PeakWeather Temperature Forecasting with Spatiotemporal Graph Neural Networks

GDL 2025

Filippo Wang, Roberts Kalvitis, Riccardo Sacco
{filippo.wang, roberts.kalvitis, riccardo.sacco}@usi.ch

Abstract

We investigate the effectiveness of spatiotemporal graph neural networks (STGNNs) for temperature forecasting using the PeakWeather dataset from Switzerland. We implement and evaluate four deep learning architectures: a Temporal Convolutional Network (TCN) as a graph-free baseline, an RNN with learnable node embeddings, a time-then-space STGNN with graph convolutions, and an attention-based STGNN with learned graph structure (STGNN-Att). A spatial graph is constructed based on geographical distance using a Gaussian kernel with K-nearest neighbor filtering. All models are trained with the Energy Score (CRPS) loss function. We compare our learned models against simple baselines (Naive, Moving Average) and the ICON-CH1-EPS numerical weather prediction model. Our experiments evaluate forecast accuracy at multiple horizons (1h to 24h) using both MAE and Energy Score metrics. Results show that proper graph connectivity is critical for spatial models to outperform temporal-only baselines, and that the Energy Score loss improves probabilistic calibration.

1 Introduction

Accurate temperature forecasting is essential for energy management, agriculture, transportation, and public safety. While numerical weather prediction (NWP) models have been the standard approach, recent advances in deep learning offer promising alternatives that can learn directly from historical observations. This project investigates whether spatiotemporal graph neural networks (STGNNs) can enhance temperature forecast accuracy by explicitly modeling spatial dependencies among weather stations.

We work with the PeakWeather dataset [1], which provides a diverse set of meteorological variables collected every 10 minutes from 302 station locations (160 of which are core meteo stations, while the rest are rain-gauges) distributed across Switzerland and is complemented with topographical indices. The geographical distribution of stations naturally forms a graph structure, where nearby stations exhibit correlated temperature patterns due to shared weather systems, terrain effects, and atmospheric dynamics. We aggregate the features to use hourly data.

Our main contributions are:

- Implementation of four temperature forecasting architectures with varying levels of spatial modeling: TCN (no graph), RNN with embeddings, STGNN with graph convolutions, and STGNN-Att with learned graph structure.
- Jupyter notebook for convenient dataset exploration.
- Evaluation using both deterministic (MAE) and probabilistic (Energy Score/CRPS) metrics at multiple forecast horizons.
- Comparison of models trained by augmenting low-quality sensors
- Comparison with operational NWP forecasts from ICON-CH1-EPS and simple algorithmic baselines such as Last-value and Moving Average

2 Related works

Multivariate Time Series forecasting Graph neural networks have been successfully applied to weather forecasting tasks where the spatial structure is naturally represented as a graph of observation stations [2]. The key insight is that message-passing operations can aggregate information from neighboring stations to improve local predictions.

Graph Deep Learning for Time Series Forecasting Introduces a comprehensive methodological framework formalizing the forecasting problem and providing design principles for graph-based predictors [3]

3 Methodology

3.1 Problem formulation

Given a spatiotemporal dataset with N weather stations and T timesteps, let $\mathbf{X} \in \mathbb{R}^{T \times F \times N}$ denote the multivariate observations where F is the number of features. We construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ where vertices $\mathcal{V}$ represent stations, edges $\mathcal{E}$ connect nearby stations, and $\mathbf{A}$ is the weighted adjacency matrix.

The forecasting task is to predict temperature at all N stations for the next $H = 24$ hours given the previous $W = 72$ hours of observations:

$$\hat{\mathbf{Y}}_{t:t+H} = f_\theta(\mathbf{X}_{t-W:t}, \mathcal{G}) \tag{1}$$

3.2 Graph construction

We construct the spatial graph based on geographical distance between stations. Let d_{ij} denote the haversine

distance between stations i and j. We compute edge weights using a Gaussian kernel:

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{\theta^2}\right) \tag{2}$$

where $\theta = \text{std}(d_{ij})$ is the standard deviation of all pairwise distances.

To create a sparse graph suitable for message-passing, we apply two filtering operations:

1. **K-Nearest Neighbors (KNN)**: For each node, retain only the K edges with highest weights.
2. **Threshold**: Remove edges with weights below a threshold τ.

With $N = 160$ stations distributed across Switzerland (latitude: 45.84–47.70N, longitude: 6.10–10.43E), the inter-station distances range from 2.36 km to 331.78 km with mean 117.76 km.

