

Short-Term Load Forecasting for Hospitals Using Transformer-Based Deep Learning with Temporal Memory and Statistical Features

Piyapong Boonsompan and Kampol Woradit[†], Non-members

ABSTRACT

Reliable electricity is essential for hospitals because many clinical and support services run 24/7. This paper presents a single-hospital case study on short-term load forecasting (STLF) using half-hourly data from Mae Moh Hospital, Thailand, from October 2023 to February 2025. To reduce data leakage, we use a leakage-aware feature pipeline that relies only on past observations and fits scaling on the training set only. We use a 48-step input window (24 hours) to predict the next two steps (+30 and +60 minutes). To evaluate robustness over time, we use rolling-origin temporal cross-validation (expanding window). We compare compact hybrid Transformer models (TransLSTM, TransGRU, TransCNN) with classical baselines (Extra Trees, Random Forest, LSTM, CNN-DR, Transformer) and a modern Transformer baseline (PatchTST). Across folds, PatchTST achieves the best average performance ($RMSE = 1.186 \pm 0.776$, $MAE = 0.762 \pm 0.530$, $R^2 = 0.879 \pm 0.064$). The proposed hybrid models remain close, and TransGRU/TransLSTM are stable across time splits. Finally, we analyze errors by time window and by fold to highlight the periods with higher uncertainty.

Keywords: Short-Term Load Forecasting, Hospital Energy, Temporal Cross-Validation, Transformer, PatchTST, Hybrid Deep Learning, Feature Engineering

1. INTRODUCTION

Thailand has announced ambitious targets for carbon neutrality by 2050 and net-zero greenhouse gas emissions by 2065 under the LT-LEDS and updated NDCs [1,2]. Achieving these goals requires greater integration of variable renewable energy sources such as solar and wind. While renewables are clean and sustainable, their variability introduces short-term uncertainty to power systems, increasing the need for flexibility on both supply and demand sides. Virtual Power Plants (VPPs)

address this challenge by aggregating distributed energy resources (DERs)-including rooftop photovoltaics, battery storage, electric vehicle chargers and flexible loads-and coordinating them through forecasting and control [3,4]. Accurate prediction of both supply and demand is therefore a fundamental enabler of reliable VPP operation [5].

Hospitals represent a unique and critical category of consumers in this energy transition. They operate 24/7 and require an uninterrupted electricity supply for medical devices, life-support systems, laboratories and IT infrastructure. Consequently, even brief outages or poor load planning can directly compromise patient safety. Hospitals also face increasing electricity costs due to high base demand, heavy air-conditioning loads and occupancy fluctuations [6,7]. Compared to other commercial buildings, hospitals are more energy-intensive and exhibit complex consumption patterns driven by visiting hours, staff shifts, equipment usage, weekends, holidays and weather variations [8-10]. As a result, accurate short-term load forecasting (STLF) is not only an economic tool, but also a matter of operational reliability and quality of care [11].

We use half-hourly electricity consumption data from Mae Moh Hospital, Lampang, Thailand. After cleaning and timestamp checks, the dataset contains 24,816 records from 1 Oct 2023 to 28 Feb 2025. The paper focuses on a practical and reproducible workflow for short-term forecasting. Because this is a single-hospital dataset, the findings should be viewed as a case study.

The contributions of this paper are summarized as follows.

1. Leakage-aware features. We design a feature pipeline with three input groups: (i) cyclic time-of-day, (ii) operational context (weekend, holiday, working hour) and (iii) temporal memory with statistics (lags, differences, rolling statistics, EWMA). All features use only past data and training-based scaling to avoid leakage.
2. Temporal validation for robustness. We evaluate models using rolling-origin temporal cross-validation (expanding window) so that training uses earlier months and testing uses later months. This better reflects real deployment than a single split.
3. Modern baseline comparison. We compare compact hybrid Transformer models with standard baselines and include PatchTST as a modern Transformer-based forecasting baseline.
4. Simple interpretability for practice. We report error

Manuscript received on September 23, 2025; revised on February 25, 2026; accepted on March 4, 2026. This paper was recommended by Associate Editor Siraporn Sakphrom.

The authors are with Department of Computer Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai, Thailand.

[†]Corresponding author: kampol.w@cmu.ac.th

©2026 Author(s). This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License. To view a copy of this license visit: <https://creativecommons.org/licenses/by-nc-nd/4.0/>.

Digital Object Identifier: 10.37936/ecti-ec.2026242.261439

patterns by daily windows (Night-Early/Daytime/Late-day) and show fold-wise robustness across time splits. These analyses help explain when the model is likely to make larger errors.

We discuss how the model may generalize to other hospitals and how transfer learning could help when additional hospital data become available. Although we only evaluate one hospital, the workflow is designed to be reusable. For another hospital with the same sampling rate, the same feature set and training steps can be applied. In practice, a simple transfer strategy is to start from a model trained on Hospital A, then fine-tune it on a few weeks of Hospital B data with train-only scaling for Hospital B. Another practical option is to freeze the main encoder and retrain only the final prediction layers. These strategies are not tested in this study due to data availability, but they provide a clear direction for multi-hospital deployment in future work.

2. RELATED WORKS

Research on energy forecasting has evolved from classical statistical methods to modern hybrid deep learning models. Early studies focused on statistical approaches like exponential smoothing and ARIMA, which handled seasonality in short-term electricity demand [12,13]. These methods provided useful baselines and remain strong references for benchmarking [14,15].

Later, data-driven methods became popular, especially ensemble models like Random Forests and gradient boosting. Reviews showed that these models achieved good performance when combined with engineered features such as weather variables and occupancy indicators [8,16,17]. Specific studies on healthcare facilities highlighted that hospitals are among the most energy-intensive building types, with high variability driven by schedules and medical equipment usage [6,7]. More recently, Koç and Seçkiner [9] analyzed hospital energy demand using machine learning regressors and confirmed the importance of weather and calendar effects. Other works also explored practical systems such as API-enabled ML platforms and monthly smart forecasting systems [18,19].

