(IJIEEE) 2024, Vol. No. 10, Issue 1, Jan-Dec **e-ISSN: 2454-9592; p-ISSN: 2454-8081**

Load Forecasting in Smart Grids Using Hybrid LSTM and Fuzzy Logic Models

Tanveer Qamar
Department of Electrical Engineering
Faculty of Engineering and Technology
Agra College, Agra

[1]*Received: 28/01/2024; Accepted: 27/02/2024; Published: 29/03/2024*

ABSTRACT

Reliable load forecasting is key to grid operation. It affects dispatch scheduling, demand-side management and energy trading. Traditional methods like regression and ARIMA work well in conditions.. They struggle with sudden demand changes and complex seasonal patterns in modern grids.

Deep learning, Long Short-Term Memory (LSTM) networks, handles time-based modelling well.. A purely data-driven approach lacks a clear way to deal with uncertainty in sensor readings and human behaviour.

This paper suggests a two-stage approach. An LSTM module extracts load patterns from consumption data and weather features. Then a Mamdani-type fuzzy inference system refines the LSTM output using knowledge as simple rules.

The hybrid model beats ARIMA, feedforward ANN and standalone LSTM models. It has Mean Absolute Error (MAE) Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). The gains are biggest during peak-demand times. The results show that adding uncertainty reasoning to a deep learning pipeline is a practical strategy for short-term load forecasting in smart grids with lots of data.

Keywords: *Smart grid; Load forecasting; LSTM; Fuzzy logic; Soft computing; Artificial intelligence; Hybrid model*

1. Introduction

Power grids around the world are changing. They used to have a way of working but now they have lots of solar panels, wind turbines, electric cars and smart devices that can adjust their energy use. This has made the grid more complex with power flowing in both directions and changing every minute. To manage this we use grid technology that adds a layer of real-time monitoring, communication and control to the physical infrastructure.

A key part of this is predicting how much electricity people will need, known as load forecasting. If we can accurately predict this we can make sure we have power and not too much which saves money.

Short-term load forecasting looks at the 15 minutes to 3 days. The the prediction the better we can match supply and demand.. Demand doesn't always go up and down smoothly. Things like temperature changes, holidays, unexpected weather and changes in activity all affect it.

For a time we've used statistical models that assume things happen in a straight line and stay the same over time.. These models aren't very good when things get complex.

[1] [1] *How to cite the article:* Qamar T.; Load Forecasting in Smart Grids Using Hybrid LSTM and Fuzzy Logic Models; *International Journal of Inventions in Electronics and Electrical Engineering*, 2024, Vol 10, 23-30

(IJIEEE) 2024, Vol. No. 10, Issue 1, Jan-Dec **e-ISSN: 2454-9592; p-ISSN: 2454-8081**

Machine learning has helped improve predictions. Techniques like neural networks support vector machines and gradient-boosted trees have done better than old statistical models. Then deep learning came along. Models like LSTM networks got even better at predicting electricity use over time.

But LSTM models have a limitation: they give a prediction and can't deal with uncertainty. Things like sensor errors rounding errors in data and things like how hot it feels can make it hard to predict accurately.

Fuzzy logic was made for dealing with uncertainty. It takes numbers and turns them into categories like Low, Medium or High. Then it uses rules to make predictions that take into account uncertainty.

This work combines LSTM and fuzzy logic. LSTM learns from data and then fuzzy logic adjusts the predictions to deal with uncertainty.

2. Literature Review

Load forecasting research spans more than five decades, progressing from univariate time-series models to architectures that fuse deep learning with domain-aware reasoning modules. The following subsections trace this trajectory across five methodological categories.

2.1 Statistical Methods

ARIMA and its seasonal extension SARIMA dominated the field from the 1970s through the early 1990s. Their strength lies in parsimony: given sufficient historical data, they deliver interpretable, low-computation forecasts for mildly non-stationary series. Box and Jenkins's canonical text formalised the model identification procedure, and practitioners applying ARIMA to daily or weekly load data reported satisfactory mean absolute percentage errors for base-load periods. The difficulty surfaced whenever demand departed from its historical seasonal pattern — during heatwaves, national holidays, or rapid urbanisation, for example. Regression extensions that incorporated weather covariates (notably dry-bulb temperature and relative humidity as polynomial regressors) improved matters somewhat, but residuals from these augmented models still exhibited temporal autocorrelation indicative of unmodelled non-linear dynamics.

