EPJ Web of Conferences **337**, 01024 (2025) https://doi.org/10.1051/epjconf/202533701024
CHEP 2024

Optimised Graph Convolution for Calorimetry Event Classification

Matthieu Melennec[1,*], *Shamik* Ghosh[1,**], and *Frédéric* Magniette[1,***]

[1]Laboratoire Leprince-Ringuet, École polytechnique - CNRS, Institut Polytechnique de Paris, Palaiseau, France

Abstract. In the recent years, high energy physics discoveries have been driven by the increasing of luminosity and/or detector granularity. This evolution gives access to bigger statistics and data samples, but can make it hard to process the detector outputs with current methods and algorithms. Graph convolution networks, have been shown to be powerful tools to address these challenges. We present our graph convolution framework for particle identification and energy regression in high granularity calorimeters. In particular, we introduce our algorithm for optimised graph construction in resource constrained environments. We also introduce our implementation of graph convolution and pooling layers. We observe satisfying accuracies, and discuss possible application to other high granularity particle detector challenges.

1 Introduction

The increase in luminosity at HL-LHC [1] and detector granularity for the CMS HGCAL upgrade [2] will allow to observe complex phenomena at an increased level of precision. This will increase the available statistics and precision by multiple orders of magnitude, which facilitates the detection of rare events, at the price of a significant increase of the number of channels. The challenge is that most of the standard techniques for reconstruction and triggering are not operative in such a context. For example, the energy threshold-based triggers fail to handle the complexity of high pile-up collisions. While deep learning methods such as convolution techniques are known to handle noisy and complex data inputs well [3], they do not generalise well to the very peculiar topologies of particle detectors. Alternatives such as spatial graph convolution have been shown to handle well such data [4]. In particular, they have proven to give excellent results on particle detector data at LHC [5] and neutrinos experiments [6] for tasks such as classification, energy regression and object segmentation. We present the use of graph convolution neural networks for calorimeter object identification and energy regression, in the context of the OGCID project [1]. In particular, we use the knowledge of the detector geometry to pre-compute proximities of potential node positions in the data to optimise the graph construction procedure, and discuss its time complexity. We also present the chosen message passing convolution operation, as well as introduce our pooling

*e-mail: matthieu.melennec@llr.in2p3.fr
**e-mail: shamik.ghosh@llr.in2p3.fr
***e-mail: frederic.magniette@llr.in2p3.fr

[1]Optimisation of Graph Convolution for particle IDentification: https://llrogcid.in2p3.fr/

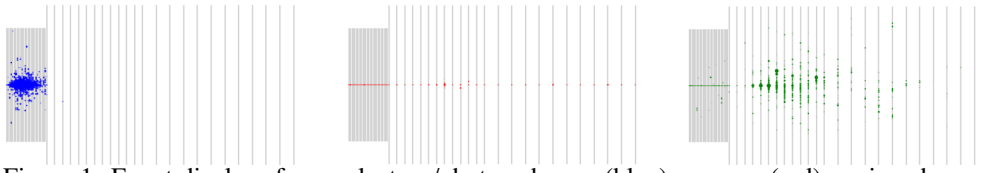


Figure 1: Event displays for an electron/photon shower (blue), a muon (red), a pion shower (blue). Photon showers are very similar to electron showers [8].

algorithm, Treclus, to coarsen local substructures, while preserving the overall structure of the graph. We present the graph readout pooling operation, used to flatten the outputs of convolution to an MLP of constant size. Finally, we present the classification and energy regression capacities of our pipeline, looking in particular at the resolution in energy to check the validity of our approach.

