International Journal of Numerical Methods for Heat & Fluid Flow

2251

Received 31 March 2025
Revised 14 October 2025
Accepted 9 December 2025

Temperature forecasting for radioactive waste monitoring using graph neural networks

Pierre Hembert
PIMM Laboratory, Arts et Métiers Institute of Technology, National Centre for Scientific Research, Paris, France, and Department of Data Chain – Sensors, Robotics and AI Data Analytics, ANDRA, Châtenay-Malabry, France

Chady Ghnatios
Department of Mechanical Engineering, University of North Florida, Jacksonville, Florida, USA

Julien Cotton
Department of Data Chain – Sensors, Robotics and AI Data Analytics, ANDRA, Châtenay-Malabry, France, and

Francisco Chinesta
PIMM Laboratory, Arts et Métiers Institute of Technology, National Centre for Scientific Research, Paris, France

Abstract

Purpose – A deep geological repository for radioactive waste, such as Andra's Cigéo project, necessitates long-term monitoring. This monitoring is achieved by collecting data from various sensors. However, due to environmental conditions (radioactivity and mechanical constraints), this set of sensors is prone to deterioration over time. Therefore, it is essential to replace the responses of faulty sensors with comprehensive predictions. Graph neural networks (GNNs) are appropriate models for these predictions, as they efficiently characterize the physical phenomena present in the system, leverage the underlying topology of the data and can be used to infer general dependencies. The purpose of this paper is to study the effectiveness of GNNs for this temperature interpolation task.

Design/methodology/approach – In this paper, the authors trained several types of GNNs for temperature forecasting using experimental data from Andra's Underground Research Laboratory. The specific experiment used to train the machine learning algorithms simulates the heating of a high-level waste (HLW) demonstrator cell by radioactive waste within a deep geological layer. The model the authors used is a forward-integrating GNN that takes the initial temperature and boundary conditions as input and outputs the temperature field at all future time steps.

Findings – By comparing GNNs to other machine learning algorithms (Gaussian processes, artificial neural networks and kriging), the authors proved their effectiveness for data completion.

Originality/value – This work is original as it proposes the use of time-integrating GNNs for data completion through transfer learning, using data collected from an industrial demonstrator that simulates the heating of a HLW cell by radioactive waste.

Keywords Graph neural network, Temperature forecast, Radioactive waste monitoring, Sensor network, Transfer learning, High-level waste demonstrator cell

Paper type Research paper

International Journal of Numerical Methods for Heat & Fluid Flow
Vol. 36 No. 6, 2026
pp. 2251-2274
Emerald Publishing Limited
0961-5539
DOI 10.1108/HFF-03-2025-0211

1. Introduction

The most hazardous and long-lasting radioactive waste, primarily produced by the nuclear power industry, cannot be safely stored above ground. For this reason, the Andra's Cigéo project aims to design and build facilities for the disposal of French high-level (HLW) and intermediate-level radioactive waste in a deep geological formation known as the Callovo–Oxfordien, as illustrated in Figure 1. These facilities also require long-term monitoring, which is facilitated by an extensive network of sensors. However, the harsh environment (radioactivity and mechanical convergence of the tunnels) to which these sensors are exposed to, leads to sensor failures (bias and drift). Given that these sensors are not easily accessible, it is crucial to identify faulty sensors based on their responses and replace their values with accurate predictions to ensure reliable and consistent monitoring of the facilities. Therefore, a method for identifying inconsistent data using graph neural networks (GNNs) (Hembert *et al.*, 2024) and another method for replacing inaccurate data based on temperature forecasting (Muñoz *et al.*, 2024) have been proposed. In this paper, we propose another framework for temperature forecasting that uses forward-integrating GNNs.

This work is based on data collected at the Andra Underground Research Laboratory (URL). More specifically, the data comes from the thermal loading of a HLW demonstrator cell. This cell is a heavily instrumented demonstrator (equipped with both point and distributed sensors), that was loaded with thermal sources (Muñoz *et al.*, 2024; Bumbieler *et al.*, 2024) to simulate the heating of the cell by HLW, as shown in Figure 2.

Figure 3 depicts the temperature distribution observed during the thermal loading experiment across all sensors. The temperature field has been reconstructed using robust principal component analysis (Muñoz *et al.*, 2024). The heating experiment starts at approximately 130 days into the study.

GNNs are well-suited for temperature forecasting in this specific context. Graph not only represent the topology of the sensor network but also account for thermal phenomena, with heat flow occurring along the edges and energy conservation principles applying at the nodes.

GNNs operate on graph structured data and can perform different tasks (i.e. classification and regression) at different levels of the graph (Sanchez-Lengeling *et al.*, 2021; Peter *et al.*, 2018; Wu *et al.*, 2019b; Zhang *et al.*, 2020; Zhou *et al.*, 2020; Hoang *et al.*, 2023; Nguyen *et al.*, 2022; Procaccini *et al.*, 2024; Khemani *et al.*, 2024; Zhang *et al.*, 2019):

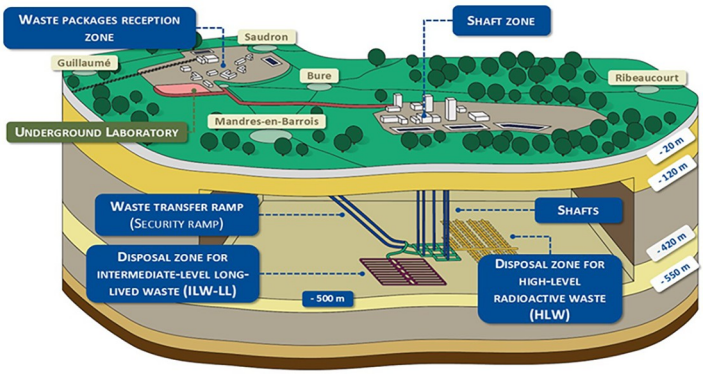

Figure 1. Andra's Industrial Centre for Geological Disposal, Cigéo

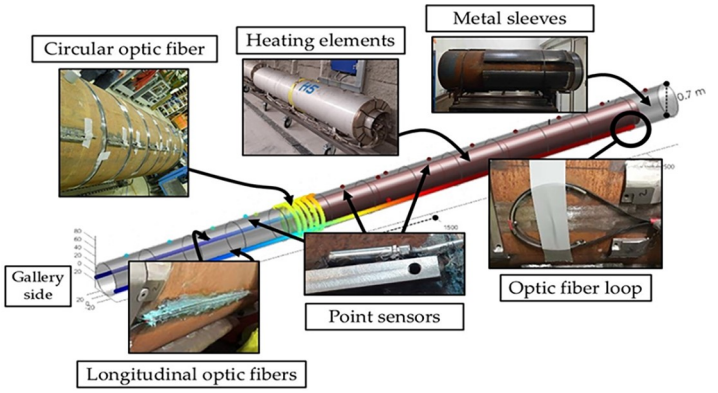

Figure 2. Thermal loading of the HLW demonstrator cell

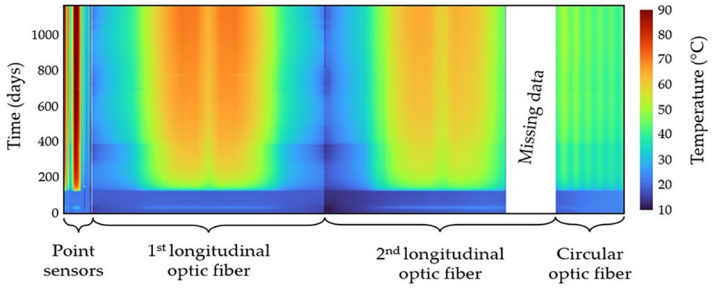

Figure 3. Reconstructed data from the thermal loading of the HLW demonstrator cell

- Tasks on the graph level produce a single prediction for the whole graph [e.g. estimating the toxicity of a molecule given its graph (Cremer *et al.*, 2023)].
- Tasks on the node level produce a prediction for each of the nodes [e.g. classifying scientific articles based on the citation graph (Maurya *et al.*, 2021; McCallum *et al.*, 2000)].
- Tasks on the edge level produce a prediction for each of the edges of the graph [e.g. friend recommendation provided a social network graph (Wang, 2022)].

GNNs in this work perform a regression task on nodes: given a temperature field (i.e. the temperature at every node of the graph) at a given time step, they predict the temperature field at the next time step.

