Deep Learning(TinyML)

Bringing Intelligence to the Edge

Chiraz Rayene Harkati

ML Engineer

December, 2025

Agenda

- AI "Born at the Edge"
- Embedded AI optimization
  - Pruning
  - Quantization
  - Distillation
  - NAS
- AI at NXP
- Lab

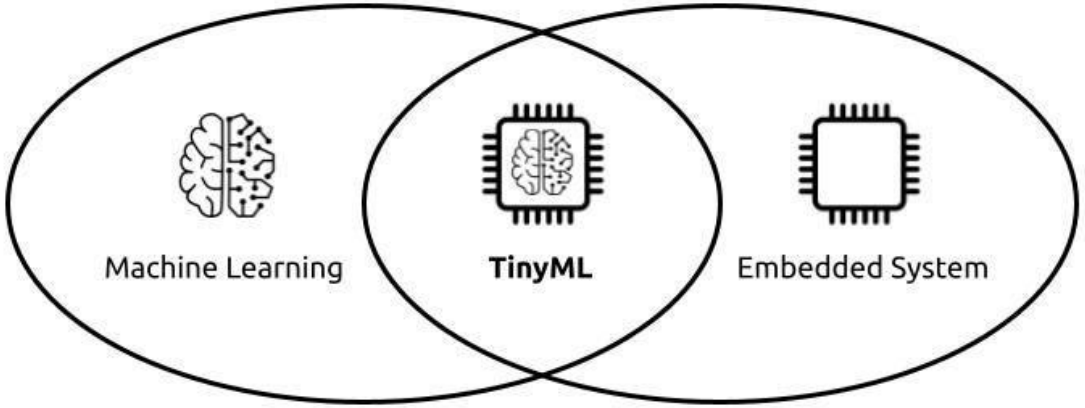
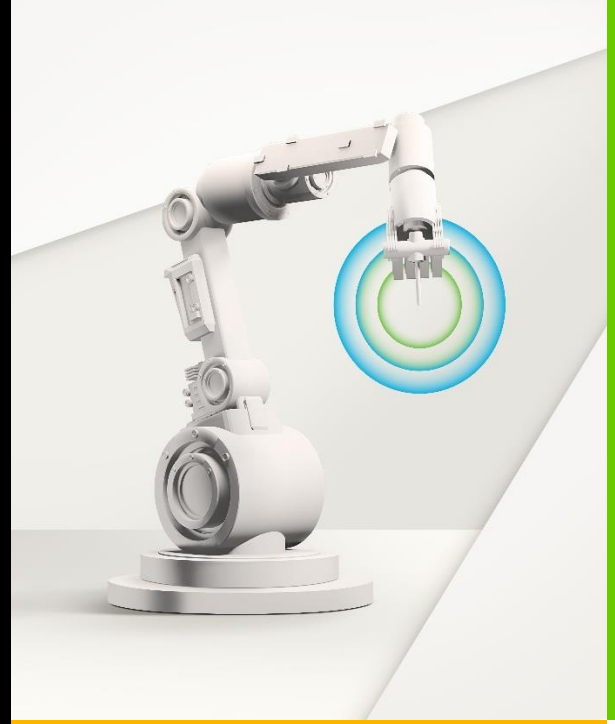


Image: https://leonardocavagnis.medium.com/tinyml-machine-learning-for-embedded-system-part-i-92a34529e899

03

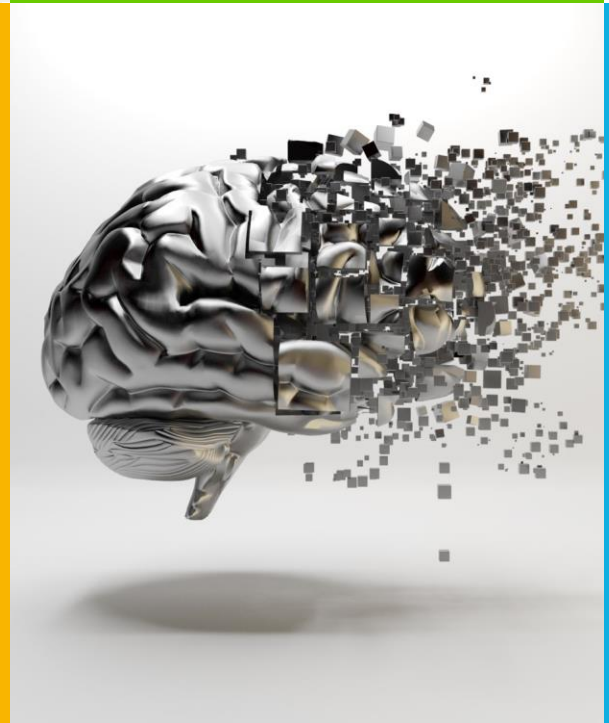
AI Born at the Edge

The ability to run AI at the edge has emerged as a transformative force.



- Real-time decisions
- Energy efficiency

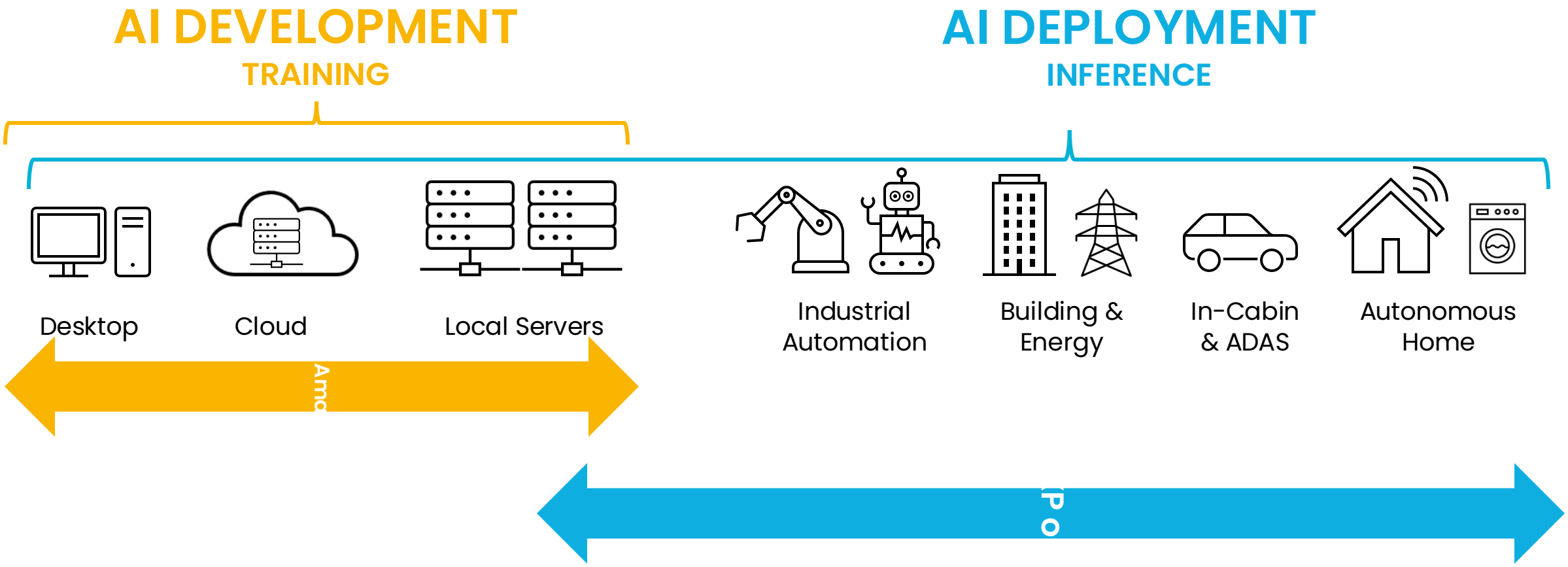
- Data privacy
- $ Cost effective
- Scalable



- Human like interactions
- Contextual understanding
- Smarter, data-driven decisions

AI spans from training in the cloud to inference at the edge

NXP's strength in processing gives us a unique opportunity to shape the deployment of AI at the edge.



