Embedding Building Operation Cycles into Transformer Models for Indoor Temperature Prediction

Shundong Li
Department of Civil, Environmental, & Architectural Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
`sli19@wpi.edu`

Nan Ma
Department of Civil, Environmental, & Architectural Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
`nma1@wpi.edu`

Abstract

Reliable forecasting of indoor thermal conditions is essential for optimizing HVAC control, reducing energy consumption, and ensuring occupant comfort in buildings. However, accurate long-term prediction of indoor temperature remains a major challenge due to temporal dependencies and variable building dynamics. In this study, we evaluated a range of models on the Smart Buildings Control Suite benchmark, spanning classical statistical approaches (spline regression, XGBoost) to advanced sequence architectures (encoder–decoder LSTM, PatchTST, Time-LLM). We further developed a domain-adapted variant of PatchTST that integrates Rotary Positional Embeddings (RoPE) aligned with building operation cycles, such as daily and weekly schedules. Results on a six-month validation window showed that classical baselines capture short-term dynamics but fail to maintain consistent accuracy over multi-week windows, while attention-based transformers substantially outperform recurrent and boosted-tree models. Most importantly, our RoPE-augmented PatchTST achieved the lowest mean absolute error (MAE=1.76 °F) in two-week forecasts across the six-month validation period, highlighting the importance of embedding building operation-specific temporal schedules into sequence models. These results indicate that domain-aware transformers can support reliable long-horizon indoor temperature prediction and thermal comfort management, ultimately facilitating more sustainable and energy-efficient building operation.

1 Introduction

Heating, ventilation, and air-conditioning (HVAC) systems account for nearly 40% of building energy consumption worldwide, making them one of the largest drivers of both operational costs and greenhouse gas emissions in the built environment [1]. Reliable forecasting of indoor thermal conditions is therefore critical for supporting intelligent HVAC control strategies that balance energy efficiency with occupant comfort. Long-horizon forecasts [2] are particularly important in demand response, predictive control, and energy scheduling applications, where anticipating indoor temperature

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: UrbanAI: Harnessing Artificial Intelligence for Smart Cities.

trajectories allows systems to proactively adjust setpoints, avoid peak loads, and maintain thermal comfort under dynamic external conditions.

Previous research has explored a wide range of methods for short- and medium-term building thermal forecasting. Classical statistical approaches, such as autoregressive models and ensemble learning techniques, have demonstrated effectiveness over short horizons but generally fail to capture nonlinear building dynamics over longer timescales [3, 4]. More recently, machine learning and deep learning approaches—including recurrent neural networks (RNNs) and long short-term memory (LSTM) architectures have improved predictive accuracy by learning temporal dependencies directly from data [5, 6]. However, most existing studies have focused on short horizons (hours to a few days) and have not systematically evaluated model performance across multi-week forecasting windows, which are critical for building operation and urban energy system planning [7]. Moreover, while generic sequence models can capture temporal patterns, they often neglect domain-specific characteristics such as the strong daily and weekly cycles inherent to building operation.

To address these challenges, we evaluated a diverse suite of models on the Smart Buildings Control Suite [8] benchmark, ranging from spline regression and XGBoost to encoder–decoder LSTM and transformer variants. We further introduced a domain-adapted PatchTST model that integrates Rotary Positional Embeddings (RoPE) aligned with building operation cycles, allowing the model to better represent temporal regularities. Our experiments provide a systematic comparison of statistical, recurrent, and attention-based models for indoor temperature forecasting, while demonstrating that embedding building-specific cycles significantly improves long-horizon predictive performance. This study, therefore, makes two primary contributions: (i) a comprehensive benchmarking of classical and modern sequence models for long-horizon indoor temperature prediction in a standardized building control dataset, and (ii) the introduction of a domain-aware transformer that incorporates building operation cycles into positional encoding. Together, these advances highlight the importance of domain-informed sequence learning for reliable prediction of indoor thermal conditions, with implications for thermal comfort management, energy-efficient HVAC operation, and building automation.

