EPJ Web of Conferences **251**, 04008 (2021) https://doi.org/10.1051/epjconf/202125104008
CHEP 2021

A Deep Learning approach to LHCb Calorimeter reconstruction using a Cellular Automaton

Núria Valls Canudas[1,*], *Xavier* Vilasis Cardona[1,**], *Míriam* Calvo Gómez[1,***], and *Elisabet* Golobardes Ribé[1,****] *on behalf of the LHCb Real Time Analysis Project*

[1]DS4DS, La Salle-Universitat Ramon Llull, Quatre Camins 30, 08022 Barcelona, Spain

Abstract. The optimization of reconstruction algorithms has become a key aspect in LHCb as it is currently undergoing a major upgrade that will considerably increase the data processing rate. Aiming to accelerate the second most time consuming reconstruction process of the trigger, we propose an alternative reconstruction algorithm for the Electromagnetic Calorimeter of LHCb. Together with the use of deep learning techniques and the understanding of the current algorithm, our proposal decomposes the reconstruction process into small parts that benefit the generalized learning of small neural network architectures and simplifies the training dataset. This approach takes as input the full simulation data of the calorimeter and outputs a list of reconstructed clusters in a nearly constant time without any dependency in the event complexity.

1 Introduction

The increasing popularity of machine learning techniques in recent years has pushed many improvements in a wide range of computational challenges in high energy physics (HEP). Through the years, the Large Hadron Collider (LHC) experiment has proven to make a long use of this techniques to automatize and accelerate various tasks. Focusing in one of the main experiments, LHCb is currently facing an upgrade involving an increase in complexity and volume of data up to an average rate of 32 Tbit/s for the full 30 MHz bunch crossing rate of LHC [1]. This evidences the need to improve the detectors software in efficiency and performance. Moreover, the vision of future upgrades [2] enhances the importance of developing scalable and flexible software methodologies. Within the current upgrade, LHCb has started a migration process to a full software trigger that aims to make a complete reconstruction at an early stage before any data pruning or storage [3]. The first steps to be done in this direction start with reducing the computational complexity of the reconstruction software to accomplish the time and throughput requirements of the detector. The current time performance analysis of the reconstruction process in LHCb points the Calorimeter reconstruction as the second most computational expensive process representing 25% of the LHCb reconstruction time [4].

The LHCb Electromagnetic Calorimeter (ECAL) [5] can be represented as a 2 dimensional grid built from individual modules made from lead absorber plates inter spaced with

*e-mail: nuria.valls@salle.url.edu
**e-mail: xavier.vilasis@salle.url.edu
***e-mail: miriam.calvo@salle.url.edu
****e-mail: elisabet.golobardes@salle.url.edu

scintillator tiles as active material. The structure is segmented into three regions of different dimension. Each region has identical sized modules with different number of readout cells. Hence, the region closest to the beam pipe (inner) consists of 167 modules containing 9 cells of 4x4 cm^2 area, the middle region has 448 modules containing 4 cells of 6x6 cm^2 and the outer region has 2688 modules that are made from a single cell of 12x12 cm^2. The data processed in the reconstruction phase comes from the readout cells at each of the three regions.

In this article, we propose a reconstruction strategy for the LHCb calorimeter response that aims to reproduce the steps of the currently used algorithm in a specific formulation that facilitates the generalized learning of small deep learning structures. With this methodology we achieve to have an efficient cluster reconstruction algorithm that performs at a nearly constant speed with independence of the events complexity. To start, an analysis of background methodologies and other deep learning approaches for the LHCb calorimeter reconstruction is presented; then, the insights of our proposal are explained starting with an overview of the current implementation of the calorimeter reconstruction. Next, the results obtained in the testing of the proposal are shown, followed by a discussion and conclusions.

2 Background

The current reconstruction strategy for the LHCb Calorimeter detector is based on the mathematical model called Cellular Automaton [6]. Due to the detectors geometry, the cellular automata strategy has long been used in calorimeters for high energy physics [7, 8]. Such method provides an efficient reconstruction of the clusters of an event although the currently used formulation requires several iterative processes along the data cells that are programmed as loops in the algorithms code. As this is causing a strong dependency of the algorithms complexity on the number of clusters in the data, other approaches have tried to avoid this dependency exploiting the architecture similarities between cellular automata and neural networks [9, 10]. However, these approaches focus in a proof of concept rather than providing a specific reconstruction solution for the LHCb calorimeter data.

