Eur. Phys. J. C (2019) 79:608
https://doi.org/10.1140/epjc/s10052-019-7113-9

THE EUROPEAN PHYSICAL JOURNAL C

Special Article - Tools for Experiment and Theory

Learning representations of irregular particle-detector geometry with distance-weighted graph networks

Shah Rukh Qasim[1,2,a], **Jan Kieseler**[1,b], **Yutaro Iiyama**[1,c], **Maurizio Pierini**[1,d]

[1] CERN, Geneva, Switzerland
[2] National University of Sciences and Technology, Islamabad, Pakistan

Received: 7 March 2019 / Accepted: 5 July 2019 / Published online: 18 July 2019

Abstract We explore the use of graph networks to deal with irregular-geometry detectors in the context of particle reconstruction. Thanks to their representation-learning capabilities, graph networks can exploit the full detector granularity, while natively managing the event sparsity and arbitrarily complex detector geometries. We introduce two distance-weighted graph network architectures, dubbed GarNet and GravNet layers, and apply them to a typical particle reconstruction task. The performance of the new architectures is evaluated on a data set of simulated particle interactions on a toy model of a highly granular calorimeter, loosely inspired by the endcap calorimeter to be installed in the CMS detector for the High-Luminosity LHC phase. We study the clustering of energy depositions, which is the basis for calorimetric particle reconstruction, and provide a quantitative comparison to alternative approaches. The proposed algorithms provide an interesting alternative to existing methods, offering equally performing or less resource-demanding solutions with less underlying assumptions on the detector geometry and, consequently, the possibility to generalize to other detectors.

1 Introduction

Traditionally, Machine Learning (ML) techniques are a key ingredient to event processing at particle colliders, employed in tasks such as particle reconstruction (clustering), identification (classification), and energy or direction measurement (regression) in calorimeters and tracking devices. The first applications of Neural Networks to High Energy Physics (HEP) date back to the '80s [1–4]. Starting with the MiniBooNE experiment [5], Boosted Decision Trees became the state of the art, and played a crucial role in the discovery of the Higgs boson by the ATLAS and CMS experiments [6]. Recently, a series of studies on different aspects of LHC data taking and data processing workflows have demonstrated the potential of Deep Learning (DL) in collider applications, both as a way to speed up current algorithms and to improve their performance. Nevertheless, the list of DL models actually deployed in the centralized workflows of the LHC experiments remains quite short.[1] Many of these studies, which are typically proof-of-concept demonstrations, are based on convolutional neural networks (CNN) [10], which perform computing vision tasks by applying translation-invariant kernels to *raw* digital images. CNN architectures applied on HEP data thus imposes a requirement for the particle detectors to be represented as regular arrays of sensors. This requirement, common to many of the approaches described in Sect. 2, creates problems for realistic applications of CNNs in collider experiments.[2]

In this work, we propose novel Deep Learning architectures based on graph networks to improve the performance and reduce the execution time of typical particle-reconstruction tasks, such as cluster reconstruction and particle identification. In contrast to CNNs, graph networks can learn optimal detector-hits representations without making specific assumptions on the detector geometry. In particular, no data preprocessing is required, even for detectors with irregular geometries. We consider the specific case of

[a] e-mail: 14beesqasim@seecs.edu.pk
[b] e-mail: jan.kieseler@cern.ch
[c] e-mail: yutaro.iiyama@cern.ch
[d] e-mail: maurizio.pierini@cern.ch

[1] As an example, at the moment such a list for the CMS experiment consists of a set of b-tagging algorithms [7,8] and a data quality monitoring algorithm for the muon drift tube chambers [9]. Other applications exist at the analysis level, downstream from the centralized event processing. In data analyses, one typically considers abstract four-momenta and not the low-level quantities such as detector hits, making the use of DL techniques easier.

[2] The picture is completely different in other HEP domains. For instance, CNNs have been successfully deployed in neutrino experiments, where the regular-array assumption meets the geometry of a typical detector.

particle reconstruction in calorimeters, for which this characteristic of graph networks may become especially relevant in the near future. In view of the High-Luminosity LHC phase, the endcap calorimeter of the CMS detector will be replaced by a novel-design digital calorimeter, the High Granularity Calorimeter (HGCAL), consisting of arrays of hexagonal silicon sensor cells interleaved with absorber layers [11]. Being positioned close to the beam pipe and exposed to $\sim$ 200 proton-proton collisions on average per bunch crossing, this detector will be characterized by high occupancy over its large number of readout channels. Downstream in the data processing pipeline, the unprecedented number of sensors and their geometry will cause an increase in event size and consequently the computational needs, necessitating novel data processing approaches given the expected computing limitations [12]. The detector we consider in this study, described in detail in Sect. 4, is loosely inspired by the HGCAL geometry. In particular, it features a similarly irregular sensor structure, with sensor sizes varying with the detector depth as well as within a single layer. On the other hand, the HGCAL hexagonal sensors were traded for square-shaped sensors, in order to keep the computing resources needed to generate the training data set within a manageable limit.

As a benchmark application, we consider the basis for all further particle reconstruction tasks in a calorimeter: clustering of the recorded energy deposits into disentangled showers from individual particles. To this purpose, we introduce two novel distance-weighted graph network architectures, the GarNet and the GravNet layers, which are designed to provide a good balance between performance and computing resources needs for inference. While our discussion is limited to a calorimetry-related problem, the design of these new layer architectures is such that it automatically generalizes to any kind of sparse data, such as *hits* collected by a typical tracking device or reconstructed particle candidates inside a hadronic jet. We believe that architectures of this kind are more practical to deploy in a realistic experimental environment and could become relevant for the LHC experiments, both for offline and real-time event processing and selection as well as shower simulation.

This paper is structured as follows: Sect. 2 reviews related previous works. In Sect. 3, we describe the GravNet and GarNet architectures. Section 4 describes the data set used for this study. Section 5 introduces the metric used to optimize the networks. Section 6 describes the models. The results are presented in Sects. 7 and 8 in terms of accuracy and computational efficiency, respectively. Conclusions are presented in Sect. 9.

