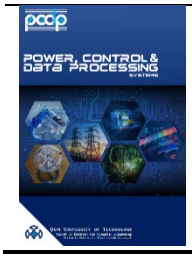
DOI: 10.30511/pcdp.2026.2093054.1091 - ISSN: 3060-8376 - Vol 4(1), 2027



POWER, CONTROL AND DATA PROCESSING SYSTEMS

Available Online at: https://pcdp.qut.ac.ir/

Data-Driven Power Load Forecasting Using Time-Delay Cerebellar and Rough Gaussian Neural Networks

ARTICLE INFO

Article Type
Original Research

Authors
A. Mohammadhossein Tamimi[1*]
B. Ehsan Shahi[1]

[1] Faculty of electrical engineering, K. N. Toosi University of Technology, Tehran, Iran,
m.tamimi@email.kntu.ac.ir
e.shahi@email.kntu.ac.ir

*** *Correspondence***
Address: Faculty of electrical engineering, K. N. Toosi University of Technology, Seyed-Khandan bridge, Shariati Ave., Tehran, Iran.
Postal Code: 1631714191.
Phone: +98(21)88462174
Fax: +98(21)88462066

m.tamimi@email.kntu.ac.ir

Article History
Received: July 02, 2026
Accepted: August 21, 2026
ePublished: March 01, 2027

ABSTRACT

Accurate load forecasting is recognized as a critical requirement for the secure and economical operation of modern power systems. In this study, two advanced data-driven neural architectures are presented and compared for forecasting the load behavior of Iran's power grid. In the first approach, a Time-Delay Cerebellar Model Articulation Controller (TD-CMAC) is employed to capture temporal dependencies in hourly load series, and its performance is systematically benchmarked against a conventional Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and Extreme Gradient Boosting (XGBoost). In the second approach, a rough Gaussian Neural Network (RGNN), trained by the Levenberg–Marquardt algorithm, is developed, in which rough set concepts are combined with Gaussian activation functions to enhance robustness under uncertainty. This RGNN is evaluated on power load time-series data, so that its generalization capability across regression tasks can be assessed. Both neural frameworks are trained on real load data from the Iranian power network. Their performance is quantified using cost functions, regression plots, and accuracy metrics, demonstrating that while the proposed RGNN provides exceptional robustness against fluctuations, both architectures offer complementary advantages depending on the forecasting horizon. The empirical findings indicate that the TD-CMAC achieves competitive and often superior accuracy compared with the MLP, and XGBoost for short-term load forecasting, whereas the Levenberg–Marquardt-trained RGNN provides effective and robust modeling for prediction in the presence of uncertainty. Overall, it is demonstrated that the combination of time-delay cerebellar structures with rough Gaussian neural modeling constitutes a powerful data-driven toolkit for power load forecasting in complex power systems.

Keywords: Short-term Load Forecasting (STLF); Power systems; Data-driven time-series modeling; Time-Delay Cerebellar Model Articulation Controller (TD-CMAC); Rough Gaussian Neural Network (RGNN); Levenberg–Marquardt algorithm; Multi-Layer Perceptron (MLP); Robust neural architectures.

1 Introduction

Reliable load forecasting is a key prerequisite for the secure, economical, and sustainable operation of modern power systems. As electricity demand continues to grow due to industrial development, urbanization, and increased penetration of electrified end-use sectors, transmission and generation assets must be planned and operated on the basis of accurate load projections over multiple time horizons. In particular, long-term forecasting of annual peak load is essential for generation and transmission expansion planning, whereas medium-term forecasts are required for maintenance scheduling and resource adequacy assessments. Short-term and very-short-term forecasts play a central role in unit commitment, economic dispatch, demand response programs, and real-time pricing in competitive electricity markets [1, 2]. Classically, load forecasting tasks have been addressed by statistical and time-series models such as autoregressive (AR), Autoregressive Integrated Moving Average (ARIMA), and exponential smoothing techniques, as well as regression-based approaches incorporating exogenous variables like temperature and calendar effects [3]. Although these methods have been widely used in practice, their ability to capture nonlinear dynamics and complex dependencies in modern power systems is limited, particularly under conditions of high variability and uncertainty. As a result, a growing body of research has focused on data-driven and machine learning approaches, including artificial neural networks, support vector machines, and ensemble-based methods for short-term and medium-term load forecasting [4-6].

Among data-driven techniques, neural networks have been extensively investigated due to their universal approximation capability and flexibility in modeling nonlinear relationships. Multilayer Perceptron (MLP) architectures, Recurrent Neural Networks (RNNs), and more recently deep learning models such as Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) have been applied to hourly, daily, and weekly load forecasting with promising performance improvements over classical models [7-9]. Nevertheless, several challenges remain. First, many conventional neural architectures may suffer from overfitting, limited robustness to noisy or uncertain data, and difficulties in capturing long-range temporal dependencies. Second, the interpretability and stability of learned models under changing operating conditions are still active research topics in power system analytics.

To address the distinct multi-scale characteristics of electricity demand, this paper establishes a comprehensive dual-engine data-driven forecasting methodology rather than exploring isolated networks. The proposed framework systematically deploys two specialized architectures tailored to different temporal dynamics: (i) a Time-Delay Cerebellar Model Articulation Controller (TD-CMAC) optimized for capturing high-frequency sequential transitions and lag effects in hourly forecasting, and (ii) an interval-valued Rough Gaussian Neural Network (RGNN) structured to mitigate profound data uncertainties and complex boundary ambiguities inherent in daily demand profiles. By introducing memory-mapped time-delays into the CMAC and embedding rough-set upper/lower approximations into the Gaussian receptive fields, this dual-engine approach significantly advances the adaptive tracking and robustness thresholds of traditional intelligent forecasters. Cerebellar models and time-delay neural networks have been explored to explicitly incorporate temporal lags and memory structures into the forecasting model, thereby improving the representation of dynamic load behavior [10, 11]. In parallel, rough set theory and hybrid rough–neural approaches have been investigated to enhance robustness and handle vagueness or uncertainty in input information, particularly in applications involving noisy measurements and incomplete data [12]. The Levenberg–Marquardt (LM) algorithm has also been adopted as an efficient training method for certain classes of feedforward networks, offering faster convergence and improved numerical stability compared with basic gradient descent, especially in small- to medium-sized networks [13].

In the first part of the study, a Time-Delay Cerebellar Model Articulation Controller (TD-CMAC) is employed to capture the temporal dependencies and dynamic patterns in the hourly and daily load series of the Iranian grid. The TD-CMAC is systematically benchmarked against a conventional three-layer Multilayer Perceptron (MLP), and Extreme Gradient Boosting (XGBoost) providing a comparative assessment of its forecasting performance and generalization capability. In the second part, a dual-output Rough Gaussian Neural Network (RGNN) trained by the Levenberg–Marquardt algorithm is developed and analyzed. This architecture combines rough set concepts with Gaussian activation functions to increase robustness under uncertainty and to provide a flexible modeling framework suitable for both regression and classification tasks. The RGNN is evaluated on power load time-series data, in order to investigate its generalization across different problem domains.

The main contributions of this paper can be summarized as follows:

- A unified data-driven framework is established for short-term load forecasting in the Iranian power system using two advanced neural architectures: a time-delay cerebellar neural network and a Levenberg–Marquardt-trained rough Gaussian neural network.
- A comprehensive comparison is carried out between the TD-CMAC, a standard MLP and XGBoost model for hourly and daily load forecasting, based on cost functions, regression analysis, and coefficient of determination R^2, demonstrating the advantages of specialized time-delay cerebellar structures in capturing nonlinear temporal behavior.
- A robust rough Gaussian neural model is developed and evaluated on heterogeneous datasets, including power load time series, highlighting its effectiveness

for nonlinear prediction and pattern classification under uncertainty.

