COMPARATIVE STUDY OF TRANSFORMER- AND LSTM-BASED MACHINE LEARNING METHODS FOR TRANSIENT THERMAL FIELD RECONSTRUCTION

Wiera Bielajewa,[1] Michelle Tindall,[2] & Perumal Nithiarasu[1,*]

[1]*Zienkiewicz Institute for Modelling, Data and AI, Swansea University, Bay Campus, Swansea, SA1 8EN, UK*

[2]*Culham Science Centre, United Kingdom Atomic Energy Authority (UKAEA), Abingdon, OX14 3DB, UK*

*Address all correspondence to: Perumal Nithiarasu, Zienkiewicz Institute for Modelling, Data and AI, Swansea University, Bay Campus, Swansea, SA1 8EN, UK, E-mail: p.nithiarasu@swansea.ac.uk

Original Manuscript Submitted: 08/03/2023; Final Draft Received: 11/13/2023

Solution reconstruction from limited number of measurements is useful in many areas of heat transfer applications. Unlike the standard problems, such reconstruction problems are ill-posed; thus, the non-uniqueness of solution and inherent instability severely complicates the modelling process. Consequently, more conventional inverse analysis methods to reconstruct solutions remain computationally intractable and lacking sufficient flexibility, especially when dealing with time-dependent problems. Aided by powerful Graphical Processing Units (GPUs), Machine Learning (ML) methods rose in popularity due to their flexibility and ability to efficiently process large amounts of data. In recent years, the Transformer-based ML models have gained recognition for their remarkable performance in Natural Language Processing (NLP) tasks as well as time-series analysis, overshadowing the performance of the ML models conventionally used for sequence processing, such as the long short-term memory (LSTM) models. These achievements make Transformer-based models seemingly ideal candidates for reconstructing full solutions from a few measurements. This article compares the performance of these novel Transformer-based models with a simple LSTM model in reconstructing transient one-dimensional (1D) and two-dimensional (2D) thermal fields using sparse spatial measurements. Counterintuitively, the simple LSTM model achieves higher or comparable prediction accuracy compared to the complex Transformer-based models while also exhibiting shorter or comparable training times, which may render Transformer-based models a suboptimal choice for reconstructing transient solutions. Instead, more traditional sequence processing ML models, such as LSTM, might be preferred for this purpose.

KEY WORDS: *Machine Learning, Transformer, Transient problem, Solution reconstruction, Conduction, Computational heat transfer, Sparse measurements*

Nomenclature

α	Thermal diffusivity
α_x	Thermal diffusivity in x direction
α_y	Thermal diffusivity in y direction
$[\boldsymbol{K}]$	Key matrix for self-attention operation
$[\boldsymbol{Q}]$	Query matrix for self-attention operation
$[\boldsymbol{U_{pred}}]$	Predicted temperature matrix

$[\boldsymbol{U_{true}}]$ True temperature matrix

$[\boldsymbol{V}]$ Value matrix for self-attention operation

$[\boldsymbol{W}]'$ Scaled weight matrix for self-attention operation

$[\boldsymbol{W}]$ Normalised weight matrix for self-attention operation

$[\boldsymbol{W_k}]$ Key weight matrix for self-attention operation

$[\boldsymbol{W_q}]$ Query weight matrix for self-attention operation

$[\boldsymbol{W_v}]$ Value weight matrix for self-attention operation

$[\boldsymbol{X}]$ Input matrix for self-attention operation

$[\boldsymbol{Y}]$ Output matrix for self-attention operation

$\{\boldsymbol{k_i}\}$ Key vector for self-attention operation

$\{\boldsymbol{q_i}\}$ Query vector for self-attention operation

$\{\boldsymbol{v_i}\}$ Value vector for self-attention operation

$\{\boldsymbol{x_i}\}$ Input vector for self-attention operation

$\{\boldsymbol{y_i}\}$ Output vector for self-attention operation

b Bias of the RNN cell

b_f, b_i, $b_{\tilde{c}}$, b_0 Biases of the modified LSTM cell

b_i, $b_{\tilde{c}}$, b_o Biases of the original LSTM cell

c_t Original LSTM cell state at time t

d_{model} Model dimension

f_t Forget gate value of the modified LSTM cell at time t

h Number of heads in multi-head (self-)attention

h_t Hidden state (or recurrent information) of the RNN, original and modified LSTM cells at time t

it Iteration number

l Prediction window size (sequence length)

lr Learning rate

N_{out} Number of output channels

N_{total} Total number of temperature data points

$PE_{i,n}$ Prediction error at node n at time step i

t Time

u Temperature

$u_{pred\_i,n}$ Predicted temperature at node n at time step i

u_{pred_k} k^{th} temperature value predicted by the ML model

u_{true_mean} True temperature mean

$u_{true\_i,n}$ True temperature at node n at time step i

u_{true_k} k^{th} true temperature value out of N_{total} temperature data points

W_h, W_x Weights of the RNN cell

$W_{fh}, W_{fx}, W_{ih}, W_{ix}, W_{\tilde{c}h}, W_{\tilde{c}x}, W_{oh}, W_{ox}$ Weights of the modified LSTM cell

$W_{ih}, W_{ix}, W_{\tilde{c}h}, W_{\tilde{c}x}, W_{oh}, W_{ox}$ Weights of the original LSTM cell

x Spatial coordinate

x_t Input of the RNN, original and modified LSTM cells at time t

y Spatial coordinate

y_t Output of the RNN cell at time t

1. INTRODUCTION

Transient inverse analysis is a research area in computational engineering which addresses solving various time-dependent inverse problems, including solution reconstruction from limited number of measurements. An inverse problem significantly differs from a standard forward problem. Generally, transient forward problems are well-posed; therefore, given the appropriate initial and boundary conditions the numerical solution can be calculated with a defined accuracy (Tarantola, 2004). Contrariwise, one type of transient inverse problem involves reconstructing the full data inside a problem domain using the available sparse data (observations or measurements). Unlike the standard forward problem, the inverse problem is ill-posed (Tarantola, 2004); consequently, the non-uniqueness of the solution and inherent instability severely complicates the modelling process oftentimes making it computationally intractable, especially for challenging problems.

In order to combat the limitations of the more conventional approaches to reconstructing transient solutions, this paper explores the use of two types of Machine Learning (ML) models to aid the process of obtaining solutions within an acceptable margin of uncertainty. With the advent of powerful Graphical Processing Units (GPUs) ML became popular in many areas of science and engineering due to its flexibility and the ability to process vast amounts of data within a feasible timescale. In recent years, the Transformer-based ML models have attained recognition by achieving outstanding performance in various Natural Language Processing (NLP) tasks, as evidenced by the well-known ChatGPT chatbot (Qin et al., 2023), as well as numerous time-series analysis problems (Lim and Zohren, 2021; Wen et al., 2023; Wu et al., 2021; Zhou et al., 2021, 2022). These successes in the area of temporal sequence transformation make them seemingly ideally suited for transient inverse analysis. This paper compares the performance of the novel complex Transformer-based models with the performance of simple long short-term memory (LSTM) model (Hochreiter and Schmidhuber, 1997), which is the ML model type traditionally used for the sequential data processing (Yu et al., 2019), for the task of reconstructing the transient one-dimensional (1D) and two-dimensional (2D) thermal fields.

The aim of the current work is to showcase the suitability of the increasingly popular Transformer-based models for reconstructing transient solutions, a problem encountered in many industrial applications, against more conventional ML models, such as LSTM. The source codes used to produce the results can be found at `https://zenodo.org/doi/10.5281/zenodo.8208285`.

2. BACKGROUND

This section provides the background information and add context necessary for understanding the significance of the study presented in this paper.

2.1 Transient Inverse Problem

The general definition of a transient forward problem can be given as determining the time-dependent effects of the given causes using the applicable physical model of a system. In order to solve the transient forward problem and obtain a full solution inside a domain using standard methods, the system parameters, the boundary conditions, and the initial conditions of the system should be prescribed. Contrariwise, the transient inverse problems can be divided into two types:

1. Determining the system parameters from the observed causes and effects. This is a classic definition of an inverse problem (Tarantola, 2004).

2. Determining the causes from the observed time-dependent effects. In essence, it is a task of using the available sparse data inside a domain (observations or measurements) to reconstruct the full data solutions.

Various methods have been employed over the years to deal with the inverse problems; however, historically, more attention has been given to the inverse problems falling under the first type. The more traditional methods include functional analytic regularisation as well as the statistical regularisation, with the most well-known example being the Bayesian inversion (Arridge et al., 2019; Tarantola, 2004). A search-and-optimisation-based approach is another way to obtain solutions to transient problems. For instance, Bangian-Tabrizi and Jaluria (2018) solved the inverse 2D natural convection problem in steady state using a Particle Swarm Optimisation (PSO) algorithm (Zhang et al., 2015). Whereas Arridge et al. (2019) and Tamaddon-Jahromi et al. (2020) provide a more comprehensive review of the various methods used to solve inverse problems.

This paper focuses on the transient inverse problem of the second type, the reconstruction of transient thermal fields in particular. Perhaps the most common sources of the sparse data observed inside a domain are the data obtained from physical experiments. Fusion energy technology research facilities, designed to test components' suitability for the extreme environment inside a fusion rector, regularly encounter transient inverse problems stemming from the sparse experimental data. The HIVE (Heating by Induction to Verify Extremes) experimental facility (Hancock, 2018) is an illustrative example of that: inverse analysis has to be performed to reconstruct the full temperature field using the temperature measurements recorded by a few thermocouples. Figure 1 shows an example of the HIVE experimental setup.

2.2 Long Short-Term Memory (LSTM)

Recurrent neural networks (RNNs) are extensively used for various sequential data processing tasks (Yu et al., 2019). RRNs typically consist of a number of standard recurrent cells (Figure 2), the mathematical representation of which

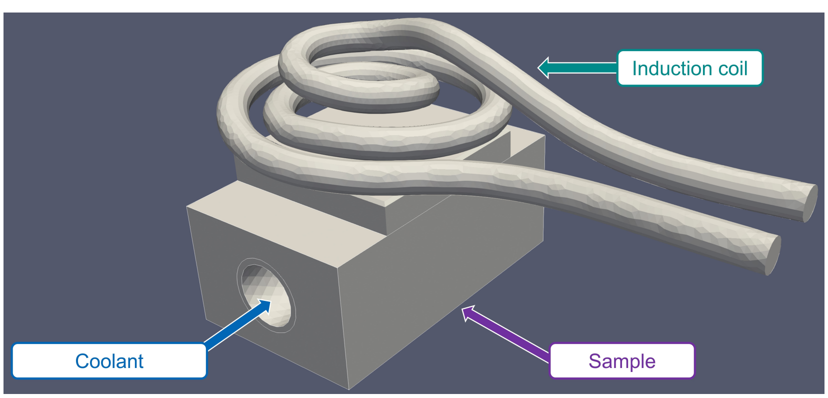

FIG. 1: The arrangement of induction coil and sample inside HIVE (Hancock, 2018).

can be written as:

$$
\begin{aligned}
h_t &= \sigma(W_h h_{t-1} + W_x x_t + b) \\
y_t &= h_t
\end{aligned}
\tag{1}
$$

where x_t, y_t, and h_t represent the input, output, and the hidden state (or recurrent information) of the cell at time t respectively, h_{t-1} is the hidden state at time $t-1$, while W_h and W_x are the weights of the cell, and b is the bias. The operator σ is a sigmoid function.

