Embedding Techniques for Modeling Ambiguous Time Series in River Water Networks*

Benjamin Fankhauser[1][0000−0002−7982−2669], Vidushi Bigler[2][0000−0001−6043−8264], and Kaspar Riesen[1][0000−0002−9145−3157]

[1] Institute of Computer Science, University of Bern, Bern, Switzerland
{benjamin.fankhauser,kaspar.riesen}@unibe.ch
[2] Institute for Optimisation and Data Analysis, Bern University of Applied Sciences, Biel, Switzerland
vidushi.bigler@bfh.ch

Abstract. Increasing river water temperature poses a growing threat to freshwater ecosystems, creating a demand for reliable and adaptable predictive models. The underlying time series problem is ambiguous, featuring non-linear effects such as snowmelt, groundwater inflow, and anthropogenic influences. In this work, we focus on systematically understanding the role of embedding techniques in this setting. We evaluate three embedding approaches – Concatenation, Gated Memory, and α-Interpolation – within LSTM-based architectures and assess their influence on predictive performance and robustness. Our results demonstrate that embeddings play a crucial role in capturing node-specific characteristics and significantly affect model behavior. While simple concatenation provides a strong baseline, both Gated Memory and α-Interpolation yield modest but consistent improvements. Moreover, the results indicate that Gated Memory embeddings are particularly beneficial in low-data regimes with fewer sensor nodes, which is highly relevant for hydrological datasets with limited temporal coverage. In an ablation study we examine that the inclusion of spatial coordinates reinforces the importance of structured input representations. Last but not least, we demonstrate that the observed effects generalize beyond LSTMs by transferring the embedding strategies to transformer-based models, where they similarly improve consistency and robustness.

Keywords: River Water Temperature Prediction · Time Series Forecasting · LSTM · Embeddings · Deep Learning · Climate Change

1 Introduction and Related Work

River water temperature is tightly coupled to atmospheric conditions through energy exchange processes, making it especially sensitive to changes in climate

* Supported by Swiss National Science Foundation (SNSF) Grant Nr. PT00P2_206252. Data are kindly provided by the Federal Office for the Environment.

variables. Ongoing climate change alters air temperature and drives sustained increases in river water temperature [1]. In Switzerland, for example, river temperatures have shown a steady rise over the past four decades [2]. This trend threatens aquatic ecosystems, as warmer water holds less dissolved oxygen and accelerates organismal metabolism [3]. Prolonged exposure to such conditions can degrade water quality [4], foster harmful algal blooms, and, in severe cases, trigger ecosystem collapse [5, 6]. These challenges are compounded by the complex physical processes governing river temperature dynamics.

River temperature modeling is a challenging pattern recognition problem, as several effects interplay in non-linear ways [7]: Solar radiation and thermal inputs from large cities heat up the river; tributaries merge, and cold groundwater enters at often unknown locations; in mountain regions, snow and glacier melt supply cold water in spring and summer. This complexity makes accurate modeling of river water temperature a challenging time-series problem.

Modeling river water temperature from air temperature is a time-series problem for which numerous methods have been proposed. For example, statistical models like *Air2Stream* [8], which incorporate air temperature and river discharge, are widely deployed. *Long short-term memory* (LSTM) models have also been applied to this task [9, 10]. Recent advances in deep learning have enabled more sophisticated architectures [11, 12], including transformer-based models [13]. In previous research, the effects of neighboring water stations [14] and spatio-temporal neural networks based on LSTMs and *graph neural networks* (GNNs) [15] were studied. Embedding techniques in conjunction with these models show particularly promising results, and the present work further investigates and substantiates these findings. That is, this paper investigates pattern recognition models for river water temperature modeling that rely on both established and novel embedding techniques.

Actually, node-specific embeddings are sometimes applied implicitly [13, 16] – for example, by concatenating coordinates or other variables to the time series, or by simply training separate models for each station. While these approaches introduce station-specific information, their role is rarely made explicit. To the best of our knowledge, no studies have systematically examined the effects of such implicit embeddings or thoroughly investigated alternative embedding strategies in sensor networks. This lack of understanding motivates a more principled analysis of embedding design choices.

The main contribution of this paper is threefold. First, we formally introduce a straightforward embedding technique for LSTMs on sensor networks (*concatenation embedding*). Second, we propose two novel embedding techniques (*gated memory embedding* and α*-Interpolation embedding*). Third, we compare and analyze the models on a real-world water temperature prediction task, with a particular focus on consistency and robustness across settings, and examine differences among the embeddings in terms of predictive performance, stability, and representation of node-specific characteristics.

The remainder of this paper is structured as follows. In Section 2, we discuss the node embedding techniques in detail. In Section 3, we provide an exper-

imental evaluation to assess their predictive performance. Section 4 presents supplementary studies that compare the techniques to transformers and beyond predictive performance. Finally, we conclude the paper and propose possible future research directions in Section 5.

