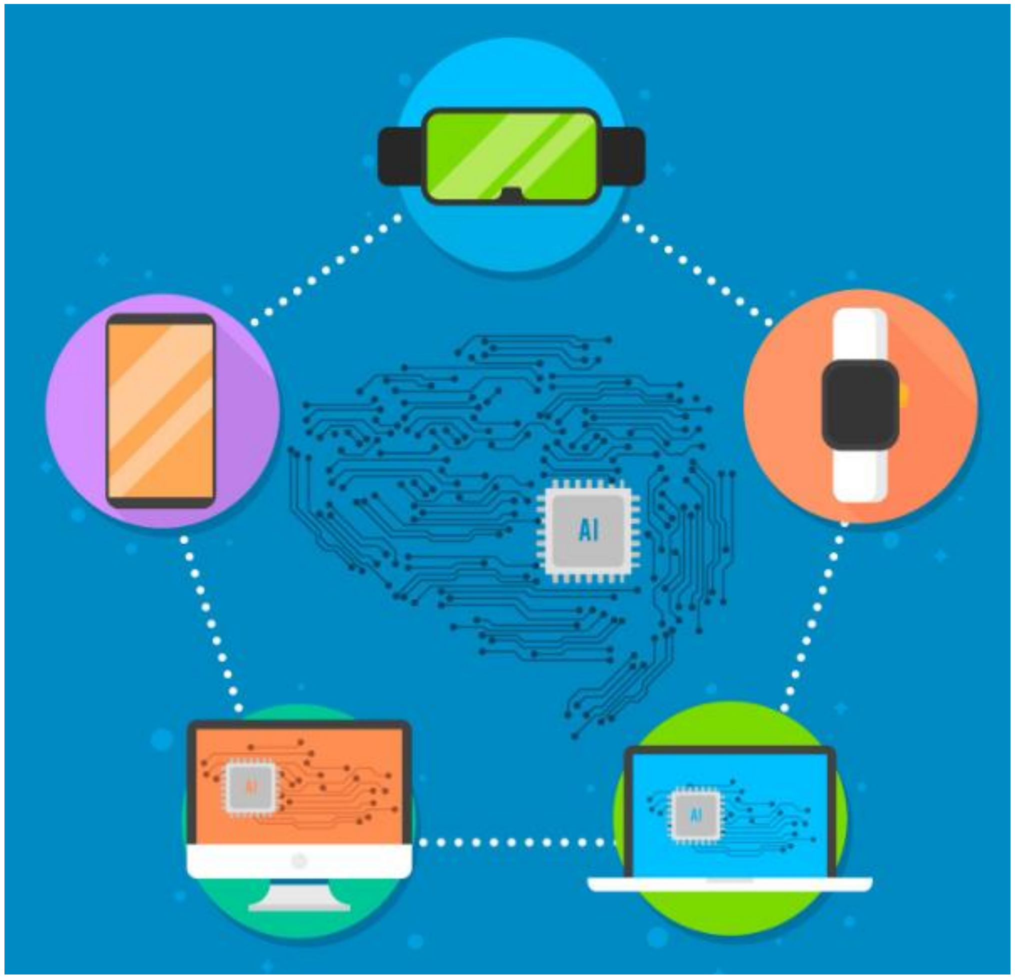
Accelerating Neural Networks on FPGAs

- Moiz Zaheer Malik
- B.Eng. Electronic Engineering
- Hochschule Hamm-Lippstadt



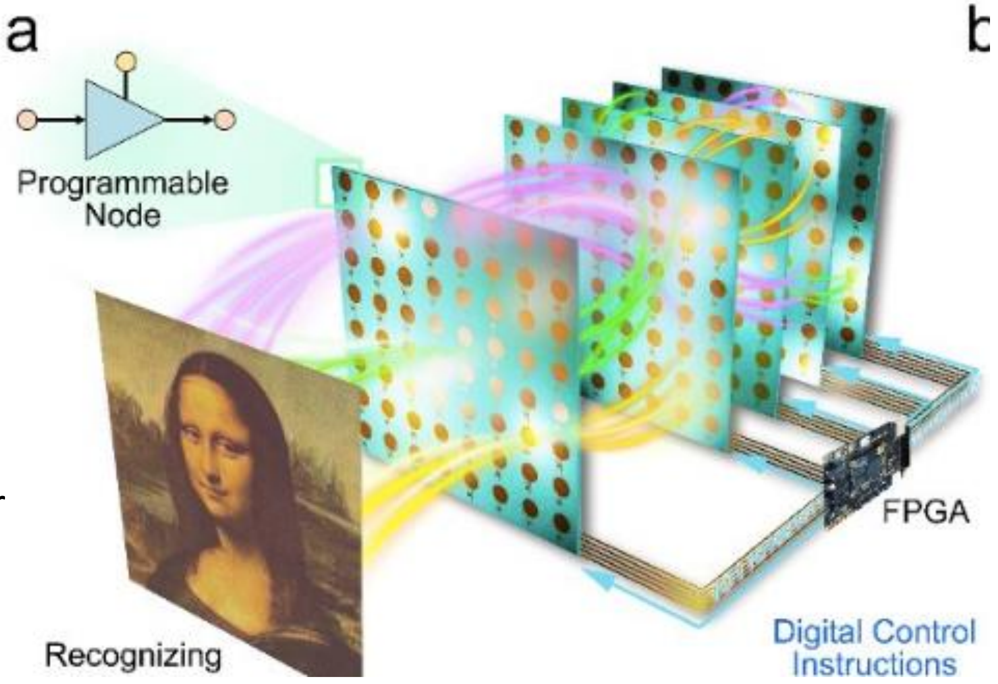
Agenda

- Motivation and problem statement
- What is an FPGA
- How FPGA programming works
- Basic FPGA structure
- Neural networks overview
- Training vs inference
- Why use FPGAs for neural networks
- Acceleration techniques
- Example FPGA architectures
- Challenges
- Conclusion

Motivation

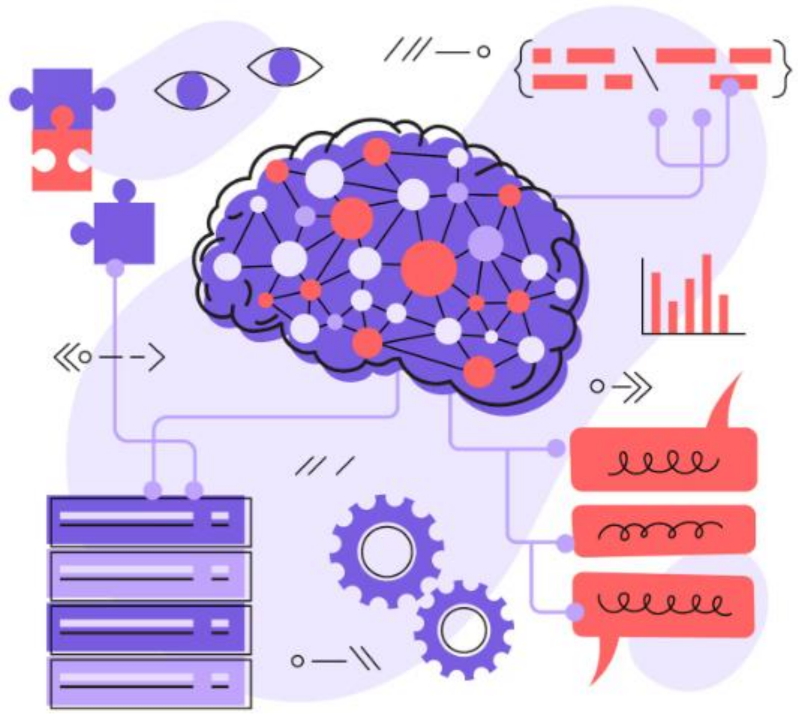
Neural networks are used everywhere

- Neural networks are used everywhere
- Image recognition
- Self driving cars
- Speech and language processing
- They need very high computing power



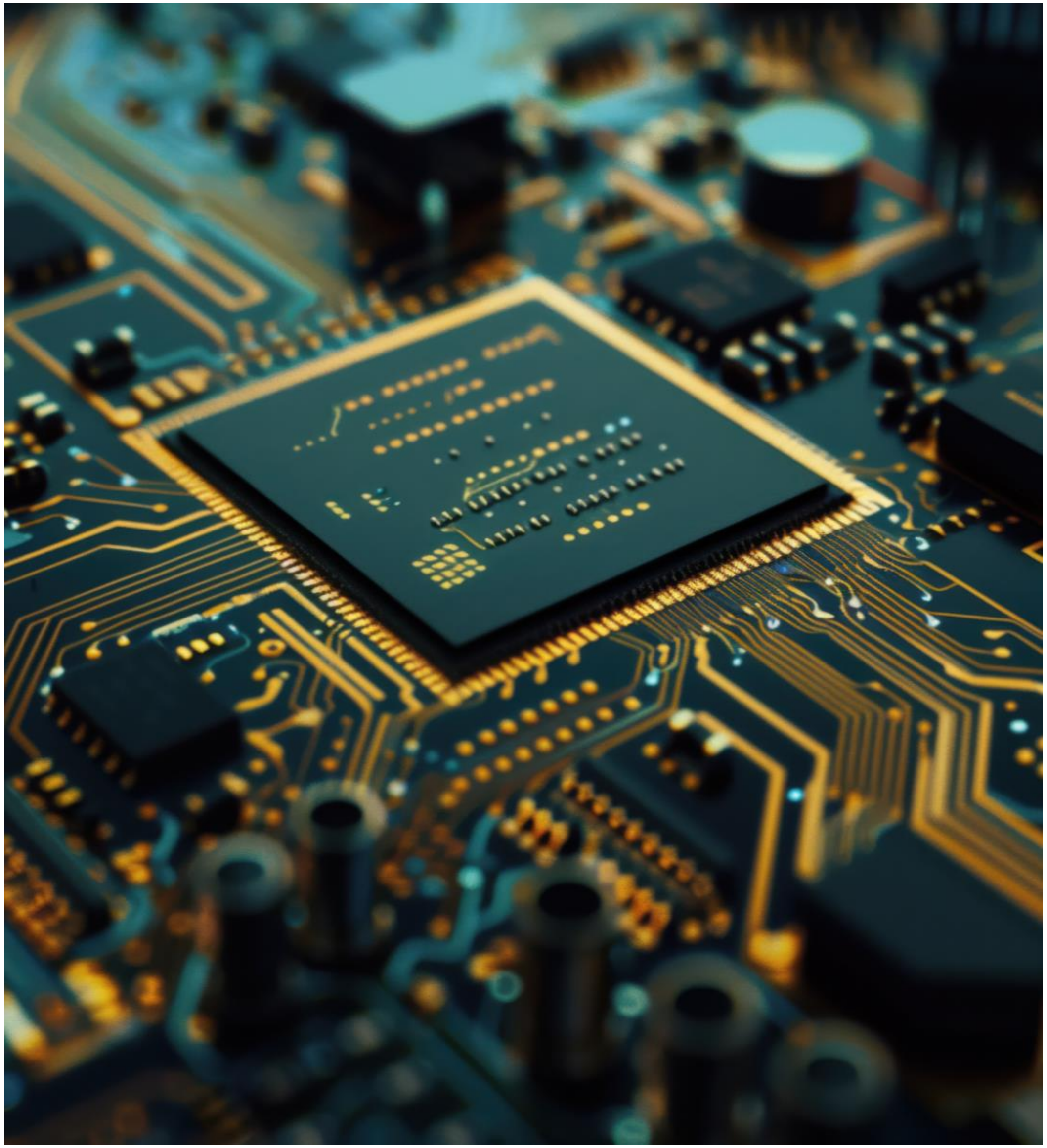
Problem Statement

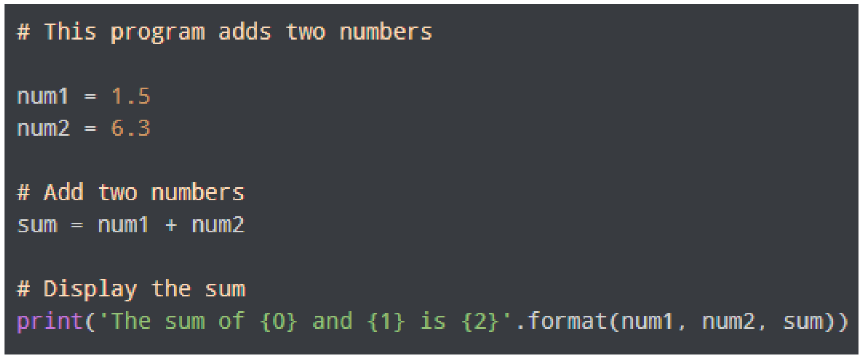
- Modern neural networks are very large
- Millions of parameters