GNNs rely on the iterative applications of a mechanism that updates the embeddings of the graph (i.e. the information contained in the nodes and the edges) without altering the connectivity of the graph. This mechanism, which exploits the topology of the graph, is known as message-passing and has been formalized by Gilmer *et al.* (2017). It can be broken down into elementary operations performed on individual nodes, which include the following (Sanchez-Lengeling *et al.*, 2021; Peter *et al.*, 2018; Wu *et al.*, 2019b; Zhang *et al.*, 2020; Zhou *et al.*, 2020; Hoang *et al.*, 2023; Nguyen *et al.*, 2022; Procaccini *et al.*, 2024; Khemani *et al.*, 2024; Zhang *et al.*, 2019; Gilmer *et al.*, 2017; Daigavane *et al.*, 2021):

- Pick the individual node v (in the set of the graph's nodes V) whose embedding is to be updated.
- Determine the neighborhood $\mathcal{N}(v)$ of the chosen node v. Then, gather all the neighboring nodes' (and edges) embeddings $(x_u)_{u\in\mathcal{N}(v)}$ (these are the messages).
- Apply a node order equivariant concatenation function α to transform the neighboring nodes' (edges) embeddings into a node embedding-like vector $\alpha\left((x_u)_{u\in\mathcal{N}(v)}\right) \sim x_v$.
- Use a ϕ update function that inputs the concatenated messages $\alpha\left((x_u)_{u\in\mathcal{N}(v)}\right)$ and the selected node's embedding x_v and outputs the updated node embedding x_v'. Generally, this update function ϕ is discovered by a neural network (usually a multi-layer perceptron [MLP]).

By subsequently using the same process for every node in the graph, a new graph with updated embeddings is generated. Figure 4 presents this mechanism and demonstrates how it leverages the topological information contained in the graph by incorporating neighboring elements when updating the nodes' embeddings. Equation (1) outlines how this mechanism is implemented in a GNN layer to update the node embeddings from layer l to updated embeddings at layer $l + 1$ (Peter *et al.*, 2018; Gilmer *et al.*, 2017):

$$\forall v \in V,\ x_v^{l+1} = \phi^l\left(\alpha^l\left(\left(x_u^l\right)_{u\in\mathcal{N}(v)}\right), x_v^l\right) \tag{1}$$

An important aspect of message passing is that both the concatenation function α and the update function ϕ are shared across all nodes (Daigavane *et al.*, 2021), thereby ensuring consistency throughout the graph and granting GNNs their generalization capabilities. This scheme is fundamental to the different GNN models. Furthermore, the update function ϕ and the concatenation function α can differ for each layer l.

Many GNN layers have been developed over the years (Peter *et al.*, 2018; Wu *et al.*, 2019b; Zhang *et al.*, 2020; Zhou *et al.*, 2020; Hoang *et al.*, 2023; Nguyen *et al.*, 2022; Chen *et al.*, 2021). In this study, we use three distinct types of layers, namely, the graph convolutional network (GCN) (Bruna *et al.*, 2014; Henaff *et al.*, 2015; Defferrard *et al.*, 2017; Kipf and Welling, 2017), the GraphSAGE (Hamilton *et al.*, 2018) and the graph attention network (GAT) (Veličković *et al.*, 2018; Zheng *et al.*, 2022; Wang *et al.*, 2021; Knyazev *et al.*, 2019).

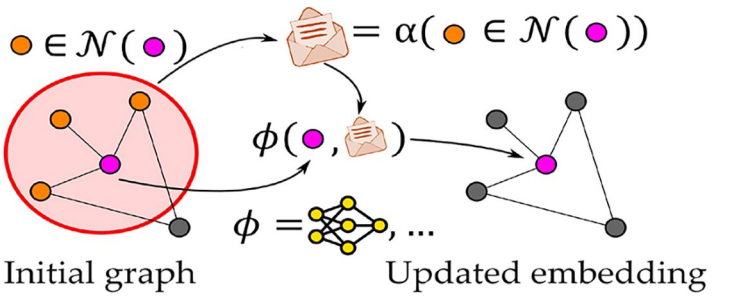

Figure 4. Message passing mechanism

GCN is a fundamental GNN layer. The GCN layer used in this paper is based on the works of Kipf and Welling (2017). Equation (2) presents the proposed GCN layer (Zhang *et al.*, 2020; Daigavane *et al.*, 2021; Wu *et al.*, 2019a; Dehmamy *et al.*, 2019). W_0^l and W_1^l are two distinct MLPs that consider the previous node embedding and the node's neighborhood, respectively. σ^l represents an often nonlinear activation function (e.g. rectified linear unit (ReLU) and exponential linear unit (ELU)). $c_{u,v}$ is the normalization scheme depending upon the size of the neighborhoods of nodes u and $vc_{u,v} = \sqrt{|\mathcal{N}(u)|} \cdot \sqrt{|\mathcal{N}(v)|}$:

$$\forall v \in V,\ x_v^{l+1} = \sigma^l\left(x_v^l \cdot W_0^l + \left[\sum_{u \in \mathcal{N}(v)} \frac{x_u^l}{c_{u,v}}\right] \cdot W_1^l\right) \tag{2}$$

GraphSAGE layers rely on a node-order invariant aggregation function *Agg* to concatenate information of neighboring nodes (rather than using an MLP). This design choice ensures that GraphSAGE is less computationally intensive than GCN and thus making it more suitable for larger graphs. The GraphSAGE layer implemented in this work is adapted from the framework introduced by Hamilton *et al.* (2018). Equation (3) introduces the proposed GraphSAGE layer (Daigavane *et al.*, 2021):

$$\forall v \in V,\ x_v^{l+1} = \sigma^l\left(\left[x_v^l, Agg_{u \in \mathcal{N}(v)}\left(x_u^l\right)\right] \cdot W^l\right) \tag{3}$$

GAT layers depend on an attention mechanism to assess the significance of connections between two nodes. Consequently, GAT layers enable greater generalization at a higher computational cost compared to GCN or GraphSAGE. The GAT layer used in this work is based on the work of Veličković *et al.* (2018). Equation (4) defines the proposed GAT layer (Daigavane *et al.*, 2021; Knyazev *et al.*, 2019), while the attention mechanism $a_{u,v}^l$, implemented through the attention MLP A^l, which takes the embeddings of two connected nodes as inputs, is detailed in equation (5):

$$\forall v \in V,\ x_v^{l+1} = \sigma^l\left(\left[a_{v,v}^l \cdot x_v^l + \sum_{u \in \mathcal{N}(v)} a_{u,v}^l \cdot x_u^l\right] \cdot W^l\right) \tag{4}$$

$$\forall v \in V,\ \forall u \in \mathcal{N}(v),\ a_{u,v}^l = \frac{\left[x_u^l, x_v^l\right] \cdot A^l}{\sum_{w \in \mathcal{N}(v)} \left(\left[x_v^l, x_w^l\right] \cdot A^l\right)} \tag{5}$$

2. Related works

The aim of this paper is to train a time-integrating GNN. This approach allows the GNN to predict the temperature field at subsequent time steps, given an initial temperature distribution and specific boundary conditions. By repeatedly applying this GNN, the temperature field can be forecasted until it reaches a hypothetical steady-state. The use of time-integrating GNNs is warranted in this context because once a sensor fails, it usually does not recover (especially if the optic fiber is damaged) and there is no possible way to replace this sensor in the radioactive waste storage facility.

GNNs have been used as surrogate models for thermal prediction (Boussaid *et al.*, 2023; Mozaffar *et al.*, 2021; Pfaff *et al.*, 2021; Yang *et al.*, 2024; Jia *et al.*, 2023;

Sanchis-Alepuz and Stipsitz, 2022). For instance, Boussaid *et al.* (2023) apply GAT to model district heating networks, where edges represent pipes and nodes represent various other components (e.g. valves, pumps and exchangers). Yang *et al.* (2024) use graphs to represent houses with rooms as nodes and walls as edges. They then implement a time-integrating GNN, based on GraphSAGE (Hamilton *et al.*, 2018) or message-passing neural networks (Gilmer *et al.*, 2017), which can predict the temperature field and heat flows given a set of initial conditions. By iteratively applying this GNN, a hypothetical steady state can be reached. Jia *et al.* (2023) use a similar framework to predict temperature in buildings with GNNs that combine GAT and gated recurrent units.

Muñoz *et al.* (2024) use hybrid twins that incorporate long short-term memory networks for temperature forecasting based on data obtained from the HLW demonstrator cell at Andra's URL. We propose to achieve a similar prediction using forward-integrating GNNs. In particular, we suggest the following transfer problem: GNNs are trained on all time steps of one of the longitudinal optic fiber. Then, given only the initial temperature distribution and the boundary conditions of the second longitudinal optic fiber, the previously trained GNN predicts the temperature at all times in the second longitudinal fiber.

The novelty of this work lies in the use time-integrating GNN for temperature forecasting based on optic fiber data from the Andra HLW demonstrator cell. The primary objective is to address a transfer problem: training on one longitudinal optic fiber and predicting the temperature distribution in the other. This transfer problem suggests that if sensors on one HLW cell fail, GNNs can be trained using data from other HLW cells with functional sensors, allowing for temperature predictions based on the most recent available temperature field.

3. Materials and methods

This section presents the data available for training our machine learning algorithm. Next, the application of the forward integration GNN for temperature prediction is discussed. Finally, a method for evaluating the performance of the GNN in the context of a transfer problem is introduced.

3.1 The training data set

As presented in Section 1, the data originates from a HLW demonstrator cell at Andra's URL. In particular, the data from the first longitudinal optic fiber is used to train our machine learning algorithms, as shown in Figure 5. Only the first half of the distributed sensor is used, as the second half is symmetrical, rendering that information redundant. In addition, the data is collected after 200 days to ensure that it does not include periods when the heat sources are not fully active. Figure 6 shows the same data represented as a time series.