2 Embedding Techniques

In this work, we formulate river water temperature modeling as a time-series problem on a sensor network, where each node corresponds to a monitoring station. Rather than training separate models, we deliberately adopt a single shared model across all nodes and rely on node-specific embeddings to capture local characteristics. This formulation allows us to explicitly separate global dynamics from local variations, making the role of embeddings central to the modeling approach.

This design is particularly well suited to the present setting. While air temperature – the primary input signal – is nearly uniform across Switzerland, local hydrological and geomorphological conditions induce substantial variability in river temperature responses. As a result, the mapping from air temperature to water temperature cannot be described by a single global function without accounting for node-specific effects.

This is illustrated in Fig. 1, which shows air and water temperature time series for two stations over one year. Despite the strong similarity of the input (air temperature) signals at stations 2056 and 2386 (orange line), their corresponding water temperature dynamics differ significantly (blue lines). This discrepancy highlights a key challenge: similar inputs can lead to systematically different outputs depending on location. Consequently, simple input–output models are insufficient, and explicitly modeling node-specific characteristics – via embeddings – is essential for capturing these differences.

In the remainder of this section we first briefly review the vanilla LSTM and then describe different embedding techniques considered in this work. To the best of our knowledge, only one of these embedding approaches has been proposed and evaluated in previous studies [17], while the remaining techniques have not been presented in this context.[3]

2.1 Vanilla LSTM

The LSTM is a *recurrent neural network* (RNN) that is applied at each time step, with its hidden and cell state (h, c) propagated through time to encode the temporal context of the sequence. For completeness, the following formulations already contain the final linear layer suited to the problem definition, detailed in Section 3.1.

The vanilla LSTM consists of three gates: the forget gate f, the input gate i, and the output gate o. Further, a candidate state $\tilde{c}$ is computed in a manner

[3] Code is available under `https://tinyurl.com/srn-lstm-embedding`

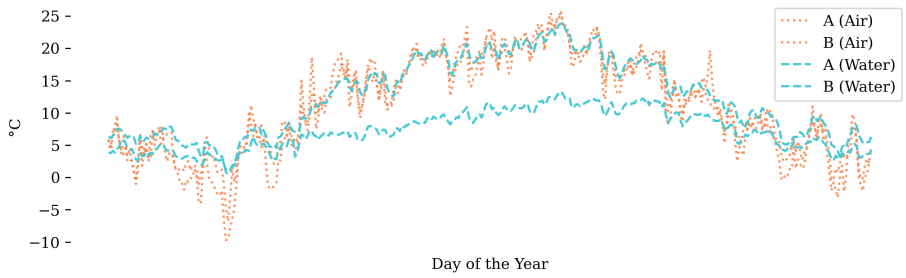

Fig. 1: Air and water temperature time series for two distinct stations, shown as two air–water pairs over the course of one year. Notably, air temperature patterns are highly similar across both stations.

similar to a gate. The memory cell c is then updated using the Hadamard product $\odot$ of the forget gate, the input gate and the candidate state. The output of the LSTM is the hidden state h, obtained as the Hadamard product of the output gate and the activated memory cell. Both the memory cell c and the hidden state h are propagated through time. Finally, a linear transformation is applied to project the hidden state into the target space, yielding the final prediction $\hat{y}$.

Formally, for node k at time step t, the LSTM is defined as:

$$
\begin{aligned}
&\text{LSTM}(x_k^{(t)}) : \\
&\qquad f^{(t)} = \sigma(\boldsymbol{W}_f x_k^{(t)} + \boldsymbol{U}_f h^{(t-1)} + \boldsymbol{b}_f) \\
&\qquad i^{(t)} = \sigma(\boldsymbol{W}_i x_k^{(t)} + \boldsymbol{U}_i h^{(t-1)} + \boldsymbol{b}_i) \\
&\qquad o^{(t)} = \sigma(\boldsymbol{W}_o x_k^{(t)} + \boldsymbol{U}_o h^{(t-1)} + \boldsymbol{b}_o) \\
&\qquad \tilde{c}^{(t)} = \theta(\boldsymbol{W}_c x_k^{(t)} + \boldsymbol{U}_c h^{(t-1)} + \boldsymbol{b}_c) \\
&\qquad c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} \\
&\qquad h^{(t)} = o^{(t)} \odot \theta(c^{(t)}) \\
&\qquad \hat{y}^{(t)} = \boldsymbol{W}_y h^{(t)} + \boldsymbol{b}_y
\end{aligned}
$$

Here $\sigma(\cdot)$ denotes the sigmoid activation function and $\theta(\cdot)$ the tanh function. Bold symbols denote the learnable network weights. All weight matrices are shared among all nodes and only the input time series x_k depends on node k.

