Contents lists available at ScienceDirect

Nuclear Inst. and Methods in Physics Research, A

journal homepage: www.elsevier.com/locate/nima

Full Length Article

Nanosecond hardware regression trees in FPGA at the LHC

Pavel Serhiayenka [a], Stephen T. Roche [a,b], Benjamin T. Carlson [a,c], Tae Min Hong [a,*]

[a] Department of Physics and Astronomy, University of Pittsburgh, United States of America
[b] School of Medicine, Saint Louis University, United States of America
[c] Department of Physics and Engineering, Westmont College, United States of America

ARTICLE INFO

Keywords:
Data processing methods
Data reduction methods
Digital electronic circuits
Trigger algorithms
Trigger concepts and systems (hardware and software)

ABSTRACT

We present a generic parallel implementation of the decision tree-based machine learning (ML) method in hardware description language (HDL) on field programmable gate arrays (FPGA). A regression problem in high energy physics at the Large Hadron Collider (LHC) is considered: the estimation of the magnitude of missing transverse momentum using boosted decision trees (BDT). A forest of twenty decision trees each with a maximum depth of ten using eight input variables of 16-bit precision is executed with a latency of less than 10 ns using $\mathcal{O}(0.1\%)$ resources on Xilinx UltraScale+ VU9P — approximately ten times faster and five times smaller compared to similar designs using high level synthesis (HLS) — without the use of digital signal processors (DSP) while eliminating the use of block RAM (BRAM). We also demonstrate a potential application in the estimation of muon momentum for ATLAS RPC at the High-Luminosity LHC (HL-LHC).

1. Introduction

Deep architectures for machine learning (ML) methods continue to empower high energy physics experiments, such as the ATLAS [1] and CMS [2] experiments at the Large Hadron Collider (LHC) [3]. Various simplified approaches for ML designs aimed for trigger systems using field programmable gate arrays (FPGA) exist in the literature. Many use high level synthesis (HLS), in which C-like syntax is converted to an register-transfer level (RTL) circuit using a translator provided by the vendor. The HLS version of neural networks by hls4 ml [4] gave rise to a body of work in the implementation of artificial intelligence (AI), such as autoencoders and convolutional neural networks [5,6], on firmware. Similarly, the HLS version of decision trees [7,8] by `fwXmachina` (developed by us) also gave rise to an autoencoder [9]. There are also non-HLS approaches using hardware description language (HDL) for neural networks [10] and decision trees [11], as well as more dedicated applications, e.g., [12,13].

In this paper, we present a more efficient implementation of decision trees with lower latency. We illustrate the physics potential using the problem in our previous paper [14], the estimation of missing transverse momentum ($E_{\mathrm{T}}^{\mathrm{miss}}$) at the LHC. $E_{\mathrm{T}}^{\mathrm{miss}}$ is an important signature of physics with minimal interaction with the detector materials, including neutrinos and beyond the Standard Model (BSM) particles, including dark matter [15,16] and supersymmetry (SUSY) [17–19]. The High-Luminosity LHC (HL-LHC) [20] upgrades of the trigger systems of ATLAS and CMS experiments [21,22] provide an opportunity to implement AI/ML with deeper designs for a variety of applications.

This paper is organized as follows. Section 2 describes the ML training and firmware implementation on FPGA. Section 3 presents the results, with comparisons to previous publications. Section 4 concludes. Appendix A provides the technical details of the subscore adder. Appendix B presents a study, following Ref. [23], of the estimation of muon momentum at the HL-LHC using hits in the resistive plate chamber (RPC) subdetector in the ATLAS level-0 trigger system.

2. Method

The ML training is described followed by the firmware design. *T*MVA [24] is used to train a boosted decision tree (BDT) for regression with the "truth" generator-level $E_{\mathrm{T}}^{\mathrm{miss}}$ as the target variable. The setup follows our previous publication [8] and uses the data samples described therein. These variables include four calculations of the event $E_{\mathrm{T}}^{\mathrm{miss}}$ using calorimeter deposits, hadronic jets, tracks from charged particles, and four estimates of the total transverse energy deposited in the detector.

The firmware design is a VHDL adaptation of the Deep Decision Tree Engine (DDTE) originally designed in HLS [8]. In Python, the `fwXmachina` program writes VHDL that reflects a given ML configuration from the training. In Fig. 1, each tree is represented by a HDL Tree Engine (HTE) and the forest is managed by the HDL Tree Manager. The summing of the subscores is either clocked or combinational; see Appendix A. We note that operations are triggered by the rising edge

* Corresponding author.
E-mail address: tmhong@pitt.edu (T.M. Hong).

https://doi.org/10.1016/j.nima.2025.170209
Received 1 October 2024; Received in revised form 25 November 2024; Accepted 7 January 2025

