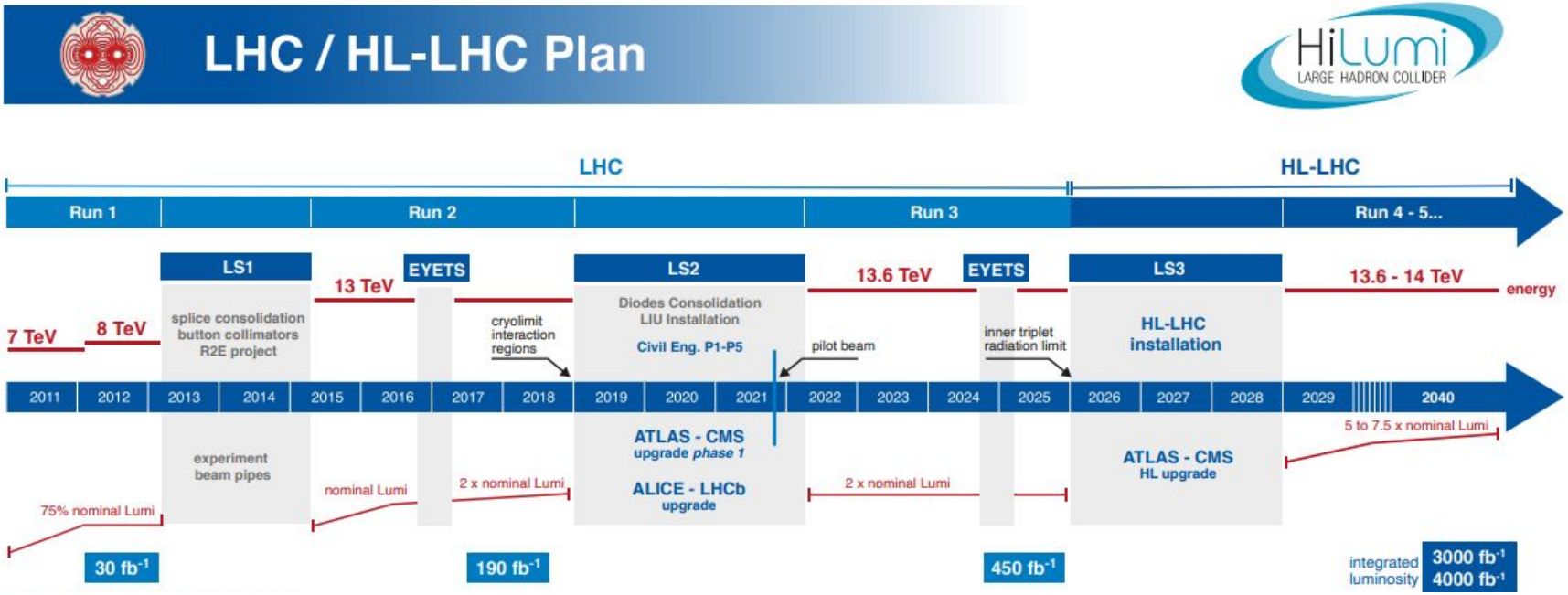
Embedded Recurrent Neural Networks on FPGAs for Real-Time Computation of the Energy Deposited in the ATLAS Liquid Argon Calorimeter

IJCLab Seminar
26/06/2024

Georges Aad
CPPM

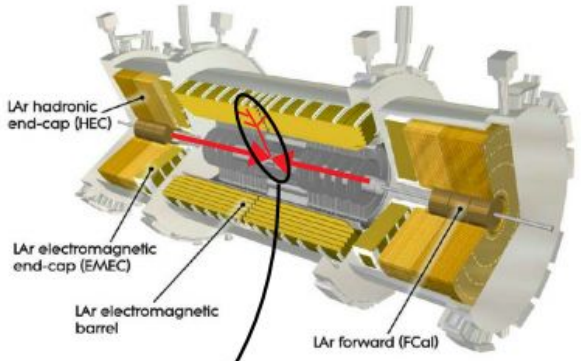
Introduction

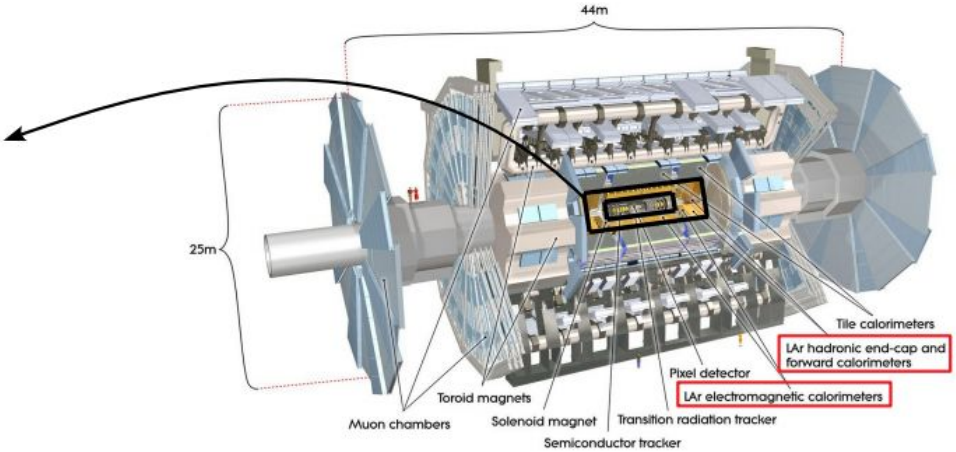
- LHC upgrade during the long shutdown starting 2026 leading to the HL-LHC
  - Increase the instantaneous luminosity by a factor 5 to 7 with respect to the LHC design value
  - 140 to 200 simultaneous proton-proton collisions (pileup)
- ATLAS will be upgraded to cope with the HL-LHC conditions
  - Increase the level 1 trigger frequency from 100 kHz to 1 MHz
  - ➢ New readout electronics for the liquid argon calorimeter



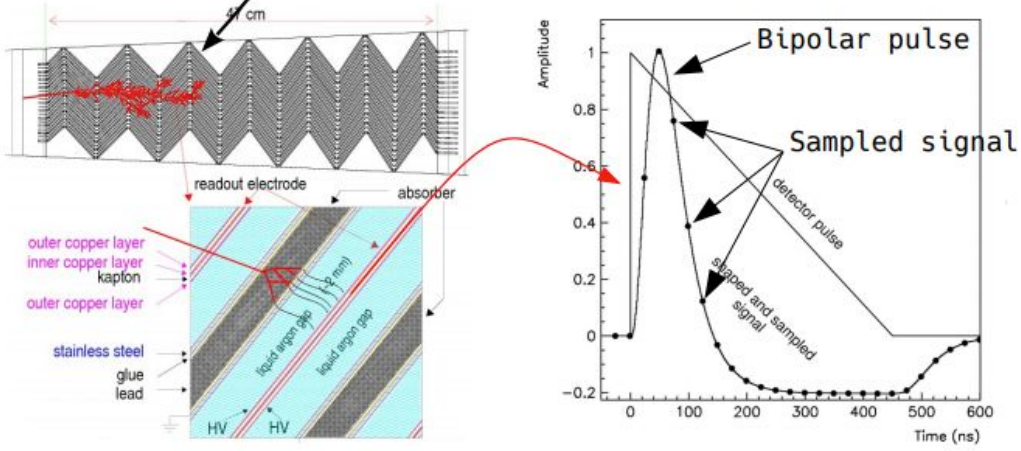
The ATLAS Liquid Argon Calorimeter

- Measures the energy of electromagnetically interacting particles mainly electrons and photons
- Trigger capabilities at the first level of triggering (implemented in hardware)
  - Fast processing of the data needed (at 40 MHz)





180000 channels

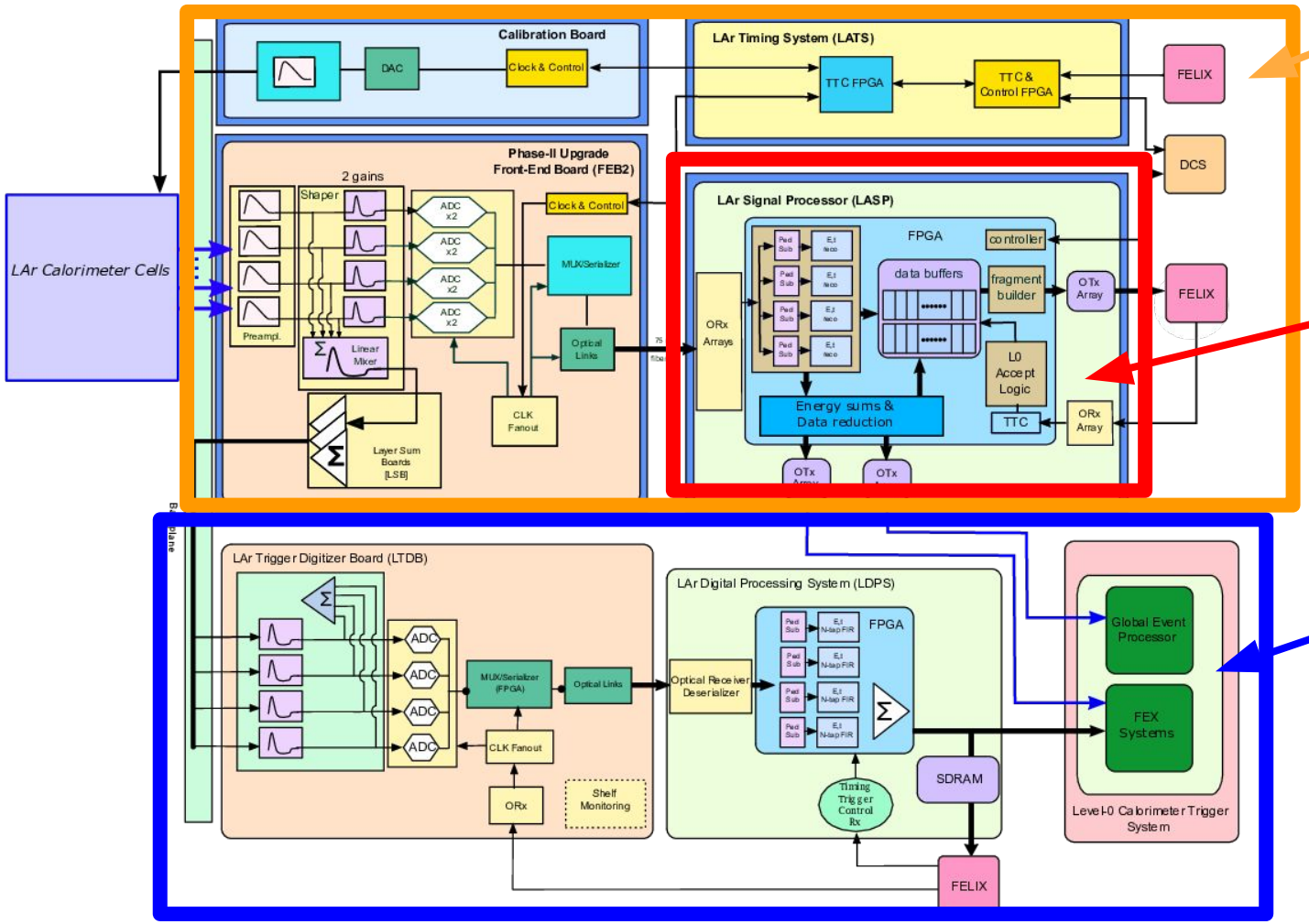


- Electronic signal amplitude proportional to the deposited energy in the calorimeter
- Shaped and sampled at 40 MHz

LAr Phase-II Upgrade

CERN-LHCC-2017-020

- Full electronics of the readout path will be exchanged
  - New on-detector electronics to digitize the signal at 40 MHz and send it to the backend
  - New off-detector electronics to compute the energy at 40 MHz



