Predicting Heat Transfer Coefficient Using Bidirectional Long Short-Term Memory

Ankan Basu, Aritra Saha, and Sumanta Banerjee

Abstract The prediction of Heat Transfer Coefficient (HTC) plays a critical role in optimizing the thermal performance of systems during heat treatment processes. Traditional numerical methods struggle to solve the Inverse Heat Transfer Problem (IHTP) associated with HTC prediction. In this paper, a novel approach has been proposed that leverages the power of machine learning, specifically Bidirectional Long Short-Term Memory (BiLSTM) networks, to estimate the HTC value. The developed model demonstrates remarkable precision, achieving an impressive $\approx 98.75\%$ accuracy. This outperforms conventional feed-forward networks. The proposed machine learning approach offers several advantages over traditional methods. It provides rapid estimations of the key characteristics of the HTC function, offering quick insights into the heat transfer process. Furthermore, machine learning algorithms have the capability to learn from data and capture complex relationships, making them highly suitable for HTC estimation tasks. Compared to heuristic search algorithms and swarm-based approaches, the present approach significantly reduces computational requirements and maintains excellent predictive performance. The results obtained highlight the potential of machine learning in optimizing heat treatment processes and improving overall performance.

A. Basu (✉)
Department of Computer Science and Engineering, Jadavpur University, Kolkata, India
e-mail: ankanb.cse.pg@jadavpuruniversity.in

A. Saha
Department of Computer Science and Engineering, Heritage Institute of Technology, Kolkata, India

Fakultät Für Elektrotechnik Und Informatik, Gottfried Wilhelm Leibniz Universität Hannover, Hannover, Germany

A. Saha
e-mail: aritra.saha.cse23@heritageit.edu.in

S. Banerjee
Department of Mechanical Engineering, Heritage Institute of Technology, Kolkata, India
e-mail: sumanta.banerjee@heritageit.edu

A. Mukhopadhyay and K. Ghosh (eds.), *Advances in Thermo-Fluid Engineering*, Lecture Notes in Mechanical Engineering,
https://doi.org/10.1007/978-981-97-7296-4_9

614263_1_En_9_Chapter ✓ TYPESET ☐ DISK ☐ LE ✓ CP Disp.:19/10/2024 Pages: 17 Layout: T1-Standard

Keywords HTC Prediction · Inverse Heat Transfer Problem (IHTP) · Machine learning · Bidirectional Long Short-Term Memory (BiLSTM) · Neural networks · Heat treatment optimization

1 Introduction

The Heat Transfer Coefficient (HTC), denoted as "h", is a measure of how effectively thermal energy is transferred between two surfaces, or a surface and a fluid. It quantifies the rate of heat transfer per unit area per unit temperature difference.

In order to optimize the thermal performance and properties of systems during heat treatment, it is crucial to determine the HTC value, which indicates the amount of heat exchanged between the workpiece and the cooling medium (water, oil, etc.). For example, heat treatment methods can be employed to customize the microstructure of materials. This process entails subjecting a workpiece to elevated temperatures and then carefully controlling its cooling process. By doing so, specific mechanical properties of the material can be enhanced [1, 2].

The (metallurgical) properties of cast metals are, in general, influenced by their cooling rates during solidification or in thermal treatment modes. In instances of continuous casting, secondary cooling conditions play a crucial role in determining product quality and process efficiency. Significantly rapid cooling rates can induce thermal stresses within the product, which results in cracks or fractures. Conversely, the metallurgical and economic benefits of a particular casting process, subjected to very low cooling rates, might be adversely affected. As a result, emphasis must be placed on the fundamental understanding of the variation trends of (local and/or average) heat transfer coefficients associated with various cooling techniques. This understanding aids in calculating cooling rates and evaluating the suitability of these methods for the production processes.

In order to achieve the desired micro-structural changes within a workpiece, the engineering problem ultimately boils down to the estimation of the (average value of) HTC. This estimation is useful, for instance, to set up appropriate thermal boundary conditions, duration of quenching, and so on [3]. The prediction of HTC involves solving an "Inverse Heat Transfer Problem" (IHTP), which cannot be easily solved using traditional numerical methods due to various factors such as non-uniqueness of the solution, ill-posedness, complex nature of physical systems, nonlinearities, limited observations, etc. To tackle this challenge, various techniques have been employed, which include heuristic search algorithms such as Genetic Algorithms [4–6], Particle Swarm Optimization [4, 7], and other approaches based on Swarm theory [8]. However, these methods typically exhibit high computational requirements [9].

Alternatively, machine learning methods can be employed to address the same objective, providing quick estimations of the key characteristics of the HTC function. Machine learning techniques have the ability to learn from data and capture complex relationships, making them suitable for HTC estimation tasks [10]. Use of Artificial Neural Networks (ANNs) for solving IHTPs is a powerful approach [11–22]. In case of IHTP, ANNs can be effectively utilized to approximate the nonlinear relationship between the input data (temperature measurements) and the output data (heat flux, boundary conditions, etc.).

