# Spatiotemporal Graph Neural Networks for Probabilistic Postprocessing of Wind Speed Forecasts

**GDL 2025**
Group id: team 1
Project id: MeteoSwiss

Edoardo Leali, Matteo Vitali, Sidharth Padmanabhan
{edoardopierluigi.leali, matteo.vitali, sidharth.padmanabhan}@usi.ch

**Abstract**

Numerical Weather Prediction (NWP) models are the cornerstone of modern forecasting, yet they suffer from systematic biases and an inability to fully resolve local topographical effects due to limited grid resolution and other factors. To mitigate these errors and provide reliable uncertainty quantification, ensemble forecasts requires statistical postprocessing. This process aims to correct predictions on basis of actual in-site observations. The project explores the application of Spatiotemporal Graph Neural Networks (STGNNs) for the postprocessing of wind speed forecasts across the complex terrain of Switzerland. The primary objective is to determine if explicitly modeling the spatial dependencies between weather stations, encoded as a graph structure, yields superior calibration compared to independent point-based methods.

## 1 Introduction

NWP models simulate the atmosphere using fundamental physical laws. To quantify forecast uncertainty and generate a range of possible outcomes, operational centers run ensemble forecasts—multiple simulations created by slightly perturbing the initial conditions. However, these raw ensembles often suffer from systematic biases and insufficient spread, making statistical postprocessing necessary to calibrate the forecasts against real-world observations.

From a machine learning perspective, this postprocessing is a supervised learning task. The model is designed to map local environmental conditions to a calibrated probability distribution of wind speed. Inputs and outputs are defined as follows:

- **Inputs:** For every station $s$ and forecast lead time $t$, the model receives a feature vector $x_{s,t}$. This vector aggregates dynamic predictors derived from the raw NWP ensemble (e.g., ensemble mean wind speed, standard deviation, and pressure gradients) and static descriptors of the local terrain (e.g., elevation, slope, and topographical position index).
- **Model Task:** The neural network functions as a parameter estimator. By leveraging the graph structure $G = (\mathcal{V}, \mathcal{E})$—where nodes represent stations and edges represent geographical proximity—the model learns to correct local biases by aggregating information from neighboring stations and analyzing the temporal evolution of forecast errors.
- **Outputs:** Instead of a single point forecast, the model outputs the parameters of a conditional probability distribution $P(y_{s,t}|x_{s,t})$ for the observed wind speed $y_{s,t}$. Specifically, we predict mean $\mu_{s,t}$ and std $\sigma_{s,t}$ of a Log-Normal distribution. This approach provides a full uncertainty quantification for every prediction.

The innovation of the approach lies in the fact that standard postprocessing methods typically treat weather stations as independent entities, ignoring the strong spatial correlations inherent in fluid dynamics. This project aims to model these dependencies via Spatiotemporal Graph Neural Networks (STGNNs), trying to yield superior calibration by capturing both the temporal trajectory of the forecast and the spatial propagation of local weather effects.

## 2 Related Works

The transition from classical statistical methods, such as Ensemble Model Output Statistics (EMOS), to deep learning approaches was notably advanced by Rasp and Lerch [1], who demonstrated that neural networks could learn non-linear dependencies between ensemble statistics and observed distributions, significantly outperforming traditional techniques. However, their approach modeled grid points independently, discarding spatial context.

Recently, the integration of spatial information has gained traction. Feik et al. [2] successfully applied Graph Neural Networks (GNNs) to postprocessing, showing that aggregating information from neighboring stations improves forecast skill, particularly in data-sparse regions. This aligns with internal experiments conducted at MeteoSwiss, which utilized a Bi-directional STGNN and hybrid TCN-GNN architectures on fixed distance-based graphs.

Beyond these foundational baselines, the broader spatiotemporal forecasting literature offers advanced architectures relevant to this domain. For instance, Graph WaveNet [3] combines graph convolutions with dilated causal convolutions to capture long-range temporal dependencies without the computational overhead of recurrent networks. Furthermore, attention-based mechanisms, such as Graph Attention Networks (GATs), allow for dynamic edge weighting, potentially enabling the model to learn anisotropic weather patterns that static distance heuristics fail to capture.

## 3 Methodology

Four distinct models were implemented and compared:

- **Model0 (Baseline):** a simple TCN that processes each station independently. Serves as a control baseline to quantify the performance gap when spatial dependencies are ignored.
- **STGNN1 (Bi-STGNN):** an improved version of the already present Bi-directional STGNN
- **STGNN2 (TCN-GNN):** an improved version of the already present hybrid TCN-GNN
- **STGNN3 (TGMM):** Temporal Graph MLP-Mixer, dual-branch architecture that decouples mixing operations, Node-Mixing: processes temporal dynamics, while Global Patch-Mixing captures non-local spatial dependencies.Making it possible to reconstruct structural information gaps without the bottleneck of local message passing.