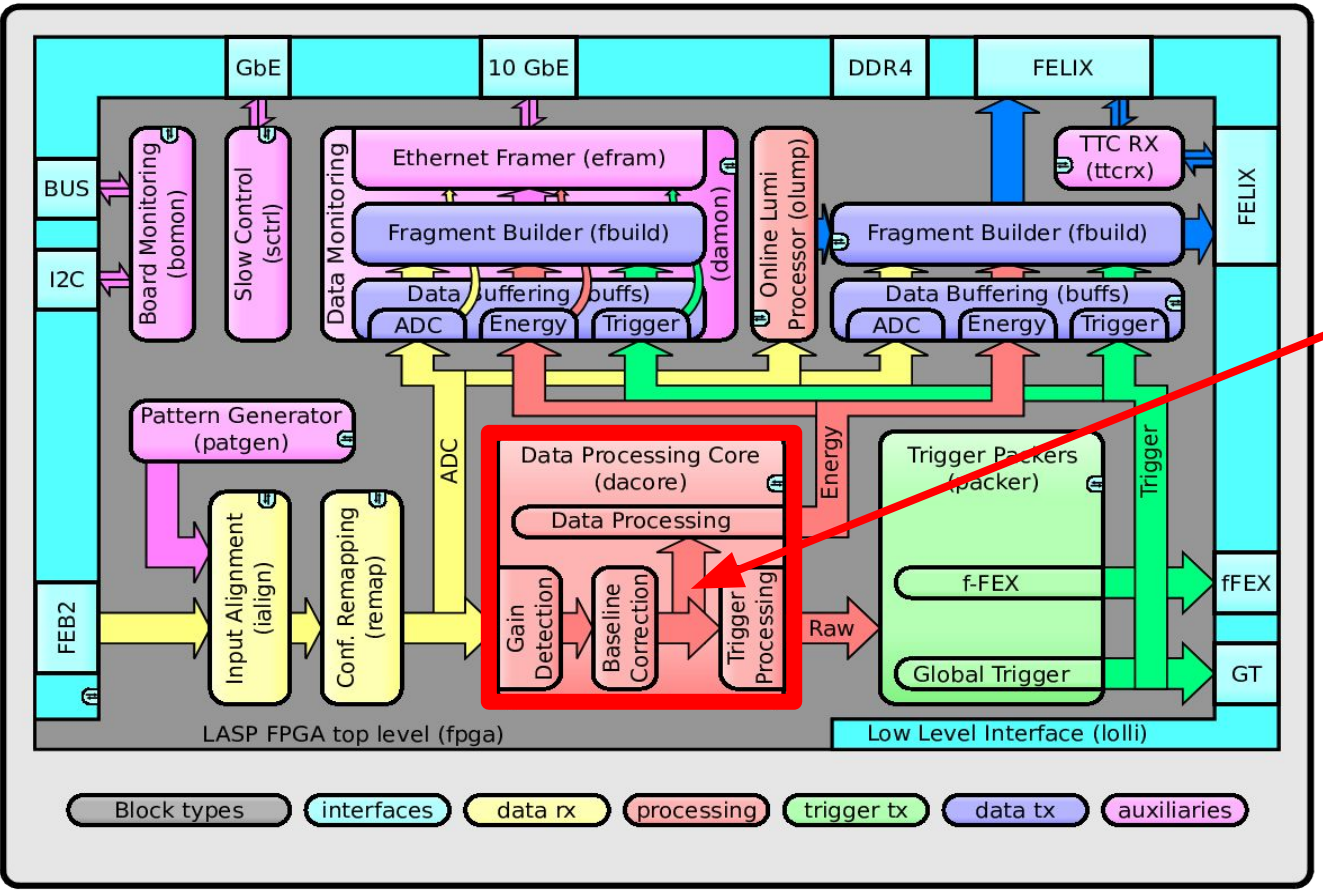
Phase 2 electronics
Trigger path
+
readout path
180k channels

LASP board
Off-detector board
responsible of energy computation

Phase 1 electronics
Trigger path
35k channels

LASP Firmware

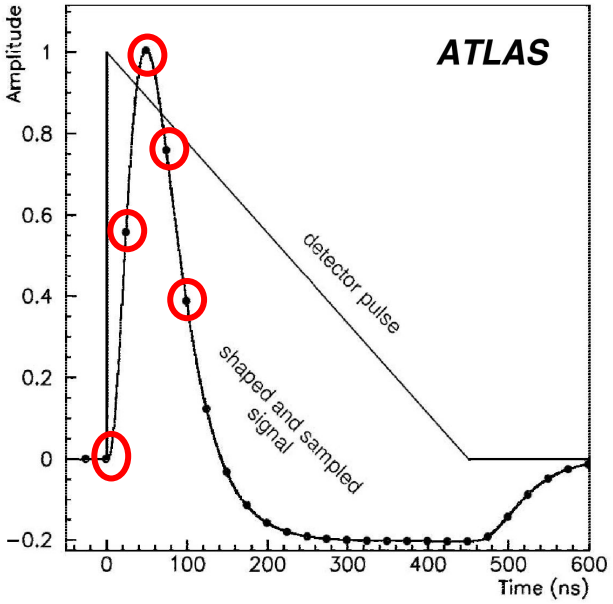
- LASP board containing 2 processing units based on INTEL FPGAs
  - Demonstrator board available with Stratix 10 FPGAs
  - Final board will be equipped with Agilex FPGAs
- One FPGA should process 384 channels
  - About 125 ns allocated latency for energy computation

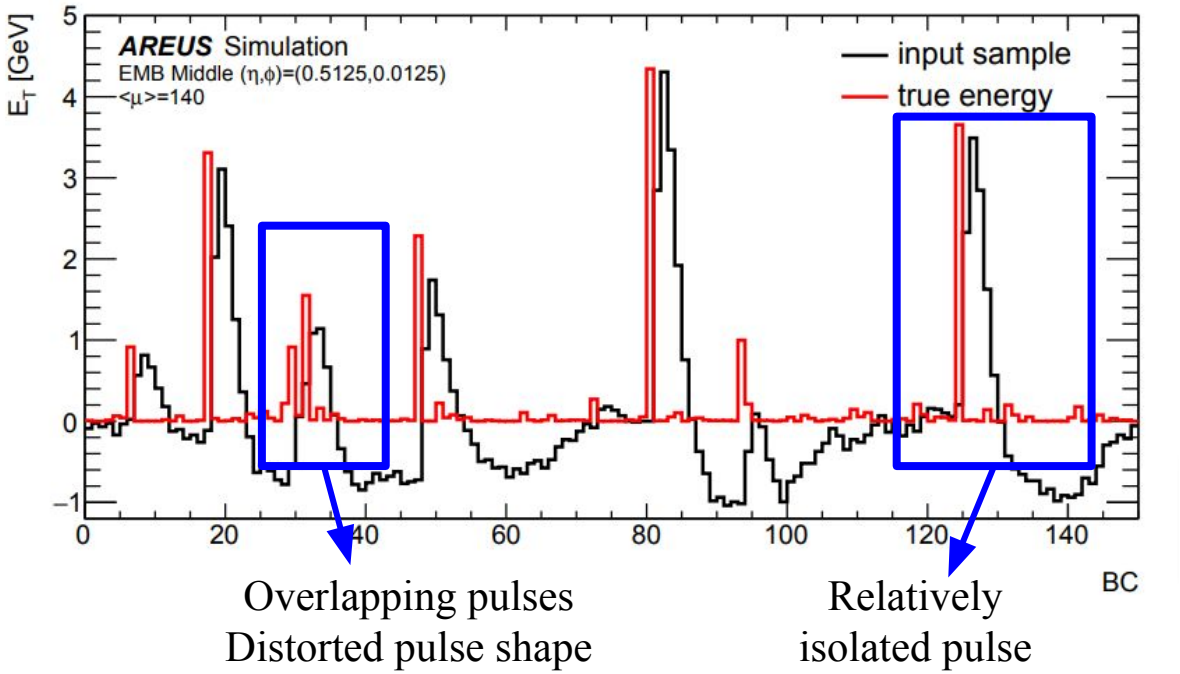


Compute energy at 40 MHz
Assign the energy to the correct bunch crossing (collision time)

Energy reconstruction

- Legacy energy reconstruction using an optimal filtering algorithm with maximum finder (OFMax)
  - Optimal filtering to reconstruct the pulse and determine its amplitude ($\propto$ energy)
  - Max finder to determine the correct time (bunch crossing)
- Not robust in case of distorted shapes due to pileup
  - Use NNs to recover performance at the HL-LHC





Energy from Optimal-Filter (OF)

n = 5 in this talk

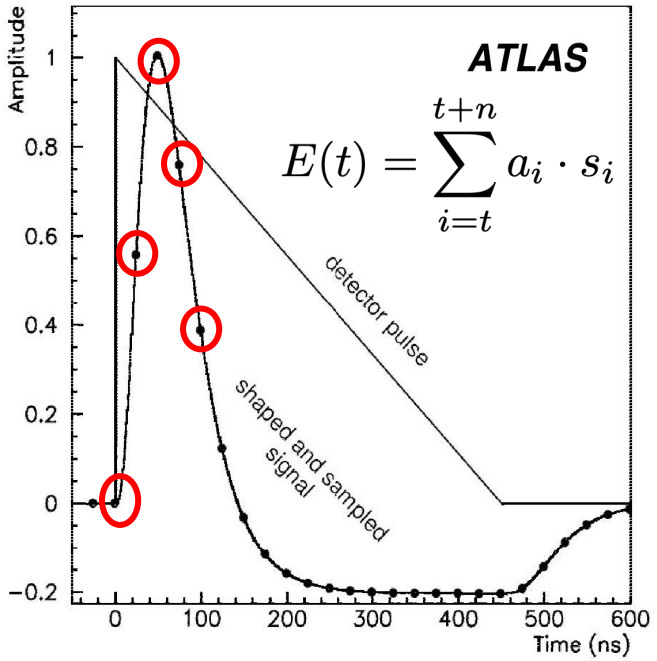
$$E(t) = \sum_{i=t}^{t+n} a_i \cdot s_i$$

Pulse Samples

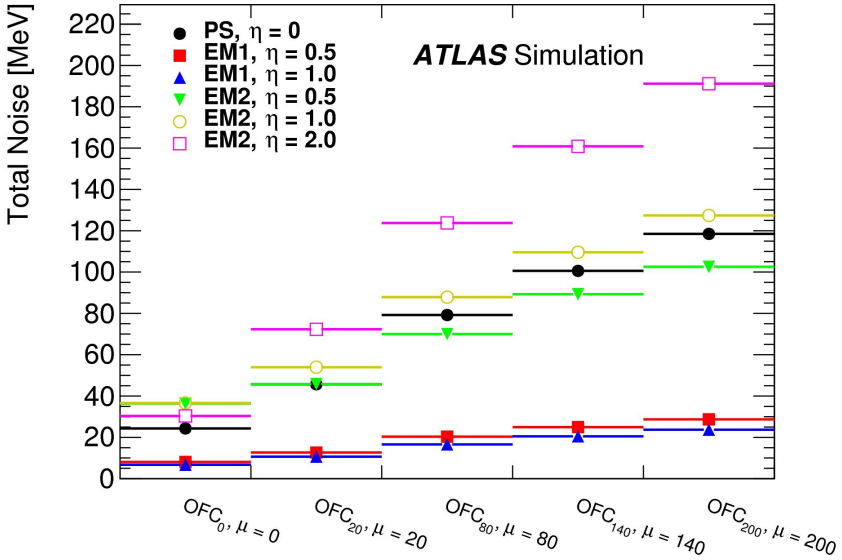
Pre-set coefficients (fit of the peak)

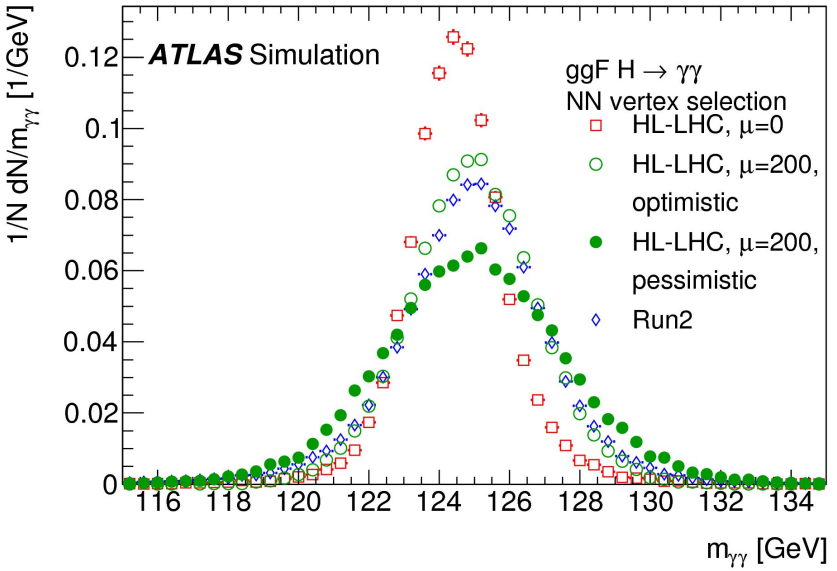
Energy reconstruction at the HL-LHC

- Increased noise due to increased pileup
  - Up to a factor of 2 with respect to Run 3
- About 30% degradation in $m_{\gamma\gamma}$ resolution
- Better energy reconstruction algorithms needed
  - Neural networks are obvious candidates