2 Related work

In recent years, DL models, and in particular CNNs, have become very popular in different areas of HEP. CNNs were successfully applied to calorimeter-oriented tasks, including particle identification [11,13–16], energy regression [11,13,15,16], hadronic jet identification [17–20], fast simulation [13,21–24] and pileup subtraction in jets [25]. Many of these works assume a simplified detector description: the detector is represented as a regular array of sensors expressed as 2D or 3D images, and the problem of overlapping regions at the transition between detector components (e.g. barrel and endcap) is ignored. Sometimes the fixed-grid pixel shape is intended to reflect the typical angular resolution of the detector, which is implicitly assumed to be a constant, while in reality it depends on the energy of the incoming particle.

In order to overcome this practical difficulty with CNN architectures, different HEP-related studies investigating alternative architectures have been performed. In the context of jet identification, several authors studied models based on recurrent [7,8,26] and recursive [27] networks, graph networks [28], and DeepSets [29]. Recurrent architectures have also been studied for event classification [30]. In general, these approaches take as input a particle-based representation of an event and thus are easier to apply in applications running after a global event reconstruction based on a particle-flow algorithm [31,32].

Outside the HEP domain, overcoming the necessity for a regular structure motivated original research to use graph-based networks [33], which in general are suited for processing point-wise data with no regular structure by representing them as vertices in a graph. A comprehensive overview of various graph-based networks can be found in Ref. [34]. In a typical implementation of a graph-based network, the vertices are connected according to some predefined criteria at the preprocessing stage. The connections between the vertices (edges) then define paths of information exchange [35,36]. In some cases, the edge and vertex properties are used to infer attention (weight) assigned to each neighbour during this information exchange, while leaving the neighbour relations (adjacency matrix) unchanged [37]. Some of these architectures have already been considered for collider physics, in the context of jet tagging [38], event topology classification [39], and for pileup subtraction [40].

Particularly interesting for irregular detectors are, however, networks that are capable of learning the geometry, as studied in combination with message passing [41]. Within this approach, the adjacency matrix is trainable. In other words, the neighbour relations, which encode the detector geometry, are not imposed at the preprocessing stage but are inferred from the input data. Although this approach is promising, its downside is the need to connect all vertices with each other, which makes it computationally challeng-

ing for graphs with a large number of vertices as the memory requirement becomes forbiddingly high. This problem is overcome by defining only a subset of connections between neighbours in a learnable space representation, where the edge properties of each vertex to a limited number of its neighbours are used to calculate a new feature representation per vertex, which is then passed to the next layer of similar structure [42]. This approach is implemented in the EdgeConv layer and the corresponding DGCNN model [42]. The neighbours are selected based on the new vertex features, which makes it particularly challenging to create a gradient for training with respect to the neighbour selection. The DGCNN model works around this issue by using the edge features themselves. However, due to the dynamic calculation of neighbour relations in high-dimensional space, this network still requires substantial computing resources, which would make its use for triggering purposes in collider detectors unfeasible.

3 The GravNet and GarNet layers

The neural network layers proposed in this study are designed to provide competitive performance on particle reconstruction tasks while dealing with data sparsity in an efficient way. These architectures aim to keep a trainable space representation at minimal computational costs. The layers receive as input a $B \times V \times F_{\text{IN}}$ data set, consisting of a batch of B examples, each represented by a set of V detector hits, embedded in the network set through F_{IN} features. For instance, the F_{IN} features could include the Cartesian coordinates of a given sensor, its address (layer number, module number, etc.), the sensor time stamp, the recorded energy, etc.

A pictorial representation of the operations performed by the two layers is shown in Fig. 1. For both architectures, the first step is to apply a dense[3] neural network to each of the V detector hits, deriving from the F_{IN} features two output arrays: the first array (S) is interpreted as a set of coordinates in some learned representation space (for the GravNet layer) or as the distance between the considered vertex and a set of S aggregators (for the GarNet layer); the second array (F_{LR}) is interpreted as a learned representation of the vertex features. At this point, a given input example of initial dimension $V \times F_{\text{IN}}$ is converted into a graph with V vertices in the abstract space identified by S. Each vertex is represented by the F_{LR} features, derived from the initial inputs. The projection from the $V \times F_{\text{IN}}$ to this graph is linear, with trainable weights and bias vectors.

The main difference between the GravNet and the GarNet architectures is in the way the V vertices are connected when building the graph. In the case of the GravNet layer, the Euclidean distances d_{jk} between (j, k) pairs of vertices in the S space are used to associate to each vertex its closest N neighbors. In the case of the GarNet layer, the graph is built connecting each of the V vertices to a set of $\dim(S)$ aggregators. What is learned by S, in this case, is the distance between a vertex and each of the aggregators.

Once the edges of the graph are built, each vertex (aggregator) of the GravNet (GarNet) layer collects the information associated with the F_{LR} features across its edges. This is done in three steps:

1. The quantities
$$\tilde{f}^i_{jk} = f^i_j \times V(d_{jk}) \tag{1}$$
are computed for the feature f^i of each of the vertices v_j connected to a given vertex or aggregator v_k, scaling the original value by a *potential*, function of the euclidean distance d_{jk}, giving the *gravitational network* GravNet its name. The potential function is introduced to enhance the contribution of close-by vertices. For this reason, V has to be a decreasing function of d_{jk}. In this study, we use a Gaussian potential $V(d_{jk}) = \exp(-d^2_{jk})$ for the GravNet layer[4] and an exponential potential $V(d_{jk}) = \exp(-|d_{jk}|)$ for the GarNet layer.
2. The $\tilde{f}^i_{jk}$ functions computed from all the edges associated to a vertex of aggregator v_k are combined, generating a new feature $\tilde{f}^i_k$ of v_k. For instance, we consider the average of the $\tilde{f}^i_{jk}$ across the j edges and their maximum. In our case, it was particularly crucial to extend the choice of aggregator functions beyond the maximum, which was already explored for similar architectures [42]. In fact, the mean function (as any other similar function) helped improve the convergence of the model, by taking into account the contribution of all the vertices.
3. Each adopted combination rule in the previous step generates a new set of features $\tilde{F}_{\text{LR}}$. All of them are concatenated to the original F_{IN} vector. This extended vector is transformed into a set of F_{OUT} new vertex features, using a fully connected dense layer with tanh activation. The concatenation is done for each initial vertex. In the case of the GarNet layer, this requires an additional step of passing the $\tilde{f}^i_k$ features of the v_k aggrega-

[3] Here and in the following, *dense* layer refers to a learnable weight-matrix multiplication and bias vector addition with respect to the last feature dimension, with shared weights over all other dimensions. In this case, the weights and bias are applied to the vertex features F_{IN} and shared over the vertices V. This can also be thought of as a 2D convolution with a 1×1 kernel.

