High-Level Synthesis for Machine Learning

Nicolò Ghielmetti

FPGA Developers Forum

12th June 2024

Agenda

- In this presentation I'll walk you through:
  - **hls4ml** workflow
  - Exploited techniques for model compression and/or high performance
    - Post-Training Quantisation, Quantisation-Aware Training, Pruning, FIFO depth optimisation
  - Applications implemented using **hls4ml**
    - CMS Anomaly Detection - currently online
    - Low latency segmentation for autonomous vehicles
    - High-performance edge computing image segmentation for earth observation satellites
  - Future developments
    - Quantisation in different ways - more optimised or flexible
    - Transformers and graph networks - NN partitioning
    - Sweet spot area between fully on-chip architectures and DPU

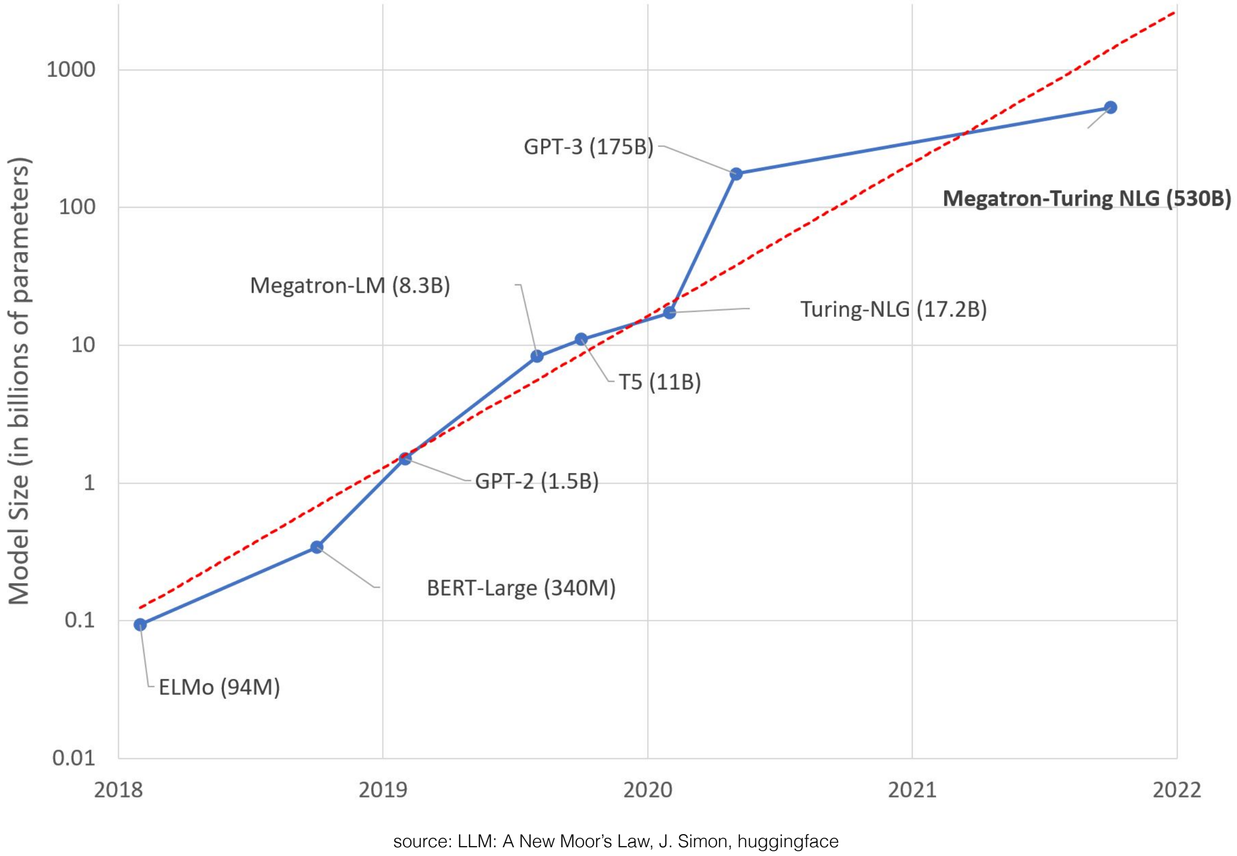
About me

- MSc in Computer Science Engineering @ Politecnico di Milano
  - Thesis: “Precision vs. Efficiency: A Bit-Level Variable-Length Floating-Point Approach to Neural Network Quantisation”
- Early Career Professional at CERN (previously Technical Student) working on **hls4ml** repository and applications mainly to Knowledge Transfer related projects, EP department
  - **hls4ml** submission for TinyMLPerf
  - Real-time semantic segmentation on FPGAs for autonomous vehicles with **hls4ml**
  - Edge-SpAIce (Work-in-Progress)

nicolo.ghielmetti@cern.ch
@nghielme

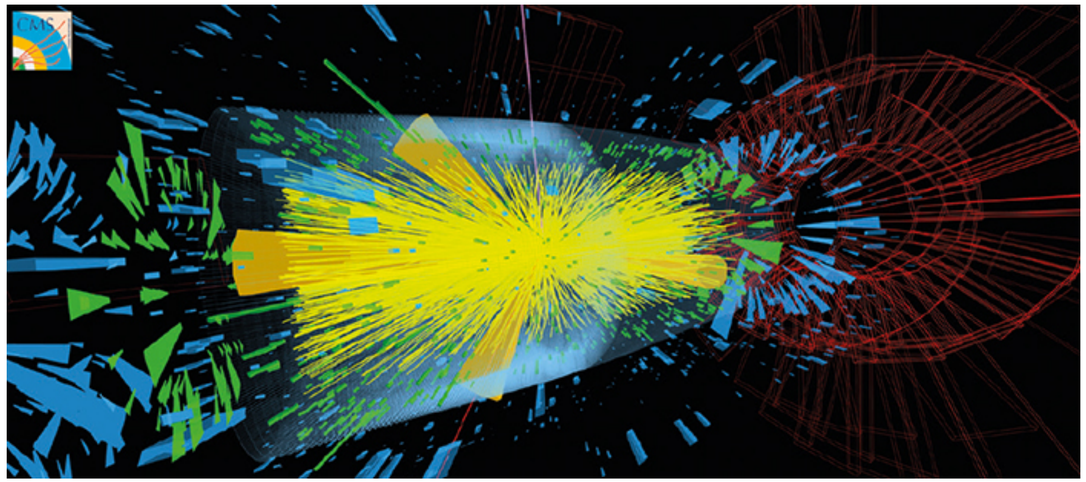
Motivation

- Machine Learning algorithms predictive performance improved a lot during the last years
  - Is predictive performance all we should care about?