2.2 Concatenation Embedding

This technique can be seen as a straightforward embedding technique for the vanilla LSTM. The embedding – either learned during training or derived from static node-attributes – is concatenated with the node-specific input time series $x_k^{(t)}$ at each time step t. This approach is formally defined as follows:

$$
\begin{aligned}
\text{CONCAT\_LSTM}(x_k^{(t)}, \mathbf{e}_k) := LSTM(x_k^{(t)} || \mathbf{e}_k) : \\
f^{(t)} &= \sigma(\boldsymbol{W}_f x_k^{(t)} + \boldsymbol{V}_f \mathbf{e}_k + \boldsymbol{U}_f h^{(t-1)} + \boldsymbol{b}_f) \\
i^{(t)} &= \sigma(\boldsymbol{W}_i x_k^{(t)} + \boldsymbol{V}_i \mathbf{e}_k + \boldsymbol{U}_i h^{(t-1)} + \boldsymbol{b}_i) \\
o^{(t)} &= \sigma(\boldsymbol{W}_o x_k^{(t)} + \boldsymbol{V}_o \mathbf{e}_k + \boldsymbol{U}_o h^{(t-1)} + \boldsymbol{b}_o) \\
\tilde{c}^{(t)} &= \theta(\boldsymbol{W}_c x_k^{(t)} + \boldsymbol{V}_c \mathbf{e}_k + \boldsymbol{U}_c h^{(t-1)} + \boldsymbol{b}_c) \\
c^{(t)} &= f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} \\
h^{(t)} &= o^{(t)} \odot \theta(c^{(t)}) \\
\hat{y}^{(t)} &= \boldsymbol{W}_y h^{(t)} + \boldsymbol{b}_y
\end{aligned} \tag{1}
$$

Note that this concatenation leads to an additional matrix multiplication in each gate, resulting in four matrix multiplications with $\boldsymbol{V}_f$, $\boldsymbol{V}_i$, $\boldsymbol{V}_o$, $\boldsymbol{V}_c$. During inference, these products remain constant for a given node and are added to the corresponding gates, allowing the embedding to influence each gate across all hidden dimensions. All weight matrices are shared among all nodes and only the time series x_k and the embedding $\mathbf{e}_k$ are node-specific.

2.3 Gated Memory Embedding

The gated memory embedding extends the vanilla LSTM by introducing an additional embedding gate e that directly modulates the memory cell. In contrast to the concatenation embedding described in Section 2.2, which influences the gates indirectly through augmented inputs, this approach integrates the embedding explicitly into the cell state update. As such, it represents a natural architectural extension of the LSTM rather than an input-level modification. Formally:

$$
\begin{aligned}
\text{GATED\_MEMORY\_EMBEDDING\_LSTM}(x_k^{(t)}, \mathbf{e}_k) : \\
f^{(t)} &= \sigma(\boldsymbol{W}_f x_k^{(t)} + \boldsymbol{U}_f h^{(t-1)} + \boldsymbol{b}_f) \\
i^{(t)} &= \sigma(\boldsymbol{W}_i x_k^{(t)} + \boldsymbol{U}_i h^{(t-1)} + \boldsymbol{b}_i) \\
o^{(t)} &= \sigma(\boldsymbol{W}_o x_k^{(t)} + \boldsymbol{U}_o h^{(t-1)} + \boldsymbol{b}_o) \\
e^{(t)} &= \sigma(\boldsymbol{W}_e x_k^{(t)} + \boldsymbol{U}_e h^{(t-1)} + \boldsymbol{b}_e) \qquad (2) \\
\tilde{c}^{(t)} &= \theta(\boldsymbol{W}_c x_k^{(t)} + \boldsymbol{U}_c h^{(t-1)} + \boldsymbol{b}_c) \\
\tilde{\mathbf{e}} &= \theta(\boldsymbol{W}_{\tilde{\mathbf{e}}} \mathbf{e}_k + \boldsymbol{b}_e) \qquad (3) \\
c^{(t)} &= f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} + e^{(t)} \odot \tilde{\mathbf{e}} \qquad (4) \\
h^{(t)} &= o^{(t)} \odot \theta(c^{(t)}) \\
\hat{y}^{(t)} &= \boldsymbol{W}_y h^{(t)} + \boldsymbol{b}_y
\end{aligned}
$$

Note that Eq. (2) defines the so-called *embedding gate* e which controls how strongly the embedding contributes at each time step, while Eq. (3) projects the node-specific embedding into the hidden space. Finally, Eq. (4) controls which part and amount of the projected embedding $\tilde{\mathbf{e}}$ is affecting the memory cell of the LSTM. Again, all weight matrices are shared across nodes; only the input time series x_k and the embedding $\mathbf{e}_k$ are node-specific.

2.4 α-Interpolation Embedding

A different way to encode an embedding is to interpolate between two weight matrices. Assume two matrices of identical dimensions, $\boldsymbol{M}_1 \in \mathbb{R}^{m\times n}$ and $\boldsymbol{M}_2 \in \mathbb{R}^{m\times n}$. Let the scalar $\alpha \in [0,1]$ serve as the embedding. Then, we obtain a new matrix $\boldsymbol{M}_\alpha \in \mathbb{R}^{m\times n}$ via the convex combination

$$\boldsymbol{M}(\alpha) = \alpha\, \boldsymbol{M}_1 + (1-\alpha)\, \boldsymbol{M}_2\,.$$