Within the last years, the evolution of deep learning models has encouraged the use of image processing techniques also in the HEP field. An early stage project has shown promising results when approaching the LHCb calorimeter reconstruction with a deep neural network structure [11]. In this paper, a different reconstruction strategy is proposed aiming to simplify the learning process of the neural networks. It is built over the formulation of the current reconstruction algorithm but using a set of deep learning structures, each one trained to solve one step of the reconstruction process at a time. With this approach, we reproduce the same steps of the current implementation of the algorithm but using deep architectures that are efficient in solving one step of the reconstruction with the advantage of being constant cost functions.

3 Proposal

3.1 Formulation of the problem

Entering in the detail of the classical implementation of the calorimeter reconstruction, the Cellular Automata based algorithm is segmented in four different sequential steps. The first one is the **seed finder** which detects and identifies all the seeds in a raw image. A seed is defined as a hit that has the maximum local energy value among its neighbours. In this case, the neighborhood of a cell is defined by its eight adjacent cells in the calorimeter grid. The second step of the reconstruction process is the **cluster reconstructor**, which is an iterative

process that assigns all the cells with a significant energy to the closest seed. The third step is the **overlap solver** which fractions the energy of cells that have been tagged to more than one cluster. Finally there is a fourth step which is the **cluster corrections** that are applied after the full reconstruction.

Looking at the first three steps, one can think of a set of rules and possible states that applied to a certain neighbourhood can model the behavior of each step. By doing this we can formulate the three steps as three independent cellular automata models. The advantage of such a formulation is that the ruleset of some cellular automata models have proven to be learned by deep convolutional architectures [12]. As stated in the cited article, a deep enough architecture can learn and apply the generalized rules to unseen data with a proper training.

Note that in the approach presented in this paper, the three regions of the calorimeter are processed separately with its own channel and results are aggregated at the last stage of the process to complete the full calorimeter reconstruction.

3.2 Seed finder implementation

Our goal is to implement a deep neural network and train it to learn the rules of one of the three steps of the reconstruction process. In this case, starting with the seed finder. The cellular automaton model that defines this step is binary, meaning it has only two possible states as output (one if the cell is a seed, zero otherwise) and a neighbourhood definition of 8 cells. The ruleset in this case is defined by the following function:

$$f(c_{i,j}^{t+1}) = \begin{cases} 1, & \text{if } c_{i,j}^{t} > c_{i+M,j+N}^{t} \text{ for } M \in \{-1,0,1\},\ N \in \{-1,0,1\} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Where the energy value of each cell is defined as $c_{i,j}^{t}$ and the case $M = 0, N = 0$ is excluded.

One of the key points in the training process of the network is realizing that there is no need to use the real images of data from the calorimeter. Instead, it can be shown a set of randomly generated images with the particularity of having a uniform distribution of all the possible input samples defined in the ruleset. This dataset does not even need to have the same range of values as the calorimeter data because the definition of the ruleset only takes into account the result of comparisons between the value of the cells. Hence it does not have a dependence with its numeric value scale. Therefore, the dataset used in the training of the seed finder step is build of a set of images with random values at each cell in a range from 0 to 99. With this, the network achieves a generalized knowledge of the defined ruleset that can be later applied with success to the real data from the calorimeter.

The output format obtained from the network consists of a same sized image as the input data with ones in the cells that match the requirements for being a cluster seed.

3.2.1 Detail of the networks

As the three regions of the calorimeter are processed independently, every region needs its own seed finder network. Although the network and data shapes differ between regions, all of them have the same structure, which consists of a 2-dimensional convolutional layer followed by two or three dense layers (depending on the region) and finally a two neuron dense layer as the output layer. The three networks are trained as classifiers, understanding the output class as the state of a cell in the cellular automata formulation. In Table 1, a summary of the network parameters can be found for each of the three regions. Results at this point are measured in accuracy, understood as the number of correct predictions over the expected

Table 1. Parameter summary of the seed finder networks

	Image shape	Training set	Neurons per layer	Trainable parameters
Outer	64×52	10000	[20, 20, 20, 10, 2]	1272
Middle	64×40	10000	[20, 20, 20, 10, 2]	1272
Inner	48×36	10000	[10, 10, 10, 2]	342

output of the network. For the three networks the accuracy is over 98% when testing with samples of full LHCb simulated data.