ATL-COM-LARG-2017-030





Energy reconstruction with NNs

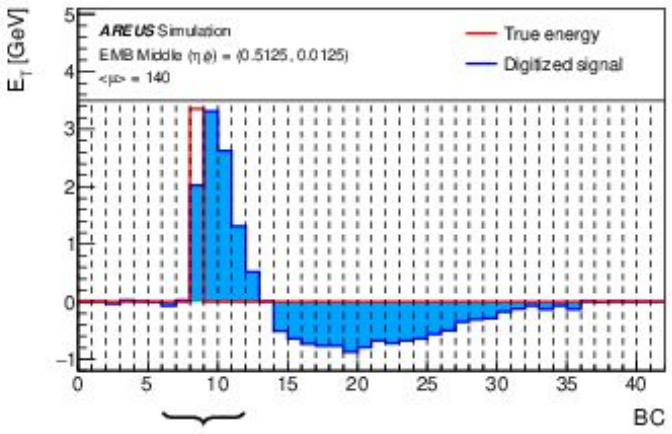
Two neural networks types tested:
Convolutional Neural Networks (CNNs) (Dresden)
and
Recursive Neural Networks (RNNs) (CPPM)

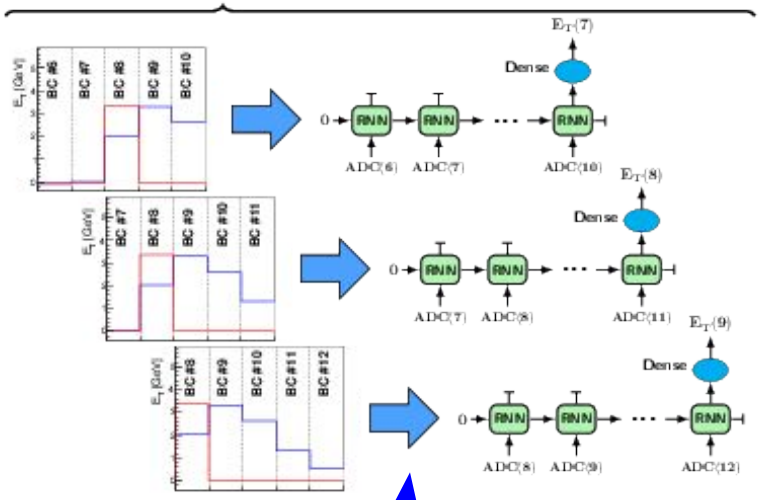
This talk will cover only RNNs

RNN structure

Comput Softw Big Sci 5, 19 (2021)

- Single cell architecture
  - Full history learned
  - Less robust against intermittent problems such as noise bursts
  - Need large cells to handle full history
- Sliding window architecture
  - Learn only local effects (what we need)
  - Intermittent problems have only short time effect
  - Suitable for small cells

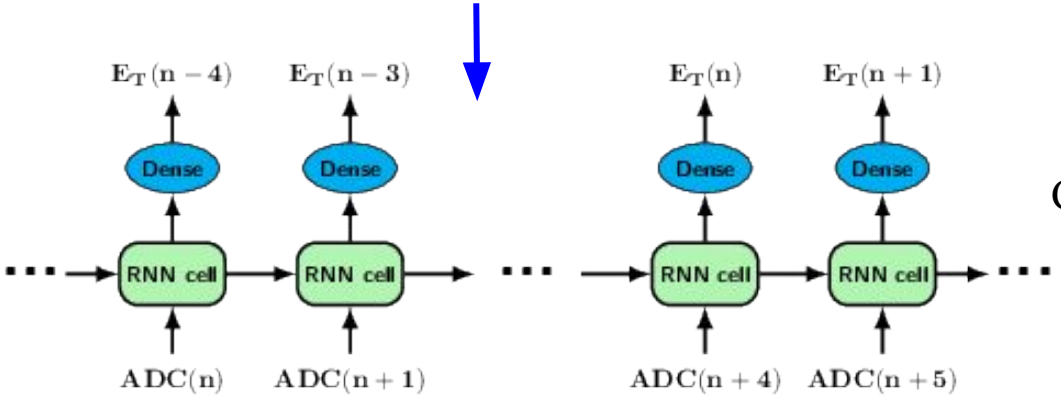




Single cell architecture

Continuous computation with a single cell

Takes into account full past info (from the beginning of run)



Sliding windows architecture

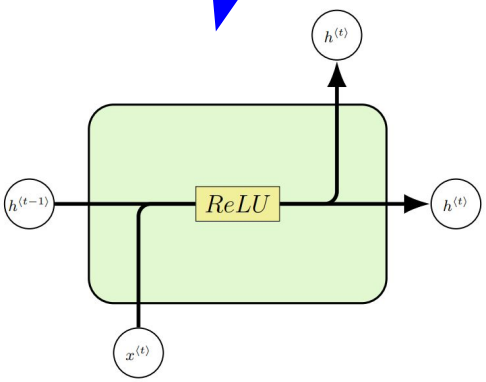
Computation on a moving slice (fixed intervals)

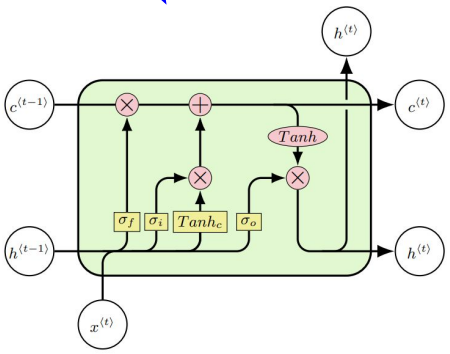
Takes into account a limited set in the past (1 sample in the past for this example)

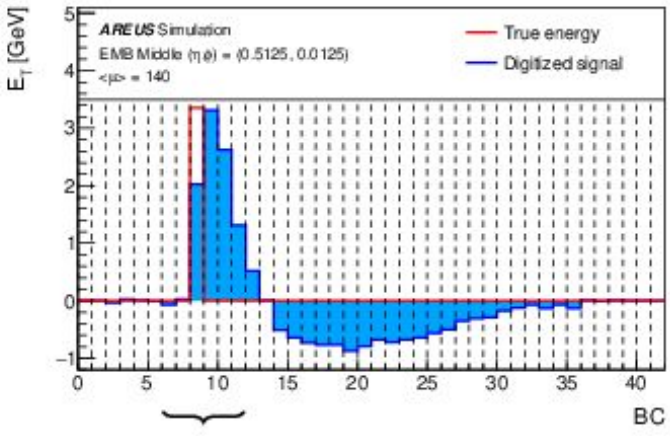
RNN structure

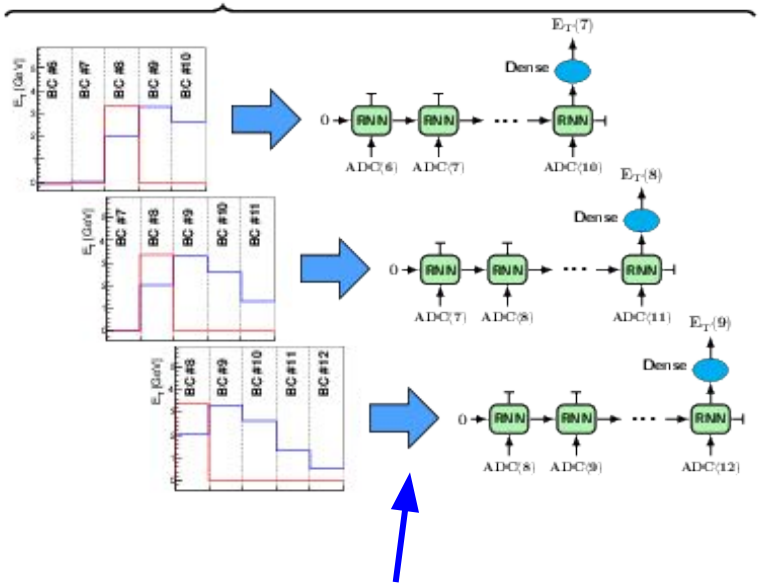
Comput Softw Big Sci 5, 19 (2021)

- Sliding window architecture retained
- Sequence of RNN cells each taking as input an ADC sample at a given BCID
  - 4 samples on the pulse
  - N samples prior to the pulse to correct for pileup
- Two general parameters control network size
  - Sequence length (number of samples)
  - NN units (internal dimension of the NN cells)
- Several cell structures tested
  - Vanilla RNN, GRU, LSTM









Sliding windows architecture

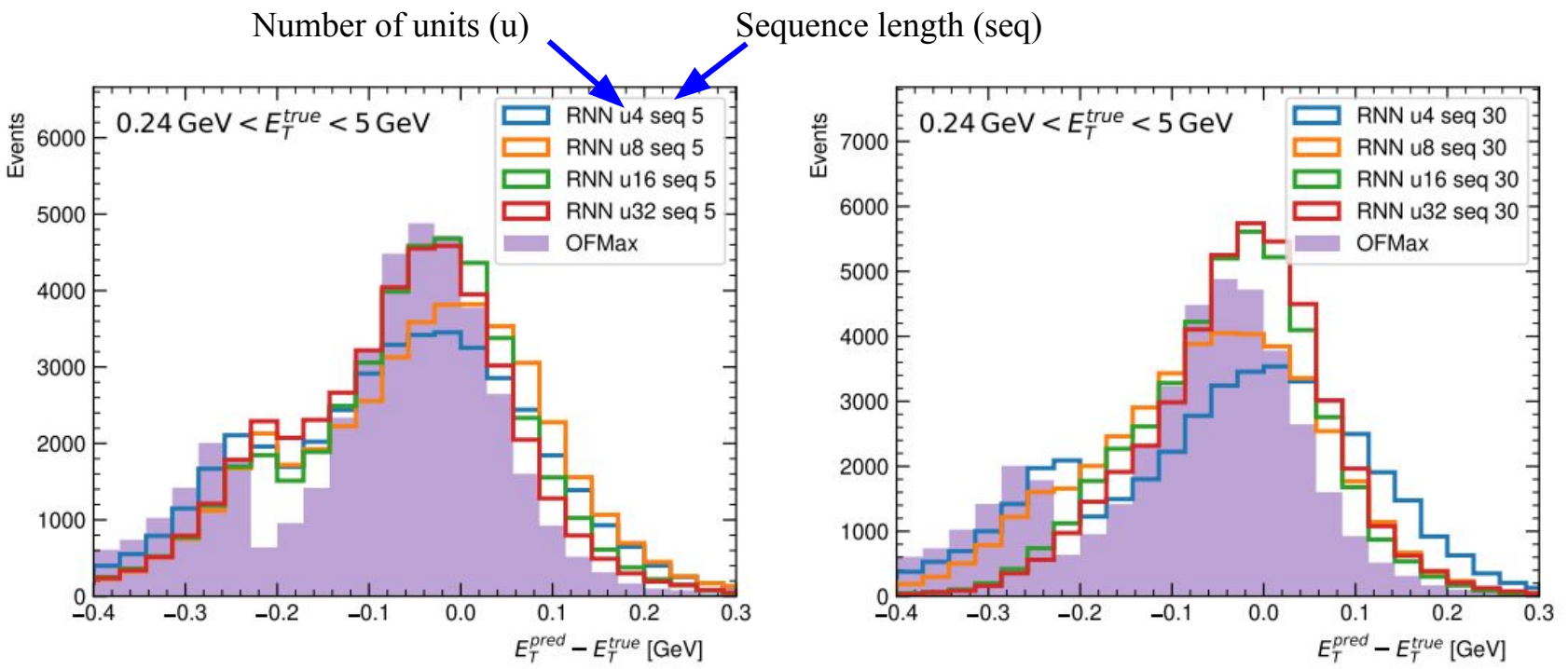
Computation on a moving slice (fixed intervals)

Takes into account a limited set in the past

(1 sample in the past for this example)