Deep learning has since become mainstream in short-term load forecasting. LSTM and GRU networks were widely applied to capture temporal dependencies [20], while CNNs and temporal convolutional networks (TCNs) offered faster sequence modeling [21]. More advanced probabilistic and interpretable models, such as DeepAR [22] and N-BEATS [23], further expanded the design space. In parallel, Transformer-based models gained popularity for long-sequence and multi-horizon forecasting. The Temporal Fusion Transformer (TFT) showed how attention and gating improve interpretability [24], while Informer, Autoformer, Frequency enhanced decomposed transformer (FEDformer) and Patch time series Transformer (PatchTST) extended Transformer design with efficient attention, decomposition and patching mechanisms [25-29]. More recent work also

combined clustering with Transformers for smart grid STLF [30].

Transformer-based forecasting has recently become a strong baseline for time-series prediction. Several variants were proposed to improve efficiency and accuracy, such as Informer, Autoformer, FEDformer and PatchTST [25-28]. PatchTST is especially popular because it represents the input sequence as patches, which often improves learning while keeping the model structure simple [28].

Although Transformer-based models provide strong capability for capturing long-range temporal dependencies, hospital load forecasting also requires sensitivity to short-term transitions and local demand spikes. Self-attention enables the model to observe the full 24-hour context, while recurrent or convolutional layers are effective at modeling step-by-step dynamics and short-term variations. For this reason, we focus on compact hybrid architectures (Transformer+LSTM/GRU/CNN) that combine global context awareness with local temporal tracking, while keeping the model practical for hospital deployment in terms of size and training cost.

As later demonstrated in Section 4 RESULTS AND DISCUSSION, this design choice is empirically supported, where the proposed hybrid models show stable and competitive performance across temporal folds, particularly in periods with higher variability such as daytime transitions.

3. METHODOLOGY

We explain how we transform raw half-hour hospital data into features and how we design hybrid deep learning models. We provide both equations and simple explanations for clarity.

3.1 Data Preprocessing

Data Cleaning makes the dataset clean and consistent before it is used by the models. If the data is not cleaned, the models may learn from wrong or incomplete information, which leads to poor forecasts. In this study, three simple steps were applied:

1. Parse and sort timestamps, remove duplicates. All data records were aligned on a fixed 30-minute time grid. The time column was converted into a standard format and sorted in order. Duplicate rows were removed so that each half-hour appears only once. This ensures that the time series is continuous and correct.
2. Convert all values to numeric. Some columns in the raw dataset included text, symbols, or mixed formats (for example, numbers with attached units or missing values written as words). All of these fields were cleaned and converted into numeric form. Any entries that could not be converted were marked as missing. This ensures that both electricity consumption and weather variables are represented as numbers that the model can directly use for calculations.

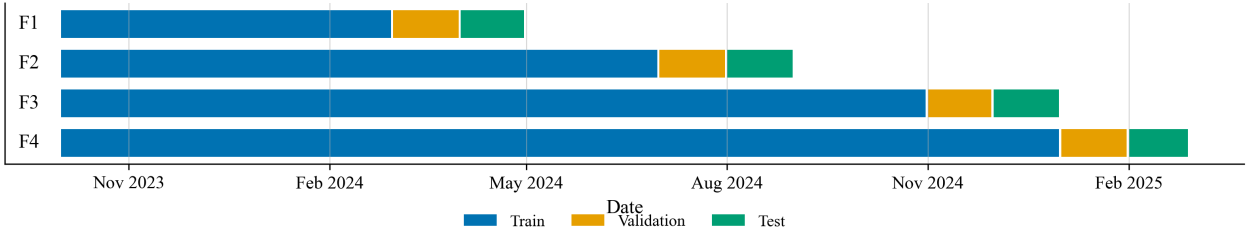


Fig. 1: Rolling-origin temporal cross-validation with an expanding window (F1-F4).

- Fill short gaps in load and weather variables. Missing values were rare. When they occurred, we used linear interpolation to fill short gaps in load and weather variables to keep the half-hour series continuous. In our dataset, this step was rarely needed because the records were mostly complete and consistent.

In summary, preprocessing ensures that the dataset is clean, well-structured and fully numeric. This provides the models with reliable inputs and improves both training stability and forecasting accuracy.

3.2 Data Splitting (Rolling-origin Temporal Cross-Validation)

A single split can give an optimistic or unstable estimate if the data distribution changes over time. To better reflect real deployment, we use rolling-origin temporal cross-validation with an expanding training window. Each fold trains on earlier months, validates on the next month and tests on the following month. Fig. 1 shows the split design.

3.3 Feature Engineering

We designed three groups of features to enrich the input data and help the models capture temporal patterns of hospital electricity consumption. All features are constructed in a leakage-aware manner, meaning that only past and present information is used and normalization is performed with training data only. This avoids the use of unseen future data. Observed load as an input. At each time step, we include the observed electricity load itself as part of the input feature vector, denoted as $load_t$. This is the measured load at the current timestamp within the lookback window and is available in practice; the model never uses future load values when forming inputs.

3.3.1 Cyclic time-of-day

This group encodes the daily rhythm of hospital operations. Electricity demand typically follows a repeated cycle higher during working hours, lower at night. By transforming the time-of-day into smooth cyclic features, the model can easily learn where a given timestamp falls within the daily cycle and better capture these recurring load patterns.

To represent the time-of-day cycle, we use two cyclic features: $time_sin = \sin(\frac{2\pi t}{T})$ and $time_cos = \cos(\frac{2\pi t}{T})$,

where t is the minute-of-day and $T = 1440$ minutes. This encoding avoids a discontinuity between 23:59 and 00:00 and helps models learn daily patterns more smoothly.

3.3.2 Operational context features

We created three new binary features (0 or 1) to represent the hospital's daily context:

- Weekend indicator: 1 if the day is Saturday or Sunday, otherwise 0.
- Public holiday indicator: 1 if the day is a public holiday, otherwise 0.
- Working hour indicator: 1 if the time is within regular working hours (08:00-16:00), otherwise 0.

These features allow the model to recognize important differences in hospital energy use. For example, electricity demand is usually lower during weekends and holidays, but higher during working hours when medical equipment and staff activity are at peak levels. Including these simple contextual features helps the model better align its forecasts with real operating conditions in the hospital.

3.3.3 Temporal memory and statistics

We add four groups of history-based features. They are built only from past data with no future information and are updated at each timestamp in a sliding manner. This design makes the inputs realistic and leakage-free, while giving the model clear signals about level, change and variability.