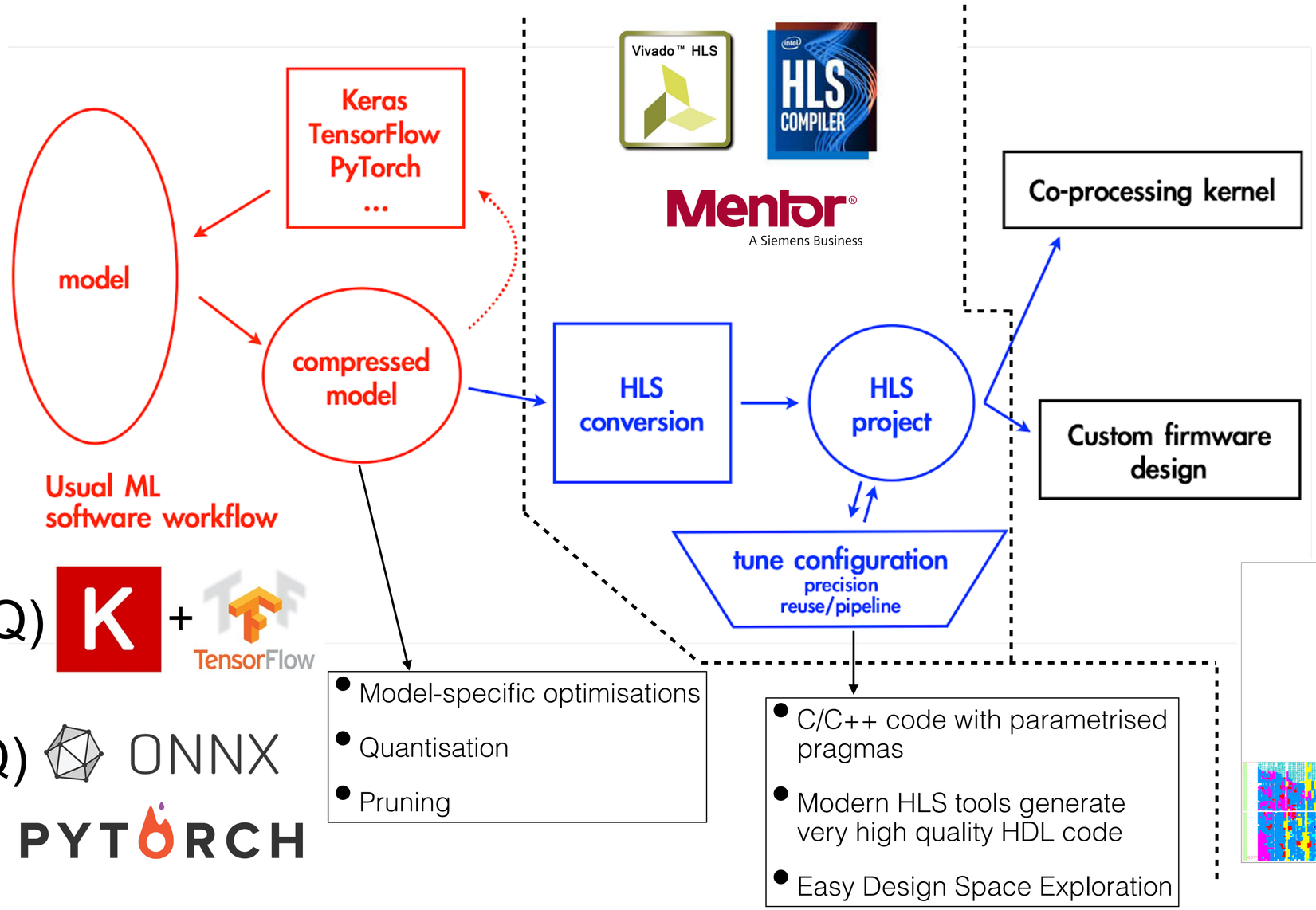
source: LLM: A New Moor's Law, J. Simon, huggingface

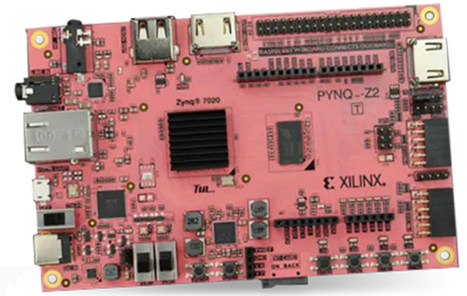
CMS Experiment
40MHz proton-proton collisions filtered down to 100kHz

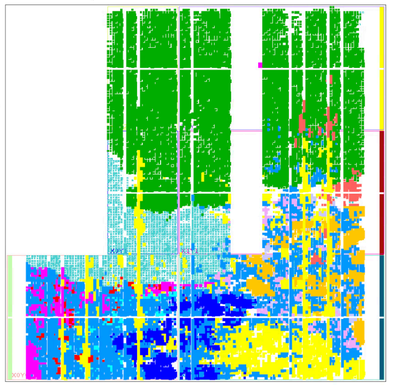


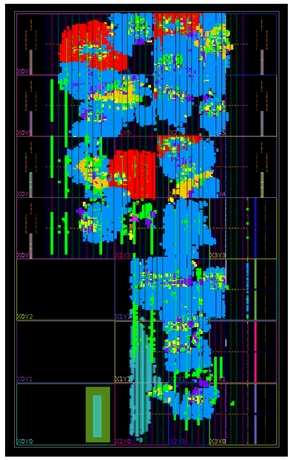
CubeSat
Computation is constrained by limited power environment

