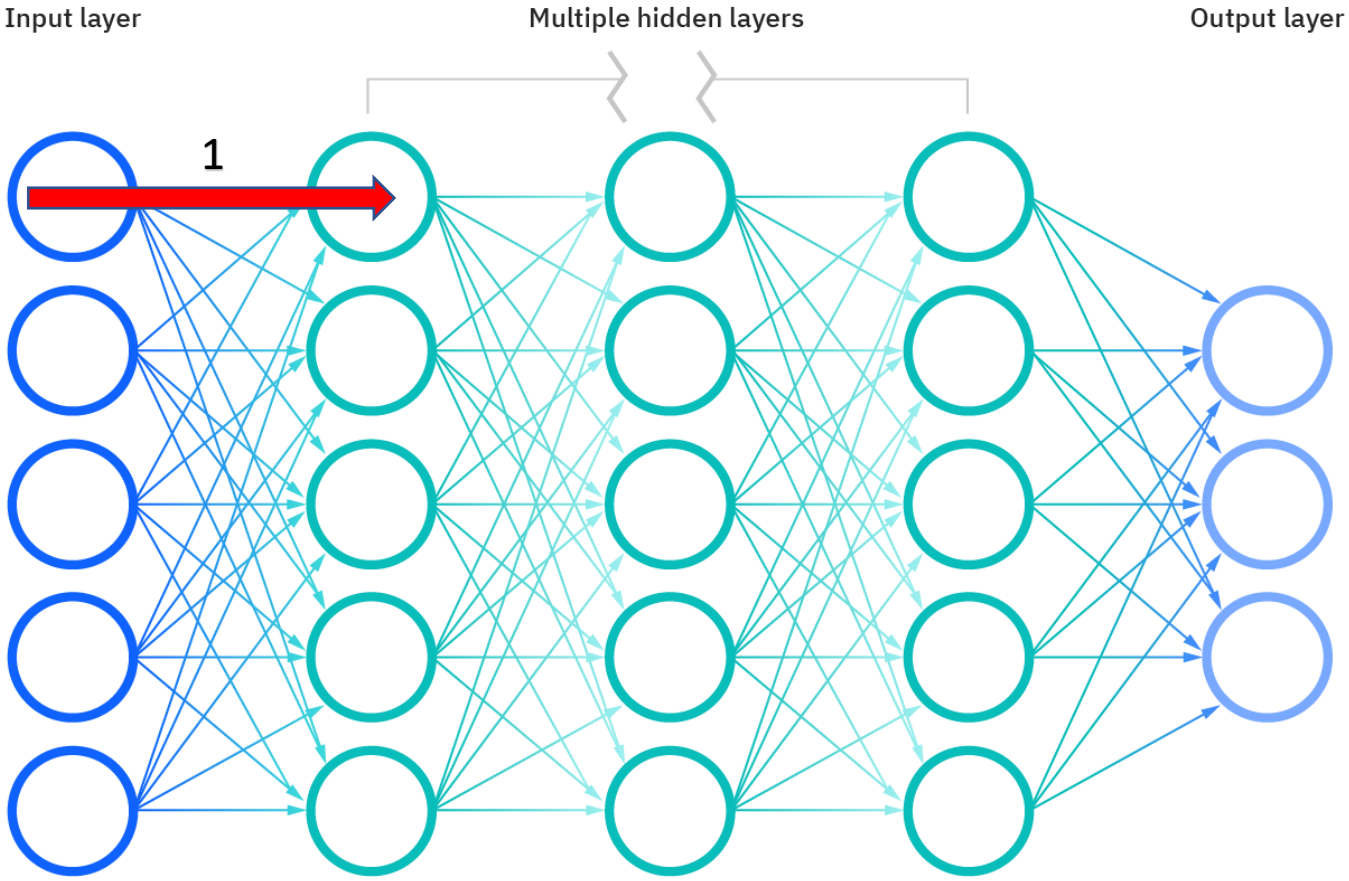


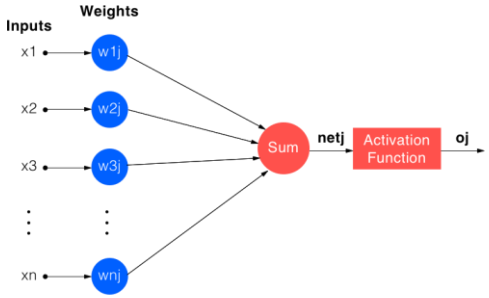
Deep neural network



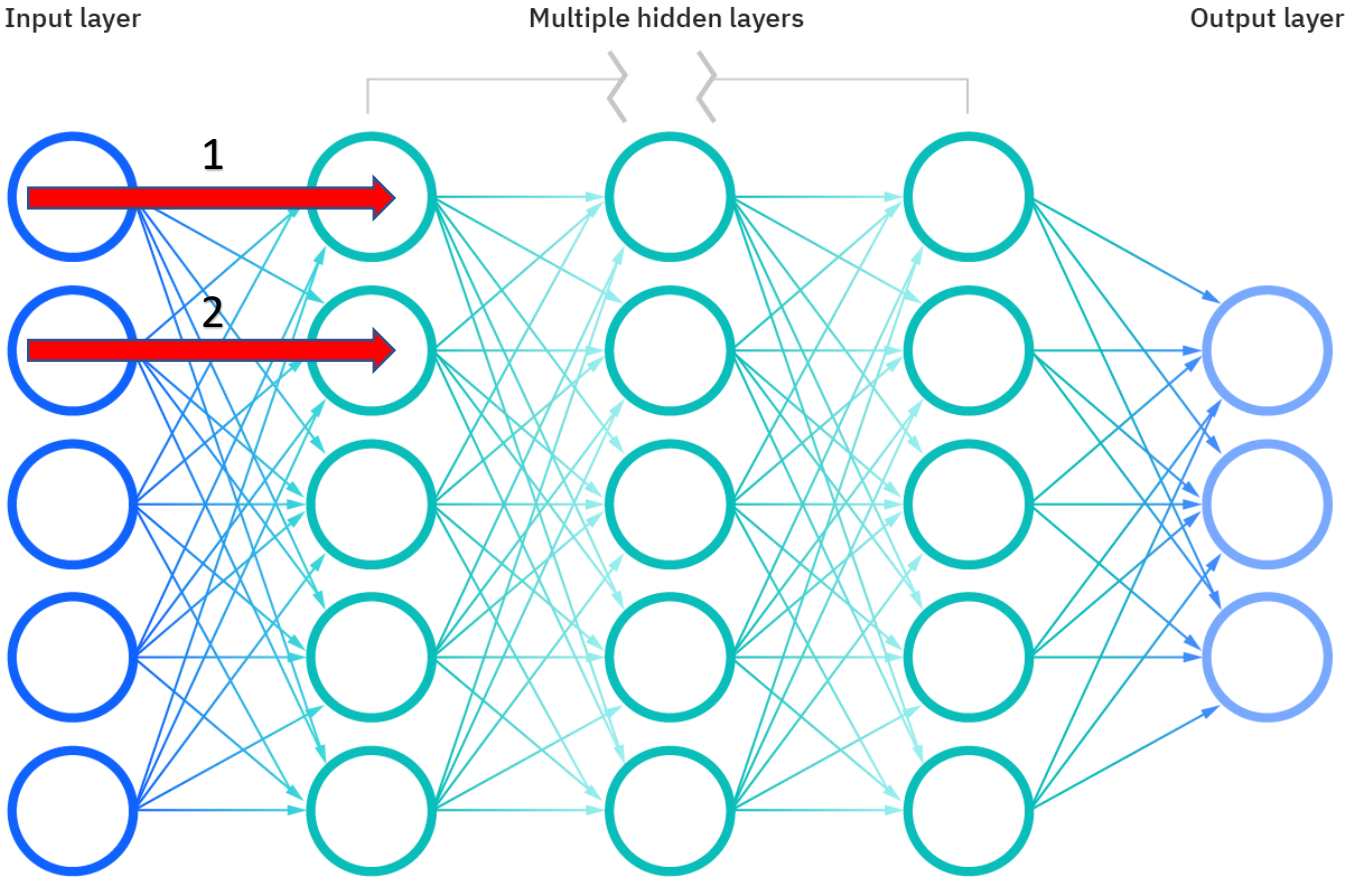


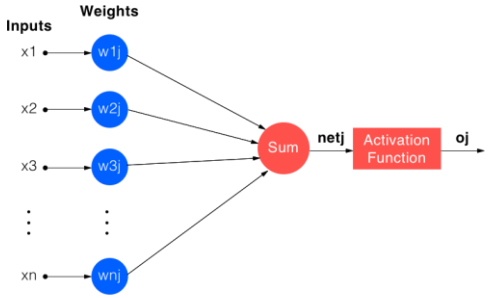
Deep neural network



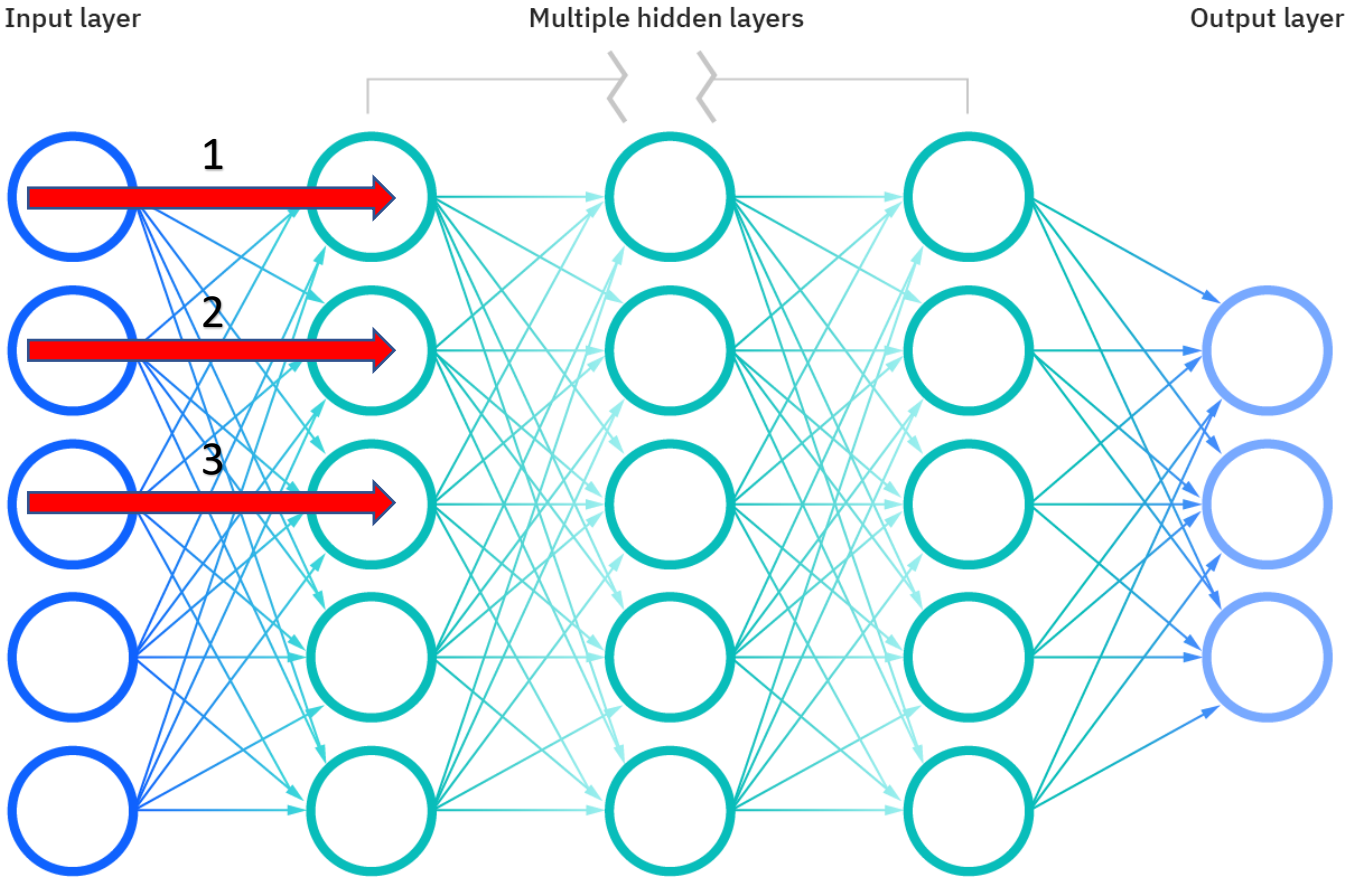


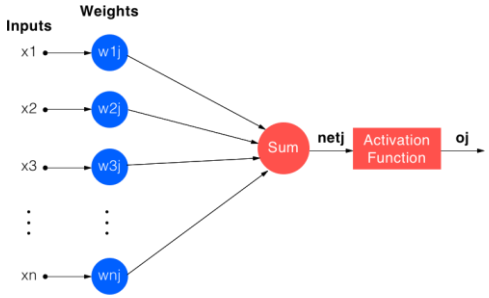
Deep neural network



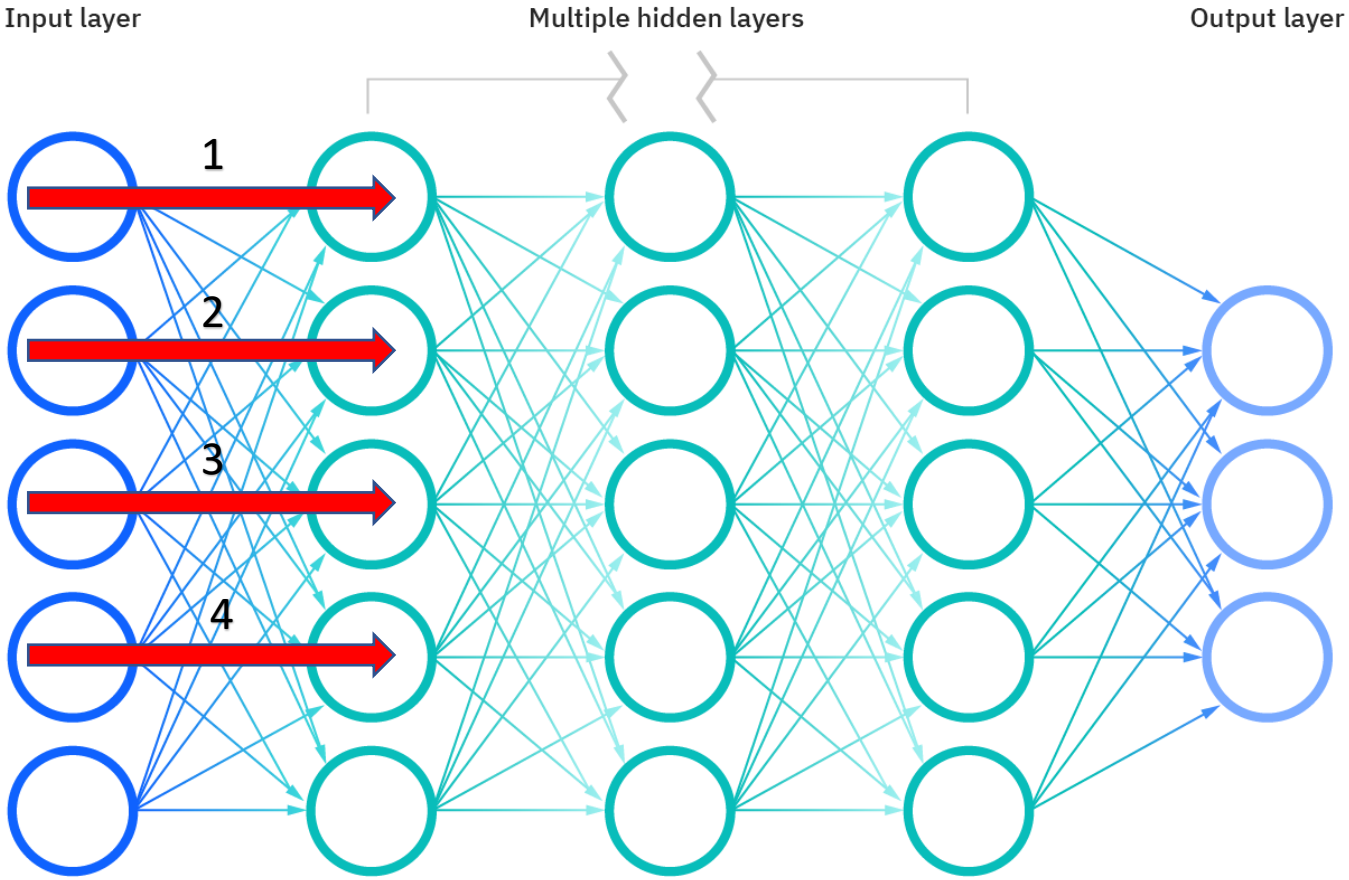


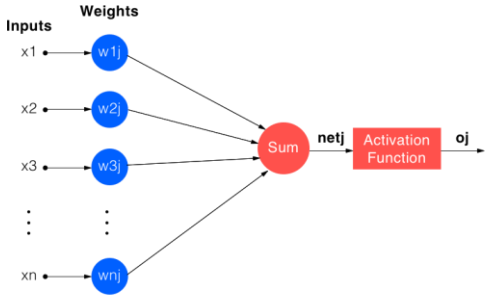
Deep neural network



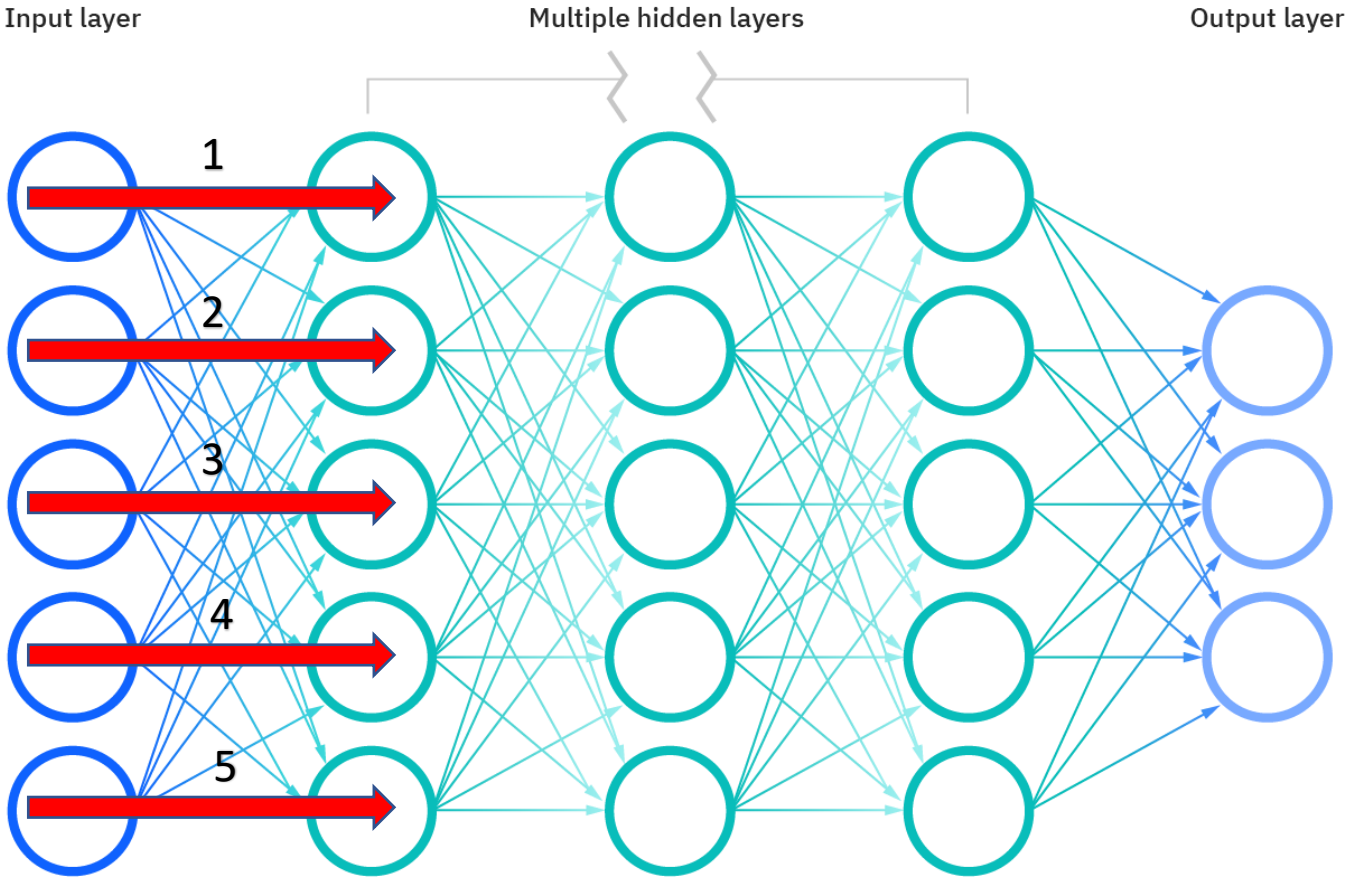


