Real-time AI at the edge of particle physics detectors

Jennifer Ngadiuba (Fermilab)

51st SLAC Summer Institute
August 7-18, 2023
Stanford National Accelerator Laboratory
Exponential growth of network size to make models more capable!
GPT-3: 175 billion parameters (0.16% of the human brain)

Made possible by parallel evolution of hardware: Graphical Processing Units (GPUs)

GPT-3 training: 36 years on eight V100 GPUs, or seven months with 512 V100 GPUs !!! [Source]
O(TB) memory needed → hardly fit on your phone!
Not really the best example for the topic of this lecture but gives a sense of the scale of AI today!
Cloud vs Edge Computing

AI is commonly used by your smartphone (eg., face and speech recognition)

Typically the AI is too complex to be run on the phone hardware

Your data are sent to a data center in the cloud through internet where AI is executed and answer is sent back to the phone
Cloud vs Edge Computing

Your data are sent to a data center in the cloud through internet where AI is executed and answer is sent back to the phone.

**CHALLENGE:** as the density of phones increases it is difficult for networks to keep up

→ **bandwidth and latency issues**

**Latency** = how much time can you wait before Alexa answers you?

→ $O(1)$ s! (depending on your patience)
Cloud vs Edge Computing

Your data are sent to a data center in the cloud through the internet where AI is executed and the answer is sent back to the phone.

Cloud

Latency = how much time there is before causing an accident?

→ O(10) ms, i.e. x10 shorter than human reaction time!

One more example: **self-driving cars**

Latency constraint here is a major challenge for cloud computing!
Cloud vs Edge Computing

SOLUTION:
execute the AI directly on device (aka at the edge)

Not without major challenges:
AI execution must be efficient

efficiency = best accuracy at lowest cost
cost = power, latency/throughput, and area
The need for specialized hardware

The need for efficient AI at the edge triggered a proliferation of AI-specialized chips

https://cloud.google.com/tpu/

Efficient hardware not enough…

Innovative training methods able to maintain accuracy at low computational complexity

SqueezeLLM: Dense-and-Sparse Quantization

Sehoon Kim*
selhoonkim@berkeley.edu
UC Berkeley
Zhen Dong
zhendong@berkeley.edu
UC Berkeley
Coleman Hooper*
chooper@berkeley.edu
UC Berkeley
Xiuyu Li
xiuyu@berkeley.edu
UC Berkeley
Amir Gholami†
amirgh@berkeley.edu
ICSI, UC Berkeley
Sheng Shen
sheng.s@berkeley.edu
UC Berkeley
Michael W. Mahoney
mmahoney@stat.berkeley.edu
ICSI, LBNL, UC Berkeley
Kurt Keutzer
keutzer@berkeley.edu
UC Berkeley

FPGA Resource-aware Structured Pruning for Real-Time Neural Networks

Benjamin Ramhorst and George A. Constantinides
Department of Electrical and Electronic Engineering
Imperial College London
London, United Kingdom
{benjamin.ramhorst19, g.constantinides}@imperial.ac.uk

Vladimir Lončar*
Laboratory for Nuclear Science
Massachusetts Institute of Technology
Cambridge, United States of America
vloncar@mit.edu

Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors

Claudionor N. Coelho Jr.
Palo Alto Networks (California, USA)
Aki Kuusela, Shan Li, and Hao Zhuang
Google LLC (California, USA)
Thea Aarrestad† Vladimir Lončar† Maurizio Pierini, Adrian Alan Pol, and Sioni Summers
European Organization for Nuclear Research (CERN) (Geneva, Switzerland)
Jennifer Ngadiuba
California Institute of Technology (Caltech) (California, USA)
(Dated: June 22, 2021)

