SOLID TEMPERATURE PREDICTION FOR CLT WALLS IN FIRE USING SUPERVISED MACHINE LEARNING

Arwa Abougharib[1], M.Z. Naser[2], Gregory Paradis[3], Felix Wiesner[4]

ABSTRACT: Machine learning (ML) tools have proven valuable and efficient in predicting fire behaviour of structural elements, including timber elements, under standard fire conditions. However, available ML analysis for structural timber elements under non-standard fire exposure is limited. The viability of using ML models to trace in-solid temperatures over time and depth in structural timber elements was explored using a dataset of axially loaded Cross-Laminated Timber (CLT) walls. Three supervised ML approaches were compared: Long Short-Term Memory (LSTM) Recurrent Neural Network, Gradient Boosting (GB), and Symbolic Regression (SR). The latter was found superior for solid-temperature prediction and lent support to opting for simpler and more interpretable models over more complex models. Our results also highlight that this problem should be framed as a traditional regression problem rather than a time-series forecast. These findings provide valuable guidance for further ML tool development for larger timber fire datasets that capture a wider range of fires.

KEYWORDS: cross-laminated timber, temperature modelling, explainable machine learning, scientific machine learning, time series forecasting, symbolic regression.

1 - INTRODUCTION

Wood is a renewable resource with a competitive strength-to-weight ratio relative to reinforced concrete. Specifically, engineered timber has the potential to decarbonise construction. Even though mass timber can achieve comparable fire resistance ratings as steel or concrete in standard fire conditions, its structural fire performance may vary based on timber species, moisture content, chemical treatment, encapsulation, adhesive, layup, applied mechanical stresses, and geometry. Moreover, compartments with exposed timber experience more intense and prolonged fires than those with non-combustible linings due to radiation feedback [1], [2], and structural failure may occur even after the end of flaming combustionas thermal waves continue to propagate within the timber members [3].Hence, modelling the temperature evolution within structural timber both during and after a fire is essential to accurately estimate its structural response.

While Finite Element Models (FEM) have been developed to approximate the behaviour of timber in compartment fires, their use is constrained by long computation times and they were derived from standard fire test results; thus their validation range is limited and they cannot be adapted for different fires, for example those that exhibit a cooling phase [4]. In contrast, Machine Learning (ML) models leverage currently available data without requiring intermediate physics to be modelled explicitly and, once trained, can generate predictions almost instantly, allowing practicing fire safety engineers to assess a wide range of fire scenarios efficiently [5]. Therefore, the opportunities for applying ML techniques to model the structural fire performance of timber are worth exploring. Specifically, we are interested in modelling solid timber temperature as it is a key determinant of strength and stiffness.

In this work, three ML approaches were applied to a dataset of 24 axially loaded Cross-Laminated Timber

[1] Arwa Abougharib, Department of Wood Science, Faculty of Forestry, University of British Columbia, Vancouver, Canada. e-mail: arwaabg@student.ubc.ca

[2] M.Z. Naser, Glenn Department of Civil Engineering, Clemson University, Clemson, SC, USA. e-mail: mznaser@clemson.edu

[3] Gregory Paradis, Department of Forest Resources Management, Faculty of Forestry, University of British Columbia, Vancouver, Canada. e-mail: gregory.paradis@ubc.ca

[4] Felix Wiesner, Department of Wood Science, Faculty of Forestry, University of British Columbia, Vancouver, Canada. e-mail: felix.wiesner@ubc.ca

https://doi.org/10.52202/080513-0030

(CLT) wall strips exposed to one-sided radiative heat exposure at their mid-height. The raw dataset from [6] recorded seven variables for each test: time, adhesive type, number of layers, heat flux, thickness, thermocouple depth, and in-depth temperature.

The objective is to investigate whether ML, specifically scientific ML (SciML), can reliably predict the temperature evolution within the CLT wall strips, given the geometry, layup, and applied heat flux, and how SciML compares in performance to traditional ML.

SciML refers to a class of ML methodologies that caters to scientists, engineers, and policymakers in high-stakes applications that require trustworthy models. These models must be explainable at the very least [7], and ideally, they should integrate domain knowledge, cause-and-effect relationships [8], and/or the physical laws governing a problem [9]. Black-box models are generally to be avoided if a SciML model with similar performance can be found [10] and, if used, the black-box should be accompanied by a post-hoc explainability method. Non-SciML models remain purely correlational or data-driven and are not designed for extrapolation as they are unaware of physical or causal constraints in the problem and may even leverage spurious and unphysical correlations within the data to minimize a chosen error metric. For scientific contexts, if domain knowledge cannot be incorporated into a data-driven model, at the very least, it should be explainable so that expert users can interpret the model's logic and uncover its hidden biases or problematic correlations. Thus, explainable ML can be considered a subclass of SciML and the first step towards curating ML for scientific discovery.

In this study, three ML models were investigated: (1) a Long Short-Term Memory Recurrent Neural Network (LSTM RNN, or simply 'LSTM') to represent a black-box neural network which does not require feature engineering, (2) Gradient-Boosted (GB) tree methods, including three GB algorithms (XGBoost, CatBoost, and Light GB Machine), which are more interpretable than LSTM as they involve explicit feature engineering, and (3) Symbolic Regression (SR), which is an inherently interpretable ML method as it results in an explicit functional expression of the target in terms of the features.

