Evaluation of Spatial-Temporal Graph Neural Network on Temperature Forecast

Ambrose Lee [1]

Abstract

Weather forecast has been playing a critical role in contemporary society, and improving the accuracy of temperature forecast could bring numerous social and economic benefits. In this work, our objective is to evaluate different machine learning techniques for temperature prediction, and these results are going to serve as the foundation for upcoming more complicated weather forecast task. We experimented with different machine learning models and graph neural networks to predict the mean temperature one day after. We evaluated the accuracy of different models, and how number of layers and batch size affect the performance of the model. Code is available at https://github.com/ambr-0-se/TemperatureForecast.

1. Introduction

Weather forecasts have played an important role in modern society, not only for normal people but also to various industries. Governments rely on long-term temperature forecast to devise climate policy, while the agricultural sector plan and revise their farming strategies according to short-term temperature forecast (Lobell & Field, 2007). A more accurate temperature could bring immense economic and social benefits, incentivizing researchers to investigate the field.

Recently, graph neural networks (GNNs) have gained traction in machine learning community, and have been applied to various tasks, most commonly node classification and link predictions (Zhou et al., 2018). Designed to process graph-structured data, GNN excels in a wide range of application, but some studies found that GNNs are subject to limited representation power, and many model architectures and kernels fail to surpass 1-WL test (Xu et al., 2018).

This paper serves as a foundation work of predicting weather with graph neural networks. Specifically, we examined models and techniques that perform well on the dataset, and these results would serve as the corner stone of the coming more specific and complicated weather forecast task. Through this experiment, we aimed to study that 1) despite the limited representation power of GNN, how well does it perform in weather forecast; 2) compare the performance of GNN with traditional ML models; and 3) how do different data preprocessing methods, GNN architectures, and hyperparameters affect the accuracy of weather prediction.

2. Related Works

Numerical weather prediction (NWP) has been the gold standard of weather forecast. It is a branch of scientific computing involving solving complex mathematical equations, such as partial differential equations, with a lot of variables to simulate weather. However, as the method is computationally intensive and induces larger errors when time goes on (Brotzge et al., 2023), some people complement it with machine learning methods.

Deep learning models such as LSTM leverage temporal data to discover trend and give accurate predictions (Li et al., 2023). GNNs (Lam et al., 2022) produce comprehensive representations that captures spatial relationship through message passing mechanism. Generating realistic forecasts with vision transformer (ViT) is another a promising direction(Kurth et al., 2023).

3. Methodology

3.1. Dataset

The experiment leverages a weather dataset retrieved from European Climate Assessment & Dataset (ECA&D) (Klein Tank & Coauthors, 2002; Huber et al., 2024). It contains daily observations at meteorological stations in diferent parts of Europe. The dataset was constructed of the daily weather observations of 18 stations across 2000-2010. The objective is to train models that predict the mean temperature of the coming day using previous data with spectral and temporal features.

[*]Equal contribution [1]School of Computing and Data Science, The University of Hong Kong, Hong Kong. Correspondence to: Ambrose Lee <ambr0se@connect.hku.hk>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

3.2. Data Preprocessing

To create the graph, we first calculated the distance between 18 nodes (places), and each region is given an incoming edge with its k-nearest neighbour (default k=3). Then, we generated features for each region. As the set of features differs among regions, we choose features that appear above 80% of the region. For region that does not have the specific feature, we will impute the average value of the feature to the missing column. After filtering less common features, we obtain an unified features consisting of ['sunshine', 'humidity', 'temp_max', 'pressure', 'precipitation', 'temp_min', 'global_radiation', 'temp_mean']. After that, we shuffle the data and split them into train, validation and test set (with a ratio of 6:2:2). For each (x, y) in the dataset, the input x is the graph that contains unified features of 18 regions in the past few days (depends on the temporal window, default temporal_window=3) and edges (with distance as attribute), and the prediction y is the mean temperature of the 18 regions in the coming day.

3.3. Models, Training and Evaluation

In this experiment, we compared the performance of traditional models and GNNs.