However, a RNN comprised of the standard recurrent cells tend to experience difficulties during the training process due to the vanishing or exploding gradients between inputs that are far apart in time (Bengio et al., 1994). In order to address this problem of long-term dependencies, the long short-term memory (LSTM) cell, a type of RNN, was developed more than two decades ago (Hochreiter and Schmidhuber, 1997) and successfully applied to a wide range of sequential tasks, such as speech recognition (Hsu et al., 2016), trajectory prediction (Altché and de La Fortelle, 2017), and prediction of the remaining useful life (Ren et al., 2021), to name a few. The original LSTM cell contains only input and output gates (Figure 3) and is represented by the following expressions:

$$
\begin{aligned}
i_t &= \sigma(W_{ih} h_{t-1} + W_{ix} x_t + b_i) \\
\tilde{c}_t &= \tanh(W_{\tilde{c}h} h_{t-1} + W_{\tilde{c}x} x_t + b_{\tilde{c}}) \\
c_t &= c_{t-1} + i_t \odot \tilde{c}_t \\
o_t &= \sigma(W_{oh} h_{t-1} + W_{ox} x_t + b_0) \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
\tag{2}
$$

where c_t and c_{t-1} represent a LSTM cell state at times t and $t-1$ respectively, while W_{ih}, W_{ix}, $W_{\tilde{c}h}$, $W_{\tilde{c}x}$, W_{oh}, W_{ox} are the weights, b_i, $b_{\tilde{c}}$, b_o are the biases. The operator $\odot$ is the Hadamard product (also known as the element-wise product). The input gate determines what information should be stored in the cell state, whereas the output gate chooses what information should be extracted from the cell state for the output.

In the present work, a LSTM network comprising a modified version of the original LSTM cells is used. The modification incorporates a forget gate, which decides what information should be eliminated from the cell state (Gers

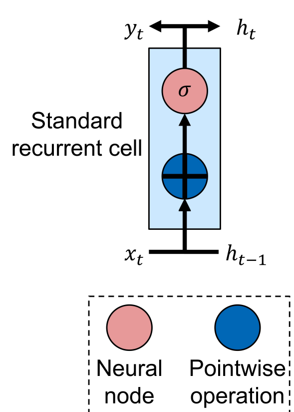

FIG. 2: Standard recurrent cell for RNNs.

et al., 2000). Figure 4 shows the modified LSTM cell, and the following equations represent the cell:

$$
\begin{aligned}
f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \\
i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \\
\tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_0) \\
h_t &= o_t \odot \tanh(c_t) \qquad (3)
\end{aligned}
$$

where f_t represent the value of the forget gate at a time t.

It should be noted that in this sub-section t is the t^{th} time step in a sequence as it is the commonly used notation for RNNs; however, t means the total sequence length in the subsequent sub-sections.

2.3 Transformers

The classic Transformer model was created by the Google research team in 2017 and successfully applied to the Natural Language Processing (NLP), such as natural language generation and machine translation (Vaswani et al., 2017). Nowadays, Transformer is considered to be the best model for the various NLP tasks (Wolf et al., 2020), with Chat Generative Pre-trained Transformer (ChatGPT) (Qin et al., 2023), an Artificial Intelligence (AI) chatbot, being probably the most famous example of the impressive results Transformers are capable of achieving. However, NLP is essentially a sequence-to-sequence transformation task, since the model input and output are almost always an ordered series of elements. This fact makes the Transformer an appropriate model for time-series prediction. Indeed, in recent years a number of Transformer-based models were successfully applied to various time-series tasks, such as weather, electricity consumption, and exchange rate prediction. Wen et al. (2023) reviewed the state-of-the-art Transformer-based models used to analyse the time series; whereas, Lim and Zohren (2021) conducted a survey of Deep Learning (DL) methods used for time-series forecasting. The aforementioned surveys might not be completely exhaustive; as Transformers are gaining popularity, the new Transformer-based models and the numerous variations of the already existing ones are created every year. Consequently, it is quite challenging to keep track of

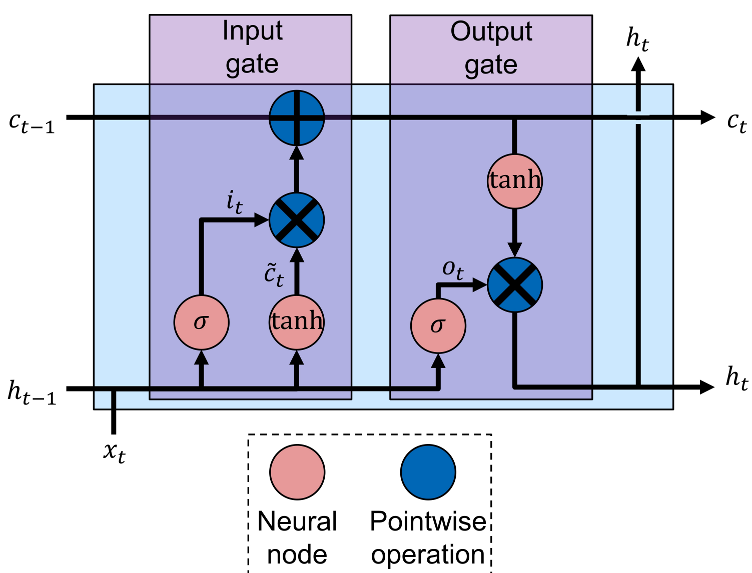

FIG. 3: Original LSTM cell (Hochreiter and Schmidhuber, 1997).

all new developments, particularly due to the fact that these advances are oftentimes made in widely different areas of research, such as weather prediction (Wu et al., 2022) and computer vision (Ivanovic and Pavone, 2019). Meaning that the process of comparing the new models between each other is complicated by the fact that they have been tested on the research-area-specific datasets, therefore it is not immediately obvious which model is better suited for engineering applications.

The primary advantage of the Transformer-based models lies in the absence of any recurrent connections, which are present in RNNs (Hochreiter and Schmidhuber, 1997). The elimination of the recurrent connections should significantly reduce the training times and make the model more parallelisable, which is beneficial for the model training on GPUs. Furthermore, they were found to be excellent at detecting the long- and- short-term sequence dependencies, which should make the model more accurate (Vaswani et al., 2017).

Figure 5 shows a simplified transformer block. This paper will not provide a detailed explanation of how Transformers work. Readers are referred to Bloem (2019) and Vaswani et al. (2017) for a more detailed introduction to self-attention and the classic Transformer. Nevertheless, a brief overview of the classic self-attention is provided in the next sub-section.

2.4 Self-Attention

At the core of any Transformer-based model is a self-attention operation or a variation thereof. Figure 6 shows the classic self-attention operation used in the classic Transformer (Vaswani et al., 2017). The input is t vectors of size K, while the generated output is a different set of t vectors of the same size. The input vectors are used to calculate queries, keys, and values using query, key, and value weight matrices:

$$\begin{aligned}\{\boldsymbol{q_i}\} &= [\boldsymbol{W_q}]\{\boldsymbol{x_i}\} \\ \{\boldsymbol{k_i}\} &= [\boldsymbol{W_k}]\{\boldsymbol{x_i}\} \\ \{\boldsymbol{v_i}\} &= [\boldsymbol{W_v}]\{\boldsymbol{x_i}\}\end{aligned} \tag{4}$$

where $\{\boldsymbol{q_i}\}$, $\{\boldsymbol{k_i}\}$, $\{\boldsymbol{v_i}\}$, and $\{\boldsymbol{x_i}\}$ are i^{th} query, key, value, and input vectors, respectively; $[\boldsymbol{W_q}]$, $[\boldsymbol{W_k}]$, and $[\boldsymbol{W_v}]$ are the query, key, and value weight matrices, respectively. All query, key, and value vectors can be concatenated to obtain query, key, and value matrices, which are $[\boldsymbol{Q}]$, $[\boldsymbol{K}]$, and $[\boldsymbol{V}]$, respectively. The scaled weight matrix $[\boldsymbol{W}]'$ is

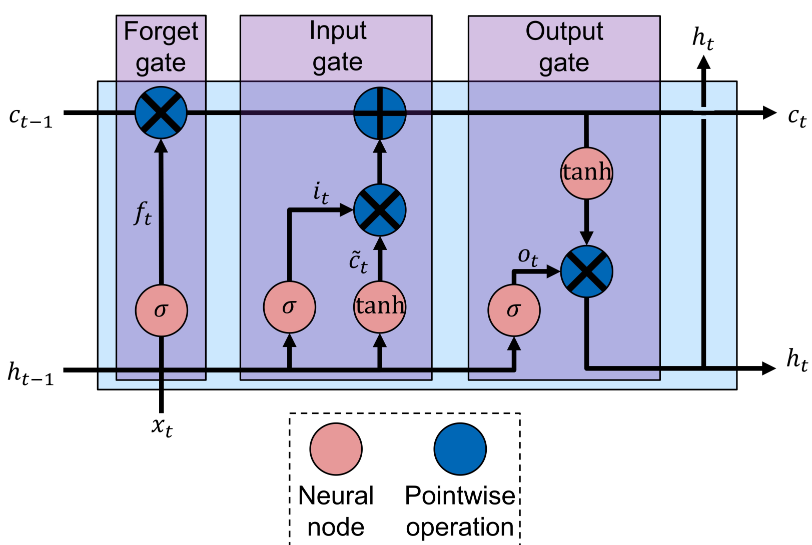

FIG. 4: Modified LSTM cell with a forget gate.

calculated using the following equation:

$$[\boldsymbol{W}]' = \frac{[\boldsymbol{Q}][\boldsymbol{K}]^T}{\sqrt{K}} \tag{5}$$

The scaled weight matrix is normalised using a softmax function (Goodfellow et al., 2016):

$$[\boldsymbol{W}] = \mathrm{softmax}\big([\boldsymbol{W}]'\big) \tag{6}$$

Finally, the normalised weight matrix $[\boldsymbol{W}]$ is then multiplied by the values vector to finally obtain the output matrix $[\boldsymbol{Y}]$, consisting of output vectors $\{\boldsymbol{y_i}\}$:

$$[\boldsymbol{Y}] = [\boldsymbol{W}][\boldsymbol{V}] \tag{7}$$

The query, key, and value weight matrices are trainable parameters. The Transformer-based models tend to employ several self-attention operations in parallel (Figure 7), which allows for a more efficient extraction of the various features in the given time series. The self-attention operation comprises a number of matrix multiplications, which is beneficial as they can be performed using a highly optimised and efficient matrix multiplication code.