- Billions of operations
- CPUs are too slow
- GPUs consume a lot of power
- Not good for real time systems



What is an FPGA?

- FPGA is programmable hardware
- Custom logic design
- Parallel processing
- Low power consumption

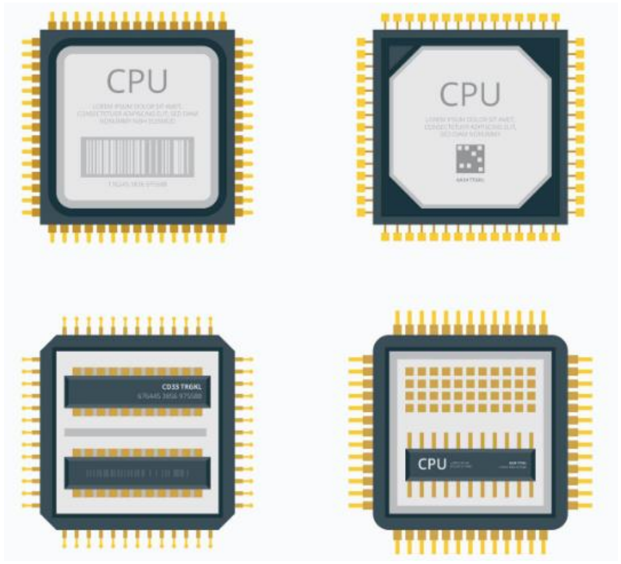




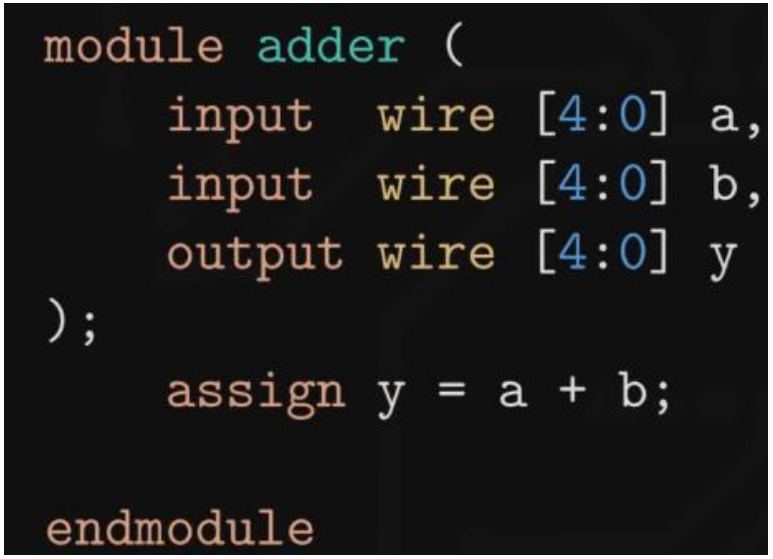
```
# This program adds two numbers

num1 = 1.5
num2 = 6.3

# Add two numbers
sum = num1 + num2

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

$\longrightarrow$

Compilation

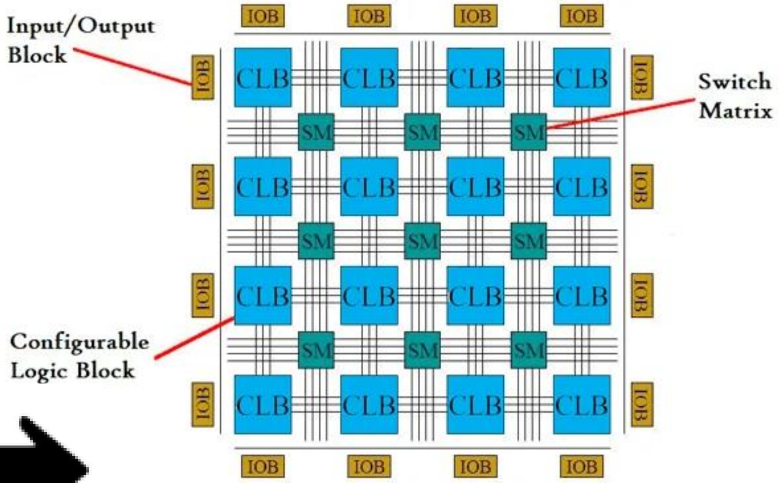


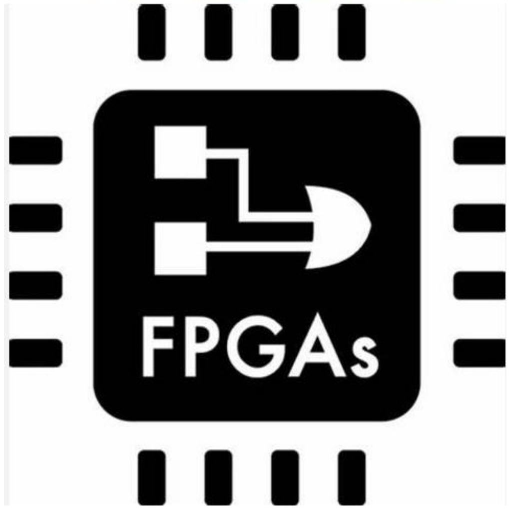
- Code is written by the developer
- Code is compiled
- Compilation produces a set of instructions
- These instructions run on a processor
- The processor executes instructions one by one



```
module adder (
    input  wire [4:0] a,
    input  wire [4:0] b,
    output wire [4:0] y
);
    assign y = a + b;

endmodule