All models were trained by minimizing the Continuous Ranked Probability Score (CRPS), a proper scoring rule that jointly optimizes the sharpness and calibration of the probabilistic output. To account for sensor failures or maintenance periods in the observational record, we utilized a masked loss function that excludes missing target values (NaNs) from the gradient computation.

For the final assessment, two primary metrics were employed: CRPS and Mean Absolute Error (MAE) for deterministic accuracy (using the predicted median).

Additionally, Rank Histograms (Talagrand diagrams) were used to diagnostically assess the calibration of the ensembles, ensuring the predicted spread accurately reflects the true forecast uncertainty across different lead times.

NOTE: For STGNN1 and STGNN2, the report follows a clear structure: we identify problems in the baseline models, describe how we fixed them, and explain what this means in practice. This match the method and reasoning we adopted, while completing the tasks.

## 4 Implementation

### 4.1 Datasets

We use a wind speed dataset from MeteoSwiss. The training and validation data come from reforecasts (February 2020–September 2023), while the test data come from operational forecasts (May 2024–January 2025).

The dataset includes 40 available features, from which we select 18 predictors for model training based on the default configuration. These consist of:

- **4 NWP ensemble features** from ICON-CH2-EPS: ensemble mean and standard deviation of 10-meter wind speed, plus East-West (Geneva-Güttingen) and North-South (Basel-Lugano) pressure gradients
- **4 temporal features**: cyclical encodings of hour-of-day and day-of-year (sine and cosine transformations to capture periodic patterns)
- **10 terrain features**: station elevation, distance to Alpine ridge, topographic position indices at different scales, terrain roughness, valley indicators, and terrain gradients at multiple spatial scales

This feature selection focuses on the most meteorologically relevant predictors while keeping computational costs manageable.

To enable graph-based learning, we build a fixed spatial graph where nodes represent weather stations. We compute the adjacency matrix $\mathbf{A}$ using a Gaussian kernel over Haversine distances between station coordinates. To keep computation efficient and focus on local connections, we use $k$-Nearest Neighbors ($k = 5$) with a minimum edge weight threshold. This graph structure assumes that nearby stations share similar weather conditions.

### 4.2 Models

**Model0 (TCN Baseline).**

To measure whether graph-based spatial modeling actually helps, we implement a purely temporal baseline using Temporal Convolutional Networks [**bai2018empirical**]. Model0 processes each station independently through stacked dilated convolutions with exponentially growing receptive fields (dilation $= 2^l$ for layer $l$). Each layer produces both a residual output and a skip connection, allowing the model to extract features at multiple time scales. Final predictions combine these features: $\mathbf{h}_{\text{out}} = \sum_{l=1}^{L} \mathbf{h}_l^{\text{skip}} + \mathbf{h}_L$.

An important design choice is the use of **non-causal convolutions** (causal_conv $=$ False). Unlike true forecasting tasks where future timeteps are unknown, postprocessing has access to the entire forecast window simultaneously—all lead times $\tau = 1, \ldots, 96$ are available at once from the NWP model output. Non-causal convolutions can look both backward and forward in time within this window, using symmetric padding rather than left-padding only. This allows the model to use full temporal context: when correcting the forecast at lead time $\tau$, it can observe patterns from both $\tau - k$ and $\tau + k$, helping identify whether a predicted wind event is an isolated spike or part of a sustained pattern (e.g., multi-day weather events). The same reasoning applies to following architectures.

This architecture captures long-range temporal patterns through dilated convolutions, but it completely ignores spatial relationships between stations. This makes it a good baseline to test whether adding spatial modeling improves performance.

**STGNN1 (Enhanced Bidirectional STGNN).**

The original BiDirectionalSTGNN used bidirectional recurrent graph networks with simple message passing (GatedGraphNetwork) and additive temporal residuals. However, it had three main problems:

1. **No deep spatial learning**: Each graph layer received the same concatenated input (raw features + hidden states), which prevented the model from learning higher-level spatial patterns where deeper layers build on earlier layers.
2. **Equal weighting of neighbors**: All neighboring stations were treated equally, ignoring important directional weather patterns (e.g., upwind/downwind relationships during Föhn winds, elevation effects).
3. **Poor gradient flow**: Simple additive residuals ($\mathbf{h}_t = \mathbf{h}_{t-1} + \Delta\mathbf{h}_t$) didn't allow gradients to flow well

through long time sequences, causing vanishing gradients during training.