3. METHODOLOGY

3.1 Selected Models

In this paper LSTM model and four Transformer-based models are applied to the 1D and 2D heat conduction problems. Table 1 provides a summary of these models; the classic Transformer will hereafter be referred to as the Transformer. The self-attention operation used in the Transformer (Vaswani et al., 2017) is described in Subsection 2.4. Informer (Zhou et al., 2021), Autoformer (Wu et al., 2021), and FEDformer (Zhou et al., 2022) were developed with an aim of increasing the Transformer's efficiency by reducing its complexity and adapting the architecture specifically to time-series processing (Table 1).

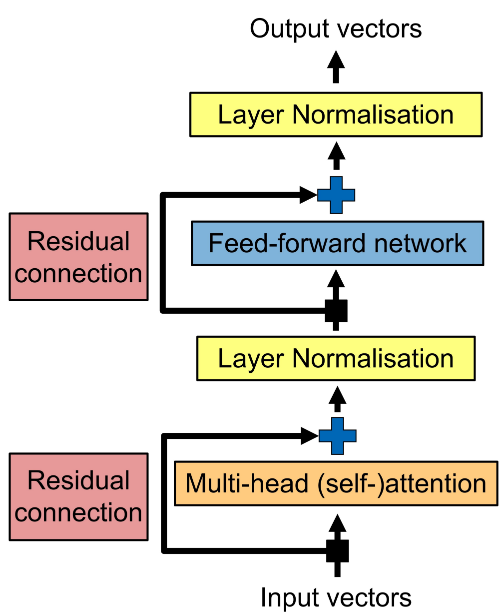

FIG. 5: Simplified transformer block, which consists of multi-head (self-)attention, layer normalisation, feed-forward network, and another layer normalisation. The reader is referred to Bloem (2019) for a more detailed introduction to the classic Transformer.

3.2 Model Structure

Table 2 presents the model hyperparameters used for the Transformer-based models in this paper. Three layers, two encoder layers and one decoder layer, are used; three values of the input and output time-series length (sequence length hyperparamter) are considered: 25, 50, and 100. For greater clarity, the sequence length l will be referred to as the prediction window size, with the prediction window being defined as the time interval for which a prediction is

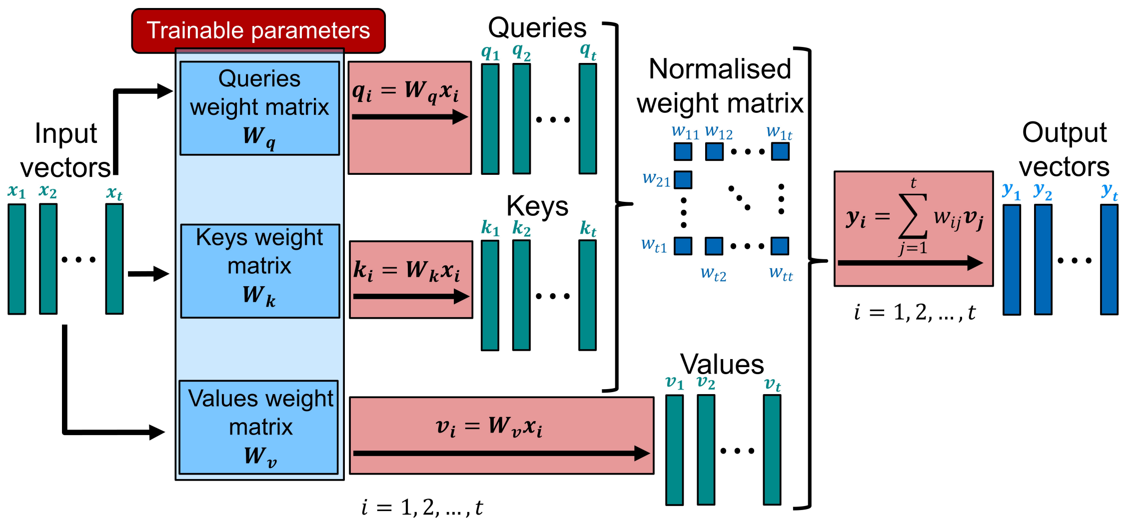

FIG. 6: Classic self-attention used in the classic Transformer (Vaswani et al., 2017). The input is t vectors of size K, while the generated output is a different set of t vectors of the same size. The input vectors together with the query, key, and value weight matrices are used to calculate queries, keys, and values; a weight matrix is produced by multiplying queries and keys, and it is subsequently scaled and normalised.

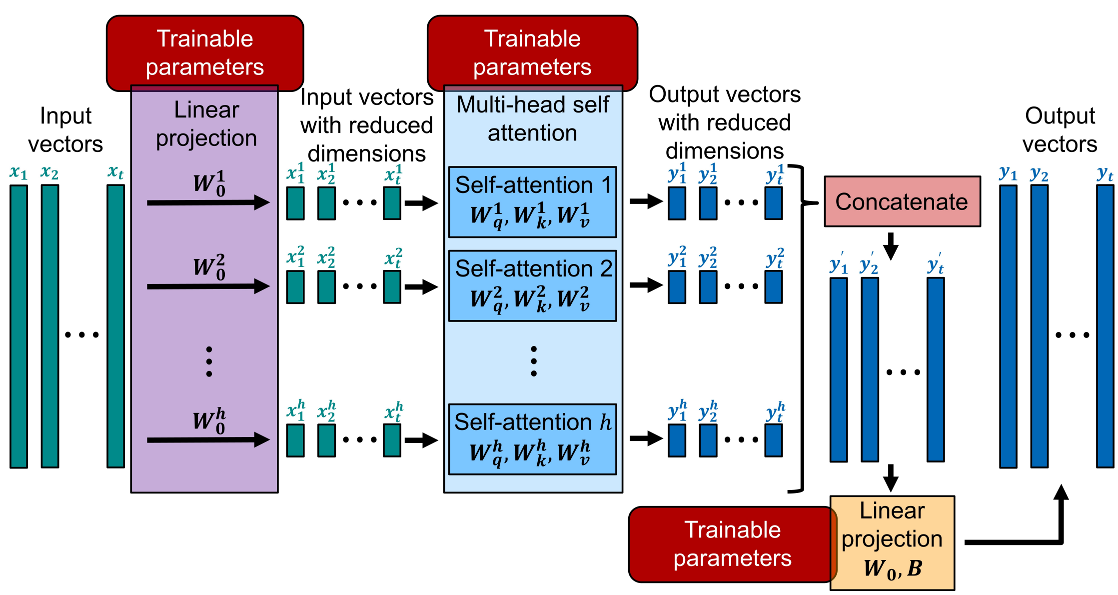

FIG. 7: Classic multi-head (self-)attention operation. Several self-attention operations are employed in parallel, which allows for a more efficient extraction of the various features in the given time series. The output vectors with reduced dimensions are concatenated and then the linear projection is applied to them to obtain the output vectors.

made by the model. The hyperparameters No. 4-9 are assigned the values used in literature (Vaswani et al., 2017; Wu et al., 2021; Zhou et al., 2021, 2022).

Table 3 presents the model hyperparameters used for the LSTM model in this paper. Only one layer is used; the model dimension is fixed at 512 to match the Transformer-based models. Three values of the prediction window size l are considered: 25, 50, and 100. Furthermore, the feed-forward layer is added after one LSTM layer in order to reshape an output and directly predict the temperature over multiple time steps in one inference step.

3.3 Training

All Transformer-based models are trained using an Adam optimiser (Kingma and Ba, 2017). Additionally, the warm-up stage is applied to the learning rate as it was shown to improve the training process of the Transformer-based architectures (Xiong et al., 2020). The learning rate during the warm-up stage is given by:

$$lr(it) = \frac{it}{T_{warmup}} lr_{max} \quad \text{for} \quad it \leq T_{warmup} \tag{8}$$

where lr is the learning rate, and it is an iteration number. While the learning rate after the warm-up stage is given by:

$$lr(it) = \frac{lr(it-1)\sqrt{T_{warmup}}}{\sqrt{it}} \quad \text{for} \quad it > T_{warmup} \tag{9}$$

TABLE 1: The summary of the models considered in this paper.

Model type	LSTM	Transformer	Informer	Autoformer	FEDformer
Original purpose	Sequential data	Linguistic data	Temporal data	Temporal data	Temporal data
Self-attention type	N/A	Classic self-attention	Sparse self-attention	Auto-Correlation	Discrete Fourier Transform (DFT)

TABLE 2: Hyperparameters selected for all Transformer-based models.

No.	Hyperparameter	Value(s)
1.	Encoder layers	2
2.	Decoder layers	1
3.	Prediction window size (sequence length) l	25, 50, and 100
4.	Model dimension d_{model}	512
5.	Multi-head (self-)attention heads h	8
6.	Feed-forward network dimension	2048
7.	Dropout rate	0.05
8.	Activation function	GELU
9.	Attention factor	3

TABLE 3: Hyperparameters selected for LSTM models.

No.	Hyperparameter	Value(s)
1.	LSTM layers	1
2.	Prediction window size (sequence length) l	25, 50, and 100
3.	Model dimension d_{model}	512
4.	Dropout rate	0.05

Transformer-based models are trained using lr_{max} and T_{warmup} values provided in Table 4 for 500 epochs with the batch size of 32; then the best option is selected for each model type listed in Table 1 using Normalised Root Mean Square Error (NRMSE) as a metric:

$$NRMSE = \frac{RMSE}{u_{true\_mean}} \quad \text{and} \quad RMSE = \sqrt{\frac{\sum_{k=1}^{N_{total}} (u_{true\_k} - u_{pred\_k})^2}{N_{total}}} \tag{10}$$

where N_{total} is a total number of temperature data points, u_{true_mean} is the true temperature mean, u_{true_k} is the k^{th} true temperature value out of N_{total} temperature data points, while u_{pred_k} is the k^{th} temperature value predicted by the ML model. Further details are provided in Section 4.

The LSTM model is trained using just an Adam optimiser (Kingma and Ba, 2017) for 500 epochs. Finally, NVIDIA A100 40GB GPU is used for training of all ML models considered in this paper. Figure 8 shows the convergence during the training, with the best options shown for Transformer, Informer, Autoformer, and FEDformer.

All models are initialised with a fixed random seed to ensure the repeatability of the results. This approach guarantees the consistency of the initial weights and any stochastic processes within the training algorithm across different runs. Thus, the potential variability in performance due to random initialization is mitigated allowing for a balanced comparison of the model architectures.

TABLE 4: Learning rates used to train the models for 500 epochs; the best option is selected for each model type listed in Table 1 using Normalised Root Mean Square Error (NRMSE) as a metric (Eq. 10).