2 Materials and methodology

The experiments are based on the Smart Buildings Control Suite [8], the first open-source benchmark designed to evaluate and scale HVAC control policies across diverse commercial buildings. The dataset contains multiple time series, each associated with a unique identifier corresponding to a physical device. However, the identifiers are not directly stored as simple device labels; rather, they are embedded within composite strings. To analyze the data set in a structured way, we first extracted the underlying device identifiers by parsing these composite keys and isolating the portion preceding the delimiter symbol (the character '@'). This step allowed us to represent each device consistently across the different data sources.

After standardizing the identifiers, we identified 138 unique devices in the exogenous variable dataset (e.g., weather conditions, setpoints, and other influencing factors) and 123 in the temperature measurement dataset. Cross-comparison revealed that 123 devices were common to both sources, forming the primary set for modeling tasks. An additional 15 devices appeared only in the exogenous dataset, contributing input features but lacking associated temperature records, while all devices with temperature measurements also had corresponding exogenous data. This reconciliation established a coherent mapping across modalities, allowing that subsequent modeling focused on the shared subset where both predictive features (exogenous variables) and targets (temperatures) were available.

2.1 Device type analysis and filtering

After reconciling the device identifiers, we investigated the distribution of device types to better understand the nature of the sensing equipment included in the dataset. Each device is labeled with a categorical type code that indicates its function in the building HVAC system. The analysis revealed two distinct categories:

- Type 4 devices (122 in total): These correspond primarily to variable air volume (VAV) units, which regulate airflow and are closely related to local zone conditions such as temperature and carbon dioxide levels.

- Type 6 device (1 in total): This entry was associated with an air handling unit (AHU), representing a centralized component responsible for conditioning and distributing air across multiple zones.

Because the vast majority of target devices in the dataset are VAV units, and only a single device was classified as an AHU, we determined that including the lone AHU in training and evaluation would add heterogeneity without offering meaningful statistical strength. Its operational characteristics differ substantially from the VAV population, making it an outlier in both function and representation.

For this reason, we excluded the single type 6 (AHU) device from both the training and validation sets. This step ensured that our modeling tasks focused on a more homogeneous population of devices, where statistical comparisons and model performance could be evaluated consistently.

2.2 Feature inspection and engineering

To better understand the available signals from the VAV units, we examined the distribution of variable types across VAV units to understand the available signals, identifying common features like airflow, cooling and heating setpoints, and damper commands, along with additional signals such as discharge air temperature, heating valve commands, and CO2 measurements. This confirmed a mostly consistent set of operational variables across devices, with some variation in CO2-related data. Based on this, we engineered features by computing time-based changes in setpoints to capture control adjustments, applying one-hour rolling averages to smooth command signals and sensor readings, and aggregating data to reduce noise while preserving trends. For model training, we built per-device time series data frames combining temperature targets with aligned exogenous variables at five-minute intervals, resulting in a consistent, device-level dataset structured for supervised learning.

2.3 Temporal segmentation and dataset construction

Following feature engineering, we segmented each device's time series into fixed-length windows for supervised learning, testing durations of two weeks (4032 timestamps), one month, and two months. Overlapping segments were used for training to boost sample size, while validation segments remained non-overlapping to prevent data leakage. We filtered out any segments containing zero values in the zone air temperature column, as these were deemed invalid. This cleaning step ensured the dataset reflected realistic indoor conditions. For the two-week window, which became our primary focus due to modeling constraints, this process yielded 1,078 training and 906 validation segments—offering a good trade-off between data volume and the ability to model short-term dynamics effectively.

2.4 Reproducibility and experimental setup

To ensure consistent and reproducible results across experiments, we fixed the random seed to a common value (42) across all libraries and frameworks involved in data processing and model training. Specifically, the seed was set for Python's built-in random number generator, NumPy, PyTorch on both CPU and GPU, and CUDA operations. We further enforced deterministic behavior in PyTorch by disabling nondeterministic optimizations and setting the PYTHONHASHSEED environment variable. Furthermore, the same seed was applied to the PyTorch random number generator object used during data loading and batching. This standardized seeding procedure reduced the influence of randomness in weight initialization, data shuffling, and numerical kernels, enabling fair comparisons between different model architectures.