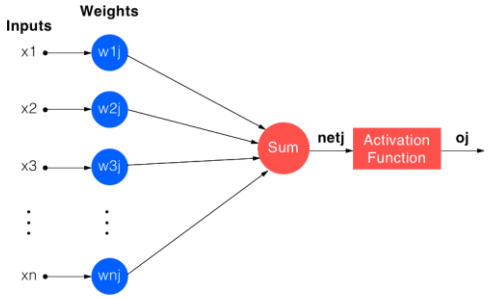
Deep neural network



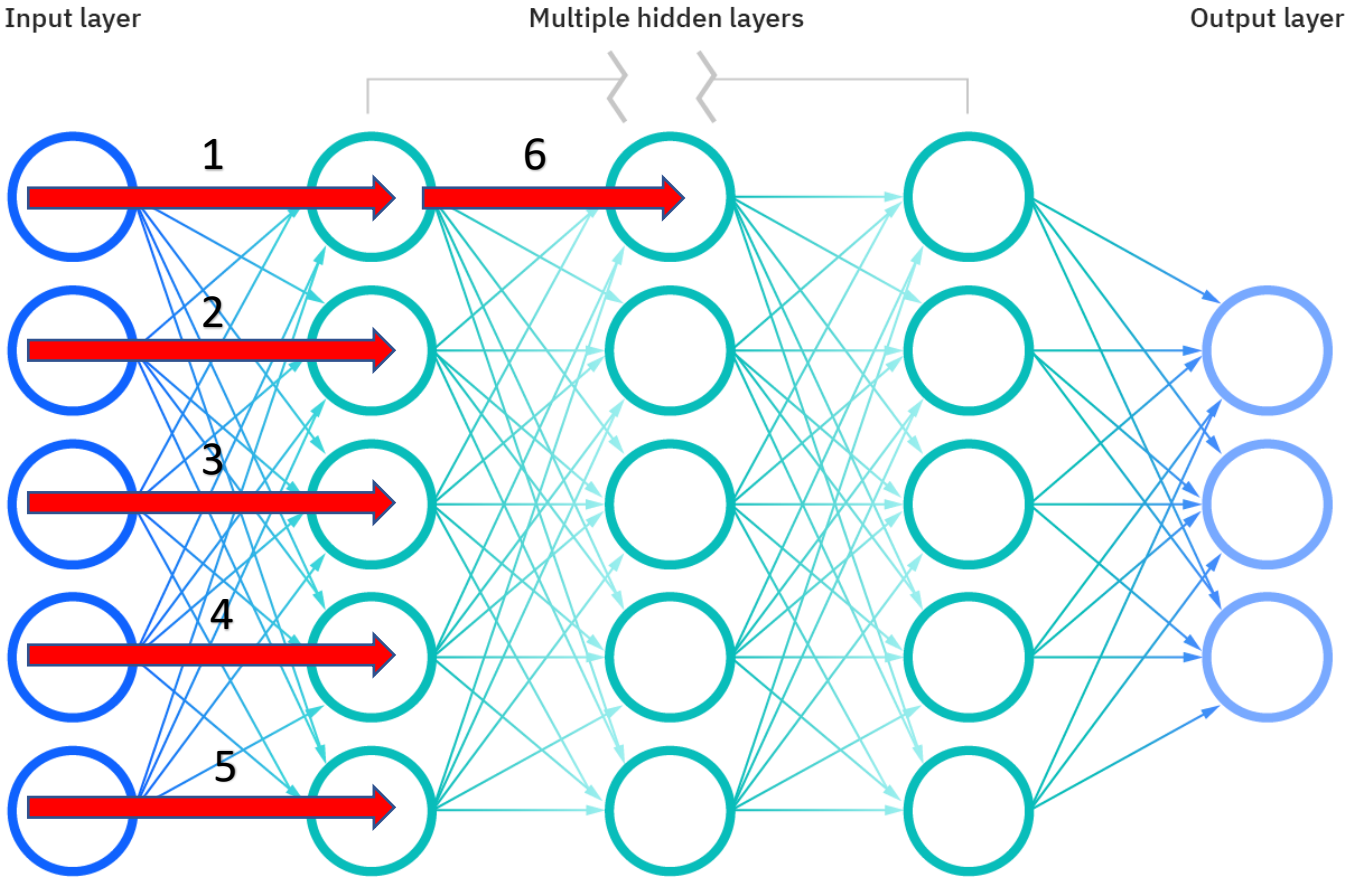


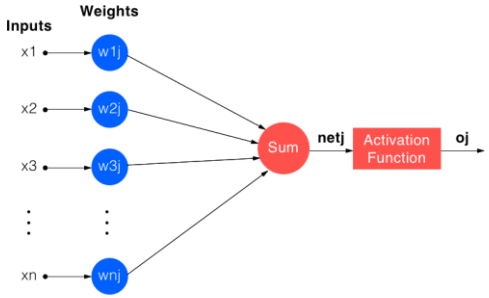
Deep neural network



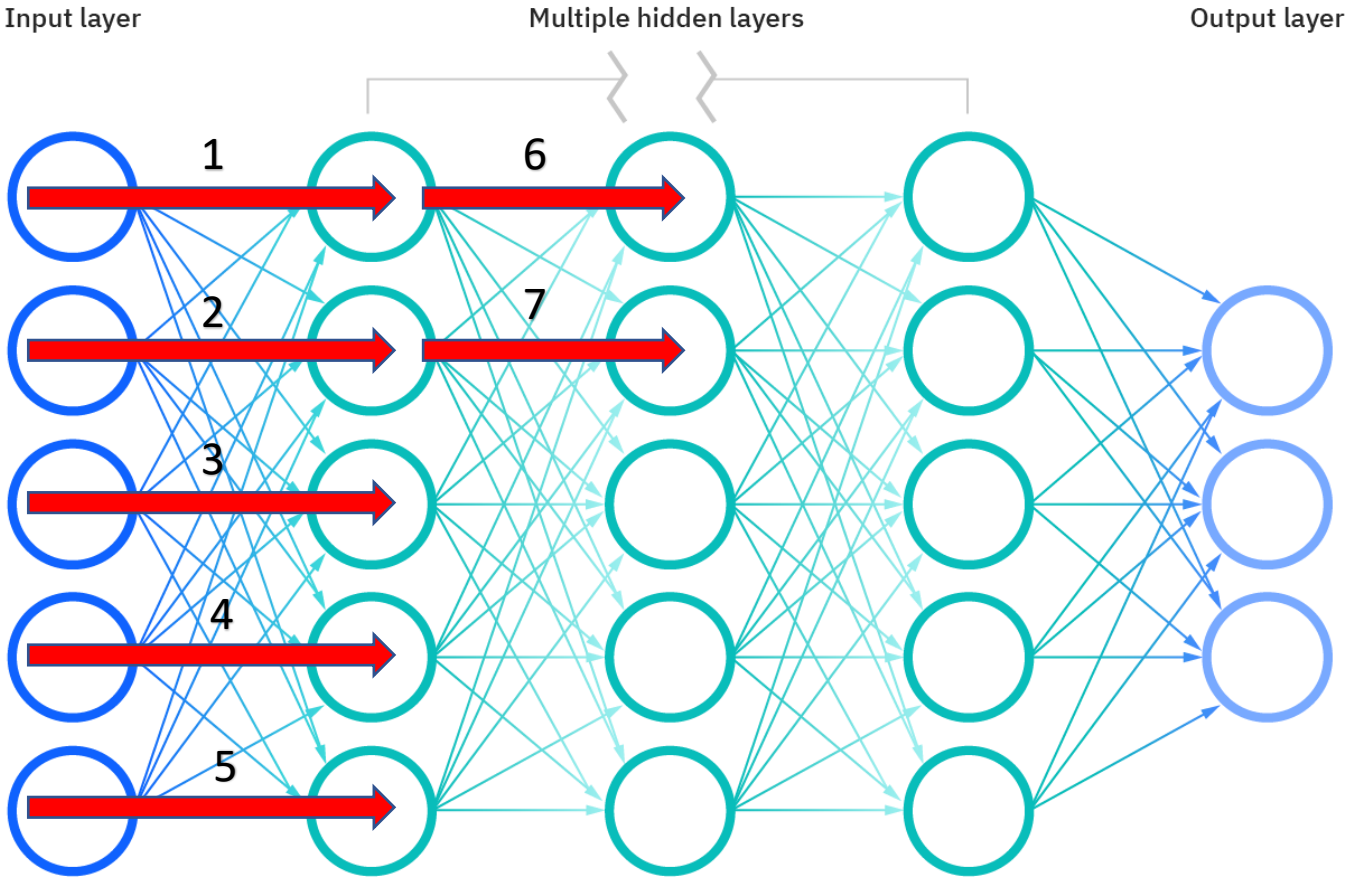


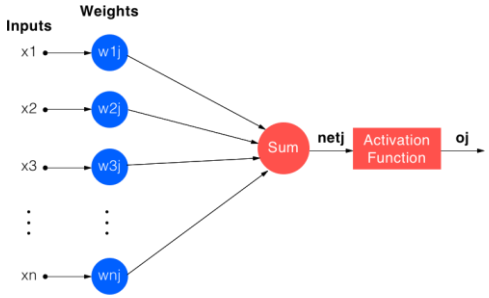
Deep neural network



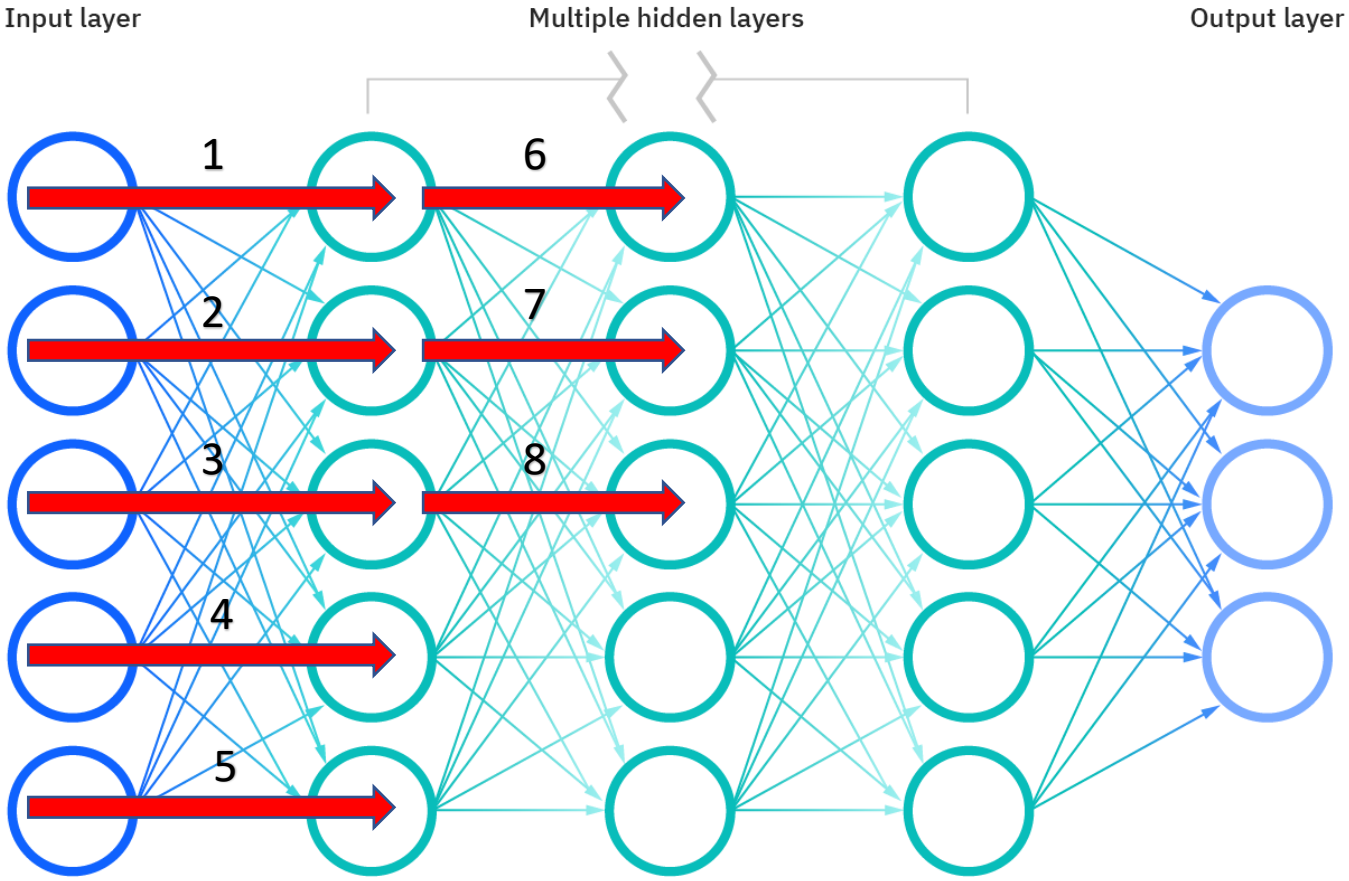


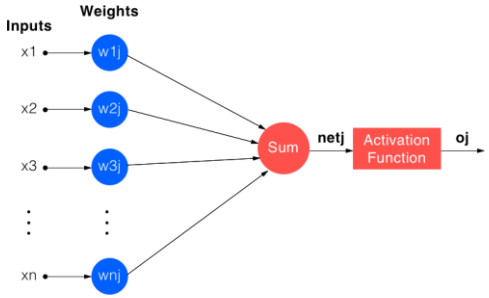
Deep neural network



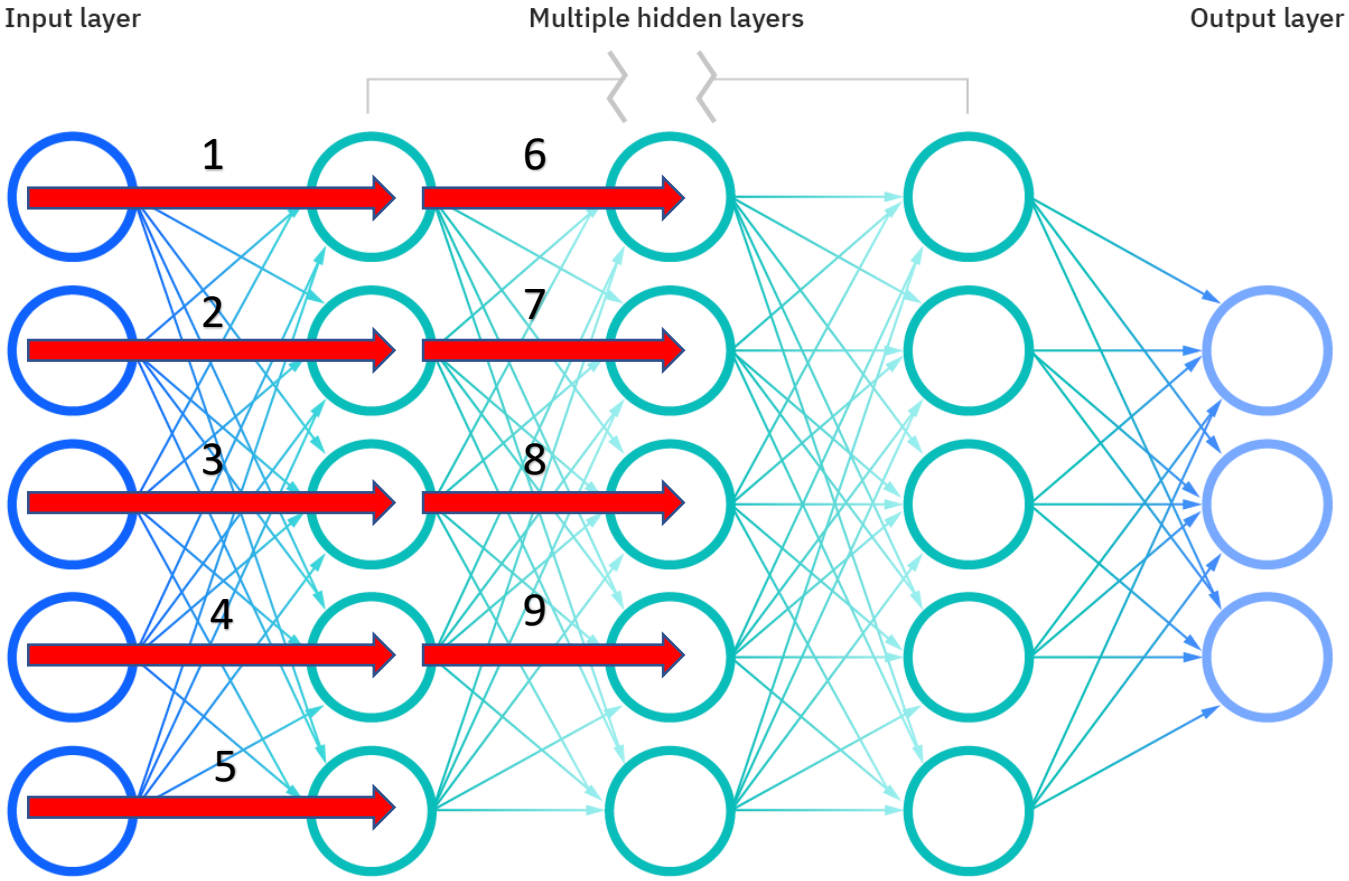


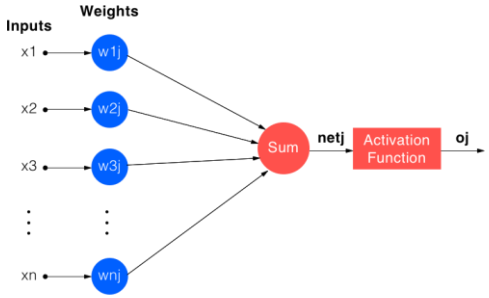