Sreekanth et al. [11] have used ANN to determine the surface HTC at the liquid–solid interface using the solid's internal temperature profile information. Their particular case has relevance and applications in food process engineering. Soeiro et. al. [12] have used ANN to obtain initial estimates for solving IHTP using Levenberg–Marquardt method. Mirsephai et al. [13] have utilized ANN to calculate the heat emitted to irradiative batch drying process. They have utilized the temperature history for a point on the bottom surface of a simple laboratory drying furnace with a halogen lamp. GC et al. [15] developed recurrent and genetic algorithm tuned neural networks to develop an input–output relationship in squeeze casting process. Zhang et al. [16] have developed multi-domain physics-informed neural network (M-PINN) for solving forward and inverse steady-state heat conduction problems in multilayer media. The authors have decomposed the multilayer media into several sub-domains (using the domain decomposition technique). Then, a fully connected neural network was employed to estimate each sub-domain's temperature field. Thereafter, the sub-networks were combined to form a large total network by using continuity conditions on the interfaces. Han, et al. [17] have employed an encoder-decoder (based on LSTM) neural network to estimate instantaneous heat flux at the tool-chip region during turning process, and demonstrated the potential of their proposed method through both numerical and experimental tests. Zhu et al. [18] have proposed an approach of combining the Convolution Neural Network (CNN) and the Long Short-Term Memory (LSTM) to perform real-time prediction of multi-dimensional thermal boundary condition parameters based on the target's temporal temperature field image. The authors performed experimental studies to verify the success of application of their technique and showed its significance for the study of the transient IHTP. Sajjad et al. [19] developed deep neural networks to predict the pool boiling HTC for sintered coated porous surfaces. The best model had five layers with (11, 30, 15, 1, 1) nodes in the layers, from input to output, and had an error of 5.74%. Cui et al. [20] presented a novel approach using Radial Base Function (RBF) neural network and reported satisfactory results for predicting simulation results of combustion and heat transfer characteristics in supercritical CO_2 CFB boiler. Doner et al. [21] utilized deep neural network (the best model having 10 layers with 10 nodes) to analyse the heat transfer in a bubbling fluidised bed combustion system and reported high accuracy of the prediction results on different values of the input data.

Szénási et al. [22] have reported promising results while implementing ANNs to estimate HTC values for immersion quenching processes. Nonetheless, the researchers have expressed concern on the accuracy of the employed method, indicating the need for further refinement. Their concern entails searching for the right network architecture, determining the optimal number of nodes, and other improvements. However, owing to extensive data, this process is quite time-consuming, requiring several weeks to train each network and test potential architectures. As a solution, the authors have decided to release the proposed database to all researchers in the field, enabling collaboration and exploration of potential solutions.

The object of the present computational study is primarily aimed at prediction of the (local) coefficients of heat transfer from hot metal surfaces to the coolant, using a generic instance of temperature profile using machine learning methods, particularly Bidirectional LSTM (BiLSTM). This study, therefore, warrants practical relevance in industrial applications involving continuous casting of metals [23, 24]. The performance of BiLSTM has been compared with the Feed-forward Neural Network (FFNN), the Recurrent Neural Network (RNN), and the LSTM-based models. It is observed that the BiLSTM yields higher accuracy and performs significantly better than the other models applied here.

2 Description of the ML Algorithms

In this study, the FFNN, RNN, LSTM and BiLSTM were used to predict HTC from temperature values. A brief description of these algorithms is presented below.

The ANNs are computational models that draw inspiration from the structure and functionality of the human brain [25]. The fundamental building block of a neural network is the artificial neuron, arranged in layers. Each neuron takes one or more inputs, applies a mathematical operation to them (usually a weighted sum), and passes the result through an activation function to produce an output. These outputs are then propagated through the network, with each layer transforming the input until a final output is generated [26, 27].

This simple type of neural network is called feed-forward network (FFNN), as the information-flow is unidirectional from input to output. There are no feedback connections to feed the outputs back into the model. When these FFNNs are extended to include feedback connections, they are called RNNs [28]

The RNNs are neural networks designed for sequential data [29, 30]. Unlike traditional feed-forward neural networks, RNNs have recurrent connections that allow information to persist and flow through the network. RNNs excel in tasks where temporal dependencies and context are important (Fig. 1).

At each time step, the RNN produces a hidden state (denoted by $h_{(t)}$), which is fed back to the network, and an output (denoted by $y_{(t)}$). The hidden state of a particular time step t (i.e., $h_{(t)}$) is calculated using the previous hidden state (i.e., $h_{(t-1)}$) and the current input (i.e., $x_{(t)}$). Then this new hidden state (i.e., $h_{(t)}$) is used to calculate the final output (i.e., $y_{(t)}$) of that time step.

This is summarized in the formulae below:

$$\left.\begin{aligned} h(t) &= \phi_h\left(W_{hh}^T h_{(t-1)} + W_{xh}^T x_{(t)} + b_h\right) \\ y_t &= \phi_y\left(W_{hy}^T h_{(t)} + b_y\right) \end{aligned}\right\} \tag{1}$$

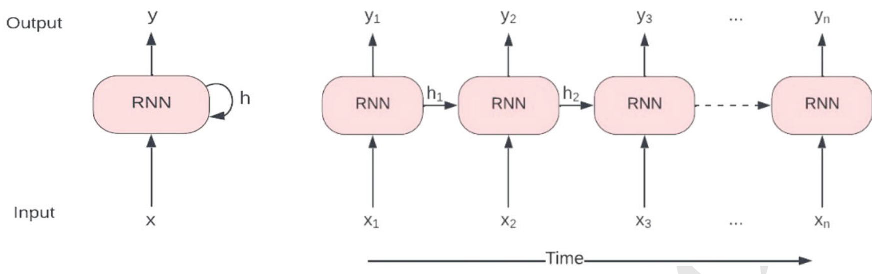

Fig. 1 **a** RNN cell, and **b** the RNN unrolled through time

In Eq. (1) W_{hh} is the weight matrix associated with the hidden state ($h_{(t-1)}$), W_{xh} is the matrix of weights associated with the input $x_{(t)}$, and b_h is the bias for calculating the new hidden state $h_{(t)}$. Likewise, W_{hy} denotes the matrix of weights and b_y denotes the bias for calculating the new output (denoted by $y_{(t)}$) at time step t. The respective activation functions are ϕ_h and ϕ_y. Usually in RNN *tanh* activation function is used.

