Weather Forecasting Using GNN

Presented by: Badal Basnet (04), Prastut Bhattarai (07), Shaswat Bhushan Jangam (11), Samir (14)

Supervisor: Harish Chandra Bhandari

Kathmandu University

December 21, 2025

Outline

Introduction

- Weather forecasting is crucial for understanding and predicting atmospheric conditions.
- Numerical Weather Prediction (NWP) is a method using computer tools and mathematical models.

Project Objectives

1. Construct a graph dataset for processing from the weather data.
2. Predict the node feature for a long time step (7-day forecast).
3. Observe the accuracy of the trained model.

Graph Neural Network (GNN)

- Graph Neural Networks are a class of neural networks designed for graph-structured data.
- They can operate on graphs and capture complex relationships in data.
- GNNs have found applications in various domains, including social network analysis, recommendation systems, and, in our case, spatio-temporal forecasting.

Spatio-temporal graph

We construct a graph with static structure and time-varying features. Mathematically, it is represented as $G(V, E, X_V(t))$, where:

- V is the set of nodes representing spatial entities.
- E is the set of edges representing relationships or connections between spatial entities.
- X is the set of node features representing the spatial attributes.
- T is the set of timestamps representing the temporal evolution.

Graph Convolutional Networks (GCN)

Mathematical Formulation:

$$H^{(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

where:

- $H^{(l)}$ is the node feature matrix at layer l.
- $\hat{A}$ is the adjacency matrix with added self-loops.
- $\hat{D}$ is the degree matrix of $\hat{A}$.
- $W^{(l)}$ is the weight matrix at layer l.
- σ is the activation function.

Static Temporal Graph Convolutional Network (TGCN)

Mathematical Formulation:

$$H_v^{(t+1)} = \sigma\left(\sum_{u \in N(v)} A_{u,v} \cdot W \cdot H_u^{(t)} + b\right)$$

Explanation:

- $H_v^{(t+1)}$: Hidden state of node v at time $t+1$
- $H_u^{(t)}$: Hidden state of node u at time t
- $N(v)$: Neighbors of node v
- $A_{u,v}$: Adjacency matrix element between nodes u and v
- W: Learn able weight matrix
- b: Bias term
- σ: Activation function (e.g., ReLU)

Training

Mathematical Formulation:

$$\text{cost} = \text{cost} + \frac{1}{T} \sum_{t=0}^{T} (\hat{y}^{(t)} - y^{(t)})^2$$

$$\text{cost} = \frac{\text{cost}}{T+1}$$

Explanation:

- cost: Accumulated cost over time
- $\hat{y}^{(t)}$: Predicted value at time t
- $y^{(t)}$: True value at time t
- T: Temporal depth

The cost is updated for each time step by adding the squared difference between the predicted and true values. Finally, the cost is normalized by the temporal depth.

Optimization

Optimization Algorithm:

- Adam Optimizer

Hyper parameters:

Hyper parameter	**Value**
Learning Rate	0.01
Number of Epochs	100
Train and Test ratio	50:50

Table: Hyper parameters used in training.

MODEL Architecture

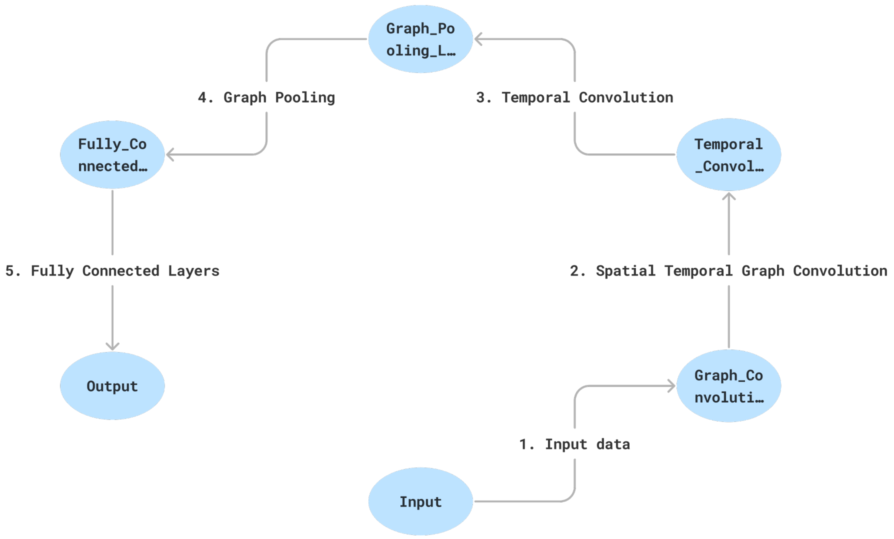


Figure: STGCN MODEL Architecture

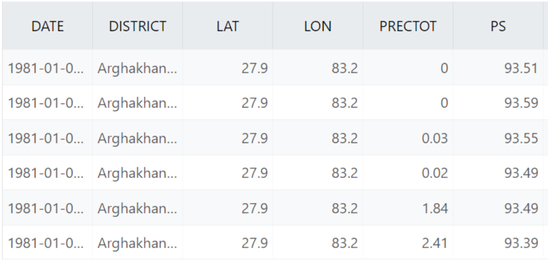
Data Collection

We collected data from the **OPEN DATA NEPAL** website,

Dataset Details:

- **Title:** District Wise Daily Climate Data for Nepal
- **Source:** NASA Langley Research Center (LaRC) POWER Project
- **Funding:** NASA Earth Science/Applied Science Program

Dataset Description: The dataset contains climate data for Nepal, providing insights into various climate parameters.