3.3 Cluster reconstructor and overlap solver implementation

Once proven that the formulation and network training achieved good results in the seed finder, the same approach is applied to the following steps of the reconstruction process.

3.3.1 Output definition

The final objective of the reconstruction is to have a list of clusters, each one identified with the unique cell identifier (cell ID) of the seed. A list of cell IDs and energies of the cells associated to each cluster are also needed. Translating this into the image format of the data needed for this proposal the best approximation is to build what we call a reconstructed cluster stack. Our representation of a cluster in this format is defined by a 3 by 3 image where each pixel represents a cell. The central cell stands for the seed of the cluster and each cell in the image has the reconstructed energy assigned to that cluster as value. Aiming to keep the information of locality and also to have a fixed length output, the reconstructed cluster stack has a fixed number of cluster images for each region: 3328 for the outer (64×52), 2560 for the middle (64×40) and 1728 for the inner (48×36). Thus we achieve to locate each reconstructed cluster at the position of the stack concerning the position of the seed in the raw data image. The cluster images in the stack concerning positions in which there is no seed in the raw image are filled with zeros.

3.3.2 Formulation

The cluster reconstructor step and the overlap solver step have in common that its definition of neighbourhood is bigger than the 3 by 3 used in the seed finder. Developing a bit more this concept, we can look at the left image in Figure 1 as an example of a 5 by 5 cell window of the calorimeter data that is centered on a seed. Remembering the job of the cluster reconstructor, it must assign cells to the closest seed. Also in case one cell is close to more than one seed, it must be tagged as *overlapping*. In the example, the cells that need to be tagged are the ones around the central blue cross, marked in light blue, considering that the same process will be repeated for windows centered in all the seeds of the image. But if one looks only at the eight neighbouring cells around the blue cross seed, it won't see the purple seed nor the green one, which both are causing overlap with the cells marked with a circle. In this particular case, there is no need to see the orange seed as any seed located further than two cell square around the central seed will never cause overlap with any cell from the blue cluster by construction. Therefore in order to correctly tag the cells around a seed in the cluster reconstructor step, the automata formulation needs to define the neighborhood as a 5 by 5 window around each seed in the input image. These definitions are done under the assumption that all the reconstructed clusters are built as the 3 by 3 cells around a seed as in the LHCb reconstruction.

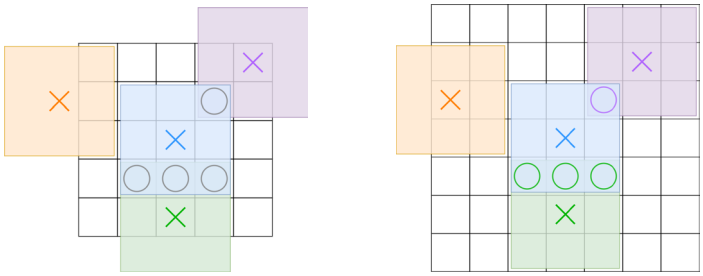

Figure 1. From left to right: diagram of an example distribution of clusters in a 5 by 5 cell window centered on a seed and diagram of the same distribution of clusters in a 7 by 7 cell window. In both diagrams the cluster seeds are marked with crosses and the overlap cells are marked with circles.

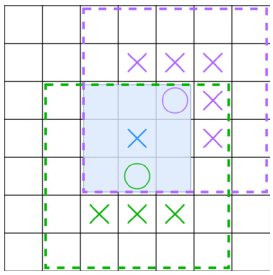

Figure 2. Diagram representing the possible seed positions overlapping with the central cluster in a 7 by 7 window. The seed positions are marked with crosses and the two of the overlap cells that need to be predicted are marked with a circle.