However, traditional RNNs can struggle with long-term dependencies due to the vanishing and exploding gradient problem, where gradients diminish or explode as they propagate through time during training [30, 31].

The LSTM is an extension of RNNs that addresses the vanishing gradient problem [32]. Their ability to model long-term dependencies and retain contextual information makes them particularly suitable for tasks requiring an understanding of sequential patterns and dynamics such as speech recognition, language modeling, machine translation, sentiment analysis, and more (Fig. 1).

The LSTM introduces a memory cell to capture long-term dependencies in sequential data (refer to Fig. 2). LSTM has separate connections for long-term and short-term memories. The LSTM cell includes specialized gates (input, forget, and output) to control information flow.

The long-term (memory) state $c_{(t-1)}$ flows through the network (in a left-to-right direction in Fig. 2). Initially, it undergoes a Forget Gate, allowing certain memories to be discarded. Then, new memories are introduced through an addition operation, where the Input Gate selects the memories to be added. The outcome $c_{(t)}$ is then directly forwarded without any additional transformations. Consequently, some information (memories) is discarded while some new ones are added at each time step. Thereafter the long-term state is copied and the *tanh* function is applied to it, followed by filtering via the Output Gate. As a result, the short-term state $h_{(t)}$ (equivalent to the cell's output $y_{(t)}$ for that particular time step) is generated.

In an LSTM cell, there are four fully connected (FC) feed-forward neural network layers. The input vector $x_{(t)}$ of the current time step and the short-term memory (hidden state) of the previous time step $h_{(t-1)}$ are fed as inputs to the FC layers. Among these layers, the main layer produces the output $g_{(t)}$; instead of going straight out, only the crucial parts of it are stored in the long-term state, and the rest is discarded. The remaining three FC layers control the gates. Let $i_{(t)}$ represent the output of the FC layer, which controls the input gate (Fig. 2); $f_{(t)}$ represent the output of the FC layer,

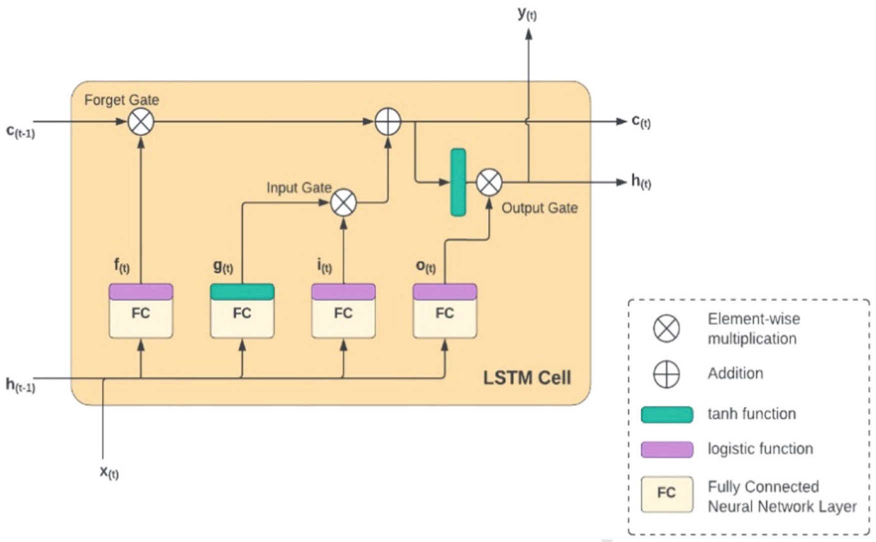

Fig. 2 LSTM Cell showing the long-term and short-term memories and the different gates

which controls the Forget Gate; and $o_{(t)}$ the output of the FC layer, which controls the Output Gate. Equation (2) provides a summary of these above operations at an instance of time (t).

$$\left.\begin{aligned}
i_{(t)} &= \sigma\left(W_{xi}^{T}x_{(t)} + W_{hi}^{T}h_{(t-1)+b_i}\right)\\
f_{(t)} &= \sigma\left(W_{xf}^{T}x_{(t)} + W_{hf}^{T}h_{(t-1)+bf}\right)\\
o_{(t)} &= \sigma\left(W_{xo}^{T}x_{(t)} + W_{ho}^{T}h_{(t-1)+b_o}\right)\\
g_{(t)} &= \tanh\sigma\left(W_{xg}^{T}x_{(t)} + W_{hg}^{T}h_{(t-1)+b_g}\right)\\
c_{(t)} &= f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)}\\
y_{(t)} &= h_{(t)} = o_{(t)} \otimes \tanh\left(c_{(t)}\right)
\end{aligned}\right\} \tag{2}$$

In Eq. (2), the weight matrices of each of the four FC layers for their connection to $x_{(t)}$ are denoted by W_{xi}, W_{xf}, W_{xo}, W_{xg}. Similarly, the weight matrices of each of the FC layers, for their connection to $h_{(t-1)}$ (hidden state of previous time step), are denoted as W_{hi}, W_{hf}, W_{ho}, and W_{hg}. The bias terms associated with each of the FC layers are denoted as b_i, b_f, b_o, and b_g. The symbol $\otimes$ represents element-wise multiplication between two vectors.