Our improved architecture fixes these three problems:

1. **Layer-by-layer processing**: We implement true deep learning where each layer takes the previous layer's output as input:

$$\mathbf{h}_t^{(l)} = \text{GRU}\left(\text{GAT}^{(l)}(\mathbf{h}_t^{(l-1)}, \mathcal{E}), \mathbf{h}_{t-1}^{(l)}\right)$$

   In this case, GATv2 processes one timestep at a time, providing spatially-aggregated features to the GRU before temporal updates. This lets the model learn patterns at multiple scales: early layers capture local station relationships while deeper layers learn regional weather patterns. We also use a three-layer readout MLP ($2H \cdot L \to 2H \to H \to H$) with LayerNorm and SiLU activations to better combine the bidirectional states.
2. **Attention-based aggregation**: We replace uniform message passing with GATv2 [4], which learns to weight neighbors dynamically using multi-head attention. The attention mechanism for a single head computes:

$$\alpha_{ij} = \text{softmax}_j\left(\mathbf{a}^\top \text{LeakyReLU}(\mathbf{W}[\mathbf{h}_i|\mathbf{h}_j])\right)$$

   This allows the model to automatically focus on the most relevant neighboring stations (e.g., upwind nodes) depending on the current state.
3. **GRU cells for time modeling**: We replace manual residuals with GRU cells that have learnable gates: $\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$. The update gate $\mathbf{z}_t$ balances old vs. new memory ,while the reset gate $\mathbf{r}_t$ (used to compute the candidate state $\tilde{\mathbf{h}}_t$) allows the model to drop irrelevant historical context. The GRU gating controls temporal information flow, deciding how much past state versus new observations to retain at each timestep. This mechanism ensures robust gradient flow across long sequences.

From a weather perspective, these improvements let the model: automatically identify and weight the most relevant neighboring stations based on current conditions, retain important long-term temporal patterns needed for bias correction, and learn spatial relationships at multiple scales from local to regional.

**STGNN2 (Enhanced TCN-GNN).**

The baseline TCN-GNN architecture uses an alternating spatiotemporal convolution approach, where each layer applies a temporal convolution followed by graph convolution, repeating this pattern across multiple layers. While computationally efficient, this design has four main problems for postprocessing:

1. **Equal weighting of neighbors**: same limitation of STGNN1
2. **Limited spatial reach**: Local $k$-hop message passing can't efficiently capture long-range patterns like pressure gradients between distant stations (e.g., Basel-Lugano pressure differences that indicate Föhn conditions).
3. **Same weight for all time scales**: Summing all skip connections equally and adding the final layer output. This treats all temporal scales the same, even though early layers capture hourly patterns while deeper layers capture multi-day trends.
4. **No explicit time horizon information**: The model has to learn how forecast uncertainty grows with lead time purely from data, even though we know this pattern from meteorology.

Our improved architecture adds four key features:

1. **Direction-aware spatial aggregation**: Given the good performances with STGNN1,we use again GATv2Conv [4] with multi-head attention. Despite that, the implementation is diffent; unlike STGNN1's per-timestep processing, GATv2Conv here refines the entire temporal sequence after TCN processing, operating on all timesteps simultaneously.
2. **Global spatial attention**: After the TCN-GNN blocks, we add a multi-head self-attention layer (4 heads) that looks at all stations at once: $\mathbf{H}_{\text{global}} = \text{MultiHeadAttn}(\mathbf{H}_{\text{deep}}, \mathbf{H}_{\text{deep}}, \mathbf{H}_{\text{deep}})$. This allows distant stations to exchange information directly without needing multiple hops, which is important for large-scale weather patterns.
3. **Smart ensemble correction**: We add a GRU-inspired gate (without recurrence) that blends deep features with raw input:

$$\mathbf{h}_{\text{final}} = \mathbf{g} \odot \mathbf{h}_{\text{refined}} + (1 - \mathbf{g}) \odot \mathbf{h}_{\text{raw}}$$

   where $\mathbf{g} = \sigma(\mathbf{W}_g \mathbf{h}_{\text{refined}})$.
   Unlike STGNN1's recurrent temporal GRU gates, this is a non-recurrent gate that operates on the aggregated spatial features, deciding how much to trust deep processing versus raw predictions. This lets the model keep well-calibrated ensemble predictions when they're already good, but apply corrections when there are systematic biases.
4. **Forecast horizon embeddings**: We add learnable embeddings $\mathbf{e}_\tau \in \mathbb{R}^H$ for each forecast lead time $\tau \in \{1, \ldots, 96\}$ to the initial encoding. This gives the model direct information about the forecast horizon so it can learn how uncertainty should grow with time.

