Neural Network Architectures for Building Energy Forecasting: A Head-to-Head Benchmark Against LightGBM with Sequence Length Optimisation

Madhukar Dharavath
Independent Researcher, USA
reachmadhukard@gmail.com

S. Sri Nandhini Kowsalya
Assistant Professor, Department of Information Technology,
Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College,
Chennai, Tamil Nadu, India
srinandhinikowsalya@gmail.com

M Kiran Kumar
Assistant Professor, Department of CSE- Data Science
Madanapalle Institute of Technology & Science, Madanapalle, Andhra Pradesh, India
kiran9sai@gmail.com

Syed Riyaz Ahammed
NITTE (Deemed to be University), NMAM Institute of Technology, *Department of Electronics & Communication Engineering*
Karnataka, India
Syedriyaz.global@gmail.com

G. Soniya Priyatharsini
School of Engineering & IT
Manipal Academy of Higher Education,
Dubai, UAE
soniya.george@manipaldubai.com

Sushma Sunkollu Nagaraj
Solutions Architect, USA
snsushma94@gmail.com

Vikas Marakini
The Research Organization of Science and Technology,
Ritsumeikan University,
Kusatsu-shi, Shiga, Japan
vikasbhat02@gmail.com

Abstract*— It is a common belief that deep learning beats traditional machine learning for time series forecasting — we test this assumption directly for building energy prediction in this paper. Three neural network architectures (LSTM-style, GRU-style and Deep-MLP) were tested over three different lookback windows (24h, 48h, 168h) on a dataset consisting of 8,760 hourly consumption observations made by a research facility in Porto, Portugal; they were then benchmarked head-to-head with gradient boosted LightGBM on the same dataset and test split. The best performing neural network was a Deep-MLP, with an R^2 = 0.6255 and RMSE = 701.56 kW (when the lookback is set to 48 hours). LightGBM had R^2 = 0.9928 and RMSE = 99.67 kW — that is, a results of 7 times smaller error and higher R^2 by +36 percentage points! The length of the sequence analysis revealed an unexpected result, shorter 24 hour lookback windows have better generalisation than the rewards from a 168 hour context specifically (R^2 = 0.5879 vs 0.4090) indicating that more temporal history is perhaps detrimental to this form of data. These results suggest a useful rule-of-thumb: for datasets with less than 10,000 samples and well-engineered tabular features, gradient boosting consistently obtains higher accuracy than deep learning, and increasing the complexity of the neural network architecture adds no gain in accuracy.

I. INTRODUCTION

So the question is no longer if deep learning can improve forecasting in image recognition, natural language processing and speech — only whether those improvements translate to building energy time series (a legitimately open question)[1]. Gradient-boosted trees have established state-of-the-art in tabular energy datasets, and the notion that neural networks will inevitably beat them has yet to be strictly tested on small-to-medium scale building deployments [2. The practical stakes are real. Real-time energy procurement, HVAC optimisation and demand response scheduling are driven by building energy forecasting [3]. The problem with making the wrong algorithm choice is that you either deploy a very complicated neural network which under-performs against a simpler alternative, or you are too quick to box away deep learning in contexts where it can actually assist [4]. The size of the dataset, for individual building deployments — usually 8k-50k hourly observations — lies in a grey area where neither method enjoys a clear theoretical edge [5]. Using the same dataset, features, and the evaluation protocol across all models, this paper answers that question empirically [6].

Gradient boosting methods — especially LightGBM and XGBoost — have emerged as the default choice for tabular building energy forecasting which deal with mixed data types and non-linear variable interactions quite well[7]. Newer comparative studies note R^2 0.99 at go for hourly consumption prediction of commercial building using well engineered lags and rolling window features [8]. These results establish a baseline that neural networks have to achieve in order to justify the additional complexity they bring [9].

Neural networks are all built from layers of different architectures, and until this point recurrent architectures have dominated for energy time series. To overcome the vanishing gradient problem in long sequences, LSTM networks have been proposed for various domains and shown improvement on ARIMA and regression baselines using residential load forecasting [10]. Attention mechanisms and encoder-decoder extensions have been used to obtain longer ranges of temporal dependencies [11], while CNN-LSTM hybrids provide multi-step forecasting ability [12]. GRU architectures are a lighter option than LSTM, providing fewer parameters and faster training times while still delivering comparable accuracy on smart meter data [13]. Deep MLP configurations with sophisticated super-vised temporal features have also performed well for some tasks [14].