In our experiments, we set $K = 12$ and $\tau = 0.5$, resulting in 1,866 edges with an average degree of 11.66 neighbors per station.

3.3 Model architectures

We implemented four architectures with increasing levels of spatial modeling:

Model 0: Temporal Convolutional Network (TCN)

The TCN processes each station independently using dilated causal convolutions. This serves as a graph-free baseline to isolate the contribution of spatial modeling.

Architecture:

1. **Input projection**: Linear layer projecting input features to hidden dimension $h = 128$.
2. **Temporal blocks**: Stack of $L = 3$ residual blocks with dilated causal convolutions. Each block contains two convolution layers with kernel size $k = 3$, dilation 2^l for block l, batch normalization, ReLU activation, and dropout ($p = 0.2$).
3. **Output MLP**: Two-layer MLP projecting the final hidden state to $H \times 1$ outputs.

The forward pass processes input $\mathbf{x} \in \mathbb{R}^{B\times T\times N\times F}$ independently for each node:

$$\mathbf{h} = \text{TCN}(\text{proj}(\mathbf{x})), \quad \hat{\mathbf{y}} = \text{MLP}(\mathbf{h}[:, -1, :, :]) \tag{3}$$

Model 1: RNN with Node Embeddings

The RNN model adds learnable node embeddings to capture station-specific patterns while still processing temporal information independently.

Architecture:

1. **Node embeddings**: Learnable embedding matrix $\mathbf{E} \in \mathbb{R}^{N\times d_e}$ with $d_e = 32$.
2. **Conditional encoder**: Projects input features concatenated with embeddings to hidden dimension $h = 128$.
3. **GRU layers**: Stack of $L_t = 2$ GRU layers processing the temporal sequence.
4. **MLP decoder**: Multi-step decoder generating all H horizon outputs.
5. **Sampling layer**: Generates probabilistic outputs via linear noise injection.

Embeddings are concatenated before encoding and decoding stages:

$$\mathbf{x}_{\text{enc}} = [\mathbf{x}; \mathbf{e}], \quad \mathbf{h} = \text{GRU}(\text{encoder}(\mathbf{x}_{\text{enc}})) \tag{4}$$

Model 2: Spatiotemporal GNN (STGNN)

The STGNN extends the RNN model with graph convolution layers for spatial message passing, following a time-then-space paradigm.

Architecture:

1. **Temporal encoding**: Same as RNN (GRU with $L_t = 2$ layers).
2. **Spatial encoding**: Stack of $L_g = 2$ GraphConv layers that aggregate neighbor information:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}_{\text{root}}^{(l)}\mathbf{h}_i^{(l)} + \sum_{j\in\mathcal{N}(i)} \tilde{w}_{ij}\mathbf{W}^{(l)}\mathbf{h}_j^{(l)}\right) \tag{5}$$

   where $\tilde{w}_{ij}$ are normalized edge weights and σ is the ELU activation.
3. **Skip connection**: Residual connection from temporal to spatial output.
4. **Decoder**: Same MLP decoder and sampling layer as RNN.

The time-then-space paradigm first compresses temporal information, then propagates it spatially:

$$\mathbf{h}_t = \text{GRU}(\mathbf{x}), \quad \mathbf{h}_s = \text{GraphConv}(\mathbf{h}_t, \mathcal{G}), \quad \hat{\mathbf{y}} = \text{decoder}(\mathbf{h}_s + \mathbf{h}_t) \tag{6}$$

Model 3: STGNN-Att (Attention-Based STGNN)

This model architecture is heavily based on the paper "Attention-Based Spatial-Temporal Graph Neural Network With Long-Term Dependencies for Traffic Speed Prediction" [4]. Although the model architecture proposed in that work uses two stages: Long-term feature stage and Traffic speed prediction stage. We decided to only implement the second stage, tailoring it to our task of predicting air temperature.

When looking for suitable architecture we decided that the hardest part is going to be predicting the long temperatures like 12h, 18h, 24h. So we wanted to use model architecture that would best capture the long term features. Although the original model was meant for traffic speed prediction, we believe it was an interesting experiment to see whether improved results carry over from different tasks and domains and results clearly show that it works also for different kind of tasks.