These improvements directly address postprocessing challenges: GATv2Conv attention captures direction-dependent patterns neded for wind forecasting, global attention handles long-range dependencies like pressure gradients across Switzerland (e.g., the North Föhn in Ticino driven by pressure differences between Zurich and Lugano), the GRU-inspired gating prevents over-correction of already-calibrated forecasts, and horizon embeddings encode how forecast uncertainty naturally grows with lead time.

**STGNN3 (Temporal Graph MLP-Mixer)**

Standard spatiotemporal graph neural networks (STGNNs) often rely on localized message-passing or global attention mechanisms with quadratic $\mathcal{O}(N^2)$ complexity in the number of nodes. While effective, these approaches face limitations in capturing long-range spatial interactions efficiently and can suffer from over-smoothing in deep

architectures. Inspired by recent successes of MLP-based architectures in vision and graph learning [5], we propose **STGNN3**, an adaptation of the Temporal Graph MLP-Mixer (T-GMM) [6] tailored for high-resolution probabilistic meteorological postprocessing. Our model, processes the spatiotemporal graph at dual resolutions to capture both local dynamics and global patterns, incorporating two key modifications for our domain: 1) a structured patch encoder and 2) a probabilistic output head for uncertainty quantification.

The core of STGNN3 is a dual-branch architecture that operates on two complementary views of the input spatiotemporal graph

1. **Node Branch (Fine-Grained):** Operates on the full graph ($N$ nodes). A Spatiotemporal Mixer (MLP mixer) captures dependencies across Time ($T$) and Space ($N$). This branch preserves high-frequency local details (e.g., specific station variance).
2. **Patch Branch (Coarse-Grained):** Operates on $P$ patches. By coarsening the graph, the Spatiotemporal Mixer in this branch can learn long-range spatial correlations with reduced computational complexity for global attention).

The outputs of both branches are fused via a learned weighted sum, allowing the model to adaptively balance local and global information.

**Patch Encoder (GINE).** The original T-GMM uses simple mean pooling to aggregate node features within a patch. This discards potentially informative local topology and edge attributes (e.g., inter-station distances). We instead employ a 2-layer Graph Isomorphism Network with edge attributes to encode the sub-graph within each patch $p$:

$$h_v^{(l+1)} = \mathrm{MLP}^{(l)}[t]\Big((1+\epsilon^{(l)})\cdot h_v^{(l)} + \sum_{u\in\mathcal{N}(v)} \mathrm{ReLU}\big(h_u^{(l)} + e_{uv}\big)\Big) \tag{1}$$

where $e_{uv}$ represents edge attributes (distance). This ensures that the patch representation $x_p = \mathrm{MeanPool}(\{h_v | v \in \mathcal{V}_p\})$ encodes the internal geometric structure of the patch, rather than just being a simple average of raw node features.

**Spatiotemporal MLP-Mixer.** The mixing block in each branch follows a factorized design that applies MLPs sequentially along the time ($T$), space ($N$ or $P$), and feature ($F$) axes. For an input tensor $X \in \mathbb{R}^{B\times T\times S\times F}$ (where $S$ is $N$ or $P$), the operations are:

$$U = X + W_{\text{space}} \cdot \sigma\left(W'_{\text{space}} \cdot \mathrm{LN}(X)_{(S,F)}\right) \quad \text{(Spatial Mixing)} \tag{2}$$

$$V = U + W_{\text{feat}} \cdot \sigma\left(W'_{\text{feat}} \cdot \mathrm{LN}(U)_{(T,S)}\right) \quad \text{(Feature Mixing)} \tag{3}$$

$$Y = V + W_{\text{time}} \cdot \sigma\left(W'_{\text{time}} \cdot \mathrm{LN}(V)_{(T,F)}\right) \quad \text{(Temporal Mixing)} \tag{4}$$

Here, $\mathrm{LN}(\cdot)$ denotes LayerNorm, $\sigma$ is the GELU activation, and $\cdot_{(a,b)}$ indicates rearrangement to apply the linear projection along axes $a$ and $b$ (with other dimensions flattened). Crucially, this *factorized* design avoids a full $\mathcal{O}(T\cdot S\cdot F)$ parameterization. Instead, the MLPs operate independently on each axis, yielding parameter counts proportional to $T+S+F$ and compute complexity linear in each dimension.