DATE	DISTRICT	LAT	LON	PRECTOT	PS
1981-01-0...	Arghakhan...	27.9	83.2	0	93.51
1981-01-0...	Arghakhan...	27.9	83.2	0	93.59
1981-01-0...	Arghakhan...	27.9	83.2	0.03	93.55
1981-01-0...	Arghakhan...	27.9	83.2	0.02	93.49
1981-01-0...	Arghakhan...	27.9	83.2	1.84	93.49
1981-01-0...	Arghakhan...	27.9	83.2	2.41	93.39

Figure: CSV-Tabluar view of the dataset

NODE Definition

In our implementation, we consider **DISTRICT** as a node, each representing a weather station. The following are the districts treated as nodes:

- Arghakhanchi, Baglung, Baitadi, Bajang, Banke, Bara, Bardiya, Bhaktapur, Darchula, Dhading, ...

Total Nodes: 62
Mathematically,
$G(V, E, X_V(t))$,
where:

- V (node) represents the set of districts (DISTRICT).

Process: We collect climate data from different districts, treating each district as a weather station. These districts serve as nodes in our spatio-temporal graph. The collected data forms the node features, and the relationships between districts form the edges.

EDGE: Implementation

To establish connections between DISTRICT nodes, we utilize the geographical coordinates (latitude and longitude). The process involves determining the distance between each pair of nodes and connecting them if the distance is less than 40 kilometers.

Geodesic Distance Formula: The geodesic distance D between two points given their latitude ϕ_1, ϕ_2 and longitude λ_1, λ_2 is calculated using the Haversine formula:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cdot\cos(\phi_2)\cdot\sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2\cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$D = R\cdot c$$

where $\Delta\phi = \phi_2 - \phi_1$, $\Delta\lambda = \lambda_2 - \lambda_1$, and R is the Earth's radius (mean radius = 6,371 km).

By applying this formula, we connect nodes within a 40-kilometer range,

Node Feature (Time-varying Feature)

For the node features, we focus on a single feature: **PS (Surface Pressure)**. This feature serves as a temporal attribute for the static graph. Mathematically, our representation is given by $G(V, E, X_V(t))$, where:

- $V(t)$ represents the node features at time t.

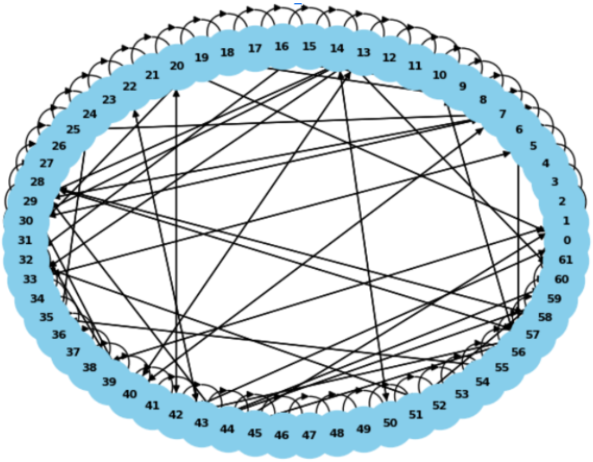
In our context:

- $V(t) = \text{PS}[t]$, where t denotes time.

Surface Pressure (PS): Surface pressure is a critical meteorological parameter, and by considering it as a time-varying feature, we aim to capture its temporal evolution across the graph.
This single feature, evolving over time, contributes to the overall spatio-temporal dynamics of the weather data in our graph.

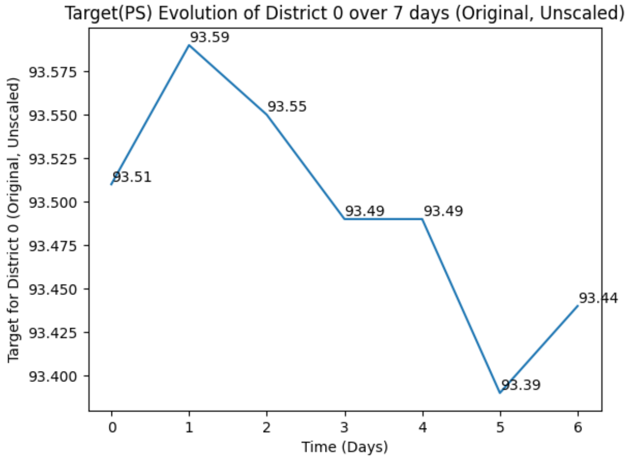
Graph Plot using NetworkX and Matplotlib

