Fast and resource-efficient Deep Neural Network on FPGA for the Phase-II Level-0 muon barrel trigger of the ATLAS experiment

Stefano Giagu[1,*] *on behalf of the ATLAS Collaboration*

[1]Dipartimento di Fisica, Sapienza Università di Roma and INFN Sezione di Roma, Roma, IT

Abstract. The Level-0 muon trigger system of the ATLAS experiment will undergo a full upgrade for the High Luminosity LHC to stand the challenging requirements imposed by the increase in instantaneous luminosity. The upgraded trigger system will send raw hit data to off-detector processors, where trigger algorithms run on a new generation of FPGAs. To exploit the flexibility provided by the FPGA systems, ATLAS is developing novel precision deep neural network architectures based on trained ternary quantisation, optimised to run on FPGAs for efficient reconstruction and identification of muons in the ATLAS "Level-0" trigger. Physics performance in terms of efficiency and fake rates and FPGA logic resource occupancy and timing obtained with the developed algorithms are discussed.

1 Introduction

The high-luminosity phase of the Large Hadron Collider (HL-LHC) at CERN is expected to start operation in 2027, to ultimately reach a peak instantaneous luminosity of $L = 7.5 \times 10^{34}$ cm^{-2} s^{-1}, corresponding to approximately 200 inelastic proton-proton collisions per bunch crossing, which delivers to the ATLAS experiment [1] more than ten times the total integrated luminosity collected in all previous LHC runs. Meeting these requirements poses significant challenges to the ATLAS trigger and DAQ systems to fully exploit the physics potential of the machine. To be able to handle the amount of data produced at peak luminosity by the HL-LHC the detectors of the ATLAS experiment will undergo a full upgrade. In particular, the Level-0 muon barrel trigger will be improved with the addition of a Resistive Plate Chamber (RPC) station (BI chamber) in the innermost radius of the Muon Spectrometer [2], and by moving the trigger logic off-detector, where flexible algorithms run on latest generation Field Programmable Gate Array (FPGA) processors [3]. Classification and regression methods based on modern Machine Learning (ML) are well suited to solve the limitations in terms of performance and flexibility of the conventional algorithms, and they can be a promising and viable solution to exploit the flexibility of FPGAs for real-time applications in the LHC detector triggers. In this work, we explore the implementation of deep convolutional neural networks in FPGAs based on trained ternary quantisation networks [4, 5], optimised to cope with the tight requirements in terms of resource usage ($\mathcal{O}(30\%)$) and latency ($\mathcal{O}(1\mu\text{s})$) imposed by the FPGA architecture and trigger constraints. We demonstrate that it is possible to

[*]e-mail: stefano.giagu@roma1.infn.it

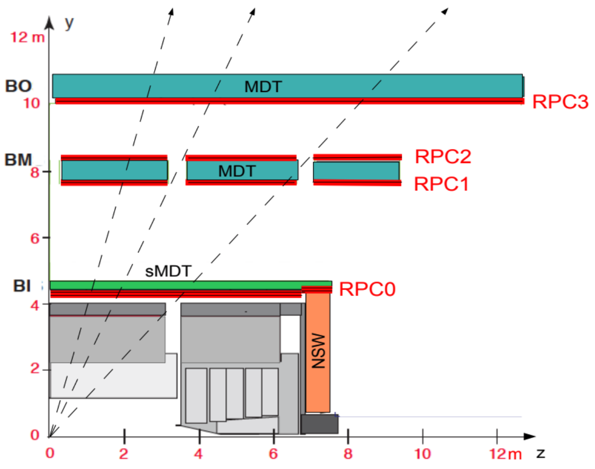

Figure 1. Sketch of a transverse section of the barrel region representing one of the sectors that contain a barrel toroid coil and its support structures. The four groups of RPC chambers (red) are shown as well as the MDT chambers (green and cyan) on the BI, BM, and BO stations. The three dashed lines represent muon trajectories traversing different combinations of RPC chambers [3].

reach state-of-the-art performance in muon reconstruction and identification in the ATLAS Level-0 muon trigger with microsecond latency.