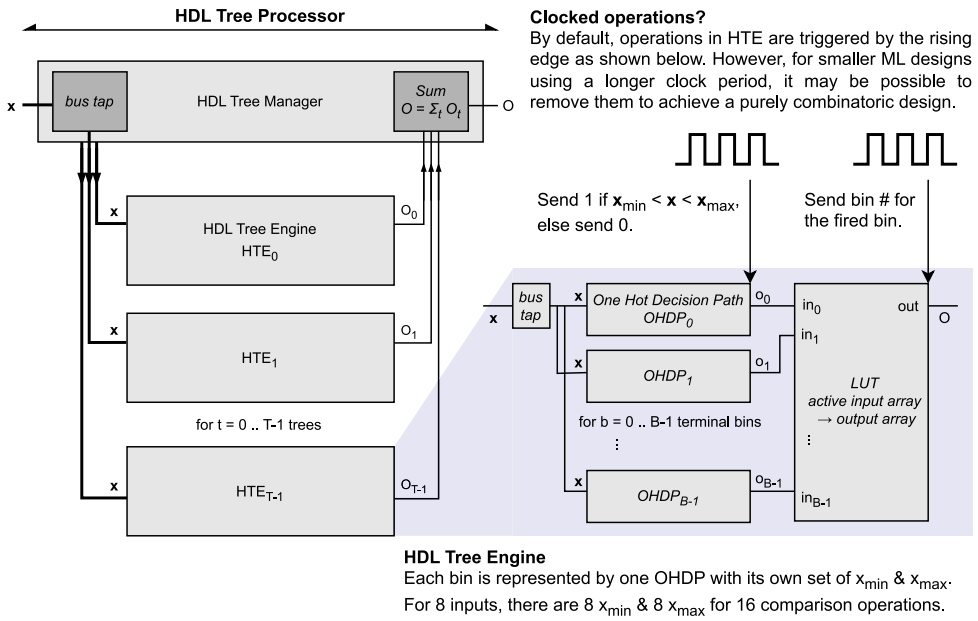

Fig. 1. Block diagram of the VHDL version of the Deep Decision Tree Engine (DDTE). Each tree is represented by HDL Tree Engine (HTE), which are composed of One Hot Decision Paths (OHDP) corresponding to Parallel Decision Paths (PDP) (from Figure 2 of Ref. [8]). The inputs are represented by a vector of values **x** and the outputs of the HTE by a single value O_t.

of the clock, but it may be possible to remove them for smaller designs and slower clock speeds.

The DDTE proceeds in two steps. The first step bins the inputs and returns a subscore for each tree represented by HTE in Fig. 1. For example, four trees produce four subscores. The second step combines the subscores to produce the final score. Multiplication and division are avoided in the firmware by pre-weighting each subscore by the tree's boost weight.

3. Results

The regression of $E_{\mathrm{T}}^{\mathrm{miss}}$ described in Section 2 is used as a benchmark model. Results of four configurations are presented, summarized in Table 1, followed by scaling properties based on one of the configurations. The configurations are

(1) $N_{\mathrm{tree}} = 40$ with $D = 6$ using the pipeline adder,
(2) $N_{\mathrm{tree}} = 10$ with $D = 8$ using the combinational adder,
(3) $N_{\mathrm{tree}} = 20$ with $D = 10$ using the combinational adder, and
(4) $N_{\mathrm{tree}} = 100$ with $D = 12$ using the pipeline adder.

where N_{tree} is the number of trees in the forest, and D is the maximum depth. We take this opportunity to define the number of bins for the forest N_{bin}, which has a value of up to 2^D. We note that due to the behavior of the BDT training, the number of bins does not necessarily correspond to a fully populated binary tree, so we define a quantity called "effective depth" d so that $2^d = N_{\mathrm{bin}}/N_{\mathrm{tree}}$ represents the number of bins of a fully populated decision tree of depth d. All configurations use $N_{\mathrm{var}} = 8$ input variables of 16-bits each, and a clock speed of 320 MHz. The testbench to reproduce this configuration is available online [25].

The first two configurations are identical to those reported previously [8], which used high level synthesis (HLS) to write the firmware. The latter two demonstrate the speed and efficiency of the new design. Configuration (1) yields a design that is 50% faster at 25 ns latency and 20 times smaller usage of flip flops, with similar look up table usage using a pipeline adder. Configuration (2) yields a design that is 5 times faster at about 20 ns latency and 7 times smaller usage of flip flops, 5 times smaller usage of look up tables, and eliminates the use of block RAM (BRAM) using a combinational adder. Configuration (3) has twice as many trees as configuration (2), but the design demonstrates the combinational adder's preference for a slower clock speed. Using twice as many trees but a clock speed of 200 MHz rather than 320 MHz, the latency is halved with similar resource usage; this result is noted in the abstract. Lastly, configuration (4) has an order of magnitude more trees than configuration (2), but the latency grows by less than 10 ns and the resource usage remains modest using a pipeline adder. No digital signal processors (DSP), BRAM, or URAM are used in the four configurations. For configurations (2), (3), and (4), the effective depth d is similar at approximately 7, while it is 5.4 for configuration (1).

Scaling properties are studied by changing one parameter at a time with respect to configuration (1) as the benchmark. The algorithm latency for configuration (1) is eight clock ticks. The latency can be as low as two clock ticks, but it grows logarithmically to ten clock ticks for 150 trees as shown in Fig. 2. The logarithmic dependence is due to the adder being arranged in a binary tree structure: two subscores are iteratively summed until a final score is reached. In contrast to the pipelined adder, the combinational adder always has a latency of 2 clock ticks. However, it has limited use cases as the model has to be relatively simple. For further discussion about the adder, we refer to Appendix A. Lastly, the interval period between successive inputs is one clock tick for all tested configurations, so the firmware can take in an input at every clock tick and generate an output after the algorithm latency.

