hls4ml enabling real-time deep learning in particle physics

Jennifer Ngadiuba (Fermilab) on behalf of the hls4ml team

CPAD Instrumentation Frontier Workshop
Stony Brook University, March 18-22, 2021
hls4ml @ the LHC

- **hls4ml** is a library for automatic translation of deep learning models to FPGA firmware for inference with ultra low latency

- **First target applications:**
  hardware trigger of LHC experiments and detector front-end electronics

A 2-tier event filter reduces data rates by ~4 orders of magnitude
**hls4ml @ the LHC**

- **hls4ml** is a library for automatic translation of deep learning models to FPGA firmware for inference with ultra low latency.

- **First target applications:**
  hardware trigger of LHC experiments and detector front-end electronics.

---

**J. Krupa’s talk:** accelerate DL using co-processors (GPUs or FPGAs)

---
**hls4ml** @ the LHC

- **hls4ml** is a library for automatic translation of deep learning models to FPGA firmware for inference with ultra low latency

- **First target applications:**
  - hardware trigger of LHC experiments and detector front-end electronics

**THIS TALK!** Limited resources and strict latency constraints
Bring DL to FPGA for L1 trigger with high level synthesis for machine learning

2.1 hls4ml

Our basic task is to translate a trained neural network by taking a model architecture, weights, and biases and implementing them in HLS in an automated fashion. This automated procedure is the task of the software/firmware package, hls4ml. A schematic of a typical workflow is illustrated in Fig. 1.

The part of the workflow that is illustrated in red indicates the usual software workflow required to design a neural network for a specific task. This usual machine learning workflow, with tools such as Keras and PyTorch, involves a training step and possible compression steps (more discussion below in Sec. 2.3) before settling on a final model. The blue section of the workflow is the task of hls4ml which translates a model into an HLS project that produces a firmware block. This automated tool has a number of configurable parameters which can help the user customize the network translation for their application.

The time to perform the hls4ml translation is much shorter (minutes to hours) than a custom design of a neural network and can be used to rapidly prototype machine learning algorithms without dedicated engineering support. For physicists, this makes designing physics algorithms for the trigger or DAQ significantly more accessible and efficient, thus allowing the "time to physics" to be greatly reduced.

https://fastmachinelearning.org/hls4ml/
Bring DL to FPGA for L1 trigger with high level synthesis for machine learning

- User-friendly automated tool
- Easy to tune the inference performance for your specific application: precision, resource vs latency/throughput tradeoff
- Can be used as API
- Includes several debugging utilities
- Most common DL layers and activation functions supported
**hls4ml**: recent developments

- Since CPAD19 but the library has been significantly expanded!

• Quantization-aware training and pruning
  - Google QKeras [arxiv.2006.10159] **THIS TALK!**
  - PyTorch Brevitas [arxiv.2102.11289] **coming soon**

• Convolutional neural networks [arxiv.2101.05108]

• Custom architectures as graph neural networks:
  - Interaction networks for tracking [arxiv.2012.01563] **THIS TALK!**

• Workflow for DL-dedicated ASICs [arxiv.2103.05579] **J. Hirschauer talk**

• Support for other vendors: Intel and Mentor HLS **coming soon**
Neural network inference on FPGA

Neural network inference = matrix multiplication

Efficient implementation on FPGA uses DIGITAL SIGNAL PROCESSORS

There are about 5–10k DSPs in modern FPGAs!

\[
\begin{bmatrix}
  w_{11} & w_{21} \\
  w_{12} & w_{22} \\
  w_{13} & w_{23}
\end{bmatrix}
\begin{bmatrix}
  i_1 \\
  i_2
\end{bmatrix}
=
\begin{bmatrix}
  (w_{11} \times i_1) + (w_{21} \times i_2) \\
  (w_{12} \times i_1) + (w_{22} \times i_2) \\
  (w_{13} \times i_1) + (w_{23} \times i_2)
\end{bmatrix}
\]

• DSPs are the most precious resource when mapping a NN into FPGA!
• Usage can be controlled in hls4ml by tuning how much to parallelize
  → this affects the latency and it’s a trade off that depends on the application
Efficient NN design: quantization

- Post-training quantization on FPGA allows for large area reduction but severe model performance drop for too few bits.

Figure 11: DSP usage in the pruned 3-layer model as a function of the network precision. The various curves illustrate resource usage for different resource usage factors.

Figure 12: FF and LUT usage in the pruned 3-layer model as a function of the network precision. The various curves illustrate resource usage for different resource usage factors.

The corresponding to the four layers of neuron values that must be computed, with each increment in reuse factor. This is in line with expectations from Eq. 2.4 where additional reuse of multipliers in a given layer calculation incurs added latency. In the right plot of Fig. 13, the initiation interval is shown for different reuse factors. By design, the initiation interval and the reuse factor match as a new input can be introduced to the algorithm only when all multiplications for a given DSP multiplier are completed.

At very low network precision, the HLS synthesis initiation interval is smaller than the reuse factor. This is because multiplications are no longer implemented in DSPs but through FFs and LUTs.

Severe performance drop below 12 bits.

ap_fixed<width,integer>

0101.10111010101

integer  fractional  width
Efficient NN design: compression

- Neural Network compression is a widespread technique to reduce the size, energy consumption, and overtraining of deep neural networks.