3 Results and discussion on model selection

3.1 Model testing and selection

We evaluated five models spanning a spectrum from simple statistical learners to state-of-the-art sequence models. This progression allowed us to benchmark the performance of traditional approaches against recent advances in time-series deep learning. The models were chosen to cover both lightweight methods (Spline regression, XGBoost) and more expressive architectures (Encoder–Decoder LSTM, PatchTST, Time-LLM). By analyzing this diverse set, we aimed to assess the trade-off between model complexity, interpretability, and predictive accuracy.

For model training, we standardized the experimental setup by fixing the maximum number of epochs to 200 and applying early stopping based on validation mean absolute error (MAE). Specifically, we employed patience thresholds of 10 and 20 epochs, meaning training was terminated if validation performance failed to improve within the specified patience window. Exceptions were made for spline regression, which does not involve iterative training, and for Time-LLM, where the computational cost was prohibitively high; in the latter case, reduced patience thresholds of 3 and 5 epochs were used. Across all deep learning models (LSTM, PatchTST and Time-LLM), the learning rate was set to 1×10^{-3}, a common default that balances convergence speed with stability. We used the Adam optimizer and mean squared error (MSE) loss for the deep learning models. For all models trained iteratively, the best checkpoint was selected according to the lowest validation MAE achieved during training.

Spline Regression Spline regression was selected as a baseline because of its ability to model smooth, nonlinear relationships between input and output variables. By fitting piecewise polynomial curves with continuity constraints, splines can capture gradual changes and seasonal variations in building dynamics while remaining computationally lightweight. Although limited in handling sequential dependencies across long horizons, splines provide a simple and interpretable benchmark. We used the SplineTransformer available in scikit-learn [9] to map standardized input features into a higher-dimensional spline basis, followed by a linear regressor trained with stochastic gradient descent (SGD). The training loop was implemented in a mini-batch style with early stopping, enabling the model to handle large time-series datasets efficiently. A cubic spline basis (degree=3) with 20 knots was used in our default configuration.

XGBoost Gradient-boosted decision trees, implemented through XGBoost [10], offer a more flexible nonlinear baseline that can handle complex feature interactions. By training on exogenous variables to predict temperature at each timestamp, our approach utilizes boosting to correct errors iteratively, yielding robust short-term predictions. Its strengths lie in handling heterogeneous feature types, resistance to overfitting through regularization, and efficiency on tabular datasets. However, as a point-wise predictor, it lacks the capacity to model temporal dependencies explicitly. We configured XGBoost with 1,000 boosting rounds and an early stopping patience of 50 rounds, using a learning rate of 0.05 and a maximum tree depth of 6 to balance flexibility with generalization. To prevent overfitting, we applied row ($subsample = 0.8$) and column ($colsample\_bytree = 0.8$) sampling at each iteration, coupled with built-in L2 regularization.

Encoder–Decoder LSTM Long Short-Term Memory (LSTM) networks [11] are a well-established recurrent architecture for sequential modeling. We adopted an encoder–decoder LSTM to perform auto-regressive prediction, where past temperature sequences and exogenous variables are encoded into a latent state and decoded into future temperature values. This design explicitly leverages temporal correlations, enabling the model to capture delayed effects of control signals or weather disturbances. The encoder–decoder LSTM was trained with a hidden dimension of 128, two recurrent layers, and a dropout of 0.2, striking a balance between model capacity and regularization. In addition, we varied the use of teacher forcing, comparing runs where the decoder was conditioned on previous ground-truth values against purely autoregressive rollouts using its own predictions. Teacher forcing often led to faster convergence during training, while autoregressive decoding provided a more realistic evaluation of deployment performance.

PatchTST PatchTST [12] represents a recent advance in transformer-based sequence modeling tailored for time series. Instead of processing data point by point, the model segments sequences into patches, which are then embedded and passed through attention layers. This design captures both local and long-range dependencies efficiently, making it well-suited for multi-week forecasting tasks in building environments. By leveraging attention, PatchTST can balance contributions from multiple signals (e.g., setpoints, airflow, CO_2 concentration) and adaptively learn which inputs matter most at different horizons. Recent work has already begun to explore the suitability of PatchTST-style models in building energy applications. In our baseline PatchTST implementation, each patch consisted of a fixed temporal window (48 steps) that was linearly embedded into a hidden dimension of 512, followed by a two-layer transformer encoder with four attention heads. Positional encodings were added to maintain temporal ordering, and the decoder reconstructed the full sequence by overlap-adding predicted patches. Dropout (0.2) was applied throughout to mitigate overfitting.