Deep neural network



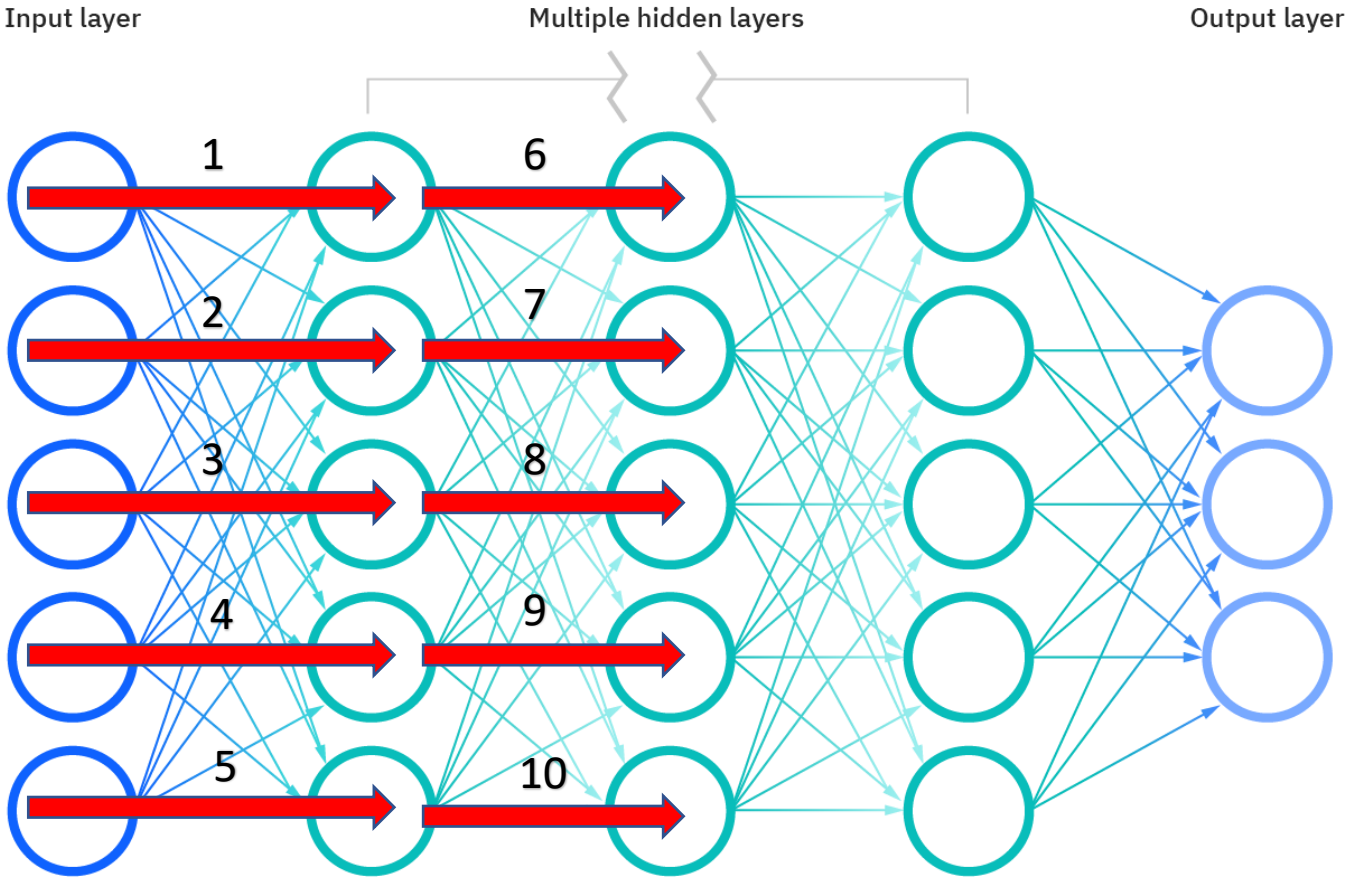


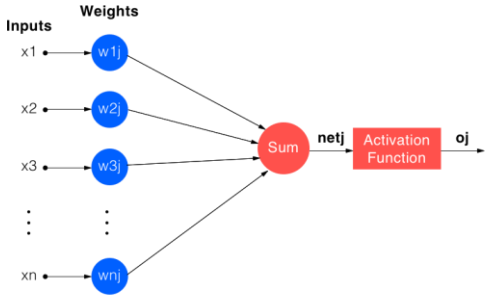
Deep neural network



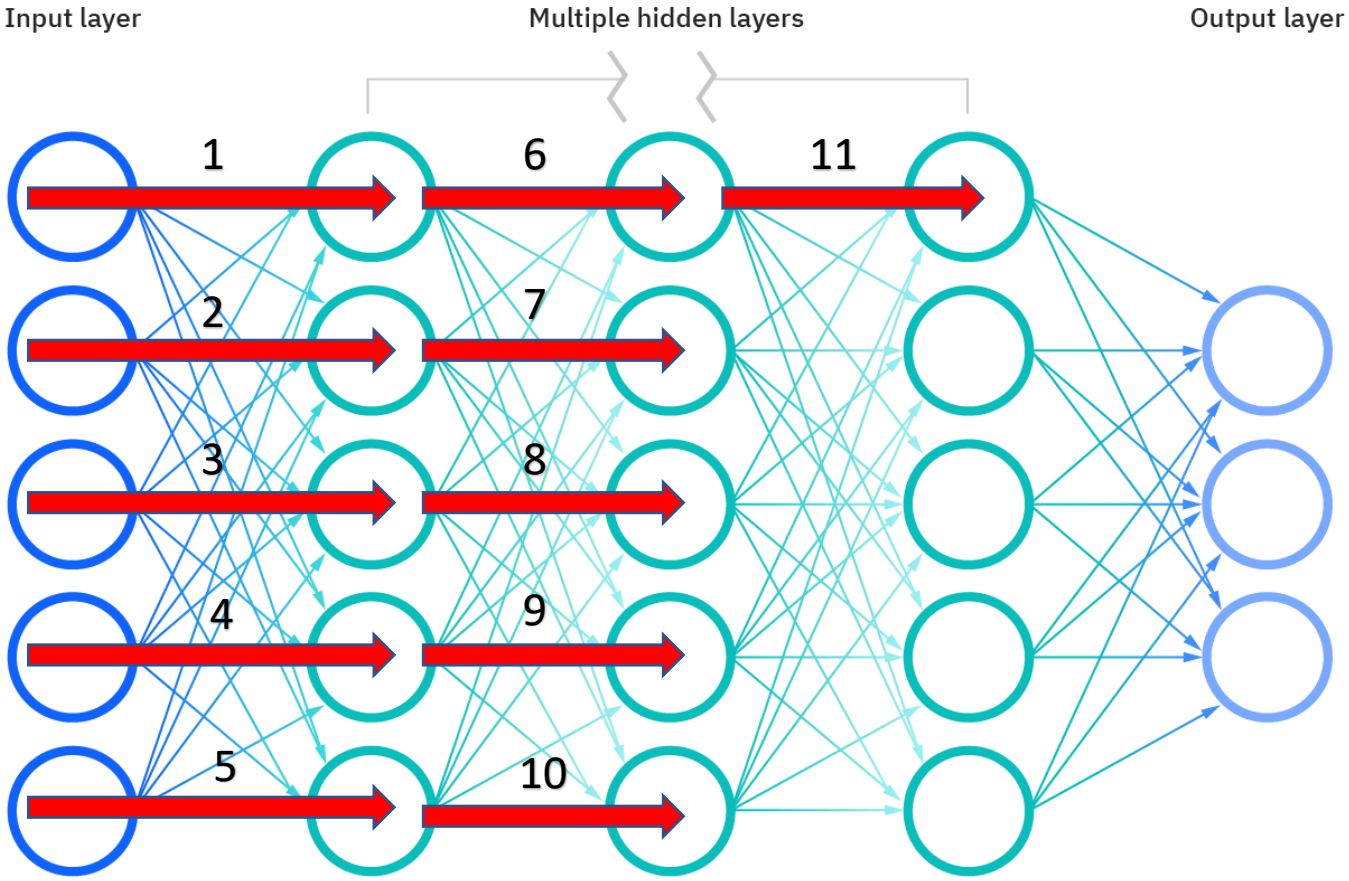


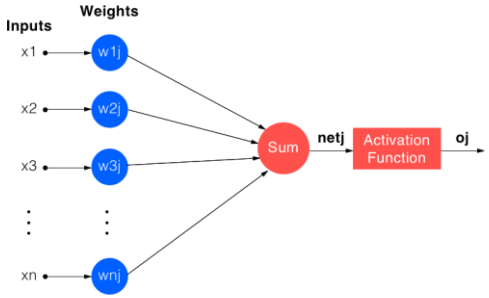
Deep neural network



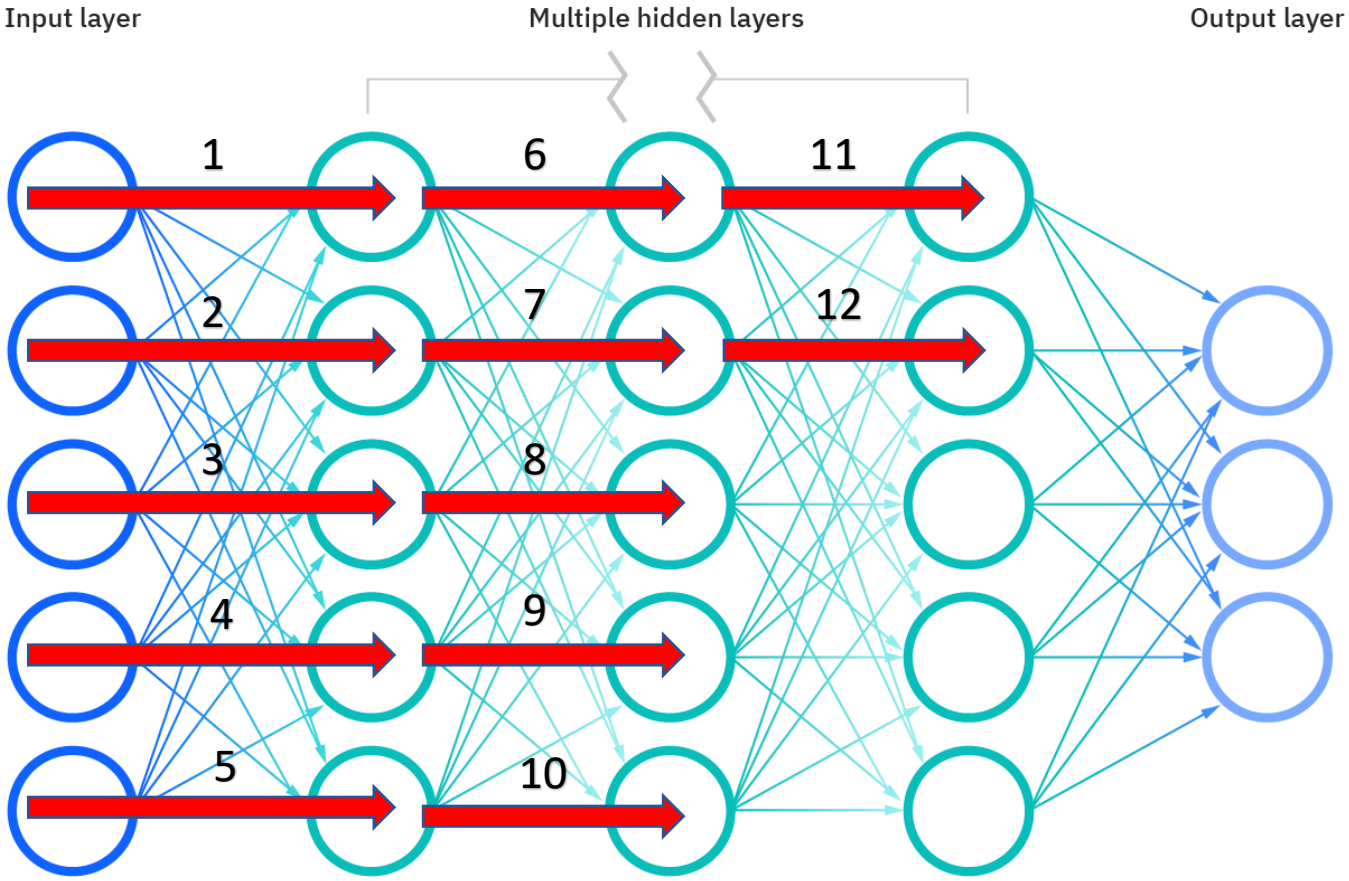


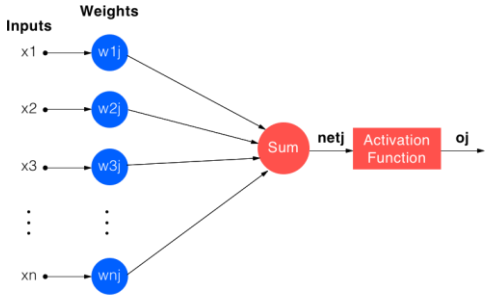
Deep neural network



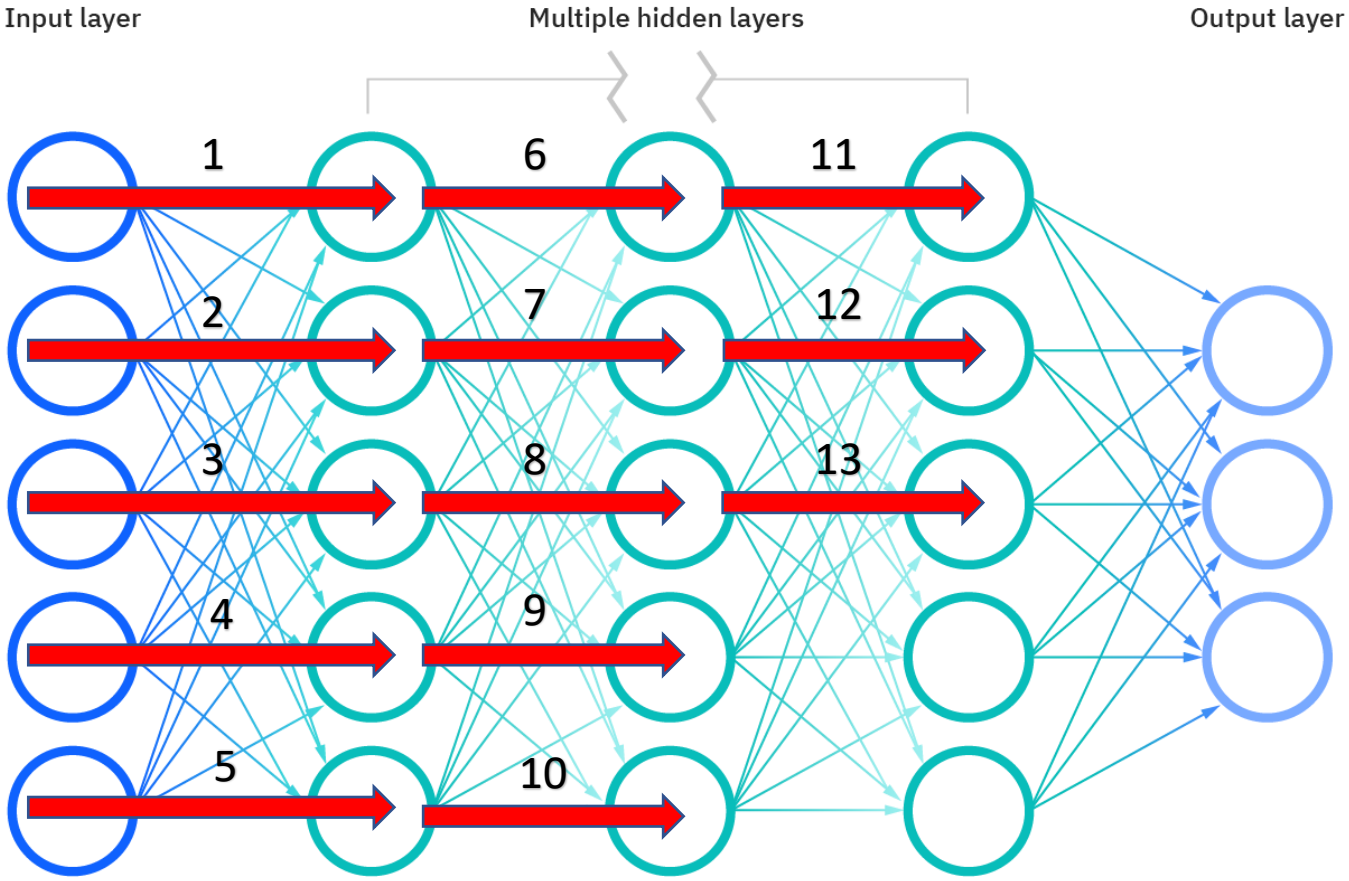


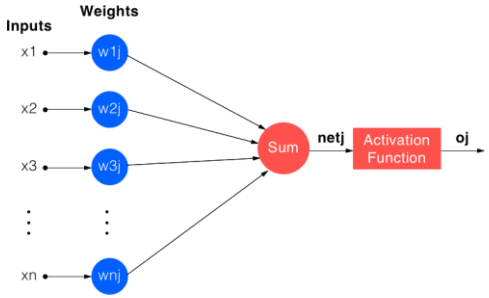
Deep neural network