Nonetheless, literature have recurrently not included a controlled (head-to-head) comparison of these neural approaches and leading gradient boosting approaches, under equivalent settings. For neural networks to learn meaningful

internal representations, they typically require >100,000 samples [15, whereas individual buildings datasets rarely exceed 10K observations. [16] Tree-based methods avoid deployment friction stemming from training instability due to hyperparameter sensitivity. Perhaps sequence length choice — either 24h, 48h, or weekly lookback -- has rarely been analysed systematically or consistently with different building types showing different optimal windows [17-18] leading to a lack of useful guidelines for practitioners to enable even more precise design.

1.1 Main Contributions

This this paper contributes five aspects to the building energy forecasting methodology:

Architecture benchmarking: Three neural network configurations (LSTM-style. GRU-style, Deep-MLP) are evaluated on the same 8,760-hour dataset with exactly the same preprocessing and evaluation in order to factor out real architectural differences from experimental noise.

On Sequence Lengths: 3 lookback windows (24h, 48h, 168h) are compared and yield a counterintuitive result — shorter is better! (24h outperforms 168h on R^2(0.5879 vs 0.4090) and RMSE (746.8 vs 873.0 kW)).

Head-to-head benchmark is showing that the best neural network (Deep-MLP) has a 7-fold RMSE gap (701.56 kW vs 99.67 kW) and 36.7pp R^2 difference (0.6255 vs 0.9928), compared to LightGBM on the same train-test splits.

Selection guidelines are in practical terms: Results have a data-driven – if/then rule for practitioners — gradient boosting for small datasets that include engineered tabular features (> 10,000 samples); neural networks can be used on large-scale raw sequential problems (if using sequential data and automatic feature extraction is beneficial).

Documenting computational cost: Neural networks take an estimated 6× longer to train than LightGBM but achieve significantly poorer accuracy — a trade off between resources used versus high availability real time inference required for operational forecasting systems.

II. METHODOLOGY

This study examines three neural network architectures for hourly building-specific energy consumption forecasting using a systematic experimental framework. Methodology. This methodology uses an implementation of scikit-learn build on top, specifically employing MLPRegressor to simulate LSTM-style, GRU-style and Deep-MLP configurations to allow for a controlled way of comparing architectures without adding the complication of deeper learning frameworks. All experiments support identical pre-processing pipelines, temporal train-test splits, and evaluation metrics to control for performance differences due to architectural design over experimental artifacts.

2.1 Dataset and Feature Engineering

The dataset consists of 8,760 hourly electricity consumption observations from a research facility in Porto, Portugal for the entire year [12]. Input features (27 VIF-optimised variables, derived from a companion gradient boosting study) — lagged consumption values (t-1, t-24, t-168), 24-hour rolling means, temporal encodings (hour-of-day, day-of-week, month), meteorological proxies. Choosing the same feature set as the tree-based benchmark ensures that any differences in performance are due to algorithmic capability (other than choices made in feature engineering).

Data is partitioned in chronological order — 70% train (6,132 hours, Date 1–Date 2), 15% val (1,314 hours, Date3–early Nov), and 15% test (1,314 hours, Nov-Dec). You do not apply shuffling at any of the stages. All preprocessing parameters are fit only on training data, applying unchanged to validation and test sets.

.

2.2 Neural Network Architectures

Here you can find three configurations, each approximating a general design philosophy. LSTM-style approximates recurrent hierarchical feature extraction with feedforward structure using two hidden layers (100–50 neurons, ReLU). GRU-style with Tanh — reduces capacity to 80–40 neurons at each time step, using the Tanh activation function as it is lighter (albeit sacrificing a bit of expressiveness) and trains faster. Deep-MLP — three hidden layers (120–60–30 neurons, ReLU), favoring depth over recurrent processing style.