- Lags (13 features). Past load values are attached at fixed intervals: 30, 60, 90, 120 minutes (1–4 steps), 3 hours (6 steps), 6 hours (12 steps), 12 hours (24 steps), 24 hours (48 steps), 2 days (96 steps), 3 days (144 steps), 6 days (288 steps), 7 days (336 steps) and 14 days (672 steps). Current demand often resembles the same time yesterday or last week, so these 13 lag features provide memory of recurring patterns and improve accuracy during periods such as daytime ramps and evening declines.
- Differences (2 features). We compute the change from the previous half-hour and the change between the same times on consecutive days. These values highlight both short-term fluctuations and daily-scale shifts, allowing the model to capture whether demand is rising or falling and by how much.

Pseudo-code: Leakage-aware temporal cross-validation pipeline

Input: Half-hourly load and context data; lookback $L = 48$; horizon $H = 2$; folds F1-F4

Output: Fold-wise metrics and predictions

1. Parse timestamps, sort records, and remove duplicates
2. Build leakage-aware features using only past values (lags, differences, rolling stats, EWMA)
3. For each fold $f \in \{F1, \dots, F4\}$:
 - 3.1 Split data into train/val/test by time (Fig. 1)
 - 3.2 Fit scalers on train only; transform val/test
 - 3.3 Create sequences with input length L and horizon H
 - 3.4 Train the model with early stopping on the validation set
 - 3.5 Evaluate on the test set and store predictions/metrics
4. Report mean $\pm$ std across folds

Fig. 2: Pseudo-code of the proposed leakage-aware workflow with rolling-origin temporal cross-validation.

- Rolling statistics (24 features). For windows of 2 hours (4 steps), 6 hours (12 steps), 12 hours (24 steps), 24 hours (48 steps), 48 hours (96 steps) and 1 week (336 steps), we calculate the mean, standard deviation, minimum and maximum. This results in 6 windows $\times$ 4 statistics = 24 features. The rolling mean serves as a local baseline, the standard deviation measures short-term volatility and the min/max track recent extremes. Together, these features help the model adapt to both stable and highly variable conditions.
- Exponentially Weighted Moving Average (1 feature). This is a type of running average that pays more attention to the most recent values while still keeping some influence from older ones. We use the past 24 half-hour steps (12 hours). This helps the model follow the latest trend in energy use more quickly, while still remembering the longer-term pattern. As a result, the model can react better when hospital demand suddenly changes.

All features are computed only from past observations at each timestamp. Scaling is fitted on the training data and applied consistently to validation and test sets. In total, this group contributes 40 features (13 lags + 2 differences + 24 rolling stats + 1 EWMA). These features provide the model with baseline levels, variability, recent trends and memory of prior demand, which leads to more stable training and more accurate short-term forecasts.

3.4 Sequence Construction

We frame the problem as a multi-step forecasting task. Specifically, we use a sliding window of the past 24 hours (48 half-hour steps) as input, with each step containing all engineered features. The output is the next two half-hour load values, corresponding to forecasts at +30 minutes and +60 minutes. This setup is practical for hospital operations because it provides actionable predictions one hour ahead while preserving a realistic sequential structure. The models compared in this

study, including both classical baselines and the proposed hybrid Transformer architectures, are summarized in Table 1.

3.5 Hybrid Transformer Models

We use hybrid models that start with a Transformer encoder to capture global patterns over the past 24 hours and then refine these representations with an additional block (CNN, GRU, or LSTM) before producing the final forecast. Below we explain the step-by-step equations and variables in plain language.

3.5.1 PatchTST baseline

We include PatchTST as a modern Transformer-based forecasting baseline. The 48-step input is reshaped into non-overlapping patches with patch length $P = 4$ (stride $S = 4$ in half-hour steps). Each patch is projected to $d_{model} = 64$ and processed by a compact Transformer encoder (2 layers, 4 heads, dropout 0.1). We use the same training protocol with optimizer, early stopping and evaluation as the other deep models for a fair comparison.

Transformer Encoder. For each time step t in the lookback window, the input vector includes the observed load $load_t$ together with calendar, weather, context features and history-based statistics. The targets are the next two loads ($load_{t+1}$, $load_{t+2}$), so all inputs use only information available up to time t .

Each time step t has input features (load, calendar, weather). These are embedded and combined with a positional encoding:

$$h_t^{(0)} = z_t + p_t \quad (1)$$

where: $z_t \in \mathbb{R}$: represents the embedding of input features. $p_t \in \mathbb{R}$ denotes the positional encoding reflecting temporal order and $h_t^{(0)}$ is the initial hidden state at step t .

Stacking all steps gives $H^{(0)} \in \mathbb{R}^{L \times d}$, where $L = 48$ step (24h). The Transformer projects this sequence into queries, keys and values:

$$Q = H^{(0)}W_Q, \quad K = H^{(0)}W_K, \quad V = H^{(0)}W_V \quad (2)$$

where: W_Q, W_K, W_V are trainable weight matrices and Q, K, V are representations used to measure similarity between time steps.

Self-attention then computes how each step relates to others:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3)$$

This produces $H \in \mathbb{R}^{L \times d}$, a new sequence of hidden states that captures global dependencies across the whole 24-hour history. d_k is the key dimension used for scaling.

Table 1: Summary of compared models.

Model	Main idea / practical note
ETR, RandomForest	Strong non-linear baselines with feature engineering; easy to run
CNN-DR	Captures local patterns; often good at sharp changes
LSTM	Sequence model with memory; good for smooth temporal transitions
Transformer	Attention over full input window; captures global daily context
PatchTST	Modern Transformer baseline using patches; strong accuracy in many tasks
TransCNN	Transformer context + CNN for local spikes
TransGRU	Transformer context + GRU for efficient step-by-step updates
TransLSTM	Transformer context + LSTM for stable tracking of ramps and transitions

3.5.2 Transformer + 1D-CNN (TransCNN)

The TransCNN model combines a Transformer encoder with one-dimensional convolutional layers (1D-CNN). The Transformer first captures global dependencies across the 24-hour history and then CNN layers refine local short-term patterns which are fluctuations before producing the final prediction. The process can be described below.

$$c_1 = \text{ReLU}(W_1 * H + b_1) \quad (4)$$

$$c_2 = \text{ReLU}(W_2 * c_1 + b_2) \quad (5)$$

where: c_1 , c_2 are feature maps that capture short-term spikes and ramps. convolution operation sliding over time. ReLU introduces nonlinearity and keeps only positive signals.