TinyBERT: Distilling BERT for Natural Language Understanding

Xiaoqi Jiao1,* Yichun Yin2,* Lifeng Shang2,† Xin Jiang2
Xiao Chen2, Linlin Li3, Fang Wang4 and Qun Liu2
1Key Laboratory of Information Storage System, Huazhong University of Science and Technology, Wuhan National Laboratory for Optoelectronics
2Huawei Noah’s Ark Lab
3Huawei Technologies Co., Ltd.
{jiaoxiaoqi, wangfang}@hust.edu.cn
{yinyichun, shang.lifeng, jiang.xin}@huawei.com
{chen.xiao2, lynn.lilinlin, qun.liu}@huawei.com
Big Science in 21st century

Probing the fundamental structure of nature requires complex experimental devices, large infrastructures and big collaborations.

- The Large Hadron Collider
- LIGO/VIRGO interferometers
- Vera C. Rubin Observatory
- CMB-S4
- The DUNE neutrino experiment
Big Science = Big Data

• Requirements for ML in particle physics go far beyond industrial and commercial applications because of extreme environments:
  - speed, throughput, fidelity, interpretability, and reliability

• At the extreme edge of throughput requirements HEP experiments need efficient real time ML able to meet the most challenging latency constraint!

https://a3d3.ai/
Big Data @ the Energy Frontier

The Large Hadron Collider (LHC)

Collision frequency: 40 MHz
Particles per collision: $O(10^3)$
Detector resolution: $O(10^8)$ channels

Extreme data rates of $\sim$PB/s!
Collisions which produce interesting products (ex: Higgs boson) are typically very rare

The probability of producing a Higgs boson is 5-9 orders of magnitude smaller than producing only jets.

Standard Model Production Cross Section Measurements

Becoming more and more rare →

How often it is produced →

ATLAS Preliminary
Run 1, 2 $\sqrt{s} = 7, 8, 13$ TeV

Status: March 2017
Data reduction workflow @ LHC

- **Level-1 Trigger (hardware)**
  - 99.75% rejected
  - Decision in ~4 $\mu$s

- **High-Level Trigger (software)**
  - 99% rejected
  - Decision in ~100s ms

- After trigger, 99.99975% of events are gone forever
Data reduction workflow @ LHC

99.75% events rejected!

Hardware based
Runs on FPGAs in real time
O(μs) latency

100 KHz 1 MHz 1 MB/evt

L1 trigger 1 kHz
High-Level trigger

1 kHz
Offline reconstruction

1 KHz
3-30 kB/evt

Data volume

Computing time

Information

High-level analysis
Data reduction workflow @ LHC

99.75% events rejected!
Hardware based
Runs on FPGAs in real time
O(μs) latency

99% events rejected!
Software based
Runs on CPUs in real time
O(100 ms) latency

L1 trigger
100 KHz
High-Level trigger
1 KHz
1 MB/evt
Offline reconstruction
1 KHz
3-30 kB/evt