Resource usage is considered as a function of N_{tree} in the forest and N_{bit} of each input variable. The dependence of look up tables (LUT) and flip flops (FF) on N_{tree} is expected to be linear as the combinational logic scales linearly. The left plot in Fig. 3 shows this to be roughly true until about eighty trees, but the dependence seems to flatten out afterwards. For the number of bits, we observe a linear relationship for both LUT and FF on the right plot of Fig. 3, as expected. The separate study for the combinational adder was not done, but we anticipate that usage results would be similar to the pipeline design because only the clock is removed.

4. Conclusions

The improved ML architecture for FPGA presented in this paper demonstrate improvements in the resource utilization on FPGA and

Table 1
FPGA results and comparison with Refs. [7,8,11]. All results in the table uses the same FPGA model Xilinx Ultrascale+ VU9P (vu9p-flgb2104-2L-e) with the following available resources 1.2 M LUT, 2.4 M FF, 6.8 k DSP, and 4.3 k BRAM. Effective depth d is defined as so that $2^d = N_{\text{bin}}/N_{\text{tree}}$.

Goal	5 classif'n	2 classif'n	$E_{\text{T}}^{\text{miss}}$ regression		$E_{\text{T}}^{\text{miss}}$ regression			
Reference	[11]	[7]	[8]		This paper			
Setup								
Design	VHDL	HLS	HLS	HLS	VHDL	VHDL	VHDL	VHDL
Sum strategy	–	–	–	–	pipeline	combin.	combin.	pipeline
Parallelize	–	cutwise	pathwise	pathwise	pathwise	pathwise	pathwise	pathwise
Clock (MHz)	250	320	320	320	320	320	200	320
Bit precision	fixed_{18}	int_8	int_{16}	int_{16}	int_{16}	int_{16}	int_{16}	int_{16}
N_{var}	16	4	8	8	8	8	8	8
N_{tree}	100	100	40	10	40	10	20	100
Max. depth D	4	4	6	8	6	8	10	12
N_{bin}	–	–	1.7 k	1.4 k	1.7 k	1.4 k	2.9 k	15.7 k
Effective depth d	–	–	5.4	7.2	5.4	7.2	7.2	7.3
Config. label	–	–	(1)	(2)	(1)	(2)	(3)	(4)
Notable			identical		identical		slower clock	larger forest
Results								
LUT	96 k	1 k	6.4 k	75 k	5.1 k	10 k	15.5 k	38 k
FF	43 k	0.1 k	35 k	24 k	1.6 k	4.7 k	6.6 k	19.4 k
DSP	0	2	0	0	0	0	0	0
BRAM	0	5.5	0	10	0	0	0	0
URAM	–	0	0	0	0	0	0	0
Latency (ns)	52 ns	9.375 ns	38 ns	119 ns	25 ns	19 ns	10 ns	28 ns
" (tick)	13	3	12	38	8	6	2	9
Interval (tick)	1	1	1	1	1	1	1	1
Notable			benchmark				in abstract	

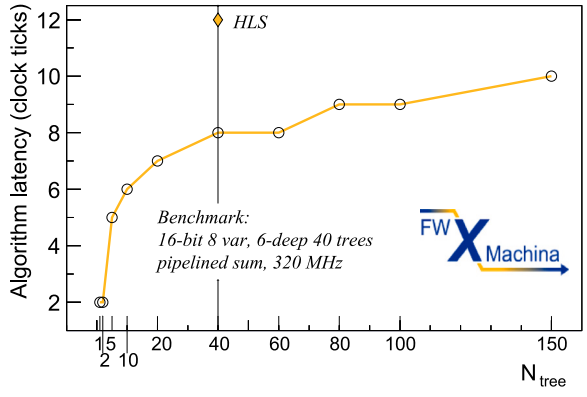

Fig. 2. Algorithm latency scaling vs. N_{tree}. The scaling is done with respect to the parameters stated on the plot, corresponding to the benchmark configuration in Table 1. Diamond represents the HLS results [8].

lower latency. These tools may be of use in constrained environments such as in the trigger systems of ATLAS [21] and CMS [22], which are being upgraded for the High-Luminosity LHC to account for the substantial increase in the average number of simultaneous proton collisions per bunch crossing, reaching values of up to 200 [20].

CRediT authorship contribution statement

Pavel Serhiayenka: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Data curation. **Stephen T. Roche:** Writing – review & editing, Software, Data curation. **Benjamin T. Carlson:** Writing – review & editing, Writing – original draft, Funding acquisition, Data curation. **Tae Min Hong:** Writing – original draft, Visualization, Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare competing interests. TMH, BTC, and STR have filed a patent on the firmware design of the decision tree with the University of Pittsburgh. It is currently pending as US Patent Application Publication No. US 2024/0054399. Other authors declare no competing interests.

Acknowledgments

We thank Connor Marsh for assistance with FPGA data collection. We thank Elliot Lipeles for discussions regarding tails of neural network distributions. We thank Yuvaraj Elangovan and Isabelle Taylor of the Electronics Shop. Work performed in the University of Pittsburgh Dietrich School Electronics Shop Core Facility (RRID:SCR_025113) and services and instruments used in this project were graciously supported, in part, by the University of Pittsburgh, United States. TMH was supported by the US Department of Energy [award no. DE-SC0007914]. BTC was supported by the National Science Foundation, United States [award no. 2209370]. STR was supported by the Emil Sanielevici Scholarship.

Appendix A. Adder details

Two firmware designs are implemented to execute the addition of the tree subscores. The first is combinational, which organizes addition into a tree structure, independent of the clock, as shown in the right diagram of Fig. 4. The second technique is a pipeline, which also organizes addition into a tree structure but with clocks inserted in between each level of the adder. Flip flops are used to separate the addition operations into distinct, clocked stages as shown in the left of Fig. 4. This results in an adder that is dependent on the clock. Only one level of the adding process is computed during each clock tick, which results in as many clock ticks are there are levels.