The Bidirectional RNNs enhance the standard RNN architecture by incorporating information from both past and future contexts [33]. BiLSTM is a similar extension of LSTM [34].

In BiLSTM, two LSTMs separately process the information in the forward and the reverse directions (refer to Fig. 3). The input sequence is $[x_1, x_2, x_3, \ldots, x_n]$

while the output sequence is [y_1, y_2, y_3, …, y_n]. The outputs of the two LSTMs can be merged in several ways; for example, by adding, averaging, concatenating, and so on, to form the final output sequence. By considering both preceding and succeeding elements, BiLSTMs capture a more comprehensive understanding of the input sequence, thereby improving the model's ability to make informed predictions by incorporating future context. In Fig. 3, it may be noted that between two consecutive LSTM time steps, there are two separate connections (one each for the long-term memory and the short-term memory). However, for compactness of representation, only one connection has been shown (Fig. 4).

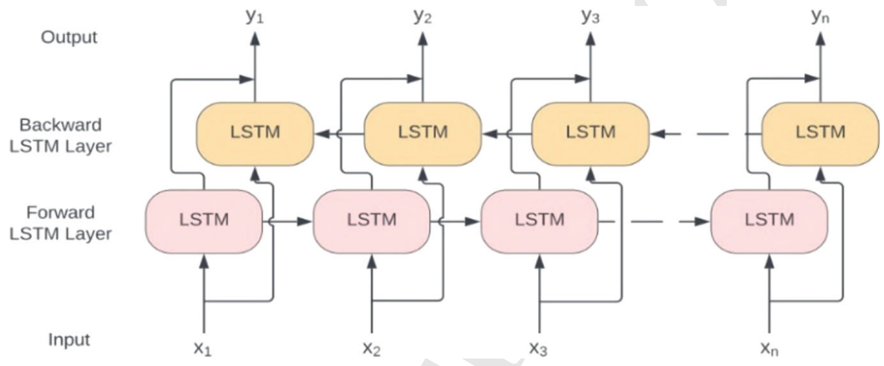

Fig. 3 Schematic representation of bidirectional LSTM unrolled through time

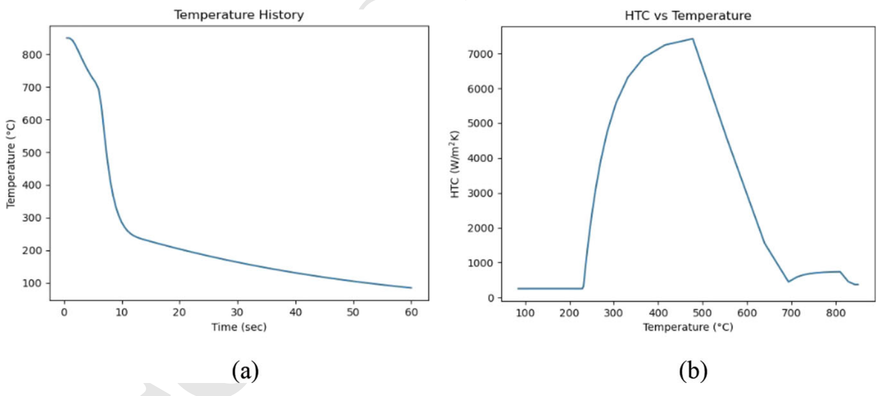

Fig. 4 A sample data from the dataset **a** temperature data plotted against time, and **b** HTC data plotted against temperature

3 Methodology

3.1 Data

In the present study, the dataset created by Szénási et al. [22] has been utilized. The latter [22] presents an innovative approach to create random HTC functions. The method relies on control points and extra parameters that describe the shapes of HTC profiles.

The dataset comprises information collected during the quenching process of a cylindrical bar (made from Inconel 600 alloy, having a diameter of 20 mm and length of 60 mm) from 850 °C to room temperature. Temperature data was recorded every 0.5 s for a duration of 60 s. Consequently, each temperature data point is represented by a vector of size 120.

The HTC values were measured at 5 control points. The dataset includes htc_header files that provide local information at these control points, such as (local value of) temperature ($P_i.temp$), HTC ($P_i.htc$), and alpha (α_i, a parameter used for interpolating and calculating HTC values at other temperatures in the range). The ranges of the parameters are specified below [22]:

$$\left.\begin{array}{c} 0°C \leq P_i.temp \leq 850°C;\ \forall i \in \{1, 2, \ldots, N\} \\ 0\frac{W}{m^2K} \leq P_i.htc \leq 1200\frac{W}{m^2K};\ \forall i \in \{1, 2, \ldots, N\} \\ P_i.temp < P_2.temp < \cdots < P_N.temp \\ -1.0 \leq \alpha_i \leq +1.0\ \forall i \in \{1, 2, \ldots, N\} \end{array}\right\} \quad (3)$$

In Eq. (3), $P_i$denotes the the i-th control point. The total number of control points is denoted by N (here, 5). The data descriptor presents formulas for interpolating and determining HTC values at any temperature (within the temperature range of 0 °C to 850 °C). In order to interpolate and find the HTC value at temperature T [HTC(T)] between two adjacent control points P_i and P_{i+1}, the following formulae (Eq. 4) are used (C_α is a scaling factor; usually 7 [22]):

$$\begin{aligned} \alpha_{i'} &= \frac{1}{\alpha_i C_\alpha} \\ \delta(T) &= \frac{T - P_i.temp}{P_{i+1}.temp - P_i.temp} \\ \gamma(T) &= \frac{1 - e^{\frac{-\delta(T)}{\alpha_{i'}}}}{1 - e^{\frac{-1}{\alpha_{i'}}}} \\ HTC(T) &= \begin{cases} P_i.htc + \gamma(T)(P_{i+1}.htc - P_i.htc), & if\ \alpha_i \neq 0 \\ P_i.htc + \delta(T)(P_{i+1}.htc - P_i.htc), & if\ \alpha_i \neq 0 \end{cases} \end{aligned} \quad (4)$$

The dataset contains HTC values for a total of 86 temperature-points (separated by 10 °C interval), which can be found using the same interpolation formulae as above (refer to Eq. 4). In this work, Eq. (4) has been used to calculate the corresponding HTC values for each of the (120) specified temperatures (recorded at 0.5-s intervals). Hence, each HTC data point is also represented by a vector of size 120.

