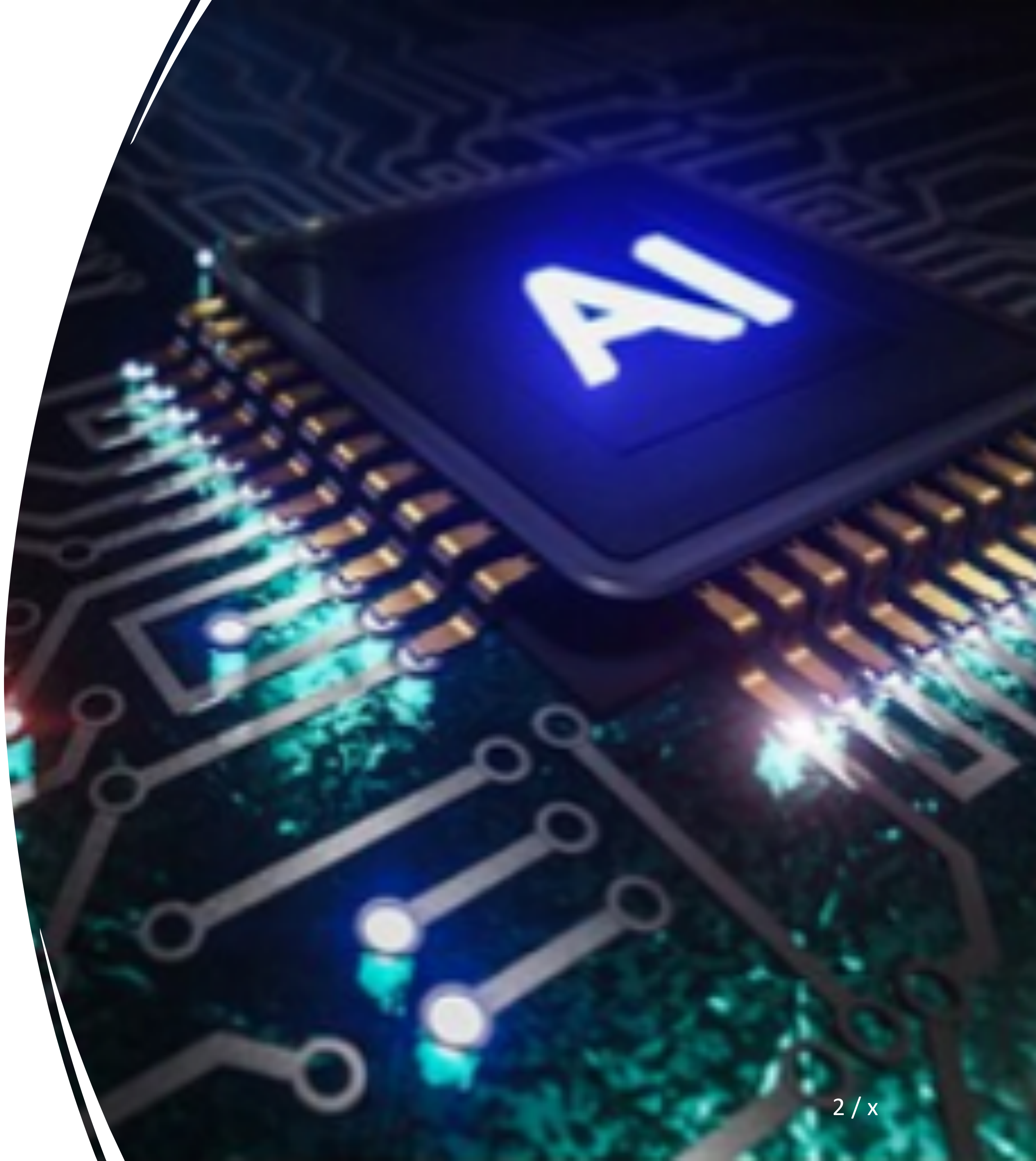
EMBEDDED AI: INTELLIGENCE ON DEVICES

Serge Pacome Bosson, Ing. Radek Holota, Ph.D, prof. Ing. Václav Šmídl, Ph.D.

FACULTY OF ELECTRICAL
ENGINEERING
UNIVERSITY OF WEST BOHEMIA

MOTIVATION

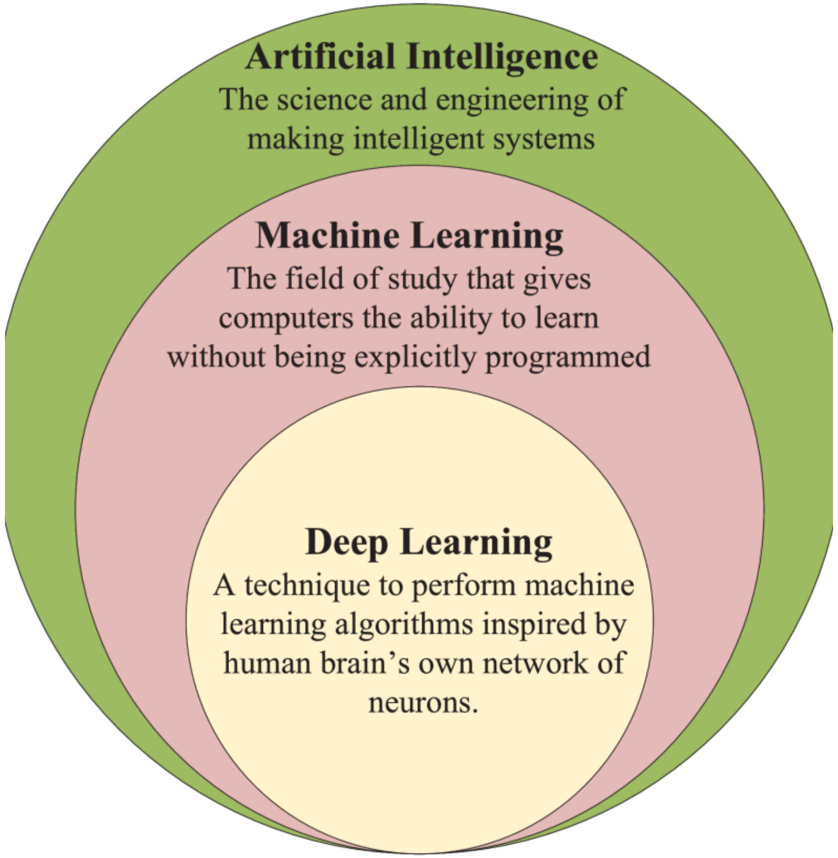
- Deploy Machine Learning Algorithms onto an Embedded System.



WHAT IS AI AND DEEP LEARNING?

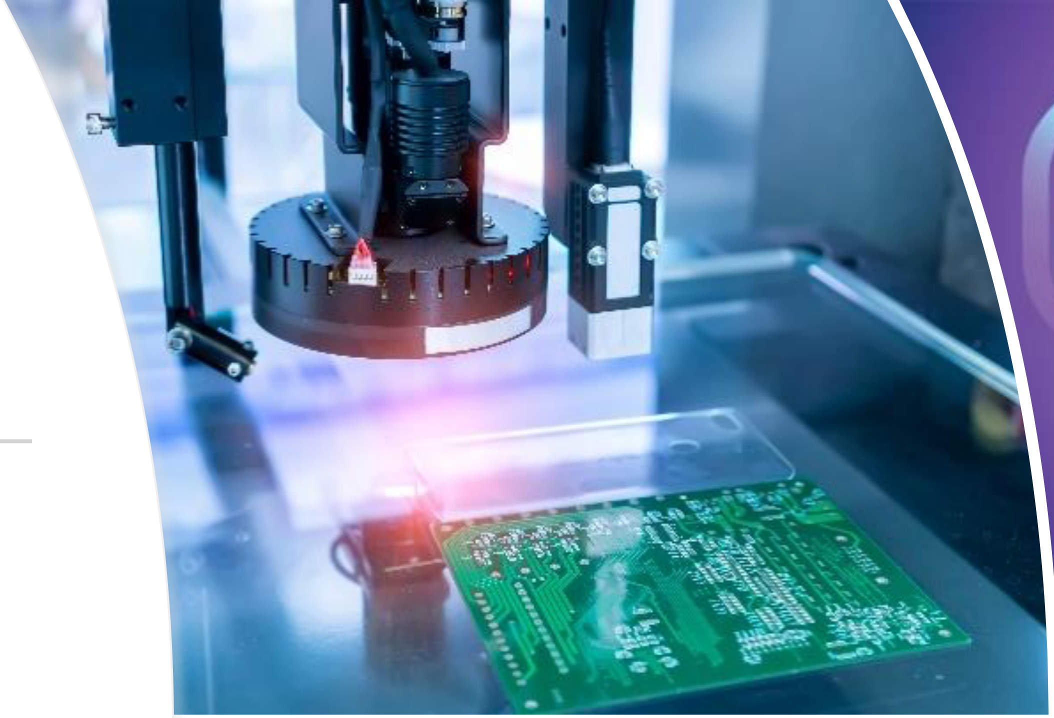
- Artificial intelligence (AI): Using human-like intelligence to solve tasks.
- Machine learning (ML): Algorithm uses data to find patterns.
- Deep learning (DL): Very large algorithms using raw data input.