2 The D2 Dataset

The data used in this study are extracted from the OGCID/D2 simulation dataset [7]. It is a set of single particle interaction with a simplified high granularity calorimeter inspired by CMS HGCAL [2]. This detector architecture features two main sections: the electromagnetic calorimeter (ECAL) and the hadronic calorimeter (HCAL), optimised for detecting particles of varying energies. The ECAL comprises 26 layers of lead absorbers (6.05 mm thick), while the HCAL has 24 layers of stainless steel absorbers, with 12 layers of 45 mm thickness and 12 of 80 mm, providing high resolution sampling of particle showers. Active silicon layers are placed between absorbers, segmented into hexagonal cells with a thickness of 0.32 mm and a transverse area of approximately 1 cm^2 for ECAL and 4 cm^2 for HCAL. The ECAL layers are comprised of 3571 Si sensor cells for a total width of 680 mm, and the HCAL layers have 1801 Si cells and are 960 mm wide, for a total of 136,070 channels[2] (versus 6 million for HGCAL [2]). The D2 dataset is a collection of simulated interactions between this detector and four types of particles (muon, positively charged pions, electrons and photons) at different energies and with a flat incidence angle (aligned with the longitudinal detector axis). The dataset includes detailed informations on each particle event, namely the energy measurement in each cell but also extracted variables of interest describing the geometry and the energy repartition of the hits. For this study, we looked at electrons, photons, muons and pions with energies ranging from 10 to 100 GeV and aligned with the longitudinal axis. The event data that was used was the energy measurement in each cell.

3 Code Architecture

The pipeline used in this work is based on message passing convolution [9]. After generating graphs from the raw event data, these go through several layers of message passing convolution and pooling (CP layers). The resulting graphs are then flattened into a one-dimensional tensor by a readout pooling layer, which flattens the node features of the graphs in an orderly fashion. It also returns a structured tensor of constant size, independent of the input graph's size. This layer is essential to then feed the convoluted features to a multi-layer perception that will output our objective (particle ID or energy) by following a systematic pattern.

[2] https://llrogcid.in2p3.fr/hgcal-like-simulations/

Sensor id	pos. 1	pos.2	$\cdots$	k
1	s_1^1	s_1^2	$\cdots$	s_1^k
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
i	s_i^1	s_i^2	$\cdots$	s_i^k

Table 1: Proximity table. s_i^k is the sensor in k-th position in the row for sensor s_i. Neighbours are ordered such that for all $i, k, d(s_i, s_i^k) \leq d(s_i, s_i^{k+1})$.

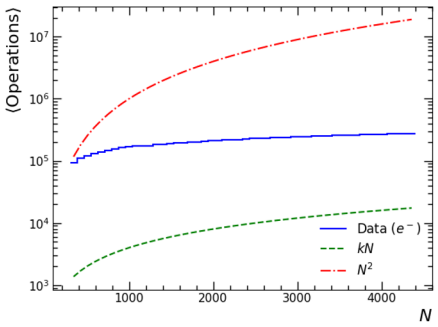


Figure 2: Average number of operations for the PT-KNN algorithm. A comparison shows linear and quadratic complexities. Asymptotically, the average number of operations behaves like $O(kN)$

Algorithm 1: PTKNN: Proximity table k nearest neighbours graph generation algorithm. k is the number of neighbours wanted for each node.

Input: hits, k
Output: $G = \{V, E\}$
$V \leftarrow \{\}$
$E \leftarrow \{\}$
for $s_i \in$ hits **do**
    $V \leftarrow V \cup \{s^i\}$
    counter $\leftarrow 0$
    while counter $< k$ **do**
        for $s_i^k \in \{s_i^k\}_k$ **do**
            if $s_i^k \in$ hits **then**
                $V \leftarrow V \cup \{s_i^k\}$
                $E \leftarrow E \cup \{(s_i, s_i^k)\}$
                counter $\leftarrow$ counter $+1$
            end if
        end for
    end while
end for
return $G = \{V, E\}$

3.1 Graph Construction and Proximity Tables

Every calorimeter event is represented as a graph, the nodes of which correspond to every hit cell of the event, linked by arbitrarily defined edges. Our edge building strategy uses the k nearest neighbours (KNN) algorithm to encode a sense of geometric locality in the graph structure. Our implementation of KNN is based off of so-called proximity tables (PTs), which contain a pre-computed order of neighbourhood for every sensor of the detector. That is, for every sensor of the detector, all other sensors are ordered according to a user-defined metric, as in Table 1. The graph building procedure then follows algorithm 1. Note that while it is standard to use the euclidean distance for the KNN algorithm, other metrics can be used to encode physical properties in the graph structure. In this study, the choice of the metric seemed not to have an impact on the performance of the reconstruction, and we therefore only use the euclidean distance. We also use the full proximity table for the D2 detector, i.e. m^2 entries with m the number of Si sensors in the detector. Intuitively, one understands that the PT-KNN algorithm brings an improvement to the KNN algorithm with regards to time complexity. Indeed, the KNN complexity, as used here for graph construction where for every nodes, we look through all the other nodes for the k nearest, is constantly quadratic. For the PT-KNN algorithm, while the worse case complexity can in theory increase to $O(NM)$, with N the number of nodes and M the number of cells in the detector if $M \gg N$, this risk is mostly mitigated by the geometric locality of the shower (figure 1). In practice, in the range of number of nodes N corresponding to the size of a shower in the detector, the observed