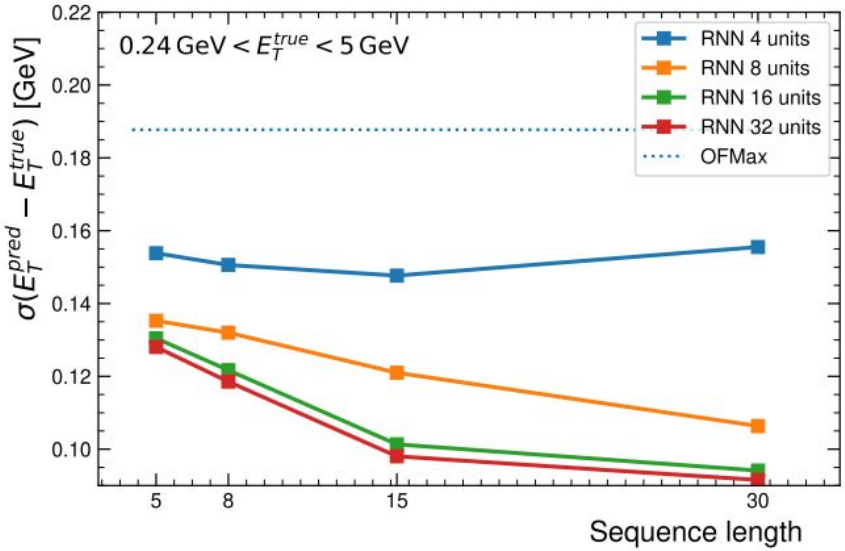
RNN Performance

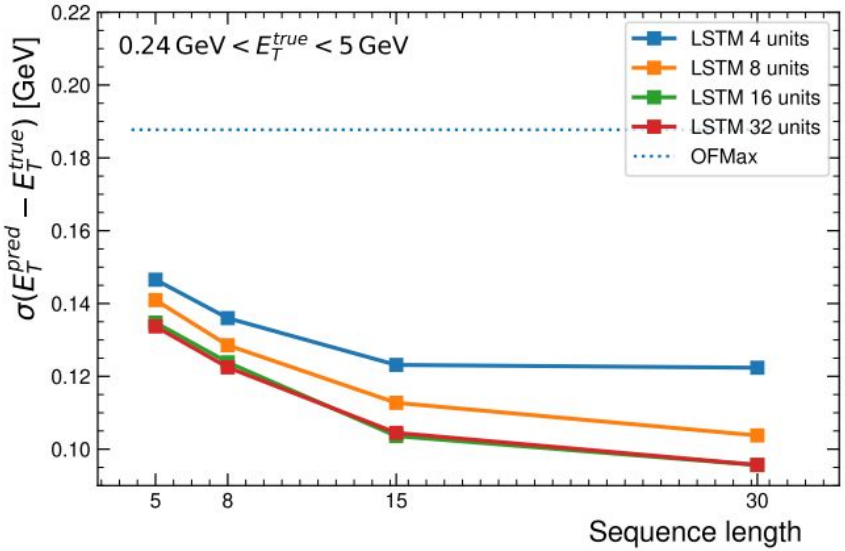
- Compare energy resolution between RNNs and OFMax
  - RNNs with increased size
  - Keep size under control to fit FPGAs
- Second peak in resolution due to overlapping events
- Use Std. Dev. as metric (although the shape is not very gaussian)



RNN Performance

- Compare energy resolution between RNNs and OFMax
  - RNNs with increased size
  - Keep size under control to fit FPGAs
- Second peak in resolution due to overlapping events
- Use Std. Dev. as metric (although the shape is not very gaussian)



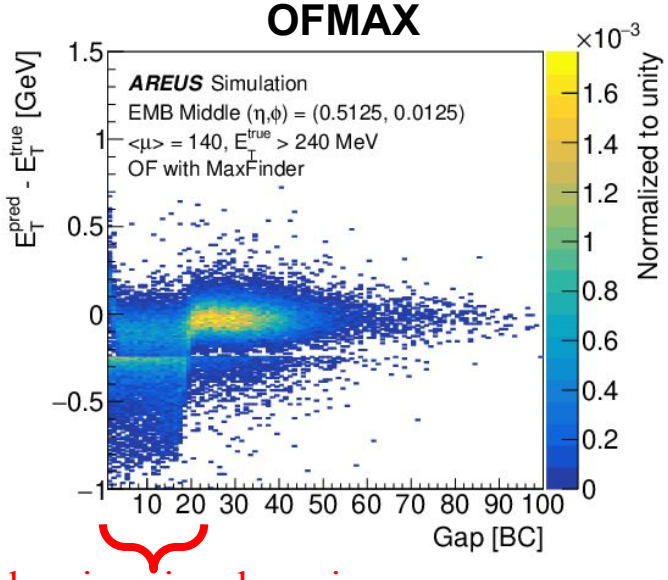


Performance as Function of Time Gap

Lauri Laatu, PhD thesis (2023)

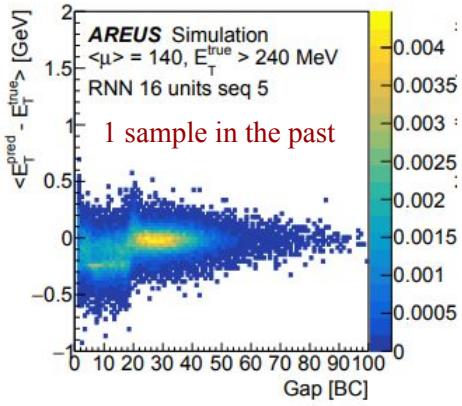
- Energy resolution as function of the time gap between two pulses to isolate pileup effects
- Clear drop in OFMax performance when pulses overlap
  - Time gap of less than ~ 20 BC
- Neural networks recover the performance in this region
  - Strongly dependent on the number of samples used in the past (prior to the energy deposit)

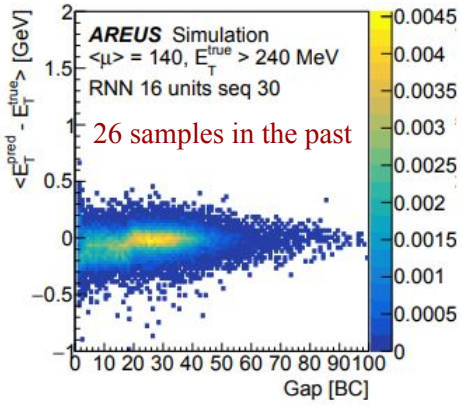
OFMAX



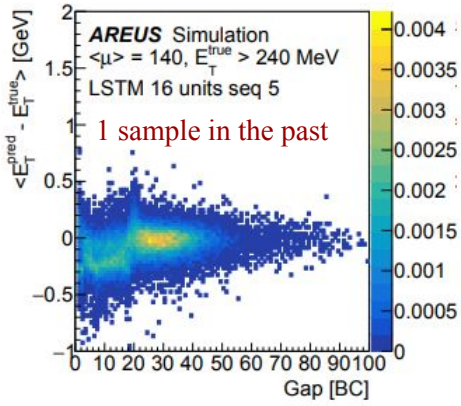
Overlapping signals region

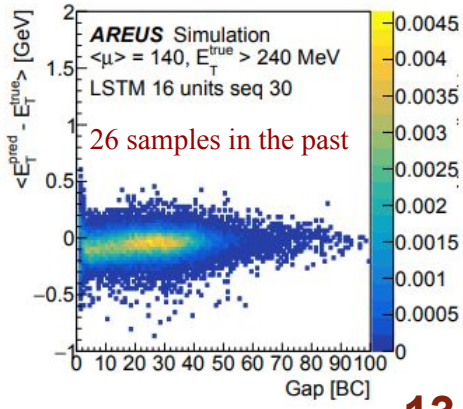
Vanilla RNN





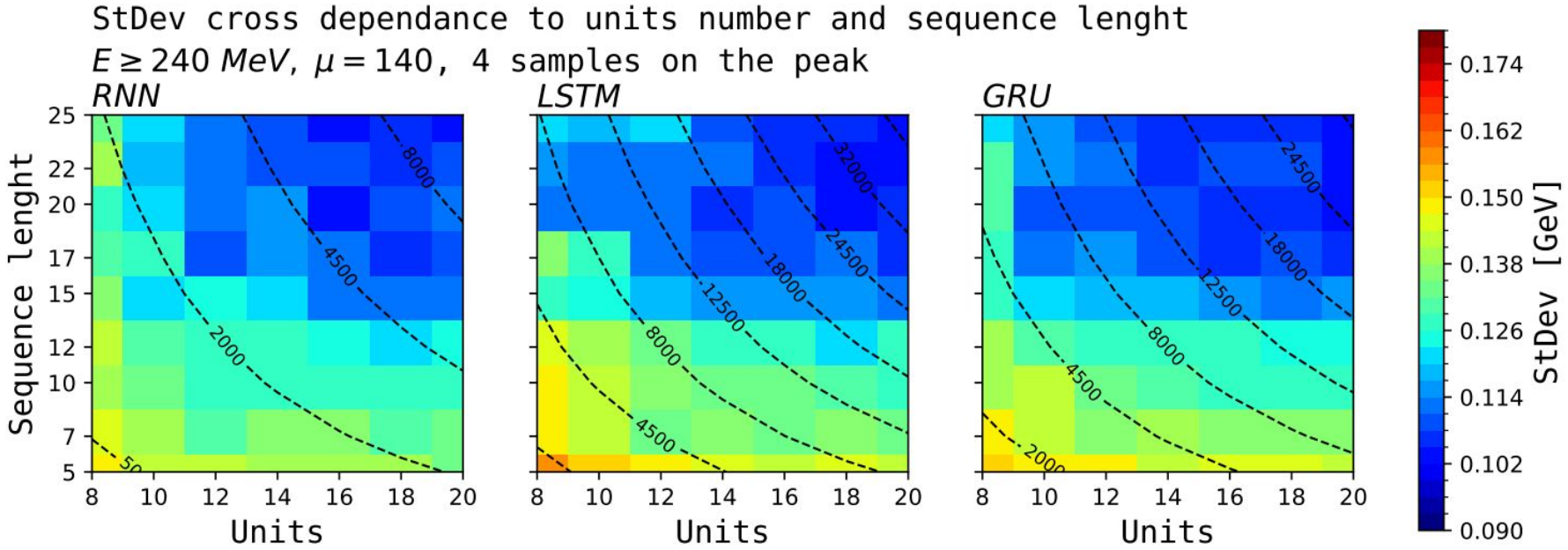
LSTM



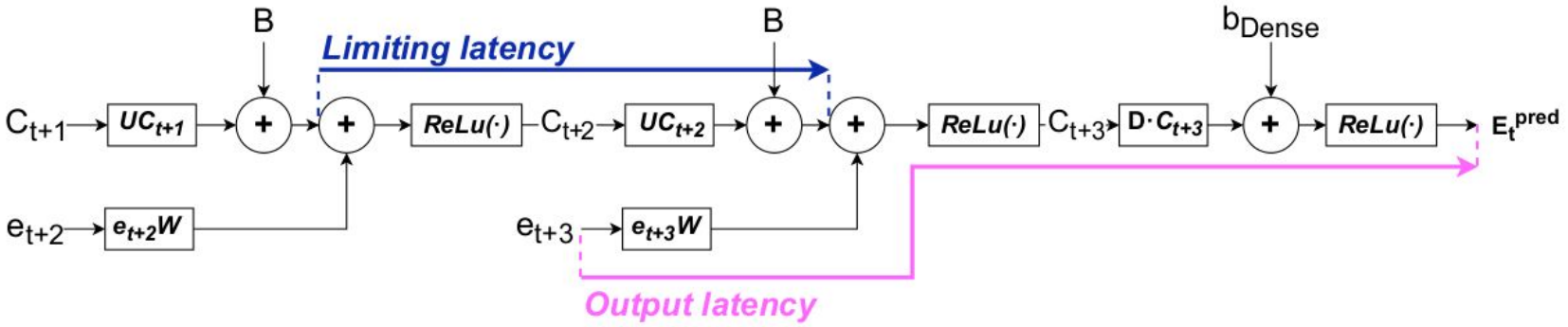


RNN Performance vs RNN Cell Type

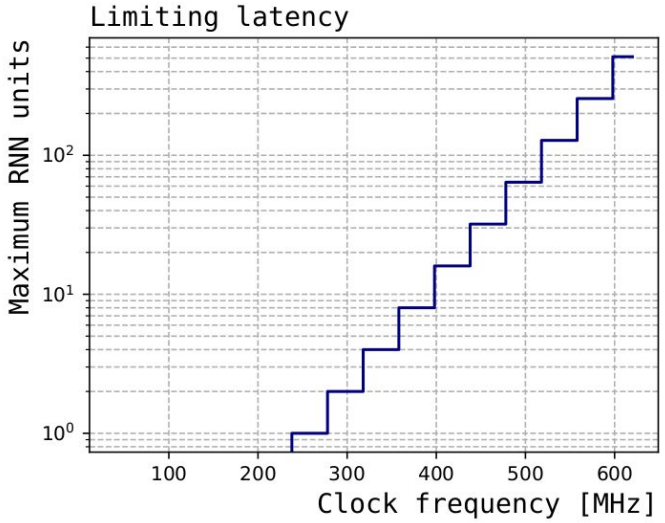
- Checking performance of Vanilla-RNN, GRU and LSTM
  - Increased NN size by increasing sequence length and number of units
- Network size probed by number of multiplications (MAC units)
  - Dashed lines in the plots
- Vanilla-RNN can reach the same performance with much less required MACs
  - Best adapted to fit in FPGAs
  - However best performance still too big for FPGA (can fit NNs with O(1000) MACs)

