Heterogeneous Point Set Transformers for Segmentation of Multiple View Particle Detectors

Edgar E. Robles
Department of Computer Science
University of California, Irvine
Irvine, CA 92697

Dikshant Sagar
Department of Computer Science
University of California, Irvine
Irvine, CA 92697

Alejandro Yankelevich
Department of Physics
University of California, Irvine
Irvine, CA 92697

Jianming Bian
Department of Physics
University of California, Irvine
Irvine, CA 92697

Pierre Baldi
Department of Computer Science
University of California, Irvine
Irvine, CA 92697

For the NOvA Collaboration

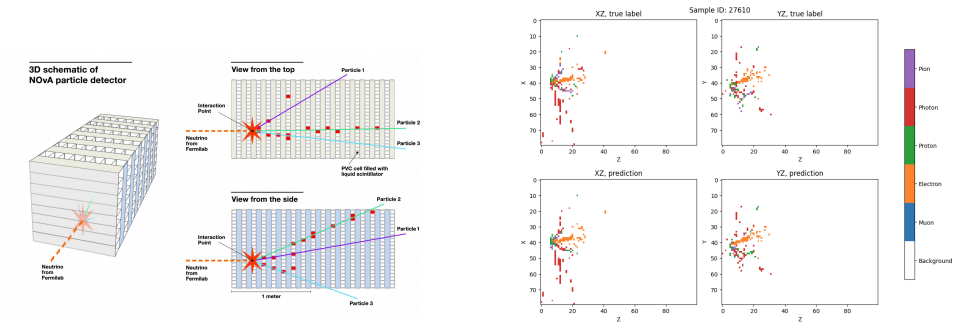
Abstract

NOvA is a long-baseline neutrino oscillation experiment that detects neutrino particles from the NuMI beam at Fermilab. Before data from this experiment can be used in analyses, raw hits in the detector must be matched to their source particles, and the type of each particle must be identified. This task has commonly been done using a mix of traditional clustering approaches and convolutional neural networks (CNNs). Due to the construction of the detector, the data is presented as two sparse 2D images: an XZ and a YZ view of the detector, rather than a 3D representation. We propose a point set neural network that operates on the sparse matrices with an operation that mixes information from both views. Our model uses less than 10% of the memory required using previous methods while achieving a 96.8% AUC score, a higher score than obtained when both views are processed independently (85.4%).

1 Introduction

Experiments in the physical sciences often create large amounts of data, which is hard to process at scale by human experts. The advent of high quality machine learning models has improved performance across many of these data processing tasks [5], but with an increase in quality, there has also been an increase in computational costs. Even large experimental collaborations in the field of particle physics often face strict limits in resource utilization.

NOvA [3] is a long-baseline neutrino oscillation experiment in which neutrinos from Fermilab's NuMI beam interact in both the near detector (ND) at in Batavia, IL and larger far detector (FD) in Ash River, MN spaced $810\,\mathrm{km}$ apart. The detectors consist of long $4\,\mathrm{cm} \times 6\,\mathrm{cm}$ polyvinyl chloride (PVC) plastic extrusions, referred to as "cells", filled with liquid scintillator. These cells are arranged in planes such that the cells' orientations alternate horizontally and vertically between neighboring planes (Figure 1a). The full FD considered here is $16\,\mathrm{m} \times 16\,\mathrm{m} \times 60\,\mathrm{m}$ with 344064 cells. When neutrinos interact within the detector, the resulting particles, e.g. pions, photons, protons, electrons and muons, produce scintillation light as they travel through the detector and appear as



(a) Schematic of NOvA detector and generation of top and side views from vertical and horizontal planes respectively.

(b) Example event display image of a neutrino interaction producing a prominent electron shower and secondary particles shown with true particle classes (top) and HPST predicted classes (bottom).

Figure 1: Visual representations of the NOvA detector and data.

tracks or showers referred to as "prongs" (Figure 1b) [1]. The task at hand is then to perform instance segmentation over these prongs as well as to classify each detection into its corresponding particle type.

The generated images are often very sparse, consisting of an empty background in most of the image except for a few prongs. When performing computations such as the ones used in segmentation machine learning models, sparse matrices have to be converted into dense matrices, which can slow down training and inference. There have been implementations of differentiable convolution operations on sparse matrices, such as Nvidia's MinkowskiEngine [7]. However, the operations need to approximate a convolution in order to save memory. An alternative to using sparse matrices is representing the sparse image as a point cloud, which allows the coordinates and values to be processed.

