oceanoGRAPH: Graph Neural Networks in Ocean Temperature Prediction

Jacob Lessard & Soumya Menon

December 3rd, 2024

Abstract

The oceanoGRAPH project aims to investigate the feasibility of using graph neural networks (GNN) in ocean temperature prediction, particularly with the Argo program dataset. The Argo program conducts oceanographic research through fleets of floats deployed across the globe and has led to immense data collection in the field. However, the process involved is cost intensive and is exposed to errors and miscalibrations due to hardware and physical issues in the ocean. The goal of this project is to utilise the available float data in combination with graph neural networks as a better form of data imputation and as a cheaper means of artificially expanding the float fleet. The paper investigates multiple GNN architectures through fine tuning data preprocessing and hyperparameter tuning. The optimal architecture involved using Huber loss, along with constructing our input graph with a radius threshold of 500km, producing an average training and validation loss of ≈ 0.25.

1 Introduction

1.1 Argo Program

The Argo program is an international program that collects data from across the world's oceans using a collection of more than 3500 free floating instruments known as floats. Each of these floats remain at sea for 3 to 5 years at depths of up to 6 kilometers, measuring key variables for the ocean such as temperature, pressure, salinity (see figure 1). In addition, models such as the BGC-Argo floats also measure more specialized quantities, such as oxygen content, pH, chlorophyll A content, and more bio-geochemical variables. Altogether, the Argo program collects more than 13,000 data profiles per month, and with data available from 2016 to November 2024, there are nearly 3 million data profiles available.

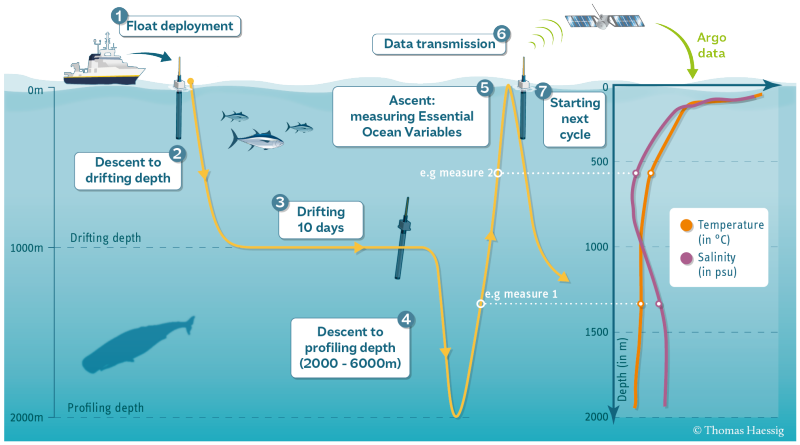

Figure 1: Diagram depicting how Argo floats collect data [2]

1.2 Graph Neural Networks

Graph neural networks are a form of neural networks that are an architecturally similar extension to the standard neural network architecture, with the usage of graph-based data as input. The goal of these networks is to leverage the information present in the graph, like the 'topological dependency of information in each node' (Scarselli et al. 2009), in learning underlying trends in the dataset. Unlike models that are used on unstructured data, GNN's take advantage of the structure in the data to ideally perform better. By capturing the dependence in graphs through their edges, GNN's have produced many promising results especially in deep learning tasks [1]. A key feature about these networks is the variety of machine learning tasks they can do, including ones that are unique to the graphical data structure, such as edge prediction and graph generation.

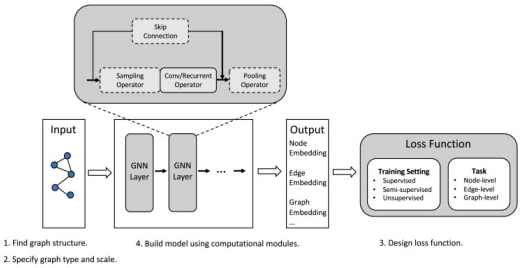

Figure 2: Standard GNN Architecture (Zhou et al.)

Their applications go far and wide, and many major companies have their own proprietary implementations of GNNs that they use. Some examples include AliGraph by the AliBaba Group [5] and PinSage by Pinterest [6].

2 Dataset and Preprocessing

2.1 Argo Dataset

The Argo dataset was retrieved using ArgoPy and the ERDDAP server provided by the National Oceanic and Atmospheric Administration (NOAA) [12]. Due to limits on the size of requests for the data as well as data availability, we decomposed our area of interest into 4 regions encompassing all 360 degrees of longitude and the latitudes between 50°N and 15°S. Moreover, we selected the time frame of 2013 November 1st to the 11th, we chose this date due to data quantity. Furthermore, we chose a date range since the Argo floats all send data to their respective satellite once every 10 days, and so we are making the assumption that regions of the ocean do not vary in temperature greatly over a 10-day period, we acknowledge that this is a simplifying assumption, however without it the quantity of data on a given single day is not large enough to effectively train a sophisticated model.

