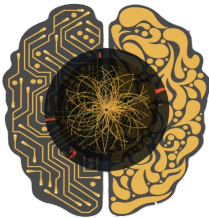


The **hls4ml** project and the future of deploying ultrafast deep learning on specialized hardware

Vladimir Lončar
For the FastML team
fastmachinelearning.org



Agenda

Introduction to **hls4ml** project

Brief history of the project

The current state of the project and recent developments

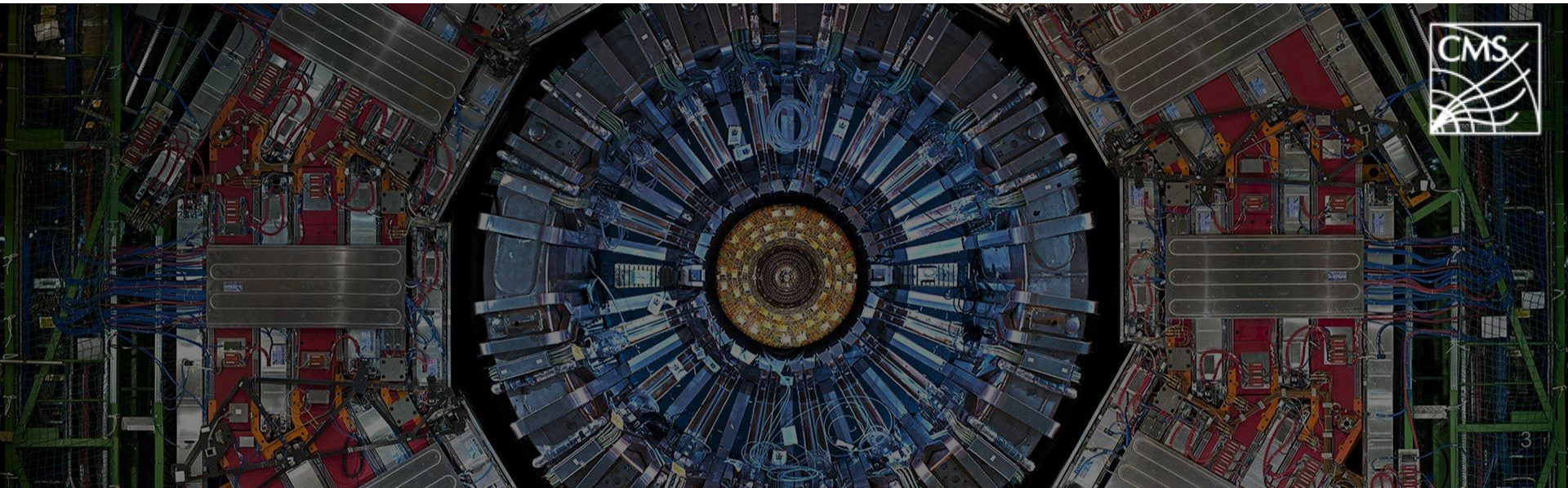
Potential new directions

The original motivation: triggering at (HL-)LHC

At the LHC proton beams collide at a frequency of 40 MHz

Extreme data rates of $O(100 \text{ TB/s})$

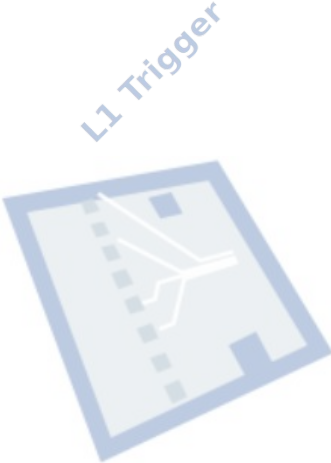
“Triggering” - Filter events to reduce data rates to manageable levels



Machine learning at LHC



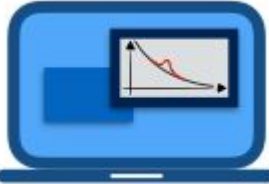
1 ns



1 μ s



100 ms



1 s

ML is already successfully employed in
offline analysis

Machine learning at LHC

40 MHz
pp
collisions



1 ns

L1 Trigger



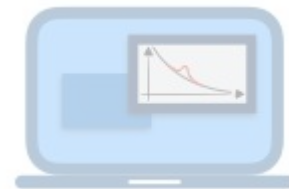
1 μ s

High-Level
Trigger



100 ms

Offline
Computing



1 s



Deploy ML algorithms very early

Challenge: strict latency constraints!

