A Comparative Evaluation of Imputation Models for Agricultural Weather Networks

Awanish Khanal
Washington State University
Pullman, WA, USA
awanish.khanal@wsu.edu

Monowar Hasan
Washington State University
Pullman, WA, USA
monowar.hasan@wsu.edu

Abstract

High-resolution weather data are essential for irrigation scheduling, frost protection, and pest and disease risk modeling. However, weather stations frequently experience multi-hour to multi-day outages, leading to substantial downtime for weather-driven decision-making. To mitigate this, stakeholders often rely on "imputation" models to reconstruct missing data. Despite the existence of many statistical and machine-learning models, it remains unclear *which imputation methods are most suitable* for operational agricultural settings. This paper evaluates *twelve imputation methods*—statistical models, classical machine learning algorithms, and deep neural networks—to identify the most suitable model for agricultural applications. We tested the models using data from five meteorological towers (three in Jena, Germany, and two in Sunnyside, Washington, USA). We perform a thorough performance-engineering study: in addition to accuracy, we evaluate runtime, inference throughput, peak memory usage, GPU usage, energy consumption, and monetary cost. Our findings are surprising: among all complex and advanced (neural network) models, *a properly tuned Random Forest (RF) model consistently outperforms* them across multiple evaluation categories (e.g., accuracy, latency, throughput, and cost). Specifically, an RF achieves competitive error with no GPU dependencies, modest memory usage, and substantially lower energy expenditure than deep learning baselines. Our research shows that classical machine learning models remain a compelling choice for scalable, cost-aware weather data imputation in agricultural decision-support systems.

CCS Concepts

• **Applied computing** → **Agriculture**; • **Computing methodologies** → *Machine learning*; • **Information systems** → Data analytics.

Keywords

Weather Data Imputation, Performance Evaluation, Energy and Cost Efficiency, Machine Learning, Spatio-temporal Modeling

ACM Reference Format:
Awanish Khanal and Monowar Hasan. 2026. A Comparative Evaluation of Imputation Models for Agricultural Weather Networks. In *Proceedings of the 17th ACM/SPEC International Conference on Performance Engineering (ICPE '26), May 04–08, 2026, Florence, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3777884.3797006

ICPE '26, Florence, Italy

ACM ISBN 979-8-4007-2325-4/2026/05
https://doi.org/10.1145/3777884.3797006

1 Introduction

High-resolution weather measurements have become essential for almost every modern agricultural decision. Accurate observations of temperature, humidity, wind speed, solar radiation, and other meteorological parameters enable farmers to make informed and timely decisions. These metrics are crucial for predicting critical agricultural events, such as pest outbreaks, the ideal timing for irrigation based on plant water stress, the risk of frost damage, and the likelihood of disease occurrences [7, 18, 36]. Agriculture stakeholders (e.g., growers, suppliers) can effectively anticipate potential problems and respond proactively when they have reliable data. For instance, growers can optimize irrigation schedules to use water more efficiently, apply protective measures such as frost fans precisely when needed to safeguard crops, or strategically time pesticide applications to manage pests effectively. These data-driven decisions power modern digital agriculture and underpin agricultural efficiency and profitability in recent years.

However, weather stations are not immune to occasional data loss and anomalies. For instance, antennas drop their telemetry link when a cell tower reboots, radiation shields clog with dust or snow, or sensors that look fine in the field get rejected minutes later by automated quality-control flags [22]. These interruptions usually last for several hours to more than a day. However, even a short gap of a few hours can jeopardize critical agricultural workflows [36]. For example, if the temperature falls below freezing during the gap, wind machines or overhead sprinklers may spin up too late, leaving tender blossoms ice-burned. The opposite happens on a hot summer afternoon: evapotranspiration models fed by incomplete data overestimate crop water use, so growers pump thousands of gallons more than they need. Hence, a missing record is not a benign statistical artifact; it is a potential source of immediate economic loss that can destroy yields and tighten already-thin profit margins in agriculture.

One remedy is to *impute* missing weather samples before they reach downstream tools. A large body of work has explored methods ranging from simple climatological averages and Kalman filters to tree ensembles and deep sequence models with attention or generative structure. Most studies report only forecast accuracy (e.g., MAE, RMSE) under idealized outage patterns. However, a comparative evaluation of state-of-the-art imputation models is lacking. Besides, there is a little discussion of the *systems cost* of deploying these imputers in real networks. For instance, *(a) how much CPU/GPU time do they require? (b) how much energy do they consume per million gap fills? (c) can they be retrained overnight across dozens of towers, or do they demand dedicated GPU servers?*

These questions are central to the operation of regional weather networks but are not explicitly evaluated in the literature.

This paper explores weather-gap imputation as a performance-engineering problem. Using more than a decade of high-quality 10-minute observations from three towers near Jena, Germany [29], and two semi-arid agricultural stations from the AgWeatherNet network in Washington State, USA [41], we benchmark 12 imputation methods spanning statistical, classical machine learning, and deep learning families. In addition to accuracy, we measure runtime, throughput, peak memory, GPU usage, energy, and monetary cost on a realistic cloud hardware configuration. We further study whether lightweight spatio-temporal representations (e.g., GraphSAGE-GRU embeddings [16, 25]) provide enough benefit to justify their extra GPU budget. Our results show that *(a) a tuned RF model still delivers the best overall trade-off between accuracy and resource usage*, and *(b) more complex deep sequence models offer only modest gains at substantially higher compute cost.*

1.1 The Context and Our Research

Imputing missing values is the most direct way to repair the weather data streams before they enter the downstream agricultural decision tools. A wide range of imputation techniques exists across the time-series literature: simple statistical approaches such as climatological means and Kalman smoothers [20, 24, 30, 31], distance-weighted KNN and spline interpolation [15, 26, 39], tree-ensemble imputers such as MissForest [8, 37], and modern deep-learning models, including BRITS/LSTM, GAIN, and attention transformers like TFT [10, 12–14, 17, 21, 43]. What remains unclear, however, is which of these families is *most reliable* under the types of short, irregular outages that occur in agricultural weather networks.

Lightweight statistical models are fast and easy to deploy, but tend to degrade sharply during abrupt weather transitions, sensor drift, or clustered outages [22, 26, 27, 31]. Classical machine-learning regressors often perform well on clean datasets, but struggle to capture nonlinear meteorological dynamics or tail regimes such as heatwaves, high radiation, or high-wind conditions [1, 6, 11, 28, 32, 38, 40]. Deep sequence models offer greater representational capacity, but their substantial training and inference cost raises the question of whether their additional complexity is justified for sub-day gaps in high-frequency weather records.

These considerations motivate our first research question:

> *For realistic multi-hour outages in agricultural weather streams, which imputation family provides the best balance of accuracy, robustness, and computational cost?*

A second challenge is that weather stations rarely operate as isolated sensors. Towers within the same region often share synoptic forcing, radiative conditions, and boundary-layer structures. This raises a natural question: can imputers exploit spatial and short-term temporal context to improve gap filling without excessively increasing model complexity? To examine this, we introduce a lightweight spatio-temporal encoder (GraphSAGE-GRU) that learns cross-tower representations and exposes them as frozen auxiliary features to the downstream imputers.

This leads to our second research question:

> *Do simple, learned spatio-temporal embeddings measurably improve imputation accuracy, and is the gain worth the additional GPU budget?*

To answer these questions, we assemble a unified performance benchmark across two independent datasets: three long-record Jena towers and two semi-arid AgWeatherNet towers. We evaluate 12 imputers spanning statistical, classical machine learning, and deep learning families and measure both accuracy and system metrics (runtime, throughput, energy, and cost). The result is a performance-grounded assessment of *which methods offer the strongest balance of accuracy and operational practicality* for real-world agricultural deployments.

1.2 Our Contributions

This paper makes the following contributions.