[4] A gravitational potential $(-1/d)$ has singularities at $d = 0$ and therefore cannot be used, however the potential we are using has a similar qualitative effect of pulling together vertices.

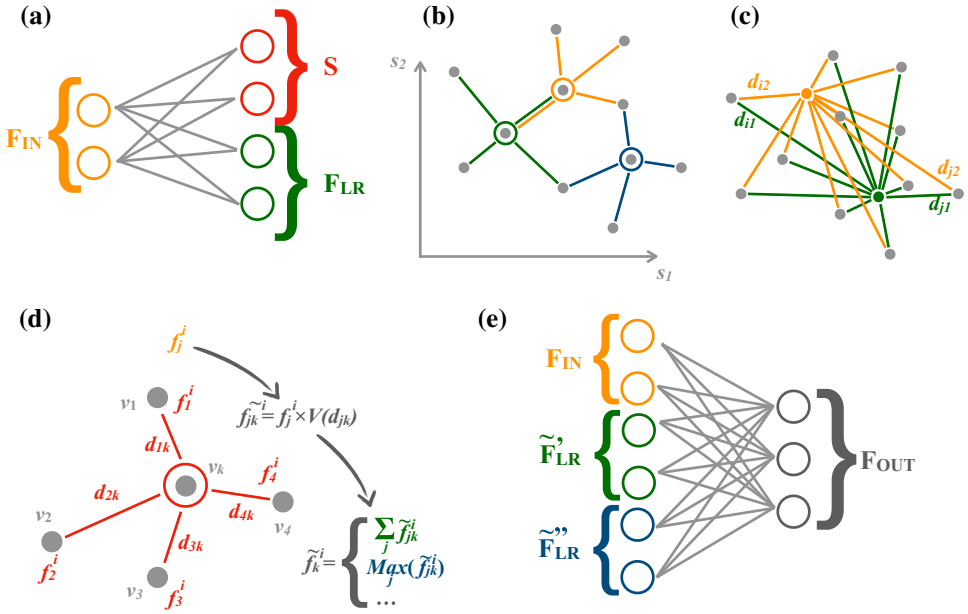


Fig. 1 Pictorial representation of the data flow across the GARNET and the GRAVNET layers. **a** The input features F_{IN} of each $v_i \in V$ are processed by a dense neural network with two output arrays: a set of learned features F_{LR} and spatial information S in some learned representation space. **b** In the case of the GRAVNET layer, the S quantities are interpreted as the coordinates of the vertices in some abstract space. The graph is built in this space, connecting each v_i to its N closest neighbors (N = 4 in the figure), using the euclidean distance d_{ij} between the vertices to rank the neighbors. **c** In the case of the GARNET layer, the S quantities are interpreted as the distances between the vertices and a set of S aggregators in some abstract space. The graph is then built connecting each v_i vertex to each a_j aggregator, and the S quantities are the d_{ij} euclidean distances. **d** Once the graph structure is established, the f_j^i features of the v_j vertices connected to a given vertex or aggregator v_k are converted into the $\tilde{f}_{jk}^i$ quantities, through a potential (function of d_{jk}). The corresponding information is then gathered across the graph and turned into a new feature $\tilde{f}_k^i$ of v_k (e.g. summing over the edges, or taking the maximum). **e** For each choice of gathering function, a new set of features $\tilde{f}_k^i \in \tilde{F}_{\text{LR}}$ is generated. The $\tilde{F}_{\text{LR}}$ vector is concatenated to the initial F_{IN} vector. The resulting feature vector is given as input to a dense neural network with tanh activation, which returns the output representation F_{OUT}

tors back to the initial vertices, weighted by the $V(d_{jk})$ potential. This information exchange of the *garnered* information through the aggregators defines the GARNET name.

The full process transforms the initial $B \times V \times F_{\text{IN}}$ data set into a $B \times V \times F_{\text{OUT}}$ data set. As common with graph networks, the main advantage comes from the fact that the F_{OUT} output (unlike the F_{IN} input) carries collective information from each vertex and its surrounding, providing a more informative input to downstream processing. Thanks to the distinction between learned space information S and learned features F_{LR}, the dimensionality of connections in the graph is kept under control, resulting in a smaller memory consumption than, for instance, the EDGECONV layer.

The two layer architectures and the models based on them, described in the following sections, are implemented in TensorFlow [43].[5]

[5] The code for the models and layers can be found in https://github.com/jkiesele/caloGraphNN.

4 Data set

The data set used in this paper is based on a simplified calorimeter with irregular geometry, built in GEANT4 [44]. The calorimeter is made entirely of Tungsten, with a width of 30 cm × 30 cm in the x and y directions and a length of 2 m in the longitudinal direction (z), which corresponds to 20 nuclear interaction lengths. The longitudinal dimension is further split into 20 layers of equal thickness. Each layer contains square sensor cells, with a fine segmentation in the quadrant with $x > 0$ and $y > 0$ and a lower granularity elsewhere. The total number of cells and their individual sizes vary by layer, replicating the basic features of a slightly irregular calorimeter. For more details, see Fig. 2 and Table 1.

Charged pions are generated at $z = -2$ m; the x and y coordinates of the generation vertex are randomly sampled within $|x| < 5$ cm and $|y| < 5$ cm. The x and y components of the particle momentum are set to 0, while the z component is sampled uniformly between 10 and 100 GeV. The particles therefore impinge the calorimeter front face perpendicularly and shower along the longitudinal direction.