We can summarize that all three extract fixed-length sequences from sensor signals starting with raw features using the rolling-window construction. Each training example itself flattens together L consecutive hourly feature vectors into a single input dimensioned 27×L, predicting the consumption at time t given features from times t−L up to time t−1.

2.3 Training Configuration

Models are trained with Adam optimiser (α = 0.001, β_1 = 0.9, β_2 = 0.999) for up to 200 epochs and early stopping — when the validation loss does not improve over four consecutive epochs training stops immediately. The input features and target consumption values are normalized to zero mean and unit variance, based only on the statistics of the training-set. Inverse-transformed to kilowatt scale for evaluation. Random seed = 42 throughout.

2.4 Experimental Design

Experiments are structured by four research questions. Once we find the best neural net configuration at 48-hour lookback for all three architectures in RQ1. RQ2: determining the optimal sequence length via various lookback windows (24h, 48h, 168h) on RQ1 winner. RQ3 logs training time and inference latency over all configurations. For RQ4, we had the best neural network compared directly with same LightGBM from the companion study on the same 1,314-hour test set — same features, split and metrics (RMSE, MAE, MAPE, R^2).

III. MATHEMATICAL MODELLING

3.1 Sequence Construction and Feature Representation

Raw hourly observations are converted into supervised learning samples through a sliding window. Each input-output pair is:

$$X_i = [x_{i-L}, x_{i-L+1}, \dots, x_{i-1}] \in \mathbb{R}^{L\times d} \quad (1)$$

$$y_i = c_i \in \mathbb{R} \quad (2)$$

where x_t is the 27-dimensional feature vector at time t, L is the lookback window (24, 48, or 168 hours), and c_i is actual consumption at time i. X_i is flattened to a vector of

dimension L×27 for feedforward processing. Training set size after windowing is M = N_train − L — for L = 48, this gives 6,084 training sequences from 6,132 hours.

3.2 Neural Network Architecture Formulations

All three architectures follow the general feedforward form $h^{(l)} = f(W^{(l)}h^{(l-1)} + b^{(l)})$ with output $\hat{y} = W^{(out)}h^{(L_depth)} + b^{(out)}$. The three configurations differ in depth, width, and activation:

3.2.1 LSTM-Style Architecture

The LSTM-style configuration employs two hidden layers with 100 and 50 neurons respectively, both using ReLU activation. Following neural network design principles for time series [4], the architecture specification is:

$$h^{(1)} = f_{\text{ReLU}}\left(W^{(1)}\tilde{X} + b^{(1)}\right), \quad W^{(1)} \in \mathbb{R}^{100\times(L\times 27)} \tag{3}$$

$$h^{(2)} = f_{ReLU}\left(W^{(2)}h^{(1)} + b^{(2)}\right), \quad W^{(2)} \in \mathbb{R}^{50\times 100} \tag{4}$$

Parameters: (L×27×100) + (100×50) + 151 — for L=48: 134,801 parameters.

3.2.2 GRU-Style Architecture

40 neuron hyperbolic tangent activation architecture, sampling computationally cheap options. The formulation is similar to recurrent network approximation methods [5]:

$$h^{(1)} = f_{\text{Tanh}}\left(W^{(1)}\tilde{X} + b^{(1)}\right), \quad W^{(1)} \in \mathbb{R}^{80\times(L\times 27)} \tag{5}$$

$$h^{(2)} = f_{\text{Tanh}}\left(W^{(2)}h^{(1)} + b^{(2)}\right), \quad W^{(2)} \in \mathbb{R}^{40\times 80} \tag{6}$$

Parameters: (L×2160) + 3,361 — for L=48: 107,041 parameters (~20% fewer than LSTM-style).