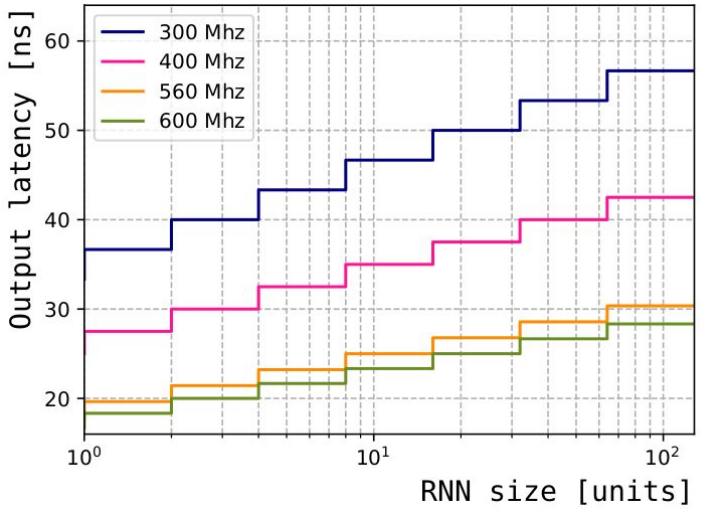


RNN latency

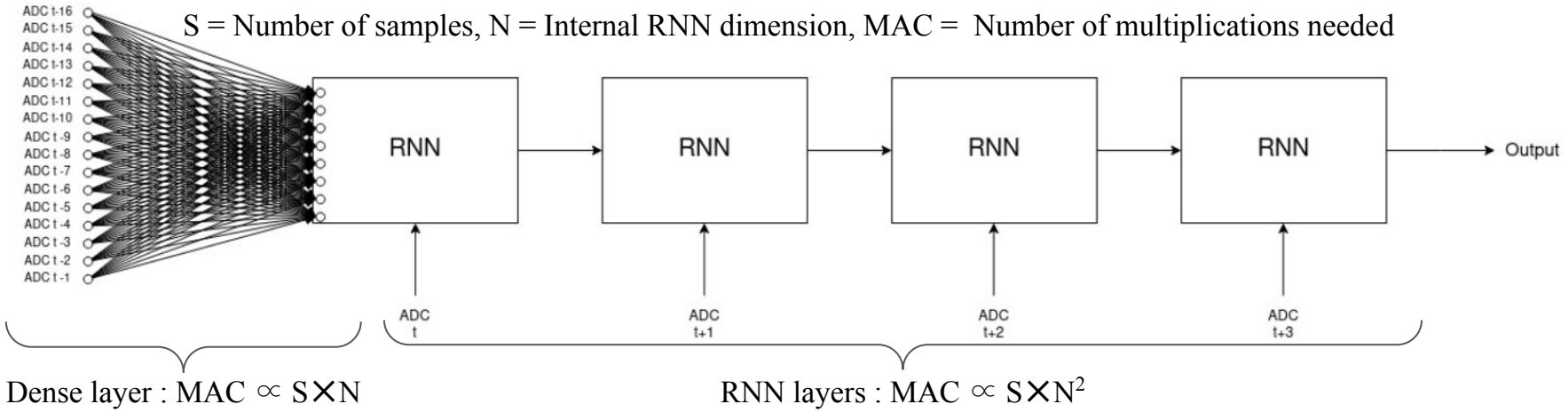


- Minimum achievable latency for Vanilla RNN estimated as function of the NN size
  - Additioning the number of clock cycles needed for fundamental blocks
- Two latencies are important:
  - Limiting latency: available time between 2 samples
  - Output latency: time to finish the computation after the last sample
- RNN cells with up to 100 units possible (latency is not the limiting factor at high frequency)



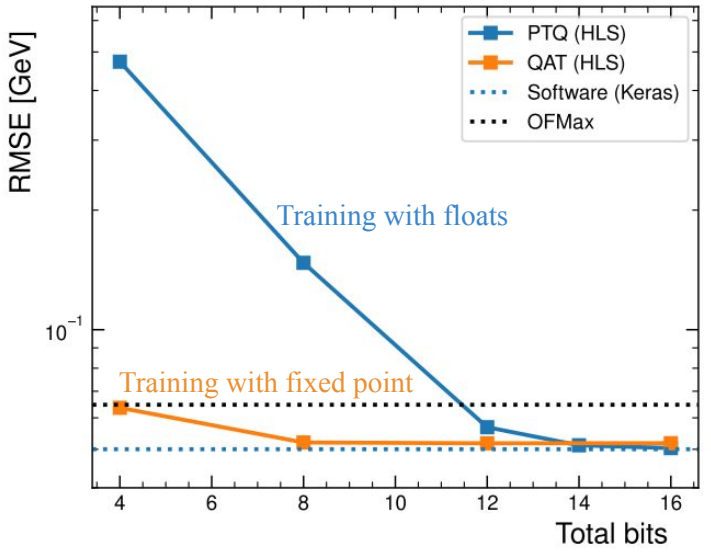


Optimisation of computational resources



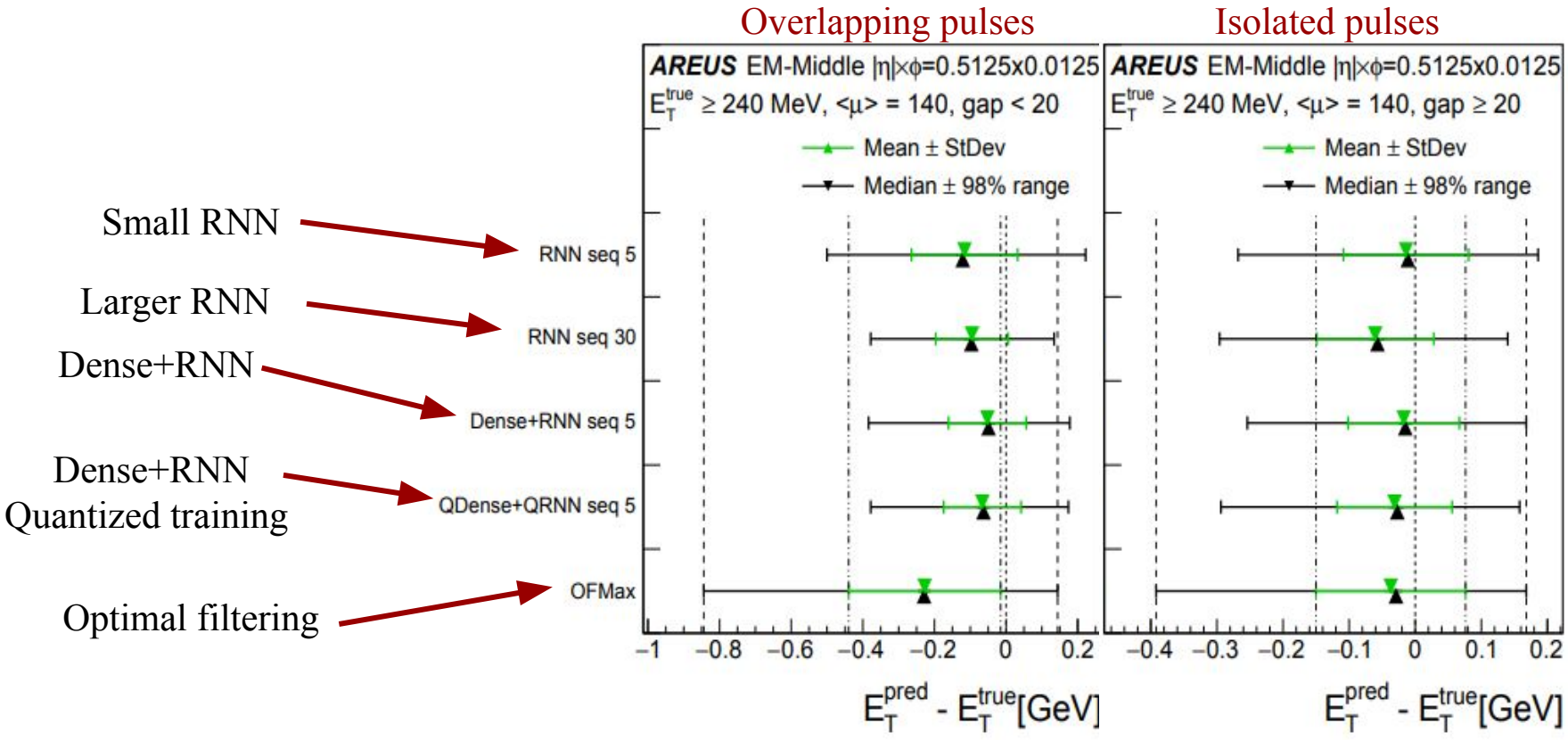
- Long sequences needed to efficiently correct for pileup
  - Significant computational resources needed for RNN cells
- Replace RNN cells in the past by a dense layer
  - Dense to correct to pileup, RNN to compute the amplitude
  - Reduce the number of needed multiplications by a factor 4
    - For a network with dimension 30 and sequence length 20
  - No effect on performance
- Reduce number of bits needed for arithmetic computation
  - Replace floating point with fixed point operation
  - Train the network directly with fixed point (QAT)
  - Quantization aware training (QAT) can reduce the number of needed bits by a factor 2

Simulation of the energie resolution in firmware as function of the number of bits



RNN performance (summary)

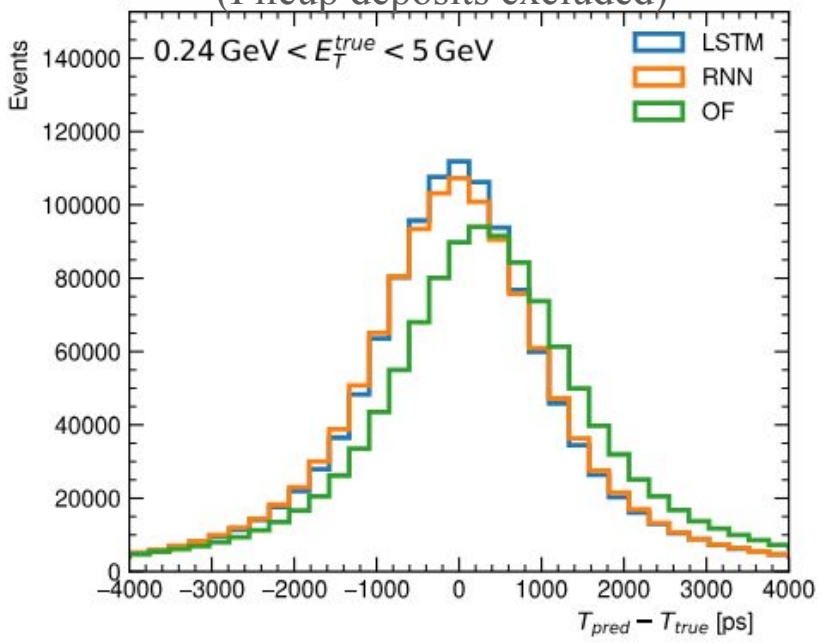
- Small RNNs (sequence length of 5 samples) can outperform OFMax overall
  - But not in all regions
  - Larger networks needed
- Several optimisation carried out to improve the performance
  - Keeping the network suitable for FPGA processing



