Spatiotemporal Graph Neural Networks for Probabilistic Postprocessing of Wind Speed

Edoardo Leali Matteo Vitali Sidharth Padmanabhan

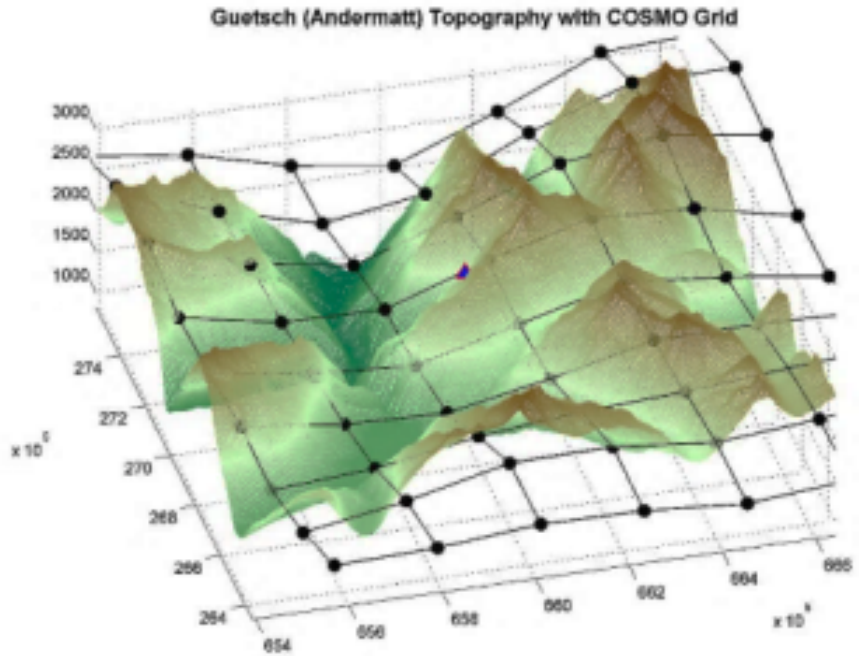
USI – MeteoSwiss

January 5, 2026

The Problem: NWP Systematic Biases

Challenges in NWP models:

- Terrain resolution
  - → Grid smoothing flattens mountains
- Precipitation biases
  - → "Drizzle" bias: too much light rain. Underestimate extreme events.
- Temperature biases
  - → Damped day-night temperature range



Our Task

Statistical postprocessing to correct these systematic biases

Key Distinction

We're **NOT forecasting** from scratch → We're **correcting** existing forecasts

Dataset & Graph Structure

MeteoSwiss Wind Speed Dataset

- **Train/Val:** Reforecasts (Feb 2020 – Sep 2023)
- **Test:** Operational forecasts (May 2024 – Jan 2025)
- 18 predictors
- 96-hour forecast horizon

Graph with the weather stations

Spatial Graph Construction:

- **Nodes:** Weather stations
- **Edges:** k-NN (k=5)
  - Haversine distance
  - Gaussian kernel weights
- **Fixed graph**

The Machine Learning Task

Problem Formulation:

Input: $\mathbf{X} \in \mathbb{R}^{T \times N \times F}$ spatiotemporal data

- $T = 96$ timesteps (lead times 1–96 hours)
- $N =$ number of stations (nodes in graph)
- $F = 18$ features:
  - 4 NWP ensemble features (mean, std, pressure gradients)
  - 4 temporal features (hour-of-day, day-of-year encodings)
  - 10 terrain features (elevation, roughness, etc.)

Output: Probabilistic distribution

$$y_{s,t} \sim \text{LogNormal}(\mu_{s,t}, \sigma_{s,t})$$

- Why LogNormal? Wind speeds are strictly positive and right-skewed

Learning Objective

Supervised Learning Framework:

- **Input:** $\mathbf{X} \in \mathbb{R}^{T \times N \times F}$ (predictor features)
- **Output:** $(\mu_{s,t}, \sigma_{s,t})$ parameters for LogNormal distribution
- **Target:** $y_{s,t}$ (observed wind speed at station s, lead time t)
- **Loss:** CRPS (Continuous Ranked Probability Score)

CRPS Properties:

- Proper scoring rule: jointly optimizes sharpness & calibration
- Masked implementation: excludes missing observations (sensor failures)

$$\text{CRPS}(F, y) = \int_{-\infty}^{\infty} \left[F(t) - H(t - y)\right]^2 dt$$

where F is the predictive CDF, y is the observation, and $H(t - y)$ is the Heaviside function (0 when $t < y$, 1 otherwise)

Complete Experimental Pipeline

1. Model Implementation

1. **Model0:** Pure temporal (no graph) – *baseline*
2. **STGNN1:** Bidirectional RNN where the recurrent cell is a MP cell
3. **STGNN2:** TCN-GNN
4. **STGNN3:** Temporal Graph MLP-Mixer (our model)

2. Hyperparameter Tuning

- Optuna framework with TPE sampling
- Search space: learning rate, hidden channels, layers, dropout

3. Multi-seed Evaluation

- 5 independent random seeds for statistical robustness
- Report mean $\pm$ std across seeds

4. Calibration Analysis

- Rank histograms (Talagrand diagrams)
- Assess predictive distribution calibration

Metrics: CRPS (probabilistic) + MAE (deterministic)

Model Implementations

Baseline - Enhanced Architectures - Our Model

Recap: Temporal Convolutional Networks

A Temporal Convolutional Network processes sequential data using **dilated 1D convolutions** instead of recurrent units.

Key Innovation:

- **Exponential dilation:** $d = 2^l$ for layer l
  - Layer 1: dilation $= 1$ (hourly patterns)
  - Layer 2: dilation $= 2$
  - Layer 3: dilation $= 4$
  - Layer 6: dilation $= 32$ (multi-day)

Advantages:

- Parallelizable (unlike RNNs)
- No vanishing gradients
- Multi-scale temporal patterns
- For postprocessing: non-causal convolutions
  - $\rightarrow$ Look forward & backward in forecast window

Residual Connections:

- output $=$ Layer(input) $+$ input
- Network learns *corrections*, not full transformations
- Provides gradient highway $\rightarrow$ stable deep training

Model0: Non-Graph Temporal Baseline

Architecture: Temporal Convolutional Network (No Graph)

Forward Pass:

1. **Encoder:** Linear projection $\mathbb{R}^{18} \to \mathbb{R}^{128}$
   - Projects input features to hidden dimension
2. **Rearrange for TCN:** $[B, T, N, C] \to [(B \times N), C, T]$
   - Treats each station as **independent sequence**
   - N stations processed in parallel, no spatial info shared
3. **Stacked TCN Layers:** 2 layers with dilations $[1, 2]$
   - Each layer returns: $(x_{\text{residual}}, x_{\text{skip}})$
   - Collects multi-scale temporal features
4. **Multi-Scale Aggregation:** $\sum_{l=1}^{L} \text{skip}_l + x_{\text{final}}$
   - Combines all temporal scales into final representation
5. **Rearrange Back:** $[(B \times N), C, T] \to [B, T, N, C]$
6. **Probabilistic Output:** Projects to LogNormal(μ, σ) parameters

Model0: Performance

Results:

- **CRPS: 0.67** vs. NWP 0.98
- Strong improvement over raw ensemble
- *Most gain comes from learning temporal correction?*

Key Insight

This purely temporal baseline is **surprisingly good!**
$\rightarrow$ Graph structure needs to prove its worth

Challenge for graph models: Beat this strong baseline by capturing spatial dependencies

STGNN1: What the Baseline Does

Architecture: Bidirectional Recurrent GNN

Approach:

1. At each timestep t: aggregate spatial neighbors
2. Then: update temporal state with RNN
3. Process both directions:
   - Forward: $t = 1 \rightarrow 96$
   - Backward: $t = 96 \rightarrow 1$

Components:

- Simple message passing (GatedGraphNetwork)
- Uniform neighbor weighting
- Additive residuals: $\mathbf{h}_t = \mathbf{h}_{t-1} + \Delta\mathbf{h}_t$

STGNN1: Problems & Solutions

Limitations:

- **Vanishing Gradients**
  - Additive residuals: $\mathbf{h}_t = \mathbf{h}_{t-1} + \Delta\mathbf{h}_t$
  - 96 timesteps $\rightarrow$ exponential decay
- **Equal Neighbor Weighting**
  - All neighbors treated equally
  - Wind is directional – upwind matters more!
- **No Hierarchical Learning**
  - Flat spatial processing
  - Early vs. deep layers should learn different scales

Our Improvements:

- **GRU Cells**
  - Gated updates: $\mathbf{h}_t = (1 - z_t) \odot \mathbf{h}_{t-1} + z_t \odot \tilde{\mathbf{h}}_t$
  - Robust gradient flow
- **GATv2 Attention**
  - Dynamic neighbor weighting
  - 2 attention heads
- **Layer-by-Layer Processing**
  - Deep spatial learning
  - Hierarchical pattern extraction

STGNN1: The Gradient Problem & Solution

Before:

Baseline used additive residuals:

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \Delta\mathbf{h}_t$$

Result: Exponential gradient decay during backpropagation $\rightarrow$ model cannot learn.

