PolarBERT: A Foundation Model for IceCube

Inar Timiryasov
University of Copenhagen
Niels Bohr Institute
Jagtvej 155A, DK-2200
Copenhagen, Denmark
inar.timiryasov@nbi.ku.dk

Jean-Loup Tastet
University of Copenhagen
Department of Computer Science
Universitetsparken 1, DK-2100
Copenhagen, Denmark
jeta@di.ku.dk

Oleg Ruchayskiy
University of Copenhagen
Niels Bohr Institute
Jagtvej 155A, DK-2200
Copenhagen, Denmark
oleg.ruchayskiy@nbi.ku.dk

Abstract

The IceCube Neutrino Observatory at the South Pole is a cubic kilometer of Antarctic ice, instrumented with 5,160 digital optical modules. These modules collect light induced by neutrino interactions in the ice. This data is then used to identify the neutrino directions, their energies, and types, which are essential inputs for both particle physics and astrophysics. Deep learning methods, such as graph neural networks, have been successfully applied to the steady stream of incoming data IceCube is receiving. In this work, we build a foundation model on the IceCube data in a self-supervised way without any data labeling. This pre-trained model can be fine-tuned for the downstream task of directional reconstruction of neutrino events in a sample-efficient way.

1 Introduction

The IceCube Neutrino Observatory [1] is a cubic-kilometer-sized neutrino detector located at the South Pole. It consists of 5,160 digital optical modules (DOMs) distributed over 86 vertical strings deep inside the Antarctic ice. The DOMs are sensitive to Cherenkov light produced by charged particles moving through the ice faster than the speed of light in ice. The main goal of the IceCube experiment is to identify the sources of cosmic rays and to study the properties of neutrinos.

Traditionally, IceCube data analysis is based on sophisticated maximum likelihood estimators [2]. These methods are used to reconstruct the direction of the incoming neutrino and to estimate the energy of the neutrino and its type. The likelihood-based methods are computationally expensive and require significant human effort to tune the parameters of the reconstruction algorithm.

Recently, deep learning methods have been successfully applied to IceCube data analysis, see, e.g., [3, 4, 5]. These models are trained using synthetic Monte Carlo data. Simulating light propagation in a cubic kilometer of ice is a challenging and expensive process. IceCube accumulates vast amounts of real data, with a detection rate of 2.15 kHz [6] yielding approximately 70 billion events annually. While atmospheric muons constitute the majority of these events, they share fundamental physical properties with signal events. This abundance of real data naturally motivates the exploration of self-supervised learning methods for IceCube observations.

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

In this work, we propose a foundation model for IceCube, which we call PolarBERT. The model is similar to BERT [7] but instead of masked token prediction, we perform "masked DOM prediction". We pre-train PolarBERT on the publicly available high-quality Monte Carlo data prepared by the IceCube collaboration for the Kaggle competition [8]. Using the directional labels available in the dataset, we then fine-tune the model for the downstream task of directional reconstruction of neutrino events.

Remarkably, the foundation model can be trained without requiring explicit knowledge of the IceCube detector geometry or individual optical module sensitivities. This suggests that when trained on real experimental data, the model could autonomously learn crucial detector characteristics, including ice transparency and DOM quantum efficiency, directly from the observations. However, it is important to note that while the model may internally encode these properties, extracting such information from the trained model remains challenging and is not guaranteed to be feasible.

2 Related work

Recently, several foundation models have been proposed in particle physics, particularly for jet reconstruction at the Large Hadron Collider [9, 10, 11, 12]. For an up-to-date review of the field, see [13].

The main challenge in self-supervised pre-training for particle physics lies in data representation. To apply self-supervised learning, raw data must be converted into a sequence of tokens, allowing the model to predict a probability distribution over these tokens. For example, the authors of [12] and [11] use VQ-VAE to convert jet constituent momenta into tokens from a fixed-size dictionary. Alternative approaches often embed raw particle features directly into the model space, such as in jet-transformer [14], which is trained in a supervised manner.

For IceCube, the 5,160 DOMs provide a natural tokenization scheme. However, each pulse registered by a DOM contains multiple features, with time of the pulse and charge being the most significant. Given the nature of IceCube data, we adopt a hybrid approach: we treat DOMs as tokens and linearly embed other features into a high-dimensional space. The unique timestamp of each pulse also provides a natural ordering for the pulse sequence.