high level synthesis for machine learning





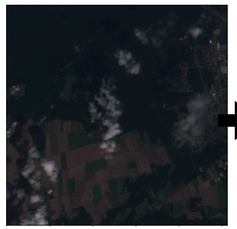




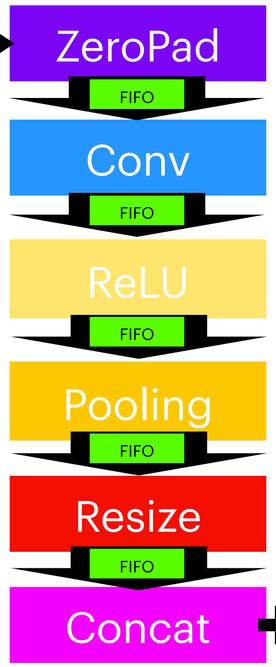
fastmachinelearning.org/hls4ml/

hls4ml architecture vs DPU

- Dataflow architecture: each layer is an independent compute unit (Task-Level Parallelism)
- The layers exchange data by using variable length FIFOs. Large input data can lead to the implementation of large and resource consuming FIFOs
- The whole neural network is implemented **on-chip**

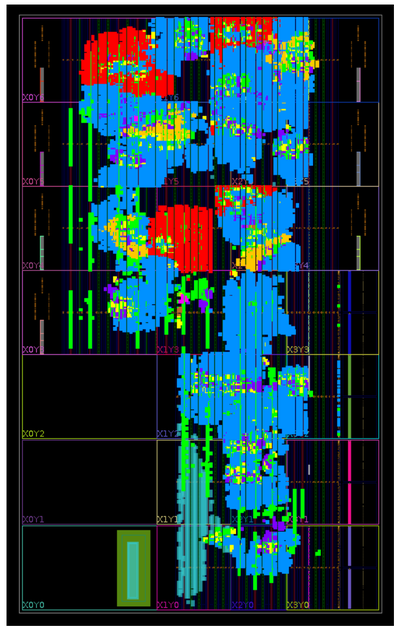


Input image
(from off-chip memory)

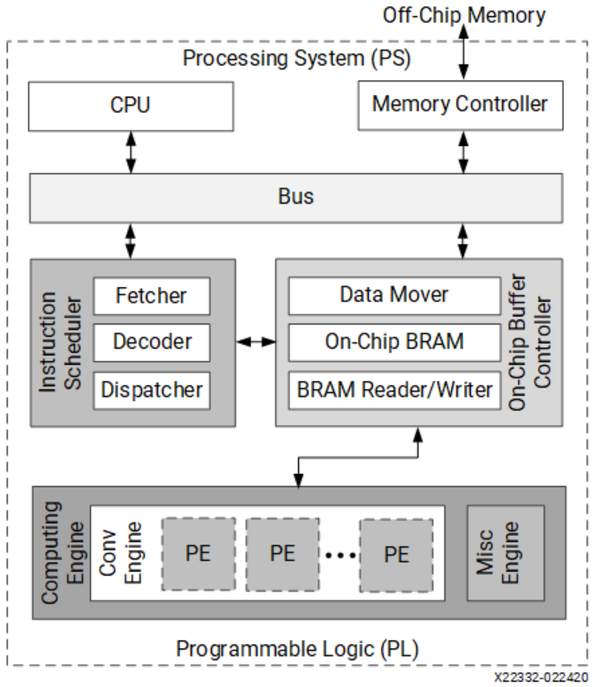


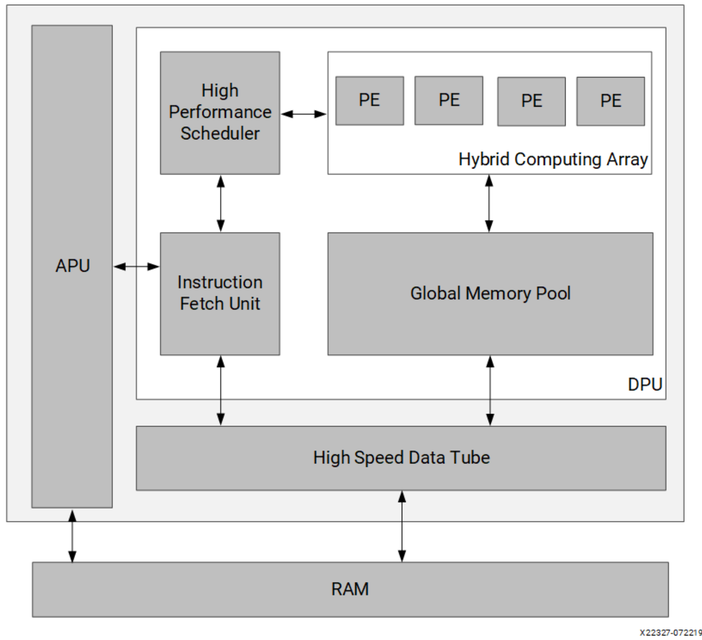
Output segmented image
(to off-chip memory)

FPGA floorplan
UNet segmentation network



- Deep Learning Processing Unit (DPU) architecture: by fetching instructions from **off-chip memory**, the DPU execute different parts of the neural network
- Typically 8-bit fixed point quantisation
- On-chip memory is used to buffer input, intermediate, and output data