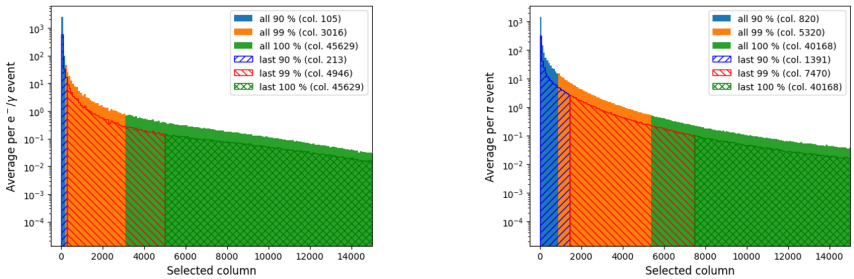


Figure 3: Column selections for PT-KNN algorithm with $k = 4$, euclidean distance at energies 10 to 100 GeV.

average last selected column $\langle c_{\text{last}} \rangle$ gives an average complexity $N \langle c_{\text{last}} \rangle \sim (\log N)^2$. To verify this, we assume the nodes are spread following a normal distribution and combine it with the number $N_{\text{cells}} \propto r$ of cells found in a radius r around the Gaussian's mean. This allows us to recover the radius where on average k nodes can be found and estimate $\langle c_{\text{last}} \rangle = A \log^2 N \log(N/(N-k)) + k$, which when fitted to the data gives $(\log N)^2/N$ at leading order[3]. In the asymptotic range $N \to \infty$ where $N \gg (\log N)^2$, one must consider the lower bound kN imposed by the PT-KNN algorithm, and we obtain a linear asymptotic complexity. The value of $\langle c_{\text{last}} \rangle$ is particularly relevant in the context of constrained computing, such as particle detector triggering algorithms. Indeed, reducing the size of the proximity table would greatly reduce the memory footprint of the PT. Hence knowing how far the PT is explored allows us to truncate the rows to $\lambda \langle c_{\text{last}} \rangle$ columns for some λ to be determined. In practice, we observe that for most hits, the PT-KNN algorithm only explores 5% of the corresponding row in the PT (figure 3). Hence a strong truncation of the depth of the PT would likely not have too big an impact on the graph structure while drastically reducing the size of the PT. The truncation of the proximity table also ensures that the worse case complexity reduces to $O(Nm_{\max})$ with $m_{\max}$ the number of columns kept with typically $m_{\max} \sim N$. The resulting graphs are composed of vertices $v \in V$, holding as features the energy x_v measured by the corresponding sensor and the position $\vec{u}_v$ of that sensor. Note that in order to respect the symmetries of shower formations in the detector, the position of the node is kept as a "hidden feature," not used during the convolution operation. The positional information of the graph is encoded in the edges $e_{vw} \in E$, which carry as feature $x_{vw} = \left\| \vec{u}_v - \vec{u}_w \right\|_2$ the euclidean distance between the end nodes $v, w \in V$, holding information about *relative* positions. We observed that when increasing k, the improvement in classification and regression power formed a plateau from $k = 4$, while being more computationally expensive. As such, All the graphs built in this work use $k = 4$.