- Machine learning provides significant advantages over classical computing:
  - Scalability
  - Less R&D effort
  - More accurate



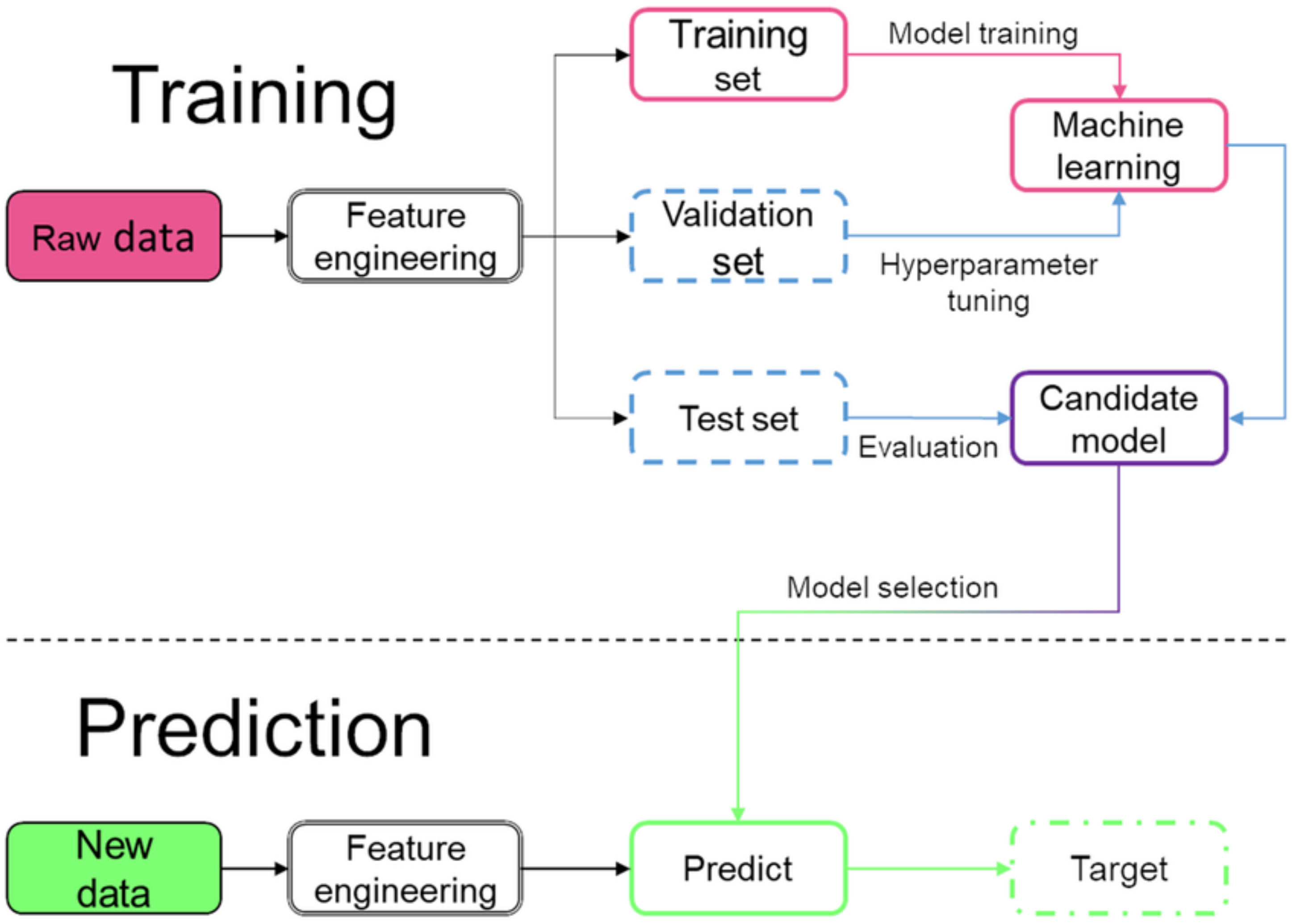
WHERE DO WE USE AI ?

- Drive assistant
- Machine Vision.
- Fault diagnosis.
- Robotics.
- Security and
  home Automation cameras.
- Speech recognition,
  text analysis, translation.
- And Many More!



PRINCIPLE

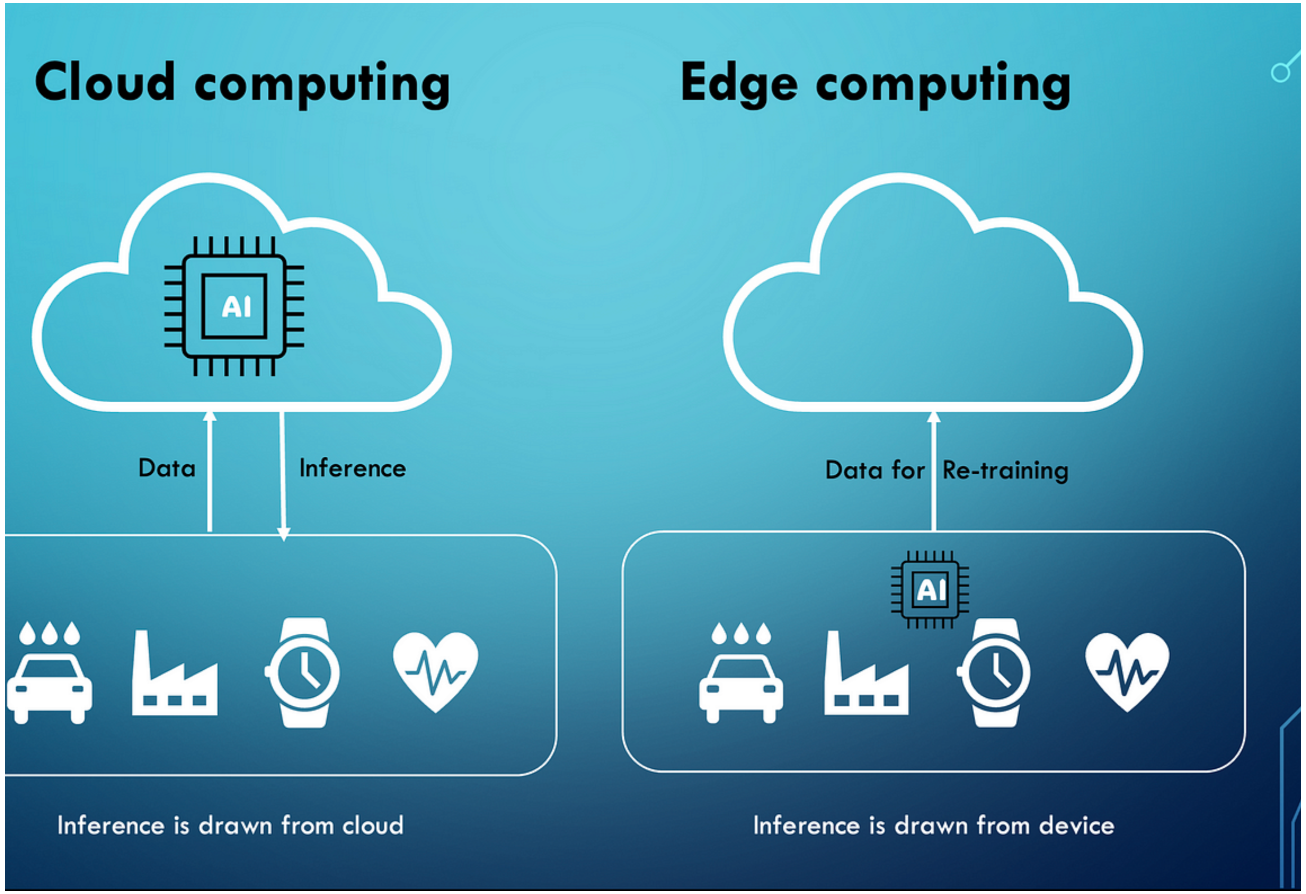
- Collect data.
- Design a ML model.
- Train the model.
- Move trained model to platform for inference.



AI INFERENCE

- Edge AI: run ML models where data is generated.
  - Algorithms run on embedded systems.
- Cloud AI: run ML models on cloud servers.
  - Algorithms run on data centers.

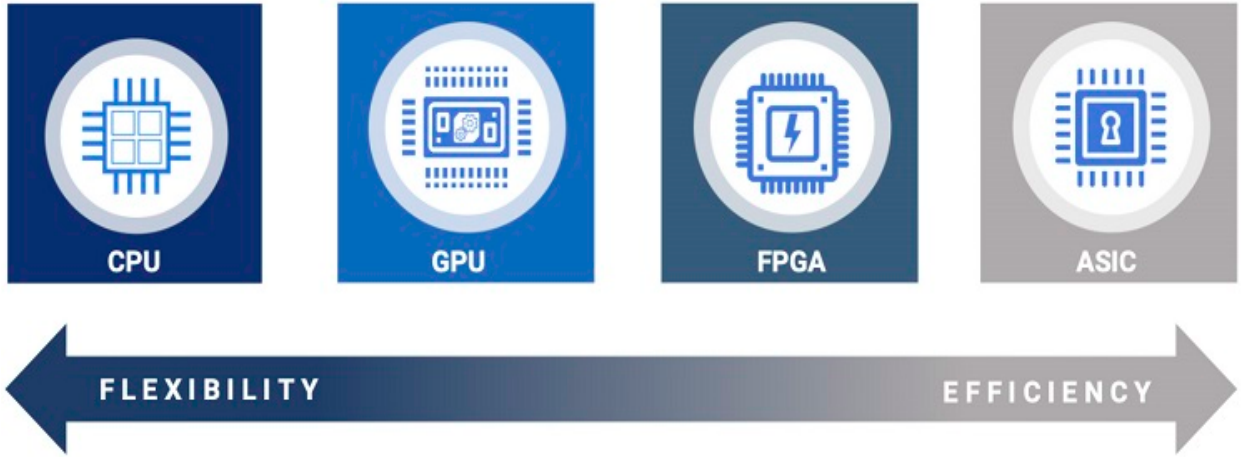
- Benefits of Edge AI:
  - Reduced latency.
  - Improved privacy and security.
  - Enhanced energy efficiency.
  - Real-time decision-making.