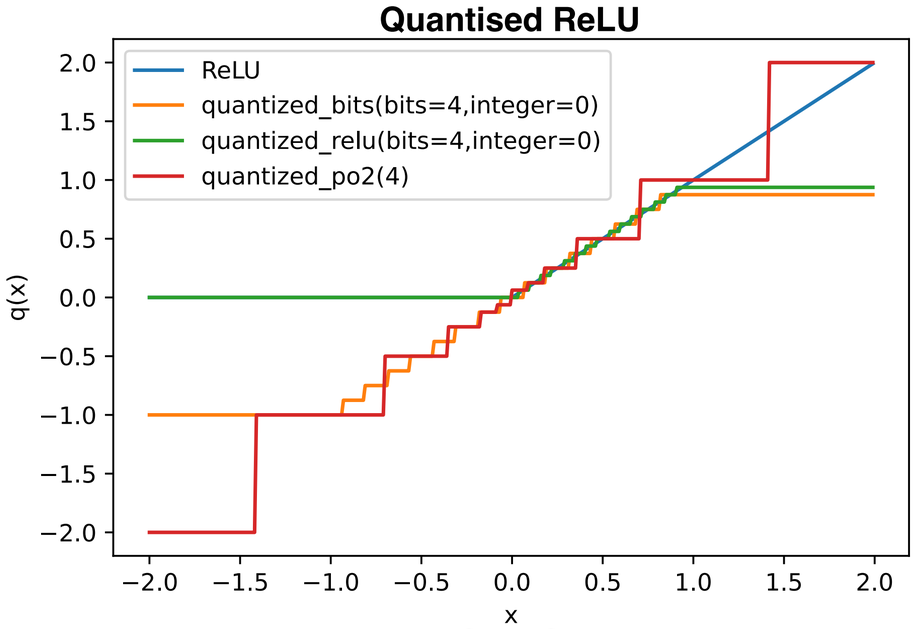


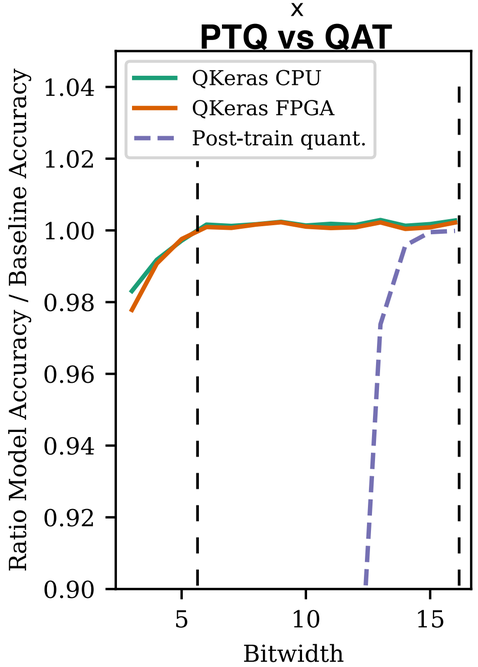


source: DPUCZDX8G for Zynq UltraScale+ MPSoCs Product Guide

hls4ml Model Compression: Quantization

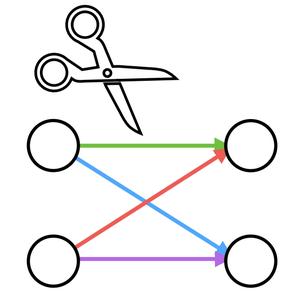
- Typically neural networks are trained using floating point datatypes for weights and activation functions, supported by CPUs and GPUs
  - However on FPGAs integer (or fixed point) operations result much more efficient for resources and latency
- Problem: how to pass from floating point to fixed point without loosing precision?
  - "Direct" approach: cast floating point to a "*big enough*" fixed point representation after training, aka **Post-Training Quantisation (PTQ)**
  - Introduce quantisation in the loop: consider limited precision and range of quantised datatypes as constraints for weights and activations during training, aka **Quantisation-Aware Training (QAT)**
- PTQ can be applied directly to a floating point trained model but it is not easy to achieve good prediction accuracy after quantisation with low bitwidth fixed-point representations
- QAT requires to customise the model in order to simulate quantisation during training but results in better prediction accuracy for lower bitwidths

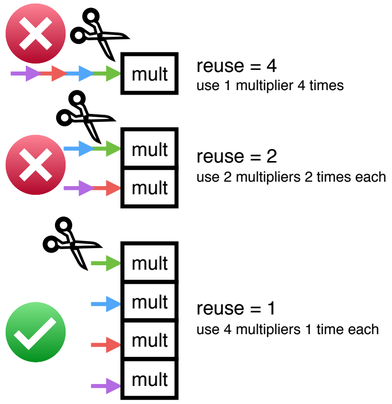


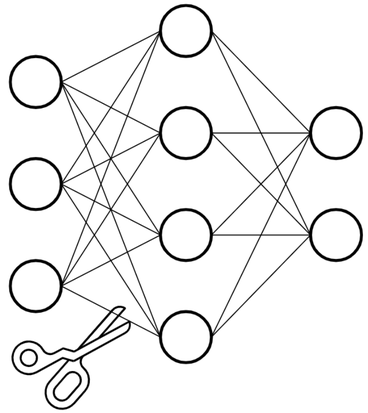


hls4ml Model Compression: Pruning

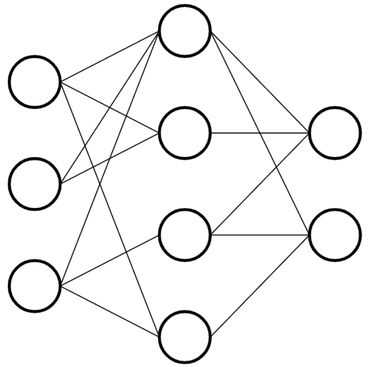
- A neural network may contain many redundant connections
- Pruning methods generally remove some connections from the final model, usually applied at training time
- **hls4ml**'s fully unrolled implementations can avoid unnecessary logic for pruned connections and save resources
- Different methods:
  - Regularisation - penalise low value weights, then make them 0
  - Set a target level of sparsity (i.e. weights set to 0) of the neural network after pruning
  - Structured pruning - remove continuous blocks of weights
  - Filter pruning - remove entire filters of CNN