To extend the representative power of α, we generalize it to a vector $\boldsymbol{\alpha} \in \mathbb{R}^n$, enabling *column-wise interpolation* between $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$:

$$\boldsymbol{M}(\boldsymbol{\alpha}) = \boldsymbol{M}_1\, \mathrm{diag}(\boldsymbol{\alpha}) + \boldsymbol{M}_2\, \mathrm{diag}(\mathbf{1}-\boldsymbol{\alpha}) \tag{5}$$

The gradient of the loss function L with respect to the learnable parameters $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ follows directly from this formulation, making it well-suited for first-order optimization methods. The corresponding derivatives are given by

$$\frac{\partial L}{\partial \boldsymbol{M}_1} = \boldsymbol{\alpha}\, \frac{\partial L}{\partial \boldsymbol{M}}\,, \text{ and}$$
$$\frac{\partial L}{\partial \boldsymbol{M}_2} = (\mathbf{1}-\boldsymbol{\alpha})\, \frac{\partial L}{\partial \boldsymbol{M}}\,.$$

.

The α-Interpolation LSTM is a vanilla LSTM where each learnable matrix is replaced by the column-wise interpolated matrix $\boldsymbol{M}(\boldsymbol{\alpha})$ defined in Eq. (5). Formally:

$$\begin{aligned}
&\alpha\text{-INTERPOLATION_LSTM}(x_k^{(t)}, \mathbf{e}_k):\\
&\qquad \boldsymbol{\alpha} = \sigma(\boldsymbol{W}_{\mathrm{e}}\mathbf{e}_k + \boldsymbol{b}_e) \qquad (6)\\
&\qquad f^{(t)} = \sigma(\boldsymbol{W}_f(\boldsymbol{\alpha})\, x_k^{(t)} + \boldsymbol{U}_f(\boldsymbol{\alpha})h^{(t-1)} + \boldsymbol{b}_f(\boldsymbol{\alpha}))\\
&\qquad i^{(t)} = \sigma(\boldsymbol{W}_i(\boldsymbol{\alpha})x_k^{(t)} + \boldsymbol{U}_i(\boldsymbol{\alpha})h^{(t-1)} + \boldsymbol{b}_i(\boldsymbol{\alpha}))\\
&\qquad o^{(t)} = \sigma(\boldsymbol{W}_o(\boldsymbol{\alpha})x_k^{(t)} + \boldsymbol{U}_o(\boldsymbol{\alpha})h^{(t-1)} + \boldsymbol{b}_o(\boldsymbol{\alpha}))\\
&\qquad \tilde{c}^{(t)} = \theta(\boldsymbol{W}_c(\boldsymbol{\alpha})x_k^{(t)} + \boldsymbol{U}_c(\boldsymbol{\alpha})h^{(t-1)} + \boldsymbol{b}_c(\boldsymbol{\alpha}))
\end{aligned}$$

$$
\begin{aligned}
c^{(t)} &= f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} \\
h^{(t)} &= o^{(t)} \odot \theta(c^{(t)}) \\
\hat{y}^{(t)} &= \boldsymbol{W}_y h^{(t)} + \boldsymbol{b}_y
\end{aligned} \tag{7}
$$

Eq. (6) projects the node-specific embedding $\mathbf{e}_k$ to the interpolation vector $\boldsymbol{\alpha}$, whose dimensionality matches the number of columns required for the α-interpolation. The sigmoid activation function $\sigma(\cdot)$ constrains the values of $\boldsymbol{\alpha}$ to the interval $[0, 1]$, ensuring that the resulting weight matrices are obtained as convex combinations.

All LSTM weight matrices $\boldsymbol{W}_j(\boldsymbol{\alpha})$, $\boldsymbol{U}_j(\boldsymbol{\alpha})$, and bias terms $\boldsymbol{b}_j(\boldsymbol{\alpha})$ are constructed using the column-wise matrix interpolation defined in Eq. (5) (with $j \in \{f, i, o, c\}$). This allows the embedding to directly modulate the model parameters while preserving the overall LSTM structure.

For a fair comparison across models, matrix interpolation is not applied to the final linear transformation producing the target prediction $\hat{y}$ in Eq. (7).

2.5 Intuition and Motivation

Each embedding technique offers distinct benefits and limitations in terms of expressive power, temporal flexibility, and computational complexity.

- The *concatenation embedding* is quite powerful, as the embedding acts on every time step and on every gate. Thus, this embedding has the possibility to infer with all dimensions of the hidden space. However, once the training is completed, the projected embedding for a particular node remains constant across time steps and therefore does not capture temporal variation.
- The *gated memory embedding* is less expressive than concatenation, but explicitly introduces a notion of time: the value of the embedding gate e depends on the hidden state h and can vary throughout the sequence. This allows the final embedding to modulate the memory cell differently depending on the temporal context.
- The α-*Interpolation embedding* performs column-wise interpolations between the weight matrices of two learned LSTMs. These two LSTMs define strict boundaries on the representational capacity of the embedding. Since this approach only modifies the parameterization during inference, the inference-time computations remain identical to that of a vanilla LSTM.