The resulting total energy deposit in each cell, as well as the cell position, width, and layer number, are recorded

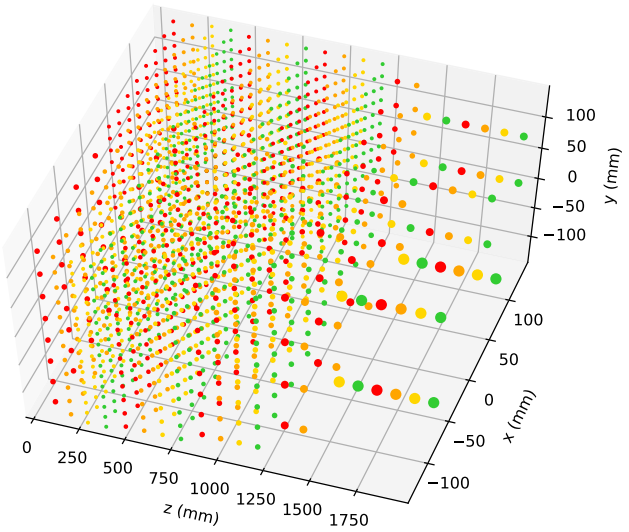


Fig. 2 Calorimeter geometry. The markers indicate the centre of the sensors, their size the sensor size. Layers are colour-coded for better visualisation

Table 1 Number of cells in the finely segmented quadrant and the rest of the layer, for the benchmark calorimeter geometry described in the text

Layer	Cells ($x > 0$, $y > 0$)	Cells elsewhere
0	64	48
1	64	108
2–3	100	192
4–7	64	108
8–11	64	48
12–13	16	12
14–19	4	3

for each event. These quantities correspond to the F_{IN} feature vector given as input to the graph models (see Sect. 3). Each example consists of the result of two overlapping showers. Cell by cell, the energy of two showers is summed and the fraction belonging to each of the showers in each cell is defined as the ground truth. In addition, the position of the largest energy deposit per shower is recorded. If this position is the same for the two overlapping showers, they are considered not separable and the event is discarded. This applies to about 5% of the events.

In total 16,000,000 events are generated. Out of these, 100,000 are used for validation and 250,000 for testing. The rest is used for training.

5 Clustering metrics

To identify individual showers and use their properties, e.g. for a subsequent particle identification task, the energy deposits should be clustered so that overlapping parts are identified without removing important parts of the original shower. Therefore, the clustering algorithms should predict the energy fraction of each sensor belonging to each shower. Lower energy deposits are slightly less important. These considerations define the loss function:

$$L = \sum_k \frac{\sum_i \sqrt{E_i t_{ik}}(p_{ik} - t_{ik})^2}{\sum_i \sqrt{E_i t_{ik}}}, \tag{2}$$

where p_{ik} and t_{ik} are the predicted and true energy fractions in sensor i and shower k. These are weighted by the square root of $E_i t_i$, which is the total energy deposit in sensor i belonging to shower k, to introduce a mild energy scaling within each shower.

In addition, in each event we randomly label one of the showers as the *test* shower and the other as the *noise* shower, and define the clustering energy response R_k of shower k (k = test, noise) as:

$$R_k = \frac{\sum_i E_i p_{ik}}{\sum_i E_i t_{ik}}. \tag{3}$$

6 Models

The models need to incorporate neural network layers to identify localized structures as well as to perform information exchange globally between the sensors. This can be achieved either by multiple message passing iterations between neighbouring sensors or a direct global information exchange. Here, we employ a combination of both. The input to all models is an array of sensors, each holding its recorded energy deposits, global position coordinates, sensor size, and layer number. We compare three different graph-network approaches to a CNN based approach (Binning), presented as a baseline. Each model is designed to contain approximately 100,000 free parameters. The model structure is as follows:

– *Binning:* a regular grid of $20 \times 20 \times 20$ pixels is imposed on the irregular geometry. Each pixel contains the information of at most one sensor.[6] The information is concatenated to the mean of these features in all pixels, preprocessed by one $1 \times 1 \times 1$ CNN layer with 20 nodes, and then fed through eight blocks of CNN layers. Each block consists of a CNN layer with a kernel of $7 \times 7 \times 1$ followed by a layer with a kernel of $1 \times 1 \times 3$, each containing 14 filters. The output of each block is passed to the next block and simultaneously added to a list of all block outputs. All CNN layers employ tanh activation functions. Finally, the full list of block outputs per pixel

[6] Alternative configurations with more than one sensor per pixel were also investigated and showed similar performance.

is reshaped to represent the vertices of the graph and fed through a dense layer with 128 nodes and ReLU activation. Different CNN models have also been tested and showed similar or worse performance.

– *DGCNN model:* Adapting the model proposed in Ref. [42] to our problem, the sensor features are interpreted as positions of points in a 16-dimensional space and fed through one global space transformation followed by four blocks comprising one EdgeConv layer. Our EdgeConv layer has a similar configuration as in Ref. [42], with 40 neighbouring vertices and three internal dense layers with ReLu activation acting on the edges with 64 nodes each. The output of the EdgeConv layer is concatenated with its mean over all vertices and fed to one dense layer with 64 nodes and ReLu activation which concludes the block. The output of each block is passed to the next block and simultaneously added to a list of all block outputs per vertex together with the mean over vertices. This list is finally fed to a dense layer with 32 nodes and ReLU activation.
– *GravNet model:* The model consists of four blocks. Each block starts with concatenating the mean of the vertex features to the vertex features, three dense layers with 64 nodes and tanh activation, and one GravNet layer with $S = 4$ coordinate dimensions, $F_{\mathrm{LR}} = 22$ features to propagate, and $F_{\mathrm{OUT}} = 48$ output nodes per vertex. For each vertex, 40 neighbours are considered. The output of each block is passed as input to the next block and added to a list containing the output of all blocks. This determines the full vector of vertex features passed to a final dense layer with 128 nodes and ReLU activation.
– *GarNet model:* The original vertex features are concatenated with the mean of the vertex features and then passed on to one dense layer with 32 nodes and tanh activation before entering 11 subsequent GarNet layers. These layers contain $S = 4$ aggregators, to which $F_{\mathrm{LR}} = 20$ features are passed, and $F_{\mathrm{OUT}} = 32$ output nodes. The output of each layer is passed to the next and added to a vector containing the concatenated outputs of each GarNet layer. The latter is finally passed to a dense layer with 48 nodes and ReLU activation.

