# Weather Modelling using Ground Sensor Data and Hierarchical Transformers

A Preprint

**Yash Bonde**[*]
Raipur, India
bonde.yash97@gmail.com

October 5, 2020

## Abstract

Weather forecasting is a crucial human endeavour on which many aspects of our modern society depend upon, from energy, transportation to sports events and defence. There is a growing community applying machine learning to weather forecasting and modelling, majority of the results, however have not been consequential. Most of the studies do not consider the geospatial and temporal nature of weather phenomenon and often attempt to predict lone features, without considering the external effects. Larger research efforts attempt modelling using large datasets, usually satellite imagery, and compute unavailable to average researcher. We demonstrate a new method which uses ground sensor data and graph neural networks to create temporal weather models. For training we use data from National Meteorological Institute - Brazil, which has thousands of ground sensor data points logged every hour for over 20 years. To validate we fill the corrupted parts of data and show that our model successfully predicted the features. We further demonstrate that this model is a platform that can be extended to more datapoints, ultimately creating a global weather modeller.

***K*eywords** Weather Modelling · Graph Neural Networks · Hierarchical Transformers

## 1 Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

[*] yashbonde.github.io

# 2 Related Work

## 2.1 Lone Features

## 2.2 Satellite Features

# 3 Weather Dataset

We use ground sensor data instead of satellite images, one of the largest available datasets is from the National Meteorological Institute - Brazil [2]. Historical data is available at from year 2000 to the day downloaded. It contains an hourly data with 21 features, discussed in detail in Appendix 6

## 3.1 Sampling

The data used did not have all the weather stations active at the same time instead, they were active in buckets at different times. Fig 3.1 shows the number of stations active on any given moment, for consistency we use (DD/MM/YYYY) format for dates.

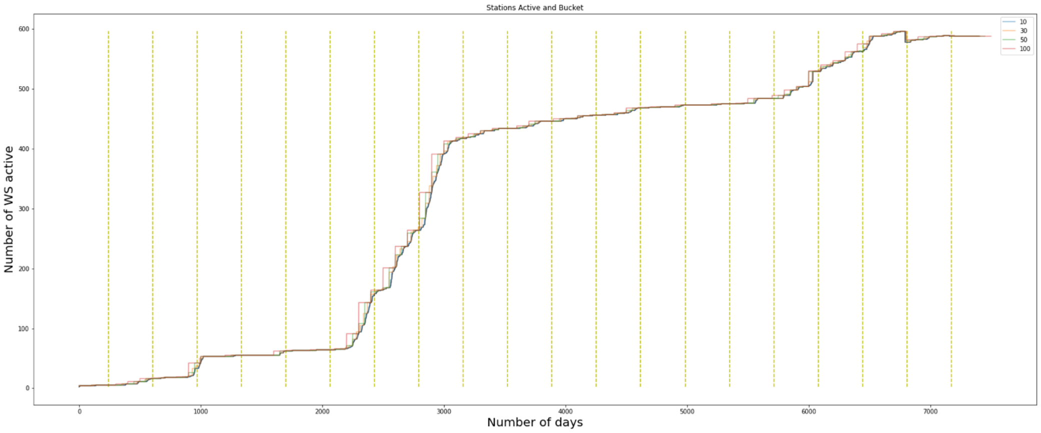

Figure 1: **Activity Plot**, x-axis represents the days since first record (07/05/2000) till (30/08/2020) and y-axis represents the number of stations active on that day. Each day is further sampled to 24 hours (not shown for visual representation). Vertical yellow lines represent 1st January from 2001 to 2020. Overlay lines represent the maximum nodes active in sliding window of $d$ days.

## 3.2 Great Circle Distance

Great circle distance is defined as the the distance between two points on surface of a sphere. Let $A$ and $B$ be two points on the surface with latitude $\phi_A, \phi_B$ and longitudes $\lambda_A, \lambda_B$ in radians. Then $\Delta\lambda$ and $\Delta\phi$ is the absolute difference between the two; central angle $\Delta\sigma$ is given as

$$\Delta\sigma = \arccos(\sin\phi_A \sin\phi_B + \cos\phi_A \cos\phi_B \cos(\Delta\lambda)) \tag{1}$$

However this cannot be programmed directly because it causes rounding errors in float values. So we use the following formula, distances from which have been manually verified.

$$\Delta\sigma = \arctan \frac{\sqrt{(\cos\phi_B \sin(\Delta\lambda))^2 + (\cos\phi_A \cos\phi_B - \sin\phi_A \cos\phi_B \cos(\Delta\lambda))^2}}{\sin\phi_A \sin\phi_B + \cos\phi_A \cos\phi_B \cos(\Delta\lambda)} \tag{2}$$

Great circle distance $d$ given as $d = r\Delta\sigma$, $r$ being the radius of the sphere.