Time-LLM Finally, we evaluated Time-LLM [13], a large language model (LLM)–augmented architecture that integrates foundation models with time-series forecasting. The key motivation is to leverage pretrained representations and attention mechanisms to capture complex temporal patterns and cross-variable relationships beyond what classical time-series models achieve. For building dynamics, this could mean recognizing higher-order structures, such as recurring daily cycles, seasonal shifts, or nonlinear responses to setpoint changes. While computationally heavy, Time-LLM provides a cutting-edge reference point for exploring how language-model architectures could transfer to structured temporal prediction tasks. We evaluated Time-LLM using three different Llama3[14] backbones (Llama-3.1-8B, Llama-3.2-3B and Llama-3.2-1B). Due to Colab T4 GPU VRAM constraints, we only managed to implement Llama-3.2-1B as the LLM backbone. For time-LLM configurations, the model used a compact embedding dimension ($d_{\text{model}} = 16$), 4 attention heads, 2 encoder layers, and 1 decoder layer, with a feedforward size of 64, dropout of 0.1, GELU activation, and temporal embeddings set to "timeF". Patch-level processing employed a patch length of 16, and stride of 8 to capture local temporal structure. The LLM integration relied on Llama-3.2-1B with a hidden size of 2048 and 6 transformer layers, allowing the model to align high-level temporal reasoning with exogenous features. Training was performed in mini-batches of size 4, with evaluation also at batch size 4,

Table 1: Validation performance (MAE) of tested models on the two-week dataset.

Model	**Validation MAE**
Spline	15.380
XGBoost	8.222
Encoder–Decoder LSTM	9.940
PatchTST	**3.853**
Time-LLM	8.892

The results shown in Table 1 highlight a clear performance gap between classical baselines and modern sequence models. Spline regression performed poorest, as expected, since it cannot capture sequential dependencies or device-specific dynamics. XGBoost provided a substantially stronger baseline by modeling nonlinear interactions among exogenous variables, though it remained limited by its pointwise prediction strategy. The encoder–decoder LSTM demonstrated modest gains from temporal modeling but struggled to maintain accuracy across the long two-week horizon. In contrast, PatchTST achieved the best performance by a wide margin, confirming the advantage of attention-based architectures in capturing both local fluctuations and long-range dependencies in building temperature dynamics. Interestingly, Time-LLM, while representing a cutting-edge approach, underperformed PatchTST despite its computational cost, suggesting that specialized time-series transformers may currently be better suited to this task than the pretrained LLM augmented approach.

4 PatchTST hyperparameter selection and domain adaptation with RoPE

In our implementation of PatchTST, every patch was flattened and linearly projected into a hidden representation, after which temporal order was encoded through positional information. The sequence of patch embeddings was then processed by a Transformer encoder stack with multi-head self-attention and feedforward submodules. Finally, a linear output head produced patch-level forecasts, which were reassembled into a continuous prediction sequence using an overlap-add procedure. Dropout regularization was applied both at the embedding stage and within the encoder to mitigate overfitting and improve generalization.

4.1 Hyperparameter selection

We performed hyperparameter exploration by testing different combinations of patch length (24, 48, 96), stride (half or a quarter of the corresponding patch length), hidden dimension (128, 256, 512), and dropout rate (0.0, 0.1, 0.2), using validation MAE as the selection criterion. The search was conducted using mostly the same training setting from the model selection part, except the patience window was set 5 to balance thoroughness and computational feasibility. For each hyperparameter setting, we trained the model with three different seeds (0,1,2) to calculate average validation MAE and standard deviation of validation MAE. The full set of parameter search results is summarized in Table 4. Overall, larger patch lengths generally improved predictive stability, with hidden dimensions of 512 providing the best trade-off between capacity and efficiency. Among the tested configurations, the best-performing model used a patch length of 48 timesteps, stride of 12, hidden dimension of 512, and dropout of 0.1, achieving an average validation MAE of 2.093 with a standard deviation of 0.056. This configuration was selected as the final PatchTST model hyperparameter for further configurations.