In all cases, each output vertex of these model building blocks is fed through one dense layer with ReLU activation and three nodes, followed by a dense layer with two output nodes and softmax activation. This last processing step determines the energy fraction belonging to each shower. Batch normalisation [45] is applied in all models to the input and after each block.

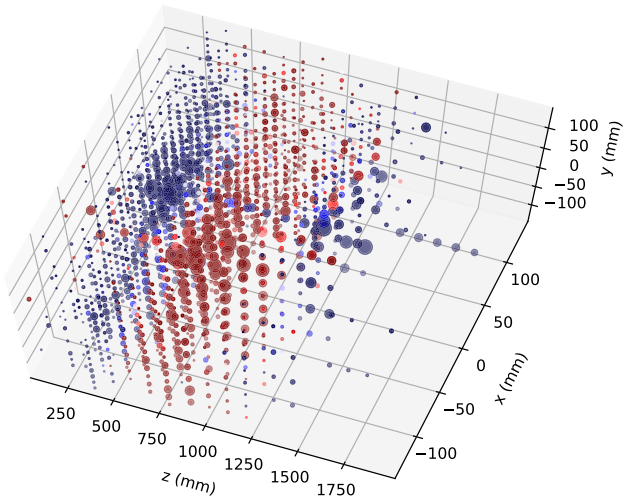
All models are trained on the full training data set using the Adam optimizer [46] and an initial learning rate of about 3×10^{-4}, the exact value depending on the model. The learning rate is reduced exponentially in steps to the minimum of 3×10^{-6} after 2 million iterations. Once the learning rate has reached the minimum level, it is modulated by 10% at a fixed frequency, following the method proposed in Ref. [47].

7 Clustering performance

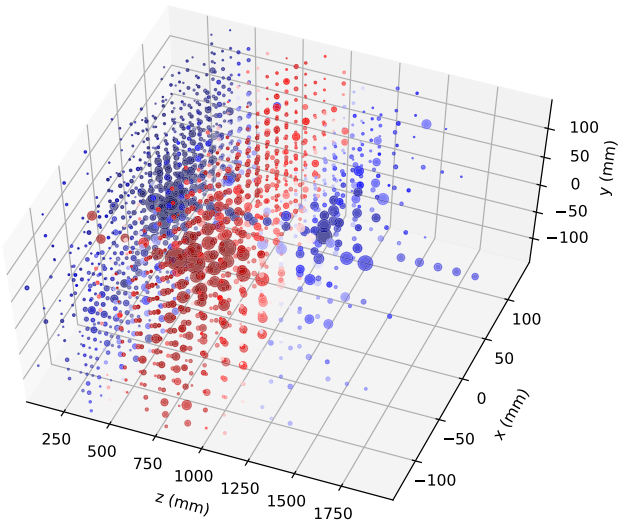
All approaches described in Sect. 6 perform well for clustering purposes. An example is shown in Fig. 3, where two charged pions with an energy of approximately 50 GeV enter the calorimeter. One pion loses a significant fraction of energy in an electromagnetic shower in the first calorimeter layers. The remaining energy is carried by a single particle passing the central part of the calorimeter before showering. The second pion passes the first layers as a minimally ionizing particle and showers in the central part of the calorimeter. Even though the two showers largely overlap, the GravNet network (shown here as an example) is able to identify and separate the two showers very well. The track within the calorimeter is well identified and reconstructed and the energy fractions properly assigned, even in the parts where the two showers heavily overlap. Similar performance can be observed with the other investigated methods.

Quantitatively, the models are compared with respect to multiple performance metrics. The first two are the mean and the variance of the loss function value (μ_L and σ_L) computed according to Eq. (2) over the test events. The mean and the variance of the test shower response (μ_R and σ_R), where the response is defined in Eq. (3), are also compared. While the test shower response follows an approximately normal distribution over majority of the test events, a small outlier population, where the shower clustering fails, are seen to lead μ_R and σ_R to misparametrize the core of the distribution. Therefore, response kernel mean μ_R^* and variance σ_R^*, restricted to test showers with response between 0.2 and 2.8, are added to the set of evaluation metrics. In addition, we also compare the clustering accuracy (A), defined as the fraction of showers with response between 0.7 and 1.3. Finally, the above set of metrics is duplicated, with the second set using only the sensors with energy fractions between 0.2 and 0.8 in the computation of the loss function and the response. The second set of metrics characterizes the performance of the models in particularly challenging case of reconstructing significantly overlapping clusters. The two sets of metrics are called *inclusive* and *overlap-specific* in the remainder of the discussion.

The metric values are listed in Table 2. Comparing the inclusive metrics, it can be seen that the GravNet layer outperforms the other approaches, including even the more resource-intensive DGCNN model. The GarNet model performance is in between the DGCNN model and the binning approach in terms of reconstruction of individual shower hit fractions, parametrized by μ_L and σ_L. However, in charac-



(a) Truth



(b) Reconstructed

Fig. 3 Comparison of true energy fractions and energy fractions reconstructed by the GravNet model for two charged pions with an energy of approximately 50 GeV showering in different parts of the calorimeter. Colours indicate the fraction belonging to each of the showers. The size of the markers scales with the square root of the energy deposit in each sensor

teristics related to clustering response, the binning model outperforms the GarNet and DGCNN model slightly. On the other hand, with respect to overlap-specific metrics, the graph based approaches outperform the binning approach. The DGCNN and GravNet model perform equally well, and the GarNet model lies in between the binning approach and GravNet.

One should notice that part of the incorrectly predicted events are actually correctly clustered events in which the test shower is labelled as noise shower (shower swapping). Since the labelling is irrelevant in a clustering problem, this behavior is not a real inefficiency of the algorithm. We denote by s the fraction of events where this behaviour is observed. In Table 3, we calculate the loss for both choices and evaluate the performance parameters for the assignment that minimizes the loss. The binning model shows the largest fraction of swapped showers. The difference in response between the best-performing GravNet model and the GarNet model is enhanced, while the difference between the GravNet and DGCNN model scales similarly, likely because of their similar general structure.

