Score-based Graph Generative Model for Neutrino Events Classification and Reconstruction

Yiming Sun[*1] **Zixing Song**[*2] **Irwin King**[2]
[1]Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China
[2]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Sha Tin, N.T., Hong Kong SAR
sunyiming2000@mail.ustc.edu.cn, {zxsong, king}@cse.cuhk.edu.hk

Abstract

The IceCube Neutrino Observatory is an astroparticle physics experiment to investigate neutrinos from the universe. Graph Neural Networks (GNNs) have achieved great success in this area due to their powerful modeling ability for the irregular grid structure of the detectors. Unlike existing GNN-based methods which apply GNNs directly on a simple constructed graph like kNN graph, we focus on the graph construction step via the score-based generative model to enhance the quality of the constructed graph for GNNs to operate on for downstream tasks. Extensive experiments on the classification and reconstruction of neutrino events verify the efficacy of our method.

1 Introduction

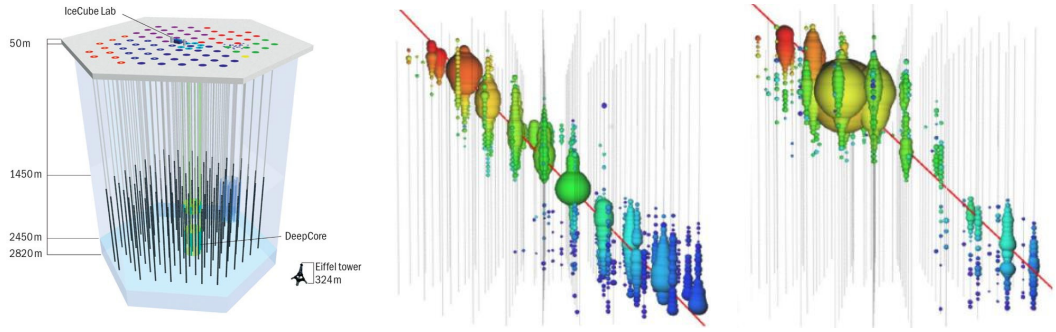
The IceCube Neutrino Observatory is a cubic-kilometer particle detector to observe neutrinos from the universe. The detector consists of 5160 digital optical modules (DOMs). DOMs are attached to 86 vertical strings and the strings are deployed on a hexagonal grid [5] which is shown in Figure 1(a). We only sketch a portion of them as a top-view of the observatory such that 60 DOMs are in each string and overlapping in depth within Figure 1(a). The sensors detect and record the Cherenkov radiation emitted by secondary charged particles produced in the primary neutrino interactions. We can use the information to analyze the direction, energy and type of muons and neutrinos.

Event classification aims to observe a flux of muons from neutrino interactions or in down-going events. The signal event refers to the muons produced in the neutrino-ice interaction, while the background event includes muons from cosmic-ray interactions with the atmosphere. Our first downstream task is to distinguish the signal from the background and the main difference between them is light emission from the track. As shown in Figure 1(c), signal has an uneven light track, while the average emission of a bundle of muons in the background is more smooth, as shown in Figure 1(b) [2, 4].

After removing the background events, we can further reconstruct the events of our interests. Strictly speaking, we need to predict the neutrino energy, the zenith angle of arrival direction, and the event topology of each interaction. These parameters can be analyzed to estimate properties, including traveled distance and neutrino flavor [6, 10, 3].

[*]Equal contribution. Work done when the first author did an internship at CUHK.

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

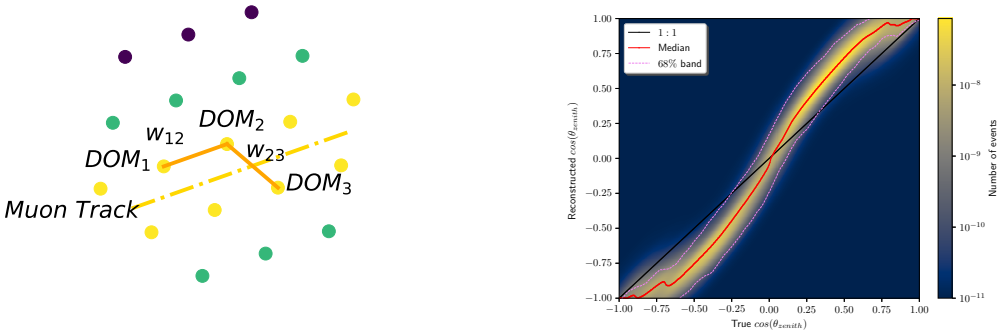


(a) Icecube detector schematic (Credit: IceCube Collaboration)