```

Synthesis

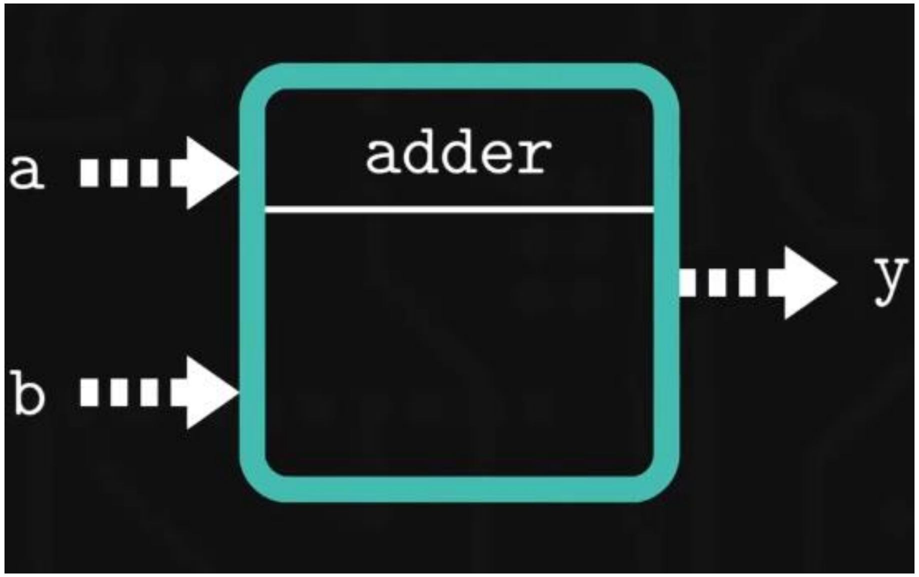




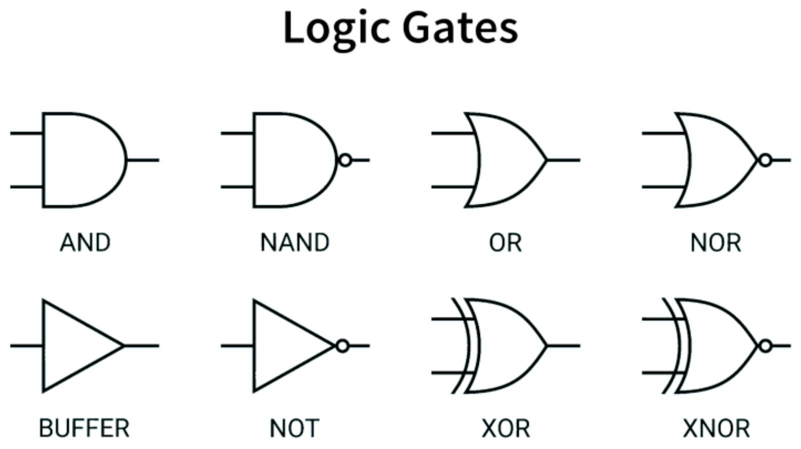
- Code can be written in a different way
- This code is synthesized for an FPGA
- The result is hardware, not instructions
- This type of code describes hardware behavior
- It is called Hardware Description Language (HDL)

Synthesis

- Synthesis maps code to physical hardware
- Code describes operations, for example adding two numbers
- Inputs: A and B
- Output: Y
- Synthesizer creates hardware blocks for this logic
- The result is real hardware, not software instructions



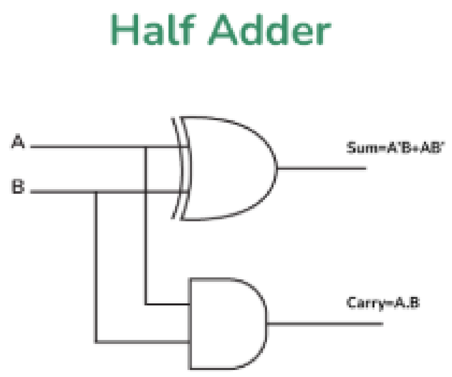
Logic blocks inside an FPGA

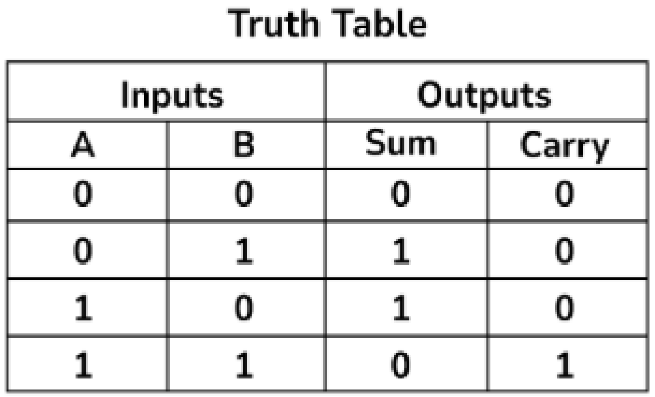


- Logic blocks are made of registers and logic gates
- Logic gates can be AND, OR, XNOR, NOT
- Gates are connected together to form logic blocks

Example logic blocks:
- Half adder
- Full adder