In Fig. 4, the performance of the models are compared in bins of the test shower energy with respect to inclusive and overlap-specific μ_R and σ_R. For the inclusive metrics, the GravNet model outperforms the other models in the full range, and the GarNet model shows the worst performance, albeit in a comparable range. The resource-intensive DGCNN model lies in between GravNet and GarNet.

The overall upward bias in the response for lower shower energies warrants an explanation. This bias is a result of edge effects, induced by our choice of using an adapted mean-square error loss to predict a quantity bounded in [0, 1] (the energy fraction). This choice of loss function creates an expectation value larger than 0 at a peak value of 0 (and vice-versa at a fraction of 1), and therefore pushes the prediction away from being exactly 0 or 1, leading to an underestimation at high energies and an overestimation at low energies. The design of a customized loss function that eliminates this bias is left to future studies. For the moment, we are interested in a performance comparison between models, all affected by this bias.

For overlap-specific metrics, the edge effects are highly suppressed. The figures confirm that the graph-based models outperform the binning method at all test shower energies. It is also seen that the GravNet and the DGCNN model show similar performance.

8 Resource requirements

In addition to the clustering performance, it is important to take into account the computational resources demanded by each model during inference. The required inference time and memory consumption can have a significant impact on the applicability of the network for reconstruction tasks in constrained environments, such as the online and offline central-processing workflows of a typical collider-physics experiment. We evaluate the inference time t and memory consumption m for the models studied here on one NVIDIA GTX 1080 Ti GPU for batch sizes of 1 and 100, denoted as (t_1, m_1) and (t_{100}, m_{100}), respectively. The inference time is also evaluated on one Intel Xeon E5-2650 CPU core (t_{10}^{CPU}) for a fixed batch size of 10. As shown in Fig. 5, memory consumption and execution times differ significantly between the models.

Table 2 Mean and variance of loss, response, and response within the Gaussian kernel as well as clustering accuracy

	μ_L	σ_L	μ_R	σ_R	μ_R^*	σ_R^*	A
Inclusive							
Binning	0.191	0.017	1.083	0.183	1.046	0.057	0.867
DGCNN	0.174	0.012	1.082	0.179	1.045	0.052	0.881
GarNet	0.182	0.011	1.086	0.190	1.048	0.055	0.872
GravNet	0.172	0.012	1.077	0.173	1.042	0.049	0.886
Overlap-specific							
Binning	0.163	0.0045	1.005	0.099	1.004	0.096	0.697
DGCNN	0.154	0.0046	1.004	0.090	1.002	0.087	0.728
GarNet	0.157	0.0048	1.005	0.095	1.004	0.092	0.714
GravNet	0.156	0.0047	1.004	0.091	1.003	0.088	0.721

Table 3 Mean and variance of loss, response, and response within the Gaussian kernel as well as clustering accuracy corrected for shower swapping. The last column shows the fraction of swapped showers

	μ_L	σ_L	μ_R	σ_R	μ_R^*	σ_R^*	A	s [%]
Inclusive								
Binning	0.179	0.007	1.076	0.139	1.047	0.054	0.875	3.2
DGCNN	0.167	0.006	1.076	0.138	1.047	0.050	0.887	2.6
GarNet	0.176	0.006	1.081	0.149	1.049	0.054	0.877	2.5
GravNet	0.164	0.006	1.071	0.126	1.044	0.047	0.892	2.7
Overlap-specific								
Binning	0.160	0.0037	1.005	0.098	1.004	0.095	0.699	3.2
DGCNN	0.152	0.0038	1.003	0.089	1.002	0.086	0.729	2.6
GarNet	0.154	0.0040	1.005	0.094	1.003	0.091	0.715	2.5
GravNet	0.152	0.0039	1.004	0.090	1.003	0.087	0.722	2.7

The binning approach outperforms all other models, because of the highly optimized CNN implementations. The DGCNN model requires the largest amount of memory, while the model using the GravNet layers requires about 50% less. The GarNet model provides a good compromise of memory consumption with respect to performance. In terms of inference time, the binning model is the fastest and the graph-based models show a similar behaviour for small batch sizes on a GPU. The GarNet and the GravNet model benefit from parallelizing over a larger batch. In particular, the GarNet model is mostly sequential, which also explains the outstanding performance on a single CPU core, with almost a factor of 10 shorter inference time compared to the DGCNN model.

9 Conclusions

In this work, we introduced the GarNet and GravNet layers, which are distance-weighted graph networks capable of learning irregular patterns of sparse data, such as the detector hits in a particle physics detector with realistic geometry. Using as a benchmark problem the hit clustering in a highly granular calorimeter, we show how these network architectures offer a good compromise between clustering performance and computational resource needs, when compared to CNN-based and other graph-based networks. In the specific case considered here, the performance of the GarNet and GravNet models are comparable to the CNN and graph baselines. On the other hand, the simulated calorimeter in the benchmark study is only slightly irregular and can still be represented by an almost regular array. In more realistic applications, e.g. with the hexagonal sensors and the non-projective geometry of the future HGCAL detector of CMS, the difference in performance between the graph-based approaches and the CNN-based approaches is expected to increase further, making the GarNet approach a very efficient candidate for fast and accurate inference and the GravNet approach a good candidate for high-performance reconstruction with significantly less resource requirements but similar performance compared to the DGCNN model for a similar number of free parameters.