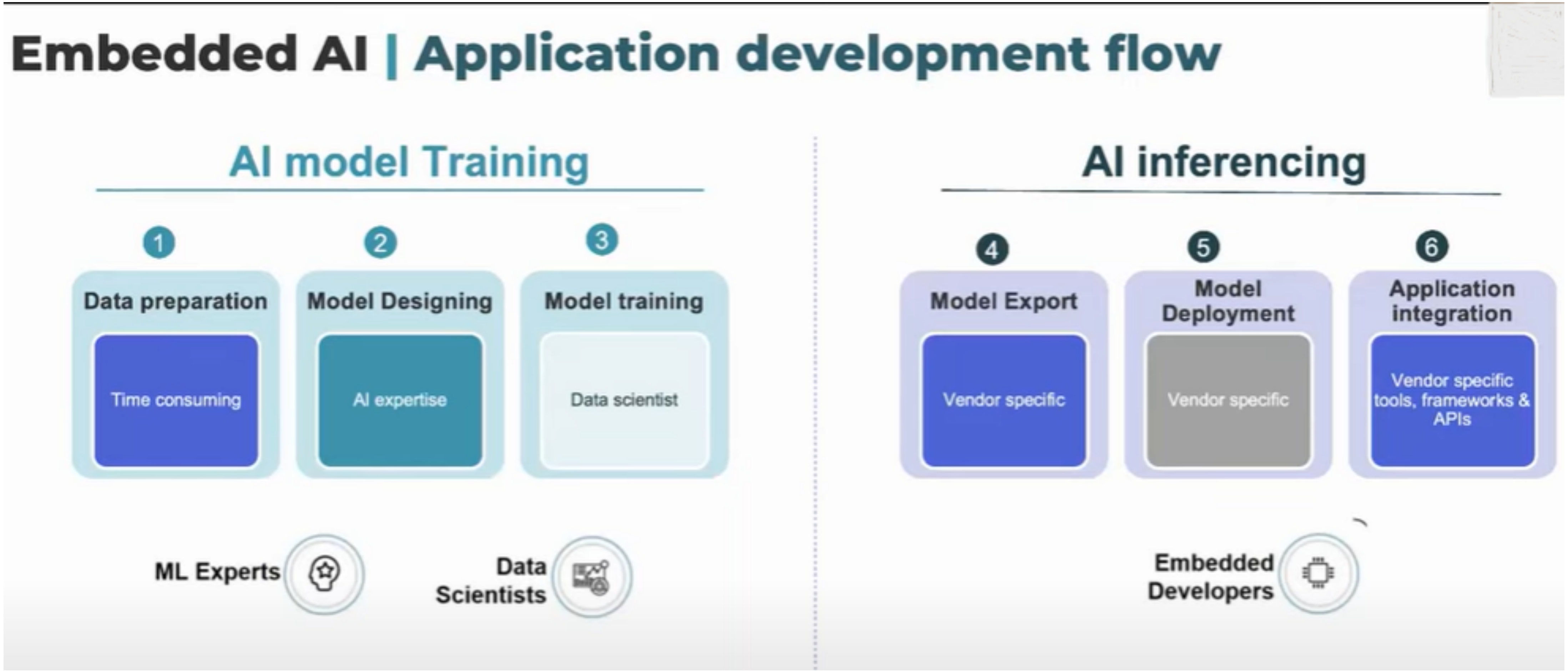
HARDWARE FOR EMBEDDED AI

The brain of an embedded AI device is usually based on:

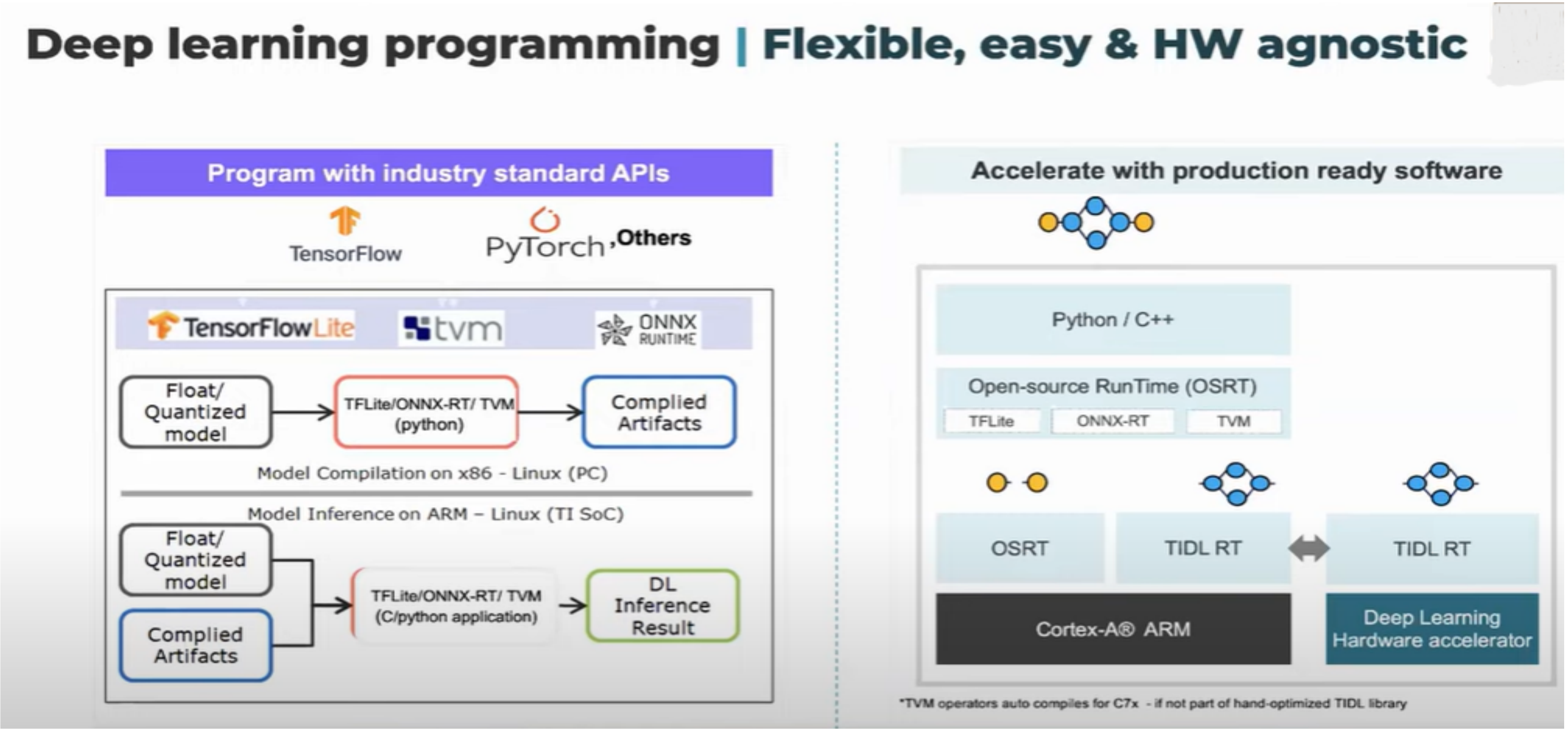
- Microprocessor (CPU).
- Graphical processing unit (GPU).
- Field programmable gate array (FPGA).
- Application specific integrated circuit (ASIC).