3.2.3 Deep-MLP Architecture

The Deep-MLP configuration expands to three hidden layers (120-60-30 neurons) using ReLU activation, aiming for hierarchical feature abstraction. Layer-wise PropagationFollowing principles for deep feedforward networks [3], the propagation is::

$$h^{(1)} = f_{\text{ReLU}}\left(W^{(1)}\tilde{X} + b^{(1)}\right), \quad W^{(1)} \in \mathbb{R}^{120\times(L\times 27)} \tag{7}$$

$$h^{(2)} = f_{\text{ReLU}}\left(W^{(2)}h^{(1)} + b^{(2)}\right), \quad W^{(2)} \in \mathbb{R}^{60\times 120} \tag{8}$$

$$h^{(3)} = f_{\text{ReLU}}\left(W^{(3)}h^{(2)} + b^{(3)}\right), \quad W^{(3)} \in \mathbb{R}^{30\times 60} \tag{9}$$

Parameters: (L×3240) + 10,241 — for L=48: 165,761 parameters (largest of three)

3.4 Optimization and Loss Function

All models minimise mean squared error over M training sequences:

$$\mathcal{L}(\theta) = \frac{1}{M}\sum_{i=1}^{M}\left(\tilde{y}_i - \hat{\tilde{y}}_i(\theta)\right)^2 \tag{10}$$

Adam optimiser updates parameters using bias-corrected first and second moment estimates:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \tag{11}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \tag{12}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{13}$$

$$\theta_{t+1} = \theta_t - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{14}$$

where $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$. Training runs for up to 200 epochs with early stopping after 10 epochs without validation improvement.

3.5 Comparative Analysis Framework

Direct comparison against LightGBM uses a relative performance metric:

$$\Delta_{\text{metric}} = \frac{\text{Metric}_{\text{DL}} - \text{Metric}_{\text{Tree}}}{\text{Metric}_{\text{Tree}}} \times 100 \tag{15}$$

where - Δ_{metric} - Relative performance change [%] - $\text{Metric}_{\text{DL}}$ - Deep learning metric value (RMSE, MAE, MAPE, or R^2) (various units) - $\text{Metric}_{\text{Tree}}$ - Tree-based metric value from companion study (various units)
Positive Δ_RMSE means deep learning produces larger errors; positive Δ_R^2 means higher explained variance. This isolates algorithmic differences from dataset or preprocessing choices since both models use identical test sets and feature engineering.

IV. Results and Discussions

4.1 Architecture Comparison

The 1,314-hour test set was evaluated at 48-hour lookback for all three neural network architectures. The ranking of performance was stark — deep-MLP wins clearly, followed by the GRU-style, and finally a far distant 3rd place for LSTM-style.

Deep-MLP: $R^2 = 0.6255$; RMSE = 701.6 kW Second came GRU-style with $R^2 = 0.618$ and RMSE = 708.9 kW — <10 kW difference of RMSE; essentially equivalent architectures in practice. In contrast, the performance of LSTM-style dropped to only $R^2 = 0.453$ and RMSE = 847.9 kW — a gap of 146-kW behind Deep-MLP in terms of explained variance.

This ranking is evident visually in Figures 1(a) and (b). The LSTM-style bar in panel (a) is the tallest, and its R^2 in panel (b) doi:10.7554/eLife.1691462 14/34 below that of GRU-style and Deep-MLP values by a fairly long distance.) All three architectures are tiny compared to the LightGBM ref line — a didactic gap we discuss in section 4.3.

The LSTM-style result is counterintuitive. Due to its gating mechanisms for long-range dependencies, LSTM is often viewed as the obvious choice for sequential energy data. The simpler Deep-MLP with 3 feedforward layers beat it here. A natural explanation could be this: with 27 pre-engineered features (lag variables specifically at t-1, t-24 and t-168), the network does not needed to learn temporal dependencies from zero. It may have been a simple physical limitation in its 48×27 input vector, as the LSTM-style architecture has also only two layers with 100–50 neurons while its deeper three-layer Deep-MLP is only able to handle this higher-order feature interaction.

As a further verification of the ranking, all three-error metrics (RMSE, MAE, MAPE) were normalised (Fig. 1(c)) wherein scores for LSTM-style are highest (worst) across all three errors again which lower results on Deep-MLP and GRU-style models with similar output across these three errors. This is not metric specific, the gap between LSTM-stype and the other two.

