life.augmented

Introduzione all'Intelligenza Artificiale nei Microcontrollori

Valeria Tomaselli

Tiny Machine Learning Power

FarmBeats project
Microsoft

life.augmented

Edge AI/TinyML

‘A Tsunami of TinyML Devices is Coming’

R. El-Ouazzane, STMicroelectronics

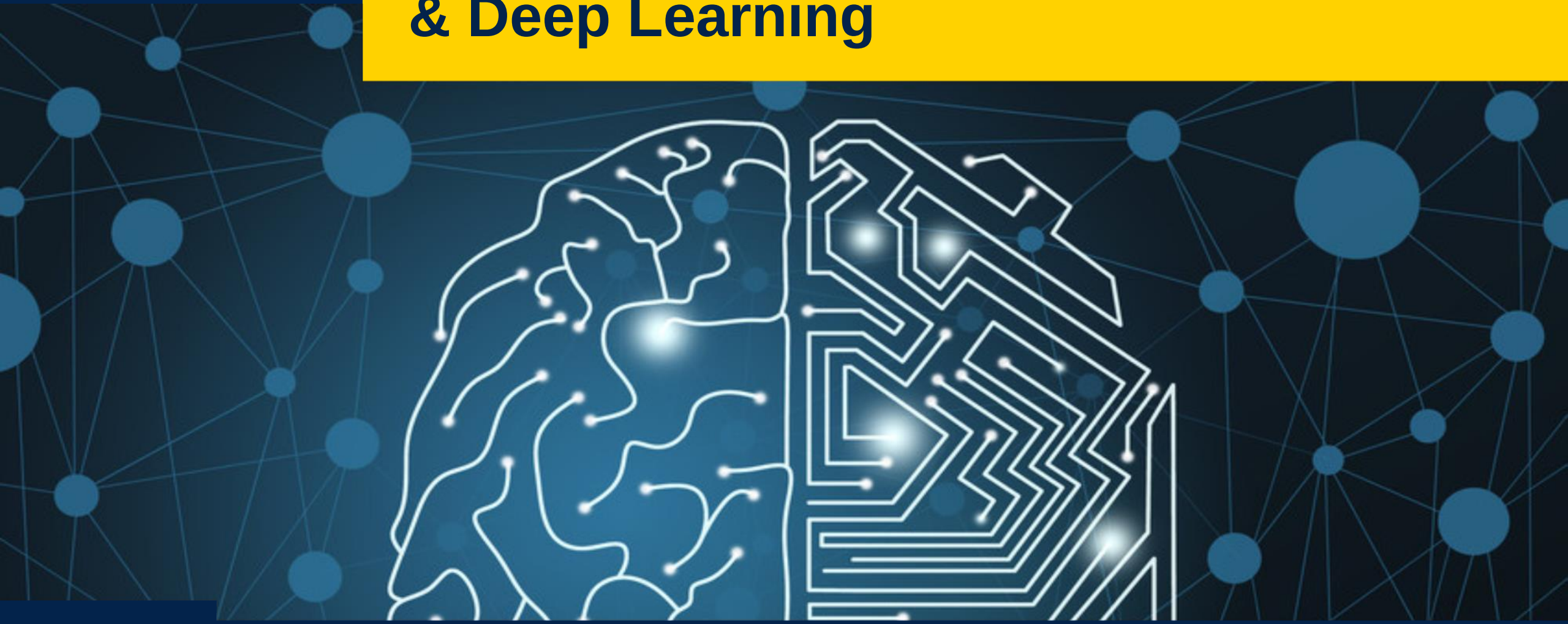
life.augmented

Agenda

1. Introduction to AI & Deep Learning
2. Tiny Machine Learning (TinyML) and its benefits
3. TinyML use cases and examples
4. Neural Networks
5. Training Neural Networks
6. TinyML deployment challenges
7. ST Edge AI Core STM32Cube.AI
8. Next Steps & Conclusion

life.augmented

Introduction to Artificial Intelligence (AI) & Deep Learning



life.augmented

AI is used today in almost every market segment

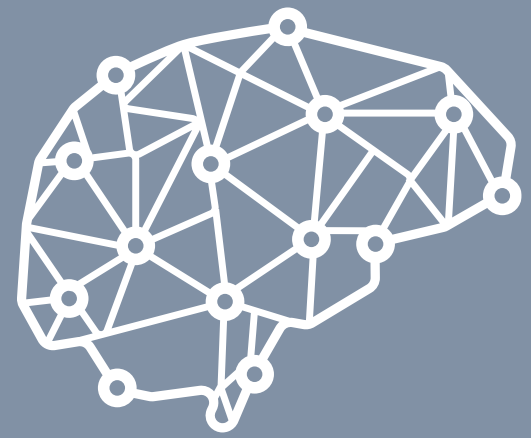
Drones
Flying & landing

Smart Industry
Predictive maintenance

Smart Home
Event detection

Smart Farming
Optimizing conditions

Personal Healthcare
Body measurements



White goods
Smart control

Self-driving Cars
Environment sensing

Smart Buildings
Energy saving

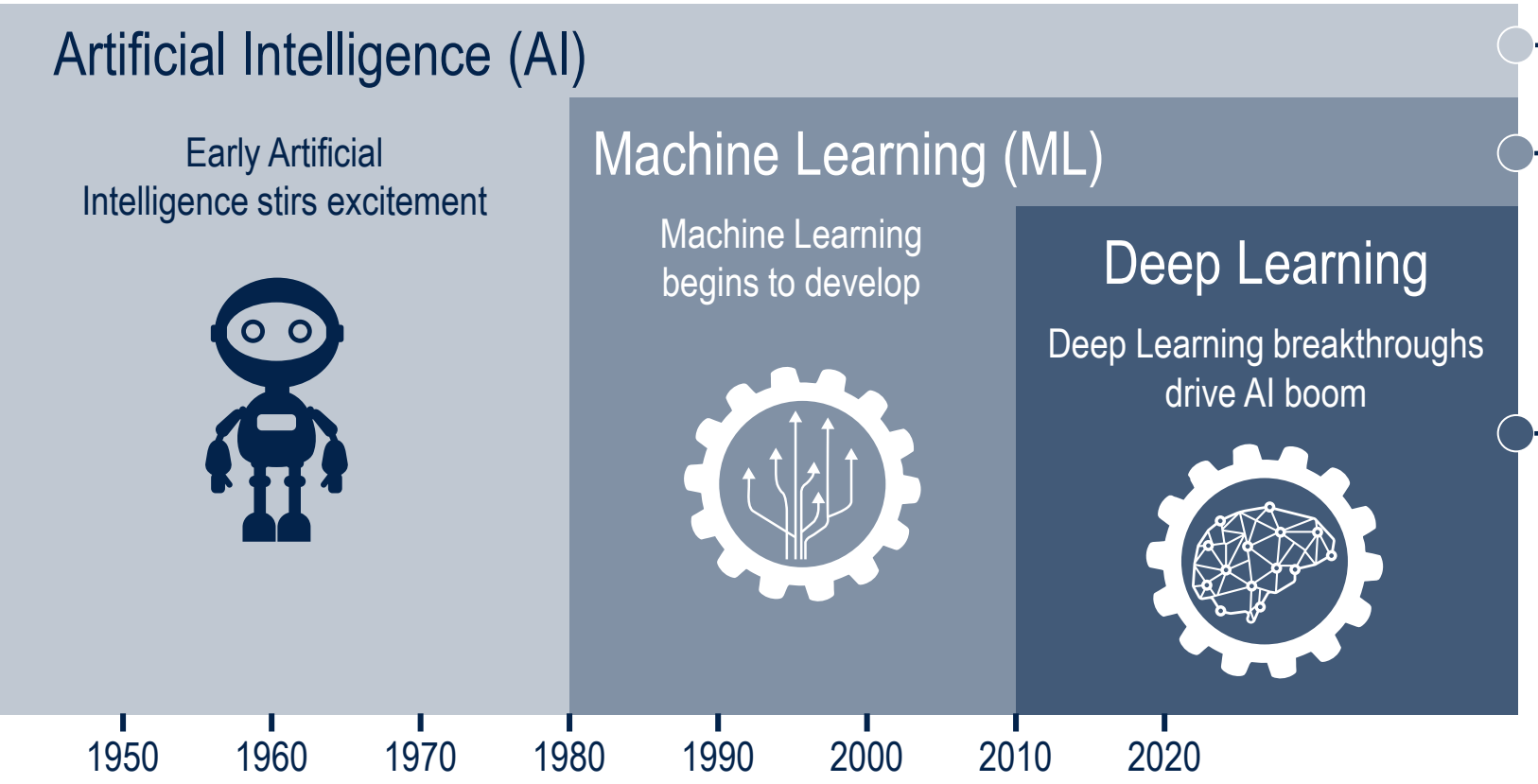
Robots
Collision detection

Personal electronics
Activity recognition

life.augmented

What is AI?

The evolution of AI



Any technique that enables computer to mimic **human behavior**

Subset of AI. Algorithms and methodologies that improve over time through **learning from data**

Subset of ML. Learning algorithms that derive meaning from a **huge amount of data**, by using a hierarchy of multiple layers **that mimic the neural networks of the human brain**

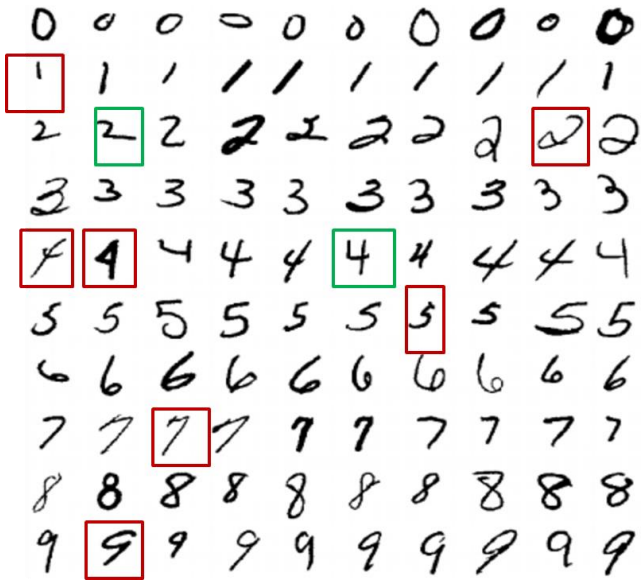
life.augmented

Machine Learning: Why do we need it?

When a **complex task or problem** involves a **large amount of data** and **lots of variables**, but no existing formula or equation can solve it

An example of difficult program

- How to recognize the handwritten digits?
- What makes all these numbers to be identifiable?
- Is there a pattern?
- What is it that makes a 2 to be identified as a 2?



Some examples from MNIST database

(Mixed standard institute for standard and technology)

Standard vs Machine Learning algorithm approach

Standard programming
“A priori” approach

Design algorithm specific for the given problem

Machine Learning
“Empirical” approach

Log of Input Data to the system

+

Desired Output from the system

General ML model trained for the specific problem

Tiny Machine Learning (TinyML) and its benefits

life.augmented

Signals turning into data

Embedded applications will collect more data in the future

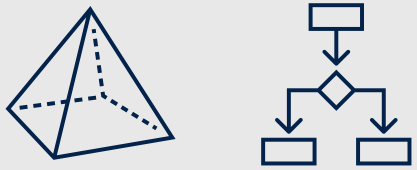
Growing demand for data-driven insights

Increasing use of sensors

Proliferation of IoT devices

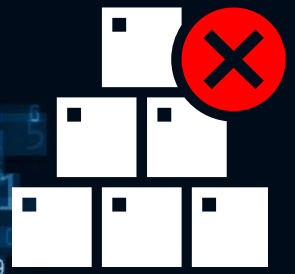
life.augmented

AI offers the best approach to process this growing amount of data



Algorithms and **predefined models** to analyze data and make predictions or decisions

Machine learning algorithms to automatically **learn** patterns and relationships **from data**



Traditional approaches have limitations:

- when dealing with **large datasets**
- when the **phenomena are too complex**

AI-based data processing offers a more flexible and powerful approach to analyzing and making decisions from large data collections

The rise of edge AI

Ultra-low latency
Real-time applications

Reduced data transmission
Generate meaningful information

Enhanced privacy and security
No data sharing in the cloud

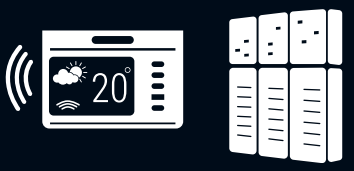
Power efficiency
Low-data / Low-power

Improved accuracy
analyze data from a wide range of sensors and sources

Edge AI benefits many application domains:

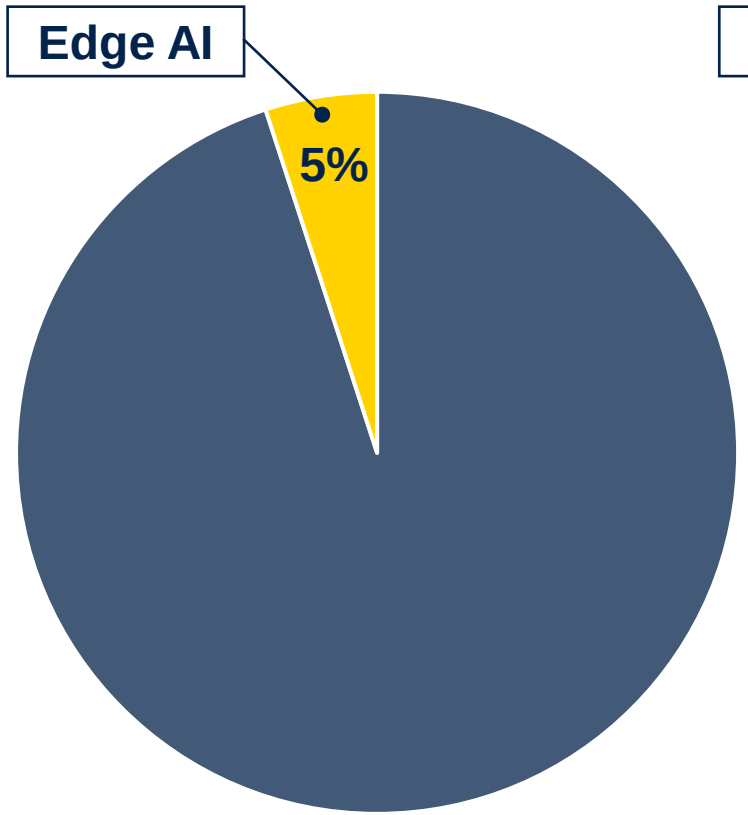
Industrial maintenance
Condition monitoring
Predictive maintenance

Control systems
From home heating systems to industrial machines

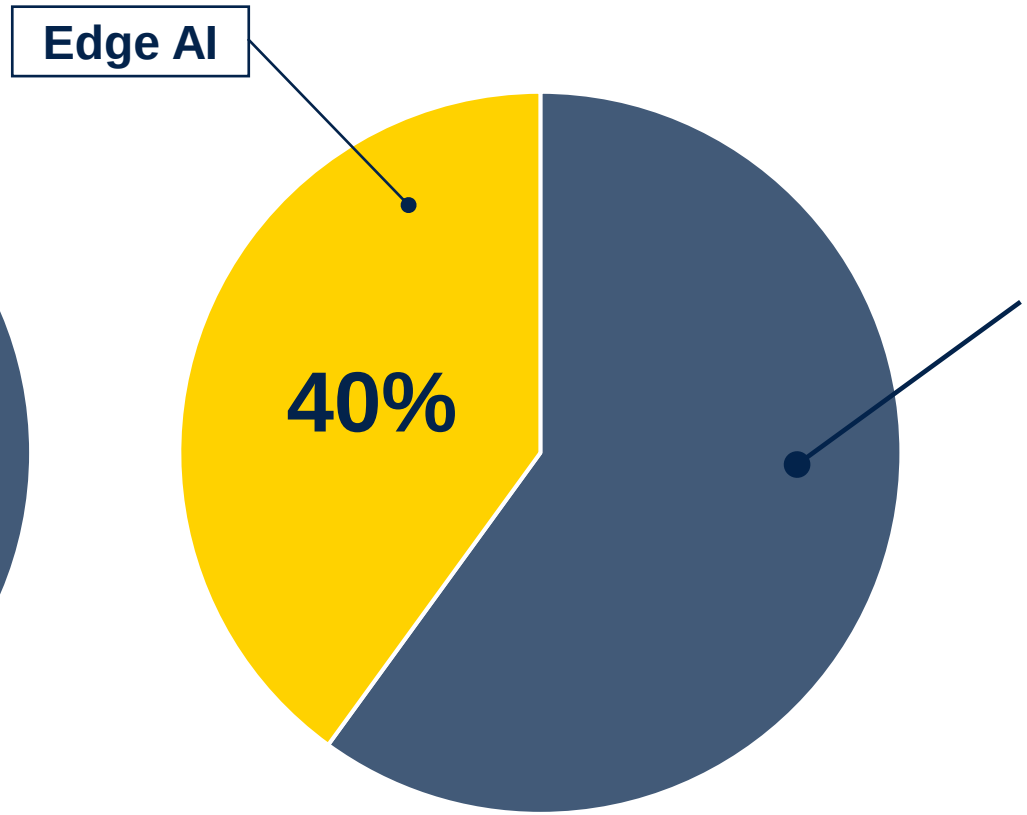


Internet of Things (IoT)
Smart cities, smart buildings, connected homes, and industrial automation

Edge AI adoption is rapidly accelerating



2020

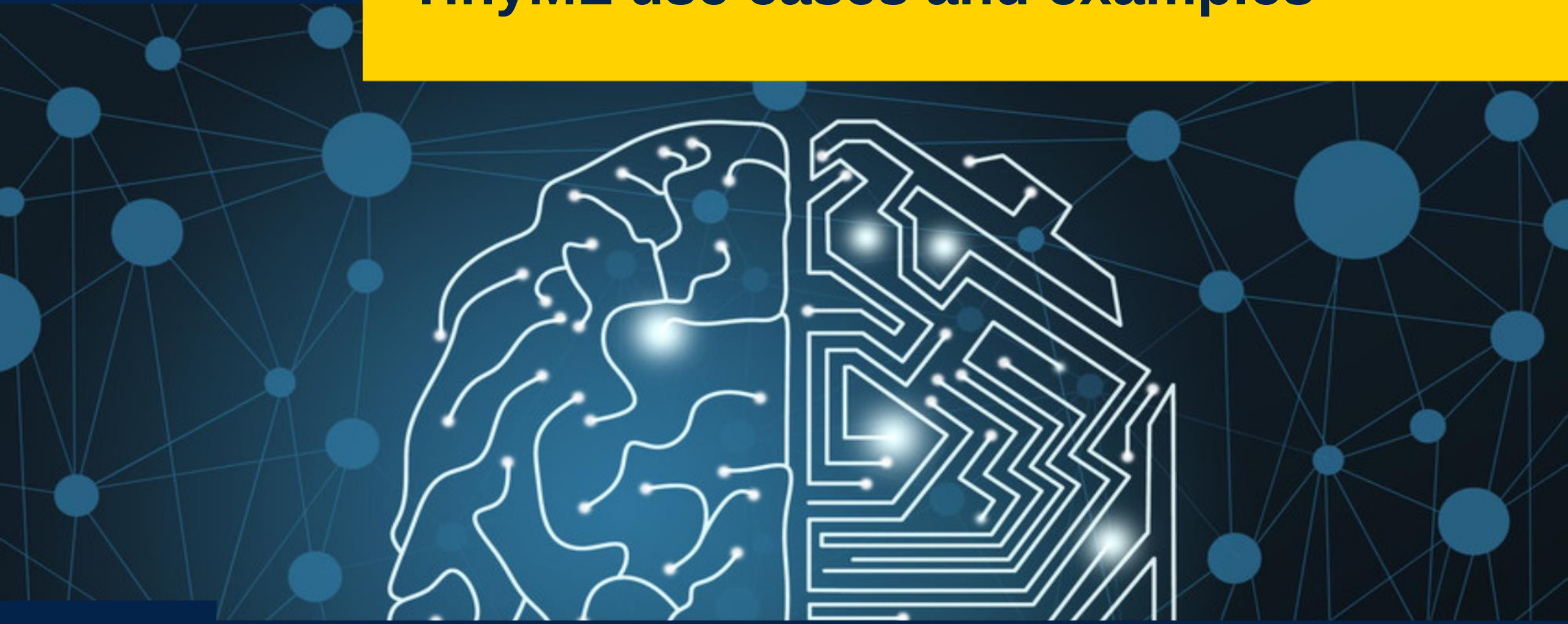


2023

80% of customers who have not yet started are planning **to launch an edge AI project in 2024**

Source: ST customer survey – December 2023

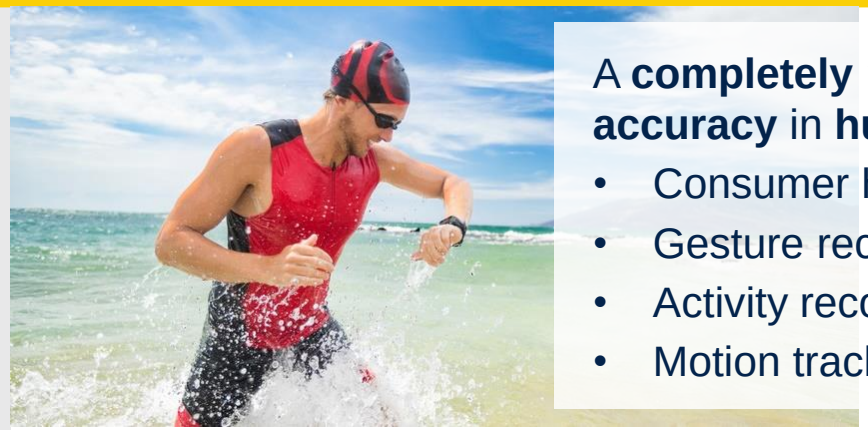
TinyML use cases and examples



life.augmented

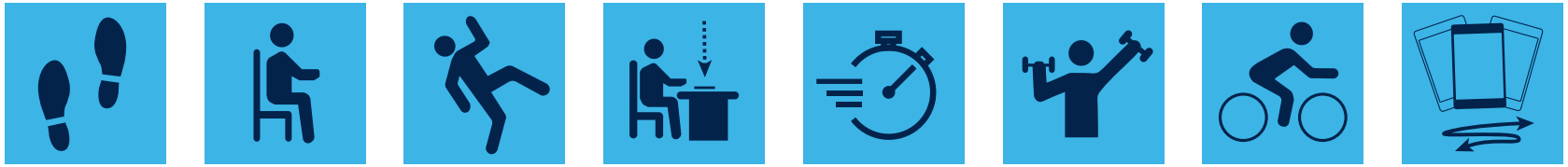
ISPU for personal application

AI at the edge with ultralow power 6-axis IMU for consumer market



A **completely new level of capabilities** and **detection accuracy** in **human activity recognition** applications:

- Consumer health
- Gesture recognition
- Activity recognition
- Motion tracking



Gait analysis | Pose estimation | Fall detection | Carry position | Active time | Fitness activities | Activity recognition | Gesture recognition

And more…

life.augmented

ISPU to approach the Industry 5.0

AI at the edge with ultralow power 6-axis IMU for industrial market

Higher **detection accuracy**, **always on monitoring** in **anomaly detection** applications

- Home alarms
- Robotics
- Condition monitoring

A **completely new level of capabilities** and **detection accuracy** in **asset tracking** applications

life.augmented

AI Solutions on STM32

A full development ecosystem to create your AI application

STM32Cube.AI

AI extension for STM32CubeMX to **map pre-trained Neural Networks**

STM32 **Community** with dedicated Neural Networks topic and **AI expert partners**

Trainings, hands on, MOOCs and partners **videos**

FP-AI-VISION1

Person presence detection

Food classification

FP-AI-SENSING1

People activity recognition

Audio scene classification

FP-AI-NANOEDG1

Condition-based monitoring

life.augmented

Automotive Edge AI Solutions

AI at the edge with Stellar E microcontroller

Better efficiency & maintenance

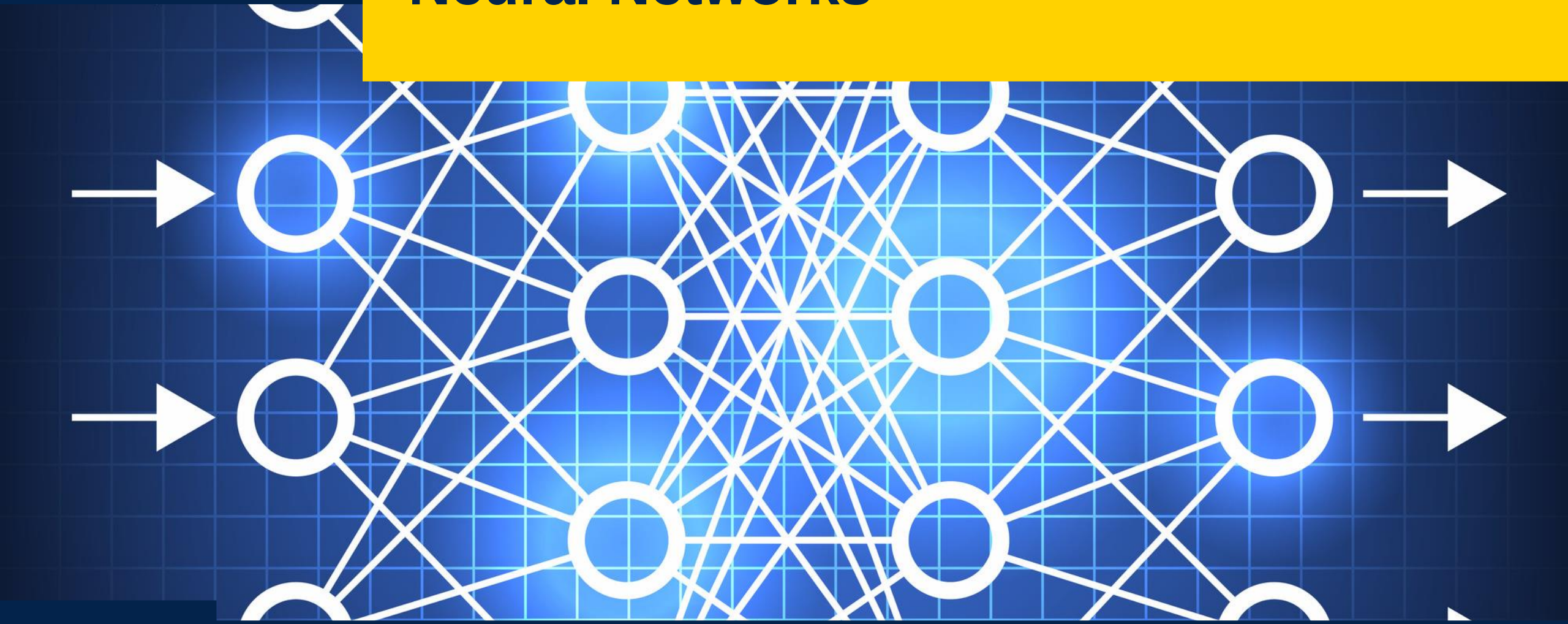
- Protecting critical functions
- Improving energy management
- Enabling predictive maintenance

A personalized driving experience

- optimize comfort and convenience
- power infotainment systems

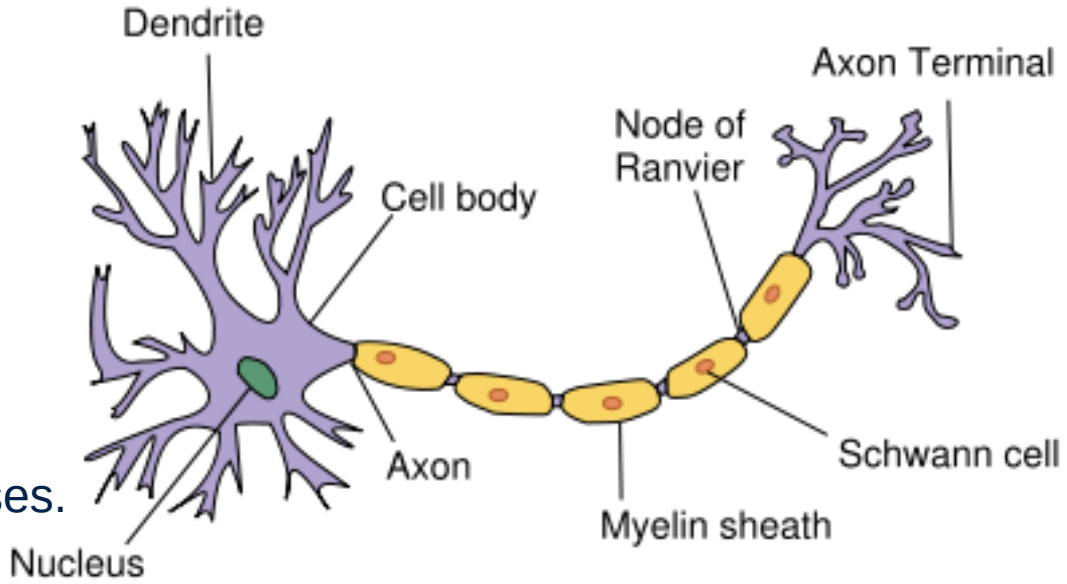
life.augmented

Neural Networks



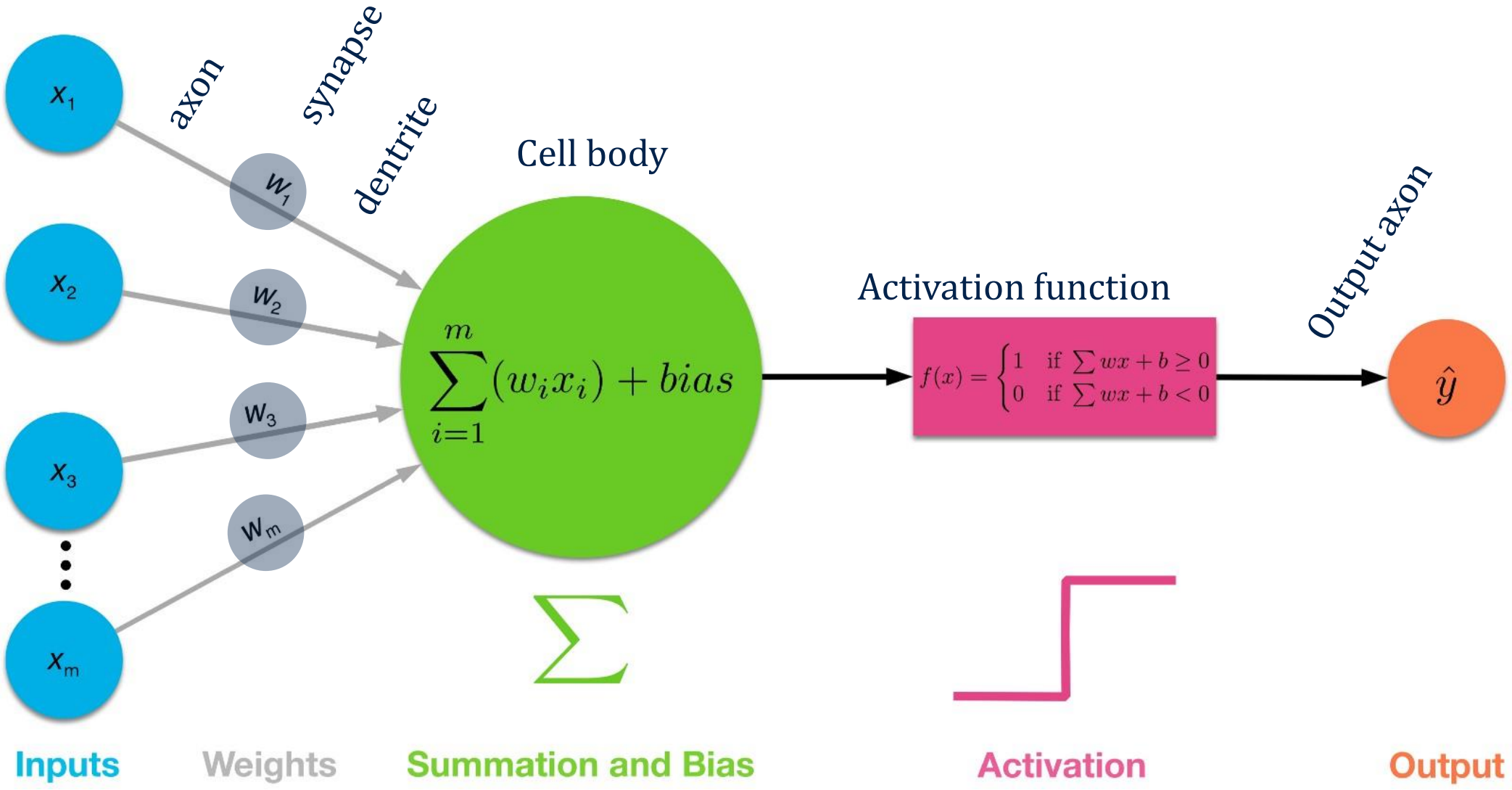
What are Neural Networks?

- Also referred to as Artificial Neural Networks.
- Inspired by human neural system.
- Human neuron has three main components
  - Dendrites
    - Take inputs from other neurons in terms of electrical pulses.
  - Cell body
    - Makes the inferences and decides the actions to take.
  - Axon terminals
    - Send the outputs to other neurons in terms of electrical pulses.



