Transformer Oil Temperature Prediction

IT1244 Report

Jaisev Singh Sachdev (A0336866M), Muhammad Ilham Alfarisi (A0305879M), Nabil Rakaiza Abror (A0305715J), Teo Ting Wei (A0272933Y)

National University of Singapore

Introduction

The task involves predicting the hourly oil temperature (OT) of electric transformers based on historical load data, without using oil temperature information from any previous time point. The dataset consists of two transformers and contains six load features recorded at each time point. The objective is to forecast transformer oil temperature across three different time horizons. These forecasts include 1-hour, 1-day, and 1-week ahead predictions using past N time points of data.

Accurately predicting transformer oil temperature is critical for the safe and efficient operation of electrical grids. Overheating can lead to equipment failure and costly maintenance. Traditional monitoring methods rely heavily on manual inspection and reactive maintenance, which can be time-consuming and prone to human error. By developing predictive models using historical load data, operators can anticipate potential overheating events and plan proactive maintenance, improving reliability, reducing energy waste, and enhancing operational safety. Additionally, the project provides an opportunity to compare machine learning and deep learning approaches for time series forecasting, gaining insights into their effectiveness for industrial applications.

AI/ML Techniques Overview

To address this forecasting problem, we will explore and compare traditional machine learning and modern deep learning algorithms. The traditional machine learning will involve using established algorithms that are powerful at learning simple or complex patterns from tabular feature sets. In parallel, we will implement deep learning models that are adept at capturing temporal dependencies and patterns over time. These models will be trained on the historical load data to find relationships that predict oil temperature. These are the models that we are going to use for this problem:

- Linear Regression
- Lasso Regression
- Ridge Regression
- Polynomial Regression
- Random Forest Regression
- SVR
- XGBoost Regression
- CatBoost Regression
- LightGBM Regression
- Recurrent Neural Network (RNN)
- Gated Recurrent Unit (GRU)
- Long Short-Term Memory (LSTM)
- Convolutional Neural Network (CNN)

By comparing the performance of both traditional and deep learning techniques, we aim to identify the most robust and accurate method for predicting transformer oil temperature using only load data.

Related Works

Traditional machine learning methods, such as Random Forest, Decision Tree, Support Vector Regression, and K-Nearest Neighbor, were used in earlier research (Ofori et al., 2023) to forecast transformer health using variables like top-oil, hot-spot, and ambient temperatures. Despite achieving excellent accuracy (up to 94.4%), these models were limited in their ability to capture the temporal dependencies present in continuous load and thermal variations since they relied on static relationships. Accuracy in this source refers to a score computed by the model (0-100 range) that considers RMSE and MAE. To address this limitation, our study evaluates a wide range of models, from traditional regressors (Linear, Ridge, Random Forest, and CatBoost) to time-series deep learning models (RNN, GRU, LSTM, and CNN). By analysing short and long-horizon predictions, we aim to determine

which modelling approach best captures the nonlinear, time-dependent behaviour of transformer thermal profiles.

The use of deep learning models designed specifically for long-sequence time-series forecasting has been recently researched (Boujamza and Elhaq, 2025). They developed the Neural Hierarchical Interpolation for Time Series (NHITS) model to predict transformer oil temperature. Its main strength lies in incorporating Multirate signal sampling via MaxPool layers and hierarchical interpolation to merge predictions across different time scales. By achieving a 51% reduction in mean squared error and a 38% reduction in mean absolute error when compared to the Informer model, NHITS demonstrated its superior computing capability and ability to capture intricate temperature patterns in transformers. Our study differs by evaluating a broader range of models across both traditional machine learning and deep learning, instead of focusing on a single architecture. Unlike the other studies (Ofori et al., 2023; Boujamza & Elhaq, 2025) that relied on past oil temperature data, our study compares models without the use of past oil temperature data, offering a practical solution for situations where direct temperature measurements are unavailable. Furthermore, our analysis supports their findings by investigating the effects of temporal depth and model complexity on performance stability across multiple forecast horizons.

Dataset

Dataset Description and Quality

This investigation utilizes two datasets from two different transformers. Each dataset includes an oil temperature as the target, a datetime variable, and six other variables detailing electric load. The data is recorded at a 15-minute sampling interval. The table below explains the variables contained in the datasets.

