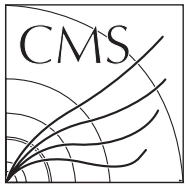
Available on CMS information server

CMS CR -2023/215



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

23 October 2023 (v3, 03 November 2023)

Design and implementation of Neural Network based conditions for the CMS Level-1 Global Trigger upgrade for the HL-LHC

Gabriele Bortolato, María Cepeda, Jaana Heikkilä, Benjamin Huber, Elias Leutgeb, Dinyar Rabady, Hannes Sakulin for the CMS Collaboration

Abstract

The CMS detector will be upgraded to maintain, or even improve, the physics acceptance under the harsh data taking conditions foreseen during the High-Luminosity LHC operations. In particular, the trigger system (Level-1 and High Level Triggers) will be completely redesigned to utilize detailed information from sub-detectors at the bunch crossing rate: the upgraded Global Trigger will use high-precision trigger objects to provide the Level-1 decision. Besides cut-based algorithms, novel machine-learning-based algorithms will also be included in the Global Trigger to achieve a higher selection efficiency and detect unexpected signals. Implementation of these novel algorithms is presented, focusing on how the neural network models can be optimized to ensure a feasible hardware implementation. The performance and resource usage of the optimized neural network models are discussed in detail.

Presented at *TWEPP2023 Topical Workshop on Electronics for Particle Physics*

Prepared for submission to JINST

Topical Workshop on Electronics for Particle Physics
2-6 October 2023
Geremeas, Sardinia, Italy

Design and implementation of Neural Network based conditions for the CMS Level-1 Global Trigger upgrade for the HL-LHC

G. Bortolato,[a,b,*] **M. Cepeda,**[c] **J. Heikkilä,**[d] **B. Huber,**[a,e] **E. Leutgeb,**[a,e] **D. Rabady,**[a] **H. Sakulin**[a] **on behalf of the CMS Collaboration**

[a] *Experimental Physics Department, CERN, 1211 Genève 23, Switzerland*
[b] *Department of Physics and Astronomy "Galileo Galilei", Padova University, Via Marzolo 8, 35131 Padova, Italy*
[c] *CIEMAT, Avda. Complutense 40, 28040 Madrid, Spain*
[d] *Universität Zürich, Winterthurerstrasse 190, 8057 Zürich, Switzerland*
[e] *Technische Universität Wien, Karlsplatz 13, 1040 Wien, Austria*

E-mail: gabriele.bortolato@cern.ch

Abstract: The CMS detector [1] will be upgraded to maintain, or even improve, the physics acceptance under the harsh data taking conditions foreseen during the High-Luminosity LHC operations. In particular, the trigger system (Level-1 and High Level Triggers) will be completely redesigned to utilize detailed information from sub-detectors at the bunch crossing rate: the upgraded Global Trigger will use high-precision trigger objects to provide the Level-1 decision. Besides cut-based algorithms, novel machine-learning-based algorithms will also be included in the Global Trigger to achieve a higher selection efficiency and detect unexpected signals. Implementation of these novel algorithms is presented, focusing on how the neural network models can be optimized to ensure a feasible hardware implementation. The performance and resource usage of the optimized neural network models are discussed in detail.

Keywords: Trigger concepts and systems (hardware and software); Trigger algorithms

[*] Corresponding author.

Contents

1 Introduction

The new CMS trigger system for the High-Luminosity LHC upgrade [2] will exploit detailed information from the calorimeter, muon and tracker subsystems at the bunch crossing rate. The final stage of the Level-1 Trigger apparatus, the Global Trigger (GT), will receive high-precision trigger objects from the upstream systems. Implemented in modern Field Programmable Gate Arrays (FPGA), it will determine the Level-1 decision based on a trigger menu consisting of more than 1000 trigger algorithms. The current system relies on cut-based algorithms that act on specific combinations of reconstructed particle properties. To reach higher selection efficiency and selection of unexpected signals, the upgraded GT will include also neural-network-based conditions. Implementing these neural-network-based conditions in the GT algorithm chain requires meeting stringent requirements in terms of latency and resources. The upgrade targets a total latency of 1 μs (40 Bunch Crossings, BX) for the entire GT. Three quarters of it is used by high speed serial links, demultiplexers, distribution and the Final-OR stage. Given neural networks (NN) are typically resource intensive, extensive optimization is required during and after training to ensure they can be integrated alongside the cut-based algorithms while meeting the target latency of ~10 BXs. Two different flavours of NNs are considered: deep binary classifiers and deep auto-encoders. To reduce the models' resource usage and latency, multiple optimizations have been applied. Some of these optimizations, such as synapse pruning, hyper-parameter quantization and precision tuning, can be performed without completely redesigning the model. However, others require a new model to be designed and trained from scratch. In this work a technique known as knowledge distillation was used to further reduce the resource usage of the final NN model.

