Neural Network-Based Primary Vertex Reconstruction with FPGAs for the Upgrade of the CMS Level-1 Trigger System

C. Brown[1], A. Bundock[2], M. Komm[3], V. Loncar[3], M. Pierini[3], B. Radburn-Smith[1], A. Shtipliyski[1], S. Summers[3], J.-S. Dancu[1], and A. Tapper[1] for the CMS Collaboration

[1] Imperial College London (Blackett Laboratory, Prince Consort Rd, South Kensington, London, SW7 2BW, United Kingdom)

[2] University of Bristol (HH Wills Physics Laboratory, Tyndall Ave, Bristol BS8 1TL, United Kingdom)

[3] CERN (CH-1211 Geneva 23, Switzerland)

E-mail: cebrown@cern.ch

Abstract.
The CMS experiment will be upgraded to maintain physics sensitivity and exploit the improved performance of the High Luminosity LHC. Part of this upgrade will see the first level (Level-1) trigger use charged particle tracks reconstructed within the full outer silicon tracker volume as an input for the first time and new algorithms are being designed to make use of these tracks. One such algorithm is primary vertex finding which is used to identify the hard scatter in an event and separate the primary interaction from additional simultaneous interactions. This work presents a novel approach to regress the primary vertex position and to reject tracks from additional soft interactions, which uses an end-to-end neural network. This neural network possesses simultaneous knowledge of all stages in the reconstruction chain, which allows for end-to-end optimisation. The improved performance of this network versus a baseline approach in the primary vertex regression and track-to-vertex classification is shown. A quantised and pruned version of the neural network is deployed on an FPGA to match the stringent timing and computing requirements of the Level-1 Trigger.

1. Introduction

The HL-LHC will produce up to 200 simultaneous proton-proton interactions per bunch crossing (pile-up) in the CMS detector. While most proton-proton interactions are inelastic, a hard scatter, which reveals the interactions CMS aims to probe, is far rarer making the identification of this primary interaction key for triggering. Due to the increased Pile-Up (PU), the CMS Level-1 (L1) Trigger is to be upgraded [1] and novel algorithms are being developed to maintain the physics sensitivity of the detector. Part of the L1 Trigger upgrade is to introduce track finding, which will use outer tracker modules [2] to reconstruct tracks with a transverse momentum (p_{T}) $> 2\,\mathrm{GeV}$. This information can be used to separate the Primary Vertex (PV) from PU. The main downstream user of the PV is Pile-Up Per Particle Identification (PUPPI) [3] which will perform calculations on the tracks associated to this vertex. This makes the PV regression and the association of tracks to this vertex important to utilise the physics performance of the PUPPI algorithm while reducing the impact of PU [1]. As with the current system, the L1 trigger will be implemented on custom hardware running FPGAs with strict resource limitations. Thanks to larger front-end buffers, the latency will be increased to 12.5 µs which when combined with more powerful FPGAs allows for more complex algorithms to be used.

While simple histogramming and cut-based methods can lead to effective vertexing strategies and are well within the latency budget they do not take into account all the information from the track finder and so are susceptible to non-genuine tracks, called fakes, and the differences in track parameter resolution in different regions of the detector [4]. Modern Deep Neural Networks (DNNs) are able to find optimal solutions from low-level information such as track features thus skipping the lengthy development processes of more traditional approaches. Tools such as hls4ml [5] and QKeras [6] allow these DNNs to be compressed to fit in FPGA hardware.

2. Baseline Approach

The PV is the location of the hard proton-proton scatter in an event. Offline, it is defined as the reconstructed vertex with the highest sum of track p_{T}^2 [7].

The baseline approach to vertex finding bins all tracks in z_0 weighted by their p_{T} in a 256-bin histogram spanning a z_0 range of -15 to 15 cm, as is shown in Fig. 1. Where z_0 is defined as the distance of a reconstructed track from the beamspot, along the beam line. A three-bin window is then passed across this histogram to find the three consecutive bins with the highest combined p_{T}. The centre of the middle of these three bins is returned as the PV. While this method is fast (a latency of 30 clock cycles at 360 MHz) and has low resource usage, it has some key issues. The first is the lack of correction for the degradation in z_0 resolution for high η tracks, which leads to a worsening of the resolution of the PV. Secondly, it does not account for high p_{T} fakes which, when associated with clusters of PU tracks, can appear as high p_{T} vertices.