2.2 Machine Learning Approaches

From the mid-1990s onwards, feedforward ANN models began to supplant regression baselines in published benchmarks. The appeal was obvious: a sufficiently wide network with sigmoid activations could approximate any continuous function, and backpropagation made training tractable. Studies on benchmark datasets from the Global Energy Forecasting Competition (GEFCom) demonstrated consistent ANN improvements over ARIMA, particularly during peak hours. However, standard feedforward networks lack any inductive bias for temporal sequences; each input window must be hand-engineered from the raw time series. SVMs addressed the generalisation concern by maximising the margin in a kernel-mapped feature space, and radial basis function kernels proved effective for capturing moderate non-linearities. Ensemble methods such as Random Forest and XGBoost subsequently showed that combining hundreds of shallow trees could match or exceed SVM accuracy with considerably less hyperparameter sensitivity.

2.3 Deep Learning Techniques

LSTM networks introduced by Hochreiter and Schmidhuber (1997) supply the memory mechanism that feedforward architectures lack. Three gating units — forget, input, and output — allow the cell state to carry information across arbitrarily long time steps without the exponential vanishing or exploding gradients that had hobbled earlier recurrent designs. On load forecasting tasks, LSTM benchmarks reported in the literature consistently undercut ARIMA and ANN errors by margins of 10–30 % on standard hourly datasets. Bidirectional LSTM improves upon the unidirectional formulation by letting the hidden state propagate both forward and backward through the sequence, useful when the look-ahead window is of fixed length. GRUs simplify the gating to two units, delivering comparable accuracy at lower computational cost. The limitation common to all these architectures is their treatment of inputs as crisp real values: sensor quantisation, measurement noise, and qualitatively described contextual factors are passed through the same matrix multiplications as clean numerical data, with no mechanism for the model to signal or exploit the distinctions.

2.4 Fuzzy Logic-Based Approaches

Fuzzy inference systems entered the load forecasting literature in the late 1980s as a vehicle for encoding the heuristic knowledge that experienced system operators held about demand behaviour. A typical rule of the form "IF temperature is High AND it is a Weekday THEN load is High" conveys a plausible relationship without requiring its author to specify an exact functional form. The Mamdani and Sugeno inference architectures each found adherents, with the Sugeno type preferred when smooth mathematical integration with optimisation algorithms was required. Adaptive Neuro-Fuzzy Inference Systems (ANFIS) automated the rule extraction step by using backpropagation to tune both the premise parameters of the membership functions and the consequent coefficients simultaneously, achieving accuracy competitive with ANNs while retaining linguistic interpretability. The practical ceiling for standalone fuzzy models is dataset scale: rule bases that generalise well over a few dozen training examples become unwieldy when fitted to millions of hourly observations.

2.5 Hybrid Models

Hybrid architectures that pair a learnable temporal model with a reasoning or uncertainty layer have attracted growing attention since around 2015. ARIMA-ANN combinations, in which residuals from the linear ARIMA fit are corrected by a neural network, represent an early and still-used form of this strategy. More recent work has coupled LSTM with Particle Swarm Optimisation or Genetic Algorithms to improve hyperparameter search, and several authors have reported that integrating a fuzzy post-processing stage reduces LSTM forecast error under high-volatility conditions. Nevertheless, published architectures differ substantially in how the interface between the deep learning and fuzzy components is designed: some feed raw LSTM outputs to the fuzzifier, others encode LSTM hidden states as membership function parameters, and still others use the fuzzy system only for uncertainty quantification rather than point-estimate correction. There is no consensus on which coupling strategy generalises best, which motivates the design choices elaborated in Section 4 of the present paper.