Architecture:

The proposed model follows an attention-based spatial–temporal graph neural network design and is composed of four main components:

Feature projection. The input consists of historical air temperature measurements $\mathbf{X} \in \mathbb{R}^{T\times N\times F}$ for N nodes over T time steps, optionally augmented with exogenous variables. These features are projected into a latent space:

$$\mathbf{H}^{(0)} = \text{GELU}(\mathbf{X}\mathbf{W}_{\text{proj}} + \mathbf{b}_{\text{proj}}) \tag{7}$$

Graph structure learning. Instead of relying on a predefined or distance-based graph, the model learns the adjacency matrix directly from data. A learnable parameter matrix $\mathbf{\Theta} \in \mathbb{R}^{N \times N}$ is transformed into a soft adjacency matrix using Gumbel-Sigmoid relaxation:

$$\mathbf{A} = \sigma\left(\frac{\mathbf{\Theta} + \mathbf{G}}{\tau}\right) \qquad (8)$$
$$\mathbf{G} = \log(\mathbf{U}) - \log(1 - \mathbf{U}), \quad \mathbf{U} \sim \text{Uniform}(0, 1)$$

where τ is the temperature parameter controlling the sharpness of the sampling.

To encourage sparsity and local consistency, a k-nearest-neighbor graph $\mathbf{A}_{\text{KNN}}$ is computed from node embeddings derived from temporally averaged features $\bar{\mathbf{H}} = \frac{1}{T}\sum_t \mathbf{H}_t^{(0)}$. The KNN graph provides a regularization signal via binary cross-entropy loss $\mathcal{L}_{\text{graph}} = \text{BCE}(\mathbf{A}, \mathbf{A}_{\text{KNN}})$, but does not participate in message passing.

Spatial–temporal attention backbone. The core of the model is a Spatial-Temporal Attention WaveNet (STAWnet) backbone consisting of stacked ST-Blocks. Temporal dependencies are captured using gated causal convolutions with multiple dilation rates $d \in \{1, 2, 4\}$:

$$\mathbf{H}_{\text{temp}} = \tanh(\mathbf{W}_f * \mathbf{H}) \odot \sigma(\mathbf{W}_g * \mathbf{H}) + \mathbf{H} \qquad (9)$$

where $*$ denotes causal dilated convolution and $\odot$ is element-wise multiplication.

Spatial dependencies are modeled using dynamic multi-head attention. The learned adjacency is incorporated as an attention bias:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} + \log \mathbf{A}\right)\mathbf{V} \qquad (10)$$

Skip connections aggregate information across temporal scales.

Readout and prediction. The backbone produces a node representation $\mathbf{Z} \in \mathbb{R}^{N \times D}$ summarizing temporal dynamics and spatial context. A feed-forward network with ReLU activation maps these to future time steps. A sampling-based probabilistic decoder generates forecasts, enabling uncertainty estimation via Monte Carlo sampling during inference.

$$\hat{\mathbf{Y}} = \text{SampleDecoder}(\text{MLP}(\mathbf{Z})) \qquad (11)$$

4 Implementation

4.1 Framework

We implement all models using PyTorch with the `torch-spatiotemporal` library for graph operations. Training is managed with PyTorch Lightning, and hyperparameters are configured via Hydra. Experiments are tracked with MLflow.

4.2 Datasets

We use the PeakWeather dataset, which provides hourly meteorological observations from the SwissMetNet network. We focus on 160 meteorological stations (`station_type: meteo_station`), excluding rain gauges for the main experiments.

Target variable. Air temperature (°C).

Covariates. All available meteorological variables (humidity, pressure, wind speed/direction, precipitation, radiation, etc.) are used as input features. Binary masks indicate missing observations.

Temporal encodings. We include cyclical encodings for annual ($\sin$ / $\cos$ of day-of-year) and daily ($\sin$ / $\cos$ of hour-of-day) patterns to capture temperature seasonality.

Static features. Topographic attributes crucial for temperature modeling:

- Geographic coordinates (latitude, longitude)
- Station height (altitude in meters)
- Aspect and slope at 2km and 10km scales
- Topographic Position Index (TPI) distinguishing valleys from ridges

Data Split. We use temporal splitting:

- **Training:** 61,272 samples (May 15, 2023 – December 31, 2023)
- **Validation:** 2,184 samples (January 1 – March 31, 2024)
- **Test:** 8,738 samples (April 1, 2024 onwards)

Preprocessing. We apply global standardization (z-score normalization) across all stations. We preferred this method over per-node normalization because some stations have missing data for entire periods, which would result in undefined scaling factors. Missing values replaced with zeros.