OT	Oil Temperature (in Celsius)
date	Datetime of observation
HULL	High Useless Load
HUFL	High Useful Load
MULL	Medium Useless Load
MUFL	Medium Useful Load
LULL	Low Useless Load
LUFL	Low Useful Load

Figure 1. *Dataset Description*

A preliminary data quality assessment was conducted, and no missing values or duplicated records were found. However, a potential data anomaly was identified exclusively within the trans_2.csv dataset. The source and nature of this discrepancy are undetermined at this stage of the analysis.

Data Preprocessing

During data preprocessing, the feature engineering process was employed to transform the time series prediction task into a supervised learning problem. This was achieved using a sliding window (or lag) approach, where a sequence of the past N observations was used as the input feature vector to predict a future target value. Additionally, time-based features were extracted from the datetime variable to capture seasonality. This methodology was adapted to develop models for three distinct forecasting horizons: predicting the oil temperature at the next hour, the next day, and the next week.

Subsequently, a standardization process was applied to standardize the features. To prevent look-ahead bias and data leakage, the standardization must be computed exclusively on the training dataset. These parameters were then applied to transform both the training and the test datasets to ensure that no information from the test set influenced the standardization process.

Exploratory Data Analysis

An exploratory data analysis was conducted to identify underlying temporal patterns within the data. Analysis of the oil temperature (OT) plotted on a yearly basis revealed a distinct seasonal trend, which was consistently observed across both transformer datasets. This seasonality is characterized by a tendency for oil temperatures to peak during the middle of the year and reach their lower level at the beginning and end of the year.

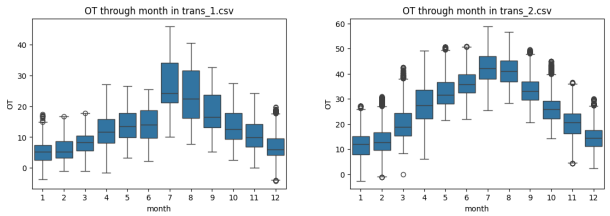

Figure 2. *Boxplot of OT*

Methods

The core problem defined is a supervised regression task: given past load observations from a transformer, the goal is to forecast its oil temperature (OT) at a future time step. Mathematically, we can describe this as learning a mapping

$$f : (x_{t-L}, x_{t-(L+1)}, ..., x_{t-(N+L)}) \rightarrow y_t$$

Figure 3. *Mathematical Mapping*

where $x_i \in \mathbb{R}^6$ represents the six load variables at time i, N is the size of the historical window, L is the prediction horizon offset (e.g., 4 for next hour, 96 for next day, 672 for next week), and yt is the target oil temperature at time t. The model f(·) is trained to minimize a regression loss such as the Root Mean Squared Error (RMSE).

We selected AI/ML techniques that balance interpretability, generalization, and temporal learning capacity. Traditional models (Linear, Ridge, Lasso, Random Forest, XGBoost) handle structured lag features effectively and allow feature importance analysis. They are efficient baselines for tabular time-series data and provide strong interpretability. Deep learning models are included to explore their ability to learn temporal dependencies directly from sequential inputs without explicit lag engineering. In particular, LSTMs are designed to overcome vanishing-gradient issues found in classical RNNs (cf. Week 10 lecture), making them well-suited for long-range dependencies in load and temperature sequences.

Workflow

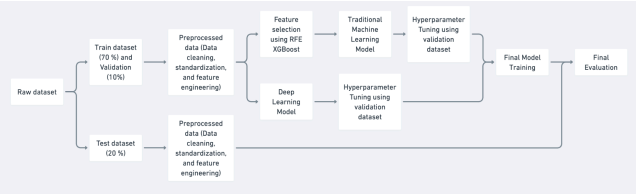

Figure 4. *Project Workflow*

As illustrated in the figure above, after preprocessing (cf. 2.2), feature selection is conducted using Recursive Feature Elimination (RFE) with an XGBoost regressor as the base estimator to identify the most informative features. Base model choice is based on the base model's efficiency in feature importance ranking. Prior studies have shown that XGBoost-based RFE can significantly improve predictive performance over other models (Suprihadi et al., 2025, DOAJ). The selected features are then used to train traditional machine learning models and deep learning models (cf. 1.3). Each model is further refined through hyperparameter tuning using the validation dataset based on the RMSE to ensure optimal performance. Once the best parameters are found, the final models are retrained on the combined training and validation data before being evaluated on the independent test set.