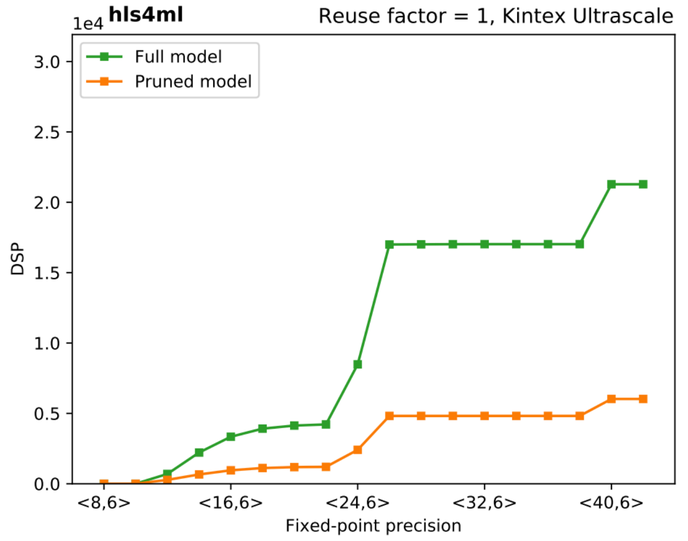




Before pruning

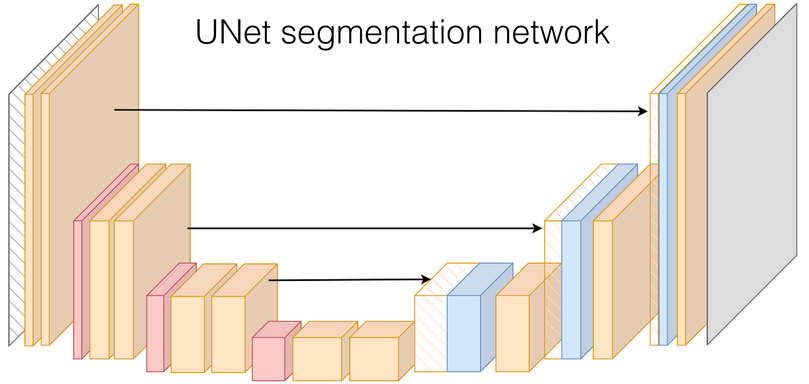


After pruning



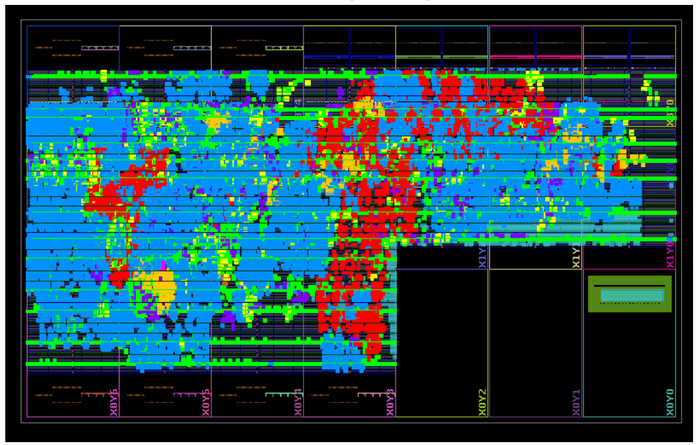
hls4ml HW Compression: FIFO Depth Optimization

- By default **hls4ml** set FIFO depths equal to ***w x h*** of input tensors in order to avoid race conditions by layers accessing the FIFOs
- For small input data and small networks this “worst-case scenario” approach could be acceptable
- For large networks, the use of large FIFOs may lead to the impossibility of implementing the neural network on FPGAs
- By using part of the input dataset as a **calibration** data set, the optimisation aims to identify, through co-simulation, the maximum size to be allocated to the system's FIFOs



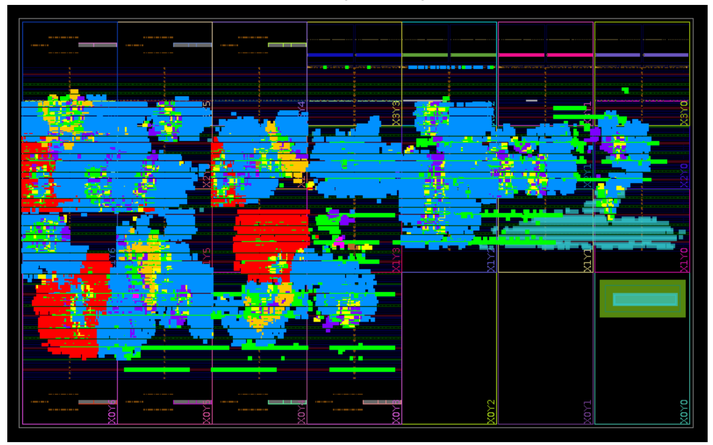
Layer FIFO: 0,1,2,3… … … … …,178,179

Before FIFO depth optimisation

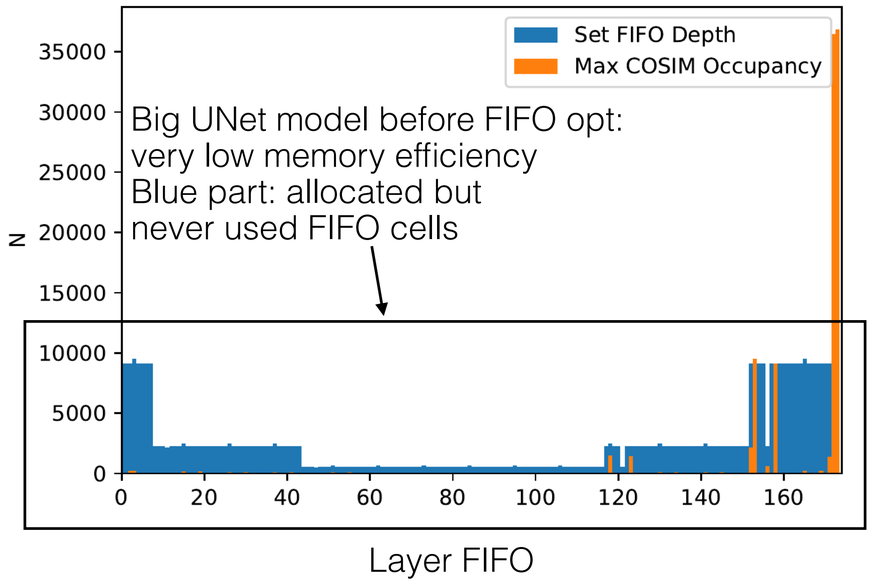


FIFOs

After FIFO depth optimisation



FIFOs



Applications

hls4ml applications: Anomaly Detection at CMS

- CMS experiment exploits a trigger system composed by around 100 FPGAs to filter 40MHz proton-proton collisions down to 100KHz

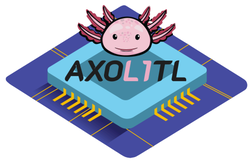
- AXOL1TL is a trigger algorithm designed to detect new physics based on a Variational AutoEncoder (VAE)