4.3 Exploratory data analysis

To better understand the data distribution and inform model design, we performed an analysis of the spatial topography, temporal patterns, and feature correlations.

Spatial distribution. The dataset covers a diverse topography typical of the Alpine region. Figure 1 shows the locations of the 160 weather stations colored by elevation. Altitudes range significantly from 203m to 3571m, with a mean elevation of 1044m. This variation necessitates the inclusion of static topographic features in our models to account for adiabatic lapse rates and local microclimates.

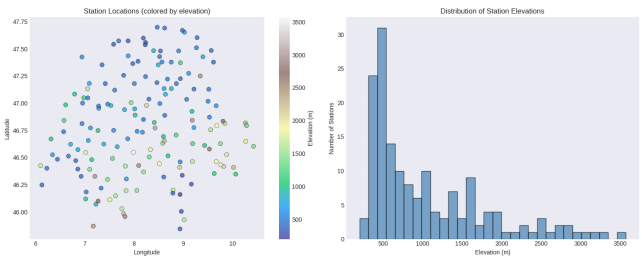

Figure 1. Spatial distribution of meteorological stations in Switzerland colored by elevation (left), and histogram of station elevations (right). The complex terrain justifies the use of graph-based methods.

Temporal periodicity. Temperature data exhibits strong periodic components. Figure 2 illustrates the seasonal and diurnal patterns for a representative station (BAS). The monthly averages show the expected summer peak, while the hourly profile demonstrates a consistent daily cycle peaking in the afternoon. These strong signals justify the use of cyclical temporal encodings ($\sin$ / $\cos$ of hour and day) as input features.

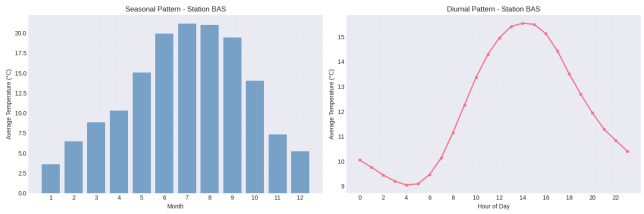

Figure 2. Average temperature patterns for station BAS. Left: Monthly seasonality. Right: Diurnal cycle showing daily heating and cooling.

Correlations. To identify the most predictive covariates, we analyzed feature correlations (Figure 3). Temperature shows a strong negative correlation with humidity (-0.55) and a positive correlation with sunshine duration ($+0.45$), confirming meteorological intuition. Other variables like pressure and wind direction show weaker linear correlations but likely capture non-linear interactions relevant for forecasting.

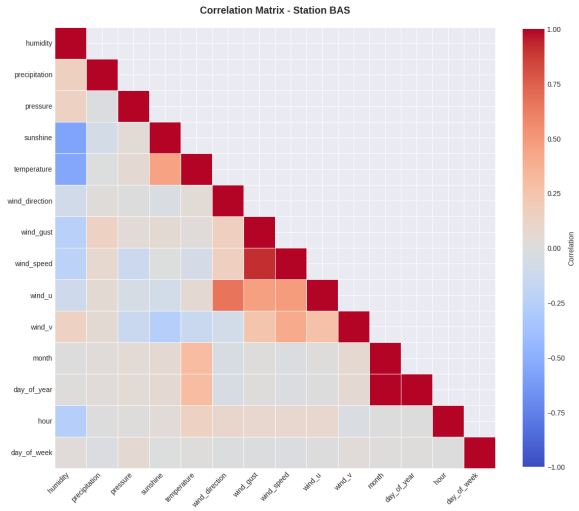

Figure 3. Feature correlation heatmap for dynamic variables. Humidity and sunshine duration are the strongest predictors of temperature.

Missing Data and Rain Gauges. Analysis of the temperature data availability reveals heterogeneous coverage across stations. Of the 160 meteo stations, 84 (52.5%) have complete temperature records with no missing values, while 66 stations (41.2%) exhibit partial missingness. Notably, 10 meteo stations (6.2%) have no temperature data whatsoever, despite being equipped with full sensor suites. Figure 4 shows the spatial distribution of missing data, with stations colored by their missing percentage. Overall, 6.98% of temperature values are missing across the dataset; however, excluding the 10 stations with complete data absence, the effective missing rate drops to only 0.77%, indicating that data gaps are highly concentrated in a small subset of stations.

Figure 4. Spatial distribution of missing temperature data across meteo stations. Green indicates complete data availability; red indicates high missing percentages. Ten stations have 100% missing temperature data.