**Probabilistic Output Head.** For uncertainty quantification, we replace the standard deterministic readout with a probabilistic head that outputs parameters of a Log-Normal distribution. This choice aligns with the non-negative, skewed nature of wind speed while providing a closed-form continuous ranked probability score (CRPS) for efficient training. The head consists of a two-layer MLP that maps the fused node-level features to location $\mu_i$ and scale $\sigma_i$ parameters for each node $i$ and lead time. The model is trained end-to-end by minimizing the CRPS between the predicted Log-Normal distribution and the target observation.

Addressing these TGMM (STGNN3) provides a computationally efficient yet expressive architecture for spatiotemporal postprocessing

### 4.3 Hyperparameter Tuning

For all models (Model0, STGNN1, STGNN2, TGMM), we employed an automated hyperparameter optimization framework using Optuna with Tree-structured Parzen Estimator (TPE) sampling and median pruning. The tuning process optimized four key hyperparameters across a predefined search space:

- **Learning rate**: log-uniform distribution in $[10^{-5}, 10^{-3}]$
- **Hidden channels**: categorical selection from $\{32, 64, 128\}$
- **Number of layers**: categorical selection from $\{1, 2, 3, 4\}$
- **Dropout probability**: uniform distribution in $[0.1, 0.5]$

Each tuning run targeted 15 successful trials, with early stopping via MedianPruner (5 startup trials, 5 warmup steps) to eliminate underperforming configurations. Training was conducted for 8 epochs per trial on the validation set. To handle memory constraints, the framework automatically adapted batch sizes from 64 to smaller values when out-of-memory (OOM) errors occurred, ensuring robust convergence across different model architectures.

**TGMM:** for our proposed TGMM architecture, we conducted a focused grid search on the validation set:

- Architecture: Patch Size $P = 80$ (METIS), Hidden dimensions $H_{\text{node}} = 64$, $H_{\text{patch}} = 128$, Depth $L = 2$ mixer layers
- Training: AdamW optimizer ($lr = 10^{-3}$), 50 epochs, Early stopping (patience=10)

### 4.4 Experimental setup

Training uses:

- GPU acceleration: NVIDIA RTX 5080 and RTX 3070
- Loss: CRPS with masked NaNs
- Compute Time: an overall estimation for all models would be around 1 minute per epoch

## 5 Results

We evaluate MAE and CRPS on the test set for the raw NWP ensemble, the non-graph baseline (Model0), the default STGNN2 baseline, and three enhanced architectures: our tuned STGNN1, our tuned STGNN2, and STGNN3 (TGMM).

**Evaluation protocol.** All reported results are aggregated over 5 independent random seeds to ensure statistical robustness. For each model, we report the mean and standard deviation of the metrics across seeds. Final training used the optimized hyperparameters, extended training (up to 50 epochs), and early stopping based on validation CRPS.

**Table 1.** Test CRPS and MAE (overall across all stations and lead times). Values are reported as mean ± standard deviation across seeds.

| Methods | CRPS ↓ | MAE ↓ |
|---|---|---|
| NWP ensemble | $0.9808 \pm 0.0000$ | $1.2060 \pm 0.0000$ |
| Model0 | $0.6715 \pm 0.0011$ | $0.9802 \pm 0.0037$ |
| STGNN1 (ours) | $0.6422 \pm 0.0046$ | $0.9456 \pm 0.0129$ |
| STGNN2 (default) | $0.6648 \pm 0.0031$ | $0.9776 \pm 0.0080$ |
| STGNN2 (ours) | $\mathbf{0.6209 \pm 0.0053}$ | $0.9077 \pm 0.0129$ |
| STGNN3 (TGMM) | $0.6243 \pm 0.0040$ | $\mathbf{0.9017 \pm 0.0061}$ |

Model0 already provides a strong baseline, substantially improving over the raw NWP ensemble in both CRPS and MAE. This indicates that learning a purely temporal postprocessing mapping from ensemble features captures a large fraction of the available skill.

Among the graph-based models, both tuned STGNN1 and STGNN2 further improve upon Model0. In particular, our modified STGNN2 achieves the best overall CRPS ($0.6209 \pm 0.0053$), while STGNN3 (TGMM) attains the lowest MAE ($0.9017 \pm 0.0061$), suggesting improved point forecast accuracy. The default STGNN2 configuration performs competitively but is consistently outperformed by its tuned counterpart, highlighting the importance of architectural and optimization choices.

At short lead times ($t = 1$ h), all tuned graph-based models yield significant improvements over NWP and Model0. For instance, STGNN2 (ours) reduces CRPS from $0.9804$ (NWP) to $0.5777$, while MAE decreases from $1.2060$ to $0.8505$. At longer lead times ($t = 96$ h), the gains are smaller but remain consistent across models (e.g. CRPS $0.6866$ vs. $0.9846$ for NWP), reflecting the increasing difficulty of the forecasting task.