Inputs go in, output comes out

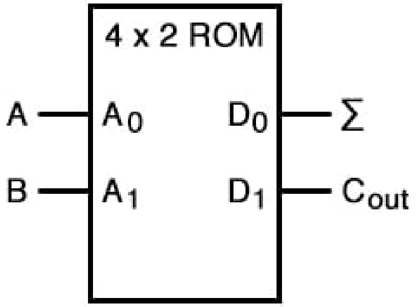




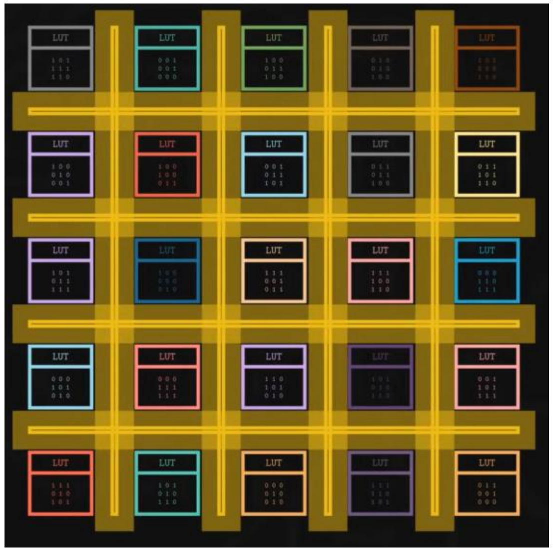
Truth Table

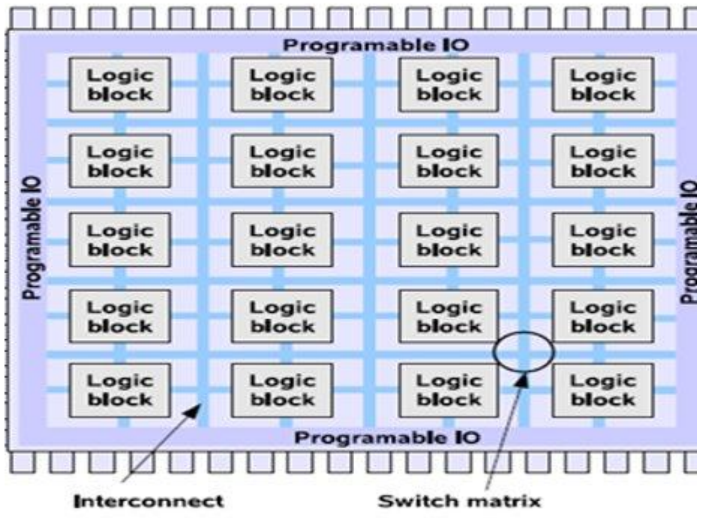
Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

LUTs and FPGA structure



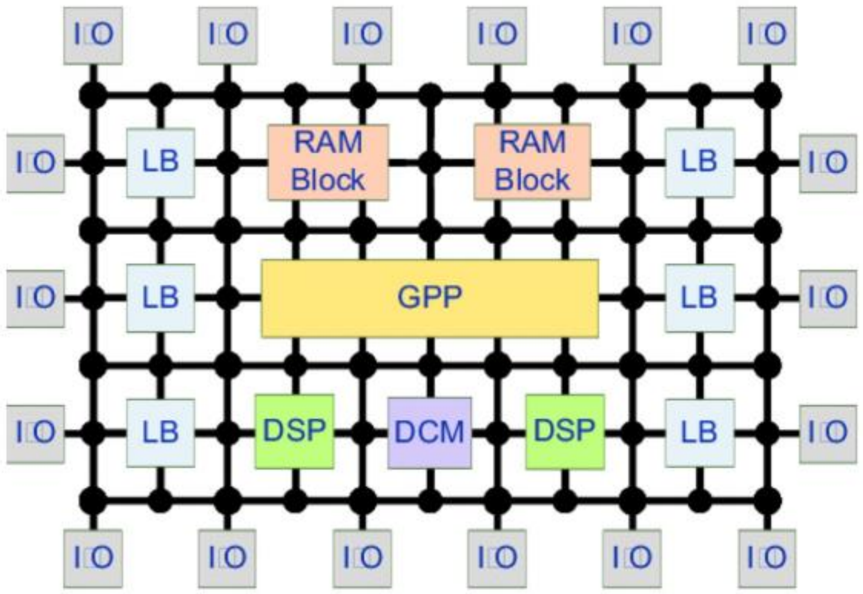
Address		Data	
A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

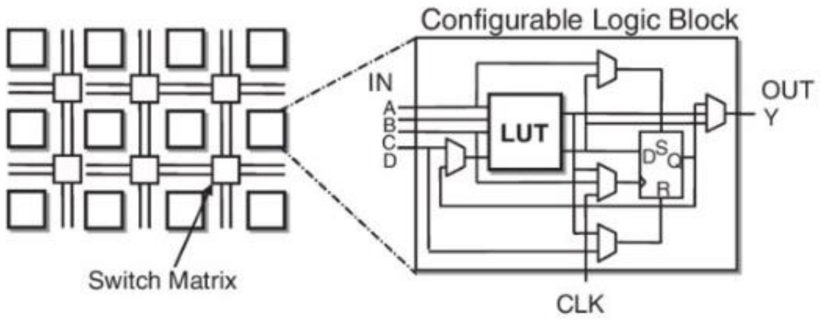




- Logic gates are implemented using lookup tables (LUTs)
- A LUT implements logic using stored values
- Multiple LUTs are combined together
- LUTs are connected through a programmable switching fabric
- This interconnection is programmable
- All together, this forms an FPGA
- The FPGA behavior depends on how we code it

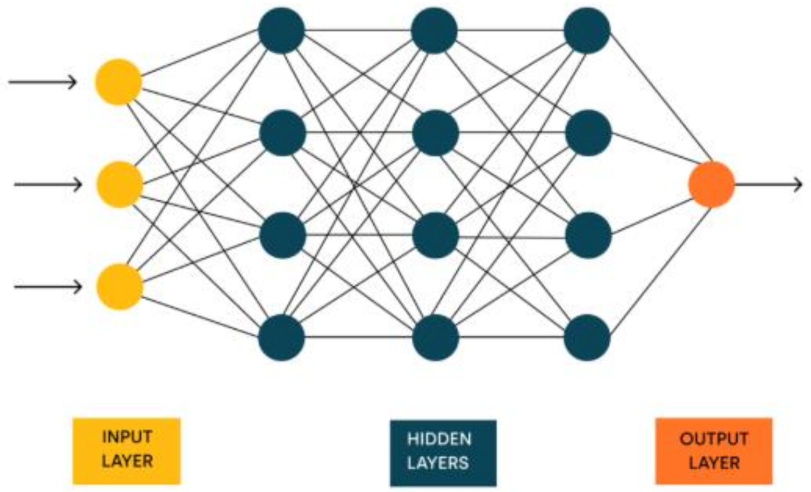
Inside an FPGA

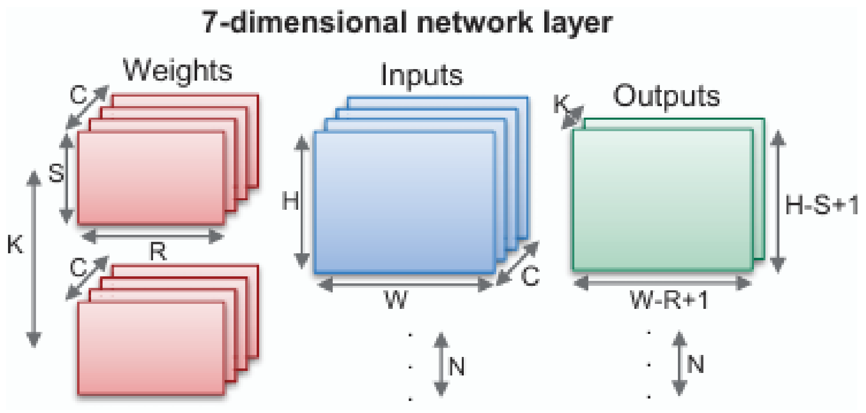




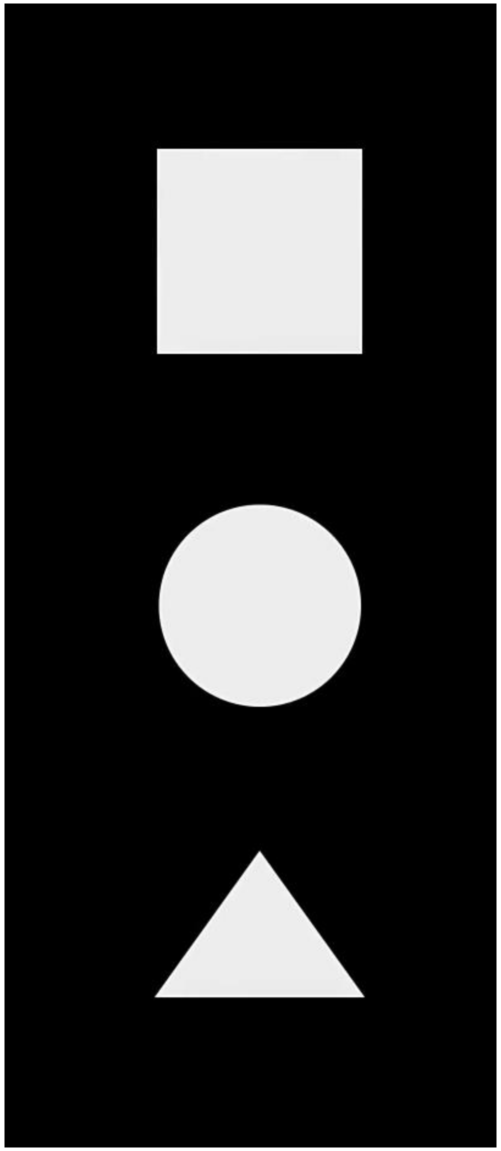
What is a Neural Network?

