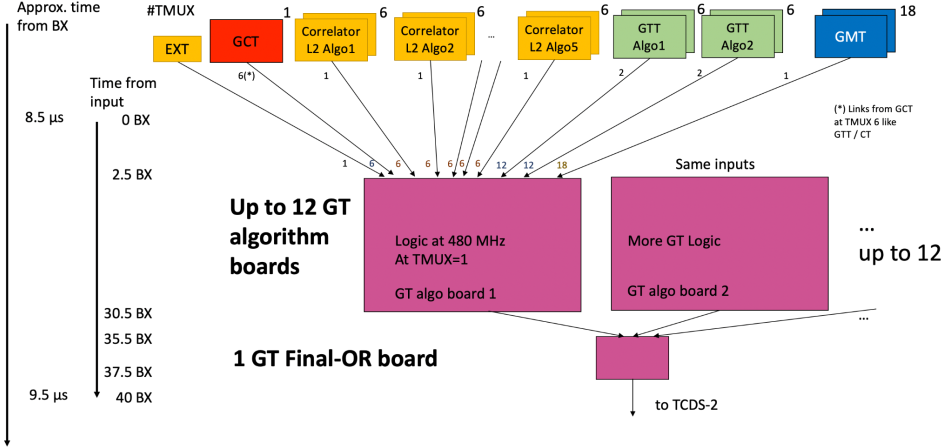
Use of topological correlations in ML-based conditions for the CMS Level-1 Global Trigger upgrade for the HL-LHC

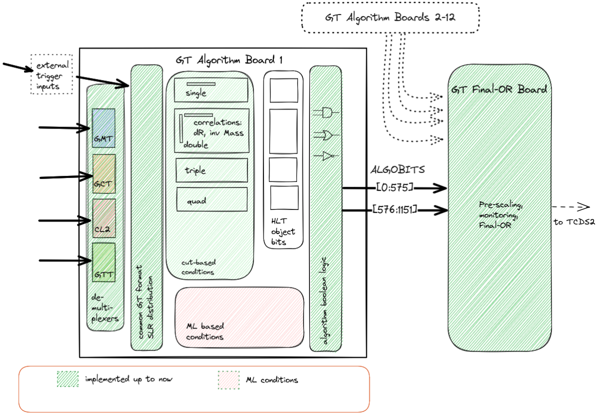
Gabriele Bortolato[1,2], Benjamin Huber[1,3], Elias Leutgeb[1,3,4], Dinyar Rabady[1], Hannes Sakulin[1] on behalf of the CMS Collaboration

[1]CERN, [2]Universitá degli Studi di Padova, [3] Technische Universität Wien, [3] Imperial College

CMS Phase-2 Global Trigger



CMS Phase-2 Global Trigger



A lot of work done by the GT team in the past years!

- Algorithms share the same input and output structure
- Fixed latency for all of them
- Each algorithm can be placed wherever we want in terms of board and Super-Logic-Region

Beyond the cut-based algorithms @ P2GT

Goals

- Implement ML algorithms in Global Trigger FPGAs
- Explore different ML algorithms architectures, Deep Neural networks, Boosted Decision Trees and so on

Hardware implementation constraints

- **Latency**: Algorithms must fit into ~8 BX (200 ns)
  - total budget 1 μs, part of it will be used by the existing GT infrastructure, time de-multiplexing logic and data transmission
- **Resources**: Up to 12 boards for the full Phase-2 menu
  - as of today we can fit ~1000 traditional cut-based algos in 3-4 Serenity boards equipped with VU13P FPGA parts (Virtex Ultrascale+)

FPGA	LUT [k]	FF [k]	DSP	BRAM [Mb]	URAM [Mb]	N SLR
VU13P	1,728	3,456	12,288	94.5	360.0	4
VU9P	1,182	2,364	6,840	74.9	270.0	3

P2GT NN development workflow

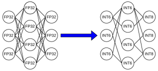
Step 1 : Model definition

Model definitoon and training with the commonly used frameworks.

TensorFlow dmlc XGBoost PyTorch

Step 2 : Optimizations

Hyperparemeter quantization, model compression via pruing and knowledge distilalition.



Step 3 : FPGA Porting

Python model transaltion to HLS and finally to HDL.

hls4ml Conifer AMD Vitis

Step 4 : Interfaces and Deploy

Interface layer to adjust input and output data formats.
Bitfile generation for the target FPGA.

AMD Vivado

Studied architectures

Binary Classifier

Pros

- Generally small footprint
- Straight forward training

Cons

- Need to train one model for each signal signature

Auto-Encoder

Pros

- 1 model to tackle different scenarios

Cons

- Model is very large
- Usually quite resource hungry
- Training not as straight forward as BC

Input variables

Candidate objects comes from different subsystems and up to 12 objects per collection are available every BX (40 MHz).