The sequence is then condensed by global average pooling:

$$g = \frac{1}{L} \sum_{t=1}^L c_2, t \quad (6)$$

Finally, a linear layer maps to the forecast:

$$\hat{y} = W_o g + b_o \quad (7)$$

Global average pooling compresses the sequence c_2 into a single summary vector g . A linear layer with weights W_o and bias b_o then maps g to the final forecast $\hat{y} \in \mathbb{R}^H$, where $H = 2$ corresponds to the +30 minute and +60 minute predictions.

TransCNN combines the strengths of both models: the Transformer captures the overall daily rhythm and long-term dependencies, while CNN layers focus on local spikes and short bursts (e.g., equipment being switched on). This balance makes it well-suited for hospital load forecasting where both gradual cycles and sudden changes are important.

3.5.3 Transformer + GRU (TransGRU)

This model first uses a Transformer to learn the big-picture patterns over the past 24 hours, then a GRU to track step-by-step changes and finally a linear layer to predict the next two steps.

$$u_t = \sigma(W_u h_t + U_u r_{t-1}) \quad (8)$$

$$r_t = \sigma(W_r h_t + U_r r_{t-1}) \quad (9)$$

$$\tilde{r}_t = \tanh(W_h h_t + U_h(r_t \odot r_{t-1})) \quad (10)$$

$$r_t^{new} = (1 - u_t) \odot r_{t-1} + u_t \odot \tilde{r}_t \quad (11)$$

where: u_t (update gate): works like a switch that decides how much of the new information at time t should be added into memory. r_t (reset gate): decides how much of the old memory should be cleared away before processing new input. $\tilde{r}_t$ (candidate state): the temporary new memory, created from the current input and part of the old memory. r_t^{new} (updated state): the final memory at time t , made by mixing the old memory and the candidate state according to the update gate.

After the last time step L , we take r_L :

$$\hat{y} = W_o r_L + b_o \quad (12)$$

GRU is well-suited for hospital load forecasting because it can update its memory step by step in a simple and efficient way. Using only two gates, it decides how much past information should be kept and how much new information should be added. This structure makes the model smaller and faster while still capturing the short-term patterns that often occur in hospital demand, such as sudden increases when equipment is switched on or decreases during quieter hours. In practice, this efficiency is important for hospitals, where forecasting systems may need to run on limited servers and provide rapid updates. TransGRU combines this efficiency with the Transformer's ability to see the whole-day context, resulting in a model that is both lightweight and responsive, making it practical for real-time hospital energy management.

3.5.4 Transformer + LSTM (TransLSTM)

This model also starts with a Transformer for global context, but then uses an LSTM, which keeps an explicit memory cell to model longer dependencies and smoother transitions.

$$i_t = \sigma(W_i h_t + U_i m_{t-1}) \quad (13)$$

$$f_t = \sigma(W_f h_t + U_f m_{t-1}) \quad (14)$$

$$g_t = \tanh(W_g h_t + U_g m_{t-1}) \quad (15)$$

$$o_t = \sigma(W_o h_t + U_o m_{t-1}) \quad (16)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (17)$$

$$m_t = o_t \odot \tanh(c_t) \quad (18)$$

where: i_t (input gate): decides how much of the new information at time t should be stored in memory.

f_t (forget gate): chooses how much of the old memory should be erased.

o_t (output gate): controls how much of the memory should be revealed as output.

g_t (candidate state): the proposed new content that could be added to memory.

c_t (cell state): the long-term memory that carries information across many time steps.

m_t (hidden state): the short-term output that represents what the model focuses on at the current step.

After the last step L , we take m_L :

$$\hat{y} = W_o m_L + b_o \quad (19)$$

LSTM is especially good at remembering information over long periods of time. It keeps two kinds of states: a hidden state that captures short-term patterns and a memory cell that stores longer-term context. This design allows LSTM to link what happened many hours earlier with what is happening now. For hospital energy demand, this is very important because the load does not only depend on the last few minutes but also follows daily cycles, such as the steady rise in the morning and daytime when activity begins, the flat plateau around midday when demand is stable and the decline in the evening after visiting hours. By combining LSTM with a Transformer, the model benefits from both: the Transformer provides a broad view of the entire day, while the LSTM ensures smooth tracking of how demand changes from one period to the next. This combination captures both long-term cycles and fine-grained transitions, which leads to highly accurate forecasts that reflect real hospital operations.

Overall, each hybrid model has its own advantage depending on the load pattern and deployment needs.

TransCNN is good for capturing local spikes and short bursts in demand.

TransGRU is lightweight, efficient and effective for sequential updates.

TransLSTM is the strongest for modeling both short and long-term hospital cycles, typically giving the best accuracy.

Together, these hybrids show how combining Transformer's global context with different sequence modules improves hospital load forecasting in different ways.

The main hyperparameters used for all models are listed in Table 2. The models use 48 past half-hourly steps (24 hours) as input and predict 2 future steps (1 hour ahead). The Transformer backbone has 2 encoder layers, 8 attention heads, hidden size 128 and a feed-forward network (FFN) of 512 units, where FFN expands and projects features for richer representation. Training was performed with AdamW (Adam with decoupled Weight

Table 2: Hyperparameters for all deep models.

Hyperparameter	Value
Input length (lookback)	48 (30-min steps; 24 hrs)
Output horizon	2 (+30 and +60 minutes)
Transformer layers	2
Attention heads	4
Hidden dimension	64
FFN dimension	128
Dropout	0.1
Optimizer	AdamW
Batch size	256
Max epochs	30 (early stopping)
Early stopping patience	6
Gradient clipping	1.0
Loss function	MSE

Table 3: Model-specific components.