L1 trigger hardware

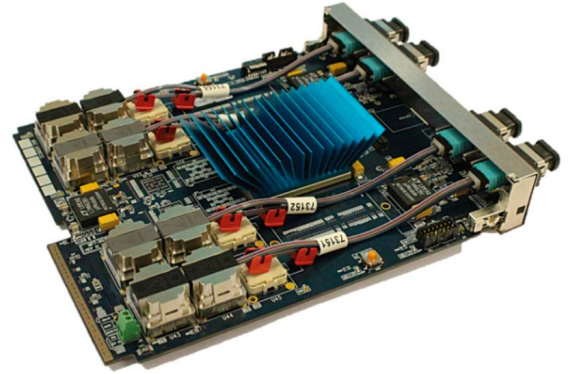
We need fast processing of raw data $O(\mu\text{s})$

- Not possible to use common hardware, such as Intel CPUs

Must be flexible and modular to support reconfiguration and upgrade/maintenance of modules

→ Field-programmable gate arrays (FPGAs)

At HL-LHC hundreds of such devices will be used



High-level synthesis for machine learning

FPGAs are programmed using Hardware Description Languages (HDLs)

- VHDL, Verilog

High Level Synthesis (HLS)

- Develop in C/C++ and compile to HDL
- Drastic decrease in firmware development time!

hls4ml is a library for translating neural networks into FPGA firmware

- Available as a Python package: `pip install hls4ml`

Inference with extremely low latency by using on-chip weights

- Much faster access times → lower latency

High-level synthesis for machine learning

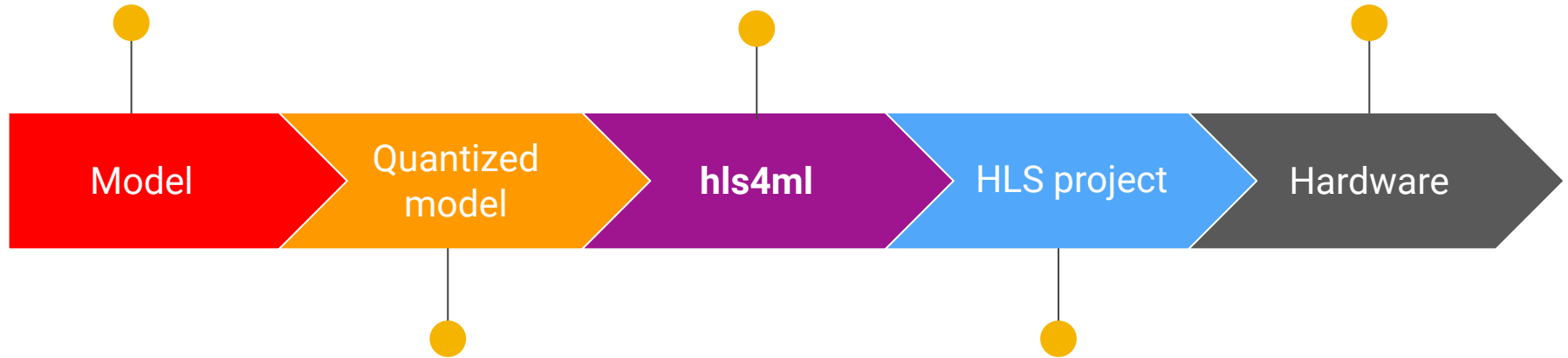
Supported DL frameworks:

-  Keras
-  PyTorch
-  ONNX

Model conversion,
optimization, profiling &
tuning



Xilinx and Intel/Altera FPGAs



Quantization and pruning
techniques:

- [QKeras + AutoQ](#)
- [QONNX](#)



A growing project


First paper published in 2018

- Currently ~20 papers published

In development since late 2017

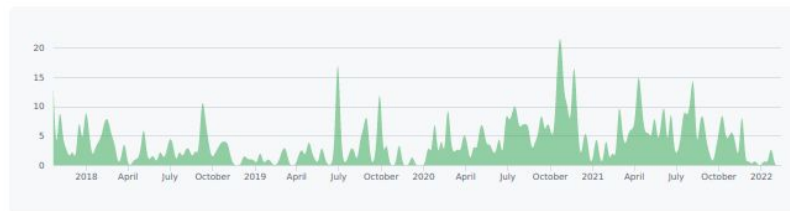
Commits on Oct 25, 2017

Initial commit

 benjaminkreis committed on Oct 25, 2017

Growing number of code contributions

Contributions to master, excluding merge commits and bot accounts



Jinst

PUBLISHED BY IOP PUBLISHING FOR SISSA MEDIALAB

RECEIVED: May 10, 2018

ACCEPTED: July 17, 2018

PUBLISHED: July 27, 2018

Fast inference of deep neural networks in FPGAs for particle physics

J. Duarte,^a S. Han,^b P. Harris,^b S. Jindariani,^a E. Kreinar,^c B. Kreis,^a J. Ngadiuba,^d M. Pierini,^d R. Rivera,^a N. Tran^{a,1} and Z. Wu^e

^aFermi National Accelerator Laboratory, Batavia, IL 60510, U.S.A.

^bMassachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

^cHawkEye360, Herndon, VA 20170, U.S.A.

^dCERN, CH-1211 Geneva 23, Switzerland

^eUniversity of Illinois at Chicago, Chicago, IL 60607, U.S.A.

E-mail: hls4ml.help@gmail.com

ABSTRACT: Recent results at the Large Hadron Collider (LHC) have pointed to enhanced physics capabilities through the improvement of the real-time event processing techniques. Machine learning methods are ubiquitous and have proven to be very powerful in LHC physics, and particle physics as a whole. However, exploration of the use of such techniques in low-latency, low-power FPGA (Field Programmable Gate Array) hardware has only just begun. FPGA-based trigger and data

The current feature-set

Supported network architectures:

- **DNNs** - [arxiv:1804:06913](https://arxiv.org/abs/1804.06913)
- **CNNs** - [arxiv:2101:05108](https://arxiv.org/abs/2101.05108)
- **Graph NNs** - GarNet architecture - [arxiv:2008.03601](https://arxiv.org/abs/2008.03601)

Tunable features

- User controllable trade-off between resource usage and latency/throughput
- Weights can be stored in registers or block RAM

Model compression

- Resource-efficient designs for pruned models
- Quantization with QKeras integration - [arxiv:2006.10159](https://arxiv.org/abs/2006.10159)
- Binary/Ternary layers (computation without using DSPs) - [arxiv:2003.06308](https://arxiv.org/abs/2003.06308)

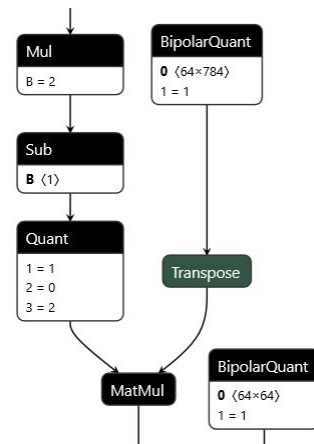
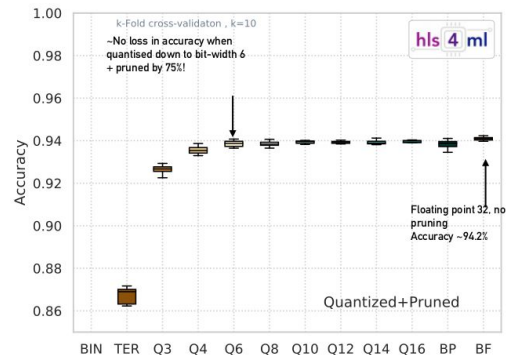
Interoperability with the other ML tools