Having fixed the architectural backbone, we next examined how the model responds to variations in training features and loss functions. Across these experiments, MSE provided the most stable optimization and best validation performance. The lowest MAE (1.945) shown in Table 2 was achieved when combining engineered features with MSE, confirming the value of both feature engineering and the choice of loss. In contrast, using L1 or Huber loss generally degraded performance, particularly when applied to the engineered feature set.

Table 2: Validation MAE under different feature sets and training loss functions (PatchTST, two-week dataset).

Feature Set	Loss Function	Val MAE
Original	MSE	2.119
Engineered	MSE	**1.945**
Original	L1	2.347
Engineered	L1	2.506
Original	Huber	2.134
Engineered	Huber	2.404

4.2 RoPE integration

Transformers rely on positional encodings to provide a sense of order to otherwise permutation-invariant attention mechanisms. Rotary Positional Embeddings (RoPE) [15] achieve this by rotating query and key vectors in the complex plane according to their position index, enabling attention to capture relative positional information in addition to absolute order. Unlike fixed sinusoidal encodings, RoPE introduces position-dependent rotations directly into the dot-product attention step, which has been shown to improve extrapolation and long-range sequence modeling.

In our implementation, we augmented the PatchTST architecture with RoPE by applying rotation to the query and key projections of each attention head. This required building cosine and sine caches parameterized by a "rope base" value, which controls the frequency spectrum of positional encodings. Larger bases stretch positional sensitivity across longer contexts, while smaller bases compress it into finer-grained periodic cycles.

To explore both general-purpose and domain-specific configurations, we tested three choices for the rope base parameter:

- 10,000 — the default value used in Llama3 and other large language models. This serves as a standard benchmark, reflecting how RoPE is commonly deployed in NLP.
- 288 — corresponding to the number of 5-minute intervals in a single day (12×24). This setting directly encodes daily periodicity, aligning the positional embedding frequencies with diurnal cycles that dominate indoor environmental conditions.
- 2016 — corresponding to the number of 5-minute intervals in a week ($12 \times 24 \times 7$). This captures weekly rhythms, such as work-week occupancy patterns and weekend fluctuations, which are highly relevant in commercial buildings.

By aligning rope base values with natural temporal cycles in the domain, the model is encouraged to attend to regularities at daily and weekly horizons while still retaining flexibility from the standard large-base configuration. This approach allows RoPE to encode both general long-term dependencies and building-specific periodic structures that drive indoor temperature dynamics.

4.3 Evaluating RoPE base across scenarios

The results in Table 3 indicate that aligning RoPE with domain-specific cycles can be beneficial. For both feature settings, a weekly cycle (2016) yielded the lowest average validation MAE, outperforming

Table 3: Validation performance of PatchTST with RoPE under different base parameters and feature sets.

RoPE Base	Feature Set	MAE (avg)	MAE (std)
10000	Original (v2)	1.860	1.640
288	Original (v2)	1.919	1.796
2016	Original (v2)	**1.763**	1.560
10000	Engineered (v3)	1.863	1.689
288	Engineered (v3)	1.925	1.696
2016	Engineered (v3)	1.827	1.591

both the Llama3 default (10,000) and the daily-aligned base (288). This suggests that weekly rhythms in occupancy and HVAC operation are especially salient in building dynamics.