Results and Discussion

Evaluation

To evaluate our models, we used Root Mean Squared Error (RMSE) as the primary performance metric. RMSE penalizes larger prediction errors more heavily compared to smaller ones, making it suitable for a regression problem such as oil temperature prediction, where large deviations may indicate critical system misreadings.

We performed 5-fold time series cross-validation on the training set to ensure generalizability and to tune hyperparameters for each model. The final reported performance metrics were computed using the held-out test set, ensuring no temporal overlap with training data.

Results

The results presented in this report were obtained from our predictive models, which were executed under three distinct configurations. These configurations were designed to evaluate the model's performance across different temporal prediction horizons. Specifically, configuration 1 was set to predict the subsequent hour, configuration 2 was set to predict the subsequent day, and configuration 3 was set to predict the subsequent week. The performance metrics for each configuration are detailed in the following sections.

trans_1.csv

Model	RMSE Config 1	N Config 1	RMSE Config 2	N Config 2	RMSE Config 3	N Config 3
Linear Regression	8.8183	96	9.5943	48	8.0108	96
Lasso Regression	3.7147	48	3.9557	48	7.5609	96
Ridge Regression	5.7780	96	5.9664	96	5.9832	48
Polynomial Regression	9.7473	48	9.5042	12	8.8322	48
Random Forest	7.1539	12	7.2666	12	7.0035	48
SVR	3.8596	48	3.9334	48	4.8916	12
XGBoost Regression	4.8211	96	5.354612	96	5.1504	12
CatBoost Regression	4.7769	12	5.4113	24	4.6232	48
LightGBM	6.3048	12	6.5638	48	6.0005	24
RNN	**2.9893**	**48**	**3.3693**	**48**	4.1205	48
GRU	7.8746	48	8.2626	96	6.8129	288
LSTM	7.7124	48	6.995	72	3.4154	288

CNN	6.791	96	6.6451	96	**3.2618**	**288**

trans_2.csv

Model	RMSE Config 1	N Config 1	RMSE Config 2	N Config 2	RMSE Config 3	N Config 3
Linear Regression	7.1978	48	7.5437	24	7.6011	24
Lasso Regression	7.9911	24	7.8695	96	8.0752	12
Ridge Regression	7.2721	96	7.4769	24	8.1041	48
Polynomial Regression	7.3243	12	6.2414	12	6.6715	48
Random Forest	5.8423	48	6.3275	48	7.0506	48
SVR	7.1686	48	6.6075	48	6.4664	48
XGBoost Regression	5.5907	96	5.7065	96	6.1630	48
CatBoost Regression	**5.4323**	**24**	**5.5296**	**24**	5.8292	48
LightGBM	5.5461	48	5.7138	12	6.2871	48
RNN	7.1798	12	7.3623	12	7.6517	12
GRU	7.1925	48	7.4244	72	**5.2874**	**192**
LSTM	6.9913	48	6.7168	128	6.23	192
CNN	7.6351	96	8.3158	192	8.0396	192

Discussion

Based on the results depicting the performance of each model across both transformers (using RMSE), various conclusions can be derived. Firstly, it is evident that deep learning models consistently outperform the traditional regressors in predictive performance (except CatBoost Regression for Transformer 2), reflecting their superior ability to capture temporal dependencies in oil temperature data.

For trans_1, the CNN (RMSE ≈ 3.26, N=288) and LSTM (RMSE ≈ 3.41, N=288) achieved the best accuracy, showing that longer look-back windows substantially improve predictive performance. In trans_2, the GRU (RMSE ≈ 5.29, N=192) performed best, indicating that it is more effective in handling increased irregular patterns. In general, trans_2's higher overall RMSEs imply greater data variability or external influences affecting thermal behaviour.