3 IceCube Data

Publicly available data. The IceCube collaboration has released a high-quality Monte Carlo dataset for the Kaggle competition [8]. The dataset consists of 131,000,000 simulated neutrino events. These events were caused by neutrinos (or background muons) of all flavors with energies ranging from 100 GeV to 100 PeV. The events were simulated with an energy spectrum of $E^{-1.5}$ for energies up to 1 PeV and E^{-1} for higher energies. The neutrino directions (azimuth and zenith) are provided as labels. Other ground truth parameters, such as energy or neutrino type, are not specified. The number of pulses per event—a proxy for the neutrino energy—ranges from 2 to 100,000. The data is available in `parquet` format on the Kaggle platform. For every pulse, the following data is provided: the *time* of the pulse in nanoseconds within the current event time window; *DOM ID*; *charge*, an estimate of the amount of light in the pulse, in units of photoelectrons (p.e.); and a Boolean variable *auxiliary*. If auxiliary is True, the pulse was not fully digitized, is of lower quality, and was more likely to originate from noise. In the future, we plan to pre-train the model on real data and use synthetic data only at the fine-tuning stage. However, a study using real data can only be conducted as part of the IceCube collaboration. Therefore, in this study, we use synthetic data for both stages as a proof of concept.

IceCube Data as Time Series. Since all pulses within an event have unique timestamps, we can treat each event as a time series. The time series data is a sequence of pulses, where each pulse is represented by a vector of features. These features are the time, DOM ID, charge, and auxiliary flag (see Figure 1).

Down Sampling. The main challenge of the IceCube data is that the number of pulses per event varies significantly. This problem is not specific to our time series representation. In this work, we fix the maximum sequence length `seq_len` and down-sample the events with more pulses. We also pad events with fewer pulses with zeros. There are several ways to down-sample events. We

prioritize fully digitized pulses (*auxiliary* = `False`), which we refer to as *primary pulses*. If an event contains more than `seq_len` primary pulses, we randomly select `seq_len` of them. If an event contains fewer than `seq_len` primary pulses, we add randomly selected auxiliary pulses (*auxiliary* = `True`). A similar strategy has been used by the top Kaggle solutions [15]. We found that down-sampling reduces the reconstruction accuracy for the most energetic events. Due to the power-law distribution of neutrino energies, there are relatively few high-energy events, so this is not an issue when considering average accuracy, as was the case in the Kaggle competition. For physics purposes, fast reconstruction of the most energetic events is crucial and will be addressed in future work.

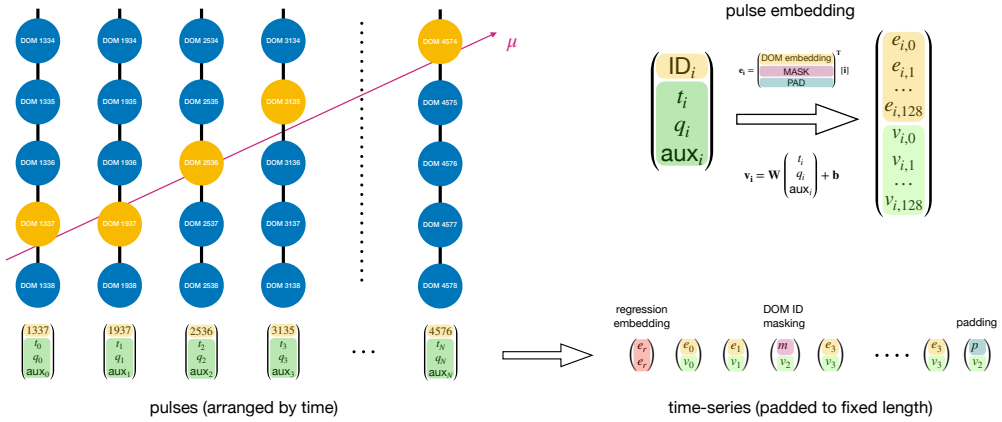

Figure 1: A sketch representing how pulses are embedded into the model space. Each DOM ID corresponds to a trainable embedding vector. There are also embedding vectors for masking and padding. The time and charge are linearly embedded into the model space.