The advantage of the combinational adder is speed. Once the adder receives a set of subscores, it will begin to compute the sum immediately from start to finish. The disadvantage is that it cannot support

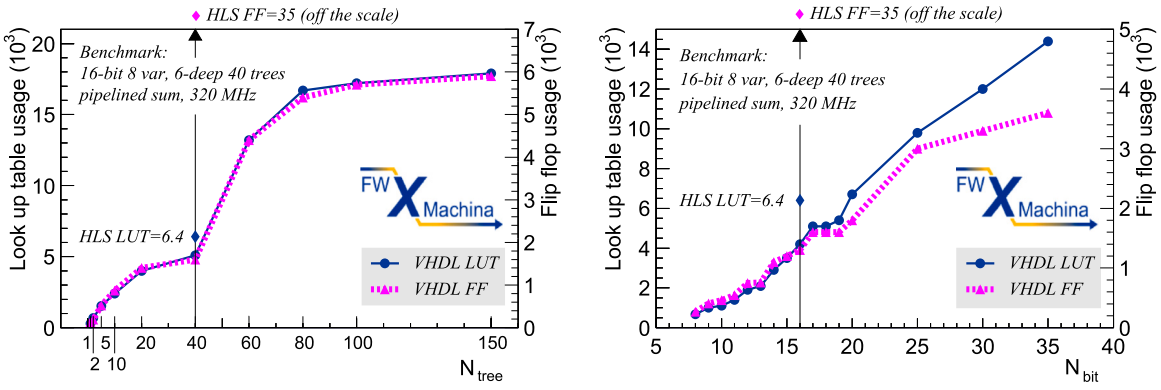

Fig. 3. Resource usage scaling vs. N_{tree} (left) and N_{bit} (right). On each plot, look up table usage is shown in circles with the scale on the left side and flip flop usage is shown in triangles with the scale on the right side. The scaling is done with respect to the parameters stated on the plot, corresponding to the benchmark configuration in Table 1. Diamond represents the HLS results [8], with flip flop usage being off the scale and noted as such with the arrow.

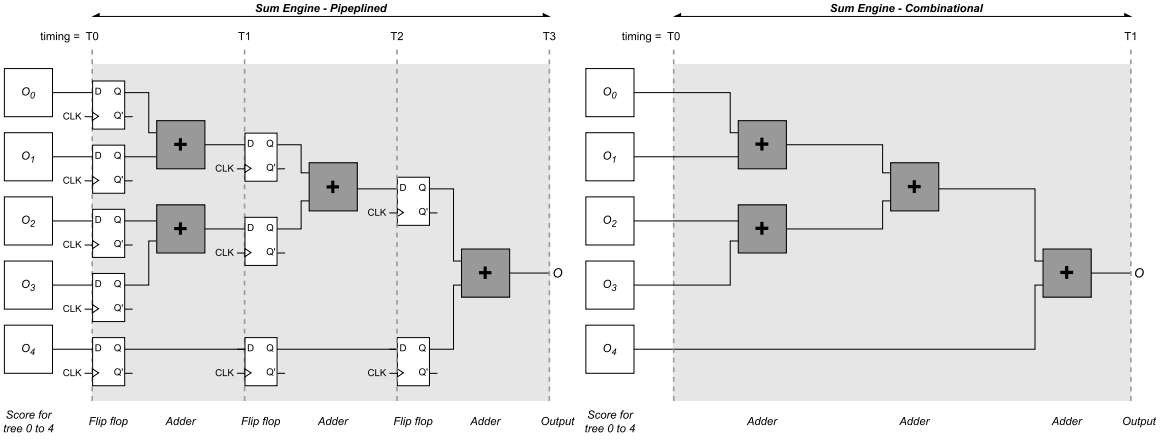

Fig. 4. Two adder designs using pipeline (left) and combinational logic (right) for the sum block in Fig. 1.

a large number of trees without timing violations, as it assumes that all the trees' scores may be added within a single clock tick. Because the pipeline adder distributes the subscore addition process into steps, assuming all other parameters are equal, it can operate for a larger number of trees than the combinational. Though, the pipeline adder does have upper limits of operation, these limits are much higher than those of the combinational. The disadvantage of the pipeline adder is that it will have a higher latency compared to the same size design using a combinational adder. While the combinational latency is always 2 ticks, the pipeline latency is represented by 2 plus the ceiling integer of $\log_2$ of the number of trees, i.e., latency in clock ticks $= 2 + \lceil \log_2 N_{\text{trees}} \rceil$. Therefore, combinational may be desired with pipeline reserved for scenarios where timing violations occur.

Appendix B. Muon momentum for ATLAS RPC at HL-LHC

We present an application of ML regression for muon momentum estimation in the ATLAS level-0 trigger system for HL-LHC upgrade, which detects muons with resistive plate chamber (RPC) detector in the barrel section [26,27]. There are three cylindrical 2 cm-wide RPC layers; the trajectory of a muon passing between them can be used to determine its charge and p_{T} as they are immersed in a toroidal magnetic field.[1] Ref. [12] presents an HDL implementation of neural network (NN) regression on FPGA to estimate the p_{T} using hit information. We apply our BDT to this problem following the detailed description by the same authors [23].