2.6 Implicit Embedding Techniques

We observe that, in practice and hydrological research, several implicit embeddings arise [12, 13, 16]. These are techniques in which an embedding emerges implicitly through model design, rather than being introduced as an explicit architectural component. While often not framed as such, these approaches effectively encode node-specific information and thus serve a similar functional role. A few examples include:

- *Static Features*: Node-specific information is concatenated with, or otherwise incorporated into, the time-series input. For example, node coordinates are often included, allowing the network to distinguish between nodes even when their time series are otherwise very similar.
- *Node-specific modeling*: Each node is trained with a separate model, which can be interpreted as an embedding technique with maximal flexibility, as node identity is fully captured by model parameters.
- *Fixed-position modeling*: The data is organized in tabular form and end-to-end learning is applied to the entire table. In this setup, each node consistently appears at the same input position in the network, implicitly encoding node identity through positional structure.

All these approaches are valid and commonly used. However, their implicit nature makes it difficult to isolate, analyze, and compare their effects systematically. Their widespread adoption further highlights the potential – and necessity – of explicit and well-designed embedding techniques.

3 Experimental Evaluation

In this section, we empirically evaluate the proposed embedding techniques on a real-world river water temperature modeling task. We first formalize the prediction problem and describe the datasets used in our experiments. We then detail the training procedure and evaluation protocol, followed by a quantitative comparison of the different models with respect to predictive performance.

3.1 Problem Definition and Baseline Models

We focus on the task of water temperature modeling, which involves predicting the estimated water temperature $\hat{w}^{(t)}$ at time step t, given an observed air temperature time series $(a^{(1)}, a^{(2)}, \ldots, a^{(t)})$ and a node specific embedding $\mathbf{e}_k$. The objective is to learn a model function f, as well as the node-specific embedding $\mathbf{e}_k$, such that

$$f(a^{(1)}, a^{(2)}, \ldots, a^{(t)}, \mathbf{e}_k) = \hat{w}^{(t)}, \tag{8}$$

where $\hat{w}^{(t)}$ denotes the model's prediction (which is generally distinct from the actual measured water temperature $w^{(t)}$). Note that f is does not refer to a forecasting model with a positive prediction horizon, but rather an air-to-water model that can be used in hydrological downstream applications.

To measure and compare our predictions, we compute the widely used *Root Mean Squared Error* (RMSE) as well as the *Mean Absolute Error* (MAE).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(w^{(i)} - \hat{w}^{(i)})^2}, \quad \text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|w^{(i)} - \hat{w}^{(i)}| \tag{9}$$

For both errors, values close to 0 indicate better prediction qualities.

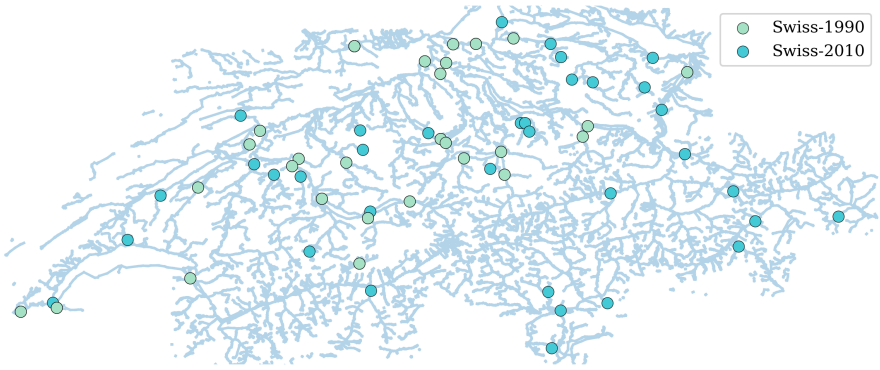

Fig. 2: Spatial distribution of water monitoring stations across Switzerland's water bodies. We distinguish two datasets: Swiss-1990 and Swiss-2010.

We use a single vanilla LSTM across all nodes as a weak baseline model, which does not incorporate any form of embedding. This baseline serves as a reference point to isolate the effect of node-specific information and is not expected to perform well. A secondary baseline is the concatenation embedding, which provides a simple yet competitive approach to incorporating node-specific features and allows us to compare the more complex embedding techniques against a strong and transparent reference.

3.2 Dataset

Switzerland's *Federal Office for the Environment* (FOEN) has monitored river temperature for more than 40 years. Measurements are recorded at water stations every 10 minutes and reported as daily averages (with sensors routinely inspected and calibrated). In response to climate change, the FOEN expanded its network from about 40 to 80 stations between 2000 and 2010. Air temperature data from nearby *MeteoSwiss* stations supports the joint analysis of atmospheric and hydrological conditions.

Based on this data, two publicly available datasets[4] have been proposed, viz. Swiss-1990 and Swiss-2010 [15]. Both datasets measure water temperatures in Switzerland, either since 1990 on 28 water stations or since 2010 on 63 water stations. Note that the Swiss-1990 stations form a subset of Swiss-2010. Fig. 2 shows the distribution of water monitoring stations across Switzerland's water bodies.