For short-term horizons, models like RNN and SVR performed best, achieving low RMSEs ~2.99–3.86 for one-hour forecasts. Their ability to capture recent temporal fluctuations makes them well-suited for immediate predictions. However, their performance declined over longer horizons due to their limited memory depth. In contrast, for medium to long-term forecasts, LSTM and CNN excelled, with the CNN achieving the lowest RMSE of 3.26 (N=288) for trans_1 and the GRU (RMSE ≈ 5.29, N=192) for trans_2. These deep learning models utilise longer look-back periods to model complex temporal dependencies, making them stronger for weekly predictions. While CatBoost and XGBoost offered competitive results for shorter horizons, their performance plateaued as the forecasting window increased.

Overall, it can be concluded that shorter horizons benefit from simpler, faster models, while longer horizons require deeper architectures that retain temporal patterns over extended periods.

Societal Implications

While the above results demonstrate strong predictive performance, several avenues remain for further improvement. Incorporating exogenous variables such as weather, temperature, and humidity could enhance the model's ability to capture broader contextual influences affecting transformer temperature. Moreover, attention based architectures such as the transformer models may also offer superior capability in modeling long-range temporal dependencies compared to LSTM networks.

Lastly, it is also essential to address practical considerations. Generally, the development of accurate and reliable predictive models can hold significant societal value. By preventing transformer failures and enabling more efficient maintenance scheduling, such models can enhance operational safety and allow human operators to concentrate on higher-level system management tasks. Although exceeding human expertise is not the primary objective, a reliable forecasting system can greatly support planning as well as mitigating overheating risks, ultimately improving both the safety and efficiency of power system operations.

References

Boujamza, A.; and Lissane Elhaq, S. 2025. Predicting Oil Temperature in Electrical Transformers Using Neural Hierarchical Interpolation. Journal of Engineering 2025: 9714104. https://onlinelibrary.wiley.com/doi/full/10.1155/je/9714104

Ofori, D.; Armah, E.; Marfo, B.; Otchere, I. K.; Quansah, A.; and Kyeremeh, K. A. 2023. Predictive Health Monitoring of a Power Transformer Using Machine Learning. Electrical and Electronic Engineering 13(1): 12–18. http://article.sapub.org/10.5923.j.eee.20231301.03.html

Suprihadi, E.; Danila, N.; and Ali, Z. 2025. Enhancing financial product forecasting accuracy using EMD and feature selection with ensemble models. Journal of Open Innovation: Technology, Market and Complexity 11(2): 100531. https://www.sciencedirect.com/science/article/pii/S2199853125000666

Appendix

Linear Regression

We evaluated two options: enabling/disabling the intercept (fit_intercept ∈ {True, False}) and enforcing non-negativity on coefficients (positive ∈ {False, True}) to test whether a constrained solution improved generalization. The selected setting minimized cross-validated error.

Lasso Regression

A pipeline with feature standardization (StandardScaler) and a Lasso regressor (coordinate-descent) was tuned via grid search. The search varied the ℓ1 penalty strength (alpha) over [0.0001, 0.001, 0.01, 0.1, 1, 10], toggled the intercept term (fit_intercept ∈ {True, False}), and compared coordinate update strategies (selection ∈ {'cyclic','random'}). To avoid premature convergence warnings on high-dimensional, lagged features, max_iter was set to 80,000.

Ridge Regression

Ridge was trained where hyperparameter search explored the ℓ2 penalty (alpha) across [0.01, 0.1, 1, 10, 100], the intercept setting (fit_intercept ∈ {True, False}), and multiple solvers ('auto','svd','cholesky','lsqr','sag','saga') to balance numerical stability and speed for different data sizes.

XGBoost Regressor

A tree-based gradient boosting model (XGBRegressor, objective='reg:squarederror') was tuned with histogram-based splitting (tree_method='hist') and full CPU parallelism (n_jobs = –1). Since trees are scale-invariant, the scaler was set to passthrough. The grid spanned the learning rate (learning_rate ∈ {0.03, 0.1, 0.5}), tree depth (max_depth ∈ {3, 5, 7}), minimum child weight (min_child_weight ∈ {1, 3}), row subsampling (subsample ∈ {0.8, 1.0}), column subsampling per tree (colsample_bytree ∈ {0.8, 1.0}), and regularization terms (reg_alpha ∈ {0, 1e-2}, reg_lambda ∈ {1.0, 5.0}).

