Using SoC/FPGA
to Accelerate Machine Learning Algorithms

Romina Soledad Molina, Ph.D.

School on Applied AI for Sustainable Development | smr 4210 | Trieste, Italy
March 2026

Using SoC/FPGA
to Accelerate Machine Learning Algorithms

Outline

- Summary of the Key Concepts
- High-Level Synthesis for Machine Learning
- The hls4ml Workflow
- Deploying Deep Neural Networks on Embedded Architectures

Summary of the Key Concepts

Summary of the Key Concepts

Rise of Real-Time Machine Learning

Machine learning is increasingly deployed in latency-critical domains, such as High-Energy Physics (HEP), Robotics, Autonomous systems, Internet of Things (IoT)

Summary of the Key Concepts

Rise of Real-Time Machine Learning

Machine learning is increasingly deployed in latency-critical domains, such as High-Energy Physics (HEP), Robotics, Autonomous systems, Internet of Things (IoT)

These applications demand fast and efficient inference, with strict requirements on:

Low latency, low power consumption.

Summary of the Key Concepts

Rise of Real-Time Machine Learning
Machine learning is increasingly deployed in latency-critical domains, such as High-Energy Physics (HEP), Robotics, Autonomous systems, Internet of Things (IoT)

These applications demand fast and efficient inference, with strict requirements on:
Low latency, low power consumption.

Why FPGAs?
Highly parallel and reconfigurable architectures
Well suited for deterministic low-latency execution
Energy-efficient compared to general-purpose processors

Summary of the Key Concepts

Rise of Real-Time Machine Learning
Machine learning is increasingly deployed in latency-critical domains, such as High-Energy Physics (HEP), Robotics, Autonomous systems, Internet of Things (IoT)

These applications demand fast and efficient inference, with strict requirements on:
Low latency, low power consumption.

Why FPGAs?
Highly parallel and reconfigurable architectures
Well suited for deterministic low-latency execution
Energy-efficient compared to general-purpose processors

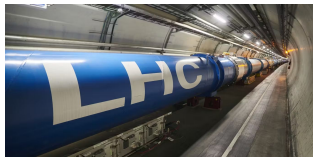
The Challenge
ML models are software-native by design
Mapping them efficiently to hardware is: complex, time-consuming, often manual
There is a strong need for: automation, abstraction, hardware-aware ML workflows

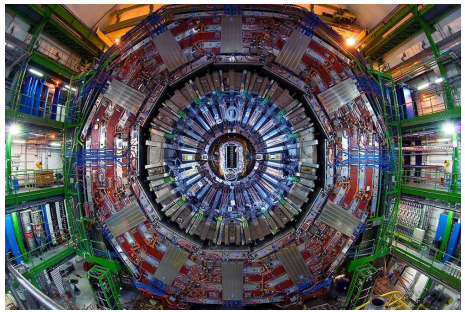
High-Level Synthesis for Machine Learning

High-Level Synthesis for Machine Learning

hls4ml

"**The inspiration for the creation of the hls4ml** package stems from the **high energy physics community at the CERN Large Hadron Collider (LHC)**. While machine learning has already been proven to be extremely useful in analysis of data from detectors at the LHC, it is typically performed in an "offline" environment after the data is taken and agglomerated." [hls4ml]

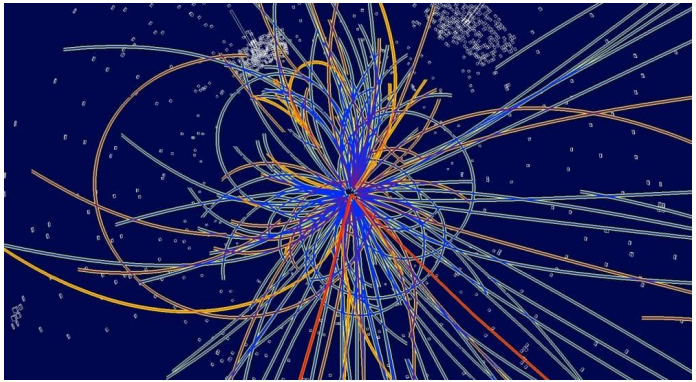




[hls4ml] https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

"However, **one of the largest problems at detectors on the LHC is that collisions, or "events", generate too much data for everything to be saved**. As such, filters called "triggers" are used to determine whether a given event should be kept. Using **FPGAs allows for significantly lower latency so machine learning algorithms can essentially be run "live" at the detector level for event selection**. As a result, more events with potential signs of new physics can be preserved for analysis." [hls4ml]