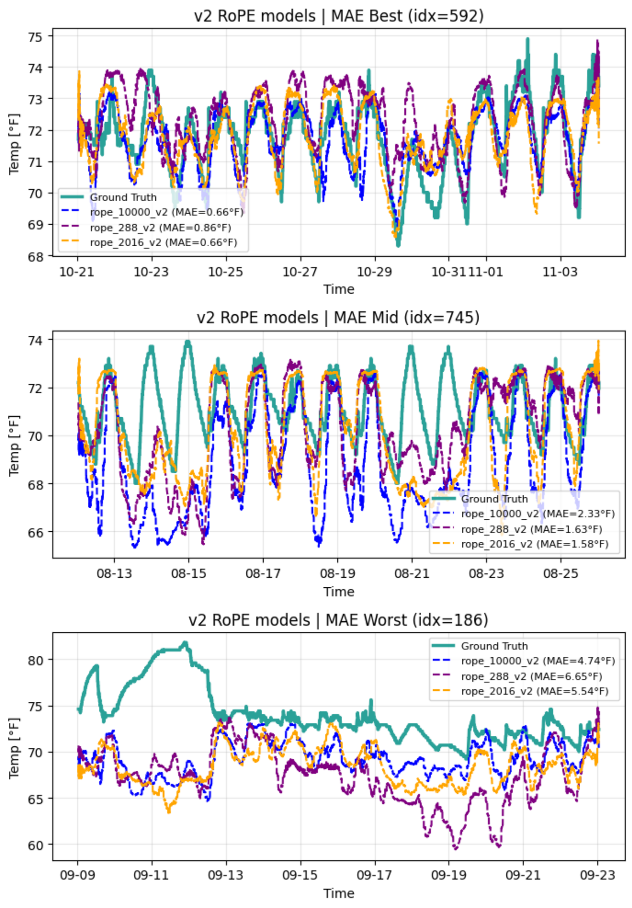

Figure 1: Best, median, and worst-case validation predictions for the PatchTST model with RoPE base = 2016. The plots compare predicted base = 2016 (dashed yellow), base = 288 (dashed purple), base = 10,000 (dashed blue) and ground-truth (teal) temperature trajectories, with MAE shown for each segment.

To complement the aggregated MAE results, we further examine case studies of the best, median, and worst validation sequences using the top-performing PatchTST model (RoPE base = 2016 with the original feature set). These examples shown in Figure 1 illustrate how the model behaves under both typical and challenging conditions.

In the best-case sequence, RoPE bases of 10,000 and 2016 achieved identical MAE (0.66). However, the base = 2016 model better captured the overall trajectory, while the base = 10,000 model showed mismatches during the period from 10-27 to 10-29. Notably, both models under-predicted the double peak between 11-01 and 11-03, whereas the base = 288 model came closest to matching the true peak values. This improvement came at a cost: the base = 288 model consistently over-predicted peaks in the preceding days, leading to the highest overall MAE among the three bases.

In the median sequence, the ground truth displayed a repeating pattern of 2 consecutive days, 5 days, and then 2 days—resembling a workday–weekend structure. All three models struggled with the shorter "weekend" segments, but the domain-specific bases (2016 and 288) more accurately captured the five-day "workday" peaks than the base = 10,000 model.

The worst-case sequence highlights a systematic underestimation of temperature levels during the first two days across all models, likely reflecting atypical operating conditions or shifts underrepresented in training. This case underscores a limitation of the approach: while RoPE-based PatchTST captures dominant periodic patterns well, it remains vulnerable to rare dynamics.

5 Conclusions

Our experiments demonstrate that modern sequence models substantially outperform classical approaches in modeling building temperature dynamics. Among the baselines, spline regression was unable to capture temporal dependencies, and XGBoost—though stronger—remained constrained by its point-wise prediction strategy. Sequence models provided notable improvements, with encoder–decoder LSTM achieving moderate gains but struggling over long horizons. The attention-based PatchTST achieved the best performance by a wide margin, reducing validation MAE to 3.853 on the two-week dataset. This confirms the value of attention mechanisms in learning both local fluctuations and long-range dependencies.

Performance improved further with careful tuning. The best results (MAE 1.945) came from models trained with MSE loss and augmented by engineered features, underscoring the importance of both representation and objective alignment. Incorporating Rotary Positional Embeddings (RoPE) revealed another key insight: aligning positional encoding with domain-relevant cycles improved prediction stability. A weekly RoPE base (2016) consistently outperformed both a daily-aligned base and the standard setting, suggesting that weekly rhythms—such as occupancy and HVAC schedules—play a major role in building thermal behavior. This configuration achieved the lowest overall MAE (1.763) across the full six-month validation period segmented into two-weeks chunks.