4 Model design

Pulse embeddings. The pulses are encoded as follows (see Figure 1). DOM IDs are embedded using a standard dimensionality embedding layer $d_{\text{model}}/2$. The vocabulary size of the embedding vector is $N_{\text{DOMs}} + 3$, where $N_{\text{DOMs}} = 5160$ is the number of DOMs, and there are three additional special "tokens": the padding token, the mask token, and the regression token. The time, charge, and auxiliary flag are linearly projected into a $d_{\text{model}}/2$ dimensional space. Before applying the affine transformation, we rescale the features as follows: time $\to \frac{\text{time}-10^4\text{ns}}{3\times10^4\text{ ns}}$, charge $\to \frac{1}{3}\log_{10}(\text{charge})$, aux $\to$ aux $- 0.5$. The resulting two vectors—DOM embedding and other features—are then concatenated into a single vector of dimensionality d_{model}. We prepend a special trainable regression embedding, used for the downstream task, to each sequence. In this work, we downsample and pad all events to `seq_len`$= 128$.

The rationale behind this embedding approach stems from the unique challenges of handling mixed data types. Masking time and charge values would necessitate their digitization, similar to approaches used in jet physics [12, 11], or require more sophisticated objectives as recently proposed in [16]. Therefore, to leverage the natural "tokenization" offered by IceCube DOMs, we divided the features into two distinct categories. Our experimental investigations included frequency-based untrainable time encodings—analogous to those in the original transformer—though these proved suboptimal for model training. Future work will explore incorporating DOM spatial positions into the masked embedding components.

Transformer model. The model architecture is inspired by BERT [7]. The encoder-only architecture has demonstrated empirical success with IceCube data [15]. The model consists of an embedding layer and an encoder-only, pre-layer norm variant of the transformer [17]. The model dimension is d_{model}. The transformer encoder consists of N_L layers. We have experimented with both the standard MLP and SwiGLU [18] layers but did not observe significant differences in performance. For the rest of this work, we use the standard MLP with ReLU non-linearity. We do not use positional encodings since the timestamps already provide information about the relative position of pulses in the sequence.

Masking strategy. During pre-training, we mask a fraction of pulses by replacing the DOM embedding with a trainable masking embedding. We found that a masking fraction of 0.25 leads to the best downstream performance. Note that we only mask DOM embeddings and not other features like time. We have found it crucial to mask only primary pulses. The auxiliary pulses are mostly due to random noise, and the model cannot predict in which DOM an auxiliary pulse takes place. Nevertheless, we discovered that complete removal of auxiliary pulses degraded model performance, suggesting they contribute meaningful contextual information despite their stochastic nature.

Pre-training. To enhance the model's ability to capture important event information in the regression embedding, we introduced an auxiliary prediction task. In addition to the masked DOM prediction, we task the model with predicting the logarithm of the total charge of all pulses. This auxiliary feature is available in the unlabeled data and cannot be linearly extracted from the input, as we provide the model with logarithms of individual pulse charges. The resulting loss is a combination of the usual cross-entropy loss for the masked DOMs and an MSE loss for the regression task:

$$l = L_{CE} + \lambda L_{MSE}, \tag{1}$$

where L_{MSE} is the mean squared error loss between true and predicted logarithm of total charge, and the default value of $\lambda = 1$. The second term quickly reaches nearly zero values during training, but removing this term significantly decreases the fine-tuning capacity.

Fine-tuning. Once the model is pre-trained, we remove the unembedding layer and add an MLP prediction head (with a single hidden layer of size $4 \times d_{\text{model}}$). This prediction head converts the regression embedding into a three-dimensional unit vector. This unit vector is used to predict the direction with the mean angular error loss [8].

Currently, neutrino direction represents the sole ground-truth label available in the kaggle dataset. We plan to use open-source data from the Prometheus project [19] when a large enough dataset becomes available.