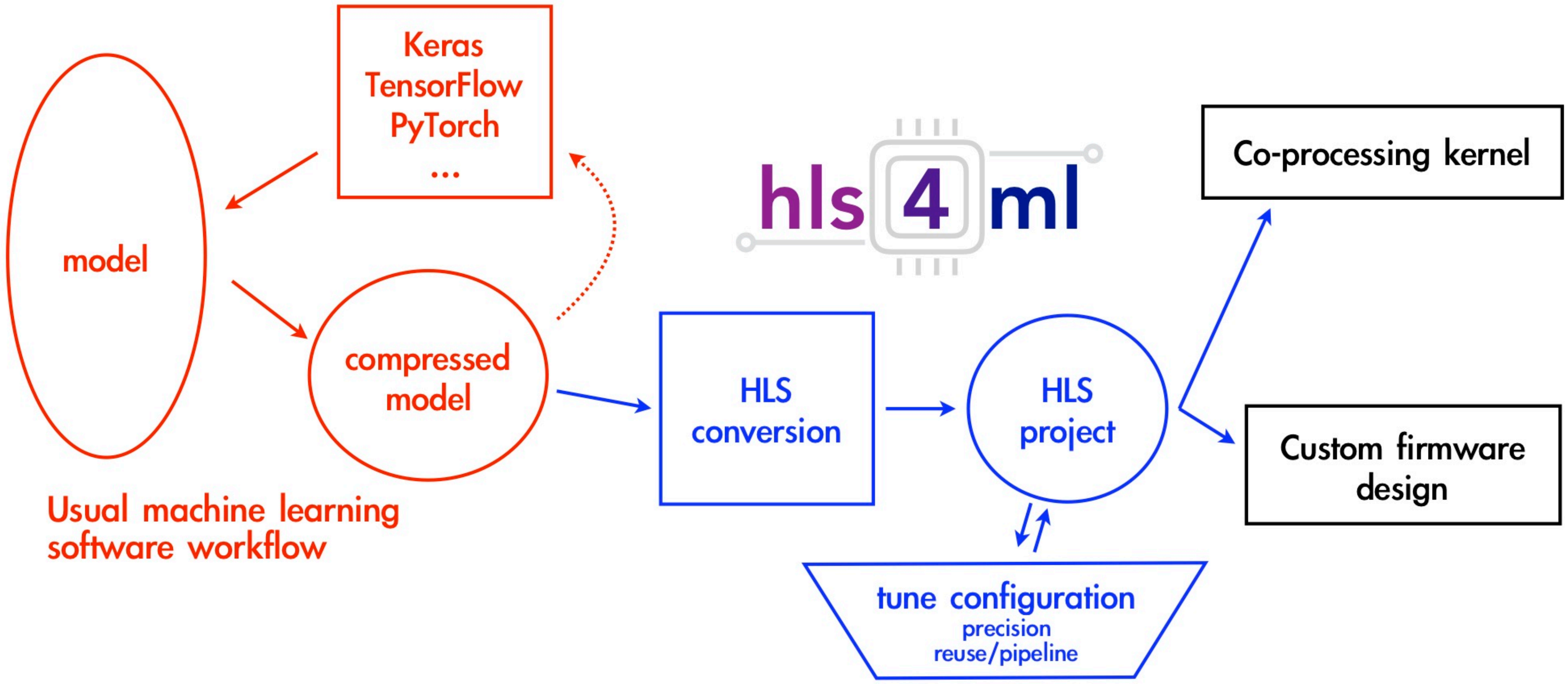
AI INFERENCE



ML COMPILER FRAMEWORK FOR CPU AND GPU: OSRT TECHNOLOGY



ML COMPILER FRAMEWORK FOR FPGA AND ASIC : HLS TECHNOLOGY



CAS STUDY: FUNCTION APPROXIMATION IN EMBEDDED SYSTEMS

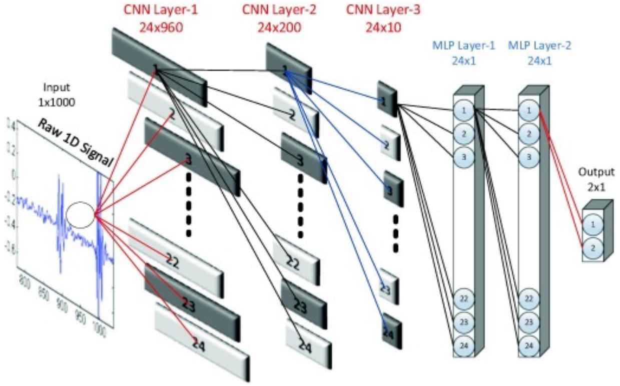
Function approximation based on machine learning algorithms can find practical application in electrical engineering:

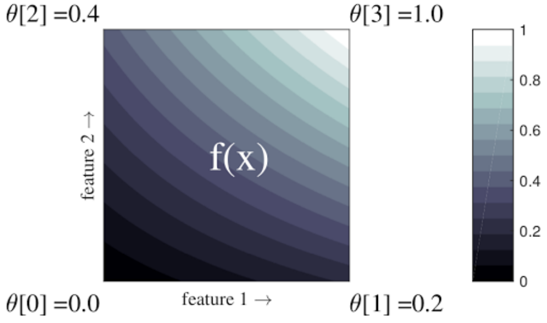
- Flux linkages approximation.
- Online fault diagnosis.
- AI robotics.
- Advanced control in power electronics and drives.
- Optimization problems.
- Etc…

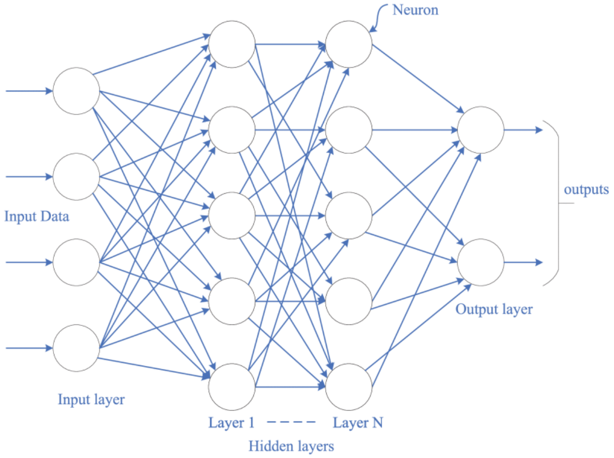
CAS STUDY: FUNCTION APPROXIMATION IN EMBEDDED SYSTEMS

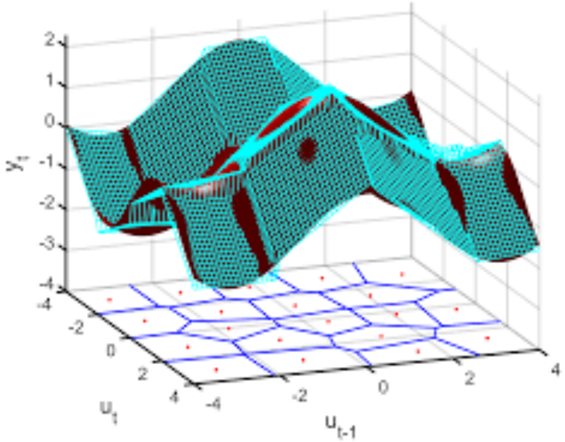
Approximators that admit efficient implementation in conventional industrial computer systems:

- Multilayer perceptron (MLP).
- 1D Convolutional neural network (1D CNN).
- Piece-wise affine (PWA).
- Lattice interpolated look-up table (LUT).





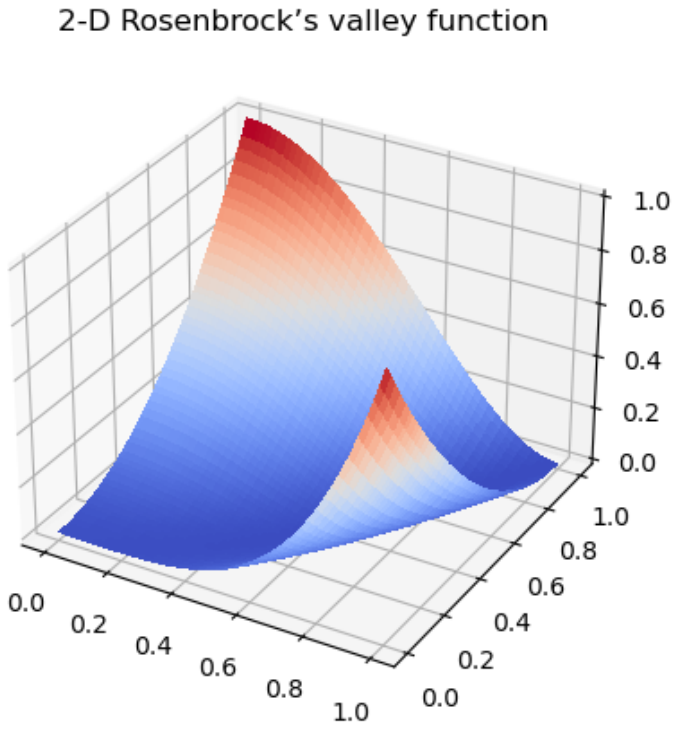




DATA PREPARATION

Evaluate the regression power of the reviewed approximators on the classical optimization problem called Rosenbrock's valey:

- Generated 10K points from the uniform distribution [0 1]^D.
  - D: Number of dimension: 2D 5D 8D 12D 15D.
- Compute D-dimensional Rosenbrock's function and scale them in the range of [0 1].