Model	Extra components
Transformer	Last-step representation + FC head (2-step output)
TransLSTM	LSTM (hidden=64, layers=1) after Transformer + FC
TransGRU	GRU (hidden=64, layers=1) after Transformer + FC
TransCNN	2xConv1d (channels=64, k=3) + pooling + FC
CNN-DR	2xConv1d (channels=64; k=5 then k=3) + pooling + FC
PatchTST	Patch length $P = 4$, stride $S = 4$ (non-overlapping) + Transformer encoder + FC

decay) (lr=0.001, weight decay=0.001), batch size 128, 120 epochs, dropout 0.2 and gradient clipping 1.0. The loss combines MSE and Huber loss with horizon weights [0.7, 0.3] to emphasize the first step. Table 3 shows model specific components: the baseline Transformer uses pooling and fully connected (FC) layers for prediction, while TransLSTM and TransGRU add recurrent layers (128 units) and TransCNN uses two 1D convolutions (128 filters, kernel=3) with max pooling before the FC layers. Here, FC denotes fully connected layers that map the feature representation to the final 2-step forecast.

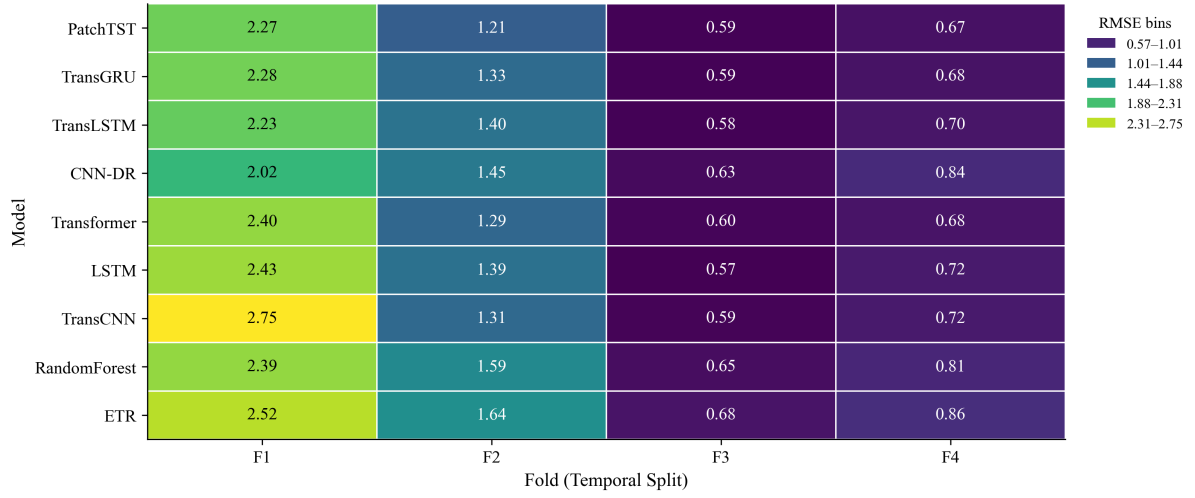
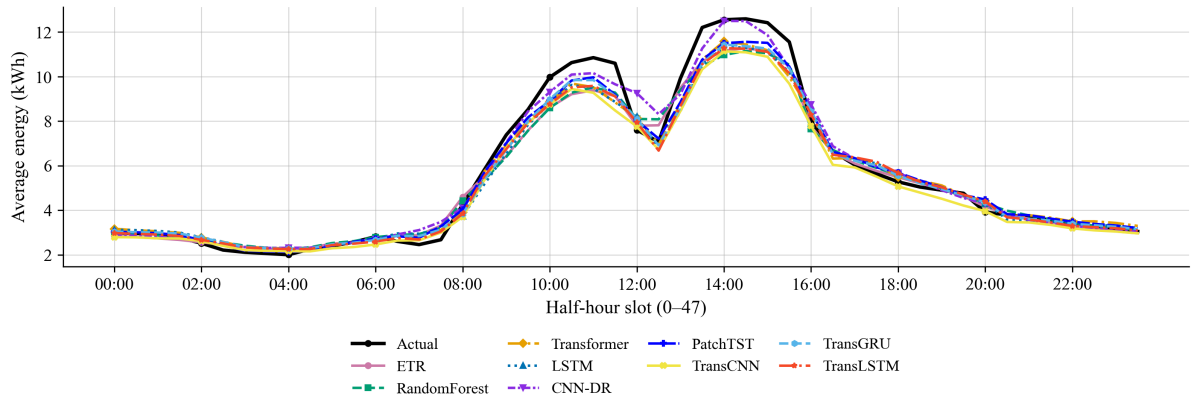
3.6 Training and Evaluation Metrics

This model also starts with a Transformer for global context, but then uses an LSTM, which keeps an explicit memory cell to model longer dependencies and smoother transitions.

To train the forecasting models, we used the Mean Squared Error (MSE) as the loss function. MSE calculates the average squared difference between the predicted values and the actual values. Because the differences are squared, larger errors are penalized more heavily than smaller ones. This encourages the model to produce forecasts that are close to the actual consumption values across both prediction steps (+30 minutes and +60 minutes). MSE is widely used in energy forecasting

Table 4: Temporal cross-validation results (mean $\pm$ std across folds, RMSE/MAE).

Model	RMSE	MAE	R^2
PatchTST	1.186 $\pm$ 0.776	0.762 $\pm$ 0.530	0.879 $\pm$ 0.064
TransGRU	1.218 $\pm$ 0.783	0.793 $\pm$ 0.553	0.874 $\pm$ 0.061
TransLSTM	1.228 $\pm$ 0.762	0.798 $\pm$ 0.554	0.873 $\pm$ 0.054
CNN-DR	1.233 $\pm$ 0.630	0.878 $\pm$ 0.521	0.867 $\pm$ 0.025
Transformer	1.242 $\pm$ 0.830	0.829 $\pm$ 0.635	0.869 $\pm$ 0.073
LSTM	1.278 $\pm$ 0.848	0.841 $\pm$ 0.611	0.863 $\pm$ 0.073
TransCNN	1.342 $\pm$ 0.989	0.868 $\pm$ 0.652	0.847 $\pm$ 0.108
RandomForest	1.360 $\pm$ 0.803	0.910 $\pm$ 0.599	0.844 $\pm$ 0.055
ETR	1.426 $\pm$ 0.838	0.969 $\pm$ 0.646	0.827 $\pm$ 0.061

**Fig. 3:** Temporal cross-validation robustness heat map of all models measured by RMSE across four folds.**Fig. 4:** Comparison between the actual load profile and model predictions based on the hourly average energy consumption during working days.

because it provides a clear signal during training and helps the model converge to stable solutions.