* Discuss the dataset that I am using * How we are breaking it down * How training and validation examples are created

[2] https://portal.inmet.gov.br/dadoshistoricos

## 4 Graph Neural Networks

Graph Neural Networks are a category of neural networks that provide a framework that take graph data structure as input and extract embeddings from it. A graph is a triplet of nodes $(V)$, edges $(E)$ and global vector $(u)$. GNN blocks process the combinations of this information to compute further embeddings.

GNN blocks can be joined to created all the modern neural network structures, feed forward, encode-process-decode and recurrent. They are a versatile set of networks and have demonstrated their use in modelling physical systems [1, 2, 3, 4], Chemistry problems, Neural alogrithmic execution and many more.

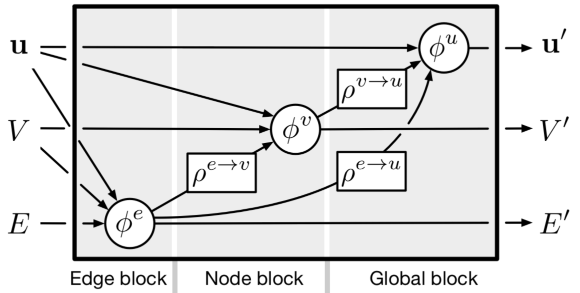

Figure 2: Full GNN block from [5], $u$ is the global vector, $V$ is the set of node vectors and $E$ is the set of edge vectors, $\rho$ are aggregation methods and $\phi$ are the neural networks

All neural network architectures can be fit in this framework, [5] gave a mathematical model that can be modified and fit to any problem. We first demonstrate how transformers are a subset of fully connected graph networks and then modify the attention mechanism to obtain a global embedding. For full block see Figure 4. The core idea of such a GNN is the message passing, where information in the nodes is passed along to other nodes. This then tells each node to consider the new information obtained from its surroundings to further compute its own vectors. Depending on the requirements the aggregation methods ($\rho$) and neural network architectures $\phi$ can be modified. A generalised node function uses edge, global and node values to extract the new embedding, Algorithm 1.

**Algorithm 1:** Generalised GNN node embedding with edge features

```
Given $n$ nodes, $e$ edges;
for $i \in \{1, ..., N^n\}$ do
    let $E_i' = \{e_k, r_k, s_k\}_{r_k=i;k=1:N^e}$;
    $\bar{e}_i \leftarrow \rho^{e\to v}(E_i)$;
    $v_i' \leftarrow \phi^v(\bar{e}_i, v_i, u)$
end
```

When there is no edge attributes in the system, the node gathering can be modified to use only the node to node connections.

$$v_i' \leftarrow \rho^{v\to v}\phi(v_i, v_j)\forall j \in \{1, ..., N^n\} \tag{3}$$

### 4.1 Transformers are GNNs

Transformers [**?** ] is an architecture that uses parallel feedforward computation that can be scaled to billions of parameters [**?** ] and a multi-head attention to gather the information from each tokens to other tokens. The multi-head attention uses a softmax over dot product of input vectors that are split into multiple heads. The attention mechanism takes query $Q$ and computes the data from memory (called value) $V$ using key $K$.

$$V' \leftarrow softmax\Big(\frac{QK^T}{\sqrt{d_k}} + A\Big)V \tag{4}$$

Here $V$ is the set of all token vectors and $A$ is the attention matrix that tells model where to attend to, this can be used for ignoring padding, casual masking. Keys $K$, value $V$ and query $Q$ are computed by passing node embeddings through a feedforward neural network. This can be compared to Eq. 3, where $\phi$ is the neural network and $\rho$ is the dot product. This shows that transformers can be considered as fully connected GNNs and after training the attention layers demonstrate learning the relevant edge connections for each input token.

This detaches the dependence of learning network from inputs, and so this can be used to input text [**? ?** ], audio [**?** ] and even images [**?** ]. Moreover the structure of attention mask allows input of partially corrupted features by simply ignoring them.

### 4.2 Hierarchical Transformers

We propose an encoder-processor-decoder format, where encoder is tasked with creating global vectors from raw features, processor takes global vectors over multiple time steps and decoder takes the computed global vector for each time step and predicts the future features. In this paper we use transformers as GNNs, each learning a different part of "algorithm" to predict weather.

In all cases we use the following nomenclature, $T$: time-steps in consideration, $N$: number of nodes in the graph, here number of weather stations, $E$: embedding dimension for node vectors, $G$: embedding dimension for global vector $(G > E)$.