[hls4ml] https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

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]

[a] *Fermi National Accelerator Laboratory, Batavia, IL 60510, U.S.A.*
[b] *Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.*
[c] *HawkEye360, Herndon, VA 20170, U.S.A.*
[d] *CERN, CH-1211 Geneva 23, Switzerland*
[e] *University 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 acquisition systems have extremely low, sub-microsecond latency requirements that are unique to particle physics. We present a case study for neural network inference in FPGAs focusing on a classifier for jet substructure which would enable, among many other physics scenarios, searches for new dark sector particles and novel measurements of the Higgs boson. While we focus on a specific example, the lessons are far-reaching. A companion compiler package for this work is developed based on High-Level Synthesis (HLS) called `hls4ml` to build machine learning models in FPGAs. The use of HLS increases accessibility across a broad user community and allows for a drastic

Duarte, J., Han, S., Harris, P., Jindariani, S., Kreinar, E., Kreis, B., ... & Wu, Z. (2018). Fast inference of deep neural networks in FPGAs for particle physics. *Journal of instrumentation*, *13*(07), P07027.

High-Level Synthesis for Machine Learning

- Binary & Ternary neural networks: [2020 Mach. Learn.: Sci. Technol]
  - Compressed weights for low resource inference
- Boosted Decision Trees: [JINST 15 P05026 (2020)]
  - Low latency for Decision Tree ensembles
- GarNet / GravNet: [arXiv: 2008.03601]
  - Distance weighted graph neural networks suitable for sparse/irregular point-cloud data
- Quantization aware training QKeras + support in hls4ml: [arXiv: 2006.10159]
- Convolutional neural networks: Mach. Learn.: Sci. Technol. 2 045015 (2021)

High-Level Synthesis for Machine Learning

- Python package.
- It enables the transformation of neural network models into firmware deployable on FPGAs for efficient inference.
  - https://fastmachinelearning.org/hls4ml/
  - https://github.com/hls-fpga-machine-learning/hls4ml
  - pip install hls4ml
- Extremely configurable: precision, resource vs latency/throughput tradeoff.
- Easy to use.

Duarte, J., Han, S., Harris, P., Jindariani, S., Kreinar, E., Kreis, B., ... & Wu, Z. (2018). Fast inference of deep neural networks in FPGAs for particle physics. *Journal of instrumentation*, *13*(07), P07027.

High-Level Synthesis for Machine Learning

ML framework support:

- (Q)Keras
- PyTorch
- (Q)ONNX

Neural networks architectures:

- Fully Connected NN
- Convolutional NN
- Recurrent NN
- Graph NN

HLS backends:

- Vivado HLS
- Intel HLS
- Vitis HLS
- Catapult HLS
- oneAPI (experimental)

High-Level Synthesis for Machine Learning

hls4ml is tested on the following platforms:

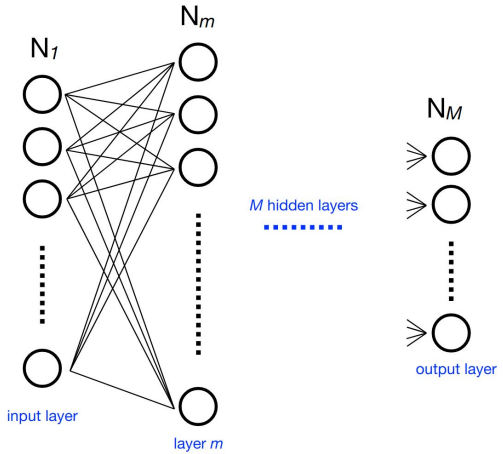
Vivado HLS versions 2018.2 to 2020.1
Intel HLS versions 20.1 to 21.4. Versions > 21.4 have not been tested.
Vitis HLS versions 2022.2 to 2024.1. Versions <= 2022.1 are known not to work.
Catapult HLS versions 2024.1_1 to 2024.2
oneAPI versions 2024.1 to 2025.0

High-Level Synthesis for Machine Learning

How does it work?

With hls4ml, each layer of output values is calculated independently in sequence, using pipelining to speed up the process by accepting new inputs after an initiation interval. The activations, if nontrivial, are precomputed.