For GNNs, we included graph convolutional network (GCN) (Kipf & Welling, 2016), graph attention network (GAT) (Veličković et al., 2017) and GraphSAGE (Hamilton et al., 2017). Here, GCN utilises edge attributes by taking the inverse of each edge attribute (distance) as edge weight. We implemented the GNN architectures and training with PyTorch Geometric.

For traditional ML models, we included linear regression, multi-layer perceptron and random forest. To train the traditional models, we convert the graph features into tabular features, which input is the concatenation of the node attributes and edge attributes of the 18 regions. Additionally, we included a baseline that predicts today's mean temperature with yesterday's mean temperature.

The default setting of the graph is forming incoming edges with the top-3 nearest neighbours, and each node has a temporal windows of 3. For all trainable models, their parameters learnt using Adam optimiser with rate 0.01. The number of layers and batch size of GNNs are 3 and 64 respectively.

To compare the performance of different models, we used mean square error (MSE) as the evaluation metrics. We trained each model for 3 times, and show the mean and standard deviation of the results on the table.

Table 1. Result comparison between models under default setting

MODELS	TRAIN LOSS	VALIDATION LOSS	TEST LOSS	TIME (IN S)
YESTERDAY	4.7±0.0	4.8±0.0	4.7±0.0	**0.0**
GCN	21.2±0.0	21.4±0.2	21.2±0.4	41.6
GCN_EDGE	15.9±0.2	16.5±0.3	16.5±0.4	42.2
GAT	26.9±3.4	27.1±7.6	27.3±8.0	58.5
GAT_EDGE	24.4±0.0	21.2±0.2	21.0±0.2	59.1
GRAPHSAGE	4.6±0.2	6.5±1.6	6.6±1.7	43.9
LR	4.4±0.0	4.7±0.0	4.5±0.0	0.1
MLP	5.4±0.1	6.0±0.3	5.5±0.3	1.1
RF	**0.6±0.0**	**4.3±0.0**	**4.1±0.0**	38.7

Table 2. Test loss of models with different k-nearest neighbour

k	1	3	5	7	10
GCN	**16.4±0.4**	21.1±0.5	21.2±0.2	22.5±0.1	24.5±0.2
GCN_edge	**13.7±0.3**	14.7±0.3	16.2±0.3	17.4±0.2	18.4±0.3
GAT	**15.9±1.5**	18.0±4.1	18.9±8.1	20.3±5.2	29.5±4.4
GAT_edge	13.4±1.8	9.0±4.9	6.4±3.2	8.1±3.3	**5.3±0.5**
GraphSAGE	6.8±1.4	7.8±1.5	11.7±1.7	8.6±1.2	**4.7±1.3**
LR	**4.4±0.0**	4.5±0.1	4.4±0.0	4.5±0.0	4.6±0.0
MLP	5.3±0.4	**5.2±0.6**	5.7±0.1	5.5±0.3	6.1±0.4
RF	4.1±0.0	4.1±0.0	**4.1±0.0**	4.1±0.0	4.1±0.0

4. Experiment & Analysis

4.1. Evaluation on models under default setting

From Table 1, we find that most models performed worse than the baseline model. The baseline model, which predicts current mean temperature with yesterday mean temperature, only has a MSE of 4.7 on testing data. While traditional models like linear regression and random forest got better results (4.5 and 4.1 respectively), most GNNs attained unfavourable results. Except GraphSAGE, other GNNs models converged early (Figure 1 and Figure 2)had MSE larger than 15 on training, validation and test set, implying they cannot effectively learn the prediction task from the dataset. To improve the result, we experimented with different data preprocessing and training hyperparameters.