2 - METHODOLOGY

2.1. DATASET DESCRIPTION

The dataset was collected from experiments on simply supported CLT wall strips exposed to one-sided radiative heat exposure at their mid-height on a central 300 by 300 mm^2 area while under sustained compressive load. All samples had a thickness of approximately 100 mm, a width of 300 mm, and a height of 1700 mm.

Four CLT configurations and three heating regimes resulted in 12 experiments with two repeats for each configuration. The three heating regimes were: (1) 50 kW/m^2 to failure, (2) 50 kW/m^2 for either 15 or 25 min, followed by a cooling phase, and (3) 15 kW/m^2 to failure. In all cases, the applied load was kept constant. In the first two heating regimes, a heat flux of 50 kW/m^2 was applied to cause ignition and sustain burning, while 15 kW/m^2 was used as an exposure below the critical heat flux for auto-ignition. The four CLT wall configurations were distinguished by two types of adhesives, Polyurethane (PU) and Melamine Formaldehyde (MF), and two layups: three-ply and five-ply. Each record in the dataset is a temperature reading at a given instant and depth from the heated surface. The temperatures were measured using 1.5 mm diameter Inconel sheathed, Type K thermocouples inserted from the unexposed side of the wall, i.e., parallel to the direction of heat transfer. Table 1 describes the features in the original dataset. In addition, the original dataset contained a feature for the load applied. This was dropped as it should not be relevant to temperature prediction and to prevent an ML model from leveraging any spurious correlations between the load and the temperature.

Table 1: Description of variables in the original temperature dataset

Variable	Description	Input / Target
Signal ID	Test number and thermocouple ID	Unique identifier
Layers	Number of plies (3 or 5)	Input
Flux	Applied radiative heat flux (kW/m^2)	Input
Adhesive	Binary variable that indicates which of two types of adhesives was used: 0 for MF and 1 for PU.	Input
Thickness	Initial wall thickness (mm)	Input
Heating duration	Duration for which the CLT wall was exposed to the heating panel (seconds)	Input
Depth	Depth of the thermocouple from the exposed side (mm)	Input
Time	Time elapsed from the start of heating (seconds)	Input
Temperature	Measured temperature (°C)	Target

2.2. DATA PRE-PROCESSING

Data Cleaning

Firstly, signal streams from faulty thermocouples were excluded, and signal smoothing was achieved using a Savitzky-Golay filter. Next, the temperatures were capped at 600 °C, making the conservative assumption that char will have completed oxidation (i.e., conversion to ash) by this temperature [11]; higher temperatures recorded would have been due to surface regression and thus gas phase temperatures. Secondly, series from samples heated to failure were truncated at the time the sample failed, whereas time series from experiments with a cooling phase were truncated after five minutes of reaching a steady state, which was determined as zero first-order and zero second-order time differentials. Finally, all signals were downsampled from seconds to minutes; data loss in this case was prevented with prior signal smoothing.

The next step in data pre-processing was splitting the cleaned dataset into either two subsets (training and testing) or three subsets for training, validation, and testing.

A key consideration was the time-series nature of the problem. In a traditional regression problem, it is assumed that the observations are “i.i.d.”: *independent* (each data point does not depend on any other data point) and *identically distributed* (all come from the same probability distribution). However, this is not always true for every dataset. Data points may be interconnected, e.g., spatial data, and their distributions may be time-dependent, e.g., time-series data. In time-series data, the points are neither independent (the next data point depends on the past data points) nor come from the same distribution (the mean and variance change over time). Thus, this regression problem qualifies as a Time-Series Forecast (TSF), as the temperature readings have a temporal order and sequential dependency, and the information value of any forecasted temperature is lost if its associated time stamp is not provided. Dealing with time-series data has several consequences relevant to the data-splitting step:

- *No random sampling*: When the data set is split for training, validation, and testing, the split must respect the chronological order. Otherwise, data leakage between the testing and training sets, also known as look-ahead bias, will lead to non-generalizable, over-fitted models. Past data points should be used for training and future data points for validation or testing.
- *Feature engineering*: since a dataset with sequential dependency exhibits *auto*-correlation (the target variable depends on its *own* past values besides other variables), additional features are often added to the feature set, which are the lags, differences, or other transformations on past values of the target variable, e.g., using Y_{t-1} and Y_{t-2} and moving averages as features to predict Y_t at each time step t.
- *Static vs. Dynamic exogenous features*: Attention must be given to exogenous features (i.e., predictors other than the transformations on past values of the target, e.g., layers, adhesive, flux) that are also time-dependent. In this problem, all the exogenous features are static except for the applied flux, as it is stepped rather than constant for the samples with a cooling phase. Thus, the applied flux is considered a dynamic feature, and a schedule of the applied flux must be provided to a TSF algorithm to obtain a temperature forecast. Other static values, such as the adhesive, number of layers, and initial thickness, are static features; only a single value is needed for each static input in the prediction step.