2 Neural network model development

Deep binary classifiers and deep auto-encoder are studied. The primary purpose of the deep binary classifiers is to discern specific signal signatures, while the deep auto-encoders are designed to learn the unlabeled data and flag anything that deviates from it as anomalous. The latter rely on an

unsupervised learning technique, and in this particular case aim to understand efficient encoding of the feature of the well-understood physics scenarios ("background"). They aim to encode the input data into a lower-dimensional representation (latent space) and then decode it back to its original form, attempting to minimize the reconstruction error. As a result, any signature which differs substantially from the background will be reconstructed poorly. The distance between the input and the reconstructed event is then used as anomaly score. To compare the performances of deep binary classifiers and auto-encoders, we consider three different signal signatures denoted as A, B and C. Each unique signal signature will be associated with its own trained binary classifier, whereas a single auto-encoder will be trained using background events. Binary classifiers are trained with a mixture of signal and background events and supervised learning is used.

Hardware used for real-time inference in the Level-1 Trigger has limited computational capacity due to size and latency constraints. Incorporating resource-intensive models without a loss in performance poses a great challenge. Significant model compression is then necessary. Hyper-parameter and input/output precision quantization using qkeras [3] is employed to reduce the multiplication's complexity. Additionally, synapse pruning is implemented through Tensorflow model optimization [4]. These optimization processes occur during training and result in a reduction in model size by more than threefold compared to the uncompressed model implementation [5]. Despite the aforementioned compression methods, the auto-encoders typically remain too large to be implemented in FPGAs. To tackle this challenge, one more compression technique is harnessed: a basic implementation of knowledge distillation [6]. First, a bigger auto-encoder (referred to as the "teacher") is trained with only background events. A secondary, more compact model (referred to as the "student") is then trained to reproduce the teacher's anomaly score using the background events and random samples[1]. The anomaly score is computed as Mean Squared Error (MSE). Figure 1 depicts the training procedure for the two different approaches.

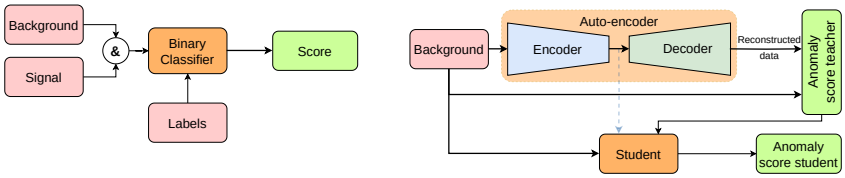


Figure 1. Left: In the supervised learning a model is trained knowing the output labels. Right: auto-encoders rely on unsupervised learning where the model is trained with only background events. With the knowledge distillation technique the student model is trained to behave like the teacher model.

Data pre-processing is performed with a normalization layer: the training dataset's variables are re-scaled to have a mean equal to zero and a standard deviation equal to one. Such re-scaling parameters are applied during training and in the hardware inference. In Table 1, input variables are listed for the two model topologies. P_T, η, ϕ are the representation of the candidate particle's momentum [7]. Since the usage of the ϕ variables does not result in notable increase in the signal efficiency in the case of binary classifiers, they are ignored during training. Binary classifiers feature a single hidden layer with 64 nodes with ReLU activations, the output is a single node with

[1] ϕ and η uniformly distributed and a decaying distribution for p_T and E_T^{Miss}, required to increase the phase space of the student's training dataset

Table 1. Input variable of the two model topologies.