Figure 6 presents the data collected from the first half of the distributed sensor at specified sample points. The sample point $x = 0p$ (the optic fiber has a spatial resolution of $p = 5cm$) corresponds to the gallery, while $x = 485$ marks the boundary between the cell and the rock. The distributed sensor has 485 measurement points.

Before training, all data presented in Figure 5 is normalized using the scheme proposed in equation (6), with T_{max} the maximum temperature and T_{min} the minimum temperature. Predictions are then generated in the normalized space and subsequently scaled back to their original values. For clarity, the normalized temperature will continue to be denoted as T; however, it is important to note that for every training and testing task, the temperature is normalized using equation (6):

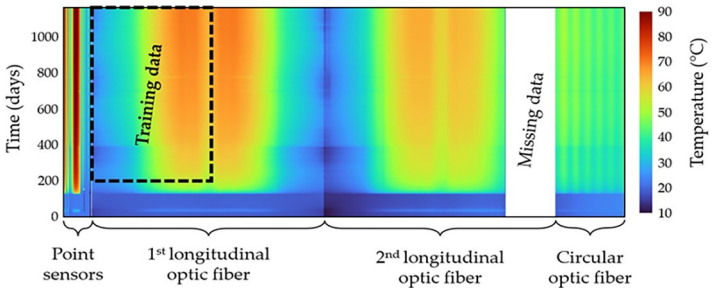

Figure 5. Subset of the data used for the training

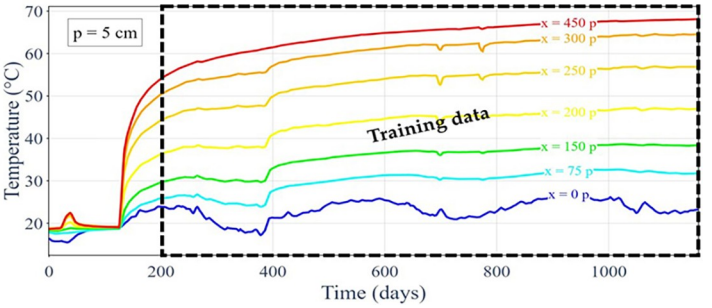

Figure 6. Temperature evolution at set sample points of the first distributed sensor

$$u(x,t) = \frac{T(x,t) - T_{min}}{T_{max}} \tag{6}$$

At any given time j (in days), the input data can be represented as a one-dimensional graph, while the corresponding output data is the same graph at time $j+1$ as shown in Figure 7. Therefore, the complete training database consists of temperature graphs from day 200 to day 1160 as inputs and temperature graphs from day 201 to day 1161 as their respective outputs.

3.2 The forward integration graph neural network model

Let us denote the GNN model as $\mathcal{G}$, $T(\chi, j)$ as the temperature of the entire graph at time j and $\widehat{T}_{\mathcal{G}}(\chi, j)$ as the GNN prediction for the temperature of the entire graph at time j. This GNN transforms the one-dimensional graph at time j into a prediction of the graph at time $j+1$ as shown in equation (7). Thus, by applying the GNN $\mathcal{G}n$ times, the predicted temperature graph at time $j+n$ is obtained:

$$\begin{aligned} \widehat{T}_{\mathcal{G}}(\chi, j+1) &= \mathcal{G}(T(\chi, j)) \\ \widehat{T}_{\mathcal{G}}(\chi, j+n) &= \underbrace{\mathcal{G}^{\circ} \ldots {}^{\circ}\mathcal{G}}_{\mathcal{G}^n} \ (T(\chi, j)) \end{aligned} \tag{7}$$

The boundary conditions are enforced at the nodes located at the extremities of the graph (v_0 and v_{485}) by taking the measured temperature at the prediction time, as established in equation (8):

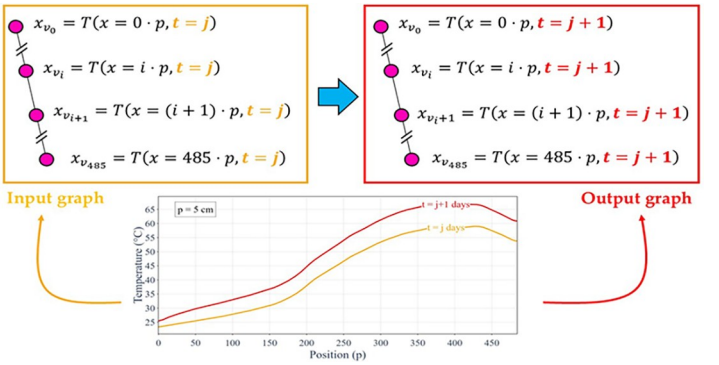

Figure 7. Training input and output graphs

$$\begin{aligned} \widehat{T}_{\mathcal{G}}(x_{v_0}, j+1) &= T(x_{v_0}, j+1) \\ \widehat{T}_{\mathcal{G}}(x_{v_{485}}, j+1) &= T(x_{v_{485}}, j+1) \end{aligned} \tag{8}$$

Let T_v^j denote the temperature at node v at time j. Equation (9) introduces the specific one-layer GCN architecture used in this paper. The activation functions σ from equation (2) are integrated in both MLPs W_0 and W_1. Additionally, MLP W_0 receives as input a source constant s_v which is dependent on the position of the node v:

$$\forall v \in V,\ \widehat{T}_v^{j+1} = \left[T_v^j,\ s_v\right] \cdot W_0 + \left[\sum_{u \in \mathcal{N}(v)} \frac{T_u^j}{c_{u,v}}\right] \cdot W_1 \tag{9}$$

Equation (10) reveals the one-layer GraphSAGE architecture that uses an average aggregation function used in this study. Similar to the GCN, the activation function σ from equation (3) is incorporated into the MLP $W$and the same source constant s_v is used:

$$\forall v \in V,\ \widehat{T}_v^{j+1} = \left[T_v^j,\ \sum_{u \in \mathcal{N}(v)} \frac{T_u^j}{|\mathcal{N}(v)|},\ s_v\right] \cdot W \tag{10}$$

Equation (11) showcases the one-layer GAT architecture that generates the results presented in this paper. It uses both an update MLP and an attention MLP, which incorporate multiple activation functions σ. Additionally, the update MLP receives the source constant s_v as input:

$$\begin{aligned} \forall v \in V,\ \widehat{T}_v^{j+1} &= \left[T_v^j \cdot a_{v,v}^j + \sum_{u \in \mathcal{N}(v)} T_u^j \cdot a_{u,v}^j,\ s_v\right] \cdot W \\ \forall v \in V,\ \forall u \in \mathcal{N}(v),\ a_{u,v}^j &= \frac{\left[T_u^j,\ T_v^j\right] \cdot A}{\sum_{w \in \mathcal{N}(v)} \left(\left[T_v^j,\ T_w^j\right] \cdot A\right)} \end{aligned} \tag{11}$$

Table 1 presents the selected hyper-parameters for each architecture (GCN, GraphSAGE and GAT). The sizes of the MLPs (W, A) indicate the number of neurons per layer of the MLP, while the activation function used for each layer is represented by MLP activations. To identify the best hyper-parameters, a small hyper-parametric study has been conducted. This study compared GNN results with varying numbers of neuron layers, numbers of neurons per layer and a few activation functions. Then, the architectures with the best performances have been identified.

The loss function used to train our GNNs is the mean squared error (MSE). It is calculated by comparing the predicted output denoted as $\widehat{y}$ and the expected output y as introduced in equation (12):

$$\begin{aligned} &\mathcal{L}(y,\widehat{y}) = \frac{1}{n_y}\cdot \sum_{i=1}^{n_y}(y_i-\widehat{y}_i)^2 \\ &y = T(\chi, j+1) \;\&\; \widehat{y} = \widehat{T}_{\mathcal{G}}(\chi, j+1) = \mathcal{G}(T(\chi, j)) \end{aligned} \tag{12}$$

The learning parameters used are:

- the optimizer is Adam stochastic gradient descent algorithm (Kingma and Ba, 2017);
- 1,000 training epochs in total;
- a learning rate of 10^{-3} for the first 500 training epochs and of 10^{-4} for the last 500;
- a batch size of 100 per iteration of the gradient descent algorithm;
- a validation split of 15%; and
- the training database is shuffled before training.

All the neural networks used in this paper were implemented using the Keras Python library developed by Chollet (2015).

3.3 The testing framework

The testing is conducted on half of the measurements (due to symmetry) from the second linear optic fiber, as illustrated in Figure 8. More precisely, the temperature graph from the second distributed sensor at 200 days is used as the sole input. The testing data was chosen different from the training data, see Figure 5, for two main reasons: to ensure that the trained GNN is not subject to overfitting and to show the potential for generalization for the task at hand (transfer learning). The boundary values for nodes v_0 and v_{485} are also provided at all time steps.