(b) Light deposition for muon bundles from background (Credit:[1])

(c) Light deposition for single muon from signal (Credit:[1])

Figure 1: Background knowledge of Icecube experiment



(a) DOM strings sketch

(b) Zenith angle reconstruction based on [1]

Figure 2: Motivations of our work and limitations of previous works

2 Motivations

In an event, the light can activate a small number of DOMs, which output the information about the energy and time of the interaction. Because of the sparseness of active sensors for low-energy events and the irregular sensor geometry, previous methods based on convolutional neural network (CNN) have confronted difficulties [6], while GNN is more suitable in this situation. Given the x, y, z coordinates of each DOM activated by the event, they can be formed as a geometric graph first. GNN can then be applied to generate a representation vector of this graph for predicting the event information like its type and energy as a classification task.

However, existing GNN-based methods [6, 1] mostly focus on the second graph classification component and omit the importance of the first graph construction step. One calculates the pairwise Euclidean distance and uses the result to connect k nearest neighbors [6]. Another applies a Gaussian kernel to the pairwise distances $d_{ij} = e^{-\frac{1}{2}\|x_i - x_j\|^2/\sigma^2}$ [1]. These vanilla construction methods leave out many factors. In [1], the learnable parameter σ is the only parameter in this network, so it is sensitive to noise. Moreover, it considers active DOMs in each event as different graphs, resulting in unstableness and difficulty while training. Besides, they fail to take factors other than pairwise distance into consideration. In Figure 2(a), DOMs are arranged in hexagonal distribution so the distances between DOM_1 DOM_2 and DOM_2 DOM_3 are the same, leading to the same edge weights using [1]. So since the weights reflect the influences between nodes, DOM_3 and DOM_1 have the same impact on DOM_2 in this model. However, DOM_2 and DOM_3 are supposed to share more information because the muon track passes between them. Furthermore, because of the symmetry properties of Gaussian kernel, the graph constructed in the previous model [7] remains

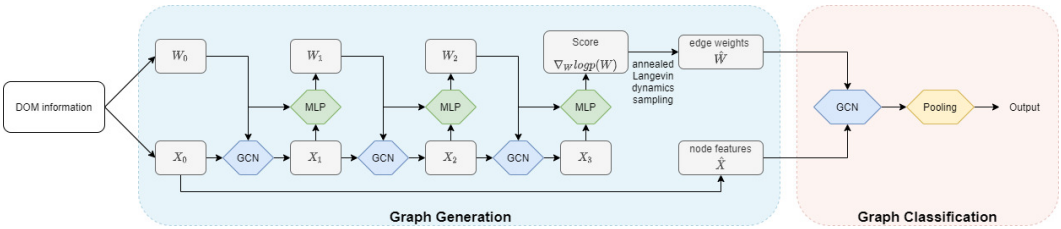


Figure 3: Model structure

roughly the same even if zenith angle varies, so it does not achieve satisfying performance in zenith angle reconstruction. Our initial empirical experiments demonstrate this issue in Figure 2(b) as most of the predictions of incident angle deviate from the ground truth.

We hold the assumption that graph structure is vital to this task. Motivated by [7, 9], we focus on graph generation to open up improvement space for the following graph classification and reconstruction task. We generate the edge automatically, learning from information obtained by DOMs. We will illustrate it thoroughly in Section 3. We also include all the DOMs in this graph, so the number of nodes does not change, stabilizing the training process.

3 Model Architecture

This model has two main components: graph generation and graph classification. We will train the two steps together. If the network outputs a binary label between signal and background as a graph classification problem, we use the cross-entropy loss. If it is a graph regression problem to predict values like zenith angle in the event reconstruction problem, we apply the mean squared error (MSE) loss.