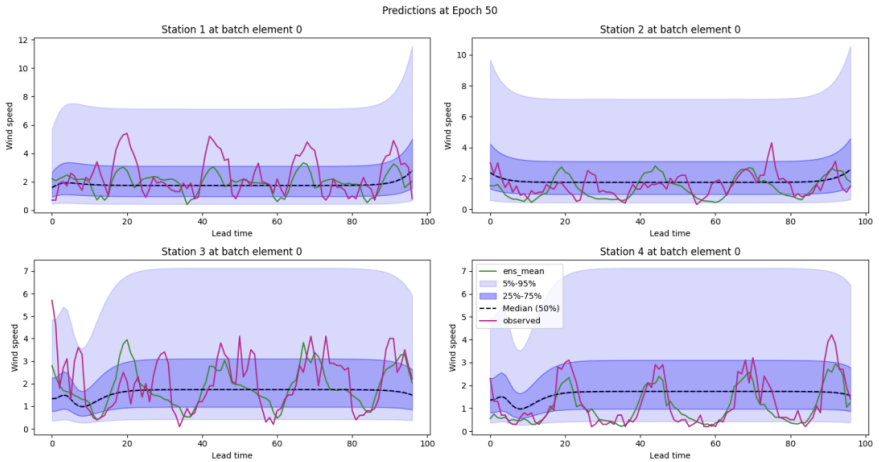


Figure 2. BiDirectional STGNN (STGNN1 default) at epoch 50.

After:

Replace with GRU cells:

$$\mathbf{h}_t = (1 - z_t) \odot \mathbf{h}_{t-1} + z_t \odot \tilde{\mathbf{h}}_t$$

Gating creates **multiplicative skip connections** $\rightarrow$ gradients flow without decay. Model finally learns temporal patterns

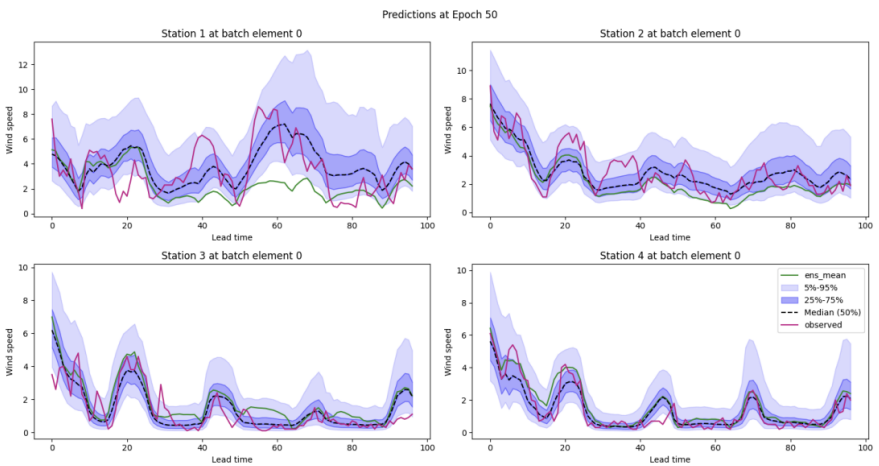


Figure 3. Our tuned STGNN1 at epoch 50.