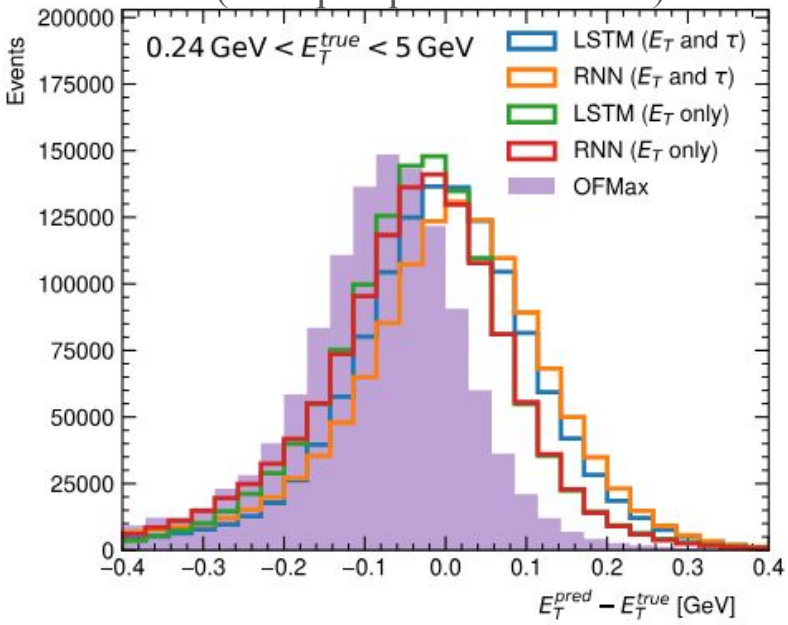
Computing the deposit time

- OFMax can also compute the time of the deposit
  - Phase with respect to the time of training
- Can be done easily with the NN by adding one additional neuron at the output for the time
  - Adds n MAC units (n is the internal dimension of the network)
- Achieved better resolution than OFMax
  - But degradation of the energy resolution observed

Time resolution
(Pileup deposits excluded)



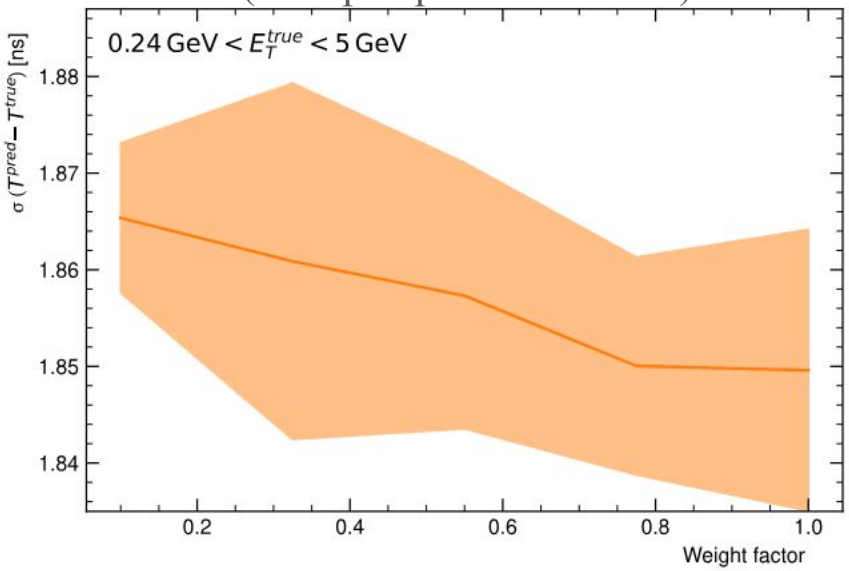
Energy resolution
(Pileup deposits excluded)



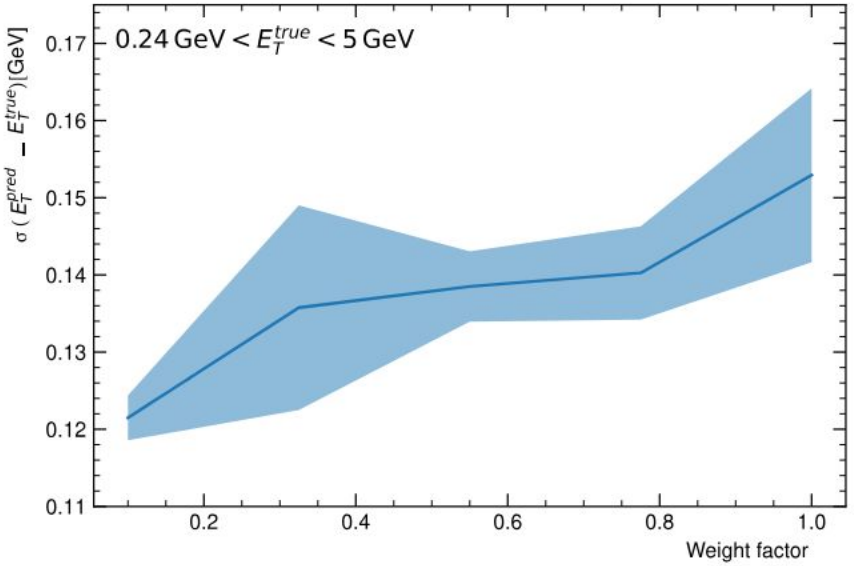
Computing the deposit time

- OFMax can also compute the time of the deposit
  - Phase with respect to the time of training
- Can be done easily with the NN by adding one additional neuron at the output for the time
  - Adds n MAC units (n is the internal dimension of the network)
- Achieved better resolution than OFMax
  - But degradation of the energy resolution observed
  - Can be mitigated by weighting the loss function $L = MSE(e) + w \cdot MSE(t)$

Time resolution
(Pileup deposits excluded)



Energy resolution
(Pileup deposits excluded)

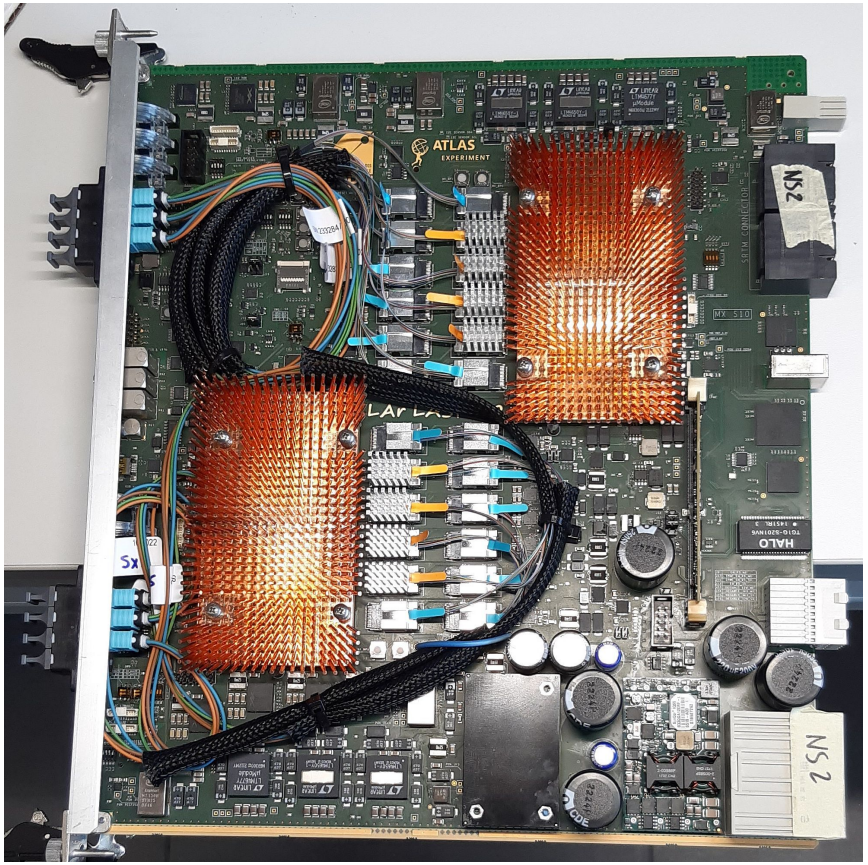


Firmware Implementation

- Implemented on Stratix 10 FPGA
  - Reference 1SG280HU1F50E2VG
  - Implementation on Agilex ongoing
- Challenges:
  - 384 channels per FPGA
  - 125 ns latency

- Preliminary implementation in HLS (High Level Synthesis) shows that LSTM is too large to fit
- Focus on Vanilla RNN
- Start with small RNN
  - 8 units and sequence length of 5
  - 89 parameters
  - 368 multiplications/accumulations (MAC) needed

LASP demonstrator board



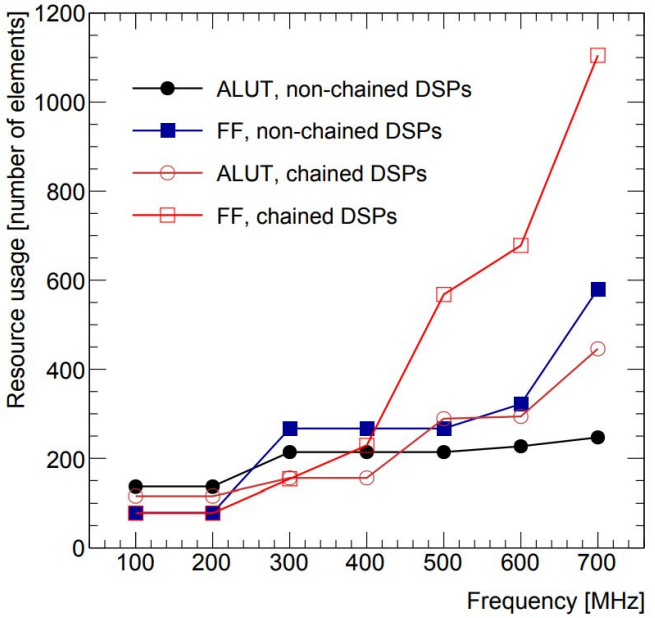
Implementation in HLS

JINST **18** (2023) P05017
Etienne Fortin, PhD thesis (2022)

- Optimisation needed to fit RNNs within resource and latency limitations
  - Impossible to fit 384 RNN instenses in the FPGAs
  - Need to serialize (time multiplexing)
  - Need to go to high frequency (multiple of 40 MHz)
- Optimisation of vector/matrix multiplications
  - Most elementary operation inside neural networks
  - Naive C++: let HLS do it all
  - ACC37: Accumulate (sum) in DSPs by chaining them
  - ACC19: Accumulate in general logic elements (ALUT)

$$A.B = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_8 \end{bmatrix} . \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_8 \end{bmatrix} = \sum_{i=0}^{7} a_i.b_i$$

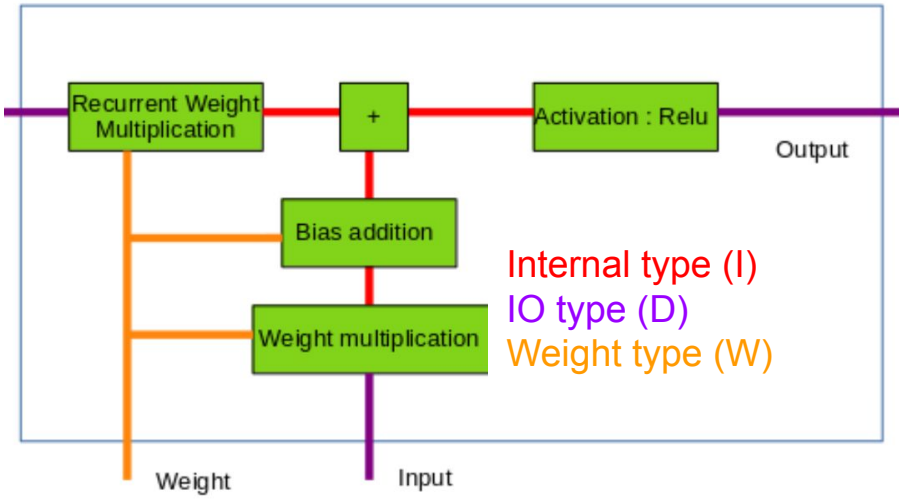
	Implementation	ALUTs	FF	DSP
@100 MHz	C++ style	709	222	8
	ACC37	116	79	4
	ACC19	137	78	4