Overall, these results indicate that incorporating spatial structure through carefully designed spatiotemporal graph models yields consistent improvements over strong non-graph baselines, while the choice of architecture plays a critical role in determining the magnitude of the gains.

### 5.1 Qualitative analysis of predictive distributions

To complement the aggregated CRPS and MAE scores, Figures 1–6 show example forecasts over four stations and all lead times. For each panel we display the NWP ensemble mean (green), the predicted median and central quantiles (shaded blue areas), and the observed wind speed (magenta).

Note that the epochs differ between models (e.g. epoch 70 for Model0 vs. 30–50 for the more complex STGNNs). This reflects the fact that deeper graph-based architectures

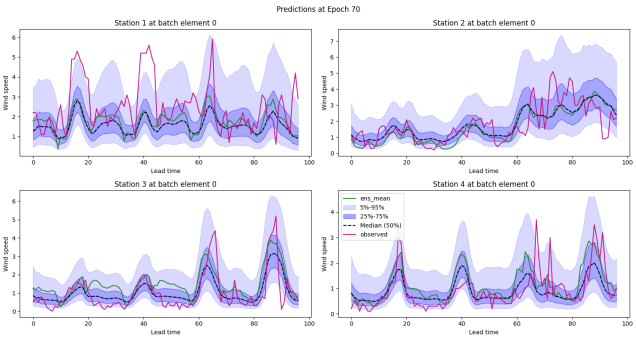


**Figure 1.** Model0 at epoch 70. Shaded areas denote 5–95% and 25–75% predictive intervals, the dashed line is the predictive median, the green line is the NWP ensemble mean, and the magenta line the observed wind speed.

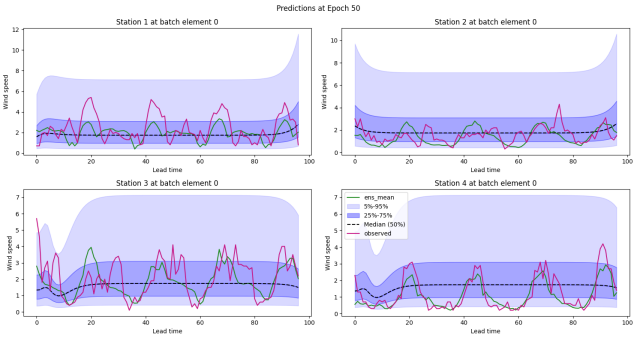


**Figure 2.** BiDirectional STGNN (STGNN1 default) at epoch 50.

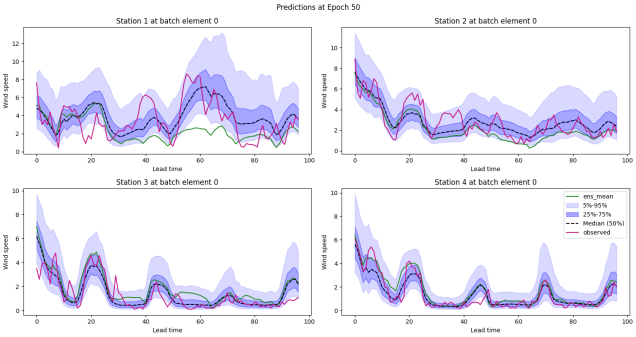


**Figure 3.** Our tuned STGNN1 at epoch 50.

are more expensive per epoch and reach their best validation CRPS earlier in training.

Note: the baseline STGNN1 (bidirectional rnn) was dropped early because, as pretty evident in figure 2 basically no improvement is made even with 50 epochs.

### 5.2 Rank histograms (Talagrand diagrams)

Calibration is further assessed using rank histograms constructed from 20 samples drawn from the predictive distributions, resulting in 21 possible ranks for each observation. Figure 7 shows the Talagrand diagrams at lead time $t = 1$ h for all models. The red dashed line indicates the expected relative frequency under perfect calibration (uniform histogram).

Rank histograms are a standard diagnostic tool in ensemble weather forecasting, as they provide direct insight into the statistical consistency between predictive