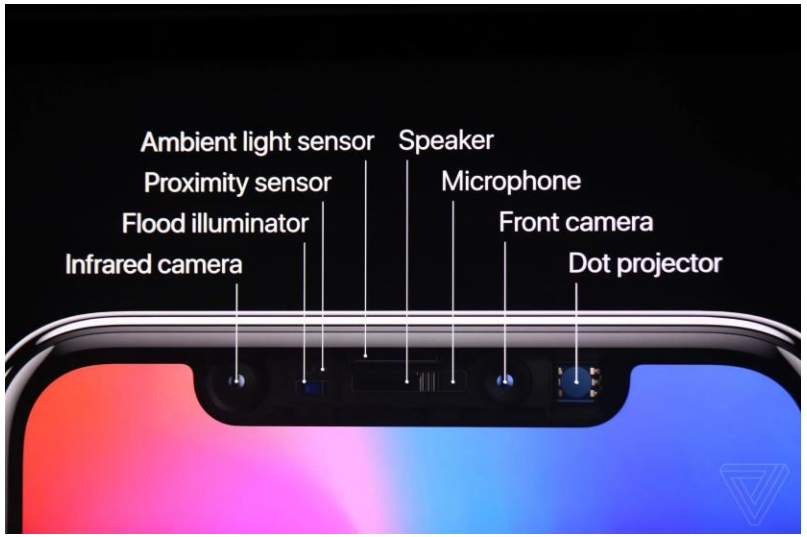
Real world examples

Cars

Waymo: 100% autonomous since 2019

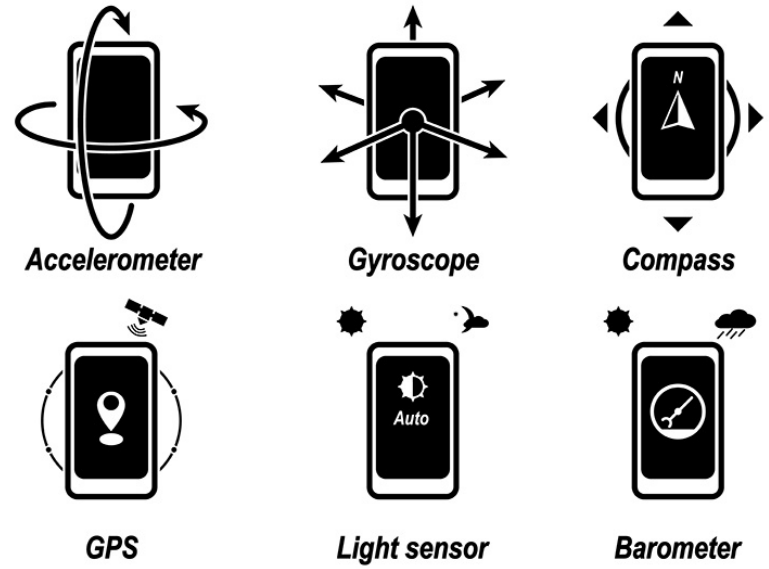
Real world examples

Smartphones

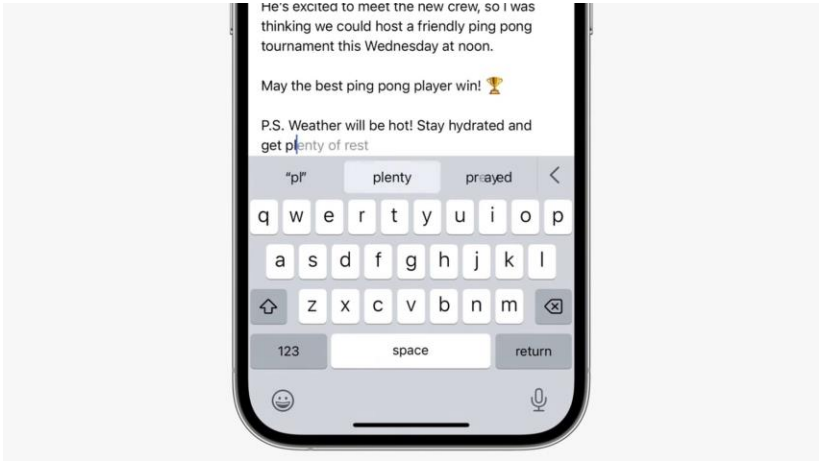


FaceID sensors

Camera image processing



Phone sensors



Autocorrect with language model

Real world examples

Smart watches

- Heart rate sensor, ECG, blood oxygen…
- Applications:
  - Activity recognition
  - Sleep tracking
  - Assistant(Siri, Ok google etc)

- Everything with machine learning models!

Apple Watch and Galaxy Watch

Classical computing

A laptop or desktop CPU

- 2 to 16 cores
- Running at 2 to 5GHz
- Between 15W and 200W
- Typically, x86 instruction set
- Large chips
- Expensive (tens to hundreds of €)

Regular 15 inch HP laptop

intel. AMD

Classical computing

A laptop or desktop CPU

- Thousands of cores
- Running at 1 to 3GHz
- Between 50W and 400W
- Extremely fast for deep learning
- Very large chips
- Very expensive (hundreds to thousands of €)

RTX 4090 Graphics card (Nvidia)

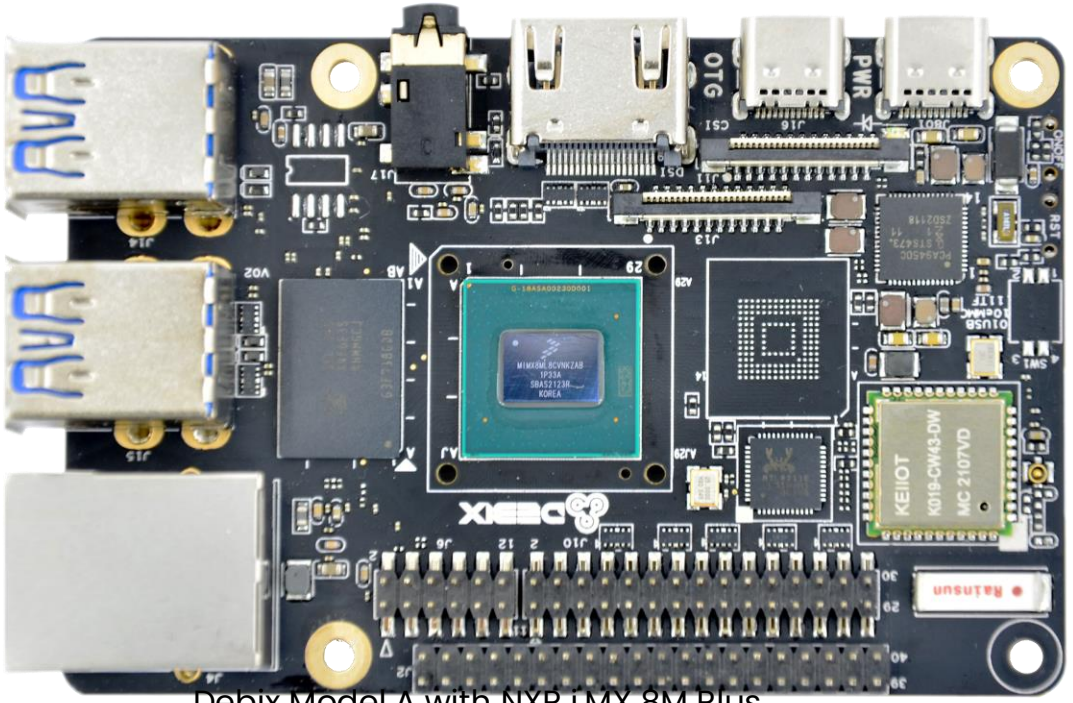
NVIDIA

AMD

Edge computing

ARM Microprocessors

- 1 to 8 cores
- Running at around 1GHz
- Low power (1 to 10W)
- Cheap (few €)
- Reduced instruction set (ARM)
- Typical in phones, tablets, etc.

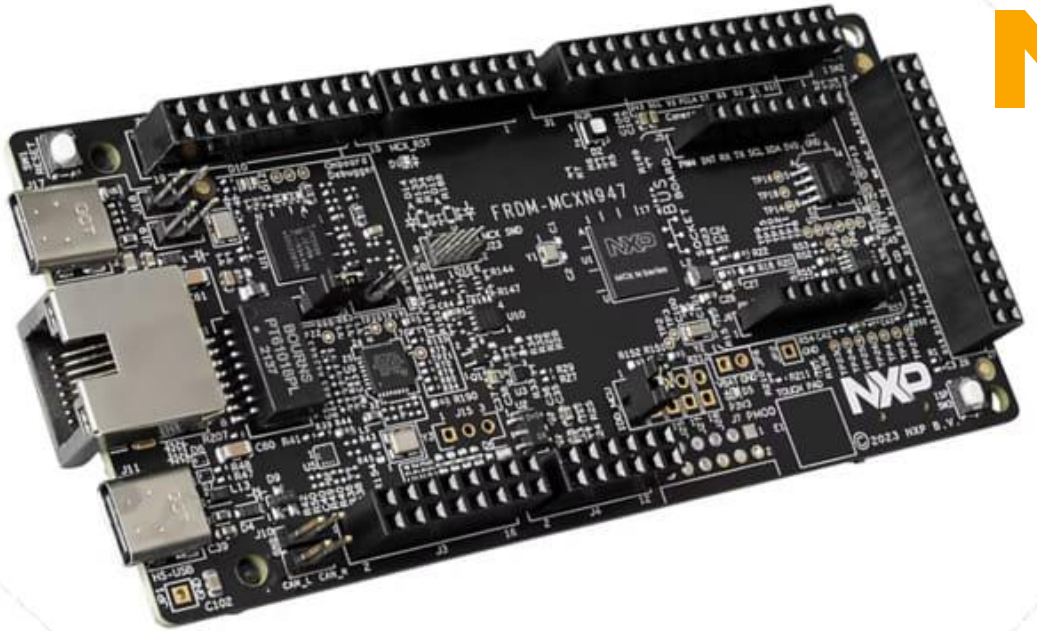


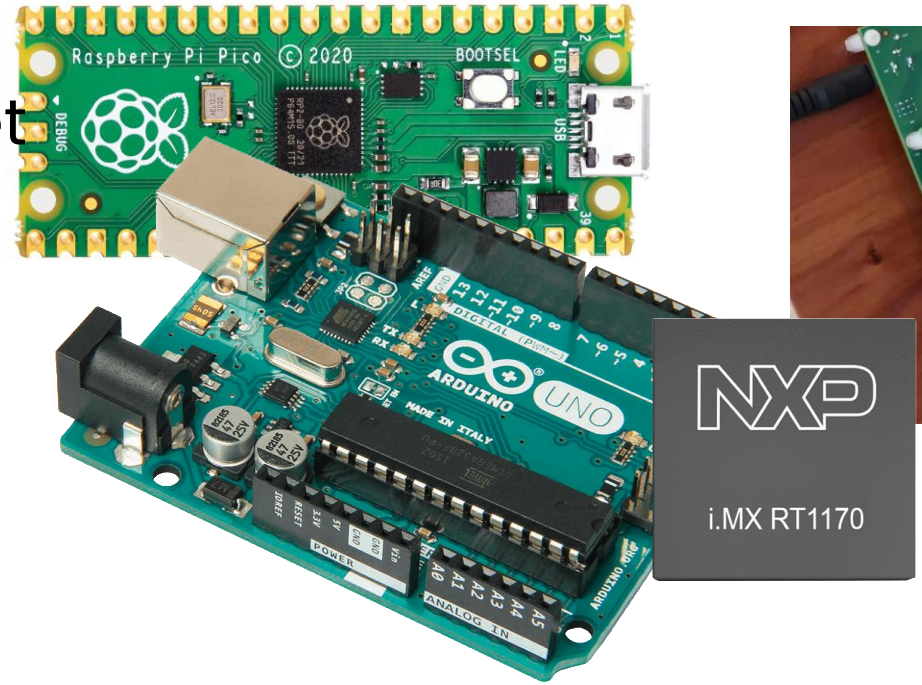
Debix Model A with NXP i.MX 8M Plus processor

Edge computing

Microcontrollers

- 1 to 2 cores
- 10MHz to 1GHz
- Very low power (mW to 1W)
- Very cheap (few cents)
- Very reduced instruction set (sometimes no FPU)
- Typical in appliances, watches, IoT.