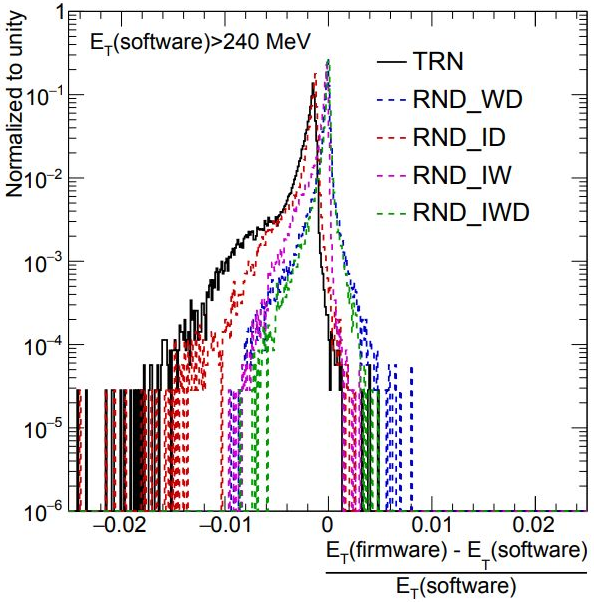


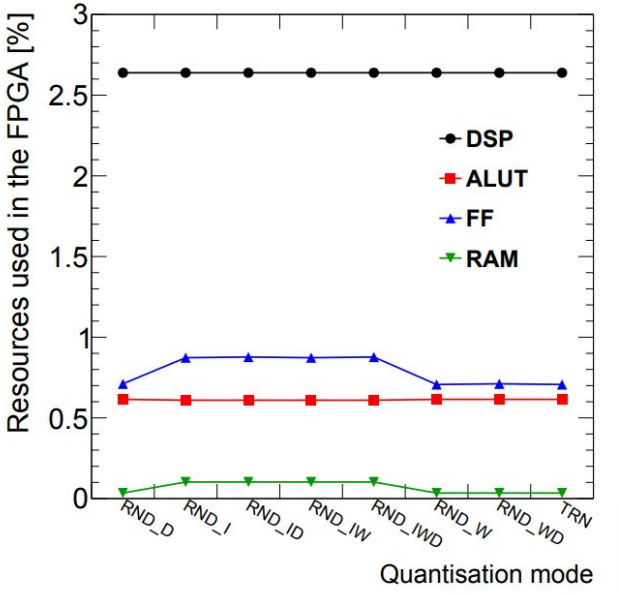
- Best strategy depends on frequency
  - Accumulate in DSP at low frequency
  - Accumulate in ALUT at high frequency
- Chaining DSPs at high frequency needs more logic than what is gained by performing sums inside DSPs

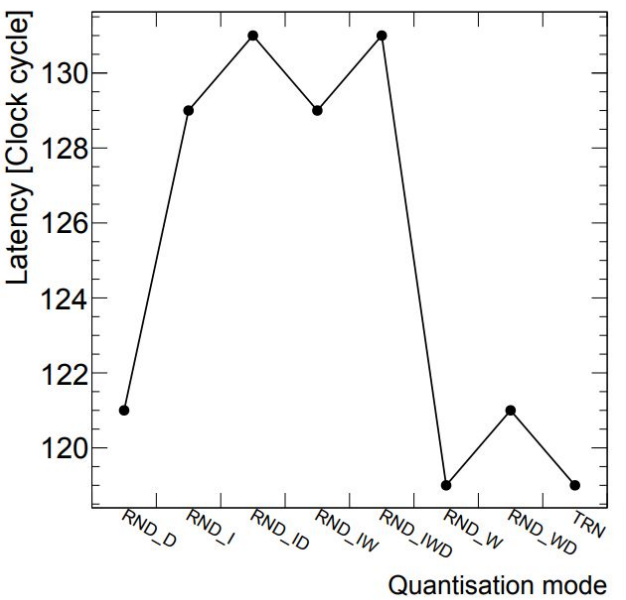
Rounding vs Truncation

- Compromise between resolution and resource usage and latency
  - Truncation of IO and Internal types leads to important reduction of latency with small impact on energy resolution
  - Weight type rounded in software
    - No impact on latency
- Use truncation in the firmware







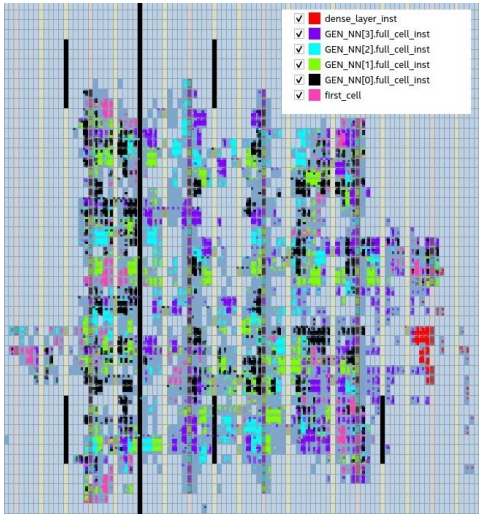


Implementation in VHDL

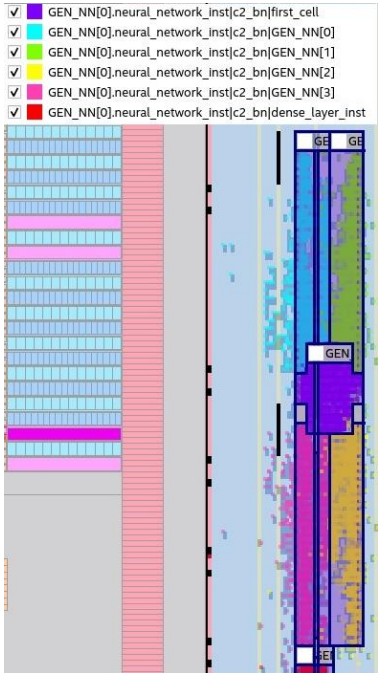
JINST **18** (2023) P05017
Etienne Fortin, PhD thesis (2022)

- HLS did not allow to reach the target frequency and resource usage
  - Increase of the needed logic (per network) and the latency as we add networks to the FPGA
- Move to VHDL for the final fine tuning
- Force placement of the RNN components
  - Allow to better tackle timing violations and improve the maximum reachable latency (FMax)
- Use incremental compilation
  - Freeze networks with no timing violations and recompile only the rest

HLS placement



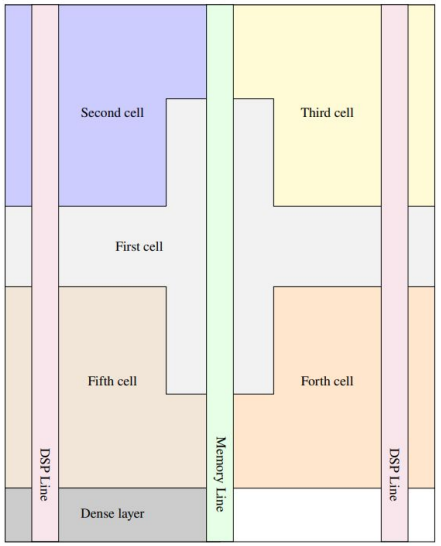
VHDL forced placement



Optimized placement of RNN cells

First cells in the middle and connected to all cells (common computations done only in first cell)

Dense layer next to last cell



RNN firmware results

JINST 18 (2023) P05017

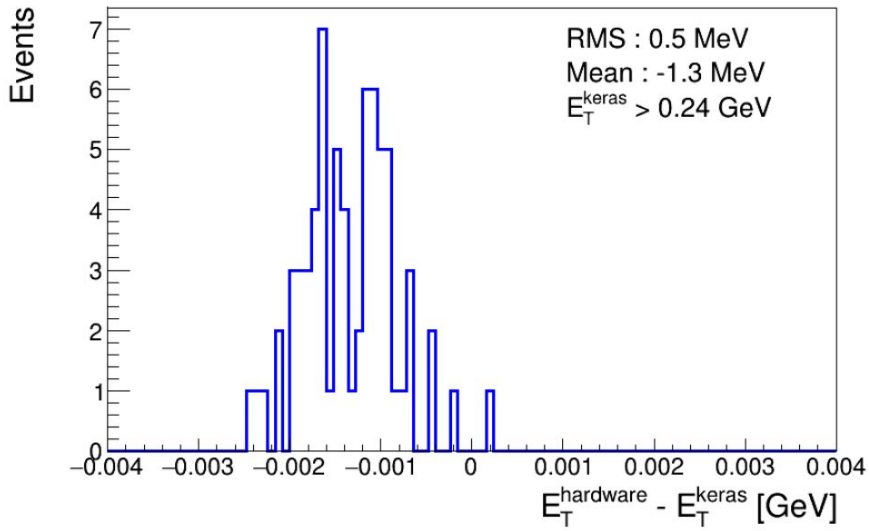
*based on experience with the phase-I upgrade

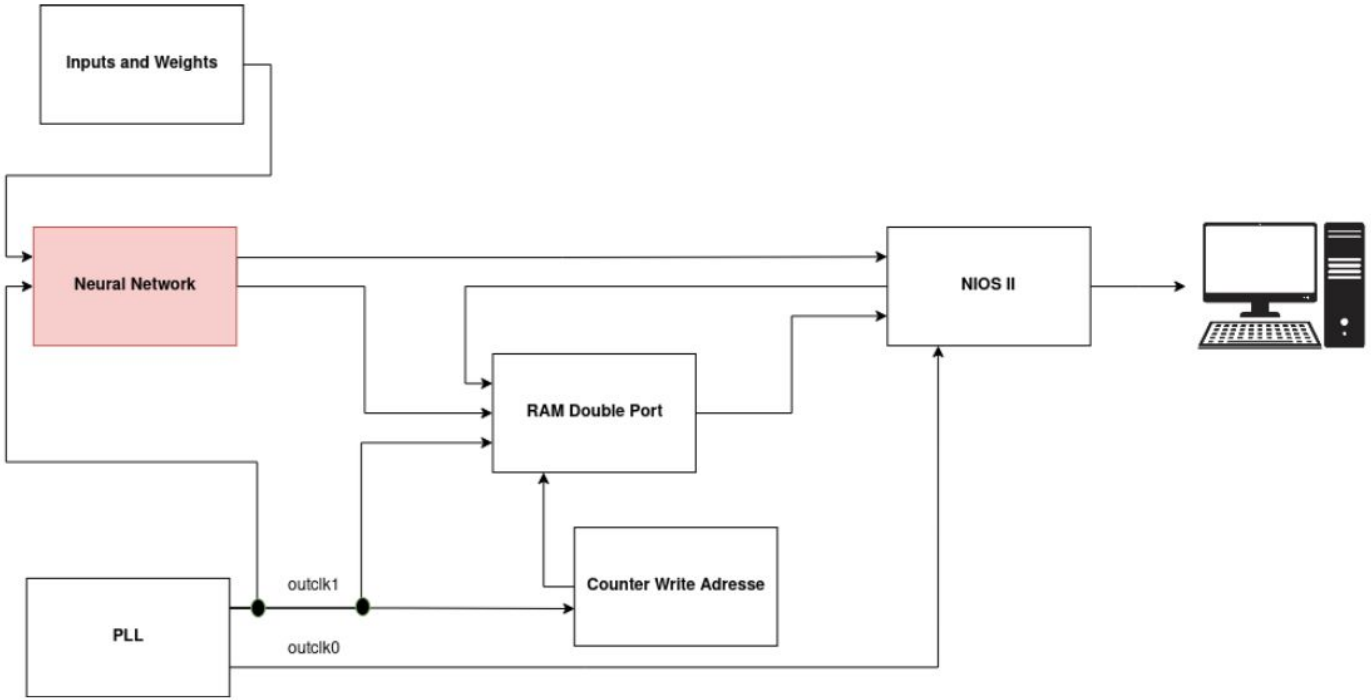
	N networks x multiplexing	ALM	DSP	FMax	latency
Target	**384 channels**	**30%***	**70%***	**Multiplexing x 40 MHz**	**125 ns**
"Naive" HLS	384x1	226%	529%	-	322 ns
HLS optimized	37x10	90%	100%	393 MHz	277 ns
VHDL optimized	28x14	18%	66%	561 MHz	116 ns

- HLS allows fast development and optimisation
  - However less control on hardware specific implementation
- VHDL is needed to fine tune the design and fit the LAr requirements
- Vanilla RNN firmware produced and fits the requirements
  - Better performance expected with the Agilex FPGA

Testing on hardware

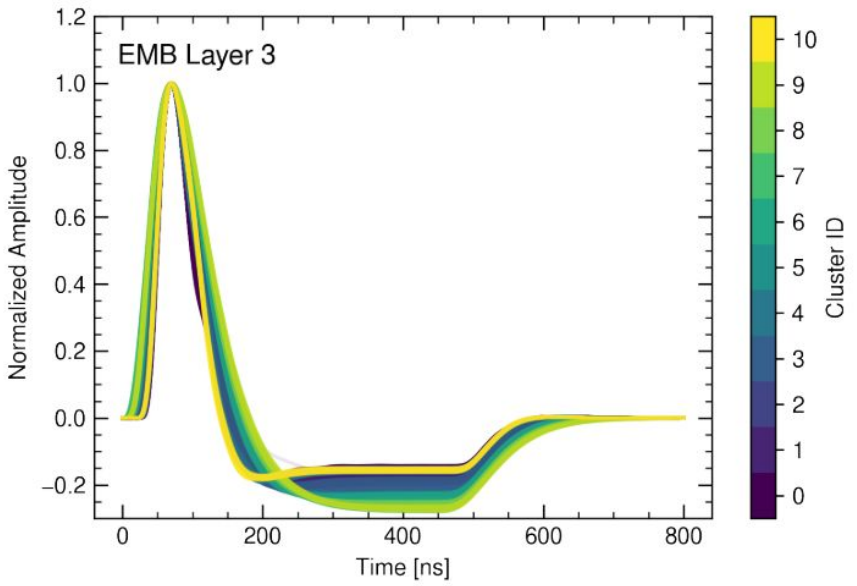
- VHDL implementation tested on Startix 10 DevKit
- Test firmware to inject input and weights and collect the output is built
  - Data extraction using a JTAG-UART connection with a NIOS
- Data match firmware simulation bit-by-bit
- Firmware resolution < 0.1% as expected from simulation