- Made of layers
- Uses weights
- Processes data step by step
- Produces output

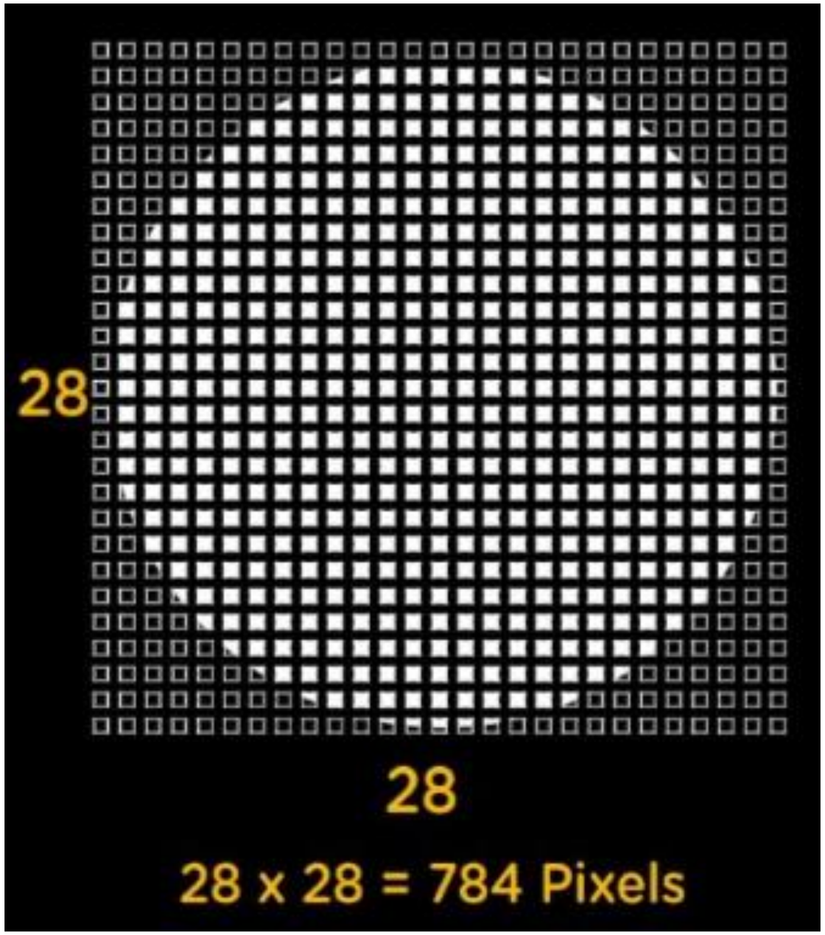




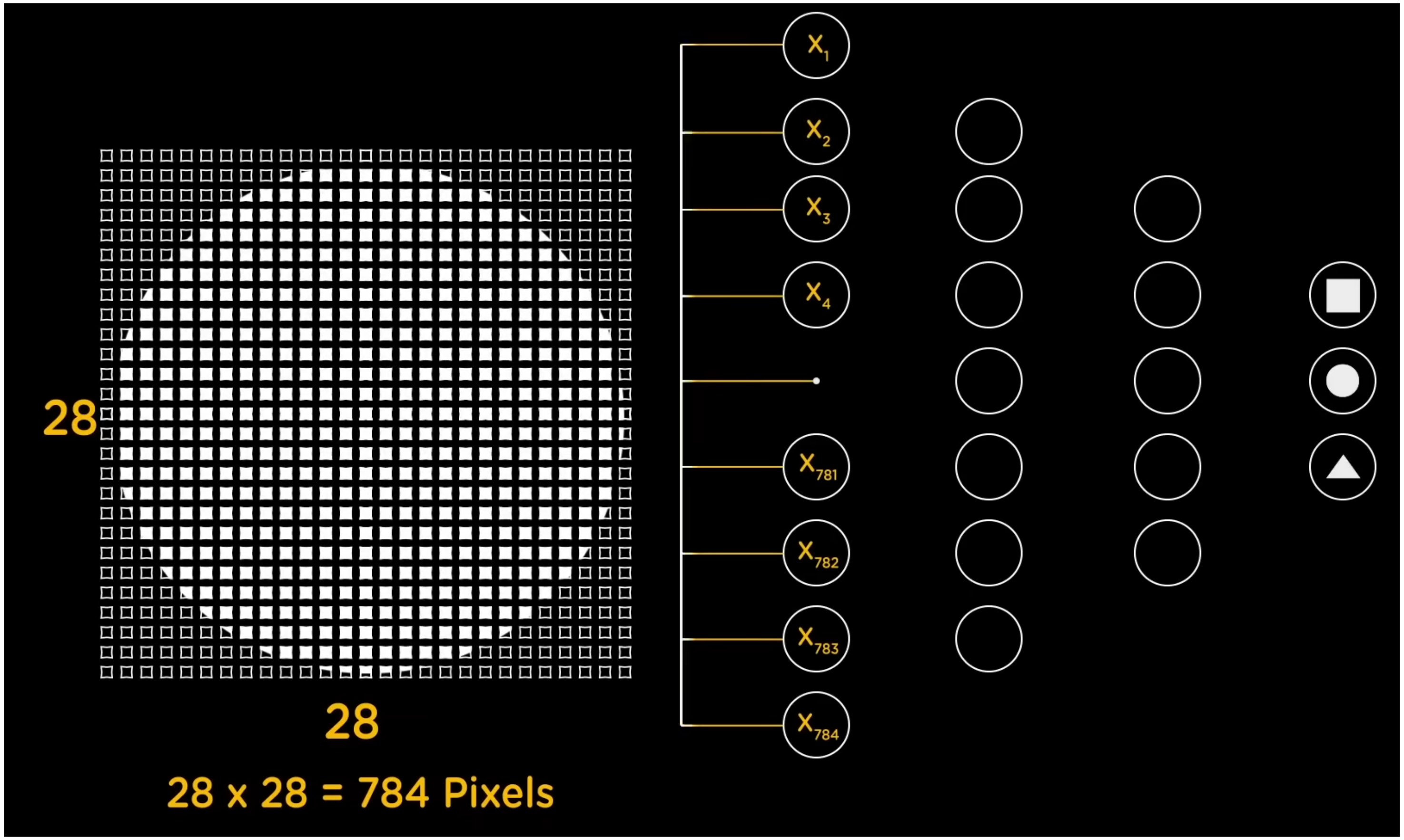
How Neural Networks Work