- **A unified accuracy and systems benchmark for weather-data imputation.** We evaluate twelve representative imputation methods—*four statistical*, *four classical machine learning*, and *four deep sequence* models—on two independent data sources (one from Europe and the other in North America). The weather streams used in this work are: a 14-year, 10-minute-resolution archive from three Jena towers (Germany) and a 2-year, 5-minute-resolution semi-arid agricultural dataset from two AgWeatherNet stations (Washington, USA). All methods are compared under identical conditions, synthetic outages, and stress-test regimes.
- **A performance-grounded evaluation spanning runtime, throughput, memory, and energy cost.** Beyond MAE/RMSE, we measure *training time*, *inference latency*, *CPU/GPU utilization*, *energy footprint*, and *monetary cost* on a realistic cloud configuration (2 vCPUs + NVIDIA T4). This allows us to assess not only *which* imputer is most accurate, but which one is *most practical* for deployment in regional agricultural networks.
- **A lightweight spatio-temporal representation learning module (GraphSAGE-GRU) and its impact.** We introduce a simple encoder that learns short-range spatial and temporal context across towers, freezes the resulting embeddings, and exposes them as auxiliary features to downstream imputers. We quantify both the *accuracy gains* and the additional *GPU cost.*

1.3 Key Findings

Across both datasets and all outage regimes, a clear pattern emerges. A well-tuned **RF** provides the best overall balance of accuracy, robustness, and computational efficiency. We compute errors over the masked set M (Sec. 3.1), comparing ground truth y_t vs. imputed $\hat{y}_t$. We report MAE (Mean Absolute Error) and RMSE (Root Mean Square Error) in °C. On the Jena tower, RF reaches **0.46°C MAE**, outperforming all statistical baselines and remaining consistently ahead of the deep sequence models (TFT, BiLSTM+Attn, SDAE). On the more volatile AgWeatherNet station, RF again achieves the lowest error (**0.72°C MAE**), preserving its margin under temperature and multivariate extremes.

Deep learning models do offer competitive accuracy, but at substantially higher runtime and energy cost—often **10–60×** more expensive per million imputations than RF. Meanwhile, the GraphSAGE-GRU embeddings yield **modest but consistent**

improvements (0.02–0.08°C), with the largest gains under stress-regimes, but they require additional GPU training and storage.

Overall, the benchmark suggests that in operational agricultural settings where outages are short, retraining budgets are finite, and imputation must scale across many towers, **RF delivers the most substantial return on compute and accuracy**. Deep sequence models and spatio-temporal embeddings offer incremental gains, but their higher cost makes them best suited for specialized or resource-rich deployments rather than routine gap-filling.

2 Evaluation Setup

2.1 Dataset

A robust assessment of imputation methods requires long, high-quality time series with diverse climatological regimes and realistic patterns of operational outages. To this end, we assemble two independent datasets: *(a)* three meteorological towers located near Jena, Germany, and *(b)* two agricultural monitoring towers situated near Sunnyside in the Yakima Valley region of Washington State, USA. The Jena towers report 10-minute observations, while the AgWeatherNet towers report at a 5-minute cadence.

Jena Towers (Germany) [29]. The Jena stations provide a continuous 2011–2024 archive (∼1.4 million 10-minute timestamps), covering multiple ENSO phases, the 2018 European heatwave, winter cold-pool episodes, and a broad range of synoptic conditions. The three towers (*WS Saaleaue*, *WS Beutenberg*, and *Versuchsbeete*) share nearly identical hardware and metadata formats, making them well suited for a long-horizon evaluation of outage patterns and recovery behavior.

Sunnyside Towers (AgWeatherNet, USA) [41]. The two Sunnyside stations provide a complementary dataset sampled from March 2023 to March 2025. Although the record is shorter than Jena's, the AgWeatherNet network covers a semi-arid, irrigated agricultural environment with steep diurnal temperature cycles, strong radiative forcing, and sensor-level constraints such as battery fluctuations and logger temperature drift. These characteristics yield a distinct failure profile relative to the Jena towers.

2.1.1 Independent Training and Evaluation. All models are *trained and evaluated separately* on the Jena and AgWeatherNet datasets. This prevents information leakage across climates, hardware configurations, sampling intervals, or temporal coverage. The Sunnyside towers serve as a *short, independent testbed* that allows us to assess whether the model ranking and robustness trends observed in Jena carry over to a different sensor design, measurement cadence, and microclimate.

This two-dataset design ensures that all conclusions about accuracy, runtime, throughput, memory footprint, and energy cost remain valid across distinct operational settings, not only within a single long-term tower archive.

2.1.2 Raw Variables. Table 1 summarizes the principal meteorological channels available at the Jena and Sunnyside towers. The Jena stations provide an extensive suite of derived thermodynamic and radiation measurements, while the AgWeatherNet towers include additional soil and diagnostic variables typical of agricultural monitoring systems.

Table 1: Key meteorological variables at Jena and AgWeatherNet towers (native 10-min and 5-min data, each aggregated to a 10-minute grid).

Attributes	Jena towers	AgWeatherNet towers
Air temperature	Air temperature (2 m, aspirated)	Air temperature
Humidity / vapour	Relative humidity; specific humidity; dew-point temperature; saturation vapour pressure; actual vapour pressure; vapour-pressure deficit; water-vapour concentration; air density	Vapor pressure; vapour-pressure deficit (VPD); Relative-Humi-sensor temperature
Pressure	Atmospheric pressure	Atmospheric pressure; reference calibration pressure
Wind	Wind speed (10 m); wind direction (10 m)	Wind speed; gust speed; wind direction
Radiation	Short-wave downwelling radiation; net radiation; infrared canopy temperature; sunshine duration	Solar radiation
Precipitation	Precipitation (tipping-bucket)	Precipitation; maximum precipitation rate
Soil/canopy	Soil temperature (6 depths); soil moisture (5 depths)	Soil temperature; matric potential

2.1.3 Feature Set. We use the following feature sets.

Core Meteorological Attruibutes. We retain variables with first-order physical relevance to near-surface air temperature. For the Jena dataset, this includes `Tpot`, `Tdew`, `RH`, and `VPdef` (moisture control) [32]; `SWDR` and `Rn` (radiative forcing) [3, 11]; `rain` (evaporative cooling) [28]; and `wv` (mechanical mixing) [38]. For the AgWeatherNet dataset, we select the physically analogous subset: air temperature, vapor pressure, VPD, wind speed and direction, solar radiation, and precipitation. In both datasets, soil and diagnostic channels contribute negligible predictive value (importance < 0.02) and are excluded.

Temporal Encodings. Sine-cosine encodings of hour-of-day and day-of-year are added to expose diurnal and seasonal structure to non-sequence models [10, 14].

Lagged Context. Single and double 10-minute lags of air temperature and relative humidity (T_{-1}, T_{-2}, RH_{-1}) provide short-term autoregressive memory.

Multi-Tower Alignment. Within each dataset, predictors are standardized using statistics derived from that dataset's training split and aligned to its own 10-minute clock. Native missing values are preserved so that each imputation method encounters realistic operational data. When cross-tower encoders (e.g., GraphSAGE-GRU) are used, they operate only within a dataset (i.e., Jena towers form one graph; Sunnyside towers form a separate graph). Downstream tabular models are always trained within the same dataset and never transfer parameters across datasets.

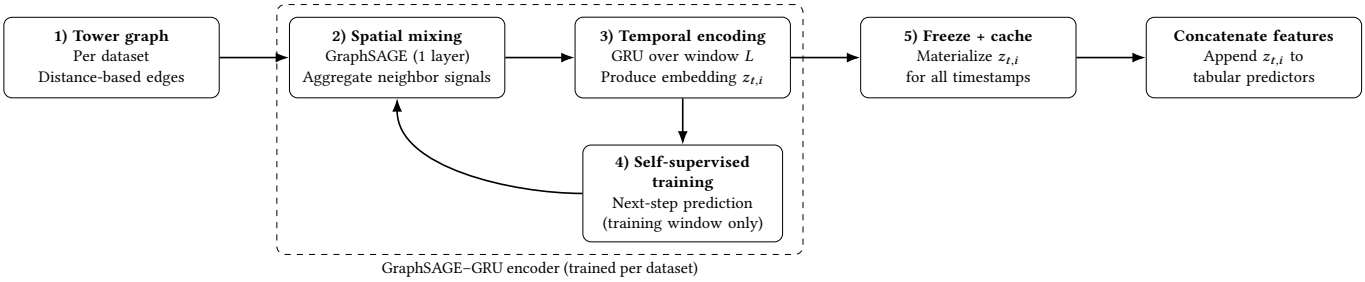

Figure 1: GraphSAGE–GRU as a lightweight feature generator. The encoder is trained self-supervised per dataset, frozen, and its embeddings are concatenated to the original tabular features for downstream imputation models.