The design of the NOvA detectors only allows images to be produced over two planes: the XZ plane (top view) and YZ plane (side view), in which each view only consists of vertical or horizontal cells, respectively (Figure 1a). Therefore, the Y position of a hit in a vertical plane fundamentally cannot be known. The existing method for the instance segmentation task approaches this by applying the fuzzy k-means clustering algorithm on each view independently followed by matching instances across the two views [4]. Similarly for the particle classification task, existing methods use two independent CNNs to embed each image [2, 10, 6] or use each view as a channel to an image [11]. We can extend the framework of point transformers by using heterogeneous attention [9]. In this paper we leverage point set transformers, along with implementing a novel method to allow information to flow between both views of the dataset, allowing for a model that is able to draw information from both views. The model proposed is known as a heterogeneous point set transformer (HPST).

2 Methods

2.1 Notation

Consider a dataset $\mathcal{X}$ of size N, where each sample $X^{(i)}$ represents an event from the particle detector. Each event $X^{(i)}$ is split into M views, each view denoted by $X^{(i,j)}$ In the case of the NOvA detector, we have $M = 2$ views. Each view has a variable number of detections $K^{(i,j)}$. Each detection is described by coordinates $x_k^{(i,j)} \in \mathbb{R}^c$ and values $v_k^{(i,j)} \in \mathbb{R}^d$.

For each pair of points we define an intra-view distance $d_{jj}(x_k^{(i,j)}, x_{k'}^{(i,j)})$ for points within the same view and therefore vector space and an inter-view distance $d_{jj'}(x_k^{(i,j)}, x_{k'}^{(i,j')})$ for points between different views. Additionally, based on these distances we will define an edge $e_{k,k'}^{(i)} \in \{0, 1\}$ which connects two nodes that may be in the same or different views.

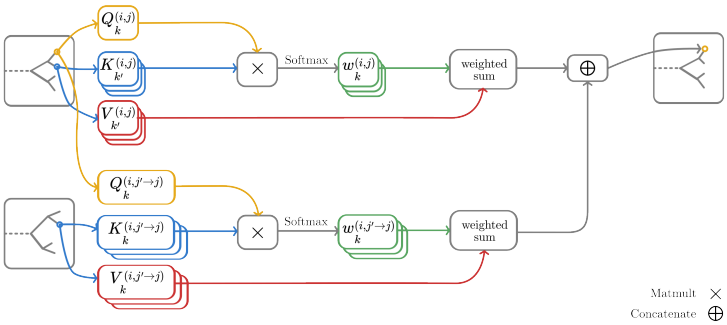


Figure 2: Block diagram of the attention mechanism. The top path describes the intra-view attention mechanism, and the bottom path describes the inter-view mechanism.

2.2 Heterogeneous attention

In order to calculate attention, we need to encode each point into a set of keys, queries and values, each of these is done by taking the input point $x_k^{(i,j)}$ and multiplying it by a linear layer as such: $Q_k^{(i,j'\to j)} = W_Q^{(j'\to j)} x_k^{(i,j)}$, $K_k^{(i,j'\to j)} = W_K^{(j'\to j)} x_k^{(i,j)}$, $V_k^{(i,j'\to j)} = W_V^{(j'\to j)} x_k^{(i,j)}$.

We use the inter-view distance to build a nearest neighbors graph, then, for each point we calculate the query $Q_k^{(i,j'\to j)}$, i.e., the query on point k from view j' to view j on sample i, and then for each of its neighbors k' we calculate both $K_{k'}^{(i,j'\to j)}$ and $V_{k'}^{(i,j'\to j)}$, that is, the key and values on point k' from view j' to view j on sample i. Using these, we can calculate $w_{kk'}^{(i,j'\to j)} = Q_k^{(i,j'\to j)T} K_{k'}^{(i,j'\to j)}$. RPE (relative positional encoding, $\mathrm{RPE}(x,y) = W(x-y)$, where W is a learnable linear layer as used in [14]) is not used between different views. This weight is then normalized using a softmax operation over its neighbors and then used in a weighted sum to calculate the output of the attention module,

$$h'^{(i,j)}_k = \sum_{k'} \mathrm{softmax}_\ell(w_{k\ell}^{(i,j'\to j)})_{k'} V_{k'}^{(i,j'\to j)}.$$

Pooling is done within the same view, using a voxel pooling method [12], creating a grid and then averaging out the values of all the points within each point of the grid, and positioning the point in the barycenter of all the points within the created voxel. Unpooling is performed using skip connections, the points are upsampled to the same coordinates that they were previously pooled from, only using information from the same view.