The authors [22] have already divided the data into training, validation and testing datasets which have been employed to train and evaluate the neural networks in this study. The training dataset comprises of one million data points, while the testing and validation datasets consist of 100,000 data points each.

3.2 Model

Several neural network architectures based on FFNN, RNN, LSTM and BiLSTM were tested in this study. In the best model architecture selected, three BiLSTM layers are stacked on top of one another (refer to Fig. 5). The first layer has 2 × 50 neurons, while the next two layers have 2 × 100 neurons each. This yields an output of dimension 200 in each time step. This output is passed to a feed-forward layer with one output node. The activation function is linear in the last feed-forward layer.

In addition to this model, several other models were trained to compare their performances. At first, simple FFNNs were trained. The FFNNs had 120 input neurons and 120 output neurons, and experimentation was conducted by varying

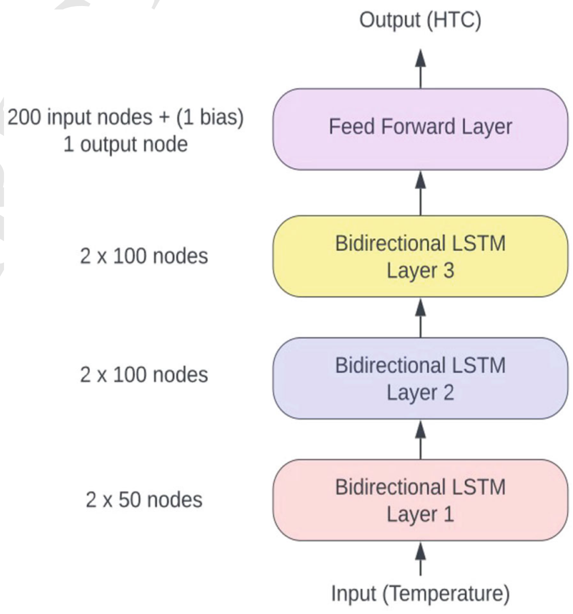
Output (HTC)
200 input nodes + (1 bias)
1 output node
Feed Forward Layer
2 x 100 nodes
Bidirectional LSTM Layer 3
2 x 100 nodes
Bidirectional LSTM Layer 2
2 x 50 nodes
Bidirectional LSTM Layer 1
Input (Temperature)

Fig. 5 Schematic representation of proposed model

the number of hidden layers. For the first configuration, a single hidden layer (with 50 neurons) was employed. In the second configuration, another hidden layer (with 100 neurons) was added on top of the first one. Lastly, in the third configuration, a third hidden layer (with 100 neurons) was introduced. The Rectified Linear Unit (ReLU) was the activation function for the hidden layers. The output layer had a linear activation function.

Then, the RNN-based models were trained, which are similar to the proposed best model. It contained RNN layers stacked on top of each other and a feed-forward layer (with one output) in the end. Experimentation was conducted with varying numbers of RNN layers. The first configuration had an RNN layer with 50 nodes; then another RNN layer with 100 nodes; and, finally, a feed-forward layer with one output node. In the next configuration, another RNN layer (having 100 nodes) was added after the last RNN layer. In the third configuration, a fourth RNN layer (having 100 nodes) was added after the last RNN layer.

In the next instance, the LSTM-based models were trained. They are exactly similar to the RNN-based models, except for the use of LSTM instead of simple RNN. Thereafter, the BiLSTM-based model was trained, which had the same configuration as the RNN-based models, but which made use of BiLSTM instead of the simple RNN.

The number of nodes (in respective layers) employed in FFNN, RNN, LSTM, and BiLSTM for the model variants have been specified in Table 1.

In all of the above models, the mean squared error (MSE) is used as the loss function (to be minimized), and the Adaptive Moment Estimation (Adam) is chosen as optimizer. The batch size is selected as 32 in line with the reviewed literature [35]. The MSE, the Mean Absolute Percentage Error (MAPE), and the Coefficient of Determination (R^2), selected as the metric for evaluating the performance of the models, are defined below. All the above models were trained until the MAPE values reflected no significant change over successive training epochs.

In Eq. (5) below, $\hat{y}$ denotes the predicted or estimated value, while the true value is denoted by y_i, , and $\bar{y}$ denotes the mean of y. The number of samples is denoted by N. The choice of MAPE as a metric for performance evaluation has been motivated by the wide range of HTC values in the dataset (spanning over $0\frac{W}{m^2K} \leq HTC \leq 12000\frac{W}{m^2K}$;). Using the absolute value of the error would not be

Table 1 Number of nodes (in respective layers) employed in FFNN, RNN, LSTM, and BiLSTM for the model variants

Model index	Feed-Forward (FFNN)	RNN based (RNN)	LSTM based (LSTM)	Bidirectional LSTM based (BiLSTM)
(1)	120, 50, 120	1, 50, 100, 1	1, 50, 100, 1	2*(1, 50, 100), 1
(2)	120, 50, 100, 120	1, 50, 100, 100, 1	1, 50, 100, 100, 1	2*(1, 50, 100, 100), 1
(3)	120, 50, 100, 100, 120	1, 50, 100, 100, 100, 1	1, 50, 100, 100, 100, 1	2*(1, 50, 100, 100, 100), 1

an appropriate measure, since the significance of a particular error value may vary (depending on whether the true value is low or high).