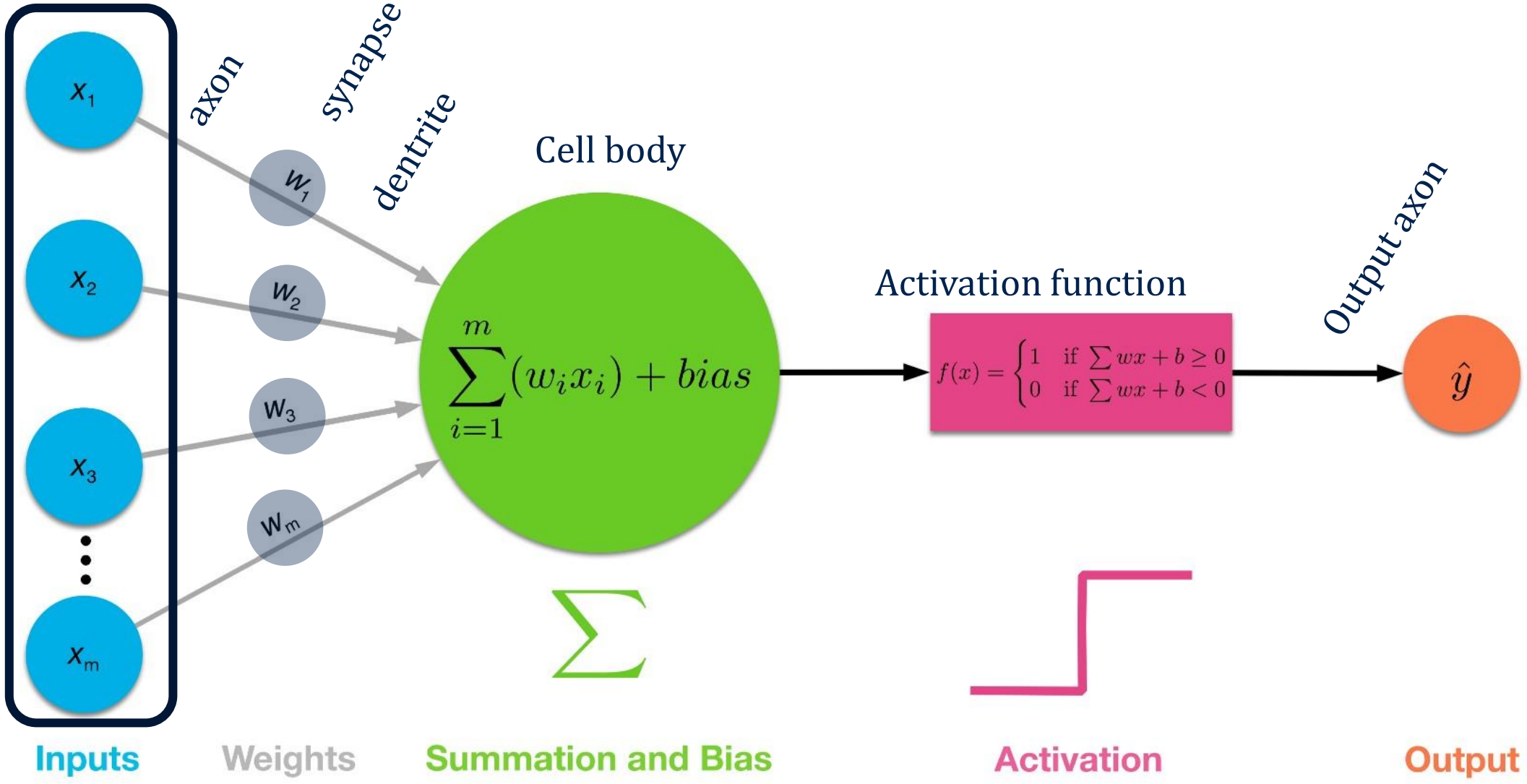
Artificial Neuron

- The heart of a neural network



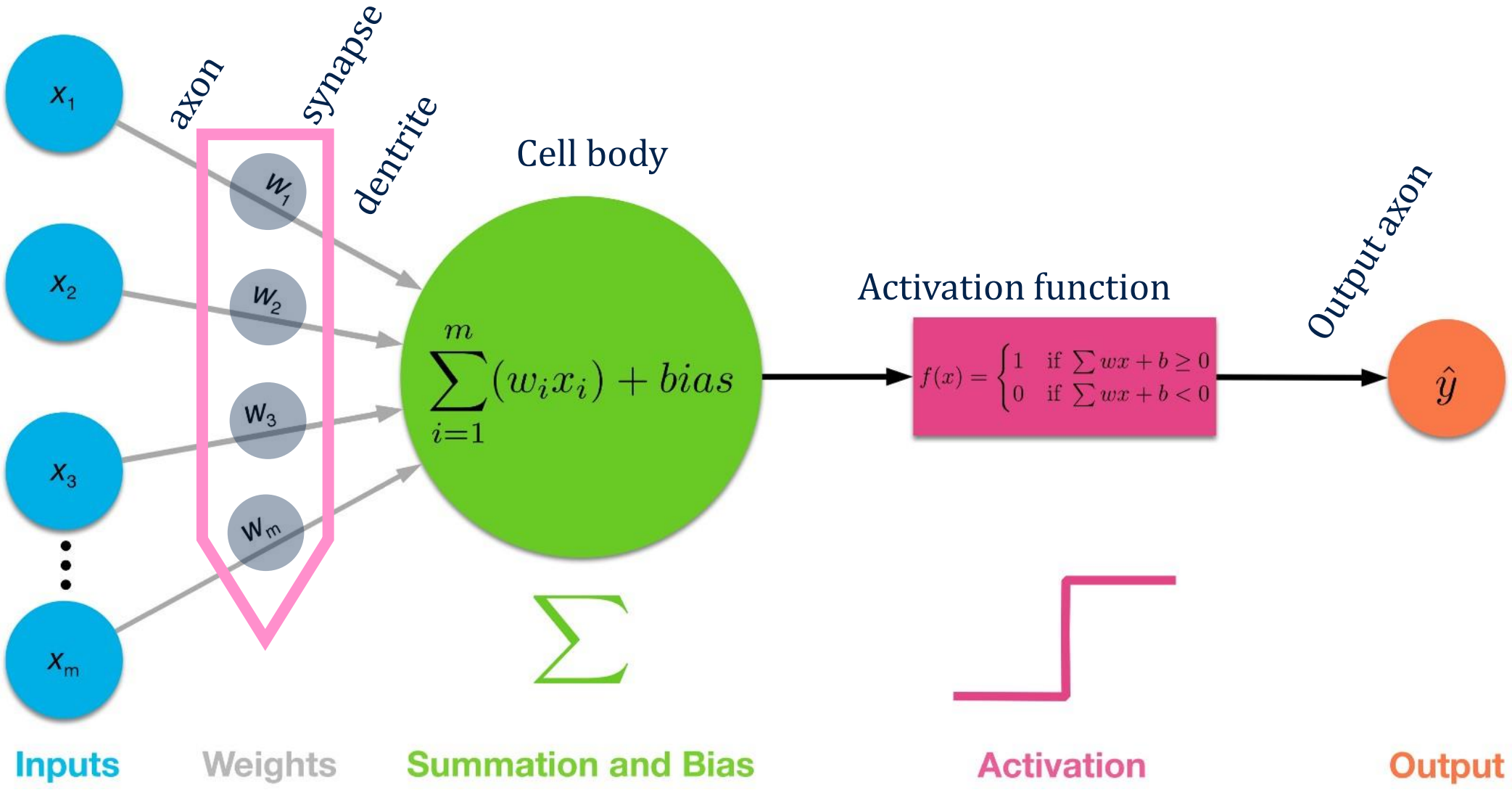
Artificial Neuron

- The heart of a neural network



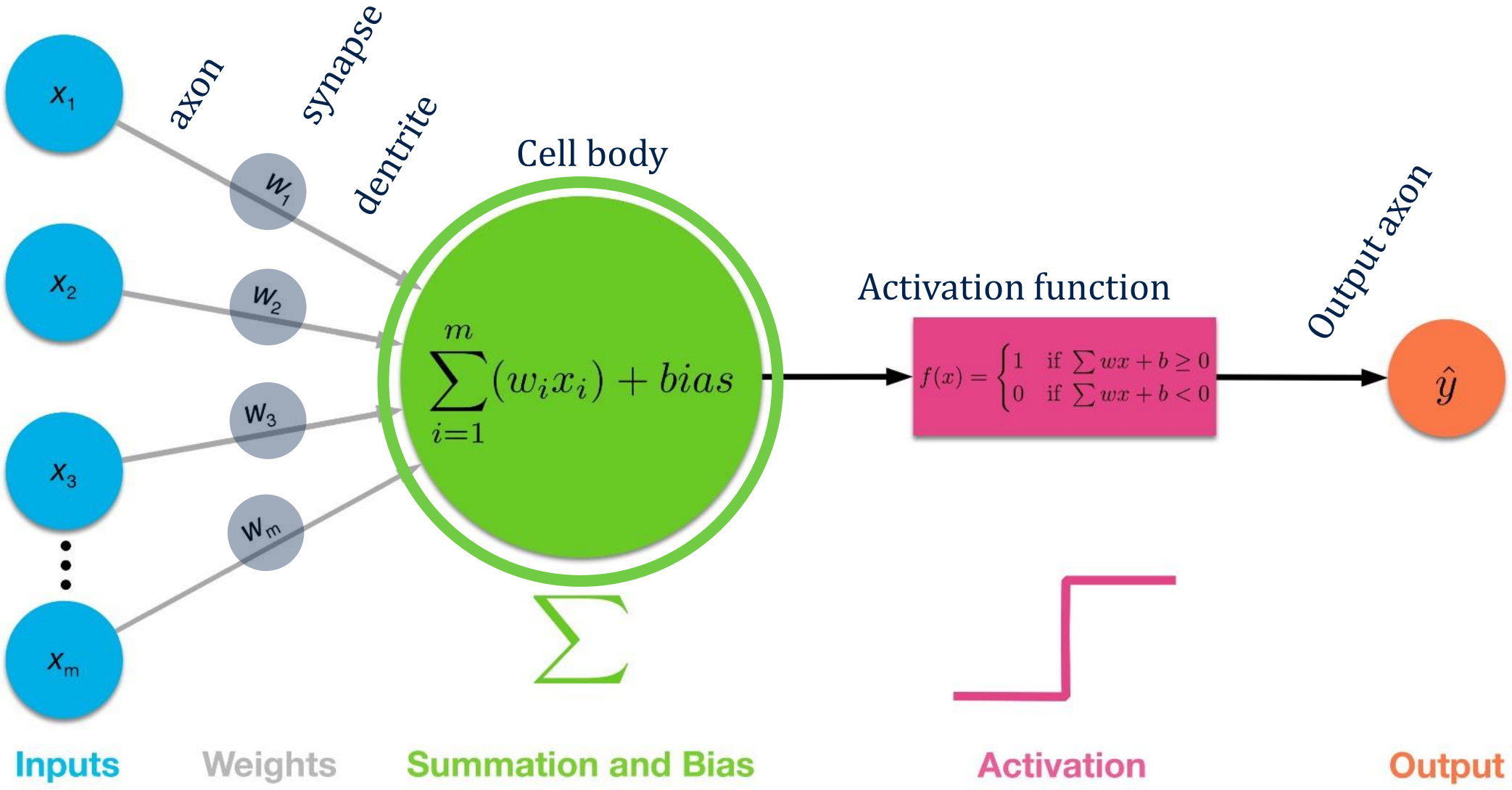
Artificial Neuron

- The heart of a neural network



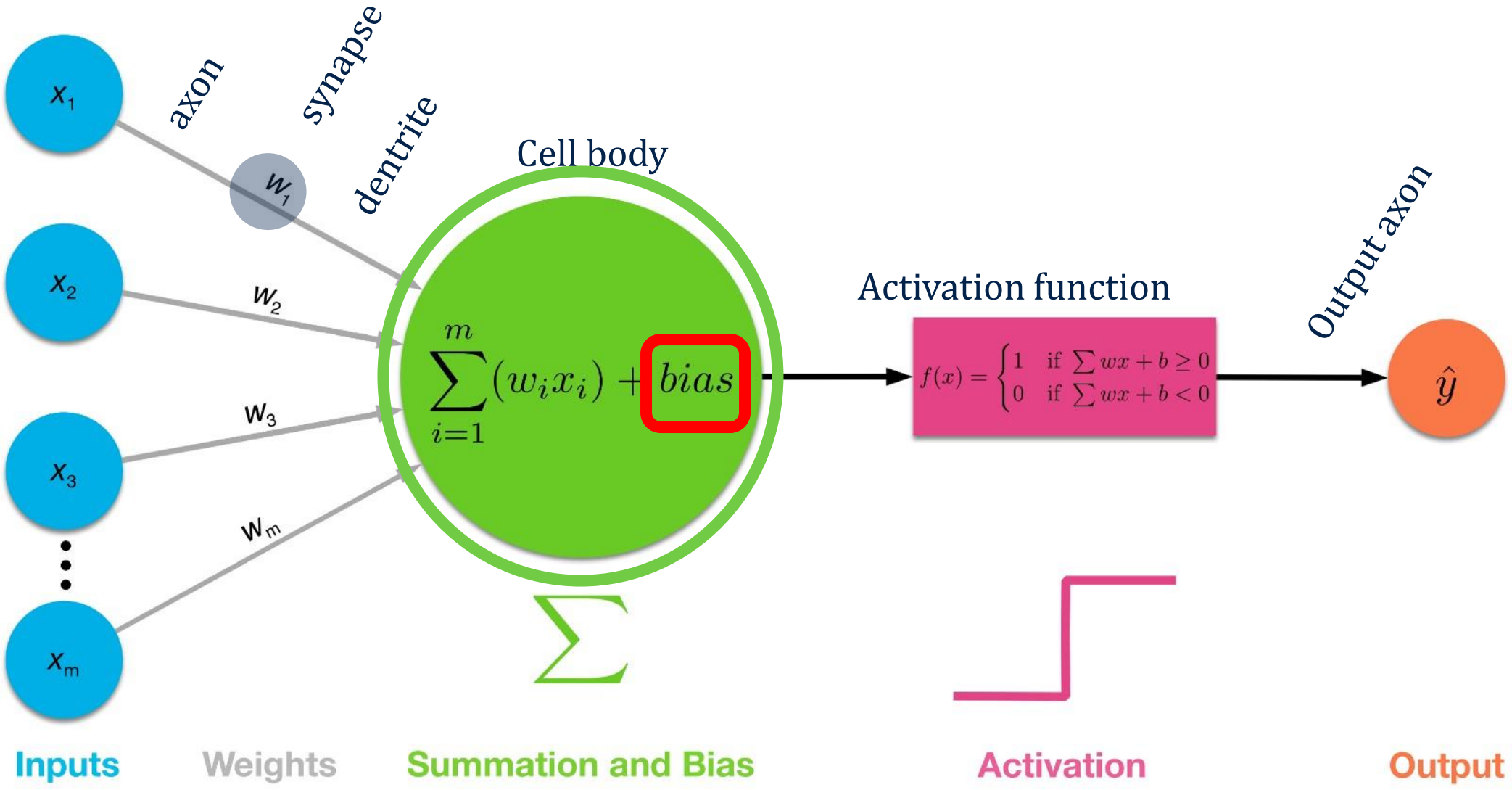
Artificial Neuron

- The heart of a neural network



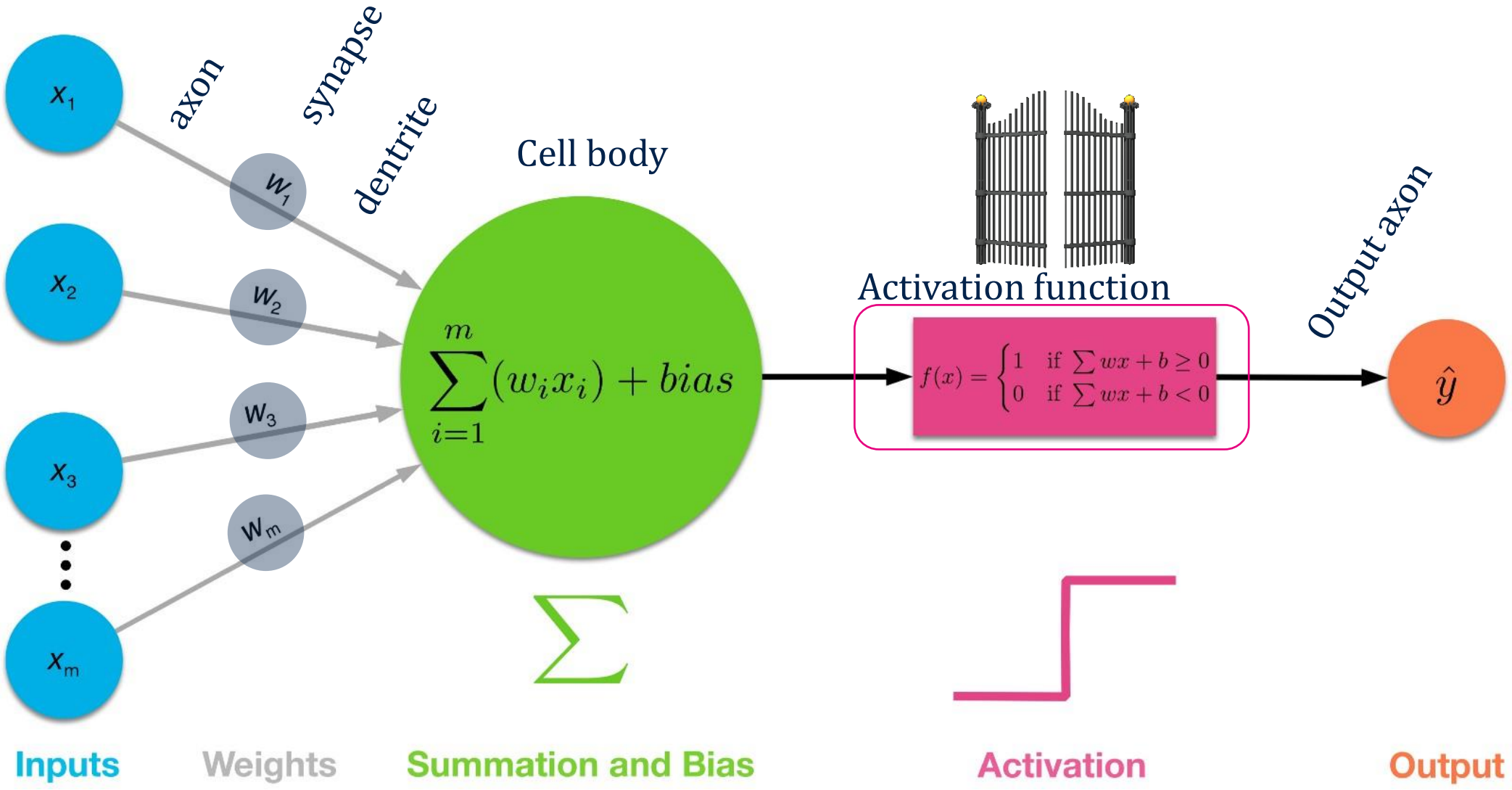
Artificial Neuron

- The heart of a neural network



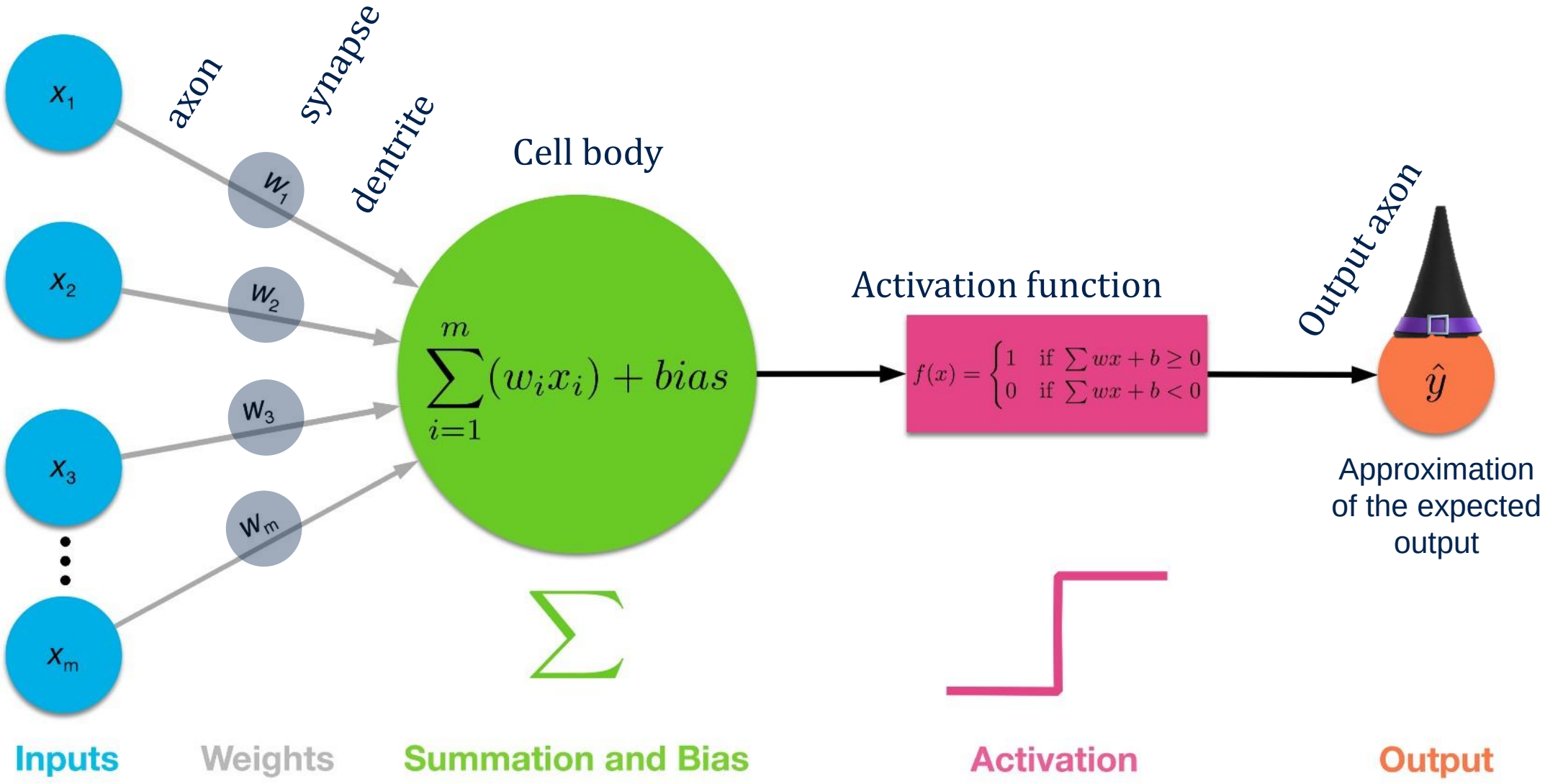
Artificial Neuron

- The heart of a neural network



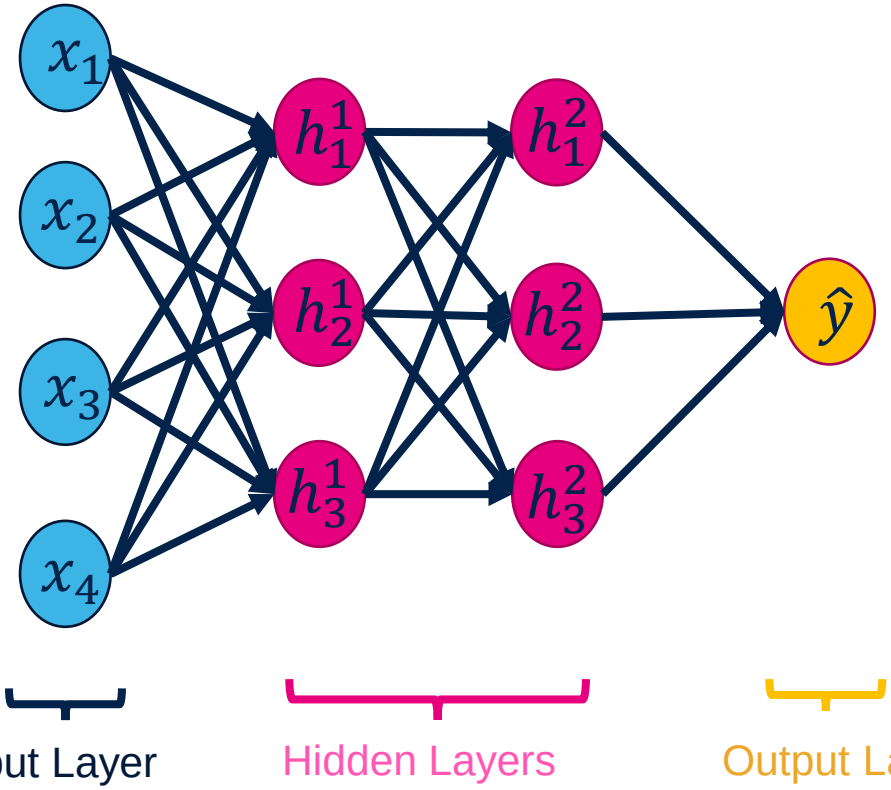
Artificial Neuron

- The heart of a neural network



Neural Networks

- Notations

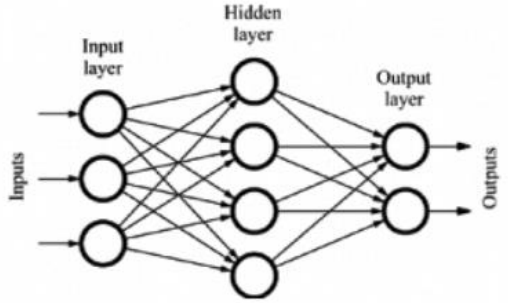


$$x_n, n = \{1, 2, 3, \dots\}$$

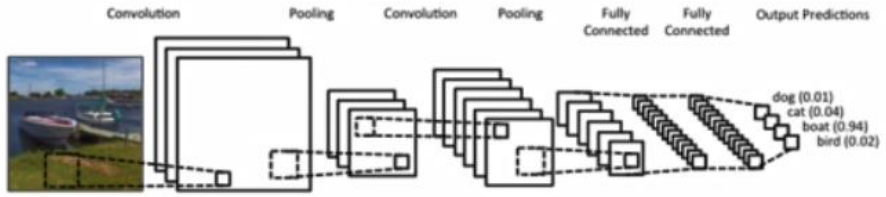
$$h_p^m, m = \{1, 2, 3, \dots\}, p = \{1, 2, 3, \dots\}$$

Different types of Neural Networks

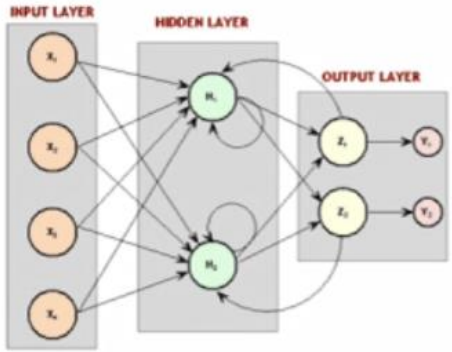
- Multi Layer Perceptrons



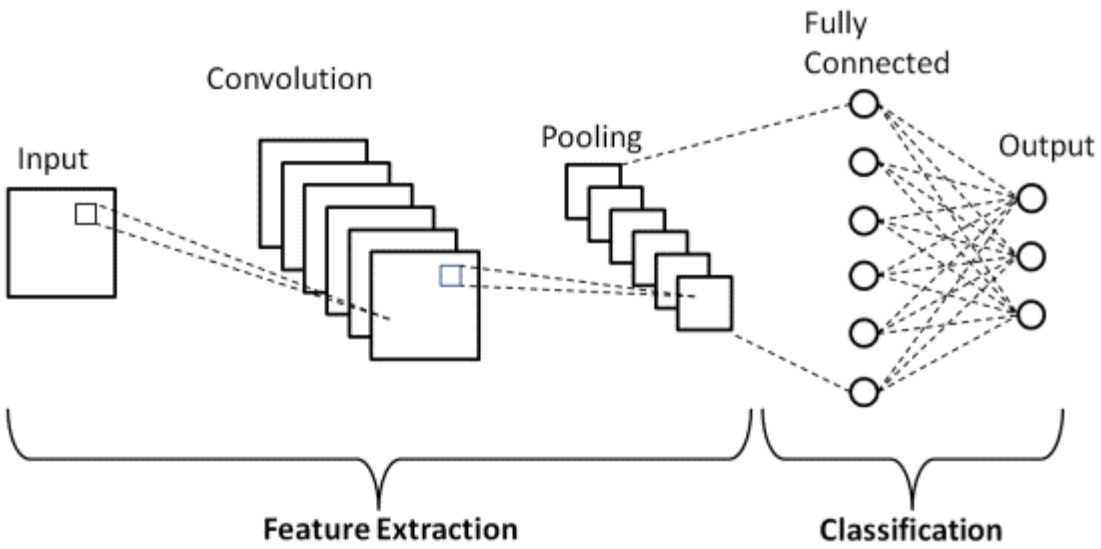
- Convolutional Neural Networks

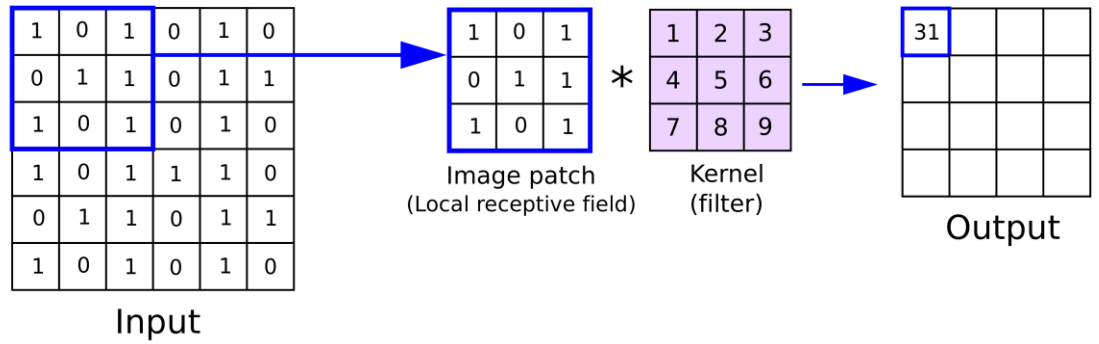


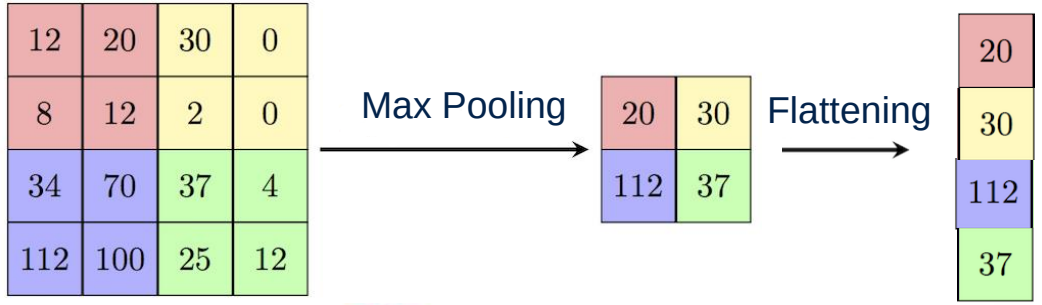
- Recurrent Neural Networks

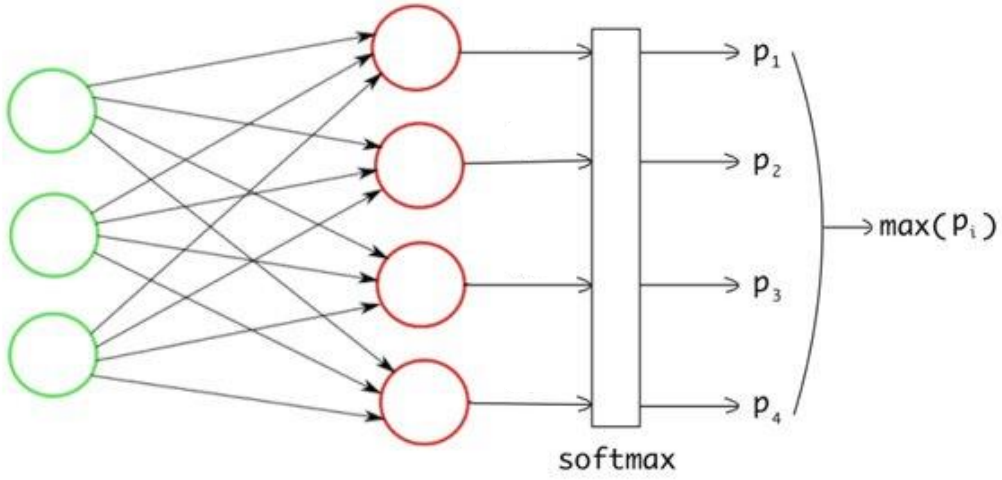


Convolutional Neural Network



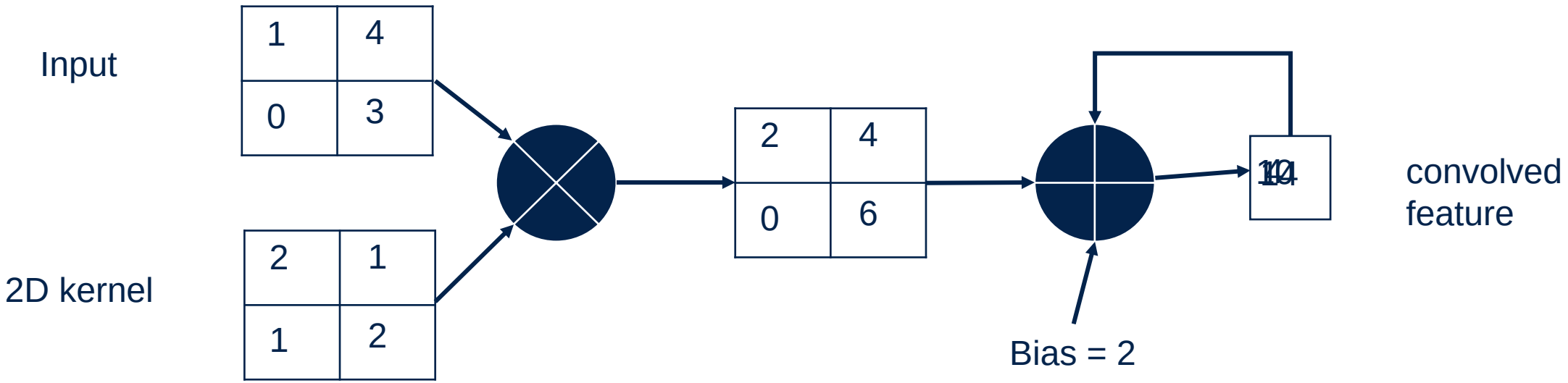






life.augmented

2D Convolutions



3D Convolutions

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

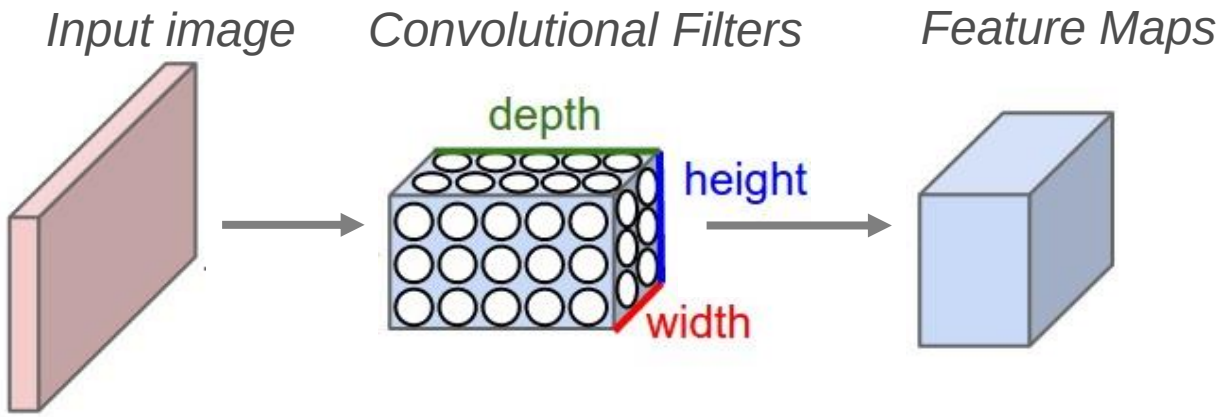
o[:,:,0]

2	3	3
3	7	3
8	10	-3

o[:,:,1]

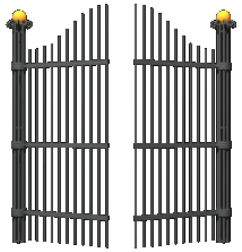
-8	-8	-3
-3	1	0
-3	-8	-5

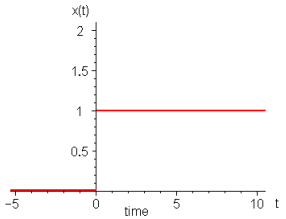
toggle movement

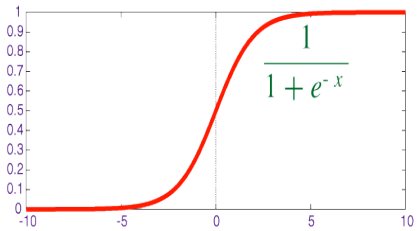


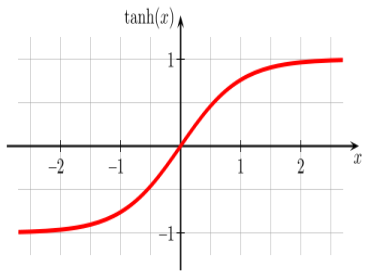
Most widely used activations

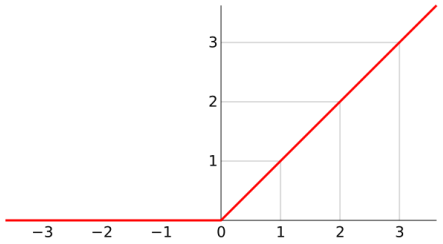
- Unit step
  - Threshold
- Sigmoid Function
  - Like a step function but smoother
  - Best to predict probabilities
- Tan hyperbolic
  - Stretched out version of the sigmoid function
- ReLU
  - Computationally efficient







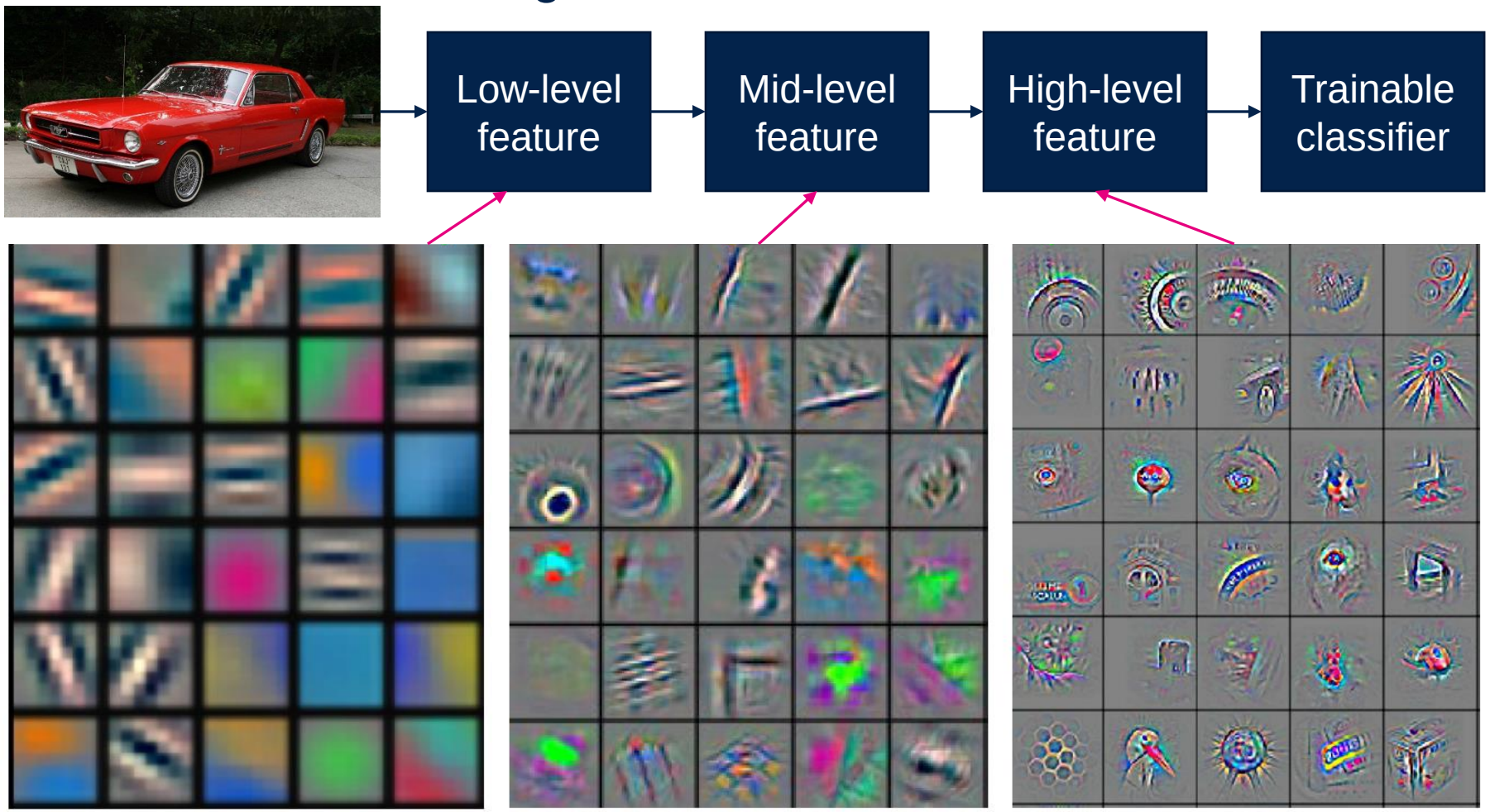




- **Function choice depends on the characteristics of the data.**
- **For example Sigmoid Function works good for classification purposes, resulting in Faster training and convergence.**
- **ReLU is good for approximation. As it is simple so always start from this if you don't know the data characteristics. Helps against gradient vanishing**
- **We can also define custom activations.**

Learning hierarchical representations

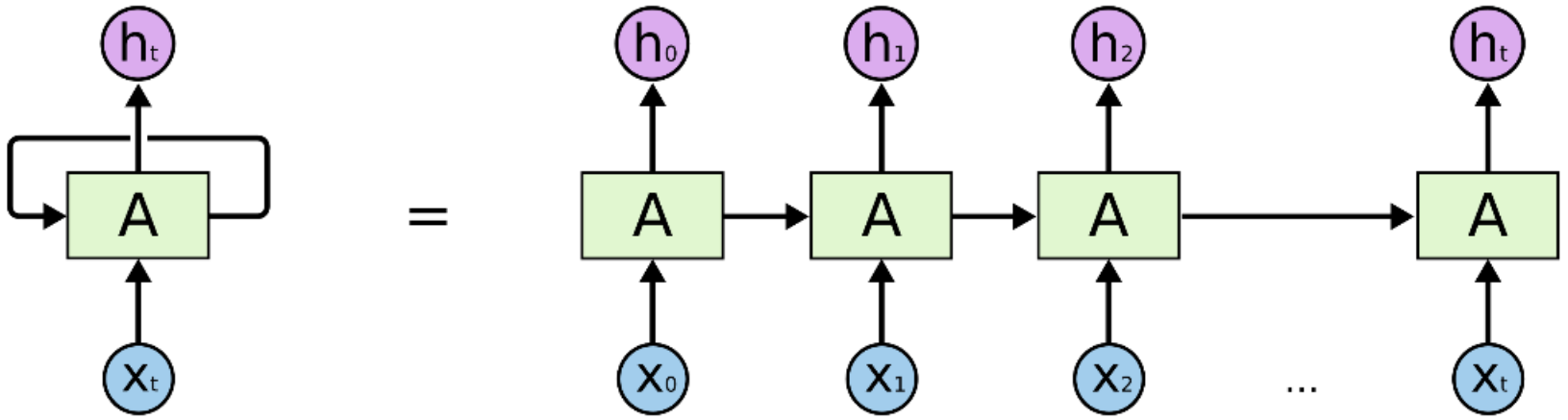
- More than one stage of non-linear feature transformation



Feature visualization on convolutional net on ImageNet [Zeiler & Fergus 2013]

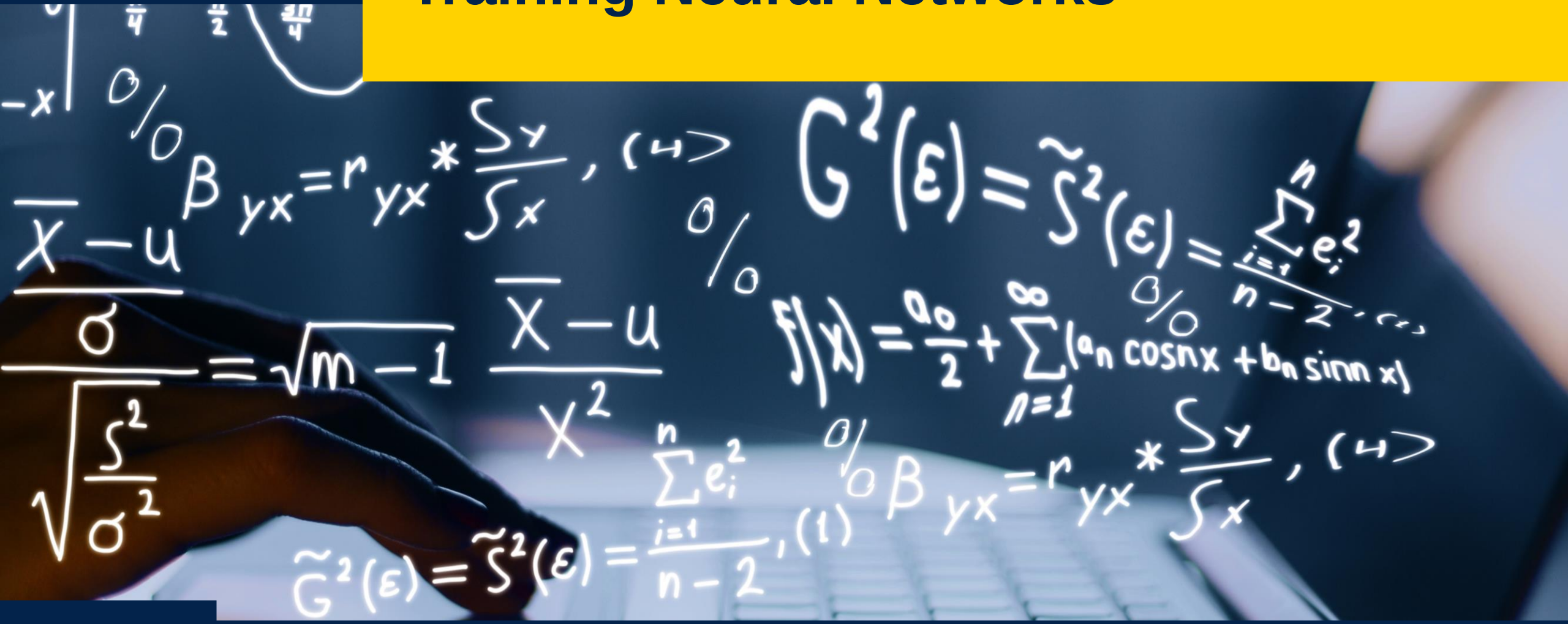
Recurrent Neural Networks

- Convolutional neural networks have no internal state persistence
- Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.