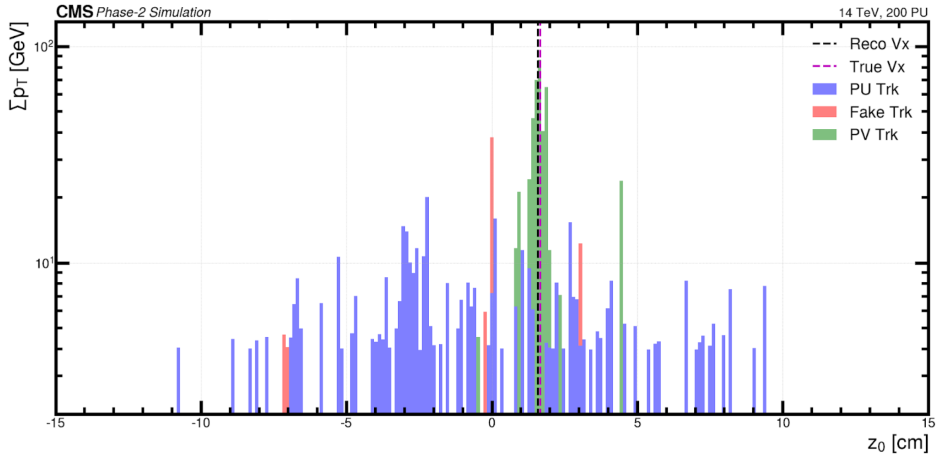

Figure 1. A single simulated event of a hadronic $t\bar{t}$ decay with PU of 200 showing all tracks histogrammed in z_0 weighted by p_{T} and coloured by their track type. Also shown is the reconstructed PV using the baseline approach and true generator PV.

The baseline approach to track-to-vertex association uses an η-dependent window in z_0 around the PV. This is reasonably effective with a true positive rate (correctly assigning PV tracks to the PV) of 91% and a false positive rate (assigning either a PU or fake track to the vertex) of 10%. Again, this method is fast and simple but fails to take into account more complex track features, such as the quality of the track fit, and is therefore heavily dependent on the resolution of the tracks it is provided.

3. End-to-End Neural Network Approach

The end-to-end Neural Network (NN) approach uses the same concepts as the baseline approach and expands on them with interconnected neural networks that are trainable end-to-end as shown in Fig. 2. Instead of weighting by p_{T} in a 256 bin histogram, a three layer DNN is used to learn an ideal weight function per track from input track features. The features used are the track p_{T}, η and the output of a BDT trained to distinguish non-genuine and real tracks [8] (labelled as MVA in eq. 1). These learned track weights are then used in combination with the track's z_0 to fill a histogram. This histogram is used as the input to a 1D convolution of kernel size three, depth one and stride one. The convolved histogram is passed through an ArgMax to obtain the bin position with the largest entry, as in the baseline approach.

Instead of a cut-based approach to track-to-vertex association a three layer DNN is used, which uses the same input track features as the weight network and additionally the distance from the PV to the track in z_0. Using a SoftMax final output activation, a likelihood that a track belongs to the PV is returned.

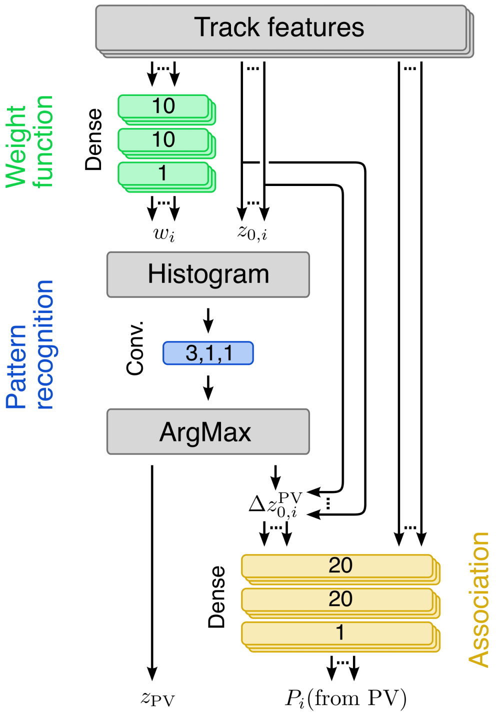

Figure 2. End-to-end network architecture showing the three distinct networks in colour as well as the position of the histogram layer and ArgMax.

3.1. Back-Propagation

The end-to-end network is trained in one cycle with a two part loss function. The first is a Huber loss [9] for the event level regression of the PV versus the true generator-level vertex. The second is a binary cross entropy loss that is used at the track level comparing the output track-to-vertex association probability to the simulation truth track label. These two losses are equally weighted.