Generally, two-way splits of the original dataset into training and validation are common in most ML studies. The training set is used to tune model parameters, i.e., calibrate it, and the validation set would be used to evaluate it. However, if the validation set is used to further refine the model, e.g., as with k-fold cross-validation, or to tune model hyperparameters, then the validation set no longer represents ‘unseen’ data and using a third set, called a testing set, is recommended for an objective evaluation of the model.

Global Splitting Strategies

This dataset contains 12 unique experimental configurations, each replicated once. In each experiment, 16 thermocouples were drilled into the CLT wall sample at different depths. The temperature recorded over time from each thermocouple is considered a time series. Hence, there are $12 \times 2 \times 16 = 384$ time series. Due to the deletion of signals from faulty thermocouples, this number was reduced to 373. Another decision must be made regarding how a global splitting strategy might apply to all the series. There are two options:

- *Split by series*. For example, allocate 268 entire series for training (70% of 373) and 116 entire series for validation (30% of 373).
- *Split within series*. Each of the 373 series will be split independently using any of the local splitting strategies discussed next.

Local Splitting Strategies

For time-series data, there are a number of strategies that respect the temporal order. The most basic is the chronological split: *a single train-validate split*, using a given time as a cut-off point; all measurements recorded before it would be in the training set, and all points occurring after it would be in the validation/testing set. The downside of this simple strategy is that important phenomena that tend to occur later in the series, e.g., smouldering, reignitions, may not make it into the training set and, therefore, not captured by the model unless the training window is made long enough to capture all desired features. A long training window would result in a short validation window, which is problematic as the validation windows would almost always be in the decay phase where observations were very similar, leading to overly optimistic validation scores.

To address the shortcomings of the single train-validate split, other splitting strategies employ *multiple train-validate splits*, often used in conjunction with iterative training/validation methods. These include the rolling window (fixed-size) split, the expanding window split, and the stratified time-series split.

Herein, we divided the dataset into training and validation sets only; a third testing set was deemed unnecessary as the validation set was not used for hyperparameter tuning.

The two global splitting strategies (within series vs by series) will be examined for each of the three candidate algorithms. When the by-series split was implemented, the allocation of each series to either the training or validation set was conducted using stratification. Within each set, the three heating scenarios—heated to failure, heated for 15 minutes, and heated for 25 minutes—were represented in the same proportions as they appear in the original dataset. As for the within-series split, a chronological split was applied to each series, with the first 70% of data points in a series being assigned to the training set and the trailing 30% to the validation set. Windowing techniques would have complicated the analysis by adding another hyper-parameter that requires tuning: the window size.

Performance Metrics for Time-Series

Defining the forecast error at time t as $e_t = Y_t - F_t$, where Y_t is the actual value at time t and F_t is the corresponding prediction, Hyndman classified forecast-error metrics into four groups [12]:

1. *Scale-dependent metrics*, e.g., Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), which compute some aggregate (e.g., mean, median, geometric mean) of the squared or absolute forecast errors to prevent positive and negative errors offsetting each other. As these metrics are not unit-free, they must not be used to compare series that involve different units.
2. *Percentage error metrics*, e.g., Mean Absolute Percentage Error (MAPE), which aggregate the values of $\frac{100\, e_t}{Y_t}$, and are scale-free, so they can be used for comparison across series with different units. Their disadvantage is that they are unstable when Y_t approaches zero or is missing.
3. *Relative error metrics*, which aggregate the relative errors defined as $r_t = \frac{e_t}{e_t^*}$ where e_t^* is the forecast error of a benchmark method, usually the naïve method. The naïve method (also called random walk) is a benchmark forecasting method in which the next observation is the same as the most recent observation, i.e., $F_t^* \coloneqq Y_{t-1}$. This may also lead to division by zero in an individual period when the naïve forecast errors are small or zero, as in the case of intermittent series. An example is the Median Relative Absolute Error.
4. *Scaled error metrics* account for temporal dependence by comparing predictions to a baseline forecast. The forecast error e_t for each period is normalized by the aggregate of squared or absolute forecast errors generated by a baseline method over all periods.
   Unlike percentage-error or relative-error metrics, the denominator here is unlikely to be zero, even in intermittent series with some missing data, as it is calculated on the entirety of the training set. The baseline method is typically the naïve method or can be a more sophisticated method, such as Autoregressive Integrated Moving Average (ARIMA). Examples of scaled error metrics include the Median Absolute Scaled Error (MdASE) and the Mean Square Scaled Error.

Besides scaled-error metrics, there are other metrics in the realm of forecasting that are more ‘aware’ of temporal dependence, such as Dynamic Time Warping Distance, which measures the alignment of two series and is useful when time delays are important, and Mean Directional Accuracy which measures the accuracy of the forecasting method in predicting the correct direction of change. Eventually, more than one metric may be needed, and the ideal set of metrics depends on the application and the forecast goals.

For this problem, we desire an interpretable error metric with the same units as the data (°C), e.g., MAE or RMSE (as it is difficult to interpret an error in squared °C), or as a percentage (e.g., MAPE). This makes absolute error metrics attractive, but they do not serve as good loss functions in ML methods relying on gradient descent for optimization, since the absolute function is not differentiable everywhere, unlike the square and square root functions. Hence, the RMSE provides a good balance between training speed and interpretability.

Moreover, relative error metrics should be precluded, as they would be skewed or undefined in the decay phase of the temperature-time curve, where the temperatures become almost stable, introducing a risk of near-zero denominators.