Target Variable. In all experiments, the 10-minute near-surface air temperature (T or its AgWeatherNet equivalent) is treated as the imputation target for each dataset independently.

2.2 Spatio-Temporal Representation Learning with GraphSAGE–GRU

The five-tower benchmark exhibits meaningful spatial and temporal structure: nearby towers often share synoptic forcing, radiative conditions, and boundary-layer dynamics. To examine whether such a structure yields informative representations for downstream imputation, we introduce a lightweight encoder that combines spatial graph aggregation with temporal recurrence. The encoder is trained separately on each dataset (three Jena towers and two AgWeatherNet towers), and no cross-dataset parameters are shared.

Overview (How the Encoder Is Used). We use GraphSAGE–GRU only as a lightweight feature generator:

(1) Build a small tower graph within each dataset, with edge weights based on inter-tower distance (Eq. (1)).
(2) At each time t, apply a one-layer GraphSAGE operator to mix tower i's features with a weighted summary of other towers (Eq. (2)).
(3) Feed each tower's sequence through a GRU over a short history window to obtain $z_{t,i}$ (Eq. (3)–(6)).
(4) Train the encoder via next-step prediction on the training window.
(5) Freeze the encoder, materialize $z_{t,i}$ for all timestamps, and concatenate $z_{t,i}$ to the original tabular features for downstream imputers (Eq. (7)).

Figure 1 summarizes the GraphSAGE–GRU feature-generation pipeline.

Spatial Graph Construction. Within each dataset, we construct a fully connected undirected graph $G = (V, E)$ where each node $i \in V$ corresponds to a tower reporting at time t. Edges are weighted by a radial basis kernel of the inter-tower distance:

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{2\sigma^2}\right), \qquad i \neq j. \tag{1}$$

We tune σ per dataset via a small sweep $\sigma \in \{\alpha \cdot \text{median}(\{d_{ij}\}_{i<j}) : \alpha \in \{0.5, 1, 2\}\}$ and select the value minimizing validation MAE for the encoder's self-supervised next-step prediction task.

Let $x_{t,i} \in \mathbb{R}^F$ be the feature vector of tower i at time t (as defined in Sec. 2.1.3). We use a single-layer GraphSAGE operator to obtain a spatial embedding:

$$h_{t,i} = \sigma\big(W_1 x_{t,i} \,\|\, W_2 \,\text{AGG}\big\{ w_{ij} x_{t,j} : j \in \mathcal{N}(i)\big\}\big), \tag{2}$$

where $\|$ denotes concatenation, σ is a nonlinear activation (ReLU [33]), W_1, W_2 are learnable matrices, and AGG$(\cdot)$ is mean aggregation weighted by w_{ij}. Because each dataset has at most three towers (Jena) or two towers (Sunnyside), the spatial operator remains intentionally minimal while still capturing shared mesoscale variability.

Temporal Encoding. To summarize recent evolution, each spatial embedding sequence $\{h_{t-L+1,i}, \ldots, h_{t,i}\}$ is passed through a Gated Recurrent unit (GRU) [16]:

$$r_t = \sigma\big(W_r h_{t,i} + U_r z_{t-1,i}\big), \tag{3}$$

$$u_t = \sigma\big(W_u h_{t,i} + U_u z_{t-1,i}\big), \tag{4}$$

$$\tilde{z}_t = \tanh\big(W_h h_{t,i} + U_h (r_t \odot z_{t-1,i})\big), \tag{5}$$

$$z_{t,i} = u_t \odot z_{t-1,i} + (1 - u_t) \odot \tilde{z}_t, \tag{6}$$

where $z_{t,i}$ is the resulting temporal embedding for tower i at time t, u_t and r_t are update and reset gates, and $W_\cdot$, $U_\cdot$ are learned parameters. The final representation $z_{t,i} \in \mathbb{R}^d$ provides a compact summary of short-term temperature, humidity, and radiation dynamics conditioned on local spatial context.

Frozen Embeddings for Downstream Models. After training, the encoder is frozen and applied across the full timeline to generate a representation sequence $\{z_{t,i}\}$ for each tower. These embeddings are appended to that tower's tabular feature vector:

$$X_{t,i}^{\text{extended}} = \big[X_{t,i}^{\text{original}} \,\|\, z_{t,i}\big], \tag{7}$$

where $X_{t,i}^{\text{original}}$ denotes the predictors from Sec. 2.1.3. The downstream imputation models (Sec. 2.3) are trained separately on the Jena and AgWeatherNet datasets using these extended features. This design allows us to test whether learned spatio-temporal structure improves imputation accuracy over purely tabular features, without altering model capacity or violating dataset isolation.

Scope. The encoder is used solely to produce auxiliary features; all primary imputation models remain unchanged. We use the encoder to examine whether spatial–temporal structure, even in a

small multi-tower setting, yields embeddings that classical models (e.g., RF) can exploit. A performance-oriented analysis (runtime, throughput, memory, and energy) is discussed in Sec. 3.

2.3 Models: Twelve Ways to Impute

With more than a decade of weather observations in hand (see Sec. 2), the next step is to select a *diverse yet manageable* line of imputation techniques that span the statistical to deep spectrum. Table 2 details the type/category of the various imputation models employed in the study. We also list the selected hyperparameters for each model. We use Adam optimizer ($\eta = 10^{-3}$) and batch size 256 for training.

2.3.1 Why These Twelve Models? We select these 12 models as they satisfy the following criteria.

(C1) *Distinct Inductive Bias.* Our chosen set of models represents every major modeling paradigm: linear regression, kernel method, tree ensemble, autoencoder, recurrent sequence model, GAN, and attention transformer, so any single family does not dominate the roster.
(C2) *Field Relevance.* The algorithms are heavily used in prior time-series imputation literature, so there is evidence that practitioners or researchers already trust them.
(C3) *Hardware Requirements.* The set spans the full compute ladder: light statistical baselines that run easily on general-purpose computers or even embedded boards; tree ensembles and KNN that fit comfortably on a laptop CPU (including RF); and GPU-accelerated deep learners (e.g., TFT) for machines with dedicated hardware. We deliberately aim to reflect real-world deployments, ranging from low-power edge devices in field stations to high-power GPU servers or clusters, so that every practitioner can find a feasible option.

Applying (C1)–(C3) produces the twelve-model suite as listed in Table 2: four statistical baselines, four classical machine learning models, and four deep-learning sequence models. Together, this set of models helps us study if the added representational capacity translates into better imputation of high-frequency weather data and, more importantly, how much any extra accuracy translates into agronomic benefit.

2.3.2 Training Process. All models ingest the same feature matrix from Sec. 2.1.3. Continuous predictors are z-score standardized. After hyperparameter tuning, see Table 2 for the final hyperparameter settings.

Thus far, the imputation methods introduced range from linear baselines to GPU-hungry Transformers, and they were trained on a common feature set. The following section (see Sec. 3) presents an evaluation of these methods to answer a performance-centric question: *which model offers the best trade-off between imputation accuracy and resource usage (runtime, throughput, memory, and energy cost)?*

3 Experiments and Discussion

3.1 Hyperparameter Tuning and Test Set Construction

Because the Jena network provides a long archive (2011–2024), we perform full hyperparameter tuning on 2011–2022 and reserve 2023–2024 as unseen test years. In contrast, the AgWeatherNet stations provide only two years of data (March 2023–March 2025), which is insufficient for rolling cross-validation. For AgWeatherNet, we therefore use a fixed one-year calibration window (March 2023–March 2024) for training and hold out the final twelve months (March 2024–March 2025) as the test set.

Rolling-Origin Cross-Validation for Hyperparameters (Jena). To tune hyperparameters in a time-series-consistent way, we adopt a rolling-origin strategy inside the 2011–2022 training window. Rather than validating on every individual year (which would be both expensive and highly autocorrelated), we select four representative validation years spaced across the archive. Each fold trains on an expanding prefix and validates on the immediately following year.

For a given configuration θ, we compute the cross-validation MAE:

$$\mathrm{CV}(\theta) = \frac{1}{4}\sum_{k=1}^{4} \mathrm{MAE}^{(k)}(\theta), \tag{8}$$

where $\mathrm{MAE}^{(k)}(\theta)$ is the MAE on the validation year of fold k. The configuration $\theta^\star$ minimizing this score is selected. Each model is then retrained on the full 2011–2022 window using $\theta^\star$ before moving on to the test years.