STGNN1: Weather Perspective

How Our Improvements Address Weather Postprocessing:

- **GRU Cells**
  - Track sustained weather events (Föhn episodes, persistent winds)
  - Maintain temporal context across 96-hour forecast window
- **GATv2 Attention**
  - Identifies relevant upwind stations for wind correction: upwind stations weighted more heavily
  - Captures local effects: valley channeling, terrain blocking
- **Layer-by-Layer Spatial Processing**
  - Early layers: local station correlations (valley networks)
  - Deep layers: regional patterns (Alpine pressure systems)

Result: CRPS 0.64 – modest improvement over Model0, proves attention helps

STGNN2: What the Baseline Does

Architecture: Hybrid TCN-GNN

Approach:

- Stack L layers of: [TCN $\rightarrow$ GNN]

Each layer:

1. **TCN:** Dilated temporal convolution
   - Captures multi-scale temporal patterns
   - Dilation $= 2^l$ for layer l
2. **GNN:** Message passing for spatial aggregation
   - Simple GatedGraphNetwork
   - Uniform neighbor weighting
3. **Skip:** Store for final multi-scale combination

Final output: sum(all skips) + final layer

STGNN2: Limitations & Solutions

Limitations:

- **Equal Neighbor Weighting**
  - All neighbors treated equally
- **Limited Spatial Reach**
  - Can't capture long-range patterns
  - Requires many hops through graph

Our Solutions:

- **Direction-Aware GATv2**
  - Dynamic neighbor weighting
- **Global Spatial Attention**
  - Direct long-range communication
  - No multi-hop bottleneck

Additional Feature: Smart Ensemble Correction

- Creates two parallel paths:
  - *Shallow*: Simple linear projection
  - *Deep*: Full TCN-GNN-Attention
- Learned gate decides when to use which:

$$\mathbf{h} = g \odot \mathbf{h}_{\text{deep}} + (1 - g) \odot \mathbf{h}_{\text{shallow}}$$

- Trust ensemble when good ($g \approx 0$), correct when biased ($g \approx 1$)

STGNN2: Weather Perspective

How Our Improvements Address Weather Postprocessing:

- **GATv2 Attention**
  - Identifies relevant upwind stations for wind correction: upwind stations weighted more heavily
  - Captures local effects: valley channeling, terrain blocking
- **Global Attention**
  - Long-range dependencies (pressure gradients)
  - Example: North Föhn in Ticino driven by Zürich-Lugano pressure difference
- **GRU-inspired Gating**
  - Trusts good ensemble predictions
  - Prevents over-correction

STGNN2: Results

Method	CRPS ↓	MAE ↓
NWP ensemble	0.98	1.21
Model0 (no graph)	0.67	0.98
STGNN1 (ours)	0.64	0.95
STGNN2 (ours)	**0.62**	**0.91**

Key Findings:

- **Best CRPS: 0.62** – significant improvement at *all* lead times
- Outperforms both Model0 and STGNN1
- Consistent gains from short (t=1h) to long (t=96h) lead times

STGNN3 (TGMM): From ViT to Graph MLP-Mixer Architectural Change

The Journey of Modern Deep Learning:

1. **Vision Transformers (ViT, 2020)**
   - Treated images as sequences of patches (tokens)
   - Used Self-Attention for global context
   - Problem: $O(N^2)$ complexity
2. **MLP-Mixer (Google, 2021)**
   - Key insight: Attention isn't necessary!
   - Simple MLPs + transposition = global mixing
   - Advantage: $O(N)$ complexity
3. **Graph MLP-Mixer (ICML 2023)**
   - Adapts "Patch-and-Mix" to **static graphs**
   - Uses METIS partitioning to create graph "patches"
4. **Temporal Graph MLP-Mixer (2025)**
   - Extends Graph MLP-Mixer to **dynamic spatiotemporal** data
   - Adds **temporal mixing** dimension for time-series

STGNN3 (TGMM): Why MLP-Mixer for Graphs?

Problems with Standard GNNs:

- **Over-squashing**
  - → Long-range info compressed into small vectors
  - → "Traffic jam" for distant signals
- **Limited Expressivity**
  - → Bounded by 1-WL test
  - → Can't distinguish certain graph structures
- **Local Receptive Field**
  - → Need L layers for L-hop reach
  - → Slow information propagation

MLP-Mixer Solution:

- **Global Mixing**
  - → Every patch talks to every patch
  - → "Teleportation" bypasses traffic jam
- **High Expressivity**
  - → Exceeds 1-WL test limit
  - → Distinguishes complex structures
- **Linear Complexity**
  - → $O(N)$ vs. $O(N^2)$ for Transformers
  - → Scalable to large graphs

STGNN3 (TGMM): Creating "Patches" from Graphs

Challenge: Images have grids; graphs have irregular topology.