Multi-step evaluation. All models predict two future steps (+30 and +60 minutes). For reporting RMSE, MAE and R^2 , we aggregate errors from both horizons and compute each metric over the combined set of predictions. This provides a single score per model under

the same 2-step forecasting task.

Time-window definition. We group each target timestamp into three daily windows: Night-Early (00:00-07:59), Daytime (08:00-15:59) and Late-day (16:00-23:59). For each window, MAE is computed using errors from both horizons that fall within that window.

For evaluation, we applied three metrics that provide

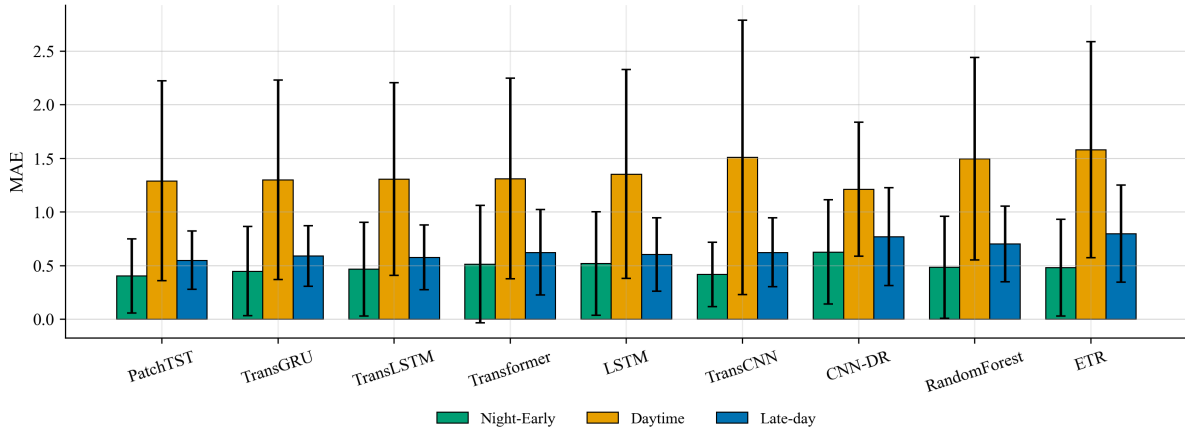


Fig. 5: MAE comparison across night-early, daytime, late-day averaged over temporal folds with standard deviation.

Table 5: Clock-time windows used for time-window error analysis.

Window	Clock-time range
Night-Early	00:00-07:59
Daytime	08:00-15:59
Late-day	16:00-23:59

different perspectives on model performance:

1. Root Mean Squared Error (RMSE): This is the square root of the MSE and has the same unit as the original data (kWh). RMSE emphasizes large errors, which makes it especially useful in hospital settings where big mistakes during peak demand periods can have serious consequences for energy management.
2. Mean Absolute Error (MAE): This metric calculates the average of the absolute differences between predictions and actual values. Unlike RMSE, it treats all errors equally, without extra penalty for larger ones. MAE is easy to interpret as the typical size of an error in the forecasts.
3. Coefficient of Determination (R^2): This measures how well the model explains the variability of the actual data. An R^2 value close to 1 means the predictions capture most of the fluctuations in energy consumption, while lower values indicate poorer performance.

All metrics were reported together for the +30 minutes and +60 minutes horizons. This separation is important because forecasting further into the future is usually more difficult and errors often increase with horizon length. By examining both horizons, we can understand not only how well the models perform in the near term but also how quickly accuracy declines as the forecast window extends. For hospitals, this is particularly critical: highly accurate short-term forecasts enable proactive actions such as scheduling equipment usage, reducing peak loads and ensuring reliable participation in Virtual Power Plant programs.

4. RESULTS AND DISCUSSION

4.1 Temporal cross-validation Performance (Mean±Std Across Folds)

We evaluate all models using rolling-origin temporal cross-validation (F1-F4). For each fold, metrics are computed on the test period using both forecast steps ($H = 2$), then we report mean±std across folds. Table 4 summarizes the results. Overall, PatchTST gives the best average accuracy. The proposed hybrid models are close behind and show stable performance across time splits.

4.2 Robustness Across Time Splits

Fig. 3 shows the fold-wise RMSE for each model. This visualization helps confirm robustness: F1 is generally the hardest split (higher RMSE), while later folds tend to be easier. PatchTST performs well across all folds. The hybrid models (TransGRU and TransLSTM) also remain stable and competitive.

4.3 Hourly Profile on Working Days (All Models)

To connect the behavior of the model with hospital operations, Fig. 4 shows the average half-hour load curve on working days for the +30 minutes horizon. The true load typically rises in the morning, stays relatively high through midday and decreases in the evening. Most models track the mid-day level well, while larger errors appear during the daytime ramp and late-day transitions.

4.4 Error by Daily Time Window

We group each target timestamp into three clock-time windows:

Night-Early (00:00-07:59), Daytime (08:00-15:59) and Late-day (16:00-23:59), as shown in Table 5.

Because the dataset is sampled every 30 minutes, the last timestamps inside these windows are 07:30 and 23:30. For each window, MAE is computed by pooling absolute errors from both horizons (+30 and +60 minutes) whose target timestamps fall inside that window.

Table 6: MAE by time window (mean $\pm$ std across folds).

Model	RMSE	MAE	R^2
PatchTST	0.402 $\pm$ 0.347	1.308 $\pm$ 0.979	0.576 $\pm$ 0.269
TransGRU	0.448 $\pm$ 0.415	1.321 $\pm$ 0.974	0.611 $\pm$ 0.280
TransLSTM	0.466 $\pm$ 0.438	1.324 $\pm$ 0.937	0.604 $\pm$ 0.301
CNN-DR	0.627 $\pm$ 0.486	1.204 $\pm$ 0.623	0.805 $\pm$ 0.470
Transformer	0.512 $\pm$ 0.548	1.327 $\pm$ 0.969	0.649 $\pm$ 0.397
LSTM	0.519 $\pm$ 0.483	1.373 $\pm$ 1.018	0.631 $\pm$ 0.338
TransCNN	0.417 $\pm$ 0.300	1.537 $\pm$ 1.335	0.650 $\pm$ 0.326
RandomForest	0.484 $\pm$ 0.476	1.513 $\pm$ 0.986	0.733 $\pm$ 0.348
ETR	0.480 $\pm$ 0.451	1.602 $\pm$ 1.052	0.825 $\pm$ 0.442