1.1 Related work

ML inference on FPGAs has received increasing interest in recent years [6, 7], and several studies have been recently presented in the context of high energy physics (see for example: [8–10]). In this work, we get inspiration from these and other studies but follow an approach more oriented in achieving a robust and working implementation for a specific deep neural network architecture. In particular a ternary convolutional neural network with performance matching the requirements of the ATLAS Phase-II Level-0 muon trigger.

2 Conventional RPC-based trigger algorithm

A conventional Phase-II RPC-based trigger algorithm ("Standard Algorithm" in the following) has been implemented in the ATLAS simulation [3], and it is a direct extension of what has been used before the upgrade with the additional BI RPC station (see Figure 1). It operates as a pattern-finding algorithm, as illustrated in Figure 2. For each hit found in a predefined detector layer (usually the innermost layer), a coincidence window is opened toward the adjacent layer. The dimension of the window is inversely proportional to the transverse momentum (p_T) threshold of the trigger so that if the muon p_T is not high enough, the magnetic field will curve the particle outside the coincidence window of the next layer. If a new hit is found, the process is recursively repeated until all the layers are analysed. If the number of hits is greater than a given threshold, then the event is triggered. To compare performance with the deep neural network algorithm, the configuration which requires at least three hits out of four RPC stations is required. The Standard Algorithm is reliable and fast. However, some limitations in terms of robustness are observed. The geometrical acceptance of the coincidence window and configuration logic set an upper limit on the maximum efficiency.

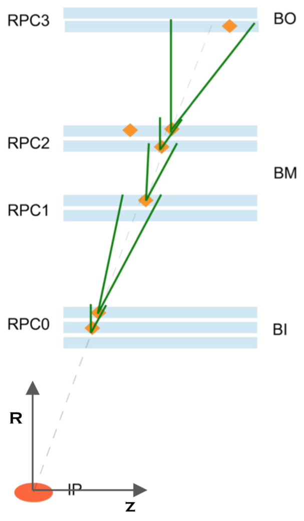

Figure 2. Illustration of the conventional RPC-based trigger algorithm. Hits are linked between consecutive planes based on predefined coincidence windows centred on a straight-line extrapolation from the nominal interaction point. Patterns of linked hits that satisfy quality requirements are selected as muon track candidates. The width of the coincidence windows defines the p_T threshold [3].

Moreover, the algorithm effectively measures the muon momentum from the deflection of the trajectory with respect to a straight line from the interaction point, limiting the possibility to trigger with high efficiency neutral long-lived particles decaying in muons.

3 Deep Neural Network approach

To overcome the limitations of the Standard Algorithm, we have used an ML-based approach based on the implementation of a Convolutional Neural Network (CNN) on an FPGA. CNN is a regularised version of multilayer perceptrons well known to excel in classification and regression task analysing visual imagery. As shown in Figure 3 [11], RPC detector strips can be arranged in image-like objects, to be fed to CNN as training inputs. Each bin of the vertical axis corresponds to a detector layer (3 detector layers for the inner station, 4 for the middle and 2 for the outer station). The horizontal axis maps the η coordinates of each physical RPC strip: for the i-th strip $\eta^i_{\text{bin}} = 384\frac{\eta^i-\eta^{\text{min}}}{\eta^{\text{max}}-\eta^{\text{min}}}$, where η^{max} and η^{min} are respectively the maximum ($\eta^{\text{max}} = 0.95$) and the minimum ($\eta^{\text{min}} = 0.07$) η values for the barrel RPC strips chosen to prevent muons from falling outside any layer of a specified sector; and 384 is a realistic number of strips per layer. This provides a convenient representation for the RPC hits data, in which an infinite momentum muon appears in the image as a vertical pattern of pixels, independently of the pseudorapidity η, while lower momentum muons appear ideally as inclined pixel patterns with slopes inversely proportional to the muon p_T.