METIS Partitioning

- Split graph into k clusters
- Minimize inter-cluster edge cuts
- Result: meaningful communities

1-Hop Overlap ("Halo")

- Add 1-hop neighbors to each patch
- Preserves boundary information
- Analogous to image border padding

Weather Station Graphs

- Patches = regional clusters
- Capture regional weather dynamics

Key Insight

Graph patching converts irregular graphs into token sequences, enabling Mixer-style architectures.

STGNN3 (TGMM): The 3D Temporal Mixer

Fundamental Architecture Shift

- **Original T-GMM:** $[B, T_{\text{hist}}, N, F] \rightarrow t{+}1$ (autoregressive)
- **Our Model:** $[B, T_{\text{lead}}, S, F] \rightarrow$ **all 97h at once**

3D Mixer Axes

Spatial Mixing

- MLP across stations
- Regional patterns

Feature Mixing

- MLP across variables
- Physical interactions

Lead-Time Mixing

- MLP across horizon
- Temporal evolution

Key Innovation: Parallel Multi-Step Forecasting

Traditional autoregressive approaches predict sequentially (slow, error-prone). Our non-autoregressive method predicts the complete 97h forecast trajectory in **one forward pass**, enabling:

- Faster inference
- No error compounding (each hour predicted independently)
- Consistent trajectories (coherent forecast evolution)

STGNN3 (TGMM): Why Non-Autoregressive

- Weather prediction is an **initial-value problem**
- Model learns $X(t_0) \mapsto \{X(t_0+1h), \ldots, X(t_0+97h)\}$ **in one step**
- Conceptually aligned with Numerical Weather Prediction (NWP)

Operational Perspective

Autoregressive Limitations

- Error accumulation
- Inconsistent temporal evolution
- Sequential inference (slow)
- Exposure bias during training

TGMM Advantages

- **Parallel forecasts**: all lead times at once
- **Coherent trajectories**: physically consistent
- **Fast inference**: operationally viable
- **Joint uncertainty**: lead-time covariance

Technical Impact

This architecture enables **operational-grade forecasting** that is fast, consistent, and probabilistic meeting the strict requirements of energy grid management and weather services while maintaining the theoretical advantages of MLP-Mixer efficiency.

STGNN3 (TGMM): Dual-Stream Processing

Original Papers: Single processing stream (patch or node)
Our Adaptation: **Two parallel streams**

Patch Stream

- Input: METIS subgraphs
- View: Coarse-grained / Regional
- Captures: Large-scale weather patterns

Node Stream

- Input: Individual stations
- View: Fine-grained / Local
- Captures: Microclimates
- Example: "Station X is in a valley"

Readout: Combines both streams:

$$\mathbf{h}_{\text{final}} = \text{MLP}\left([\mathbf{h}_{\text{patch}} \,\|\, \mathbf{h}_{\text{node}}]\right)$$

Why Both?

Regional context (patches) + local precision (nodes) = best of both worlds

STGNN3 (TGMM): Complete Architecture

Full Pipeline:

1. **Input:** $\mathbf{X} \in \mathbb{R}^{B \times L \times N \times F}$ + Validity Mask
2. **METIS Partitioning:** Graph $\rightarrow$ 80–100 patches with 1-hop overlap
3. **GNN Encoder:** GINEConv on patches (3 layers, local structure)
4. **Dual 3D Mixer:**
   - Patch Mixer (2 layers): Global regional patterns
   - Node Mixer (2 layers): Local station details
5. **Readout:** Concatenate + MLP $\rightarrow (\mu, \sigma)$ for each station/lead time
6. **Output:** LogNormal(μ, σ) for 97 lead times $\times$ N stations

Key Properties:

- Linear complexity $O(N)$ – scalable to 1000+ stations
- Global receptive field – captures long-range dependencies
- Probabilistic output

STGNN3 (TGMM): Results

Method	CRPS ↓	MAE ↓
NWP ensemble	0.9808 ± 0.0000	1.2060 ± 0.0000
Model0 (no graph)	0.6715 ± 0.0011	0.9802 ± 0.0037
STGNN1 (ours)	0.6422 ± 0.0046	0.9456 ± 0.0129
STGNN2 (ours)	0.6209 ± 0.0053	0.9077 ± 0.0129
STGNN3 (TGMM)	0.6243 ± 0.0040	0.9017 ± 0.0061

Key Findings:

- Strong MAE performance — highly accurate point predictions
- Competitive CRPS, similar to STGNN2
- Low variance across seeds (± 0.0040)
- Atleast 10x more efficient than comparable architectures

Trade-off

STGNN2: slightly better probabilistic skill (CRPS). TGMM: excellent deterministic accuracy + stability + efficiency.