It should also be noted that the GarNet and GravNet architectures make no specific assumption on the structure

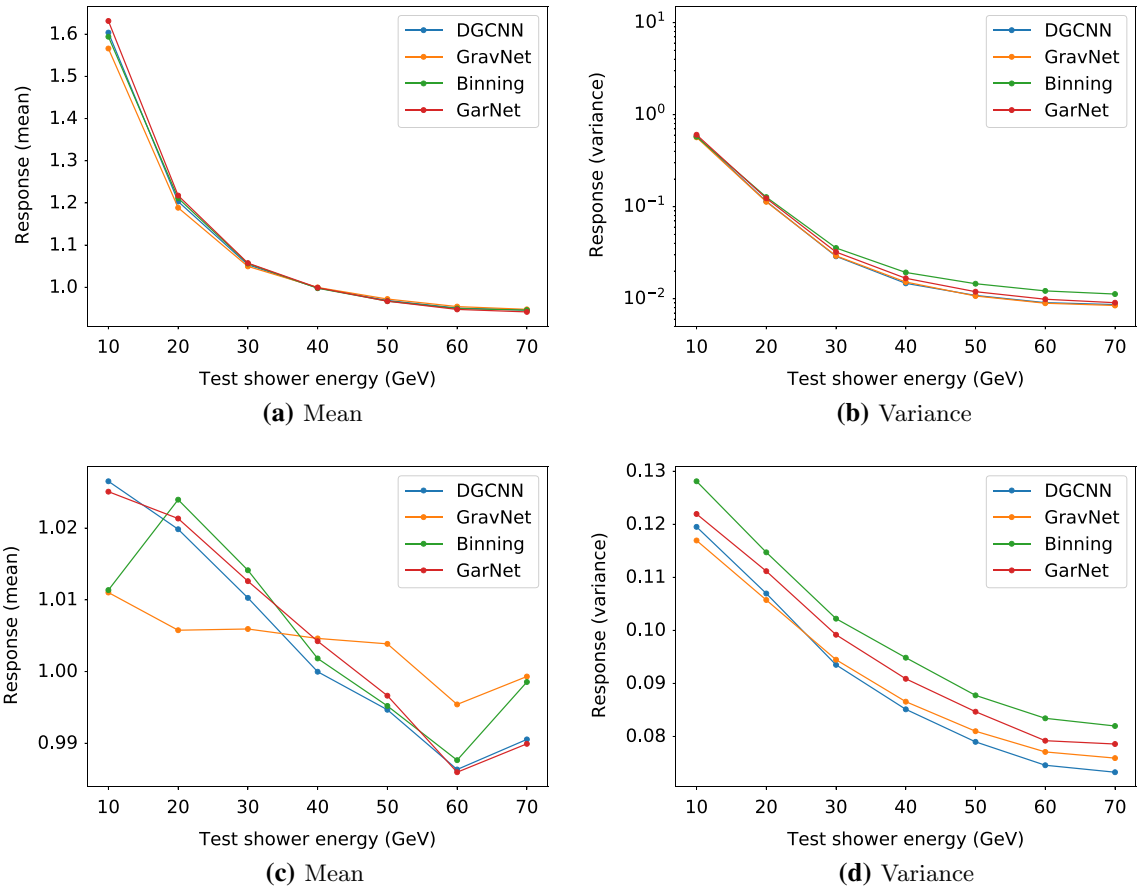


Fig. 4 Mean (left) and variance (right) of the test shower response as a function of the test shower energy for full shower (top) and for overlapping shower (bottom), computed summing the true deposited energy. Swapping of the showers is allowed here

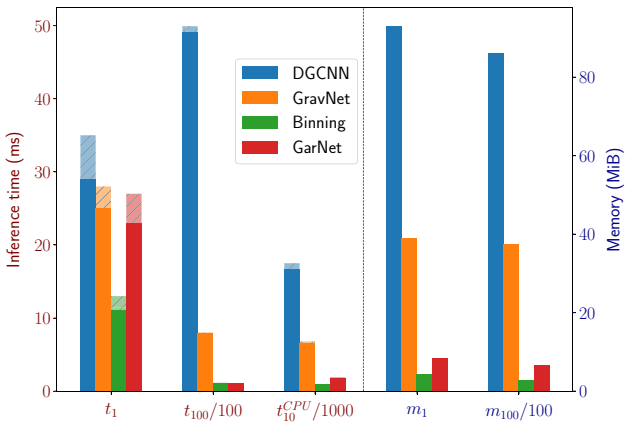


Fig. 5 Comparison of inference time for the network architectures described in the text, evaluated on CPUs and GPUs with different choices of batch size. The shaded area represents the $+1\sigma$ statistical uncertainty band

of the underlying data, and thus can be employed for many other applications related to particle and event reconstruction, such as tracking and jet identification. Exploring the extent of usability of these architectures will be the focus of follow-up work.

Note added

After the completion of this work, Ref. [28] appeared, discussing the application of a similar approach to the problem of jet tagging.

Acknowledgements We thank our CMS colleagues for many suggestions received in the development of this work. The training of the models was performed on the GPU clusters of the CERN TechLab and the CERN CMG group. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement no. 772369).

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Authors' comment: There is limited value in publishing the dataset for this rather specific problem of calorimeter clustering, given the size and format of the files. The dataset will be soon superseded by more realistic simulations of the future HGCAL calorimeter in CMS.]

Funded by SCOAP[3].