Option No.	**Constant lr or with warm-up**	lr_{max}	T_{warmup}
1.	Constant $lr = 1e^{-4}$	N/A	N/A
2.	With warm-up	$1e^{-3}$	4000
3.	With warm-up	$1e^{-3}$	2000
4.	With warm-up	$1e^{-3}$	500
5.	With warm-up	$5e^{-4}$	4000
6.	With warm-up	$5e^{-4}$	2000
7.	With warm-up	$5e^{-4}$	500

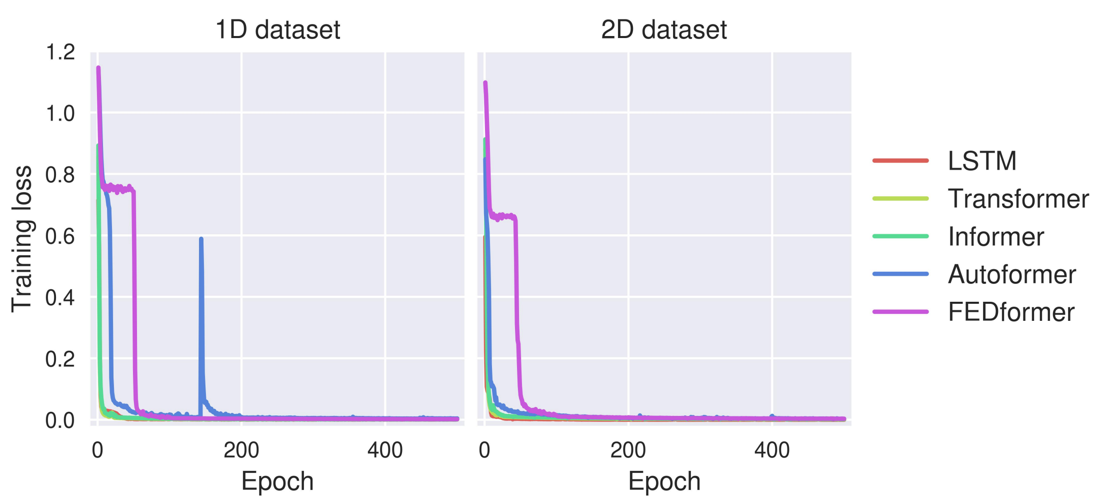

FIG. 8: Model convergence during the training process for 1D and 2D transient heat conduction cases discussed in Section 4.

4. RESULTS AND DISCUSSION

4.1 One-Dimensional Transient Heat Conduction

The linear 1D transient heat conduction equation is given by the following expression:

$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial^2 u(x,t)}{\partial x^2} \tag{11}$$

where u is temperature, α is thermal diffusivity, t is time, and x is a space coordinate.

For this case α is set to be equal to $8.6e^{-4} m^2/s$, while the boundary and initial conditions are given by (Figure 9):

$$\text{Boundary Conditions: } u(x = x_A, t) = 255.372K \quad \text{and} \quad \frac{\partial u(x = x_B, t)}{\partial x} = 0$$
$$\text{Initial Conditions: } u(x, t = 0) = 272.039K \tag{12}$$

The Ground Truth (GT) solution is obtained using the Finite Difference Method (FDM) implemented in the PyPDE Python package (Zwicker, 2020). Figure 9 shows the grid used to generate the GT; Figure 10 shows the GT used for model testing. The simulation is run for $1000s$ and the temperature is recorded every second: the first $700s$ of the obtained data is used for ML training, the next $100s$ for ML validation, and finally the last $200s$ is used for ML testing. The temperature values at the six input channels (Figure 9) are given to the ML model, whereas the temperature values at the 194 output channels are generated by the ML model (Figure 11). The locations of the six input channels are randomly selected.

For the training, validation, and testing the prediction window is moved by one time step forward, which is equal to one second in this case, to generate one input-output sample. For example, assuming that the prediction window size l is equal to 50, the first prediction window spans from $1s$ to $50s$ of $200s$ used for ML testing, the second one spans from $2s$ to $51s$, the third from $3s$ to $52s$ etc. Consequently, for testing when $l = 50$ there are $200 - 50 + 1 = 151$ prediction windows for which the predicted temperature matrix $[\boldsymbol{U_{pred}}]$ and true temperature matrix $[\boldsymbol{U_{true}}]$ are generated. $[\boldsymbol{U_{pred}}]$ and $[\boldsymbol{U_{true}}]$ are matrices of dimension $((50 \cdot 151) \times N_{out})$, where N_{out} is the number of output channels, which is equal to 194 for this case. Therefore, for Eq. 10 N_{total} can be calculated as:

$$N_{total} = (50 \cdot 151) \cdot N_{out} = (50 \cdot 151) \cdot 194 \tag{13}$$

For other values of the prediction window size l and N_{out} the calculations are performed in the same manner.

Table 5 provides testing NRMSEs calculated using Eq. 10 with Eq. 13, as well as the training times. In order to visualise the error distribution in space and time, four consecutive prediction windows (out of 151 prediction windows) are selected for the prediction window size $l = 50$; these are four prediction windows spanning from $1s$ to $50s$ of $200s$ used for ML testing, from $51s$ to $100s$, from $101s$ to $150s$, and finally from $151s$ to $200s$. Figure 12 (top row) shows the prediction error distribution defined using Eq. 14, whereas the bottom row of this Figure shows the prediction errors averaged at each time step using Eq. 15.

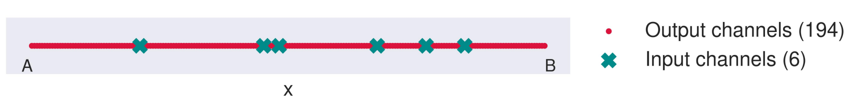

FIG. 9: The 1D grid used to generate the Ground Truth (GT) for 1D transient heat conduction. The input and output channels of the mesh nodes are highlighted, with the six green crosses being the input channels and the 194 red dots being the output channels. The temperature values at the six input channels are given to the ML model, whereas the temperature values at the 194 output channels are generated by the ML model at every time step, which is equal to 1s in this case. The locations of the six input channels are randomly selected.

$$PE_{i,n} = \frac{|u_{pred\_i,n} - u_{true\_i,n}|}{u_{true\_i,n}} \tag{14}$$

$$PE_i = \frac{\sum_{n=1}^{N_{out}} PE_{i,n}}{N_{out}} \tag{15}$$

where $PE_{i,n}$ is a prediction error at node n at time step i, PE_i is a prediction error averaged at time step i; $u_{pred\_i,n}$ and $u_{true\_i,n}$ are the predicted and true temperatures, respectively, at node n at time step i.

Table 5 shows that for all three values of l LSTM model achieves the lowest training times, whereas FEDformer model consistently has the highest training times. For $l = 25$ and $l = 50$ LSTM model attains the lowest NRMSE; and, indeed, this correlates with the prediction error distribution on Figure 12. For $l = 100$ Transformer displayed the lowest NRMSE; however, the NRMSE of LSTM is only 0.004% higher. Finally, Autoformer model obtained the NRMSE more than two times higher than LSTM model.

4.2 Two-Dimensional Transient Heat Conduction

The linear 2D transient heat conduction equation is given by the following expression:

$$\frac{\partial u(x,y,t)}{\partial t} = \alpha_x \frac{\partial^2 u(x,y,t)}{\partial x^2} + \alpha_y \frac{\partial^2 u(x,y,t)}{\partial y^2} \tag{16}$$

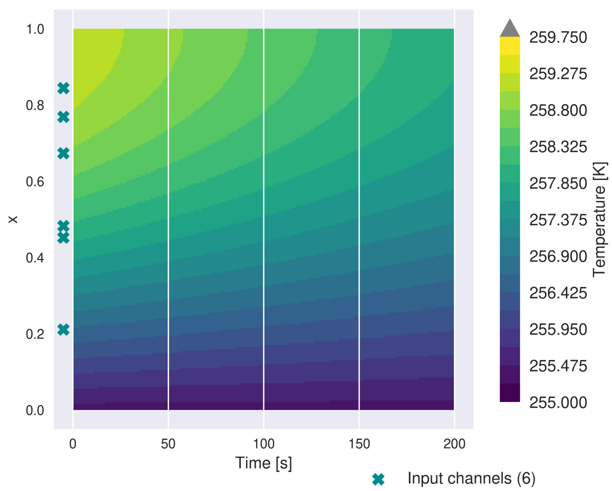

FIG. 10: The Ground Truth (GT) for 1D transient heat conduction problem generated using the Finite Difference Method (FDM) implemented in the PyPDE Python package (Zwicker, 2020). This region is used for the model testing. The temperature values at the six input channels (green crosses) are given to the ML model, whereas the temperature values at the 194 output channels (not shown on this figure, please refer to Figure 9) are generated by the ML model. The locations of the six input channels (green crosses) are randomly selected.

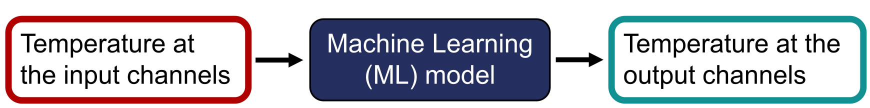

FIG. 11: Outline of the Machine Learning (ML) model used for the transient thermal field reconstruction. The temperature values at the six input channels are given to the ML model, whereas the temperature values at the 194 output channels are generated by the ML model. The locations of the six input channels are randomly selected.

where x and y are the spatial coordinates, and α_x and α_y are the thermal diffusivities in x and y directions.

For this case α_x is set to be equal to $13.9e^{-4}m^2/s$, while α_y is set to be equal to $3.3e^{-4}m^2/s$. The boundary and initial conditions are given by Eq. 17, and Figure 13 shows the location of the domain boundaries.

$$\begin{aligned}&\text{Boundary Conditions:}\\ &1.\quad u(x,y,t) = 255.372K \quad \text{for} \quad x,y \in [AB] \cup [BC]\\ &2.\quad \nabla u(x,y,t) = 0 \quad \text{for} \quad x,y \in [CD] \cup [DA]\\ &\text{Initial Conditions: } u(x,y,t=0) = 272.039K\end{aligned} \tag{17}$$

TABLE 5: Model testing errors and training times for the 1D heat conduction calculated using Eq. 10. The best results are highlighted in **green bold** and the worst results are highlighted with a red underline.