The same explanation is applied to the overlap solving step. Looking at the right image in Figure 1 there is the same example but in a bigger cell window of 7 by 7. In this case, the overlap solver task is to fraction the energy of the cells marked as *overlapping* so that the implied clusters are assigned a proportion of the cells energy according to the total energy of the clusters. Hence in the example, a 5 by 5 window is not enough as we are not seeing the full range of the purple and green clusters. Instead, we need a 7 by 7 window centered in the blue seed so that the energy of all the light purple cells is available to correctly fraction the energy of the purple circled cell. The same happens with the green cluster, as we need to sum up the energy of the light green cells to fraction the energy of the green circled cells from the blue cluster. As in the previous example, there is no need to see all the light orange cells as the orange cluster is not overlapping with the central cluster.

Given the similarities in the neighbour observation in both steps, we have opted for a cellular automata formulation that performs the cluster reconstructor and the overlap solver at once. Accordingly, the 7 by 7 window structure holds the reconstruction step for a single cluster (the one centered at the window) and outputs a 3 by 3 cluster image from the reconstructed cluster stack.

However, the reconstruction approach presented in this paper stands for the decomposition of the problem in simplified steps. In this case, from each 7 by 7 window centered in a seed, a network needs to generate 9 values corresponding to the reconstructed energy values of the cluster. This process can be generalized as 9 smaller windows of 5 by 5 around the central cluster. As a reference, see the diagram in Figure 2; for the purple circled cell in the blue cluster, all the positions where a seed of a cluster may cause overlap are marked with purple crosses. Consequently, in order to see the full clusters that may be causing overlap, there needs to be a big enough window following the purple dashed square. This structure is repeated for the four cells at the corners of the blue cluster. In the same way, for the cells

located at the top, bottom, left and right of the blue cluster, the structure follows the same pattern as the green circled cell.

Consequently, the formulation of this step of the reconstruction focuses in transforming an image of 25 cells (5×5) into a single value. However, within the cellular automata formulation, there must be enough states such that any cell value in the calorimeter can be represented ($2^{12} = 4096$ different states). About the neighbour definition, as discussed before, consists of 24 cells in a 5 by 5 square around the current cell. Finally, the ruleset that defines this reconstruction step can be summarized as:

$$f(c_{i,j}^{t+1}) = \begin{cases} c_{i,j}^{t}, & \text{if is central seed} \\ \frac{c_{i,j}^{t} C_0}{\sum_{k=0}^{K} C_k}, & \text{for } K = num\_seeds \text{ if } c_{i,j}^{t} > 0 \text{ and is not central seed} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Where

$$C_k = \sum_{m=-1}^{1} \sum_{n=-1}^{1} c_{o+m,p+n}^{t} \tag{3}$$

and *num_seeds* refers to the number of seeds causing overlap to the central cell. Variables o and p stand for the seed coordinates of cluster k in the 5 by 5 image. For $k = 0$ the cluster is specifically the one in the center of the 7 by 7 window.

Within the window reshaping from 7 by 7 to 5 by 5, information regarding the position of the central cluster is lost. In order to distinguish the central seed from the others, a third stream of data is generated by masking the 7 by 7 windows with a tensor before the reshaping.

3.3.3 Non-linearities

Before starting with the network definition there is one key aspect worth mentioning. Observing the formulation of the seed finder step in equation 1, it can clearly be said that the function is linear as the only outputs are one or zero. However when looking at equation 2 there is a division in the result for the second condition, which does not correspond to a linear behavior. Following the universal approximation theorem [13], there needs to be at least one hidden layer to approximate non-linear functions. Therefore, the neural network structure used in the first reconstruction step cannot be used in this case as the convolution is performed with a 2-dimension convolutional layer, which performs a matrix multiplication between the data and a linear kernel. At this point, the conclusion is to proceed with the implementation of this step using a multi layered perceptron (MLP) structure trained to be the kernel of the convolution.