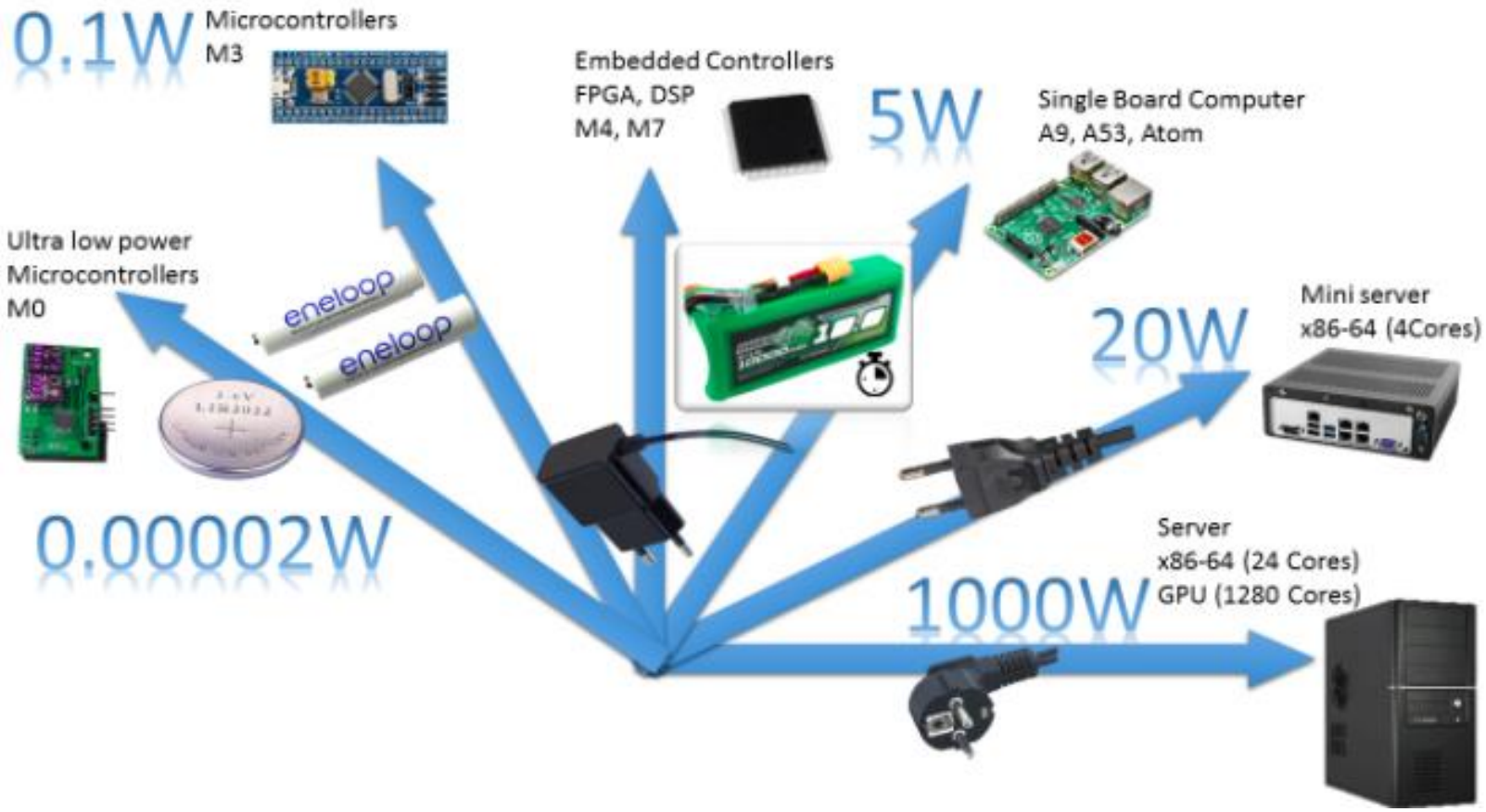




NXP i.MX RT Crossover MCU

Constraints of Edge devices

- Small memory
- Small storage
- Slow(er) compute
- Limited connectivity



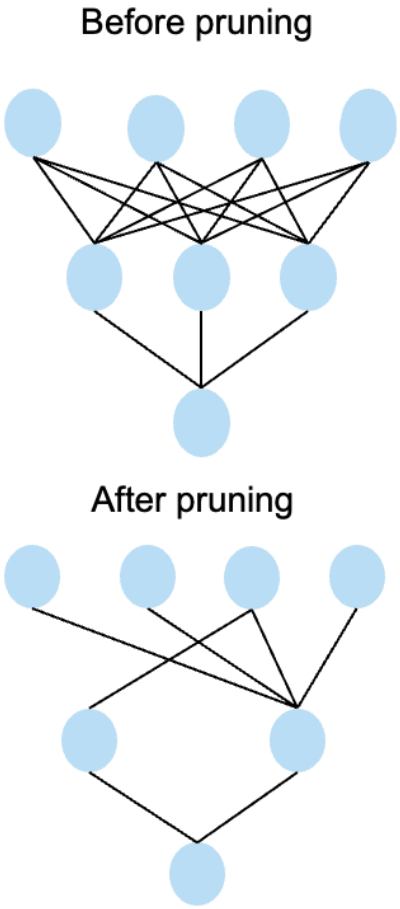
03

Embedded AI Optimization

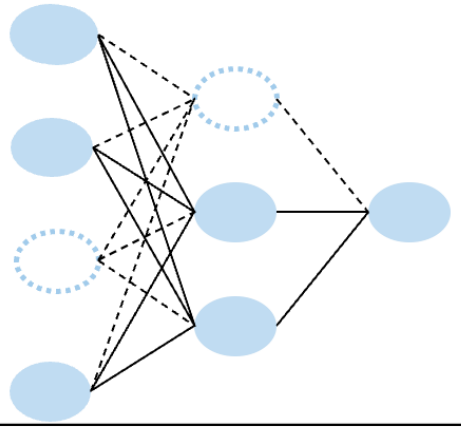
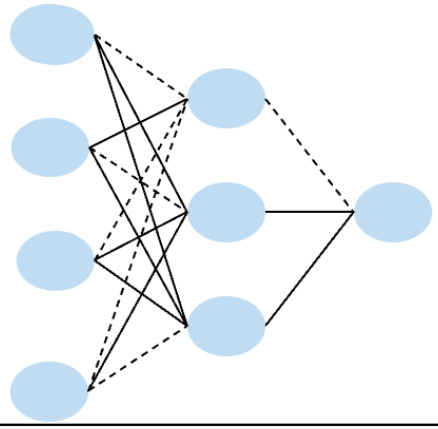
Pruning

Overview

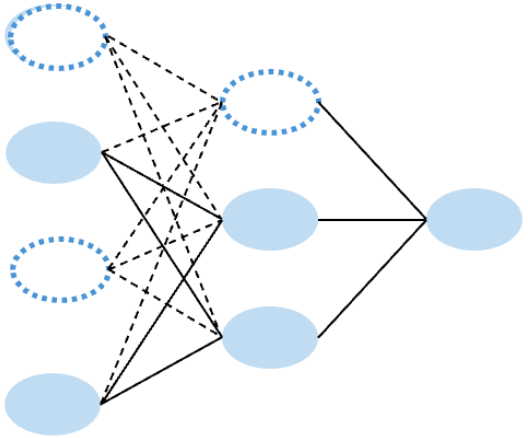
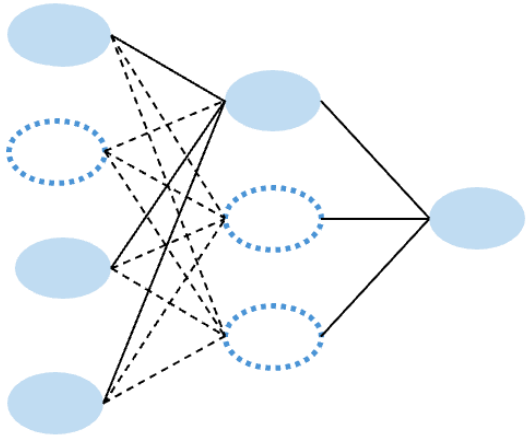
- In deep learning, pruning refers to removing unnecessary parameters that do not contribute significantly to the output.
- Motivating factors for performing pruning:
  - Identifies smaller subnetworks which can fit in memory
  - Reduces energy costs, storage constraints, inference time
  - Facilitates deployment of DL models in constrained embedded systems