3.3 Training and Hyperparameter Tuning

We use the Adam optimizer for gradient based optimization [18]. Hyperparameter selection is performed via random search using the ASHA scheduler [19].

[4] Available under `https://swiss-river-network.github.io/data`

For validation, we split the training data into 80% and 20% subsets and select models based on the lowest validation RMSE. The node-specific embeddings $\mathbf{e}_k$ are learned jointly during training by computing the gradients with respect to the embeddings and updating them accordingly.

3.4 Empirical Results

We report performance on the hold-out test data in Table 1. As a base reference system, we use a vanilla LSTM without embeddings, which allows us to directly assess the performance gains obtained by introducing embeddings. For each dataset and embedding technique, we report the mean, minimum, and maximum RMSE, as well as the mean MAE. The best result per metric is highlighted in bold.

We observe substantial variation in prediction performance across stations, with RMSE values ranging from 0.37 to 1.70 for the embedding techniques. This variability reflects the heterogeneous nature of the sensor network, where certain stations exhibit more complex or less predictable dynamics.

Compared to the baseline LSTM, all embedding techniques yield a pronounced and consistent reduction in prediction error. This confirms that incorporating node-specific information is essential for modeling ambiguous time series in this setting. Notably, even the simple concatenation embedding already provides a strong improvement over the non-embedded baseline, underlining the importance of distinguishing between nodes.

Among the evaluated methods, the more advanced embedding techniques – Gated Memory and α-Interpolation – achieve further, albeit moderate, improvements. On the Swiss-1990 dataset, the Gated Memory model attains the lowest mean RMSE and MAE, suggesting that its explicit integration of embeddings into the memory cell is particularly beneficial in smaller sensor networks. On Swiss-2010, the α-Interpolation model performs best overall, achieving the lowest mean RMSE and MAE.

When compared directly to concatenation, both advanced methods outperform it across several metrics, including mean RMSE, minimum RMSE, and mean MAE. However, statistical analysis using the Wilcoxon signed-rank test indicates that these improvements are only statistically significant for α-Interpolation on Swiss-2010 ($p < 0.01$). This suggests that, while more sophisticated embedding mechanisms can provide gains, their impact depends on the dataset and may not always be substantial.

Overall, these results highlight two key findings:

1. the inclusion of embeddings is critical for achieving strong predictive performance, and
2. the choice of embedding technique provides additional, but comparatively smaller, improvements.

Table 1: RMSE performance on the hold-out test sets for water temperature modeling. Bold values indicate best performance. *Denotes significant improvements over Concatenation ($p < 0.01$).

		RMSE			MAE
Dataset	**Method**	**Mean**	**Min**	**Max**	**Mean**
Swiss-1990	LSTM (Baseline)	1.510	.56	3.04	1.25
	Concatenation	.800	.48	1.39	.62
	Gated Memory	**.784**	.46	1.39	**.60**
	α-Interpolation	.786	**.45**	**1.34**	**.60**
Swiss-2010	LSTM (Baseline)	1.510	.58	3.27	1.26
	Concatenation	.809	.46	1.63	.62
	Gated Memory	.796	.40	**1.62**	.61
	α-Interpolation*	**.782**	**.37**	1.70	**.60**

4 Additional Studies

In this section, we analyze the proposed embedding techniques beyond RMSE and MAE performance, examining the structure of their embedding spaces, their interaction with static features, and their comparative performance against transformer architectures.

4.1 Robustness of the Embedding Space

In this experiment, we assess the robustness of the embedding spaces produced by the different embedding techniques. We define robustness as local stability of the embedding space, i.e. small perturbations to an embedding should lead to only minimal changes in predictive performance.

To evaluate this property, we train models with identical embedding dimensionality and select the best-performing model based on validation performance. We then perturb each learned embedding with scaled Gaussian noise,

$$\tilde{\mathbf{e}}_k = \mathbf{e}_k + \mathcal{N}(0, n)\,, \text{ where } n \in [0, 1]\,, \tag{10}$$

and measure the resulting degradation in performance.

Fig. 3 shows the robustness of each embedding technique on both datasets, showing RMSE as a function of the embedding perturbation magnitude. Across Swiss-1990 and Swiss-2010, the concatenation embedding is the least robust under embedding perturbations. On Swiss-1990, the gated memory embedding exhibits the highest robustness, indicating a more stable embedding space.

Differences between the different embedding techniques are more pronounced on Swiss-1990, which contains only half the number of nodes compared to Swiss-2010, suggesting that embedding robustness becomes increasingly important in smaller sensor networks.

4.2 Embedding Static Features

In this experiment, we replace the learnable embeddings with three static features for each water station: the x- and y-coordinates and the altitude. While altitude constitutes a meaningful explanatory variable, spatial coordinates are only indirectly related to river water temperature and primarily encode the geographic identity of a station.

In Table 2 we present the results of this study. Overall, performance is very close to that achieved with learned embeddings, suggesting that static features can, to some extent, fulfill a similar role as learned embeddings. This suggests that spatial coordinates contribute little direct predictive power for water temperature modeling and instead primarily serve to distinguish between monitoring stations.