The dataset also contains 142 rain gauge stations, of which only 41 have temperature measurements available. Rain gauges are equipped with a limited sensor suite compared to weather stations: they record only temperature and precipitation, lacking humidity, pressure, radiation, and sunshine duration sensors. This asymmetry in feature availability presents a challenge for graph-based models, as rain gauge nodes contribute fewer informative features during message-passing. Additionally, the temperature sensors on rain gauges are of lower quality, primarily designed to distinguish between solid and liquid precipitation rather than for precise temperature measurement.

Training considerations. In our experiments, we included all 160 meteo stations in the graph topology, including the 10 nodes with no temperature data. These nodes receive zero-imputed temperature values during training and contribute to spatial message-passing despite providing no ground-truth supervision signal. This design choice maintains the full spatial connectivity of the station network but introduces noise in the learning process. We hypothesize that training exclusively on the 150 stations with temperature data would yield improved forecasting performance by eliminating uninformative nodes from the optimization objective. This investigation is left for future work.

4.4 Hyperparameters

Training configuration.

- Optimizer: Adam with learning rate 10^{-3} and weight decay 10^{-4}
- Learning rate scheduler: MultiStep (decay $\gamma = 0.3$ at epochs 40, 60, 100)
- Batch size: 16 (with gradient accumulation over 2 steps for effective batch size 32)
- Epochs: 120 maximum with early stopping (patience 25)
- Loss function: Energy Score (CRPS)
- MC samples: 16 (train), 11 (validation), 100 (test)

The hyperparameters for each of the models can be found in Table 1.

Table 1. Model hyperparameters.

Model	Hyperparameters
TCN	Hidden size: 128 Temporal layers: 3 Dropout: 0.2 Kernel size: 3
RNN	Hidden size: 128 Embedding size: 32 Temporal layers: 2
STGNN	Hidden size: 128 Embedding size: 32 Temporal layers: 2 Graph layers: 2
STGNN-Att	Hidden size: 128 staw kernel size: 3 staw dilations: [1,2,4] staw attention heads: 4 graph knn: 10 graph tau: 1.0 dropout: 0.1

5 Results

5.1 Overall performance

Table 2 presents aggregate performance across all forecast horizons, averaged over 5 random seeds. The spatially-aware models (STGNN and STGNN-Att) achieve the best overall performance with MAE of 1.44°C and Energy Score of 1.038, representing improvements of **17.4%** (MAE) and **26.8%** (Energy Score) over the TCN baseline, and **6.9%** (MAE) and **7.1%** (Energy Score) over the RNN with node embeddings.

Table 2. Overall performance on test set (mean ± std across 5 seeds).

Model	Test MAE (°C)	Test Energy Score
TCN	1.745 ± 0.0122	1.417 ± 0.0109
RNN	1.549 ± 0.0068	1.117 ± 0.0059
STGNN	$\mathbf{1.442 \pm 0.0044}$	$\mathbf{1.038 \pm 0.0035}$
STGNN-Att	1.443 ± 0.0139	1.038 ± 0.0098

The STGNN exhibits the lowest variance across seeds (±0.0044 MAE). The STGNN-Att model matches STGNN performance but with higher variance (±0.0139 MAE).

5.2 Horizon-specific analysis

Tables 3 and 4 report MAE and Energy Score at the required forecast horizons ($t = 1, 3, 6, 12, 18, 24$ hours).

Table 3. Temperature forecasting MAE (°C) at different horizons. Values show mean ± std across 5 seeds for learned models.

Method	1h	3h	6h	12h	18h	24h
Naive	0.73	1.69	2.84	3.91	3.41	2.40
Moving Avg	2.36	2.47	2.61	2.83	3.01	3.19
ICON-NWP	1.23	1.31	1.38	**1.44**	**1.47**	**1.49**
TCN	0.85 ± 0.027	1.18 ± 0.022	1.48 ± 0.008	1.83 ± 0.016	2.03 ± 0.020	2.19 ± 0.020
RNN	0.56 ± 0.004	0.93 ± 0.004	1.26 ± 0.005	1.63 ± 0.008	1.86 ± 0.010	2.04 ± 0.011
STGNN	$\mathbf{0.56 \pm 0.010}$	$\mathbf{0.86 \pm 0.003}$	$\mathbf{1.14 \pm 0.004}$	1.50 ± 0.004	1.75 ± 0.006	1.95 ± 0.008
STGNN-Att	0.60 ± 0.021	0.91 ± 0.018	1.17 ± 0.021	1.49 ± 0.013	1.73 ± 0.010	1.92 ± 0.013

Table 4. Temperature forecasting Energy Score (CRPS) at different horizons. Values show mean ± std across 5 seeds for learned models.