Model type	LSTM	Transformer	Informer	Autoformer	FEDformer
Prediction window size $l = 25$					
NRMSE (Eq. 10) [%]	**0.188**	0.189	0.192	0.497	0.362
Training time [min]	**35.9**	39.1	40.2	40.9	46.4
Prediction window size $l = 50$					
NRMSE (Eq. 10) [%]	**0.189**	0.190	0.190	0.493	0.365
Training time [min]	**36.0**	38.5	39.8	42.4	52.3
Prediction window size $l = 100$					
NRMSE (Eq. 10) [%]	0.196	**0.192**	0.195	0.468	0.371
Training time [min]	**36.3**	38.5	40.9	43.8	65.2

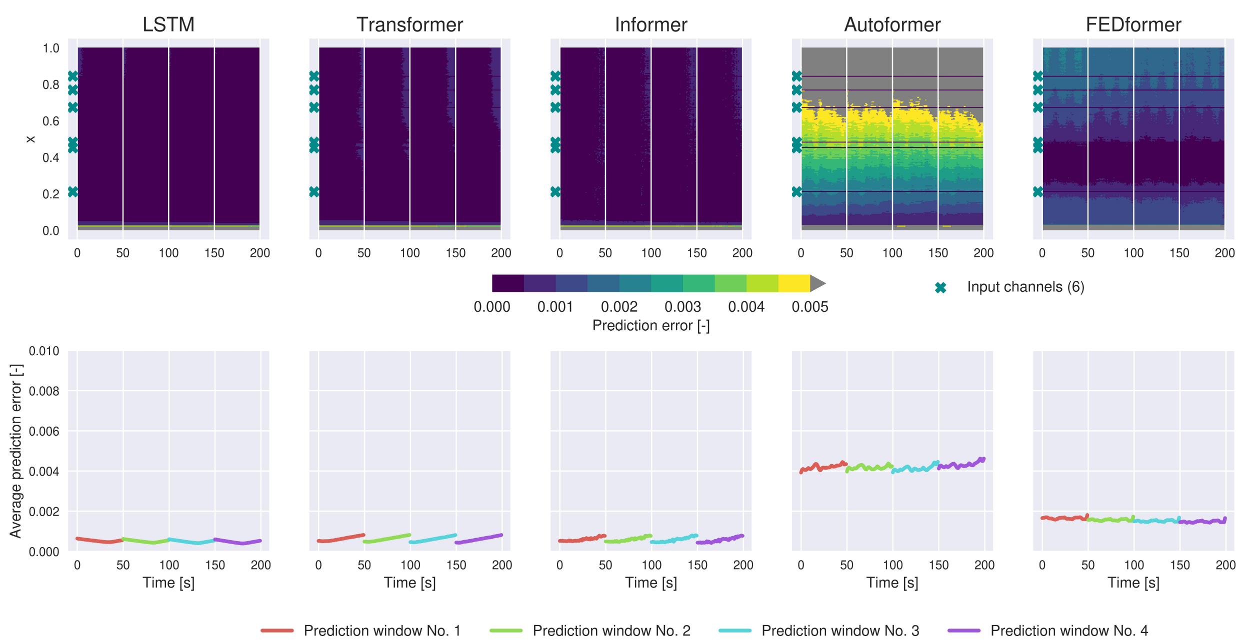

FIG. 12: Prediction error distribution calculated using Eq. 14 (top row) and prediction errors averaged at each time step calculated using Eq. 15 (bottom row) for the 1D heat conduction for five models with the prediction window size $l = 50$. For this error visualisation four prediction windows located consecutively to one another are selected. The six green crosses are the input channels.

The Ground Truth (GT) solution is obtained using the Finite Element Method (FEM) implemented in Code_Aster open-source software (Électricité de France (EDF), 1989–2023). Figure 13 shows the mesh used to generate the GT; Figure 14 shows the GT used for model testing. The simulation is run for 1000s and the temperature is recorded every second: the first 700s of the obtained data is used for ML training, the next 100s for ML validation, and finally the last 200s is used for ML testing. The temperature values at the twelve input channels are given to the ML model, whereas the temperature values at the 1142 output channels are generated by the ML model (Figures 11 and 13). The locations of the twelve input channels are randomly selected. Similar to 1D case (Sub-section 4.1), for the training, validation, and testing the prediction window is moved by one time step forward, which is one second in this case, to generate one input-output sample; the procedures for calculating the total number of prediction windows and NRMSEs are exactly the same as in the 1D case.

Table 6 provides testing NRMSEs calculated using Eq. 10 with Eq. 13, as well as the training times. The prediction error distributions for the prediction window size $l = 50$ are visualised in the similar way to 1D case (Sub-section 4.1). Figure 15 (top row) shows the time-averaged prediction error distribution defined using Eq. 18, whereas the bottom row of this figure shows the prediction errors averaged at each time step using Eq. 15 (please note that the different scales are used for these charts in comparison with Figure 12).

$$PE_n = \frac{\sum_{i=1}^{200} PE_{i,n}}{200} \tag{18}$$

where PE_n is a prediction error averaged at node n. Figure 16 shows the prediction error distribution variation with time; the four selected time instances correspond to the middle of each prediction window in Figure 15. This Figure highlights the areas where the prediction errors tend to increase with time.

Table 6 shows that for all three values of l FEDformer displayed the lowest NRMSEs and the highest training times. However, the NRMSEs achieved by LSTM model is only 0.1-0.2% higher than FEDformer, whilst LSTM model's training is 20-41% faster than FEDformer's. Moreover, one-layer LSTM architecture is significantly less complex than FEDformer structure, meaning that it is easier to troubleshoot it.

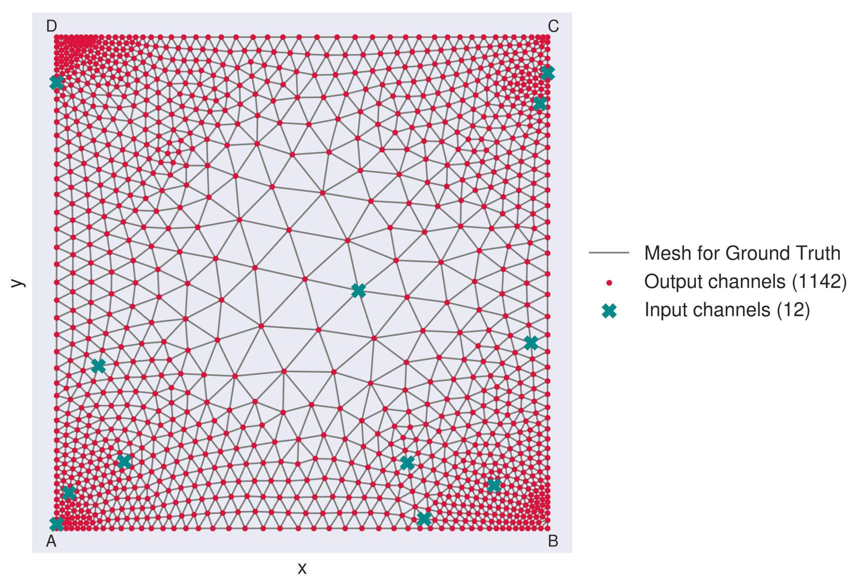

FIG. 13: The 2D mesh used to generate the Ground Truth (GT) for 2D transient heat conduction. The input and output channels of the mesh nodes are highlighted, with the twelve green crosses being the input channels and the 1142 red dots being the output channels. The temperature values at the twelve input channels are given to the ML model, whereas the temperature values at the 1142 output channels are generated by the ML model. The locations of the twelve input channels are randomly selected.

Surprisingly, the overall prediction error distribution patterns display similar features for all models (Figures 15 and 16). For the Transformer-based models this can be potentially explained by the fact that all these models are structured around self-attention operation of some description. However, the fact that the simple LSTM model, the structure of which is unrelated to the self-attention operation, demonstrates almost identical error patterns is rather counterintuitive and may potentially challenge the confidence in Transformer-based models.

5. CONCLUSIONS

In conclusion, the popular Transformer-based ML models are compared with the simple one-layer LSTM models for transient thermal field reconstruction problems. Generally, there are three main reasons why it might be advantageous to use the Transformer-based models over RRNs, such as LSTM, for sequence processing (Vaswani et al., 2017):

1. Self-attention operation is more effective at capturing the long-range dependencies in a sequence, thus the Transformer-based models should be more accurate.
2. They are more parallelisable as there are no recurrent connections; consequently, the training time should be shorter.
3. Due to the attention maps (Appendix A), they are more interpretable, and thus suffer less from the "black

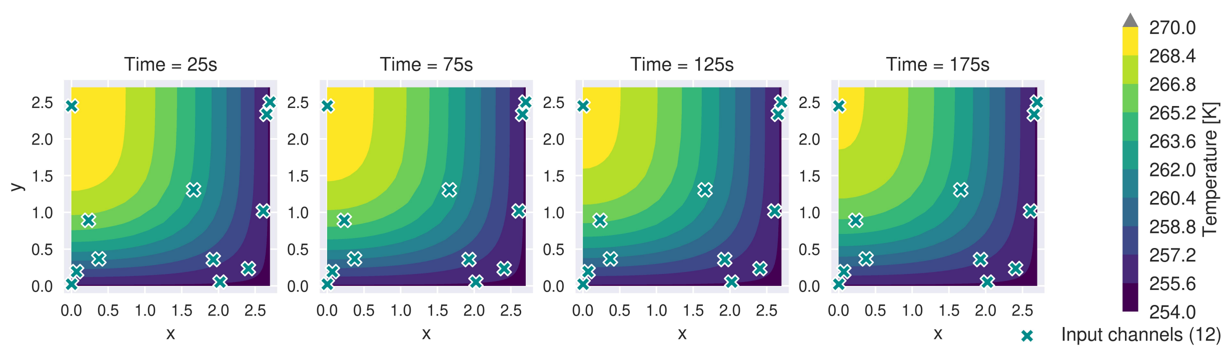

FIG. 14: The Ground Truth (GT) for 2D transient heat conduction problem generated using the Finite Element Method (FEM) implemented in Code_Aster open-source software (Électricité de France (EDF), 1989–2023). It is used for the model testing. The temperature values at the twelve input channels (green crosses) are given to the ML model, whereas the temperature values at the 1142 output channels (not shown on this figure, please refer to Figure 13) are generated by the ML model. The locations of the twelve input channels (green crosses) are randomly selected.

TABLE 6: Model testing errors and training times for the 2D heat conduction calculated using Eq. 10. The best results are highlighted in **green bold** and the worst results are highlighted with a red underline.

Model type	LSTM	Transformer	Informer	Autoformer	FEDformer
Prediction window size $l = 25$					
NRMSE (Eq. 10) [%]	2.197	2.208	2.218	2.543	**2.015**
Training time [min]	**37.9**	38.3	40.9	42.7	47.5
Prediction window size $l = 50$					
NRMSE (Eq. 10) [%]	2.170	2.191	2.193	2.415	**2.015**
Training time [min]	39.7	**39.6**	41.5	44.2	67.0
Prediction window size $l = 100$					
NRMSE (Eq. 10) [%]	2.127	2.155	2.157	2.430	**1.979**
Training time [min]	**42.0**	**42.0**	43.6	48.1	66.9

box" syndrome (Rudin, 2019), which ML models tend to be afflicted with; this should facilitate better trust in the predictions.