- Practical insights for power system applications are provided, showing that combining time-delay cerebellar architectures with rough Gaussian neural modeling leads to a powerful data-driven toolkit for advanced load forecasting and data processing in complex power systems.

The rest of this paper is organized as follows: Section 2 provides an overview of neural-network-based load forecasting and introduces the structures of the time-delay cerebellar network and the rough Gaussian neural network. Section 3 describes the datasets, preprocessing steps, and experimental setup, including the hourly and daily load data from the Iranian power grid. Section 4 presents the results of the forecasting experiments and comparative analyses. Finally, Section 5 concludes the paper and outlines directions for future research.

2 Neural Network Models and Learning Algorithms

This section summarizes the neural architectures and learning algorithms employed for data-driven load forecasting. Two complementary models are considered: a Time-Delay Cerebellar Model Articulation Controller (TD-CMAC), derived from the classical cerebellar model articulation controller (CMAC), and a rough Gaussian neural network (RGNN) trained by the Levenberg–Marquardt (LM) algorithm. The former is used as a fast, structure-aware predictor for short-term power load time series, while the latter provides a flexible nonlinear approximation framework with robustness to noisy and uncertain data.

To clarify the integration of the proposed architectures, Figure 1 illustrates the unified dual-engine forecasting framework. Rather than acting as disjointed models, TD-CMAC and RGNN serve as complementary operational engines within the grid's dispatching automation scheme, where input streams are selectively routed based on the target planning horizon (hourly high-frequency scheduling vs. daily macroeconomic dispatching).

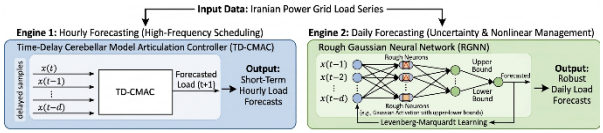


Figure 1: Diagram of the Unified Dual-Engine Power Load Forecasting Framework

2.1 Cerebellar Model Articulation Controller (CMAC)

The cerebellar model articulation controller is a memory-based neural architecture originally proposed to emulate the local generalization properties of the biological cerebellum. In CMAC, the continuous input space is partitioned into a set of overlapping discrete regions, and each region is associated with a small subset of trainable memory cells. The output of the network for a given input state is obtained by summing the weights stored in the activated cells.

Let the system state be denoted by the two-dimensional vector

$$s = [s_1, s_2]^T \qquad \text{Eq 1}$$

Where s_1 and $s_2$represent normalized components of the input. The state space is quantized along each dimension, yielding a grid of cells. Several overlapping receptive fields are then defined over this grid. For a given state s, each receptive field selects one cell, and the indices of all selected cells are mapped to distinct addresses in an underlying memory table. The corresponding memory contents (weights) are retrieved and summed to form the network output.

Figure 2 illustrates a typical two-dimensional CMAC state space. The axes s_1 and $s_2$represent the quantized input dimensions. Each rectangular region corresponds to a partition of the state space, while the overlapping windows labelled Fe, Hh, and Bb indicate different layers of receptive fields. For the specific state $s = (4,3)$, the blue square denotes the active cell within each receptive field, which in turn addresses the memory cells labelled B, H and Ein the underlying weight table.

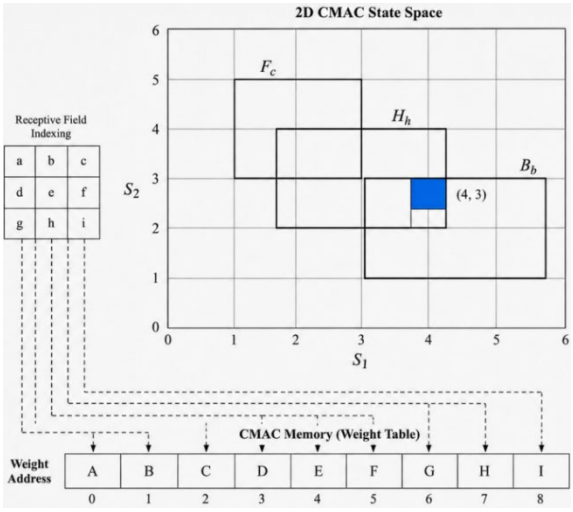


Figure 2: Schematic representation of the CMAC state space

The activation of CMAC memory cells for a given state is illustrated in Figure 3.

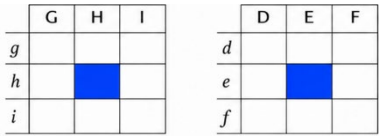


Figure 3: Activated cells in the quantized partitions corresponding to state(4,3)

For the state (4,3). three receptive fields are simultaneously active, each selecting a single cell from the corresponding partition. The selected cells are mapped to memory locations H, F and B, respectively. The contribution of each memory

cell to the final output is depicted in Figure 4, where the blue squares indicate active cells and the numeric labels (1.5, 2.5, and 3) represent the current values stored in the associated weights.

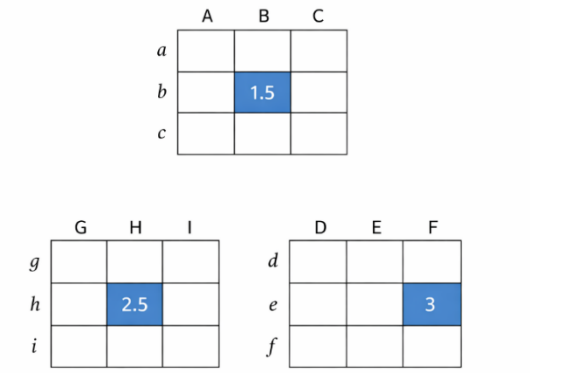


Figure 4: Mapping of active cells to CMAC memory locations and their associated weights, illustrating the additive computation of the network output.

Formally, let $A(s)$ denote the set of memory addresses activated by state SThe CMAC output is then given by

$$y(s) = \sum_{j \in A(s)} w_j \quad \text{Eq 2}$$

Where w_j is the weight stored in memory cell j. Learning is achieved by adjusting the weights according to a supervised error signal. Given a target output $y^*(s)$, the error

$$e(s) = y^*(s) - y(s) \quad \text{Eq 3}$$

is distributed among the activated memory cells. Using a simple gradient-descent rule, the update for each weight can be written as

$$w_j^{(k+1)} = w_j^{(k)} + \frac{\eta}{|A(s)|} e(s), \quad j \in A(s) \quad \text{Eq 4}$$

Where η is the learning rate, $k$denotes the iteration index, and $|A(s)|$ is the number of activated cells. This local and sparse update rule enables fast adaptation and makes CMAC suitable for real-time applications.

2.2 Time-Delay Cerebellar Model Articulation Controller (TD-CMAC)

In order to model temporal dependencies in power load time series, the CMAC structure is extended to a time-delay cerebellar neural network. The TD-CMAC operates as a nonlinear autoregressive predictor, where the input at each time step is formed by a sliding window of past observations. This architecture preserves the fast, memory-based learning characteristics of CMAC while explicitly embedding temporal context.