QKeras ([website](#))

- Quantized Keras
 - Includes quantizers for weights and activations
- Drop-in replacements for Keras layers
- Developed in collaboration with Google

QONNX ([website](#))

- A dialect of ONNX that adds quantization operators
- Collaboration with Xilinx [FINN](#) team
- Exportable from [Brevitas](#) (quantized PyTorch)
 - Enabling interoperability with FINN



Applications - Anomaly detection

We throw away most LHC collision events in our Level 1 Trigger system

- Our selections are well motivated for analyses, but: what if we are missing something?
- Can we trigger on events which look “anomalous” compared to the Standard Model background?

Unsupervised learning using autoencoders ([paper](#))

- Both DNN and CNN architectures, pruned with QKeras
- Ultra low latency ($O(100\text{ns})$)

Model	DSP [%]	LUT [%]	FF [%]	BRAM [%]	Latency [ns]	II [ns]
DNN AE QAT 8 bits	2	5	1	0.5	130	5
CNN AE QAT 4 bits	8	47	5	6	1480	895
DNN VAE PTQ 8 bits	1	3	0.5	0.3	80	5
CNN VAE PTQ 8 bits	10	12	4	2	365	115

Applications - Computer vision

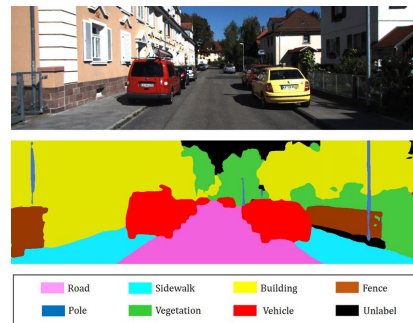
Streaming implementation for larger CNN architectures ([paper](#))

- Serial processing of input pixels
- Demonstrated Street View House Numbers digit recognition in $\sim 5 \mu\text{s}$



Collaboration with Zenseact

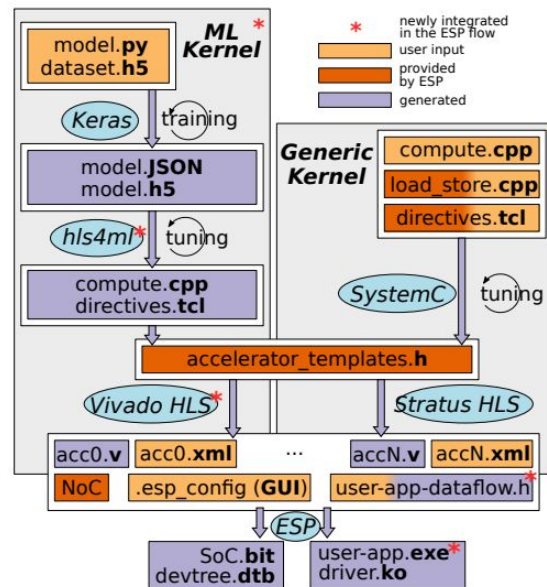
- Autonomous driving
- Semantic segmentation with ENet model in $\sim 4 \text{ ms}$



Projects integrating hls4ml

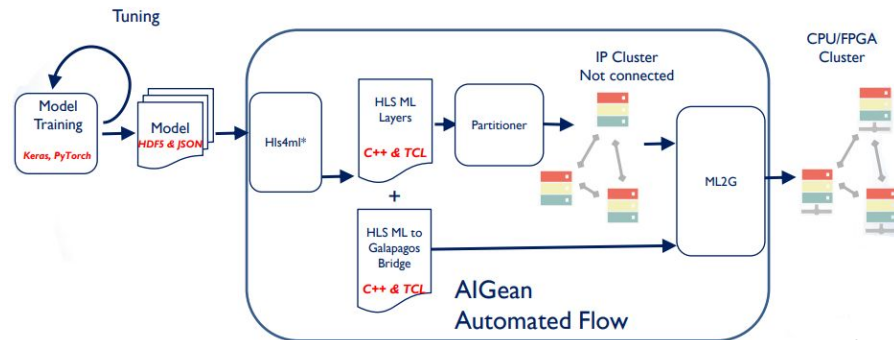
ESP4ML ([paper](#))