However, the present work demonstrates that none of the aforementioned reasons hold true for the considered cases of thermal problems concerned with solution reconstruction by providing the following arguments:

1. Transformer-based models exhibited lower or comparable prediction accuracy relative to the simple LSTM model.

2. The training times of the Transformer-based models are higher or comparable to the simple LSTM model.

3. The attention maps (Appendix A) may provide interpretability improvement for the language-related tasks, such as NLP (Vaswani et al., 2017) and computer vision (Kolesnikov et al., 2021), since the significance of the attention weights in attention maps can be easily intuitively deducted from the relationship between words in a sentence and from the parts of an image the attention operation "pays attention" to, respectively. However, this is not so easily done when dealing with just temporal data, as the relationship between values at different time steps cannot be interpreted in such an intuitive fashion, especially for the problems intrinsically based on a set of differential equations. Moreover, the LSTM layer weights can be visualised in the similar manner (Appendix B) and have the same rather low level of interpretability for the problems considered in this paper. Consequently, the interpretability advantage is negated.

Overall, the results of this study suggest that there is no significant benefit to using complex Transformer-based ML models over the conventional simper ML models, such as classic LSTM network, for solving transient thermal field reconstruction problems.

Regarding the application of these models to the practical problems, such as the one outlined in Sub-section 2.1, the appropriate number of the reliable forward simulations would need to be available in order to generate the training

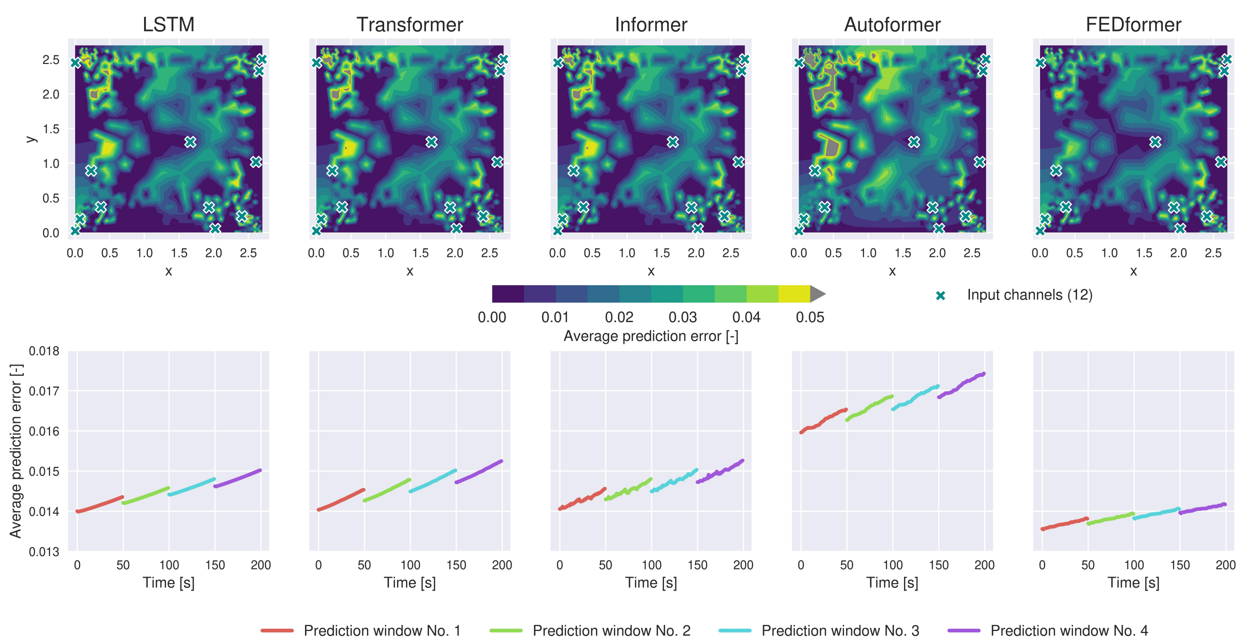

FIG. 15: Time-averaged prediction error distribution calculated using Eq 18 (top row) and prediction errors averaged at each time step calculated using Eq. 15 (bottom row) for the 2D heat conduction for five models with the prediction window size $l = 50$. For this error visualisation four prediction windows located consecutively to one another are selected. The twelve green crosses are the input channels.

data. Or, alternatively, the abundance of experimental data would need to be collected, which is rarely possible. Additionally, the computational effort of the training process and the memory usage increase with the length of the input time sequence (Table 7); consequently, for more complex problems significantly more time should be allocated for the training and VRAM (GPU memory) more carefully managed with batching.

6. ACKNOWLEDGEMENTS

This work was part-funded by the United Kingdom Atomic Energy Authority (UKAEA) and the Engineering and Physical Sciences Research Council (EPSRC) under Grant Agreement Numbers EP/W006839/1, EP/T517987/1 and EP/R012091/1. The authors acknowledge the support of Supercomputing Wales and AccelerateAI projects, which is part-funded by the European Regional Development Fund (ERDF) via the Welsh Government for giving us access to NVIDIA A100 40GB GPUs for batch training.

ACKNOWLEDGMENTS

This work was part-funded by the United Kingdom Atomic Energy Authority (UKAEA) and the Engineering and Physical Sciences Research Council (EPSRC) under Grant Agreement Numbers EP/W006839/1, EP/T517987/1 and EP/R012091/1. We acknowledge the support of Supercomputing Wales and AccelerateAI projects, which is part-funded by the European Regional Development Fund (ERDF) via the Welsh Government for giving us access to NVIDIA A100 40GB GPUs for batch training.

REFERENCES

Altché, F. and de La Fortelle, A., An LSTM Network for Highway Trajectory Prediction, *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 353–359, 2017.

Arridge, S., Maass, P., Öktem, O., and Schönlieb, C.B., Solving Inverse Problems Using Data-Driven Models, *Acta Numerica*, vol. **28**, pp. 1–174, 2019.

Bangian-Tabrizi, A. and Jaluria, Y., An Optimization Strategy for the Inverse Solution of a Convection Heat Transfer Problem, *International Journal of Heat and Mass Transfer*, vol. **124**, p. 1147 – 1155, 2018.

Bengio, Y., Simard, P., and Frasconi, P., Learning Long-Term Dependencies with Gradient Descent Is Difficult, *IEEE Transactions on Neural Networks*, vol. **5**, no. 2, pp. 157–166, 1994.

Bloem, P., Transformers from scratch, August, 2019, accessed on: March 29, 2023.
URL `https://peterbloem.nl/blog/transformers`

Électricité de France (EDF), Code_Aster: Open source finite element solver, analysis of structures and thermomechanics for studies and research, Open source on `https://code-aster.org/`, , 1989–2023.

Gers, F.A., Schmidhuber, J., and Cummins, F., Learning to Forget: Continual Prediction with LSTM, *Neural Computation*, vol. **12**, no. 10, pp. 2451–2471, 2000.

Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*: The MIT Press, 2016.

Hancock, D., Employing Additive Manufacturing for Fusion High Heat Flux Structures, Phd thesis, University of Sheffield, 2018.

Hochreiter, S. and Schmidhuber, J., Long Short-Term Memory, *Neural computation*, vol. **9**, pp. 1735–80, 1997.

Hsu, W.N., Zhang, Y., Lee, A., and Glass, J.R., Exploiting Depth and Highway Connections in Convolutional Recurrent Deep Neural Networks for Speech Recognition, *Interspeech*, 2016.

TABLE 7: Computational complexity and memory usage of the models considered in this paper.

Model type	LSTM	Transformer	Informer	Autoformer	FEDformer
Computation complexity (training)	$O(L)$	$O(L^2)$	$O(LlogL)$	$O(LlogL)$	$O(L)$
Memory usage (training)	$O(L)$	$O(L^2)$	$O(LlogL)$	$O(LlogL)$	$O(L)$

[a] L represents the length of the input time sequence.

Ivanovic, B. and Pavone, M., The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs, *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Kingma, D.P. and Ba, J., Adam: A method for stochastic optimization, , 2017.

Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Uszkoreit, J., Beyer, L., Minderer, M., Dehghani, M., Houlsby, N., Gelly, S., Unterthiner, T., and Zhai, X., An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2021.

Lim, B. and Zohren, S., Time-Series Forecasting with Deep Learning: a Survey, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. **379**, no. 2194, p. 20200209, 2021.

Qin, C., Zhang, A., Zhang, Z., Chen, J., Yasunaga, M., and Yang, D., Is chatgpt a general-purpose natural language processing task solver?, , 2023.

Ren, L., Dong, J., Wang, X., Meng, Z., Zhao, L., and Deen, M.J., A Data-Driven Auto-CNN-LSTM Prediction Model for Lithium-Ion Battery Remaining Useful Life, *IEEE Transactions on Industrial Informatics*, vol. **17**, no. 5, pp. 3478–3487, 2021.

Rudin, C., Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead, *Nature Machine Intelligence*, vol. **1**, pp. 206–215, 2019.

Tamaddon-Jahromi, H.R., Chakshu, N.K., Sazonov, I., Evans, L.M., Thomas, H., and Nithiarasu, P., Data-Driven Inverse Modelling through Neural Network (Deep Learning) and Computational Heat Transfer, *Computer Methods in Applied Mechanics and Engineering*, vol. **369**, p. 113217, 2020.

Tarantola, A., *Inverse Problem Theory and Methods for Model Parameter Estimation*: Society for Industrial and Applied Mathematics, 2004.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., and Polosukhin, I., Attention Is All You Need, *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc., 2017.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L., Transformers in time series: A survey, , 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A., Transformers: State-Of-The-Art Natural Language Processing, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, Online, pp. 38–45, 2020.

Wu, B., Wang, L., and Zeng, Y.R., Interpretable Wind Speed Prediction with Multivariate Time Series and Temporal Fusion Transformers, *Energy*, vol. **252**, p. 123990, 2022.

Wu, H., Xu, J., Wang, J., and Long, M., Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting, *Neural Information Processing Systems*, 2021.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T., On Layer Normalization in the Transformer Architecture, *Proceedings of the 37th International Conference on Machine Learning*, III, H.D. and Singh, A. (Eds.), Vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 10524–10533, 2020.

Yu, Y., Si, X., Hu, C., and Zhang, J., A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures, *Neural Computation*, vol. **31**, no. 7, pp. 1235–1270, 2019.

Zhang, Y., Wang, S., Ji, G., , A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications, *Mathematical problems in engineering*, vol. **2015**, 2015.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W., Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. **35**, no. 12, pp. 11106–11115, 2021.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R., FEDformer: Frequency Enhanced Decomposed Transformer for Long-Term Series Forecasting, *Proceedings of the 39th International Conference on Machine Learning*, Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (Eds.), Vol. 162 of *Proceedings of Machine Learning Research*, PMLR, pp. 27268–27286, 2022.