3.1 Graph generation step

Node features are initialized with the given information of each DOM, including the x, y, z position of its corresponding DOM, the sum of charge in the first pulse within the sensor, the sum of charge in all pulses within the sensor, and the time at which the first pulse crosses the activation threshold. As for initialization, the node features of active DOMs are their corresponding information and inactive ones are padded with zeros.

Score-based models [9] are introduced to avoid limitations in previous generative modeling techniques. The advantage is that they do not require normalization and can be learned directly in contrast to common likelihood-based generative models. Inspired by EDP-GNN [7], we apply score-based models to generate edge weights $\hat{W}$ of the graph formed by all the DOMs based on the gathered information from the detectors. The generated weighted graph lays the foundation for the downstream graph classification or regression task.

Naive score-based generative model. We hold the assumption that the final constructed graph with edge weights $\hat{W}$ follows a latent true distribution $p(W)$. We will train a parameterized GNN model $\mathbf{s}_\theta$ to approximate the scores of $p(W)$, which is defined as the gradient of the log-density function $\nabla_W \log p(W)$. By Langevin dynamics procedure, we can obtain samples from data distribution $p(W)$ using only its score function $\nabla_W \log p(W)$ so $\mathbf{s}_\theta$ is trained by minimizing the Fisher divergence

$$\mathbb{E}_{p(W)} \|\nabla_W \log p(W) - \mathbf{s}_\theta(W)\|_2^2.$$

Noise conditional score-based generative model. However, the estimated score functions in the naive score-based generative model are inaccurate in low-density regions. So we can use a series of Gaussian noise $\{\sigma_i\}_{i=1}^M$ to perturb W to get $\tilde{W}$ and populate these regions. We now train a noise conditional score-based model $\mathbf{s}_\theta(W, \sigma_i)$ to capture $\nabla_W \log q_{\sigma_i}(W)$. And we use the weighted average loss of every noise scale to update the aforementioned Fisher divergence as follows,

$$Loss(\theta) = \frac{1}{2M} \sum_{i=1}^{M} \sigma_i^2 \mathbb{E}_{q_{\sigma_i}(W)} \| \frac{W - \tilde{W}}{\sigma_i^2} - \mathbf{s}_\theta(W, \sigma_i) \|_2^2. \tag{1}$$

To remove the expectation symbol in Eq. (1) for implementation, we sample edge weights matrix $\tilde{W}$ from $p(\tilde{W}) = q_{\sigma_i}(W)$ with annealed Langevin dynamics from [9]. We also propose a GNN-based model to parameterize $\mathbf{s}_\theta(W, \sigma_i)$ as shown in Figure 3. As for the input, we connect all the active DOMs, which means the entity in the adjacency matrix W_0 equals one when the end-points of this edge are both active. We use Graph Convolutional Network (GCN) to update node features X^{k+1} using X^k and edge features W^k by

$$X^{k+1} = \mathrm{ReLU}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X^k W^k).$$

We use multilayer perceptron (MLP) to update edge features W_k using W^{k-1} and updated node features X^k as

$$W_{ij}^k = \begin{cases} \mathrm{MLP}(\mathrm{CONCAT}(W_{ij}^{k-1}, X_i^k, X_j^k)) & i < j \\ W_{ji}^k & \text{otherwise} \end{cases}. \tag{2}$$

Because W is symmetry, we only update the upper triangular matrix and copy it to the lower triangle. The output is a $N \times N$ score matrix, approximating the true score of $p(\tilde{W})$. We use this score matrix to sample the final edge weights $\hat{W}$ via Langevin dynamics (Appendix A.1) with the noise level being set to zero.

3.2 Graph classification step

We still apply another GCN on the constructed graph from the last step with the initial node features X_0 and the generated edge weights $\hat{W}$. The network outputs graph-level representation by summing the N nodes' features and then uses logistic regression to predict the final class.