References

1. B.H. Denby, Neural networks and cellular automata in experimental high-energy physics. Comput. Phys. Commun. **49**, 429–448 (1988)
2. C. Peterson, Track finding with neural networks. Nucl. Instrum. Methods A **279**, 537–545 (1989)
3. P. Abreu et al., Classification of the hadronic decays of the Z0 into b and c quark pairs using a neural network. Phys. Lett. B **295**, 383–395 (1992)
4. B.H. Denby, Neural networks in high-energy physics: a ten year perspective. Comput. Phys. Commun. **119**, 219–231 (1999)
5. H.-J. Yang, B.P. Roe, J. Zhu, Studies of boosted decision trees for MiniBooNE particle identification. Nucl. Instrum. Methods A **555**, 370–385 (2005)
6. A. Radovic et al., Machine learning at the energy and intensity frontiers of particle physics. Nature **560**(7716), 41 (2018)
7. V. Khachatryan et al., CMS Phase 1 heavy flavour identification performance and developments, Technical report CMS-DP-2017-013 (2017)
8. V. Khachatryan et al., New developments for jet substructure reconstruction in CMS. Technical report CMS-DP-2017-027 (2017)
9. A.A. Pol et al., Detector monitoring with artificial neural networks at the CMS experiment at the CERN large hadron collider. Comput. Softw. Big Sci. **3**, 3 (2019)
10. A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017). https://doi.org/10.1145/3065386
11. CMS Collaboration, The phase-2 upgrade of the CMS Endcap calorimeter. Technical report CERN-LHCC-2017-023. CMS-TDR-019 (2017)
12. V. Khachatryan et al., Technical proposal for the phase-II upgrade of the CMS detector. Technical report CERN-LHCC-2015-010. LHCC-P-008. CMS-TDR-15-02 (2015)
13. F. Carminati et al., Calorimetry with deep learning: particle classification, energy regression, and simulation for high-energy physics. Deep Learning for Physical Sciences workshop at NIPS 2017 (2017). https://dl4physicalsciences.github.io/files/nips_dlps_2017_15.pdf
14. D. Guest, K. Cranmer, D. Whiteson, Deep learning and its application to LHC physics. Ann. Rev. Nucl. Part. Sci. **68**, 161–181 (2018)
15. L. De Oliveira, B. Nachman, M. Paganini, Electromagnetic showers beyond shower shapes. (2018). arXiv:1806.05667 [hep-ex]
16. A. Abada et al., FCC-hh: The Hadron Collider. Eur. Phys. J. Spec. Topics **228**(4), 755–1107 (2019). https://doi.org/10.1140/epjst/e2019-900087-0
17. J. Cogan et al., Jet-images: computer vision inspired techniques for jet tagging. JHEP **02**, 118 (2015)
18. P.T. Komiske, E.M. Metodiev, M.D. Schwartz, Deep learning in color: towards automated quark/gluon jet discrimination. JHEP **01**, 110 (2017)
19. L. de Oliveira et al., Jet-images—deep learning edition. JHEP **07**, 069 (2016)
20. P. Baldi et al., Jet substructure classification in high-energy physics with deep neural networks. Phys. Rev. D **93**(9), 094034 (2016)
21. L. de Oliveira, M. Paganini, B. Nachman, Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. Comput. Softw. Big Sci. **1**(1), 4 (2017)
22. M. Paganini, L. de Oliveira, B. Nachman, CaloGAN: simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. Phys. Rev. D **97**(1), 014021 (2018)
23. G. Rukhkhattak, S. Vallecorsa, F. Carminati, Three dimensional energy parametrized generative adversarial networks for electromagnetic shower simulation. in *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018), pp. 3913–3917
24. P. Musella, F. Pandolfi, Fast and accurate simulation of particle detectors using generative adversarial networks. Comput. Softw. Big Sci. **2**(1), 8 (2018)
25. P.T. Komiske et al., Pileup mitigation with machine learning (PUMML). JHEP **12**, 51 (2017)
26. ATLAS Collaboration, Identification of jets containing *b*-hadrons with recurrent neural networks at the ATLAS experiment. Technical report ATL-PHYS-PUB-2017-003 (2017)
27. G. Louppe et al., QCD-aware recursive neural networks for jet physics. JHEP **01**, 57 (2019)
28. H. Qu, L. Gouskos, ParticleNet: jet tagging via particle clouds. (2019). arXiv:1902.08570 [hep-ph]
29. P.T. Komiske, E.M. Metodiev, J. Thaler, Energy flow networks: deep sets for particle jets. JHEP **01**, 121 (2019)
30. T.Q. Nguyen et al., Topology classification with deep learning to improve real-time event selection at the LHC. (2018). arXiv:1807.00083 [hep-ex]
31. A.M. Sirunyan et al., Particle-flow reconstruction and global event description with the CMS detector. JINST **12**(10), P10003 (2017)
32. M. Aaboud et al., Jet reconstruction and performance using particle flow with the ATLAS detector. Eur. Phys. J. C **77**(7), 466 (2017)
33. F. Scarselli et al., The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009)
34. P.W. Battaglia et al., Relational inductive biases, deep learning, and graph networks. (2018). arXiv:1806.01261
35. M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering. in *Advances in Neural Information Processing Systems* 29, eds. by D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon, R. Garnett (Curran Associates, Inc., 2016), pp. 3844–3852. http://papers.nips.cc/paper/6081-convolutional-neural-networks-ongraphs-with-fast-localized-spectral-filtering.pdf
36. P. Velickovic et al., Graph attention networks. in *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ
37. C. Selvi, E. Sivasankar, A novel adaptive genetic neural network (AGNN) model for recommender systems using modified k-means clustering approach. Multimed. Tools Appl. **78**(11), 14303–14330 (2019). https://doi.org/10.1007/s11042-018-6790-y
38. I. Henrion et al., Neural message passing for jet physics. Deep Learning for Physical Sciences workshop at NIPS 2017 (2017). https://cims.nyu.edu/~bruna/Media/nmp_jet.pdf
39. M. Abdughani et al., Probing stop with graph neural network at the LHC. (2018). arXiv:1807.09088 [hep-ph]
40. J. Arjona Martinez, O. Cerri, M. Pierini, M. Spiropulu, J.-R. Vlimant, Pileup mitigation at the large hadron collider with graph neural networks. Eur. Phys. J. Plus **134**, 333 (2018). https://doi.org/10.1140/epjp/i2019-12710-3. arXiv:1807.07988 [hep-ph]
41. J. Gilmer et al., Neural message passing for quantum chemistry in Proceedings of the 34th International Conference on Machine Learning - Volume 70. 1263–1272 (2017)
42. Y. Wang et al., Dynamic graph cnn for learning on point clouds. (2018). arXiv:1801.07829 [cs.CV]
43. M. Abadi et al., TensorFlow: large-scale machine learning on heterogeneous systems, 2015. Software available from http://www.tensorflow.org
44. S. Agostinelli et al., GEANT4: a simulation toolkit. Nucl. Instrum. Methods A **506**, 250–303 (2003)
45. S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift. in *Proceedings*

of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, pp. 448–456 (2015)
46. D.P. Kingma, J. Ba, Adam: a method for stochastic optimization. in *3rd International Conference on Learning Representations*, ICLR 2015, San Diego, CA, USA, 7–9 May 2015 (2015). arXiv:1412.6980
47. L.N. Smith, N. Topin, Super-convergence: very fast training of residual networks using large learning rates. (2017). arXiv:1708.07120