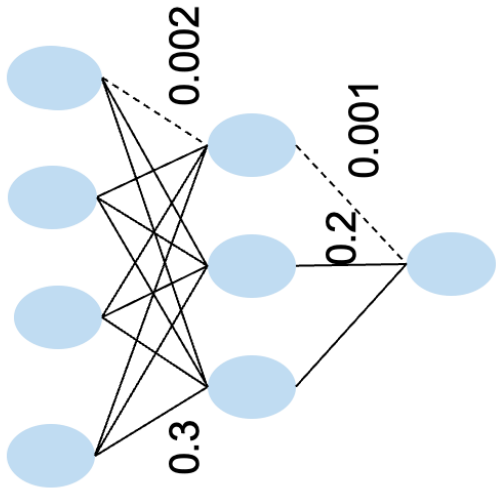
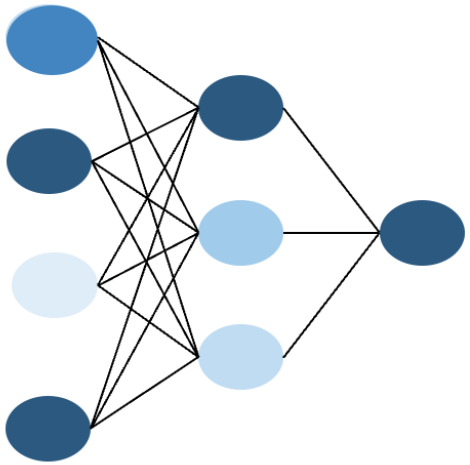
What to prune

Type	Structured	Unstructured
Definition	Removing **entire units** (neurons, filters, channels)	Removing **individual weights** of neurons
Illustration		
Pros	- Hardware acceleration - **Faster** inference	- **Flexibility** of the method - **Preservation** of model **structure**
Cons	- **Accuracy loss** - **Modified** structure	- **Accuracy loss** - More **computational cost (on normal hardware)**

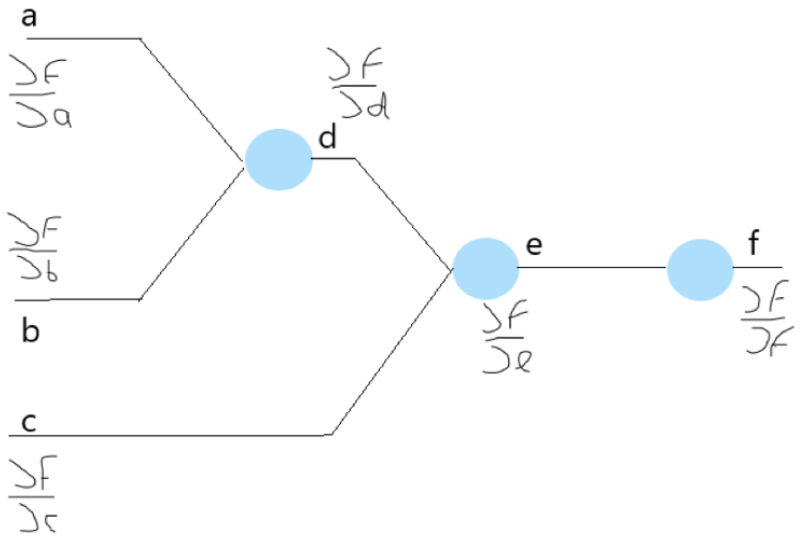
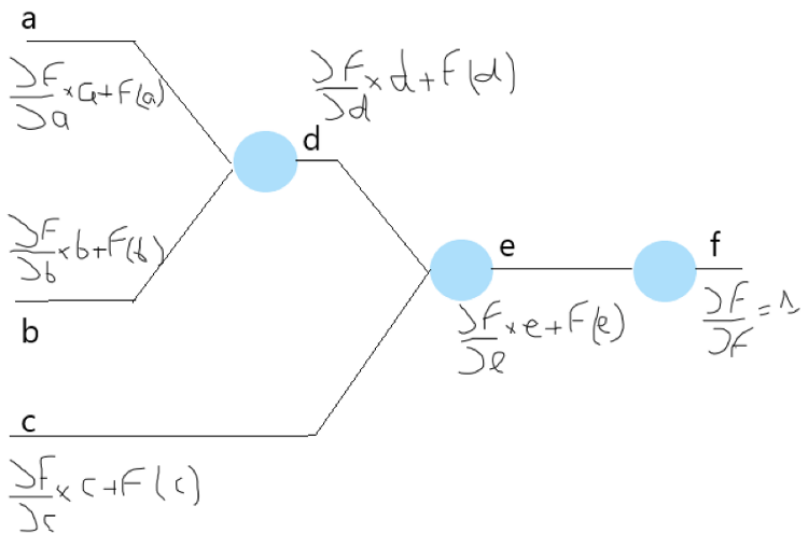
Global vs Local

Type	Global	Local
Definition	Removing a fixed percentage of weights from the **entire model**	Removing a fixed percentage of weights from **each layer** in the model
Illustration		
Pros	- **Easy** to implement - Computationally **efficient**	- Considers **local** weights importance - Can be adapted to the **behavior** of each layer
Cons	- **Important** params may be removed - Loss in **accuracy**	- **Complex** to implement and manage - Loss in **accuracy**

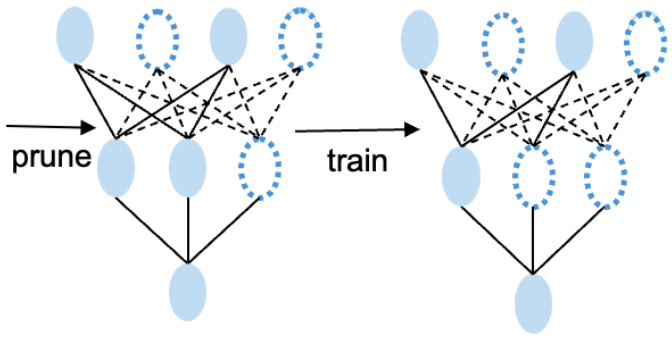
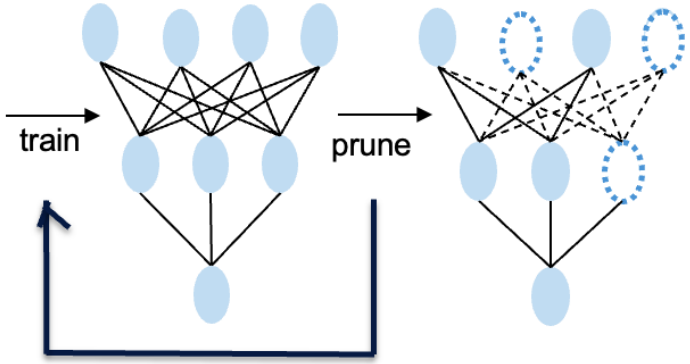
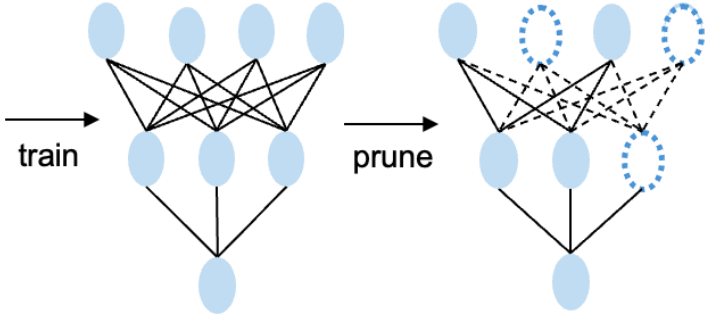
Pruning criteria

Type	Weights magnitude	Neurons activation
Definition	Removing weights with **lower magnitude** since they are the least important	Removing weights associated to neurons with **lower activations**
Illustration		
Pros	- **Easy** to implement	- More representative of real usage
Cons	- A layer can be **pruned more** than another	- **Generalization** problem - Requires training data

Pruning criteria

Type	Gradient based	Taylor expansion
Definition	Pruning weights with **lower gradients**, small updates during training	Pruning weights with **lower Taylor values**
Illustration		
Pros	- Considers **direction** and not only magnitude - Computationally **efficient (use autograd)**	- **Higher order** information - Higher **precision** and **accuracy**
Cons	- **Generalization** problem (requires data) - Dependent to **hyperparameters**	- Implementation **complexity** - Lack of **scalability**

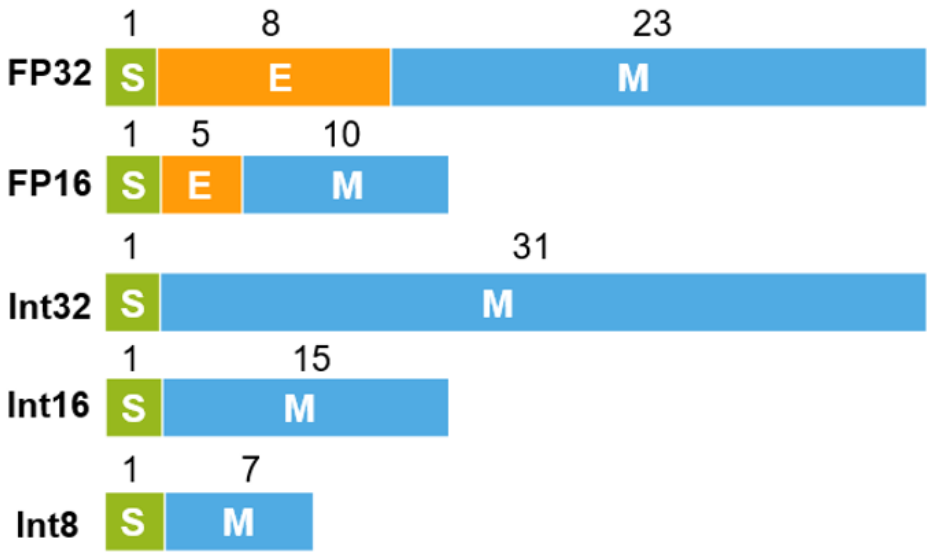
When to prune?