Unrolling recurrent neural network loops

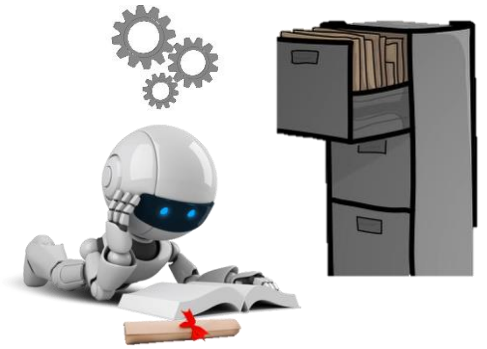
Training Neural Networks

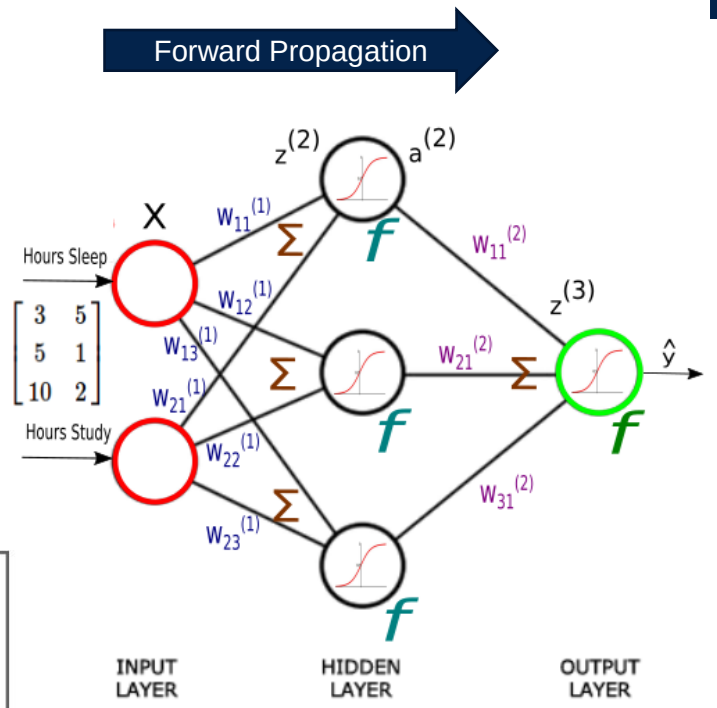


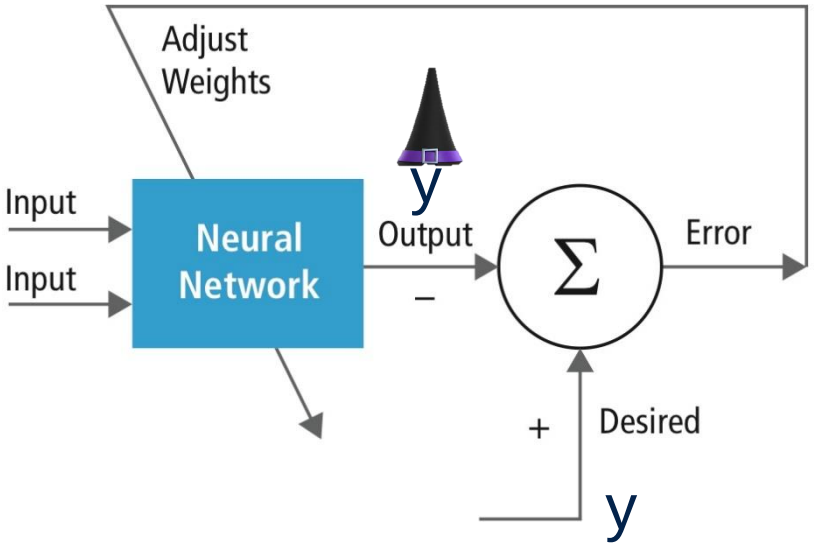
life.augmented

Training neural networks

- In supervised learning an assumption is to have a relatively large labeled dataset.
- Feed all the samples as inputs to get an output. Called forward propagation or inference run outputs.
- At start the weights can be randomized or predefined depending on the applications scenario.
- The result $\hat{y}$ is compared with ground truth output y.
- The task is to make the output value $\hat{y}$ to be as close to y as possible reducing the error expressed as Loss functions $L(\tilde{y}, y)$.

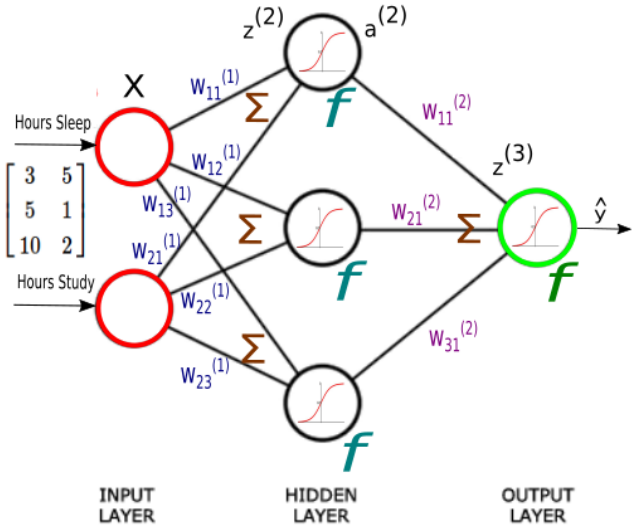


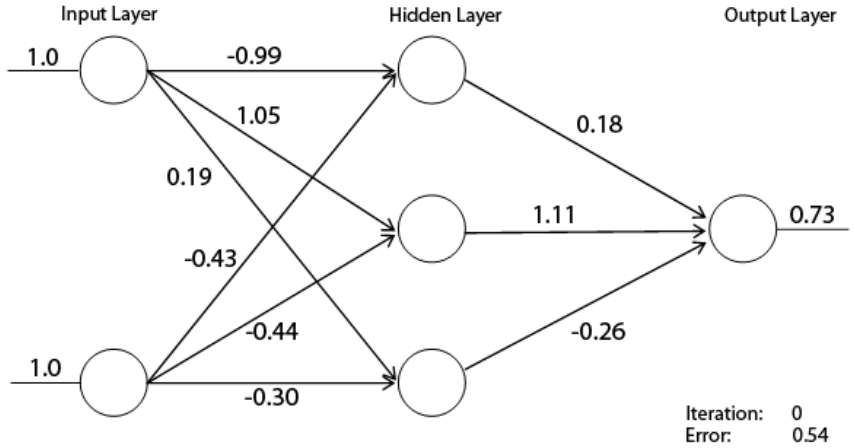




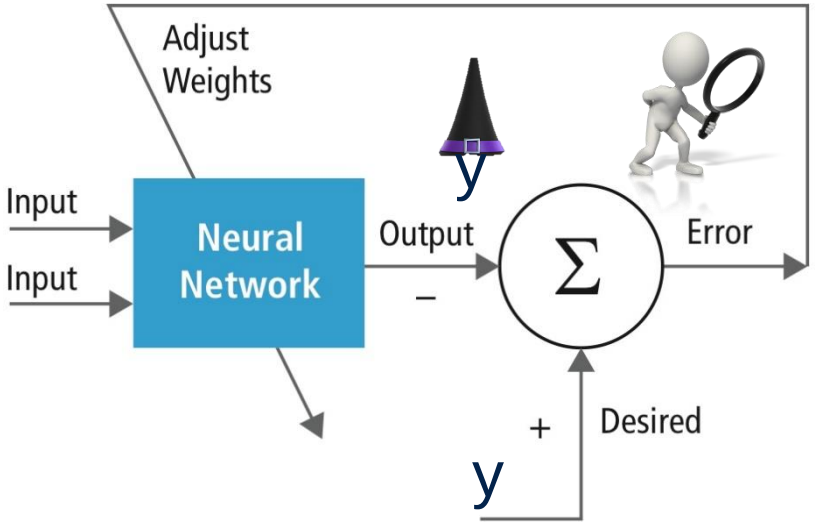
$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{c}) + c - y)^2$$

Training neural networks





Back Propagation



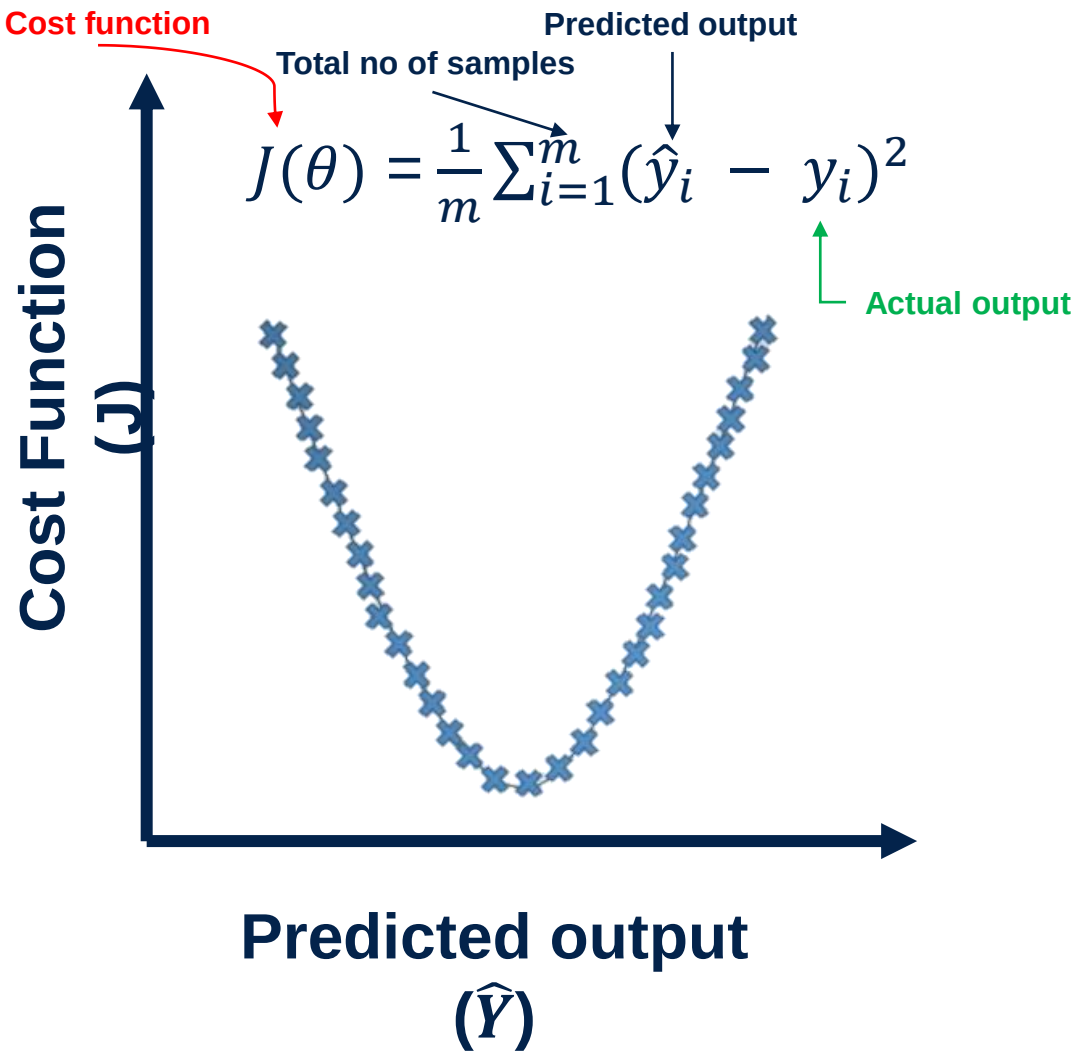
- Go back and adjust the weights slowly. Aim is

$$Error_T < Error_{T-1}$$

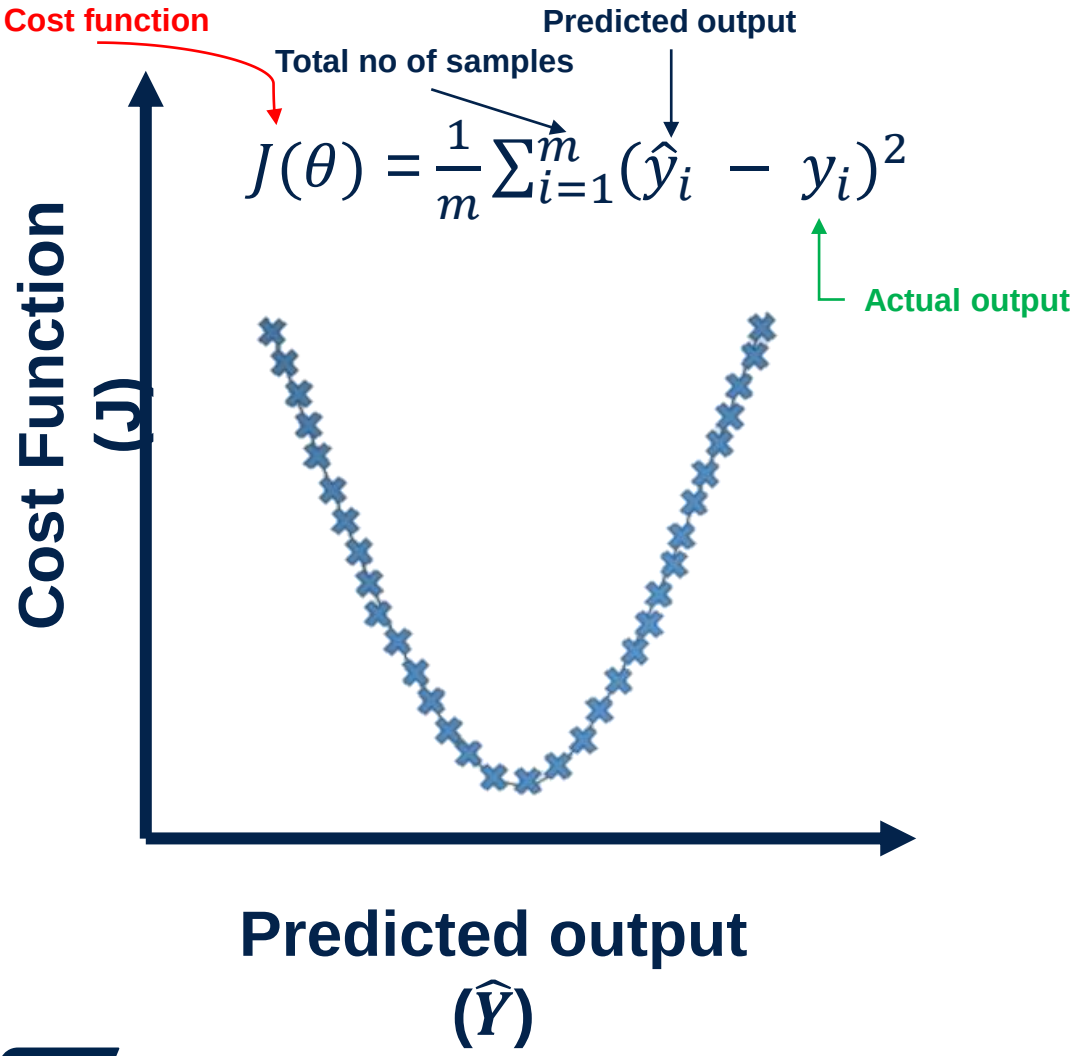
- Repeat this process until the error we get is very small.

$$\lim_{\in \to 0} Error_T < \in$$

Backpropagation



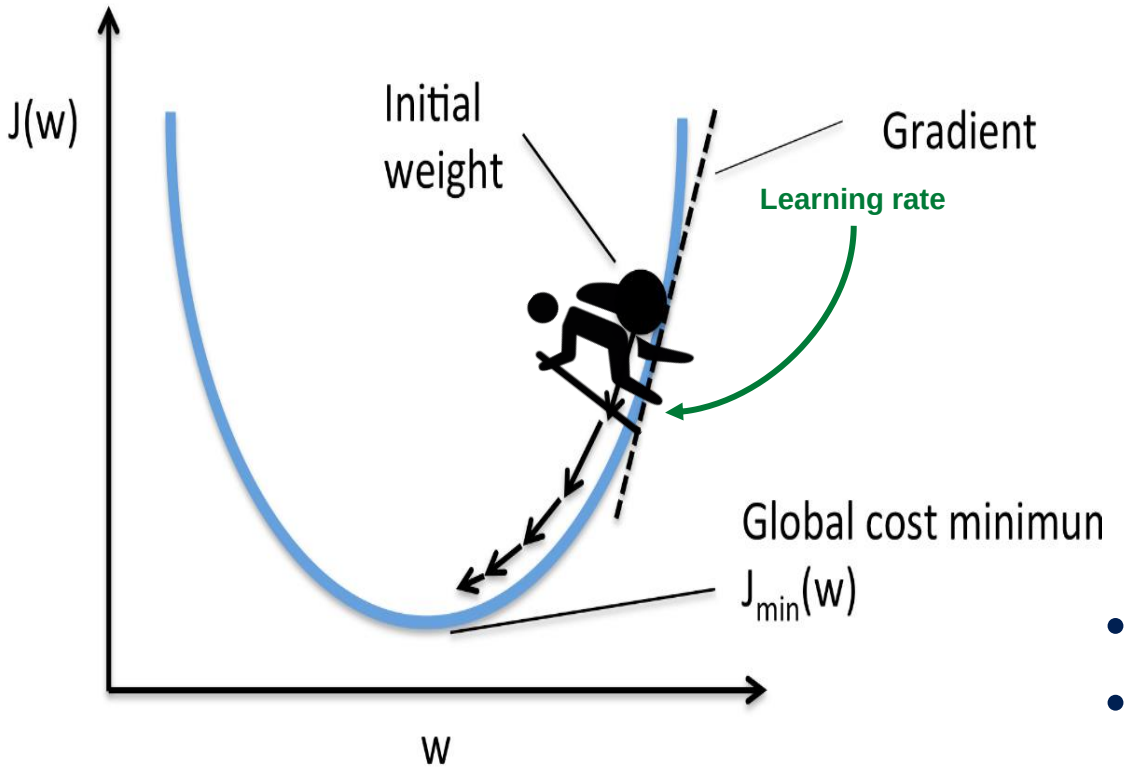
Backpropagation

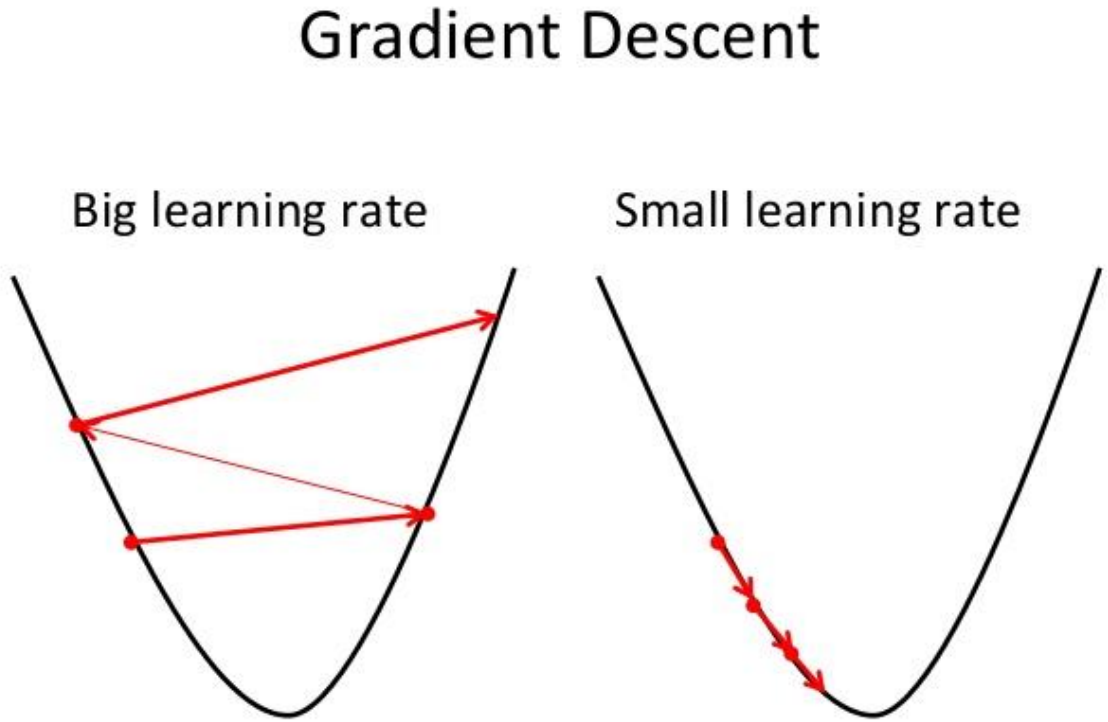


- Brute force
  - Try all the possible combination of weights.
  - Plot the cost function.
  - Use the weights which result in smallest error.
  - Sounds simple but will take too much time !!!

Backpropagation

- Enters the gradient descent





Learning rate compromise

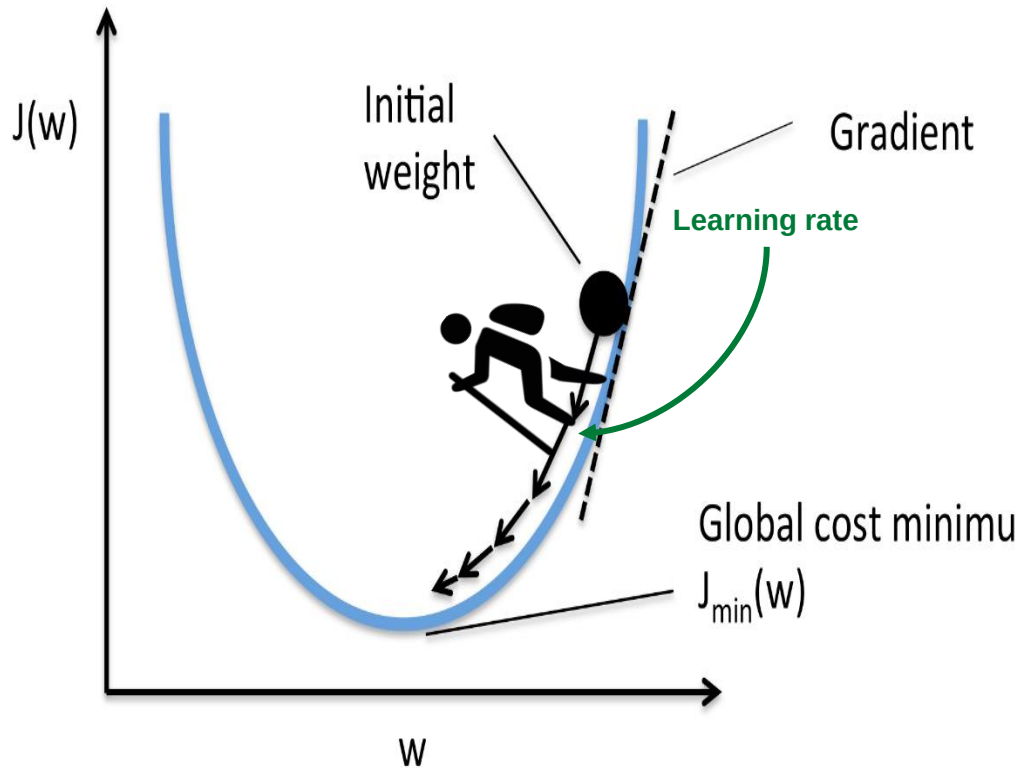
- **Big learning rate**
- **May never converge.**

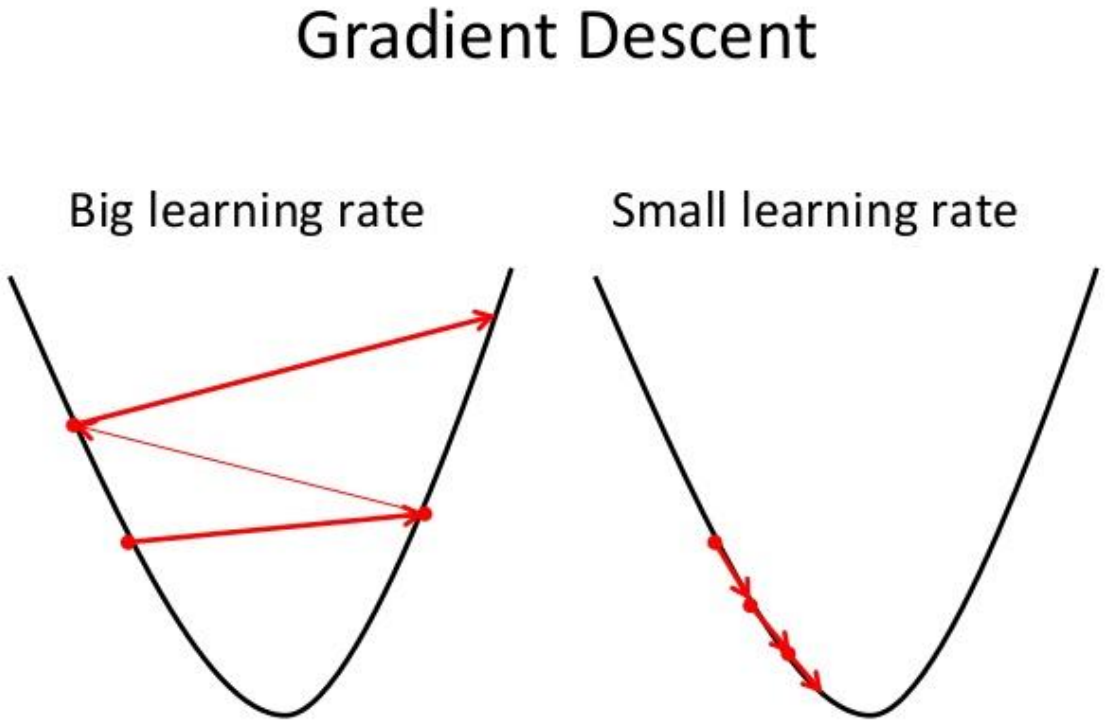
- **Small rate.**
- **Will converge for sure.**
- **A lot of small steps.**

life.augmented

Backpropagation

- Enters the gradient descent

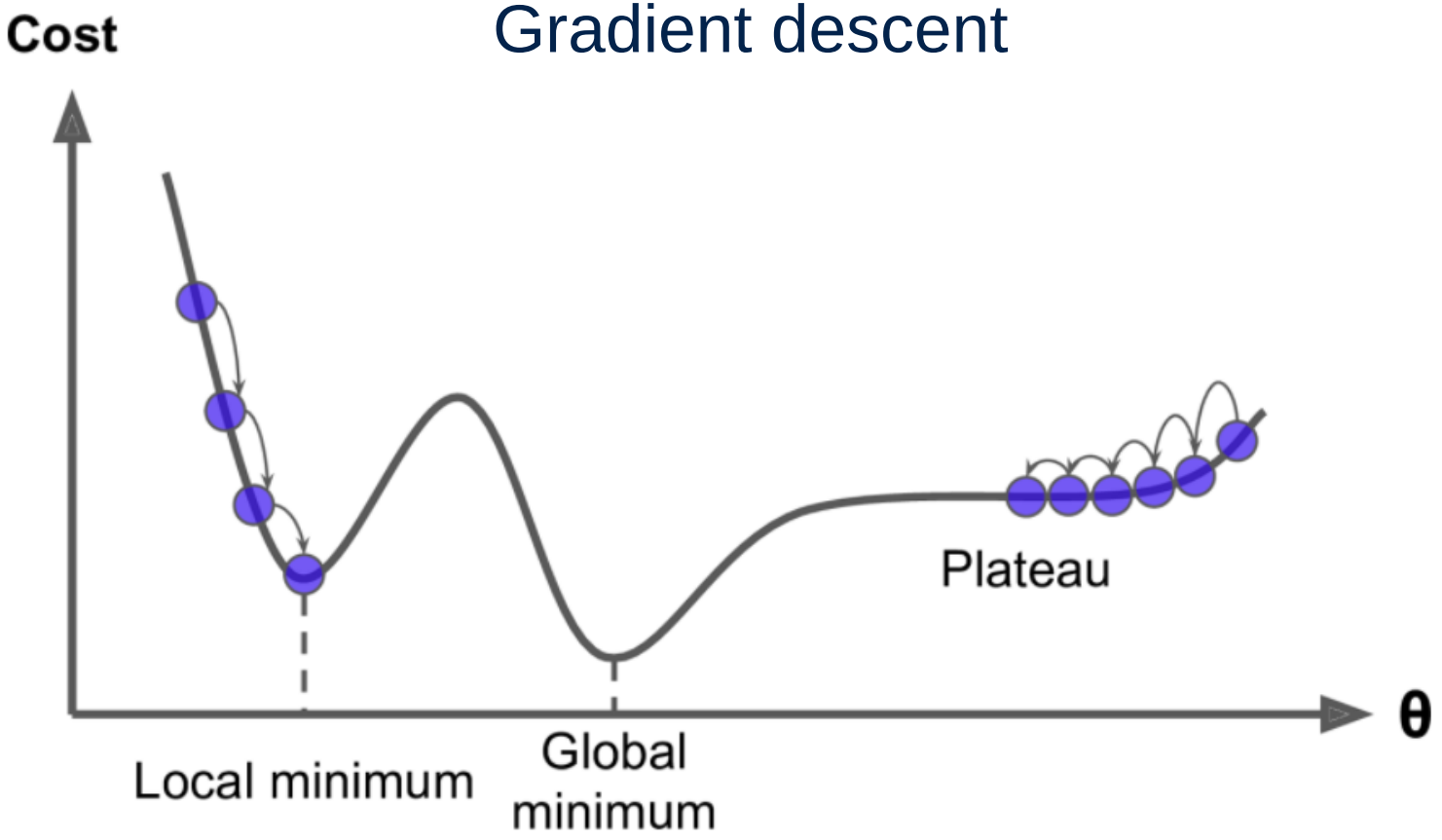




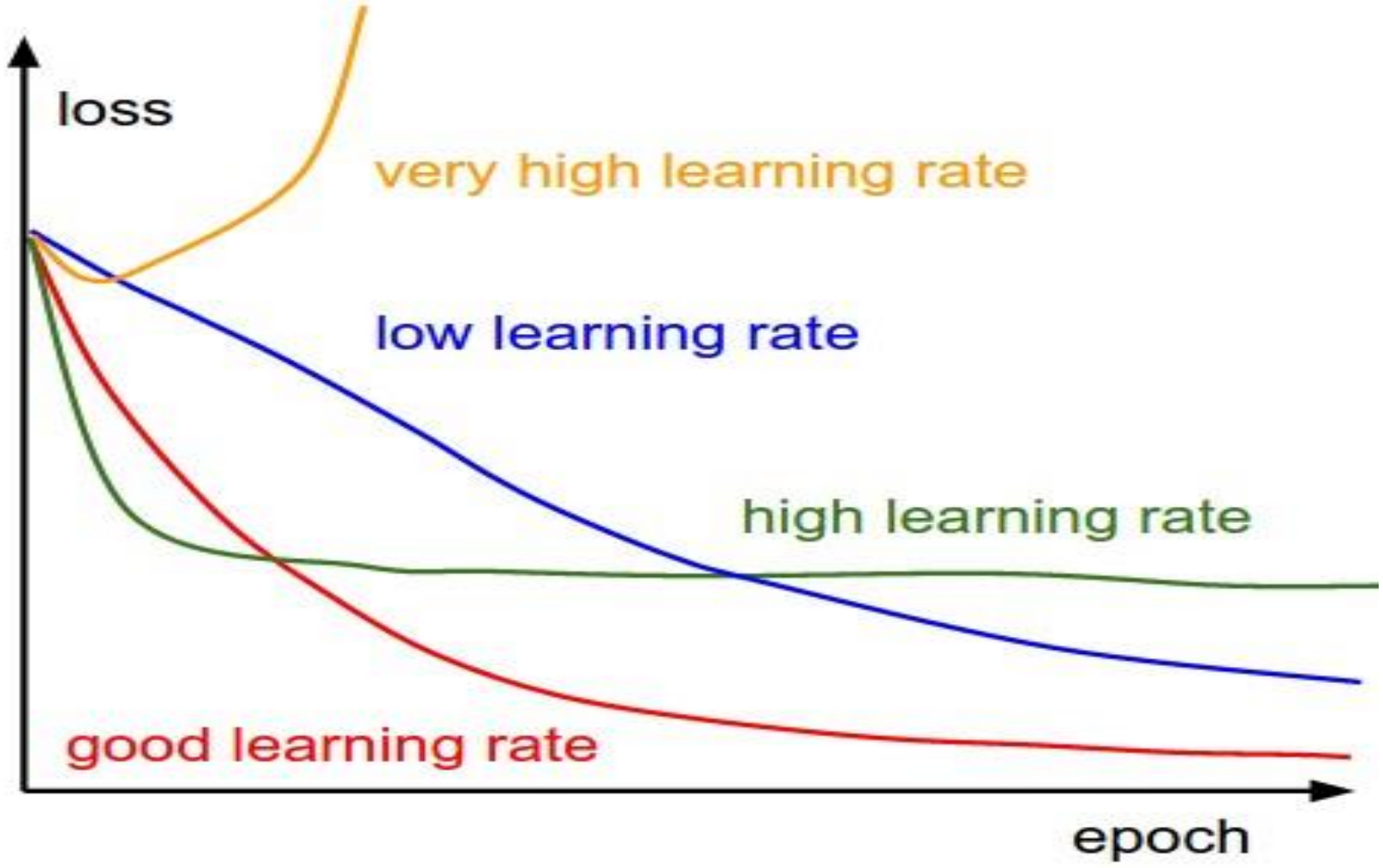
Use adaptive learning rate!

Backpropagation pitfall

Gradient descent



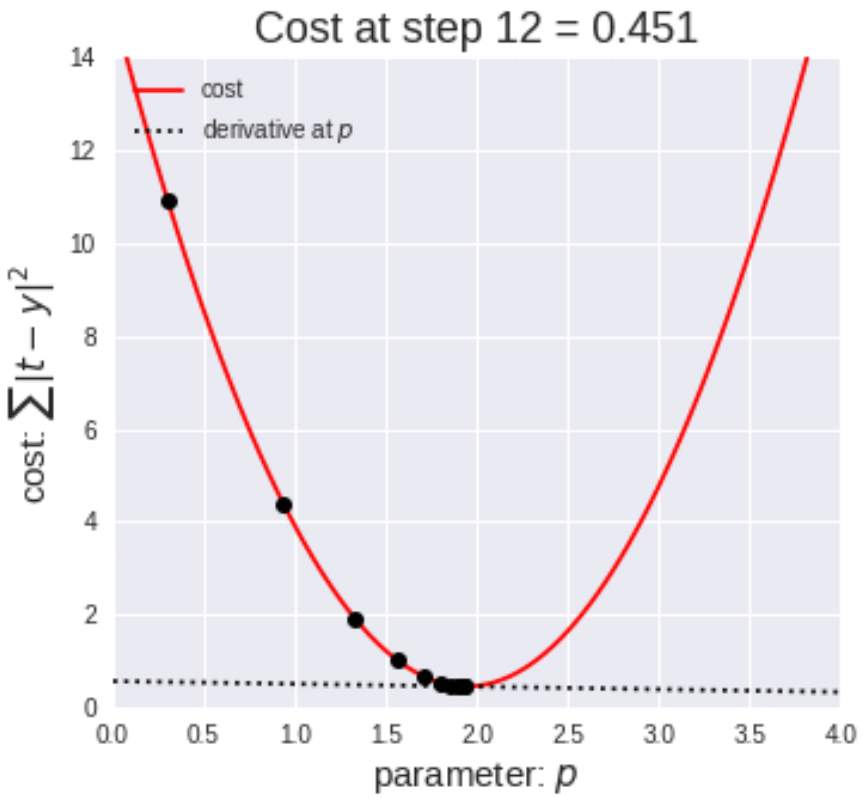
Backpropagation

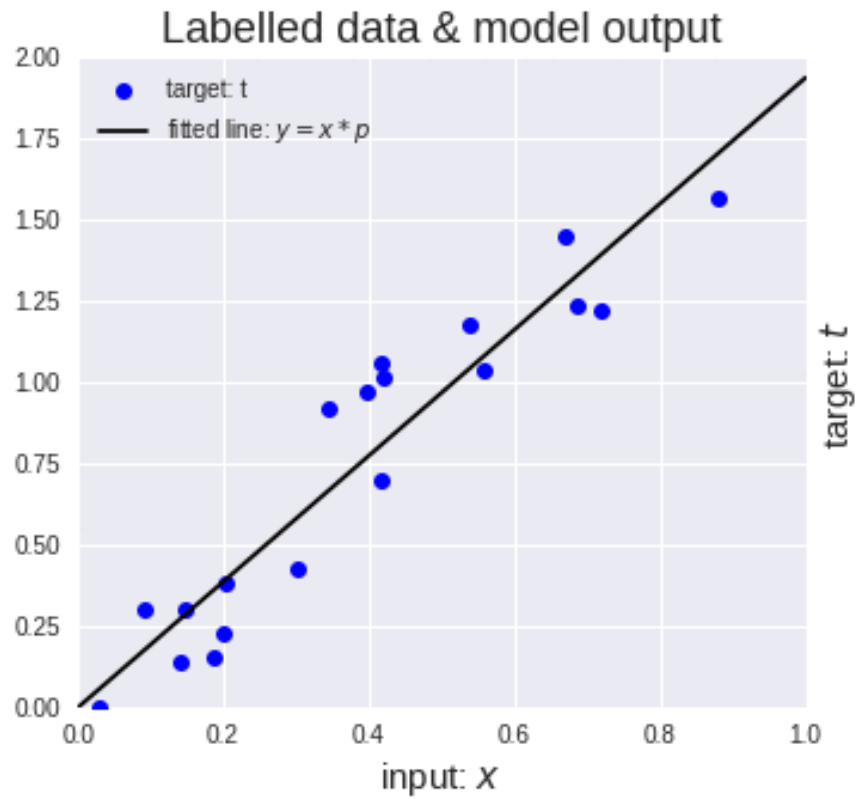


Note: One epoch means one pass of the whole training set.