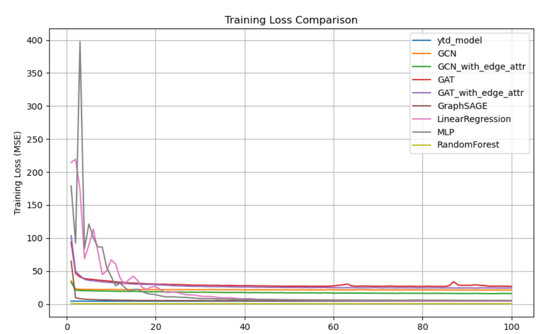

Figure 1. Training loss of all models under default setting

Table 3. Test loss of models with different temporal window

TEMPORAL WINDOW	1	3	5	10	30	50
GCN	21.1±0.1	**20.8±0.2**	21.3±0.8	21.2±0.2	21.3±0.3	22.2±1.1
GCN_EDGE	15.3±0.5	15.2±0.3	**15.0±0.5**	15.7±0.5	15.1±0.1	15.6±1.2
GAT	**19.5±1.1**	27.0±5.2	35.6±2.8	31.6±6.7	35.4±3.4	48.1±15.6
GAT_EDGE	**7.1±1.4**	9.8±2.0	14.1±4.9	30.5±12.7	34.9±10.8	55.6±19.8
GRAPHSAGE	**5.5±0.8**	6.0±0.9	6.9±1.9	8.0±0.9	8.6±1.8	12.1±1.8
LR	**3.6±0.0**	3.8±0.0	3.9±0.0	4.8±0.1	5.8±0.1	6.8±0.1
MLP	**4.0±0.1**	4.3±0.2	4.2±0.1	6.2±1.4	7.3±1.7	7.6±0.6
RF	**3.8±0.0**	4.1±0.0	4.0±0.0	4.2±0.0	4.2±0.0	4.5±0.0

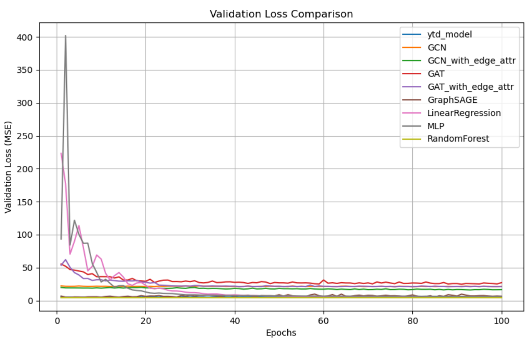

Figure 2. Validation loss of all models under default setting

Table 4. Test loss (and seconds needed for training 100 epochs) of GNNs with different batch size

BATCH SIZE	1	4	32	64	256
GCN	26.6±0.4	23.9±2.6	**20.9±0.1**	21.2±0.7	21.4±0.6
GCN_EDGE	25.1±3.2	17.8±1.6	15.0±0.5	**14.7±0.2**	15.3±0.5
GAT	59.1±17.0	59.2±17.0	25.7±5.8	**18.1±4.7**	20.5±1.3
GAT_EDGE	57.9±18.7	36.4±2.4	10.1±1.0	**6.5±1.5**	9.7±3.8
GRAPHSAGE	10.3±2.4	7.0±0.6	6.7±1.8	7.2±0.3	**6.3±0.7**

4.2. Adjusting preprocessing hyperparameters

We first investigated the impact of k (number of edges/ neighbouring nodes per station) on the results. Shown in Table 2, as k increases, most models performed worse except GAT_edge and GraphSAGE. The majority of the models perform the best when there is only 1 neighboring node, while GAT_edge and GraphSAGE achieved the optimum (5.3 and 4.7 respectively) when k=10.

In Table 3, we found that a larger temporal window does not improve the result; conversely, it significantly harmed the accuracy of GAT and GAT_edge.

4.3. Adjusting model and training hyperparameters

Furthermore, we investigated the impacts of batch size and number of layer on accuracy. In Table 4, we identified that a smaller batch size not only increases training time but also harm the results. Most models reached the best and stable results when batch size is larger than or equal to 32.