Since all the time series have the same units, using scale-dependent metrics would be acceptable. Additionally, as fire is a high-temperature scenario, it is highly unlikely that actual values would approach zero, making percentage-error metrics also a valid choice.

Scaled-error metrics, though, have more informational value than the other three classes of metrics; a sophisticated forecasting method that cannot beat the naïve prediction should be abandoned. For these reasons, we shall use the RMSE and the MdASE for training and algorithm selection, respectively.

For all scaled-error metrics, including MdASE, if the metric is less than 1, then the model performs better than the baseline model (here, the naïve model), and lower values are better, while a metric greater than 1 means the model performs worse than the baseline model. A value of 0 means the forecast exactly matches the actual values.

In contrast to the textbook formula of the baseline forecast error e_i^* which is often provided as $Y_i - F_i^* = Y_i - Y_{i-1}$ [12], we calculated the period-wise forecast error as:

$$e_i^* = Y_i - Y_0 \tag{1}$$

This deviation is due to setting F_i^* to a static value, namely, the initial value, because we are performing single-shot multi-period forecasting, in which the candidate model at the time of prediction does not and will not have access to the previous value before it makes a prediction about the next value. Here, the candidate and naive models only know the initial value and cannot use previous actual values, which is representative of the reality of structural fire modelling. The formula $Y_i - Y_{i-1}$ was built with a business and finance context in mind, assuming that the periods are in days, weeks, or months and that model users would have access to recent sales figures, for example, and be able to re-run the forecasting model utilizing the last period's actual figure to improve the forecast for the next period. This is obviously not the case in fire modelling for design, and the MdASE formula was adjusted accordingly to provide a fairer metric.

2.3. Candidate algorithms

TSF can be implemented through several paradigms [13], including classical forecasting methods (e.g., ARIMA, Prophet, and exponential smoothing), ML methods (e.g., Gradient Boosting (GB), Recurrent Neural Networks, equation discovery), and stochastic processes (e.g., Gaussian, Poisson, and Hawkes processes). Some forecasting methods only support autocorrelation, where only past values of the target variable and transformations thereof are used for inference. Not all of them can support exogenous features, i.e., interactions between the target time series and other predictor/external/exogenous variables or non-linear relationships between them–such models should be excluded from consideration. All analyses presented in this work were performed in Python.

Time-Series Forecasting with Exogenous Features

LSTM and GB are popular ML methods for TSF [14]. LSTM networks can handle non-stationary time series (i.e., the mean and variance change over time), exogenous features, and non-linear relationships and do not require explicit feature engineering. GB methods also offer similar capabilities and are faster to train compared to LSTM but require explicit feature engineering to make the series stationary. Three GB algorithms were tested in this study: XGBoost, CatBoost, and LGBM. All three GB models were implemented using Nixtla's MLForecast library [15], which offers user-friendly functions for time-series pre-processing. Stationarity is a requirement for the GB models, which was achieved through second-order differencing [15]. Stationarity was then confirmed using the Augmented Dickey-Fuller and Kwiatkowski-Phillips-Schmidt-Shin tests. These two tests, commonly referred to as 'unit root' tests, examine whether a time-series variable is stationary. Two unit root tests were applied as different tests may yield contradictory results. For the GB analysis, as it requires additional feature engineering, a new feature (cumulative incident heat flux) was added to the dynamic exogenous variables as a proxy for the radiant energy absorbed.

Symbolic Regression for Equation Discovery

SR is an interpretable representation learning ML method that searches the space of all possible mathematical formulas, including transcendental functions (e.g., exponentials, logarithms, and trigonometric functions), to express the target variable as a function of the input features. As such, SR results in a human-readable mathematical formula of the output as a function of the inputs, e.g., $y = a \cdot x^2 + b \cdot sin\,(x) + c$, rather than a stream of values to express the output. It can be considered a generalisation of the linear regression method, except that the form of the function is unconstrained [16], [17].

SR analysis was conducted in this study using the Feyn package, which is an interface to the QLattice software library [18]. Feyn was chosen as it is the more user-friendly of two popular SR packages (the other being PySR [19]). Additional features were engineered, in addition to the original features in Table 1, such as relative depth (depth ÷ initial thickness), relative time (current time - heating duration), and the initial temperature of the solid.

3 - RESULTS

3.1. Model Selection

Each of the models considered in this study was run on the pre-processed dataset for each of the two split strategies explained in §2.2. The RMSE values reported in Table 2 are before any hyper-parameter tuning.

Table 2 Training and validation RMSE (in °C) by type of data split for all ML algorithms considered (all models are untuned)

		Training	Validation
LSTM	***By series***	27310	28541
	Within series	23227	21629
CatBoost	***By series***	N/A	N/A
	Within series	9.64	545.89
LGBM	***By series***	N/A	N/A
	Within series	8.79	632.04
XGBoost	***By series***	N/A	N/A
	Within series	10.4	434.27
SR	***By series***	60.22	61.66
	Within series	57.83	102.82

The LSTM model's RMSE values are unacceptable, and this model was discarded. The model's predictions deviated from the actual range of temperatures by approximately 1.65 orders of magnitude, making the model useless for this application. In addition, the three GB algorithms could not be run for the by-series split, as the MLForecast framework prohibits generating predictions for a series that did not appear in the training set. Although the training RMSE values for the three GB models are less than 10 °C, the validation errors ranged from 430 up to 630 °C. This large discrepancy indicates overfitting.