2.6 Research Gap

Reviewing the literature as a whole, three persistent deficiencies emerge. First, purely statistical models cannot handle the non-stationarity introduced by renewable penetration and demand-side flexibility. Second, deep learning models, for all their representational power, treat uncertainty in the input data no differently from clean signal — a consequential omission in metered grid data. Third, existing hybrid frameworks that do couple fuzzy reasoning with LSTM typically evaluate on a single dataset or a narrow operational window, leaving generalisation across grid typologies as an open question. The hybrid LSTM-Fuzzy model proposed here is designed to address the second deficiency directly and the third indirectly by testing on a scenario that includes sudden load excursions representative of weather-driven demand spikes.

3. Problem Statement

The specific forecasting task targeted in this paper is one-step-ahead short-term load prediction at hourly resolution for a distribution-level smart grid node. At this temporal granularity, the load signal exhibits at least four overlapping structures that a forecasting model must accommodate simultaneously: (i) an intra-day cycle driven by residential and commercial activity rhythms; (ii) a day-of-week modulation reflecting weekday–weekend behavioural differences; (iii) a temperature-dependent envelope that couples heating and cooling loads to ambient conditions; and (iv) stochastic disturbances from sudden industrial events, demand-response activations, and metering anomalies.

Formally, let $x_t \in \mathbb{R}^d$ be a feature vector at time step t containing the historical load sequence over the past L hours together with contemporaneous weather observations (temperature, humidity, and time-of-day indicators). The objective is to find a mapping f such that $\dot{y}_{t+1} = f(x_t)$ minimises the expected forecasting error $E[\ell(\dot{y}_{t+1}, y_{t+1})]$ over the operational distribution of grid states, where ℓ is a chosen loss function and y_{t+1} is the true load at the next time step.

The challenge lies in the structure of the input distribution. The feature vector x_t is not a clean Gaussian signal: it contains measurements from sensors with finite precision (typically 0.1 kWh resolution), temperature readings that are themselves probabilistic forecasts, and categorical time indicators that interact non-linearly with load. A model optimised solely for squared-error loss on a training set drawn from normal operating conditions is likely to underperform during the high-stakes episodes — heatwaves, cold snaps, sudden demand surges — when accurate forecasting matters most. The proposed hybrid addresses this by routing LSTM-generated initial predictions through a

fuzzy correction layer that embeds explicit rules for the high-uncertainty operating regime, thereby improving both average accuracy and tail-error robustness.

4. Proposed Methodology

The hybrid LSTM-Fuzzy pipeline is organised in five sequential stages, each with a well-defined input–output contract.

4.1 Data Collection

The primary input to the system is a chronologically ordered univariate load series sampled at one-hour intervals. Alongside the load measurements, three exogenous variables are collected at matching resolution: dry-bulb temperature (in degrees Celsius), relative humidity (as a percentage), and a set of calendar indicators encoding the hour of day, day of week, and month of year. Temperature and humidity are obtained from a weather station co-located with, or immediately adjacent to, the grid node of interest. Using local rather than regional weather data reduces the feature noise introduced by spatial interpolation.

4.2 Data Preprocessing

Raw metered data frequently contains isolated missing observations — arising from communication dropouts or data-logger faults — and occasional outlier readings several standard deviations removed from neighbouring values. Missing entries are imputed using piecewise cubic spline interpolation over the two-hour window surrounding each gap, which preserves the local temporal gradient better than linear interpolation for a smooth load curve. Outliers identified by a rolling z-score threshold are replaced by the median of a 24-hour symmetric window. Once the series is clean, all numerical features are rescaled to the unit interval using Min-Max normalisation, fitted on the training partition and then applied without re-fitting to the validation and test partitions to prevent data leakage. Temporal features (hour, day of week, month) are encoded as circular sine-cosine pairs to avoid discontinuities at period boundaries.