Despite these promising results, several limitations remain. The seasonal split of training (January–June) and validation (July–December) introduces an imbalance in heating versus cooling season dynamics, which may bias models toward the operating conditions most prevalent during training. Additionally, the study did not incorporate spatial information such as floor plans or thermal zoning, which could provide critical context for heat transfer and occupancy-driven variation. Finally, while PatchTST with RoPE proved highly effective over a two-week horizon, its robustness over the full six-month validation window and across atypical operating regimes remains an open challenge.

References

[1] M González-Torres, Luis Pérez-Lombard, Juan F Coronel, Ismael R Maestre, and Da Yan. A review on buildings energy information: Trends, end-uses, fuels and drivers. *Energy Reports*, 8: 626–637, 2022.

[2] Abdul Afram and Farrokh Janabi-Sharifi. Theory and applications of hvac control systems–a review of model predictive control (mpc). *Building and environment*, 72:343–355, 2014.

[3] Liang Zhang. Data-driven building energy modeling with feature selection and active learning for data predictive control. *Energy and Buildings*, 252:111436, 2021.

[4] Qingrui Jiang, Chenyu Huang, Zhiqiang Wu, Jiawei Yao, Jinyu Wang, Xiaochang Liu, and Renlu Qiao. Predicting building energy consumption in urban neighborhoods using machine learning algorithms. *Frontiers of Urban and Rural Planning*, 2(1):6, 2024.

[5] Saeed Murtaza, Sarath Raj, Geun Young Yun, Duk-Joon Park, Ji-Hye Kim, Gwanyong Park, and Jin Woo Moon. Adaptive neural temporal hybridization for missing data imputation in building energy use datasets: An integrated lnn-lstm weighted model. *Journal of Building Engineering*, page 113774, 2025.

[6] Cunhua Zhu and Shuyuan Zhang. Enhanced learning model of lstm-gan hybrid network in the dynamic prediction of building energy consumption. *International Journal of High Speed Electronics and Systems*, page 2550008, 2025.

[7] Amin Mirakhorli and Bing Dong. Occupancy behavior based model predictive control for building indoor climate—a critical review. *Energy and Buildings*, 129:499–513, 2016.

[8] Judah Goldfeder, Victoria Dean, Zixin Jiang, Xuezheng Wang, Bing dong, Hod Lipson, and John Sipple. The smart buildings control suite: A diverse open source benchmark to evaluate and scale hvac control policies for sustainability, 2025. URL `https://arxiv.org/abs/2410.03756`.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[10] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[12] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[13] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[14] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[15] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

A Supplementary Material

patch_len	stride	hidden_dim	dropout	avg_val_mae	std_val_mae
48	12	512	0.1	2.093	0.056
24	6	512	0.1	2.112	0.043
24	12	512	0.2	2.117	0.053
24	6	512	0.2	2.126	0.028
24	12	512	0.1	2.182	0.167
48	24	512	0.2	2.188	0.074
24	6	256	0.2	2.227	0.061
24	12	256	0.2	2.229	0.052
24	12	256	0.1	2.232	0.105
96	24	128	0.2	2.251	0.270
48	24	256	0.2	2.253	0.131
96	48	128	0.1	2.261	0.152
48	12	256	0.2	2.269	0.094
48	12	512	0.2	2.280	0.161
96	24	256	0.2	2.287	0.254
96	24	512	0.1	2.317	0.125
24	6	256	0.1	2.321	0.058
24	6	128	0.1	2.326	0.141
96	24	128	0.1	2.327	0.045
96	48	256	0.2	2.340	0.043
96	48	256	0.1	2.354	0.067
96	24	512	0.2	2.356	0.028
48	24	256	0.1	2.359	0.032
24	6	128	0.2	2.392	0.151
48	24	512	0.1	2.396	0.149
24	12	128	0.2	2.406	0.161
48	12	128	0.1	2.453	0.303
48	24	128	0.2	2.464	0.058
48	24	128	0.1	2.465	0.064
48	12	128	0.2	2.466	0.033
24	12	128	0.1	2.504	0.175
96	48	512	0.1	2.514	0.258
48	12	256	0.1	2.522	0.114
96	48	512	0.2	2.542	0.119
96	24	256	0.1	2.683	0.604
96	48	128	0.2	2.766	0.506
96	24	128	0	3.138	0.598
48	12	128	0	3.304	0.604
24	6	256	0	3.347	0.818
48	24	256	0	3.425	0.535
48	12	256	0	3.429	0.516
96	24	256	0	3.438	0.411
24	12	256	0	3.442	0.677
96	48	256	0	3.465	0.571
96	48	128	0	3.472	0.660
24	6	128	0	3.518	0.616
96	24	512	0	3.836	0.061
48	24	512	0	3.871	0.011
48	24	128	0	3.872	0.104
24	12	512	0	3.875	0.006
24	6	512	0	3.878	0.001
96	48	512	0	3.883	0.008
48	12	512	0	3.885	0.010
24	12	128	0	3.957	0.035