- A system-level design flow to build and program SoC architectures for embedded applications that require the hardware acceleration of machine learning algorithms



Algean ([paper](#))

- An open framework to build and deploy machine learning (ML) algorithms on a heterogeneous cluster of devices
- A unison of `hls4ml` and Galapagos, the framework for multi-FPGA deployment



Features in development (not exhaustive)

Graph NNs with PyTorch Geometric (PyG) - [PR by Elabd et al.](#)

- Two HLS implementations:
 - Throughput-optimized: Lower latency, greater resource-usage -> Smaller graphs
 - Resource-optimized: Greater latency, lower resource-usage -> Larger graphs
- Used for particle track reconstruction at CMS L1T

Recurrent NNs

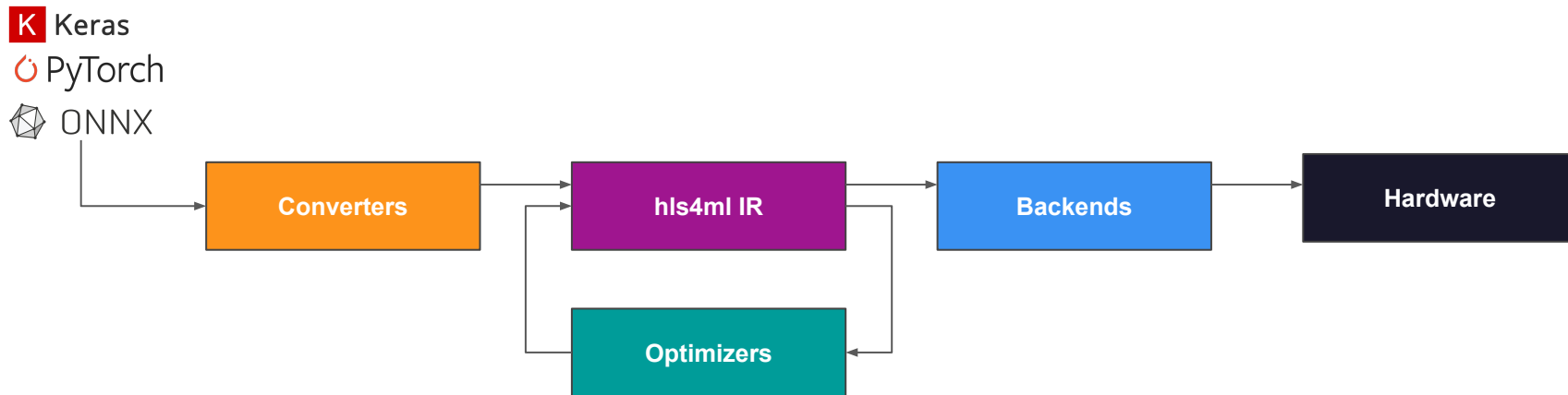
- LSTM and GRU - [PR](#)
- Low-latency LSTM implementation with balanced II - Applied in gravitational wave experiments ([paper](#))
- LSTM implementation for Intel/Altera FPGAs - Used for energy reconstruction in ATLAS LAr Calorimeter ([paper](#))

Online training with reconfigurable weights ([paper](#))

Recent changes

New modular architecture

- Overhaul of the internal representation (IR)
- Designed to be extensible and hardware-agnostic



New FPGA platforms - Intel/Altera

Initial support for **Quartus HLS** has been added to **hls4ml**

Currently only supporting DNN models

- Thanks to the flexibility of the new IR and the shared backend architecture, support for models quantized with QKeras comes automatically