Compared to the others, the SR algorithm yielded much more acceptable RMSE values in validation, performing especially well with the by-series split, for which the training and validation RMSE values are not only relatively low but also nearly equal, indicating that the model has neither over-fit nor under-fit the dataset. Thus, the SR algorithm and the by-series split were chosen for further tuning.

One of the most critical hyper-parameters in SR is complexity, as it relates directly to human interpretability. In the QLattice library, *maximum complexity* is the maximum allowable number of arcs in a graph model of the AI-generated expression, such as in Figure 1 [20] and, therefore, its inverse is a proxy for the degree of parsimony. A larger maximum complexity value set by the user enables QLattice to discover longer and more complex expressions corresponding to a greater number of feature interactions. Since the temperature profiles for samples with and without a cooling phase have different shapes, it was challenging to find a single expression that could capture the two heating scenarios while also remaining relatively parsimonious. Therefore, the SR algorithm was run independently for the two heating scenarios to generate relatively more human interpretable expressions without degrading performance. The performance of the independent expressions was evaluated and compared to the all-cases SR expression, as shown in Table 3.

Table 3 SR performance metrics for the 70-30 split by series, considering the heating scenarios collectively and separately

		All Cases	With Cooling	Without Cooling
RMSE (°C)	***Training***	60.22	42.55	46.17
	Validation	61.66	49.14	52.59
MAPE	***Training***	0.730	0.5164	0.379
	Validation	0.724	0.5336	0.383
Overall MdASE	***Training***	2	0.43	1.19
	Validation	2.09	0.66	1.41

3.2. Symbolic Regression Expressions

Heating for a fixed period with a cooling phase

The SR-generated expression of Temperature T (°C) as a function of time t (minutes) and depth d (mm) corresponding to a maximum complexity of 10 for samples with a cooling phase is provided in Eq. (2). The corresponding graph model is given in Figure 1.

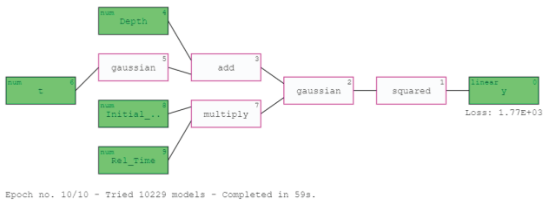

Figure 1. Graph model for the SR expression generated for samples with a cooling phase

$$T(t, d, t_{stop}, T_0) = \alpha e^{-(\beta+\gamma)} + \delta \qquad (2)$$

Where, $\alpha = 499.0$,

$\beta = 1.09\left(1 - 0.241(t - t_{stop})\right)^2 (0.101\, T_0 - 1)^2$

$\gamma = 4.0\left(-0.0344\, d - 0.361 + e^{-1.43(0.022\, t - 1)^2}\right)^2$

$\delta = 51.0$.

t_{stop} is the heating duration (minutes), and T_0 is the initial temperature of the solid.

Continuous heating at a constant flux

Similarly, the SR-generated expression for the case of continuous heating at constant flux to failure is given in (3), while the corresponding graph model is given in Figure 2.

$$T(t, d, L, F, T_0) = \alpha - \beta e^{\gamma - \delta e^{\varepsilon}} \qquad (3)$$

Where, $\alpha = 540.0$, $\beta = 532.0$, $\gamma = -0.035\left(\frac{d}{L} + 0.189\right)^2$, $\delta = 0.00205(-0.131\, T_0 - 0.641t - 1)^2$

$\varepsilon = 0.0525\, F - 10.4\, \frac{d}{L}$.

L is the initial thickness of the wall (mm) and F is the applied flux (kW/m^2).

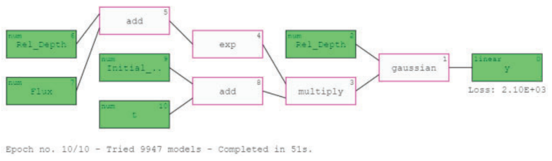

Figure 2 Graph model for the SR expression generated for samples heated to failure

4 - DISCUSSION

Plots of the predicted temperature profiles at various depths, superimposed on the actual measurements, are given in Figure 3 and Figure 4 for the two SR expressions applied to their respective heating cases. where signal streams were grouped into each plot using experimental treatment. The plot titles are coded in the format **L[a]_[b]_F[c]_H[d]**, where [a] is the number of layers, [b] is the adhesive, [c] is the flux, and [d] is the heating scenario: HF indicates heating to failure, H900 heating for 900 seconds (15 min), and H1500 heating for 1500 seconds (25 min).

For the heating-to-failure case, the predictions generally agree with the actual measurements. For experiments with a cooling phase, however, the SR model seems capable of capturing the overall shape of the graphs but does not always respect the initial temperature; the SR algorithm's objective, after all, is to optimise the overall error metric (here, this is the MSE) for all points within the given complexity constraint. Although the allowed complexity could be increased in the hope of achieving more complex models that better capture the initial growth phase, we notice that Equations (2) and (3) already contain nested exponentials, which may not be parsimonious, and increasing the complexity further would have resulted in less interpretable functional expressions.