3.2 Message Passing Convolution

Message passing graph convolution can be seen as a generalisation of image convolution, were instead of operating on pixels, we operate on the nodes and/or edges of a graph [9]. We chose the message function to be a single layer of neurons with a Leaky-ReLU activation function, taking as input χ, the concatenated features x_v, x_w and x_{vw}:

$$\phi_{\theta_\phi}(x_v, x_w, x_{vw}) = \text{Leaky-ReLU}\left(\Theta_\phi \chi\right), \quad \chi := [x_v, x_w, x_{vw}] \in \mathbb{R}^{2n+m} \tag{1}$$

[3] A fully detailed proof may be found on the OGCID project web page

where n denotes the number of features per node, m the number of features per edge, and $\Theta_\phi \in \mathbb{R}^{l\times(2n+m)}$ is a matrix of trainable weights, with l the output size of the message function. Messages from all neighbours are aggregated by a feature-wise max pooling for particle IDing problems in order to identify protruding features [10] and mean pooling for energy regression in order to smoothen the protruding features and consider the graph globally [11]. Finally, we update the features of node v by performing the aggregation of the aggregated messages with the features of the node v itself. In practice, this is equivalent to adding self-loops to the graphs, i.e. for any $v \in V, e_{vv} \in E$.

$$\gamma_{\theta_\gamma}\left(x_v^{(l)}, \square_{w\in\mathcal{N}(v)} \phi_{\theta_\phi}\left(x_v^{(l)}, x_w^{(l)}, x_{vw}\right)\right) = \square_{w\in\tilde{\mathcal{N}}(v)} \phi_{\theta_\phi}\left(x_v, x_w, x_{vw}\right) \tag{2}$$

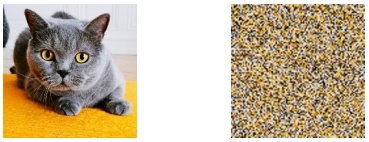
with $\tilde{\mathcal{N}}(v) = \mathcal{N}(v) \cup \{v\}$. In our case, the edge features are the euclidean distance between nodes. As such, for any node v, $x_{vv} = 0$.

3.3 Graph Pooling

Graph pooling is an important step in graph convolution, as it allows to coarsen the graph, hence looking at the graph on a more global scale [12]. First, let us express the steps of a graph pooling operation as in [13], with a clustering step, where we select which nodes will be clustered together, followed by a reduction step, where nodes within clusters are pooled, and finally a connection step, during which we update the adjacency of clusters. In our case the graphs are embedded in a euclidean space, and we therefore want to cluster together neighbours that are geometrically close to one another. To this end, we developed a threshold clustering algorithm, called Treclus, that creates a matching of the graph [14], where matched nodes are linked by an edge shorter than a threshold ε, and clusters matched nodes together. It is very important that the clustering edges make a matching to keep a linear complexity of the clustering algorithm. This however implies that clusters may only contain two nodes, while we ideally wish to find a maximum matching of the subgraph $G' = (V, E' = \{e \in E | e \leq \varepsilon\})$, which is usually a quadratic problem [15]. Our approach therefore consists in applying the Treclus algorithm multiple times in a row to increase the number of clustered short edges with limited impact on complexity. We observed that less than 10 calls to Treclus suffice to collapse all edges shorter than the 30-th percentile of edges after at most 2 CP layers. The reduction step consists of a feature wise pooling of nodes within the same clusters. The choice of the pooling operation follows the same logic as the aggregator function choice for the convolution step. However, while this applies strictly for the classification task (with a max pooling), for the energy regression step, the aggregator must be a feature-wise sum to conserve information about the number of nodes contained in the cluster. The position of the cluster is taken at random from one of the nodes it contains. This allows to keep the graph embedded in the known detector geometry (see section 3.4) while preserving the locality of the cluster, since the length of the remaining edges makes the induced error negligible.Finally, for the reconnection step, the clusters inherit their adjacencies from their nodes, i.e. two clusters are neighbours if they each contain neighbouring nodes. The length of these edges is then re-computed to smoothen the accumulation of errors due to the random choice of cluster positions.

3.4 Graph Readout Pooling

After the convolution and pooling layers, it is necessary to flatten the graph representation if the output of convolution needs to be fed into a multi-layer perceptron (MLP). This operation,



(a) (b)

Figure 5: While an image with an expected and ordered structure (4a) is easily interpreted, a random order (4b) makes it uninterpretable.

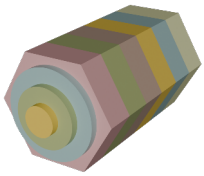


Figure 6: Radial readout regions of the detector respecting the rotational symmetry of the detector.