Table 5. Test loss of GNNs with different number of layers

# OF LAYERS	1	2	3	4	5
GCN	**18.1±0.2**	19.9±0.4	20.7±0.1	21.2±0.2	21.7±0.2
GCN_EDGE	**9.4±0.1**	13.2±0.2	14.7±0.1	16.1±0.6	17.6±0.7
GAT	**12.6±4.1**	19.7±1.5	19.0±3.4	22.9±1.8	22.4±0.4
GAT_EDGE	11.5±2.3	**5.3±0.3**	7.1±3.3	11.4±7.1	45.6±6.0
GRAPHSAGE	4.0±0.0	**3.9±0.3**	7.7±1.9	13.6±0.4	6.6±1.2

Also, more layers does not necessarily lead to better results. In this simple task, most GNNs achieved the optimal results when the model is only consist of 1-2 layer(s) according to Table 5.

Note, considering edge attributes (like GCN_edge and GAT_edge) could give models extra information and improve the results. More detailed results can be found in the 'TemperatureForecast/result_comparison/'.

4.4. Theoretical Analysis

As GNN can take in any graph structure, the existing model can still be used even when new edges or nodes are added to the graph; on the other hand, the graph structure and the sequence of the input attributes needed to be the same in order to reuse traditional ML models.

However, GNNs have a higher computational cost compared to traditional models, in turn taking longer time and more compute to train (as shown in Table 1).

5. Improvement

Witnessing the unsatisfactory performance of GNN, we were encouraged to look into the problem and explore methods for improvement. Our results below show that with suitable techniques implemented and useful features added, GNNs can also attain a competitive results as other models in predicting temperature, or even surpassing them.

5.1. Adding spectral and temporal feature

Temporal features have strong relationship with temperature. For instance, knowing the date could help the model infer the season that it is predicting, and season gives a great hint for predicting the temperature. Similarly, spectral features

Table 6. Result comparison after adding temporal-spectral features (most models have worse performance)

k	Train Loss	Validation Loss	Test Loss
GCN	19.2±0.8	14.9±0.1	14.8±0.0
GCN_edge	14.1±0.3	9.7±0.2	9.4±0.2
GAT	15.8±2.7	13.7±2.2	13.7±2.3
GAT_edge	13.8±0.5	18.5±0.7	18.5±0.7
GraphSAGE	10.4±1.5	6.9±0.9	6.7±1.0
LR	9.8±0.6	9.7±0.6	9.7±0.6
MLP	8.0±2.1	8.2±2.0	7.8±2.1
RF	**0.6±0.0**	**3.9±0.0**	**3.7±0.0**

Table 7. Result comparison after applying normalisation before training and removing year from feature

k	Train Loss	Validation Loss	Test Loss
GCN	12.0±0.0	9.8±0.0	10.3±0.0
GCN_edge	7.7±0.0	5.9±0.1	6.3±0.0
GAT	4.4±0.0	3.6±0.1	4.0±0.0
GAT_edge	6.6±2.8	8.5±3.3	6.0±2.6
GraphSAGE	4.0±0.0	4.5±0.2	**3.7±0.0**
LR	3.6±0.0	4.1±0.0	3.7±0.0
MLP	4.2±0.6	5.4±0.8	4.0±0.4
RF	**0.5±0.0**	**3.7±0.0**	4.1±0.0

help the model identify the relationship between the geographical location and weather, as some locations are more easily subject to certain kinds of weather phenomena (Exp: precipitation and sunshine). Therefore, we expand the set of features to include the 2 kinds of features.

For temporal feature, we append the date, month and year to the feature set. Due to the cyclical nature of date and month, we encode them with cyclical encoding by

$$\sin\left(\frac{2\pi \cdot \text{month}}{12}\right), \quad \cos\left(\frac{2\pi \cdot \text{month}}{12}\right)$$

$$\sin\left(\frac{2\pi \cdot \text{day}}{365}\right), \quad \cos\left(\frac{2\pi \cdot \text{day}}{365}\right)$$

We directly included the year without any preprocessing.

For spectral feature, instead of appending latitude and longitude, we added the Cartesian Coordinates (x, y, z) as it handles periodicity and it is easier for the model to calculate Euclidian distance, and Trigonometric Positional Encoding to help the model distinguish points with similar cosine and sine values but different angles. Although theoretically sound, as shown in Table 6 most of the models have worse performance after adding spectral and temporal features compared to original performance.