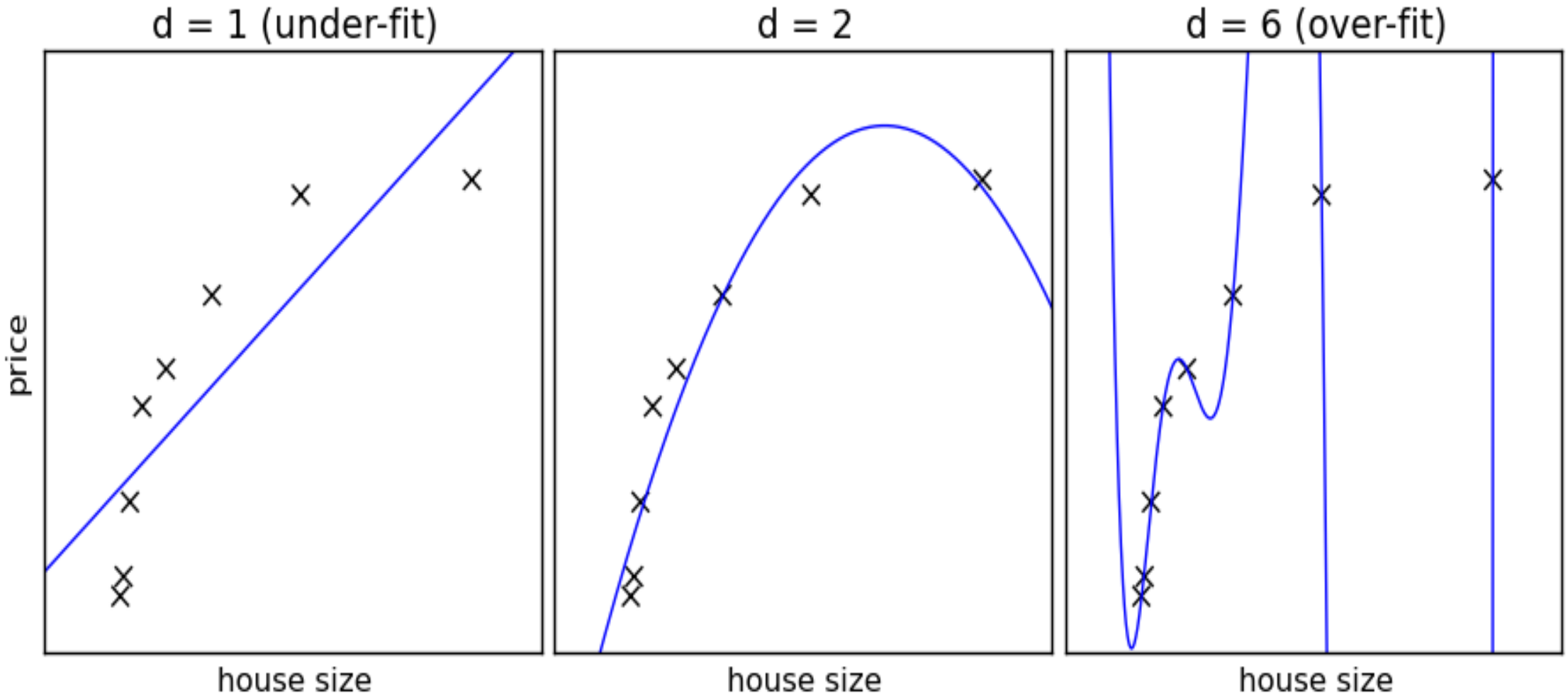
Gradient descent in action

- Example: Finding best linear fit to a set of points.





Under, Good, and Over Fit



Poor learning

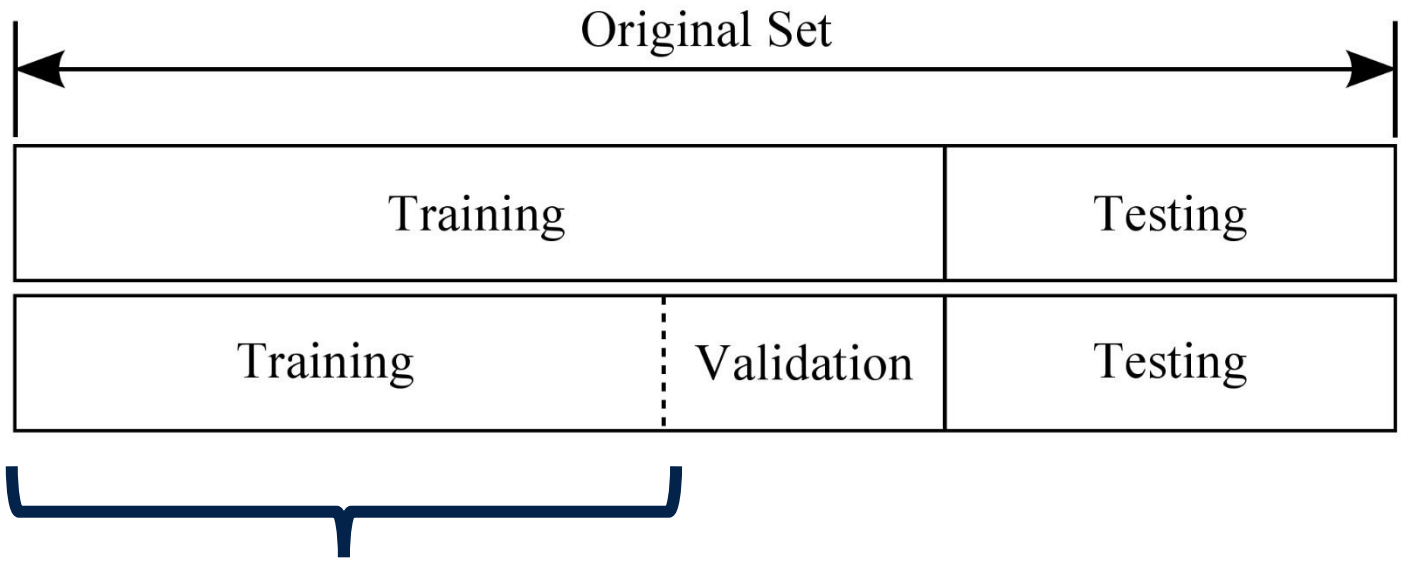
Correct learning

Wrong learning

life.augmented

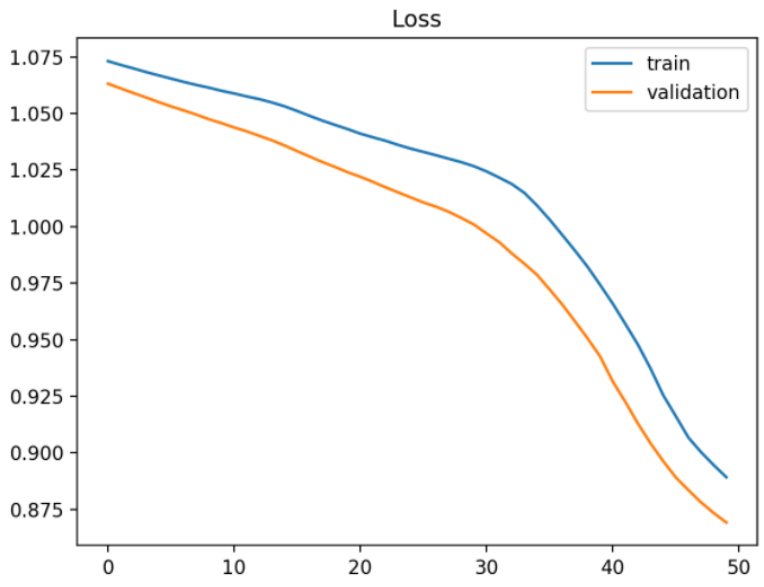
Learning datasets

- One of the challenges of working with ANN is to have a big labelled dataset.
- The dataset is usually divided into
  - Training set
    - Training set
    - Validation set
  - Testing set

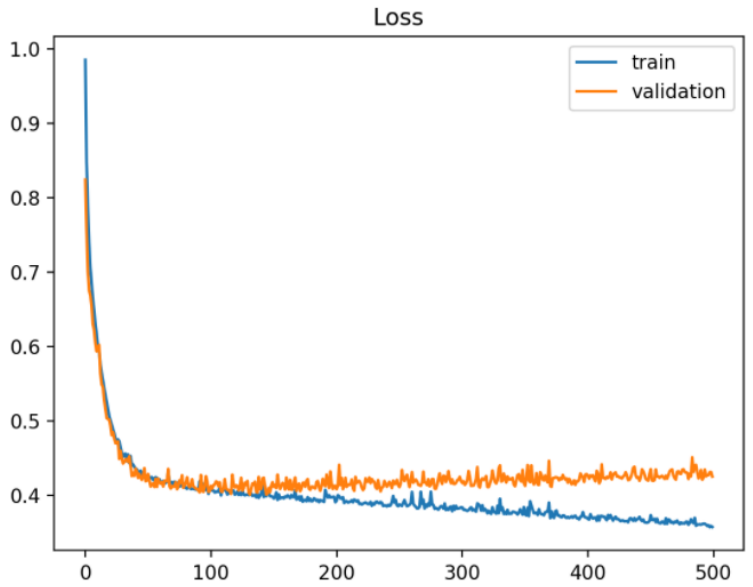


Note: One epoch means one pass of the whole training set.

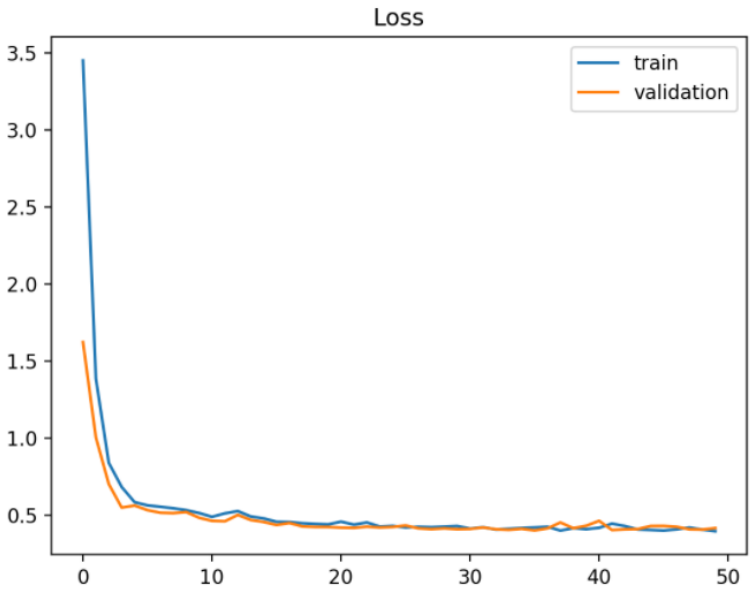
Learning Curves



Example of Training Learning Curve Showing an Underfit Model That Requires Further Training



Example of Train and Validation Learning Curves Showing an Overfit Model



Example of Train and Validation Learning Curves Showing a Good Fit

TinyML deployment challenges

life.augmented

Cloud computing vs Edge Computing

Cloud Computing

Training is still here

Edge Computing

And many more…

Inference can be here

life.augmented

What are the main deployment challenges of TinyML?

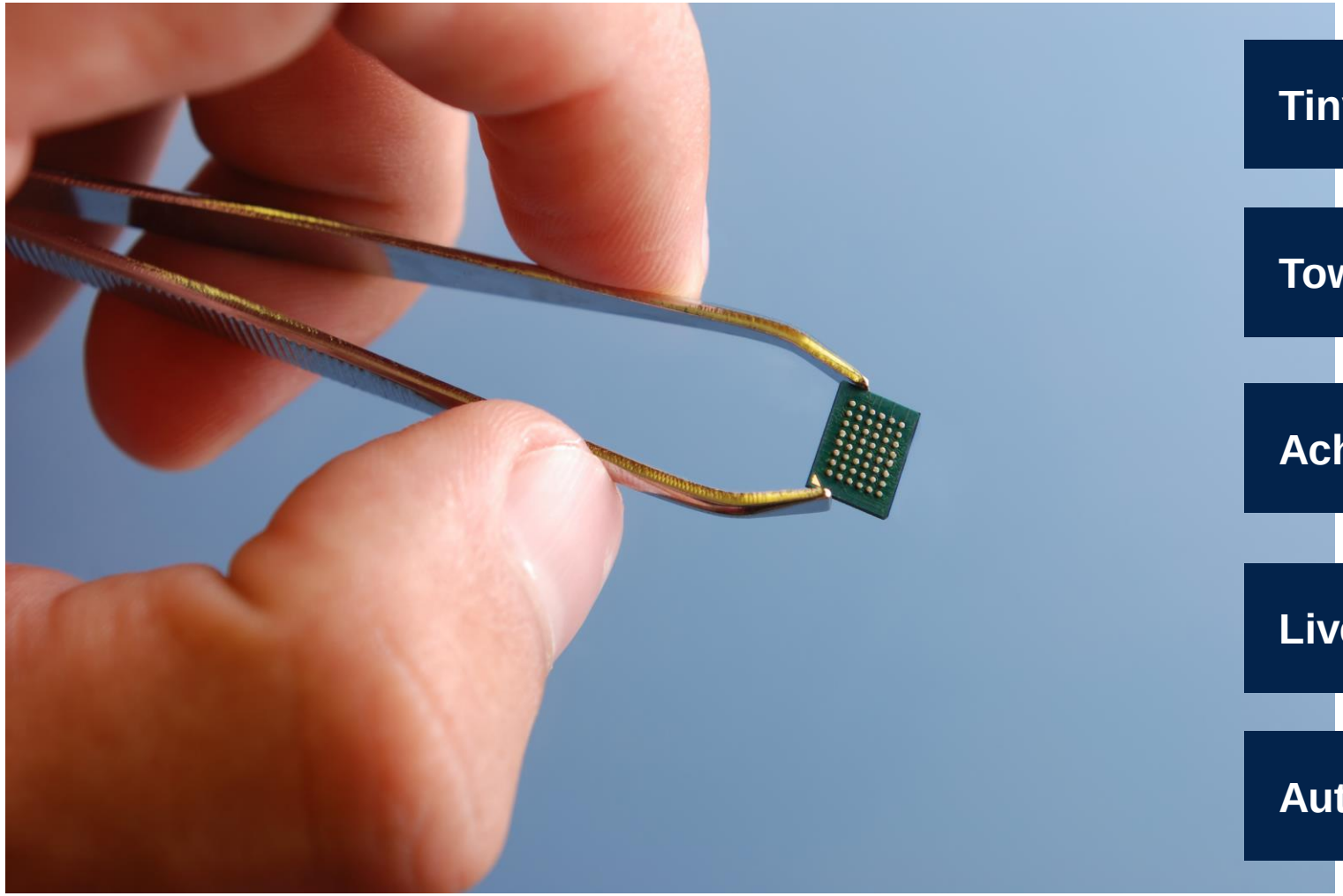
life.augmented

Tiny ML Challenges

Phenomenal Cosmic Powers!

Itty Bitty Living Space!

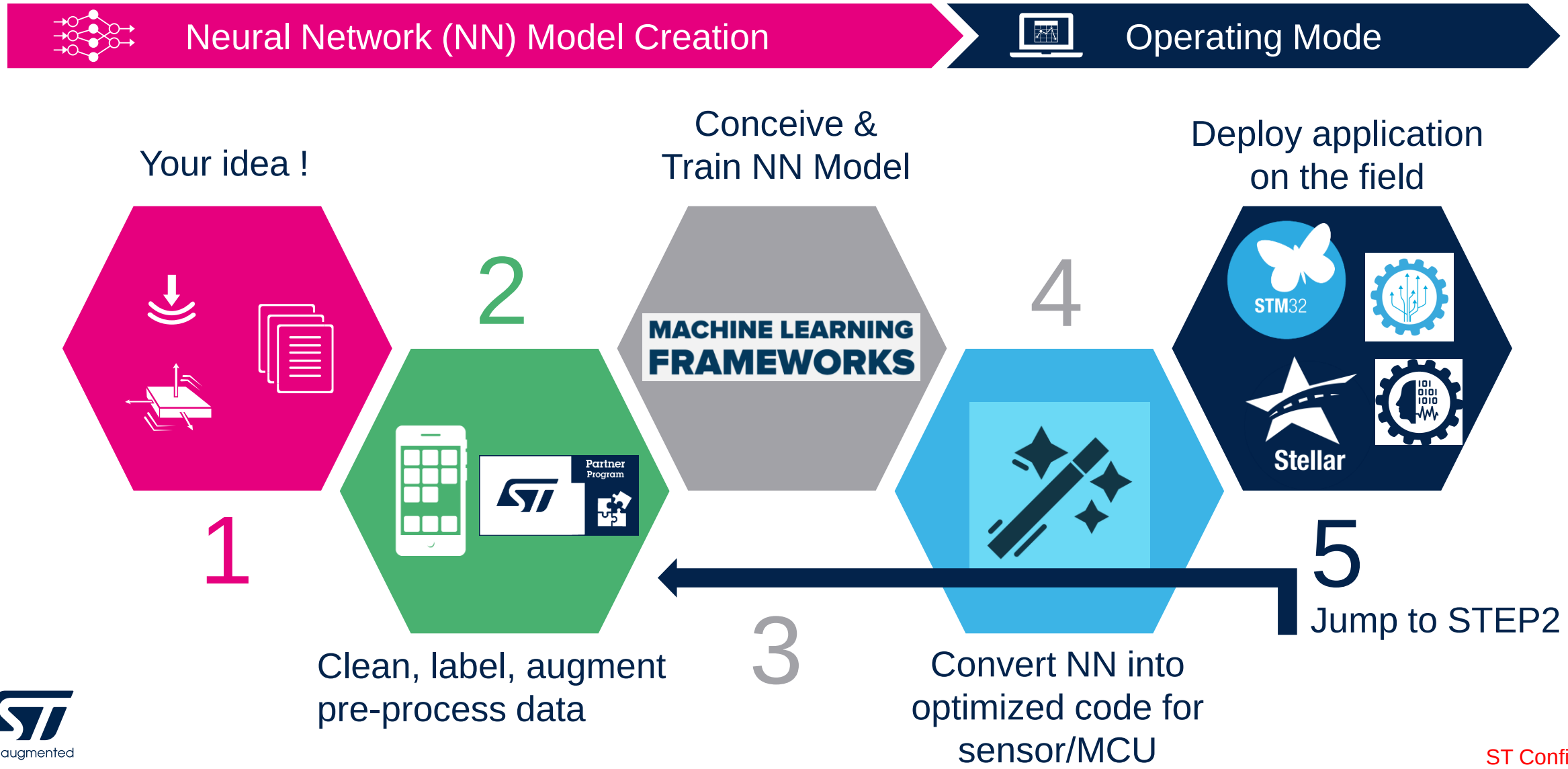
Tiny ML Challenges



- **Tiny Resources**
- **Toward Zero Power**
- **Achieve High Accuracy**
- **Live without floating point numbers**
- **Automated deployments**

life.augmented

5 Key steps for Supervised Deep Learning



life.augmented

Deploying Inference on MCU/sensor

0 $\text{Accuracy}_{\text{neural networks}} > \text{Accuracy}_{\textit{hand-crafted algorithm}}$

1 $\text{RAM}_{\text{MCU}} > \text{Activations}_{\text{Network}}$

2 $\text{ROM}_{\text{MCU}} > \text{Weights}_{\text{Network}}$

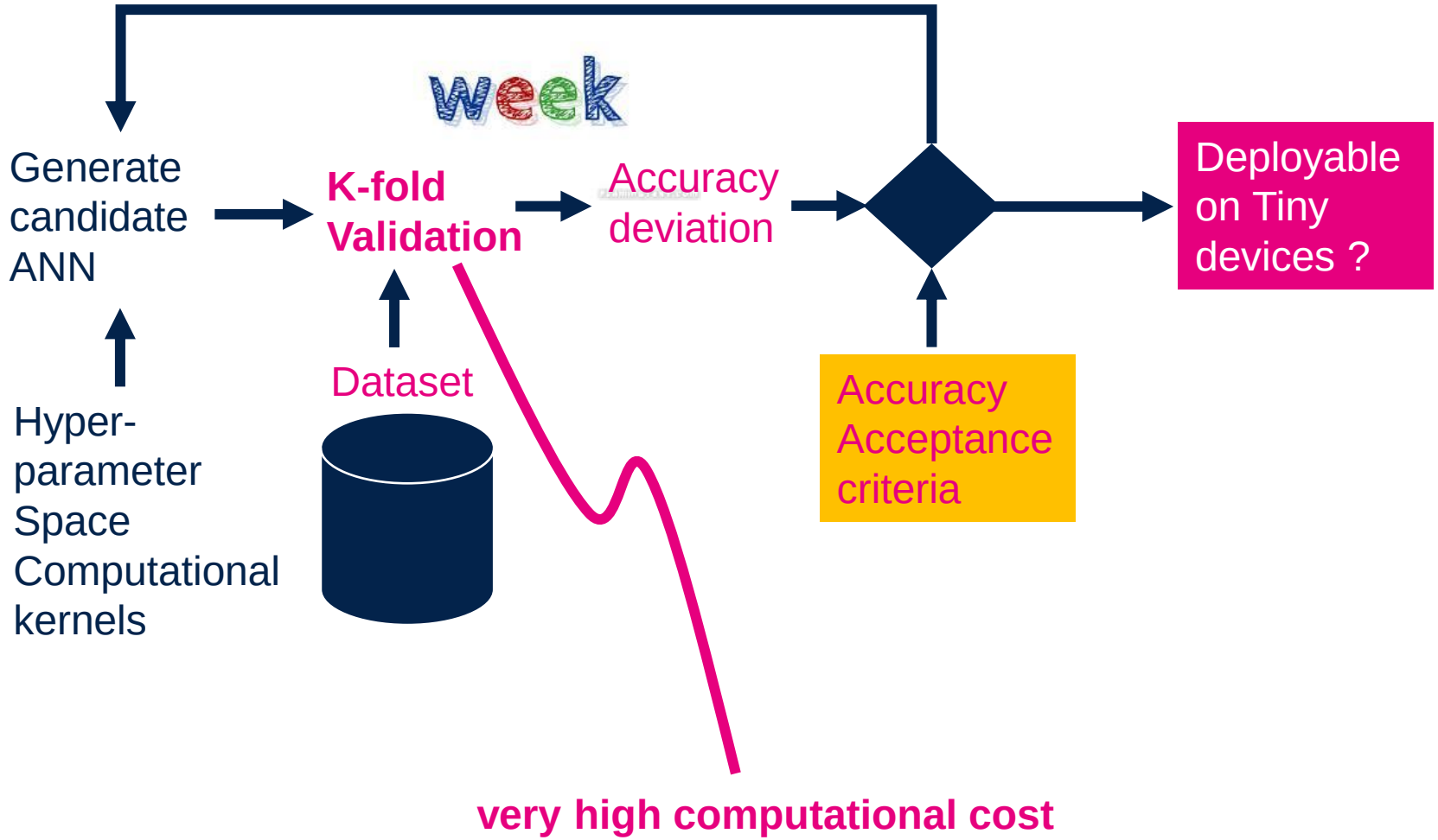
3 $\dfrac{\text{NN complexity}_{\text{FLOPS}}}{\text{MCU}_{\text{FLOPS/s}}} < \dfrac{\text{samples in a window}}{\text{f}_{\text{sensor sampling rate/s}}}$

- ISPU
- MLC
- STM32
- STM32N6
- STELLAR

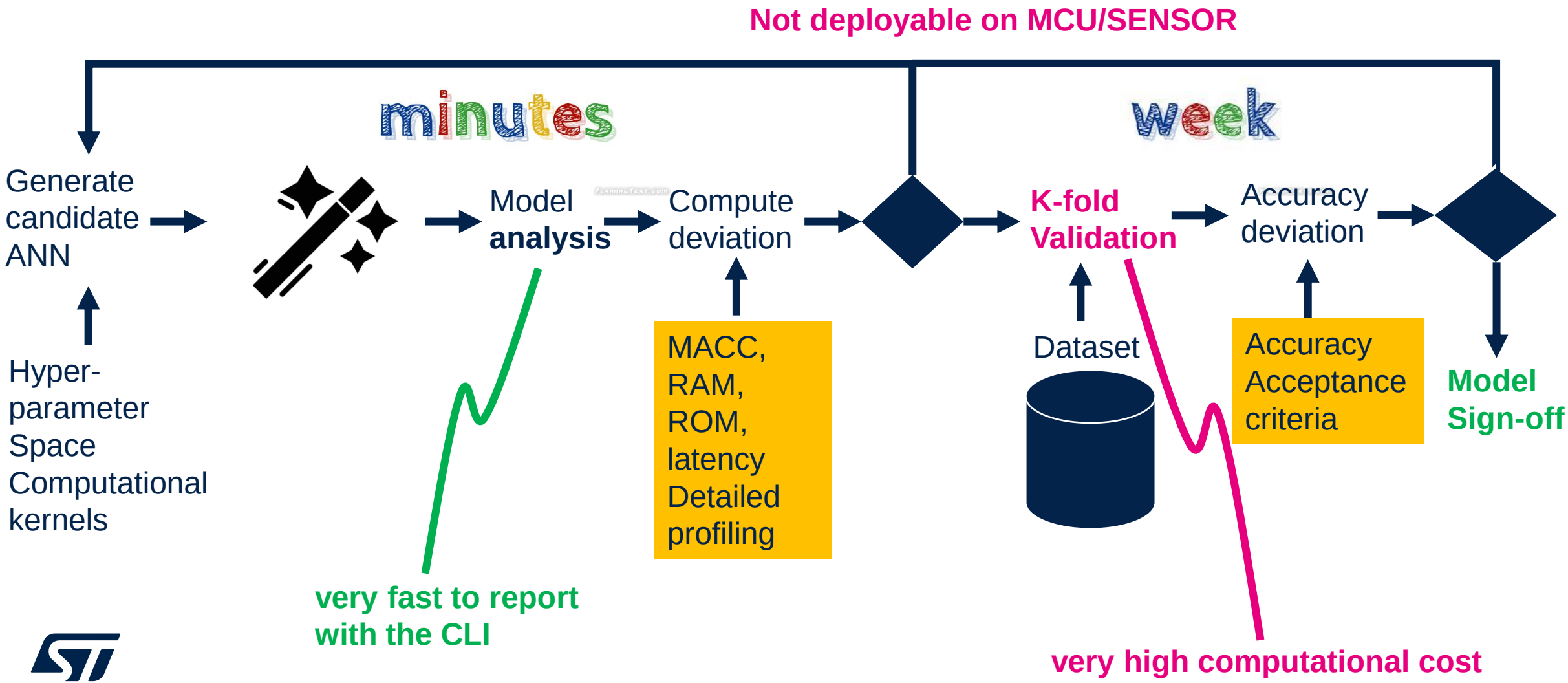
"Embedded Real-Time Fall Detection with Deep Learning on Wearable Devices," *2018 21st Euromicro Conference on Digital System Design (DSD)*, Prague, 2018, pp. 405-412, doi: 10.1109/DSD.2018.00075.

life.augmented

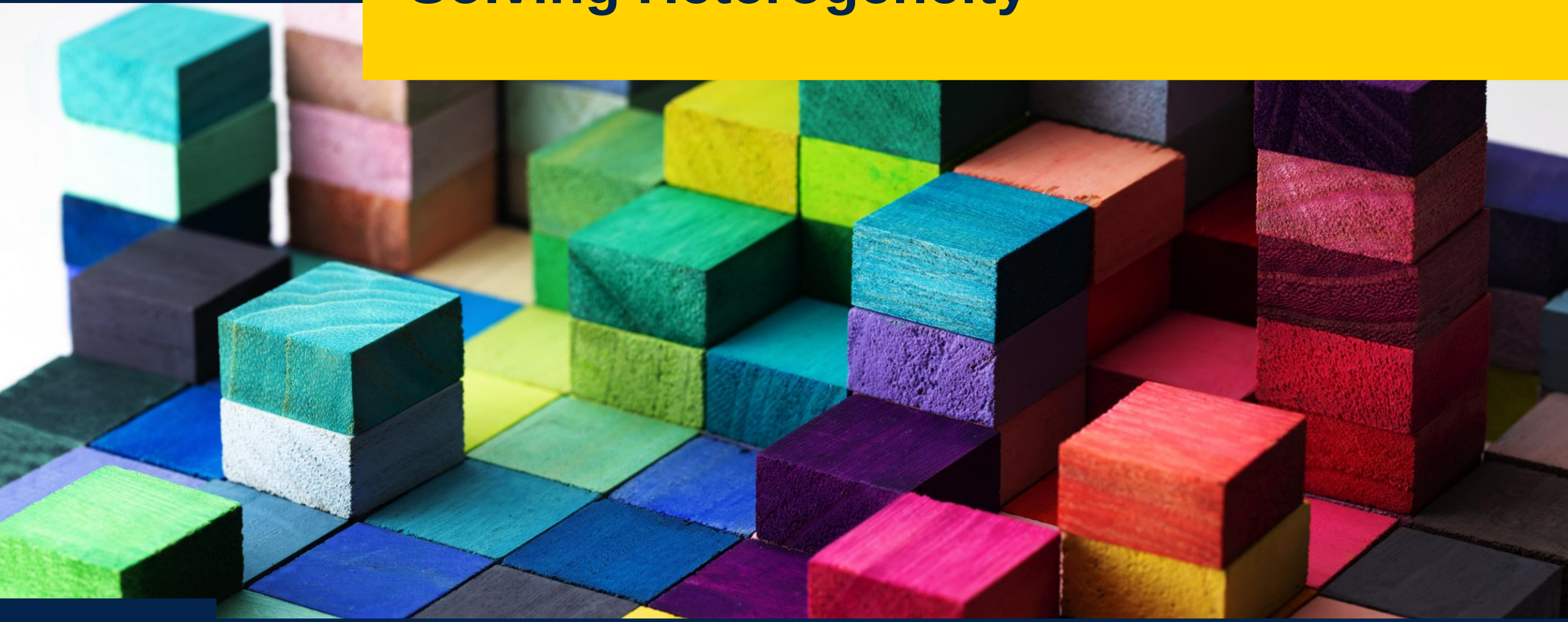
Deployment un-aware Neural Architecture Search (NAS)



Deployment aware Neural Architecture Search (NAS)

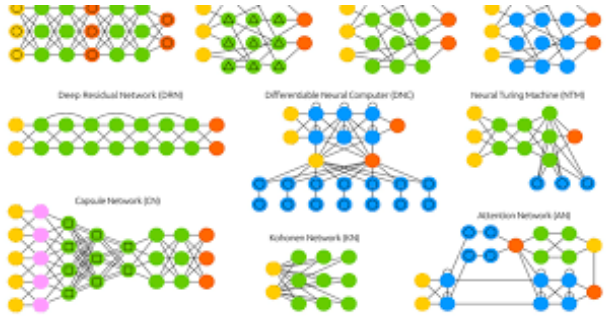


Solving Heterogeneity

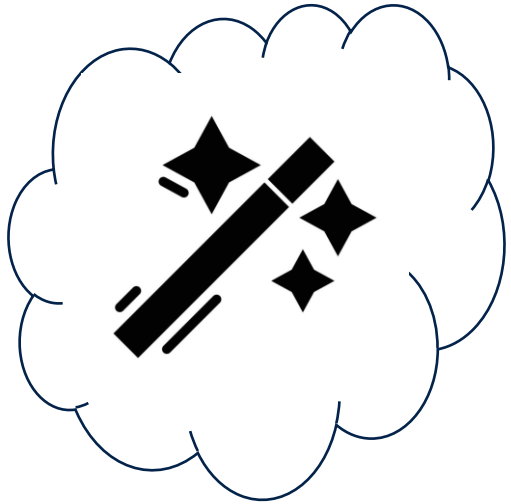


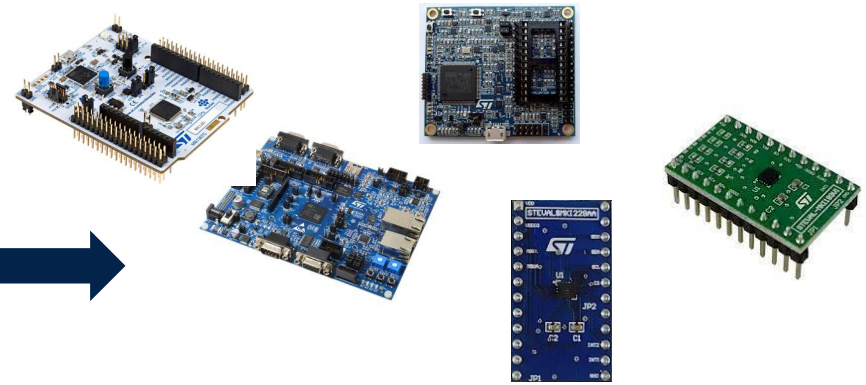
life.augmented

Machine Learning Heterogeneity



Multiple Networks





Multiple Targets

Coping with source and target heterogeneity

Maximize:

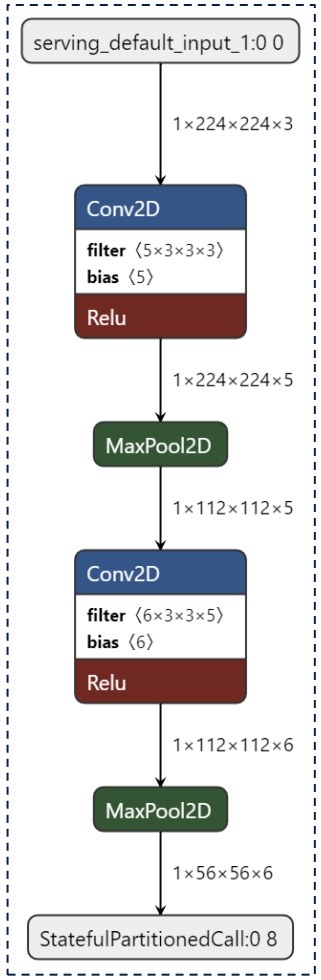
- Performance
- Sharing and Reuse
- Productivity
- Automation

Minimize:

- Differences
- Development time

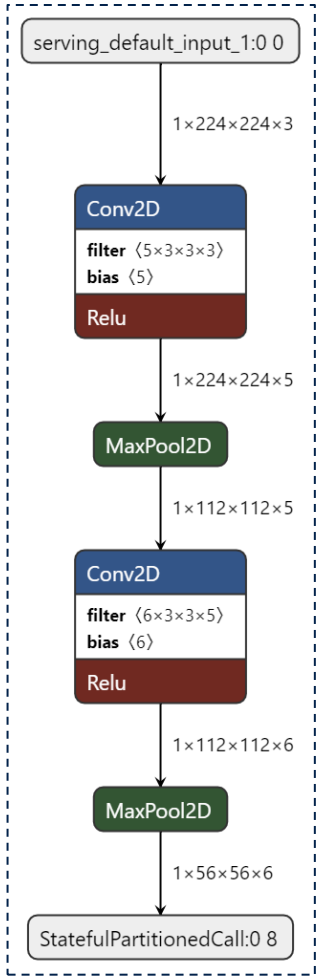
life.augmented

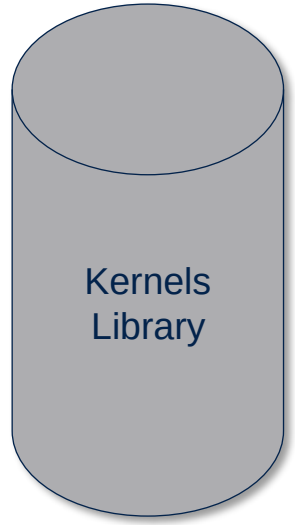
From the ML model to its deployment



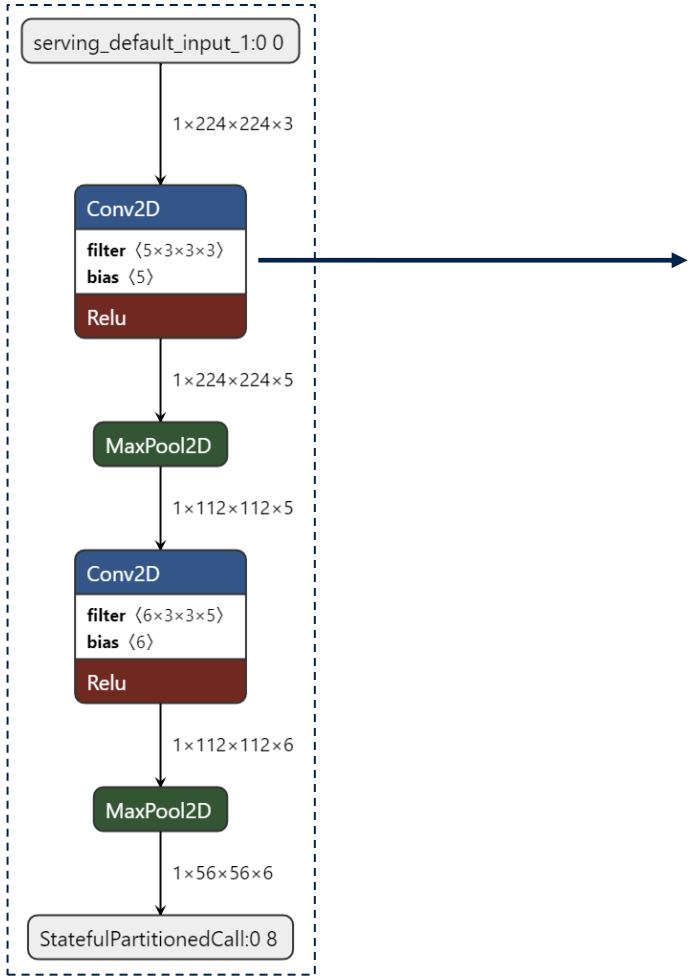
life.augmented

From the ML model to its deployment

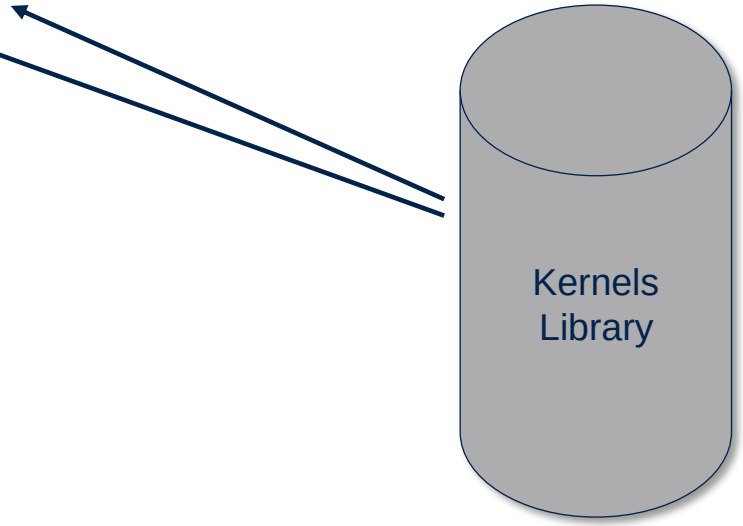




From the ML model to its deployment

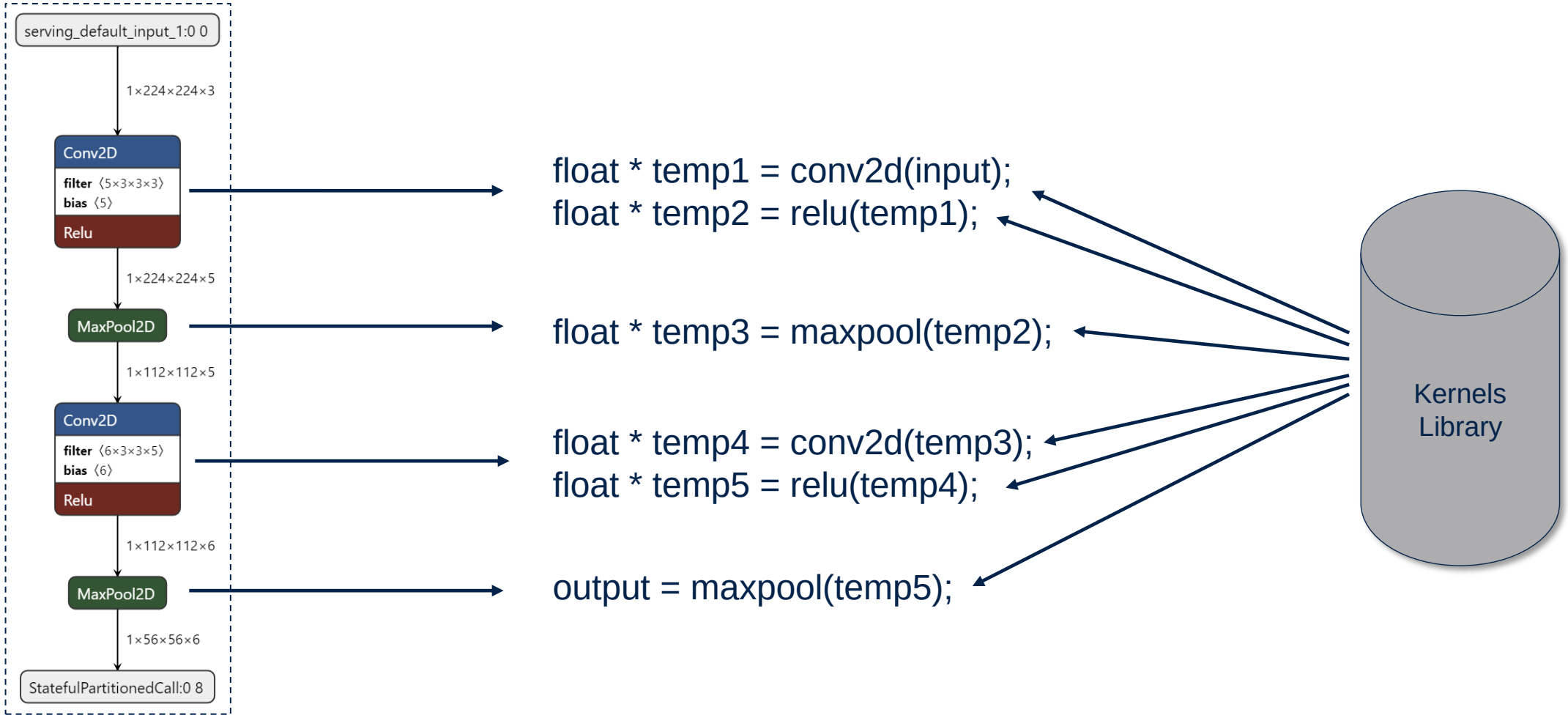



```
float * temp1 = conv2d(input);
float * temp2 = relu(temp1);
```

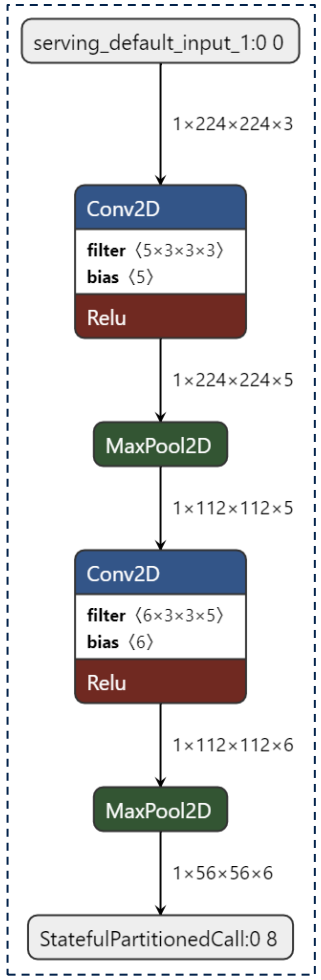


life.augmented

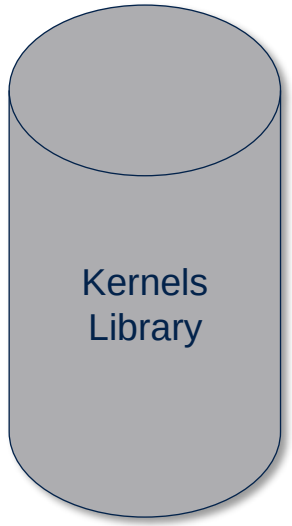
From the ML model to its deployment



From the ML model to its deployment

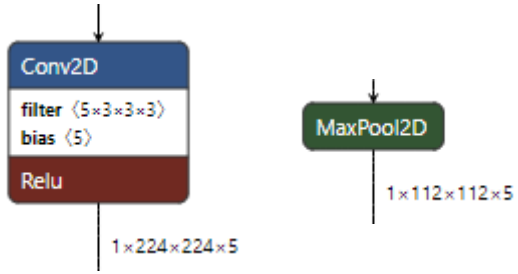



```
void network(float * input, float * output) {

    float * temp1 = conv2d(input);
    float * temp2 = relu(temp1);

    float * temp3 = maxpool(temp2);

    float * temp4 = conv2d(temp3);
    float * temp5 = relu(temp4);

    output = maxpool(temp5);

}
```