- VAE has been trained unsupervised on unbiased data comprised mostly of background events. Input data that are statistically different from previous seen data are identified as anomalies and collected for inspection

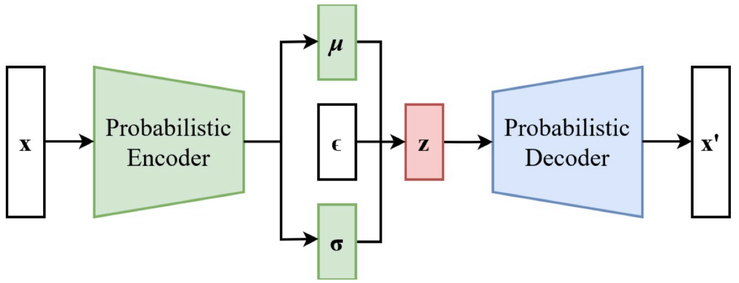
- Quantisation-Aware Training has been adopted to produce a model efficient for inference in hardware

  - Consumes 2% of LUTs of Xilinx Virtex 7 FPGAs (purple in floorplan)

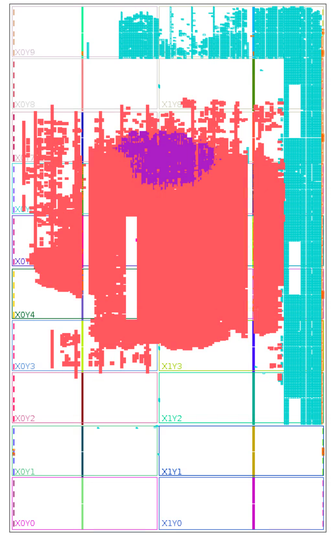
  - Inference latency: 50 ns, meeting the requirement from the Global Trigger system for deployment

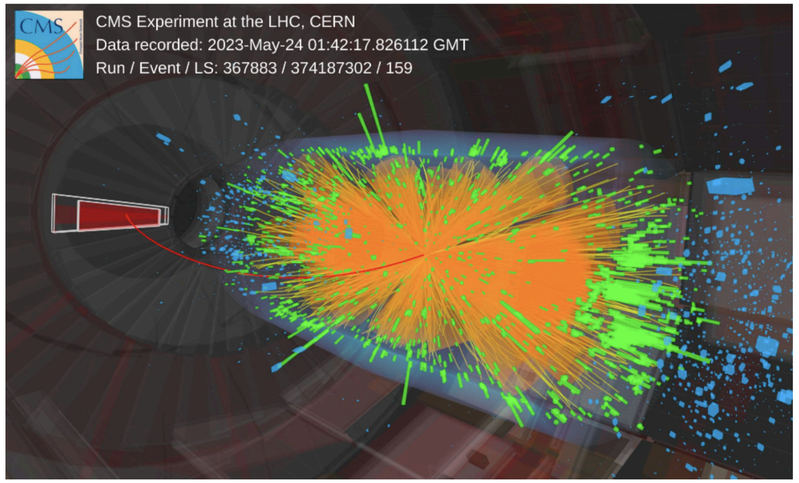


Variational Autoencoder



FPGA floorplan
Variational Autoencoder



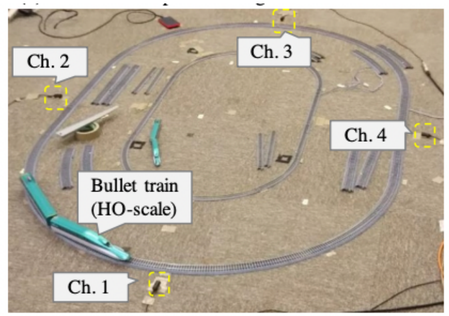


hls4ml applications: MLPerf Tiny ™

- MLPerf Tiny: benchmarks of Machine Learning for low power devices (MLPerf Tiny) organised by MLCommons
- 4 benchmark datasets, open/closed division allowing/disallowing model retraining or model modifications
- **hls4ml** in open category (for Quantisation Aware Training) achieves competitive performance

Benchmark		CIFAR-10			ToyADMOS		
Team	Device	Accuracy	Latency (ms)	Energy (uJ)	AUC	Latency (ms)	Energy (uJ)
hls4ml	**Pynq-z2[1]**	**83.5%**	**7.64**	**12266**	**0.83**	**0.019**	**30.1**
GreenWaves	GAP9 EVK[2]	85%	0.62	40.4	0.85	0.18	7.3
STMicro	Nucleo-L4R5ZI[3]	85%	54.3	8707	0.85	1.82	266.5
OctoML	Nucleo-L4R5ZI[3]	85%	389.2	21342	0.85	11.7	633.7

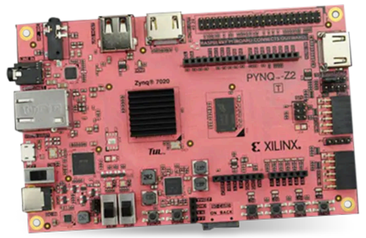
CIFAR-10 dataset



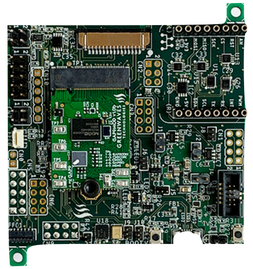
ToyADMOS dataset

hls4ml

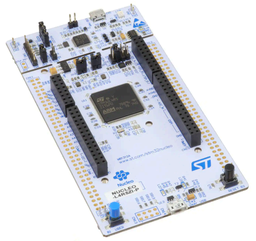
1



2



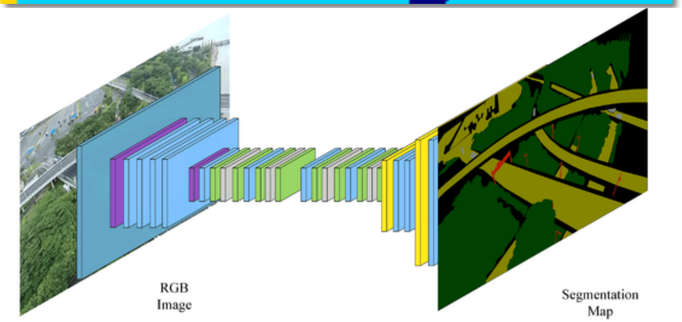
3



hls4ml applications: Autonomous Vehicles

- Not only particle physics research - through CERN Knowledge Transfer, activities at CERN contributes to society in many ways
  (e.g. WWW, LHC computing grid, this project :-))
- Project undertaken in partnership with Zenseact - Swedish autonomous vehicle ML solutions company (Paper, web)
- ENet (Efficient Net, very similar to UNet) image segmentation network trained on Cityscape dataset (road, vehicle, pedestrian, background - 4 output classes) using AutoQKeras for quantisation and Design Space Exploration
- Lowest latency model has around **10k parameters, 8 bit quantisation**
- Deployed on ZCU102 UltraScale+ MPSoC kit with **hls4ml**