This finding implies that performance gains are largely driven by the model's ability to separate node-specific dynamics rather than by meaningful spatial information. To further investigate this, we introduce a *shuffle experiment*. In which the same static features are randomly assigned to stations. This preserves separability in the embedding space, while removing geographical explainability. The results support our claim: performance remains very similar on Swiss-2010 and is slightly worse on Swiss-1990.

This raises the question of whether such node-separating mechanisms are specific to LSTM-based architectures or represent a more general modeling principle. In the following section, we therefore examine whether similar effects can be observed when embedding techniques are applied within transformer-based time series models.

4.3 LSTMs vs. Transformers

In this third experiment, we compare LSTM-based models with transformer architectures. The objective is twofold. First, we aim to situate our embedding-based approaches with respect to another state-of-the-art time-series model. Sec-

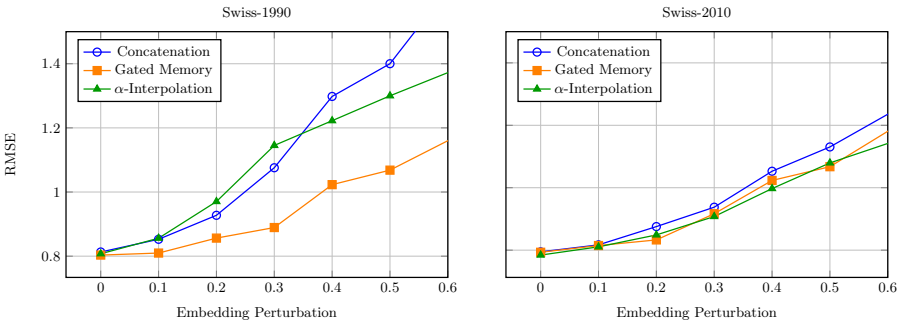

Fig. 3: RMSE as a function of embedding perturbation magnitude shown on both Swiss-1990 and Swiss-2010.

ond, this experiment demonstrates that the proposed embedding techniques are not limited to LSTMs and can be readily applied to other architectures.

Table 3 reports the results of the comparison between LSTM and transformer models on both datasets. For simplicity, we focus on the concatenation embedding only. Within this setting, predictive performance is compared directly between LSTM-based and transformer-based architectures.

Two key observations emerge. First, the concatenation embedding transfers straightforwardly to transformer models and is essential for achieving strong predictive performance. Second, transformers equipped with concatenation embeddings achieve lower RMSE than LSTMs with the same embedding technique and also slightly outperform LSTMs using gated memory and α-Interpolation embeddings (cf. Table 1).

These performance gains of the transformers, however, come at the cost of increased model complexity. Due to their higher capacity and global attention mechanisms, transformer architectures require substantially longer training times and incur greater computational overhead (cf. Table 3).

5 Conclusion and Future Work

Increasing river water temperature poses a growing threat to freshwater ecosystems and water quality, creating a demand for reliable and adaptable predictive models. Across many models for river water temperature modeling, either explicit or implicit embedding techniques are commonly employed. In this work, we systematically investigate the role of embeddings and analyze how different embedding techniques affect predictive performance and model behavior in such a sensor network. Specifically, we analyze the impact of three different embedding techniques: Concatenation, Gated Memory, and α-Interpolation embedding.

Through an extensive empirical evaluation on real-world datasets, we show that the inclusion of node-specific embeddings is substantially more important than the particular embedding mechanism employed. While more sophisticated

Table 2: RMSE performance when learnable embedding is exchanged with coordinates and altitude. Shuffle uses the same static embeddings but randomly assigned to nodes. In parentheses are the differences to the learned embeddings.

		RMSE		
Dataset	**Method**	**Learned**	**Static**	**Shuffle**
Swiss-1990	Concatenation	.800	.805 (+0.5%)	.843 (+4.3%)
	Gated Memory	.784	.797 (+1.3%)	.801 (+1.7%)
	α-Interpolation	.786	.795 (+0.9%)	.801 (+1.5%)
Swiss-2010	Concatenation	.809	.798 (−1.1%)	.807 (−0.2%)
	Gated Memory	.796	.796 (+0.0%)	.796 (+0.0%)
	α-Interpolation	.782	.796 (+1.4%)	.789 (+0.7%)

Table 3: Performance (RMSE) and training time (mm:ss) for LSTM and Transformer models using concatenation embedding.

	RMSE		Training time	
Dataset	**LSTM**	**Transformer**	**LSTM**	**Transformer**
Swiss-1990	.800	.776 (−2.4%)	04:28	18:13 (+307%)
Swiss-2010	.809	.779 (−3.0%)	14:03	21:30 (+53%)

techniques such as Gated Memory and α-Interpolation yield moderate but consistent improvements over simple concatenation, the primary benefit of embeddings lies in enabling a single shared model to capture node-specific characteristics in otherwise ambiguous time-series data. This leads to pronounced performance gains compared to non-embedded baselines.