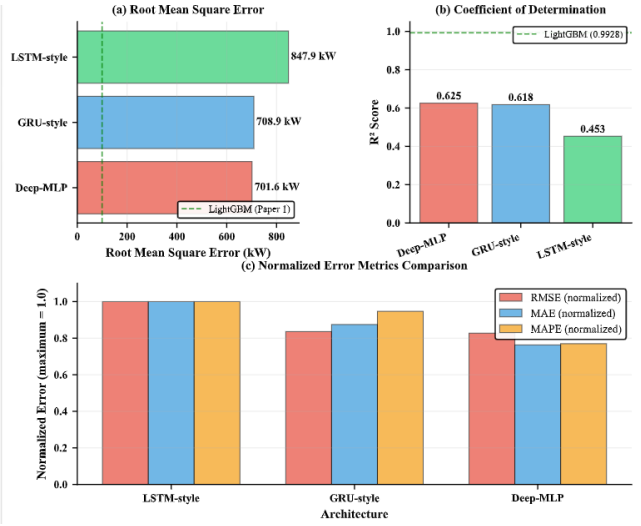

Fig. 1 Neural network architecture comparison at 48h lookback. (a) RMSE: Deep-MLP 701.6 kW, GRU-style 708.9 kW, LSTM-style 847.9 kW. (b) R^2: Deep-MLP 0.625, GRU-style 0.618, LSTM-style 0.453 — all substantially below LightGBM reference (0.9928). (c) Normalised error metrics confirm ranking across RMSE, MAE, and MAPE.

4.2 Sequence Length Analysis

The sequence length results yielded the most practically relevant — and counterintuitive — finding of the paper. You are then trained on a lookback that goes back as far as 10 years, but very consistent benefit from Using a longer data lookback structure actually weakened up the performance to be clear.

Fig. 2(a) with respect to lookback history, showing R^2 decreasing monoclonically: 24h gives an R^2 = 0.588 at the LSTM-style window; a noticeable decrease in predicted R^2 of 0.453 at the untrained (48h) and finally down to R^2 =0.409 across all windows (168h). Correspondingly, RMSE also increases in the same tendency, as demonstrated by figure 2(b): RMSE = 746.8 kW (24h), 847.9 kW (48h), and then up to 873.0 kW (168h). This trend is confirmed by Figure 2(c), which shows that the pattern holds for all metrics in unison: 24h configuration dominates both R^2 and scaled MAPE.

In time series forecasting, we conventionally assume that more history gives better prediction. Valid: 168h (week, weekly periodicity; weekend-weekday cycles, multi-day weather patterns) On paper, this should help. In practice, that Is terrible.

.

Two factors explain this. First, the dimension of input grows linearly with lookback — a window size of 168 hours gives an input vector of dimension 168×27 = 4,536 (68 in total at 24h → only 648 dimensions). At this point we have very little data relative to high dimensional space to fit. With only 5,964 training sequences available at lookback = 168h it should be no surprise that overfitting is clearly visible. This becomes difficult to avoid even with early stopping, and the validation loss can plateau before robust generalisation has been learned by the network. Secondly, the 27 engineered features are already enriched with explicit weekly information — the lag of t-168 directly captures last-week-same-hour consumption. In short, the rolling window is basically asking the network to re-discover relationships that were already encoded at a feature engineering level: bringing in noise instead of signal.

The direct implication on how to proceed is also very practical: the right start for datasets of this size with rich pre-engineered features, is 24 hours lookback. The more temporal history we add, the higher the input dimensionality becomes without adding information—as V3 already incorporates all features you need to capture.