From Table 2, we notice that for SR, a global split by time series greatly improved the validation RMSE compared to a within-series split. This may be due to the former strategy allowing the model to learn the decay phase as it is allowed to train on entire series from start to finish. In contrast, a within-series split would not allow the model to fully train on the decay phase of any series.

The error metrics in Table 3 showed significant improvement when the heating scenarios were considered separately compared to being lumped under one model. Looking at the case-specific models, the RMSE values imply that deviations 49 - 52 °C from the actual temperature are expected, corresponding to a percentage error of 38-53%.

In terms of the MdASE, the only model that could exceed the naïve forecast was the cooling case model, while the others had overall MdASE values greater than 1 when averaged over all the series. This does not necessarily mean that these models cannot provide insights. For example, given the maximum complexity constraint, none of the case-specific models included the adhesive or the number of layers in the functional expression, which tells us that these are likely insignificant predictors of temperature, as would be expected.

Interestingly, the cooling-case expression was not a function of the magnitude of applied flux. One reason may be that the heat flux was the same (0 kW/m²) for all cooling phases. Another reason may be that only samples heated with 50 kW/m² actually exhibited a cooling phase; thus, there was no variation in heat flux for the algorithm to learn from in the case of heating followed by cooling.

5 - CONCLUSION

ML can be a very useful modelling tool for scientific and engineering simulations, and interpretable ML can exceed the performance of black-box models such as LSTM and GB. This study applied three ML models to the temperature readings collected from experiments with loaded CLT wall strips exposed to one-sided heat radiation with varying experimental treatments. Symbolic Regression, an interpretable ML model, significantly outperformed a neural network, LSTM, and three gradient-boosting techniques for in-solid time-temperature prediction. This is advantageous since we desire easily interpretable models in scientific applications where users are interested in inspecting the model's logic and the reasoning behind its predictions. Moreover, more complex models would not be justified if a simple, interpretable model that performs well can be found [10]. The functional expressions generated by SR were reported, and performance was further improved when separate expressions were generated for continuous heating versus heating for a fixed period followed by a cooling phase.

However, when applied in the context of predicting temperature or other material states over time, special care must be taken in data preparation to respect the temporal order and prevent data leakage, as well as in selecting meaningful and robust performance metrics. Modellers should be aware of deployment considerations; when their model goes out to the real world, would users make one-shot multi-period predictions based on only the initial value, or would the model have access to actual values of the previous periods before making the next prediction? It is often the former for most engineering simulations, and ML modelling choices should align with that understanding. This can be challenging given that most textbook sources and out-of-the-box implementations on forecasting were made for a business context. Still, adjustments are necessary if TSF is to be applied to engineering simulation, and it is hoped that this work provides pointers to best practices for readers interested in TSF for engineering simulations.

The dataset and source code are available upon request.

6 - FUTURE WORK

This study aimed to verify whether SciML can model the solid temperatures developed in timber members in simple fire exposures where the exposure is known and controlled. Our future work is to investigate further whether ML can also substitute for both a one-zone fire model and FEA to model structural thermal response for uncontrolled and unknown exposure, e.g., a real compartment fire where the exposure is not provided to the model as an input (applied flux on the walls is not initially known), and the exposure is not simple either (neither constant nor stepped like in this study). Other considerations include correcting the measured temperatures for orientation effects [21] and, with regards to scaled error metrics, possibly substituting the naive model with a better and more relevant benchmark, such as the 1-D FEA results on a non-combustible solid with the same geometry, exposure, and initial properties of timber.