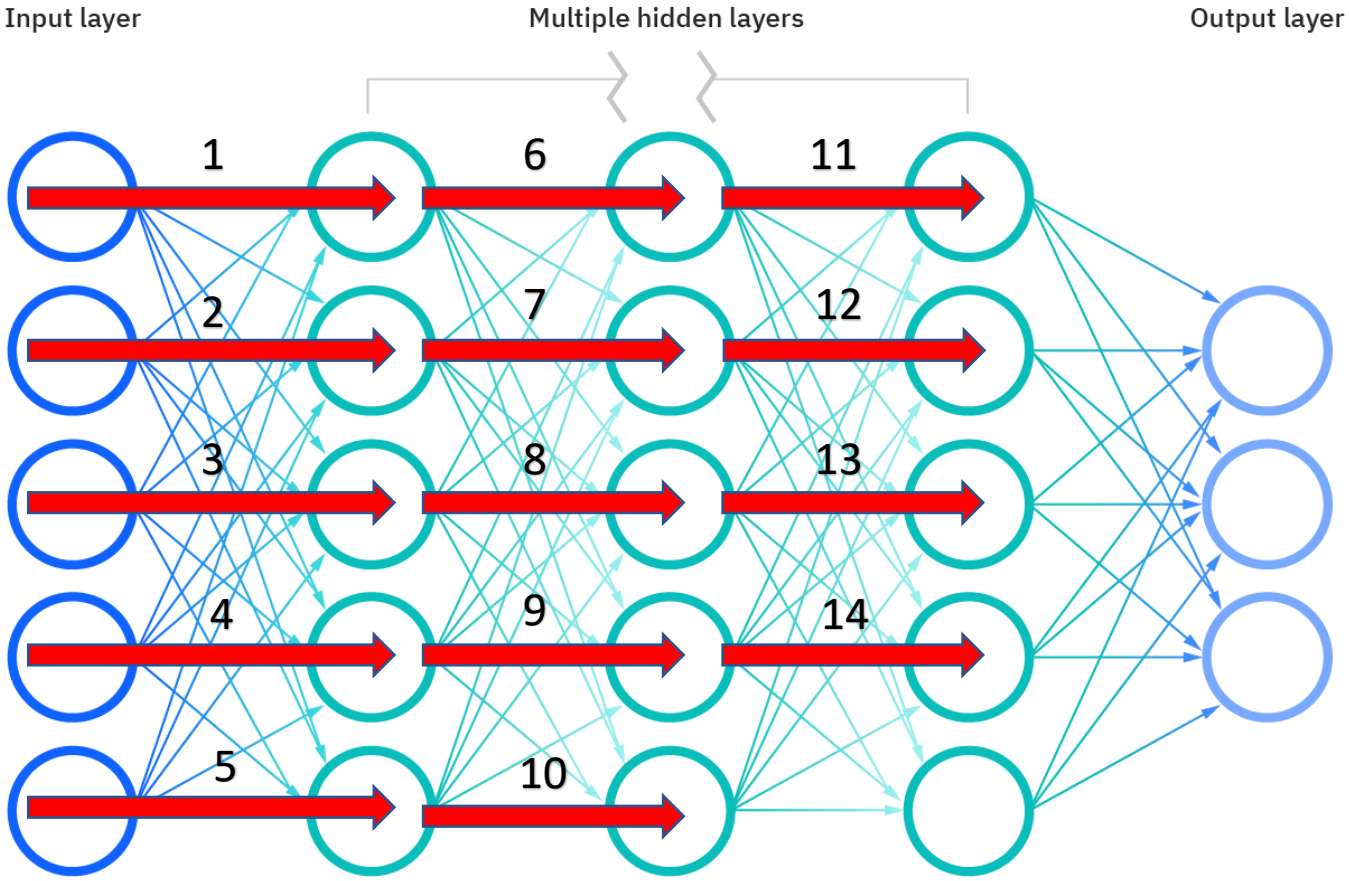


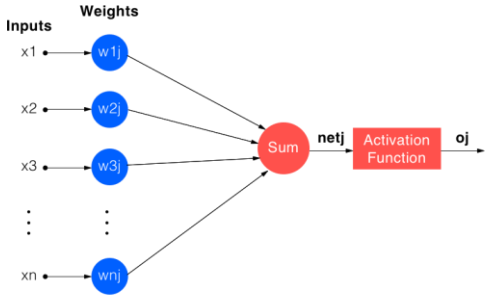
Deep neural network



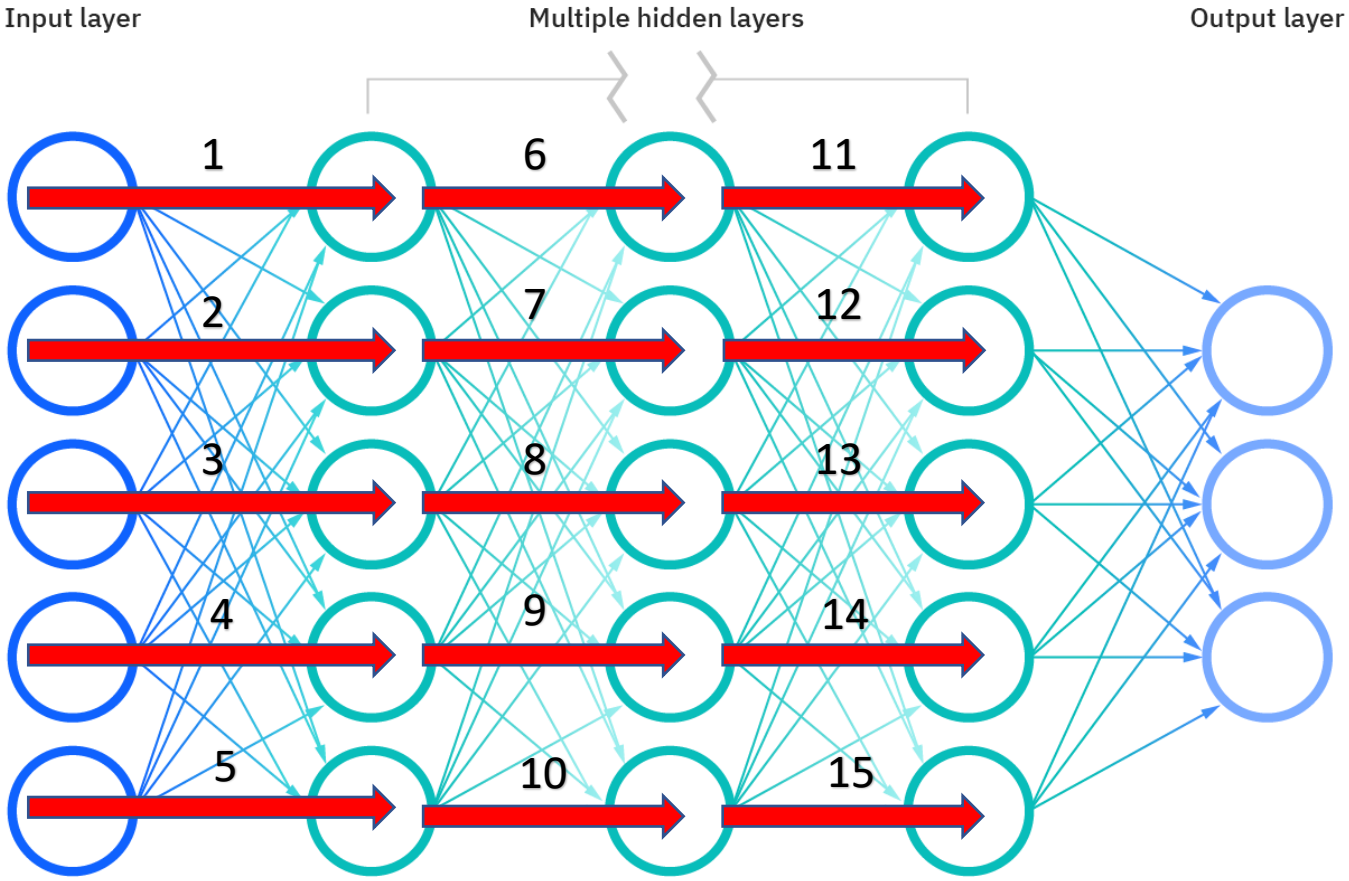


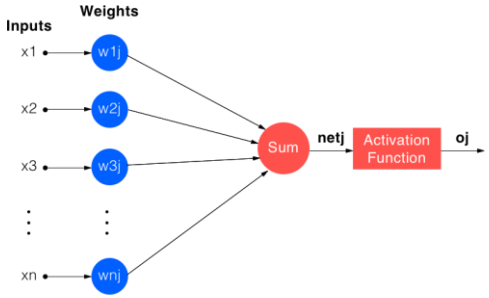
Deep neural network



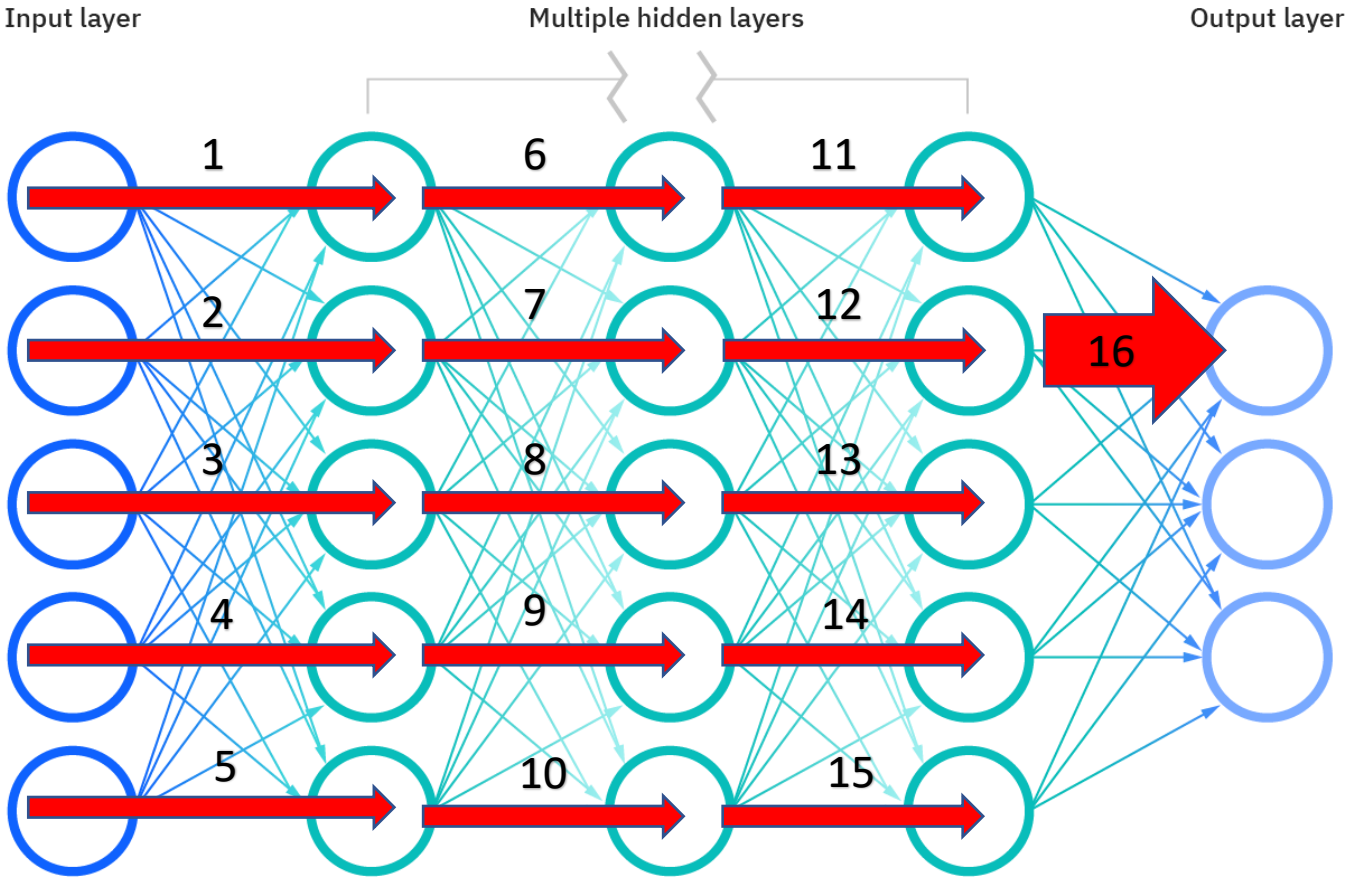


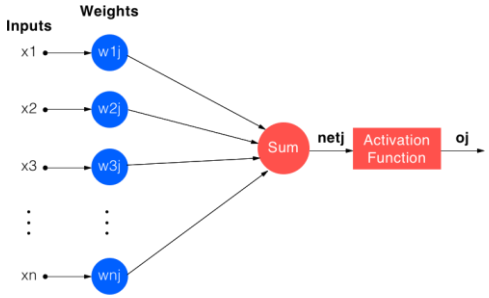
Deep neural network



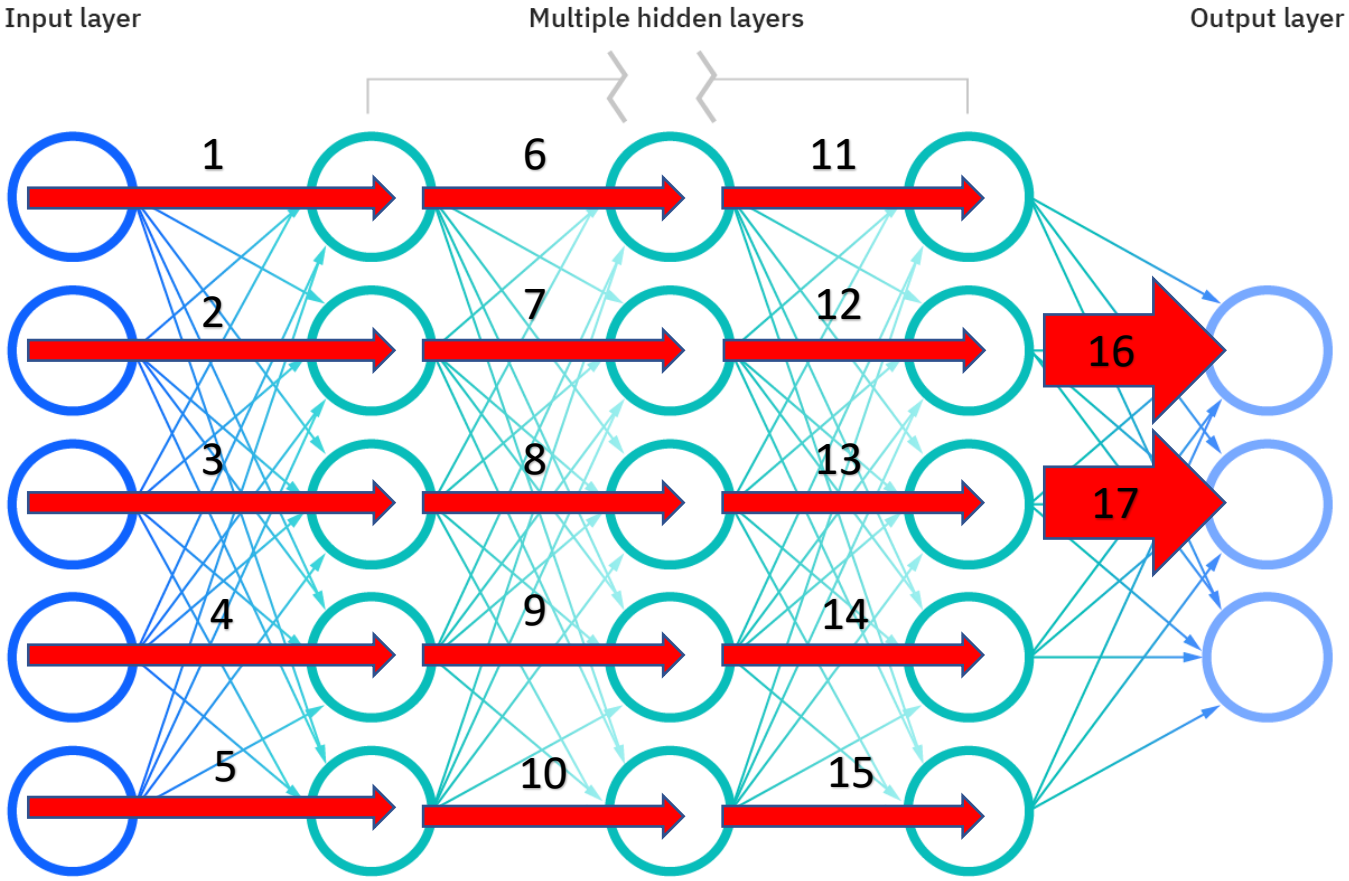


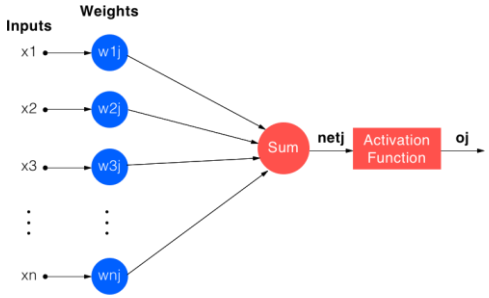
Deep neural network



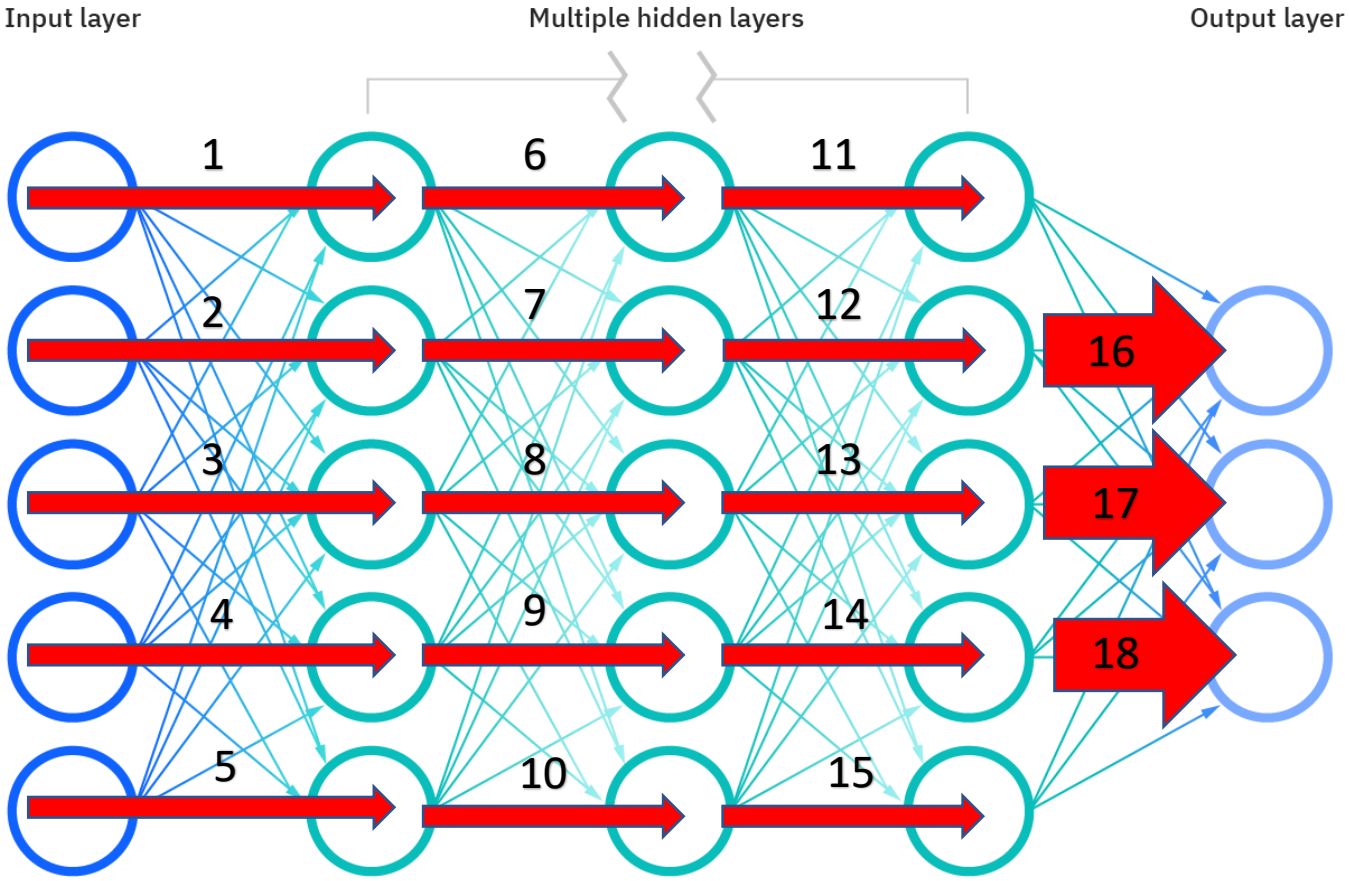