A sample of 200 k muon candidates were generated in a idealized RPC system of uniformly distributed p_{T} between 3 and 30 GeV using the publicly available code in Ref. [29]. Half of the sample is positively charged and the other half negatively charged. A small fraction of the sample (0.1%) consists of noise hits from non-muon detector background. For each muon candidate, the z-coordinate of the clusters at RPC1, RPC2, and RPC3 are recorded, labeled as z_1, z_2, and z_3, respectively. The three input variables to the regression algorithm are z_2, Δz_3, and Δz_1 with the latter two defined as the coordinate relative to z_2 as $z_3 - z_2$ and $z_2 - z_1$, respectively. The two Δz distributions are shown in Fig. 5 and they qualitatively agree with Figure 3 of Ref. [23].

The ML setup is described for the BDT as well as for the NN to reproduce results of Ref. [12]. For both methods, the sample is split evenly into training and testing sets. For the BDT, a forest of 50 trees at a maximum depth of 7 is trained using adaptive boosting (AdaBoost)

[1] An interesting and related problem in ML-based FPGA muon tracking was done for thin gap chambers (TGC) detectors located outside of the toroidal magnetic field [28], but is beyond the scope of this paper.

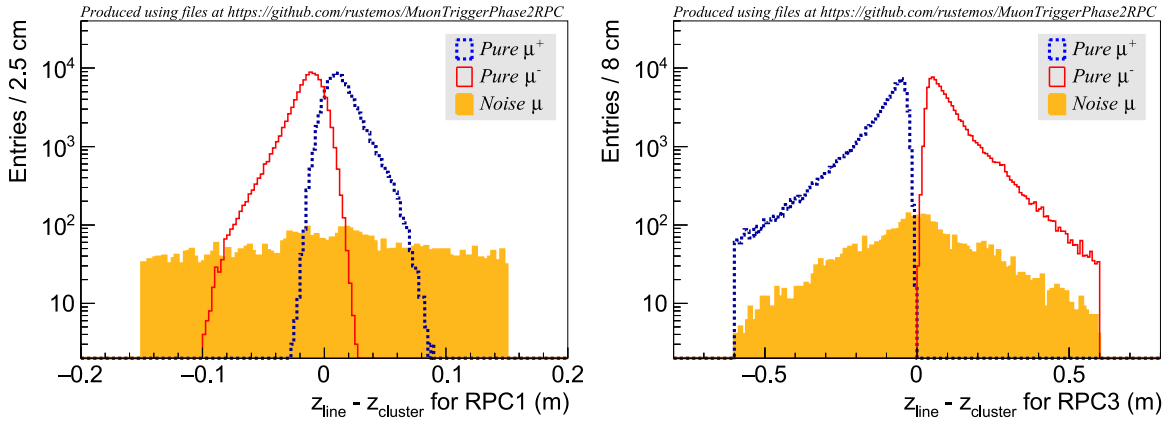

Fig. 5. Distribution of the differences of z coordinate between the impact point of the seed line and the cluster position in the RPC1 (left) and RPC3 (right). Muons are separated by charge and noise fakes. The sample is produced using the code available at Ref. [29].

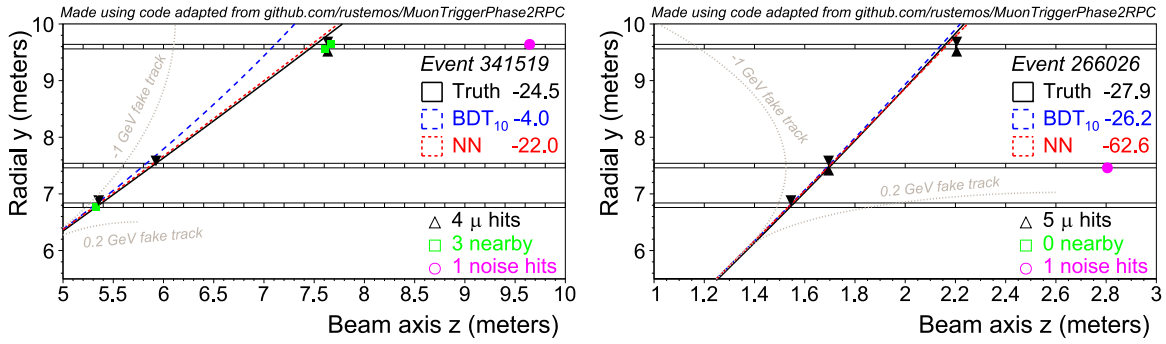

Fig. 6. Event displays showing the y-z projection of the RPC hits with projected tracks using the p_T starting at the origin. The top plot shows the case where NN gives a good estimate at $r = 0.9$ and BDT an erroneous one at $r = 0.2$, where $r \equiv (q \cdot p_T)^{\text{reco}}/(q \cdot p_T)^{\text{truth}}$. The bottom plot shows the case where BDT gives a good estimate at $r = 0.9$ and NN an erroneous one at $r = 2.2$. As a sanity check, we overlay two fake tracks (unrelated to the hits in the display) with $q \cdot p_T = -1$ GeV and 0.2 GeV to visually demonstrate the curvature dependence on $q \cdot p_T$. Muon hits and nearby hits, excluding noise hits, are used to compute the BDT and NN scores. The BDT configuration for these event displays uses 100 trees with a maximum depth of 10.

trained on the untransformed variables and set to target the product of charge and p_T: $q \cdot p_T$. We note that linear transformations on input variables may incur additional latency and resource usage so we avoid them. As done in our previous papers, floating point values are used for training and 16-bit integers are used for evaluation.