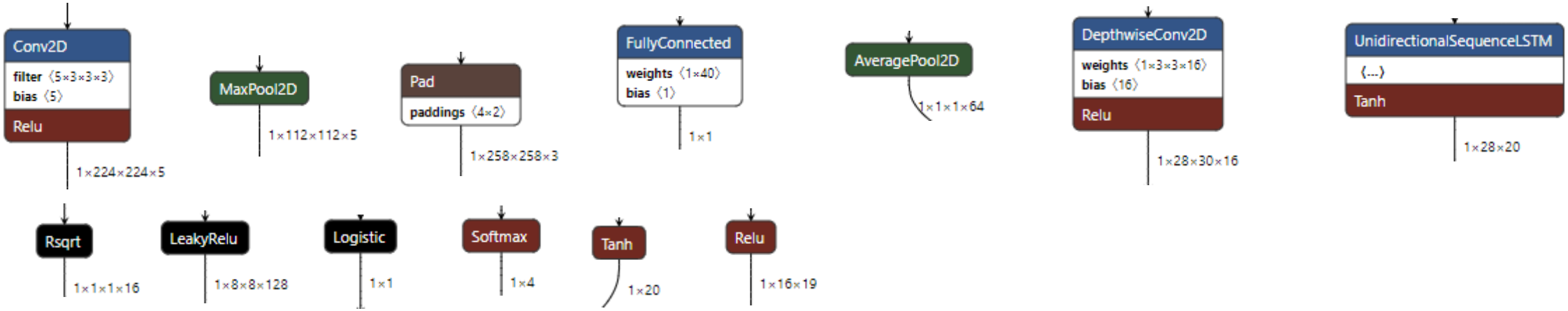
life.augmented

Heterogeneity: Operators

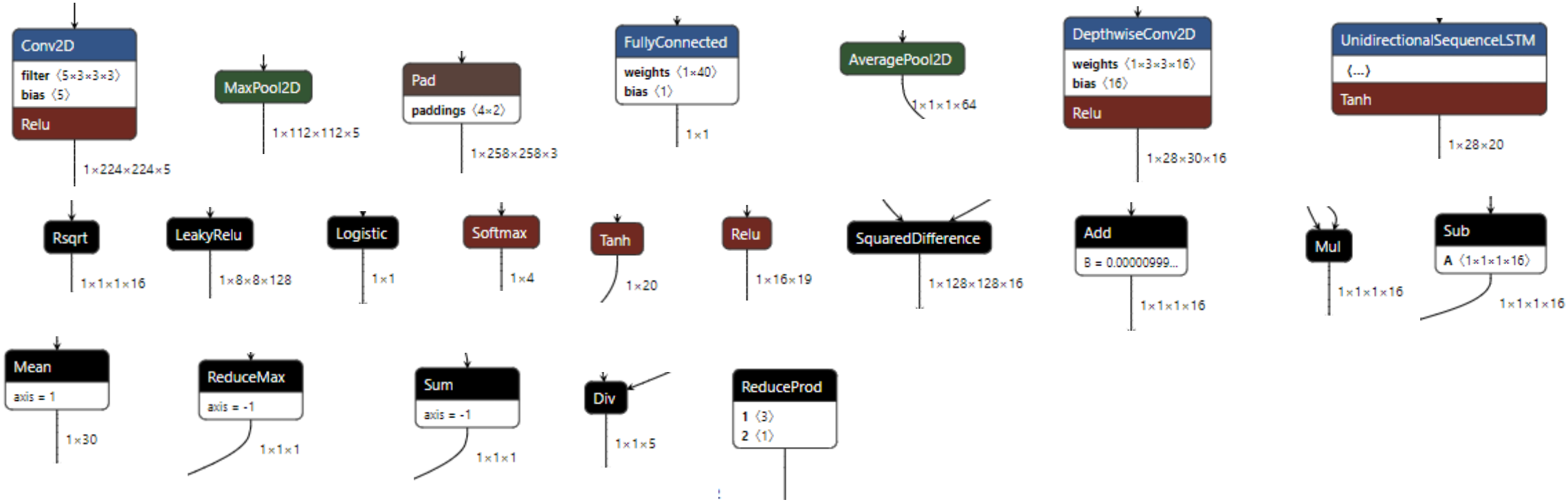


Heterogeneity: Operators

Heterogeneity: Operators

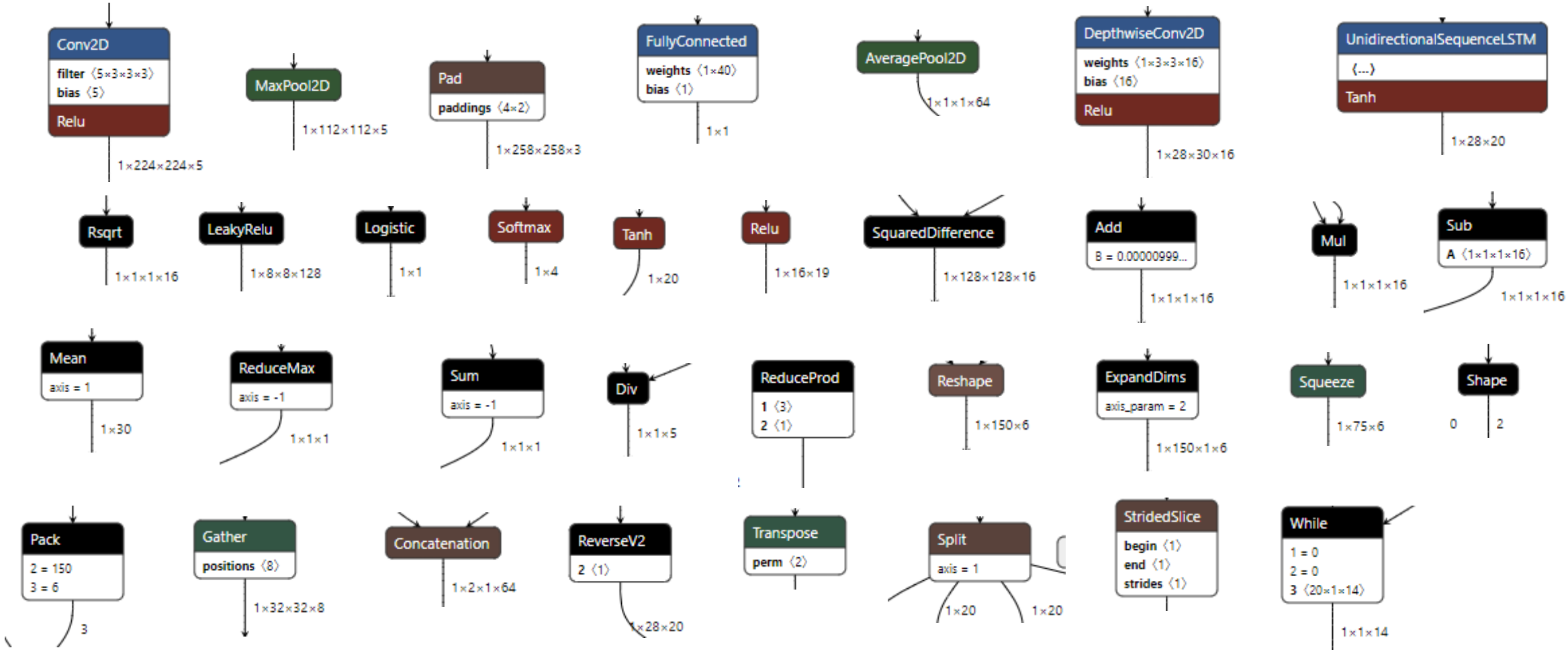


Heterogeneity: Operators



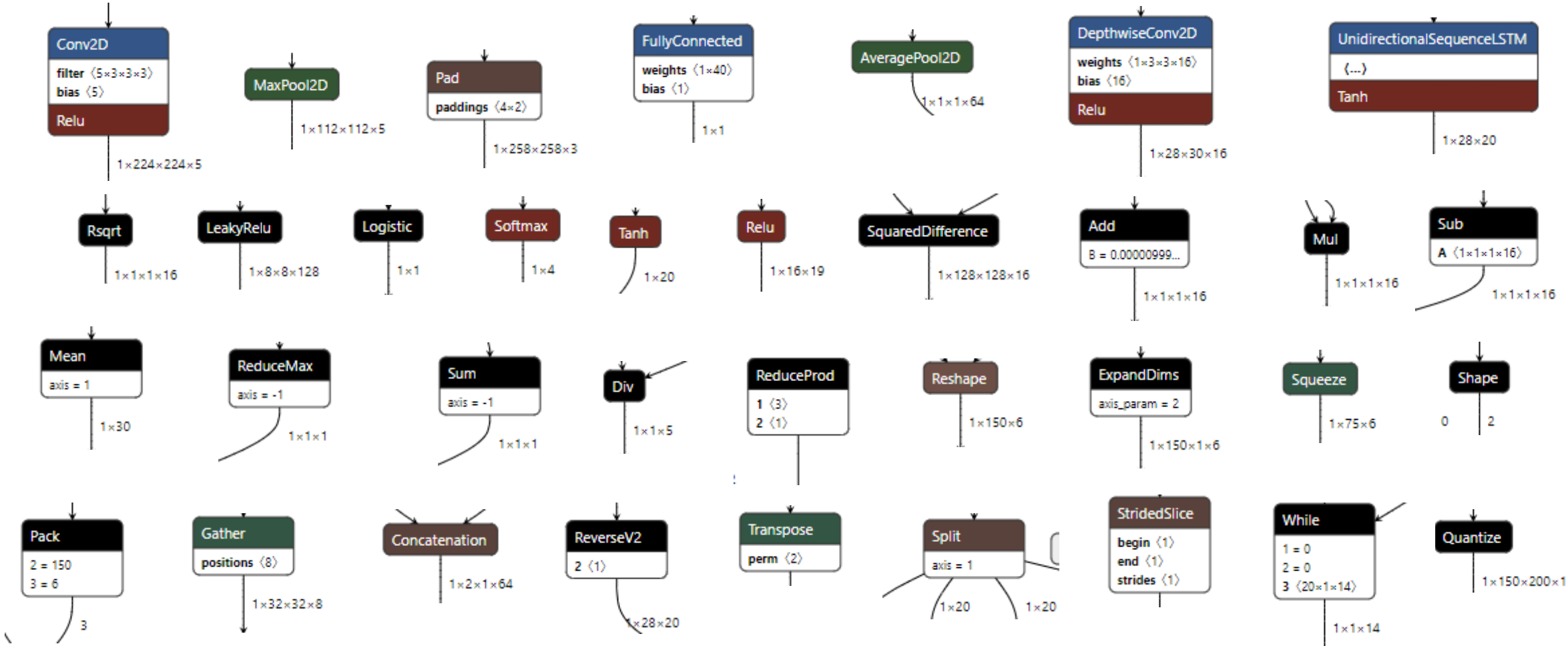
life.augmented

Heterogeneity: Operators



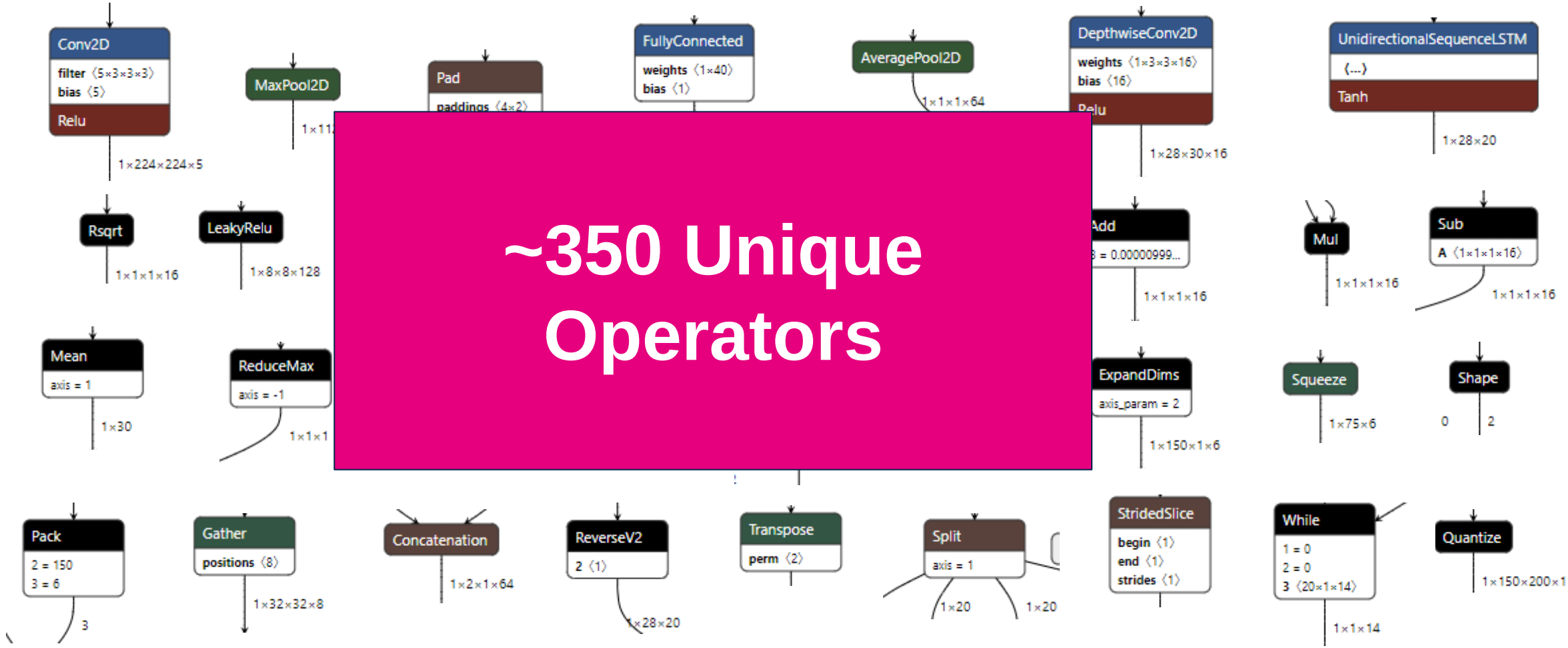
life.augmented

Heterogeneity: Operators



life.augmented

Heterogeneity: Operators

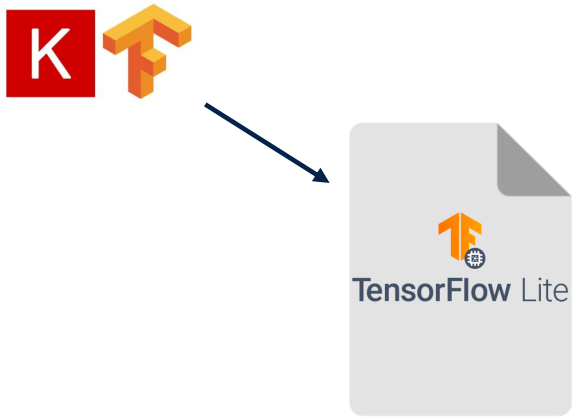


life.augmented

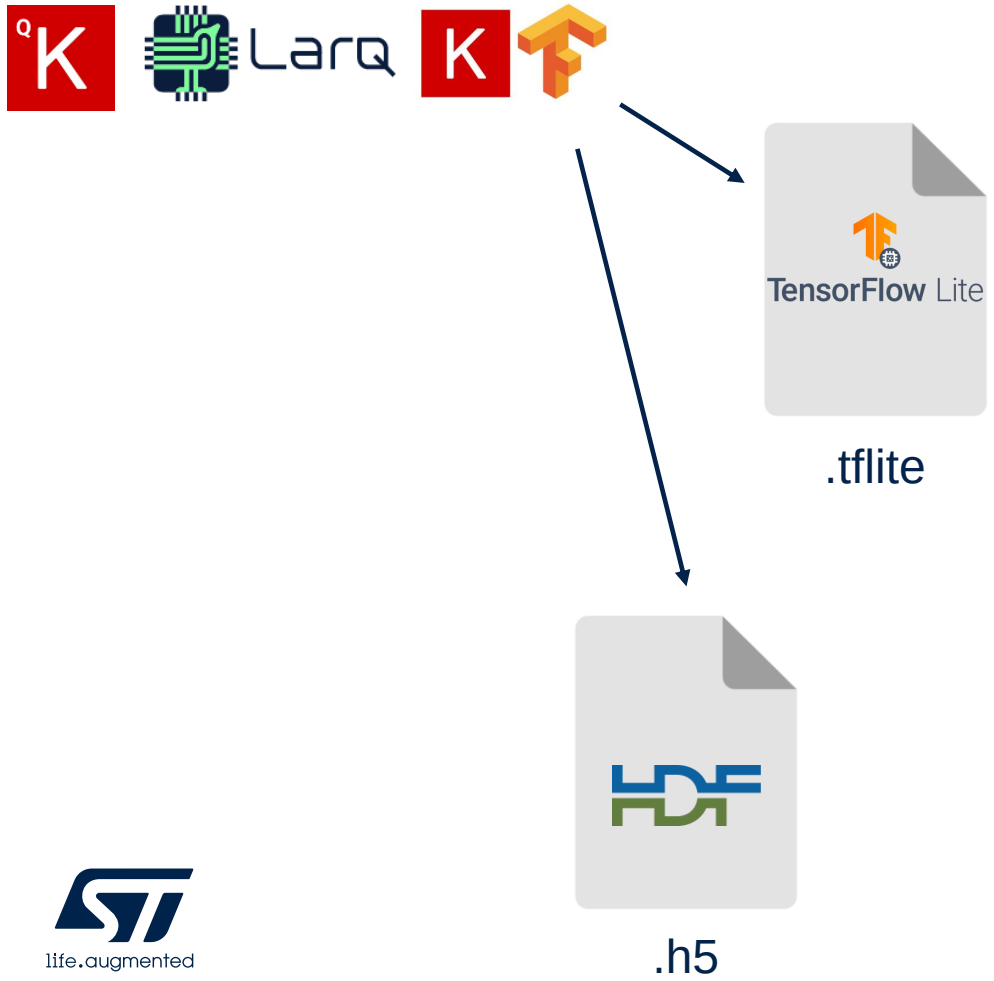
Source Model Heterogeneity: DL Formats

.tflite

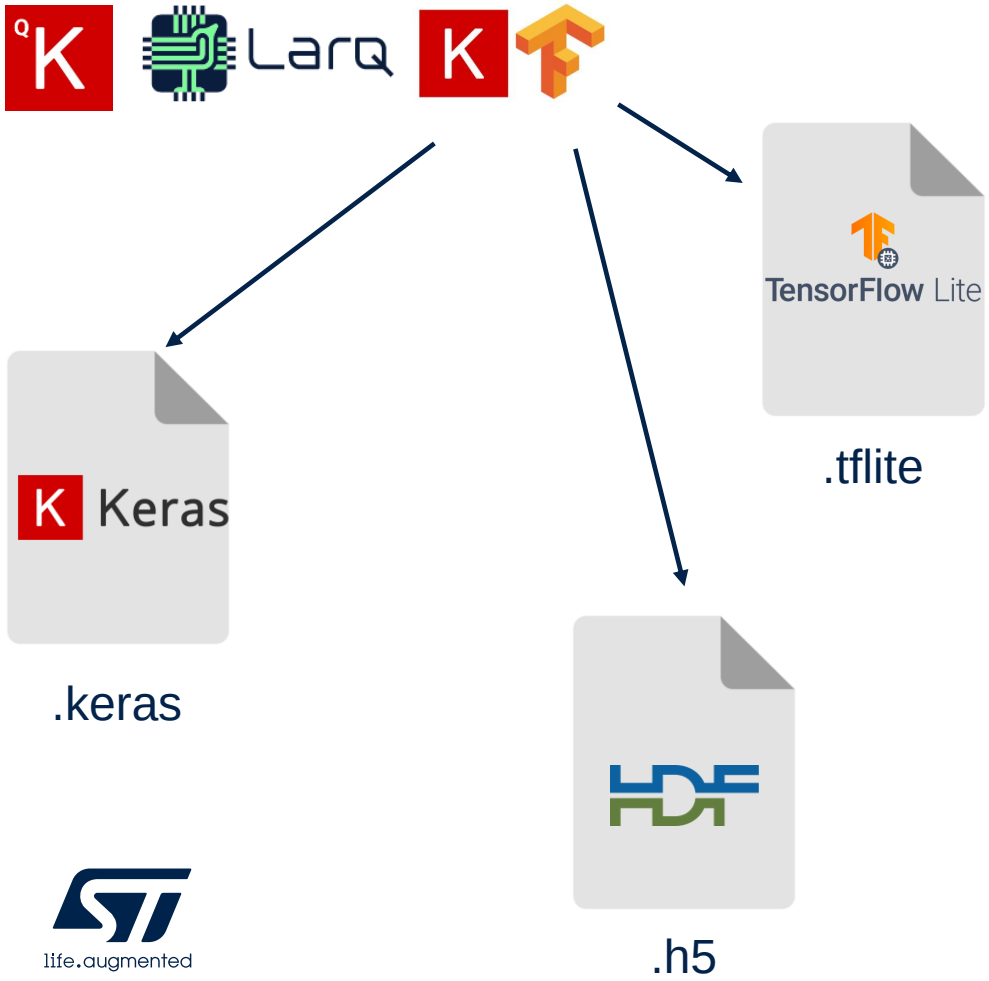
Heterogeneity: DL Formats



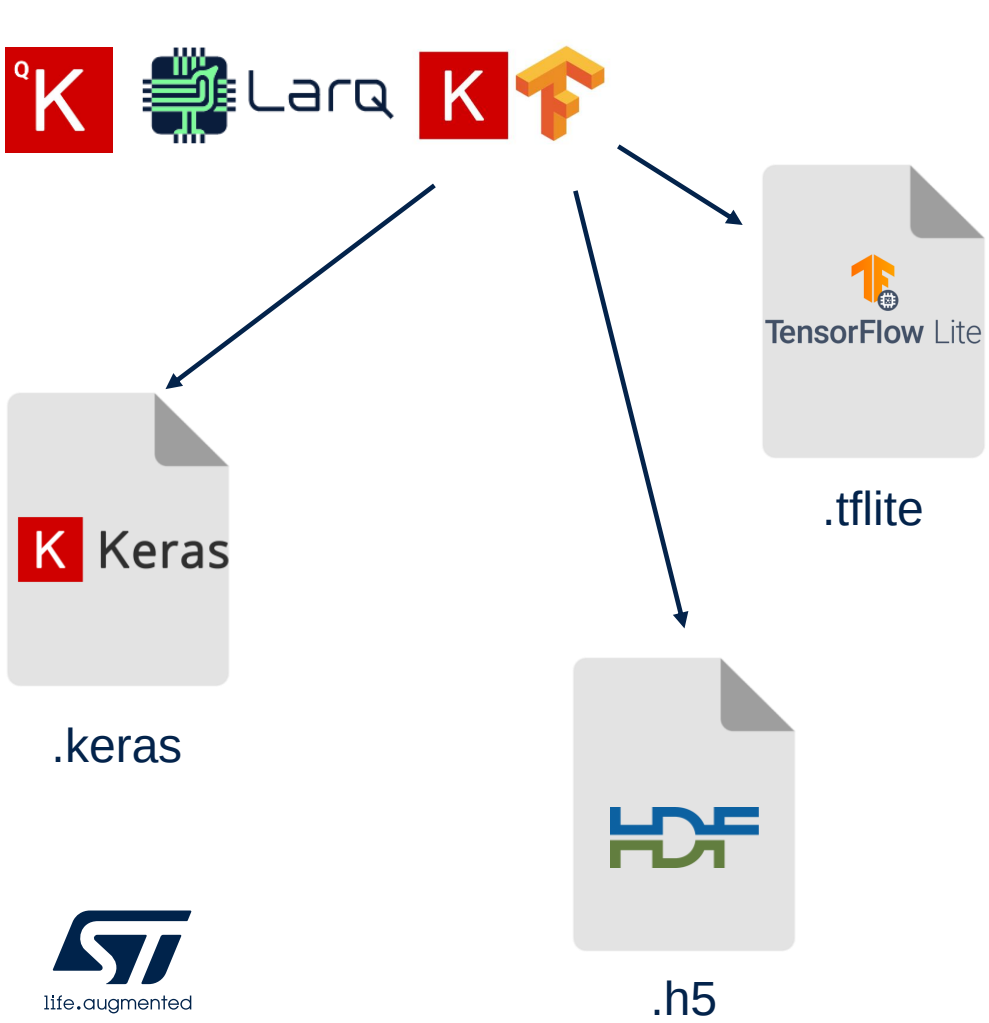
Heterogeneity: DL Formats

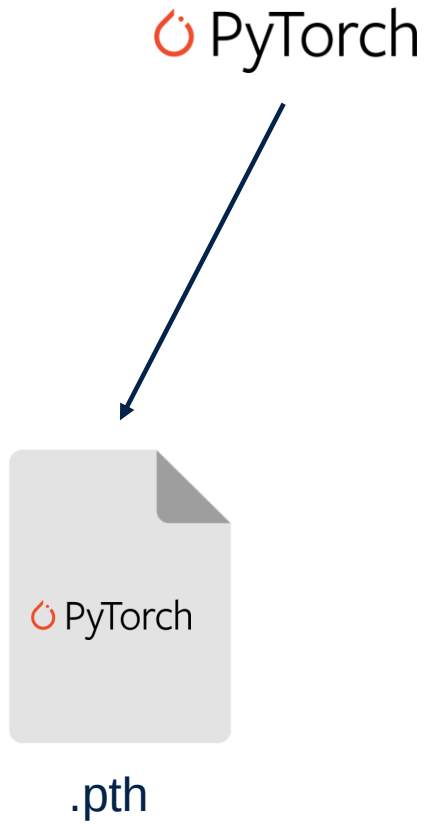


Heterogeneity: DL Formats



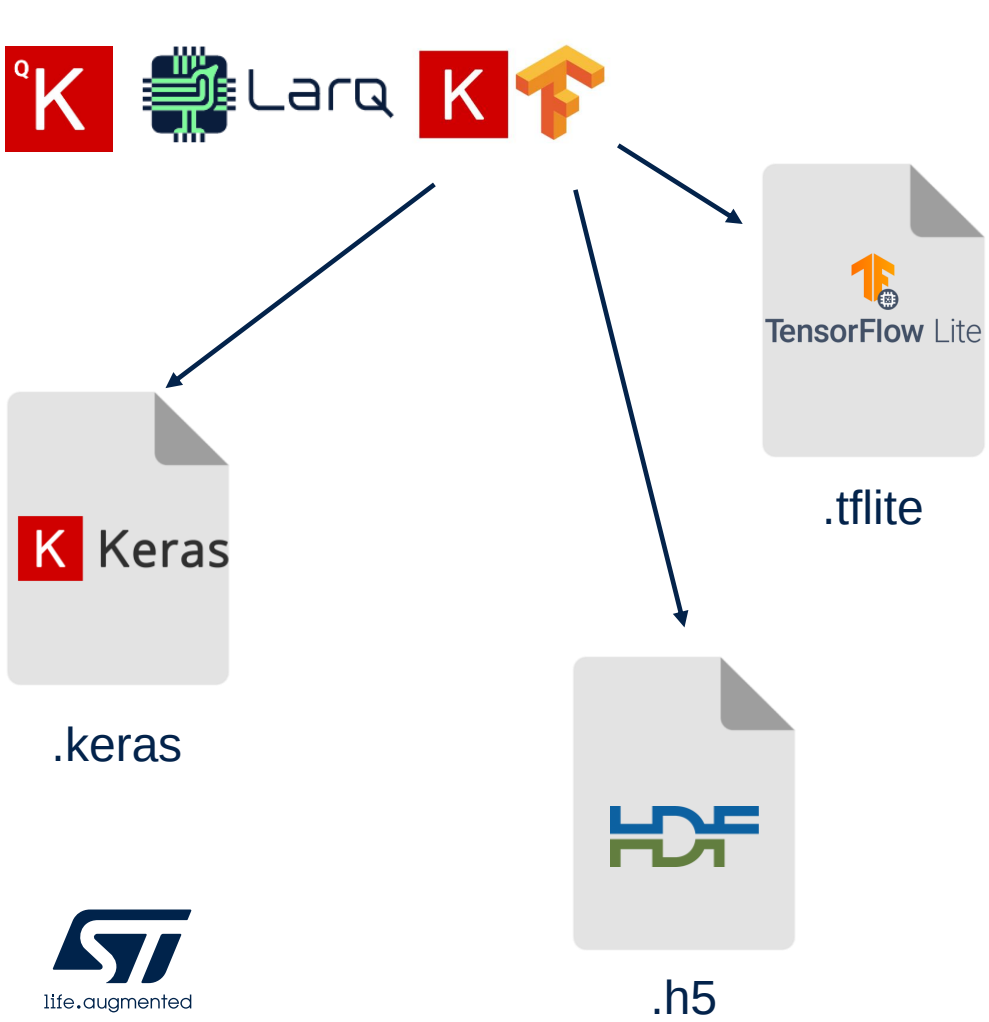
Heterogeneity: DL Formats

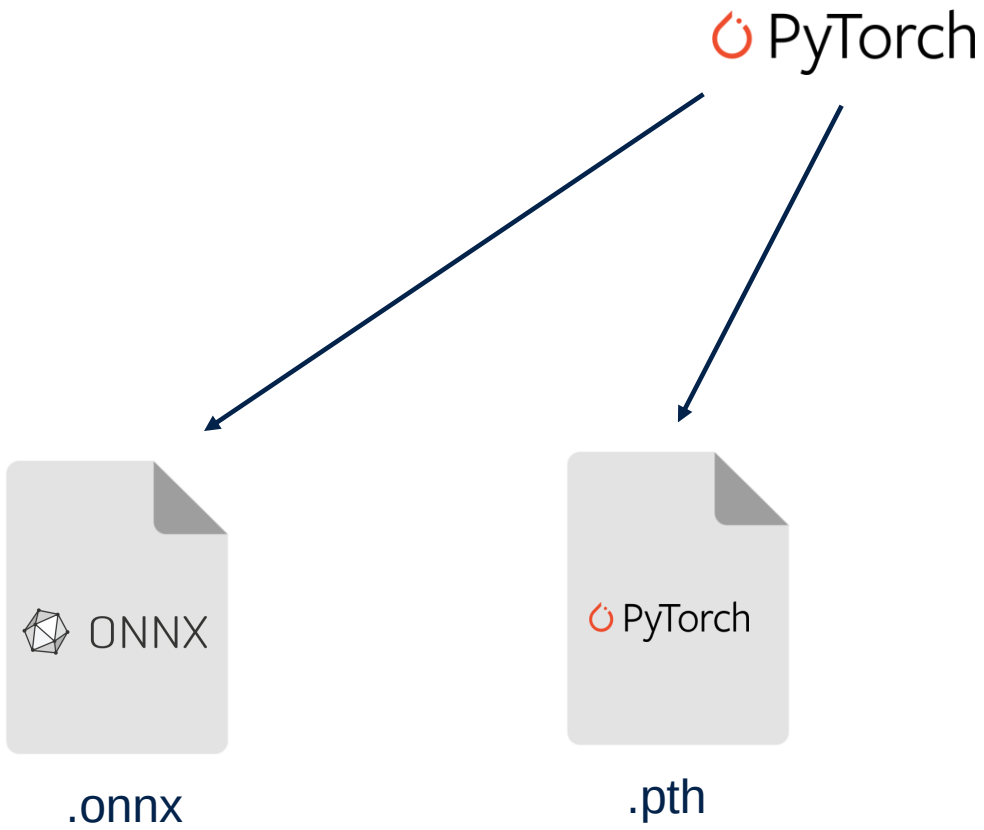




life.augmented

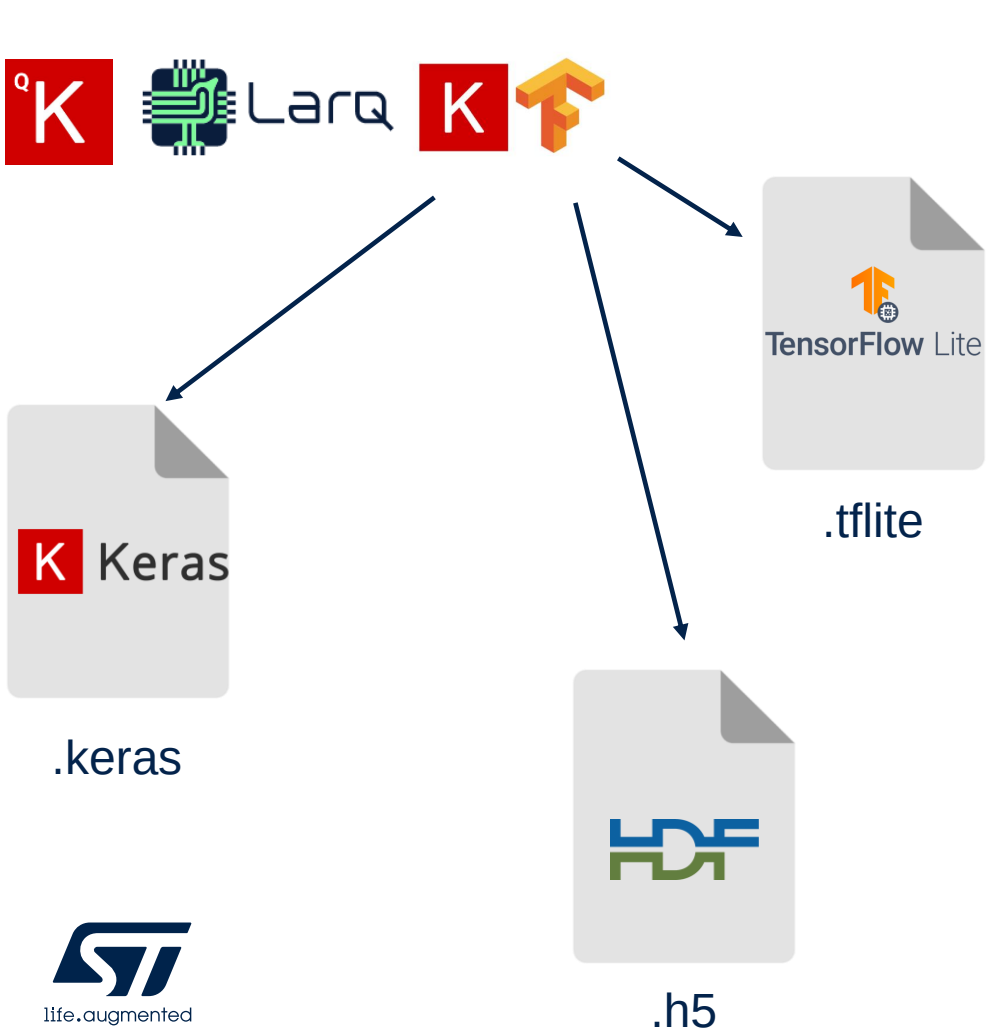
Heterogeneity: DL Formats

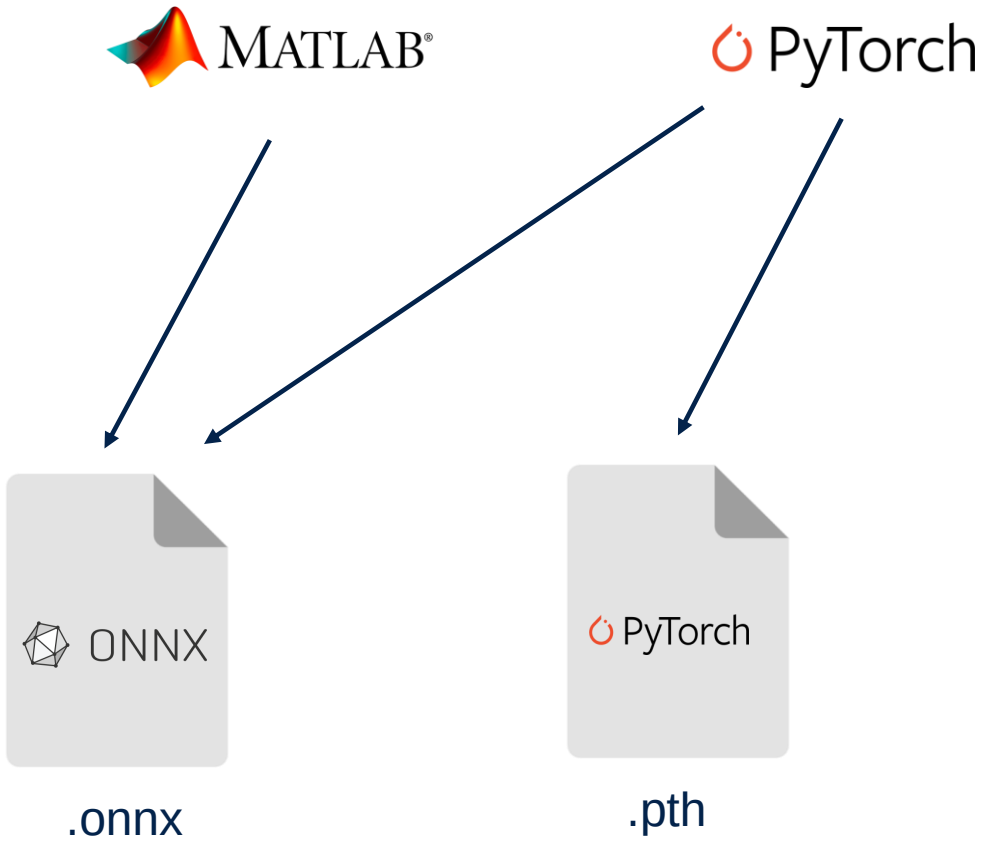




life.augmented

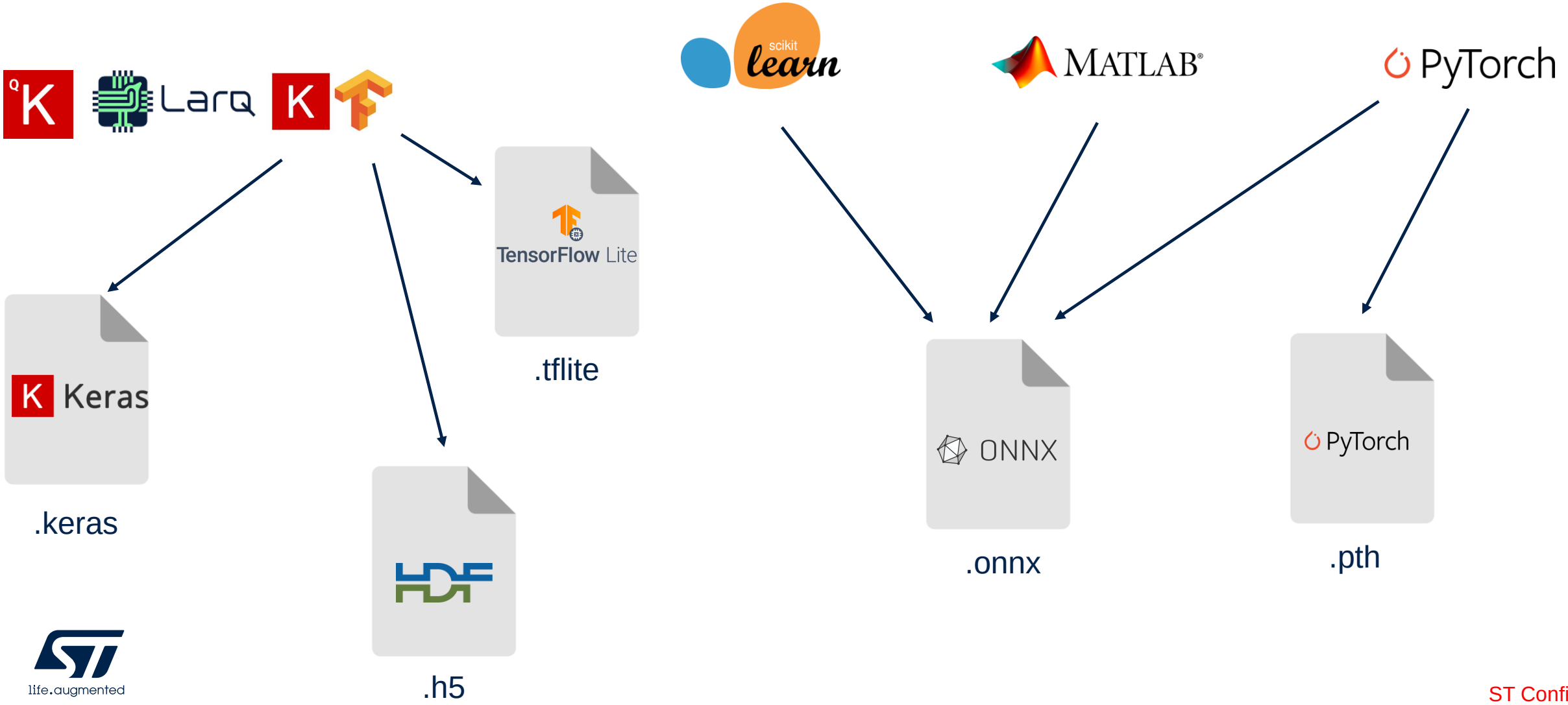
Heterogeneity: DL Formats





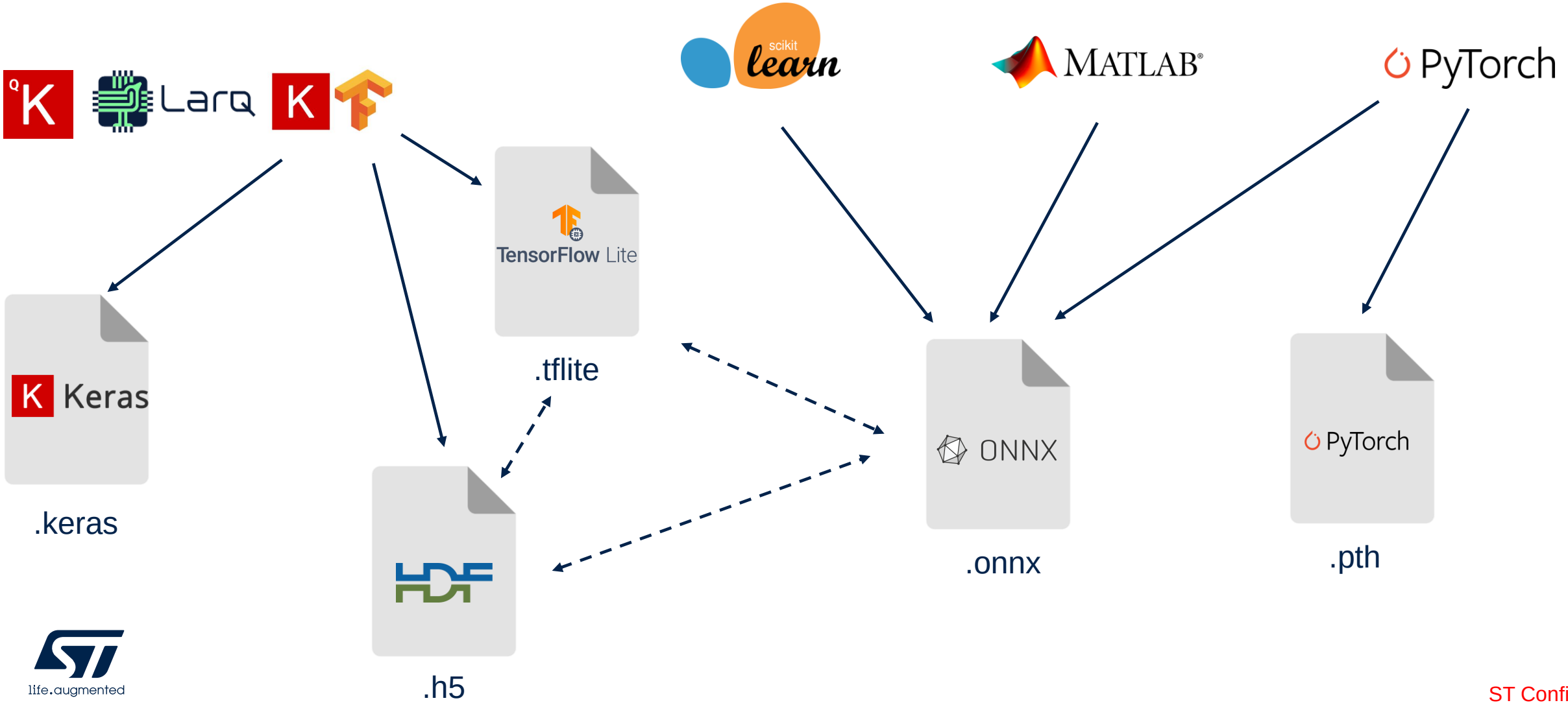
life.augmented

Heterogeneity: DL Formats



life.augmented

Heterogeneity: DL Formats



Heterogeneity: DL Differences

- Availability of operators: not all operators are available in all frameworks
- Attributes: different attributes, meaning and default values
- Data layout, i.e., channel first vs. channel last
- Quantization, i.e., many possible bit-width to represent data

Heterogeneity: DL Differences

TFLite
MEAN

ONNX
Mean

life.augmented

Heterogeneity: DL Differences

TFLite
MEAN

≠

ONNX
Mean

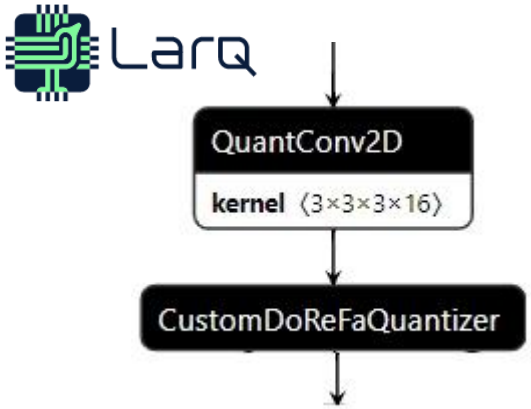
life.augmented

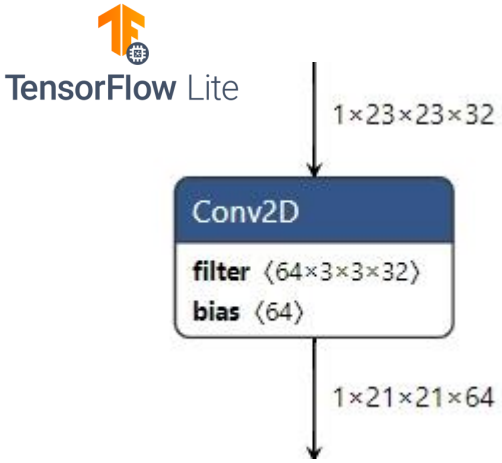
Heterogeneity: DL Differences

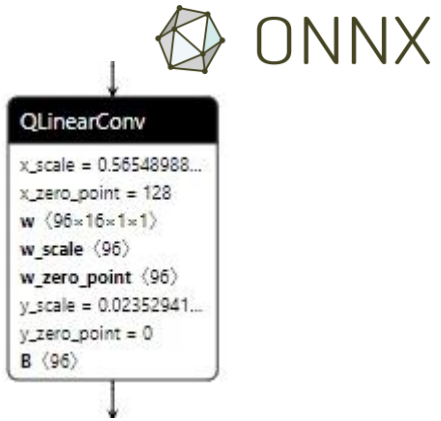
TFLite MEAN ≠ ONNX Mean

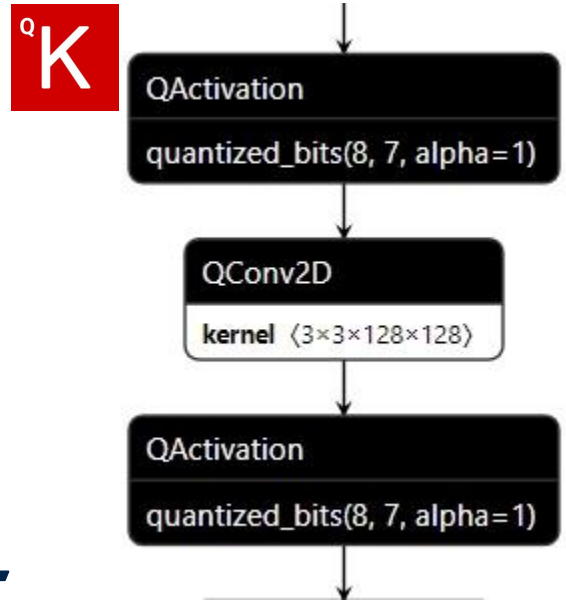
TFLite MEAN = ONNX ReduceMean

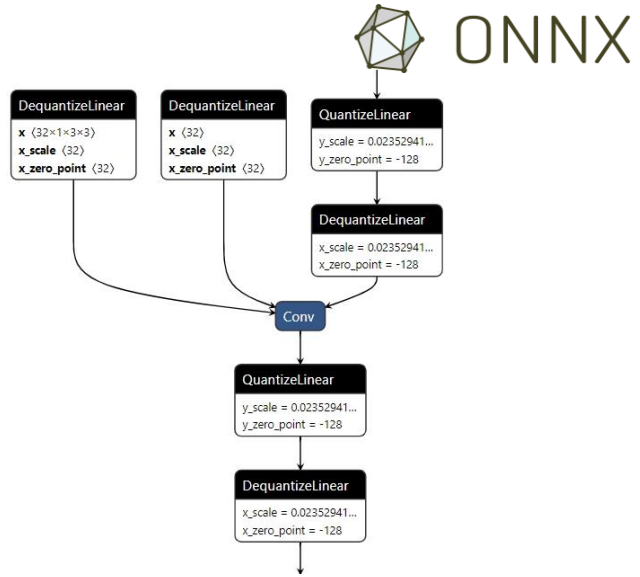
Heterogeneity: DL Formats





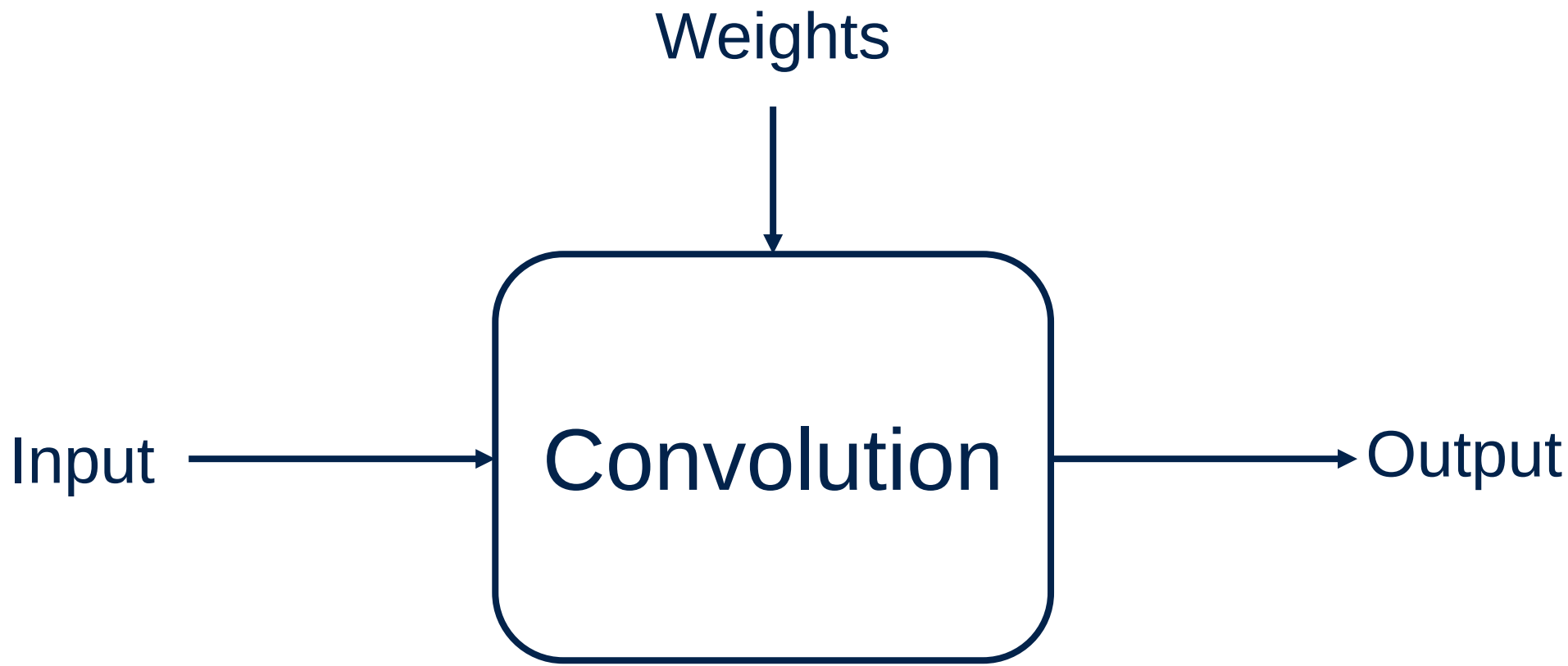






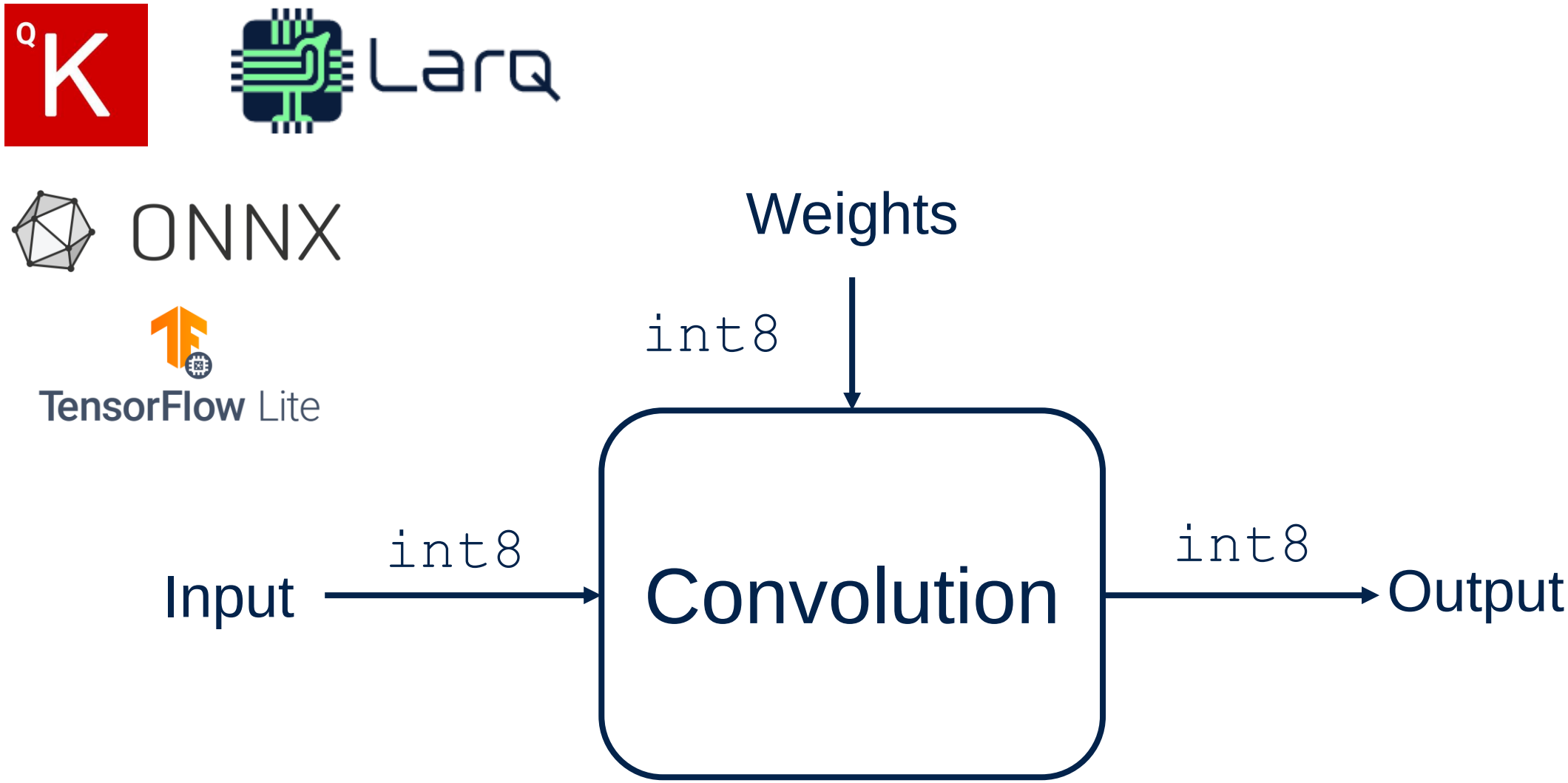
life.augmented

Heterogeneity: Quantization

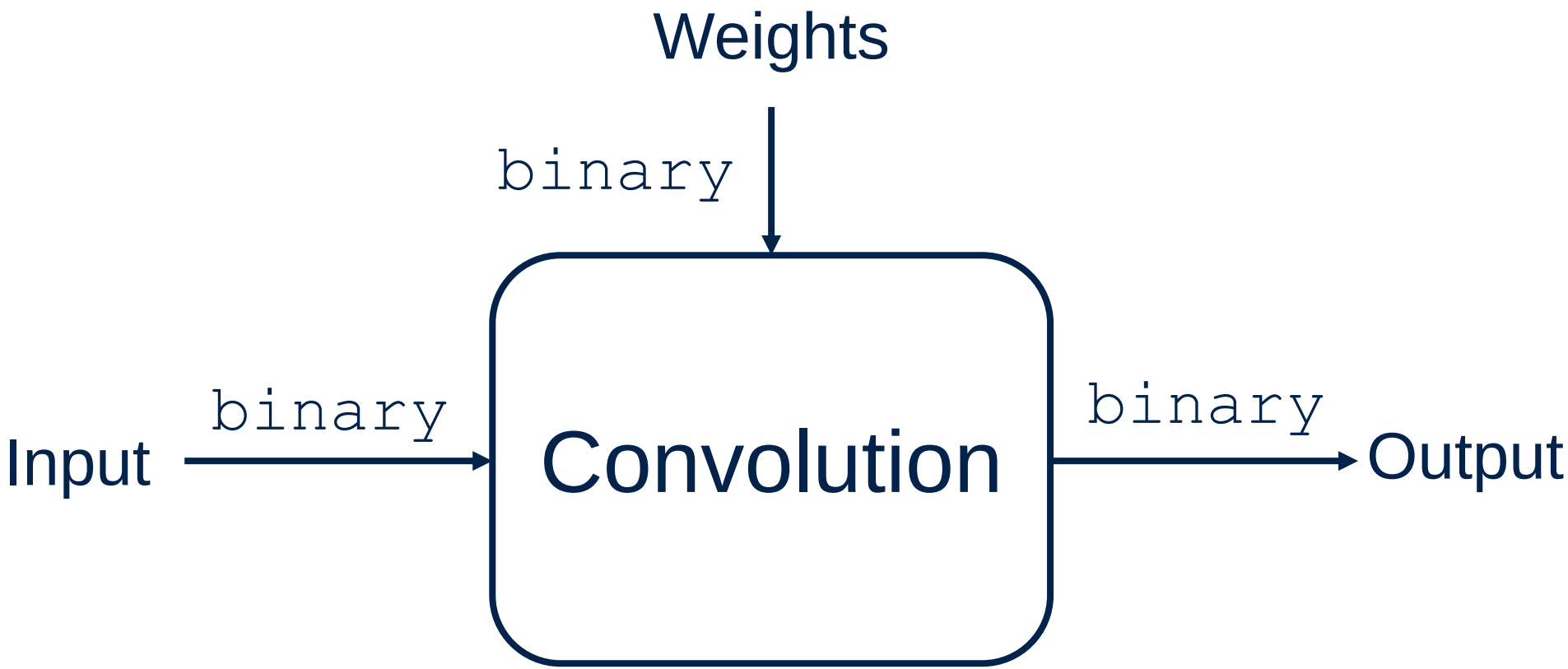


Heterogeneity: Quantization

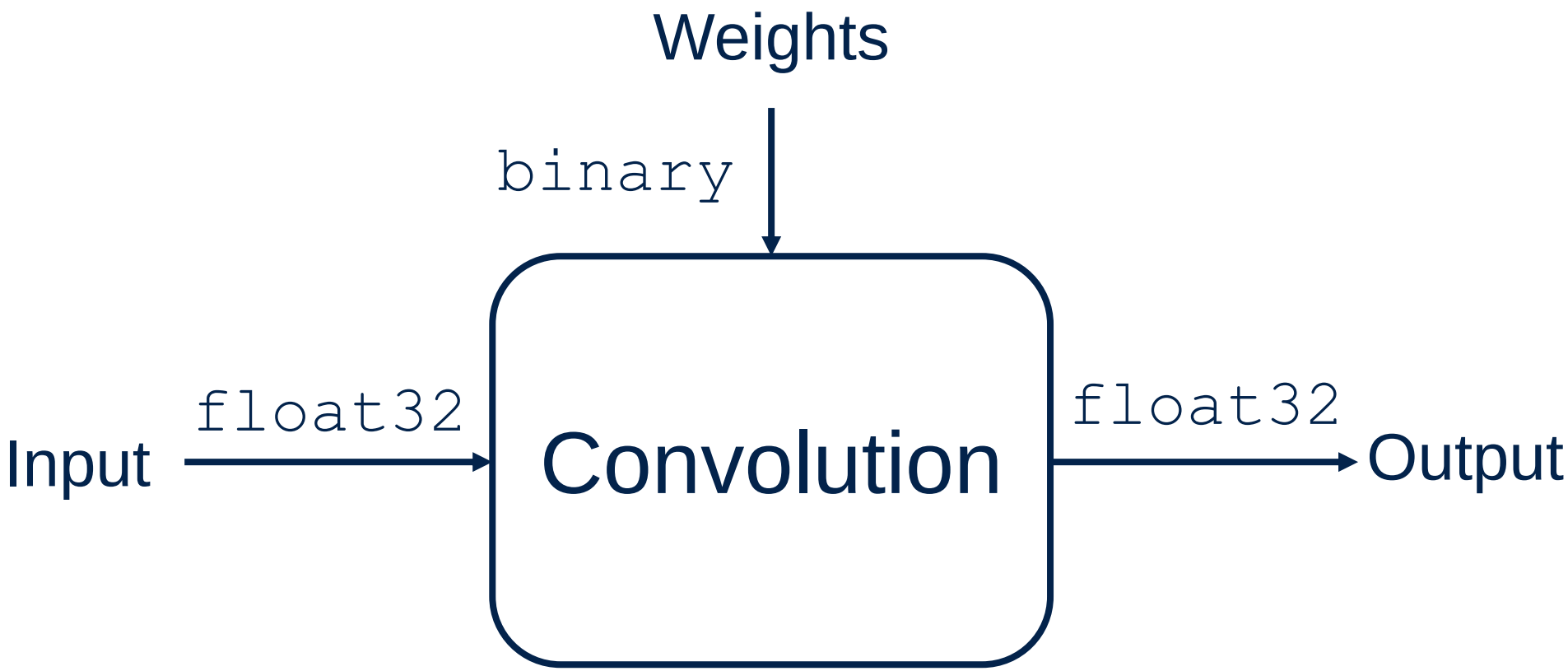
Heterogeneity: Quantization



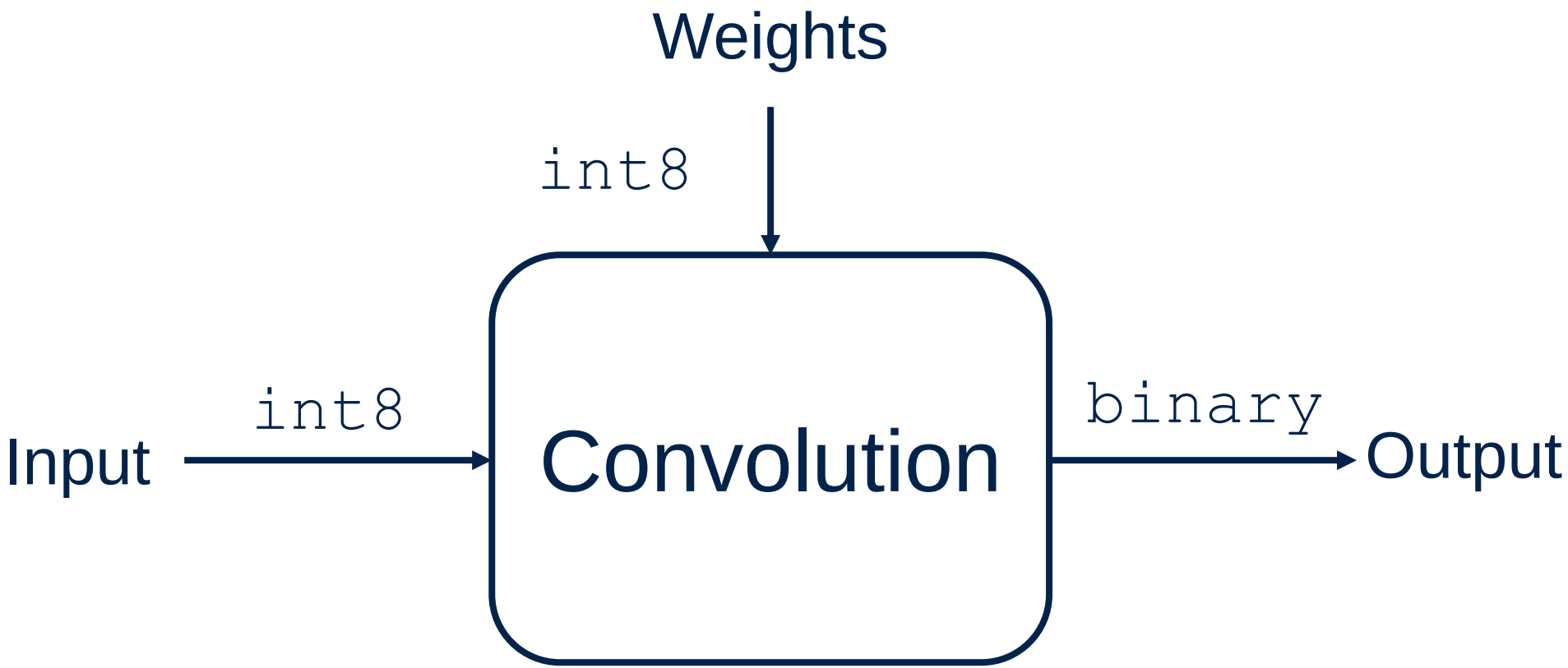
Heterogeneity: Quantization



Heterogeneity: Quantization



Heterogeneity: Quantization



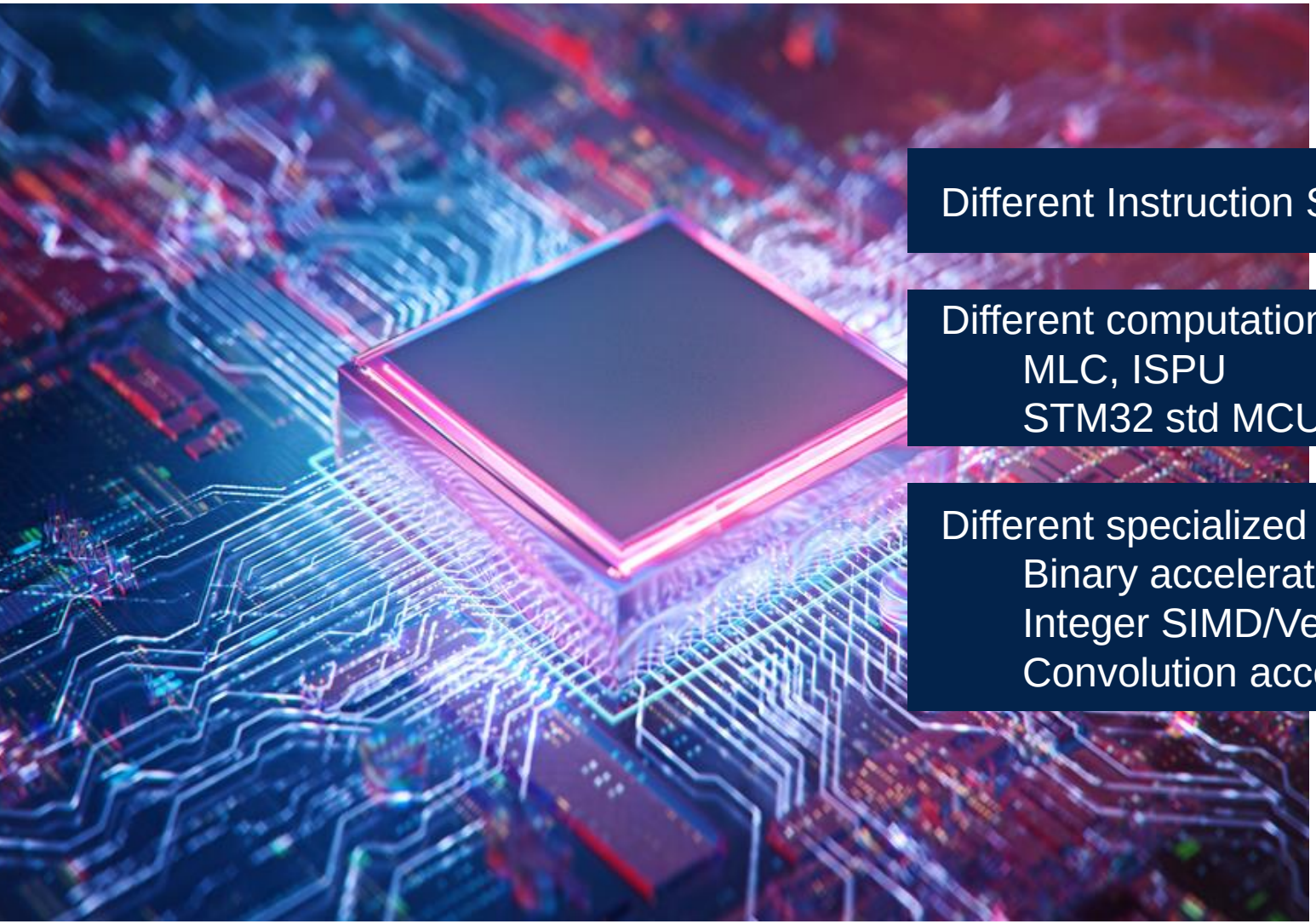
Heterogeneity: Quantization

Limiting to binary, int8, float32 there are 3^3=27 combinations to be supported for a single layer