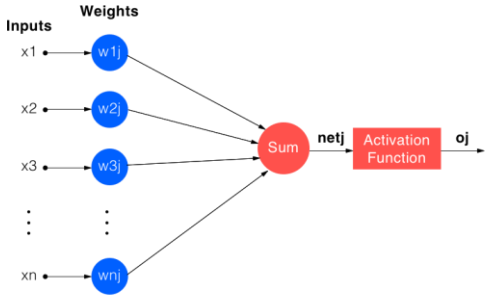
Deep neural network



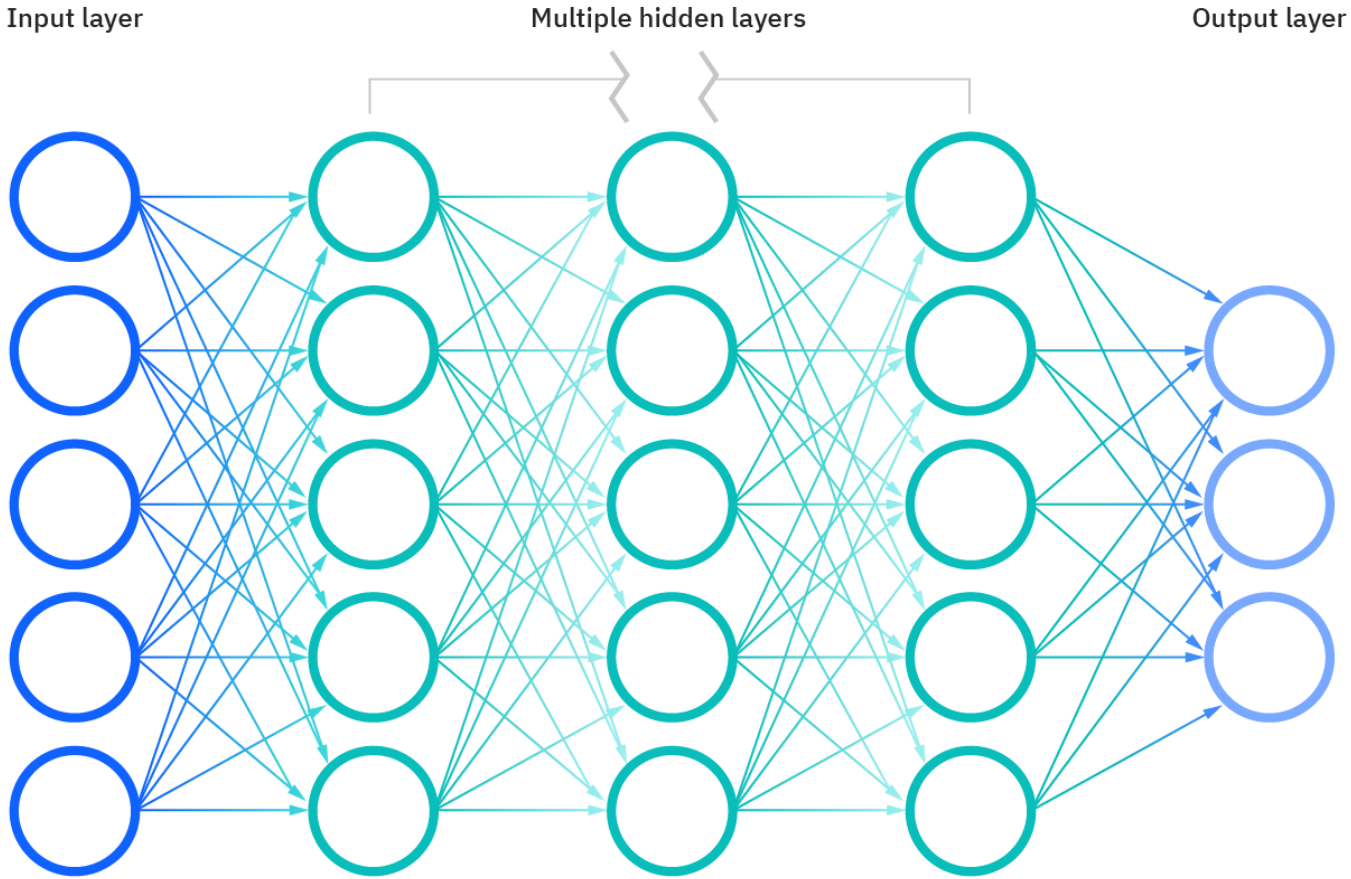


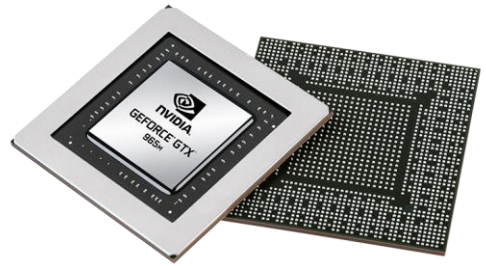
Deep neural network

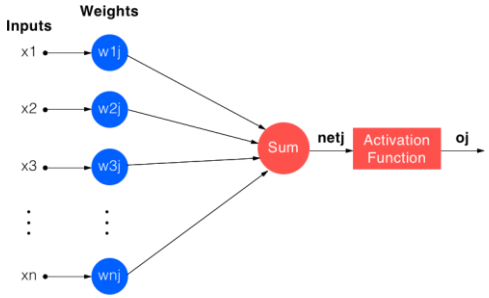




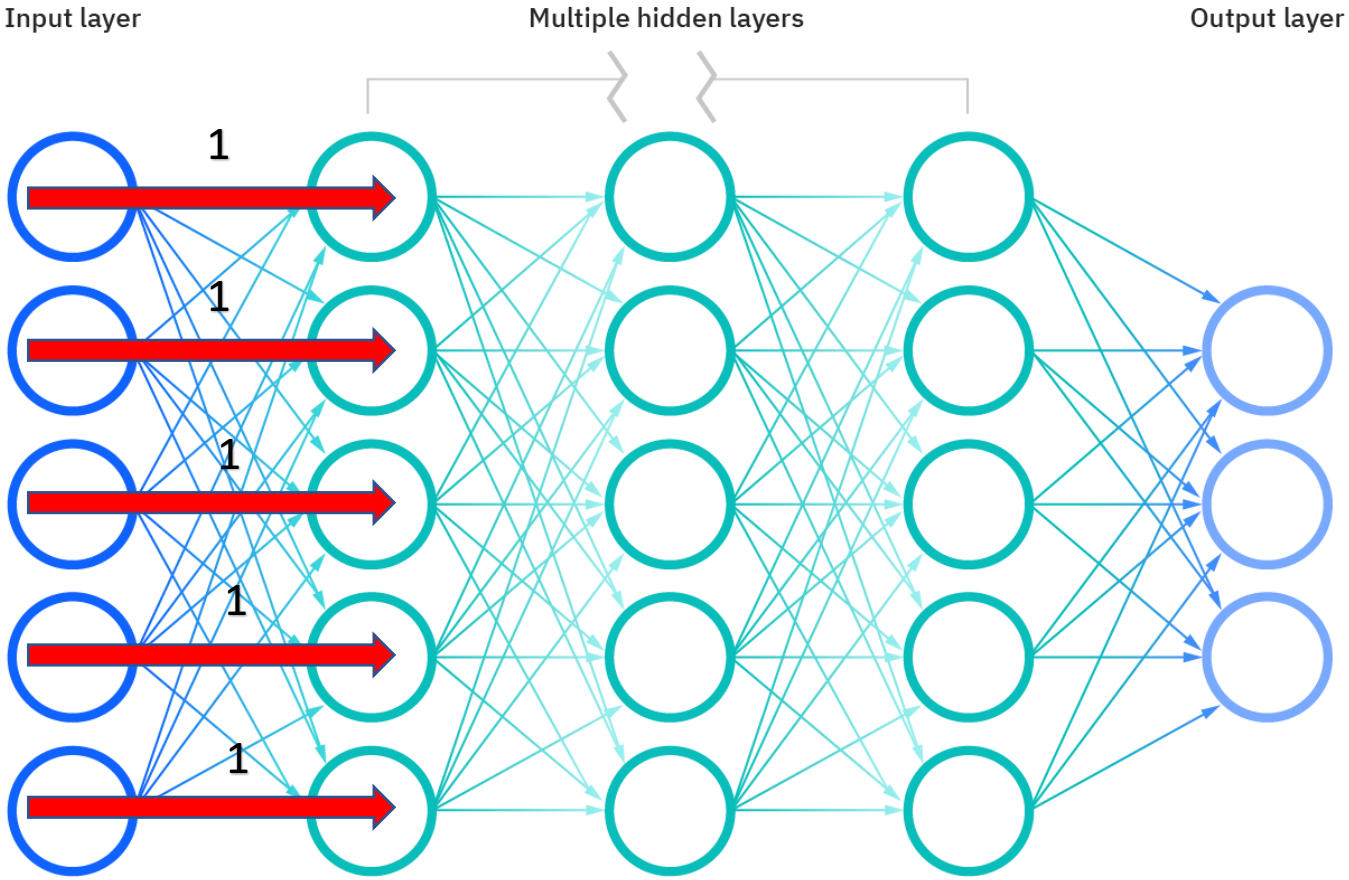
Deep neural network

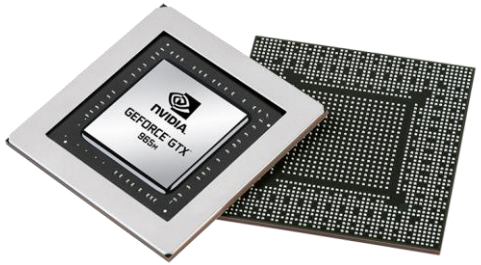


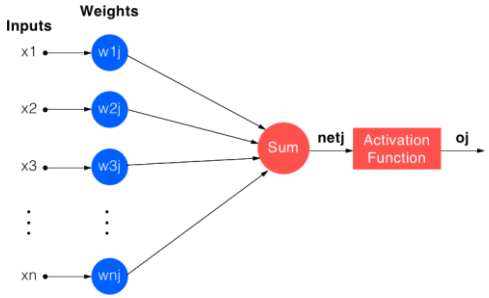




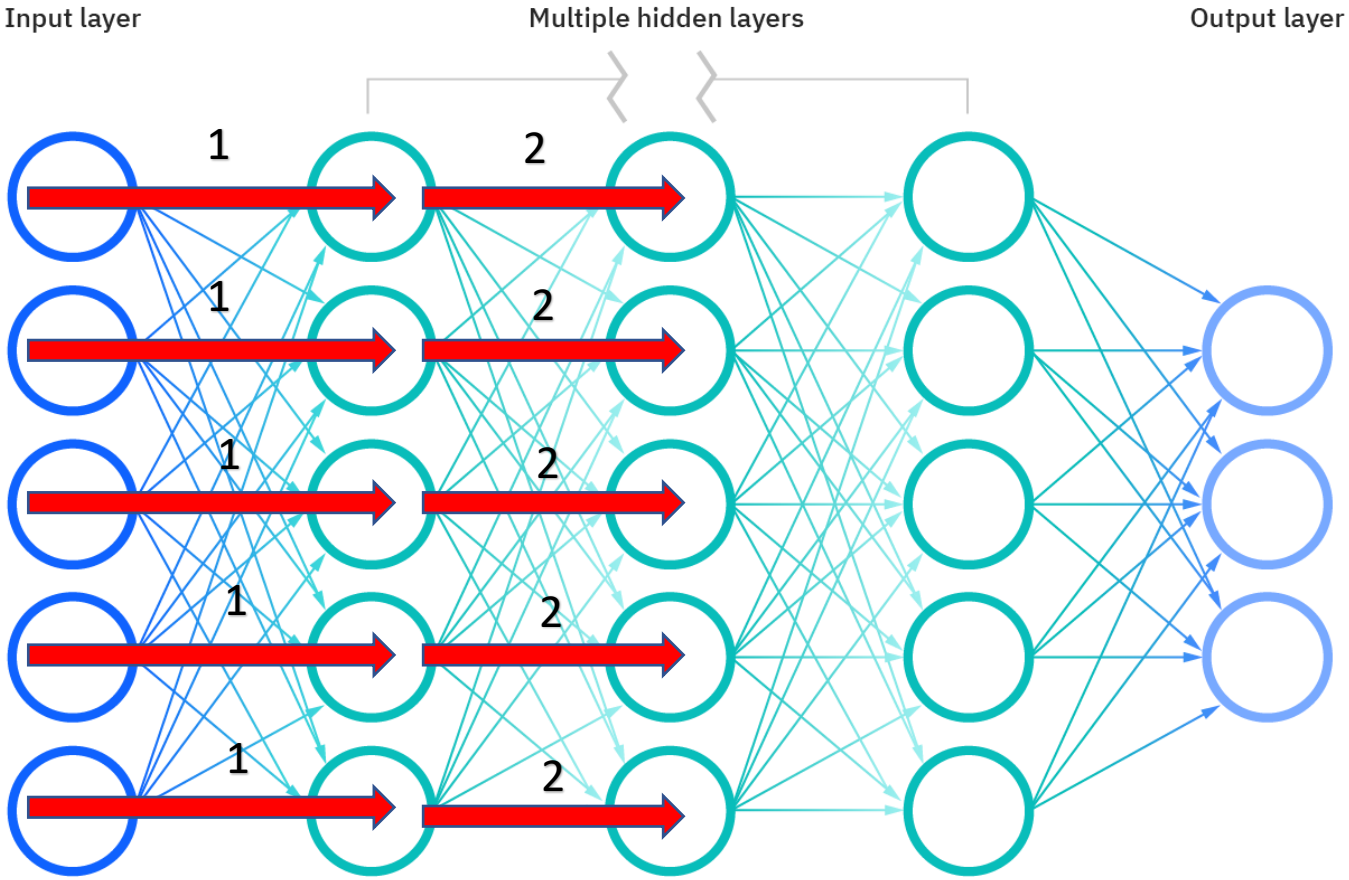
Deep neural network