For the NN, we follow the process as was done in Ref. [29]. The network is trained on linearly transformed variables — i.e., scaled and shifted to have a null mean and unit standard deviation — and set to target the charge divided by its p_T: q/p_T. The NN is trained with the default setup of Ref. [12]; floating point values are used in training. For testing, we use floating point numbers to evaluate the physics performance, which is in contrast to the 16-bit fixed point numbers used in Ref. [12]. The floating point is used to simplify the analysis, which may provide a slight improvement for the performance of the NN with respect to the fixed point, as it has been shown that using fixed point numbers results in diminished network quality [4].

The physics performance is evaluated by considering the resolution of the estimated charge times p_T, defined as the ratio of the reconstructed and the truth value, i.e., $r \equiv (q \cdot p_T)^{\text{reco}}/(q \cdot p_T)^{\text{truth}}$. The ratio is chosen as metric for comparing the BDT to the NN because it can be evaluated using a sample of simulated muons. Two events displays are shown in Fig. 6 to illustrate a case where the NN gives a good estimate whereas the BDT gives an erroneous one, and vice versa. The p_T value of the tracks are relatively high enough where the curvature is difficult to see by eye, so two fake tracks with opposite charge and very low p_T are shown to demonstrate the dependence on the curvature from $q \cdot p_T$. Lastly, we note that efficiency turn-on curves are not shown, as a realistic sample of the background composition would be needed to determine the trigger threshold value corresponding to a given background rate.

The physics results are summarized by the r distributions in the left column of Fig. 7. The resolution r is shown twice in each plot, once using hits only from the muon and nearby hits and once using those hits and the noise seed. For the results without noise hits, r is peaked at unity for both BDT and NN with standard deviation of 0.116 and 0.130, respectively. The profile plot of the standard deviation normalized by the mean defined as $s_r \equiv \text{stddev}_r/\text{mean}_r$ shows different patterns for the BDT and NN. The BDT shows a relatively flat dependence of about $s_r = 10\%$ for the full p_T range. In contrast, the NN shows a smaller value of about $s_r = 4\%$ at 3 GeV that grows linearly with p_T to about $s_r = 20\%$ at 30 GeV. For the results with noise hits, r is peaked at both 1 and -1, which signifies that the charge designation is incorrect at least half of the time, with a much broader spread of its distribution. The NN distribution with noise hits seem to be broader than the one for BDT, so the graphical axis is expanded with underflow and overflow. We note that these studies rely on the mock-up of the ATLAS RPC, so a more realistic considerations could improve the results.

The trade-off between physics and engineering performance is a consideration when implementing the algorithm on the FPGA. A simpler ML design would be faster and smaller at the cost of worse

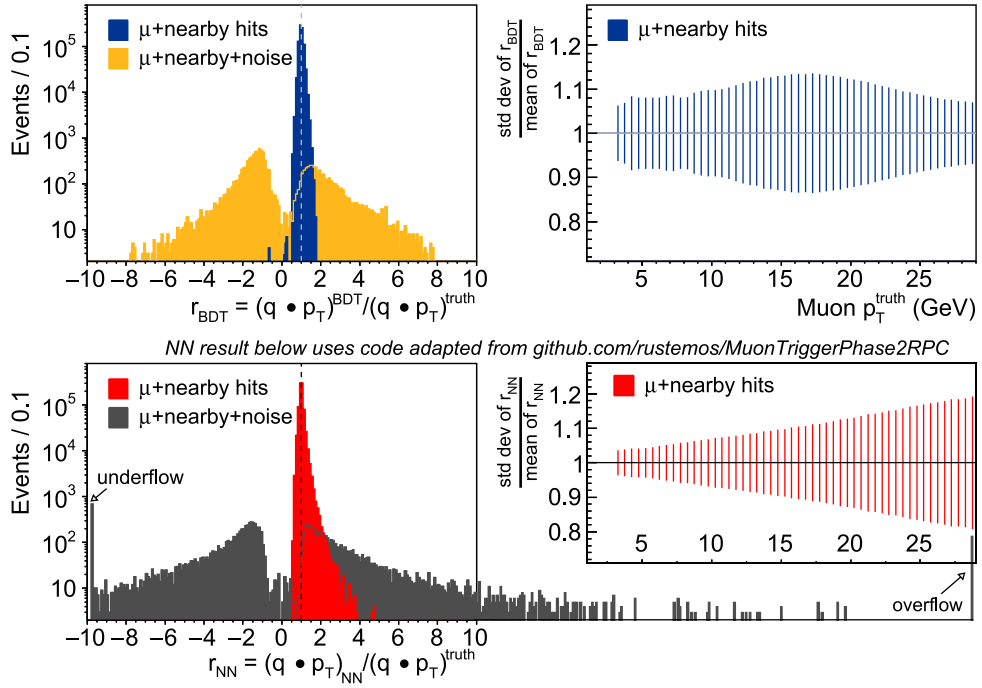

Fig. 7. (Left column) Distributions of the resolution of muon p_T defined as the ratio with respect to truth p_T. (Right column) Profile plots of the standard deviation normalized to the mean defined as $s_r \equiv \text{stddev}_r/\text{mean}_r$ is shown in slices of truth p_T. (Top row) Results are using the BDT configuration corresponding to Table 2. (Bottom row) Results are using the NN configuration described in the text. For these plots, nearby hits are included with the "hits from μ".