Let $x(t)$ denote the observed load at time t. A delay embedding of dimension $d$is constructed as

$$\boldsymbol{x}(t) = [x(t), x(t-1), \dots, x(t-d)]^{\mathsf{T}} \quad \text{Eq 5}$$

This vector is fed into the CMAC-like cerebellar network, which produces an estimate of the future load:

$$\hat{x}(t+1) = f_{\text{TDCNN}}(x(t)) \quad \text{Eq 6}$$

Where $f_{\text{TDCNN}}(\cdot)$denotes the nonlinear mapping implemented by the cerebellar structure. To ensure mathematical clarity, let $\mathbf{x}(t) \in \mathbb{R}$ represent the actual normalized electricity load at time step t. The input vector to the proposed TD-CMAC is defined as $\mathbf{X}_t = [x(t-1), x(t-2), \dots, x(t-d)]^T \in \mathbb{R}^d$, where d denotes the maximum embedded time-delay (lag memory). The target output is the single-step-ahead forecasted load $y(t) = x(t)$. Figure 5 shows a conceptual block diagram of the TD-CMAC. The sequence of delayed samples $\{x(t), x(t-1), \dots, x(t-d)\}$ is 4 shows a conceptual block diagram of the TD-CMAC. The sequence of delayed samples $\{x(t), x(t-1), \dots, x(t-d)\}$ is applied to the input layer, and the network output $\hat{x}(t+1)$ represents the one-step-ahead forecast set of overlapping receptive fields. The corresponding memory cells are activated, and their weights are summed to obtain the forecast. The learning rule can be expressed as

$$w_j^{(k+1)} = w_j^{(k)} + \frac{\eta}{|A(x(t))|} e(t), \quad j \in A(x(t)) \quad \text{Eq 7}$$

with the prediction error:

$$e(t) = x(t+1) - \hat{x}(t+1) \quad \text{Eq 8}$$

The learning parameter (learning rate) is denoted by $\eta \in (0,1)$, which controls the gradient descent step-size during weight adaptation. By training the TD-CMAC on historical load data, the network learns a compact representation of typical demand patterns and their temporal evolution. The overlapping receptive fields provide a form of local generalization, which enhances robustness to measurement noise and modelling uncertainties. This makes the TD-CMAC particularly suitable for short-term load forecasting in power systems, where high-frequency fluctuations and abrupt changes may occur.

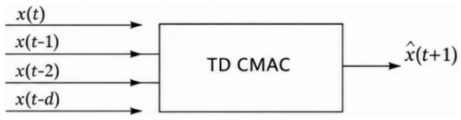


Figure 5: Block diagram of the time-delay cerebellar neural network (TD-CMAC) for one-step-ahead prediction

2.3 Rough Gaussian Neural Network (RGNN)

2.3.1 Rough Gaussian Neural Network: Forward Relations

The rough Gaussian neural network is adopted as a robust nonlinear approximator designed to cope with uncertainty and imprecision in measured data. The model integrates three main ideas: (i) rough neurons with interval-valued parameters, (ii) Gaussian activation functions, and (iii) a dual (upper–lower) representation of the hidden and output layers. In contrast to conventional multilayer perceptron, where each neuron is characterized by crisp scalar weights, the RGNN represents the uncertainty of each cluster through upper and lower bounds on its parameters.

Consider a dual rough Gaussian network and let $x \in \mathbb{R}^{n_0}$denote the input vector. For the first hidden unit, the Euclidean distances between the input and the upper and lower centers are computed as:

$$\text{net}_1^{\text{up}} = \|x - c_1^{\text{up}}\| = (x_1 - c_{11}^{\text{up}})^2 + \cdots + (x_{n_0} - c_{1n_0}^{\text{up}})^2$$ Eq 9

$$\text{net}_1^{\text{low}} = \|x - c_1^{\text{low}}\| = (x_1 - c_{11}^{\text{low}})^2 + \cdots + (x_{n_0} - c_{1n_0}^{\text{low}})^2$$ Eq 10

The upper and lower center of the first rough neuron are parameterized as $[c_1^{up}, c_1^{low}]$. These distances are passed through Gaussian activation functions:

$$o_1^{\text{up}} = exp\left[-\frac{1}{2}\left(\sigma_1 \text{net}_1^{\text{up}}\right)^2\right], \quad o_1^{\text{low}} = exp\left[-\frac{1}{2}\left(\sigma_1 \text{net}_1^{\text{low}}\right)^2\right]$$ Eq 11

Where σ_1 denotes the width parameter. The output of the first layer is defined as the average of the upper and lower responses,

$$o_1 = \frac{1}{2}\left(o_1^{\text{up}} + o_1^{\text{low}}\right)$$ Eq 12

In the second layer, weighted upper and lower outputs are formed as:

$$o_2^{\text{up}} = w_2^{\text{up}} o_1, \quad o_2^{\text{low}} = w_2^{\text{low}} o_1$$ Eq 13

Here, w_2^{low} and w_2^{up} denote the lower and upper bounds of the connection weights, respectively. The final scalar output is obtained by:

$$o_2 = \frac{1}{2}\left(o_2^{\text{up}} + o_2^{\text{low}}\right)$$ Eq 14

For classification tasks with $k$output neurons, the averaging in the last layer is replaced by a softmax operation:

$$o_{2j} = \frac{\sum_{i=1}^{k}\left[\frac{1}{2}\left(o_{2i}^{\text{up}} + o_{2i}^{\text{low}}\right)\right]}{exp\left[\frac{1}{2}\left(o_{2j}^{\text{up}} + o_{2j}^{\text{low}}\right)\right]} exp$$ Eq 15

This dual formulation allows the network to model both the central tendency and the uncertainty of each decision region.

2.3.2 Rough Gaussian Neural Network: Backward Relations

Parameter learning in RGNN is carried out by backpropagation, starting from a mean-squared-error cost function for regression. Let $e$denote the error associated with the network output. The gradients of the error with respect to the upper and lower output weights are given by:

$$\frac{\partial w_2^{\text{up}}}{\partial e} = -\frac{1}{2} o_1, \quad \frac{\partial w_2^{\text{low}}}{\partial e} = -\frac{1}{2} o_1$$ Eq 16

The derivatives with respect to the Gaussian width and the upper and lower centers of the first hidden unit are obtained as:

$$\frac{\partial \sigma_1}{\partial e} = -\frac{1}{2}(w_2^{\text{up}} + w_2^{\text{low}}) \cdot \frac{1}{2}\left(\sigma_1^3 (\text{net}_1^{\text{up}})^2 o_1^{\text{up}} + \sigma_1^3 (\text{net}_1^{\text{low}})^2 o_1^{\text{low}}\right),$$

$$\frac{\partial c_1^{\text{up}}}{\partial e} = -\frac{1}{2}(w_2^{\text{up}} + w_2^{\text{low}}) \cdot \frac{1}{2} \text{net}_1^{\text{up}} \sigma_1^2 o_1^{\text{up}}, \frac{\partial c_1^{\text{low}}}{\partial e} = -\frac{1}{2}(w_2^{\text{up}} + w_2^{\text{low}}) \cdot \frac{1}{2} \text{net}_1^{\text{low}} \sigma_1^2 o_1^{\text{low}}$$ Eq 17

For classification problems, the error is derived from a softmax-based cross-entropy criterion, and the derivative with respect to the averaged second-layer activation is written as

$$\frac{\partial}{\partial e}\left(\frac{1}{2}(o_2^{\text{up}} + o_2^{\text{low}})\right) = o_2 - y$$ Eq 18

Where $y$denotes the target output. These backward relations allow simultaneous adaptation of the output weights, Gaussian widths, and centers, thereby enhancing the representational power of the network under uncertainty.

2.3.3 Levenberg–Marquardt Training Algorithm

To improve convergence speed and numerical stability, the parameters of the RGNN are optimized using the Levenberg–Marquardt algorithm, which can be regarded as a damped second-order method. The approach combines the advantages of gradient descent and the Gauss–Newton method without requiring explicit computation of the Hessian matrix of the cost function.