Ongoing effort to bring features of Vivado backend

- Low latency CNNs
- Streaming implementations for all model architectures

Multi-FPGA-vendor / ASIC backend - Catapult

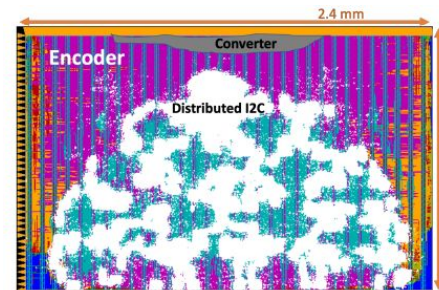
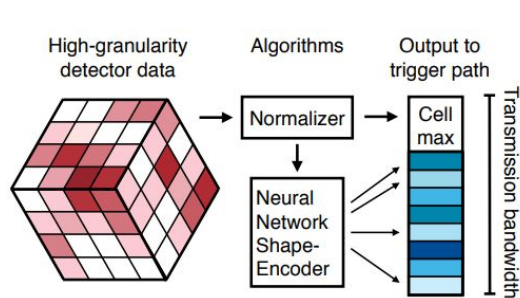
Intel and Xilinx HLS tools place restrictions on how their tools can be used

- Tied to the vendor's FPGA, cannot be used to create an ASIC

Supporting **Catapult HLS** from Siemens is underway

The early version of this backend was used for the development of [ECON-T ASIC](#)

- Data compression using an autoencoder
- Frontend chip compresses data to be sent to L1 Trigger



New feature development

Versatile IR can represent any neural network

- Conceptually simple, easy to use and debug
- Fully extensible with custom data types, precision and attributes

Backends are fully independent

- May target different hardware architectures
- May have multiple implementations for different use-cases

Fine-grained control over the translation process

- The flow of changes to IR is divided into stages for simpler handling

A stable platform enabling many new independent developments

Potential future directions

The new modular architecture of **hls4ml** allows us to position it as the research platform for the wider scientific community

Several potential directions:

- Support for deployment of novel model architectures
- Research of model compression techniques and efficient hardware implementations around them
- Deployment on specialized hardware platforms, beyond the Xilinx FPGAs
- Development of tools for interoperability with the existing ML/DL ecosystems
- ...

New model architectures - Transformers

Inference of RNN and LSTM models is an intrinsically sequential computation task

- Small amount of parallelism limits the size of the models and potential applications for extremely low latency applications

The Transformer architecture enables a high level of computation parallelism

- Suitable for massively parallel hardware like FPGAs

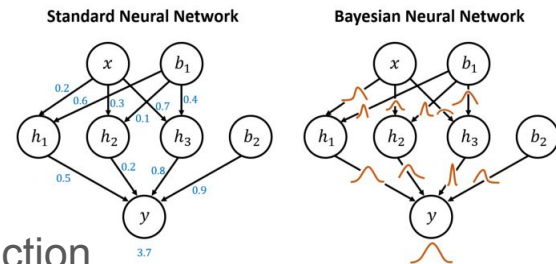
However, transformers can become very large!

- Implementations of transformers in FPGAs rely on heavy pruning ([paper](#))
- Sparse matrix multiplication is central to implementation of larger models
- Research into making transformers much smaller is ongoing

Due to the inherent parallelism, transformers may be split over multiple FPGAs with

Algean

Bayesian Neural Networks (BNNs)



BNNs are able to understand and express uncertainty in their prediction

- Allows us to build more robust models, e.g., for L1T ([paper](#))

BNNs can be constructed using DL researchers is TensorFlow Probability (TFP)

- Keras-like interface → easy to support in [hls4ml](#)

The most important missing piece in HLS code is the support for sampling from a distribution that is needed to generate the weights

- We can use the Linear Feedback Shift Register (LFSR)

Implementations of BNNs on FPGAs already exist

- BNN inference through Monte Carlo Dropout (MCD) ([paper](#))
- The uncertainty estimation and prediction is obtained by running the same input through the BNN multiple times and averaging the outputs.
- Several caching optimizations are proposed to decrease the amount of computation required making the approach very interesting for latency-oriented [hls4ml](#)