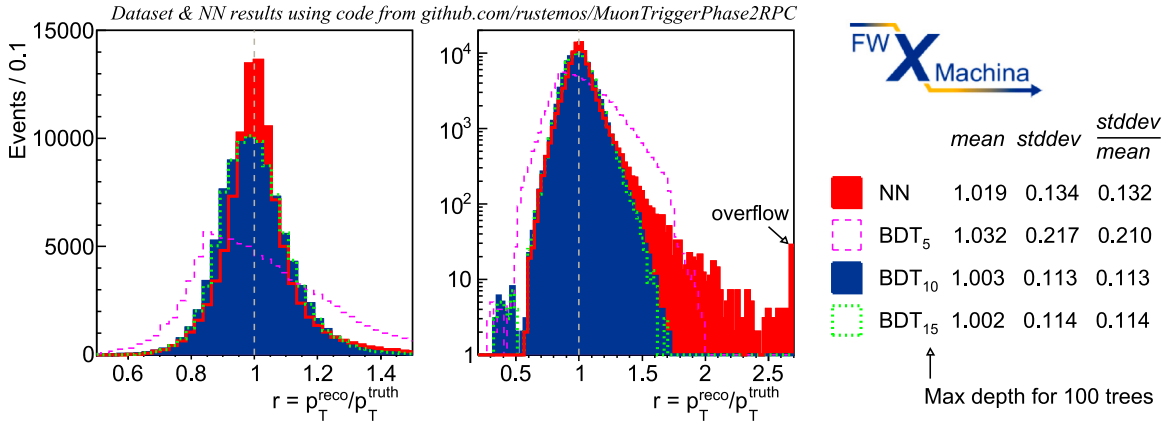

Fig. 8. Distribution of the resolution of muon p_T for the NN and BDT of varying maximum depths with 100 trees. The same distribution is shown twice with linear (left) and logarithmic axes (right).

resolution r. We studied the dependence by varying the maximum depth D—5, 10, 15—for hundred trees corresponding to BDT_5 BDT_{10}, and BDT_{15}, respectively. As done above, we consider s_r for each configuration. The NN value is $s_r = 0.13$ and is smaller than $s_r = 0.21$ obtained for the relatively shallow BDT_5. A deeper configuration of BDT_{10} yields a competitive value of $s_r = 0.11$, but no further improvement is seen for even deeper configuration as BDT_{15} yields the same value. The full table of values are given in Fig. 8. The figure shows the distributions in linear and logarithm scales. The former highlights the relative sharpness of the NN near unity with respect to BDT_{10}, whereas the latter shows the longer tail of the NN. One typical event where $r = 0.2$ for BDT_{10} and another event where $r = 2.2$ for the NN are shown in the event displays of Fig. 6.

Finally, the FPGA results are presented for a BDT of thirty trees with a maximum depth of seven; also summarized in Table 2. This combination is chosen to be close to the physics performance of the neural network. The resource utilization is about 11 k of look up tables and 4 k of flip flops with no DSP, BRAM, or URAM usage. The algorithm latency is 22 ns, which corresponds to 7 clock ticks at 320 MHz. The interval between successive inputs is 1 clock tick. The look up table and flip flop usage is comparable to the NN, but our design does not use DSP due to lack of multiplication operations. We note that the NN timing results are from simulated values whereas our results are from the actual values measured on the board, so further differences may arise with NN measurements after implementation.

Data availability

Data is available in the link given in ref [25].

Table 2
FPGA results for muon momentum estimation for ATLAS RPC at HL-LHC.

Method	Neural network [12]		BDT [this work]	
Setup				
Design	Verilog		VHDL	
Xilinx model	Kintex UltraScale XCKU060		Virtex UltraScale+ VU9P	
Clock	320 MHz		320 MHz	
Variable precision	fixed_{16}		int_{16}	
Input N_{var}	3		3	
Target variable	q/p_T		$q \cdot p_T$	
ML method	Neural network		Boosted decision tree	
Configuration	3 hidden layers		30 decision trees	
	30 nodes per layer		Maximum depth of 7	
	Fully connected layers		Boosting strategy: adaptive	
	ReLU activation function		Sum strategy: pipeline	
Results				
LUT	4659	(1.4%)	11.3 k	
FF	7370	(1.1%)	3.8 k	
DSP	157	(5.7%)	0	
BRAM	Not reported		0	
URAM	Not reported		0	
Latency	122 ns	(simulated)	22 ns	(measured)
″	39 ns	(simulated)	7 ticks	(measured)
Interval	8 tick	(simulated)	1 tick	(measured)

References

[1] ATLAS Collaboration, The ATLAS experiment at the CERN large hadron collider, J. Instrum. 3 (2008) S08003.

[2] CMS Collaboration, The CMS experiment at the CERN LHC, J. Instrum. 3 (2008) S08004.

[3] L. Evans, P. Bryant, LHC machine, J. Instrum. 3 (2008) S08001.

[4] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran, et al., Fast inference of deep neural networks in FPGAs for particle physics, J. Instrum. 13 (07) (2018) P07027.

[5] E. Govorkova, E. Puljak, T. Aarrestad, T. James, V. Loncar, M. Pierini, A.A. Pol, N. Ghielmetti, M. Graczyk, S. Summers, et al., Autoencoders on field-programmable gate arrays for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider, Nat. Mach. Intell. 4 (2022) 154–161.