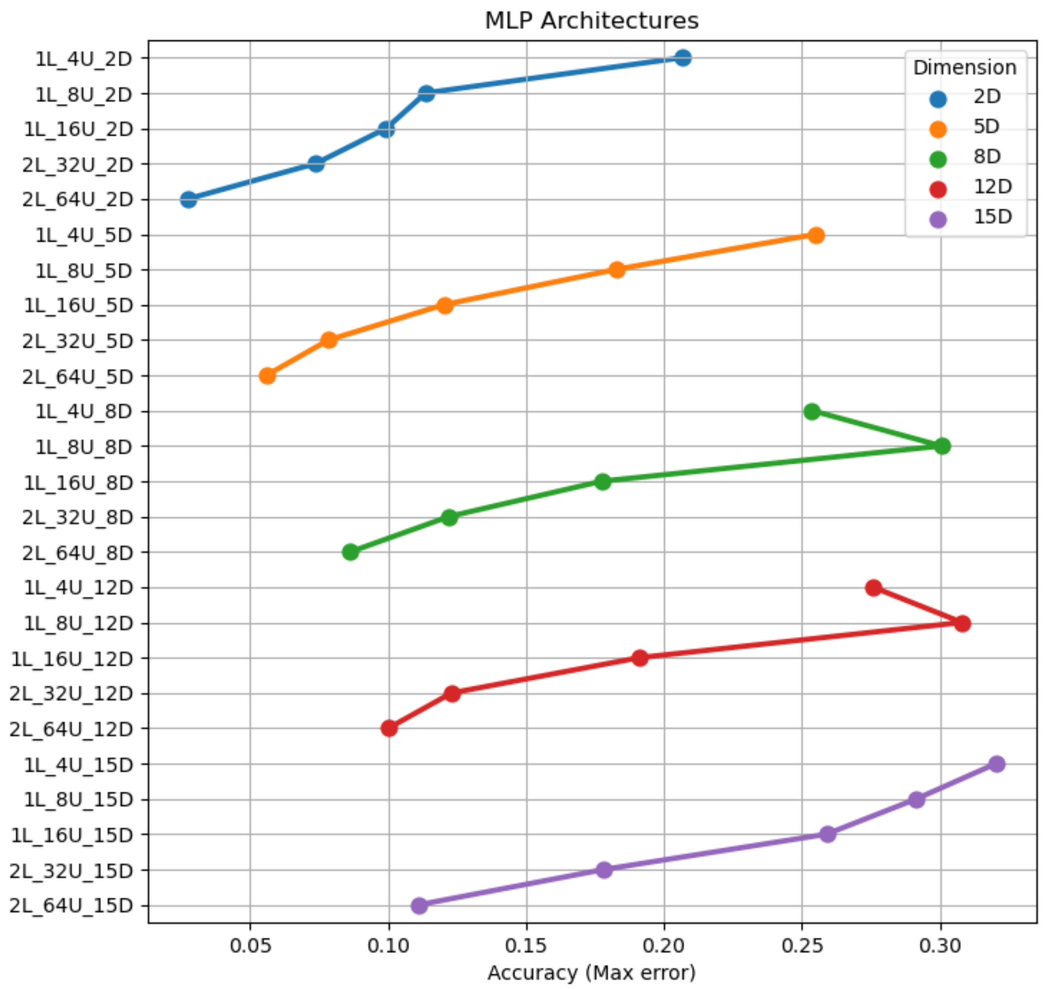


MODEL DESIGN

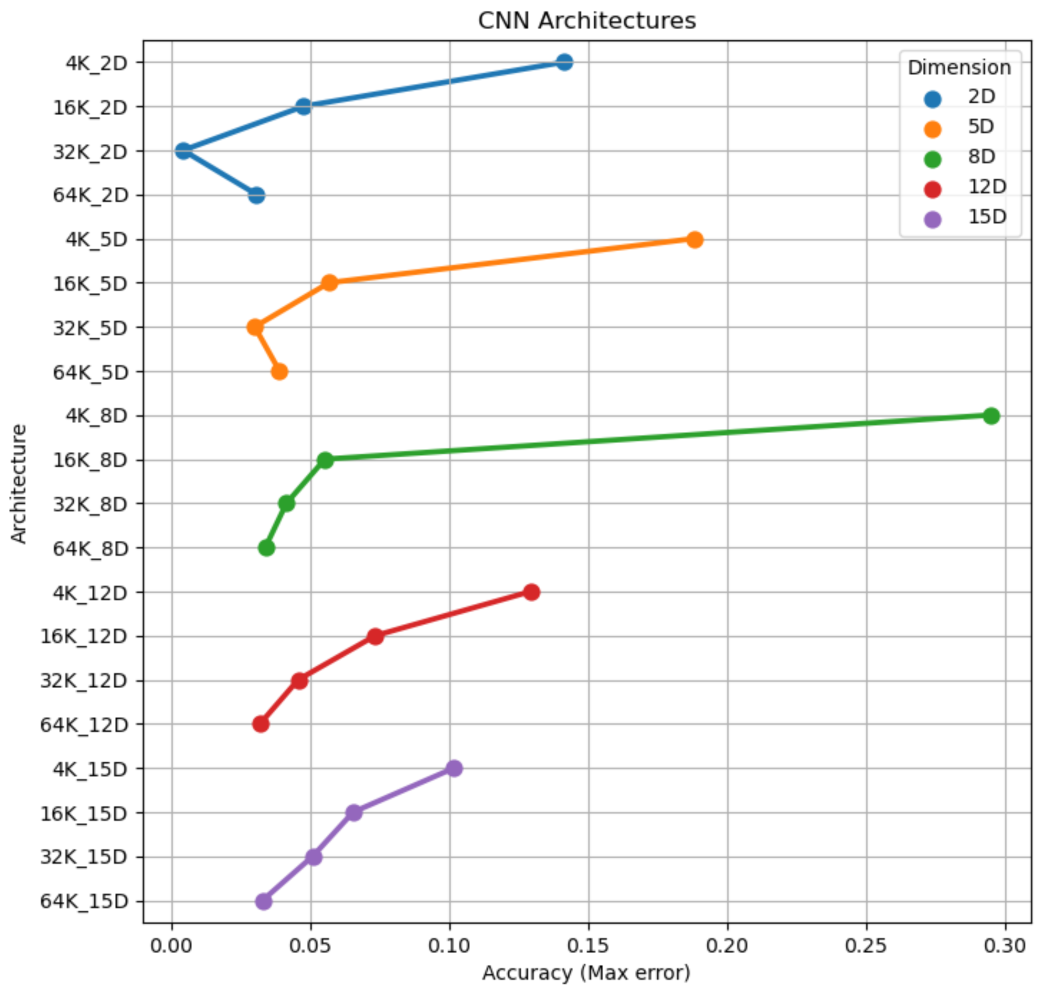
Multi-dimensional ML model design:

- MLP design using Tensorflow and Keras library:
  - Varying depth (number of layer) and width (number of neuron in each layer).
- 1-D CNN design using Tensorflow and Keras Library:
  - 1 convolution layer and variations of number of filter with fixed kernel size equal to 2.
- Lattice LUT design using Tensorflow lattice library:
  - Varying the lattice sizes.
- PWA design using Python code source available at https://github.com/bemporad/PyPARC.git and modify the code under Apache 2.0 license to add metrics for accuracy comparison with other ML models:
  - Varying number of partitions.