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$
$$\mathrm{MAPE} = \frac{1}{N}\left(\sum_{i=1}^{N}\frac{\left|y_i - \hat{y}_i\right|}{y_i} \times 100\right) \tag{5}$$
$$R^2 = \frac{\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}$$

The absolute percentage error provides a more meaningful assessment, as it considers the relative difference between the predicted and the true values. Also, the objective is to make predictions as close to the true values as possible, regardless of the magnitude of the true values. This justifies the use of the MSE as the loss function in the chosen model.

The input and output layers of the feed-forward networks have 120 nodes each, as the input (temperature) vector and the output (HTC) vector have dimensions of 120 each. But the LSTM and the RNN need only 1 (2*1 for BiLSTM input) node each for input and output, as they take one input data point at a time and the networks get unrolled over time to take sequential inputs.

4 Results and Discussion

The MAPE and R^2 scores of different neural network models are presented in Tables 2, 3, 4 and 5.

As seen in Table 2, for all tested FFNN models, the values of MAPE were significantly high despite increasing the number of hidden layers. Also, there are no signs of over fitting, as the validation and test accuracies are very close to the training accuracy.

In the present study, the RNN-based models were particularly difficult to train and the trends of error underwent fluctuations during training. For convergence within

Table 2 The MAPEs and R2 scores for the different FFNN models

Model	Training		Validation		Testing	
	MAPE	R2	MAPE	R2	MAPE	R2
FFNN (1)	12.1884	0.8864	12.2030	0.8858	12.1918	0.8884
FFNN (2)	8.6321	0.9114	8.6383	0.9116	8.6214	0.9128
FFNN (3)	8.3560	0.9123	8.3646	0.9118	8.3485	0.9139

acceptable error, the models had to be trained multiple times. This is possibly due to the vanishing and exploding gradient problems that traditional RNNs face while dealing with long sequences [30, 31]. Table 3 shows the MAPE values for the different RNN models.

Table 4 shows that, for all tested LSTM models, the values of MAPE remain virtually unchanged with the increasing complexity of the models. This has led to an adoption of the BiLSTM models for the present inverse heat transfer study, and underpins the justification of the adopted methodology.

For all tested BiLSTM models, as seen in Table 5, the values of MAPE remain almost unchanged with the increasing complexity of the models. The BiLSTM-based model is observed to outperform the other models by a significant amount. Also, there are no signs of over fitting. The present study concludes that BiLSTM (2) serves as the best model for the chosen inverse problem.

Figure 6 depicts histogram plots of the MAPE of the best-trained neural network models. The plot clearly shows the efficacy of BiLSTM over its contemporaries.

Figure 7 represents the plots of the original HTC values and the ML-predicted results using FFNN (3), RNN (2), LSTM (2) and BiLSTM (2). The plots show impressive agreement for BiLSTM (2).

Table 3 The MAPEs and R2 scores for the different RNN models

Model	Training		Validation		Testing	
	MAPE	R2	MAPE	R2	MAPE	R2
RNN (1)	7.2368	0.8684	7.2382	0.8678	7.2422	0.8708
RNN (2)	5.8751	0.9538	5.8810	0.9511	5.8768	0.9540
RNN (3)	6.6190	0.9297	6.6232	0.9290	6.6155	0.9305

Table 4 The MAPEs and R2 scores for the different LSTM models

Model		Training		Validation		Testing	
		MAPE	R2	MAPE	R2	MAPE	R2
LSTM (1)		2.4285	0.9783	2.4312	0.9780	2.4293	0.9781
LSTM (2)		2.3431	0.9780	2.3457	0.9778	2.3425	0.9777
LSTM (3)		2.2860	0.9802	2.2847	0.9802	2.2860	0.9802

Table 5 The MAPEs and R2 scores for the different BiLSTM models

Model	Training		Validation		Testing	
	MAPE	R2	MAPE	R2	MAPE	R2
BiLSTM (1)	1.8492	0.9919	1.8537	0.9907	1.8527	0.9917
BiLSTM (2)	**1.2529**	**0.9948**	**1.2572**	**0.9946**	**1.2548**	**0.9948**
BiLSTM (3)	1.3584	0.9943	1.3642	0.9939	1.3642	0.9942

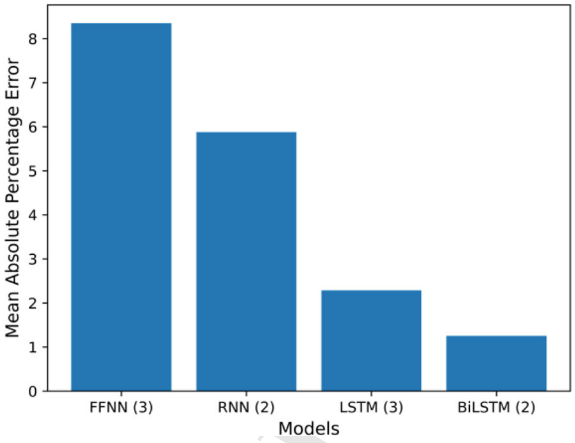

Fig. 6 Histogram depicting the MAPE (on the test dataset) of the best models in each category (FNN, RNN, LSTM, BiLSTM)