Method	1h	3h	6h	12h	18h	24h
ICON-NWP	1.00	1.09	1.13	1.17	**1.18**	**1.19**
TCN	1.03 ± 0.015	1.15 ± 0.014	1.27 ± 0.010	1.45 ± 0.013	1.56 ± 0.013	1.65 ± 0.014
RNN	0.41 ± 0.002	0.68 ± 0.003	0.91 ± 0.004	1.18 ± 0.007	1.34 ± 0.009	1.46 ± 0.009
STGNN	$\mathbf{0.41 \pm 0.007}$	$\mathbf{0.62 \pm 0.002}$	$\mathbf{0.83 \pm 0.003}$	1.08 ± 0.003	1.26 ± 0.005	1.40 ± 0.006
STGNN-Att	0.44 ± 0.014	0.66 ± 0.013	0.84 ± 0.015	$\mathbf{1.07 \pm 0.009}$	1.24 ± 0.007	1.38 ± 0.010

Short-term forecasts (1–6h). All learned models significantly outperform ICON-CH1-EPS at short horizons. At t=1h, the RNN and STGNN achieve MAE of 0.56°C compared to ICON's 1.23°C—a **54% error reduction**. This demonstrates that data-driven models effectively exploit recent observations that NWP models, constrained by physical simulation timesteps, cannot fully leverage.

Medium-term forecasts (6–12h). The gap between learned models and ICON narrows. At t=12h, STGNN achieves 1.50°C versus ICON's 1.44°C. Spatial models (STGNN, STGNN-Att) maintain an advantage over temporal-only models (RNN, TCN), with STGNN providing approximately 0.13°C improvement over RNN at t=12h.

Long-term forecasts (18–24h). ICON-CH1-EPS demonstrates superior performance at longer horizons, achieving 1.49°C at t=24h compared to 1.92–2.19°C for learned models. This reflects the advantage of physics-based NWP for capturing synoptic-scale weather evolution. However, the STGNN-Att model shows the smallest degradation at t=24h (1.92°C), suggesting that dynamically learned graph structures combined with temporal self-attention better capture long-range dependencies than fixed graph convolutions.

5.3 Model comparison

TCN vs. RNN. The RNN with node embeddings improves substantially over TCN (1.55 vs. 1.75 MAE), indicating that station-specific learned representations capture important local characteristics (elevation effects, urban heat islands, valley inversions) that pure temporal modeling misses.

RNN vs. STGNN. Adding graph convolutions yields consistent improvement across all horizons (1.44 vs. 1.55 MAE), validating our hypothesis that explicit spatial message-passing helps propagate information from neighboring stations. The benefit is most pronounced at medium horizons (6–12h) where local weather systems span multiple stations.

STGNN vs. STGNN-Att. When looking at the direct comparison in Table 4 and Table 3, the data suggests mixed conclutions since there are cases where STGNN is better and there are cases where STGNN-Att is better. Our goal was for STGNN-Att to outperform every model, but this only party came true. We are satisfied that the hypothesis of long term prediction came true, since STGNN-Att clearly captures long term features better than STGNN. When it comes to short term predictions the the baseline STGNN outpreforms the STGNN-Att.

5.4 Rain gauge integration

To assess whether lower-quality sensors improve forecasting, we conducted two experiments incorporating 41 (out of 142) rain gauge stations alongside the 160 weather stations.

Table 5. MAE (℃)tested on all stations (all) and tested on weather stations only (w.o.) on seed 1

Method	1h	3h	6h	12h	18h	24h
TCN all	1.145	1.428	1.715	2.043	2.217	2.373
TCN w.o.	1.135	1.419	1.708	2.036	2.210	2.367
RNN all	0.595	0.988	1.340	1.736	1.983	2.166
RNN w.o.	0.579	0.971	1.324	1.716	1.960	2.140
STGNN all	0.584	0.909	1.207	1.573	1.820	2.000
STGNN w.o.	0.571	0.901	1.209	1.577	1.824	2.007
STGNN-Att all	0.660	0.981	1.267	1.622	1.884	2.051
STGNN-Att w.o.	0.653	0.982	1.276	1.633	1.888	2.049