Type	Before training	While training	After training
Definition	Pruning **initialized before** training the model	Pruning at **specific/all** training **epochs**	Pruning is done **after** the training while **inference**
Illustration			
Pros	- **Faster** training - **Smaller** models	- **Adaptive**, faster convergence - All computations are done **before** inference	- Better **preservation** of accuracy - Simple and **efficient**
Cons	- Risk of **losing** important info - Lack of **prior** on what to prune	- Implementation **complexity** - **Instability** of training	- No benefits during training - May need **fine tuning**

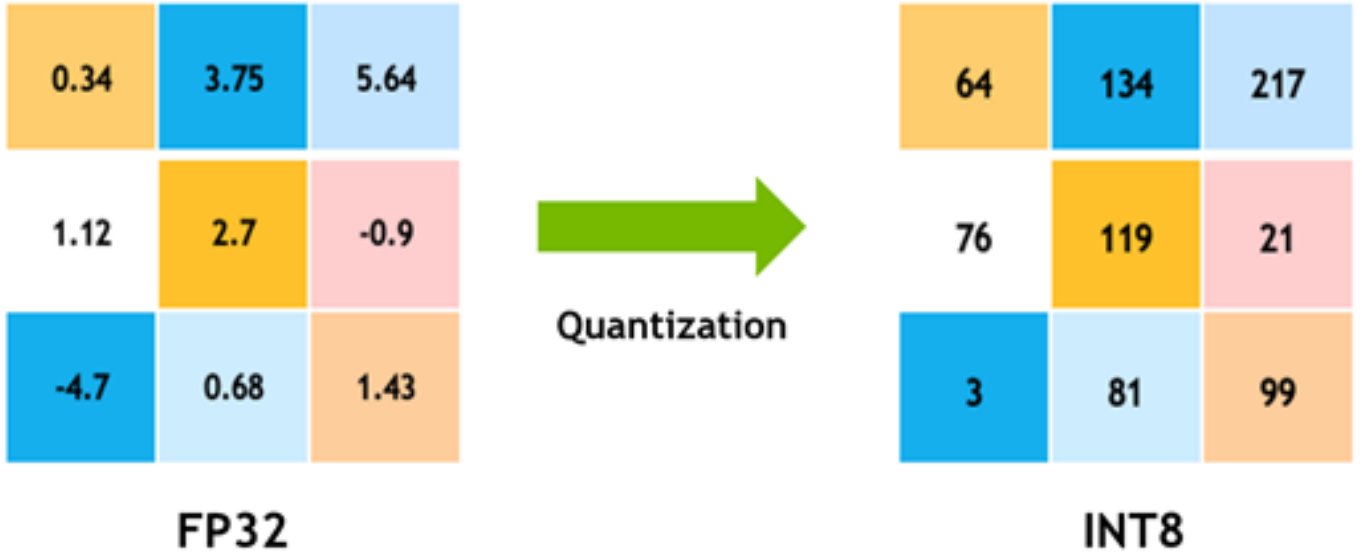
Quantization

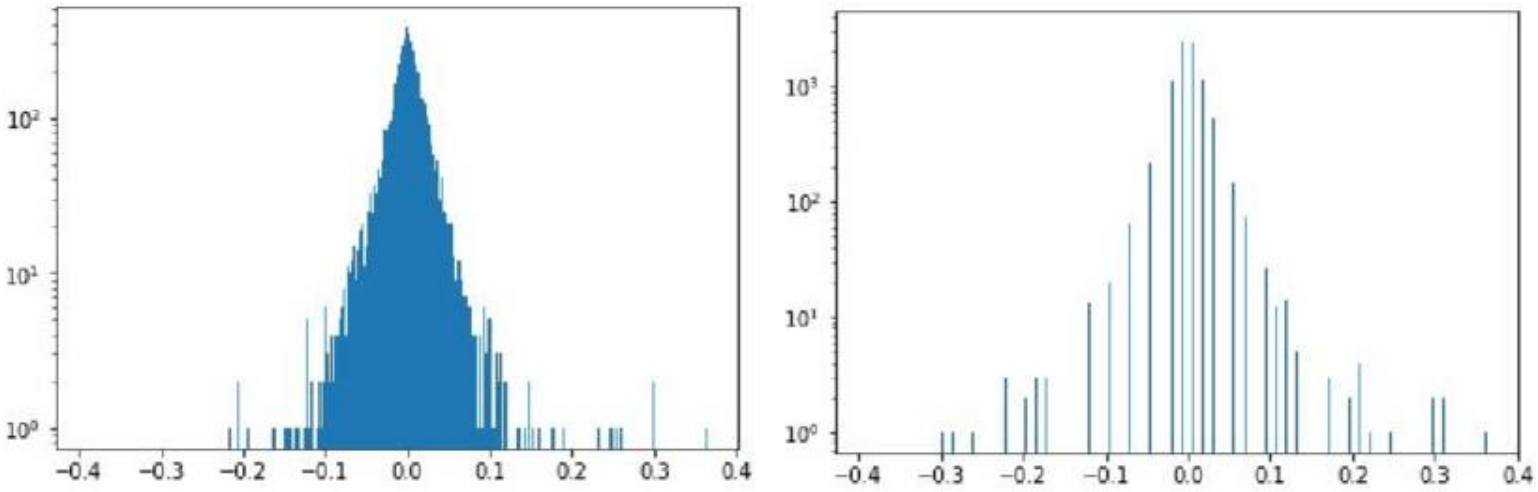
Bit depth

- In deep learning models, parameters are represented as **floating point** (32 or 16 bit) numbers which requires a significant amount of memory and computational resources.
- Quantization **shrinks** a neural network by decreasing the **precision** of its weights, biases and activations.
- **Necessary** for some constrained devices that do not support high precision representations like the NPUs
- Pros
  - Lower latency, reduced energy
  - Computational efficiency
- Cons
  - Accuracy loss
  - Precision loss in computation

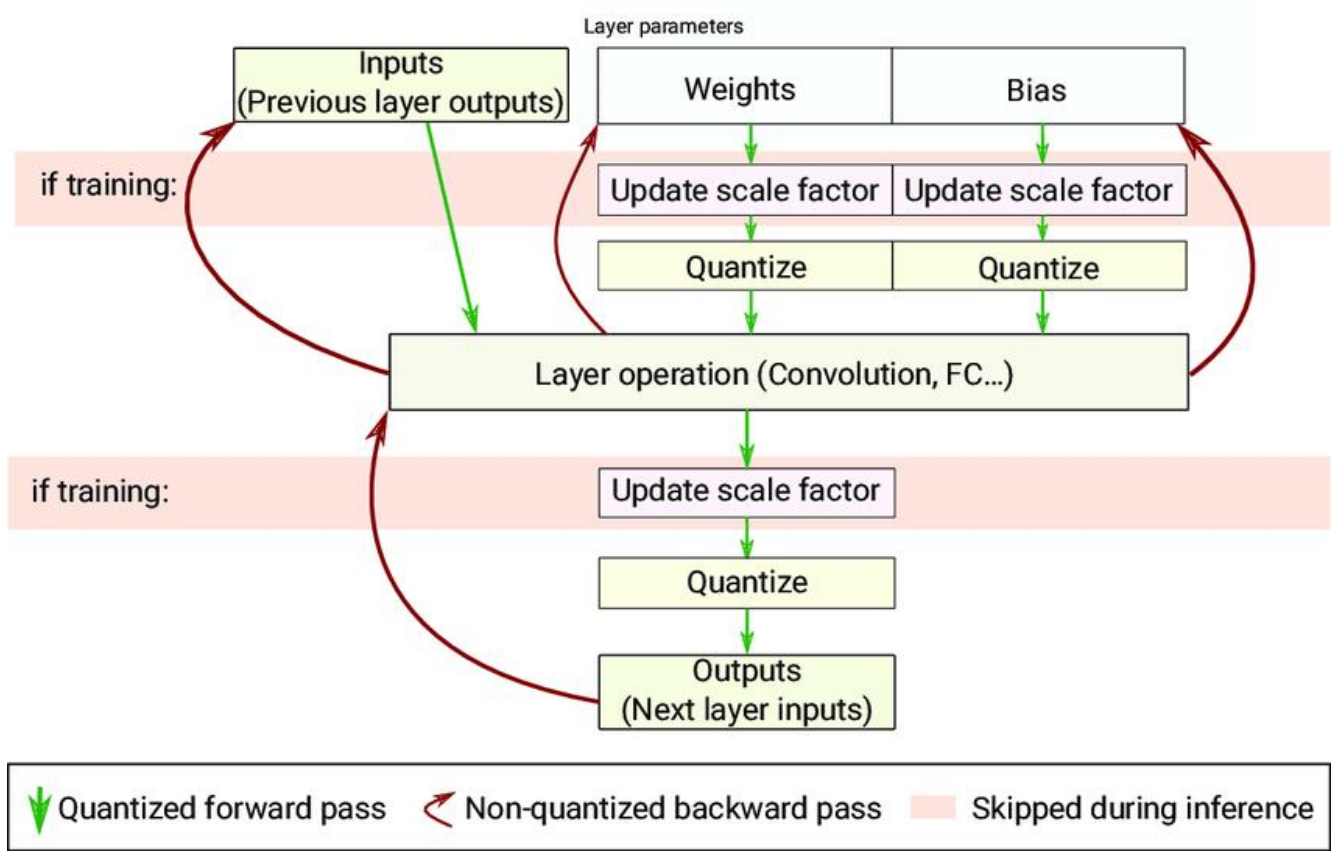


Float vs Int


0.34
3.75
5.64
1.12
2.7
-0.9
-4.7
0.68
1.43
Quantization
64
134
217
76
119
21
3
81
99
FP32
INT8