L1T Objects	L1T subsystem	Binary Classifier	Auto-Encoder
Jets	CL2	p_T, η	p_T, η, ϕ
Electrons	CL2	p_T, η, Iso, Qual	p_T, η, ϕ
Muons	GMT	p_T, η, Qual	p_T, η, ϕ
Photons	CL2	p_T, η, Iso, Qual	p_T, η, ϕ
Taus	CL2	p_T, η	p_T, η, ϕ
Missing energy	CL2	E_T^{miss}	E_T^{miss}, ϕ
HT and MHT	GTT	H_T, H_T^{miss}	H_T, H_T^{miss}
Invariant masses	GT	M_{ii}	M_{ii}

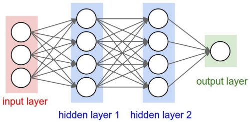
The invariant mass is computed at the GT level, more on that calculation later.

Binary classifiers

Two architectures are studied: Deep Neural Networks (DNN) and Boosted Decision Trees (BDT).

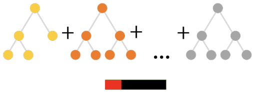
DNN

- Need to be pruned and quantized to be implementend in FPGA
- Uses DSPs to compute multiplications
- To better perform it needs a normalizer at the input stage



BDT

- Chain of logical decision
- No quantization or pruning is required
- No need of a normalizer at the input stage

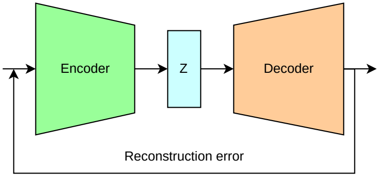


Auto-Encoder

Deep auto-encoder requires more inputs, add the φ variable as well.

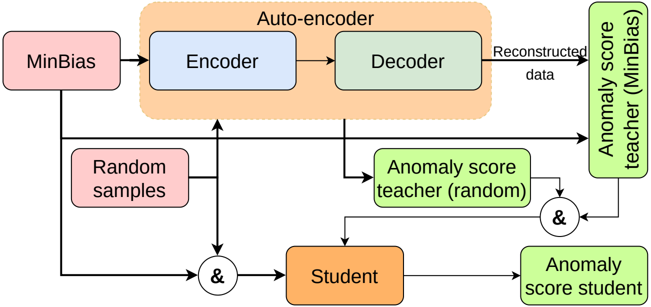
- Trained only with background events
- Encode the input in a smaller latent space and then decode it back to its original size

- Plain Auto-Encoder is too large to be implemented in the FPGA fabric, knowledge distillation is a must!

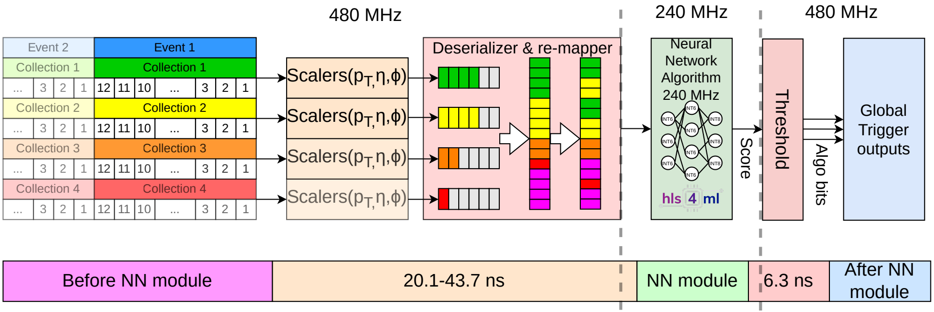


Knowledge distillation

- Such auto-encoders cannot be deployed on FPGAs due to the size and speed limitations
- Use another model (the student) and train it with the anomaly score computed with the AE (regression problem)



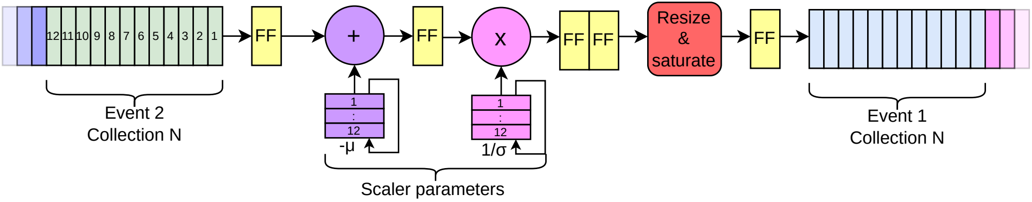
Neural Network pre-processing



Data are streamed at 480 MHz, parameters are updated on each clock cycle. Output resizing is applied to match the neural network fixed-point precision.