Table 6. Energy Score tested on all stations (all) and tested on weather stations only (w.o.) on seed 1

Method	1h	3h	6h	12h	18h	24h
TCN all	0.923	1.059	1.213	1.410	1.526	1.630
TCN w.o.	0.919	1.053	1.208	1.406	1.521	1.626
RNN all	0.420	0.689	0.930	1.201	1.367	1.490
RNN w.o.	0.408	0.677	0.920	1.187	1.352	1.472
STGNN all	0.408	0.630	0.836	1.089	1.252	1.370
STGNN w.o.	0.398	0.625	0.837	1.091	1.254	1.373
STGNN-Att all	0.454	0.675	0.869	1.114	1.295	1.408
STGNN-Att w.o.	0.454	0.675	0.869	1.121	1.298	1.406

Effect on weather station predictions

We compared the STGNN trained exclusively on weather stations against the STGNN trained on all 201 nodes but evaluated only on weather stations.

Table 7. Impact of rain gauge integration on weather station forecasting (STGNN, seed 1).

Training Data	MAE (℃)	Energy Score
Weather stations only	1.442 ± 0.004	1.038 ± 0.004
All stations (eval on weather)	1.515	1.096
Difference	**-5.1%**	**-5.6%**

Including rain gauges **does not improve** weather station predictions. Performance degrades by approximately 5% in both MAE (1.44°C $\rightarrow$ 1.52°C) and Energy Score ($1.04 \rightarrow 1.10$). This indicates that noise from lower-quality sensors propagates through graph convolutions, outweighing any benefit from increased spatial coverage.

Performance by station type

We trained on all 201 nodes and compared prediction errors between station types.

Table 8. STGNN performance by station type when trained on all 201 nodes (seed 1).

Station Type	Count	MAE (℃)	Energy Score
Weather stations	160	1.515	1.096
Rain gauges	41	1.527	1.102
Difference	—	**+0.8%**	**+0.5%**

Rain gauge predictions exhibit **0.8% higher MAE** than weather stations—a surprisingly small gap. This suggests the STGNN effectively leverages spatial context from neighboring weather stations to compensate for the limited input features at rain gauge nodes.

6 Discussion

6.1 Key findings

Deep Learning models excel at short horizons. Learned models achieve up to 54% lower error than NWP at 1-hour horizons by effectively leveraging recent observations. This advantage diminishes with forecast horizon as physical dynamics become dominant. Beyond 12 hours, ICON-CH1-EPS outperforms all learned models in MAE.

Graph structure provides consistent benefit. Spatial message-passing improves performance by 6.9% (MAE) and 7.1% (Energy Score) over temporal-only models across all horizons, validating the STGNN approach for weather station networks.

Graph connectivity is critical. Our experiments revealed that sparse graph connectivity can prevent spatial models from leveraging their architectural advantages. With only 6.48 average neighbors per station in early experiments, the STGNN could not effectively aggregate spatial information, leading to poor performance. Increasing connectivity to 11.66 neighbors was essential for spatial models to outperform temporal baselines.

Simple graphs suffice for this task. Distance-based graph construction matches learned graph structures in performance while offering better stability and interpretability. The STGNN and STGNN-Att achieve nearly identical overall MAE (1.442 vs 1.443°C), but STGNN exhibits lower variance across seeds. This suggests that for the Swiss station network, geographical proximity is a strong prior for spatial dependencies.

STGNN-Att is better at longer horizons. While STGNN dominates at short horizons (1–6h), the attention-based STGNN-Att achieves the best performance at longer horizons (12–24h). At t=24h, STGNN-Att achieves 1.92°C MAE compared to STGNN's 1.95°C, suggesting that dynamically learned graph structures combined with multi-head attention better capture long-range temporal dependencies.

Rain gauges degrade predictions. Including lower-quality sensors degrades weather station predictions by approximately 5%, indicating that noise from rain gauge sensors propagates through graph convolutions and outweighs any benefit from increased spatial coverage. However, predictions at rain gauge locations are only 0.8% worse than at weather stations, suggesting the model leverages spatial context from neighboring high-quality stations effectively.

6.2 Comparison with NWP

ICON-CH1-EPS outperforms learned models beyond 12-hour horizons, reflecting the fundamental advantage of physics-based simulation for synoptic-scale weather evolution. However, learned models offer complementary strengths for short-term nowcasting. A hybrid approach using data-driven corrections for short horizons and NWP for longer horizons could leverage both paradigms.