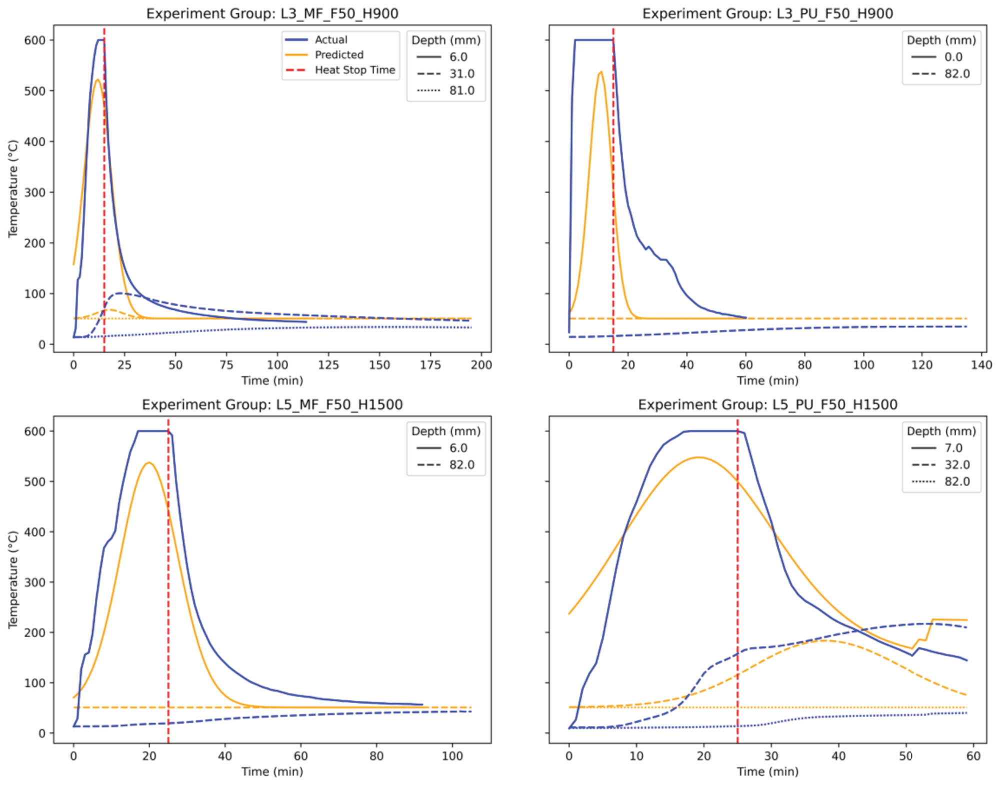

Figure 3. Actual vs. predicted temperature profiles for samples heated for a fixed period with a cooling phase

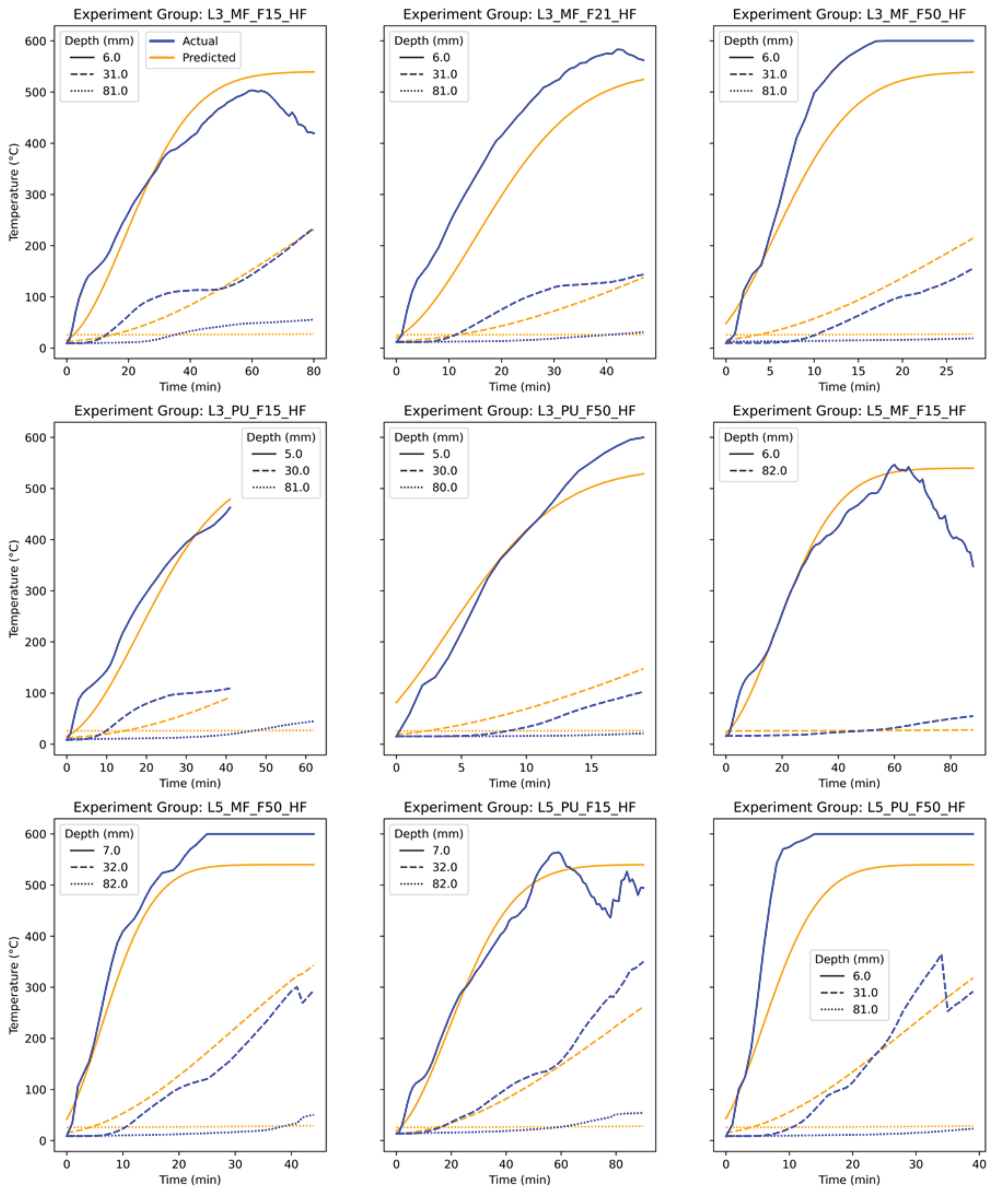

Figure 4. Actual vs. predicted temperature profiles for samples heated to failure

7 - REFERENCES

[1] P. Kotsovinos *et al.*, "Fire dynamics inside a large and open-plan compartment with exposed timber ceiling and columns: CodeRed #01," *Fire Mater.*, vol. 47, no. 4, pp. 542–568, 2023, doi: 10.1002/fam.3049.