2.3 Architecture

The network is structured like a UNet. The UNet is divided into $2n$ stages. Each stage contains m blocks, where each block has an attention module. Following each stage in the first n stages is a pooling step, which reduces the number of points. The next n stages are followed by an unpooling stage, which uses the point coordinates from a past block, as well as concatenates the features from the past block. The dimensionality of the embeddings is doubled at each stage during the first half and halved at each stage during the second half. Intra-view attention is calculated on each stage in order to ensure that the information mixing between views is done locally (in the earlier stages) as well as globally (in the later stages).

2.4 Loss function

The network performs two tasks simultaneously: an instance segmentation, selecting separate prongs from each other; and a semantic segmentation, classifying each detection into a particle type. As such, the loss function used is separated into two parts: $\mathcal{L} = \lambda \mathcal{L}_{\text{sem}} + (1-\lambda)\mathcal{L}_{\text{ins}}$. Semantic segmentation is a simple classification problem, so we use multi-class cross-entropy to calculate

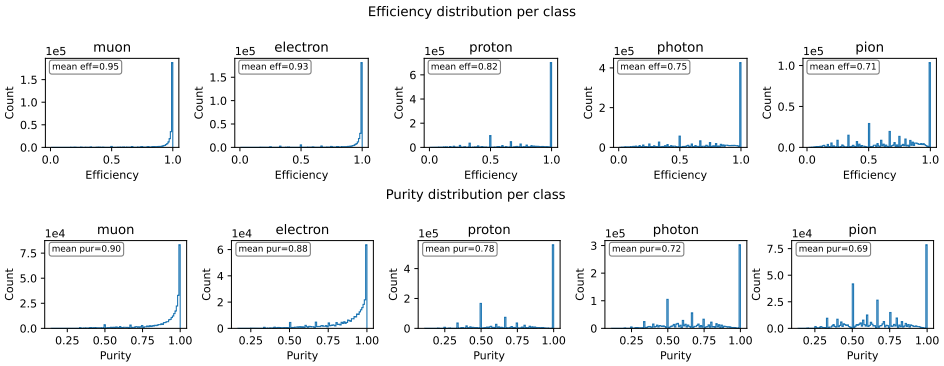


Figure 3: Distribution of prong efficiency and purity

this loss: $\mathcal{L}_{\text{sem}} = \sum_{X^{(i)} \in \mathcal{X}} \sum_{X^{(i)} \in x_k^{(i,j)}} \text{CE}\left(\text{softmax}_{k'}\left(f(X_i)_{k'}^{(i,j)}\right), y_k^{(i,j)}\right)$, where $y_k^{(i,j)}$ is the correct semantic label of the detection.

Instance segmentation is done by minimizing the loss calculated by the best assignment between the predicted labels and the real labels. If point $x_k^{(i,j)}$ belongs to the segment $L_k^{(i,j)}$, then the loss calculated is $\mathcal{L}_{\text{ins}} = \sum_{X^{(i)} \in \mathcal{X}} \min_{\phi \in \Sigma} \sum_{x_k^{(i,j)} \in X^{(i)}} \text{CE}\left(\text{softmax}_{k'}\left(f(X_i)_{k'}^{(i,j)}\right), \phi\left(L_k^{(i,j)}\right)\right)$, where Σ is the set of all permutations of labels, allowing a unique assignment of one label to another. The optimal assignment of the labels is solved using a linear sum assignment solver.

3 Experiments and Results

3.1 Experiments

The dataset used consists of neutrino interaction simulations generated by the NOvA collaboration for the purpose of preparing event reconstruction methods for future large-scale productions. This dataset contains 9246712 events, each event containing 70 hits on average in $2 \times 80 \times 100$ images.

Three models were trained and evaluated: a Mask-RCNN [8] based model, GAT [13], and our heterogeneous point set transformer. We performed a hyperparameter search: using the number of neighbor connections (4, 8), the number of stages of the neural network (2, 3, 4), the size of the embeddings inside the neural network (128, 256, 512), and the learning rate (between 1e-4 and 1e-1). The training and testing was done on a server using an Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz, 503G of RAM, and 4xNVIDIA Titan V. The hyperparameter sweep was peformed over 8 epochs, using 1% of the dataset. The model with the best accuracy on the segmentation's class labels was selected as the one with the best hyperparameters. The resulting models with the best hyperparameters were trained for 24 epochs. The code for these experiments can be found at `https://github.com/erobl/hpst`.

3.1.1 Classification and Segmentation accuracy

We present the evaluation of each model in Table 1. For each prong, we calculated the efficiency (recall) and purity (precision) of the classification, allowing for multiple predicted prongs to be assigned to a single prong. Figure 3 shows the distribution of purity and efficiency for each class. As we can see, the segmentation results are generally good, especially in the majority classes (muons and electrons). Figure 1b shows a qualitative example from the test set. This sample presents a mostly correct classification, with some confusion for secondary particles producing very few hits.