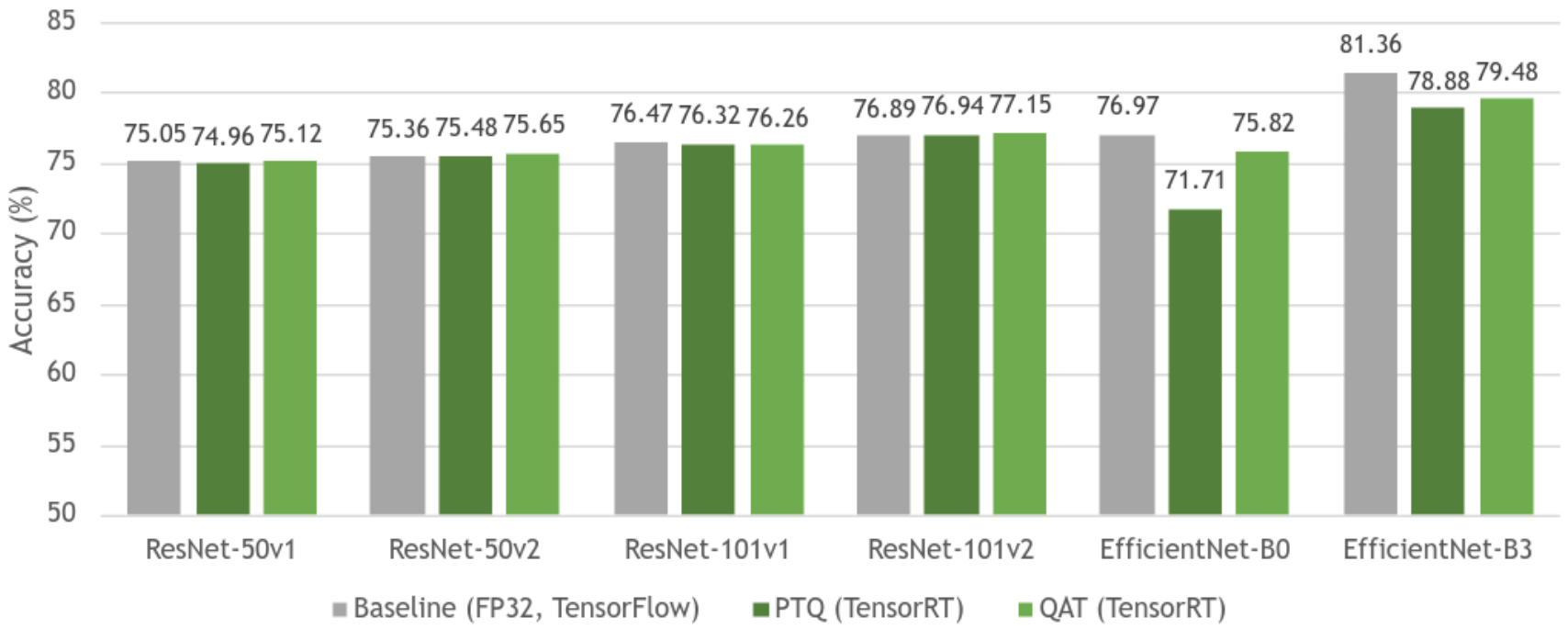

10^2
10^1
10^0
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
10^3
10^2
10^1
10^0
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4

Quantization-Aware Training (QAT)



PTQ vs QAT (Source Nvidia)

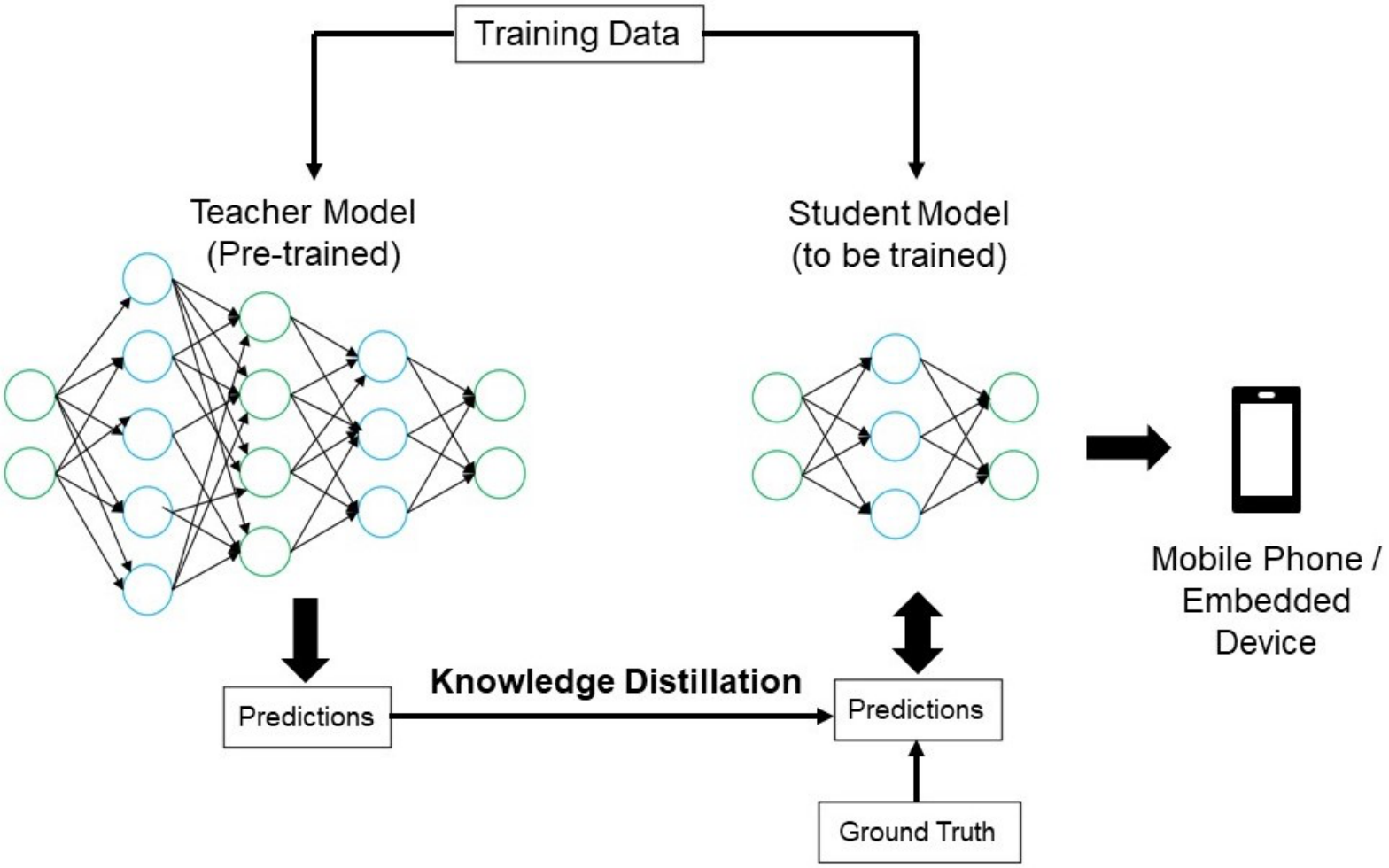
PTQ vs QAT



PTQ vs QAT (Source Nvidia)