L1T Objects	Binary classifier	Auto-encoder
First 6 jets	p_T, η	p_T, η, ϕ
First 4 electrons	p_T, η	p_T, η, ϕ
First 4 muons	p_T, η	p_T, η, ϕ
First 4 photons	p_T, η	p_T, η, ϕ
First 2 taus	p_T, η	p_T, η, ϕ
Missing energy	E_T^{miss}	E_T^{miss}, ϕ

a sigmoid activation function. The auto-encoder (teacher) features multiple hidden layers for the encoder part, a latent space with 7 nodes and a decoder with the inverted encoder's architecture. The student is a deep neural network with one output (the anomaly score). ReLU activations are used in the hidden layers and linear activation at the output.

Table 2. Relative performance of auto-encoder with respect to binary classifiers

Model	Hidden layers Architecture	Hyper-paremeter Quantization	Eff/Eff$_{BinaryBaseline}$		
			A	B	C
Keras Binary(A)	41→64→1	FP32	100.0%	-	-
hls4ml Binary(A)		<6/8,1/4>	98.8%	-	-
Keras Binary(B)	41→64→1	FP32	-	100.0%	-
hls4ml Binary(B)		<6/8,1/4>	-	99.0%	-
Keras Binary(C)	41→64→1	FP32	-	-	100.0%
hls4ml Binary(C)		<6/8,1/4>	-	-	94.4%
Keras AE (teacher)	Latent space:7 nodes	FP32	55.9%	70.0%	37.2%
Qkeras AE (student)	62→32/16/7/5→1	<8,1/2>	61.3%	72.0%	37.0%
hls4ml AE (student)			61.2%	72.0%	37.0%

To translate NN models into firmware, hls4ml [5], developed by the CMS community, is used. It translates high level description models (Keras/Qkeras) into synthesizable C code, which is then translated by the AMD VITIS High Level Synthesis compiler [8] into a VHDL module. This results in a block for integration into an FPGA design, and eventually, firmware can be built using AMD VIVADO [9]. In Table 2, signal selection efficiency at a given fixed rate for the three reference samples is compared between the binary classifiers and the auto-encoder. Keras models are trained with single-precision floating-point (FP32) without synapse pruning. In Qkeras and hls4ml models, different quantizations are applied for hidden and output layers, favoring higher bit precision in the output layer for enhanced performance. Hardware deployable models usually loose performance when quantization (fixed-point with 8/6 bits) and pruning (50%) are applied. In this work, Keras to hls4ml porting for binary classifiers incurs in less than 6% signal efficiency loss the target rate, which is small considering the reduction in the model size (see Section 4). The results presented above demonstrate that the auto-encoder is sensible to the signal samples, albeit with reduced efficiency compared to the binary classifiers, even though it was not trained with signal events.

3 Interface between the Global Trigger and neural networks

As was mentioned in Section 2, a pre-processing step is applied at the inputs of the NN models, where the re-scaling parameters have to be passed to the hardware. The mathematical operation

is described in eq. 3.1, where two passed parameters are the mean (μ) and the standard deviation (σ). Each object variable has its own set of parameters, resulting in 124 parameters in total for the student and 82 for the binary classifiers.

$$x' = \frac{x - \mu}{\sigma} \tag{3.1}$$

In the upgraded GT, within a BX, collections of up to 12 objects are streamed at 480 MHz so that these objects arrive at the algorithm logic, sequentially [10]. For each collection, the re-scale operation is implemented within a single DSP per variable. The pre-adder takes care of the mean subtraction, while the multiplier multiplies by the inverse of the standard deviation[2]. Its parameters are updated on each clock cycle to target different objects within the same collection. The process is depicted in Figure 2. A fully pipelined design with 5 stages has been chosen to reach a clock frequency of 480 MHz [11] and to reduce congestion.