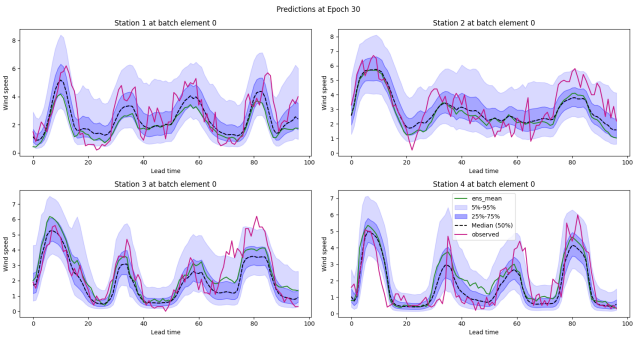


**Figure 4.** TCN-GNN (STGNN2 default) at epoch 30.

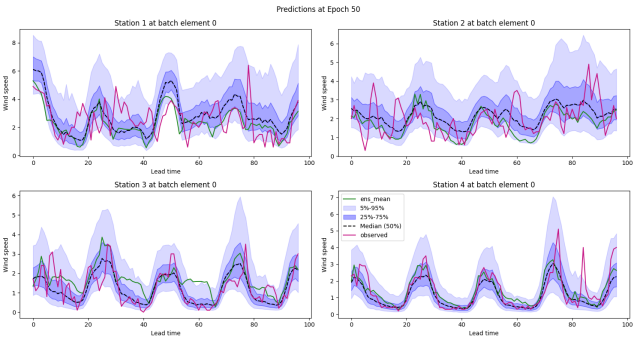


**Figure 5.** Our stgnn2 at epoch 50.

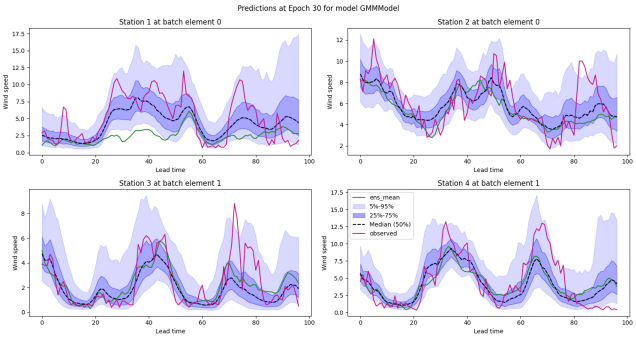


**Figure 6.** TGMM (STGNN3) at epoch 30.

distributions and observations. Unlike aggregate scores such as CRPS or MAE, which combine effects of bias, dispersion, and sharpness into a single number, rank histograms allow these aspects to be disentangled. In particular, deviations from uniformity can reveal under- or overdispersion (U-shaped or inverted U-shaped histograms) as well as systematic biases (skewed histograms), making them especially valuable for assessing probabilistic calibration in operational forecasting contexts.

For brevity, we only report the diagrams for lead time $t = 1\,\text{h}$. The histograms at longer lead times ($t = 24, 48, 96\,\text{h}$) exhibit very similar patterns and are therefore omitted to save space.

A consistent problem across all models is an elevated frequency in the first rank and a slightly reduced frequency in the last rank, while the intermediate bins remain close to the uniform reference. This pattern suggests a mild tendency of the predictive distributions to assign insufficient probability mass to very low wind-speed observations. In our opinion one possible contributing factor is the use of a

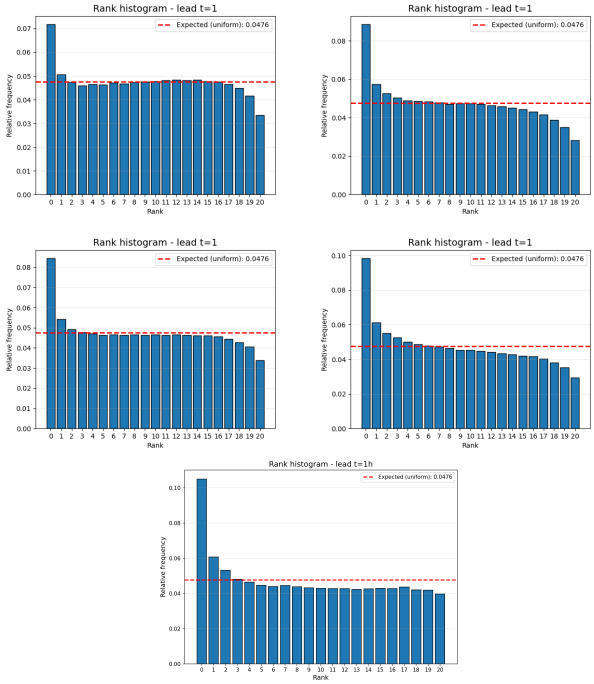


**Figure 7.** Rank histograms (Talagrand diagrams) at lead time $t = 1\,\text{h}$ using 20 samples (21 ranks). Top-left: Model0. Top-right: enhanced STGNN1. Center-left: STGNN2 default. Center-right: enhanced STGNN2 Bottom: TGMM (STGNN3)

LogNormal predictive distribution, which enforces strictly positive support and may struggle to accurately represent near-zero wind regimes. In such cases, observations close to zero are more likely to fall below all sampled ensemble members, which increases the frequency of the lowest rank. On the other hand, very high wind-speed observations are rarer and are generally better represented by the upper tail of the predictive distribution, resulting in a slightly lower occupancy of the highest rank.