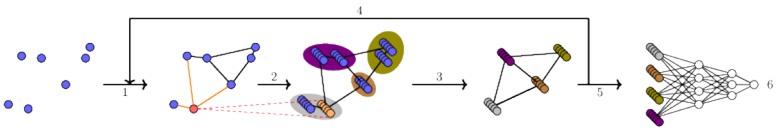


Figure 7: Typical message passing GNN pipeline. (1) Graph generation (2) Message passing convolution - Nodes collect "messages" from their neighbours (3) Pooling (4) Repeat CP layers (5) Readout (6) Objective (PID or Energy regression)

known as readout (or readout pooling), can be quite tricky, as graph-like data usually does not have any consistent ordering and may have variable sizes. The importance of ordering for the readout operation is illustrated in figure 5. The same applies to the readout of graphs and the ordering of nodes [16]. Our approach uses the fact that we know the detector geometry, and kept the nodes embedded (through the hidden features) in that geometry (see "reduction" paragraph of Sec. 3.3) to re-embed the graph back into the detector, and divide the detector into different *readout regions*. Using the same argument and procedure as in section 3.3, all nodes within the same readout region are feature-wise pooled together, and the obtained feature of every region are flattened following a known order. It is however important that this voxelisation of the detector respects the geometry of the detector (see figure 6). Indeed, the HGCAL model that we considered in this study has a rotational symmetry. If we rotated an electron shower in the detector, while it is the same input, if the readout does not respect that symmetry in the detector the model would have to learn both showers as two different inputs with identical output. This would hinder the training time and performance greatly. Note that the detector being composed of two different types of detector, electromagnetic and hadronic calorimeters, we can adapt the granularity of the readout to the task at hand: in energy regression tasks for electrons or photons, where there is no interaction with the hadronic part of the detector, a higher granularity of the readout is only needed in the ECAL.

3.5 Single Particle Reconstruction Pipeline

All the elements presented above allow us to design a pipeline for the reconstruction (particle ID and energy regression) of calorimeter events. The pipeline aims at identifying the detected particle and regressing its energy. The workflow is illustrated in figure 8. The first step is to generate graphs from raw detector data, using the PT-KNN algorithm. A first pipeline is trained for particle classification. Because electromagnetic and hadronic showers have very different signatures in the detector, there are two energy regression GNNs, for either of these signatures. In the case of muons, their energy when passing through the detector is in the

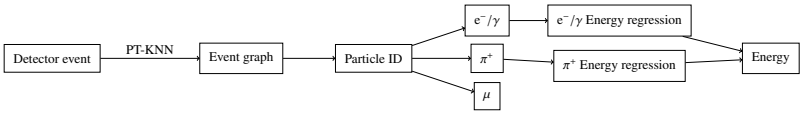


Figure 8: Algorithmic workflow of the single particle full reconstruction pipeline

minimum ionisation range [17], making it very hard to regress their energy, especially without a magnetic field, as is the case in this simulation. All pipelines have the same structure of repeated convolution and pooling layers, followed by a readout layer for the input of an MLP. Their exact definition slightly differ, most importantly by the choice of threshold for Treclus, either dataset specific or generalist (all particles at once), and in the granularity of the readout, chosen to be maximal in the regions of interest (whole detector, ECAL or HCAL) while not imposing an MLP size that would be too big. For all GNN pipelines, we apply 3 layers of convolution and pooling. At every CP layer, the number of node features doubles while the number of nodes is divided by two. For readout, the granularities depend on the task at hand. For the particle ID, we need a granular readout for the full detector, both longitudinally (for e^-/γ and π^+ classification) and radially (to distinguish μ, which do not spread radially in the detector). As such the ECAL is divided in 3 concentric region, while the HCAL has 4 longitudinal regions, divided in respectively 3, 3, 2 and 2 concentric regions. For energy regression tasks, we divide the concerned part of the detector (ECAL or HCAL) with a granular readout (3 to 6 longitudinal slices, dependent on the length of the region, divided each in 3 concentric cylinders), while keeping the rest of the detector as a single readout region. Finally, the MLP is funnel-shaped comprised of 6 layers with leaky ReLU activation.