Training Protocol for AgWeatherNet. Because AgWeatherNet provides only two years of data, cross-validation is not statistically meaningful. Hyperparameters for AWN are therefore kept fixed at the Jena-selected settings except for minor adjustments that depend on input dimensionality (e.g., channel subsets and scale differences). All AgWeatherNet models are trained on March 2023–March 2024 and evaluated on March 2024–March 2025.

Synthetic Outage Protocol. The raw archives for 2011–2024 contain too few natural outages for a controlled comparison across methods, so we inject synthetic gaps that mimic operational weather-station failures. The same protocol is used during cross-validation (on the validation year in each fold) and during final testing (on 2023–2024), and all models see identical masks for a given tower.

We divide each year into four seasons (winter: DJF, spring: MAM, summer: JJA, autumn: SON) and generate two types of outages:

- *Micro-gaps.* For each season s in year y, we draw the number of short outages $n_{y,s} \sim \mathrm{Poisson}(\lambda_{\mathrm{micro}} = 8)$. For each event, the duration $L \sim \mathrm{Gamma}(k{=}2, \theta{=}2\,\mathrm{h})$, which yields a median of ≈ 4 h and keeps 90 % of events below ≈ 9 h. Start times are drawn uniformly within the season, and events are constrained not to overlap.
- *Macro-gaps.* Each year receives three longer outages with a length $U \sim \mathrm{Uniform}[24, 72]$ h, representing maintenance or hardware replacement windows. These are placed to avoid overlap with micro-gaps.

Table 2: Imputation method and tuned hyperparameters.

Family/Model	Key hyperparameters
Statistical	
Multiple Linear Regression	N/A
ARIMA	Auto-p, d, q via AIC (max $p, q = 3$); re-fit monthly
Cubic Spline	Natural spline, shape $\alpha = 0.5$, knots at nearest obs.
Kalman Smoother	Local-level ETS; process noise $\sigma_w^2 = 0.15$, measurement $\sigma_v^2 = 0.05$ (learned by EM on 2011–2012)
Classic Machine Learning	
KNN Imputer	$k = 10$, inverse Euclidean weights;
RF	200 trees, max_depth 20, min_samples_leaf 10
SVR (lin/poly)	$C = 1, \epsilon = 0.1$, degree 3 for polynomial kernel
Deep Learning	
SDAE	3 layers 128–64–32, corruption drop 0.2, $\ell_2 = 10^{-5}$
GAIN	Generator 128–128, Discriminator 128; hint rate 0.9
Bi-LSTM+Attn	2 layers, 64 hidden, Bahdanau attention, dropout 0.1
TFT	$d_{\text{model}} = 64$, 4 heads, hidden 128, dropout 0.1

Table 3: Rolling-origin cross-validation used for hyperparameter tuning.

Fold	Training Period	Validation Period
1	2011–2014	2015
2	2011–2016	2017
3	2011–2019	2020
4	2011–2021	2022

For every selected outage interval, we blank the target channel(s).

Stress-Test Slices. Randomly placed outages provide a useful average-case benchmark, but the most demanding conditions for imputation often arise during extreme meteorological regimes or when hardware begins to degrade. To probe robustness in such settings, we define stress slices based on empirical quantiles of key variables, computed from the respective training periods of two datasets.

Let $q_p(X)$ denote the p-th quantile of variable X for a given tower. We define:

- *Temperature extremes:* $\mathcal{T}_{\text{ext}} = \{\, t : T_t > q_{0.80}(T) \text{ or } T_t < q_{0.20}(T) \,\}$, capturing heatwaves, cold-pool episodes, and strong inversions.
- *Multivariate extremes:* $\mathcal{X}_{\text{ext}} = \{\, t : \text{WS}_t > q_{0.80}(\text{WS}) \vee \text{RH}_t > q_{0.80}(\text{RH}) \vee \text{SWDR}_t > q_{0.80}(\text{SWDR}) \,\}$, representing high wind (turbulent mixing), high humidity (fog/saturation), or high shortwave radiation (midday forcing).
- *Sensor-stress regime (AgWeatherNet only):* $\mathcal{D}_{\text{stress}} = \{\, t : \text{Battery}_t < q_{0.20}(\text{Battery}) \vee \text{LoggerTemp}_t < q_{0.20}(\text{LoggerTemp}) \,\}$, covering periods when hardware is near thermal or power limits.

Unified Test Set and Evaluation. All evaluations use a single comprehensive test set per tower, where we intentionally mask the target temperature at both randomly sampled outage periods and at all predefined stress conditions.

Let $\mathcal{M}_{\text{rand}}$ denote the set of timestamps selected by the synthetic outage protocol (micro-gaps and macro-gaps) described above. Let $\mathcal{T}_{\text{ext}}$, $\mathcal{X}_{\text{ext}}$, and $\mathcal{D}_{\text{stress}}$ denote the sets of timestamps falling into temperature extremes, multivariate extremes, and (for AgWeatherNet) sensor-stress regimes as defined by the training-period quantiles, respectively. For the test years, we mask the target temperature at the union

$$\mathcal{M} = \mathcal{M}_{\text{rand}} \cup \mathcal{T}_{\text{ext}} \cup \mathcal{X}_{\text{ext}} \cup \mathcal{D}_{\text{stress}} \text{ (AgWeatherNet only)}, \quad (9)$$

while leaving all predictor variables intact. Thus, the final evaluation set includes both realistic random outages and deliberately constructed gaps at all extreme regimes.

For analysis, we report MAE and RMSE on: *(a)* the full masked test set $\mathcal{M}$ (*All Outages*), *(b)* the subset of masked timestamps in temperature extremes $\mathcal{T}_{\text{ext}}$ (*Temp-Extreme*), *(c)* the subset in multivariate extremes $\mathcal{X}_{\text{ext}}$ (*Multi-Extreme*), and *(d)* for AgWeatherNet, the subset in sensor-stress conditions $\mathcal{D}_{\text{stress}}$ (*Sensor-Stress*). By construction, the "All Outages" metric summarizes overall performance across both random and stress-driven gaps, while the three slices isolate how much accuracy degrades when the imputer is forced to reconstruct the most challenging points (e.g., heatwaves, high winds, or high-radiation periods, and degraded hardware states).

3.2 Accuracy and Robustness Across Datasets

With hyperparameters fixed via the rolling-origin procedure (Sec. 3.1), we now evaluate all imputers on the unified 2023–2024 test sets for both networks: the three 10-minute-resolution towers near Jena and the two 5-minute-resolution AgWeatherNet towers. Each model is trained strictly within its own network—2011–2022 for Jena, and March 2023–March 2024 for AWN—so no cross-site information is shared. This design isolates performance under distinct climates, sensor hardware, and archive lengths.

Table 4 reports results for the WS Saaleaue tower, while Table 5 summarizes performance for the Sunnyside-In AWN station. The

Table 4: Jena (WS Saaleaue): Accuracy on unified 2023–2024 test set. Values shown as MAE/RMSE (°C).

Class	Model	All	T-Ext	X-Ext
Stat.	MLR	1.56/1.67	1.81/1.98	2.06/2.18
	ARIMA	2.31/2.44	2.65/2.79	2.80/2.94
	Kalman	2.44/2.68	2.66/2.81	2.81/2.97
	Spline	2.80/2.98	2.98/3.16	3.17/3.32
ML	KNN	0.82/0.94	1.06/1.20	1.22/1.33
	RF	**0.46/0.57**	**0.71/0.84**	**0.82/0.93**
	SVM-lin	1.21/1.34	1.56/1.67	1.70/1.91
	SVM-poly	1.04/1.18	1.39/1.51	1.53/1.66
DL	TFT	0.68/0.79	0.92/1.06	1.15/1.26
	BiLSTM+Att	0.76/0.82	1.07/1.18	1.29/1.42
	SDAE	0.88/0.95	1.18/1.30	1.38/1.53
	GAIN	0.93/1.02	1.31/1.45	1.52/1.69

Table 5: AgWeatherNet (Sunnyside-In tower): imputation accuracy on the unified test set. Values shown as MAE / RMSE (°C).