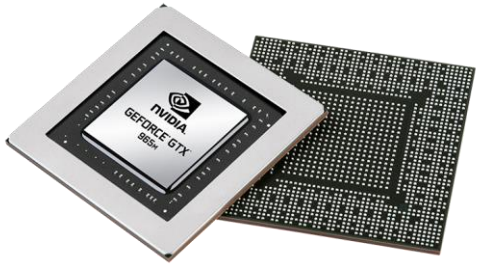


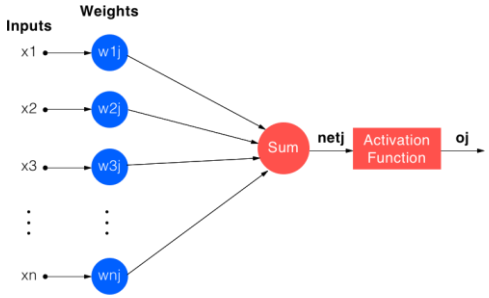




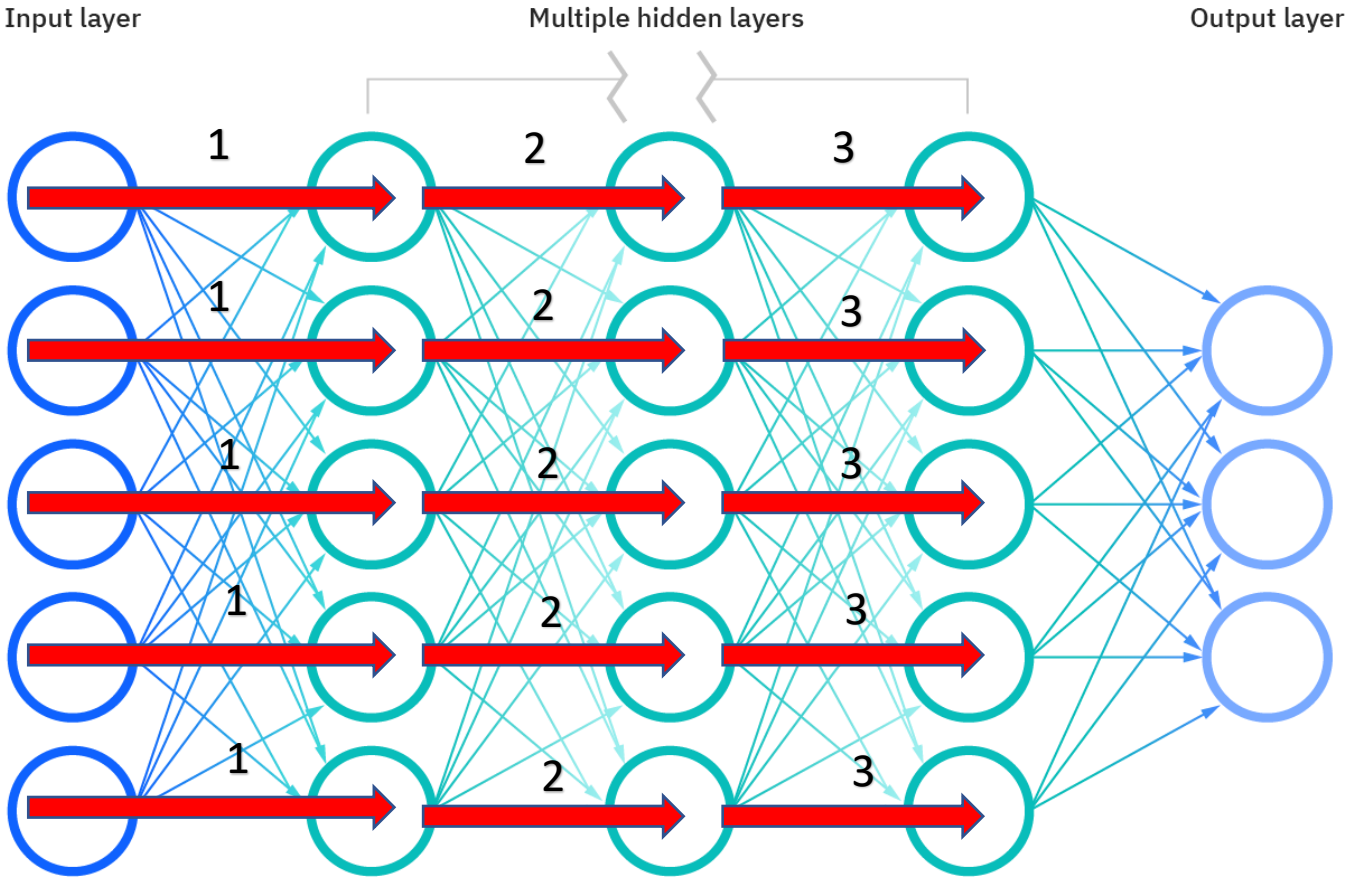
Deep neural network

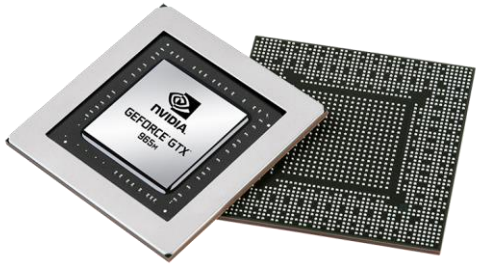


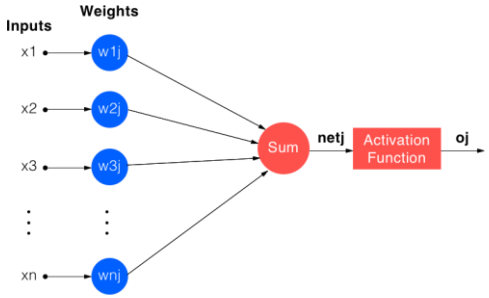




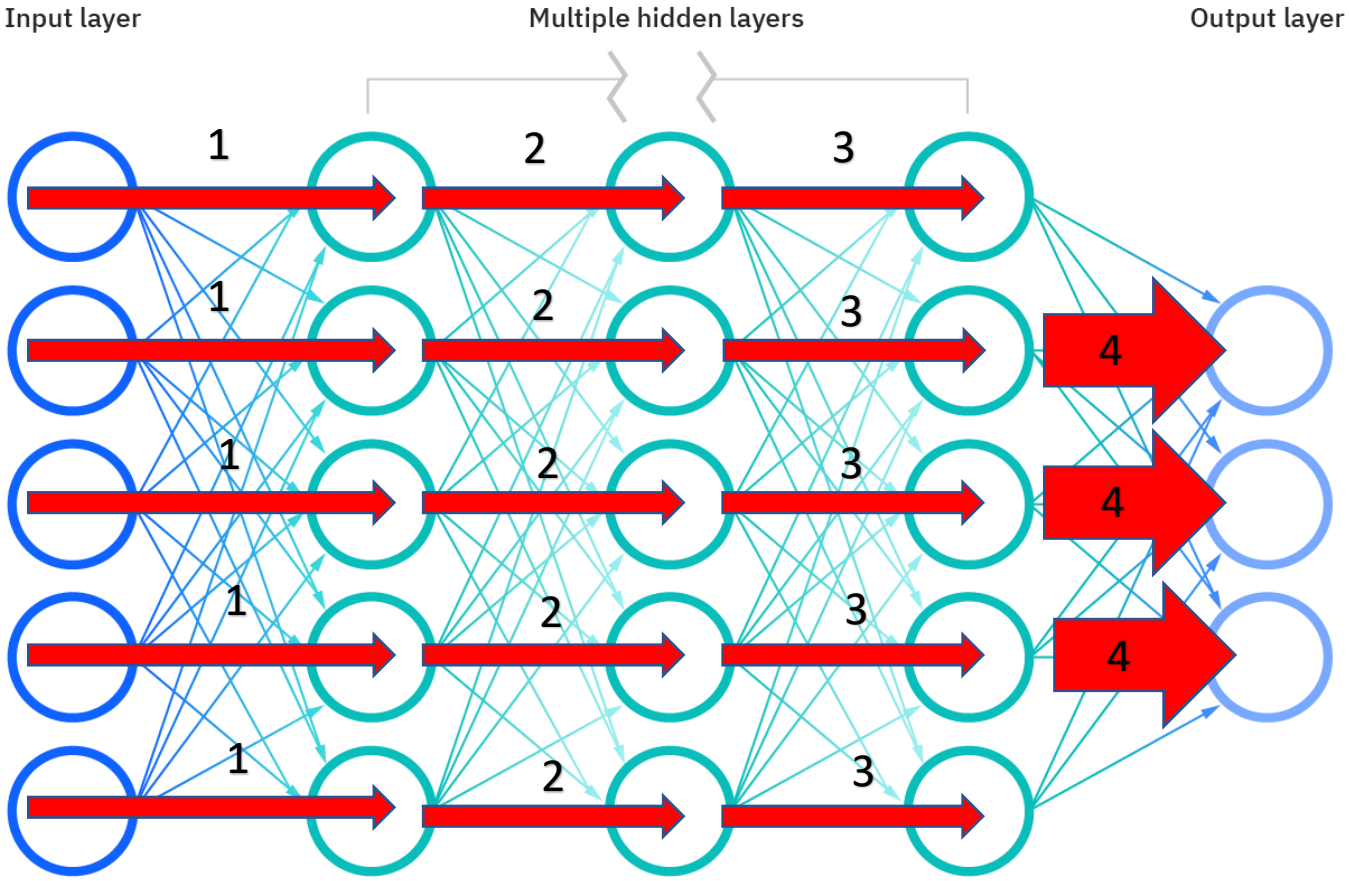
Deep neural network

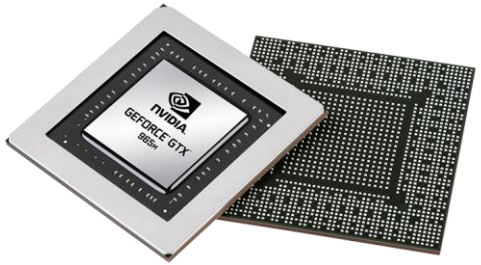






Deep neural network





Road Map

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

Field Programmable Gate Array (FPGA)