Simplifying the input network must be done before using hls4ml to generate HLS code, for optimal compression to provide a sizable speedup.

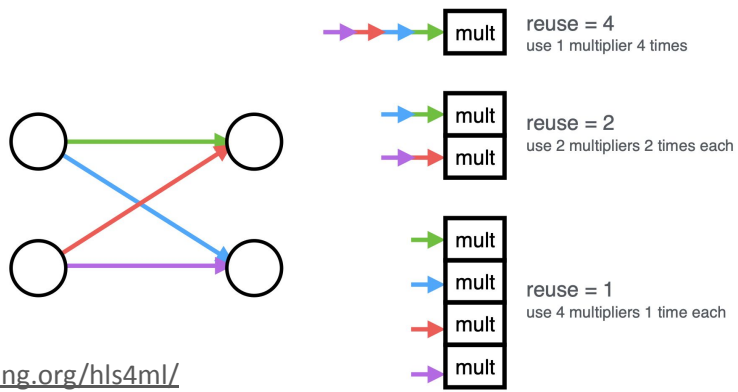


[hls4ml] https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

Trade-off between latency and FPGA resource usage determined by the parallelization of the calculations in each layer.

Reuse factor: number of times a multiplier is used to do a computation.



Fewer resources,
Lower throughput,
Higher latency

More resources,
Higher throughput,
Lower latency

https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

I/O types: supports multiple styles for handling data transfer to/from the network and between layers.

io_parallel

io_stream

https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

I/O types: supports multiple styles for handling data transfer to/from the network and between layers.

io_parallel

Data is passed in **parallel** between the layers.

This style allows for **maximum parallelism** and is well suited for MLP networks and small CNNs which aim for lowest latency.

https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

I/O types: supports multiple styles for handling data transfer to/from the network and between layers.

io_stream

Data is passed **one "pixel" at a time**.

Each pixel is an **array of channels**, which are always sent in parallel. This method for sending data between layers is recommended for larger CNN and RNN networks.

With this IO type, each layer is connected with the subsequent layer through **first-in first-out (FIFO) buffers.**

The implementation of the FIFO buffers contribute to the overall resource utilization of the design, impacting in particular the **BRAM or LUT utilization**.

https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

Strategy: the implementation of core matrix-vector multiplication routine, which can be **latency-oriented, resource-saving oriented, or specialized**.

Different strategies will have an **impact on overall latency and resource consumption** of each layer and users are advised to choose based on their design goals.

If one layer would have >**4096 elements**, we should set **['Strategy'] = 'Resource'** for that layer, or **increase the reuse factor** by hand.

4096 elements is related to the maximum size of an array to be partitioned. This value might change, it should be checked with the hardware synthesis tool.

https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

Activations - Implementation parameter

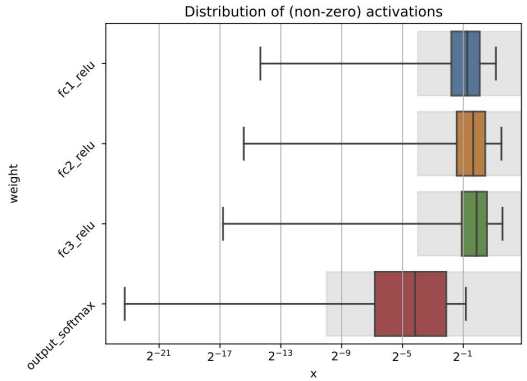
- **latency:** Good latency, high resource usage. **It does not work well if there are many output classes**.
- **stable:** Slower but with better accuracy, useful in scenarios where **higher accuracy is needed**.
- **legacy:** An older implementation with poor accuracy, but good performance. Usually the latency implementation is preferred.
- **argmax:** If you don't care about normalized outputs and only care about which one has the highest value, using argmax saves a lot of resources. This sets the highest value to 1, the others to 0.