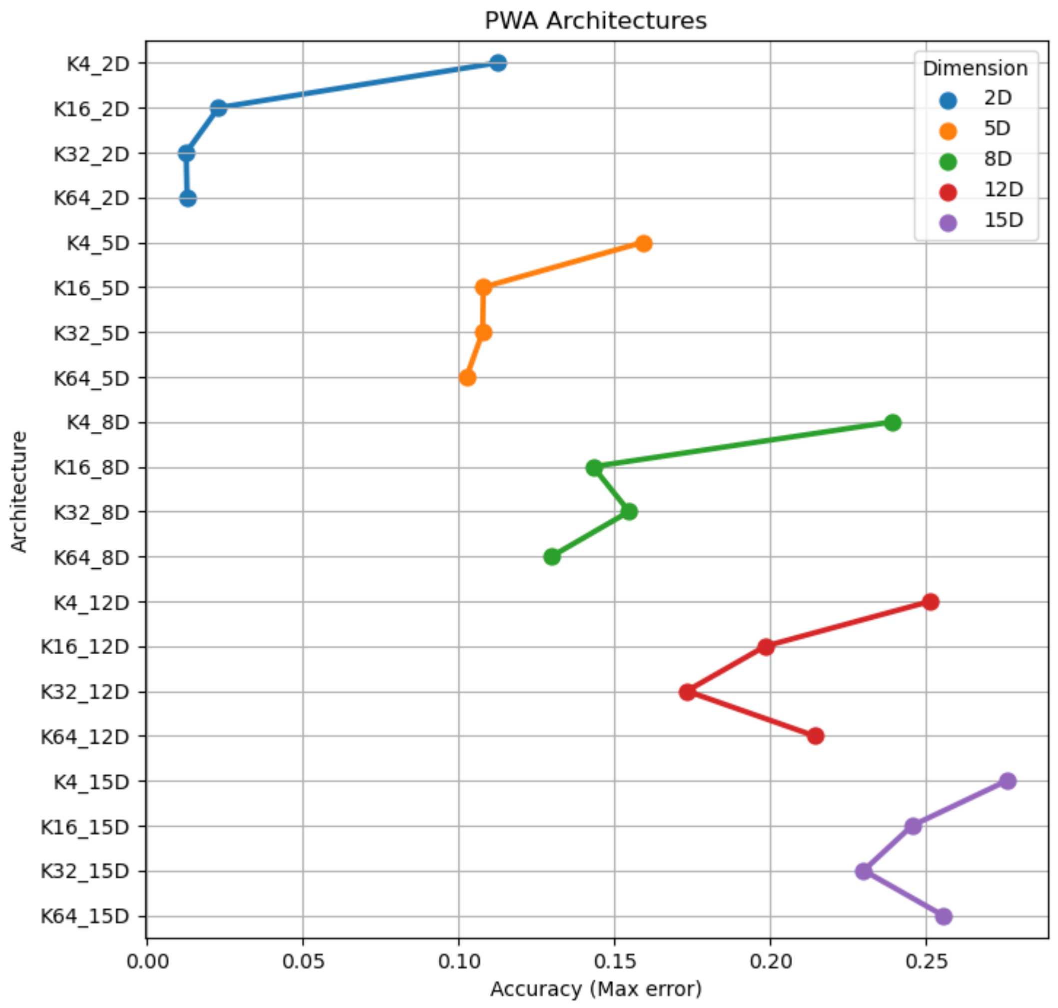
AI MODEL TRAINING: MLP Architectures



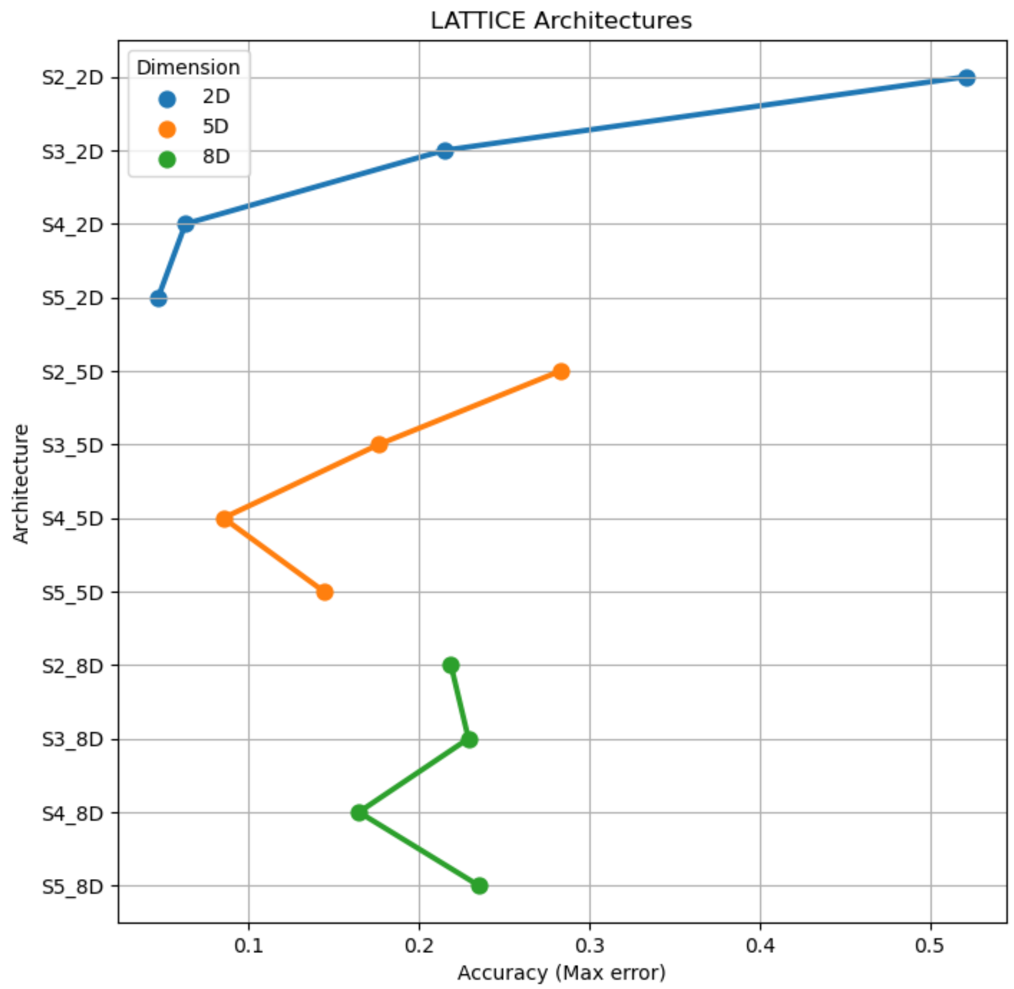
AI MODEL TRAINING: CNN Architectures



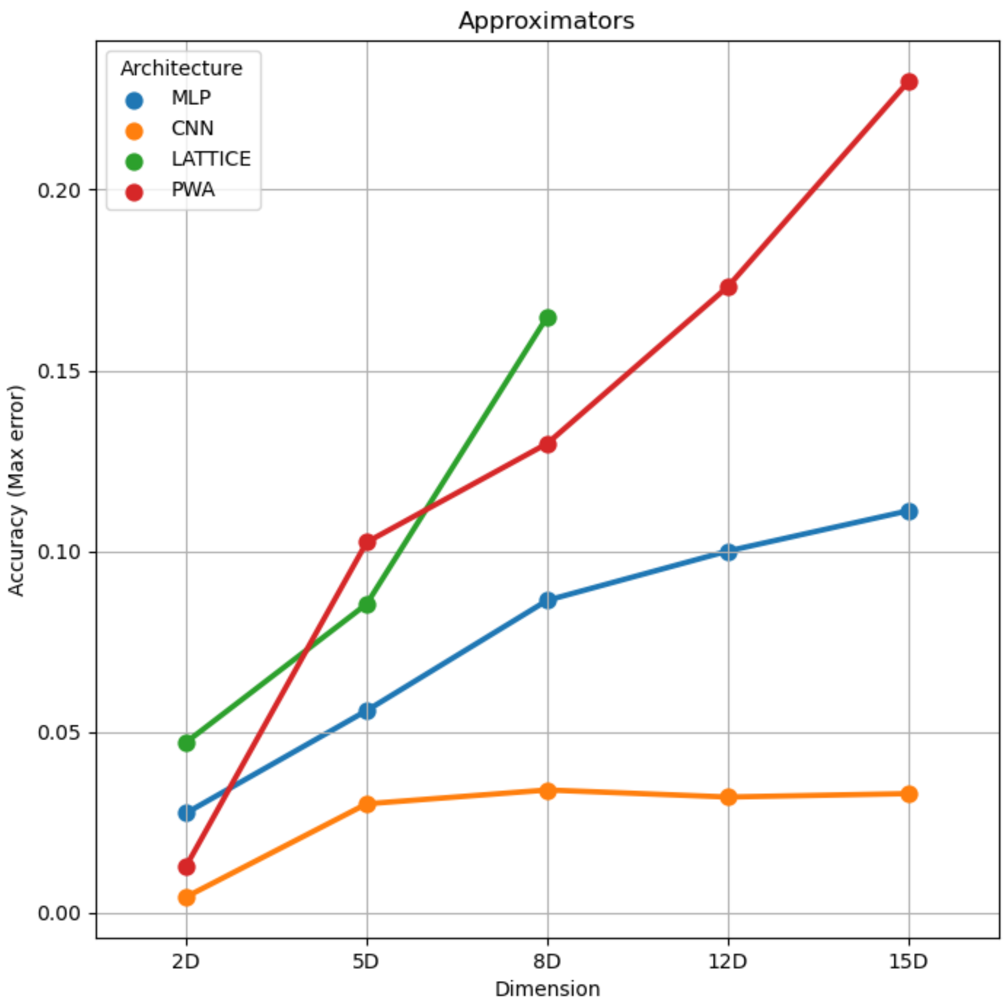
AI MODEL TRAINING: PWA Architectures



AI MODEL TRAINING: LATTICE Architectures

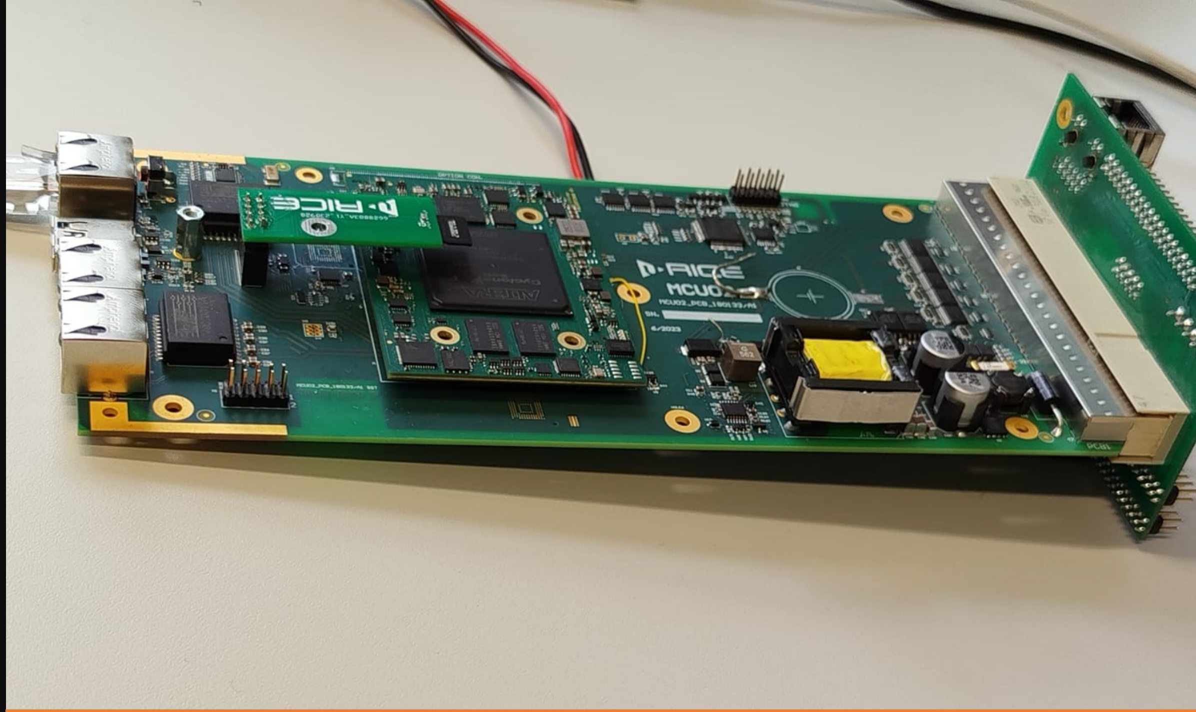