First, used for Prototyping by Electronic people

High-Level Language

Very Low-Level Language
(requires Digital Electronics knowledge)

Field Programmable Gate Array (FPGA)

First, used for Prototyping by Electronic people

High-Level Language

SystemVerilog

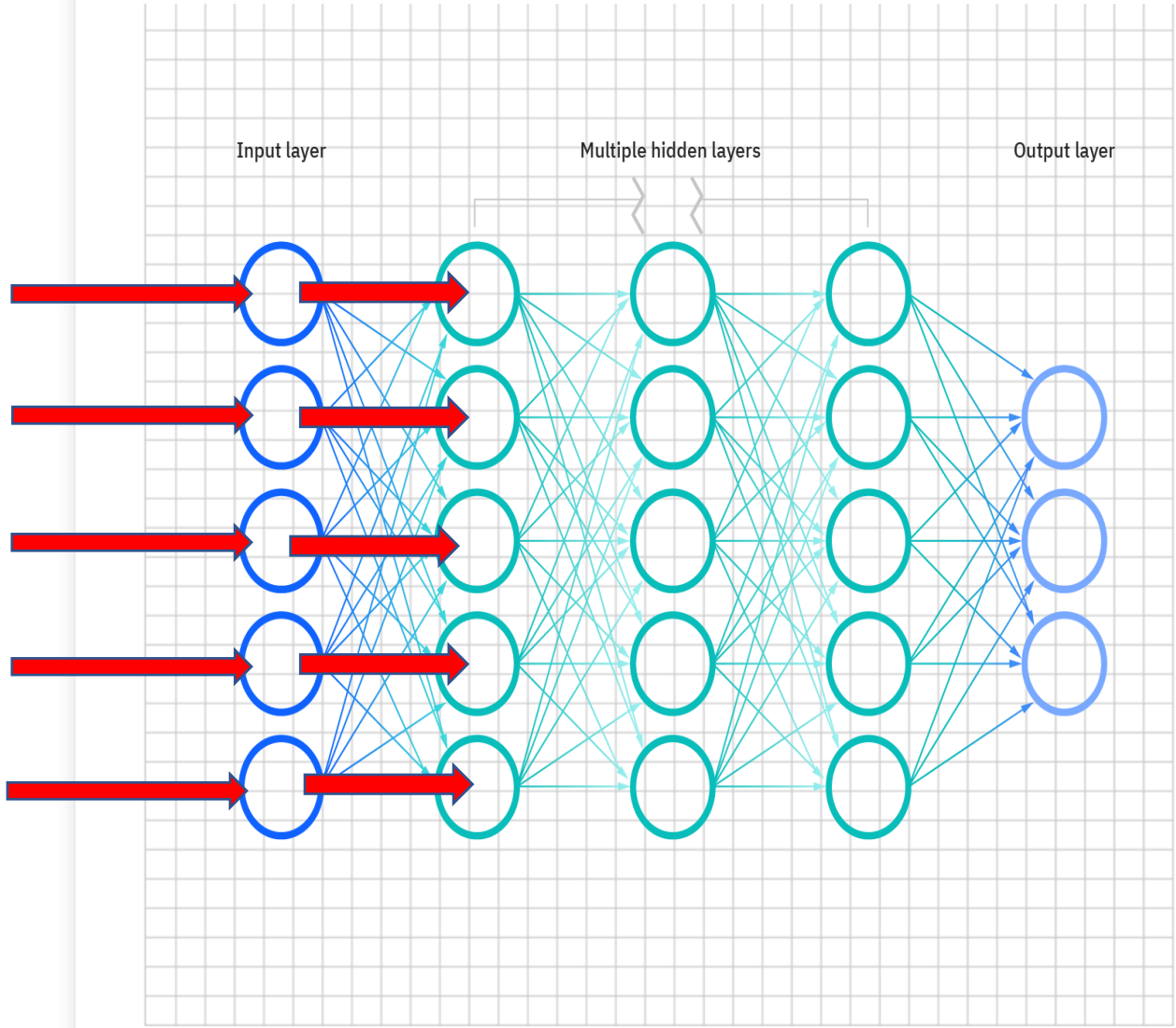
Very Low-Level Language
(requires Digital Electronics knowledge)