3.1.2 Speed and memory usage

We benchmarked the three models by running inference on 100 samples, with a batch size of 1, measuring the peak memory increase between the start of inference and the end of inference, to remove as much overhead as possible. We evaluated the time it takes for these 100 inferences and

Table 1: Speed and memory usage for each model compared to their performance.

Model	Memory usage (MiB)	Time per sample (s)	OVR AUC	Segmentation accuracy
R-CNN	151.9 ± 11.950	3.1449 ± 0.0660	0.732	0.343
GAT	29.10 ± 0.001	0.3911 ± 0.0010	0.854	0.659
HPST (ours)	29.35 ± 0.559	0.4636 ± 0.0032	0.968	0.835

the memory used in each of them. Memory usage is greatly reduced compared to the regular CNN model. Our model is able to obtain a superior performance over the other models while still keeping a comparable memory usage.

4 Limitations and Conclusions

While the claims of memory efficiency will generally hold true, for different datasets this might not be the case. The representation of a sparse matrix is more efficient than a dense matrix until a certain point, where the storage of the coordinates becomes bigger than just storing a dense matrix. Point set operations can also greatly increase in complexity as the number of points grows, resulting in a much slower algorithm. However, these are not the regimes found in the the NOvA experiment. HPSTs strike a balance between memory usage, time, and performance that is fitting for its application, they are able to train a stronger model with the same amount of parameters due to sharing information between the views, which in turn boosts the performance of the model.

Acknowledgments and Disclosure of Funding

This document was prepared by the NOvA collaboration using the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, Office of High Energy Physics HEP User Facility. Fermilab is managed by FermiForward Discovery Group, LLC, acting under Contract No. 89243024CSC000002.

References

[1] AURISANO, A., BACKHOUSE, C., HATCHER, R., MAYER, N., MUSSER, J., PATTERSON, R., SCHROETER, R., AND SOUSA, A. The nova simulation chain. *Journal of Physics: Conference Series 664*, 7 (dec 2015), 072002.

[2] AURISANO, A., RADOVIC, A., ROCCO, D., HIMMEL, A., MESSIER, M. D., NINER, E., PAWLOSKI, G., PSIHAS, F., SOUSA, A., AND VAHLE, P. A Convolutional Neural Network Neutrino Event Classifier. *JINST 11*, 09 (2016), P09001.

[3] AYRES, D. S., ET AL. The NOvA Technical Design Report.

[4] BAIRD, M., BIAN, J., MESSIER, M., NINER, E., ROCCO, D., AND SACHDEV, K. Event reconstruction techniques in nova. *J. Phys.: Conf. Ser. 664*, 7 (dec 2015), 072035.

[5] BALDI, P. *Deep Learning in Science*. Cambridge University Press, 2021.

[6] BALDI, P., BIAN, J., HERTEL, L., AND LI, L. Improved energy reconstruction in nova with regression convolutional neural networks. *Phys. Rev. D 99* (Jan 2019), 012011.

[7] CHOY, C., GWAK, J., AND SAVARESE, S. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 3075–3084.

[8] HE, K., GKIOXARI, G., DOLLÁR, P., AND GIRSHICK, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2961–2969.

[9] HU, Z., DONG, Y., WANG, K., AND SUN, Y. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020* (New York, NY, USA, 2020), WWW '20, Association for Computing Machinery, p. 2704–2710.

[10] PSIHAS, F., NINER, E., GROH, M., MURPHY, R., AURISANO, A., HIMMEL, A., LANG, K., MESSIER, M. D., RADOVIC, A., AND SOUSA, A. Context-Enriched Identification of Particles with a Convolutional Network for Neutrino Events. *Phys. Rev. D 100*, 7 (2019), 073005.

[11] SHMAKOV, A., YANKELEVICH, A. J., BIAN, J., AND BALDI, P. Interpretable joint event-particle reconstruction at nova with sparse cnns and transformers. In *Machine Learning and the Physical Sciences, NeurIPS* (2023).

[12] SIMONOVSKY, M., AND KOMODAKIS, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 29–38.

[13] VELICKOVIC, P., CUCURULL, G., CASANOVA, A., ROMERO, A., LIO, P., BENGIO, Y., ET AL. Graph attention networks. *stat 1050*, 20 (2017), 10–48550.

[14] WU, X., LAO, Y., JIANG, L., LIU, X., AND ZHAO, H. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS* (2022).