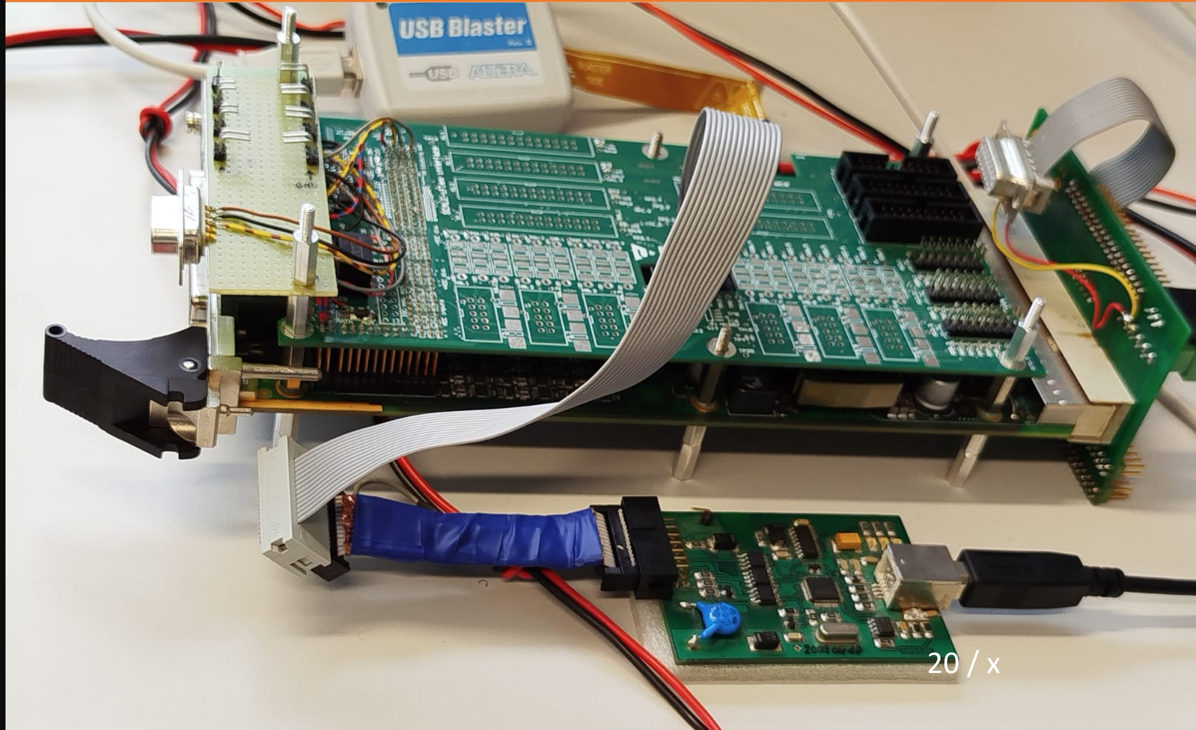
AI MODEL TRAINING: APPROXIMATORS



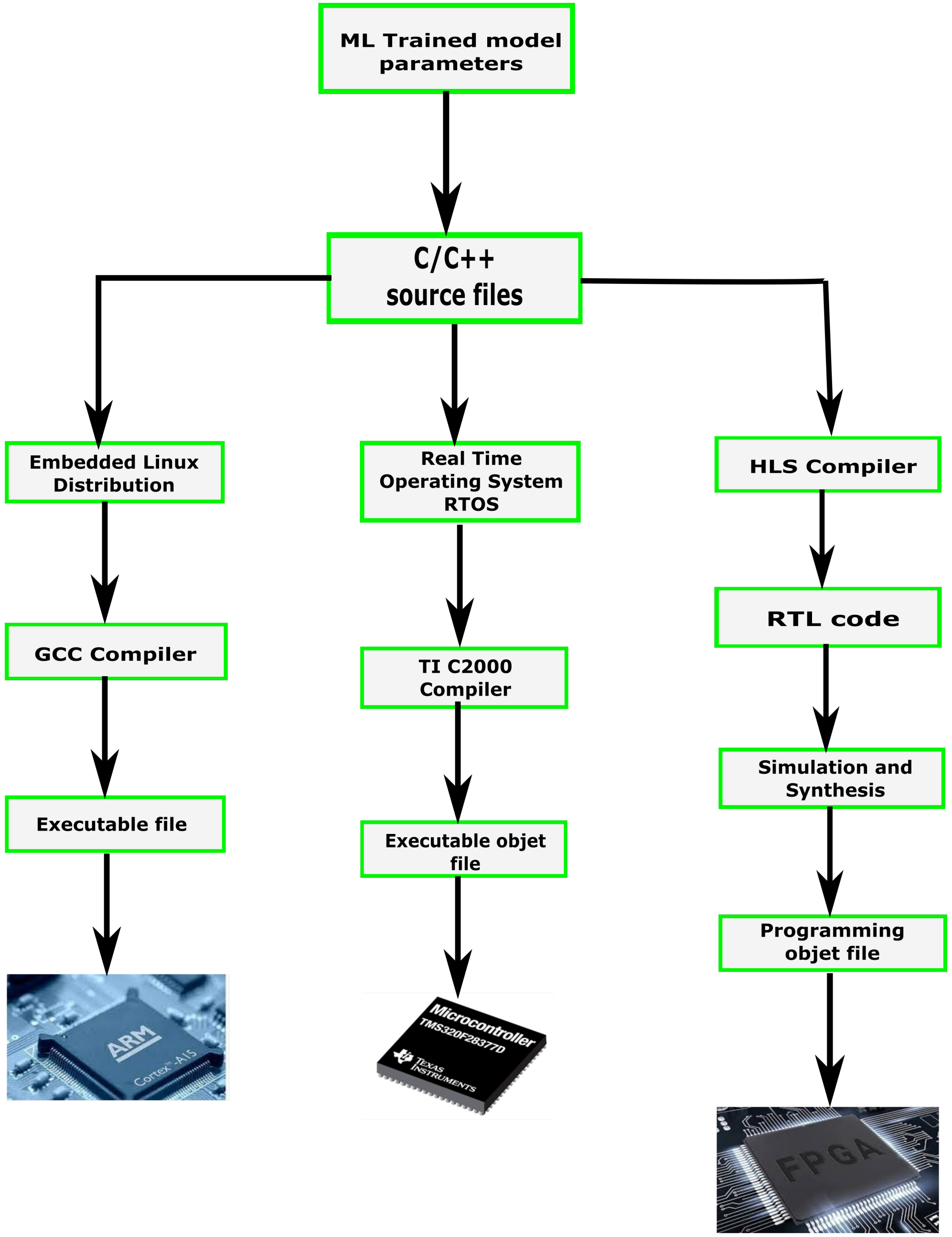
AI MODEL INFERENCE

- Our industrial computer systems are based on:
  - SoC FPGA Cortex-A Arm – embedded Linux.
  - Cyclone V FPGA.
  - Texas instruments dual core MCU.