[2] F. Wiesner *et al.*, "Large-scale compartment fires to develop a self-extinction design framework for mass

timber-Part 2: Results, analysis and design implications," *Fire Saf. J.*, p. 104346, Jan. 2025, doi: 10.1016/j.firesaf.2025.104346.

[3] T. Gernay *et al.*, "Experimental investigation of structural failure during the cooling phase of a fire: Timber columns," *Fire Mater.*, vol. 47, no. 4, pp. 445–460, Jun. 2023, doi: 10.1002/fam.3110.

[4] E. Philion, B. Chorlton, J. Gales, and P. Kotsovinos, "Structural Fire Modeling Strategies for Exposed Mass Timber Compartments and Experimental Gaps for Model Validation," *J. Perform. Constr. Facil.*, vol. 36, no. 6, p. 04022049, Dec. 2022, doi: 10.1061/(ASCE)CF.1943-5509.0001761.

[5] M. Z. Naser, "Mechanistically Informed Machine Learning and Artificial Intelligence in Fire Engineering and Sciences," *Fire Technol.*, vol. 57, no. 6, pp. 2741–2784, Nov. 2021, doi: 10.1007/s10694-020-01069-8.

[6] F. Wiesner, R. Hadden, S. Deeny, and L. Bisby, "Structural fire engineering considerations for cross-laminated timber walls," *Constr. Build. Mater.*, vol. 323, p. 126605, Mar. 2022, doi: 10.1016/j.conbuildmat.2022.126605.

[7] M. Z. Naser, "An engineer's guide to eXplainable Artificial Intelligence and Interpretable Machine Learning: Navigating causality, forced goodness, and the false perception of inference," *Autom. Constr.*, vol. 129, p. 103821, 2021.

[8] M. Z. Naser and A. T. G. Tapeh, "Causality, causal discovery, causal inference and counterfactuals in Civil Engineering: Causal machine learning and case studies for knowledge discovery," *Comput. Concr.*, vol. 31, no. 4, pp. 277–292, 2023.

[9] P. Y. Lu, "Interpretable Physics-informed Machine Learning Methods for Scientific Modeling and Data Analysis," Thesis, Massachusetts Institute of Technology, 2022. Accessed: Dec. 05, 2023. [Online]. Available: https://dspace.mit.edu/handle/1721.1/150769

[10] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, May 2019, doi: 10.1038/s42256-019-0048-x.

[11] L. Schmidt and R. M. Hadden, "Conditions for the onset of char oxidation in timber," *J. Phys. Conf. Ser.*, vol. 2885, no. 1, p. 012010, Nov. 2024, doi: 10.1088/1742-6596/2885/1/012010.

[12] R. J. Hyndman, "Another look at measures of forecast accuracy for intermittent demand," 2006.

[13] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward," *PLoS ONE*, vol. 13, no. 3, p. e0194889, Mar. 2018, doi: 10.1371/journal.pone.0194889.

[14] A. Frifra, M. Maanan, M. Maanan, and H. Rhinane, "Harnessing LSTM and XGBoost algorithms for storm prediction," *Sci. Rep.*, vol. 14, no. 1, p. 11381, May 2024, doi: 10.1038/s41598-024-62182-0.

[15] *mlforecast: Scalable machine learning based time series forecasting*. Python. Accessed: Feb. 17, 2025. [Online]. Available: https://github.com/Nixtla/mlforecast

[16] R. Ruggiero, "Symbolic Regression: The Forgotten Machine Learning Method," Medium. Accessed: Jan. 15, 2024. [Online]. Available: https://towardsdatascience.com/symbolic-regression-the-forgotten-machine-learning-method-ac50365a7d95

[17] C. S. Wilstrup, "Symbolic Regression: a Simple and Friendly Introduction," Medium. Accessed: Jan. 15, 2024. [Online]. Available: https://medium.com/@wilstrup/symbolic-regression-a-simple-and-friendly-introduction-16bcadbe870a

[18] Abzu, *feyn: Feyn is a symbolic regression package named after Richard Feynman, that uses the QLattice as a simulator to generate models.* C, Python. Accessed: Jan. 15, 2024. [MacOS, POSIX :: Linux]. Available: https://abzu.ai

[19] M. Cranmer, *MilesCranmer/PySR*. (Jan. 14, 2024). Python. Accessed: Jan. 15, 2024. [Online]. Available: https://github.com/MilesCranmer/PySR

[20] "Model complexity · Feyn Documentation." Accessed: Feb. 17, 2025. [Online]. Available: https://docs.abzu.ai/index.html

[21] A. Béreyziat, M. Audebert, S. Durif, and A. Bouchair, "Temperature Measurements in Timber Exposed to Fire Using Thermocouples," in *Wood & Fire Safety 2024*, L. Makovická Osvaldová, L. E. Hasburgh, and O. Das, Eds., Cham: Springer Nature Switzerland, 2024, pp. 341–348. doi: 10.1007/978-3-031-59177-8_40.