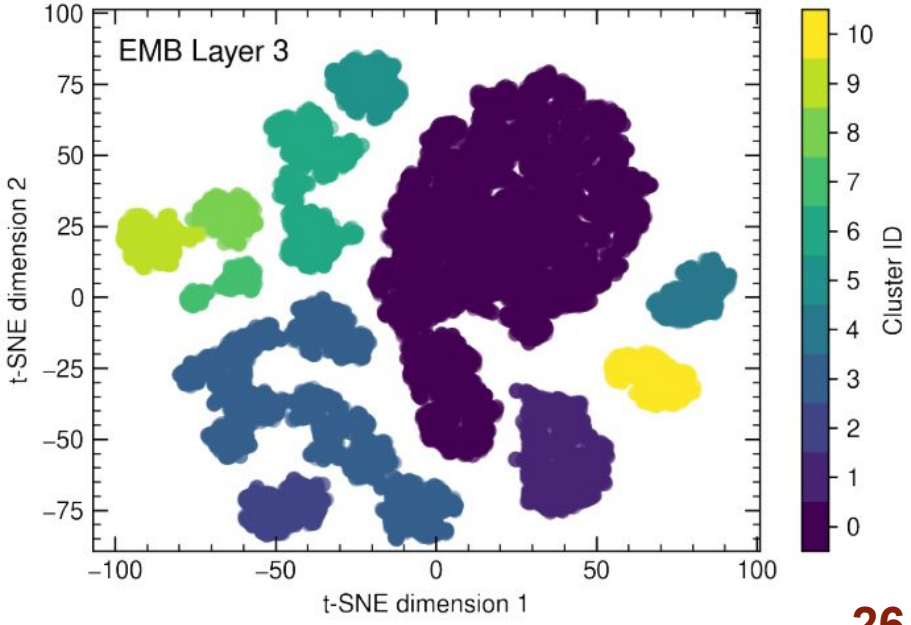




From single cell to the full detector

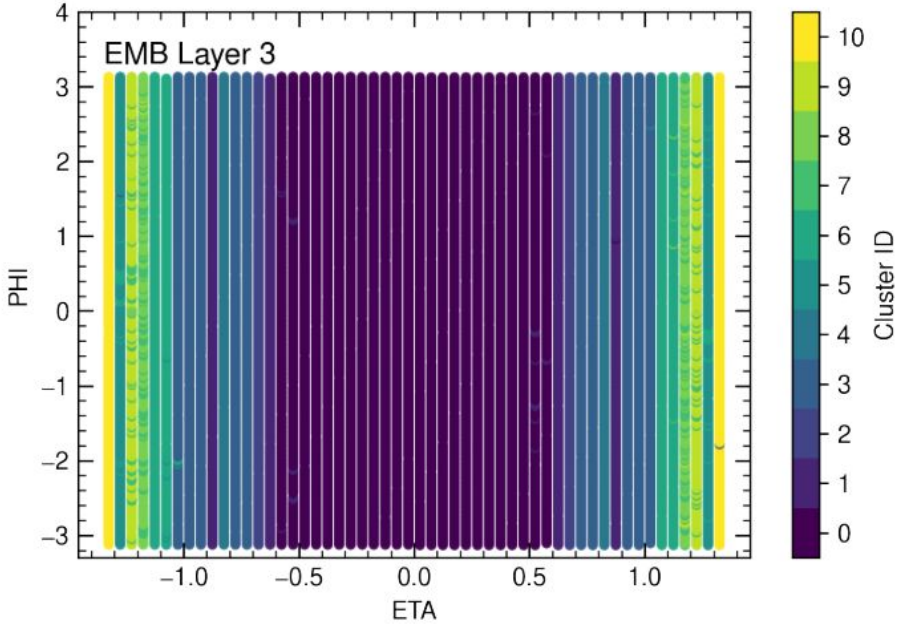
- Training 180000 NNs is not a raisonnable task
  - Not just CPU/GPU but also need to validate then
- Group cells with “similar” pulse shape into a single NN
- Cells are grouped using an unsupervised clustering method
  - t-SNE to reduce the dimensionality: from n samples of the pulse to 2 dimensions
  - DBSCAN to cluster in two dimensions

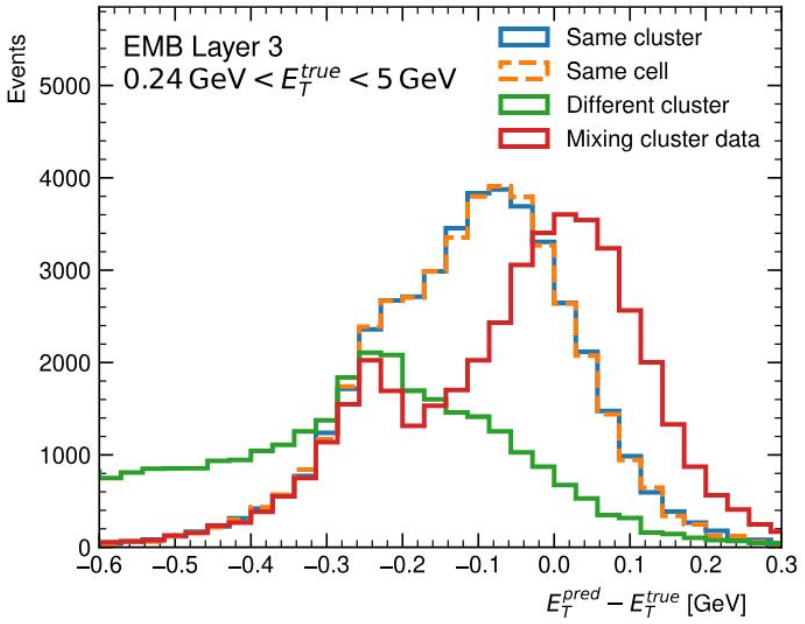




From single cell to the full detector

- Clusters manage to catch the geometric symmetries in the detector
  - Symmetry in phi
  - Changing cell size (capacitance) and thus pulse shape in eta
- Confirmed clustering does not degrade RNN performance
  - Same resolution training on cells from the same clusters
  - Dramatic degradation of resolution if training on a random cell outside the cluster
    - Training on all clusters at once does not recover the performance



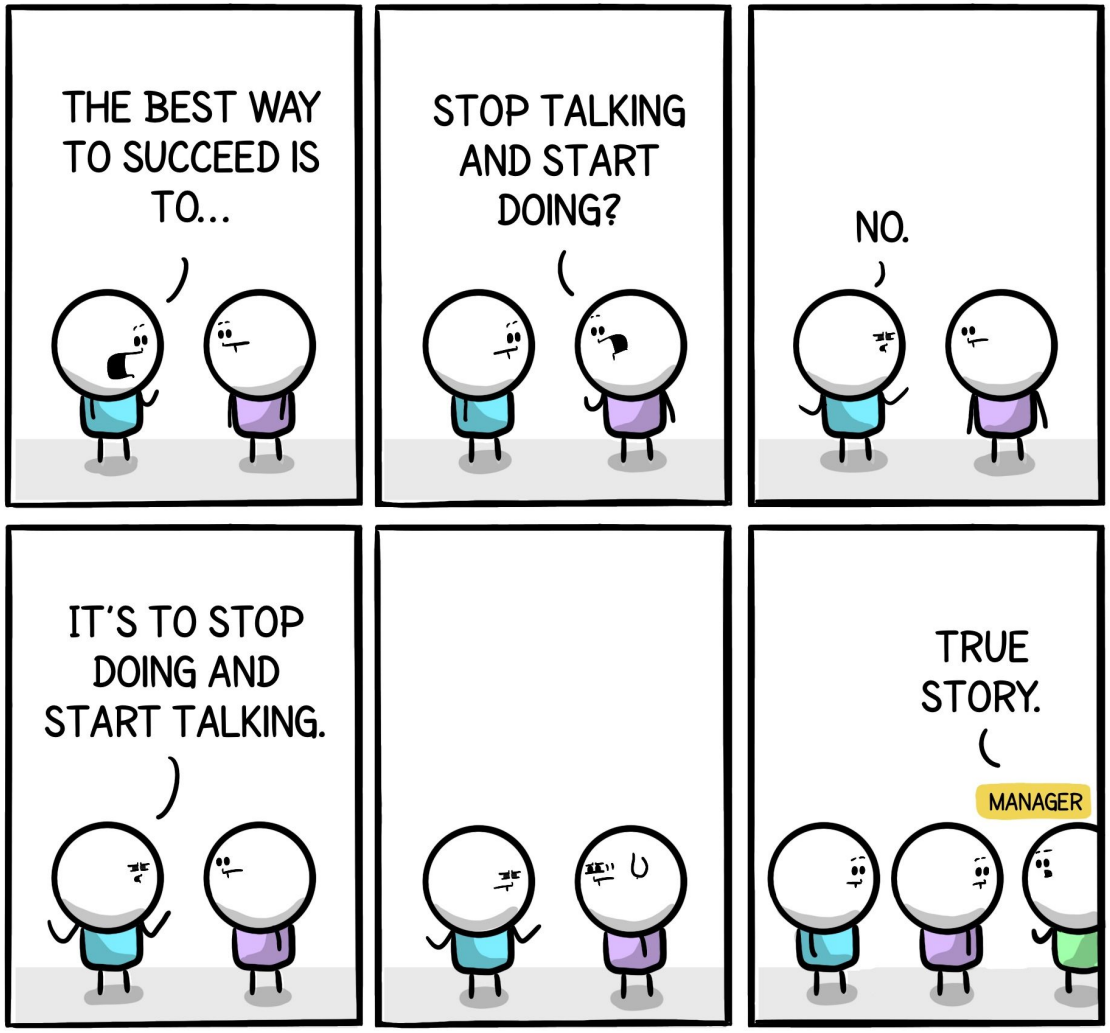


Conclusions

- Neural networks can outperform the optimal filtering algorithm for the energy reconstruction in the ATLAS LAr Calorimeter
  - Particularly in the region with overlap between multiple pulses (high pileup)
- Several optimisations carried out to improve the RNN performance while keeping minimal resource usage
  - Assessing the improvement on object reconstruction (electrons, photons) is ongoing
  - Implementation in athena (ATLAS simulation and reconstruction software) in advanced stage
    - Using the clustering technique to group cells for the training

- Small Vanilla RNN implemented on Stratix 10 FPGAs
- HLS implementation allows very fast prototyping
  - Added support for both Vanilla RNNs and LSTMs on INTEL FPGAs to HLS4ML
  - HLS design did not fit the stringent resource and latency requirements
- Final implementation done in VHDL
  - Fits requirements and successfully tested on hardware
- Next steps is to implement larger networks in Agilex FPGAs

Backup



Hello. I make comics about work.
Follow me on Instagram / Twitter / Facebook.

Work Chronicles
workchronicles.com

RNN configuration

Table 2 Configurable key parameters of the single-cell and sliding-window algorithms.

		Single-cell LSTM	Sliding-window LSTM	Sliding-window Vanilla RNN
Time inference	Receptive Field	∞	5	5
	Samples after deposit	5	4	4
RNN layer	Dimension	10	10	8
	Activation	tanh	tanh	ReLU
	Recurrent Activation	sigmoid	sigmoid	N/A
Dense layer	Dimension	1	1	1
	Activation	ReLU	ReLU	ReLU
Number of Parameters		491	491	89
MAC units		480	2360	368

Dot product implementations

Naive C++ implementation

```
for (int i=0; i < 8; i++){
  acc += a[i] * b[i];
}
```

ACC37 implementation

```
for (int i=0; i < 4; i++){
  tmp[i] = a[i]*b[i] + a[7-i]*b[7-i];
}
for (int i=0; i < 4; i++){
  acc += tmp[i];
}
```

ACC19 implementation

```
for (int i=0; i < 4; i++){
  tmp[i] = hls_fpga_reg(a[i]*b[i] + a[7-i]*b[7-i]);
}
for (int i=0; i < 4; i++){
  acc += tmp[i];
}
```

Stratix10 DSP