Overall, the approximately uniform distribution of the intermediate ranks indicates good calibration of the predictive distributions. Whereas the remaining imbalance at the extremes suggests that alternative distributional models could further improve calibration, particularly in low-wind regimes.

### 5.3 Graph Refinement (Bonus 1: Terrain-Aware kNN)

To test whether a more geographically realistic graph improves postprocessing, we replaced the baseline distance-only kNN graph (k=5, $\theta$ from global distance std, threshold=0.6) with a terrain-aware variant. This second graph combines: (i) distance bandwidth from the median nearest-neighbor spacing ($\theta_{\text{strategy}} = \text{nn\_median}$, $\theta_{\text{scale}} = 0.7$); (ii) sparsity/robustness controls ($k = 8$, threshold=0.35, max_distance_km=120, mutual=true); and (iii) an orography penalty using ridge distance (orography_var = `terrain:distance_to_alpine_ridge`, scale=15,000, $\alpha = 1.0$) to down-weight cross-ridge edges.

**Results (STGNN2).**

The CRPS improvement is small but consistent, suggesting that penalizing cross-ridge links modestly helps calibration.

**Table 2.** Impact of enhanced graph construction on STGNN2 performance. Results aggregated over 2 seeds with standard deviations.

| Configuration | CRPS | MAE |
|---|---|---|
| Baseline | $0.6209 \pm 0.0053$ | $0.9077 \pm 0.0129$ |
| Enhanced | $0.6173 \pm 0.0044$ | $0.9002 \pm 0.0105$ |

Further tuning of the ridge penalty (scale/$\alpha$) and the distance cap may yield larger gains.

## 6 Discussion and conclusion

Our experiments show that even a relatively simple non-graph temporal model (Model0) already provides a strong postprocessing baseline. It clearly improves over the raw NWP ensemble in terms of both CRPS and MAE, suggesting that much of the gain comes from learning a good temporal mapping from ensemble features to observations.

In contrast, the bidirectional STGNN (STGNN1) performs poorly in our setting: the default configuration is consistently worse than both NWP and Model0, and our enhanced version only partially mitigates this issue. This indicates that the current architecture and/or training strategy for STGNN1 is not well suited to the task, and that simply adding graph structure does not automatically translate into better skill.

The TCN-GNN (STGNN2) in its default configuration is more promising. It matches or exceeds Model0 and NWP in terms of CRPS and MAE, while also showing improved calibration in the rank histograms at short lead times. This suggests that combining temporal convolutions with graph convolutions is an effective way to exploit spatial dependencies between stations.

By abandoning the strict local connectivity assumptions of GAT/GCN (STGNN1/2) and resolving high-frequency temporal fluctuations and non-local spatial dependencies via Node & Global Patch-Mixing, the TGMM (STGNN3) overcomes the trade-offs inherent in purely local message passing, surpassing other model in efficiency and performance setting a benchmark.

Overall, we conclude that carefully designed spatiotemporal graph models can bring additional benefits over strong non-graph baselines, but that these benefits are architecture- and training-dependent. Future work should focus on simplifying and regularising the graph-based models (especially STGNN1), exploring alternative graph constructions, and systematically analysing model robustness across lead times and weather regimes.

## References


[1] Stephan Rasp and Sebastian Lerch. "Neural networks for postprocessing ensemble weather forecasts". In: *Monthly Weather Review* 146.11 (2018), pp. 3885–3900.

[2] Moritz Feik, Sebastian Lerch, and Jan Stühmer. "Graph neural networks and spatial information learning for post-processing ensemble weather forecasts". In: *arXiv preprint arXiv:2407.11050* (2024).

[3] Zonghan Wu et al. "Graph WaveNet for Deep Spatial-Temporal Graph Modeling". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 1907–1913. DOI: 10.24963/ijcai.2019/264. URL: https://doi.org/10.24963/ijcai.2019/264.

[4] Shaked Brody, Uri Alon, and Eran Yahav. "How Attentive are Graph Attention Networks?" In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=F72ximsx7C1.

[5] Xiaoxin He et al. "A generalization of vit/mlp-mixer to graphs". In: *International conference on machine learning*. PMLR. 2023, pp. 12724–12745.

[6] Muhammad Bilal and Luis Carretero Lopez. "Temporal Graph MLP Mixer for Spatio-Temporal Forecasting". In: *arXiv preprint arXiv:2501.10214* (2025).