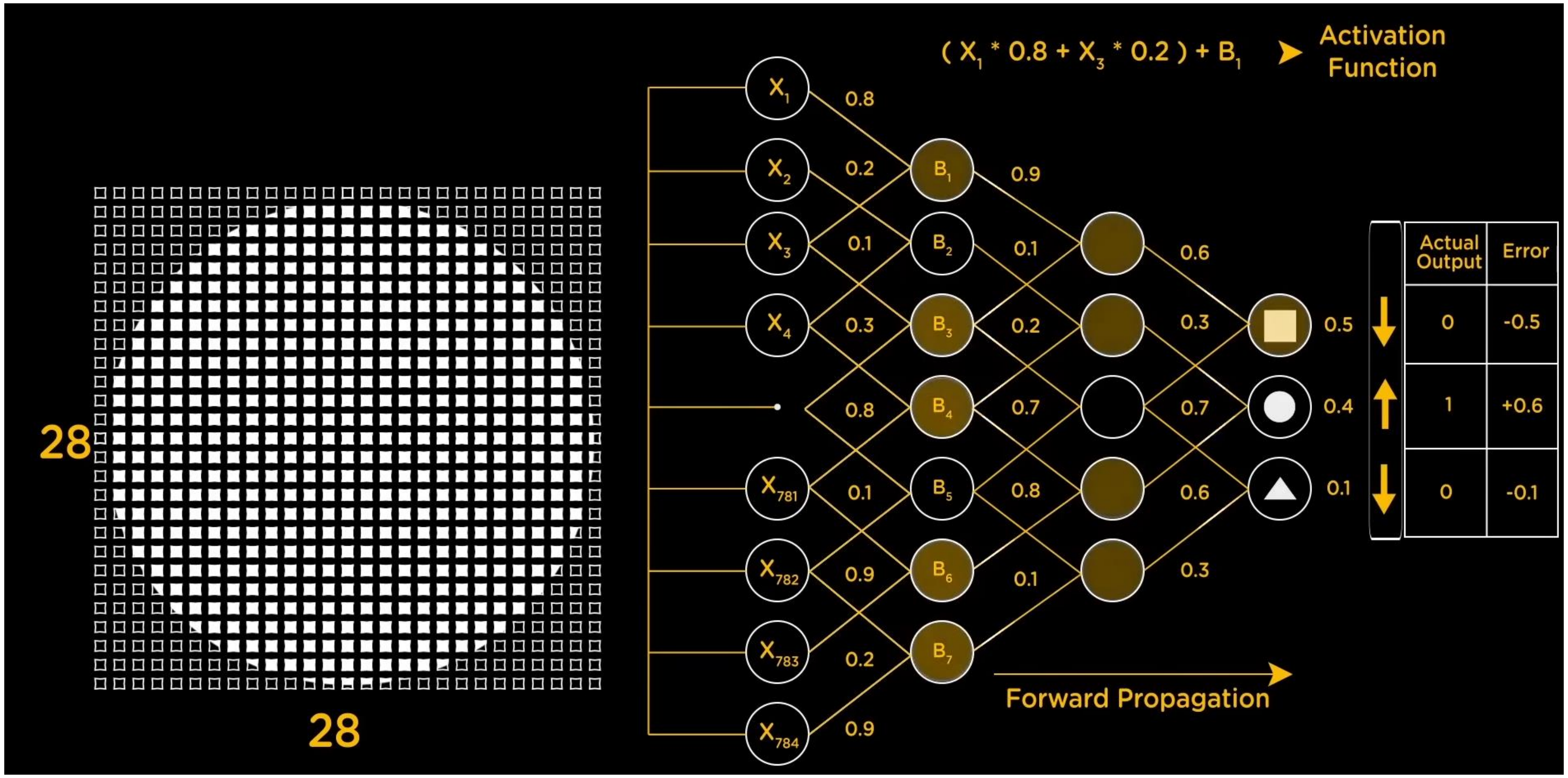
How Neural Networks Work



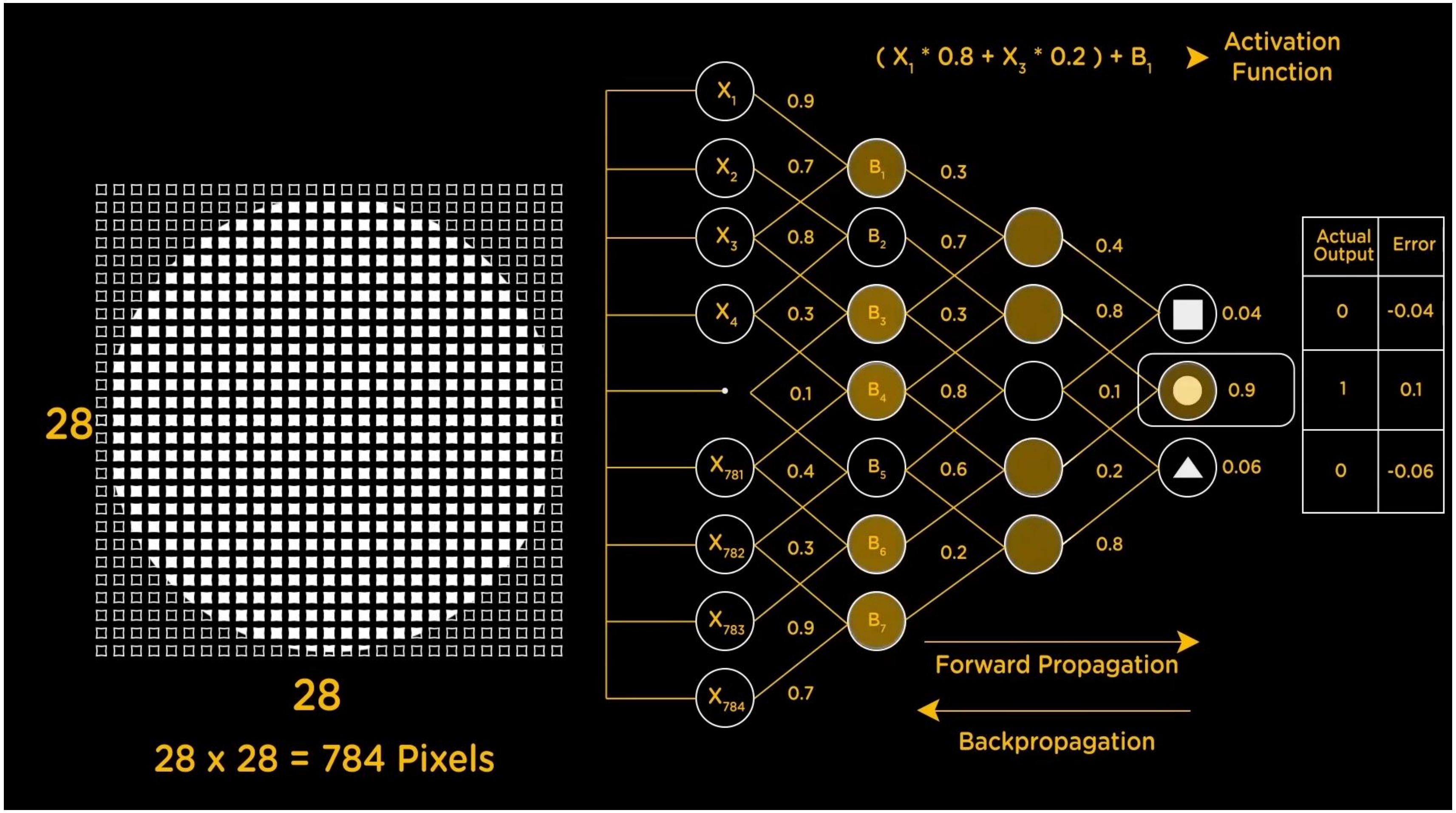
How Neural Networks Work

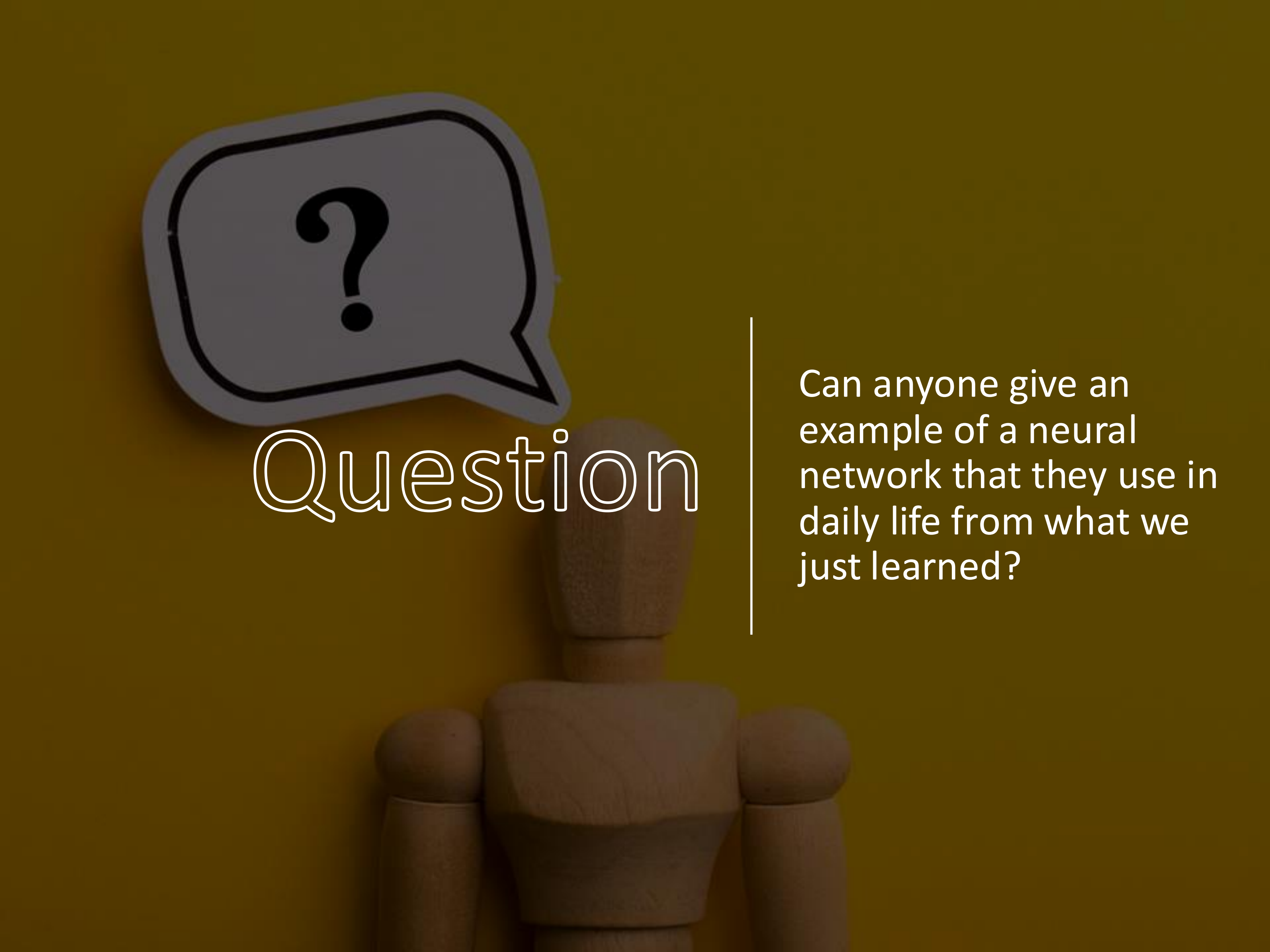


How Neural Networks Work