No data stored before this line

Data volume
Computing time
High-level analysis
Information

Data reduction work @ LHC
Data reduction workflow @ LHC

99.75% events rejected!
Hardware based
Runs on FPGAs in real time
O(μs) latency

99% events rejected!
Software based
Runs on CPUs in real time
O(100 ms) latency

Software based
Run on data centres (LHC grid)
O(s) computing time

Data volume
L1 trigger
100 KHz

High-Level trigger
1 MHz
1 MB/evt

Offline reconstruction
1 MHz
3-30 kB/evt

Information
High-level analysis
Computing time

99.75% events rejected!
Hardware based
Runs on FPGAs in real time
O(μs) latency

99% events rejected!
Software based
Runs on CPUs in real time
O(100 ms) latency

Software based
Run on data centres (LHC grid)
O(s) computing time

Data volume
L1 trigger
100 KHz

High-Level trigger
1 MHz
1 MB/evt

Offline reconstruction
1 MHz
3-30 kB/evt

Information
High-level analysis
Computing time
Data reduction workflow @ LHC

99.75% events rejected!
Hardware based
Runs on FPGAs in real time
O(μs) latency

100 KHz
L1 trigger

40 MHz

99% events rejected!
Software based
Runs on CPUs in real time
O(100 ms) latency

1 KHz
High-Level trigger

1 MB/evt

1 KHz
Offline reconstruction

3-30 kB/evt

No data stored before this line

Software based
Run on data centres (LHC grid)
O(s) computing time

Run by individuals on the grid

13
Make physics discoveries with 0.0025% of the events!
(the rest is lost...)
Let’s zoom in!

99.75% events rejected!
Hardware based
Runs on FPGAs in real time
O(μs) latency

100 KHz
L1 trigger
Data volume

99% events rejected!
Software based
Runs on CPUs in real time
O(100 ms) latency

1 kHz
High-Level trigger

1 MB/evt
 Offline reconstruction

1 kHz
3-30 kB/evt
No data stored before this line

1 kHz
Run by individuals on the grid

Software based
Run on data centres (LHC grid)
O(s) computing time

Computing time
Information
High-level analysis

40 MHz
Take away message

• We must reduce data rates to manageable levels for offline processing and storage by filtering collision events $\rightarrow$ triggering
  - not all collisions are interesting (ex, about 1 collision per trillion will produce a Higgs boson)
• We must pick the good ones $\rightarrow$ use deep learning for highest accuracy
• We must do it as fast as one microsecond $\rightarrow$ deploy FPGAs
• Chip area (aka resources) limited as also needed by other physics tasks

the trigger system
O(100) FPGAs
in: 40 MHz
out: 100 KHz

detector front end electronics
detector data
optical links
How to fit DL model here?

- Detector front end electronics
- Detector data
- Optical links
- The trigger system
  - O(100) FPGAs
  - In: 40 MHz
  - Out: 100 KHz
What are FPGAs?

**Field Programmable Gate Arrays**
are reprogrammable integrated circuits

Contain many different building blocks
(‘resources’) which are connected together as you desire
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

Look Up Tables (LUTs) perform arbitrary functions on small bitwidth inputs (2-6 bits)
→ used for boolean operations, arithmetics, memory

Flip-flops register data in time with the clock pulse
Field Programmable Gate Arrays are reprogrammable integrated circuits.

DSPs are specialized units for multiplication and arithmetic → faster and more efficient than LUTs for these type of operations → for deep learning, they are often the most precious resource.

Also contain embedded components:

- **Digital Signal Processors (DSPs):** logic units used for multiplications.
Machine learning algorithms are ubiquitous in HEP.

FPGA usage is broad across HEP experiments, centered on DAQ and trigger development.

Some early adaptations of ML techniques in trigger [1].

FPGA development is becoming more accessible.

High-Level Synthesis, OpenCL,

FPGA interest in industry is growing.

Programmable hardware with structures that map nicely onto ML architectures.

What are FPGAs?

Field Programmable Gate Arrays are reprogrammable integrated circuits.

BRAMs are small, fast memories (ex, 18 Kb each)

→ more efficient than LUTs when large memory is required.

Modern FPGAs have ~100 Mb of BRAMs, chained together as needed.

FPGA diagram

Also contain embedded components:

Digital Signal Processors (DSPs): logic units used for multiplications.

Random-access memories (RAMs): embedded memory elements.
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

Contain array of logic cells embedded with DSPs, BRAMs, etc.

Support highly parallel algorithm implementation

Low power per Op (relative to CPU/GPU)

Also contain embedded components:

- Digital Signal Processors (DSPs): logic units used for multiplications
- Random-access memories (RAMs): embedded memory elements
Why are FPGAs fast?

• Fine-grained / resource parallelism
  - use the many resources to work on different parts of the problem simultaneously
  - allows us to achieve low latency

• Most problems have at least some sequential aspect, limiting how low latency we can go
  - but we can still take advantage of it with…

• Pipeline parallelism
  - instruct the FPGA to work on different data simultaneously
  - allows us to achieve high throughput
How are FPGAs programmed?