The objective is to apply the GNN repeatedly to generate predictions for the entire test data set, as presented in Figure 9. Then, these predictions will be compared to the measured temperature using relative error, denoted as ε_r. This relative error is calculated based on the real temperature T and the predicted temperature $\widehat{T}$, as defined in equation (13):

Table 1. Architectures description

Architecture	MLP sizes	MLP activations	Number of unique MLPs
GCN	8,8,1	ReLU, ReLU, linear	$2 \rightarrow W_0, W_1$
GraphSAGE (avg)	8,8,1	ReLU, ReLU, linear	$1 \rightarrow W$
GAT	8,8,1	ReLU, ReLU, linear	$2 \rightarrow W, A$

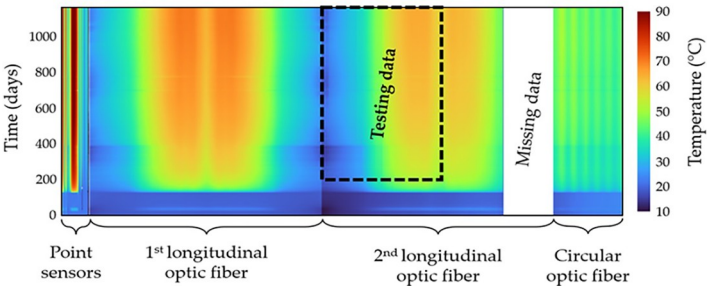

Figure 8. Data used to test GNNs

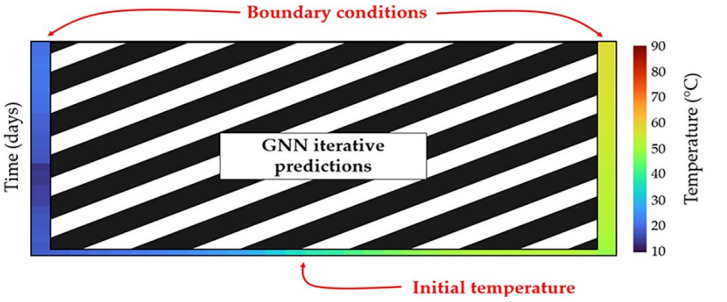

Figure 9. Testing framework

$$\varepsilon_r = \frac{T(x,\,t) - \widehat{T}(x,\,t)}{T(x,\,t)} \tag{13}$$

3.4 Alternative testing frameworks

Two alternative testing frameworks that use the same test data set are proposed. These frameworks reflect practical realities and incorporate additional data.

The first framework addresses a partial failure of the distributed sensor. When an optic fiber breaks, the sensor responses recorded before the disconnection remain consistent with the measurements, while all responses recorded afterward become invalid. Therefore, we propose simulating a break in the middle of the distributed sensor; the break happens at a set point in the middle of the distributed sensor. The predictions for the first half are replaced by the actual sensor responses, while the second half is predicted by the machine learning algorithm, as shown in Figure 10. The data replacement operates as follows: after a prediction for the temperature is performed, the data is replaced by the actual sensor responses before making the next prediction.

The second framework is informed by multiple point sensors located around the HLW demonstrator cell, as depicted in Figure 2. In this case, the sensors have, respectively, been placed at the quartiles of the distributed sensor. These sensors can be used to replace the algorithmic predictions at specific locations along the distributed sensor. In this scenario, we consider three-point sensors evenly distributed along the optic fiber, as illustrated in Figure 11.

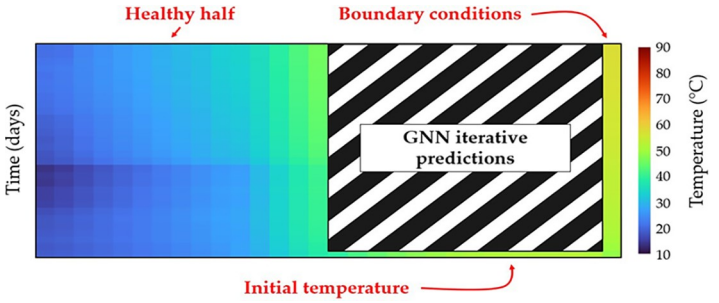

Figure 10. Alternative testing framework: partial failure of the distributed senso

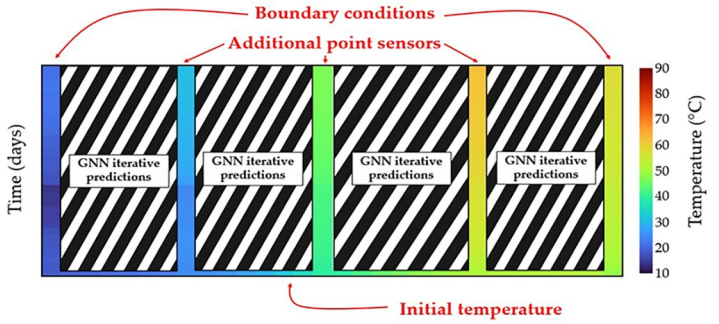

Figure 11. Alternative testing framework: three additional point sensors

3.5 Comparative methods

Other machine learning algorithms are trained on the same database and tested using the same framework.

The first method used for comparison is kriging (Le Riche, 2014; Le Riche and Durrande, 2019), which takes the initial temperature distribution and the boundary conditions as inputs, as highlighted in Figure 9. This method is implemented using the PyKrige Python library developed by Murphy (2014) with an exponential variogram. Since this method is not trained, it will serve as a baseline for performance evaluation.

The second method tested is a straightforward time-integrating artificial neural network (ANN) (Aggarwal, 2018; Russell *et al.*, 2020) that is trained using the same data. However, instead of using a graph, it uses a vector that represents the temperature field at a specific time j and predicts the temperature field at time $j + 1$. This neural network consists of two hidden layers, each containing 3,880 neurons, that uses the ReLU activation function. It is trained with the same learning parameters as the GNNs (i.e. same loss, optimizer, epochs, learning rate, batch size, validation split and shuffle).

The third comparative method used is a Gaussian process (GP) (Rasmussen *et al.*, 2005). It is implemented using the scikit-learn Python library developed by Pedregosa *et al.* (2011). Additionally, this method is used as a time integrator, which uses the same inputs and outputs as the ANN. The kernel used in this work is the sum of a Matern kernel and a white kernel. The alpha is set at the default value ($\alpha = 10^{-10}$) and the default optimizer is used.

4. Results

This section presents the predicted temperatures and the associated errors for the various machine learning algorithms. The fitting curves resulting from the training of our three GNN models are available in Appendix.

4.1 Graph convolutional network

Figure 12 presents a side-by-side comparison of the temperature measured in the second longitudinal optic fiber on the left and the temperature predicted by the GCN on the right. Both figures exhibit similar trends; however, the GCN appears to underestimate the temperature in the right section of the optic fiber. Furthermore, the GCN prediction does not perfectly align with the observed temperature in the left portion of the graph (closer to the gallery).

Then, by applying the relative error ε_r presented in equation (13) to each data point, the relative error heatmap shown in Figure 13 is generated. For the majority of the points, the relative error ranges between 0% and 10%, except in the left portion of the graph, where it is higher and can locally reach up to 45%. In fact, in this area, we can notice that the real temperature fields shown in Figure 12 are not smooth and exhibit artifacts resulting from natural temperature variations and closing of the demonstrator cell. These artifacts cannot (and should not) be possibly predicted by a machine learning algorithm, as it lacks confirmation regarding this action. Nevertheless, the network is capable of accurately predicting both the evolution and the final state.

Finally, we can compute the MSE across the entire optic fiber (for every node in the graphs) at each time step *j* as shown in equation (14). This results in a time series of errors for a given time, as shown in Figure 14. The initial error is zero because the initial temperature is

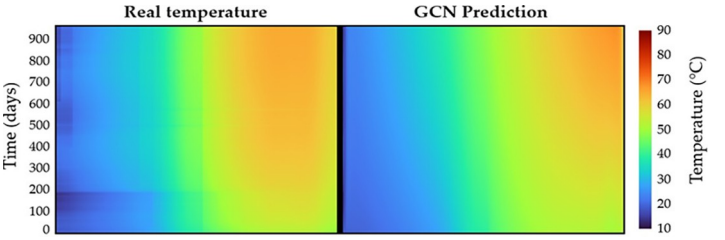

Figure 12. Side-by-side comparison of the real and GCN-predicted temperature

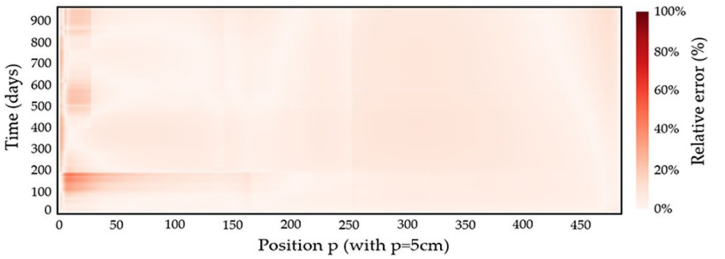

Figure 13. Relative error between the real and the GCN-predicted temperature

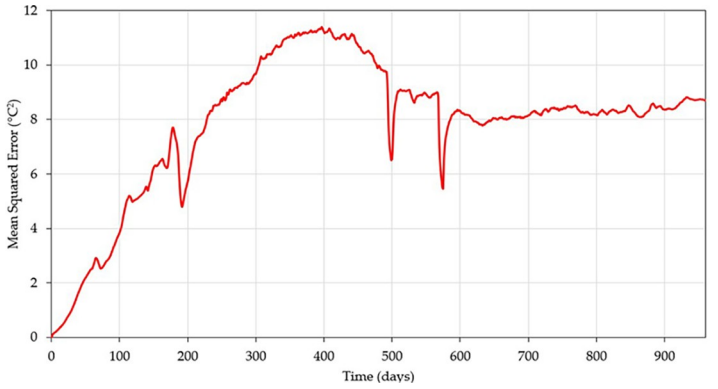

Figure 14. Mean squared error between the real and the GCN-predicted temperature for every time step

provided as an input to the GCN. A spike can be observed between 250 and 450 days. Following this period, the error starts to stabilize around $9^{\circ}C^2$. The maximum MSE is of $12^{\circ}C^2$ which corresponds to an average deviation of about $4^{\circ}C$:

$$MSE(j) = \frac{1}{485} \sum_{v \in V} \left(\widehat{T}_v^j - T_v^j \right)^2 \tag{14}$$

4.2 GraphSAGE

The same methodology can be applied to the GraphSAGE-based GNN. Figure 15 showcases the GraphSAGE predictions on the right, compared to the measured temperature on the left. The results demonstrate a trend similar to the GCN, with a slight underestimation of the temperature on the right and noticeable discrepancies in the predictions on the left.

