Automation and Machine Learning (2026)
Clausius Scientific Press, Canada

DOI: 10.23977/autml.2026.070213
ISSN 2516-5003 Vol. 7 Num. 2

Short-Term Electricity Load Forecasting Using an LSTM-GRU Hybrid Deep Learning Model

Jiajie Qiu[a], Xiaoying Su[b,*]

School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China
[a]3887756843@qq.com, [b]25980669@qq.com
**Corresponding author*

Keywords: Electricity load forecasting; LSTM; GRU; Sliding window; Temporal validation

Abstract: Short-term electricity demand forecasting requires models that represent recurrent consumption patterns while responding to abrupt variations in load. This paper presents an LSTM-GRU hybrid forecasting model and a controlled evaluation against standalone recurrent models and simple persistence baselines. The data source is the UCI Electricity Load Diagrams 2011 – 2014 collection, from which hourly aggregate demand is constructed using a fixed customer cohort. Sliding windows, training-only normalization, chronological validation, and explicit treatment of missing intervals define the data pipeline. The hybrid model passes an LSTM output sequence to a GRU layer and produces a one-hour-ahead point forecast. Model structure, input-window length, learning rate, and dropout are examined through complementary comparisons. Forecasting quality is assessed using root mean squared error, mean absolute error, and a documented masked mean absolute percentage error, together with elapsed training time. In the experiments, the hybrid model achieves an RMSE of 38.6 kW with a training time of 12.4 minutes, outperforming the standalone recurrent models and both persistence baselines. The evaluation protocol addresses temporal leakage, baseline selection, and percentage-error denominators. Subsequent work will measure peak-demand errors, stability across seasons, and performance under changing customer composition before operational use.

1. Introduction

Electricity demand forecasts support the coordination of supply and consumption over short planning horizons. Hourly load depends on recurring activity patterns, but its trajectory can also change abruptly around holidays, weather shifts, or unusual events. Long short-term memory networks introduce gated memory mechanisms for learning sequential dependencies [1]. Gated recurrent units offer an alternative recurrent formulation with a different gate structure [2]. Both architectures are evaluated under a common data series and training procedure.

Kong et al. studied LSTM-based short-term residential load forecasting [3], providing a task-specific reason to examine gated recurrence. The hybrid model uses one recurrent stage to encode a longer input sequence and another to compress the representation for prediction. Standalone models

evaluated under the same forecasting origin, data split, and validation budget provide direct controls for assessing this combination.

This paper asks whether an LSTM-GRU stack offers a useful error-time tradeoff for one-hour-ahead aggregate demand prediction. The study design distinguishes architectural comparisons from window-length and learning-rate sensitivity. It also includes persistence baselines to quantify improvement over direct reuse of recent demand. The manuscript reports the complete protocol and the experimental results obtained under it.

2. Related Theories and Technologies

2.1. Gated Recurrent Models for Demand Sequences

Deep learning constructs layered representations that can be adapted to sequential prediction [4]. In an LSTM, input, forget, and output gates regulate information entering, remaining in, and leaving a memory cell [1]. This mechanism allows the model to retain selected features across multiple time steps. A GRU combines information through update and reset gates and maintains a single hidden-state representation [2]. Its simpler state organization can reduce computation for comparable hidden dimensions.

For a forecasting origin t, the input contains observations ending at t and the target is demand at t plus one hour. The reference window contains 72 hourly observations. This window exposes several daily cycles while maintaining a bounded sequence length. A 168-hour alternative includes a full weekly cycle and increases recurrent computation. The experiments below examine the relationship between information content and sequence length.

2.2. Sequential LSTM and GRU Integration

The hybrid model begins with a 64-unit LSTM that returns its hidden state at every input time step. A 32-unit GRU processes that sequence and returns its final hidden state. A dense layer with 16 rectified linear units and a linear scalar output maps the representation to normalized next-hour load. The GRU receives an ordered sequence of LSTM hidden states, preserving the intended temporal role of the second recurrent stage.

The standalone controls use two LSTM layers with 64 and 32 units or two GRU layers with the same widths. All models share the dense head, input channels, and target. Equal hidden widths produce different parameter counts across recurrent cell types. The comparison evaluates these complete forecasting systems, with parameter-matched controls reserved for subsequent analysis of gate design and model capacity.

2.3. Normalization and Dropout Regularization

Load magnitude is standardized using the training-period mean and standard deviation. The same transformation is applied unchanged to validation and test data, and predictions are returned to kW before error calculation. This prevents future load statistics from influencing model fitting. Calendar channels use sine and cosine encodings of hour of day and day of week, plus a weekend indicator. These deterministic features are known at forecast time and require no target-dependent fitting.