STGNN3 (TGMM): When to Use?

- Large-scale graphs
- Strict runtime constraints
- Stable and consistent predictions
- Long-range spatial dependencies

The Big Picture

TGMM reflects a broader shift toward **global context with simple MLPs**, avoiding costly attention.
ViT $\rightarrow$ MLP-Mixer $\rightarrow$ Graph MLP-Mixer $\rightarrow$ Temporal Graph MLP-Mixer

Training & Evaluation

Hyperparameter Tuning & Calibration Analysis

Evaluation Protocol

Goal: Compare probabilistic and deterministic performance across models.

Metrics:

- CRPS (probabilistic skill, proper scoring rule)
- MAE (deterministic accuracy)

Aggregation:

- Overall metrics computed across **all stations** and **all lead times**:
- Key lead times: $t = 1, 24, 48, 96$ hours

Robustness:

- 5 independent random seeds
- Report mean $\pm$ std across seeds

Hyperparameter Tuning

Why tuning?

- Deep models are sensitive to hyperparameters
- Manual tuning is inefficient and not reproducible
- We tune **all architectures fairly** under the same protocol

Goal

Automatically select the best configuration for each model using **validation CRPS**, before final multi-seed testing.

Hyperparameter Tuning: How

Framework: Optuna

- **TPE sampler**: intelligent search (not grid / random)
- **MedianPruner**: stops bad trials early
- Short trials (8 epochs) for fast comparison

Why this works

TPE focuses on promising regions of the search space, while MedianPruner avoids wasting time on clearly bad configurations.

Tuning Details & Robustness

Hyperparameters tuned:

- Learning rate: $[10^{-5}, 10^{-3}]$ (log-uniform)
- Hidden channels: $\{32, 64, 128\}$
- Number of layers: $\{1, 2, 3, 4\}$
- Dropout: $[0.1, 0.5]$

Practical considerations:

- 15 successful trials per model
- Automatic batch-size backoff on OOM
- Best config reused for final 5-seed experiments

Final Test Results

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

Takeaway

Model0 is already strong → carefully designed spatiotemporal graph models yields consistent improvements.
Best overall **CRPS**: STGNN2 (ours). Best overall **MAE**: TGMM.

Lead-Time Breakdown

Why per-lead evaluation matters: forecast uncertainty increases with horizon.

Model	1h	24h	48h	96h
Model0	0.644	0.649	0.668	0.745
STGNN1 (ours)	0.608	0.618	0.638	0.702
STGNN2 (default)	0.628	0.644	0.662	0.724
STGNN2 (ours)	**0.578**	**0.594**	**0.618**	**0.687**
STGNN3 (TGMM)	0.593	0.603	0.628	0.692

Key observation

All graph-based models improve over the non-graph baseline. Our enhanced STGNN2 achieves the lowest CRPS at all lead times, with the largest gains at short horizons.

Calibration via Rank Histograms

Goal: verify statistical consistency between predicted distributions and observations.

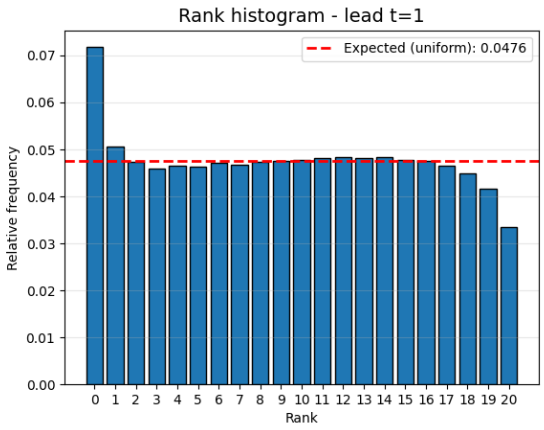
Procedure (per lead time):

1. For each case: sample **20** values
2. Insert the observation into the sorted sample set
3. Record its rank $\in \{1, \ldots, 21\}$ (**21 bins**)