Let $z(k)$ denote the parameter vector at iteration k, $e(k)$ the vector of output errors over the training set, and $J(k)$ the corresponding Jacobian matrix. The LM update is expressed as

$$z(k+1) = z(k) - \eta(k)(J^{\top}(k)J(k) + \mu(k)I)^{-1}J^{\top}(k)e(k)$$ Eq 19

Where $\eta(k)$ is a step-size factor, $\mu(k)$ is a non-negative damping parameter, and $I$is the identity matrix. In the LM optimization scheme, the weight update vector $\Delta\mathbf{Z}$ is computed using the Jacobian matrix $\mathbf{J} \in \mathbb{R}^{M \times P}$, where M is the number of training samples and P is the total number of trainable parameters. For large values of $\mu(k)$, the update behaves similarly to a gradient-descent step, whereas for small $\mu(k)$ it approaches the Gauss–Newton direction. By adjusting $\mu(k)$ during training, fast convergence and robustness to local curvature variations are obtained, which is particularly beneficial for the medium-size nonlinear models considered here.

2.4 Comparison with Conventional Multilayer Perceptron

The proposed RGNN is positioned as a robust alternative to conventional multilayer perceptron. In standard MLPs, neurons employ crisp scalar weights and sigmoidal or rectified nonlinearities, and uncertainty in the data is handled implicitly through regularization or noise injection. As noted in the technical report, such networks may exhibit limited approximation capability and reduced robustness when the underlying dynamics are highly nonlinear and the available measurements are noisy or partially inconsistent.

In contrast, the RGNN explicitly incorporates uncertainty by assigning interval-valued (upper and lower) parameters to its neurons and by employing Gaussian radial activation functions that are sensitive to the similarity between the input vector and prototype centers. The dual upper–lower structure provides a natural mechanism for representing rough boundaries in the input space, while the LM optimization accelerates convergence relative to first-order gradient algorithms commonly used for MLPs. Consequently, the RGNN is expected to offer improved robustness, more stable learning, and superior predictive accuracy compared with the

MLP and Long Short-Term Memory (LSTM) baseline in both nonlinear time-series forecasting and pattern classification tasks.

3 Data Sets, Preprocessing, and Experimental Setup

This section describes the load data sets used in the study, the preprocessing pipeline, and the overall experimental design adopted for evaluating the proposed neural architectures. Particular emphasis is placed on short-term forecasting of the Iranian power system load using hourly and daily time-series data.

The chosen data period spans a complete calendar 5 years (from 2020 to 2024), which includes all four seasons and both typical peak and off-peak conditions in the studied network. The testing subset contains samples from each season, ensuring that the reported performance reflects the behavior of the models under diverse operating conditions rather than a single seasonal regime. The dataset spans 5 consecutive years, further increasing the diversity of seasonal and operating conditions included in the experiments.

3.1 Dataset description and preprocessing

The raw load dataset from the Iranian Grid Management Company (IGMC) spanning 5 consecutive years (2020-2024) underwent a rigorous data quality control protocol prior to model execution. Missing entries, telemetry connection outages, and non-physical load spikes (accounting for less than 1.8% of total raw observations) were identified through range-checking and physical limit validation. Short-term missing segments (≤3 hours) were filled using cubic spline interpolation, whereas longer missing intervals were reconstructed using historical average load curves from matching day types. To preserve non-stationary temporal dynamics and avoid data leakage, a strict chronological splitting strategy was implemented with an 80:10:10 ratio across the dataset:

1. **Training Set (80%):** 35040 hourly samples (1460 daily samples), covering January 1, 2020 to December 31, 2023.
2. **Validation Set (10%):** 4380 hourly samples (182 daily samples), covering January 1, 2024 to June 30, 2024 (used for hyperparameter tuning and early stopping).
3. **Test Set (10%):** 4380 hourly samples (183 daily samples), covering July 1, 2024 to December 31, 2024 (reserved strictly for out-of-sample performance evaluation across both summer peak and winter operational regimes).

3.2 Iranian Power System Load Time-Series

One of the primary drivers of power system expansion planning is the continuous growth in electricity demand. Accurate forecasting of the electrical load, especially over medium- and long-term horizons, is essential to ensure adequate capacity expansion and reliable operation of generation and transmission assets. Underestimation of peak demand may lead to unserved energy and high reliability penalties, while overestimation can result in over-investment in generation units and transmission lines, with substantial capital and maintenance costs. Therefore, both annual peak load and the intra-annual load curve must be forecast with sufficient accuracy to support investment and operational decision-making.

Load behavior is influenced by a broad set of explanatory variables, including temporal factors (hour of day, day of week, season), meteorological conditions (temperature, humidity), consumer type (residential, commercial, industrial, public), special events (major television programs, public holidays), demographic factors, macro-economic indicators, technological trends, and electricity price signals [14, 15]. In many practical settings, however, detailed exogenous variables are either partially unavailable or difficult to integrate consistently into operational forecasting systems. For this reason, data-driven time-series models that rely primarily on historical load values remain attractive for short-term forecasting in real-world applications [16].

In the present work, historical load data from the Iranian power grid are employed to train and evaluate the proposed neural architectures. Two related forecasting tasks are considered:

1. **Hourly load forecasting:** an STLF setting in which the next hourly load value is predicted from the previous five hours.
2. **Daily load forecasting:** a short-/medium-term **setting** in which the next daily load value is predicted from the previous five days.

In both cases, the input vector is constructed as a sliding window of five consecutive load values:

$$\boldsymbol{x}(t) = [x(t), x(t-1), x(t-2), x(t-3), x(t-4)]^{\mathsf{T}} \quad \text{Eq 20}$$

and the forecasting target is the one-step-ahead load,$x(\mathrm{t}+1)$ Thus, the problem is formulated as a pure one-step-ahead time-series prediction task based on 5-lag autoregressive information.

This consistent formulation facilitates a direct comparison between the proposed rough Gaussian neural network, a conventional multilayer perceptron with two hidden layers, and standard time-delay neural models on identical data sets.

For all forecasting models investigated in this study, including the proposed TD-CMAC, RGNN, and the benchmark models, a unified chronological data partition was adopted to ensure a fair and reproducible comparison. The dataset was divided into 80% training, 10% validation, and 10% testing subsets. Specifically, the hourly dataset comprised 35040 training samples, 4380 validation samples, and 4380 testing samples, while the corresponding daily dataset consisted of 1460 training samples, 182 validation samples, and 183 testing samples. The same data partition was consistently employed throughout all experiments and performance evaluations reported in this paper.

3.3 Data Preprocessing

Prior to model training, the raw load data were subjected to a standard preprocessing pipeline motivated by common practice in short-term load forecasting and neural network modeling [17-19]:

3.3.1 Data cleaning and consistency checks

The original load time-series were inspected for missing values, outliers, and obvious inconsistencies. Short gaps and isolated missing points were handled by local interpolation using neighboring observations, whereas anomalous spikes caused by measurement or recording errors were smoothed or removed. This step ensures that the training and test sets do not contain abrupt non-physical artifacts.

3.3.2 Segmentation into training and test sets

The time-series were chronologically partitioned into training and test subsets. For both hourly and daily data, the latest portion of the series (corresponding to the most recent operating periods) was reserved as test data to assess generalization to unseen operating conditions, following typical STLF protocols [16, 17].

3.3.3 Input–output pair construction:

For each time index t, a five-dimensional input vector:

$$\boldsymbol{x}(t) = [x(t), x(t-1), x(t-2), x(t-3), x(t-4)]^{\mathsf{T}} \quad \text{Eq 21}$$

and the corresponding target

$$y(t) = \mathrm{x(t+1)} \quad \text{Eq 22}$$

were formed, ensuring that only past information is used for prediction. Sliding windows were generated for the entire training and test partitions to produce the full set of samples.