Dropout randomly suppresses selected activations during training as a regularization mechanism [5]. The models apply a rate of 0.15 between recurrent layers and before the dense head. Dropout is disabled during deterministic evaluation. A hybrid control with rate zero isolates the complete dropout setting, while holding model widths and all other reference parameters fixed. Recurrent-state dropout is omitted to keep the recurrence consistent across experiments.

3. Experimental Design and Implementation

3.1. Dataset Preparation

The source is UCI Electricity Load Diagrams 2011 – 2014, which supplies customer power measurements at 15-minute intervals. The analysis uses the available 2012 observations for training, 2013 for validation, and 2014 for a held-out test.

A fixed customer cohort is selected using training-period coverage only. Each customer must have at least 95% valid training intervals and a nonconstant training series. The same cohort is retained throughout validation and testing, without selecting customers based on future behavior. Aggregate demand at a timestamp is the sum of valid customer power readings. Hourly power is the arithmetic mean of four consecutive 15-minute aggregate readings, which retains kW units.

Missing timestamps, duplicate records, and clock conventions were checked before aggregation. Zeros are retained as valid measurements unless source-specific checks identify them as missing. A missing run of at most four 15-minute slots is forward-filled from earlier observations and flagged. Longer missing runs invalidate the affected hourly target and any input window containing them. Only observed test targets are evaluated.

Sliding windows are indexed by target timestamp. Training windows have training-period targets only; validation and test windows may use earlier observed context across a boundary because that context is available at prediction time. Each window ends at its forecasting origin. Training windows may be shuffled after construction, but the temporal partitions remain fixed. The reference task uses rolling one-step forecasting with newly observed loads available at each origin.

3.2. Experimental Environment

The implementation uses Python with TensorFlow and its recurrent layers on a single GPU workstation. All configurations were trained and timed on the same machine under an identical data-loading configuration.

Training time covers fitting and validation until early stopping, excluding source-file parsing and window construction. Warm-up execution and accelerator synchronization were standardized. Subsequent efficiency evaluations will report seconds per epoch and optimizer-update counts alongside total minutes, distinguishing computational speed from early stopping.

3.3. Experimental Parameter Setting

Adam is used with an initial reference learning rate of 0.001, momentum coefficients 0.9 and 0.999 [6]. The objective is mean squared error on standardized load. Batch size is 64, the maximum duration is 100 epochs, and the gradient global norm is clipped at 1.0. Early stopping monitors validation RMSE after inverse transformation, with patience ten and a minimum improvement of 0.1 kW. The lowest-validation-RMSE checkpoint is retained.

The reference hybrid uses a 72-hour window and dropout 0.15. Candidate windows are 24, 72, and 168 hours; candidate learning rates are 0.01, 0.001, and 0.0001. Architecture comparison uses the same reference window and optimization settings for every trained model. The dropout comparison changes only dropout. Window and learning-rate sweeps change one factor at a time, leaving the reference architecture fixed. All choices are made using validation performance.

Two untrained baselines are defined. Last-hour persistence predicts the next hour with the most recent observed load. Daily seasonal persistence uses the observation 24 hours before the target timestamp. Their training-time entries are marked not applicable. Both baselines were evaluated on exactly the same retained test hours as every neural model, ensuring consistent missing-data

exclusions.

3.4. Experimental Indicators

RMSE is the square root of the mean squared difference between predicted and observed hourly load. MAE is the mean absolute difference. Both metrics are computed in kW after reversing normalization. RMSE weights large deviations more strongly, whereas MAE describes a typical absolute error without squaring. Their joint interpretation is useful when rare peaks dominate operational concerns.

Percentage error is defined as masked MAPE: the average absolute error divided by actual load, multiplied by 100, using only hours whose actual load is at least 1 kW. The threshold was fixed before test evaluation. RMSE and MAE use every retained test hour. This definition avoids numerical failure near zero but can hide difficult low-load periods, which are analyzed separately.

The tables report point metrics over the retained test hours. Subsequent evaluations will use seed variability and day-block bootstrap intervals to characterize statistical uncertainty while respecting temporal dependence. Peak-hour MAE, residual bias by season, and error under demand ramps remain useful diagnostic outputs for operational conclusions.

4. Experimental Results and Analysis

4.1. Model Structure and Dropout Comparison

Table 1 reports the model and regularization comparison on the held-out 2014 test year. Trained rows use a 72-hour input window, learning rate 0.001, and dropout 0.15 .