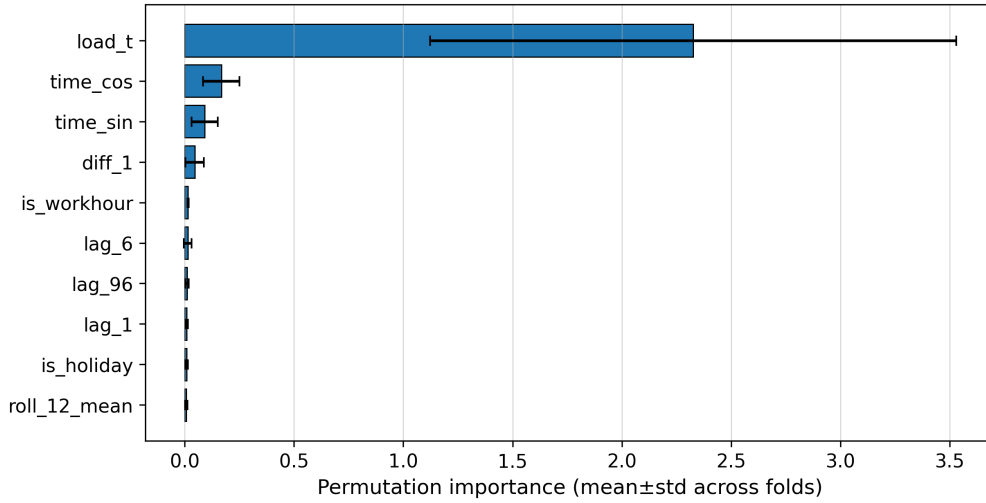
**Fig. 6:** Top-10 feature importance from RandomForest (Horizon +30 minutes).

Fig. 5 summarizes MAE by time window using temporal cross-validation mean $\pm$ std. In general, the Daytime window is the hardest due to higher activity and stronger variability. Table 6 provides the numeric values.

4.5 Notes on Interpretability

In a hospital setting, it is important to know not only how accurate the model is, but also when it may fail. The time-window analysis and the fold-wise heatmap as shown in Fig. 3 provide a simple and practical form of interpretability, they show that errors are larger in high-activity periods in the daytime and in the earliest time split. More detailed explainability methods are planned as future work when more deployment constraints and user feedback are available. We use simple clock-time windows (Night-Early, Daytime, Late-day) to summarize when errors increase, and we pool errors from both horizons (+30 and +60 minutes) for a stable estimate.

To provide a simple and practical form of interpretability, we report the top-10 feature importance from the RandomForest baseline in Fig. 6. The most important features are mainly history-based variables, together with time-of-day signals (sine/cosine) and working-hour indicators. This matches hospital operations where

demand follows daily cycles but also changes with short-term ramps and recent variability. A similar pattern is observed for Extra Trees.

5. CONCLUSION

This paper presented a leakage-aware workflow for short-term hospital load forecasting using half-hourly data from Mae Moh Hospital (Oct 2023 - Feb 2025). We evaluated compact hybrid Transformer models using rolling-origin temporal cross-validation to better reflect real deployment, where training uses earlier data and testing uses later periods. Across the temporal cross-validation, PatchTST achieved the best average performance on this dataset (RMSE = 1.186 $\pm$ 0.776, MAE = 0.762 $\pm$ 0.530, R^2 = 0.879 $\pm$ 0.064). This shows that a patch-based Transformer is a strong choice when the main goal is to maximize forecasting accuracy for Mae Moh Hospital. Although PatchTST ranks first, the proposed compact hybrid models remain competitive and stable across folds. In particular, TransGRU and TransLSTM show close results (TransGRU: RMSE = 1.218 $\pm$ 0.783, MAE = 0.793 $\pm$ 0.553; TransLSTM: RMSE = 1.228 $\pm$ 0.762, MAE = 0.798 $\pm$ 0.554). Beyond model comparison, the main contribution of this work is the leakage-aware

feature pipeline and the temporal validation protocol, together with simple robustness and time-window analyses that can be reused with different forecasting models in practice. This study is limited to a single hospital, so the results should not be generalized to all hospitals without additional validation. In future work, we plan to test multi-hospital datasets and explore practical transfer learning, such as fine-tuning a pre-trained model on a new hospital using a small amount of recent data. We also plan to add stronger explainability analyses (e.g., feature attribution and attention-based interpretation) to support real decision-making in hospital energy management.

ACKNOWLEDGEMENT

We sincerely thank the Energy Research and Development Institute Nakornping (ERDI), Chiang Mai University, and Mae Moh Hospital for providing data access, collaboration, and essential support for this study. We also highly appreciate the Electricity Generating Authority of Thailand (EGAT) for the scholarship support. The electricity consumption data from Mae Moh Hospital were provided and published with proper institutional authorization, ensuring that sensitive information was handled responsibly.