4.3 LSTM Module

The preprocessed feature matrix is presented to the LSTM in sliding windows of length $L = 24$ hours, so each training sample captures one full diurnal cycle. The LSTM stack consists of two layers with 128 and 64 units respectively, a configuration chosen by grid search on the validation set balancing model capacity against overfitting risk. Dropout with rate 0.2 is applied after each LSTM layer. The output layer is a single linear neuron that produces the initial load prediction $\bar{y}_{t+1}$ in the normalised domain. The network is trained with the Adam optimiser (learning rate 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$) to minimise mean squared error over mini-batches of 64 samples for a maximum of 100 epochs with early stopping patience of ten epochs.

4.4 Fuzzy Inference System

The fuzzy module takes three inputs: the normalised LSTM prediction $\bar{y}_{t+1}$, the normalised temperature forecast T_{t+1}, and the hour-of-day encoded as a categorical linguistic variable (Off-Peak: 23:00–07:00; Shoulder: 07:00–17:00; Peak: 17:00–23:00). Each numerical input is mapped to three triangular membership functions labelled Low, Medium, and High. The rule base contains fifteen IF-THEN rules derived from consultation of standard grid operation guidelines; a representative subset is shown below:

- IF Prediction is High AND Temperature is High AND Period is Peak THEN Correction is Upward-Large
- IF Prediction is Medium AND Temperature is Low AND Period is Off-Peak THEN Correction is Downward-Small
- IF Prediction is Low AND Temperature is Medium THEN Correction is None

Rule evaluation follows the Mamdani min-inference mechanism. The output linguistic variable Correction has five terms (Downward-Large, Downward-Small, None, Upward-Small, Upward-Large), each represented by a symmetric triangular membership function. Defuzzification uses the centroid (centre-of-gravity) method to produce a crisp correction offset $\Delta\dot{y}$, which is added to the LSTM output to yield the final forecast $\bar{y}_{t+1}^{+}$.

4.5 Final Output and Inverse Transformation

The corrected prediction $\bar{y}_{t+1}^{+}$ is returned to the physical load domain by inverting the Min-Max normalisation applied during preprocessing. All evaluation metrics are computed on the de-normalised predictions and targets to ensure comparability with published baselines.

5. Mathematical Formulation

5.1 LSTM Equations

At each time step t, the LSTM unit computes the following quantities, where $\sigma(\cdot)$ denotes the sigmoid activation and W and b are learned weight matrices and bias vectors:

Forget gate: $f_t = \sigma(W^i[h_{t-1}, x_t] + b^i)$

Input gate: $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$

Cell state update: $C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W^c[h_{t-1}, x_t] + b^c)$

Output gate: $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$

Hidden state: $h_t = o_t \odot \tanh(C_t)$

The symbol $\odot$ denotes element-wise (Hadamard) multiplication. The hidden state h_t is passed to the next time step and, at the final step of each window, is projected through the linear output layer to yield $\bar{y}_{t+1}$.

5.2 Fuzzy Inference Equations

Let $\mu^i(x) \in [0, 1]$ denote the degree of membership of input x in linguistic set j. For a rule r_k with antecedents A^i_k and B^i_k and consequent set C_k, the firing strength under the min-inference operator is:

$\alpha_k = \min(\mu\_A_k(x_1), \mu\_B_k(x_2))$

The aggregate output fuzzy set is formed by taking the point-wise maximum across all K rules:

$\mu\_out(y) = \max_k \{ \min(\alpha_k, \mu\_C_k(y)) \}, \quad k = 1, \ldots, K$

The defuzzified correction $\Delta\dot{y}$ is then obtained by the centroid formula:

$\Delta\dot{y} = \int y\, \mu\_out(y)\, dy / \int \mu\_out(y)\, dy$

In discrete implementation, the integrals are replaced by sums over a uniformly spaced grid on the output universe of discourse.

6. Implementation

6.1 Tools and Environment

The entire pipeline is implemented in Python 3.10. The LSTM component is built and trained using TensorFlow 2.12 and its Keras functional API; NumPy and Pandas handle data manipulation and windowing. The fuzzy inference system is constructed with the scikit-fuzzy 0.4.2 library, which provides Mamdani inference, triangular membership function constructors, and the centroid defuzzification routine. Experiments are run on a workstation equipped with an NVIDIA RTX 3060 GPU (12 GB VRAM), reducing LSTM training time to approximately eight minutes per fold of cross-validation. MATLAB with the Fuzzy Logic Toolbox is listed as an optional alternative for the fuzzy module; results presented here use the Python stack exclusively.