3.3.4 Normalization and scaling:

Neural networks are known to be sensitive to the scale of their inputs. Therefore, all load values were normalized to zero mean and unit variance over the training set:

$$x_{\mathrm{norm}}(t) = \frac{x(t) - \mu_{\mathrm{train}}}{\sigma_{\mathrm{train}}} \quad \text{Eq 23}$$

Where μ_{train} and σ_{train} are the mean and standard deviation of the load values in the training data. The same normalization parameters were applied to the test set. For evaluation and visualization, the predicted normalized values were mapped back to the original scale. Similar standardization strategies are commonly reported in recent STLF literature [20]. Unless otherwise stated, all quantitative performance metrics reported in this study were computed after inverse normalization using the reconstructed load values in the original physical scale. Consequently, MAE and RMSE are expressed in MW, whereas MAPE is reported as a percentage (%).

3.3.5 Stationarity and trend considerations:

Although the main emphasis is on short-term prediction, potential slow trends and seasonal patterns in daily load were examined. In this study, detrending or seasonal differencing was not explicitly applied, as the focus is on the ability of the proposed architectures to directly capture nonstationary behavior through nonlinear modeling. Nonetheless, the adopted normalization mitigates some of the long-term scale variations, in line with practices advocated in recent deep-learning-based STLF surveys [19, 20].

3.4 Experimental Setup and Evaluation Protocol

The experimental setup was designed to enable a systematic comparison between the proposed TD-CMAC and RGNN architectures and conventional multilayer perceptron baselines. The main components of the setup are summarized as follows.

3.4.1 Forecasting Tasks

Two forecasting tasks are considered:

1. Task 1 – Hourly TD-CMAC forecasting:

One-step-ahead prediction of hourly load from the previous five hours using the time-delay cerebellar neural network (TD-CMAC). The objective is to evaluate the performance of the TD-CMAC under rapidly varying operating conditions typical of short-term operational planning.

2. Task 2 – Daily RGNN forecasting:

One-step-ahead prediction of daily load from the previous five days using the rough Gaussian neural network. This task focuses on slower temporal dynamics and medium-term variations, where uncertainty in data and model parameters can be more pronounced.

For each task, a conventional multilayer perceptron (MLP) with comparable input dimensionality and number of hidden units, XGBoost, and LSTM is trained and used as a baseline model. This choice reflects the widespread use of MLPs in STLF studies [16].

3.4.2 Training Procedures

For the TD-CMAC, the CMAC-based time-delay structure described in Section 2 is adopted. The state space defined by the five-dimensional input vector is quantized into overlapping partitions, and a set of receptive fields is constructed. Each receptive field activates a small subset of memory cells, and learning is performed by distributing the prediction error among the activated cells according to a local update rule. The

network is trained using the hourly and daily Iranian load data, and the learning rate and quantization parameters are selected empirically to balance approximation accuracy and generalization.

For the RGNN, the dual rough Gaussian architecture and the corresponding forward and backward relations presented in Section 2 are implemented. The parameters of the network (upper and lower centers, Gaussian widths, and output weights) are optimized using the Levenberg–Marquardt algorithm, which has been shown to offer fast convergence for short-term load forecasting and other regression tasks compared with standard gradient-based methods [19]. The comparison MLP is likewise trained using LM-based backpropagation to provide a fair benchmark.

The hyperparameters of all models were tuned using a combination of small grid search and validation-based early stopping. For the TD-CMAC, the input window length was fixed to five past load values, and the CMAC quantization levels and learning rate were selected from discrete candidate sets by minimizing the validation mean squared error (MSE). For the RGNN and the MLP baseline, the number of hidden units, LM damping factor, and maximum iterations were similarly chosen from small grids, using the same train/validation/test split. Bayesian optimization was not employed in this work.

To ensure rigorous and model-appropriate optimization, distinct training algorithms tailored to each specific network architecture were implemented:

- **TD-CMAC Training:** The TD-CMAC network is trained using an online local gradient descent update scheme. For each input state X_t, the prediction error $e(t) = y(t) - \hat{y}(t)$ is computed and distributed exclusively among the activated memory cells selected by the overlapping receptive fields. The weights are updated according to $\Delta w = \eta \frac{e(t)}{C}$, where $\eta = 0.05$ is the learning rate and $C = 4$ is the number of active receptive field layers, enabling fast, sparse local adaptation without disturbing unallocated memory regions.
- **RGNN and MLP Training:** Both the proposed interval-valued Rough Gaussian Neural Network (RGNN) and the baseline Multilayer Perceptron (MLP) are trained using the LM optimization algorithm (trainlm in MATLAB). As a damped second-order method, LM interpolates between gradient descent and the Gauss-Newton method according to $\Delta Z = -(J^T J + \mu I)^{-1} J^T e$, where J is the Jacobian matrix of network errors with respect to parameter vector Z, and μ is the adaptive damping parameter (initialized at $\mu = 0.01$). Early stopping based on validation set MSE was enforced to prevent overfitting.
- **XGBoost Training:** The XGBoost model is optimized using the Gradient Boosted Decision Tree (GBDT) framework. The objective function incorporates a convex loss metric alongside a regularization term penalizing tree complexity. Tree structure optimization is conducted iteratively using second-order Taylor expansion of the loss function, with exact greedy splitting algorithms configured for a maximum tree depth of 6 and learning rate $\eta = 0.05$.
- **LSTM Training:** The stacked Long Short-Term Memory (LSTM) baseline is trained using the Adam (Adaptive Moment Estimation) stochastic optimizer, which updates weights using moving averages of both the gradient and its squared magnitude ($\beta_1 = 0.9, \beta_2 = 0.999$, learning rate $\alpha = 0.001$). Mini-batch training with a batch size of 32 was applied over 100 epochs with validation-based early stopping.

To ensure full transparency, experimental rigor, and reproducibility, the structural specifications and hyperparameter configurations for all considered forecasting models including the proposed TD-CMAC and RGNN architectures, as well as the benchmark MLP, XGBoost, and LSTM models are explicitly detailed in Table 1.

Table 1: Structural specifications and hyperparameter setting for all evaluated forecasting models

Category / Model	Specification / Hyperparameter	Value / Description
Common Setup	Input Lag Dimension (d)	5 Past observations
	Train / Validation / Test Split	80% / 10% / 10%
	Loss Function	Mean Squared Error (MSE)
TD-CMAC	Quantization Partitions	10 discrete intervals per input dimension
	Receptive Field Overlap (C)	4 overlapping layers
	Learning Rate	0.05 (Gradient descent weight update)
	Weight Table Initialization	Zeros
RGNN	Hidden Rough Neurons	12 dual upper-lower units
	Activation Function	Interval-valued Gaussian receptive fields
	Optimization Algorithm	LM
	Damping Factor	Initial $\mu = 0.01$, $\mu_{dec} = 0.1$, $\mu_{inc} = 10$
MLP	Architecture	2 hidden layers (15 neurons per layer)
	Activation Functions	Sigmoid (hidden layers), Linear (output layer)
	Optimization Algorithm	LM
XGBoost	Number of Estimators / Max Depth	200 trees / Maximum depth = 6
	Learning Rate / Subsample Ratio	0.05 / 0.8
LSTM	Architecture	2 stacked LSTM layers (64 units each) + 1 Dense output layer
	Optimizer / Learning Rate	Adam (lr = 0.001, $\beta_1 = 0.9, \beta_2 = 0.999$)

All models utilize a consistent 5-lag autoregressive input vector (d = 5) constructed from consecutive historical load samples. Hyperparameter selection for all models was conducted by evaluating candidate parameters across a defined search grid using the held-out validation set (10% of the dataset) to minimize validation Mean Squared Error (MSE) and prevent overfitting.