**Nodes Encoder** This is a modified GPT block that takes in nodes vector $V \in \mathbb{R}^{T\times N\times E}$ and global vector $U \in \mathbb{R}^{T\times G}$ and computes the output nodes vector $V' \in \mathbb{R}^{T\times N\times E}$ and global vector $U' \in \mathbb{R}^{T\times G}$. First $V$ is passed through a message passing layer ie. multi-head attention and then through a feed forward layer to get $V'$. This output is increased in dimensions using another feedforward layer $\phi_{E\to G}$, and pooling function $\rho_{[T\times N\times G]\to[T\times G]}$. This is added to $U$ and normalised [**?** ] to get $U'$. For details consider Fig. **??**

**Processor** Responsible for handling long term dependencies we started by using Long Short Term Memory cell [**?** ] as it has been demonstrated to successfully model very long dependencies [**? ?** ]. However in order to ensure consistency in model and code we used Transformer-XL [**?** ], that uses stored time-shifted attentions allowing for very long dependencies. This takes in the global vector for each time step $U'$ and returns computed $U'$. We apply the casual masking in order to avoid states from attending to future.

**Nodes Decoder** Similar to encoder it takes in two inputs, a nodes vector $V'$ and global vector $U''$ to give out $V'' \in \mathbb{R}^{T\times N\times E}$. Unlike encoder, decoder does not modify the global vector and instead uses it before the block. We pass $U''$ through a feedforward network that reduces the dimensions $\phi_{G\to E}$, expand the dimensions by tiling $\gamma_{[T\times G]\to[T\times N\times G]}$.

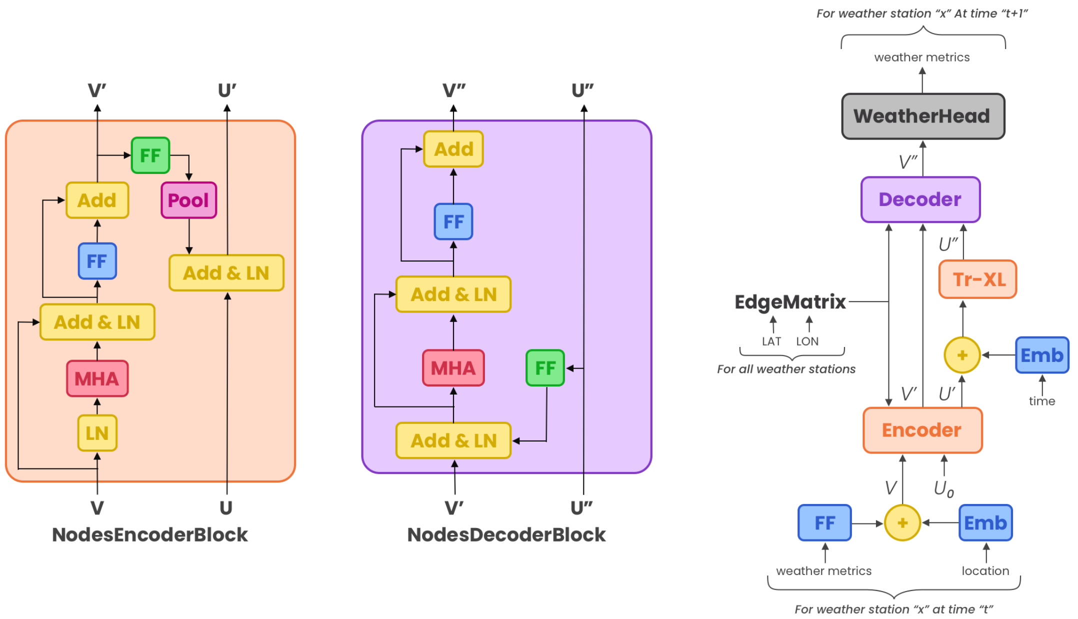

Figure 3: **left**: Nodes Encoder Block is a transformer block where nodes vector $V$ is used to extract global encoding $U$. **centre**: Decoder block is a similar transformer block but the flow of information is from global to nodes. **right**: Complete model with encoder,

Using this in vanilla format does not include the relations between different nodes. In order to pass information about the edge using an edge matrix $E$ were each element is defined as:

$$e_{ij} = \begin{cases} \frac{1}{\sqrt{d_{ij}}} & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $d_{ij}$ is the great circle distance given in Eq. 2. This edge matrix is a constant and added to attention heads along with attention mask in cases where data corruption is observed. The resulting attention formulation looks as follows

$$V' \leftarrow softmax(\frac{QK^T}{\sqrt{d_k}} + A_{\text{corrupt}} + E)V \tag{6}$$

# 5 Results

We modified code from huggingface library [6]. All the code is publicly available [3]

## 5.1 Samples

# 6 Conclusiong/Generalisation Capabilities

Write how we can generalise this system to any number of nodes and edges and scale it up to create global models

# References

[1] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv:2006.11287*, 2020.

