A Dynamic Reduction Network for Point Clouds

Lindsey Gray, Thomas Klijnsma
Fermi National Accelerator Laboratory
Batavia, IL 60510
lagray@fnal.gov

Shamik Ghosh
Saha Institute of Nuclear Physics
Kolkata, West Bengal, India
shamik.ghosh@cern.ch

March 18, 2020

Abstract

Classifying whole images is a classic problem in machine learning, and graph neural networks are a powerful methodology to learn highly irregular geometries. It is often the case that certain parts of a point cloud are more important than others when determining overall classification. On graph structures this started by pooling information at the end of convolutional filters, and has evolved to a variety of staged pooling techniques on static graphs. In this paper, a dynamic graph formulation of pooling is introduced that removes the need for predetermined graph structure. It achieves this by dynamically learning the most important relationships between data via an intermediate clustering. The network architecture yields interesting results considering representation size and efficiency. It also adapts easily to a large number of tasks from image classification to energy regression in high energy particle physics.[1][2]

Keywords Machine Learning · Geometric Deep Learning · Graph Clustering · Classification · Regression · High Energy Physics

1   Introduction

Pooling has been an essential component of modern machine learning, allowing pertinent local information to be propagated to global intermediate feature sets or final discriminators. The shape of the pooling operation is typically determined by hand, setting the size of a convolutional filter and the number of pooling steps before an output layer. This process is difficult to optimize for graph neural networks [1], since neighbourhoods of nodes may vary in size and meaning depending on the problem at hand. In the area of message passing neural networks [2] there are recent advancements in learned pooling techniques on graphs [3], and there is small but steady progress on using pooling to alter input graph structures. This text describes a new pooling architecture using dynamic graph convolutions [4] and clustering algorithms to learn an optimized representation and corresponding graph for pooling. The model used in the following text is implemented in Pytorch Geometric [5] (PyG).

This architecture was derived in the context of hadron[3] energy regression in High Energy Physics (HEP), where graph neural networks are beginning to solve difficult clustering problems [6, 7] in novel ways. The objective of that problem is to determine the original energy of a particle incident upon a device called a "sampling calorimeter." Within the calorimeter, which is made of dense material like lead or steel interspersed with lighter material, incident particles above a threshold energy will produce pairs of particles by nuclear interaction, creating a "shower" of particles. At a number of fixed depths within the calorimeter, it records scintillation or ionization signals at fixed depths as a proxy for the number of produced particles. These estimates of multiplicity can be used to infer the originating particle's energy. The energy deposition patterns of hadrons are known to have a high degree of local fluctuations in particle multiplicity during the shower's evolution. This means that throughout the shower there are randomly located regions

---

[1]This preprint will be updated with more studies on different open datasets.

[2]The code needed to reproduce the results in this text is included in the arxiv submission file.

[3]Particles that are bound together by the strong force.

that require different treatment from more homogeneous ones, and so there exists an optimal dynamic cluster for each hadron shower's data to best estimate the energy.

A human-designed algorithm called "software compensation" [8] has been developed to solve this problem as well. It is already based in the principle of reducing an objective function to generate learned weights to determine shower energies. However, there is a significant amount of specific tuning that needs to be done to make the algorithm function for varying detector designs, and its domain of applicability remains well within HEP. Using the technique described in this paper, the entire algorithm is now learned, rather than a specific part of a correction, and the algorithm can dynamically adapt to the topology of a given hadron shower. Using a machine learning algorithm for this task mitigates the need for manual specialization, and affords the possibility to investigate the applications of technique on rather different tasks from calorimetry and with more widely available datasets. Benchmarks for various estimation and classification tasks will be demonstrated in the more classic machine learning tasks.

The advantages of this dynamic reduction architecture are:

- Representation spaces with good performance are very small.
- No prior graph structure is necessary, and if one is provided it can be altered by the pooling layers, since the graph pooling structure is learned.
- Without a prior graph structure, data for training and inference need very little preprocessing beyond normalization and stacking.

2 Related Work

The architecture proposed here is similar in outcome to the techniques proposed in [3, 9], but radically different in implementation. Our focus is on learning latent representations that optimize the pooling performance of an unsupervised clustering algorithm. In particular, previous works in graph learning [10, 11] demonstrate that controlling the behavior of an unsupervised algorithm can help in learning concise representations quickly. However, aspects of the original structure of the data were kept and only messages to pass in that structure were generated. This text expands on both previous aspects by combining it with dynamic graph convolutions [4] and controlling a clustering algorithm in the latent space to produce a dynamically learned optimized pooling.
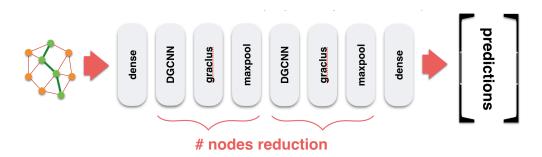
3 Dynamic Reduction Network

Figure 1: The basic workflow of the Dynamic Reduction Network, details given below.

All clustering algorithms can be treated as an indexing of nodes, so any clustering algorithm can be used in the latent space to make this demonstration. The unsupervised clustering algorithm is treated as a black box, and the supervised task is to optimize its performance for the task at hand. Using a dynamic graph convolutional approach, it is possible to learn the parameters that maximize the impact of the clustering on reducing information for further processing, and hence this network is called a dynamic reduction network (DRN). We have chosen to use a readily available greedy popularity based clustering algorithm [12] as an initial demonstrator. Other clustering algorithms will be tested and compared once their GPU implementations are made available in PyG in the future.