5.2. Normalisation before training

We suspected that the model failed to learn effectively due to the large value range for certain features. So, we normalize each feature with mean 0 and standard deviation 1 except temp_mean, and remove year from the feature list as it does not have strong relevancy with temperature forecast.

After normalization, the mean square error of GNN (including GCN, GAT and GraphSAGE) decrease substantially, as shown in table 7 . This shows that unlikely traditional models that we included in the experiment, which were not benefited hugely from normalization, GNNs depend heavily on normalization.

5.3. Temporal Encoder

Past data should help the model to discover trend and make a more accurate prediction. However, Table 3 shows that a smaller temporal leads to a better result. To help the model to learn from historical data, we added a transformer block as temporal encoder before the graph neural network. We chose transformer as the encoder for a few reasons. First, compared to other sequence models (such as LSTM and RNN), transformer is easier to train and require less historical data.

Similar to natural language processing, we view a date as a token, and its relevant features as token embeddings. In each embeddings, we included, weather features, spatial features and temporal features of the day that we are predicting on. We implemented the temporal encoder with a standard transformer block, where we added a learnable positional encoding to the features to distinguish different dates, applied single-head self-attention to the features of historical dates and layer normalization after to stablise training, residual connection and MLP to increase non-linearity.

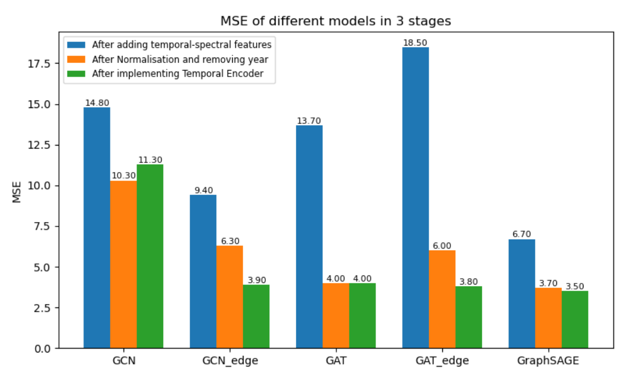

Figure 3. Result comparison of different models after the 3 stages

The result can be found in Table 8, which can be seen that all GNNs gain a huge boost in performance, with some

Table 8. Test loss of different window size after adding temporal encoder

TEMPORAL WINDOW	1	3	5
GCN	12.5±0.5	**11.3±0.2**	11.4±0.5
GCN_EDGE	5.2±0.0	**3.9±0.0**	7.6±3.0
GAT	5.5±0.6	4.2±0.3	**4.0±0.1**
GAT_EDGE	4.7±0.0	3.9±0.2	**3.8±0.1**
GRAPHSAGE	5.6±1.0	3.6±0.0	**3.5±0.1**

of the models surpassing random forest by a significant percentage. This result shows that historical data could help the model make a more accurate prediction. As graph neural network is not good at processing time-series data, adding a temporal encoder or time-series kernel in the GNN layer could combat the painpoint of the model.

Figure 4. Example 1 of unstable training: training loss of GNN after implementing temporal encoder with 5 windows

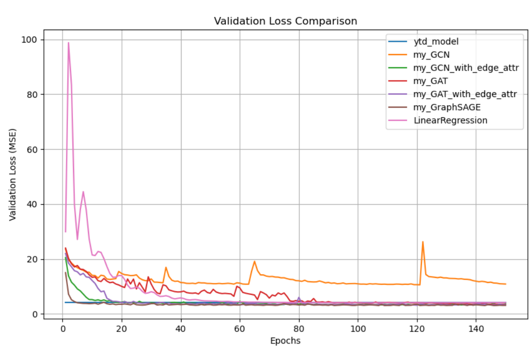

Figure 5. Example 2 of unstable training: validation loss of GNN after implementing temporal encoder with 5 windows

5.4. Other observations

During our experiment, we found that GNNs fluctuate a lot and are subject to noise, and the loss sometimes exploded in the halfway of training (as shown in Figure 4 and Figure 5, regardless of good performance in previous epochs. We think that gradient explosion is one of the causes. Besides, compared to traditional neural networks, GNNs are more prone to overfitting.