5 Experiments

PolarBERT is implemented[1] in PyTorch and utilizes `FlashAttention` [20] for numerical efficiency. A dataset with 100,000,000 neutrino events was saved as a memory-mapped file. Two separate datasets with 1,000,000 events each are used for validation and fine-tuning. For pre-training, we used the Adam optimizer [21] with a cosine schedule and tuned the learning rate (with the largest batch size fitting the device). During fine-tuning, we observed that the final results depend strongly on the initialization of the prediction head. Therefore, we opted for the recently introduced schedule-free version of Adam [22]. The schedule-free optimizer allowed for better and faster convergence of fine-tuning.

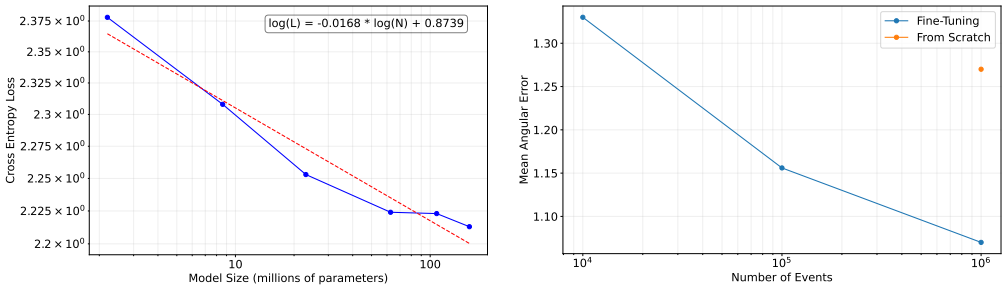

Figure 2: *Left:* The evaluation loss as a function of model size demonstrates a clear scaling law. All models were trained on 10 million neutrino events. *Right:* Downstream performance: mean angular error (lower is better) as a function of the fine-tuning dataset size.

We pre-trained a family of PolarBERT models ranging from 2.2 to 159 million parameters on a dataset containing 10 million neutrino events for one epoch. The resulting losses follow a clear power

[1] `https://github.com/timinar/PolarBERT`

law similar to that observed for large language models [23], with results presented in the left panel of Figure 2. Our largest configuration (159M parameters) required 4 hours of training on a single 40GB A100 GPU with learning rate 3×10^{-4}, cosine schedule, and batch size 512.

For comprehensive pre-training, we focused on a more efficient 8M-parameter model (8 layers, $d_{\text{model}} = 256$, MLP hidden size 1024) due to resource constraints. This model was trained on 100 million events for one epoch using batch size 2048 and maximum learning rate 5×10^{-4} with cosine annealing schedule. After 4 hours of training on a single 40GB A100 GPU, it achieved an evaluation cross-entropy loss of 2.08, demonstrating superior computational efficiency compared to the 159M model by reaching a lower final loss within the same training budget. Based on this performance, we selected this configuration for subsequent fine-tuning experiments.

We further fine-tuned the model on 10,000, 100,000, and 1,000,000 unique events. We used the pre-trained 8M backbone and added a prediction head. We did not freeze the backbone parameters since this would reduce the quality of the final results. We also trained several models from scratch on the same datasets. In both cases we limited the number of unique events used for fine-tuning, but did not limit the number of epochs and trained until the validation loss stopped decreasing.

The original PolarBERT architecture does not generalize if trained only on such small datasets, so we had to modify it by adding positional encodings and DOM coordinates to the embeddings. With these modifications, the model can be trained on 1 million events, but still significantly underperforms compared to the fine-tuning of the pre-trained model, see the right panel of Figure 2.

It is worth noting that our model, when fine-tuned on less than one percent of the dataset, does not yet reach the accuracy of the best models trained in a supervised manner [15]. However, our model could in principle be pre-trained using real data, eliminating the simulation gap and leveraging the much higher number of recorded events.

6 Conclusions

In this work, we have demonstrated the feasibility and effectiveness of using a foundation model approach for IceCube neutrino event reconstruction. Our proposed model, PolarBERT, leverages self-supervised learning on large amounts of unlabeled data to create a powerful pre-trained model that can be fine-tuned for specific tasks with limited labeled data.

Our key findings are: *(i)* Our hybrid embedding approach and masking strategy are effective in capturing relevant information from the raw data. *(ii)* We observe a clear scaling law in pre-training performance, similar to that seen in large language models. *(iii)* There are significant improvements in sample efficiency and performance when fine-tuning the pre-trained model compared to training from scratch, especially on smaller labeled datasets.