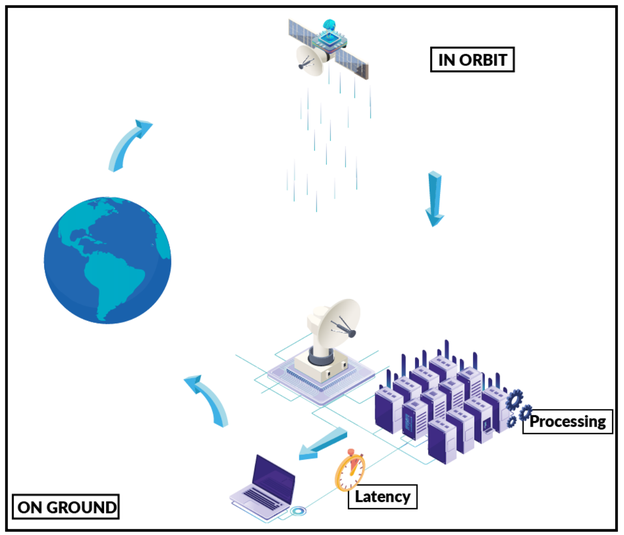
Model	Acc.	mIoU	Latency [ms] b=1	Latency [ms] b=10	BRAM	LUT	FF	DSP
EnetHQ	81.1%	36.8 %	4.9	30.6	224.5 (25%)	76,718 (30%)	87,059 (16%)	450 (18%)
Enet8Q4	77.6%	33.9 %	4.8	30.2	342.0 (37%)	166,741 (61%)	90,536 (16%)	0
Enet8Q8	77.1%	33.4 %	4.8	30.0	508.5 (56%)	126,458 (46%)	134,385 (25%)	1,502 (60%)
ENet [23]	-	63.1%	30.38 (720)[a]	-	257	62,599	192,212	689



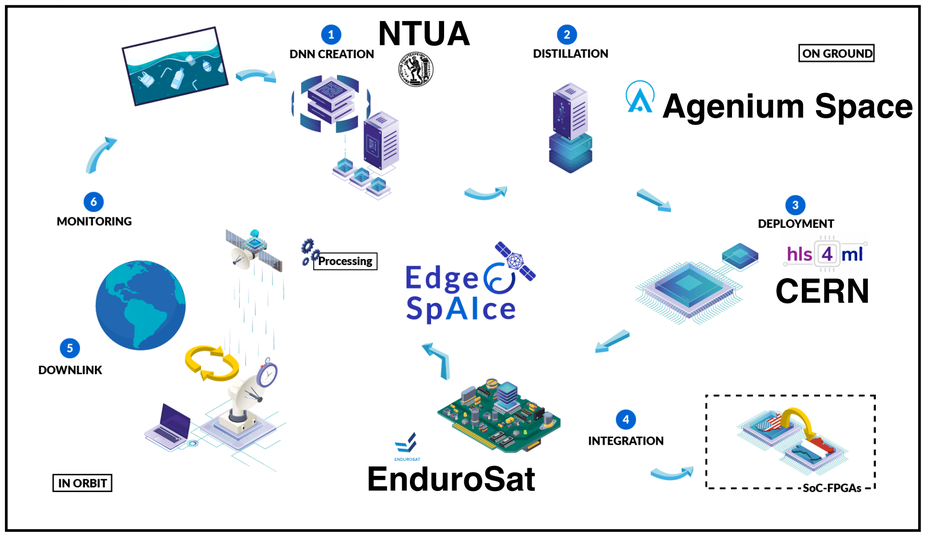
hls4ml applications: Earth Observation Satellites

- Using **hls4ml** to monitor plastics pollution in the oceans onboard Earth Observation satellites
- Satellites are normally equipped with radiation tolerant FPGA SoCs, typically used for communication purposes
- Downlink bandwidth is **limited**, and missions typically only look for certain objects (in this case, plastics pollution)
- Deploy DNN onboard satellite to identify pollution through image segmentation and downlink only selected images
- Image segmentation task applied to tiled 5-bands hyper-spectral images from MARIDA dataset
- Using **hls4ml** to reach the best *pixels / W / s* (with accuracy requirements)