**Hardware Description Languages**

HDLs are programming languages which describe electronic circuits.

**High Level Synthesis**

generate HDL from more common C/C++ code pre-processor directives and constraints used to optimize the timing

*drastic decrease in firmware development time!*

See Xilinx Vivado HLS, Intel HLS, Catapult HLS
How to fit DL model here?

the trigger system
O(100) FPGAs

in: 40 MHz
out: 100 KHz

detector front end electronics

detector data
optical links
Efficient ML design for FPGAs

FPGAs provide huge flexibility

*Performance depends on how well you take advantage of this*

In this course you will learn how to boost ML inference efficiency through:

- **compression**: reduce number of synapses or neurons

- **quantization**: reduces the precision of the calculations (inputs, weights, biases)

- **parallelization**: tune how much to parallelize to make the inference faster/slower versus FPGA resources

**Constraints:**
- Input bandwidth
- FPGA resources
- Latency
Bring DL to FPGA for L1 trigger with high level synthesis for machine learning

- A package for automatic translation of trained NN into HLS project
- Optimization for **efficient inference** (low latency, low resources)
- Very customazible: tunable precision and parallelization (area vs latency)
- Many architectures supported (dense NN, CNN, LSTM, Graph NN)
- Includes an interface to the library and several utilities
Neural Network Inference

\[ x_n = g_n(W_{n,n-1}x_{n-1} + b_n) \]

- **Activation Function**
- **Multiplication** precomputed and stored in BRAMs
- **Addition** performed in DSPs, logic cells

Input layer: 16 inputs, 64 nodes
Layer 2: 64 nodes, activation: ReLU
Layer 3: 32 nodes, activation: ReLU
Layer 4: 32 nodes, activation: ReLU
Output layer: 5 outputs, activation: SoftMax

\[ N_{\text{multiplications}} = \sum_{n=2}^{N} L_{n-1} \times L_n \]

How many resources?
- DSPs, LUTs, FFs?
- Does the model fit in the latency requirement?

Layer configuration:
- 16 inputs
- 64 nodes
- 32 nodes
- 32 nodes
- 5 outputs

Layer layout:
- Layer 1
- Layer 2
- Layer 3
- Layer 4
- Output layer

M hidden layers

Layer diagram:

---

28
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!

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

ex: Xilinx Virtex Ultrascale +
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!

Achieve lowest possible latency:
execute all multiplications at once and use all DSPs
→ CONS: LIMITED NEURAL NETWORK SIZE

reuse = 1
use 4 multipliers 1 time each
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!

Reuse the resources:
execute only few multiplications at once and use fewer DSPs!

→ CONS: LONGER LATENCY

reuse = 4
use 1 multiplier 4 times

reuse = 2
use 2 multipliers 2 times each
Neural Network inference on FPGA

Smart trick for all cases: **store weights in on-chip memory** and avoid I/O

**PROS:** achieve lowest possible latency

**CONS:** NN size limited by on-chip memory size (~100Mb)
Physics case: jet tagging

Study a multi-classification task to be implemented on FPGA: discrimination between highly energetic (boosted) $q$, $g$, $W$, $Z$, $t$ initiated jets

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t \to bW \to bbqq$</td>
<td>3-prong jet</td>
</tr>
<tr>
<td>$Z \to qq$</td>
<td>2-prong jet</td>
</tr>
<tr>
<td>$W \to qq$</td>
<td>2-prong jet</td>
</tr>
<tr>
<td>$q/g$ background</td>
<td>no substructure and/or mass $\sim 0$</td>
</tr>
</tbody>
</table>

Reconstructed as one massive jet with substructure
Physics case: jet tagging

Input variables: several observables known to have high discrimination power from offline data analyses and published studies [*]

Physics case: jet tagging

• We’ll train the five class multi-classifier on a sample of ~1M events with two boosted WW/ZZ/tt/qq/gg anti-\( k_T \) jets