VHDL

Verilog HDL

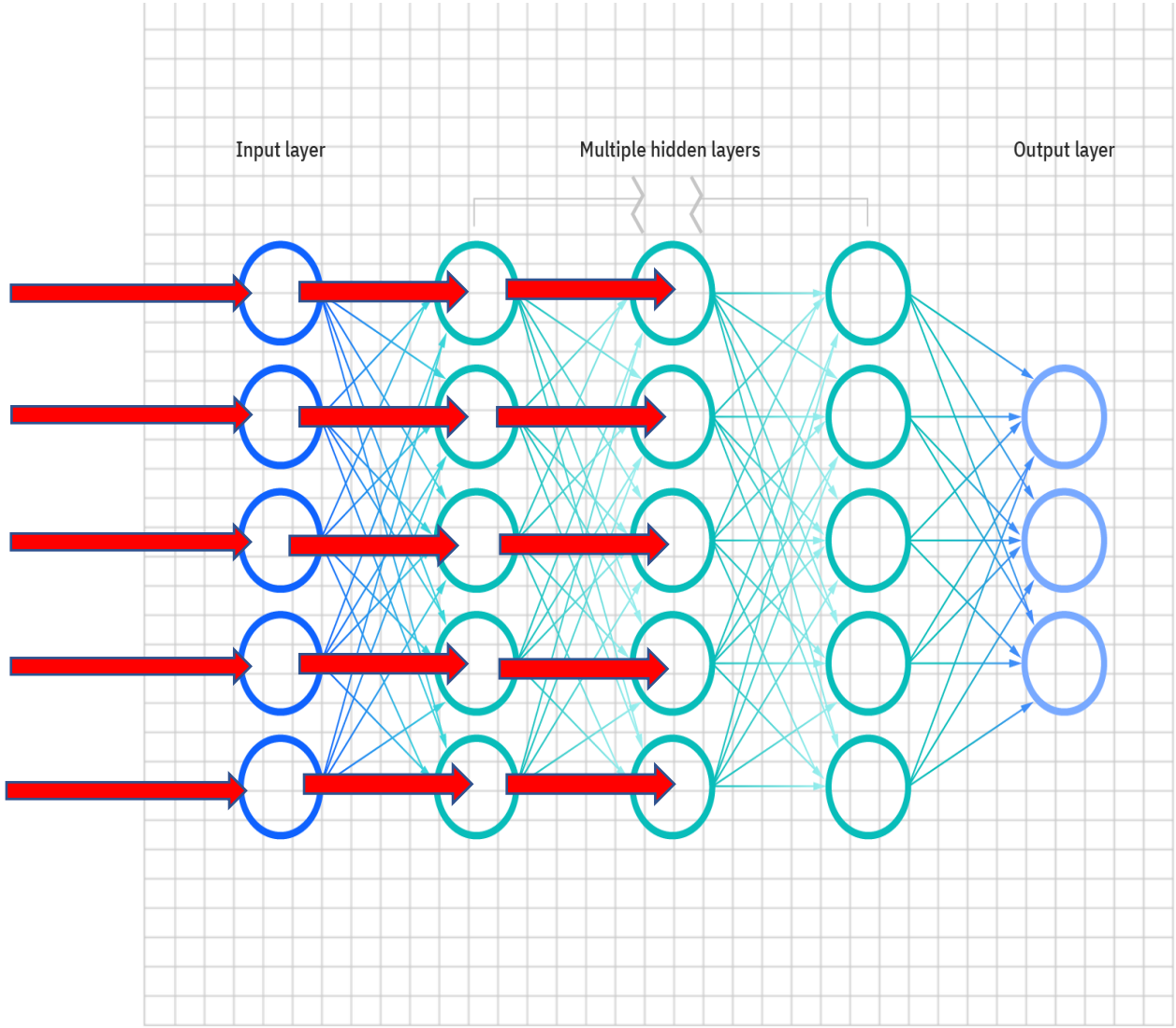
FPGAs in the Age of AI

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



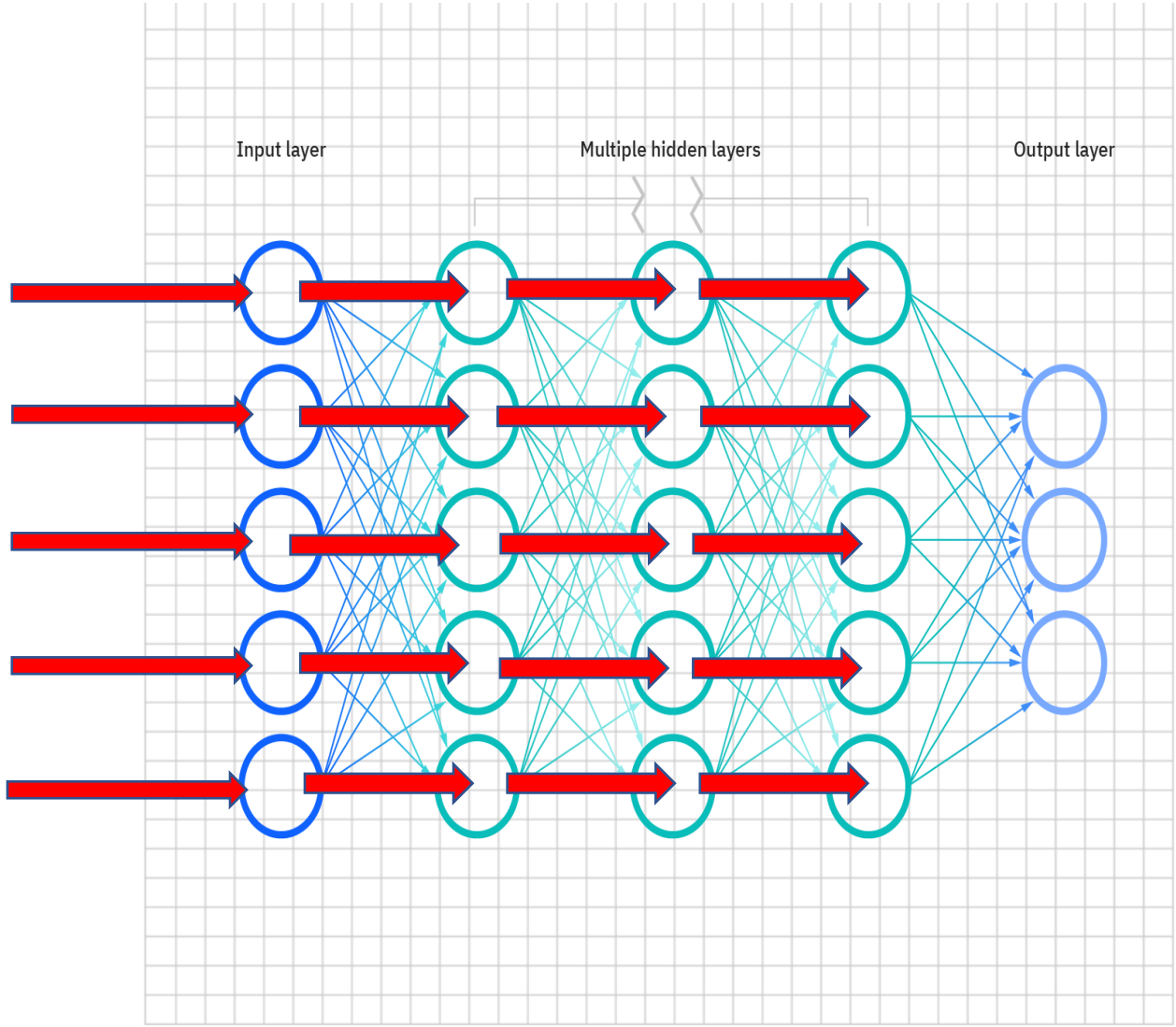
FPGAs in the Age of AI

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



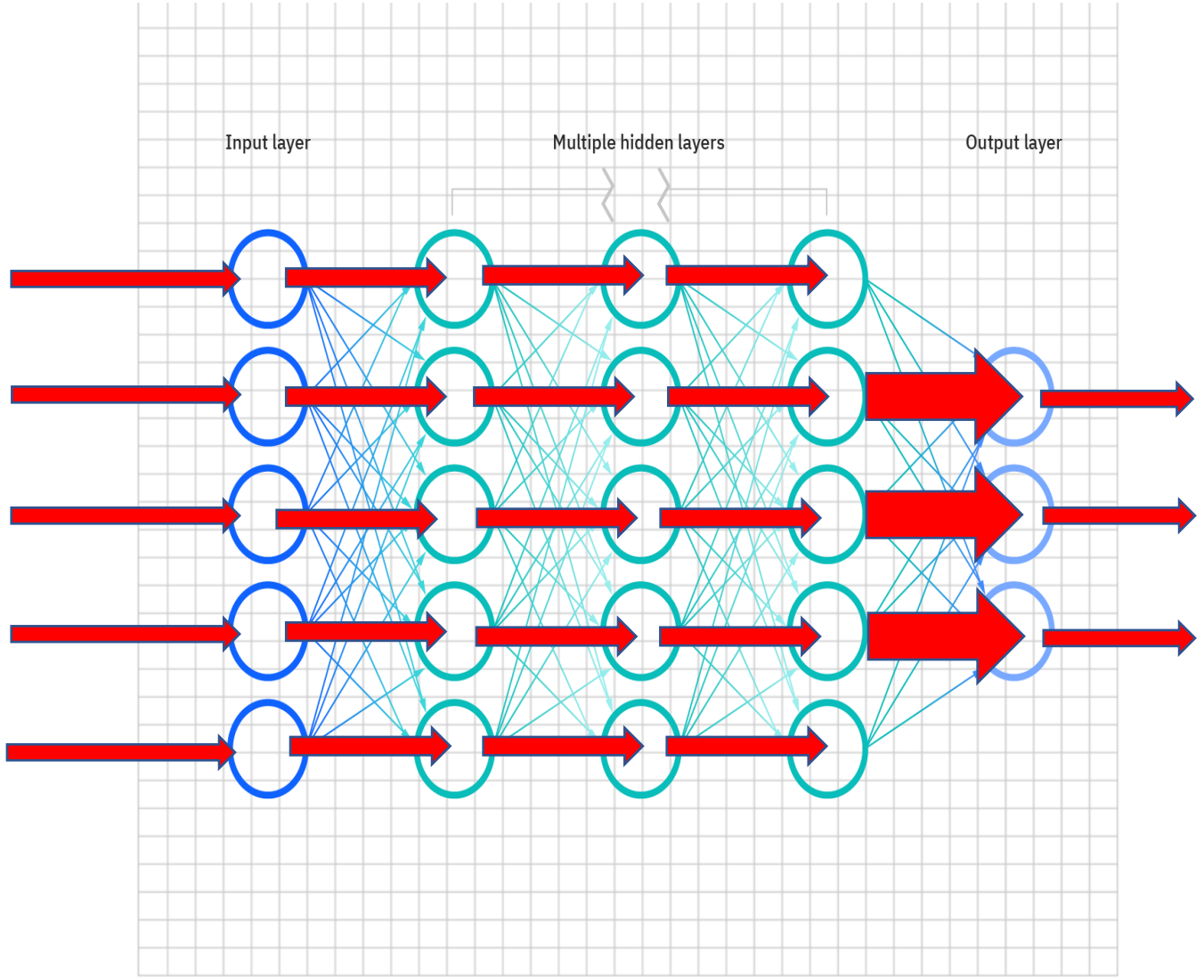
FPGAs in the Age of AI

- 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

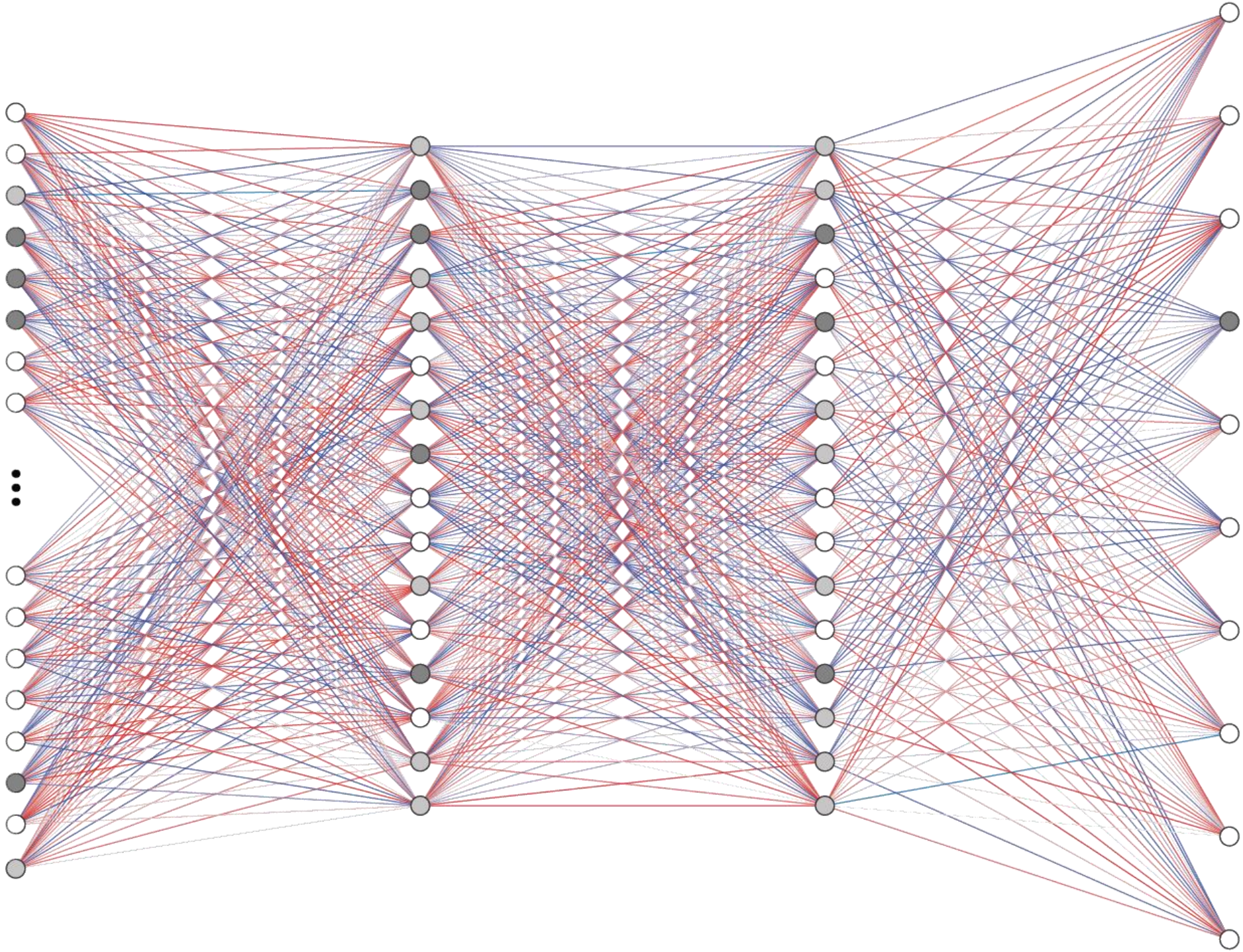
- Computing
- Processors
- How do electrons work for us?!
- Tradeoff of processors
- CPU
- GPU
- FPGA
- Accelerator

Accelerator

- Application Specific Integrated Circuit (ASIC)

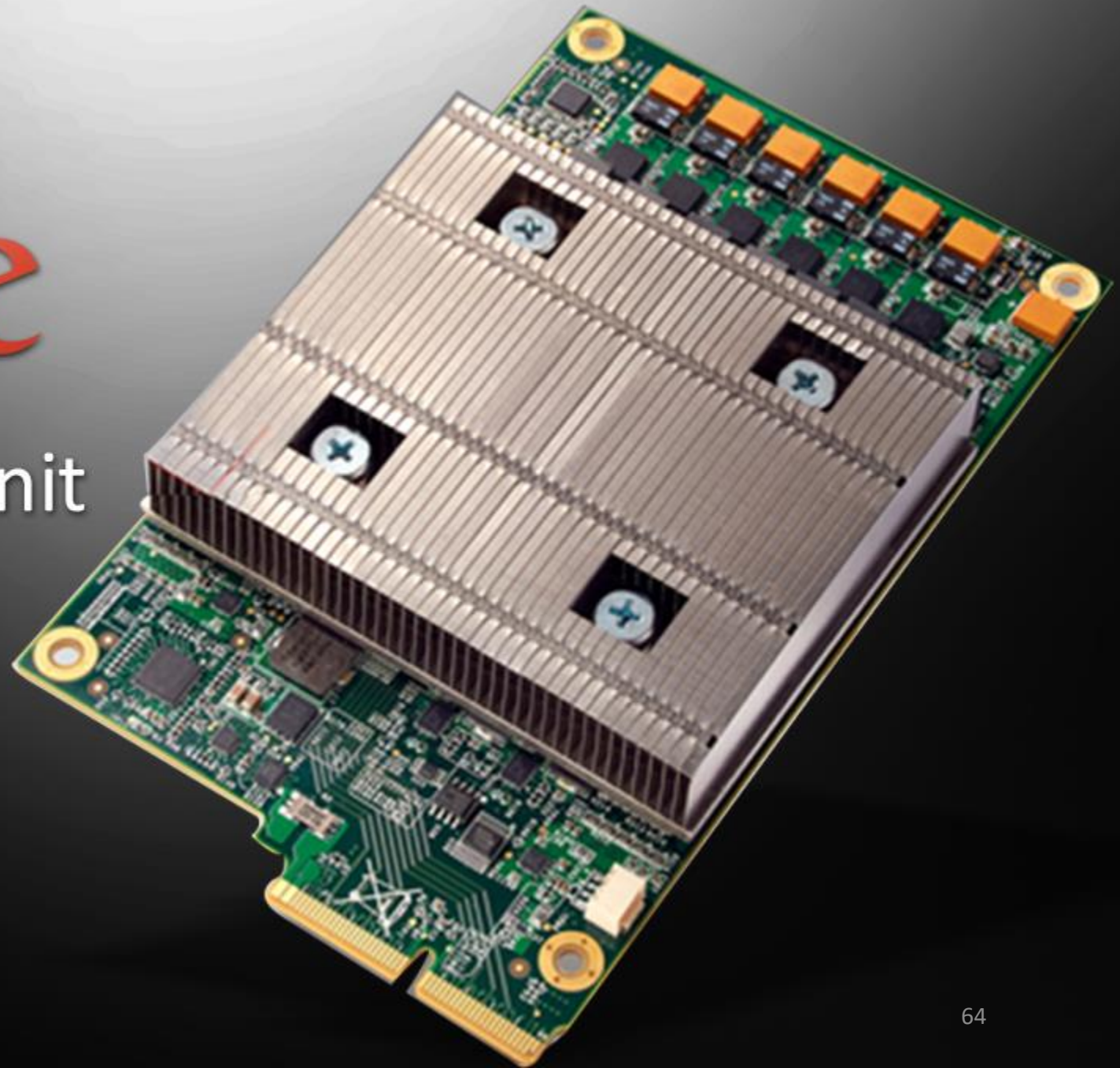
- Everything is specified by the designers

- No flexibility is expected but depends on the designers!

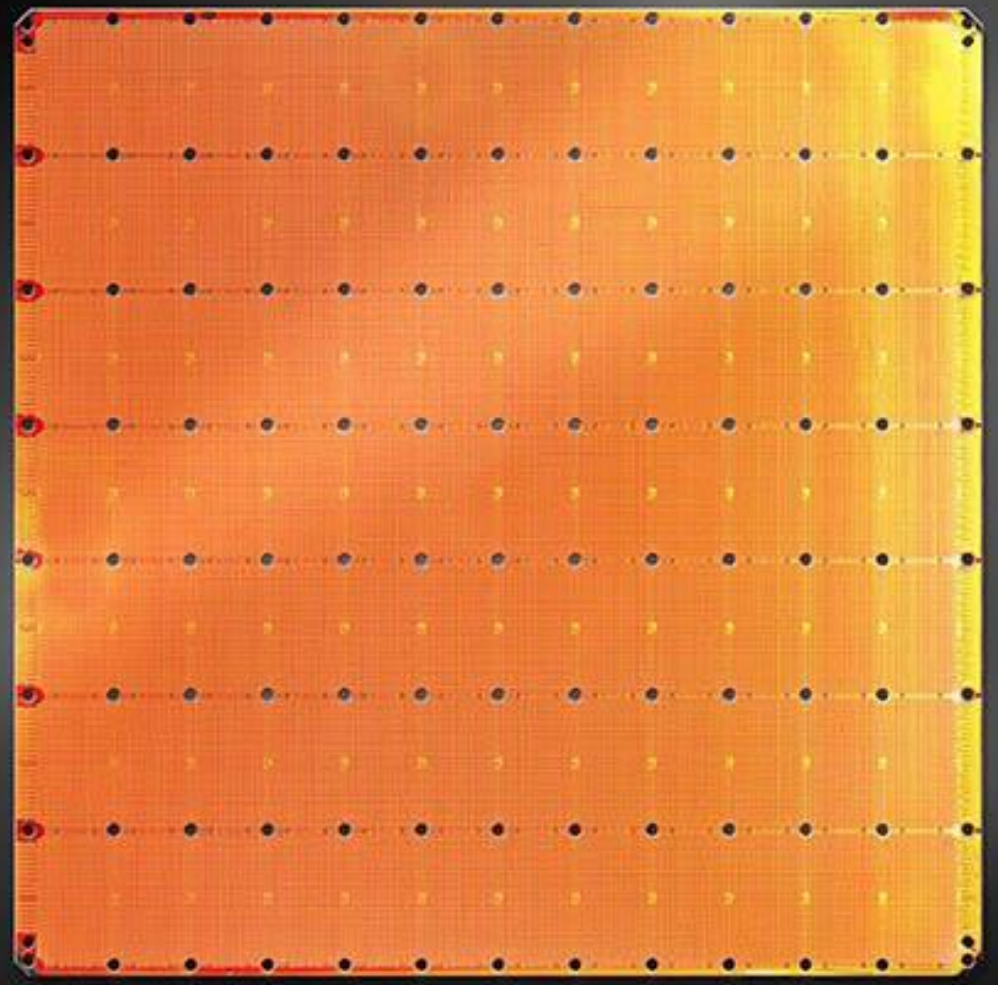


Google

Tensor Processing Unit



Cerebras Wafer Scale Engine



Cerebras WSE

1.2 Trillion Transistors

46,225 mm² Silicon

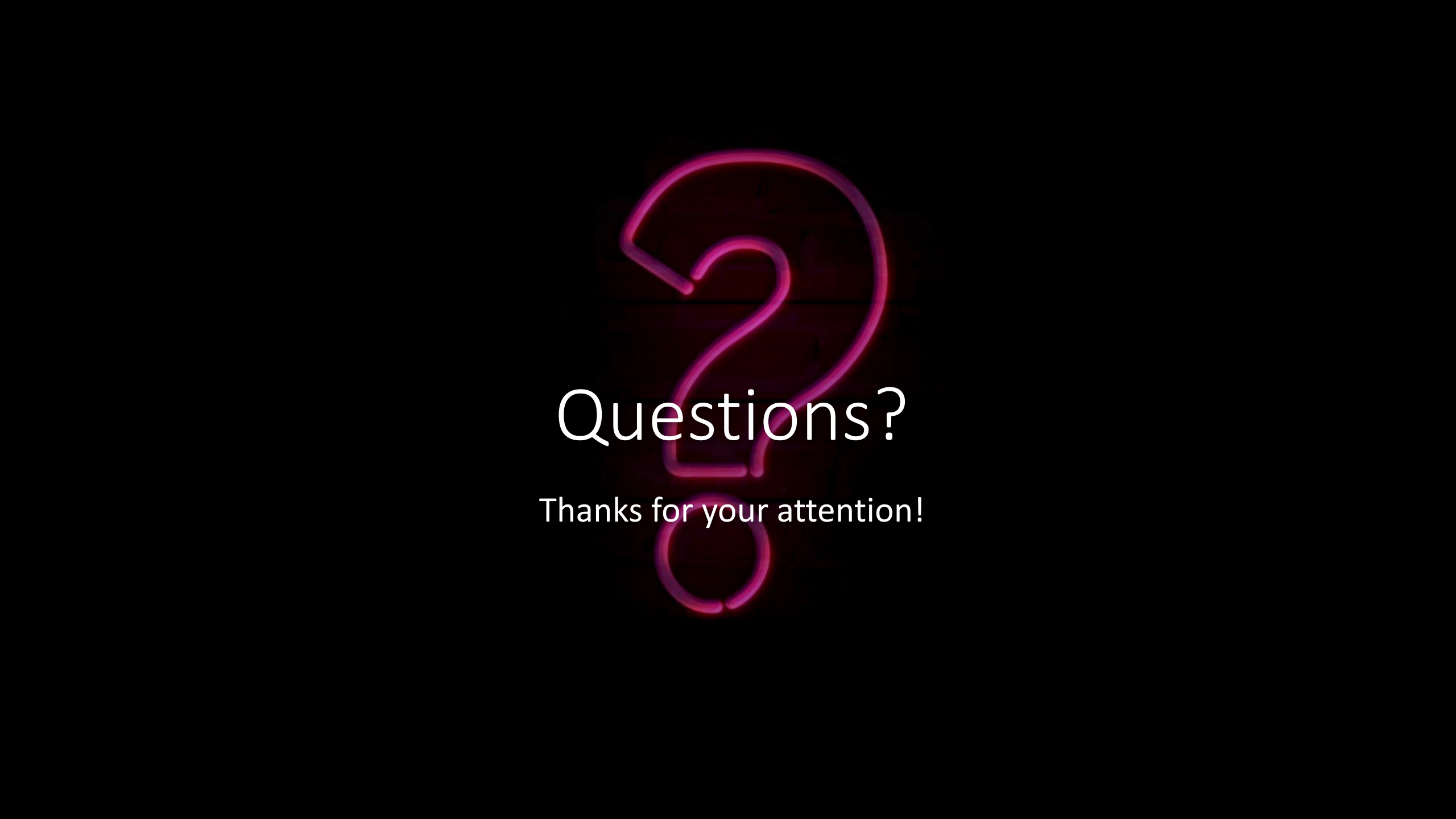
Largest GPU

21.1 Billion Transistors

815 mm² Silicon

Conclusion

- Computation
- Processors make the computation possible
- Tradeoff of different processors
- CPU
- GPU
- FPGA
- Accelerator


Questions?
Thanks for your attention!