Training data for the CNN is based on detailed Monte Carlo simulation of the ATLAS Phase-II detector, including realistic geometry, resolution effects, and cavern background evaluated from minimum bias events at HL-LHC peak luminosity conditions. Events with multiple muons are built by combining multiple single muons events with the cavern back-

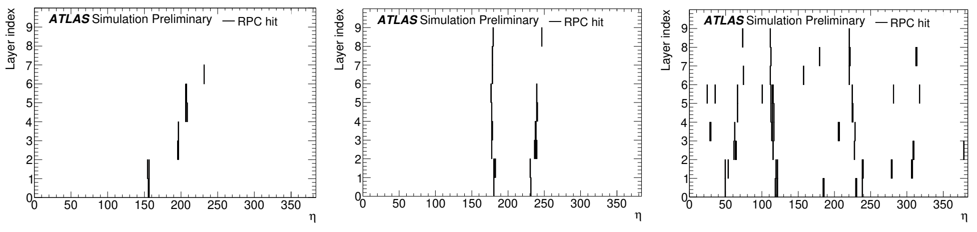

Figure 3. Examples of RPC event images used to train the CNN [11]: (left) an event with one low-p_T muon (p_T = 4 GeV); (center) an event with two high-p_T muons (15 and 12 GeV respectively); (right) an event with three muons and background noise due to pileup and cavern background.

ground. A total of one million images with muons in the range 3 – 20 GeV p_T has been used, divided between training, validation and testing sets.

Two CNN models have been trained in order to assess the performance of the new algorithm, one is a benchmark network based on state-of-the-art floating-point CNN implementation, based on a simplified VGG architecture [12], and the other one is based on a ternary-CNN (tCNN) [4, 5], that addresses the limited storage and computational resources imposed by the use of an FPGA. It can be realised by constraining weights and activation function in the network to be ternary-valued: +1, 0 and -1. The weights and neuron outputs in a tCNN can be represented using just two bits per weight instead of 32 as it would have been for a floating-point CNN, resulting in a 16-times larger compression in terms of memory occupation and simpler implementation in the FPGA firmware with better performance in term of latency.

The neural network architecture is illustrated in Figure 4. Convolutional [13] and Max-Pooling layers [14] have (4,3) and (4,1) kernels respectively, and the activation function for the hidden layers is ReLU [15] for the CNN and deterministic ternary activation for the tCNN, while a sigmoid activation [16] is used in the output layer in order to describe continuous values in output. Batch normalisation layers [17] with momentum are used in both the convolutional and fully connected stages. Both networks are trained to predict a five-component vector

$$(p_T^{\text{lead}},\ \eta^{\text{lead}},\ p_T^{\text{sub-lead}},\ \eta^{\text{sub-lead}},\ n^{\text{muons}}),$$

where "lead" stands for leading (i.e. the muon with the highest p_T) and n^{muons} represents the number of muons in an image. The MSE [16] loss function is minimised using the Adam algorithm [18] with an initial learning rate of 10^{-3} and a minibatch size of 64.

Performance of the models on test samples are shown in Figure 5 [11]. In Figure 5 (left), the trigger efficiency curves are reported. Cyan dots show the efficiency of the Standard Algorithm as a function of p_T, red squares the efficiency obtained with the reference benchmark CNN with floating-point weights, and blue triangles the efficiency obtained with the tCNN. The CNN always performs better than the Standard Algorithm (lower efficiency under the threshold and higher efficiency above the threshold). Similar results are obtained with the tCNN, which shows a reduction in the resolution in p_T manifested by a slower rise in the efficiency curve around the nominal threshold. The performance of the network in term of the number of muons identified by the tCNN vs the number of true muons reconstructed offline is reported in Figure 5 (right). Each column is normalised to unity. No trigger threshold is applied in calculating the table entries. By requiring a minimum p_T of 10 GeV, the numbers off-diagonal are further reduced, in particular, in the case where 0 muons are reconstructed in