4 Results

The models were trained on pipelines with 3 graph convolution layers and an MLP with 6 hidden layers, for a total of $\sim 10^4$ parameters. The training set for the classifier was composed of 5×10^5 events. The training sets for either regression networks were trained from a set of 2×10^6 events, labelled as either e^-/γ or π^+ by the previously trained classifier ($\sim 10^6$ graphs per regressor). We first look at the classifying performance of the GNN pipeline presented above. As can be guessed from Figure 1, this is mostly a density problem and the choice of readout regions is the key element of this task. From the confusion matrix (table 2) we can see a good classification performance. If we look at the raw detector output of misclassified elements (figure 9), we observe that most errors are due to outlier events, such as electromagnetic showers leaking into the HCAL, electron induced hadrons, later detected in the HCAL or early showering pions. The training of the energy regression pipelines was done with events classified by the Particle ID pipeline as either e^-/γ or π^+. We define the response as the ratio $E_{\text{pred}}/E_{\text{true}}$ and use it as a metric for performance. We observe a Gaussian-like distribution centred around 1 (see figure 11). The difference in ECAL and HCAL sampling fractions mean electromagnetic showers have a smaller standard deviation than for pions. We also look at the energy resolution $\sigma\left(\frac{E_{\text{pred}}}{E_{\text{true}}}\right)\Big/\mu\left(\frac{E_{\text{pred}}}{E_{\text{true}}}\right)$ as defined in [18]. In our simulation, no readout or systematic noise have been emulated and we therefore only consider $\frac{\sigma}{\mu} \propto \frac{1}{\sqrt{E}}$, as seen in figure 12. For the electromagnetic calorimeter, we find a stochastic term $S_{\text{GNN}} = 20.15\%$, close to the theoretical value S_{exp} of 20%.

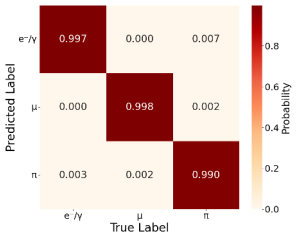
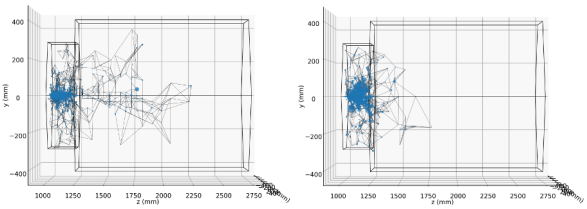


Table 2: Normalised confusion matrix of the particle classifier pipeline.



(a) (b)

Figure 9: Misclassified events with the typical signatures of the particles they are misidentified as. (8a) An electron induced hadronic shower (8b) An early showering pion ($\pi^+ \to \pi^0 \to \gamma\gamma$) [8].

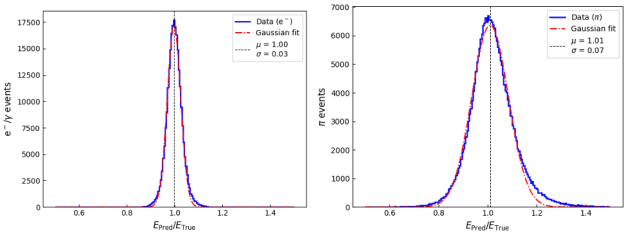


Figure 11: Energy response $E_{\text{pred}}/E_{\text{true}}$.

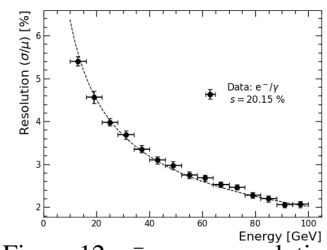


Figure 12: e^- energy resolution.