Class	Model	All	T-Ext	S-Stress	X-Ext
Stat.	MLR	1.80/1.95	2.11/2.33	2.24/2.40	2.36/2.55
	ARIMA	2.42/2.55	2.72/2.93	2.82/3.08	2.94/3.16
	Kalman	2.56/2.70	2.79/3.03	2.91/3.14	3.06/3.24
	Spline	2.86/3.04	3.11/3.32	3.19/3.42	3.36/3.55
ML	KNN	1.10/1.22	1.35/1.50	1.46/1.60	1.60/1.76
	RF	**0.72/0.84**	**0.90/1.03**	**0.98/1.10**	**1.08/1.22**
	SVM-lin	1.40/1.55	1.68/1.84	1.78/1.94	1.92/2.10
	SVM-poly	1.30/1.44	1.58/1.74	1.68/1.84	1.82/1.98
DL	TFT	0.88/1.00	1.08/1.22	1.18/1.32	1.30/1.46
	BiLSTM+Att	0.94/1.06	1.15/1.30	1.26/1.40	1.38/1.54
	SDAE	1.05/1.18	1.28/1.42	1.38/1.52	1.50/1.66
	GAIN	1.22/1.36	1.49/1.64	1.60/1.75	1.74/1.90

results of other towers are qualitatively similar, but are excluded for brevity.

Key Observations Across Jena and AgWeatherNet. Across both datasets, several consistent patterns emerge:

- ***RF remains the top performer.*** In Jena, RF attains an MAE of 0.46°C on the unified test set, rising only modestly under temperature (0.71°C) and multivariate extremes (0.82°C). Despite having barely two years of training data and a more volatile semi-arid climate, RF is still the most accurate model in AWN (MAE 0.72°C), preserving a clear margin over all alternatives.
- ***Deep sequence models outperform statistical baselines, but are less robust.*** TFT and BiLSTM+Attention consistently improve over all statistical models, yet remain 0.20–0.30°C behind RF on every slice. Their degradation under stress is sharper: for example, in AWN, the TFT error grows from 0.88°C (All) to 1.30°C (X-Ext), reflecting sensitivity to sparse tail distributions in wind, humidity, and radiation.
- ***KNN is a viable low-compute baseline but struggles in extremes.*** KNN performs reasonably in both networks and is attractive for edge deployment. However, its neighbour-based structure makes it vulnerable when temperature or radiation lies in the upper tail, where historical analogues are scarce. This leads to larger degradation in the X-Ext slices.
- **Multivariate extremes (X-Ext) are the most difficult regime.** Across all towers and methods, X-Ext yields the largest errors. These conditions—high wind, high humidity, or high solar radiation—correspond to rapid transitions in temperature tendencies, reducing the predictive value of diurnal and autoregressive structure. Sensor-stress periods in AWN also degrade performance, but less severely than X-Ext.

Summary. Despite differences in climate, hardware, sampling frequency, and archive length, the relative ordering of models is remarkably stable:

$$\text{RF} \gg \text{TFT} > \text{BiLSTM} > \text{KNN} > \text{SDAE} > \text{GAIN},$$

and this hierarchy persists even when we force models to reconstruct the hardest conditions in the record. A couple of reasons stand out why RF outperforms all other models in Table 4 and Table 5, even the powerful deep learning models. RFs exploit strong, low-order nonlinearities [9, 19] (for example, interactions between hour-of-day × day-of-year × relative humidity). On the other hand, deep learning sequence models use much of their representational capacity to model multi-day dynamics that are rarely needed when the outage times are less than a day [4]. Secondly, RF averages many weak learners, so a few bad data anomalies or spiky sensor noise do not propagate [9]. However, with deep learning models, even with dropout, there are chances of outliers amplifying during back-prop and may need careful clipping [23]. Third, it is also important to note that RF did not require an extensive hyperparameter search. The performance saturated once the ensemble reached about 200 trees with depth ≤ 20. In contrast, deep learning methods typically expose 5–10 free parameters (learning-rate schedules, layer widths, attention heads, dropout rates, etc.). Given a finite GPU budget, the hyperparameter search grid had to be limited. Hence, Tables 4 and 5 reflect not only differences in model capacity but also how efficiently each method can be tuned under realistic resource constraints.

Finally, we note that in this section each imputer operates per tower independently, using only local features. Spatio-temporal embeddings that explicitly couple towers are introduced and evaluated separately in Sec. 3.4.

3.3 Runtime, Training Cost, and Energy Footprint

Beyond accuracy, an operational imputation system must be fast, memory-efficient, and inexpensive to run at scale. In this section, we treat imputation as a performance-engineering problem and benchmark both *(a)* the *training-time* cost of fitting each model and *(b)* the *inference-time* cost of deploying it to fill gaps in operational streams. We report runtime, throughput, memory usage, and energy/monetary footprint for representative models from each family. We focus on WS Saaleaue as the primary Jena tower; the qualitative conclusions are similar for the other towers and for AgWeatherNet.

Hardware and Measurement Protocol. All measurements are collected on a cloud instance comparable to a Google Colab runtime: two virtual CPU cores at 2.3 GHz, 13 GB of host RAM, and one NVIDIA T4 GPU with 16 GB GDDR6 memory and a 70 W board power limit. Statistical and classical machine-learning methods (MLR, ARIMA, Kalman, Spline, KNN, SVM, RF) are executed on CPU with the GPU disabled. Deep learning models (SDAE, GAIN, BiLSTM+Attention, TFT) use the T4 for both training and inference.

For each tower, we consider the unified test mask M defined in Sec. 3.1, which contains both synthetic outages and stress slices. For WS Saaleaue, this corresponds to a masked subset on the order of $|M| \sim 10^4$ timestamps (i.e., a constant fraction of the two-year 10-minute timeline). For each model m, we measure

- the total *training* wall-clock time required to fit the model on the training window (2011–2022 for Jena), and
- the *inference* wall-clock time required to impute all points in M for the 2023–2024 test years.

Each inference measurement is preceded by a warm-up pass on a separate batch (10,000 rows) whose timings are discarded. We repeat inference measurements five times and report the mean; run-to-run variation is below 5% for all models.

Inference Metrics. Let $T_{\text{infer}}^{(m)}$ denote the measured wall-clock time (in seconds) required for model m to impute $N = |M|$ masked temperatures in the test set. We normalize this to a per-1000 sample inference time

$$t_{1000}^{(m)} = \frac{T_{\text{infer}}^{(m)}}{N} \cdot 1000 \quad \text{[seconds/1000 records]}. \tag{10}$$

The corresponding throughput (records per second) is

$$Q^{(m)} = \frac{1000}{t_{1000}^{(m)}} \quad \text{[records/s]}. \tag{11}$$

For energy, we distinguish between CPU- and GPU-backed models. During each timed run, we sample device power once per second: for CPU-backed models, we read the CPU package power via RAPL; for deep models we query the T4 board power via `nvidia-smi`. Let $\bar{P}^{(m)}$ denote the resulting average device power in watts. The energy required to impute 1000 records is then

$$E_{1000}^{(m)} = \frac{\bar{P}^{(m)}\, t_{1000}^{(m)}}{3600} \quad \text{[Wh/1000 records]}. \tag{12}$$

In our setup, CPU-only methods have $\bar{P}^{(m)}$ on the order of 20–30 W, while T4-backed models have $\bar{P}^{(m)}$ on the order of 70 W under inference load.

Monetary cost for inference is derived from a realistic cloud pricing. We assume an on-demand general-purpose CPU VM at a rate $c_{\text{CPU}} \approx \$0.11$/h (roughly a small 2-vCPU instance) and a T4 GPU at $c_{\text{T4}} \approx \$0.35$/h. CPU-only models are charged at c_{CPU}, while GPU-backed models incur the combined rate $c_{\text{CPU+T4}} = c_{\text{CPU}} + c_{\text{T4}} \approx \$0.46$/h. The effective inference cost per 10^6 imputations is

$$\text{Cost}_{10^6,\text{infer}}^{(m)} = \frac{10^6}{1000} \cdot \frac{t_{1000}^{(m)}}{3600} \cdot c^{(m)} \quad [\$/10^6 \text{ records}], \tag{13}$$

where $c^{(m)} = c_{\text{CPU}}$ for CPU-only models and $c^{(m)} = c_{\text{CPU+T4}}$ for GPU-backed models.