- Clock domain crossings from 480 MHz to 240 MHz and vice versa
- It uses one DSP per input variable (for the whole collection)

Neural Network pre-processing -scaler-



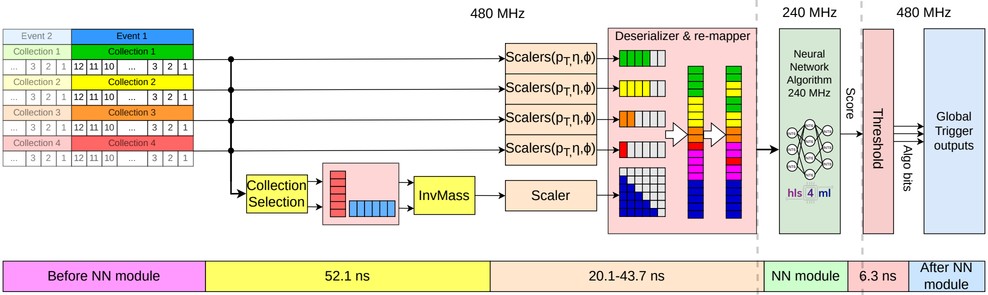
Data are streamed at 480 MHz, parameters are updated on each clock cycle. Output resizing is applied to match the neural network fixed-point precision.

- Clock domain crossings from 480 MHz to 240 MHz and vice versa
- It uses one DSP per input variable (for the whole collection)

Add invariant masses

Exploring the use of invariant mass.

- We already have the infrastructure to compute those (double object conditions)
- Just need a wrapper to prepare such variable for the hls4ml module
- At the hardware level we compute the square of the invariant mass, use the log2 to scale it down to reasonable range (easier to compute then the square root).



Invariant Mass computation

```
inv_mass_Mjj_comp_i : entity work.NN_invmass_calc
    generic map(
        collections  => (CL2Jets, CL2Jets),
        SELECTED_BXs => (others => 0)
    )
    port map(
        clk_algo          => clk_algo,
        rst_algo          => rst_algo,
        objects_valid_bx => objects_valid_bx,
        objects_bx        => objects_bx,
        invMass_o         => invMjj_val,
        valid_o           => Mjj_valid_out
    );
```

- Possibility to use any collections, e.g. Jets, Muons, ...
- Mathematical functions are stored in LUTs (*cosh*, *cos*)
- One collection is stored while the other one is streamed
- Result is a 12 objects vector at 480MHz (144 values per BX)
- Neglect the diagonal in the case of same collection

$$\frac{M^2}{2} = \underbrace{p_{T1} p_{T2}}_{\text{DSP}} (\underbrace{cosh(\Delta\eta)}_{\text{LUT}} - \underbrace{cos(\Delta\varphi)}_{\text{LUT}})$$

GT implementation

Some precautions needs to be considered if we want to implement these NNs:

- We could have other algos in the SLR! → routing congestion could arise
  Solution: Add some registers before and after the NN to help the router
- We are running at 480 (GT logic) and 240 (NNs) MHz → deal with timing violations
  Solution: Aggressive implementation strategies + Multi-cycle path constraints
- VITIS HLS complier doesn't have place & route knowledge → bigger NNs cannot be implemented
  Solution: Increased HLS compiler target frequency to 300 MHz

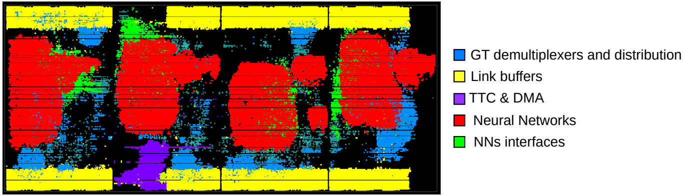
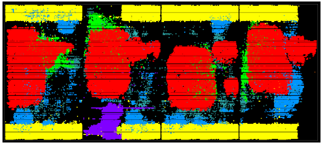


Figure: Full design floorplan (VU13P). 4 auto-encoders (student, red) with high level features as inputs. Pre-processing highlighted in green

Summary and Outlook

Summary

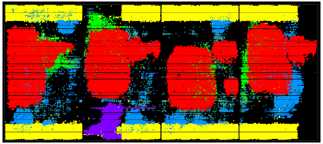
- Designed multiple small and mid-size deep NNs and BDTs
- Developed interface logic to integrate these into the P2GT firmware
- Invariant masses, isolation and quality
- Latency and resources usage under control
- Developed baseline strategy to meet timing closure



Summary and Outlook

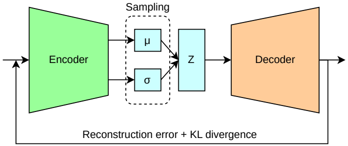
Summary

- Designed multiple small and mid-size deep NNs and BDTs
- Developed interface logic to integrate these into the P2GT firmware
- Invariant masses, isolation and quality
- Latency and resources usage under control
- Developed baseline strategy to meet timing closure



Outlook

- Finalize invariant mass wrapper module
- Implement Variation Auto-Encoder (similar to what is running now in the μGT)



BACKUP

P2GT latency break-down

-	BX	ns
Total	40	1000
Links	16	400
De-multiplexer	6	150
Int. BX delay	3	75
FinOR	$\sim$4	100
SLR distrib	$\sim$3	75
Algos	$\sim$8	200