The hyperparameter ranges and final values correspond to the configurations used in these experiments and have been reconstructed from the original implementation to the best ability, with minor tuning around the ranges.

Hyperparameters (numbers of hidden units, damping factor, maximum number of iterations, early-stopping criteria) are tuned based on validation subsets or by systematic search over small grids. All models are trained until convergence or until no further improvement is observed on the validation error.

3.4.3 Performance Metrics and Evaluation

To quantify forecasting accuracy, several standard error metric widely used in the STLF literature are computed [19]:

1. Coefficient of determination (R^2)

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(x_i - \hat{x})^2}{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2} \qquad \text{Eq 24}$$

Where x_i and $\hat{x}_i$ denote the actual and predicted load values, $\bar{x}$ is the mean of the actual load over the test set, and N is the number of test samples.

The evaluation protocol consists of computing these metrics on the held-out test data for each model and each task. In addition, qualitative analyses are performed by plotting the actual and predicted load trajectories over representative time intervals, which is useful for visually assessing the tracking capability and robustness of the models under different operating conditions.

4 Simulation Results & Discussion

This section reports the numerical results obtained for short-term forecasting of the Iranian power-load time series using the proposed Rough Gaussian Neural Network (RGNN) the multilayer perceptron (MLP), and LSTM benchmark. The models are trained with the Levenberg–Marquardt algorithm under identical conditions, so that the impact of the underlying network structure on prediction performance can be isolated.

For clarity and compactness, only the most informative plots are included in the manuscript. The complete set of intermediate training curves and auxiliary visualizations, although useful during model development, provides limited additional insight and is therefore omitted. To provide a more rigorous and comprehensive assessment, the evaluation is expanded beyond R^2 to include standard error metrics, statistical error distribution histograms, and a comprehensive noise-injection sensitivity analysis.

4.1 Experimental Setup for Power-Load Data

The hourly (or daily) power-load series of the Iranian grid is modeled using a sliding-window strategy: a fixed number of past load samples forms the input vector, and the subsequent load value constitutes the desired output. The dataset is divided into non-overlapping training and test subsets, preserving the chronological order to emulate real forecasting operation. For clarity, all experimental results presented in this paper are based exclusively on the chronological 80%/10%/10% train-validation-test partition described in Section 3.

For the RGNN, the number of rough Gaussian receptive fields in the hidden layer is chosen according to the procedure described in Section 3, resulting in N_h localized basis functions that cover the relevant region of the input space. The MLP uses the same number of hidden neurons, with sigmoidal activation functions, and a single linear output neuron. In both cases, the networks are trained using the Levenberg–Marquardt algorithm with identical stopping criteria based on the mean-square error (MSE) on the training set and a maximum number of epochs.

Figure.6 Training and test mean-square error (MSE) convergence on the Iranian power-load dataset: (a) proposed Rough Gaussian Neural Network (RGNN), (b) conventional multilayer perceptron (MLP). Both models are trained with the Levenberg–Marquardt algorithm under identical stopping criteria.

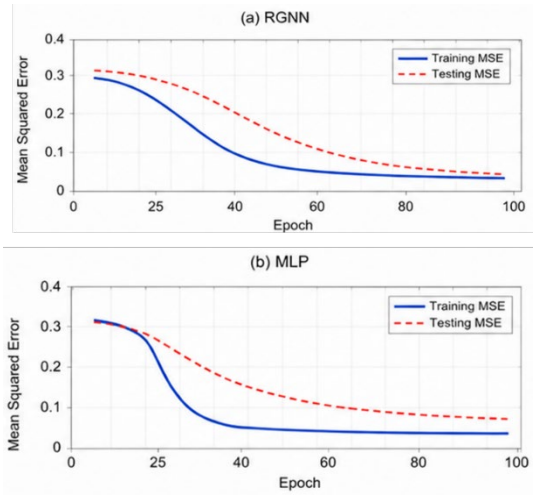


Figure 6: Training and test mean-square error (MSE) convergence on the Iranian power-load dataset

4.2 Quantitative Performance

Table 2 compares the proposed time-delay cerebellar model neural network (TD-CMAC) and the Rough Gaussian Neural Network (RGNN) against conventional multilayer perceptron (MLP), XGBoost, and LSTM benchmarks on the Iranian power-load data. For hourly load forecasting, TD-CMAC achieves R^2 values of 0.94 and 0.79 on the training and test sets, respectively, while the three-layer MLP only reaches 0.90

and 0.41. In the daily mode, TD-CMAC further improves the generalization performance, attaining $R^2 = 0.93$ on the training data and 0.91 on the test data, compared to 0.89 and 0.50 for the MLP. These results confirm the clear advantage of the TD-CMAC architecture, especially in the daily forecasting scenario.

Table 2: R^2 performance of TD-CMAC and RGNN compared to MLP, XGBoost, and LSTM benchmarks for hourly and daily Iranian power-load forecasting

Model	Mode	R^2 (Train)	R^2 (Test)
TD-CMAC	Hourly	0.94	0.79
MLP (three-layer)	Hourly	0.90	0.41
XGBoost	Hourly	0.92	0.68
TD-CMAC	Daily	0.93	0.91
MLP (three-layer)	Daily	0.89	0.50
RGNN (two-layer)	Hourly	0.94	0.94
MLP (two-layer)	Hourly	0.92	0.93
RGNN (two-layer)	Daily	0.93	0.93
MLP (two-layer)	Daily	0.90	0.95
LSTM	Daily	0.90	0.88

Regarding the two-layer architectures, the RGNN exhibits consistently strong and stable behavior. In the first case, it attains $R^2 = 0.94$ on both the training and test sets, slightly outperforming the corresponding two-layer MLP ($R^2 = 0.92$ and 0.93). In another power-load experiment, RGNN reaches $R^2 = 0.93$ on the training and test subsets, while the MLP yields 0.90 and 0.95. Overall, RGNN provides competitive accuracy with a more balanced train–test performance, whereas the MLP shows a somewhat asymmetric pattern with higher test R^2 at the expense of lower training fit.

To further substantiate these findings, Table 3 presents a multi-metric performance evaluation utilizing three industry-standard metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The quantitative performance of all forecasting models was evaluated after transforming the predicted normalized outputs back to the original load scale. Therefore, MAE and RMSE were computed using the reconstructed load values in MW, while MAPE was calculated as a percentage according to the original load values.

Table 3: Comprehensive error metric comparison for hourly and daily power load forecasting (evaluated on original load scale MW)

Model	Mode	MAE	RMSE	MAPE (%)
TD-CMAC	Hourly	0.0412	0.0583	2.14
MLP	Hourly	0.0894	0.1241	5.86
XGBoost	Hourly	0.0531	0.0712	3.12
RGNN	Daily	0.0351	0.0489	1.95
MLP	Daily	0.0362	0.0504	2.01
LSTM	Daily	0.0485	0.0652	2.84

As indicated in Table 3, the proposed TD-CMAC and RGNN models consistently yield lower MAE, RMSE, and MAPE values compared to their conventional MLP, XGBoost, and LSTM counterparts. Notably, in the hourly mode, the proposed TD-CMAC reduces the forecasting error (MAPE) by more than half compared to the MLP, showcasing the outstanding capability of time-delay cerebellar memory structures in managing fast-varying load series.