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

[3] `https://github.com/yashbonde/vv2`

[3] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[4] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018.

[5] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[6] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

# Appendices

## A Data

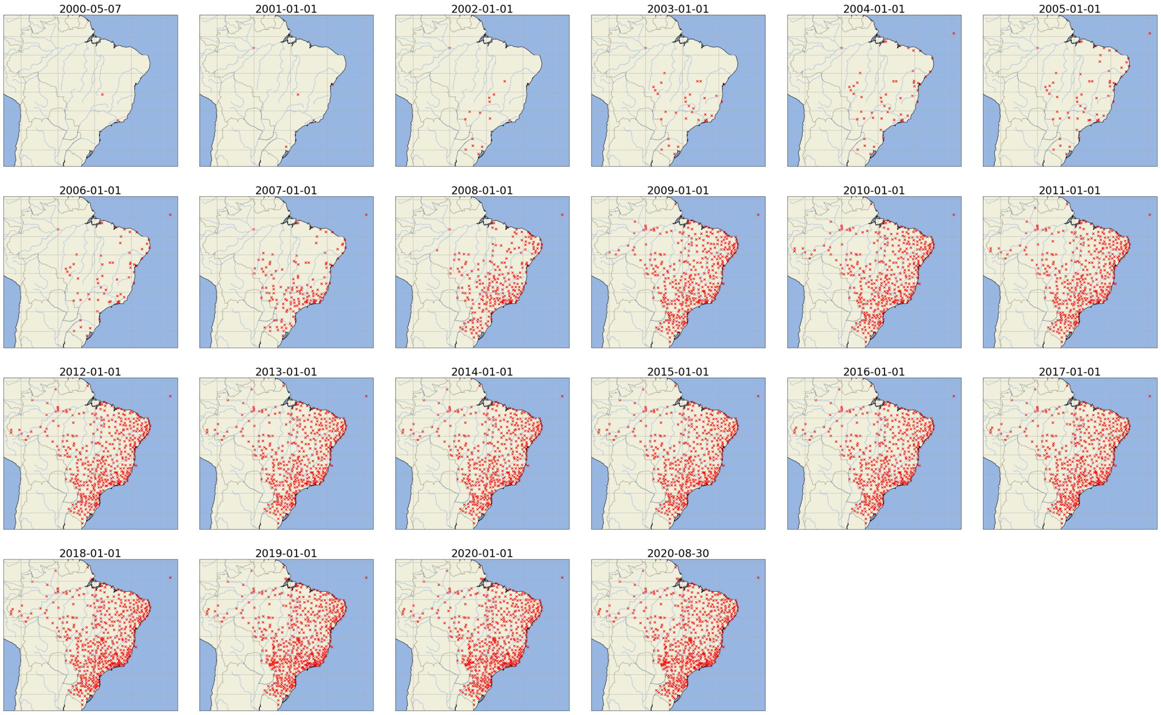

Figure 4: Plotting weather stations by location from the date of first record to the latest record with new years in between. There are sufficient rise number of weather stations after 2007.

Features and data-points available:

- `date`: recording date
- `hour`: recording hour
- `longitude` (°deg): longitude of recording weather station
- `latitude` (°deg): latitude of recording weather station
- `elevation` $(m)$: elevation from sea level of recording weather station
- `total_precipitation` $(mm)$: amount of rainfall in past one hour
- `pressure` $(mB)$: pressure at the time of logging
- `max_pressure` $(mB)$: Maximum pressure in past one hour
- `min_pressure` $(mB)$: Maximum pressure in past one hour
- `radiation` $(KJ/m^2)$: Radiation at the time of logging
- `temp` $(°C)$: Temperature at the time of logging
- `max_temp` $(°C)$: Maximum Temperature in past one hour
- `min_temp` $(°C)$: Minimum Temperature in past one hour
- `max_dew` $(°C)$: Maximum dew temperature in past one hour
- `min_dew` $(°C)$: Minimum dew temperature in past one hour
- `humidity` $(\%)$: Humidity percentage at the time of logging
- `max_humidity` $(\%)$: Maximum humidity percentage in past one hour
- `min_humidity` $(\%)$: Minimum humidity percentage in past one hour

- `wind_gust` ($m/s$): Gust (maximum wind speed) in past one hour
- `wind_speed` ($m/s$): Wind speed at the time of logging
- `wind_direction` ($^\circ$deg): Wind direction in degrees