Distillation

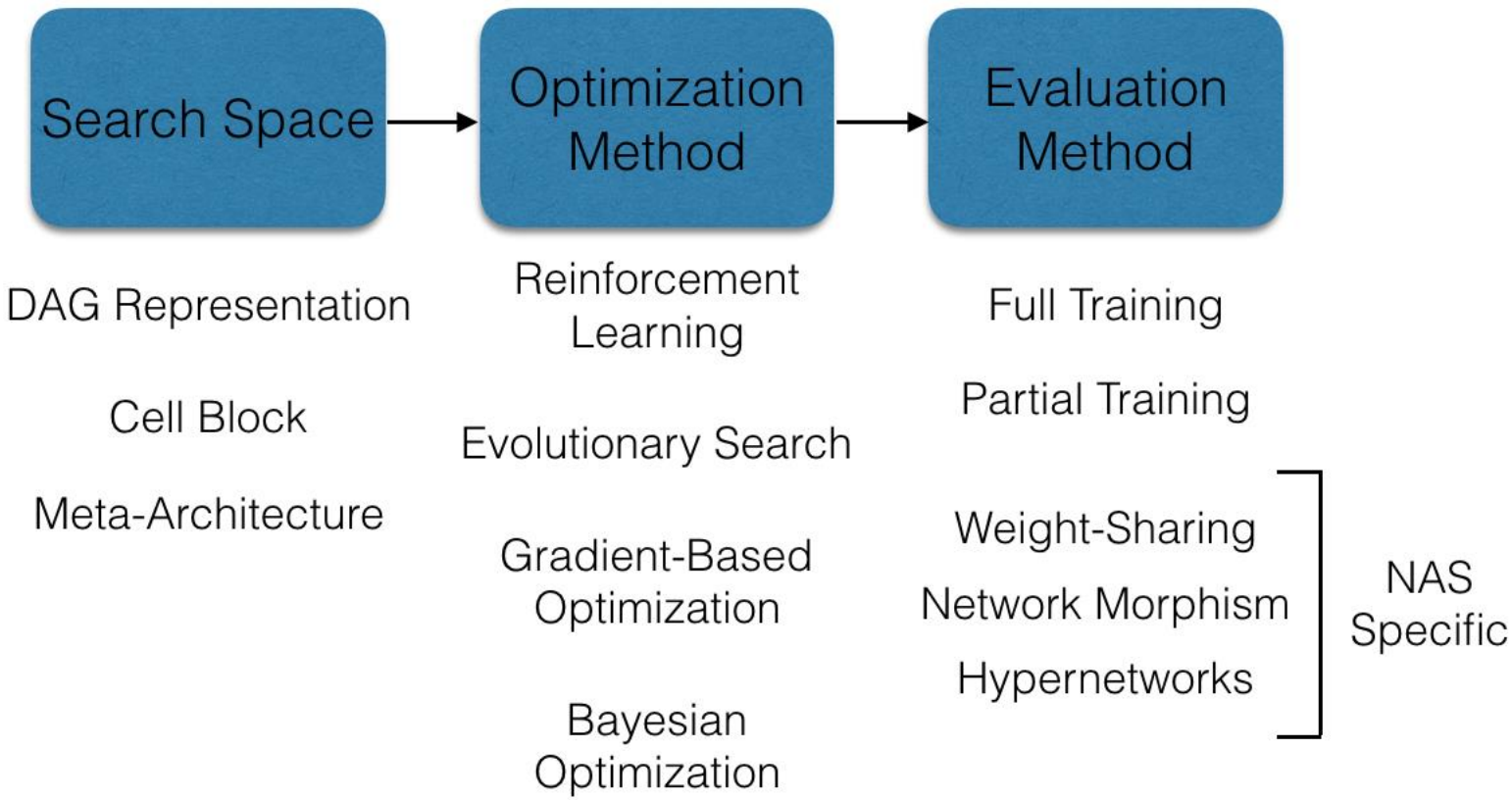
Student Teacher



Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Neural architecture search

Components of NAS



03

AI at NXP

Intelligent edge systems enabled by NXP

Expansive processor portfolio

MCX MCUs

i.MX RT Crossover MCUs

i.MX Apps Processors and beyond

AI co-processor
NPU

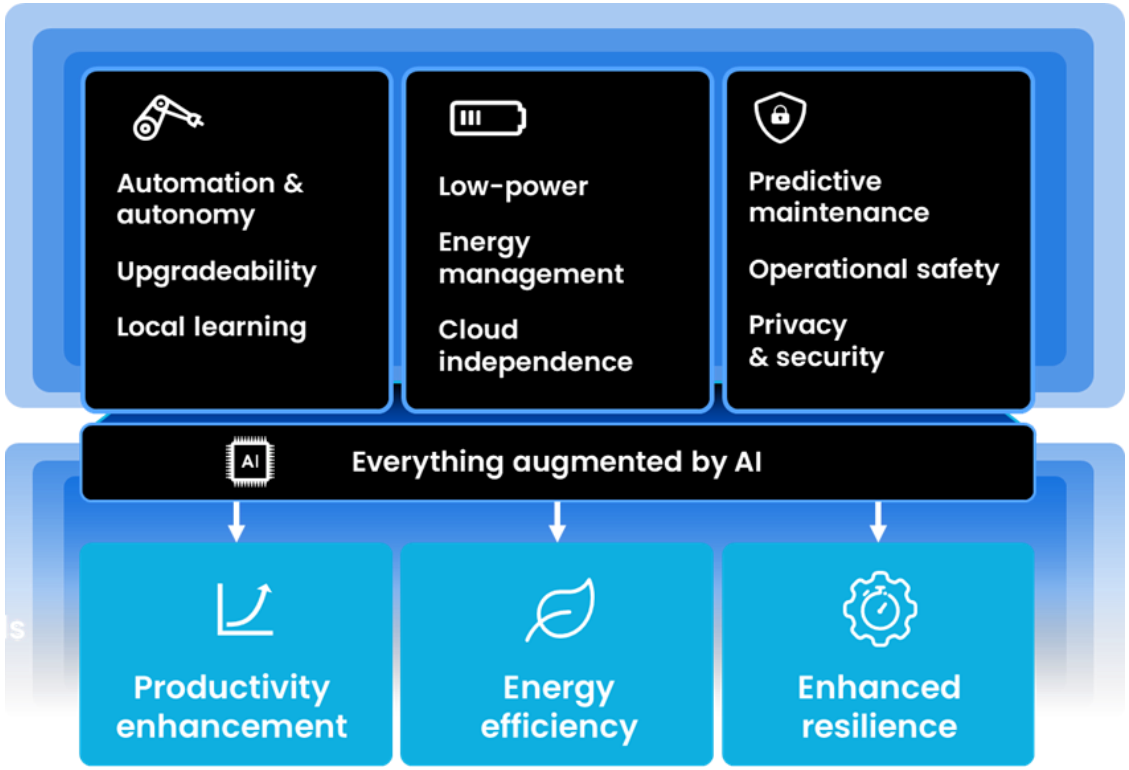
Differentiated HW and SW enablement

eIQ® Neutron NPU
Highly scalable and optimized integrated dedicated AI acceleration

eIQ® Toolkit
AI/ML software toolkit for model Creation, optimization and porting

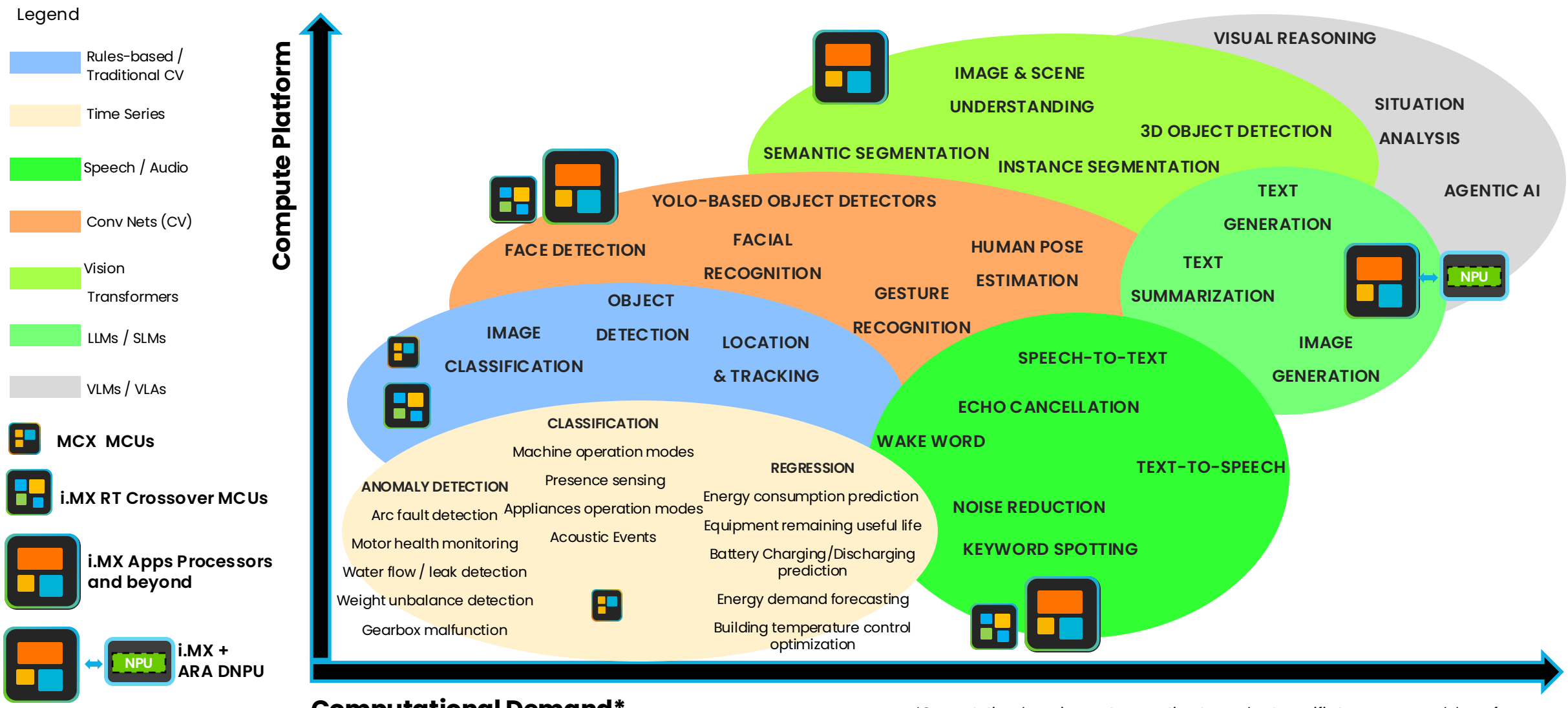
eIQ® Time Series Studio
Automated ML model creation from sensor signals

eIQ® GenAI Flow
Context aware generative AI application development



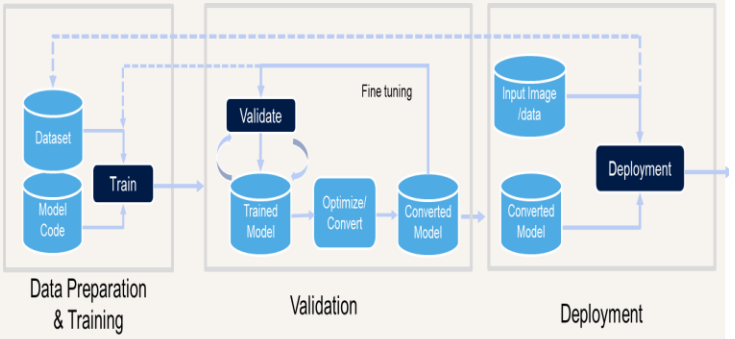
Engaging with our customers to develop system solutions and solve challenges together