Figure 16 provides the relative error for the GraphSAGE temperature prediction. Similar to the GCN, the error predominantly ranges between 0% and 10%, with the exception of the bottom left region (around 200 days), where the error is lower than that of the GCN, peaking at only 35%. The same observations apply here as well, where the errors are amplified due to experimental artifacts in the real temperature.

Figure 17 illustrates the MSE for the GraphSAGE temperature predictions across all time steps. Like for the GCN, the GraphSAGE exhibits an increase in error until 400 days, after which the error stabilizes. The maximum MSE is around $9^{\circ}C^2$ indicating an average error of $3^{\circ}C$.

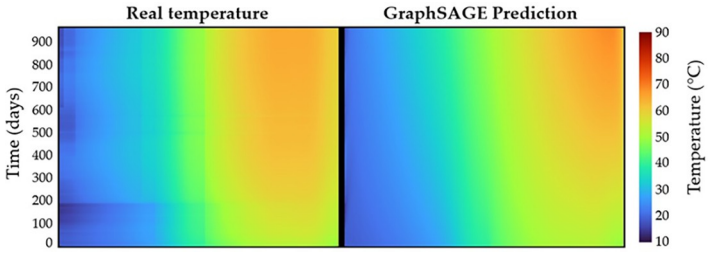

Figure 15. Side-by-side comparison of the real and GraphSAGE predicted temperature

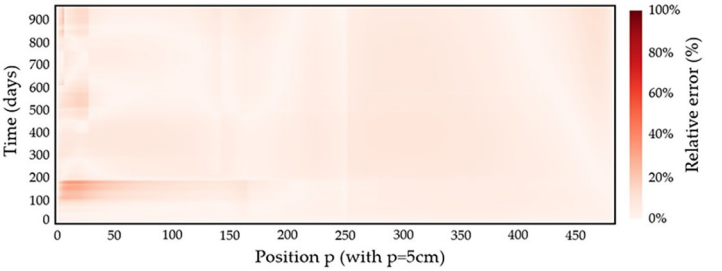

Figure 16. Relative error between the real and the GraphSAGE predicted temperature

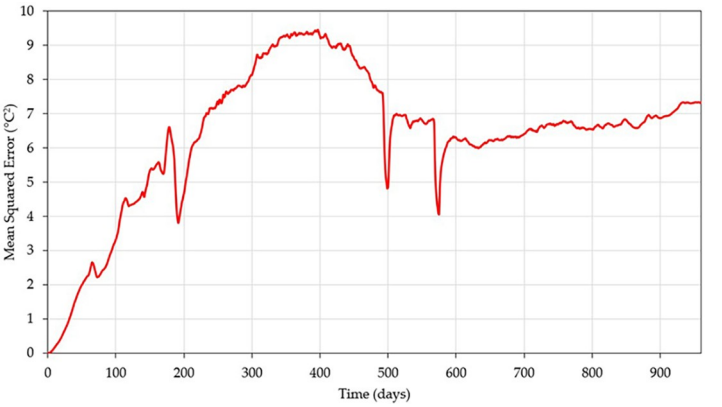

Figure 17. Mean squared error between the real and the GraphSAGE predicted temperature for every time step

4.3 Graph attention network

The same process is then applied to the GAT. Figure 18 presents a side-by-side comparison between the GAT predictions on the right and the actual temperature measured by the distributed sensor. The results are comparable to those obtained by the GCN and GraphSAGE.

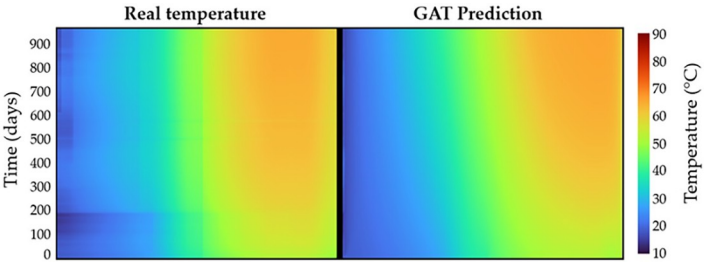

Figure 18. Side-by-side comparison of the real and GAT predicted temperature

The GAT generally outperforms the GCN and GraphSAGE in predicting temperature, as highlighted in Figure 19, where the error on the right-side peaks at approximately 6%. Moreover, the errors on the left side around the 200-day mark are comparable to those of GraphSAGE, with a local peak error of 30%. We also request that the reader take note of the experimental artifacts present in the measurements.

The maximum MSE is approximately $5\,^{\circ}C^2$ between 200 and 400 days, after which it stabilizes at around $3\,^{\circ}C^2$, indicating an error of less than $2\,^{\circ}C$. Figure 20 presents the evolution of MSE for the GAT.

4.4 Comparison between the various algorithms

Finally, Figure 21 compares the MSE of predictions from three GNN architectures as well as from ANN, GP and kriging. It is evident that GAT outperforms the other two GNN architectures. Kriging exhibits the lowest precision, with errors around $10\,^{\circ}C$. While ANN and GP outperform GNNs between 200 and 500 days, GAT yields similar or better results at other time steps. Table 2 presents the average of the time series of the MSE.

As far as training time is concerned, the machine learning algorithms can be categorized into three groups: GP, kriging, GCN and GraphSAGE, which train within 5–20 min, GAT, which can be trained under an hour and ANN, which require more than an hour.

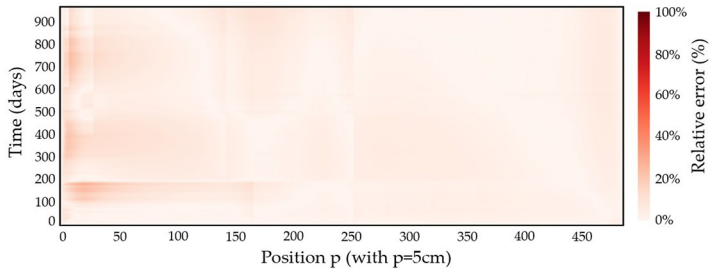

Figure 19. Relative error between the real and the GAT predicted temperature

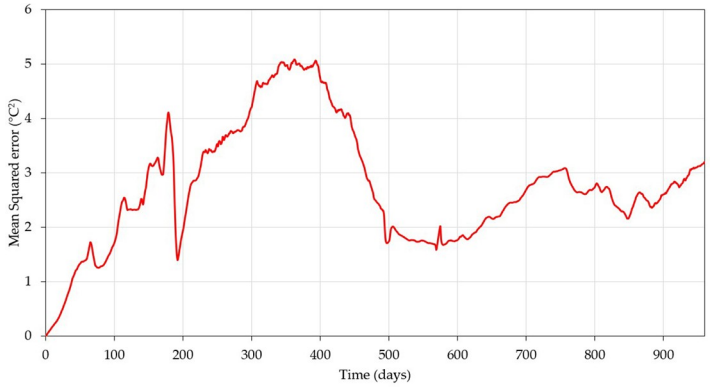

Figure 20. Mean squared error between the real and the GAT predicted temperature for every time step

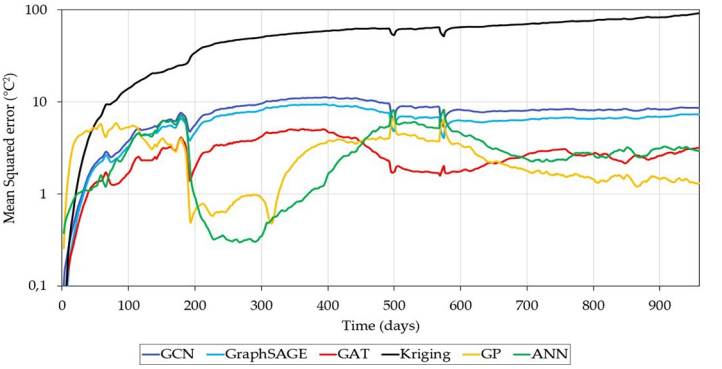

Figure 21. Comparison of the mean squared errors of the various machine learning algorithms

Table 2. Average mean squared error

Kriging	GCN	GraphSAGE	ANN	GAT	GP
55.08° C^2	7.87° C^2	6.39° C^2	2.87° C^2	2.75° C^2	2.64° C^2

Additionally, observe the significantly smaller dimensions of the GNN presented in Table 1, in contrast to the much larger size of the selected ANN, which consists of two dense layers, each containing 3,880 neurons.