6. Limitation & Future Work

One of the biggest limitations to better performance is the dataset. Although we have already added temporal and spatial features, there is still insufficient range of feature. More weather features, such as occurrence of extreme events and wind directions, could enhance the representation of the real-world situation. Besides, if the data is collected in a more frequent basis, such as every 3 hours, more trends and patterns could be extracted.

Another limitation is time and computational power. As most of the time is spent on debugging and improving the GNNs, we are not able to do hyperparameter tuning and ablation studies to discover more in-depth insights, include other SOTA models in the studies for comparison, and implement more advanced methods and sophisticated architectures to push the limit.

In the future, we hope to evaluate and compare the performance of the Google's time-series foundation model ((Das et al., 2024)) and Alibaba's DLSTM (Li et al., 2023). In addition, we hope to work on a practical dataset with more diversified features and more frequent data points.

7. Conclusion

With the findings on hyperparameter in part 4 and the 3 stages of refinement in part 5, we successfully improved the performance of GNNs and trained a reliable spectral-temporal GraphSAGE to predict tomorrow's temperature. We have shown that given limited computational power (without using any GPU and cloud server), it is achievable to train a machine learning model to predict tomorrow's mean temperature with error less than 2 degree Celsius. We believe these findings would be beneficial on future work, and we will be working on temperature prediction with more sufficient data in practical use cases.

References

Brotzge, J. A., Berchoff, D., Carlis, D. L., Carr, F. H., Carr, R. H., Gerth, J. J., Gross, B. D., Hamill, T. M., Haupt, S. E., Jacobs, N., McGovern, A., Stensrud, D. J., Szatkowski, G., Szunyogh, I., and Wang, X. Challenges and opportunities in Numerical weather prediction. *Bulletin of the American Meteorological Society*, 104(3):E698–E705, 2 2023. doi: 10.1175/bams-d-22-0172.1. URL https://journals.ametsoc.org/view/journals/bams/104/3/BAMS-D-22-0172.1.xml.

Das, A., Kong, W., Sen, R., and Zhou, Y. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs, 6 2017. URL https://arxiv.org/abs/1706.02216.

Huber, F., van Kuppevelt, D., Steinbach, P., Sauze, C., Liu, Y., and Weel, B. Will the sun shine? – an accessible dataset for teaching machine learning and deep learning, 2024. URL https://proceedings.mlr.press/v207/huber23a/huber23a.pdf.

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks, 9 2016. URL https://arxiv.org/abs/1609.02907.

Klein Tank, A. and Coauthors. Daily dataset of 20th-century surface air temperature and precipitation series for the european climate assessment. *International Journal of Climatology*, 22:1441–1453, 2002. URL http://www.ecad.eu.

Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., Miele, A., Kashinath, K., and Anandkumar, A. FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators. *association for computing machinery*, pp. 1–11, 6 2023. doi: 10.1145/3592979.3593412. URL https://doi.org/10.1145/3592979.3593412.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., and Battaglia, P. GraphCast: Learning skillful medium-range global weather forecasting, 12 2022. URL https://arxiv.org/abs/2212.12794.

Li, Y., Li, T., Lv, W., Liang, Z., and Wang, J. Prediction of daily temperature based on the robust machine learning algorithms. *Sustainability*, 15(12):9289, 6 2023. doi: 10.3390/su15129289. URL https://www.mdpi.com/2071-1050/15/12/9289.

Lobell, D. B. and Field, C. B. Global scale climate–crop yield relationships and the impacts of recent warming. *Environmental Research Letters*, 2(1):014002, 3 2007. doi: 10.1088/1748-9326/2/1/014002. URL https://doi.org/10.1088/1748-9326/2/1/014002.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks, 10 2017. URL https://arxiv.org/abs/1710.10903.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How Powerful are Graph Neural Networks?, 10 2018. URL https://arxiv.org/abs/1810.00826.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph Neural Networks: A Review of Methods and applications, 12 2018. URL https://arxiv.org/abs/1812.08434.