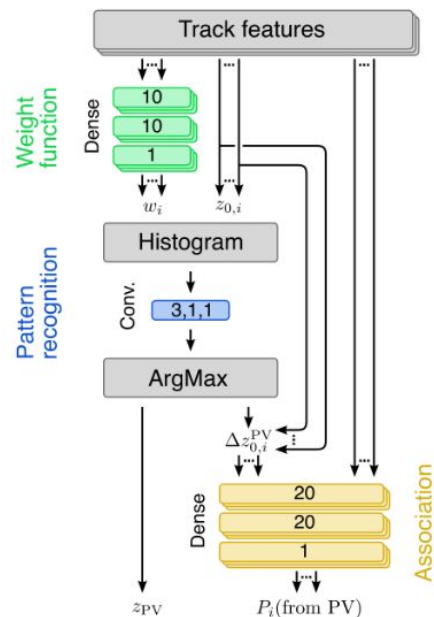
Enhancing support for CNNs

New implementation of convolution algorithm for small CNNs

- Low latency, $O(100\text{ns})$, instruction-based CNN implementation
- Used for primary vertex reconstruction for the upgrade of the CMS L1 trigger

Tiled implementation

- Process portions (*tiles*) of the input image in parallel
- Compromise between resource-intensive fully unrolled implementation and sequential streaming implementation
- Targeting $O(1\mu\text{s})$ latency



Credit: [Chris Brown et al.](#)

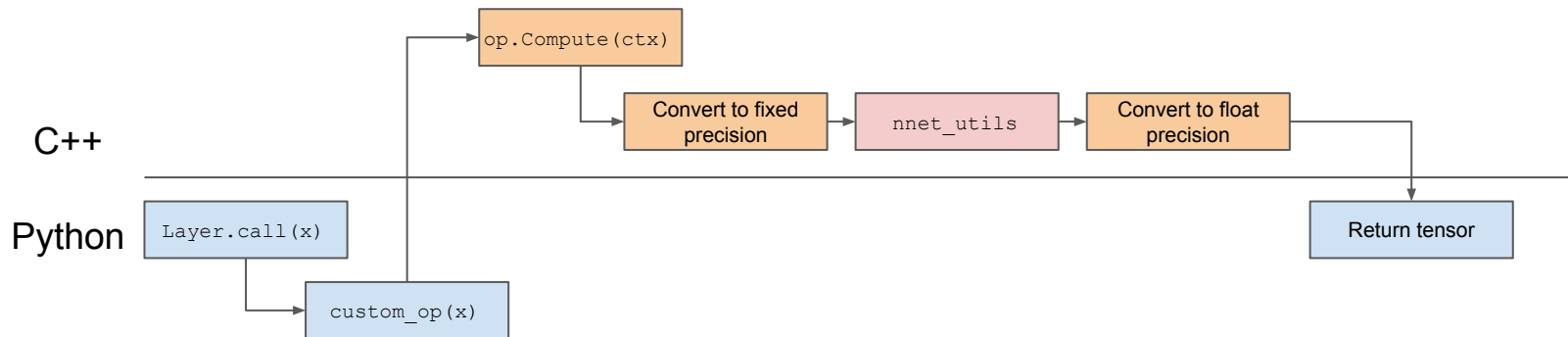
Training quantized models

hls4ml supports quantized models trained with QKeras, however getting good agreement between QKeras model and **hls4ml** model is not easy

- Main reason is the difference in implementation and data types used
- After applying optimizations like layer fusion it becomes difficult to track and debug divergence between the two models

A simpler alternative can be implemented using existing **hls4ml** routines

- Still rely on TF for backpropagation



Novel pruning techniques

Pruning is one of the two main techniques that allows us to deploy larger models

- Currently limited to fully unrolled designs

Central to the possibility of future exploration of pruning techniques is the ability to do **sparse matrix multiplications** in **hls4ml**