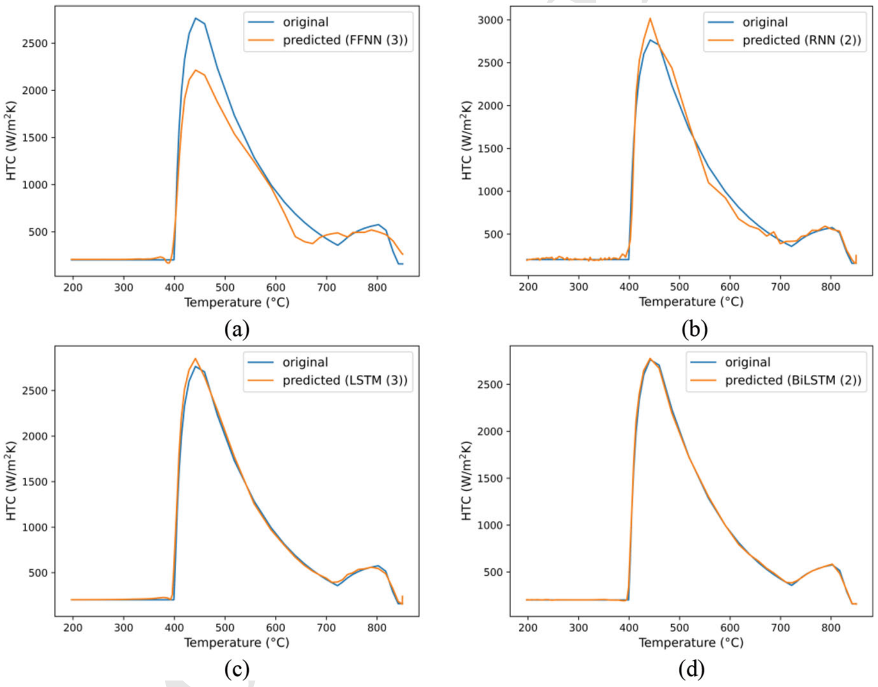

Fig. 7 Original and Predicted HTC vs Temperature plots for predictions made by **a** FFNN (3), **b** RNN (2), **c** LSTM (3), and **d** BiLSTM (2) models

The superior performance of recurrent models (RNN, LSTM, and BiLSTM), as compared to that of the simple FNN model, can be attributed to the recurrent models' capability to capture the temporal relationships within the data. The notably higher performance exhibited by the LSTM-based models, as compared to the simple RNN model, is due to their effective utilization of long and short-term memory mechanisms, enabling them to retain relevant information in long temporal data. The exceptional results demonstrated by the BiLSTM model can be attributed to their proficiency in extracting context from both past and future information in temporal data, which has made them the best model in this numerical investigation to model the complex relationship between the values of HTC with temporal temperature data.

5 Limitations and Future Work

1. ML models are specific to the dataset they are trained on. As a result, the proposed BiLSTM model in this study is specific for the particular dataset [22] used. To overcome this issue and improve the generalizability of the models, the models can be trained on different datasets containing data from using different types of workpieces and coolant fluids.
2. BiLSTM, used in this study, requires the input (temperature) and output (HTC) to be of the same dimension. As future work, encoder-decoder based neural networks may be explored which allow the input and output vectors to be of different dimensions. These methods become useful in scenarios where the interpolation functions are not readily known or estimated, or in industrial processes that inherently do not permit easy availability of input data (e.g., in metallurgical processes).
3. Models such as RNN, LSTM and BiLSTM suffer from poor parallelizability due to their sequential nature, i.e., the dependence of calculations, of a time step, on the results of the previous time step. This makes the models slow to train. This can be overcome by using non-recurrent neural networks with attention mechanism to account for the temporal dependencies in the data.
4. This study tries to model 1D heat transfer. As future work, more complex heat transfer scenarios can be explored.

6 Conclusion

In an effort to overcome the hurdles faced in using traditional numerical methods to solve ill-posed problems, this study presents a neural network-based approach to solve a representative IHTP. It outlines the capability of the neural networks, specifically the BiLSTM to map the complicated relationship between temperature (field variable) and HTC (thermal parameter) in case of one dimensional IHTP. A well-configured BiLSTM model can estimate the local values of HTC, based on the time-dependent temperature data series with sufficient accuracy. Based on comparative tests on the BiLSTM models, it proved sufficient to have three BiLSTM layers to obtain accurate results in this study. Future research scopes may include investigating more complex heat transfer scenarios, and using more advanced neural network architectures such as encoder-decoder and attention based neural networks.