The results for the alternative testing case are presented in Figures 22 and 23. GCN and GraphSAGE were excluded from the analysis because their performance was far inferior to that of GAT. Table 3 and Figure 22 indicate that, in the event of a partial failure of the distributed sensor, both GAT and ANN outperform the other options. The ANN has a slight advantage over the GAT, although it requires a significantly larger number of trainable parameters and longer training time. In addition, the precision of the prediction in the event

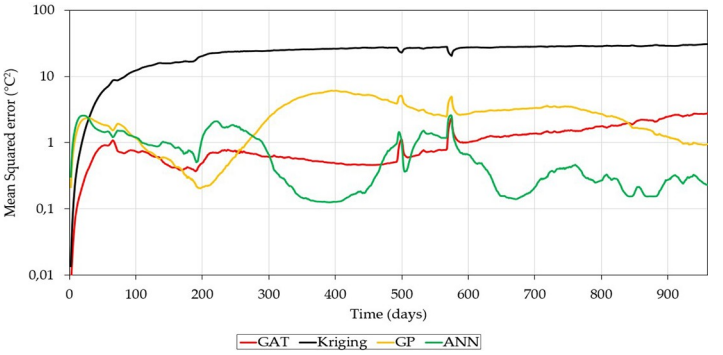

Figure 22. Comparison of the mean squared errors for the partial failure of the distributed sensor

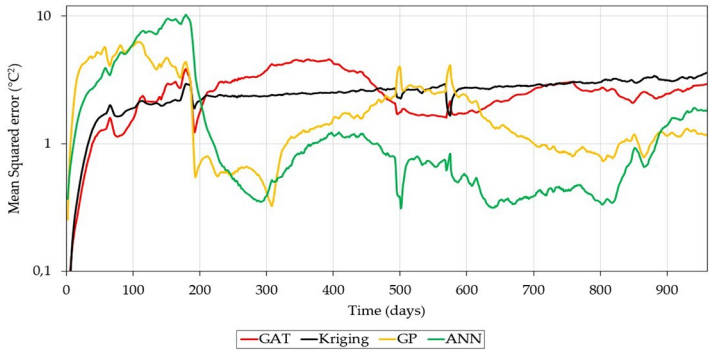

Figure 23. Comparison of the mean squared errors using three additional point sensors

Table 3. Average mean squared error for the partial failure of the distributed sensor

Kriging	GP	GAT	ANN
$23.46^{\circ}C^2$	$2.59^{\circ}C^2$	$1.06^{\circ}C^2$	$0.70^{\circ}C^2$

of a partial failure is three times more accurate than in the case of a complete failure (see Table 2 and Figure 21).

Table 4 and Figure 23 indicate that using more control points results in a slight increase in precision, except for kriging, which improves tenfold and that all algorithms perform similarly.

5. Discussion

The goal of this section is to analyze the results presented in Section 4, discuss the generalizability of the proposed model and highlight the limitations of GNNs in the context of temperature forecasting.

5.1 Appeal of graph neural networks for temperature forecasting

GNNs have proven to be a reliable tool for predicting temperature changes in distributed sensors. They can forecast temperatures up to 1,000 days in advance with an accuracy of a couple of degrees Celsius ($^{\circ}C$). GNNs significantly outperform kriging methods and yield results comparable to those of ANN and GP, which require a much larger number of training parameters. Although the GNNs performance is slightly inferior for intermediate time steps (between 200 and 500 days), they perform similarly or better for other time steps, as illustrated by Figure 21. This may raise questions about the utility of GNNs for temperature

Table 4. Average mean squared error using 3 additional point sensors

GAT	Kriging	GP	ANN
$2.55^{\circ}C^2$	$2.54^{\circ}C^2$	$1.96^{\circ}C^2$	$1.74^{\circ}C^2$

forecasting; however, it is important to note that our analysis was conducted using one-dimensional data. With more complex topological data, GNNs are expected to surpass the performance of ANN and GP due to their ability to leverage local relationships. Furthermore, GNNs can generalize to entirely different graphs (in terms of structure and size) by leveraging the individual MLPs used in the message-passing process. In contrast, ANN and GP would require retraining if any structural changes were to occur. In addition, GNNs demonstrate strong performances even with small message-passing MLPs (comprising only tens of neurons), highlighting their capacity to generalize and incorporate physical elements effectively.

In addition, the time-integrating scheme is relevant for predicting temperature in distributed sensors for two main reasons. First, when an optic fiber snaps, all values recorded after the fracture become unusable, necessitating a prediction for a continuous field. Second, the fracture is irreversible, indicating that future predictions cannot be made from the damaged section. Therefore, it is logical to use only the last accurate temperature measurement to inform future predictions, with boundary conditions provided by point sensors. Moreover, the monitoring of Andra's Cigeo project will result in a substantial amount of data available for training our machine learning algorithm.

Eventually, in the event of a partial failure of the distributed sensor (see Figure 22 and Table 3), the GAT outperforms the GP and performs slightly worse than the ANN. This is noteworthy considering that both the GP and ANN use the complete temperature field, which includes the segment composed of actual sensor responses for making predictions, while the GNN can only leverage information from neighboring nodes. Furthermore, when monitoring radioactive waste disposal facilities, a monitoring tool that can be transferred to various geometries without requiring new training is preferred. For all these reasons, temperature forecasting GNNs are prime candidates for data completion in this particular case.

5.2 Inconsistencies near the gallery

GNNs exhibit relatively poor predictive performance near the gallery (on the left). This is evident by the spike in relative error on the left side of Figures 13, 16 and 19. This phenomenon can be attributed to two factors. First, the temperature on the left side is significantly lower (approximately $15^{\circ}C$) than that of the rest of the field, resulting in similar absolute errors that lead to larger relative errors. Second, the boundary conditions near the gallery (on the left) are inconsistent due to natural temperature variations. This can be observed on the dark blue curve ($x = 0p$) in Figure 6.

This is further demonstrated by comparing the predictions in the event of a complete failure (see Table 2 and Figure 21) to those in the event of a partial failure (see Table 3 and Figure 22). Indeed, if the behavior near the gallery is understood (i.e. when the failure is partial as described in Figure 10), the GAT error is reduced by a factor of three. This indicates that the majority of the errors can be attributed to the inconsistencies near the gallery.

5.3 Inconsistencies caused by closing the gallery

At the 200-day mark in the training and testing data, the HLW cell was sealed. This is evident and can be observed around the 400-day mark in Figure 6 (because the data used in the model starts at 200 days), where the temperature rises slightly more rapidly due to the insulation provided by the sealing element. This alteration in the model's physics results in an increase in the GNN prediction error between 200 and 400 days, as illustrated in Figures 14, 17, 19 and 21. The error is lower for ANN and GP because they can leverage the entire temperature field, whereas GNNs can only leverage information from neighboring nodes.

Excluding the region most affected by the sealing of the HLW cell (i.e. the area closest to the gallery), as seen in the case of a partial failure presented in Figure 10, leads to a reduction in the GAT error. This is illustrated in Figure 22, where the GAT error is comparable to the errors of the ANN and GP between 200 and 400 days. This explains the gap in errors between ANN, GP and GNNs from the 200-day mark and the 400-day mark in Figure 21.

5.4 Architecture advantages and drawbacks

Figure 21 shows that three GNN architectures were tested. Among them, the GAT produced the best results, but at a significant computational cost for both training and prediction. GraphSAGE demonstrated the second-best performance while being the least computationally intensive of the three architectures. For reference, the training time of the GAT is 5–10 times longer than that of GraphSAGE.

In the context of monitoring the HLW cell, the sensors have a sampling rate of one day. This indicates that the computational cost of the GAT does not hinder its ability to be trained and used for online temperature forecasting.

5.5 Scaling the model to more complex graphs

A logical next step is to extend the graph to include circular optical fibers and subsequently to encompass the entire sensor network used in the HLW demonstrator cell's thermal loading, as illustrated in Figure 2. In the future, this will enable the transfer of GNNs from a HLW cell equipped with functional sensors to another cell that contains defective sensors. The two primary challenges are the creation of the sensor graph and the adaptation of the GNN models to these more complex graphs.

The primary hurdle lies in constructing the graph of sensors based on their locations and the elements they are mounted on (i.e. the rock and the sleeves). This task is particularly complex because it necessitates a standardized process for connecting the various nodes according to their positions, while ensuring that the resulting graph is neither too dense nor too sparse. Indeed, a graph that is overly dense restricts the topological information and increases the computational cost of training and applying GNNs. Conversely, a graph that is too sparse results in inefficient message passing. Furthermore, considering the elements on which the sensors are mounted adds another layer of complexity. One potential solution is to create relational graphs (Schlichtkrull *et al.*, 2017), incorporating edge embeddings that depend on the elements to which both sensors are mounted.