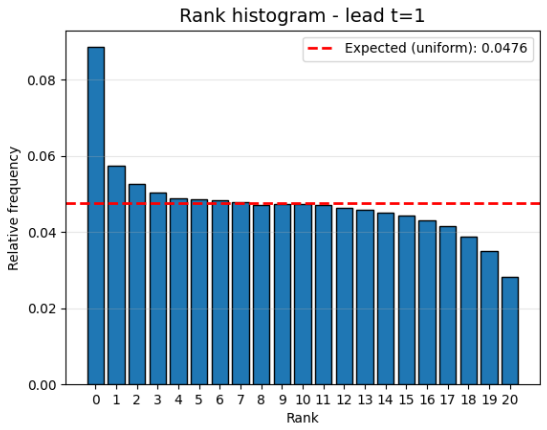
Interpretation:

- Uniform histogram $\rightarrow$ calibrated uncertainty
- U-shape $\rightarrow$ under-dispersed (too narrow)
- Inverted U $\rightarrow$ over-dispersed (too wide)
- Skew $\rightarrow$ bias

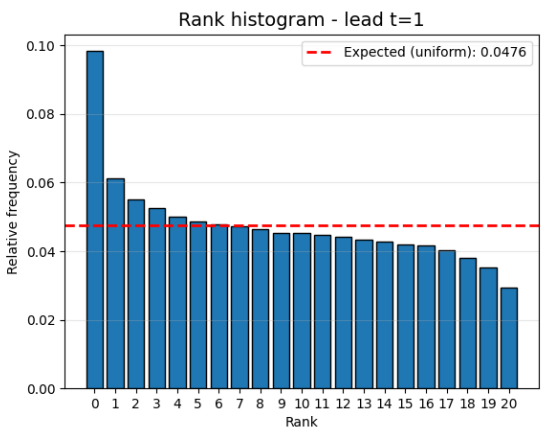
Talagrand Diagrams (Lead Time $t = 1$h)



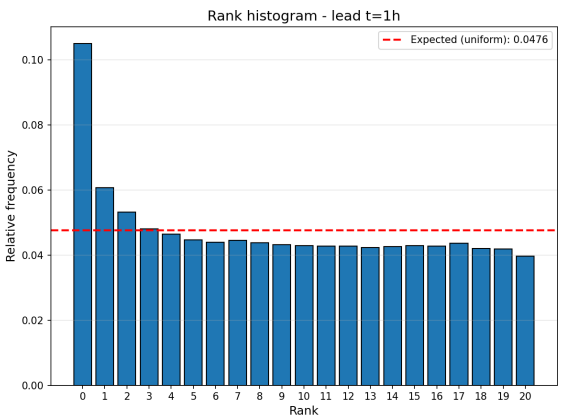
Model0



STGNN1



STGNN2



STGNN3

Why the First Rank is High

Observed pattern: more observations fall below all 20 samples than expected.

Interpretation (intuitive)

When the true wind speed is **very close to zero**, a LogNormal model can struggle to place enough probability mass near zero (it is strictly positive and often right-skewed). So the sampled ensemble values tend to be slightly above the observation, causing the observation to fall into the **lowest rank** more often.

Leads considered: $t = 1, 24, 48, 96$ hours (we show $t = 1$h for brevity; longer leads show similar qualitative patterns)

Bonus 1: Graph Refinement (Terrain-Aware kNN)

Motivation: distance-only kNN can create unrealistic links across terrain barriers.

Baseline graph:

- kNN by Haversine distance ($k = 5$)
- Gaussian kernel weights, global θ, threshold $= 0.6$

Terrain-aware variant:

- Bandwidth from median nearest-neighbor spacing: $\theta_{\text{scale}} = 0.7$
- Sparsity/robustness: $k = 8$, threshold $= 0.35$, max distance $= 120$ km, mutual edges
- Orography penalty using **distance to Alpine ridge** to down-weight cross-ridge links

Bonus 1: Results (Graph Ablation on STGNN2)

STGNN2 configuration	**CRPS**	**MAE**
Baseline distance graph	0.6209 ± 0.0053	0.9077 ± 0.0129
Terrain-aware graph	0.6173 ± 0.0044	0.9002 ± 0.0105

Takeaway

The gain is small but consistent: terrain-aware connectivity slightly improves both CRPS and MAE, suggesting that reducing unrealistic cross-ridge edges can help performance.