Further architecture refinements, as well as using real data for pre-training and scaling up the models are natural directions for future research.

7 Broader Impact Statement

We expect that this work will stimulate further research into foundation models in physics. Our mixed masking strategy and pre-training with an additional regression target could be readily applied to other particle physics settings.

The observation of a power-law relationship between model performance and size suggests that with more computing power, the performance of models in particle physics will continue to improve. This finding has implications for the future of data analysis in experimental physics.

Acknowledgements

We thank Erik Dam, Troels C. Petersen, and Raghavendra Selvan for valuable discussions, as well as the anonymous referees for their constructive feedback. This work was supported by a research grant (VIL57416) from VILLUM FONDEN. The work of IT was partially supported by the Carlsberg foundation and by the European Union's Horizon 2020 research and innovation program under the

Marie Sklodowska-Curie grant agreement No. 847523 'INTERACTIONS'. Computational resources for this work were partially provided by the SCIENCE AI Centre of Copenhagen University.

References

[1] R. Abbasi et al. The IceCube Data Acquisition System: Signal Capture, Digitization, and Timestamping. *Nucl. Instrum. Meth. A*, 601:294–316, 2009.

[2] M. G. Aartsen et al. Energy Reconstruction Methods in the IceCube Neutrino Telescope. *JINST*, 9:P03009, 2014.

[3] Jessie Micallef. Using convolutional neural networks to reconstruct energy of GeV scale IceCube neutrinos. *JINST*, 16(09):C09019, 2021.

[4] R. Abbasi et al. Graph Neural Networks for low-energy event classification & reconstruction in IceCube. *JINST*, 17(11):P11003, 2022.

[5] Andreas Søgaard et al. GraphNeT: Graph neural networks for neutrino telescope event reconstruction. *J. Open Source Softw.*, 8(85):4971, 2023.

[6] Serap Tilav, Thomas K. Gaisser, Dennis Soldin, and Paolo Desiati. Seasonal variation of atmospheric muons in IceCube. *PoS*, ICRC2019:894, 2020.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Philipp Eller. Public Kaggle Competition "IceCube – Neutrinos in Deep Ice". In *38th International Cosmic Ray Conference*, 7 2023.

[9] Thorben Finke, Michael Krämer, Alexander Mück, and Jan Tönshoff. Learning the language of QCD jets with transformers. *JHEP*, 06:184, 2023.

[10] Matthias Vigl, Nicole Hartman, and Lukas Heinrich. Finetuning Foundation Models for Joint Analysis Optimization. 1 2024.

[11] Joschka Birk, Anna Hallin, and Gregor Kasieczka. OmniJet-α: The first cross-task foundation model for particle physics. 3 2024.

[12] Lukas Heinrich, Tobias Golling, Michael Kagan, Samuel Klein, Matthew Leigh, Margarita Osadchy, and John Andrew Raine. Masked Particle Modeling on Sets: Towards Self-Supervised High Energy Physics Foundation Models. 1 2024.

[13] Matthew Feickert and Benjamin Nachman. A Living Review of Machine Learning for Particle Physics. 2 2021.

[14] Huilin Qu, Congqiao Li, and Sitian Qian. Particle Transformer for Jet Tagging. 2 2022.

[15] Habib Bukhari, Dipam Chakraborty, Philipp Eller, Takuya Ito, Maxim V. Shugaev, and Rasmus Ørsøe. IceCube – Neutrinos in Deep Ice The Top 3 Solutions from the Public Kaggle Competition. 10 2023.

[16] Matthew Leigh, Samuel Klein, François Charton, Tobias Golling, Lukas Heinrich, Michael Kagan, Inês Ochoa, and Margarita Osadchy. Is Tokenization Needed for Masked Particle Modelling? 9 2024.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[18] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[19] Jeffrey Lazar, Stephan Meighen-Berger, Christian Haack, David Kim, Santiago Giner, and Carlos A. Argüelles. Prometheus: An open-source neutrino telescope simulation. *Comput. Phys. Commun.*, 304:109298, 2024.

[20] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[21] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Aaron Defazio, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, Ashok Cutkosky, et al. The road less scheduled. *arXiv preprint arXiv:2405.15682*, 2024.

[23] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.