3.3.4 Detail of the network

Following with the MLP structure, there is no need to build three different networks for each calorimeter region whereas the window shape of the convolution does not change by region. The difference will lay in the length of the convolution, hence, the kernel can be the same for the three data channels. The structure of the network consists of four dense layers with $[64, 64, 64, 32]$ neuron distribution followed by an output dense layer of one neuron. Given the large number of states needed to represent this automaton, the approach of this step suggests to train the network as a regressor. With a total of 108.993 parameters, it takes as input three streams of data: the raw calorimeter data, the seed data generated in the first

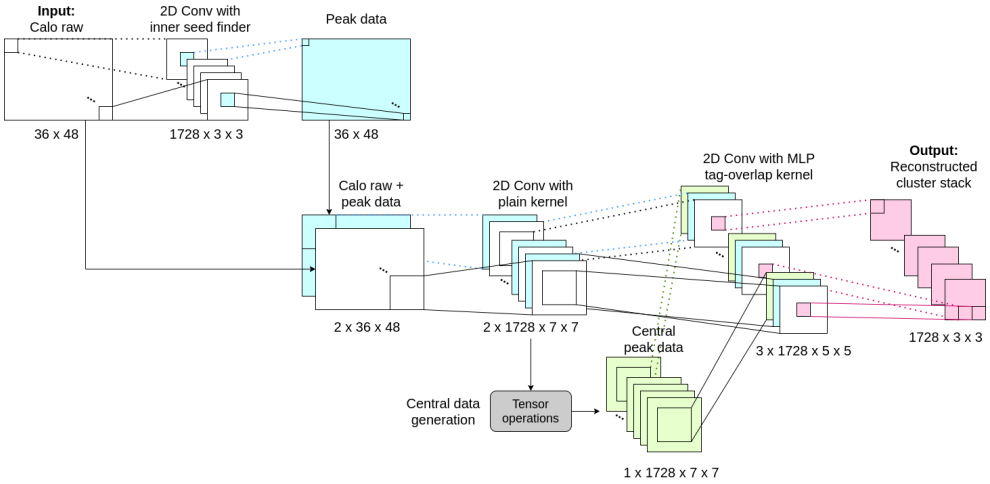

Figure 3. Detailed scheme of the inner reconstruction channel from the raw data to the reconstructed cluster stack.

step and the central cluster data generated in the 7 by 7 reshaping process. The streams are reshaped into a 5 by 5 image format and the output for each image is a single value, that must be reshaped into 3 by 3 images to obtain the reconstructed cluster stack from a concrete region. Although the kernel for the three channels is the same, the cluster reconstructor and overlap solver step still needs to be performed by the three regions independently.

The dataset used for the training contains a set of 200000 5 by 5 images. In this case, they come from the decomposition of 25000 7 by 7 images selected from full LHCb simulation data. The set is balanced within a variety in the number of seeds and overlapping cells. Given the regressive nature of this network, results in terms of training performance are measured with the relative difference of energy per cluster between the reconstructed cluster and the clusters reconstructed with the LHCb algorithm.

3.4 Overview of the proposal

In order to have an overview of the reconstruction process, there is a schematic of the reconstruction flow for the inner region in Figure 3. The top left piece of data (input) is a tensor matrix filled with the hits of an event that belong to a concrete region. From this input there are three types of data that are generated: the peak data marked in blue, the central peak data marked in green, generated with the aggregation of the input and the peak data and applying a tensor operator, and finally the reconstructed cluster stack marked in pink, produced from the aggregation of the input, peak and central peak data convoluted with the trained MLP. The same scheme is maintained for the other two regions, however the seed finder network changes and the dimensions of data are adapted to the region shapes.

4 Experimentation

Aiming to compare the computational performance of the proposed reconstruction method with a benchmark solution for the LHCb calorimeter, we have implemented a version of the classical cellular automata reconstruction algorithm with iterative coding. It has the same

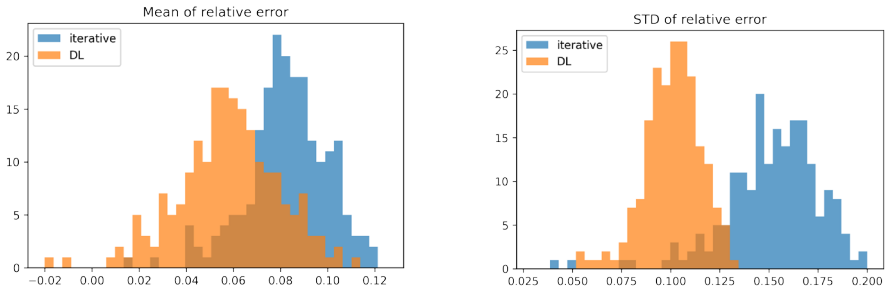

Figure 4. From left to right: Histograms of the mean value of the relative difference of energy reconstructed per cluster from an LHCb simulated event for the iterative implementation in blue and for the deep learning implementation in orange. Histograms of the standard deviation of the relative difference of energy reconstructed per cluster from an event for the same two implementations.

computational complexity as the current reconstruction algorithm in LHCb but coded in Python so that the execution conditions for the tests are the same as in the proposed method.