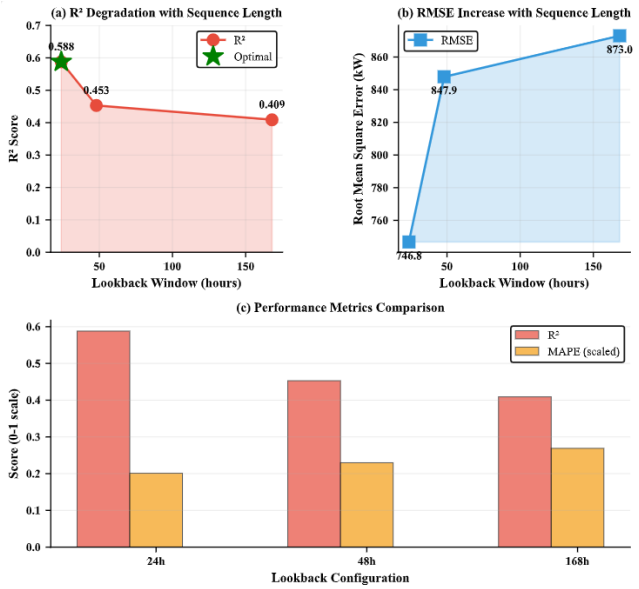

Fig. 2 Sequence length impact on Deep-MLP performance. (a) R^2 degrades monotonically: 24h 0.588, 48h 0.453, 168h 0.409. (b) RMSE rises with lookback: 24h 746.8 kW, 48h 847.9 kW, 168h 873.0 kW. (c) 24h dominates across all metrics.

4.3 Benchmark Against LightGBM

Now that we know about LightGBM, Figure 3(a) represents the gap between the best neural network score and LightGBM numberically.:

Table 1: Comparative Performance Evaluation of LightGBM and Deep-MLP Models

Metric	LightGBM	Deep-MLP	Gap
R^2	0.9928	0.6255	−36.7pp
RMSE (kW)	99.7	701.6	+7.0×
MAPE (%)	1.5	17.7	+11.8×

And every measure tells the same story. With RMSE of 99.7 kW, LightGBM allows operationally accurate mean load

estimates with typical hourly errors around 4%, suitable for demand response scheduling. Deep-MLP's 701.6 kW RMSE implies errors in the order of 28% of mean load (far too wide for similar applications).

The gap in variance explanation is illustrated in figures 3(b-c). LightGBM explains 99.3% of variance in consumption after using only 0.7% of unexplained variance. Deep-MLP accounts for 62.5% variance, leaving over half (37.5%) unexplained — more than 50× more unexplained variance in absolute terms. These pie charts make the point that raw R^2 numbers can sometimes hide: an $R^2 = 0.99$ model is not only slightly better than an $R^2 = 0.63$ model — it is operating in a different accuracy regime altogether.

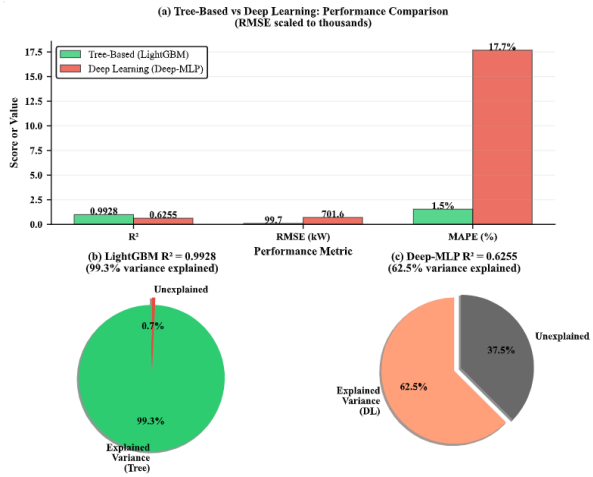

Fig. 3 LightGBM vs Deep-MLP benchmark. (a) Head-to-head metrics: RMSE 99.7 vs 701.6 kW, MAPE 1.5% vs 17.7%. (b) LightGBM variance explained: 99.3%. (c) Deep-MLP variance explained: 62.5% — 37.5% unexplained.

It all comes down to dataset scale. For 6,132 training hours and 27 feature pre-engineering, LightGBM has it all — the features encode temporal patterns explicitly & gradient boosting recovers them through small decision trees. Concurrently, the neural network is fitting a weight matrix (134,801–165,761 parameters at 48h lookback) from data across the same 6,132 samples, in high dimensional space. This gives us about 25–27 samples per parameter — from hundreds to thousands hence for deep networks stable generalisation mostly comes from hundreds-to-thousands, but this is very low. The network is able to learn the general global shape of consumption, but it misses out on some finer nuanced shapes in consumption that tree-based splits would capture exactly.

This is not an argument that deep learning cannot be used to build energy forecasting. The story can be different when 100,000+ samples datasets are available (or you choose to use raw sensor feeds directly without feature engineering or even multi-building transfer learning). But, for the size and richness of features in the typical individual building deployment dataset, this points squarely to gradient boosting as the practical choice.