Training Metrics. Let $T_{\text{train}}^{(m)}$ denote the total wall-clock time required to train model m on the Jena training window (2011–2022). This captures everything from reading the training matrix to the final fitted parameters (or checkpoint). As a coarse summary of training-time energy, we use

$$E_{\text{train}}^{(m)} = \frac{\bar{P}_{\text{train}}^{(m)}\, T_{\text{train}}^{(m)}}{3600} \quad \text{[Wh]}, \tag{14}$$

where $\bar{P}_{\text{train}}^{(m)}$ is the average device power during training (on the order of 20–30 W for CPU-only models and $\approx$ 100 W for GPU-backed models when CPU and T4 are both active).

The one-time monetary cost of training is

$$\text{Cost}_{\text{train}}^{(m)} = \frac{T_{\text{train}}^{(m)}}{3600} \cdot c^{(m)} \quad [\$], \tag{15}$$

with the same $c^{(m)}$ convention as in Eq. (13).

Representative Models. To keep the discussion focused, we report both training and inference cost metrics for one representative from each family: Cubic Spline (statistical baseline), KNN Imputer and RF (classical machine learning), and three deep learners (SDAE, BiLSTM+Attention, TFT). Accuracy for these models on the Jena tower is summarized in Table 4; here we reproduce the "All Outages" MAE so that the cost can be visually compared to the error.

Table 6 reports inference-time metrics for imputing the WS Saaleaue test set, while Table 7 summarizes training-time cost on the 2011–2022 Jena archive. Energy values are reported in milliwatt-hours (mWh) per 1000 imputed records for inference, and in Wh for training. All values are rounded and intended to reflect order-of-magnitude behaviour; exact numbers will vary across implementations and hyperparameter choices.

3.3.1 Discussion. Several trends stand out. First, classical ML models offer an extremely favorable *inference-time* accuracy–cost trade-off. RF achieves the lowest MAE on WS Saaleaue (0.46°C) while imputing 1000 records in roughly 2×10^{-2} s. This corresponds to a throughput on the order of 5×10^4 records per second and an energy usage of only ≈ 0.14 mWh per 1000 records. On a modest CPU-only instance, one million imputations from RF cost on the order of 10^{-4} dollars.

Second, deep learners are substantially more expensive per imputation. The Temporal Fusion Transformer is roughly an order of magnitude slower per 1000 records than RF and consumes about 40–50× the energy per 1000 records. Consequently, its effective inference cost per million imputations is $\approx$ 60× higher (on the order of $\$4 \times 10^{-2}$ versus $\$6 \times 10^{-4}$). BiLSTM+Attention and SDAE sit in between: they reduce error relative to purely statistical baselines but remain slower and more power-hungry than RF on this workload.

Third, *training-time* costs show a similar pattern but with different stakes. Training RF on twelve years of Jena data takes on the order of a few minutes and costs only a few thousandths of a dollar on this hardware, whereas TFT training can take roughly an hour and a half per tower, and costs on the order of tens of cents. On a single tower, this difference is modest, but for regional networks with dozens of towers and frequent retraining, deep models can accumulate substantial GPU-hours.

Finally, memory usage is modest for all methods. On our hardware, RF and KNN each fit comfortably within 1 GB of

Table 6: Inference-time runtime, throughput, and energy/cost footprint of representative imputers on the WS Saaleaue test set ($|M|$ on the order of 10^4 masked timestamps). MAE is the "All Outages" metric from Table 4. Throughput $Q^{(m)}$ and energy $E^{(m)}_{1000}$ are derived via Eq. (11)–(12), inference cost via Eq. (13).

Model	MAE [°C]	Time/1k [s]	Throughput [rec/s]	Energy/1k [mWh]	Cost/10^6 [$]
Cubic Spline (CPU)	2.80	0.01	1.0×10^5	0.07	0.0003
KNN Imputer (CPU)	0.82	0.20	5.0×10^3	1.39	0.0061
RF (CPU)	**0.46**	0.02	5.0×10^4	0.14	0.0006
SDAE (T4 GPU)	0.88	0.10	1.0×10^4	1.94	0.0128
BiLSTM+Attn (T4 GPU)	0.76	0.15	6.7×10^3	2.92	0.0192
TFT (T4 GPU)	0.68	0.30	3.3×10^3	5.83	0.0383

Table 7: Training-time wall-clock cost on the Jena training window (2011–2022) for representative imputers. Values reflect the order of magnitude observed on a 2-vCPU + T4 instance: classical models train in seconds to a few minutes, whereas deep sequence models can require tens of minutes to about an hour per tower.

Model	Train Time [min]	Train Energy [Wh]	Train Cost [$]
Cubic Spline (CPU)	≈ 0.1	≈ 0.04	≈ 0.0002
KNN Imputer (CPU)	≈ 0.3	≈ 0.13	≈ 0.0006
RF (CPU)	≈ 7	≈ 2.9	≈ 0.013
SDAE (T4 GPU)	≈ 20	≈ 33	≈ 0.15
BiLSTM+Attn (T4 GPU)	≈ 40	≈ 67	≈ 0.31
TFT (T4 GPU)	≈ 90	≈ 150	≈ 0.69

host RAM. Among deep models, BiLSTM+Attention peaks at approximately 2 GB of GPU memory, while TFT is the largest with roughly 4 GB—well below the 16 GB available on the T4. Thus, the memory is not a limiting factor for any of the models studied here; the dominant differentiators are latency, training time, and energy per imputed sample.

Taken together, these measurements reinforce the conclusion of Sec. 3.2: in this setting, a well-tuned RF provides the best overall compromise between accuracy, runtime, and compute cost. Deep sequence models do offer respectable accuracy, but their higher training and inference footprint make them harder to justify for large-scale.

3.4 Ablation: Effect of Spatio-Temporal Embeddings (GraphSAGE-GRU)

3.4.1 Training and Feature Construction. To test whether explicit spatio-temporal structure improves imputation, we add the GraphSAGE-GRU encoder from Sec. 2.2 on top of the multi-tower data. We train one encoder per network: three towers for Jena and two towers for AgWeatherNet. Each encoder takes aligned tower features on the 10-minute grid, applies a single GraphSAGE aggregation layer, and then a GRU with hidden size $d = 32$ over a short history window (e.g., the last $L = 12$ timestamps).

The encoder is trained in a self-supervised next-step prediction setup (predict T_{t+1} from the previous L steps) using mean squared error and Adam (10^{-3}). After convergence we freeze the weights and generate a 32-dimensional embedding $z_{t,i}$ for every tower i and timestamp t in the training and test windows. These embeddings are then concatenated to the original tabular features as in Eq. (7),

$$X^{\text{extended}}_{t,i} = \left[X^{\text{original}}_{t,i} \parallel z_{t,i}\right],$$

Table 8: Effect of GraphSAGE-GRU embeddings on imputation accuracy. ΔMAE=MAE(+Emb)−MAE(Base); negative is better.

Dataset	Model	MAE Base	MAE+Emb	ΔMAE
Jena	RF	0.46	0.39	−0.07
(WS Saaleaue)	SDAE	0.88	0.83	−0.05
	BiLSTM+Attn	0.76	0.72	−0.04
	TFT	0.68	0.60	−0.08
AWN	RF	0.72	0.70	−0.02
(Sunnyside-In)	SDAE	1.05	1.02	−0.03
	BiLSTM+Attn	0.94	0.90	−0.04
	TFT	0.88	0.83	−0.05

and the downstream imputers (RF, SDAE, BiLSTM+Attn, TFT) are retrained with no other changes. In all ablations, the encoder is treated as a one-time preprocessing step: once embeddings are materialized, later experiments reuse them without retraining GraphSAGE-GRU.

3.4.2 Accuracy Gains. Table 8 compares baseline models (local features only) with their embedding-augmented variants on the unified "All Outages" MAE.

On both networks, all four models benefit slightly from embeddings. These improvements are real but small compared to the gap between statistical baselines and RF / deep methods in Tables 4 and 5.

3.4.3 Performance Cost. The extra accuracy comes with additional cost from *(a)* GraphSAGE-GRU pretraining, *(b)* storing embeddings,

Table 9: Approximate GraphSAGE-GRU pretraining cost vs. MAE improvement. "Cost per 0.01°C" is Pretrain Cost/(|Δ MAE|/0.01). Inference overhead from the larger feature vectors is small compared to the one-off pretraining cost.