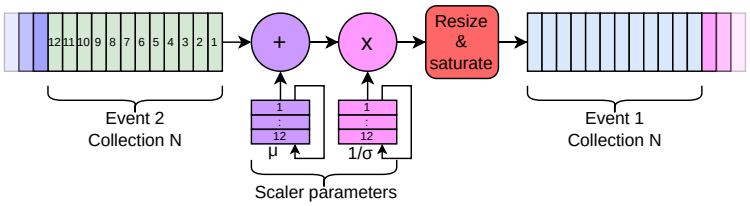


Figure 2. Re-scaler simplified architecture. Data are streamed at 480 MHz, parameters are updated on each clock cycle. Output resizing is applied to match the neural network fixed-point precision. If the result exceeds the selected range (positive or negative) the value saturates to the given limits. Register stages are inserted at the input, pre-adder, multiplier, result and resize & saturate steps.

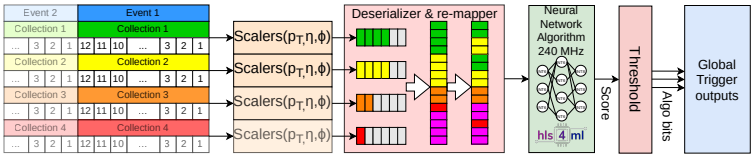


Figure 3. Interface with GT infrastructure. Data have to go through re-scaling, object selection, deserialization and bit-vector production. To extract algorithm-bits thresholds must be applied to the output score.

The interface between the Global Trigger infrastructure [10] and the neural networks consists of multiple layers. Data have to go through re-scaling, object selection, deserialization and bit-vector production. To extract algorithm-bits one or multiple thresholds must be applied to the output score. Such bits are then sent to the final decision logic. A diagram of the process is shown in Figure 3. A single input vector is injected every BX (25 ns, in the 240 MHz domain), while NN blocks run at 240 MHz. This allows the possibility to increase the Initiation Interval (II) up to 6, enabling the reuse of the multiplying logic within the dense layers. A reuse factor of 4 was selected for the student model which is a good compromise between DSP usage and latency, while a factor of 1 was selected for the binary classifiers thanks to their simpler architecture.

[2]Implementing division is rather complicated in hardware

4 Implementation of neural networks in the Global Trigger hardware

Each development step, if not managed properly, could lead to timing violations in the final hardware implementation. The hls4ml step inherits all the optimizations described in Section 2. During VITIS HLS compilation, the target clock frequency for the auto-encoder (student) was increased to 300 MHz to avoid possible timing violations, while 240 MHz was kept for binary classifiers. The clock uncertainty was increased to 33% for both. To relax any possible routing congestion within the NN block, the input vector is registered twice to allow the place and route process to focus on the NN itself rather than its external connections [12]. Crossing from 240 MHz back to 480 MHz at the output stage requires multi-cycle path constraints. Finally, timing constraints were met enabling most of the aggressive implementation strategies in VIVADO [9].

Table 3. Resource usage breakdown of the relevant modules. Uncompressed vs. compressed synthesizable models' size comparison. On the bottom, the resource usage of the re-scaler modules is shown.

Module	Clk	hls4ml (uncompr. keras)				hls4ml (compr. qkeras)			
	[MHz]	LUT [k]	FF [k]	DSP	lat[ns]	LUT [k]	FF [k]	DSP	lat[ns]
Auto-encoder	240	Non-synthesizable				42.0	15.5	301	70.8
Binary classifier (A)	240	44.1	24.0	1729	45.8	4.6	2.3	19	33.3
Binary classifier (B)	240	43.7	23.5	1766	45.8	4.6	2.3	19	33.3
Binary classifier (C)	240	45.5	23.6	1552	45.8	5.4	3.3	20	33.3

Module	Clk [MHz]	LUT [k]	FF [k]	DSP	lat [ns]
Re-scaler (AE)	480	1.1	3.3	17	22.9
Re-scaler (classifier)	480	0.8	2.2	11	22.9

A breakdown of the resource usage and latency is given in Table 3. Post Training Quantization (PTQ) was used for the uncompressed models, while Quantization Aware Training (QAT) was used for the compressed ones. The latency shown for the re-scaler modules comprises the normalization and the clock domain crossing (CDC) logic. The prototype firmware is implemented in a Serenity ATCA board [13] equipped with a Xilinx VU9P FPGA part with 3 Super Logic Regions (SLR). The prototype design features one auto-encoder (student) and the three binary classifiers, each replicated in all SLRs for a total of 3 auto-encoders (student) and 9 binary classifiers (Figure 4).