Table 4: PatchTST hyperparameter search results across tested configurations.

NeurIPS Paper Checklist

1. **Claims**

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the scope of the contribution (evaluation of baselines and introduction of domain-aware PatchTST with RoPE) and the main empirical claim (improved MAE over baselines on the Smart Buildings Control Suite). These claims are supported by experiments and results.

2. **Limitations**

   Answer: [Yes]

   Justification: The last paragraph of the "Conclusions" section discusses the seasonal imbalance between training and validation periods, lack of spatial descriptors, and challenges with rare dynamics from the training data, making clear the boundaries of the claims.

3. **Theory assumptions and proofs**

   Answer: [NA]

   Justification: The paper is empirical and does not introduce new theoretical results or proofs.

4. **Experimental result reproducibility**

   Answer: [Yes]

   Justification: The paper describes preprocessing steps, dataset splits, feature engineering, hyperparameter choices, and training details. Results can be reproduced given these descriptions, independent of the provided code.

5. **Open access to data and code**

   Answer: [Yes]

   Justification: The experiments rely on the publicly available Smart Buildings Control Suite dataset [8]. We also provide code and scripts to reproduce experiments in the supplementary material.

6. **Experimental setting/details**

   Answer: [Yes]

   Justification: Training/validation splits, hyperparameters, optimizer settings, and early stopping criteria are all described in detail in the paper.

7. **Experiment statistical significance**

   Answer: [Yes]

   Justification: Validation performance is reported with averages and standard deviations over multiple runs. Error bars quantify variability across runs, consistent with NeurIPS guidelines.

8. **Experiments compute resources**

   Answer: [Yes]

   Justification: The paper specifies training was conducted on Google Colab T4 GPUs with 16 GB VRAM. Compute limits constrained larger LLM backbones, which is noted explicitly.

9. **Code of ethics**

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. Data is publicly released under an open license, no personal identifiers are present, and no unethical practices were involved.

10. **Broader impacts**

    Answer: [Yes]

    Justification: The paper discusses the potential positive societal impact of improved energy efficiency and smarter HVAC control, as well as limitations and risks of miscalibration or misuse in automated building management.

11. **Safeguards**

    Answer: [NA]

    Justification: The models developed pose low risk of misuse and no safeguards beyond standard open-source release are required.

12. **Licenses for existing assets**

    Answer: [Yes]

    Justification: The Smart Buildings Control Suite dataset is properly cited, and its open-source status and terms of use are acknowledged. All other used software (scikit-learn, PyTorch, XGBoost) are cited with appropriate licenses.

13. **New assets**

    Answer: [NA]

    Justification: The paper does not release new datasets or pretrained models, only derived scripts/configurations.

14. **Crowdsourcing and research with human subjects**

    Answer: [NA]

    Justification: No human subjects or crowdsourcing were involved in this work.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Answer: [NA]

    Justification: No human subjects research was conducted.

16. **Declaration of LLM usage**

    Answer: [Yes]

    Justification: Time-LLM experiments rely on an LLM backbone (Llama-3.2-1B), which is explicitly described in the methods section, with configuration and training details.