Polynomial Regression

The Polynomial Regression model with degree 2 was selected to capture moderate nonlinear relationships while maintaining interpretability. Ridge regularization was applied to control overfitting from polynomial feature expansion. The regularization strength (alpha) was tuned over [0.1, 1.0, 10.0] to balance bias and variance. A higher degree was not used, as it would generate too many features and increase the risk of overfitting and computational cost.

LightGBM Regression

The LightGBM Regressor is tuned in such a way that it balances performance and efficiency. The parameters chosen was num_leaves ∈ {31, 63} for model complexity, learning_rate ∈ {0.03, 0.1} for convergence control, n_estimators ∈ {300, 600} for training depth, and reg_lambda fixed at 1.0 for regularization. This configuration offers a practical balance between accuracy, generalization, and computational cost.

CatBoost Regression

To maintain accuracy and efficiency, the CatBoost Regressor parameters included tree depth ∈ {4, 6}, learning rate ∈ {0.03, 0.1}, number of estimators ∈ {300, 600}, and L2 regularization fixed at 3. These ranges were

chosen to test the trade-off between model complexity and generalization, while keeping computation manageable.

Random Forest Regression

A hyperparameter search was conducted to optimize performance. The tuning grid included the number of estimators (n_estimators) selected from [100, 300], the maximum tree depth (max_depth) from [5, 15], the minimum samples required to split an internal node (min_samples_split) from [2, 8], and the minimum samples required to be at a leaf node (min_samples_leaf) from [1, 3].

Support Vector Regression

The Support Vector Regression (SVR) model was optimized over a grid of key hyperparameters. This search included the kernel type (kernel), testing 'linear', 'rbf', and 'poly' functions; the regularization parameter (C), with values of [0.1, 1, 10]; the kernel coefficient (gamma), using 'scale', 'auto', and 0.1; and the epsilon-tube margin (epsilon), set to [0.1, 0.4].

Recurrent Neural Network

The final architecture was determined through an iterative experimental process. It was observed that architectures utilizing more than one layer were prone to overfitting. Consequently, the selected configuration consists of a single RNN layer with 64 neurons and two dense layers with ReLU as the activation. A dropout rate of 0.2 was applied for regularization, and the model was trained for 20 epochs with a batch size of 64.

Long Short-Term Memory

We tuned a single-layer LSTM for week-ahead forecasting under strict no-peek (OFFSET=GAP=672). The search varied hidden size (`units` ∈ {64, 96, 128, 160}), dropout (`dropout` ∈ {0.1, 0.2, 0.3}), learning rate (Adam, `lr` ∈ {1e-3, 5e-4, 2e-4}), and batch size (`batch_size` ∈ {32, 64, 128}). We also compared MSE vs Huber loss (δ∈{1.0,1.5}) and enabled gradient clipping (`clipnorm`=1.0) to stabilise long-horizon training. When an official validation split was empty, we carved the last 10% of train windows as a temporary validation set (time-respecting).

Gated Recurrent Unit

We tuned a single-layer GRU for week-ahead forecasting under strict no-peek (OFFSET=GAP=672). The search varied hidden size (units ∈ {96, 128, 160}), dropout (dropout ∈ {0.10, 0.15, 0.20, 0.25}), learning rate (Adam, lr ∈ {1e-3, 5e-4, 2e-4}), and batch size (batch_size ∈ {64, 128}). We fixed Huber loss (δ=1.5) and gradient clipping (clipnorm=1.0) for robustness). Window/stride were transformer-specific to reflect thermal memory: trans_1 L=288, s=2; trans_2 L=192, s=2. The selected setting for trans_2: units=128, dropout=0.20, lr=2e-4, batch_size=128, minimised validation error (val_loss≈0.1004; MAE_z≈0.346; MAE≈4.08 °C).

Convolutional Neural Network

The CNN had lightweight residual connections to capture hours→days dependencies while preventing leakage across the 1-week gap. The tuning grid covered feature channels {96, 112, 128}, kernel {3, 5}, dropout {0.10, 0.15, 0.20}, Adam learning rate {3e-4, 2e-4, 1e-4}, and batch size {64, 128}; Huber loss and gradient clipping (1.0) were fixed. Early stopping (with ReduceLROnPlateau) monitored val_loss when a validation split existed, otherwise training loss. The best-performing config (trans_1): channels = 128, kernel = 3, dropout = 0.15, lr = 2e-4, batch = 128.