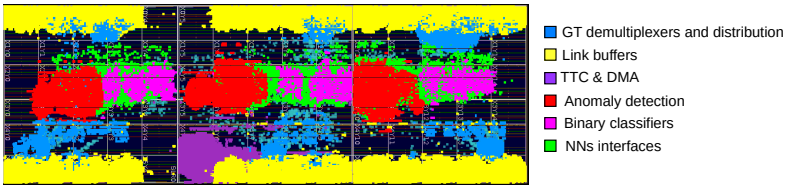


Figure 4. Full design floorplan, 3 auto-encoders (student) and 9 binary classifiers (three per signal signature). Infrastructure logic is highlighted in yellow (data regions) and purple (DMA and TTC). The full design uses 320k (27%) LUTs, 452k (19%) Registers, 723 (33%) Block RAM and 1290 (19%) DSPs.

First, data are read from the data link buffers, they are demultiplexed and distributed in the whole chip and finally are injected in the NN interfaces. Algorithm bits are written to the output channels and sent to the Final-OR board where monitoring, pre-scaling and masking take place [10].

5 Summary

The CMS Global Level-1 Trigger for Phase-2 features novel algorithms based on machine learning. In this study, we utilized quantization-aware training, pruning and knowledge distillation to compress the neural network models for the implementation in the FPGA fabric. Binary classifiers offer better performance on discerning known signal signatures with low latency and low resource usage with respect to auto-encoders, but a distinct model is needed for each signal type, requiring prior knowledge to generate the requisite dataset. Conversely, a single trained auto-encoder can be employed to detect known and unknown signatures, utilizing solely background events for its training. Furthermore, there is a notable difference in the final model sizes between the two approaches. Despite the inclusion of knowledge distillation, the auto-encoder (student) is approximately ten times larger than the binary classifier. Meeting timing constraints in this intricate architecture necessitated the employment of particular coding techniques [12] to guide the VIVADO implementation algorithm. Deep neural network models have been developed, evaluated and successfully tested in a Serenity prototype board.

References

[1] CMS Collaboration, *Development of the CMS detector for the CERN LHC Run 3*, *Journal of Instrumentation* (2023) [2309.05466].

[2] CMS Collaboration, *The Phase-2 Upgrade of the CMS Level-1 Trigger*, *Technical Design Report* (2020) .

[3] J. Coelho, Claudionor N., A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar et al., *Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors*, *arXiv e-prints* (2020) arXiv:2006.10159 [2006.10159].

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015.

[5] J. Duarte et al., *Fast inference of deep neural networks in FPGAs for particle physics*, *JINST* **13** (2018) P07027 [1804.06913].

[6] G. Hinton, J. Dean and O. Vinyals, *Distilling the Knowledge in a Neural Network*, pp. 1–9, 03, 2014.

[7] CMS Collaboration, *The CMS experiment at the CERN LHC*, *Journal of Instrumentation* **3** (2008) S08004.

[8] AMD, *Vitis High-Level Synthesis User Guide (UG1399)*, AMD (Oct, 2023).

[9] AMD, *Vivado Design Suite User Guide: Implementation (UG904)*, AMD (May, 2023).

[10] H. Sakulin et al., *Architecture and prototype of the CMS Global Level-1 Trigger for Phase-2*, *Journal of Instrumentation* **18** (2023) C01034.

[11] AMD, *Virtex UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics*, AMD (Jun, 2021).

[12] AMD, *UltraFast Design Methodology Guide for FPGAs and SoCs (UG949)*, AMD (Jun, 2023).

[13] A. Rose, D. Parker, G. Iles, O. Sahin, P.-A. Bausson, A. Tsirou et al., *Serenity: An ATCA prototyping platform for CMS Phase-2*, p. 115, 05, 2019, DOI.