All the results provided in this paper have been obtained using a computer with the following properties: memory of 15, 5 GB, processor Intel Core i7-6500U CPU @ 2.50 GHz ×4, disk capacity of 512, 1 GB with Ubuntu 20.04.1 LTS 64-bit OS. All the algorithms are coded in Python 3.8 and all the explained neural networks have been built and trained using the Tensorflow 2.3.0 library.

Regarding the methods performance, it is understood as the resemblance between the reconstructed clusters from the proposal and the actual reconstruction running in LHCb. The method proposed in this paper and also the iterative one have been compared with the LHCb reconstruction results. This provides a complete comparison between the two methods and positions the performance of the proposed one. Results are shown in Figure 4 in terms of the mean and standard deviation of the relative difference between the clusters of a full event, with respect to the current implementation, for a total of 200 different events. It can be observed that for the proposed deep learning approach (DL) the mean difference is considerably close to zero and has lower deviation than in the iterative algorithm. Although further experimentation needs to be done comparing with montecarlo truth information, the observed values of $\approx 0.06 \pm 0.1$ mean difference can be considered significantly good.

Results in terms of computational performance are measured as the time in seconds elapsed between the reading of the hits and the generation of the list of clusters for the three regions of the calorimeter. The execution of both methods has been done using a single thread in each case. Figure 5 shows a plot of the computational time measured as a mean of one hundred iteration on the same event for 200 different events. Looking at the iterative curve, although it performs really fast with small events, it shows a clear quadratic growth with the number of hits. On the other hand, the deep learning approach shows a nearly constant behavior towards the number of hits. Although it has a small positive slope of 4.97×10^{-6}, the tendency shows to be linear. However, around 72% of the events processed in the testing fall under the time performance curve of the deep learning approach. Still we have achieved a constant computational time with independence of the event complexity.

In addition, this approach could clearly benefit from executing the reconstruction of the three regions in parallel given that when plotting the time portion of the full algorithm dedicated to reconstruct each of the three regions separately, the highest curve is settled by the outer region reconstruction which cuts over 11% of the events in the iterative curve.

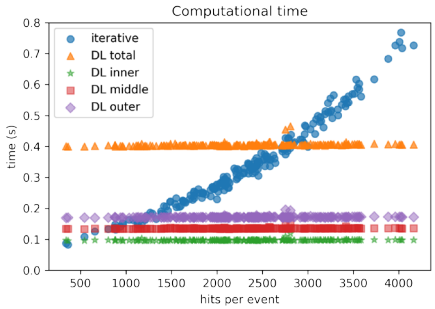

Figure 5. Scatter plot of the mean computational time over the number of hits per event from LHCb simulations. Comparing the iterative implementation and the deep learning implementation also with segmented regions.

5 Discussion and conclusions

Within the development of this proposal there are several things that have been learned. We have seen that segmenting the reconstruction steps can help in developing a deep learning solution. Moreover, we have seen that there is no need to work with full simulation data to train specific networks; when a reconstruction step is reduced to a generic application of a function with independence of the numeric value scale of the input, the training dataset of the network can be generalized to a homogeneous set of all the possible inputs randomly generated. In other words, we train networks on the rules that solve a general vision of the problem, not over real and non-generic data. It has been proved that the network achieves to learn the application of the formulated rule in a generalized context. However, when homogeneity of input samples is difficult to obtain with random generation, like in the case of the cluster reconstructor and overlap step, one can use real simulation data to extract windows of images to be used as input, hence from a set of two thousand different events more than 200.000 input samples can be extracted for the training set. The complexity reduction on the training data has also reflected into a fast training process and the simplification of the networks in terms of architecture and number of parameters compared with other deep learning approaches.