Once the complete graph is created, minimal adjustments are necessary for the existing GNN model to operate effectively on the more complex graph. In fact, the MLPs used for message passing can be used to construct the new GNNs. Additional training is required to adapt to the new topological constraints; however, aside from that, the transition is relatively straightforward. Among all architectures, GraphSAGE and the graph isomorphism network (Xu *et al.*, 2019) are the most suitable for the complete graph, as they are the least computationally intensive.

5.6 Limitations of graph neural networks for the considered problem

The primary limitations of GNNs for temperature forecasting are the computational cost and the time required to develop optimal architectures. While training a single GNN may be relatively inexpensive, achieving the best results necessitates training multiple GNNs for each set of hyper-parameters and selecting the best performer. For more complex GNNs with multiple layers of message passing, this hyper-parameter optimization demands costly benchmarks. Furthermore, even after selecting the ideal hyper-parameters, it remains necessary to train multiple GNNs due to the randomness associated with the initial weights and the back-propagation algorithm. The process of choosing an optimal GNN architecture

could be streamlined by using optimization algorithms such as particle swarm optimization (PSO) (Carvalho and Ludermir, 2007; Fernandes *et al.*, 2019), genetic algorithms (Kapanova *et al.*, 2018; Idrissi *et al.*, 2016) or other optimization techniques (Strumberger *et al.*, 2019; Ezzat *et al.*, 2021; Lankford and Grimes, 2024).

The second limitation is the amount of random access memory (RAM) and processing power required when working with large graphs. This issue can be mitigated by using subgraph learning (Zha and Yilmaz, 2023) combined with multi-threading. in conjunction with multi-threading. Subgraph learning involves creating a set of subgraphs from the entire graph and training the GNN on these subgraphs prior to reconstruction for making predictions. This approach significantly reduces the amount of RAM needed and is particularly well-suited for multi-threading applications. Moreover, when handling larger graphs, GAT models become highly computationally intensive, despite yielding the best results, which creates a dilemma between computational power and precision.

Another limitation of GNNs is the complexity of their implementation. As explained in subsection 5.5, GNNs require the construction of a graph, which can be a complex and tedious process. Additionally, selecting the most efficient message-passing mechanism is crucial. In contrast, approaches such as GP, ANN or convolutional neural networks typically require significantly less data curation and are generally simpler to implement.

6. Conclusions

In this study, we propose a process for creating temperature forecasting GNNs that ensure continuous monitoring of radioactive waste disposal facilities, particularly when distributed sensors are compromised. These GNNs yield results comparable to established models such as Kriging, ANN and GP, demonstrating their effectiveness in temperature forecasting. While the GAT architecture produces the best results, it is also more computationally intensive. Despite certain limitations, including implementation complexity and high computational costs, GNNs leverage topological and physical information, enabling them to be easily transferred to significantly different graphs. This adaptability makes them an excellent choice for monitoring multiple similar HLW cells with minor variations.

Further improvements can be made to refine our models. First, an algorithm to mitigate errors caused by evolving boundary conditions can be developed. Next, more sophisticated GNN models that use multiple layers of message-passing can be explored to enhance the accuracy of the results. The intricate architecture can then be optimized using traditional optimization techniques such as PSO, and the training parameters (e.g. optimizer and learning rate) can be fine-tuned to ensure optimal performance. Ultimately, these advanced models could be applied to the entire network of sensors monitoring the HLW cell, and a strategy for transferring GNN capabilities from one cell to another could be investigated.

Acknowledgements

The following Python libraries were instrumental in developing these models: NumPy (Harris *et al.*, 2020), Keras (Chollet, 2015), Plotly (Plotly Technologies Inc, 2015) and scikit-learn (Pedregosa *et al.*, 2011). The authors acknowledge the use of the Cassiopee Arts et Métiers Institute of Technology HPC Center made available for conducting the research reported in this paper. The authors acknowledge the role played by Andra.

References

Aggarwal, C.C. (2018), *Neural Networks and Deep Learning: A Textbook*, Springer, Google-Books-ID: achqDwAAQBAJ.

Boussaid, T., Rousset, F., Scuturici, V.-M. and Clausse, M. (2023), "Evaluation of graph neural networks as surrogate model for district heating networks simulation", In 36th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS 2023), pp. 3182-3193, *Las Palmas De Gran Canaria, Spain*, ECOS 2023.

Bruna, J., Zaremba, W., Szlam, A. and LeCun, Y. (2014), "Spectral networks and locally connected networks on graphs", arXiv:1312.6203.

Bumbieler, F., Plúa, C., Vu, M.-N. and Armand, G. (2024), "Assessment of a full-scale in-situ heater experiment based on the French high-level waste disposal cell concept conducted in the Callovo-Oxfordian claystone", *Tunnelling and Underground Space Technology*, Vol. 153, p. 106004.

Carvalho, M. and Ludermir, T.B. (2007), "Particle swarm optimization of neural network architectures and weights", In 7th International Conference on Hybrid Intelligent Systems (HIS 2007), pp. 336-339.

Chen, Z., Chen, F., Zhang, L., Ji, T., Fu, K., Zhao, L., Chen, F., Wu, L., Aggarwal, C. and Lu, C.-T. (2021), "Bridging the gap between spatial and spectral domains: a survey on graph neural networks", arXiv:2002.11867 [cs, stat].

Chollet, F. (2015), "Keras: Deep learning for humans".

Cremer, J., Medrano Sandonas, L., Tkatchenko, A., Clevert, D.-A. and De Fabritiis, G. (2023), "Equivariant graph neural networks for toxicity prediction", *Chemical Research in Toxicology*, Vol. 36 No. 10, pp. 1561-1573.

Daigavane, A., Ravindran, B. and Aggarwal, G. (2021), "Understanding convolutions on graphs", *Distill*, Vol. 6 No. 8, p. e32.

Defferrard, M., Bresson, X. and Vandergheynst, P. (2017), "Convolutional neural networks on graphs with fast localized spectral filtering", arXiv:1606.09375 [cs, stat].

Dehmamy, N., Barabási, A.-L. and Yu, R. (2019), "Understanding the representation power of graph neural networks in learning graph topology", arXiv:1907.05008 [physics, stat].

Ezzat, D., Hassanien, A.E. and Ella, H.A. (2021), "An optimized deep learning architecture for the diagnosis of COVID-19 disease based on gravitational search optimization", *Applied Soft Computing*, Vol. 98, p. 106742.

Fernandes, F., Junior, E. and Yen, G.G. (2019), "Particle swarm optimization of deep neural networks architectures for image classification", *Swarm and Evolutionary Computation*, Vol. 49, pp. 62-74.

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O. and Dahl, G.E. (2017), "Neural message passing for quantum chemistry", arXiv:1704.01212.

Hamilton, W.L., Ying, R. and Leskovec, J. (2018), "Inductive representation learning on large graphs", arXiv:1706.02216.

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T.E. (2020), "Array programming with NumPy", *Nature*, Vol. 585 No. 7825, pp. 357-362.

Hembert, P., Ghnatios, C., Cotton, J. and Chinesta, F. (2024), "Assessing sensor integrity for nuclear waste monitoring using graph neural networks", *Sensors*, Vol. 24 No. 5, p. 1580.

Henaff, M., Bruna, J. and LeCun, Y. (2015), "Deep convolutional networks on Graph-Structured data", arXiv:1506.05163.

Hoang, V.T., Jeon, H.-J., You, E.-S., Yoon, Y., Jung, S. and Joun Lee, O. (2023), "Graph representation learning and its applications: a survey", *Sensors*, Vol. 23 No. 8, p. 4168.

Idrissi, M.A.J., Ramchoun, H., Ghanou, Y. and Ettaouil, M. (2016), "Genetic algorithm for neural network architecture optimization", In 2016 3rd International Conference on Logistics Operations Management (GOL), pp. 1-4.

Jia, Y., Wang J, Hosseini M.R, Shou W, Wu P. and Mao, C. (2023), "Temporal graph attention network for building thermal load prediction", *Energy and Buildings*, Vol. 321, p. 113507.

Kapanova, K.G., Dimov, I. and Sellier, J.M. (2018), "A genetic approach to automatic neural network architecture optimization", *Neural Computing and Applications*, Vol. 29 No. 5, pp. 1481-1492.

Khemani, B., Patil, S., Kotecha, K. and Tanwar, S. (2024), "A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions", *Journal of Big Data*, Vol. 11 No. 1, p. 18.

Kingma, D.P. and Ba, J. (2017), "Adam: a method for stochastic optimization", arXiv:1412.6980 [cs].

Kipf, T. N. and Welling, M. (2017), "Semi-Supervised classification with graph convolutional networks", arXiv:1609.02907 [cs, stat].

Knyazev, B., Taylor, G.W. and Amer, M.R. (2019), "Understanding attention and generalization in graph neural networks", arXiv:1905.02850.