6.3 Limitations

- Hyperparameter tuning was limited; more extensive search might improve results.

- We did not explore ensemble methods combining multiple architectures.
- The fixed temporal resolution (1 hour) may miss sub-hourly dynamics relevant for short-term forecasting.

7 Conclusion

We presented a systematic evaluation of spatiotemporal graph neural networks for temperature forecasting using the PeakWeather dataset. Our experiments addressed all the main project deliverables:

Task 1 (Implementation): We implemented four architectures: TCN as a graph-free baseline (Model 0), RNN with learnable node embeddings (Model 1), STGNN with graph convolutions following a time-then-space paradigm (Model 2), and STGNN-Att with learned graph structure (Model 3).

Task 2 (Performance assessment): We evaluated all models on MAE and Energy Score at six forecast horizons ($t = 1, 3, 6, 12, 18, 24$h), compared against ICON-CH1-EPS, and reported results across 5 random seeds. The STGNN achieves the best trade-off between accuracy (1.44°C MAE, 1.04 Energy Score) and stability (lowest variance across seeds).

Task 3 (Rain gauge integration): Contrary to expectations, including rain gauges **does not improve** weather station predictions—performance degrades by 2–5% due to noise propagation through graph convolutions. When evaluating predictions at rain gauge locations, we observe marginally higher errors compared to weather stations, attributable to lower sensor quality and fewer available input features.

The **STGNN** with time-then-space architecture emerges as the recommended approach for this task. Distance-based graphs provide comparable performance with better interpretability. learned models excel at short-term forecasting where recent observations dominate, while NWP remains superior for longer horizons where physical dynamics become critical.

7.1 Future work

Several directions could extend this work:

- **Missing value handling**: The current pipeline fills missing observations with zeros, which likely injects artificial noise into the learning process. A more principled strategy would be to propagate the last available observation forward in time, or to explicitly leverage the provided observation masks within the model architecture.
- **Hybrid NWP integration**: Data-driven short-term predictions could be combined with Numerical Weather Prediction (NWP) forecasts for longer horizons, leveraging the complementary strengths of physics-based and learning-based approaches.
- **Higher temporal resolution**: The native 10-minute resolution of PeakWeather could be exploited to capture sub-hourly dynamics relevant for nowcasting, rather than aggregating observations to hourly intervals.
- **Graph sensitivity and construction**: A systematic analysis of the sensitivity of model performance to graph hyperparameters (e.g., distance thresholds or connectivity rules) would help clarify the robustness of graph-based approaches. Additionally, alternative graph construction heuristics beyond geographic distance could be explored and compared.
- **Learned graph structures**: Instead of relying on a fixed, predefined graph, future work could investigate models that jointly learn graph structure and model parameters. Analyzing the resulting learned graphs may provide insight into physically meaningful or emergent spatial relationships, as well as their stability across training runs.
- **Augmenting with external variables**: Incorporating additional meteorological variables from external data sources could further improve forecasting accuracy. A controlled comparison would help assess the marginal value of such auxiliary information.
- **Comparison with foundation models**: Finally, benchmarking the proposed approaches against recent foundation time-series models could provide valuable context regarding performance, generalization, and computational trade-offs.

7.2 Experimental setup

Experiments were conducted on a NVIDIA RTX 4090 GPU and a NVIDIA DGX Spark with GB10 superchip.

Code is available at: `temperature-forecasting`.

References

[1] Daniele Zambon, Michele Cattaneo, Ivan Marisca, Jonas Bhend, Daniele Nerini, and Cesare Alippi. Peakweather: Meteoswiss weather station measurements for spatiotemporal deep learning. *arXiv preprint arXiv:2506.13652*, 2025.

[2] Victor Garcia Satorras, Syama Sundar Rangapuram, and Tim Januschowski. Multivariate time series forecasting with latent graph inference. *arXiv preprint arXiv:2203.03423*, 2022.

[3] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. Graph deep learning for time series forecasting. *ACM Computing Surveys*, 57(12):1–34, 2025.

[4] Quoc-Viet Nguyen, Chun-Yu Tai, Khanh-Duy Nguyen, Min-Te Sun, Wu-Yuin Hwang, Kazuya Sakai, and Wei-Shinn Ku. Attention-based spatial-temporal graph neural network with long-term dependencies for traffic speed prediction. *IEEE Transactions on Intelligent Transportation Systems*, 26(11):18829–18838, 2025. doi: 10.1109/TITS.2025.3591264.