4 Experiments

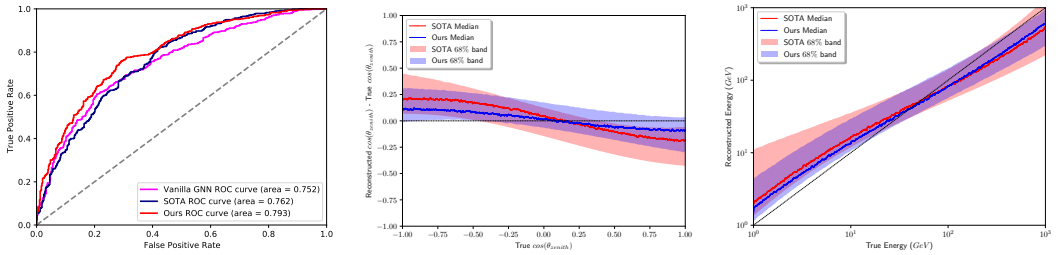
4.1 Datasets and baseline models

Event classification. The primary goal of the event classification task in the IceCube Experiment [1] is to separate the signal (muons from neutrinos, positive samples) from the background (muons from cosmic-ray shower, negative samples) based on the event information observed and collected on each DOM. Therefore, it is a binary classification problem at the graph level. We use the existing simulated Monte Carlo dataset [1] and the background event is many orders of magnitude larger than the signal event. We choose the train/validation/test split as 18,937/7,381/7,890 for the signal event and 82,118/29,813/7,890 for the background event. Note that we make the signal-to-background-ratio or signal-to-noise-ratio (SNR) consistent with the true distribution in the training and validation dataset while maintaining a 1:1 SNR on the test dataset to avoid an extremely low false-positive rate. We compare our method against two GNN-based models. One is a vanilla GNN model on the constructed Gaussian kernel graph [1] and the other is the state-of-the-art model called TAGCN on the constructed kNN graph [6]. The hyperparameters in both baselines are set as suggested in the original papers.

Event reconstruction. We can further reconstruct the selected track event by predicting its key parameters of interest, the zenith angle θ of the arrival direction, and the neutrino energy E after finishing the previous event classification task to remove all the background events. So it is a graph regression problem to predict a real value. We still use the same Mote Carlo simulation data [6] with the mixture of muon (ν_μ) and tau (ν_τ) interaction events. We neglect the event topology here because we only focus on track-like events, and all other events such as cascade-like events are filtered during the event classification task. We only employ the state-of-the-art TAGCN model [6] as the baseline here since the other Gaussian kernel graph performs much worse on this task, which is discussed earlier in the introduction section.

4.2 Results and analysis

Event classification. Figure 4(a) demonstrates the receiver operating characteristic (ROC) curve for the event classification task for our method and the other two GNN-based models. By fixing a 1:1



(a) Receiver operating characteristic curve with area under curve (AUC) for event classification.

(b) 1D slice distribution of the difference between reconstructed and true zenith angle vs. true zenith angle

(c) 1D slice distribution of the reconstructed energy value vs. true energy value

Figure 4: Experimental results on the event classification (a) and event reconstruction (b)(c) tasks

SNR on the test dataset, clearer differences among these three models can now be displayed in terms of the ROC curve. We can see that the ratio between true positive rate and false-positive rate for all the thresholds in the final predictive confidence is improved with our proposed method, leading to a significantly larger area under the ROC curve (AUC).

Event reconstruction. For both the reconstruction task with respect to the zenith angle and energy, we show the satisfactory results in Figure 4(b) and Figure 4(c) respectively. The true energy ranges from 1 GeV to 1,000 GeV, and we use the logarithm value of the original energy in base 10 for training and testing to circumvent the potential bias towards the events with higher energy. It is worth noting that the 68% bands of the reconstructed events by our method are generally more constraining and centering around the benchmark lines than those generated by the baseline SOTA model. The benchmark lines are formed by samples with zero reconstruction error as the horizontal black dashed line and the diagonal black dashed line in Figure 4(b) and Figure 4(c) respectively. Moreover, we believe the distorted shapes of the bands for the energy reconstruction in Figure 4(c) arises from the prediction bias caused by the unbalanced distribution when the energy value is too low (10 GeV) and the energy transfer phenomenon [8] when the energy value is too high (1,000 GeV).

5 Conclusion

Because the sensors in IceCube experiments are arranged irregularly and active ones are sparse for low-energy events, we employ GNN in this domain. To classify and reconstruct events, we focus on the graph generation component to improve the following graph classification and reconstruction task. We employ a score-based generative model to construct the graph automatically, learning from the dataset. Furthermore, the outstanding experiment results verify our assumption that graph generation is vital for the performance of the following GNN. Our method achieves a better performance but involves more parameters. We can use the model compression technique to improve training and predicting speed in the future.