**Figure:**

- Fully parallelized (max DSP use)
- Number of DSPs available
- Fixed-point precision
- Compression

**Diagram:**

- Before pruning
- After pruning
- Pruning synapses
- Pruning neurons

**Example:**

tensorflow sparsity toolkit

iteratively remove low magnitude weights, starting with 0 sparsity, smoothly increasing up to the set target as training proceeds.
Efficient NN design with QKeras

- QKeras is a library developed and maintained by Google to train models with quantization in the training.
  
  - Can achieve good performance with very few bits.
  
  - We’ve recently added support for QKeras-trained models to hls4ml [arxiv.2006.10159]

  - the number of bits used in training is also used in inference.

  - automatic heterogenous layer-by-layer quantization also possible.
Fast convolutional neural networks

- Brand new implementation based on streaming `hls::stream<T>`
  - collect data from input pixels until we can compute one output (FIFOs)
  - compute the value of output pixel with a single call to matrix-vector multiplication
  - can reuse existing matrix-vector multiplication used for fully connected layers
Fast convolutional neural networks

Evaluate performance on street-view house numbers dataset (32x32x3)

heterogeneously quantized model through bayesian optimization

arxiv.2101.05108
Fast convolutional neural networks

Max parallelization, i.e. reuse factor = 1

no accuracy loss down to 4 bits for Q/QP models

arxiv.2101.05108
Fast convolutional neural networks

Max parallelization

~ 5 μs inference time!

Latency (clock cycles)

Bit width

Latency (μs)

arxiv.2101.05108
hls4ml for triggering @ 40 MHz

• hls4ml enabled developments of new trigger algorithms with large gain for physics!
  - replace standard cut-based algorithms

CMS Phase-2 L1 trigger upgrade TDR

### NN VBF H→bb

<table>
<thead>
<tr>
<th>Usage</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>24 clk @ 200MHz</td>
</tr>
<tr>
<td>II</td>
<td>5</td>
</tr>
<tr>
<td>DSP48E</td>
<td>484</td>
</tr>
<tr>
<td>FF</td>
<td>32634</td>
</tr>
<tr>
<td>LUT</td>
<td>62358</td>
</tr>
</tbody>
</table>
hls4ml for triggering @ 40 MHz

• hls4ml enabled developments of new trigger algorithms with large gain for physics!
  - replace standard cut-based algorithms
  - improve physics objects reconstruction (muons, taus, jets)

36 INPUT FEATURES:
  \( \phi, \theta \) of track segments in muon stations
  track segment quality
  track segment curvature

3 HIDDEN LAYERS (30x25x20)

1 OUTPUT: muon \( p_T \)

CMS Phase-2 L1 trigger upgrade TDR

x2.5 rate reduction
**hls4ml for triggering @ 40 MHz**

- hls4ml enabled developments of new trigger algorithms with large gain for physics!
  - replace standard cut-based algorithms
  - improve physics objects reconstruction (muons, taus, jets)
  - develop new strategies like anomaly detection with autoencoders for signal-agnostic triggering

21 inputs: $p_T/\eta/\Phi$ of 4 $e/\gamma$, 4 $\mu$, 10 jets, and MET
→ input 19x3 input image

300 ns latency
30% DSPs
10% FFs
30% LUTs

with no pruning/quantization!

K. Govorkova @ Fast Machine Learning workshop 20
hls4ml for triggering @ 40 MHz

- hls4ml enabled developments of new trigger algorithms with large gain for physics!
  - replace standard cut-based algorithms
  - improve physics objects reconstruction (muons, taus, jets)
  - develop new strategies like anomaly detection with autoencoders for signal-agnostic triggering

- Allows also for integration of custom architectures like graph NNs to achieve ultra-low inference latency
  - calorimeter clusters classification [CTD 2020]
  - charged particles track reconstruction [NeurIPS 2020]
Summary

• hls4ml enables automatic translation of modern deep learning architectures to synthesizable FPGA firmware and more
  - today presented most recent developments

• Presented applications for the hardware trigger at LHC experiments but many others ongoing beyond LHC
  - eg, accelerator controls → see C. Herwig talk
  - other cases being identified with common challenges (eg., large scale LArTPC experiments or gravitational waves detection)

• The library is also expanding beyond FPGAs
  - see J. Hirschauer talk on the application of hls4ml to achieve DL-dedicated ASICs design for CMS high-granularity calorimeter (and our recent paper arxiv.2103.05579)

• Very active developers team... stay tuned for new features and applications!
high level synthesis for machine learning

For more info:

https://fastmachinelearning.org/hls4ml/

Fast inference of deep neural networks in FPGAs for particle physics [JINST 13 P07027 (2018)]
Fast inference of Boosted Decision Trees in FPGAs for particle physics [JINST 15 P05026 (2020)]
Compressing deep neural networks on FPGAs to binary and ternary precision with HLS4ML [2020 Mach. Learn.: Sci. Technol]
Automatic deep heterogeneous quantization of Deep Neural Networks for ultra low-area, low-latency inference on the edge at particle colliders [arxiv.2006.10159]
Distance-Weighted Graph Neural Networks on FPGAs for Real-Time Particle Reconstruction in High Energy Physics [arxiv.2008.03601]
Fast convolutional neural networks on FPGAs with hls4ml [arxiv.2101.05108]
Accelerated Charged Particle Tracking with Graph Neural Networks on FPGAs [arxiv.2012.01563]

Thank you!