Input

Output

Heterogeneity: Execution Targets

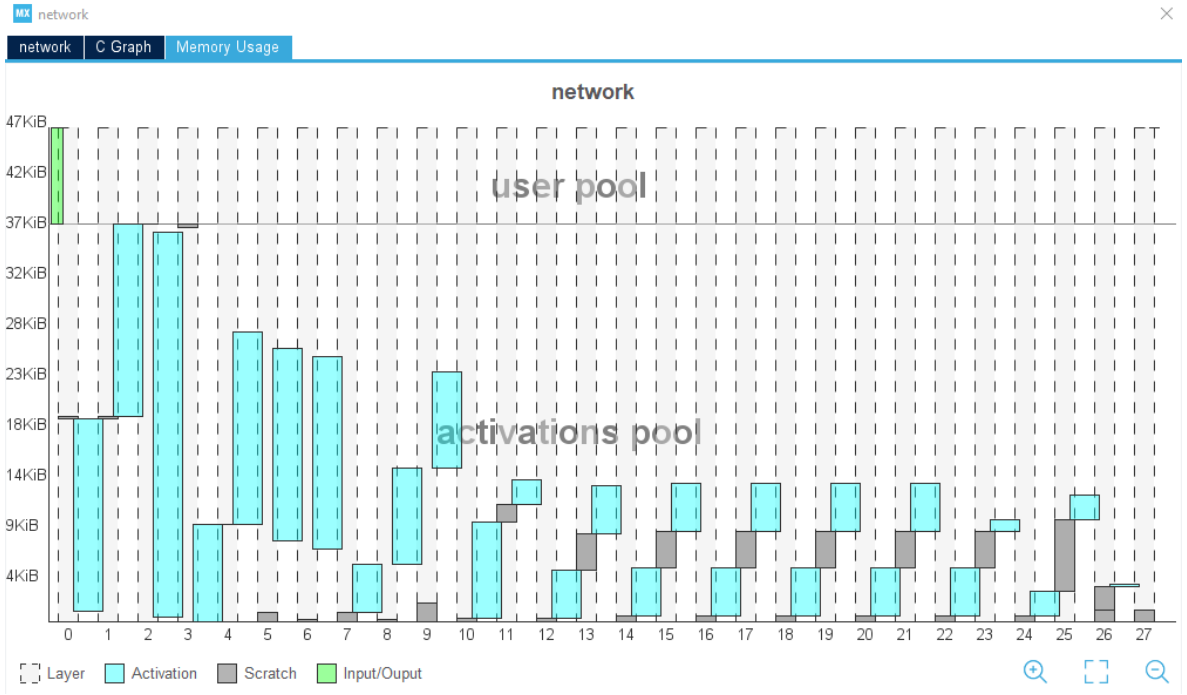


Different Instruction Sets (ARM Cortex M 0/4/7/33/55/85, STRed)

Different computational power, e.g.,
- MLC, ISPU
- STM32 std MCU family, STM32N6

Different specialized functional units, e.g.,
- Binary accelerator in ISPU
- Integer SIMD/Vector ISA in ARM
- Convolution accelerators in STM32N6

Heterogeneity: Execution Targets



Memory optimizer

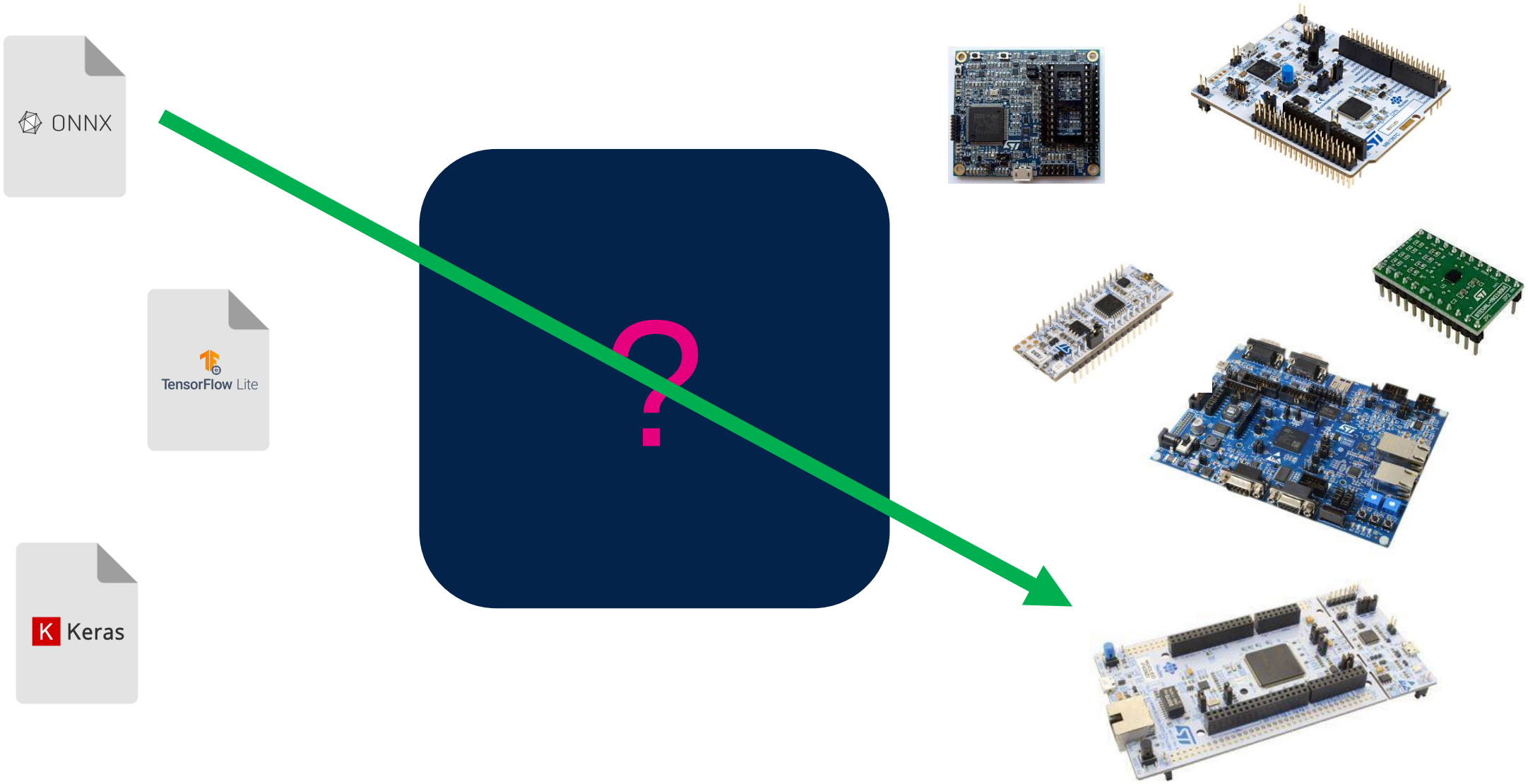
Optimize memory allocation to get the best performance while respecting the constraints of the embedded design

- Memory allocation
- Internal/external memory repartition
- Model-only update option

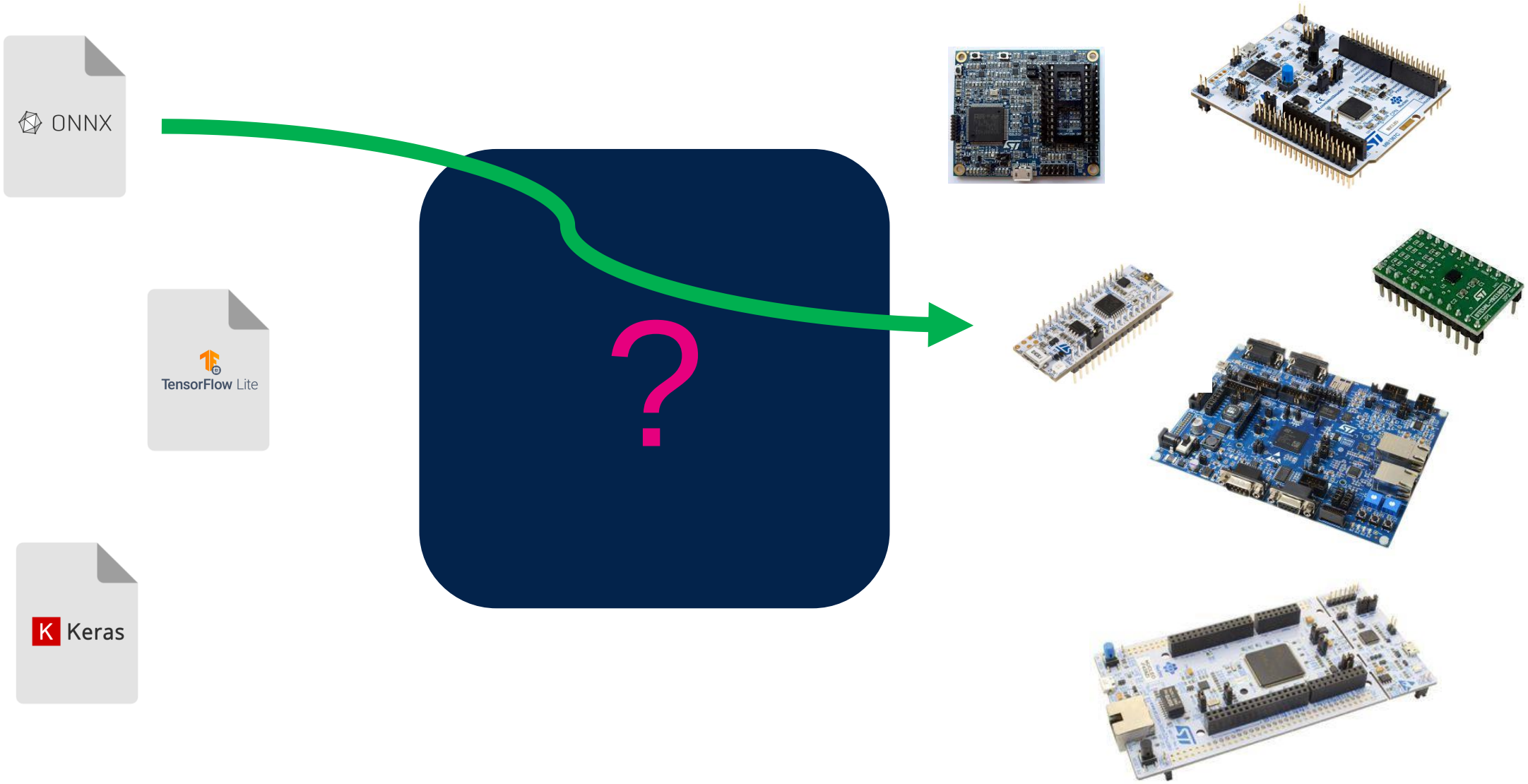
- Different memory infrastructure e.g.,
  - Single memory component
  - Multiple and homogeneous memories
  - Multiple and heterogeneous memories

life.augmented

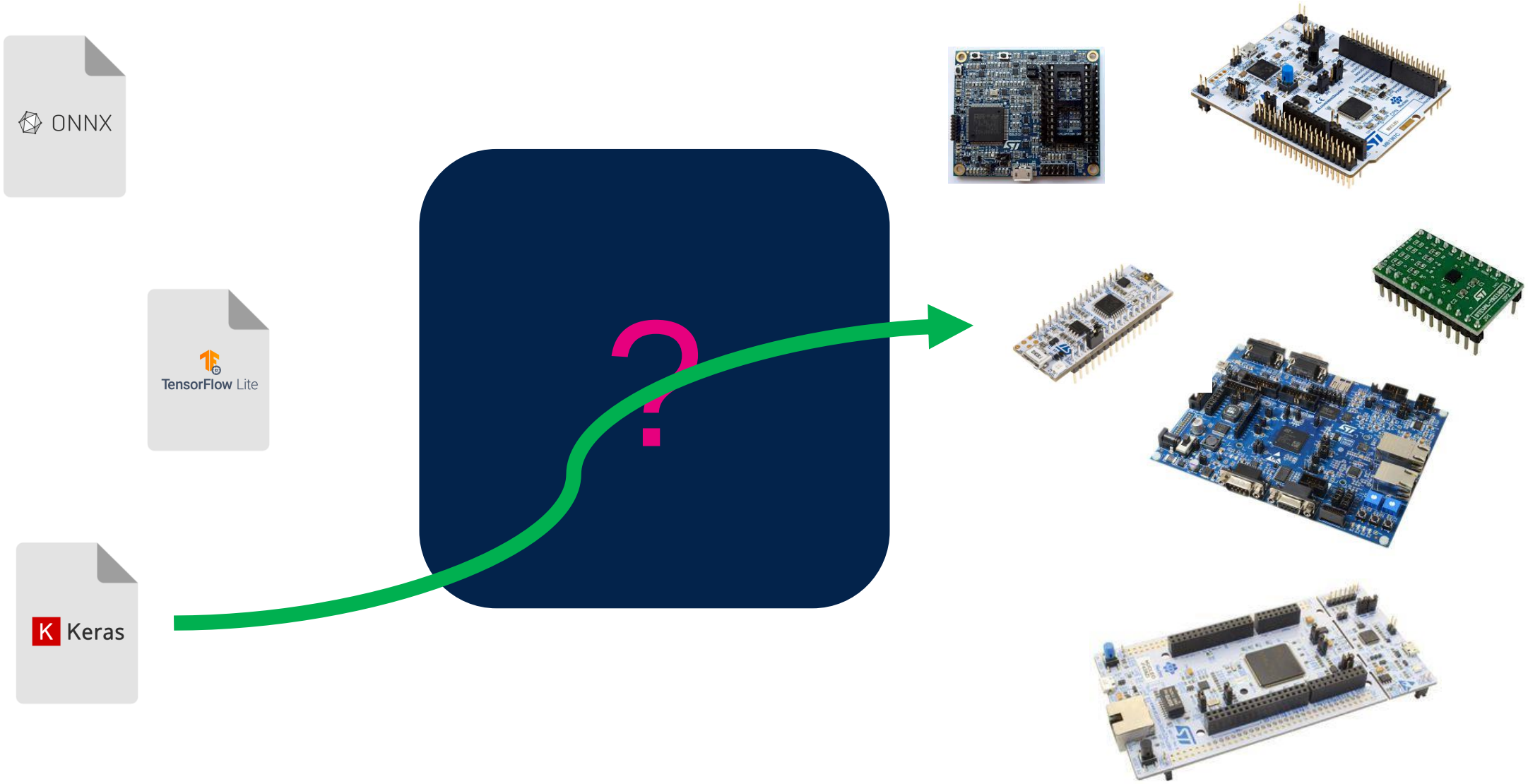
Paths from Source to Target



Paths from Source to Target

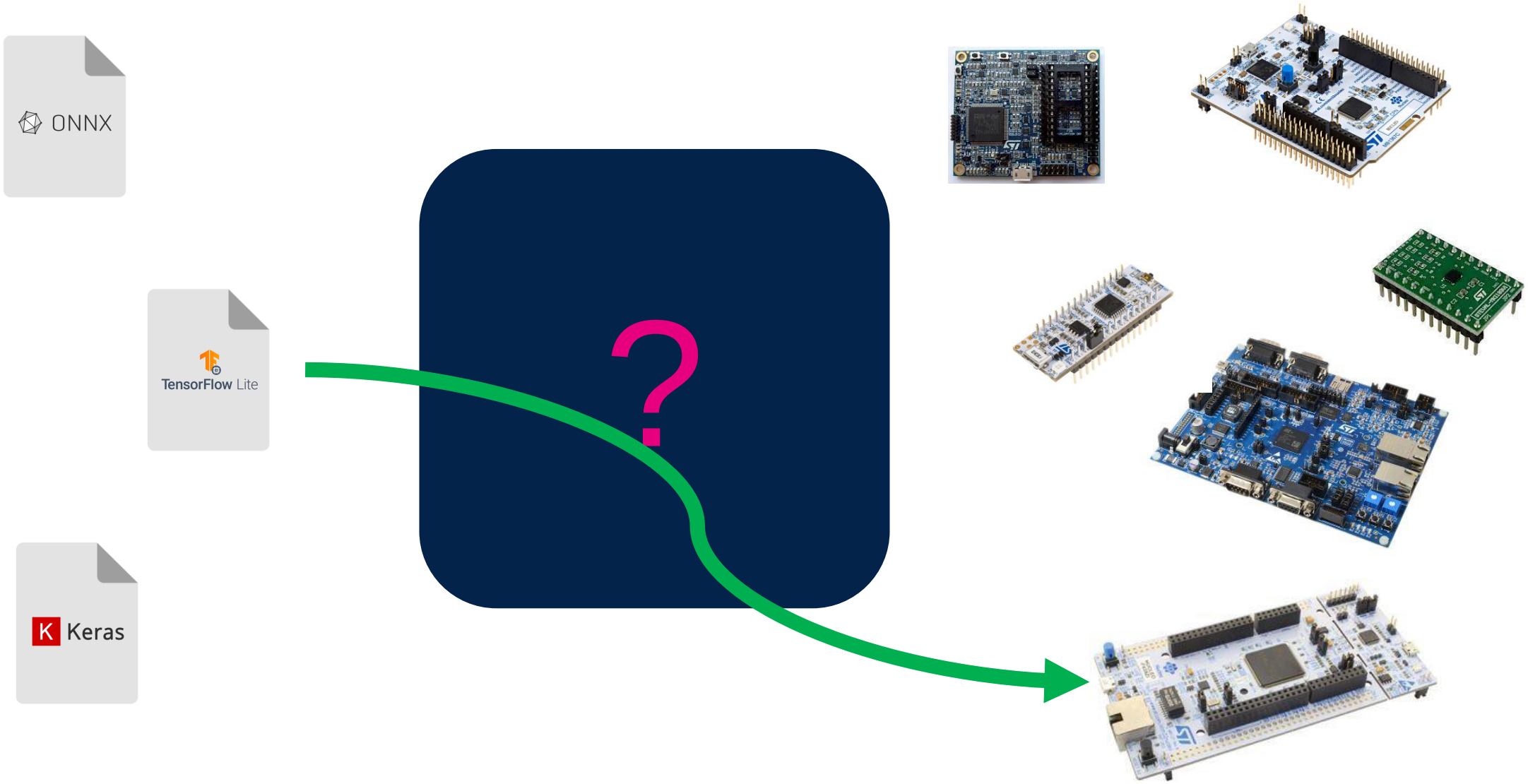


Paths from Source to Target

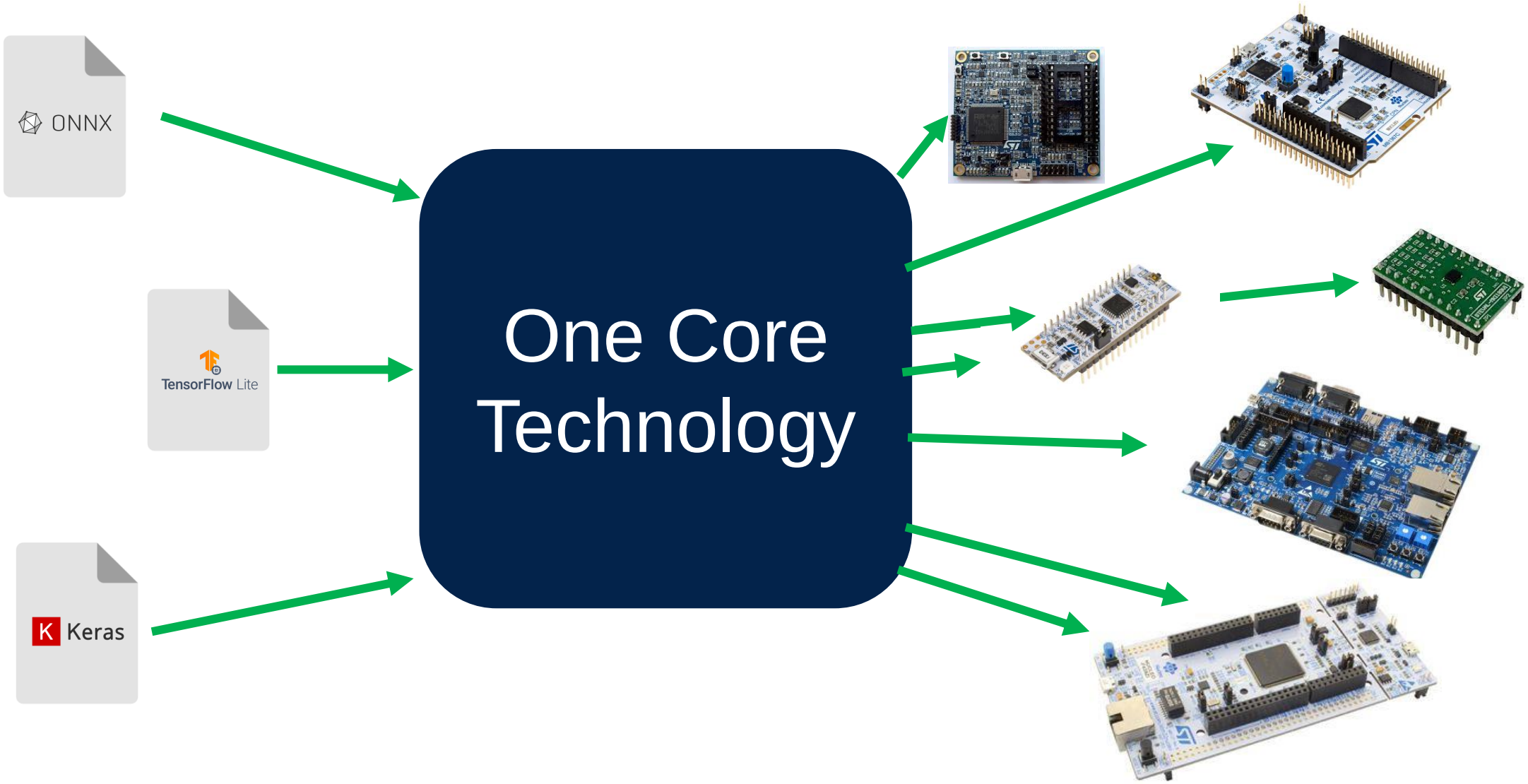


life.augmented

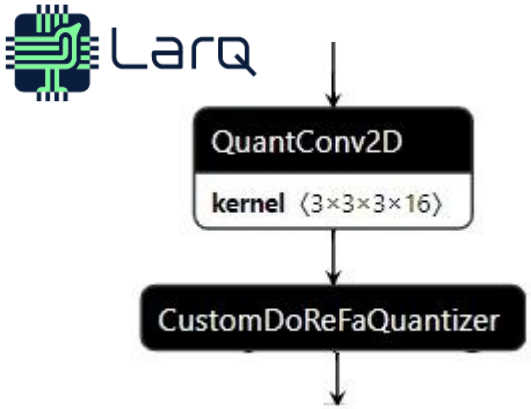
Paths from Source to Target

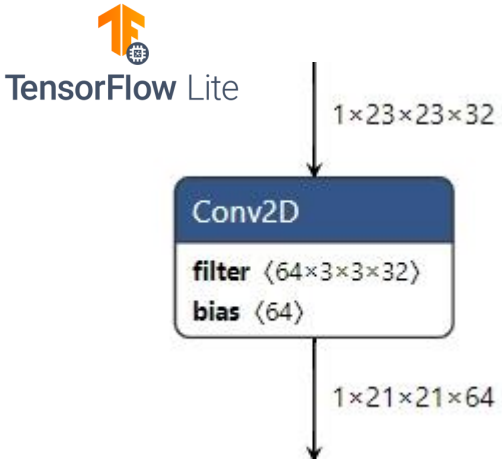


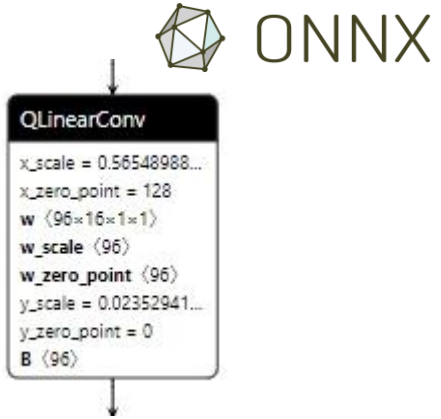
Homogenize Developer Experience

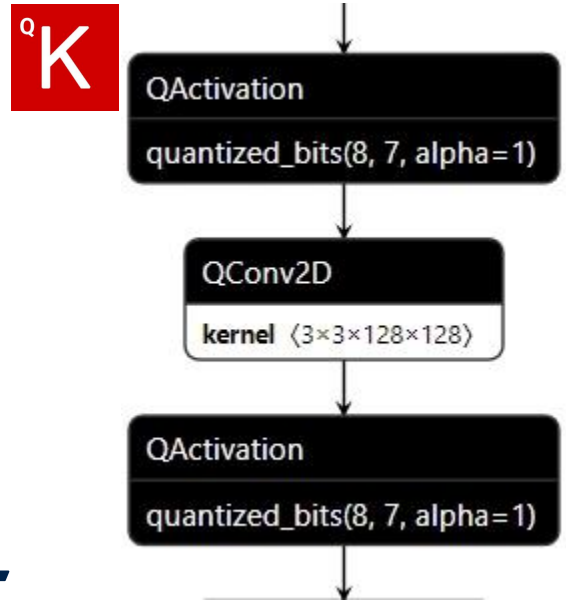


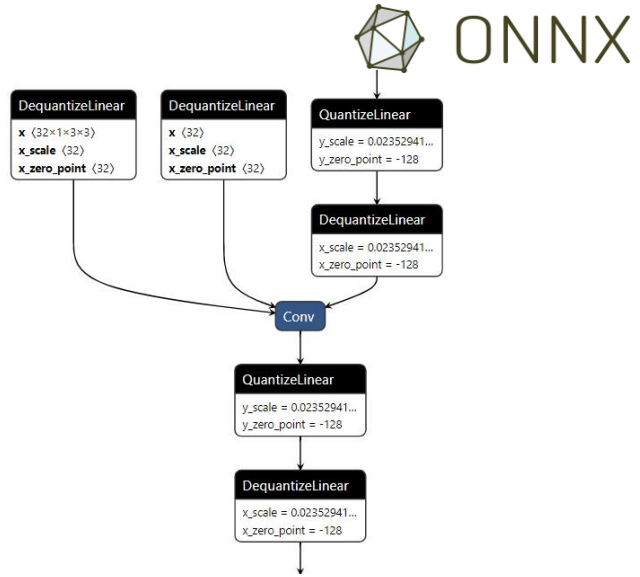
Removing Deep Learning Frameworks Differences





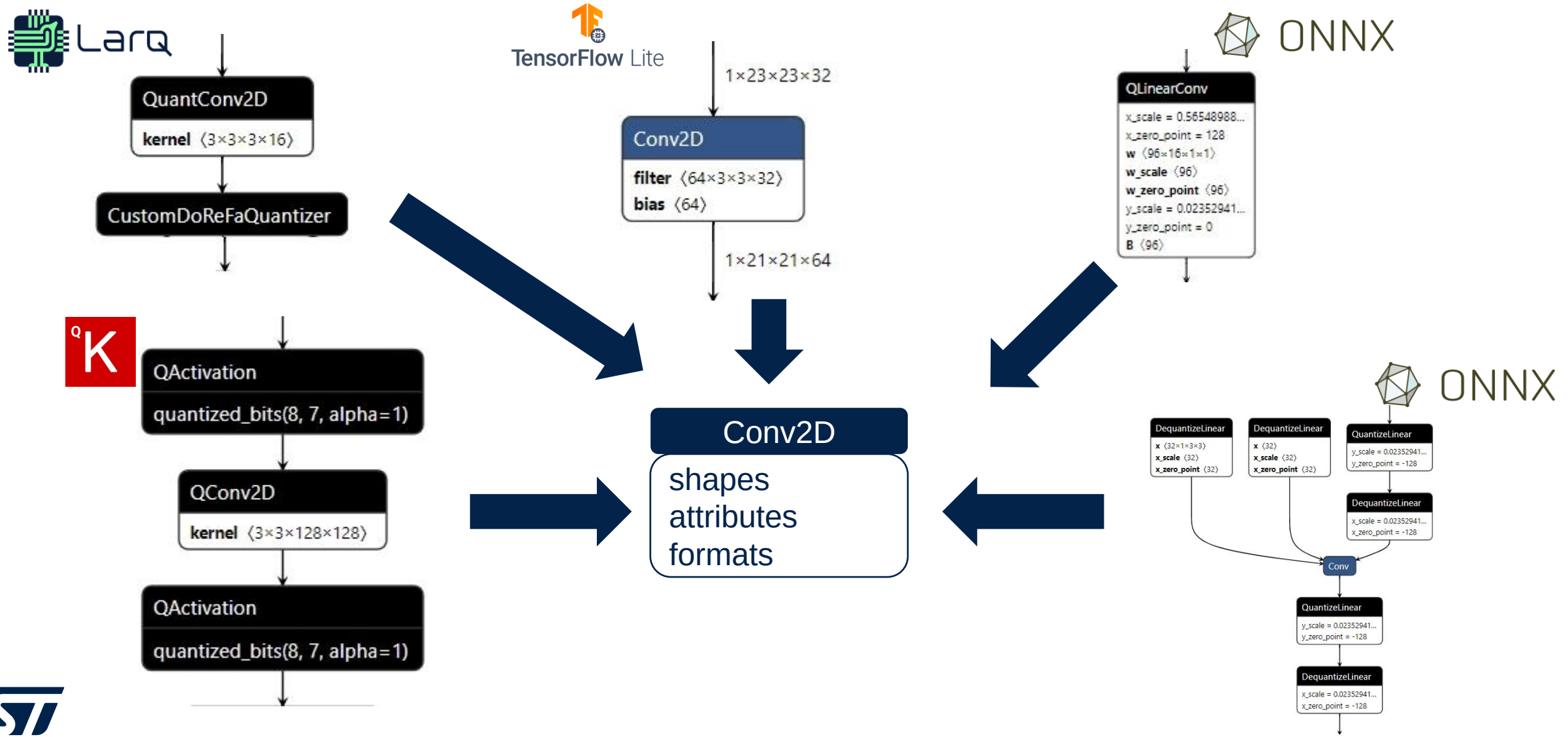






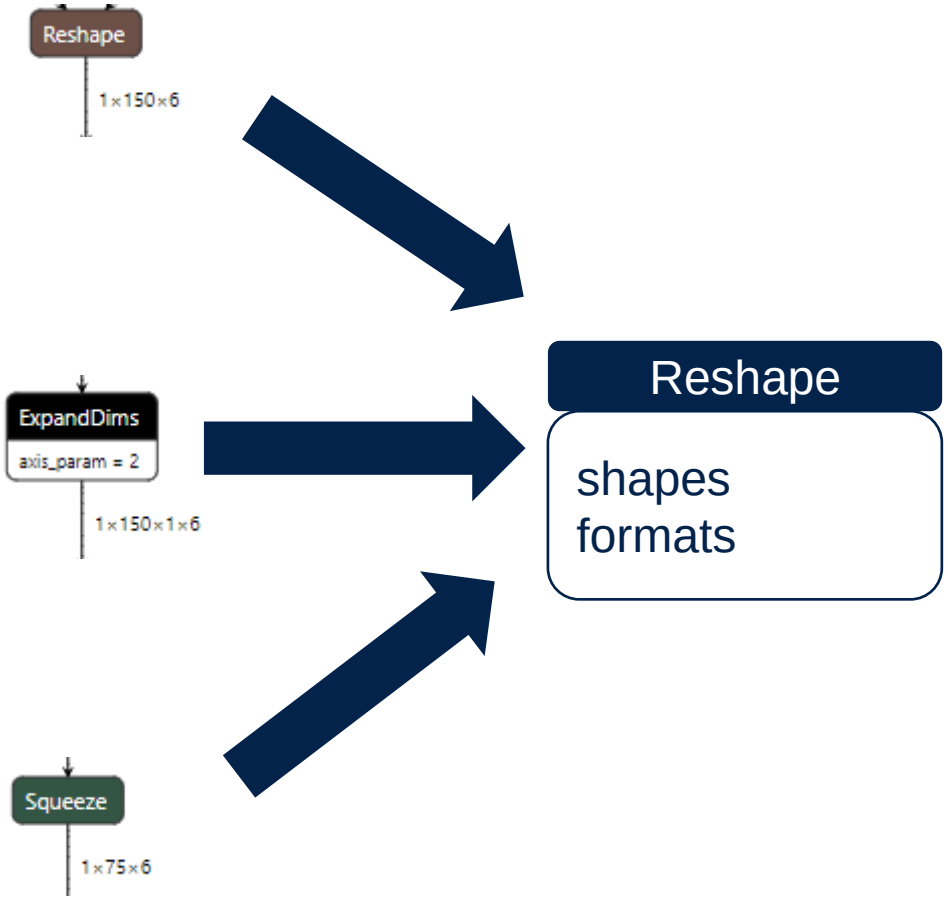
life.augmented

Removing Deep Learning Frameworks Differences

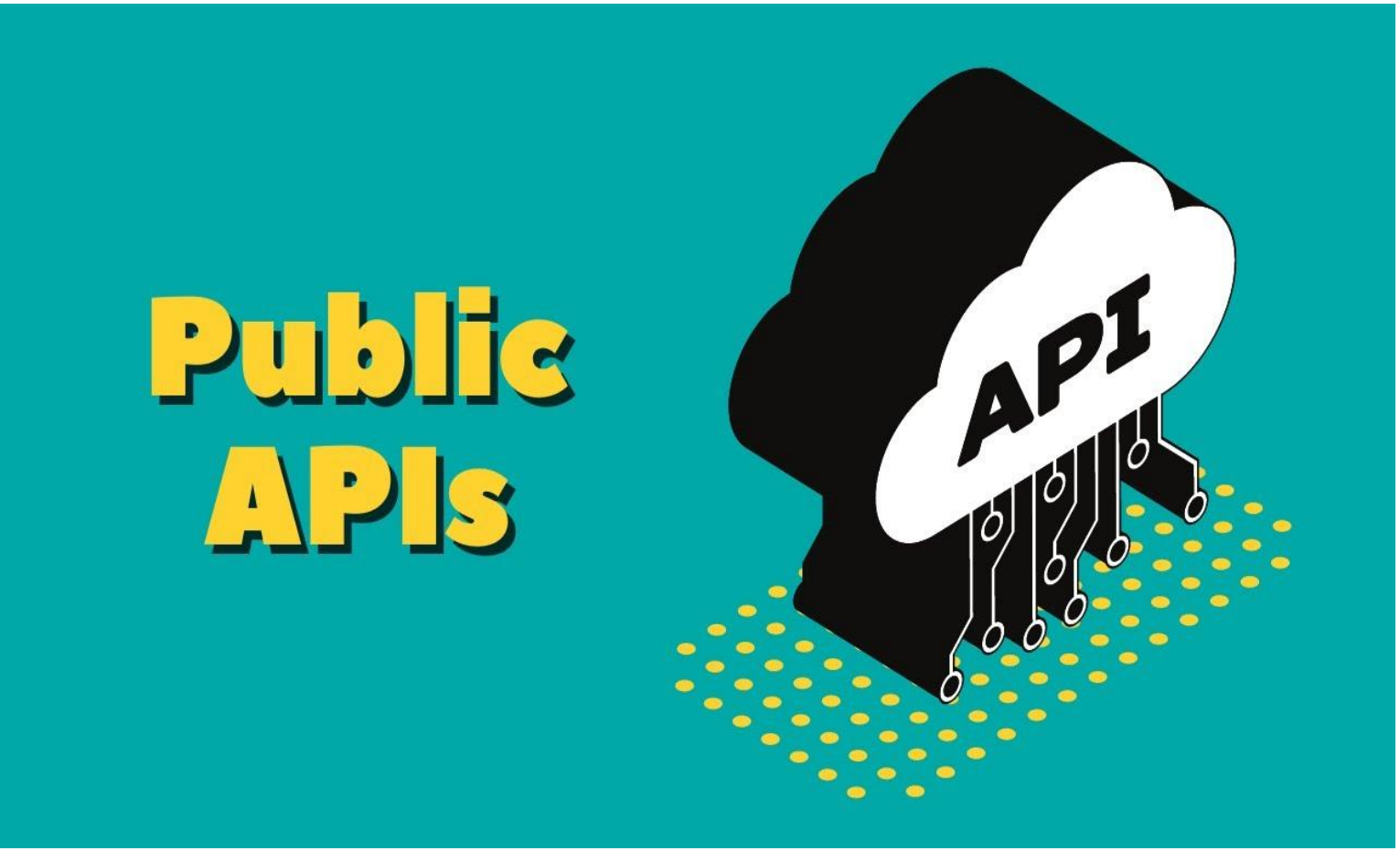


life.augmented

Removing Deep Learning Frameworks Differences

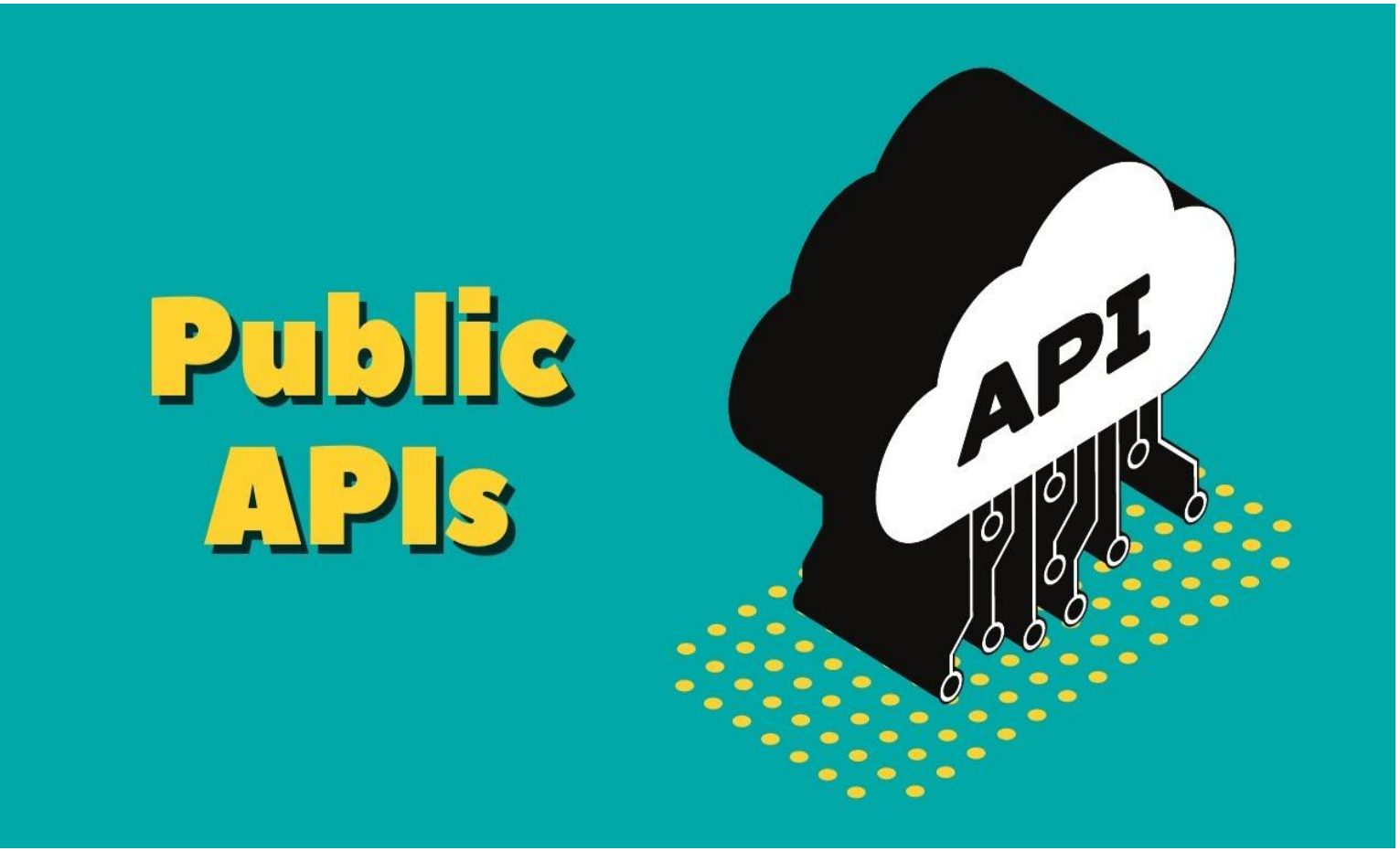


Hiding Target Differences



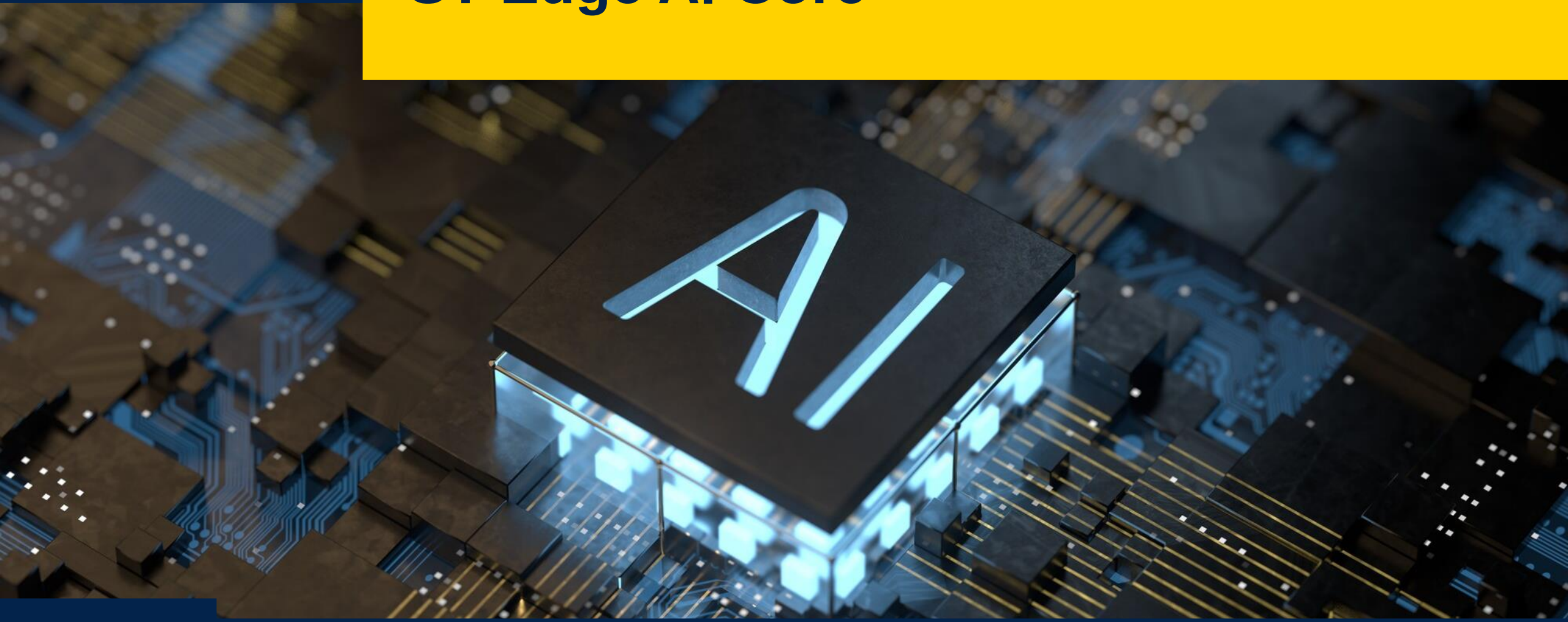
- **Same public APIs** for all the execution targets
  - Same application code means portability on all the supported target
- Internal code is optimized for the different targets exploiting the heterogeneous hardware capabilities
- **Extra APIs** to support advanced features of hardware