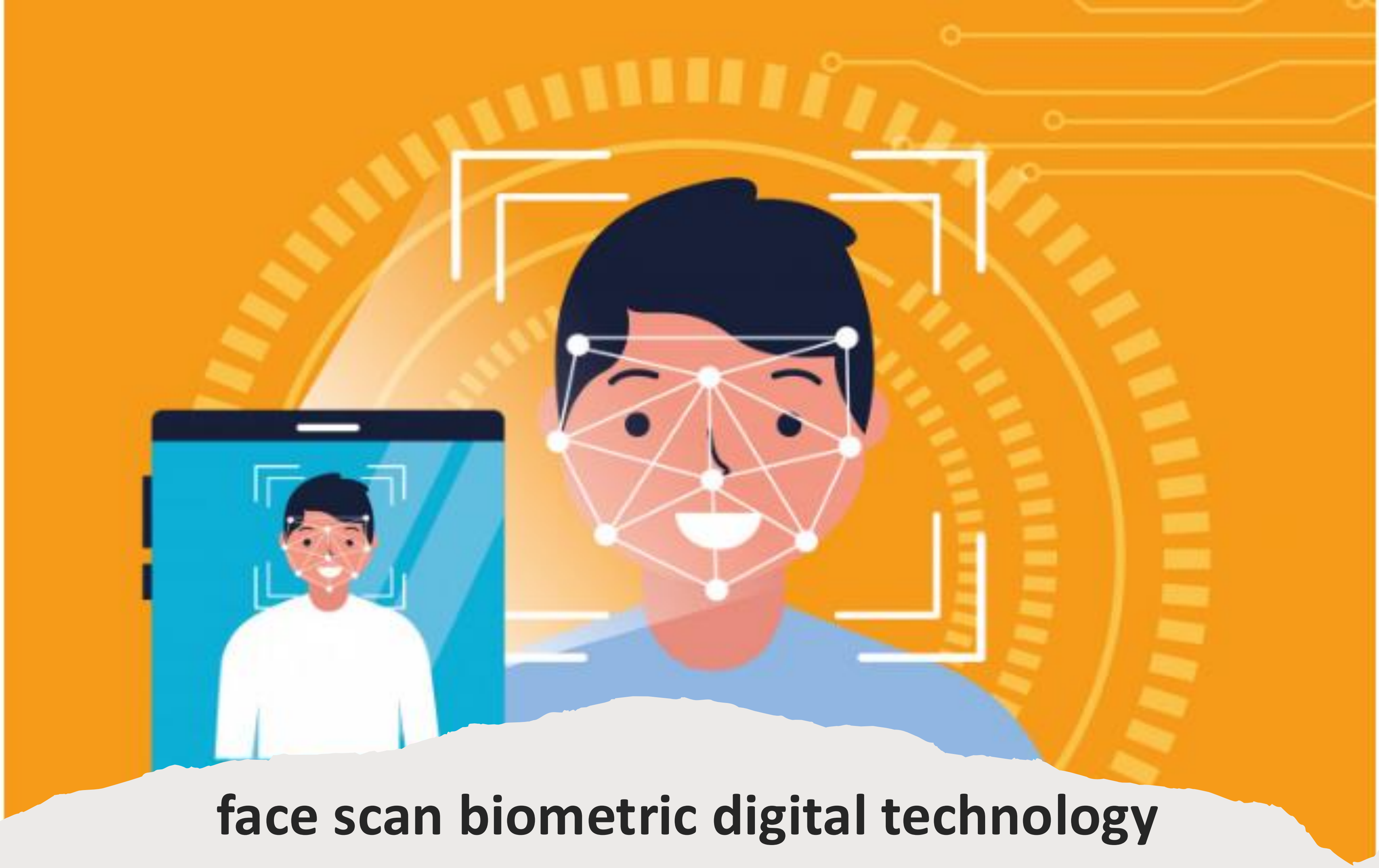
How Neural Networks Work





Question

Can anyone give an example of a neural network that they use in daily life from what we just learned?