Beyond predictive accuracy, our additional analyses reveal meaningful differences between embedding techniques in terms of robustness. In particular, the Gated Memory embedding exhibits increased stability under perturbations in smaller sensor networks. These findings suggest that embedding design should be guided not only by RMSE-based evaluation, but also by qualitative properties of the embedding space, especially in a heterogeneous and data-limited setting like water temperature modeling.

Further, we demonstrate that the benefits of embeddings are not limited to LSTM-based models. Transformer architectures also profit from explicit embeddings, achieving improved predictive performance, albeit at the cost of increased model complexity and training time.

Future work will explore the properties of embedding spaces under varying data conditions. In particular, we aim to deepen the analysis of embedding strategies for both LSTM- and transformer-based architectures and evaluate their behavior across diverse time-series benchmarks. Finally, we aim to incorporate structural information – an important predictor in sensor networks – into a novel embedding scheme.

References

1. Reid, P.C., Hari, R.E., Beaugrand, G., Livingstone, D.M., Marty, C., Straile, D., Barichivich, J., Goberville, E., Adrian, R., Aono, Y., et al.: Global impacts of the 1980s regime shift. Global change biology 22(2), 682–703 (2016)
2. Michel, A., Brauchli, T., Lehning, M., Schaefli, B., Huwald, H.: Stream temperature and discharge evolution in switzerland over the last 50 years: annual and seasonal behaviour. Hydrology and Earth System Sciences 24(1), 115–142 (2020)
3. Dahlke, F.T., Wohlrab, S., Butzin, M., Pörtner, H.O.: Thermal bottlenecks in the life cycle define climate vulnerability of fish. Science 369(6499), 65–70 (2020)
4. Caissie, D.: The thermal regime of rivers: a review. Freshwater biology 51(8), 1389–1406 (2006)

5. Karvonen, A., Rintamäki, P., Jokela, J., Valtonen, E.T.: Increasing water temperature and disease risks in aquatic systems: climate change increases the risk of some, but not all, diseases. International journal for parasitology 40(13), 1483–1488 (2010)
6. Johnson, M.F., Albertson, L.K., Algar, A.C., Dugdale, S.J., Edwards, P., England, J., Gibbins, C., Kazama, S., Komori, D., MacColl, A.D., et al.: Rising water temperature in rivers: Ecological impacts and future resilience. Wiley Interdisciplinary Reviews: Water 11(4), e1724 (2024)
7. Piccolroaz, S., Calamita, E., Majone, B., Gallice, A., Siviglia, A., Toffolon, M.: Prediction of river water temperature: a comparison between a new family of hybrid models and statistical approaches. Hydrological Processes 30(21), 3901–3917 (2016)
8. Toffolon, M., Piccolroaz, S.: A hybrid model for river water temperature as a function of air temperature and discharge. Environmental Research Letters 10(11), 114011 (2015)
9. Qiu, R., Wang, Y., Rhoads, B., Wang, D., Qiu, W., Tao, Y., Wu, J.: River water temperature forecasting using a deep learning method. Journal of Hydrology 595, 126016 (2021)
10. Piotrowski, A.P., Napiorkowski, M.J., Napiorkowski, J.J., Osuch, M.: Comparing various artificial neural network types for water temperature prediction in rivers. Journal of Hydrology 529, 302–315 (2015)
11. Jia, X., Zwart, J., Sadler, J., Appling, A., Oliver, S., Markstrom, S., Willard, J., Xu, S., Steinbach, M., Read, J., et al.: Physics-guided recurrent graph model for predicting flow and temperature in river networks. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM). pp. 612–620. SIAM (2021)
12. Moshe, Z., Metzger, A., Elidan, G., Kratzert, F., Nevo, S., El-Yaniv, R.: Hydronets: Leveraging river structure for hydrologic modeling. arXiv preprint arXiv:2007.00595 (2020)
13. Padrón, R.S., Zappa, M., Bernhard, L., Bogner, K.: Extended range forecasting of stream water temperature with deep learning models. EGUsphere 2024, 1–27 (2024)
14. Fankhauser, B., Bigler, V., Riesen, K.: Graph-based deep learning on the swiss river network. In: International Workshop on Graph-Based Representations in Pattern Recognition. pp. 172–181. Springer (2023)
15. Fankhauser, B., Bigler, V., Riesen, K.: Evaluating the role of graphs and embeddings in spatio-temporal modeling of river water temperature. Available at SSRN 5639250
16. Lim, B., Arık, S.Ö., Loeff, N., Pfister, T.: Temporal fusion transformers for interpretable multi-horizon time series forecasting. International journal of forecasting 37(4), 1748–1764 (2021)
17. Fankhauser, B., Bigler, V., Riesen, K.: Leveraging lstm embeddings for river water temperature modeling. In: IAPR Workshop on Artificial Neural Networks in Pattern Recognition. pp. 283–294. Springer (2024)
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
19. Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Ben-Tzur, J., Hardt, M., Recht, B., Talwalkar, A.: A system for massively parallel hyperparameter tuning. Proceedings of Machine Learning and Systems 2, 230–246 (2020)