Part of the end-to-end network is a histogram that is filled with a learnt track weight, which is convolved and the peak found. This contains two custom operations where the differential of the loss function with respect to the network weights are needed. The first is the histogram where each bin h_i has the input of the learnt weight w and the track's z_0. The bins are filled as:

$$h_i = \sum_{j=0}^{tracks} \delta(j \in \text{bin } i)\; w(p_{T,j}, \eta_j, MVA_j) \tag{1}$$

resulting in the following gradients

$$\frac{\partial h_i}{\partial z_0} = 0 \quad \text{and} \quad \frac{\partial h_i}{\partial w} = \sum_{j=0}^{tracks} \delta(j \in \text{bin } i) \tag{2}$$

which are implemented as custom TensorFlow [10] operations.

The second part of the PV regression is the peak finding of a convolved histogram. This in a forward pass is simply an ArgMax operation that finds the index of the highest member of a 256 element vector. However, in order to back-propagate the regression loss function the differential of this with respect to its inputs is needed which, for a standard ArgMax, is undefined. Instead, a soft ArgMax is used which combines a SoftMax, a linear layer, and a final sum to find the ArgMax of the input vector. The soft ArgMax of a vector x with N total elements is defined as:

$$\sum_{i=0}^{N} i \frac{e^{x_i/T}}{\sum_{j=0}^{N} e^{x_j/T}} \tag{3}$$

where T is a tuned hyperparameter of the network which allows this layer to return an approximate one-hot encoding.

4. Performance

The end-to-end approach outperforms the baseline approach in several key metrics. The first is the PV regression. Figure 3 shows the NN approaches in red and blue outperform the baseline in black especially in the tails of the residual where the improved filtering of fake tracks has reduced the number of high p_{T} clusters appearing to be the PV. Secondly, the NN outperforms the baseline approach in assigning tracks to this PV. The receiver operating characteristic (ROC) curve in Fig. 4 demonstrates that for a fixed false positive rate of 10 % the NN approach has a true positive rate of 96 % versus the baseline rate of 91 %.

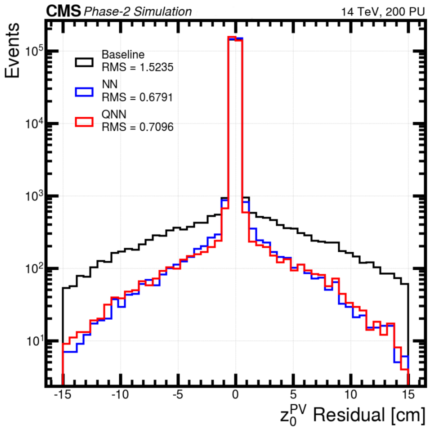

Figure 3. True PV - Reconstructed PV for the Baseline and NN approaches. NN refers to the floating point approach while QNN is the quantised approach described in Section 5.

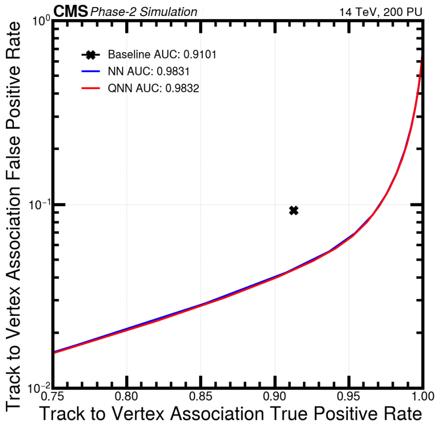

Figure 4. Receiver Operating Characteristic (ROC) curve for the Baseline and NN approaches to track to vertex association. Shown as true positive rate versus false postive rate.

5. Firmware Implementation

The hls4ml [5] package is used to realise this network in FPGA firmware. As the network has custom histogram layers, these were not converted with hls4ml but instead existing VHDL firmware from the baseline approach was reused. This means the network is split into three parts when it is converted: a weight network that takes input tracks and outputs a learnt weight; a pattern network that convolves the histogram created from the tracks; and an association network that outputs a probability the track is from the vertex, these seperate networks are highlighted in Fig. 2. As the latency budget is small, parts of the network will be implemented multiple times to exploit the parallelism of the L1 architecture, notably the weight and association networks that work on a track-by-track level and so will be replicated 18 times. The replication of elements of the network means the size in FPGA resources of the partial networks is critical for their use in firmware. A variety of tools were used to reduce the size of the network. Firstly, regularization introduced a loss function that penalizes the absolute value of the weights [11]. Secondly, pruning iteratively removed weights close to zero to remove unnecessary weights, keeping the overall network size small [12]. Finally, quantization aware training using the QKeras [6] package uses fixed point numbers for network parameters with restricted bitwidths, which, when passed to hls4ml, reduced the required resources for the network.