5 Conclusion

Our framework, by its generic and detector agnostic nature, can be generalised to other problems in high granularity particle detectors with a non-regular architecture, such as diffuse supernova neutrino background searches in Super and Hyper-Kamiokande [19] or overlapping object segmentation in granular detectors (HGCAL or Hyper-Kamiokande like). We presented a message passing graph convolution neural network architecture for particle identification and energy regression in high granularity calorimeters. We introduced the use of Proximity Tables and the PT-KNN algorithm for optimised graph construction. In particular, we discussed the improved average time complexity compared to the classical KNN algorithm. We also discussed the possibility to reduce the size of proximity tables for PT-KNN implementation in resource constrained environments such as on FPGAs. We also presented our implementation of message passing convolution and pooling layers. Treclus, our threshold based clustering algorithm for geometric pooling of graph has proven to be very adapted to coarsen graphs while preserving the original global structure of the graph. We discussed the need for a graph readout pooling to feed the result of the convolution layers to an MLP. We implemented a segmentation of the detector based on the geometry of the physical processes at hand to flatten the graph structure in a fixed size tensor while preserving the symmetries of the problem. Finally, we discussed the state of the art level classification and energy regression capacities of our GNN implementation. Looking at the energy resolution of our algorithm, we recover the expected values, proving the validity of our approach.

6 Acknowledgements

We are grateful for the Agence Nationale de la Recherche, financing the OGCID project under funding ANR-21-CE31-0030.

References

[1] I. Zurbano Fernandez et al., High-Luminosity Large Hadron Collider (HL-LHC): Technical design report, **10/2020** (2020). 10.23731/CYRM-2020-0010

[2] CMS-collaboration, Tech. rep., CERN, Geneva (2017), https://cds.cern.ch/record/2293646

[3] Z. Li et al., A survey of convolutional neural networks: Analysis, applications, and prospects, IEEE Transactions on Neural Networks and Learning Systems **PP**, 1 (2021). 10.1109/TNNLS.2021.3084827

[4] Z. Zhang et al., Deep learning on graphs: A survey, IEEE Transactions on Knowledge and Data Engineering **34**, 249 (2022). 10.1109/TKDE.2020.2981333

[5] S.R. Qasim et al., Learning representations of irregular particle-detector geometry with distance-weighted graph networks, The European Physical Journal C **79** (2019).

[6] N. Choma et al., Graph Neural Networks for IceCube Signal Classification, in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2018), pp. 386–391

[7] E. Becheva et al., High granularity calorimetry d2 dataset, https://zenodo.org/records/14260279

[8] R. Wigmans, Calorimetry: Energy Measurement in Particle Physics, International series of monographs on physics (Oxford University Press, 2017), ISBN 9780198786351

[9] J. Gilmer et al., Neural message passing for Quantum chemistry, in *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (JMLR.org, 2017), ICML'17, p. 1263–1272

[10] I. Goodfellow et al., Deep Learning (MIT Press, 2016), http://www.deeplearningbook.org

[11] F.N. Iandola et al., Densenet: Implementing efficient convnet descriptor pyramids, ArXiv **abs/1404.1869** (2014).

[12] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, in *Advances in Neural Information Processing Systems*, edited by D. Lee et al. (Curran Associates, Inc., 2016), Vol. 29, https://proceedings.neurips.cc/paper_files/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

[13] D. Grattarola et al., Understanding pooling in graph neural networks, IEEE Transactions on Neural Networks and Learning Systems **35**, 2708 (2024). 10.1109/TNNLS.2022.3190922

[14] L. Lovász, M. Plummer, Matching Theory, AMS Chelsea Publishing Series (AMS Chelsea Pub., 2009), ISBN 9780821847596

[15] S. Behnezhad et al., Approximating Maximum Matching Requires Almost Quadratic Time, in *Proceedings of the 56th Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, NY, USA, 2024), STOC 2024, p. 444–454, ISBN 9798400703836, https://doi.org/10.1145/3618260.3649785

[16] M. Zhang et al., An end-to-end deep learning architecture for graph classification, **32** (2018). 10.1609/aaai.v32i1.11782

[17] R.L. Workman et al. (Particle Data Group), Review of particle physics, PTEP **2022**, 083C01 (2022). 10.1093/ptep/ptac097

[18] C.W. Fabjan, F. Gianotti, Calorimetry for particle physics, Rev. Mod. Phys. **75**, 1243 (2003). 10.1103/RevModPhys.75.1243

[19] J.F. Beacom, M.R. Vagins, Antineutrino spectroscopy with large water Čerenkov detectors, Phys. Rev. Lett. **93**, 171101 (2004). 10.1103/PhysRevLett.93.171101