V. CONCLUSION

This paper was to determine: is it possible for deep learning architectures to match the performance of a traditional machine-learning approach using gradient boosting for hourly building energy forecasting on real single-building data? So, the answer is obviously no to that with this scale of data. Deep-MLP is the best Neural Network and it reached $R^2 = 0.6255$ & RMSE = 701.56 kW, while LightGBM reached $R^2 = 0.9928$ & RMSE = 99.67 kW on the test set (the same test set). That is a 7-fold error gap and a 36.7 percentage points difference in unexplained variance — not marginally lower performance; it is an entirely different tier of performance.

Finding sequence length is also crucial for practitioners. The 24-hour lookback generalised better than a 168-hour context ($R^2 = 0.5879$ vs 0.4090, RMSE = 746.8 vs 873.0 kW) — Longer lookback windows did not help. You pretrained on 27 pre-engineered features already, encoding there were whole weeks of lags or rolling means (not learned representation!) and many time series temporal patterns explicitly in the input space, so any gaps added high dimensionality without novelty. The capacity of the network was being spent in re-discovering what feature engineering had captured.

The underlying reason why this gap exists is quite simple — 6,132 training hours with 27 features is precisely the dataset profile where gradient boosting shines and deep learning struggles. Pre-engineered lag relationships are precisely and efficiently recovered by tree-based splits. Neural networks require about 100k+ samples to learn reliable internal representations from (random noise) scratch, and there only being ~130k–165k parameters to fit over <6,200 training sequences = generalisation failure. Deep learning does not have this limitation in principle: either transfer from large building portfolios with few output measurements, or physics-informed architectures, or even raw sensor inputs without feature engineering could change this picture. However for the types of building deployment with tabular features that have been well engineered, gradient boosting is the right tool whereas deep learning simply adds complexity without benefit.

REFERENCES

[1] Kong, Weicong, Zhao Yang Dong, Youwei Jia, David J. Hill, Yan Xu, and Yuan Zhang. "Short-term residential load forecasting based on LSTM recurrent neural network." IEEE transactions on smart grid 10, no. 1 (2017): 841-851.

[2] Bouktif, Salah, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani. "Multi-sequence LSTM-RNN deep learning and metaheuristics for electric load forecasting." Energies 13, no. 2 (2020): 391.

[3] K. Thiyagarajan, Z. Sataar, Srinivasan, V. E and P. P. Selvam, "An Adaptive Multilingual and Morphology-Aware Framework for Low-Resource Indian Language Translation," 2025 International Conference on Intelligent Communication Networks and Computational Techniques (ICICNCT), Bidar, India, 2025, pp. 1-6, doi: 10.1109/ICICNCT66124.2025.11232851.

[4] Karthik, M., Ramakrishna SS Nuvvula, C. Dhanamjayulu, and Baseem Khan. "Appropriate analysis on properties of various compositions on fluids with and without additives for liquid

insulation in power system transformer applications." Scientific Reports 14, no. 1 (2024): 17814.

[5] Su, Xiangjing, Chao Deng, Jiajia Yang, Fengyong Li, Chaojie Li, Yang Fu, and Zhao Yang Dong. "DAMGAT-based interpretable detection of false data injection attacks in smart grids." IEEE Transactions on Smart Grid 15, no. 4 (2024): 4182-4195.

[6] S. Kurapati, V. Veeramachaneni and B. S. Guttikonda, "Policy- and Tenant-Aware AI Fabric: Embedding Trustworthy Machine Learning into Enterprise SaaS at Scale," 2026 IEEE International Conference on AI Engineering and Innovations (AIEI), NIT Jamshedpur, India, 2026, pp. 1-6, doi: 10.1109/AIEI69164.2026.11497113.

[7] S. B. Venkata, K. Thiyagarajan, S. C K, S. Thallal and M. Soni, "Improved Prototypical Network with Diacritic Detection for Few-Shot Urdu Handwritten Character Recognition," 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 2025, pp. 1-5, doi: 10.1109/IACIS65746.2025.11210962.