Lankford, S. and Grimes, D. (2024), "Neural architecture search using particle swarm and ant colony optimization", arXiv:2403.03781 [cs].

Le Riche, R. (2014), "Introduction to kriging".

Le Riche, R. and Durrande, N. (2019), "An overview of kriging for researchers".

Maurya, S.K., Liu, X. and Murata, T. (2021), "Simplifying approach to node classification in graph neural networks", arXiv:2111.06748 [cs, stat].

McCallum, A.K., Nigam, K., Rennie, J. and Seymore, K. (2000), "Automating the construction of internet portals with machine learning", *Information Retrieval*, Vol. 3 No. 2, pp. 127-163.

Mozaffar, M., Liao, S., Lin, H., Ehmann, K. and Cao, J. (2021), "Geometry-agnostic data-driven thermal modeling of additive manufacturing processes using graph neural networks", *Additive Manufacturing*, Vol. 48, p. 102449.

Muñoz, D., Thomas, A.E., Cotton, J., Bertrand, J. and Chinesta, F. (2024), "Hybrid twins modeling of a high-level radioactive waste cell demonstrator for long-term temperature monitoring and forecasting", *Sensors*, Vol. 24 No. 15, p. 4931.

Murphy, B.S. (2014), "PyKrige: development of a kriging toolkit for Python. 2014:H51K–0753", Conference Name: AGU Fall Meeting Abstracts ADS Bibcode: 2014AGUFM.H51K0753M.

Nguyen, H.X., Zhu, S. and Liu, M. (2022), "A survey on graph neural networks for microservice-based cloud applications", *Sensors*, Vol. 22 No. 23, p. 9492.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, M., Brucher, M., Perrot, M. and Duchesnay, É. (2011), "Scikit-learn: machine learning in Python", *Journal of Machine Learning Research*, Vol. 12 No. 85, pp. 2825-2830.

Peter, W., Battaglia, J.B., Hamrick, V., Bapst, A., Sanchez-Gonzalez, V., Zambaldi, M., Malinowski, A., Tacchetti, D., Raposo, A., Santoro, R., Faulkner, C., Gulcehre, F., Song, A., Ballard, J., Gilmer, G., Dahl, A., Vaswani, Kelsey, A., Charles, N., Victoria, L., Chris, D., Nicolas, H., Daan, W., Pushmeet, K., Matt, B., Oriol, V., Yujia, L. and Razvan, P. (2018), "Relational inductive biases, deep learning, and graph networks", arXiv:1806.01261 [cs, stat].

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A. and Battaglia, P.W. (2021), "Learning Mesh-Based simulation with graph networks", arXiv:2010.03409 [cs].

Plotly Technologies Inc (2015), "Plotly technologies inc. Plotly: collaborative data science".

Procaccini, M., Sahebi, A. and Giorgi, R. (2024), "A survey of graph convolutional networks (GCNs) in FPGA-based accelerators", *Journal of Big Data*, Vol. 11 No. 1, p. 163.

Rasmussen, C.E., Christopher. and K.I., Williams. (2005), *Gaussian Processes for Machine Learning*, MIT Press, Google-Books-ID: GhoSngEACAAJ.

Russell, S.J., Russell, S. and Norvig, P. (2020), *Artificial Intelligence: A Modern Approach*, Pearson, Google-Books-ID: koFptAEACAAJ.

Sanchez-Lengeling, B., Reif, E., Pearce, A. and Wiltschko, A.B. (2021), "A gentle introduction to graph neural networks", *Distill*, Vol. 6 No. 8, p. e33.

Sanchis-Alepuz, H. and Stipsitz, M. (2022), "Towards real time thermal simulations for design optimization using graph neural networks", arXiv:2209.13348 [physics].

Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I. and Welling, M. (2017), "Modeling relational data with graph convolutional networks", arXiv:1703.06103 [cs, stat].

Strumberger, I., Tuba, E., Bacanin, N., Zivkovic, M., Beko, M. and Tuba, M. (2019), "Designing convolutional neural network architecture by the firefly algorithm", In 2019 International Young Engineers Forum (YEF-ECE), pp. 59-65.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y. (2018), "Graph attention networks", arXiv:1710.10903. [cs, stat].

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P. and Yu, P.S. (2021), "Heterogeneous graph attention network", arXiv:1903.07293 [cs].

Wang, Y. (2022), "Friend recommendation using GraphSAGE".

Wu, F., Zhang, T., Holanda de Souza, A., Jr, Fifty, C., Yu, T. and Weinberger, K.Q. (2019a), "Simplifying graph convolutional networks", arXiv:1902.07153 [cs, stat].

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P.S. (2019b), "A comprehensive survey on graph neural networks", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 32 No. 1, pp. 4-24, doi: 10.1109/TNNLS.2020.2978386.

Xu, K., Hu, W., Leskovec, J. and Jegelka, S. (2019), "How powerful are graph neural networks?", arXiv:1810.00826 [cs, stat].

Yang, Z., Gaidhane, A.D., Drgoňa, J., Chandan, V., Halappanavar, M.M., Liu, F. and Cao, Y. (2024), "Physics-constrained graph modeling for building thermal dynamics", *Energy and AI*, Vol. 16, p. 100346.

Zha, B. and Yilmaz, A. (2023), "Subgraph learning for topological geolocalization with graph neural networks", *Sensors*, Vol. 23 No. 11, p. 5098.

Zhang, S., Tong, H., Xu, J. and Maciejewski, R. (2019), "Graph convolutional networks: a comprehensive review", *Computational Social Networks*, Vol. 6 No. 1, p. 11.

Zhang, Z., Cui, P. and Zhu, W. (2020), "Deep learning on graphs: a survey", arXiv:1812.04202 [cs, stat].

Zheng, X., Wang, Y., Liu, Y., Li, M., Zhang, M., Jin, D., Yu, P.S. and Pan, S. (2022), "Graph neural networks for graphs with heterophily: a survey", arXiv:2202.07082 [cs].

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C. and Sun, M. (2020), "Graph neural networks: a review of methods and applications", *AI Open*, Vol. 1, pp. 57-81.

Corresponding author
Pierre Hembert can be contacted at: pierre.hembert@ensam.eu

Appendix. Training history and fitting curves

In this section, we present the fitting curves for the GCN, GraphSAGE and GAT.

Figure A1 presents the evolution of the loss, the loss on the validation data and the learning rate for the GCN. Given that the loss is higher than the validation loss, there is most likely no overfitting. Moreover, the loss seems to have converged at 800 epochs.

Figure A2 presents the evolution of the loss, the loss on the validation data and the learning rate for the GraphSAGE. Given that the loss is higher than the validation loss, there is most likely no overfitting. Moreover, the loss seems to have converged at 100 epochs.

Figure A3 presents the evolution of the loss, the loss on the validation data and the learning rate for the GAT. Given that the loss is higher than the validation loss, there is most likely no overfitting. Moreover, the loss seems to have converged at 200 epochs.

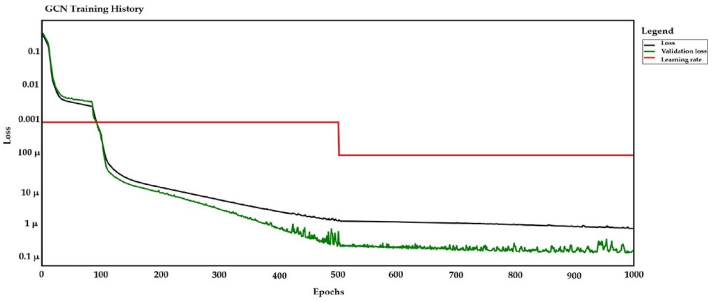

Figure A1. Fitting curves of the GCN

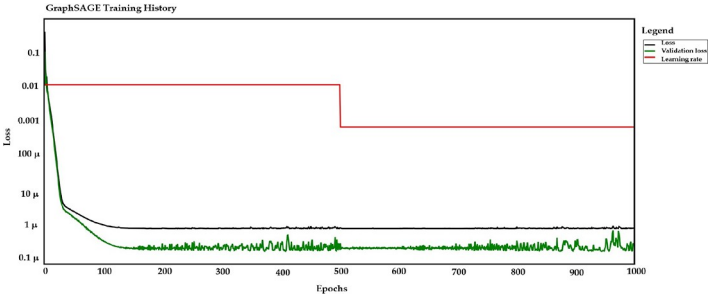

Figure A2. Fitting curves of the GraphSAGE

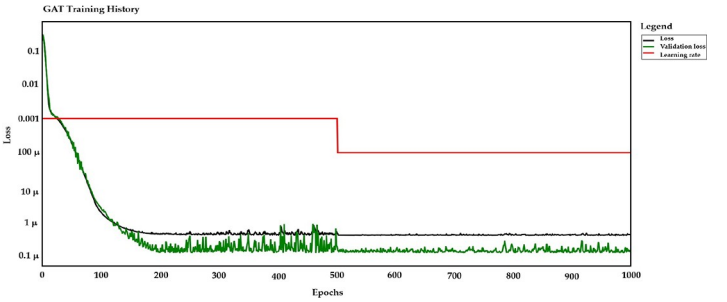

Figure A3. Fitting curves of the GAT