To visualize the constructed spatio-temporal graph, we utilize the NetworkX library for graph operations and Matplotlib for plotting.
Graph Plot:



Using Matplotlib we visualize the temporal aspect of a specific node at specific time

Temporal Plot:



Data Representation Summary

The data representation follows the format commonly used in graph-based deep learning tasks, particularly in the PyTorch Geometric library. The key components include:

- *x*: **Node Feature Matrix**
  - Shape: (62 nodes, 7 features)
  - Represents the node features of the graph. Each node has a feature vector of length 7.
- *edge_index*: **Edge Connectivity Information**
  - Shape: (2, 340)
  - Represents the connectivity of the edges in the graph. Each column in *edge_index* corresponds to an edge, with the two rows representing the source and target nodes of that edge.
- *edge_attr*: **Edge Feature Matrix**
  - Shape: (340 features,)
  - Represents the features associated with each edge in the graph. The number of features for each edge is determined by the task or data source.

Data Representation Summary

- *y*: **Node Target Matrix**
  - Shape: (62 nodes, 7 targets)
  - Represents the target values associated with each node. Each node has a target vector of length 7.

In summary, the data describes a graph with 62 nodes, where each node has 7 features and 7 targets. Edge information is captured by *edge_index*, and edge features are stored in *edge_attr*. This format is commonly used in the context of graph neural networks (GNNs) and graph-based machine learning tasks.

Visualization of Predicted Value and True Value

- We present a visual comparison between the predicted surface pressure values and the true surface pressure values for a specific time horizon.
- The figure below illustrates the model's predictions (in blue) and the actual observed values (in orange).

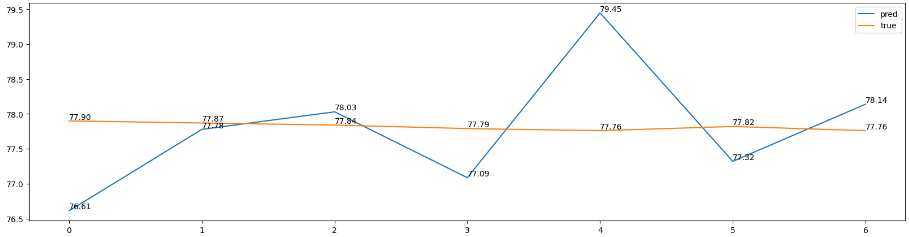


Figure: Comparison of Predicted Surface Pressure vs. True Surface Pressure of district 31(KATHMANDU) at 0 time step

OVERALL Prediction

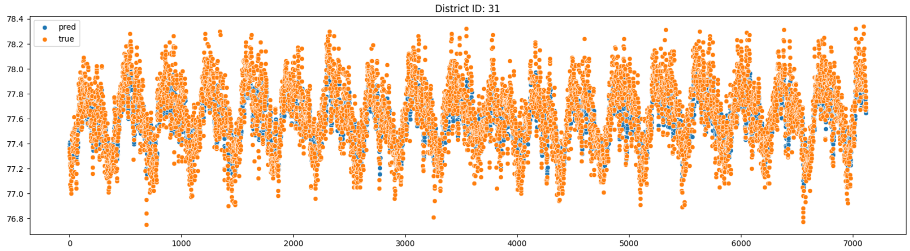


Figure: Comparison of Predicted Surface Pressure vs. True Surface Pressure of district 31 (KATHMANDU) of Test values

Conclusion

- In conclusion, our Weather Forecasting model using Spatio-temporal Graph Neural Networks (STGNN) has shown promising results in predicting surface pressure over a 7-day forecast period.
- The constructed spatio-temporal graph, incorporating district-wise weather data, effectively captured the dynamic relationships between different regions.
- The visual comparison between predicted and true surface pressure values indicates the model's ability to generalize and make accurate forecasts.
- This research contributes to the field of weather forecasting by leveraging the power of Graph Neural Networks to model complex spatio-temporal dependencies in climate data.

Future Work

- Incorporate additional meteorological features: Expand the model by including more diverse weather parameters to enhance prediction accuracy.
- Fine-tune model hyperparameters: Experiment with different configurations and hyperparameters to optimize the model's performance.
- Ensemble methods: Explore the potential of combining multiple models to further improve forecasting accuracy.
- Real-time implementation: Develop a real-time forecasting system that continuously updates predictions as new weather data becomes available.
- Climate pattern analysis: Investigate long-term climate patterns and trends using the developed model to gain insights into regional climate dynamics.
- Collaborate with domain experts: Engage with meteorologists and climatologists to validate model outputs and incorporate

Questions and Answers

- Open Floor for Questions
- Discussion and Feedback