REFERENCES

- [1] K. Thampanishvong and B. Limmeechokchai, "Reviews of low carbon scenarios, carbon neutrality, and net zero emissions in thailand: impacts on greenhouse gas emissions and the macroeconomy," *Asian Economic Papers*, vol. 22, no. 3, pp. 24-45, 2023.
- [2] R. Lorm and B. Limmeechokchai, "Thailand net zero emissions 2050: Analyses of decarbonized energy system beyond the ndc," *International Energy Journal*, vol. 24, no. 2, 2024.
- [3] D. Pudjianto, C. Ramsay, and G. Strbac, "Virtual power plant and system integration of distributed energy resources," *IET Renewable power generation*, vol. 1, no. 1, pp. 10-16, 2007.
- [4] S. Ghavidel, L. Li, J. Aghaei, T. Yu, and J. Zhu, "A review on the virtual power plant: Components and operation systems," *2016 IEEE international conference on power system technology (POWERCON)*. IEEE, 2016, pp. 1-6.
- [5] S. Abdelkader, J. Amissah, and O. AbdelRahim, "Virtual power plants: an in-depth analysis of their advancements and importance as crucial players in modern power systems," *Energy, Sustainability and Society*, vol.14, no. 1, p. 52, 2024.
- [6] A. Bagnasco, F. Fresi, M. Saviozzi, F. Silvestro, and A. Vinci, "Electrical consumption forecasting in hospital facilities: An application case," *Energy and Buildings*, vol. 103, pp. 261-270, 2015.
- [7] L. Cao, Y. Li, J. Zhang, Y. Jiang, Y. Han, and J. Wei, "Electrical load prediction of healthcare buildings through single and ensemble learning," *Energy Reports*, vol. 6, pp. 2751-2767, 2020.
- [8] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah, "A review on time series forecasting techniques for building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 74, pp.902-924, 2017.
- [9] A. Koç and S. U. Seçkiner, "Analysing and forecasting the energy consumption of healthcare facilities in the short and medium term: a case study," *Operations Research and Decisions*, vol. 34, no. 3, pp. 165-192, 2024.
- [10] Y. Mizuno, Y. Tanaka, F. Kurokawa, and N. Matsui, "A hospital grid with renewable energy system applied to virtual power plant," *2020 8th International Conference on Smart Grid (icSmartGrid)*. IEEE, 2020, pp. 203-207.
- [11] D. Morinigo-Sotelo, O. Duque-Perez, L. Garcia-Escudero, M. Fernandez-Temprano, P. Fraile-Llorente, M. Riesco-Sanz, A. Zorita-Lamadrid et al., "Short-term hourly load forecasting of a hospital using an artificial neural network," *International Conference on Renewable Energies and Power Quality*, vol. 1, 2011, p. 355.
- [12] J. W. Taylor, "Short-term electricity demand forecasting using double seasonal exponential smoothing," *Journal of the Operational Research Society*, vol. 54, no. 8, pp. 799-805, 2003.
- [13] J. W. Taylor, "Short-term load forecasting with exponentially weighted methods," *IEEE transactions on Power Systems*, vol. 27, no. 1, pp. 458-464, 2011.
- [14] R. Weron, *Modeling and forecasting electricity loads and prices: A statistical approach*. John Wiley & Sons, 2006.
- [15] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914-938, 2016.
- [16] Y. Wei, X. Zhang, Y. Shi, L. Xia, S. Pan, J. Wu, M. Han, and X. Zhao, "A review of data-driven approaches for prediction and classification of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol.82, pp. 1027-1047, 2018.
- [17] M. Bourdeau, X. qiang Zhai, E. Nefzaoui, X. Guo, and P. Chatellier, "Modeling and forecasting building energy consumption: A review of data-driven techniques," *Sustainable Cities and Society*, vol. 48, p. 101533, 2019.
- [18] T. A. Abd'Azeez and L. Olatomiwa, "A machine learning-powered energy consumption prediction system with api," *Journal of Electrical Systems and Information Technology*, vol. 12, no. 1, p. 50, 2025.
- [19] M. S. Mahmud and M. H. Chowdhury, "A smart system for monthly electrical energy consumption prediction using machine learning," *International Journal of Information Engineering and Electronic Business (IJIEEB)*, vol. 16, no. 6, pp.42-61, 2024.
- [20] H. S. Hippert, C. E. Pedreira, and R. C. Souza,

“Neural networks for short-term load forecasting: A review and evaluation,” *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44-55, 2002.

- [21] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *arXiv preprint arXiv:1803.01271*, 2018.
- [22] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, “Deepar: Probabilistic forecasting with autoregressive recurrent networks,” *International journal of forecasting*, vol. 36, no. 3, pp. 1181-1191, 2020.
- [23] B. N. Oreshkin, D. Carpow, N. Chapados, and Y. Bengio, “N-beats: Neural basis expansion analysis for interpretable time series forecasting,” *arXiv preprint arXiv:1905.10437*, 2019.
- [24] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, “Temporal fusion transformers for interpretable multi-horizon time series forecasting,” *International journal of forecasting*, vol. 37, no. 4, pp. 1748-1764, 2021.
- [25] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, “Informer: Beyond efficient transformer for long sequence time-series forecasting,” *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11106-11115.
- [26] H. Wu, J. Xu, J. Wang, and M. Long, “Autoformer: Decomposition transformers with autocorrelation for long-term series forecasting,” *Advances in neural information processing systems*, vol. 34, pp. 22419-22430, 2021.
- [27] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, “Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting,” *International conference on machine learning*. PMLR, 2022, pp. 27268-27286.
- [28] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, “A time series is worth 64 words: Long-term forecasting with transformers,” *arXiv preprint arXiv:2211.14730*, 2022.
- [29] X. Li, Y. Zhong, W. Shang, X. Zhang, B. Shan, and X. Wang, “Total electricity consumption forecasting based on transformer time series models,” *Procedia Computer Science*, vol. 214, pp. 312-320, 2022.
- [30] Y. Liu, R. Zheng, M. Liu, J. Zhu, X. Zhao, and M. Zhang, “Short-term load forecasting model based on time series clustering and transformer in smart grid,” *Electronics*, vol. 14, no. 2, p. 230, 2025.



His research interests include load forecasting, machine learning, and energy management.

Piyapong Boonsompan received the B.Eng. degree in computer engineering from Chiang Mai University, Chiang Mai, Thailand, in 2016. He is currently pursuing the master's degree with the Department of Computer Engineering, Chiang Mai University, Chiang Mai. He has been an IT Engineer with the Electricity Generating Authority of Thailand at Mae Moh Power Plant, Lampang, Thailand, since 2017, where he is responsible for data-center and server infrastructure operations.



He was a Faculty Member with the Department of Electrical Engineering at Srinakharinwirot University, Thailand, from 2010 to 2020. He is currently a faculty member with the Department of Computer Engineering at Chiang Mai University, Thailand. His research interests include wireless communications, the Internet of Things, and physical layer security. He received the ECTI-CON Best Paper Award in 2019 and the Royal Golden Jubilee Scholarship from the Thailand Research Fund for his Ph.D. Program.

Kampol Woradit received the B.Eng. (Hons.) and Ph.D. degrees in electrical engineering from Chulalongkorn University, Thailand, in 2002 and 2010, respectively. He was a Visiting Student with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, from 2007 to 2008, and was on a Postgraduate Research Attachment with the Modulation and Coding Department, Institute for Infocomm Research, A*STAR, Singapore.