Acknowledgments and Disclosure of Funding

We would like to thank the anonymous reviewers for their comments. The work described in this paper was partially supported by the National Key Research and Development Program of China (No. 2018AAA0100204) and CUHK 2410021, Research Impact Fund (RIF), R5034-18.

References

[1] Nicholas Choma, Federico Monti, Lisa Gerhardt, Tomasz Palczewski, Zahra Ronaghi, Prabhat, Wahid Bhimji, Michael M. Bronstein, Spencer R. Klein, and Joan Bruna. Graph neural networks for icecube signal classification. In *ICMLA*, pages 386–391. IEEE, 2018.

[2] Erik Ganster, Richard Naab, and Zelong Zhang. A Combined Fit of the Diffuse Neutrino Spectrum using IceCube Muon Tracks and Cascades. *arXiv e-prints*, page arXiv:2107.10003, July 2021.

[3] Mirco Hünnefeld. Combining Maximum-Likelihood with Deep Learning for Event Reconstruction in IceCube. *arXiv e-prints*, page arXiv:2107.12110, July 2021.

[4] Yang Lyu. Characterization of the PeV astrophysical neutrino energy spectrum with IceCube using down-going tracks. *arXiv e-prints*, page arXiv:2107.14298, July 2021.

[5] Jessie Micallef. Reconstructing Neutrino Energy using CNNs for GeV Scale IceCube Events. *arXiv e-prints*, page arXiv:2107.11446, July 2021.

[6] Martin Ha Minh. Reconstruction of Neutrino Events in IceCube using Graph Neural Networks. *arXiv e-prints*, page arXiv:2107.12187, July 2021.

[7] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 4474–4484. PMLR, 2020.

[8] Xavier Rodrigues, Anatoli Fedynitch, Shan Gao, Denise Boncioli, and Walter Winter. Neutrinos and ultra-high-energy cosmic-ray nuclei from blazars. *The Astrophysical Journal*, 854(1):54, 2018.

[9] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. *arXiv e-prints*, page arXiv:1907.05600, July 2019.

[10] Shiqi Yu. Direction Reconstruction using a CNN for GeV-Scale Neutrinos in IceCube. *arXiv e-prints*, page arXiv:2107.02122, July 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1
   (b) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Did you describe the limitations of your work? [Yes] See Section 5
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 4 and Appendix

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Section 4

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Annealed Langevin dynamics

We train a series of noise conditional score networks $\mathbf{s}_\theta(W, \sigma_i)$, and use annealed Langevin dynamics to generate samples. We first initialize $\tilde{W}_0$ with folded normal distribution. Then we use a series of noise conditional score-based models $\mathbf{s}_\theta(W, \sigma_i)$ where the noise scale σ_i is in descending order, and run Langevin dynamics for T steps in each model. Notice that every $\tilde{W}_t$ is supposed to be symmetric, so we only update the upper triangular matrix and copy to the lower triangle.

Algorithm 1 Annealed Langevin Dynamics Sampling.

Input: The series of guassian noise scales, $\{\sigma_i\}_{i=1}^M$; The smallest step size, ϵ; The number of iteration for each noise level, T;

Output: The edge weights sample, $\tilde{W}$ subject to $p(\tilde{W})$;

Initialize $\tilde{W}^0$ using folded normal distributions;

$(\tilde{W}^0)_{ij} = \begin{cases} |\epsilon_{ij}| & \text{if} \quad i < j \\ (\tilde{W}^0)_{ji} & \text{otherwise} \end{cases}$, where $\epsilon_{ij} \sim N(0, 1)$;

for $i = 1$ to M **do**

  $\alpha_i = \epsilon \cdot \sigma_i^2 / \sigma_M^2$;

  for $t = 1$ to T **do**

    Draw $A^t \sim N(0, 1)$;

    $(\tilde{W}^t)_{ij} = \begin{cases} (\tilde{W}^{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\hat{W}^{t-1}, \sigma_i) + \sqrt{\alpha_i} A)_{ij}, & \text{if } i < j \\ (\tilde{W}^t)_{ji}, & \text{otherwise} \end{cases}$;

  end for

  $\tilde{W}^0 = \tilde{W}^T$;

end for

return $\tilde{W}^T$;