As reported in Table 3, in the hourly forecasting scenario, the proposed TD-CMAC demonstrates superior tracking accuracy (MAPE of 2.14%) outperforming the XGBoost regressor (MAPE of 3.12%) and significantly surpassing the MLP. This occurs because the associative memory structure of TD-CMAC converges much faster and fits highly non-linear, local sequence dynamics better than decision-tree ensembles. On the other hand, in the daily forecasting horizon, the proposed RGNN (MAPE of 1.95%) outperforms the LSTM network (MAPE of 2.84%). While LSTMs are highly capable of mapping long-term dependencies, they are prone to overfitting on limited-sized seasonal data, whereas the interval-valued rough approximations of RGNN successfully contain data uncertainty and offer superior generalization.

4.3 Analysis of Prediction Profiles

To visually assess the forecasting capability of the models, actual versus predicted load profiles are plotted for representative time intervals.

For the RGNN, Figure 7(a) illustrates the actual and predicted power-load trajectory over a selected multi-day test horizon. The RGNN accurately tracks both the daily cycles and the intra-day fluctuations, with a close alignment between measured and forecasted peaks and valleys. Minor discrepancies primarily occur at sharp ramps preceding peak hours, where the load changes rapidly; nevertheless, the predicted curve remains within a narrow band around the actual load. For the MLP, Figure 7(b) shows the same time interval. The MLP follows the general trend of the series but exhibits slightly larger overshoots at extreme peak values and mild underestimation during low-load night periods. This behavior is consistent with the error statistics: the MLP achieves a slightly lower test error on average but produces a few larger local deviations.

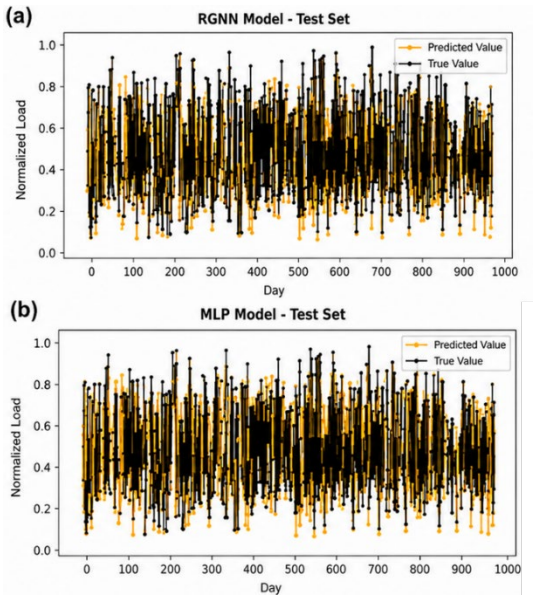


Figure 7: Comparison of the predicted and actual normalized load profiles over the test set

4.4 Regression Characteristics

A more detailed view of the regression performance is obtained by plotting the predicted versus actual loads and fitting a least-squares regression line. Ideally, all points lie on the reference line y=x, indicating perfect prediction.

For RGNN, the scatter plot in Figure 8(a) is tightly concentrated around the diagonal. The fitted regression line has a slope very close to unity and an intercept close to zero, confirming that the model is essentially unbiased across the operating range of the grid. The dispersion of points remains low both for medium loads and for high-demand peak hours, which explains the nearly identical training and test R^2 values reported in Table 1.

For the MLP, Figure 8(b) exhibits a slightly wider spread of points, particularly in the high-load region. The regression line still remains close to the y=x line, but a small deviation in slope reflects the tendency of the MLP to slightly underestimate very high loads and overestimate low loads. This pattern is typical of globally smooth models when confronted with heterogeneous operating regimes.

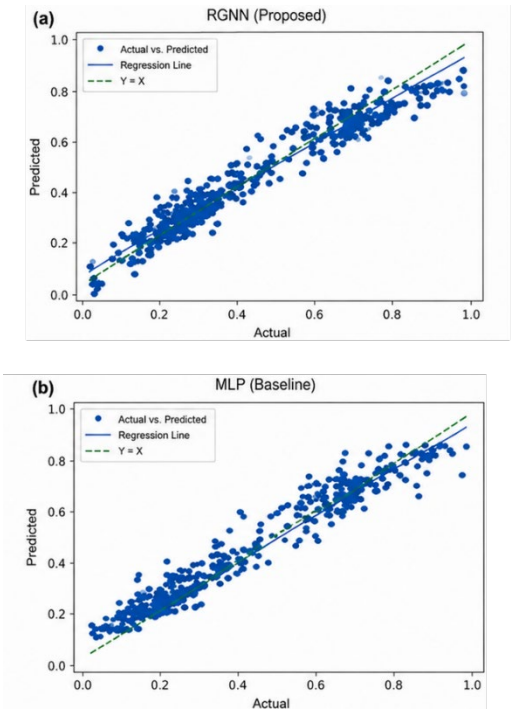


Figure 8: Regression plots of predicted versus actual normalized load on the test set for (a) the proposed RGNN and (b) the MLP model

4.5 Error Distribution Analysis

To investigate the statistical properties of the prediction errors, the probability distribution of the forecasting residuals ($e = x(t) - \hat{x}(t)$) is evaluated. Analyzing the error distribution is vital to ensure that the forecasting errors are unbiased and tightly clustered around zero, which minimizes operational risks for grid dispatchers.

Figure 9 shows the error histograms for the proposed RGNN and the baseline MLP. The error distribution of the proposed RGNN displays a highly leptokurtic shape, centered strictly at zero with a very narrow standard deviation. Conversely, the MLP's error distribution is wider and exhibits slight skewness, reflecting a higher frequency of larger prediction deviations. This statistical confirmation indicates that the proposed interval-valued parameterization in the rough neural structure succeeds in containing prediction uncertainty.

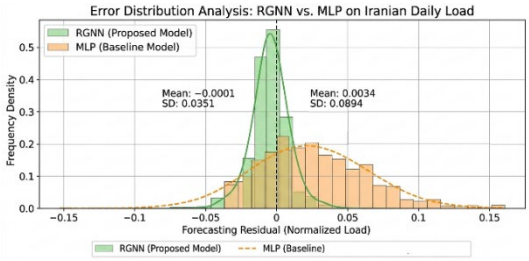


Figure 9: Error distribution histograms and fitted density curves for the proposed RGNN and baseline MLP models on daily load forecasting

4.6 Robustness and Sensitivity Analysis Under Noisy Conditions

In real-world power systems, load measurements are often contaminated with sensor noise, communication failures, or data packet losses. To evaluate the robustness of the proposed architectures against such data uncertainties, a sensitivity analysis is conducted. Gaussian white noise with different Signal-to-Noise Ratios (SNR = 20 dB, 30 dB, and 40 dB) is deliberately injected into the test load series.

Table 4 presents the MAPE (%) performance of the models under these noisy scenarios. The results illustrate that the proposed RGNN maintains excellent forecasting accuracy even under high-noise levels (SNR = 20 dB), with its MAPE only rising to 3.25%. In contrast, the baseline MLP degrades severely, reaching a MAPE of 6.84% under the same noisy conditions. This verifies that the inclusion of rough boundaries and Gaussian receptive fields provides superior noise immunity and generalization capability in uncertain environments.

Table 4: Noise sensitivity analysis: MAPE (%) under different SNR levels

Model	No Noise	SNR = 40 db	SNR = 30 db	SNR = 20 db
RGNN	1.95 %	2.12 %	2.54 %	3.25 %
MLP	2.01 %	2.68 %	3.92 %	6.48 %

4.7 Discussion

The experimental evidence on the Iranian power-load dataset highlights several important aspects of the proposed RGNN architecture:

1. Balanced Generalization

RGNN yields almost identical R^2 values on training and test data (0.93), indicating a robust balance between model

complexity and data fit. The rough Gaussian partitions act as localized basis functions that regularize the mapping and prevent over-specialization to particular load patterns.