The default model used for the MNIST superpixels [10] classification task in this paper is composed as follows:

1. The input data are normalized by fixed values such that the input data largely occupy the range $[0, 1]$, outliers are allowed.

2. A multilayer perceptron (MLP) encoding the normalized input data to a latent space of dimension N is applied to all input nodes. The default depth is 3 layers, with an intermediate layer half the size of the final output.

3. The latent space data are processed by a dynamic graph convolution layer, i.e. neighbours are found in the latent space rather than the original representation. The internal messages are created using a MLP with three layers, starting from 2N, 1.5N, and outputing a message of width N. The update function can be summation, maximum, or average.

4. The resulting latent nearest neighbours graph is then weighted by distance and clustered using a greedy clustering algorithm, pooling the node features by taking the maximum of the clustered data.

5. The reduced latent data are passed through another dynamic convolutional filter, and clustered again with the same algorithm.

6. The results of the second learned pooling step are then globally max pooled and passed through an MLP decoder to produce the output logits.

This process is summarized in Figure 1. Alterations to this model used for various test are described later in the text. The depth of the MLPs in the various encoding, message passing, and decoding steps are parameterized. The number of nearest neighbours, k, is a hyper-parameter of the model and may require tuning to a given task, depending on the relational data that is needed to make a prediction.

4 Results

This model was tested on MNIST "superpixels" dataset with 75 superpixels [4]. The superpixels dataset is a downsampled and aggregated form of the full MNIST dataset. Previous graph models are shown in Refs. [10, 11], A result of a scan in hidden-dimension (the "width") of the network is shown in 2. During the training and evaluation both the original graph structure and all pixels that are not filled are dropped, the data are "zero-suppressed". Each superpixel has a pair of coordinates defining a centroid, and an intensity. The performance using a width of 20 channels with $k = 4$ is a factor of two better on 75 superpixels than the reference models in [10, 11], and approaches the performance of models trained on full MNIST at a width of 256 channels. The data passed to the model consists only of the centroid x and y, and the gray-scale of that pixel. In Figure 2 each DRN is trained for 400 epochs using an nVidia Tesla V100 with the AdamW optimizer [13] using one-cycle cosine annealing (on the learning rate only) with a starting value of 0.001. The weight decay is held constant at a value of 0.001. Best performance is often achieved quite early, so further tuning of model training and optimization is possible, and volatile GPU usage is low. There is significant room for improving the training and evaluation time performance of this model.

Model variant	No. of parameters	Best achieved performance	Epoch of best performance	Performance at 400 epochs
DRN20	5123	*0.9761*	180	0.9731
DRN32	12797	0.9806	205	0.9792
DRN64	50157	0.9872	347	0.9866
DRN128	198605	0.9891	232	0.9884
DRN256	790413	**0.9905**	217	0.9899
MoNet	–	0.9111	–	–
SplineCNN	63786	0.9522	40	–

Figure 2: Performance for a scan in dynamic reduction network width of the architecture described above on the MNIST "superpixels" dataset [10] with 75 superpixels, and two established models. DRN<N> means a dynamic reduction network with hidden dimension N is used. Comparison to similar models on the superpixels dataset are made. All superpixel graph data has been dropped and the network is allowed to learn an entirely new representation based on the pixel data. For all DRN models, $k = 4$. The performance at a hidden dimension size of 20 is particularly interesting given the modest gains in performance from wider networks. '–' indicates unknown data.

The superpixels dataset is known to yield poor performance in CNN-based networks, and graph networks have been proposed in previous work to improve upon this. The performance demonstrated in Figure 2 shows clearly that further improvements on very under-sampled data are achievable and this model sets new performance benchmarks on these datasets, especially in terms of model size.

---

[4]This model was developed using private data of the CMS Collaboration which cannot be published here.

5 Conclusions

This text introduces the Dynamic Reduction Network as a new tool in High Energy Particle physics for processing sampled data from a highly varying multi-dimensional image. This is accomplished by designing a network that can learn effective pooling strategies by manipulating an unsupervised algorithm in a high-dimensional latent space. To demonstrate the efficacy of this network, a performance benchmark on an undersampled MNIST dataset indicates that this new architecture outperforms previous graph based architectures, even for very small numbers of parameters. This outcome suggests a powerful new technique for approaching classification and regression problems in both computer vision and high energy physics.

6 Acknowledgements

References

[1] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, Jul 2017.

[2] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[3] Frederik Diehl. Edge contraction pooling for graph neural networks, 2019.

[4] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds, 2018.

[5] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[6] Shah Rukh Qasim, Jan Kieseler, Yutaro Iiyama, and Maurizio Pierini. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *The European Physical Journal C*, 79(7):608, 2019.

[7] Xiangyang Ju, Lindsey Gray, et al. Graph neural networks for particle reconstruction in high energy physics detectors. *NeurIPS Proceedings*, 2019.

[8] Huong Lan Tran, Katja Krüger, Felix Sefkow, Steven Green, John Marshall, Mark Thomson, and Frank Simon. Software compensation in particle flow reconstruction. *The European Physical Journal C*, 77(10), Oct 2017.

[9] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling, 2019.

[10] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns, 2016.

[11] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels, 2017.

[12] Bas O. Fagginger Auer and Rob H. Bisseling. *A GPU Algorithm for Greedy Graph Matching*, page 108–119. Springer-Verlag, Berlin, Heidelberg, 2012.

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.