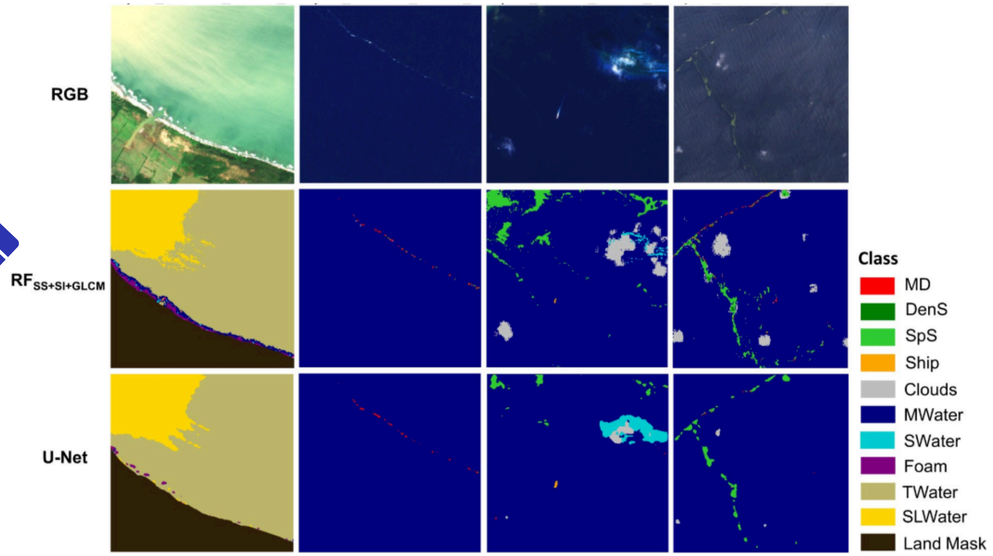
State-of-the-art



Edge SpAIce



Edge SpAIce



MARIDA

hls4ml applications: Earth Observation Satellites

- Agenium Space has experience in deploying neural networks onboard FPGAs adopting Xilinx Vitis AI tool and in NN compression exploiting **knowledge distillation**

- Vitis AI implements a DPU architecture on the FPGAs and exploits the full SoC to coordinate the computation of the network

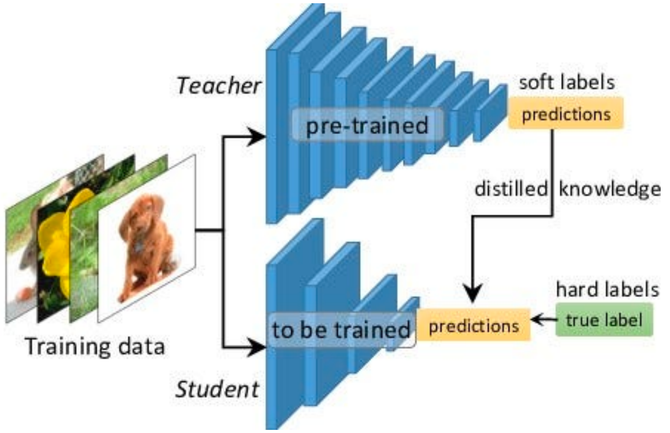
  - Many **power demanding off-chip memory accesses** since NN weights are not fully on-chip

- Adopting **hls4ml** a lower power consumption should be achieved thanks to the fully on-chip implementation

- Now working on cloud segmentation task training UNet on ALCD dataset

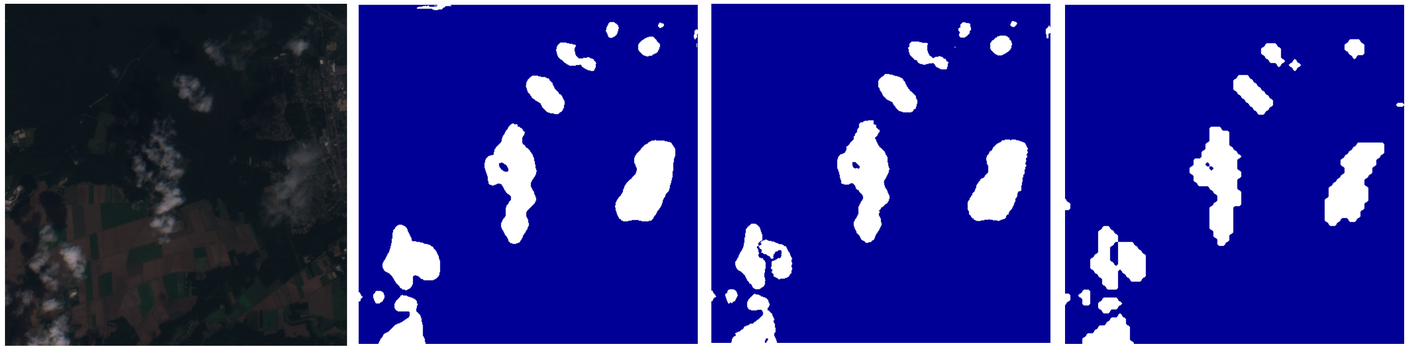
- 10k parameter model has been implemented on FPGA; working on specific optimisations to implement 100k parameter model on FPGA

Knowledge Distillation



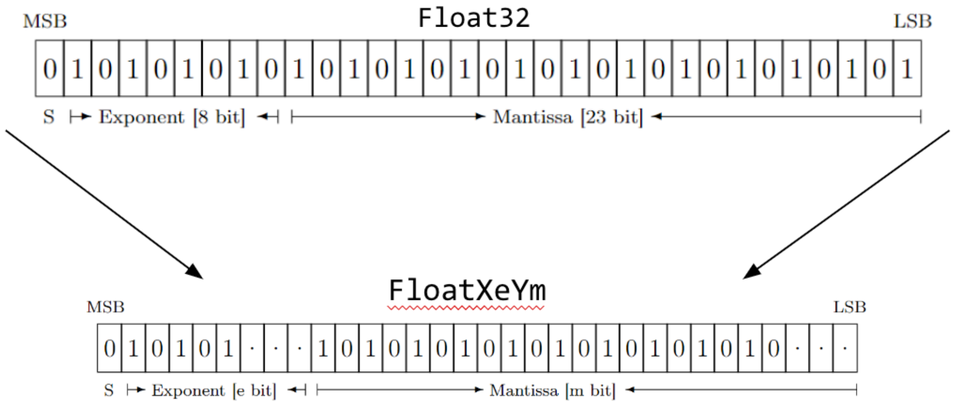
ZCU102 MPSoC - Ultrascale+ FPGA

Input cloud image | **Segmented - Float32** | **Segmented - FPGA** | **Target mask**



Future Developments

- There is a lot of active research on quantisation techniques
  - One recent approach effectively adopted on LLMs is to use compact version of floating point representation



- Find a sweet spot between current **hls4ml** fully on-chip approach and DPU
  - Fully on-chip implementations have advantages (reduced power consumption and latency) but the implementation of big models on a single board is hard/impossible
- Partitioning big models in order to deploy them in a distributed fashion on multiple FPGAs
  - Through QONNX it could be possible to estimate before HLS what is the resource cost and latency of parts of the network in order to perform the partitioning in a structured way

Thanks!
Questions?

Nicolò Ghielmetti

FPGA Developers Forum

12th June 2024