• Fully connected neural network with 16 expert-level inputs:
  - Relu activation function for intermediate layers
  - Softmax activation function for output layer

\[ \text{AUC} = \text{area under ROC curve} \]
(100% is perfect, 50% is random)
Efficient NN design: quantization

- In the FPGA we use an efficient **fixed point** representation
  - operations are integer ops, but we can represent fractional values
- But we have to make sure we’ve used the correct data types!
Efficient NN design: quantization
Efficient NN design: parallelization

3-layer pruned, Kintex Ultrascale

- Reuse Factor = 1
- Reuse Factor = 2
- Reuse Factor = 3
- Reuse Factor = 4
- Reuse Factor = 5
- Reuse Factor = 6

- Fully parallel
  - Each mult. used 1x

- Each mult. used 2x

- Each mult. used 3x

More resources

Longer latency

DSP vs. Fixed-point precision

1e3 hls4ml
Efficient NN design: parallelization

Latency of layer $m$

$$L_m = L_{\text{mult}} + (R - 1) \times I_{\text{mult}} + L_{\text{activ}}$$

- Longer latency
  - Each mult. used 6x
  - Each mult. used 3x
  - Fully parallel
    - Each mult. used 1x
- More resources
Efficient NN design: quantization

This is what we call **post-training quantization**: is there a better way to do this?

**YES**, minimize the loss function with quantization in mind with quantization-aware training

**Scan integer bits**
Fractional bits fixed to 8

**Scan fractional bits**
Integer bits fixed to 6

Full performance at 6 integer bits

Full performance at 8 fractional bits
QKeras & hls4ml

• QKeras is a **quantization aware training tool** that was developed by Google which partner with the hls4ml team for a first implementation to probe its power for efficient ML on hardware

**Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors**

Claudionor N. Coelho Jr, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol & Sioni Summers

*Nature Machine Intelligence* 3, 675–686 (2021) | [Cite this article]

1249 Accesses | 18 Citations | 25 Altmetric | [Metrics]

https://www.nature.com/articles/s42256-021-00356-5
https://github.com/google/qkeras

**Quantized neural networks on the edge**

As deep neural networks are pushed towards larger and more complex architectures, they require significant computational resources and are challenging to deploy in real-time applications. To reduce complexity, neural networks can be compressed, without a substantial decrease in model accuracy, by methods such as pruning or quantization. The latter involves using fewer bits to represent weights and biases. In a paper in this issue,... **show more**
QKeras & hls4ml

- QKeras is a *quantization aware training tool* that was developed by Google which partner with the hls4ml team for a first implementation to probe its power for efficient ML on hardware

- Simple drop-in replacement of Keras layers allows for homogeneous or heterogenous quantization accross NN layers

- During training, quantization functions only applied to the forward pass but not in the backward pass to make the gradient differentiable

```python
from tensorflow.keras.layers import Input, Activation
from keras import quantized_bits
from keras import QDense, QActivation
from keras import QBatchNormalization

x = Input((16))
x = QDense(64,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(32,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(5,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer = quantized_bits(6,0,alpha=1))(x)
x = Activation('softmax')(x)
```

Matches ap_fixed exactly!
- same granularity as hls4ml
- same precision at training and inference
QKeras & hls4ml in action

Tested the QKeras+hls4ml bundle on the 3 layers jet tagging model
Summary & Outlook

• Today we have introduced the need in particle physics to bring AI closer to the data source at the edge of particle physics detectors

• Trigger systems typically employ FPGAs for real-time data processing

• A tool exists — hls4ml — that can help physicists to bring AI to such systems in order to meet tight constraints that go beyond industry ones while giving you control on the optimization

• Next days:
  - tomorrow afternoon we will go through a demo which will serve as starting point for your own deeper navigation of the hls4ml tool
  - on Thursday we will discuss some interesting applications in HEP from anomaly detection at the LHC to detection of supernovas in LArTPC experiments