Dataset	Model	ΔMAE [°C]	Pretrain Cost [$]	Cost per 0.01°C [$]
Jena	RF	−0.07	≈ 0.20	≈ 0.03
	TFT	−0.08	≈ 0.20	≈ 0.03
AWN	RF	−0.02	≈ 0.09	≈ 0.05
	TFT	−0.05	≈ 0.09	≈ 0.02

and *(c)* a small increase in inference time due to higher feature dimensionality.

Pretraining and Storage. On Jena, training GraphSAGE-GRU for ≈ 15 epochs on a T4 takes on the order of 30 minutes (∼ 0.5 GPU-hours); on AWN, the shorter archive converges in ∼ 10–15 minutes. At the CPU+T4 rate of ≈ $0.46/h (Sec. 3.3), this corresponds to a one-off cost of roughly $0.20 (Jena) and $0.09 (AWN). Storing 32-float embeddings for all timestamps adds $O(10^8)$ floats across both networks, i.e., a few hundred megabytes—minor on our cloud instance but non-trivial for low-memory edge nodes.

Inference Overhead and Cost-per-Improvement. Assuming embeddings are precomputed, inference overhead is modest. For RF, Time/1k increases from ≈ 0.02 s to ≈ 0.023 s; for deep models, the increase is about 10–15% in Time/1k. The dominant cost is therefore the one-time encoder training, which can be amortized over all models that consume the embeddings.

Table 9 summarizes an effective cost-per-improvement by dividing the pretraining cost by the observed MAE reduction. We include RF and TFT as representatives for classical and deep models; SDAE and BiLSTM+Attn fall in the same range because they share the same encoder.

Even on this small benchmark, the cost per 0.01°C of MAE improvement is on the order of a few cents. For larger archives or many towers, total GPU-hours for spatio-temporal pretraining can therefore grow quickly.

3.4.4 Discussion. Overall, the ablation shows a clear but modest pattern: GraphSAGE-GRU embeddings yield *small* accuracy gains (0.01–0.03°C) at a *non-trivial* additional cost (extra GPU training, extra storage, slight inference slowdown). For our main use case—gap-filling for operational farm networks where RF already delivers sub-degree MAE with excellent CPU-only performance—this trade-off is hard to justify, especially on edge deployments.

We see the embeddings as most defensible when *(a)* the station network is very sparse, *(b)* key predictors are missing at some towers, or *(c)* the downstream task is extremely sensitive to tail errors. In typical agricultural weather-imputation workloads, however, a well-tuned RF with purely local features remains the simplest and most cost-effective option.

4 Related Work

Early weather-station archives were often patched using climatological normals or long-term averages [24, 42] or simple persistence rules that carry the last valid observation forward [22]. Classical state-space approaches, including Kalman filters [20, 31] and spline-based smoothers [26], have proved effective for smooth, low-frequency gaps in hourly or daily data. However, these techniques can struggle with the non-stationarity and sub-day spikes typical of real field deployments [22, 27].

Spatial interpolators can exploit neighbouring stations when a dense network exists. KNN schemes [15, 39] and related spatial methods have been used to fill daily meteorological fields, while regression on recent extrema or radiation inputs can impute daily $T_{\max}$ and $T_{\min}$ and growing-degree-day accumulations for crop models [2, 3, 5, 18, 35]. Yet accuracy typically degrades when outages exceed one to two days, or when terrain and land use create sharp microclimatic gradients [11, 32, 38].

On the machine learning side, MissForest [37] popularised Random-Forest-based imputation on mixed-type tabular data and has since been applied to the environmental and atmospheric series. More recently, Boomgard-Zagrodnik and Brown [8] used a network-scale RF to fill 15-minute temperature gaps across a mesonet, demonstrating that tree ensembles can capture local feature interactions and seasonal structure in dense agricultural networks. However, these studies typically focus on a single model family and do not compare RFs against modern deep architectures under a shared, long-term benchmark.

Deep-learning imputers have progressed rapidly in other domains. GRU- and LSTM-based models for sequences with missing values [12, 13], variational-autoencoder approaches for incomplete heterogeneous data [34], and GAN-based imputers such as GAIN [43] report strong results on clinical, sensor, and finance benchmarks. TFT [10] further combines attention, gating, and static covariates for multi-horizon forecasting and has been adopted as a powerful baseline for time-series tasks. Graph-based and diffusion-style imputers extend these ideas to spatio-temporal networks, learning over sensor or road graphs [17, 21]. Most such evaluations rely on hourly or daily data, synthetic MCAR masks, and standard machine learning metrics such as MSE, MAE, or RMSE. They rarely incorporate realistic outage patterns calibrated to sensor failures or measure runtime, throughput, and energy cost. In addition, large-scale cross-family comparisons remain scarce. Classical time-series work has focused on forecasting rather than imputation per se, and often uses cross-validation protocols that are not fully aligned with operational deployment constraints [6, 30].

Uniqueness of Our Work. To our knowledge, there is no prior study that jointly:

(1) explores statistical, classical machine learning, and deep models on 10-minute meteorological records;
(2) uses outage masks and stress slices calibrated to realistic station-failure behavior; and
(3) evaluates both accuracy and systems-level metrics such as runtime, throughput, and energy/monetary cost on a common hardware platform.

Besides, no prior study isolates the marginal benefit of augmenting tabular imputers with lightweight spatio-temporal embeddings.

This research complements the existing literature by providing a head-to-head benchmark across 12 imputation methods on 2 independent agricultural weather networks. Unlike prior work,

we: *(a)* operate under outage masks designed to mimic real sensor-failure statistics; *(b)* report accuracy alongside training time, inference latency, throughput, energy footprint, and monetary cost; and *(c)* introduce a lightweight spatio-temporal encoder whose embeddings are reused across models, allowing us to quantify the marginal accuracy gains versus the additional GPU budget. Our results show that a CPU-only RF model often matches or outperforms complex architectures while using one to two orders of magnitude less computation, and that spatio-temporal embeddings provide measurable but incremental improvements at a small, but non-zero, additional cost.

5 Discussion

Our results over two networks are encouraging, but open issues remain. It is unclear whether the same model ranking and cost profile will persist in denser mesonets, mountainous terrain, strongly maritime regimes, or for other key variables (e.g., humidity, wind, radiation) that feed directly into crop and boundary-layer models [11, 32, 38].

Another issue concerns the design of the model and hyperparameters. Deep learners expose many tunable components, and our protocol is intentionally modest, reflecting the GPU budget of an operational group rather than a large-scale autoML effort. More aggressive searches—such as Bayesian optimization or bandit-style configuration selection—could narrow the accuracy gap relative to RF, as seen in sequence-model benchmarks [4, 23], but would require additional GPU-hours and thus higher energy and monetary cost.

The GraphSAGE–GRU ablation illustrates a similar trade-off. For our two relatively small networks, the overhead of spatial–temporal embeddings is modest and can be amortized across multiple models, but the cost-per-0.01°C improvement is non-negligible. Such embeddings are therefore most appropriate when spatial correlations are stronger (e.g., dense networks or complex terrain), when key predictors are missing at some towers, or when downstream tasks are susceptible to tail errors.

Operational networks must support streaming telemetry, multi-tenant workloads, and heterogeneous hardware ranging from low-power edge loggers to regional data centers. Extending this work to larger end-to-end pipelines is a natural next step. From a performance-engineering standpoint, our results suggest a practical guideline: begin with a tuned, CPU-only RF as the default imputer, and adopt deep or graph-based architectures only when their additional accuracy demonstrably improves agronomic decisions or justifies the extra compute budget.

6 Conclusion

This paper presents a performance-oriented evaluation imputation methods on two agricultural weather networks. Across a long temperate archive (Jena) and a shorter semi-arid network (AgWeatherNet), a tuned CPU-only RF consistently offered the best trade-off, achieving the lowest imputation error under realistic outage masks and stress slices while maintaining low latency, high throughput, and negligible inference cost. Deep models such as BiLSTM+Attention and TFT provided at most modest accuracy gains over simpler baselines but required one to two orders of magnitude more training and inference compute. GraphSAGE–GRU embeddings yielded additional MAE reductions of only a few hundredths of a degree at the cost of extra GPU pretraining and storage, offering real but incremental benefits over a purely local RF. Our results point to a pragmatic design rule for operational weather-data systems: *use a well-regularized, CPU-only tree ensemble as the default imputer, and deploy deep or graph-based architectures only when their additional accuracy can be justified by the extra computational, energy, and monetary budget* required for downstream agriculture decision-making tasks.