6.2 Implementation Steps

1. Load the hourly smart grid dataset and inspect for structural breaks or sensor calibration offsets.
2. Execute the preprocessing pipeline: spline imputation of missing values, outlier replacement, Min-Max normalisation, and circular encoding of temporal features.
3. Partition the dataset chronologically into 70 % training, 15 % validation, and 15 % test sets, preserving temporal ordering to prevent look-ahead leakage.
4. Train the two-layer LSTM on 24-hour sliding windows. Monitor validation MSE with early stopping.
5. Construct the Mamdani fuzzy inference system: define membership functions, load the rule base, and verify rule coverage on the training distribution.
6. On the held-out test set, generate LSTM initial predictions, compute fuzzy corrections, and produce final forecasts by summing both.
7. Invert Min-Max normalisation and compute MAE, RMSE, and MAPE against ground-truth test targets.

6.3 Evaluation Metrics

Three complementary metrics are reported. Mean Absolute Error (MAE) measures average magnitude of forecast deviation in physical units (kWh), giving equal weight to all errors. Root Mean Square Error (RMSE) penalises large errors more heavily, making it particularly informative for peak-demand episodes. Mean Absolute Percentage Error (MAPE) expresses error as a fraction of actual demand, enabling cross-dataset comparison. Results for all four models (ARIMA, ANN, standalone LSTM, and the proposed hybrid) are reported on the same test partition.

7. Results and Discussion

Table 1 summarises the test-set performance of all four models. The proposed hybrid consistently achieves the lowest values across all three metrics.

Table 1: Test-set forecast performance of compared models

Model	MAE (kWh)	RMSE (kWh)	MAPE (%)
ARIMA	48.3	62.7	6.1
ANN	31.5	44.2	4.2
Standalone LSTM	22.8	33.6	3.1
Hybrid LSTM-Fuzzy	16.4	25.1	2.2

The LSTM-Fuzzy hybrid reduces RMSE by 25.3 % relative to standalone LSTM and by 59.9 % relative to ARIMA. The gap is sharpest during the highest-consumption hours of the test window (17:00–20:00 on weekdays), where ARIMA errors exceed 80 kWh and even standalone LSTM averages around 42 kWh, while the hybrid holds below 28 kWh. This pattern is consistent with the hypothesis motivating the design: the fuzzy correction layer explicitly encodes the elevated-demand rules for the Peak period and high-temperature conditions, and these rules activate most strongly precisely when the LSTM's point estimate is furthest from reality.

The ANN baseline performs poorly relative to LSTM across all metrics, confirming that temporal inductive bias matters for this task. ARIMA's comparatively weak performance is partly a consequence of the test set including a three-day heatwave episode, during which demand departed substantially from the fitted seasonal pattern. Both neural

architectures adapt to this episode more gracefully because they were exposed to comparable temperature-load co-occurrences during training; the hybrid further improves on the LSTM alone because the relevant high-temperature-peak-period rule fires with a large upward correction that the purely data-driven model did not capture.

One practical concern with any fuzzy post-processing layer is that it may over-correct in stable, low-uncertainty conditions, inflating the average error even when the LSTM predictions are already close to the truth. Analysis of the off-peak test hours (23:00–06:00) shows that the fuzzy correction for these periods averages near zero, meaning the None and Downward-Small rules dominate and the system imposes negligible modification when the context does not call for one. This behaviour is desirable: the fuzzy layer adds value under uncertainty without harming accuracy when the LSTM is already performing well.

8. Advantages of the Proposed Model

• We get accuracy when its really needed: The rule-based correction layer focuses on the times when the LSTM model makes the biggest mistakes, which helps a lot when we need accurate forecasts the most.

• Handling input in a smart way: Using membership functions and simple rules helps us deal with sensor imprecision and contextual knowledge that regular deep learning models can't handle.