2.2 Graph Dataset Construction

The main preprocessing step that this project involves is converting the Argo float dataset into a graph. The raw dataset (Fig. 3) is an xarray of device ID's, their position (latitude, longitude), along with their corresponding measurements. The parameters we focused on are temperature, pressure, and salinity. As the data suggests, we want to construct a graph with our nodes being each float.

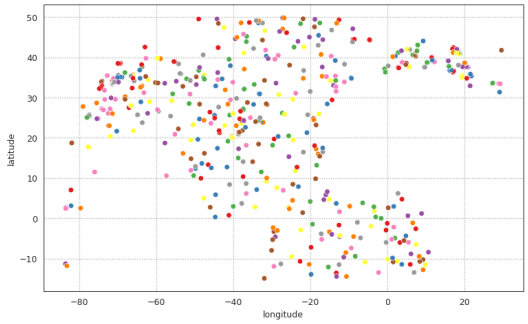

Figure 3: Argo float data (Region 1) plotted with position coordinates

Many methods can be used as criteria to create edges between these nodes, with one of the straightforward ones being implementing a radius threshold. This is a criterion that creates edges from a node to all other nodes in a fixed radius around it. The distance between nodes was calculated using the haversine distance, which measures the distance between two points on a sphere (longitude and latitude) along the great circle passing through them.

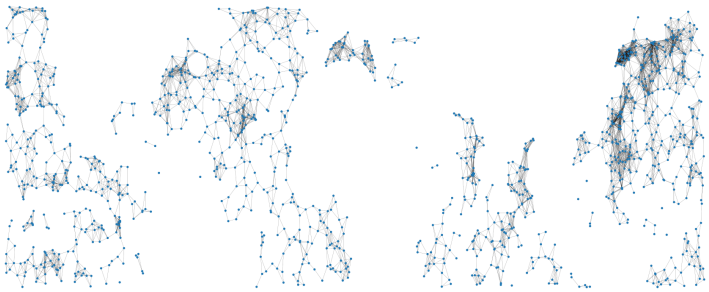

Figure 4: Graph generated with Argo float data and a radius threshold of 500km

The value of the radius threshold can also be tuned as a hyperparameter, as the density of edges in the graph will influence the performance of the model. We found a radius of 500km to work best (Fig. 4) when compared to other options of 100km, 750km, 1250km, and 1500km.

3 GNN Architectures

We decided to use two main architectures for this project: a default GNN architecture, and a GNN architecture with dropout enabled.
The first architecture we decided to use is based on Graph Convolutional Networks, first introduced by Kipf and Welling. This is a scalable implementation of CNN's that operates directly on graphical data. We also chose this for its ease-of-use due to its inbuilt functionalities available in PyTorch Geometric.

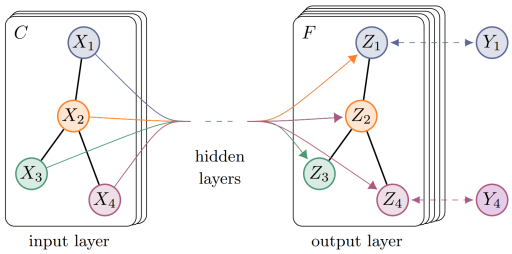

Figure 5: Graph Convulational Network (Kipf & Welling)

We construct our default architecture to have 2 layers, with 64 neurons in the hidden layer.

An alternative to the default architecture is also investigated, with the inclusion of dropout to prevent overfitting by randomly deactivating some neurons during training. Dropout is a well-researched regularization technique in CNNs to improve

robustness and decrease overfitting in the model. This architecture was constructed with 3 layers, dropout after the first layer, and 4 neurons in each hidden layer.

3.1 Loss Functions

To determine the best loss function for our model we consulted *Loss Functions and Metrics in Deep Learning. A Review* by Juan et al [9]. Since our task was to perform regression on a point-value, we considered the Mean-Squared Error (MSE), Mean-Absolute Error (MAE), the Huber loss, and the Log-Cosh loss as mentioned in the paper. We experimented with each loss function, and we concluded that the Huber loss was the most suitable for our dataset and desired task. We expected our Argo float dataset to contain outliers primarily due to equipment miscalibration and malfunction, as well as oceanographic phenomena, such as jet streams, which can strongly influence the temperature of the earth beneath them [10]. Since we anticipated these outliers, we decided that a robust loss function was necessary and thus we decided to not use the MSE as it is sensitive to outliers. Moreover, due to our desired task of temperature prediction, as well as the fact that the Argo float temperature data is precise up to 3 decimal places [11], we wanted a low penalty on small errors, thus as a result the MAE and the Log-Cosh loss were not chosen due to their sensitivity to minor errors. Hence, the Huber loss was chosen as it was the appropriate mix of insensitivity to outliers, while still minimizing the penalty to minor errors. Subsequently, we then experimented with various values for the hyperparameter δ, we determined that for our dataset $\delta = 0.5$ was optimal.