Acknowledgments

This research is partly supported by the U.S. National Science Foundation Award 2345653. Any findings, opinions, recommendations, or conclusions expressed in the paper are those of the authors and do not necessarily reflect the sponsor's views.

References

[1] Walter Acevedo and Carlos Fitzjarrald. 2003. The Influence of Cloud Cover on the Wintertime Diurnal Temperature Cycle. *Atmospheric Research* 66, 3 (2003), 203–222. doi:10.1016/S0169-8095(03)00058-6

[2] Bill Acock and Y. Pachepsky. 2000. Imputing Daily Maximum and Minimum Air Temperatures. *Agricultural and Forest Meteorology* 104 (2000), 59–73.

[3] R. G. Allen, L. S. Pereira, D. Raes, and M. Smith. 1998. *Crop Evapotranspiration: Guidelines for Computing Crop Water Requirements (FAO Irrigation and Drainage Paper 56).* Food and Agriculture Organization of the United Nations, Rome.

[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv:1803.01271* (2018).

[5] G. L. Baskerville and P. Emin. 1969. Rapid Estimation of Heat Accumulation from Maximum and Minimum Temperatures. *Ecology* 50, 3 (1969), 514–517. doi:10.2307/1933912

[6] Christoph Bergmeir and Rob J. Hyndman. 2018. A Note on the M4 Competition. *International Journal of Forecasting* 34 (2018), 1102–1107.

[7] Christopher H. Bock and Dan E. Auer. 2010. Hourly Leaf Wetness Duration Thresholds for Predicting Strawberry Powdery Mildew. *Plant Disease* 94, 6 (2010), 745–750. doi:10.1094/PDIS-94-6-0745

[8] Joseph P. Boomgard-Zagrodnik and David J. Brown. 2023. Machine Learning Imputation of Missing Mesonet Temperature Observations. *Agricultural and Forest Meteorology* 331 (2023), 109394. doi:10.1016/j.agrformet.2023.109394

[9] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. doi:10.1023/A:1010933404324

[10] Sercan O. Arık Bryan Lim, Nicolas Loeff, and Tomas Pfister. 2021. Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting. In *International Journal of Forecasting*, Vol. 37. 1748–1764. doi:10.1016/j.ijforecast.2020.11.001

[11] Gaylon S. Campbell and John M. Norman. 2012. *An Introduction to Environmental Biophysics* (3 ed.). Springer, New York.

[12] Wenjie Cao, Dong Wang, Jian Li, and Hongning Wang. 2018. BRITS: Bidirectional Recurrent Imputation for Time Series. In *Advances in Neural Information Processing Systems (NeurIPS).* 6776–6786.

[13] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent Neural Networks for Multivariate Time Series With Missing Values. *Scientific Reports* 8, 1 (2018).

[14] Lijun Chen, Zheng ya He, Peilin Zhao, Wei Cao, Lei Leong Cheong, and Philip S. Yu. 2022. SAITS: Self-Attention-based Imputation for Time Series. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[15] Qiong Chen and Meian Liu. 2015. A K-Nearest Neighbour Approach to Spatial–Temporal Gap Filling for Daily Meteorological Data. *Environmental Modelling & Software* 69 (2015), 77–88. doi:10.1016/j.envsoft.2015.03.005

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078* (2014).

[17] Andrea Cini and Federico Bonchi. 2022. GRIN: Graph Recurrent Imputation Networks for Multivariate Time Series. *ICLR* (2022).

[18] Robert W. Coates, Bruce A. King, and Allen Simmons. 2013. Smart-ET: A Web-Based Evapotranspiration Scheduling Tool for Precision Irrigation. *Computers and Electronics in Agriculture* 96 (2013), 13–22. doi:10.1016/j.compag.2013.04.012

[19] D. Richard Cutler, Thomas C. Edwards, and Kurt H. Beard. 2007. Random Forests for Classification in Ecology. *Ecology* 88, 11 (2007), 2783–2792. doi:10.1890/07-

0539.1
[20] James Durbin and Siem Jan Koopman. 2012. *Time Series Analysis by State Space Methods* (2 ed.). Oxford University Press.
[21] Zonghan Wu et al. 2023. Diffusion Imputation for Spatiotemporal Data. *arXiv:2305.05540* (2023).
[22] Christopher A. Fiebrich, Gary R. Hall, J. D. Illston, and Michael B. C. Stambaugh. 2006. Quality Assurance Procedures in the Oklahoma Mesonet. *Journal of Atmospheric and Oceanic Technology* 23, 7 (2006), 1002–1015. doi:10.1175/JTECH1909.1
[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. See Chapter 7: "Regularization and Optimization" for clipping/outlier effects.
[24] Nathan B. Guttman and Robert G. Quayle. 1990. A Historical Perspective of U.S. Climate Normals: 1895–1984. *Monthly Weather Review* 118, 3 (1990), 176–187. doi:10.1175/1520-0493(1990)118<0176:AHPOUC>2.0.CO;2
[25] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.
[26] Michael Hauschild and Jochen Fröhlich. 2019. Spline-Based Gap Filling of High-Resolution Meteorological Time Series. *Theoretical and Applied Climatology* 136 (2019), 613–629. doi:10.1007/s00704-018-2496-9
[27] Kenneth G. Hubbard and Xiaomao Lin. 2005. Radiation-Shield and Temperature Measurement Errors during the Warm Season. *Journal of Atmospheric and Oceanic Technology* 22, 11 (2005), 1576–1581. doi:10.1175/JTECH1803.1
[28] Robert A. Houze Jr. 1993. *Cloud Dynamics*. Academic Press, San Diego.
[29] Olaf Kolle. 2008. Documentation of the weather station on top of the roof of the institute building of the max-planck-institute for biogeochemistry. *Recuperado de https://www. bgc-jena. mpg. de/wetter* (2008).
[30] Roderick J. A. Little and Donald B. Rubin. 2002. *Statistical Analysis with Missing Data* (2 ed.). John Wiley & Sons, Hoboken, NJ.
[31] David W. Meek and John L. Hatfield. 1994. Data Quality Checking for Weather Station Time Series Using Kalman Filtering. *Agronomy Journal* 86, 6 (1994), 1133–1139. doi:10.2134/agronj1994.00021962008600060003x
[32] John L. Monteith and Mike H. Unsworth. 2013. *Principles of Environmental Physics* (4 ed.). Academic Press, Amsterdam.
[33] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*.
[34] Aritz Nazábal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. 2020. Handling Incomplete Heterogeneous Data using VAEs. *Pattern Recognition* (2020).
[35] L. C. Newton and M. R. Goodwin. 2017. Daily Maximum and Minimum Air-Temperature Estimation from Sparse Hourly Observations. *Agricultural and Forest Meteorology* 234 (2017), 182–194. doi:10.1016/j.agrformet.2016.12.012
[36] Richard L. Snyder and Joaquim de Melo-Abreu. 2005. *Frost Protection: Fundamentals, Practice, and Economics*. Food and Agriculture Organization of the United Nations, Rome, Italy.
[37] Daniel J. Stekhoven and Peter Bühlmann. 2012. MissForest—Non-Parametric Missing Value Imputation for Mixed-Type Data. *Bioinformatics* 28, 1 (2012), 112–118. doi:10.1093/bioinformatics/btr597
[38] Roland B. Stull. 1988. *An Introduction to Boundary Layer Meteorology*. Springer, Dordrecht.
[39] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Patrick Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Ross B. Altman. 2001. Missing Value Estimation Methods for DNA Microarrays. *Bioinformatics* 17, 6 (2001), 520–525.
[40] Russell S. Vose, David R. Easterling, and Kyung-Ja Lee. 2005. Diurnal Temperature Range Over Global Land Areas: An Update. *Journal of Climate* 18, 8 (2005), 4910–4929. doi:10.1175/JCLI3588.1
[41] Washington State University. [n. d.]. WSU AgWeatherNet (AWN). https://weather.wsu.edu/.
[42] World Meteorological Organization. 2018. *Guide to Climatological Practices* (4th ed.). Number WMO-No. 100. WMO, Geneva, Switzerland. https://library.wmo.int/doc_num.php?explnum_id=10441
[43] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing Data Imputation Using Generative Adversarial Nets. In *International Conference on Machine Learning (ICML)*. 5689–5698.