References

1. Oksman P, Yu S, Kytönen H, Louhenkilpi S (2014) The effective thermal conductivity method in continuous casting of steel Acta Polytech Hungarica
2. Trębacz L, Miłkowska-Piszczek K, Konopka K, Falkus J (2014) Numerical simulation of the continuous casting of steel on a grid platform In: Bubak M, Kitowski J, Wiatr K (eds) eScience on Distributed Computing Infrastructure. Lecture Notes in Computer Science, vol 8500 Springer, Cham. https://doi.org/10.1007/978-3-319-10894-0_29
3. Pyszko R, Píhoda M, Fojtík P, Kova M (2012) Determination of heat flux layout in the mould for continuous casting of steel Metalurgija, 51(2), pp 149-152
4. Colaco Marcelo, Orlande Helcio, Dulikravich George (2006) Inverse and optimisation problems in heat transfer J Braz Soc Mech Sci Engineering XXVIII. https://doi.org/10.1590/S1678-58782006000100001
5. M N, Özisik H R B, Orlande Inverse heat transfer: Fundamentals and applications. taylor and francis, 2000
6. Raudenský M, Woodbury KA, Kral J, Brezina T (1995) Genetic algorithm in solution of inverse heat conduction problems. Numerical heat transfer, Part B Fundamentals 28(3):293–306
7. Vakili S, Gadala MS (2009) Effectiveness and efficiency of particle swarm optimization technique in inverse heat conduction analysis. Numerical heat transfer Part B-Fundamentals 56(2):119–141
8. Sun SC, Qi H, Ren YT, Yu XY, Ruan LM (2017) Improved social spider optimization algorithms for solving inverse radiation and coupled radiation-conduction heat transfer problems. Int Commun Heat Mass Transfer 87:132–146
9. Coello Coello C, Van Veldhuizen DA, Lamont GB (2013) Evolutionary algorithms for solving multi-objective problems
10. Anderson CW, Hittle DC, Katz AD, Kretchmar RM (1997) Synthesis of reinforcement learning, neural networks and PI control applied to a simulated heating coil. Artif Intell Eng 11(4):421–429
11. Sreekanth S, Ramaswamy HS, Sablani SS, Prasher SO (1999) A neural network approach for evaluation of surface heat transfer coefficient. J Food Process Preserv 23(4):329–348
12. Soeiro FJCP, Soares PO, Campos Velho, HF, Silva Neto AJ (2004) Using neural networks to obtain initial estimates for the solution of inverse heat transfer problems. In: Inverse Problems, Design an Optimization Symposium pp 358–363
13. Mirsephai A, Mohammadzaheri M, Chen L, O'Neill B (2012) An artificial intelligence approach to inverse heat transfer modeling of an irradiative dryer. Int Commun Heat Mass Transfer 39(1):40–45
14. Zálešák M, Klimeš L, Charvát P, Cabalka M, Kdela J, Mauder T (2023) Solution approaches to inverse heat transfer problems with and without phase changes: A state-of-the-art review. Energy, 127974
15. GC MP, Shettigar AK, Krishna P Parappagoudar M B (2017) Back propagation genetic and recurrent neural network applications in modelling and analysis of squeeze casting process. Appl Soft Comput 59:418–437
16. Zhang B, Wu G, Gu Y, Wang X, Wang F (2022) Multi-domain physics-informed neural network for solving forward and inverse problems of steady-state heat conduction in multilayer media. Physics of Fluids, 34(11)

17. Han J, Xu L, Cao K, Li T, Tan X, Tang Z, Liao G (2021) Online estimation of the heat flux during turning using long short-term memory-based encoder-decoder. Case Stud Therm Eng 26:101002
18. Zhu F, Chen J, Han Y, Ren D (2022) A deep learning method for estimating thermal boundary conditionparameters in transient inverse heat transfer problem. Int J Heat Mass Transf 194:123089
19. Sajjad U, Hussain I, Hamid K et al (2021) A deep learning method for estimating the boiling heat transfer coefficient of porous surfaces. J Therm Anal Calorim 145:1911–1923. https://doi.org/10.1007/s10973-021-10606-8
20. Cui Y, Zhong W, Zhou Z, Yu A, Liu X, Xiang J (2022) Coupled simulation and deep-learning prediction of combustion and heat transfer processes in supercritical CO2 CFB boiler. Adv Powder Technol 33(1):103361
21. Doner N, Ciddi K, Yalcin IB, Sarivaz M (2023) Artificial neural network models for heat transfer in the freeboard of a bubbling fluidised bed combustion system. Case Stud Therm Eng 49:103145
22. Szénási S, Felde I (2019) Database for research projects to solve the inverse heat conduction problem. data 4(3):90. https://doi.org/10.3390/data4030090
23. Bamberger M, Prinz B (1986) Determination of heat transfer coefficients during water cooling of metals. Mater Sci Technol 2(4):410–415
24. Ramírez-López A, Aguilar-López R, Palomar-Pardavé M, Romero-Romo MA, Muñoz Negrón D (2010) Simulation of heat transfer in steel billets during continuous casting. Int J Miner Metall Mater 17:403–416
25. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5:115–133
26. Aggarwal CC (2018) Neural networks and deep learning springer international publishing AG, part of Springer Nature
27. Manaswi NK (2018) Deep learning with applications using python. Apress, Berkeley, CA
28. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press. Chapter 6: Deep Feedforward Networks, p 164
29. Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211. ISSN 0364–0213. https://doi.org/10.1016/0364-0213(90)90002-E
30. Karpathy A (2015) The unreasonable effectiveness of recurrent neural networks. Retrieved from http://karpathy.github.io/2015/05/21/rnn-effectiveness/
31. Géron A (2019) Hands-on machine learning with scikit-learn, keras, and tensorflow: concepts, tools, and techniques to build intelligent systems O'reilly media
32. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
33. Schuster M, Paliwal K (1997) Bidirectional recurrent neural networks. Signal Process, IEEE Trans On 45:2673–2681. https://doi.org/10.1109/78.650093
34. Graves A (2005) Supervised sequence labelling with recurrent neural networks Ph.D. thesis, Technical University of Munich.
35. Masters D, Luschi C (2018) Revisiting small batch training for deep neural networks arXiv preprint arXiv:1804.07612

AQ1

Author Queries

Chapter 9

Query Refs.	Details Required	Author's response
AQ1	Please check and confirm if the inserted citation of Figs. 1 and 4 are correct. If not, please suggest an alternate citation. Please note that figures and tables should be cited sequentially in the text.	