3.2 Optimizers

When training our model, we focused on using the Adam and Stochastic Gradient Descent (SGD) optimizers found in PyTorch. To compare the performance of the two optimizers, we utilized the same train, validation, and test sets as well as the same model architecture. We first performed hyperparameter tuning on the Adam optimizer by determining the optimal learning rate. Through empirical experimentation, we found that the optimal learning rate for the Adam optimizer was 0.001. We then performed hyperparameter tuning on the SGD optimizer, optimizing over both the momentum and learning rate parameters. Through empirical experimentation, we found that the optimal SGD optimizer had a momentum of 0.3 and a learning rate of 0.01. After experimentation, we found that the tuned Adam optimizer outperformed the tuned SGD optimizer.

4 Results

We found that the default GNN, the GNN with more layers, and the GNN with more layers and dropout all performed very similarly when trained for 750 epochs - even when changing the number of neurons in each layer. Overall, the best performing

model was the GNN with three hidden layers, dropout at a rate of 50%, the Adam optimizer with a learning rate of 0.3, and the graph radius threshold of 500km. When trained on 750 epochs, the best performing model had a final test Huber loss (with $\delta = 0.5$) of 0.21. Notably, even though this was the best performing model, it's performance was only negligibly better than many of the other architectures when trained on 750 epochs, it was observably the best model in the rate at which it decreased to its final loss, outperforming the others.

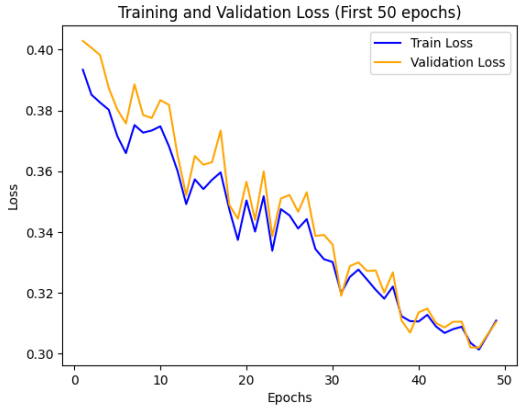

Figure 6: Huber loss with $\delta = 0.5$ for first 50 epochs of the GNN with 3 layers, dropout, and the Adam optimizer with a learning rate of 0.3.

5 Conclusions & Future Work

5.1 Conclusions

The project proposed a method for predicting ocean temperatures using a GNN trained on Argo float data. Moreover, the results of this model indicate that this can be done with a moderate degree of effectiveness. Since the model performed well in training and testing, we conclude that the proposed model can reasonably be utilized to improve Argo float data imputation and it can provide a cheaper alternative to augmenting the pre-existing Argo float fleet.

5.2 Future Work

This project has numerous areas for future work, with extensions in GNNs, the Argo dataset, and other machine learning comparisons.
The Argo program has immense scope for machine learning research. Many extensions in data preprocessing are viable for this project. One specific extension is implementing edge information through weighted edges, specifically with the weight representing the distance between nodes. This can lead to increased accuracy

and precision of regression values. Furthermore, another possible extension of this model is to apply it to other features to predict variables such as oxygen content and pH. Additionally, the inclusion of more oceanographic information, such as whether or not a given node is in a jet stream and other contextual information, could greatly improve the performance by making the structure of our data itself richer. Other extensions in architecture include experimenting with GNN specific dropout [7]. This project included a naive implementation of dropout, however dropout in GNNs are very complicated to implement and are an involved process. More experimentation into this can be beneficial for this project.

This project only scratched the surface of the iceberg of GNNs. Possible extensions of different types of predictive tasks include node classification based on region, link prediction based on parameters, even graph generation to predict where the optimal location of float deployment should be. Outside of GCNs, other types of GNN architecture like Graph Attention Networks (GATs) [8] can also be looked into.

References

[1] Zhou, Jie, et al. "Graph neural networks: A review of methods and applications." AI open 1 (2020): 57-81.

[2] "Argo." About, 19 Nov. 2024, argo.ucsd.edu/about.

[3] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The Graph Neural Network Model," in IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61-80, Jan. 2009, doi: 10.1109/TNN.2008.2005605.

[4] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).

[5] Zhu, Rong, et al. "Aligraph: A comprehensive graph neural network platform." arXiv preprint arXiv:1902.08730 (2019).

[6] Ying, Rex, et al. "Graph convolutional neural networks for web-scale recommender systems." Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018.

[7] Shu, Juan, et al. "Understanding dropout for graph neural networks." Companion Proceedings of the Web Conference 2022.

[8] Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

[9] Terven, Juan, et al. "Loss functions and metrics in deep learning. A review." arXiv preprint arXiv:2307.02694 (2023)

[10] Woollings, Tim. "What Is the Jet Stream?" NOAA Climate, 27 Jan. 2022, www.climate.gov/news-features/blogs/enso/what-jet-stream.

[11] "How Do Floats Work." Argo, argo.ucsd.edu/how-do-floats-work/. Accessed 4 Dec. 2024.

[12] "Erddap." ERDDAP - Home Page, www.ncei.noaa.gov/erddap/index.html. Accessed 4 Dec. 2024.