Despite the good results, there is still room for improvement in terms of performance. As a proof of concept, the performance comparison in this paper is done with the current implementation of the reconstruction, not the montecarlo truth. As a future improvement, montecarlo truth information will be used to test and directly train the MLP network in order to refine the reconstruction process at the time of solving the overlapping energy fractioning. In terms of computational time, there is a clear gain in the reconstruction complexity with the proposed approach. However the execution time could possibly be reduced with a vectorized implementation of the proposal. Apart from that, the proposed implementation clearly benefits from parallel execution reducing the computational time by nearly a factor 3. Moreover, its convolutional formulation could benefit even more from a GPU architecture without conditioning the efficiency as the insight neural networks and convolutional operators are highly paralelizable.

However there are some aspects from this approach that need to be worked on. Due to the region-independent strategy used in this approach, clusters that fall in the boundary regions of the calorimeter are now reconstructed partially as two separate clusters in each region. There is the idea of using a Graph Neural Network (GNN) with a similar training as the MLP in order to perform the reconstruction in the boundary regions as GNNs can model irregular neighbourhoods. Another aspect that needs to be worked is the identification of π^0 particles. By nature, π^0s arrive at the calorimeter as two very close, similar energy particles, but need to be reconstructed as one. Hence, the window that surrounds a pair of π^0s is bigger than the

defined 3 by 3. With the current training of the MLP in our proposal, the network wrongly reconstructs the π^0s as two very close overlapping clusters. There is the idea of improving this shortcoming re-training the network of the current implementation to identify the merged π^0s and make an ad-hoc reconstruction for those cases.

As a conclusion, we have achieved to implement and test an alternative approach to the LHCb calorimeter reconstruction. It adapts the current reconstruction steps to a formulation that can be learned by simple deep learning structures. With this we make sure that the reconstruction process is correct as it mimics the implementation of the current algorithm but gaining in computational complexity. Results from the testing show an interesting behavior in terms of computational time that could be promising also for a full calorimeter reconstruction implementation on GPUs for the LHCb high level trigger.

6 Acknowledgements

We acknowledge support from the spanish national agency MICINN through grant PID2019-106448GB-C32.

References

[1] LHCb Collaboration, *Framework TDR for the LHCb upgrade: technical design report*, LHCb-TDR-012 (2021).

[2] LHCb Collaboration, *Physics case for an LHCb Upgrade II-Opportunities in flavour physics, and beyond, in the HL-LHC era*, arXiv preprint arXiv:1808.08865 (2019).

[3] LHCb Collaboration, *et al.*, *LHCb Upgrade GPU High Level Trigger Technical Design Report*, CERN-LHCC-2020-006 (2020).

[4] LHCb Collaboration, *Throughput and resource usage of the LHCb upgrade HLT*, LHCB-FIGURE-2020-007 (2020).

[5] LHCb Collaboration, *LHCb calorimeters: Technical Design Report*, LHCb-TDR-002 (2000).

[6] V. Breton, N. Brun, P. Perret, *A clustering algorithm for the LHCb electromagnetic calorimeter using a cellular automaton*, CERN-LHCb-2001-123 (2001).

[7] V. Breton, *et al.*, *Application of neural networks and cellular automata to interpretation of calorimeter data*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **362**, 478–486 (1995).

[8] M. Casolino and P. Picozza, *A cellular automaton to filter events in a high energy physics discrete calorimeter*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **364**, 516–523 (1995).

[9] B. Denby, *Neural networks and cellular automata in experimental high energy physics*, Computer Physics Communications **49**, 429–448 (1988).

[10] C. Baldanza, *et al.*, *A cellular neural network for peak finding in high-energy physics*, Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA 2000)(Cat. No. 00TH8509), 443–448 (2000).

[11] M. Mazurek, *Deep learning solutions for 2D calorimetric cluster reconstruction at LHCb*, LHCb-TALK-2020-178 (2020).

[12] W. Gilpin, *Cellular automata as convolutional neural networks*, Physical Review E **100**, 032402 (2019).

[13] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems **2**, 303–314 (1989).