Zwicker, D., Py-Pde: A Python Package for Solving Partial Differential Equations, *Journal of Open Source Software*, vol. **5**, p. 2158, 2020.

7. APPENDICES

A Attention Maps for Transformer-Based Models

In general, self-attention operation or a variation thereof allows the model to attend to different parts of the input vector sequence in order to generate an output vector sequence, meaning that it concentrates more on the selected input vectors, which it deems to be more relevant, while generating a certain output vector (Vaswani et al., 2017). As it is mentioned in Sub-section 2.4, the queries and keys produced using query and key weight matrices are combined to produce a weight matrix, which can be called an attention matrix; each element of this attention matrix, which can be referred to as an attention score, represents the contribution it makes towards a certain output vector.

In order to illustrate this, the NLP example shown in Figure 17 can be considered (Bloem, 2019). The key representing the qualities the book contains and the query, which represent the reader preferences, are matched using a dot product to obtain the attention score. This score demonstrates how well the book matches the reader's preferences. Generally, the attention score indicates the degree of relevance between the key and the query; thus, it shows to what degree a certain output vector out of the output vector sequence is influenced by a certain input vector out of the input vector sequence. In the case of the transient thermal field reconstruction problems considered in this paper, each input vector i contains the information provided by the input channels at i^{th} time step, while each output vector i contains the information provided by the output channels at i^{th} time step (Fig. 9 and 13). For the classic Transformer, the attention score indicating the degree to which the output vector i is influenced by the input vector j can be calculated using a dot product between the query i and the key j (Eq. 19).

$$Attention\_score\_(i,j) = \{\boldsymbol{q_i}\}^T \cdot \{\boldsymbol{k_j}\} \quad i,j = 1,2,...,t \tag{19}$$

where $\{\boldsymbol{q_i}\}$ is the query vector i corresponding to the output vector i, $\{\boldsymbol{k_j}\}$ is the key vector j corresponding to the input vector j, and t is a sequence length (Table 2).

The attention matrix containing the attention scores can be visualised as a map by projecting it on an image where each cell (i,j) corresponds to the attention score computed for query vector i and key vector j. Figure 18 shows an example of an attention score (i,j) located on the attention map image.

Transformer-based models considered in this paper employ a multi-head attention operation (Figure 7) where several self-attention operations are performed in parallel (Sub-section 2.4). Consequently, each multi-head attention operation generates a number of attention maps equal to the number of self-attention heads the multi-head attention operation is comprised of. Eight attention heads are used in this paper (Table 2); therefore, eight attention maps can be generated per one multi-head attention operation. It is important to note that these attention scores should not be used to compare the different heads between each other, they should only be used to compare the attention scores from one attention map between each other. Furthermore, the attention matrix size for Transformer (Vaswani et al., 2017) and Informer (Zhou et al., 2021) is prediction window size l by prediction window size l (Table 2); however, the attention matrix size for Autoformer (Wu et al., 2021) and FEDformer (Zhou et al., 2022) is sequence length by d_{model}/h (Table 2), this is because the keys are projected in the sequence length direction prior to the calculation of attention scores.

Figures 19 and 20 show four out of eight attention maps for multi-head attention in the first layer of each model for $l = 50$. The attention scores are scaled between 0 and 1 with the scaler fitting done separately for each attention map.

Some distinctive characteristics can be observed in these attention maps. Transformer and Informer attention maps are diverse, which means that different attention heads tend to attend to different parts of the input vector sequence. This is beneficial for the model's performance because it signifies that the attention heads are extracting different features from the input data. Transformer's attention maps are smoother than Informer's ones, which can be attributed to the fact that Informer utilises a sparse version of Transformer's self-attention operation. On the other hand, in spite of some differences Autoformer's attention maps look vastly similar, which is undesirable since this means that the attention heads are attempting to extract the same features from the input data. The attention map similarity might be the reason why Autoformer's performance is worse than the performance displayed by other models.

Overall, the ability to produce attention maps allows the Transformer-based model to be more interpretable than other ML models, as they show how the model focuses on the various sections of the input vector sequence. By visualising the the attention scores in this manner, it is possible to see which parts of the input sequence the model "pays attention" to at each time step.

B Weight Visualisations for LSTM Models

The purpose of this appendix is to illustrate that the weight visualisations similar to the attention maps can be easily constructed for the one-layer LSTM model. These LSTM weight visualisations have the same low level of intuitive interpretability as for the Transformer-based models specifically for the transient thermal field reconstruction problems considered in this paper. Figure 21 shows the values of h_t for each time step l (Eq. 3) for LSTM models with the prediction window size equal to 50; these values are scaled between 0 and 1 with the scaler fitting done separately for each visualisation. For consistency these values are refereed to as attention scores in Figure 21.

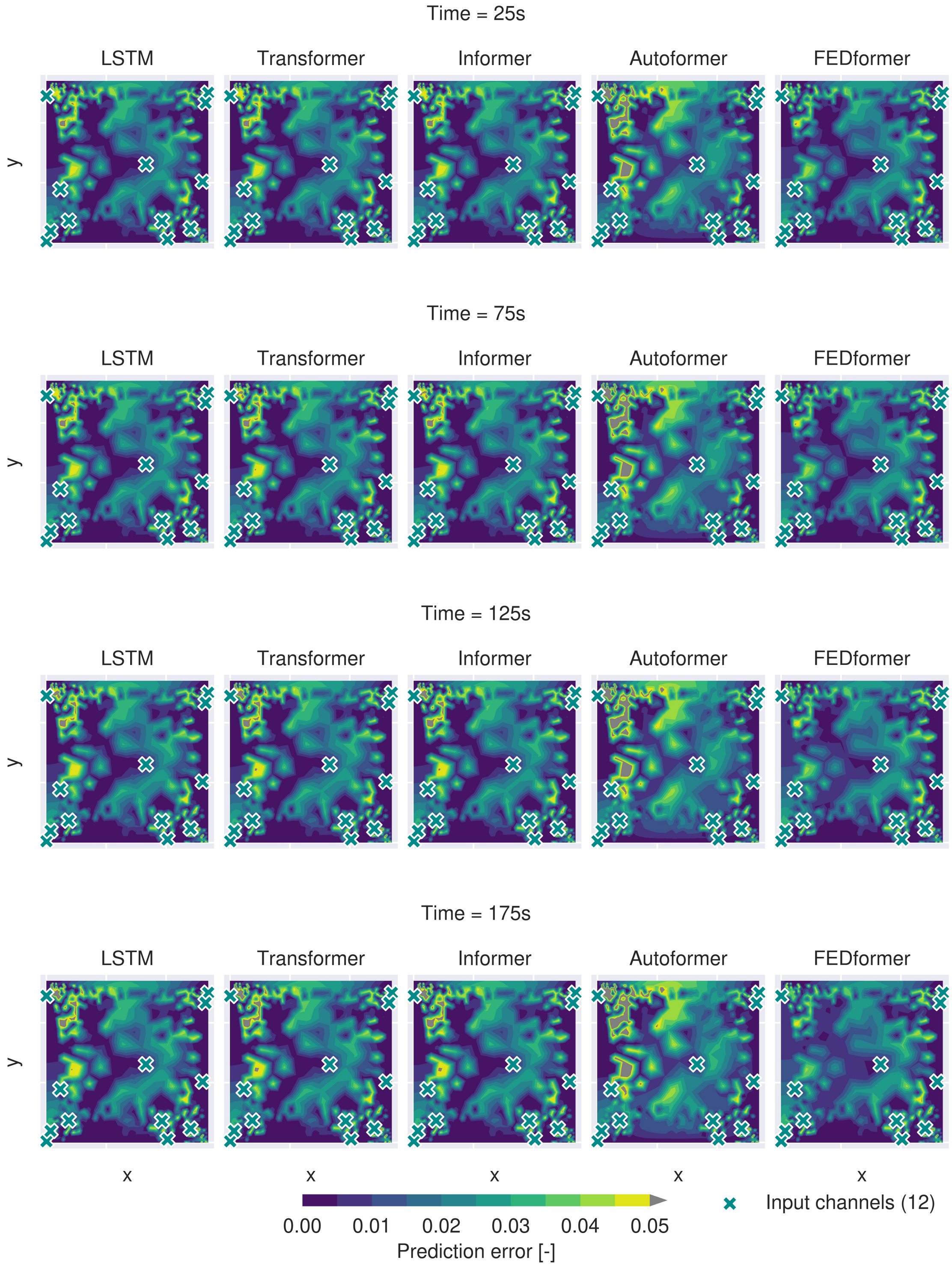

FIG. 16: Prediction error distribution dependence on time for the 2D heat conduction for five models with the prediction window size $l = 50$. The four selected time instances correspond to the middle of each prediction window in Figure 15. The prediction errors are calculated using Eq. 14 for fixed time instances i.

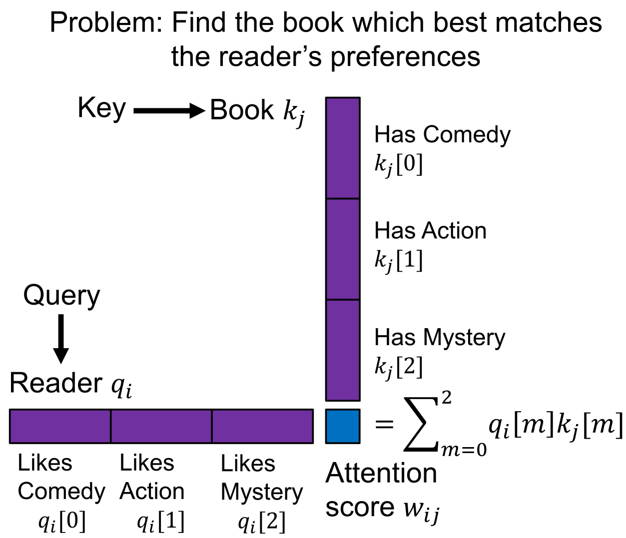

FIG. 17: The book example illustrating the self-attention operation. The key representing the qualities the book contains and the query, which represent the reader preferences, are matched using a dot product to obtain the attention score. This score demonstrates how well the book matches the reader's preferences. Generally, the attention score indicates the degree of relevance between the key and the query; thus, it shows to what degree a certain output vector out of the output vector sequence is influenced by a certain input vector out of the input vector sequence. In the case of the transient thermal field reconstruction problems considered in this paper, each input vector i contains the information provided by the input channels at i^{th} time step, while each output vector i contains the information provided by the output channels at i^{th} time step (Figures 9 and 13).