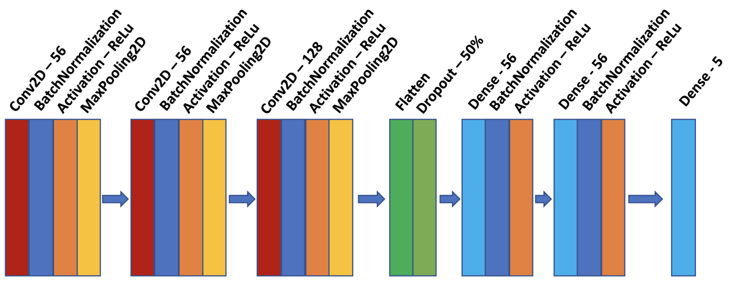

Figure 4. Schematic view of the network architecture that has been adopted. The numbers beside "Conv 2D" represent the number of filters. The numbers beside "Dense" represent the number of neurons of that layer.

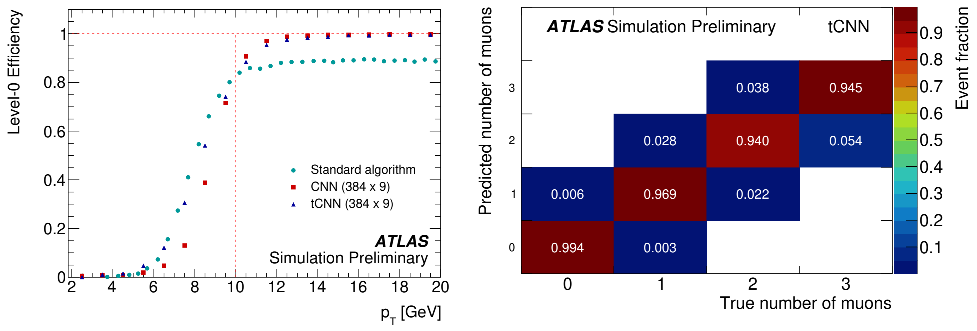

Figure 5. (Left) Trigger efficiency of different algorithms as a function of the reconstructed p_{T}. All the curves are realised taking single-muon images without a background as input and are tuned so that the efficiency reached at nominal threshold (10 GeV) is the same (82%) to ease the comparison. (Right) Confusion matrix of the predicted number of muons vs the true number of muons (i.e. offline reconstructed number of muons) [11].

the detector, and one muon is reconstructed by the network, the number decrease from 0.6% to 0.01%.

4 FPGA implementation

The implementation of the tCNN neural network model into the FPGA (Xilinx Virtex UltraScale+ XCVU13P) has been accomplished with two sequential phases. First, we have translated the model from the original Python form to a C++ code with a custom made tool[1]. During this phase, several techniques have been adopted in order to improve performance by tuning the trade-off between latency, throughput, and FPGA resource usage. In particular extensive C++ code modularisation has been used in order to reduce FPGA resource usage, while loop pipelining and vector partitioning have been implemented for latency reduction. In order to reduce the number of parameters of the neural network, the tCNN has been modified to process in parallel a predefined number of portions of the entire image. Passing a

[1]A public version of the tool is under development, check availability by contacting the authors.

Table 1. A summary of the HLS resource occupation of the implemented tCNN on a Xilinx Virtex UltraScale+ XCVU13P FPGA [11].

RAM	5%
Logic (LUT+Flip-Flops)	17%
DSP48E	11 %
Latency	1.1μs

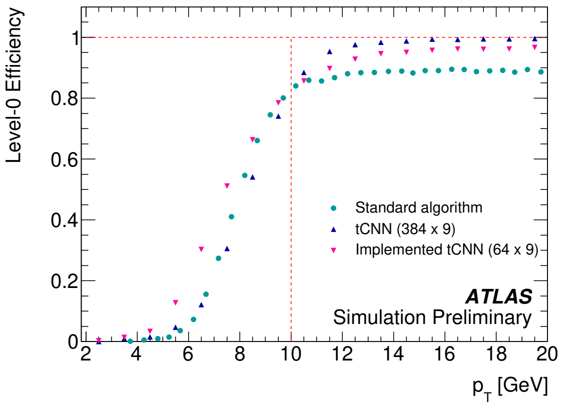

Figure 6. Trigger efficiency as a function of the reconstructed p_T for the implemented tCNN in comparison with the optimal tCNN and the Standard Algorithm [11].

smaller input to the tCNN largely reduces the total number of multiplications and therefore, memory occupancy and latency. For the final step, the translation from C++ code into VHDL code, we have used HLS, a tool developed by Xilinx [19]. FPGA latency and resource usage are reported in Table 1 [11].