[6] N. Zipper, Testing a neural network for anomaly detection in the CMS global trigger test crate during run 3, J. Instrum. 19 (03) (2024) C03029.

[7] T.M. Hong, B. Carlson, B.R. Eubanks, S.T. Racz, S. Roche, J. Stelzer, D.C. Stumpp, Nanosecond machine learning classification with deep boosted decision trees in FPGA for high energy physics, J. Instrum. 16 (08) (2021) P08016.

[8] B. Carlson, Q. Bayer, T.M. Hong, S. Roche, Nanosecond machine learning regression with deep boosted decision trees in FPGA for high energy physics, J. Instrum. 17 (09) (2022) P09039.

[9] S. Roche, Q. Bayer, B. Carlson, W. Ouligian, P. Serhiayenka, J. Stelzer, T.M. Hong, Nanosecond anomaly detection with decision trees and real-time application to exotic higgs decays, Nat. Comm. 15 (2024) 3527.

[10] N.P. Ghanathe, A. Madorsky, H. Lam, D.E. Acosta, A.D. George, M.R. Carver, Y. Xia, A. Jyothishwara, M. Hansen, Software and firmware co-development using high-level synthesis, J. Instrum. 12 (01) (2017) C01083.

[11] S. Summers, et al., Fast inference of Boosted Decision Trees in FPGAs for particle physics, J. Instrum. 15 (2020) P05026.

[12] R. Ospanov, C. Feng, W. Dong, W. Feng, S. Yang, Development of FPGA-based neural network regression models for the ATLAS Phase-II barrel muon trigger upgrade, Eur. Phys. J. Web Conf. 251 (2021) 04031.

[13] J. Gonski, Development of the ATLAS liquid argon calorimeter readout electronics and machine learning for the HL-LHC, Instruments 6 (3) (2022) 28.

[14] ATLAS Collaboration, Performance of the missing transverse momentum triggers for the ATLAS detector during Run-2 data taking, J. High Energy Phys. 08 (2020) 080.

[15] CMS Collaboration, Search for invisible decays of a higgs boson produced through vector boson fusion in proton–proton collisions at $\sqrt{s} = 13$ TeV, Phys. Lett. B 793 (2019) 520–551.

[16] ATLAS Collaboration, Search for invisible higgs-boson decays in events with vector-boson fusion signatures using 139 fb^{-1} of proton–proton data recorded by the ATLAS experiment, J. High Energy Phys. 08 (2022) 104.

[17] ATLAS Collaboration, Searches for electroweak production of supersymmetric particles with compressed mass spectra in $\sqrt{s} = 13$ TeV pp collisions with the ATLAS detector, Phys. Rev. D 101 (5) (2020) 052005.

[18] ATLAS Collaboration, Search for new phenomena in events with an energetic jet and missing transverse momentum in pp collisions at $\sqrt{s}$ =13 TeV with the ATLAS detector, Phys. Rev. D 103 (11) (2021) 112006.

[19] ATLAS Collaboration, Search for nearly mass-degenerate higgsinos using low-momentum mildly displaced tracks in pp collisions at $\sqrt{s}$=13 TeV with the ATLAS detector, Phys. Rev. Lett. 132 (22) (2024) 221801.

[20] G. Apollinari, O. Brüning, T. Nakamoto, L. Rossi, High luminosity large hadron collider HL-LHC, CERN Yellow Rep. (5) (2015) 1–19.

[21] ATLAS Collaboration, Technical design report for the phase-II upgrade of the ATLAS TDAQ system, 2017, CERN-LHCC-2017-020, ATLAS-TDR-029.

[22] CMS Collaboration, The phase-2 upgrade of the CMS level-1 trigger, 2020, CERN-LHCC-2020-004.

[23] R. Ospanov, C. Feng, W. Dong, W. Feng, K. Zhang, S. Yang, Development of a resource-efficient FPGA-based neural network regression model for the ATLAS muon trigger upgrades, Eur. Phys. J. C 82 (6) (2022) 576.

[24] A. Hoecker, P. Speckmayer, J. Stelzer, et al., TMVA - toolkit for multivariate data analysis, in: CERN-OPEN-2007-007, 2007, [physics/0703039].

[25] T.M. Hong, P. Serhiayenka, Xilinx inputs for nanosecond hardware decision trees for missing transverse energy, 2024, D-Scholarship@Pitt #46983, available online at http://d-scholarship.pitt.edu/id/eprint/46983.

[26] R. Cardarelli, R. Santonico, A. Di Biagio, A. Lucci, Progress in resistive plate counters, Nucl. Instrum. Methods Phys. Res. A 263 (1) (1988) 20–25.

[27] ATLAS Collaboration, Performance of the ATLAS RPC detector and level-1 muon barrel trigger at $\sqrt{s} = 13$ TeV, J. Instrum. 16 (07) (2021) P07029.

[28] C. Sun, T. Nakajima, Y. Mitsumori, Y. Horii, M. Tomoto, Fast muon tracking with machine learning implemented in FPGA, Nucl. Instrum. Methods A 1045 (2023) 167546.

[29] R. Ospanov, MuonTriggerPhase2RPC: python code for toy simulation of ATLAS RPC trigger, 2024, available online at http://github.com/rustemos/MuonTriggerPhase2RPC, commit tag `76ec993548d45a03e36d296a19e40baee71f3f24`, (Accessed 13 September 2024).