face scan biometric digital technology

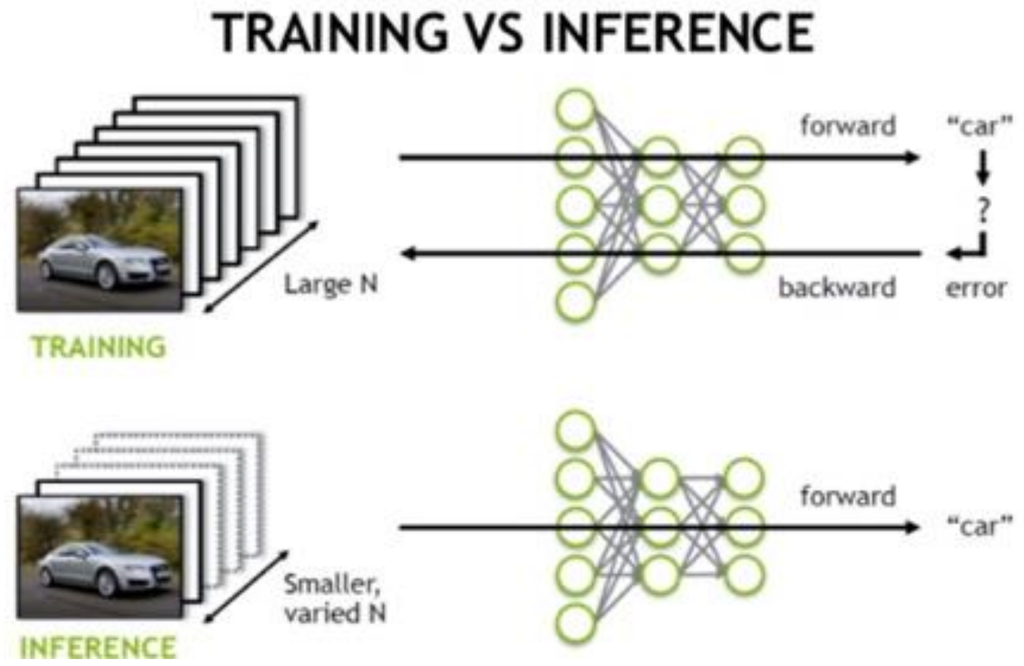
Training vs Inference

Training:

- Learn weights from data
- Very compute heavy

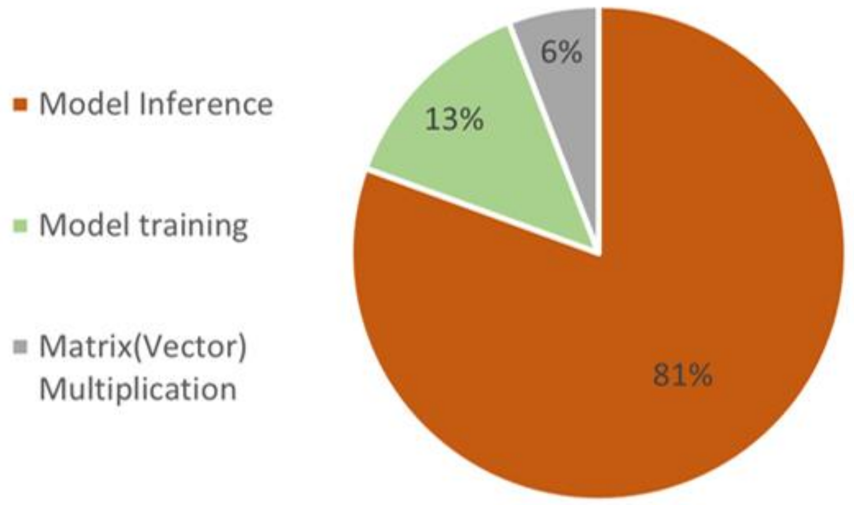
Inference:

- Use trained model
- Only forward pass
- Used in real applications
- FPGA focus: inference



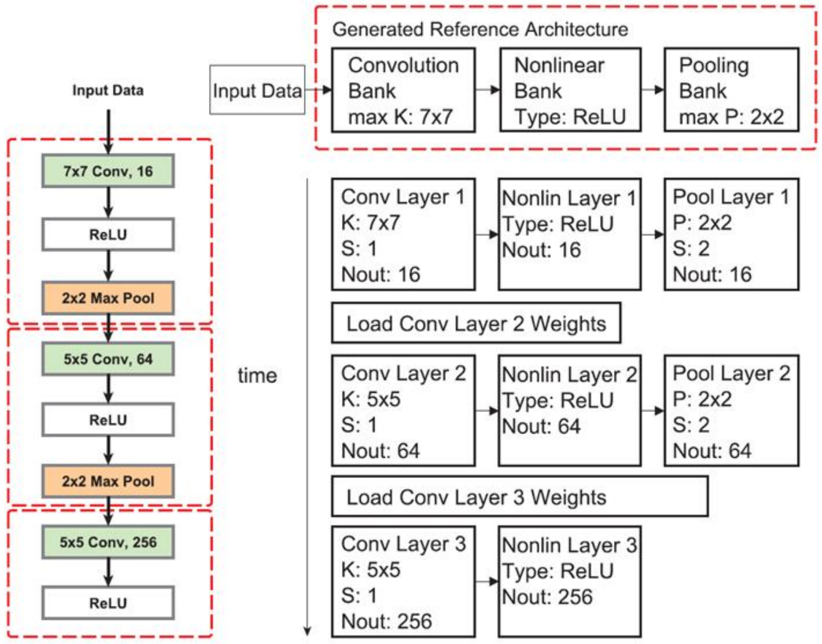
Why Use FPGA for Neural Networks?

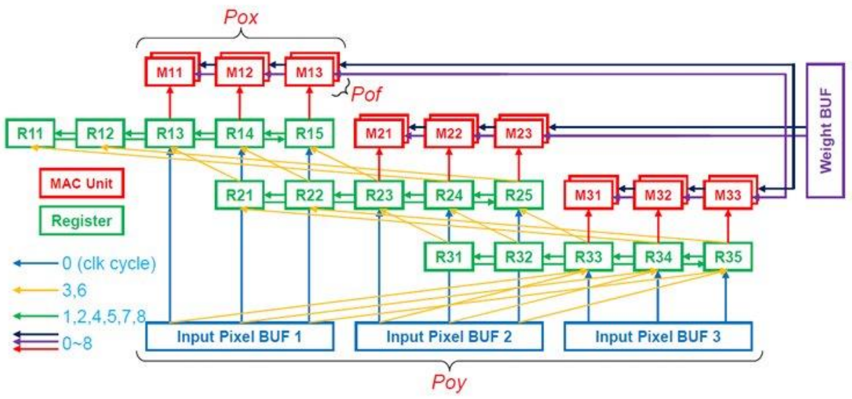
- Parallel execution
- Low latency
- Low power consumption
- Reconfigurable hardware



Main Acceleration Techniques

- Parallelism
- Pipelining
- Quantization
- Data reuse


Input Data
7x7 Conv, 16
ReLU
2x2 Max Pool
5x5 Conv, 64
ReLU
2x2 Max Pool
5x5 Conv, 256
ReLU
Input Data
Generated Reference Architecture
Convolution Bank max K: 7x7
Nonlinear Bank Type: ReLU
Pooling Bank max P: 2x2
time
Conv Layer 1 K: 7x7 S: 1 Nout: 16
Nonlin Layer 1 Type: ReLU Nout: 16
Pool Layer 1 P: 2x2 S: 2 Nout: 16
Load Conv Layer 2 Weights
Conv Layer 2 K: 5x5 S: 1 Nout: 64
Nonlin Layer 2 Type: ReLU Nout: 64
Pool Layer 2 P: 2x2 S: 2 Nout: 64
Load Conv Layer 3 Weights
Conv Layer 3 K: 5x5 S: 1 Nout: 256
Nonlin Layer 3 Type: ReLU Nout: 256



Example FPGA Architectures

- Researchers build FPGA designs for neural networks
- Some designs are made for known networks like VGG16
- Some designs use math tricks to reduce work
- Data is reused to save time and power

Challenges

- Limited hardware resources
- Memory bandwidth limits
- Design complexity
- Precision trade-offs

Conclusion

- FPGAs are good for neural networks
- High efficiency and flexibility
- Good for real-time and edge systems
- Future work improves tools and memory

Any Questions?