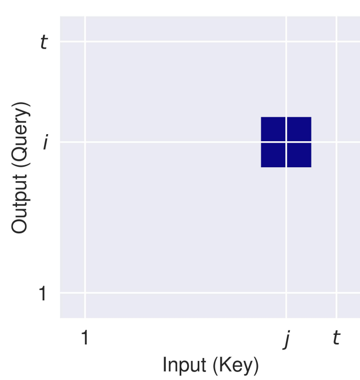

FIG. 18: Attention score example. The attention matrix containing the attention scores can be visualised as a map by projecting it on an image where each cell (i, j) corresponds to the attention score computed for query vector i and key vector j. This Figure shows an example of an attention score (i, j) located on the attention map image.

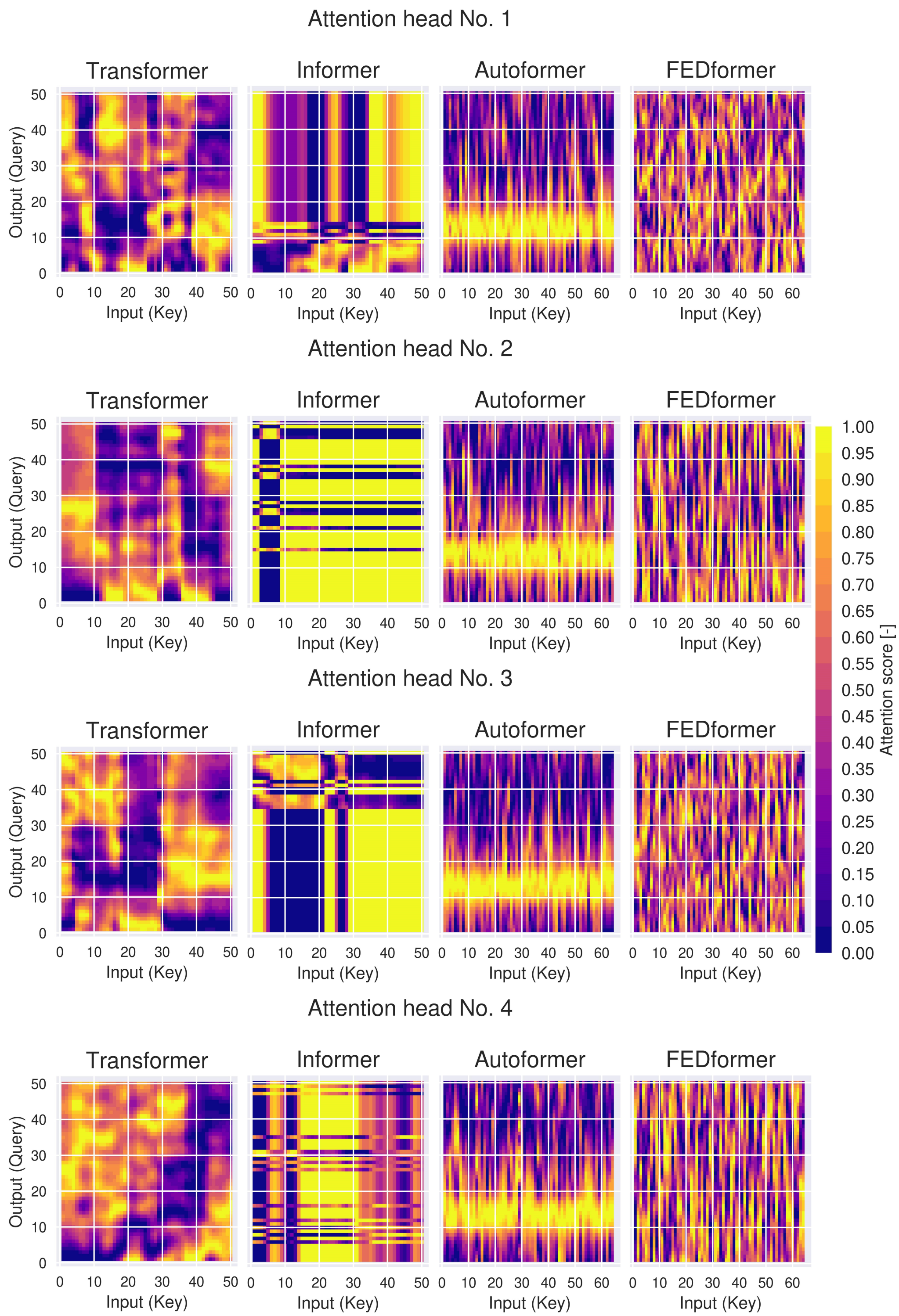

FIG. 19: Attention maps for Encoder layer No. 1 for 1D heat conduction problem for four models considered in this paper. The prediction window size l is equal to 50. This Figure show four out of eight attention maps for multi-head attention in the first layer of each model. The attention scores are scaled between 0 and 1 with the scaler fitting done separately for each attention map.

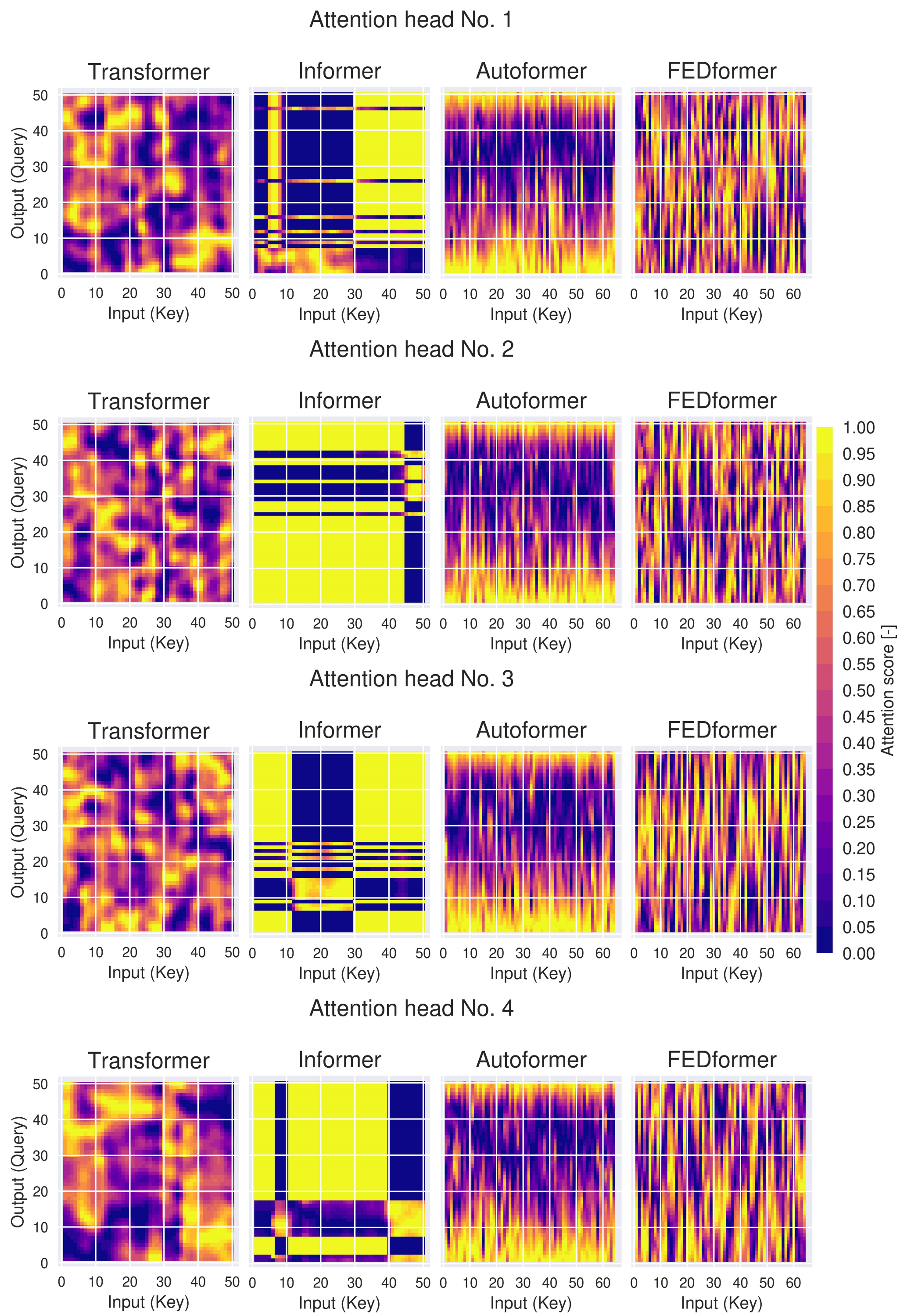

FIG. 20: Attention maps for Encoder layer No. 1 for 2D heat conduction problem for four models considered in this paper. The prediction window size l is equal to 50. This Figure shows four out of eight attention maps for multi-head attention in the first layer of each model. The attention scores are scaled between 0 and 1 with the scaler fitting done separately for each attention map.

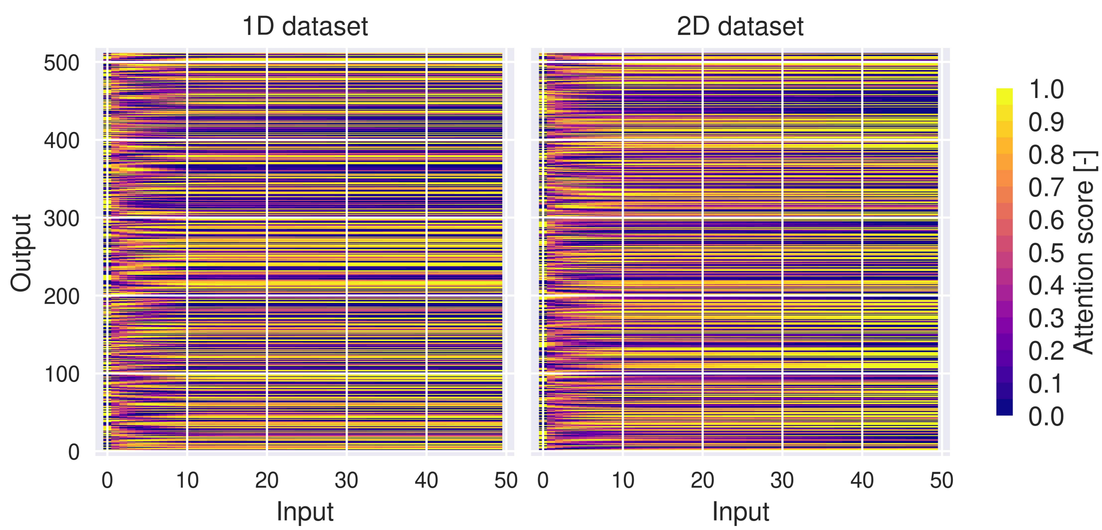

FIG. 21: Weight visualisations for 1D and 2D heat conduction problems for LSTM models with the prediction window size equal to 50. This Figure shows the values of h_t for each time step l (Eq. 3). The attention scores are scaled between 0 and 1 with the scaler fitting done separately for each weight visualisation.