Table 1. Forecasting models and regularization comparison

Model	RMSE (kW)	MAE (kW)	Masked MAPE (%)	Training time (min)
Last hour persistence	62.8	43.6	5.84	Not applicable
Daily persistence	56.2	39.8	5.17	Not applicable
LSTM	43.7	30.5	4.06	11.6
GRU	45.1	31.4	4.23	8.9
Hybrid without dropout	41.5	28.9	3.86	10.8
Hybrid with dropout	38.6	26.7	3.58	12.4

The regularized hybrid reduces RMSE by 5.1 kW relative to LSTM, equivalent to approximately 11.7% of the LSTM value, while its training time is only 0.8 minutes longer. Relative to GRU, the improvement costs 3.5 additional training minutes. These comparisons support joint reporting of predictive quality and training cost.

The dropout row has lower errors than the unregularized hybrid but requires 1.6 additional minutes, because regularization changes checkpoint selection and the stopping epoch. Both persistence baselines are clearly outperformed by every trained model on the retained test hours.

4.2. Input Window Length and Temporal Coverage

Table 2 varies the historical window for the regularized hybrid. The 72-hour row is identical to the reference row in Table 1, preserving internal consistency across comparisons.

Table 2. Input window length and forecast error

Window (hours)	RMSE (kW)	MAE (kW)	Masked MAPE (%)	Training time (min)
24	42.8	29.8	3.99	6.7
72	38.6	26.7	3.58	12.4
168	39.4	27.3	3.65	25.8

The 72-hour window improves RMSE by 4.2 kW relative to 24 hours. Extending the window to 168 hours approximately doubles the training duration, while its RMSE is 0.8 kW higher than the reference value. This result motivates further evaluation of how model capacity and training procedures affect the use of additional weekly context.

Calendar encodings and window length can interact. Because calendar channels already identify the hour and day, some periodic information is available even with a short load history. A subsequent ablation will remove calendar channels while preserving the temporal split, distinguishing the value of additional observed demand from deterministic seasonal labels.

4.3. Learning Rate and Convergence Behavior

Table 3 examines optimization at the reference 72-hour window. The retained epoch means the epoch with lowest validation RMSE, whereas elapsed training includes the patience period after that checkpoint.

Table 3. Learning rate and optimization efficiency

Learning rate	RMSE (kW)	MAE (kW)	Retained epoch	Training time (min)
0.01	47.9	33.8	18	7.1
0.001	38.6	26.7	39	12.4
0.0001	40.7	28.1	86	24.3

The largest rate stops relatively early but has the largest error. The smallest rate requires many more epochs and remains less accurate than the reference setting. Retained checkpoint epoch and elapsed time therefore provide complementary measures of optimization efficiency.

5. Research Conclusions and Prospects

5.1. Research Conclusions

The forecasting system combines an LSTM sequence encoder with GRU compression and a scalar regression head. Its evaluation protocol uses chronological partitions, training-only standardization, matched persistence baselines, and consistent missing-data exclusions. RMSE, MAE, masked MAPE, and training duration jointly define the assessment. The reference configuration achieves 38.6 kW RMSE, 26.7 kW MAE, 3.58% masked MAPE, and 12.4 training minutes on the held-out test year.

The hybrid reduces forecast errors relative to the standalone recurrent models and both persistence baselines while keeping its additional training cost moderate, indicating a favorable error-training-time tradeoff for one-hour-ahead aggregate demand prediction under the evaluated conditions. Further evaluation will address forecast-serving cost and seasonal stability.

5.2. Research Limitations and Prospects

The evaluation covers one annual test partition and a fixed customer cohort. Aggregated demand smooths customer-level volatility, and weather variables are excluded. Additional tests will address

long-term distribution change, individual households, clock transitions, and customer inactivity.

Future work will evaluate rolling seasonal origins, include weather forecasts available at the prediction time, and compare parameter-matched recurrent controls. Uncertainty intervals and peak-sensitive objectives will support applications in which large errors have unequal operational costs. Training efficiency will be reassessed after accounting for search cost, retraining frequency, and forecast-serving time. Tests under cohort changes and unusual demand events will establish the durability of the measured advantage.

References

[1] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[2] Cho K, van Merri ënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 2014: 1724-1734.

[3] Kong W, Dong Z Y, Jia Y, et al. Short-term residential load forecasting based on LSTM recurrent neural network[J]. IEEE Transactions on Smart Grid, 2019, 10(1): 841-851.

[4] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015, 521(7553): 436-444.

[5] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. Journal of Machine Learning Research, 2014, 15(1): 1929-1958.

[6] Kingma D P, Ba J. Adam: A method for stochastic optimization[C]//International Conference on Learning Representations. 2015.