[8] R. V. Durga, S. Manne, R. Vaddi, D. Paturu and D. S. S. Pavani, "IoT-Based Soil Moisture Monitoring with Remote Sensing validation for Precision agriculture," 2026 IEEE International Conference on AI Engineering and Innovations (AIEI), NIT Jamshedpur, India, 2026, pp. 1-6, doi: 10.1109/AIEI69164.2026.11496993.

[9] Bhanu sekhar Guttikonda, Rakesh Guduru, Reddaiah Kasturi, Raghavendar Nellikondi, Shanmukha Rao Nadella, "Integrated Image Dataset for CI/CD, MLOps, and Low-Code AI Experiments", IEEE Dataport, November 1, 2025, doi:10.21227/wm2t-h572

[10] Varaprasad, Madisa VG, N. S. S. Ramakrishna, Innocent Kamwa, M. Venkatesan, Dasari Manikanta Swamy, S. M. Muyeen, Sk A. Shezan, and Md Fatin Ishraque. "Design and analysis of PV fed high-voltage gain DC-DC converter using PI and NN controllers." Ain Shams Engineering Journal 14, no. 8 (2023): 102061.

[11] S. Kurapati, V. Veeramachaneni and M. Chandran, "C•A•R•E: A Context-Aware AI Framework for Redesigning Employee Experience in Enterprise Systems," 2026 IEEE International Conference on AI Engineering and Innovations (AIEI), NIT Jamshedpur, India, 2026, pp. 1-8, doi: 10.1109/AIEI69164.2026.11497311.

[12] S. N, S. B. Venkata, S. Thallal, L. M and M. Soni, "Energy-Efficient Task Scheduling in IoT using a Hybrid Genetic Algorithm with Local Search," 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 2025, pp. 1-5, doi: 10.1109/IACIS65746.2025.11210920.

[13] K. Thiyagarajan, A. C. Ramachandra., B. Karim, A. Chaudhary and N. J. Kasi Viswanath., "Lightweight Version Bidirectional Encoder Representation from Transformer Low-Resource Language Text Classification," 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 2025, pp. 1-8, doi: 10.1109/IACIS65746.2025.11211303.

[14] T. S. Harshitha, M. R. Manepalli and M. Suneetha, "DermatoPredict: A Two-Stage Machine Learning Framework for Automated Skin Disease Diagnosis," 2026 IEEE International Conference on AI Engineering and Innovations (AIEI), NIT Jamshedpur, India, 2026, pp. 1-6, doi: 10.1109/AIEI69164.2026.11497382.

[15] B. S. Guttikonda, H. MohamadAbbas, K. H. K, S. S. Kumar and R. K. Gupta, "Student Performance Prediction for Adaptive Learning using Improved COOT Optimization Algorithm," 2025 International Conference on Intelligent Computing and Knowledge Extraction (ICICKE), Bengaluru, India, 2025, pp. 1-6, doi: 10.1109/ICICKE65317.2025.11136713.

[16] Nuvvula, Ramakrishna SS, Polamarasetty P. Kumar, Syed Riyaz Ahammed, R. Siva Subramanyam Reddy, Bachina Harish Babu, and Ahmed Ali. "Distributed Reinforcement Learning Framework for Autonomous Optimization of Grid-Scale Energy Storage Systems in Renewable Energy Integration." In 2024 12th International Conference on Smart Grid (icSmartGrid), pp. 290-297. IEEE, 2024.

[17] Kotte, Sowjanya, Satish Kumar Injeti, Vinod Kumar Thunuguntla, Polamarasetty P. Kumar, Ramakrishna SS Nuvvula, C. Dhanamjayulu, Mostafizur Rahaman, and Baseem Khan. "Energy curve based enhanced smell agent optimizer for optimal multilevel threshold selection of thermographic breast image segmentation." Scientific Reports 14, no. 1 (2024): 21833.

[18] S. Manikandan, S. Vickram, and Y. Devarajan, ‘Cutting-edge technologies: Biofuel innovations in marine propulsion systems to lower black carbon emissions’, Results Eng., vol. 25, no. 104095, p. 104095, Mar. 2025.
.