The final resource usages for a Xilinx UltraScale+ VU9P with a clock frequency of 360 MHz are shown in Table 1. Both an unquantised and quantised version of each part of the network are shown, demonstrating the effectiveness of quantised aware training and pruning of the networks, especially in reducing the Digital Signal Processor (DSP) usage which is the limiting factor in these FPGAs. Also shown in Figs. 3 and 4 is the performance of the full quantised network in red, demonstrating no loss in performance when moving from an unquantised to quantised network.

Table 1. Resource usage and latencies of a Xilinx VU9P running at 360 MHz for the floating point Neural Network (NN) and the quantised and pruned version (Q) with their expected number of replications. Also included is the baseline approach, the NN approaches are additional to these resources and latency as they use existing parts of the baseline firmware. These resource usages are estimates from a Vivado synthesis of the networks and the latencies from a C-Simulation.

Network	Latency (ns)	Initiation Interval (ns)	LUTs %	DSPs %	BRAMs %	FFs %
NN (Q) Weights	22 (14)	2.7 (2.7)	2.52 (0.90)	19.98 (0.00)	0.00 (0.00)	0.72 (0.36)
NN (Q) Pattern	58 (42)	51 (35)	4.27 (4.43)	3.74 (0.00)	5.28 (5.28)	3.22 (3.15)
NN (Q) Assoc.	30 (25)	2.7 (2.7)	0.54 (7.92)	107.64 (0.54)	0.00 (0.00)	2.70 (2.34)
Baseline	44	2.7	2.40	0.00	1.90	1.40

6. Conclusion

The HL-LHC will see up to 200 PU conditions for the LHC experiments. To maintain the physics performance of the detector and exploit the high integrated luminosity, the CMS experiment is being upgraded. Upgrades to the L1 Trigger system will see charged particle tracks within the full outer silicon tracker volume used for track matching and global event variables such as the primary vertex, which is necessary to separate the hard interaction from pile-up. This work introduces a novel approach to PV finding and association of tracks to the PV using an end-to-end neural network that learns both the PV position and the likelihood of a track originating from this PV. The network uses a custom histogram layer and soft ArgMax to ensure that the loss functions can be back-propagated and is shown to outperform the baseline approach in key metrics. Finally, the implementation of this network in an FPGA is discussed and the effective use of QKeras and pruning to reduce the overall resource usage is demonstrated.

References

[1] CMS Collaboration, “The Phase-2 Upgrade of the CMS Level-1 Trigger Technical Design Report,” Tech. Rep. CERN-LHCC-2020-009. CMS-TDR-019-002, CERN, Geneva, 2020. https://cds.cern.ch/record/2272264.

[2] G. Hall, M. Raymond, and A. Rose, “2-d PT module concept for the SLHC CMS tracker,” *Journal of Instrumentation,*, vol. 5, no. 07, p. C07012, 2010.

[3] D. Bertolini, P. Harris, M. Low, and N. Tran, “Pileup Per Particle Identification,” *JHEP*, vol. 10, p. 059, 2014.

[4] S. Mersi, “Phase-2 Upgrade of the CMS Tracker,” *Nuclear and Particle Physics Proceedings*, vol. 273-275, pp. 1034 – 1041, 2016. 37th Int. Conf. on High Energy Physics (ICHEP).

[5] J. Duarte *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," , *Journal of Instrumentation*, vol. 13, no. 07, p. P07027, 2018.

[6] C. N. Coelho *et al.*, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," *Nature Machine Intelligence*, vol. 8, no. 3, pp. 675–686, 2021.

[7] CMS Collaboration, "Description and performance of track and primary-vertex reconstruction with the CMS tracker," *Journal of Instrumentation*, vol. 9, no. 10, p. P10009, 2014.

[8] S. Summers, "Application of FPGAs to Triggering in High Energy Physics." Ph.D. thesis Imperial College, London, 2018.

[9] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73 – 101, 1964.

[10] M. Abadi *et al.*, "TensorFlow," 2015. Software available from tensorflow.org.

[11] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[12] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. of the 28th Int. Conf. on Neural Information Processing Systems - Vol. 1*, NIPS'15, (Cambridge, MA, USA), p. 1135–1143, MIT Press, 2015.