The Table shows that the implemented tCNN reached the latency goal of 1 μs, and with a limited resource occupation of about 17%, to be compared with about 420 ns latency and 6% resource usage of the Standard Algorithm. However, these results have been obtained for the moment with a tCNN with a smaller number of parameters (about 1/10th of the optimal tCNN) limiting the expressive power and physics performance of the network. This is clearly visible in the efficiency curve reported in Figure 6 [11], where the implemented tCNN (purple triangles) shows a slightly lower plateau efficiency and, more importantly, a less steep turn-on with respect to the optimal tCNN. Nevertheless, the achieved initial performance is very promising, the implemented tCNN has already comparable performance with respect to the Standard Algorithm, and given the reduced resource utilisation there is large space for optimisation of the FPGA code synthesis, a work that is ongoing at the moment of the writing of these proceedings.

5 Conclusions

ML alternatives to conventional trigger algorithms have been studied for the Phase-II Level-0 muon barrel trigger of the ATLAS detector at the LHC. In particular, it has been shown that a deep neural network-based algorithm can be effectively implemented in the trigger FPGA, within the latency requirements of the ATLAS trigger, and with comparable or better performance with respect the ATLAS Standard Trigger algorithm. Work is ongoing on optimisation strategies and parameter tuning to synthesise the best performing tCNN into the trigger FPGA.

References

[1] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST, **3** S08003, (2008).

[2] ATLAS Collaboration, *Technical Design Report for the Phase-II Upgrade of the ATLAS Muon Spectrometer*, Technical Report CERN-LHCC-2017-017, ATLAS-TDR-026, (2017).

[3] ATLAS Collaboration, *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*, Technical Report CERN-LHCC-2017-020, ATLAS-TDR-029, (2017).

[4] M. Courbariaux, Y. Bengio, and J.-P. David, *Binaryconnect: Training deep neural networks with binary weights during propagations*. In Advances in Neural Information Processing Systems, pages 3123–3131, (2015).

[5] F. Li and B. Liu, *Ternary Weight Networks*, arXiv:1605.04711 [cs.CV], (2016).

[6] E. Nurvitadhi et al., *Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?*, Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 5-14, (2017).

[7] N. Suda, et al., *Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks*, Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, (2016).

[8] T. Boser, P. Calafiura and I. Johnson, *Convolutional neural networks for track reconstruction on fpgas*, NIPS 2017, (2017).

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

[10] N. Nottbeck, C. Schmitt, V. Büscher, *Implementation of high-performance, sub-microsecond deep neural networks on FPGAs for trigger applications*, JINST, **14**, P09014, (2019).

[11] ATLAS Collaboration, *L0 Muon Trigger Public Results*, (2019).

[12] K. Simonyan, A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv:1409.1556 [cs.CV], (2014).

[13] Y. LeCun, Y., *Generalization and network design strategies*. Technical Report CRG-TR-89-4, University of Toronto, (1989).

[14] Y. Zhou, and R. Chellappa, *Computation of optical flow using a neural network*. In Neural Networks 1988, IEEE International Conference on, pages 71—78. IEEE, (1988).

[15] X. Glorot, A., Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*. In AISTATS'2011, (2011).

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, (2016).

[17] S. Ioffe, S. Christian, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv:1502.03167 [cs.LG], (2015).

[18] D.P. Kingma, and J. Ba, *Adam: A method for stochastic optimization*. arXiv:1412.6980 [cs.LG] (2014).

[19] Xilinx, *Vivado Design Suite User Guide - High-Level Synthesis*, (2012).