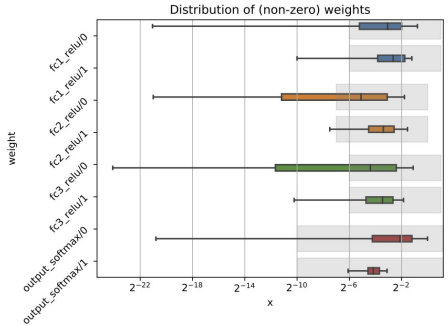
https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

Profiling

- The tools in **hls4ml.model.profiling** can help to choose the right precision for the model.
- **hls4ml.model.profiling.numerical** method with three objects: **a Keras model object**, **test data**, and an **HLSModel.**





https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for Machine Learning

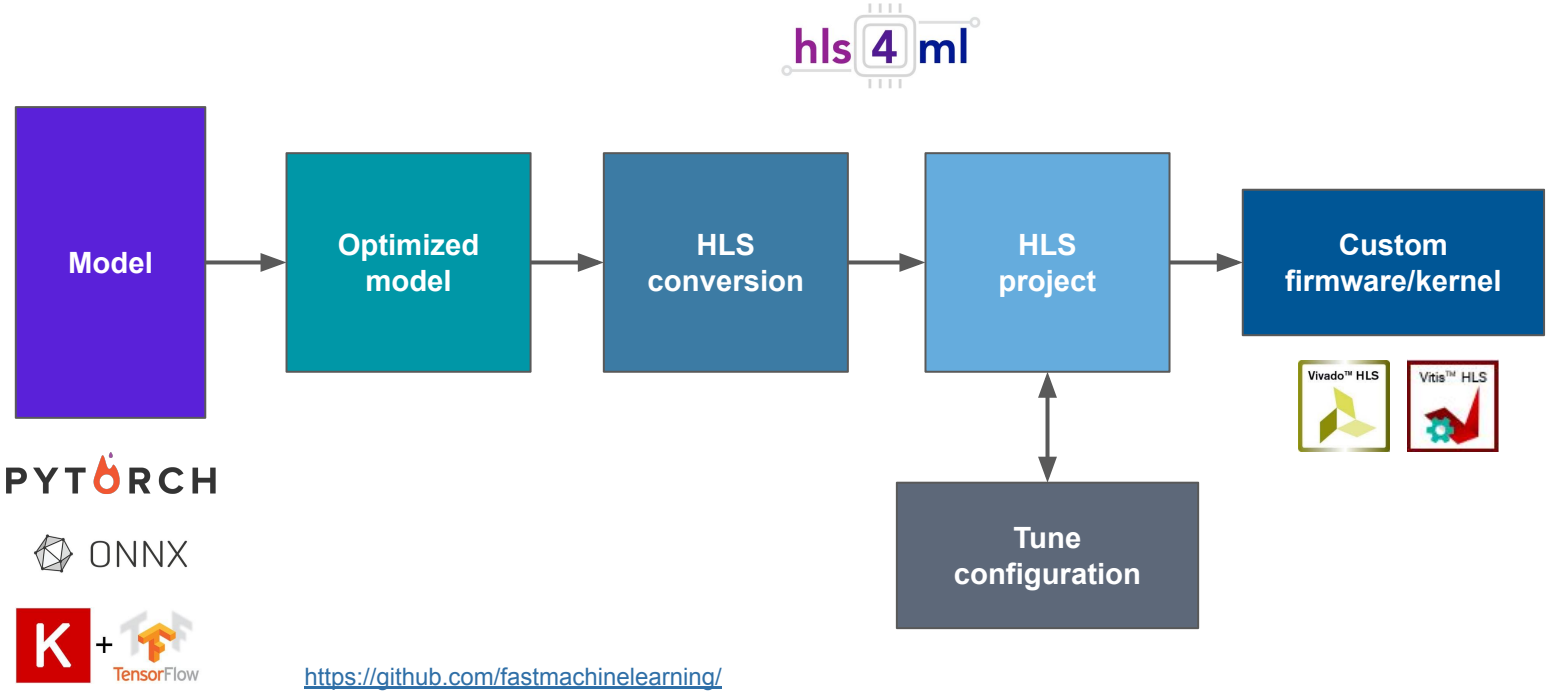
Trace

- When we start using **customised precision** throughout the model, it can be useful to collect the output from each layer. We enable this trace collection by setting **Trace = True** for each layer whose output we want to collect.

```
for layer in config['LayerName'].keys():
    config['LayerName'][layer]['Trace'] = True
```


The hls4ml
Workflow

The hls4ml Workflow



https://github.com/fastmachinelearning/

Using SoC/FPGA
to Accelerate Machine Learning Algorithms

Romina Soledad Molina, Ph.D.

School on Applied AI for Sustainable Development | smr 4210 | Trieste, Italy
March 2026