NXP MCUs + MPUs cover the full spectrum of AI/ML applications



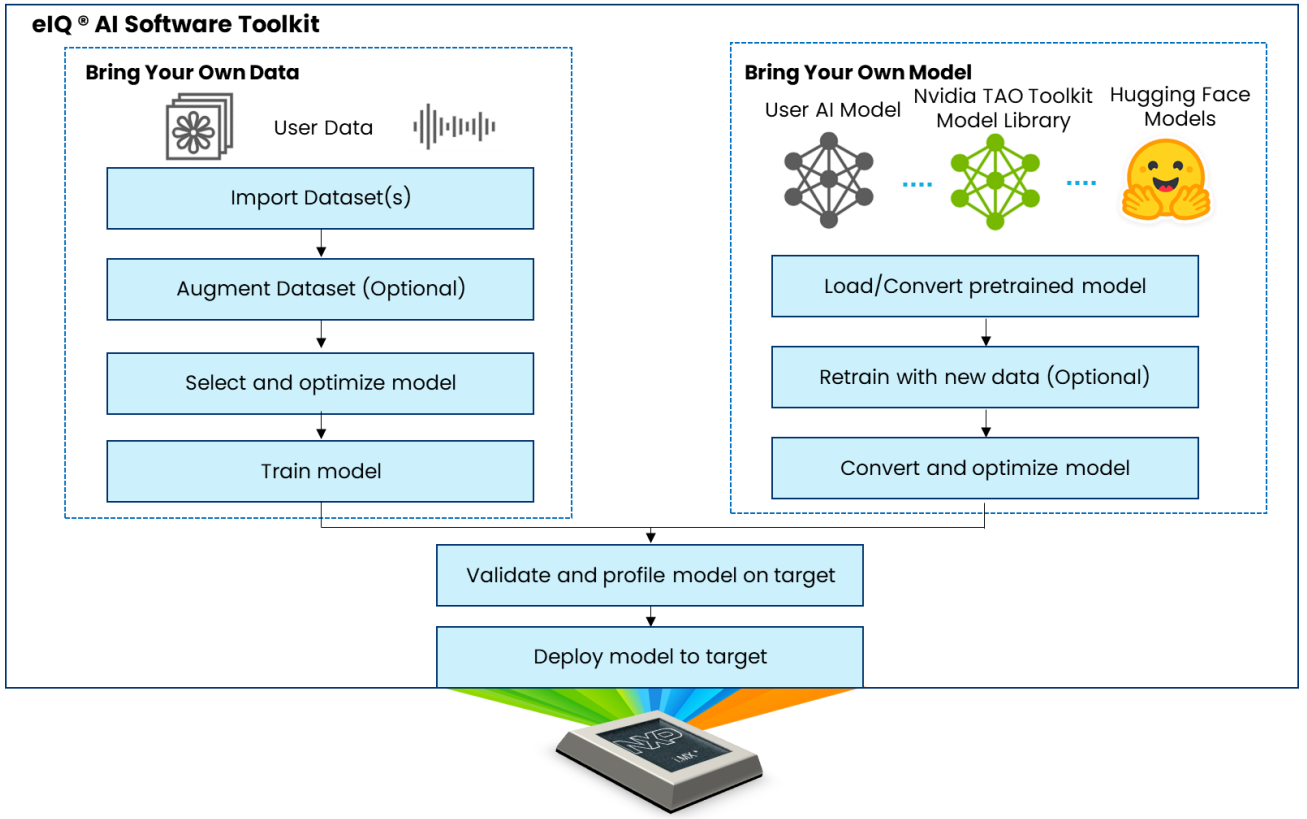
*Computational requirements are estimates and not specific to any one model or reference.

**AI model inference only is considered Demand depends on # of operations, parameters, memory footprint and is highly dependent on optimization, software framework and tools used.



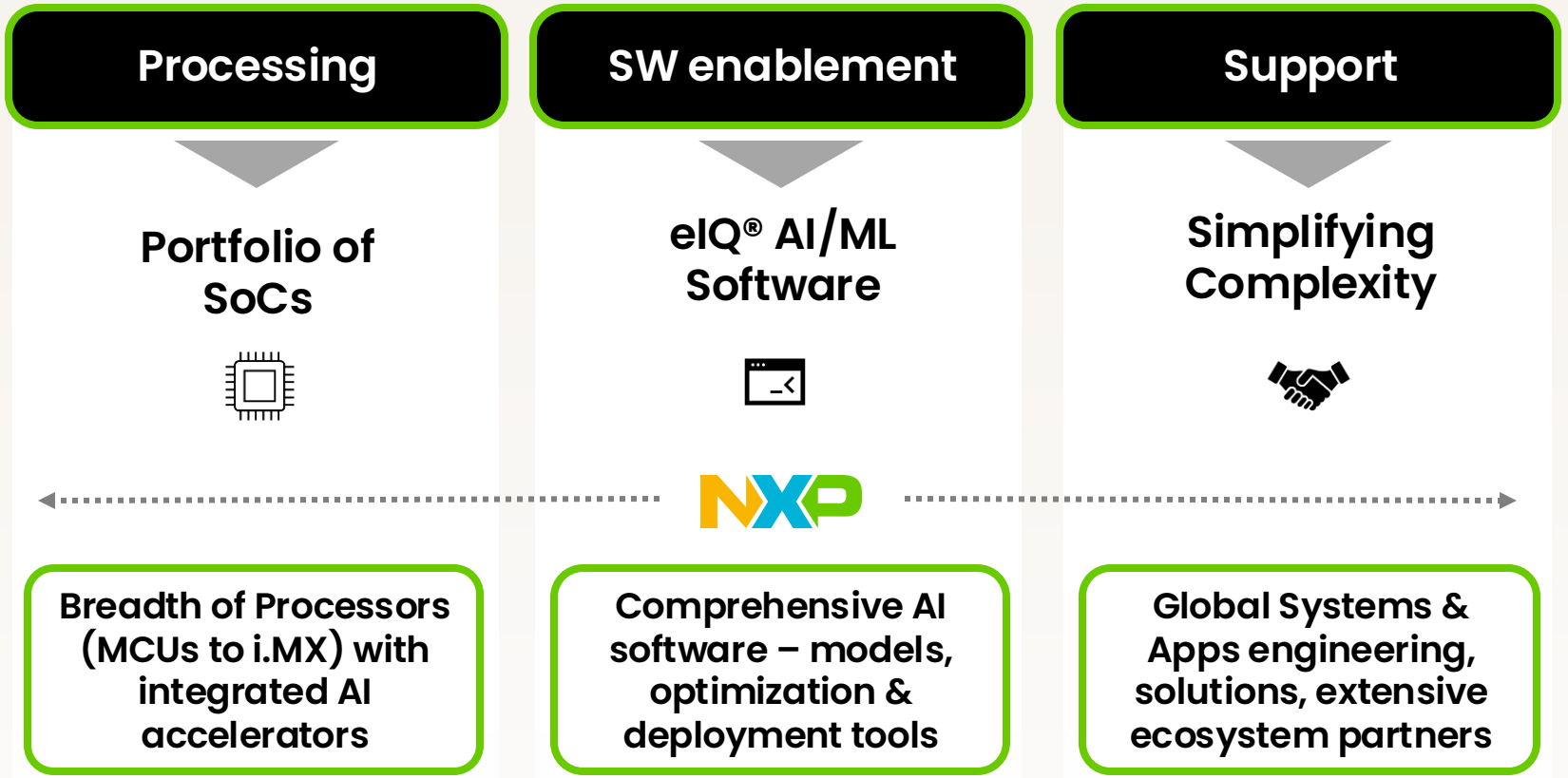
eIQ AI Software Toolkit

Support for MCX MCUs, i.MX RT Crossover MCU and i.MX Applications Processing device families



- NXP's **eIQ AI Software Toolkit** provides a collection of development tools, utilities and libraries for building machine learning applications using NXP MCUs and applications processors (MPUs).
- eIQ AI software can be leveraged as part of a user's existing flow or can be used for the complete flow depending on the machine learning application targeted.
- **Support** for multiple compute engines (CPU, **NPU**, GPU, DSP) if available on the NXP Processor

Addressing complexity with scalable solutions and enablement



Time Series Studio Target Applications

CLASSIFICATION

Machine operation modes

Presence sensing

Appliances operation modes

Acoustic Events

Exercise activity types

Gesture Types

ANOMALY DETECTION

Arc fault detection

Motor health monitoring

Water flow / leak detection

Weight unbalance detection

Bearing wear / gearbox malfunction

And many more...

REGRESSION

Energy consumption prediction

Equipment remaining useful life

Battery Charging/Discharging prediction

Energy demand forecasting

Building temperature control optimization

Our GenAI solutions:

Large, Small and Vision-Language Models (LLMs, SLMs and VLMs) are creating new use-cases for edge devices

Mobile Robotics

Speech-to-speech Interfaces

Multi-Sense Human-Machine Interface

Superior Reasoning and Visual Analysis

Enhanced Predictive Maintenance

… with the introduction of new models, discrete NPUs, and multimodal AI, **the possibilities are expanding every day!**

nxp.com