APPROXIMATORS IMPLEMENTATION: SOFTWARE DEVELOPMENT FLOW

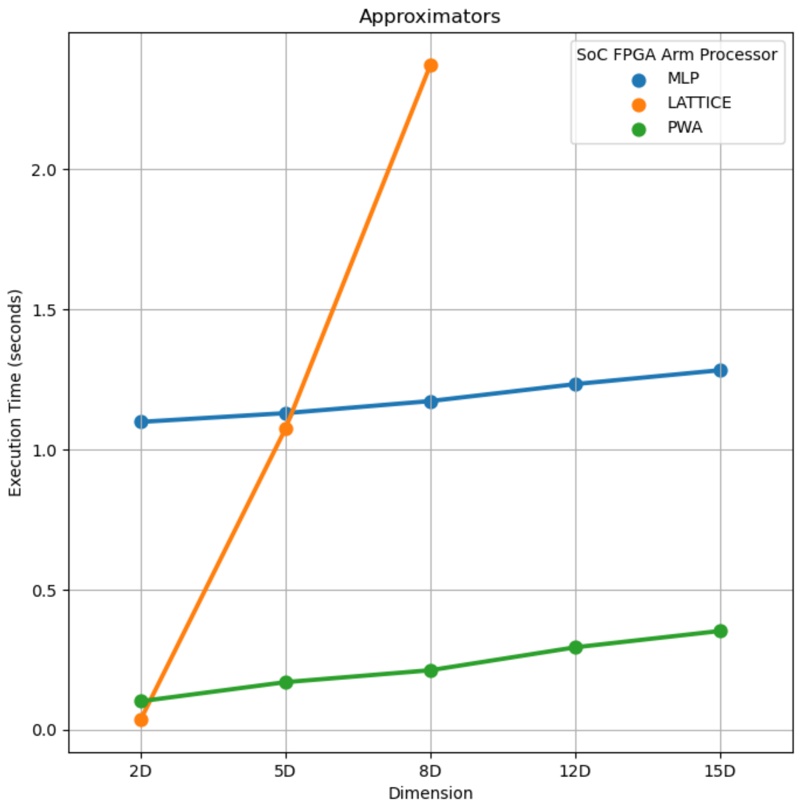


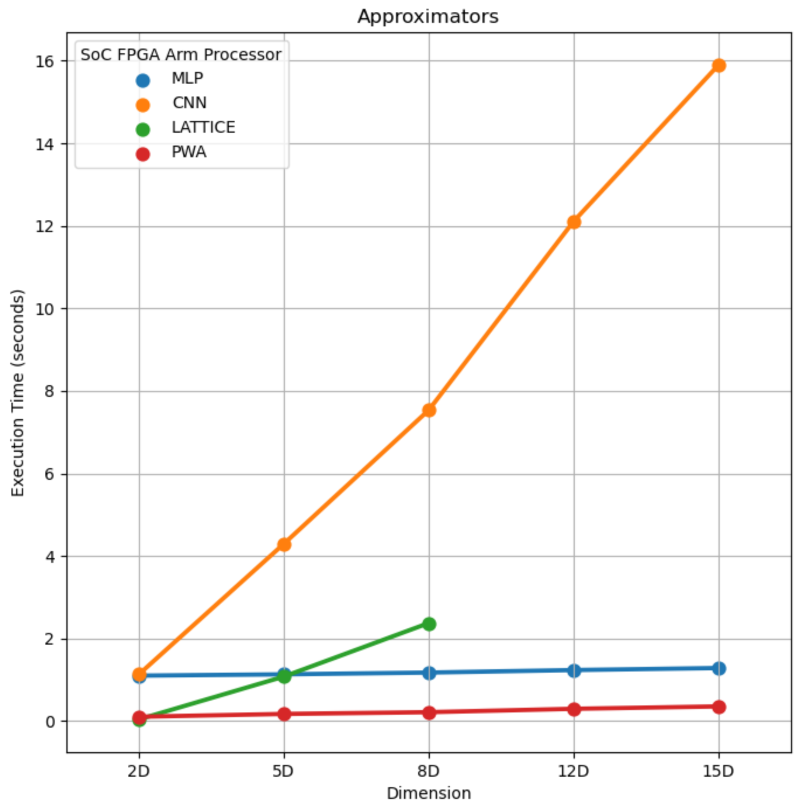
AI INFERENCE: SoC FPGA

Windows PowerShell | bosson@socfpga: ~/codes/te

```
rm -f ./obj/read_csv.o ./obj/write_csv.o ./obj/mlp_classifier.o
rm -rf MLP*
bosson@socfpga:~/codes/test1$ make
gcc -g -Wall -I ./include -c src/read_csv.c -o obj/read_csv.o
gcc -g -Wall -I ./include -c src/write_csv.c -o obj/write_csv.o
gcc -g -Wall -I ./include -c src/mlp_classifier.c -o obj/mlp_classifier.o
gcc -g -Wall src/main.c ./obj/read_csv.o ./obj/write_csv.o ./obj/mlp_classifier.o -o MLP -I ./include -lm
bosson@socfpga:~/codes/test1$ ./MLP 2 64,64 relu,relu 1 identity data/test_dataset_2.csv 4000 3 2d
Predictions:
------------
Predictions test example 4000 of 4000
Done.
Output file predictions generated to ./data/data_test_output.csv.
Execution time saved to ./data/execution_time.txt
Time it took to execute: 1.098884
bosson@socfpga:~/codes/test1$ ./MLP 2 64,64 relu,relu 1 identity data/test_dataset_5.csv 4000 6 5d
Predictions:
------------
Predictions test example 4000 of 4000
Done.
Output file predictions generated to ./data/data_test_output.csv.
Execution time saved to ./data/execution_time.txt
Time it took to execute: 1.130062
bosson@socfpga:~/codes/test1$ ./MLP 2 64,64 relu,relu 1 identity data/test_dataset_8.csv 4000 9 8d
Predictions:
------------
Predictions test example 4000 of 4000
Done.
Output file predictions generated to ./data/data_test_output.csv.
Execution time saved to ./data/execution_time.txt
Time it took to execute: 1.173177
bosson@socfpga:~/codes/test1$ ./MLP 2 64,64 relu,relu 1 identity data/test_dataset_12.csv 4000 13 12d
Predictions:
------------
Predictions test example 4000 of 4000
Done.
Output file predictions generated to ./data/data_test_output.csv.
Execution time saved to ./data/execution_time.txt
Time it took to execute: 1.233671
bosson@socfpga:~/codes/test1$ ./MLP 2 64,64 relu,relu 1 identity data/test_dataset_15.csv 4000 16 15d
Predictions:
------------
Predictions test example 4000 of 4000
Done.
```

AI INFERENCE-SoC FPGA: Execution time vs Dimension: 4K streaming data.

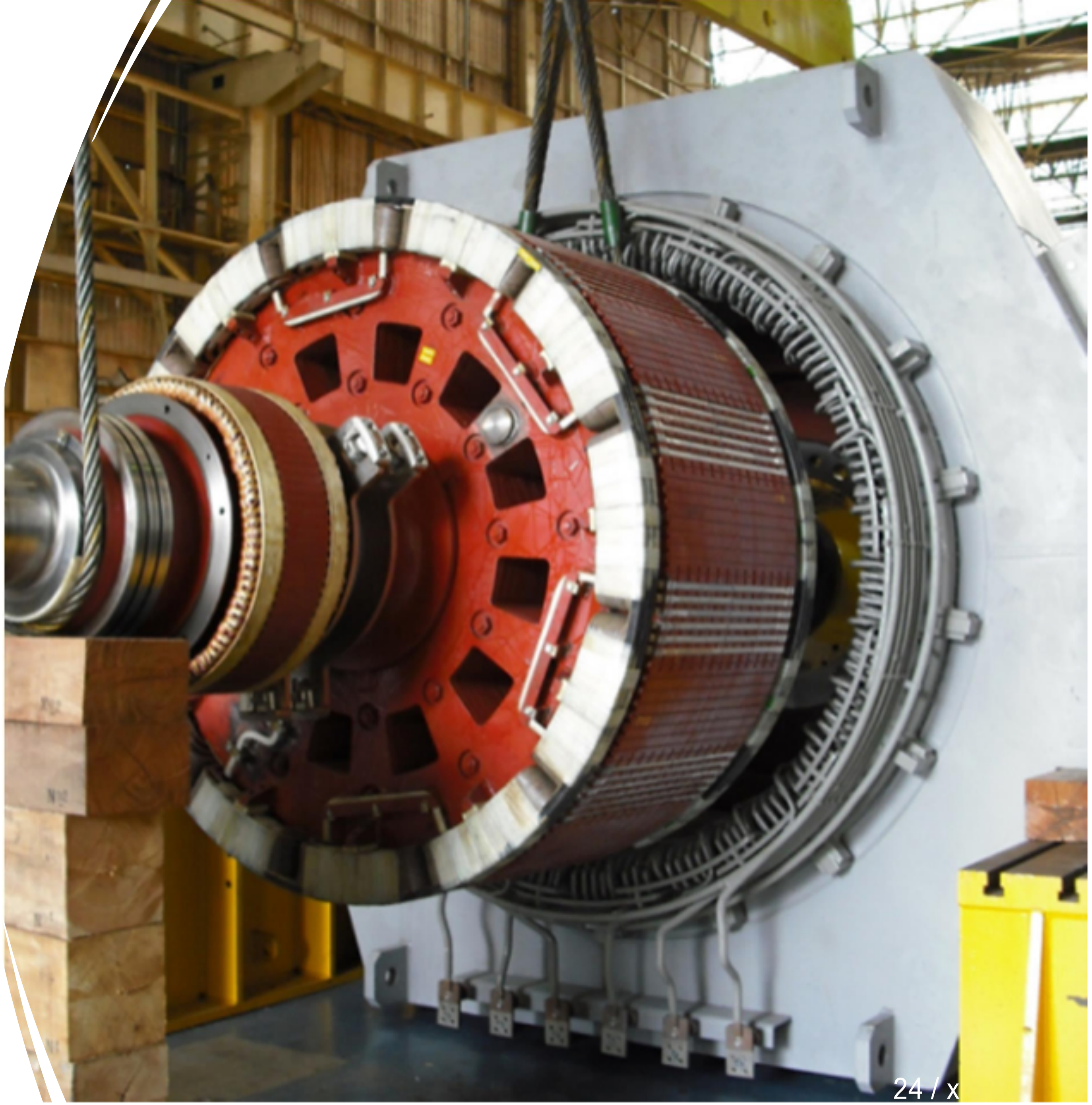




APPLICATION EXAMPLE: DIAGNOSTIC OF SYNCHRONOUS GENERATOR

Main project is online diagnostic of large SG with power range of hundred MW used in a power plant:

- Collect data from sensors installed inside the SG.
- Design a binary classification for fault detection or design a regression model for fault prediction/state of the health estimation.
- Perform some feature engineering on the raw data then use a MLP to design the diagnostic model.
- Or use a 1DCNN directly on the raw data to design a diagnostic model.
- Deploy the model inference on embedded systems.



Thank You for your attention!

Serge Pacome Bosson
377 634 408
bosson@fel.zcu.cz
fel.zcu.cz