Leverage structured sparsity for optimal implementations

- Implementations relying on common *Compressed Sparse Row* (CSR) storage format suffer from large index overhead when the values of the matrix uses narrow bit-width data type (i.e., are quantized)
- Methods based on block pruning that leverage block-sparsity to define an efficient compressed format should be explored

Pattern pruning for CNNs

Unstructured pruning of CNN kernels can achieve higher sparsity than the structured pruning methods with the same accuracy

- Unstructured pruning methods suffer from bad hardware efficiency, especially for models that are also quantized
- Structured pruning allows for efficient implementations of matrix multiplications, but suffers due to lack of intra-kernel sparsity

The pattern pruning offers a compromise ([paper](#))

- A *pattern* is defined as the positions of nonzero values in a kernel
- Offers an opportunity to exploit intra-kernel sparsity much like unstructured pruning, while benefiting from the resource saving obtained from structured pruning
- By reducing the total number of patterns in a model, we can create low-latency designs focused on unrolled implementations of only these patterns
- Iterative process leveraging existing unstructured pruning methods (e.g., from TF MOT)

Open-source FPGA backend - **Bambu**

Most HLS tools are commercial and require a (costly) licence to use

- A case can be made for fully open-source translation flow

Bambu is a free, feature-rich and actively-developed HLS compiler that works with standard C/C++

Developers of Bambu have expressed [interest](#) in enhancing **hls4ml**:

- Enhancing portability of the generated C++/HLS code
- The use of custom floating point types

The expanded flexibility of the core of **hls4ml** allow for this effort to continue

New hardware platforms - Xilinx DPU

The Xilinx **Deep Learning Processing Unit** (DPU) is a programmable engine optimized for convolutional neural networks

- Introduced as part of the Vitis AI platform that contains tools to prune and quantize CNN models

DPU has several implementations for different precision and resource/throughput requirements

Vitis AI has several limitations

- Limited customization, no support for QKeras, pruning optimizer is commercial etc

Vitis AI uses the Xilinx Intermediate Representation (**XIR**) as the graph representation of the models, and possesses tools for executing these graphs on the DPUs

- A Python interface to bridge **hls4ml** IR to XIR is possible
- Supporting the Vitis platform will significantly broaden the scope of **hls4ml** and simplify deployment on Xilinx accelerators

New hardware architectures - CPU/GPU

The extensibility of the new IR shines again

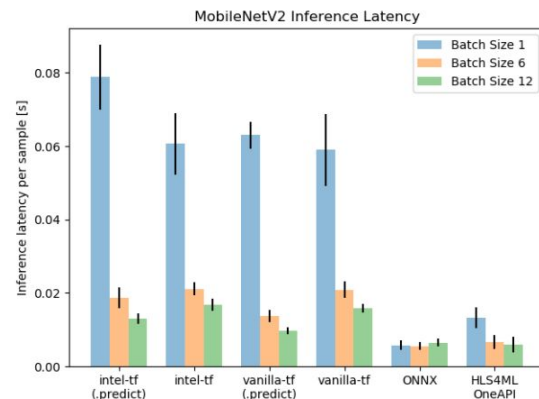
- Extension of type system to use floating point types
- Extension of NN tensors with a concept of external memory

oneAPI prototype

- Backend that emits DP C++ that uses oneDNN library

Alpaka

- An abstraction library for accelerator development
- Integrated in CMSSW, to be used for HLT development



Credit: [R. Abrahamse](#)



Summary

hls4ml - software package for translation of trained neural networks into synthesizable FPGA firmware

- Tunable resource usage latency/throughput
- Fast inference times, $O(10 \text{ ns})$ - $O(1 \text{ ms})$

Under heavy development on multiple directions

- Backends for new hardware platforms, compression tools, applications...

More information:

- Papers: <https://fastmachinelearning.org/#projects>
- Code: <https://github.com/fastmachinelearning/hls4ml>
- Tutorial: <https://github.com/fastmachinelearning/hls4ml-tutorial>