Hiding Target Differences



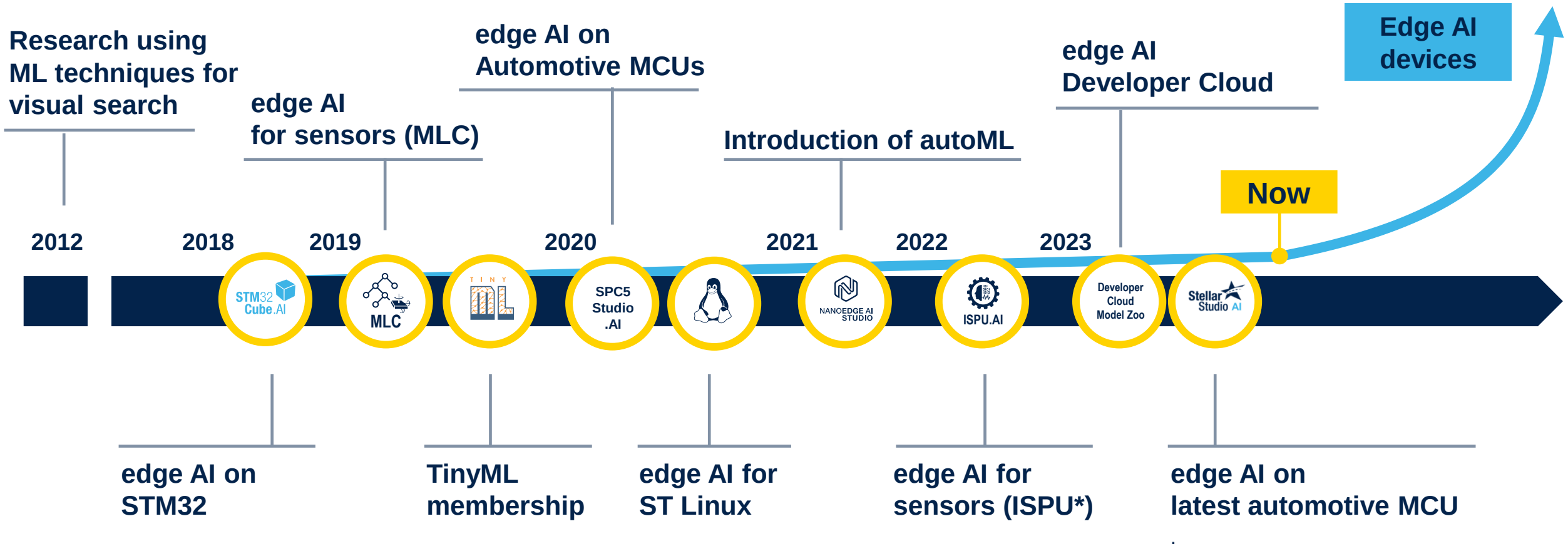
- Simplified public APIs: Init + run + deinit
- Different Optimization Objectives
  - **Time**: to minimize inference time
  - **RAM**: to minimize use of memory
  - **Balanced**: trade-off between inference time and memory usage
- Multiple networks instancing in the same application

ST Edge AI Core



life.augmented

More than a decade of research, development, and deployment



*Intelligent Sensor Processing Unit

Announcing the ST Edge AI Suite

The most complete developer-centric approach to accelerate the deployment of edge AI

Ideation
Model Zoo

Data capture
Dataset creation

Bring Your Own Data OR **Bring Your Own Model**

AutoML
End-to-end flow

Model training*
& optimization

*via partner's ecosystem

Benchmarking

Deployment

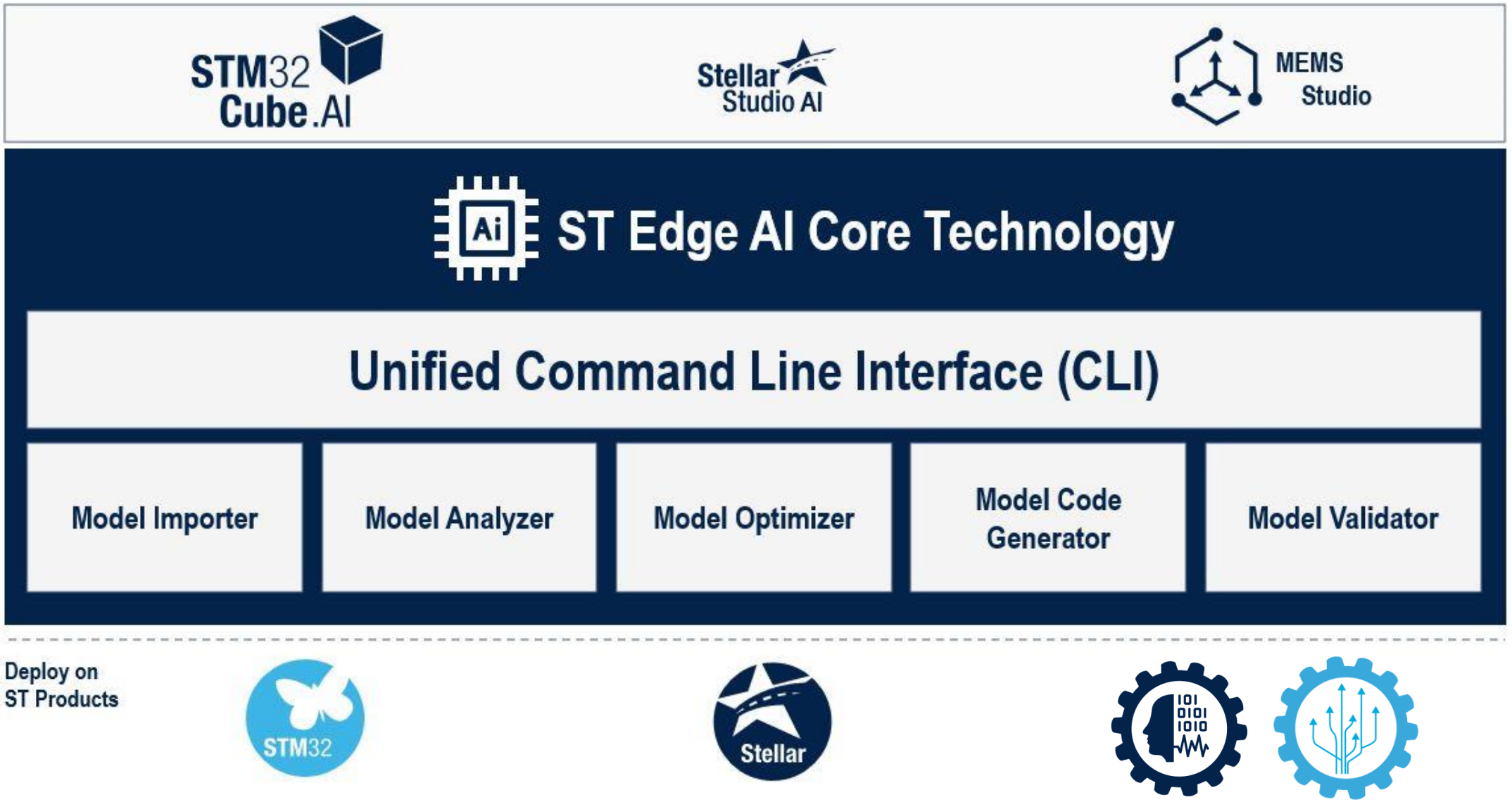
Edge AI Suite

Edge AI Resources

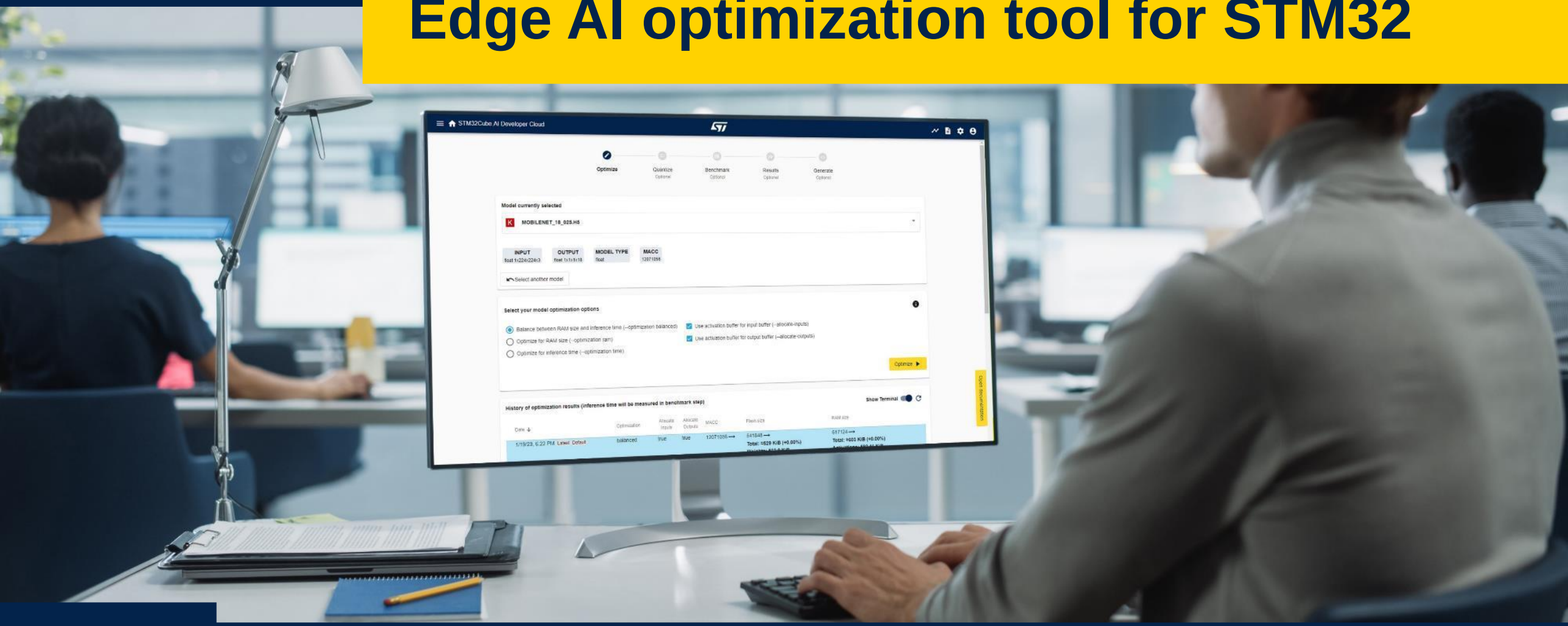
- Online documentation
- Tutorials and trainings
- Community

life.augmented

ST Edge AI Core



STM32Cube.AI
Edge AI optimization tool for STM32



life.augmented

STM32Cube.AI solutions

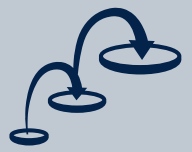
X-CUBE-AI
STM32CubeMX extension package

Developer Cloud
Online version of the tool

STM32 board farm
Online performance benchmarking facilities

STM32 Cube.AI

STM32 model zoo
Dedicated GitHub repo



Getting started
Tutorial + code examples

Partners
Dedicated partner ecosystem

life.augmented

STM32Cube.AI: your go-to solution for edge AI

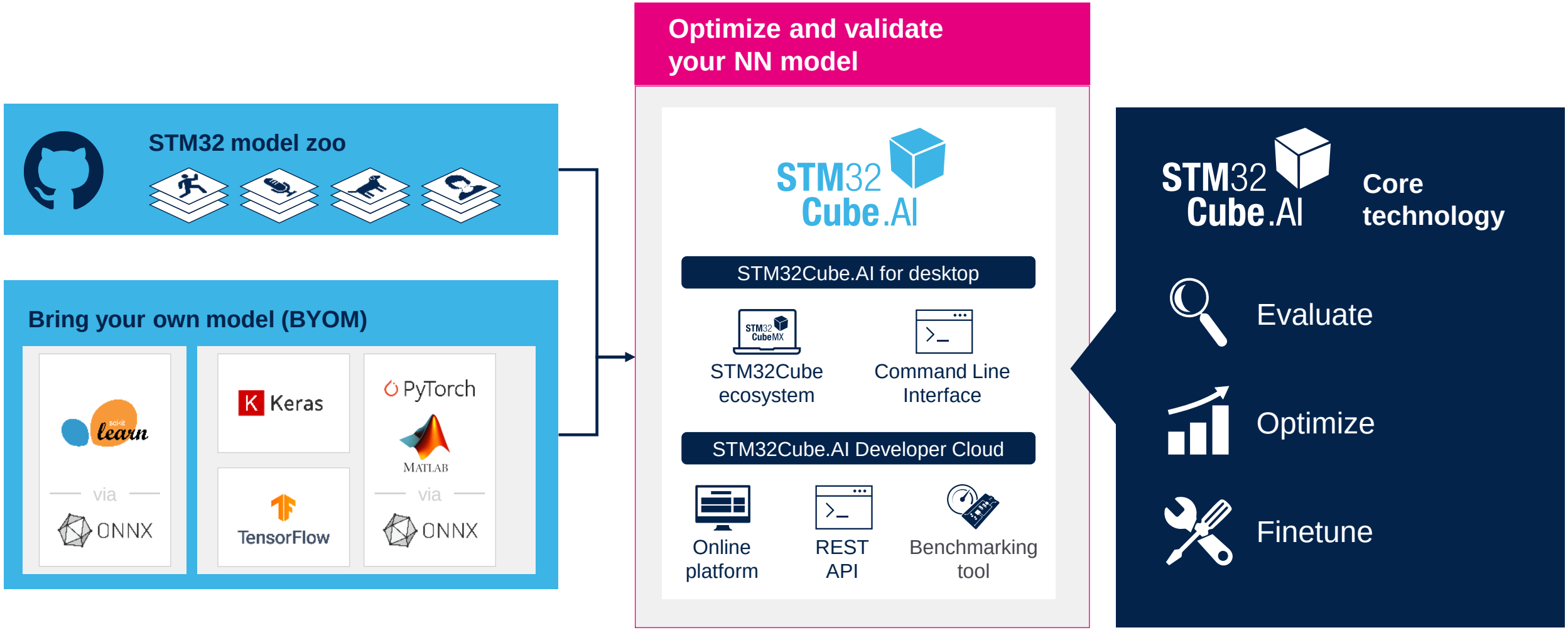
Fully **integrated** in the STM32Cube software development suite

Cutting-edge **performance,** optimized for STM32 hardware

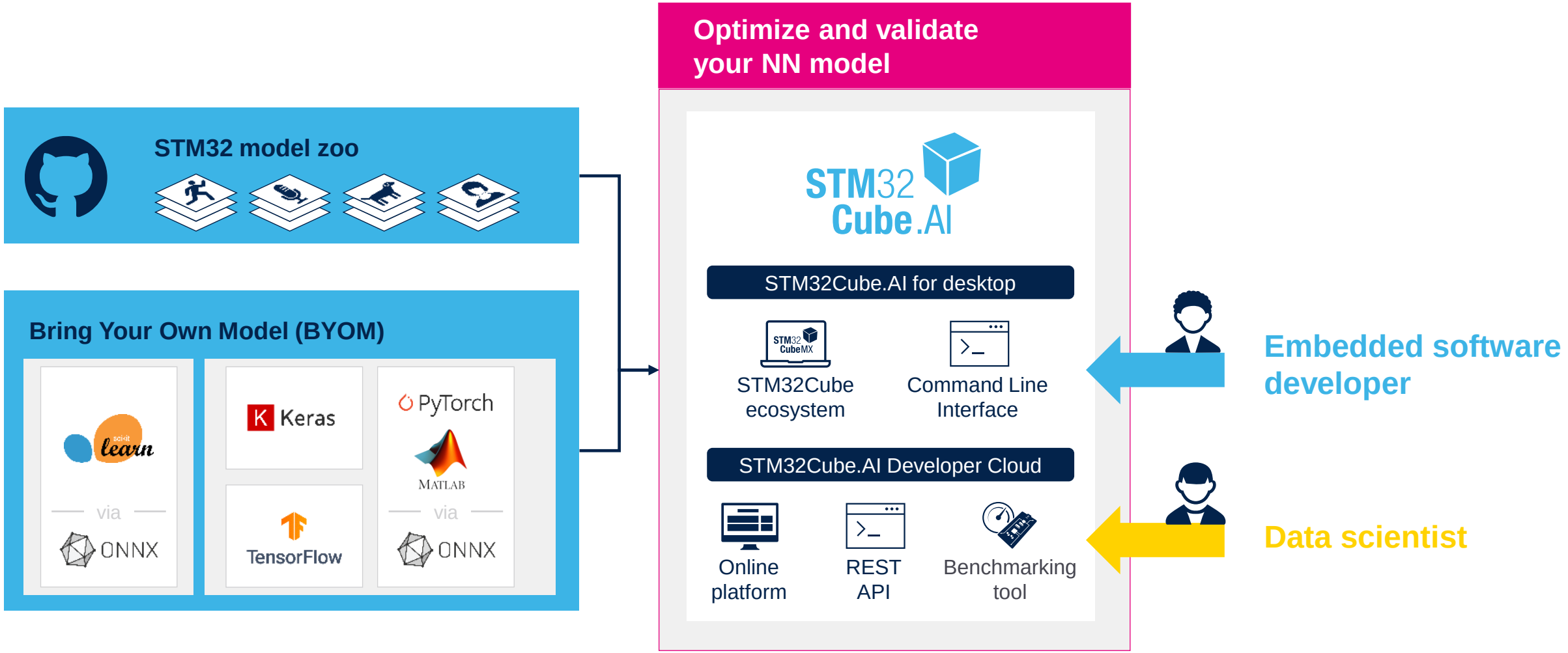
Completely **FREE** of charge

life.augmented

STM32Cube.AI core technology



Two versions of the same tool depending on your profile



STM32 benchmarking tool

The unique possibility to evaluate the performance of models remotely, on real STM32 boards

Get the real inference time from optimized models running on STM32

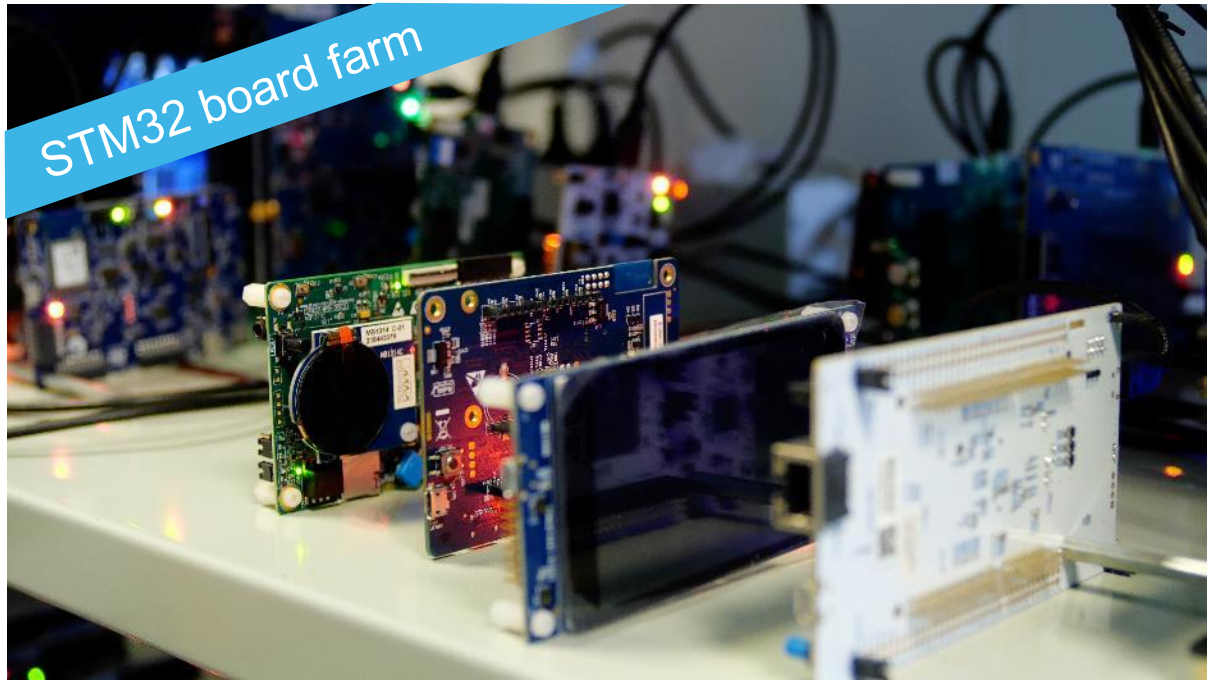
Benchmark models on a large variety of STM32 boards

Find the most appropriate board for your application

NEW

Get access to the most recent devices

A board farm is constantly updated with the latest available boards



life.augmented

STM32 model zoo

A collection of application-oriented models optimized for STM32

Human activity

Motion Sensing

Image classification

Computer vision

Audio event detection

Audio classification

Object detection

Computer vision

Hosted on GitHub

Model training scripts

- Scripts to generate and validate

Getting started application packages

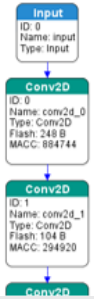
- Automatically generated from the trained models
- Easy to deploy for end-to-end evaluation

life.augmented

The 3 pillars of STM32Cube.AI

Graph optimizer

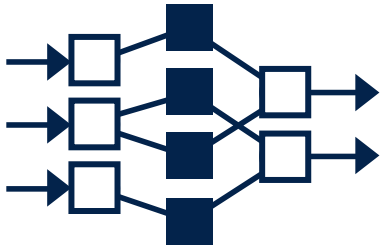
Automatically improve performance through graph simplifications & optimizations that benefit STM32 target HW architectures



- Auto graph rewrite
- Node/operator fusion
- Layout optimization
- Constant-folding…
- Operator-level info to finetune memory footprint and computation

Quantized model support

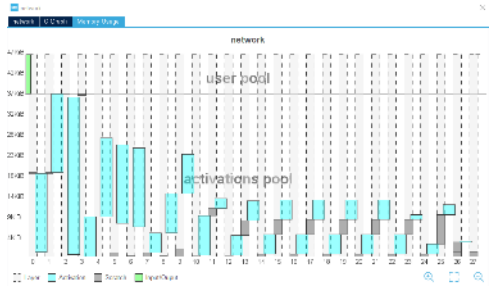
Import your quantized ANN to be compatible with STM32 embedded architectures while keeping their performance



- From FP32 to Int8 or mixed-precision
- Minimum loss of accuracy
- Code validation on target
  - Latency
  - Accuracy
  - Memory footprint

Memory optimizer

Optimize memory allocation to get the best performance while respecting the constraints of your embedded design

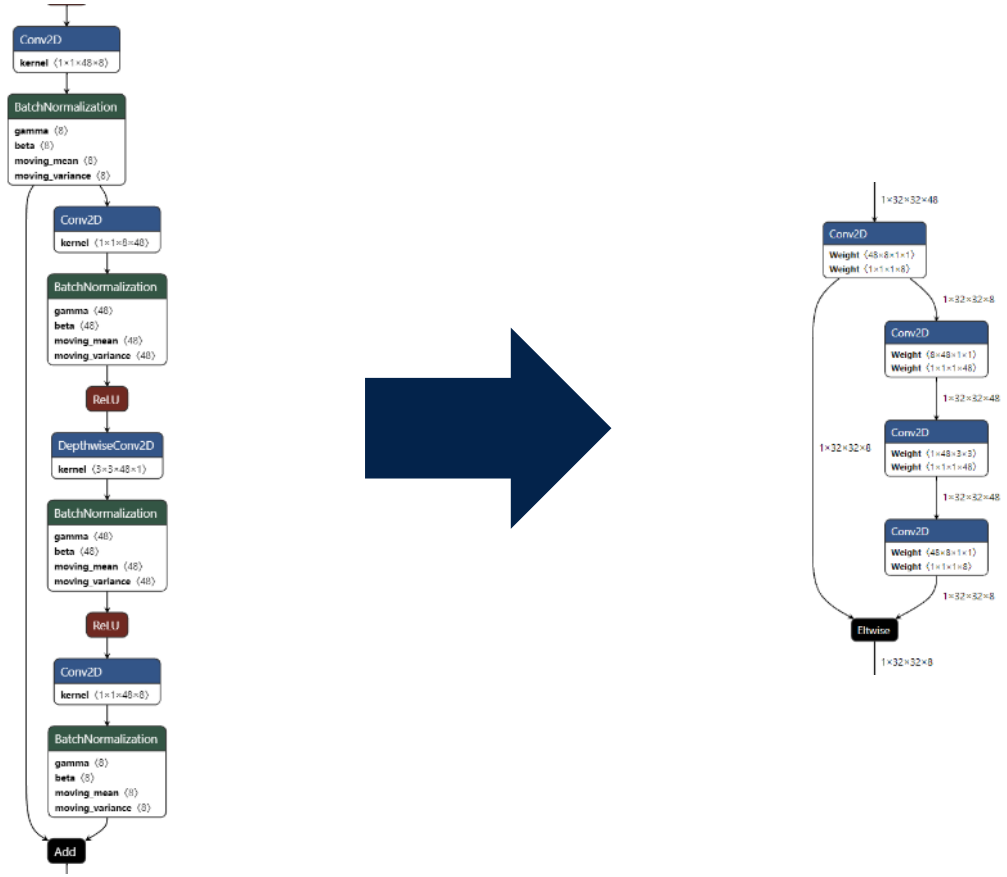


- Memory allocation
- Internal/external memory repartition
- Model-only update option

STM32Cube.AI is **free of charge**, available both in graphical interface and in command line.

Graph optimizer

Squeeze your graph to fit into an MCU!



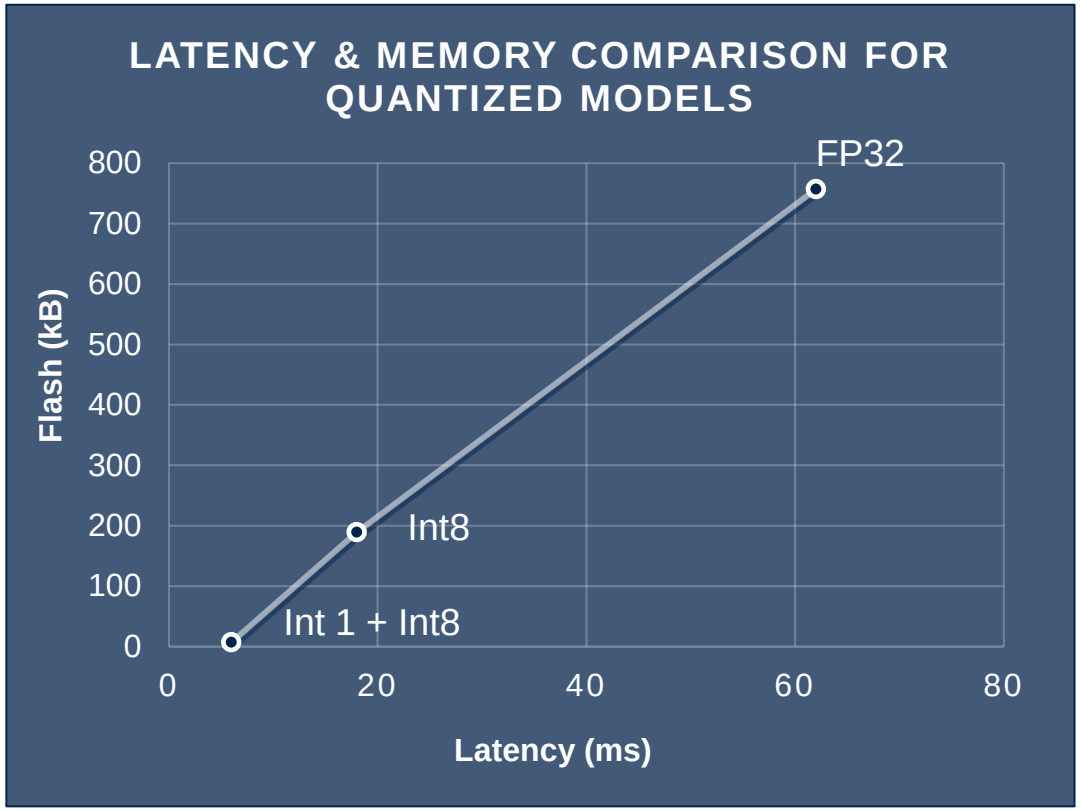
Fully automated process in the STM32Cube.AI workflow

- Your original graph is optimized at the very early stage for optimal integration into the STM32 MCU/MPU
- Loss-less conversion

life.augmented

Quantized model support

Simply use quantized networks to reduce memory footprint and inference time



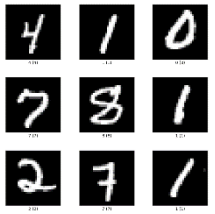
STM32Cube.AI supports quantized neural network models with **all parameter formats**:

- FP32
- Int8
- Mixed binary Int1 to Int8 (Qkeras*, Larq.dev*)

**Please contact edge.ai@st.com to request the relevant version of STM32Cube.AI*

HW Target: NUCLEO-STM32H743ZI2
Model: Low complexity handwritten digit reading
Freq: 480 MHz
Accuracy: >97% for all quantized models

Tested database: MNIST dataset

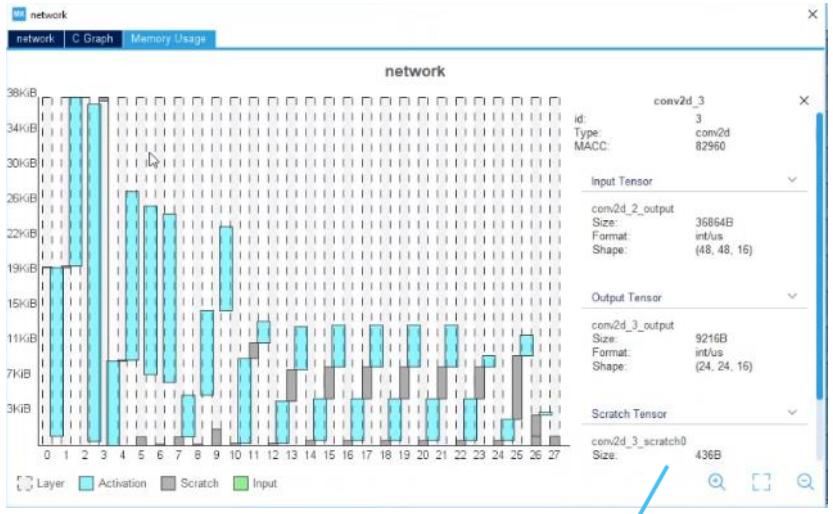


MNIST dataset

life.augmented

Memory optimizer

Optimize performance easily with the memory allocation tool



Model RAM consumption per layer

- Easily identify the most critical layers

Model memory allocation

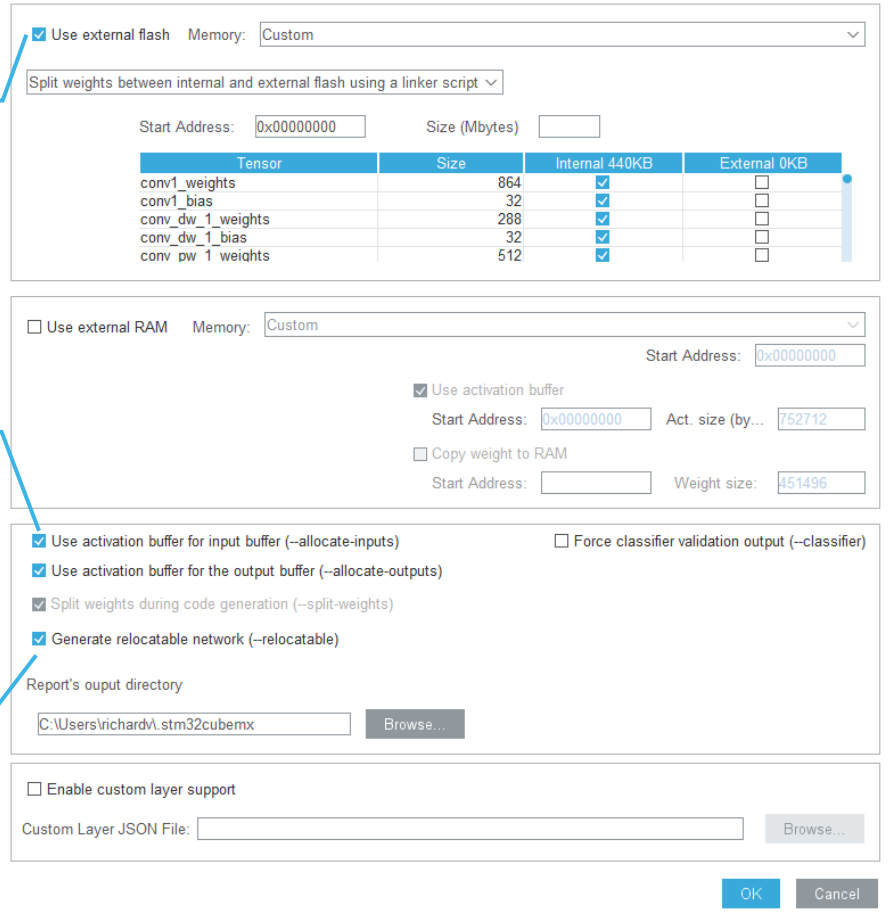
- Set your external memory
- Map in non-contiguous internal flash section
- Partition internal vs external flash memories

Re-use model input buffer to store activation data*

- Minimize RAM requirements

Relocatable network

- A separate binary is generated for the library and the network to enable standalone model upgrade

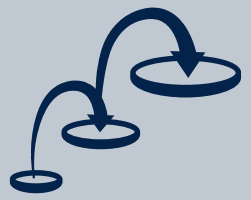


** Requires input and activation buffers in same memory*

life.augmented

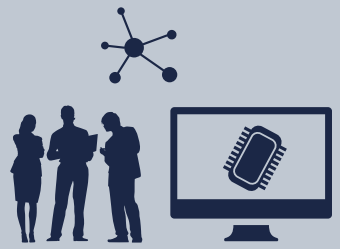
We provide everything to kick off your project

Design documentation



Getting started

We guide you step-by-step to start with the STM32 ecosystem

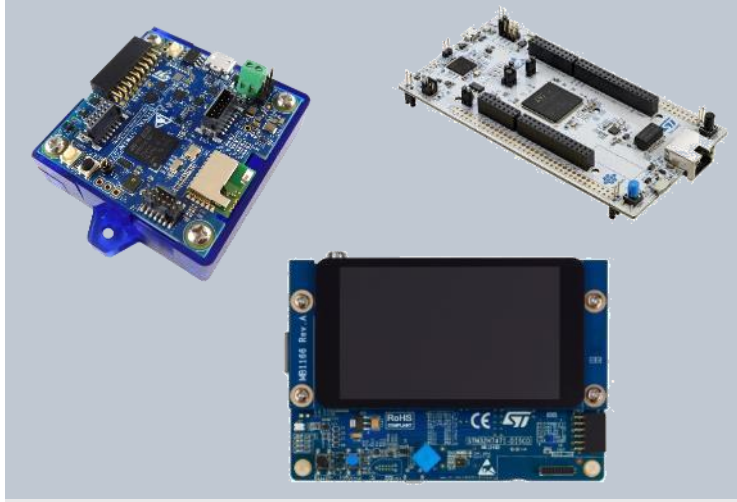


Development zone

Get started on application development and project sharing

- **Wiki by ST** is a great forum to learn and start deploying edge AI on STM32!
- Videos of application examples
- Massive open online courses (MOOCs)

Hardware and software tools



- Evaluation platforms for STM32 MCUs and MPUs
- Additional sensor boards
- Full software suite

Support and updates

- **ST Community**: STM32 ML & AI group
- Distributor certified FAE
- Support center
- Newsletter

life.augmented

Next Challenges

- On-Device Learning
- Zero Code
- Generative AI on the Edge

life.augmented

Question Session

Conclusion

“Tiny Machine Learning is not just a technology, it's a mindset that empowers us to build intelligent systems that can make a positive impact on the world."

Pete Warden, CEO at Useful Sensors (previously Technical Lead at TensorFlow Lite for Microcontrollers)

life.augmented

Our technology starts with You

Find out more at www.st.com

life.augmented