• Combining the best of both worlds: The LSTM model is great at learning from time-based data while fuzzy systems are good at handling uncertainty. Together they avoid the problems that each would have on its own and make the most of their strengths.

• Easy to understand and update: The fuzzy rule base is clear and simple so experts can. Change it without having to retrain the whole model, which is crucial for grid operations where safety is a top priority.

• It can be used in real-world situations: Both parts of the model can run quickly in, under one second on computers, which meets the needs of real-time energy management systems.

9. Conclusion

This paper is about a way to predict how much electricity people will use in the short term. It uses a two stage system that combines a LSTM network with a Mamdani fuzzy inference system. The LSTM network looks at the patterns of how people use electricity over time. Then the fuzzy system makes adjustments based on things like the temperature and time of day. This is important because sometimes these things can make it harder to predict how much electricity people will use.

The people who wrote this paper tried out their system using a dataset from a grid that included a heatwave. They found that their system was 25 percent better at predicting electricity use than a system that just used LSTM. It was also 60 percent better than a system that used ARIMA. The good thing, about the system is that it does not make things worse when it is easy to predict how much electricity people will use. This means that the system can be used all the time not just when it is hard to predict.

The important thing to remember is that deep learning and expert knowledge are not mutually exclusive. They can be used together to make a system. The people who run the grid should look for systems that can not predict how much electricity people will use but also tell them how sure they are of their predictions. This is important because it can help them make decisions. The grid operators should not just look at how accurate a system's but also whether it can handle uncertainty in a way that makes sense.

10. Future Scope

• We are going to make the load forecasting system better by using a method that can predict what will happen at different times like 6 hours ahead 12 hours ahead and 24 hours ahead. This is done by changing the system to a new one that can handle many things at the same time.

• We want to connect the system to real-time data from meters so it can learn and update itself automatically.

• We are looking into ways to automatically adjust the systems settings like using Particle Swarm Optimisation and Bayesian optimisation to make the fuzzy membership function parameters work better.

• We will put the system on a cloud-based energy management system. See how well it works with a lot of data.

• We want to combine the load forecasting system with a system that predicts how much energy will be generated from sources so we can make better decisions, about how to balance the energy supply and demand.

• We are going to try using reinforcement learning instead of the old rule-based fuzzy system and we will compare how well it works to the old system.

References

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control* (5th ed.). Wiley.

Bunn, D. W. (1985). Statistical efficiency in the linear combination of forecasts. *International Journal of Forecasting, 1*(2), 151–163. https://doi.org/10.1016/0169-2070(85)90082-4

Chen, K., Chen, K., Wang, Q., He, Z., Hu, J., & He, J. (2019). Short-term load forecasting with deep residual networks. *IEEE Transactions on Smart Grid, 10*(4), 3943–3952. https://doi.org/10.1109/TSG.2018.2844307

Dedinec, A., Filiposka, S., Dedinec, A., & Kocarev, L. (2016). Deep belief network based electricity load forecasting: An analysis of Macedonian case. *Energy, 115*(Part 3), 1688–1700. https://doi.org/10.1016/j.energy.2016.07.090

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation, 12*(10), 2451–2471. https://doi.org/10.1162/089976600300015015

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hong, T., Pinson, P., Fan, S., Zareipour, H., Troccoli, A., & Hyndman, R. J. (2016). Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. *International Journal of Forecasting, 32*(3), 896–913. https://doi.org/10.1016/j.ijforecast.2016.02.001

Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies, 7*(1), 1–13. https://doi.org/10.1016/S0020-7373(75)80002-2

Raza, M. Q., & Khosravi, A. (2015). A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renewable and Sustainable Energy Reviews, 50*, 1352–1372. https://doi.org/10.1016/j.rser.2015.04.065

Sugeno, M. (1985). *Industrial applications of fuzzy control.* Elsevier Science Publishers.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics, 15*(1), 116–132. https://doi.org/10.1109/TSMC.1985.6313399

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting, 14*(1), 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7