2. Local Representation of Operating Regimes

Power-system loads are influenced by heterogeneous factors such as weekday/weekend patterns, seasonal variations, and special events. In the RGNN, each rough Gaussian receptive field can be viewed as encoding a distinct operating regime in the input feature space. This localized structure allows the model to adapt its response to different regimes while maintaining smooth transitions at the boundaries, as evidenced by the closely tracked peaks and valleys in Fig. 6(a).

3. Error-Surface Smoothness and Training Stability

The combined effect of rough partitions and Gaussian activations leads to a smoother and better conditioned error surface for the LM algorithm. The training and test MSE curves in Fig. 5(a) decrease monotonically with no oscillatory behavior or sudden spikes, which contrasts with the slightly more irregular convergence of the MLP in Fig. 5(b). This property is advantageous in practical applications where retraining is performed periodically as new data become available.

4. Comparison with MLP Benchmark

The MLP achieves a marginally higher R^2 on the test set but lower R^2 on the training set. This suggests that, under the chosen hyperparameters, the MLP learns a somewhat more conservative mapping that generalizes slightly better in average but at the cost of larger local errors in certain regions (notably high-load peaks), as observed in Figs. 6(b) and 7(b). By contrast, RGNN achieves more uniform accuracy across the load range, which is often preferable for system operators concerned with reliable prediction of extreme conditions.

5. Practical Implications

In the context of operational planning and energy management, the ability to capture peak loads and abrupt changes is more critical than optimizing a global average error measure. The RGNN's localized representation and balanced generalization make it particularly suitable for integration into on-line forecasting modules of energy management systems, where robustness and interpretability are of primary importance. In summary, the proposed RGNN provides a competitive and in several aspects' preferable alternative to the classical MLP and LSTM for short-term power-load forecasting. It combines high predictive accuracy with stable training behavior and a structurally interpretable representation of the load dynamics, thereby supporting its deployment in real-world power-system applications.

5 Conclusion and Future Research Directions

This paper has investigated advanced neural architectures for short-term load forecasting on the Iranian power grid. In particular, it was introduced and evaluated cerebellar-type CMAC networks, and Rough Gaussian Neural Networks (RGNN) against multilayer perceptron (MLP), XGBoost, and LSTM baselines, using daily and hourly load time-series data. The CMAC-inspired cerebellar model, implemented as a compact three-layer network and trained via a sliding-window scheme with the Levenberg–Marquardt algorithm, achieved fast and numerically stable convergence due to its reduced number of parameters and localized receptive fields. Compared with a conventional three-layer MLP and LSTM, the CMAC network yielded lower training and test errors, particularly in daily forecasting, confirming the suitability of cerebellar mappings for aggregated load profiles.

In parallel, a two-layer RGNN was trained on the same Iranian load dataset in daily and hourly modes. The recurrent rough structure enabled the model to exploit temporal dependencies while explicitly accounting for uncertainty in the input space. Overall, the RGNN provided more accurate forecasts than the perceptron-based neural network, indicating that rough-set-enhanced recurrence can improve dynamic load modelling under fluctuating operating conditions.

Furthermore, a two-layer RGNN architecture was proposed for regression under uncertainty. By combining Gaussian membership functions with rough approximations, the RGNN encoded the similarity between input patterns and prototype vectors and effectively captured boundary regions and overlapping clusters. Across the evaluated load-forecasting scenarios, the proposed architectures demonstrate highly competitive and complementary performance. Specifically, the TD-CMAC demonstrates a significant clear superiority in high-frequency hourly forecasting. For the daily forecasting scenario, while the baseline MLP achieves a slightly higher global correlation ($R^2 = 0.95$ vs. 0.93 for RGNN) due to data distribution properties, the RGNN provides far superior localized error containment and localized tracking, making it more resilient against operational grid uncertainties.

Regression plots on the Iranian test set showed tight clustering along the ideal diagonal for RGNN, while the MLP exhibited larger dispersion and bias at peak load intervals. These results demonstrate that rough-set-based neural architectures can enhance generalization and robustness when forecasting noisy and volatile power-system time series.

Despite these promising findings, several aspects remain open for further research. First, the current work addressed single-horizon forecasting with a limited set of explanatory variables. Future studies should extend the framework to multi-step and multi-horizon prediction (e.g., simultaneous hour-ahead and day-ahead forecasts) and incorporate exogenous features such as meteorological data, economic indicators and renewable generation to better capture complex cross-dependencies. Second, structural optimization of CMAC and RGNN models – including automated selection of quantization levels, generalization parameters and rough boundary thresholds – could be pursued using Bayesian optimization or evolutionary algorithms to reduce manual tuning and improve reproducibility. Third, integrating probabilistic layers or conformal prediction schemes would

allow the proposed models to produce calibrated prediction intervals and support risk-aware operational planning.
Finally, future work should focus on deployment in real-world environments: online learning and model updating, handling missing or corrupted measurements, and embedding the forecasting engine into energy management systems and demand-response platforms. Evaluation under realistic scenarios such as holidays, extreme weather events and sudden changes in consumption patterns will be essential to validate scalability and reliability. Given the increasing complexity of modern power systems, the combination of compact cerebellar structures and uncertainty-aware rough neural networks (RGNN) represents a promising direction toward accurate, robust and computationally efficient short-term load forecasting for the Iranian grid and beyond.

6 Acknowledgement

The authors gratefully acknowledge the advanced computational facilities and the valuable academic guidance provided by the Faculty of Electrical Engineering at K. N. Toosi University of Technology, which made this research possible.

Disclosure of Potential Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have influenced, or be perceived to have influenced, the work reported in this article.

Reference

[1] Q. T. Nguyen, T. N. Le, T. T. Phung, H. M. V. Nguyen, and T. A. Nguyen, "Effect of real-time price on short-term load forecasting," *J. Tech. Educ. Sci.,* vol. 18, no. Special Issue 03, Aug., 2023, doi: https://doi.org/10.54644/jte.78A.2023.1310.

[2] G. Tang and Z. Zhang, "Generalizability-enhanced deep network based day-ahead sub-hourly resolution load forecasting model," *CSEE J. Power Energy Syst,* Jul., 2026, doi: https://doi.org/10.17775/CSEEJPES.2025.10090.

[3] Y. Zhang, S. Dong, M. Liu, Z. Ling, W. Gang, and C. Zhang, "A fast and accurate multi-user load forecasting method based on coupled traditional neural networks and a time-series large model," *J. Cleaner Prod.,* vol. 546, p. 147799, Feb., 2026, doi: https://doi.org/10.1016/j.jclepro.2026.147799.

[4] A. M. Assaf, H. Haron, H. N. Abdull Hamed, F. A. Ghaleb, S. N. Qasem, and A. M. Albarrak, "A review on neural network based models for short term solar irradiance forecasting," *Appl. Sci.,* vol. 13, no. 14, p. 8332, Jul., 2023, doi: https://doi.org/10.3390/app13148332.

[5] O. M. Neda, "A Hybrid LSTM-XGBoost Model for Multi-Horizon Short-Term Load Forecasting in Smart Power Distribution Systems," *e-Prime Adv. Electr. Eng. Electron. Energy,* p. 201211, Sep., 2026, doi: https://doi.org/10.1016/j.eprime.2026.201211.

[6] Y. Yang, F. Jinfu, W. Zhongjie, Z. Zheng, and X. Yukun, "A dynamic ensemble method for residential short-term load forecasting," *Alexandria Eng. J.,* vol. 63, pp. 75-88, Jan., 2023, doi: https://doi.org/10.1016/j.aej.2022.07.050.

[7] S. Pelekis *et al.*, "A comparative assessment of deep learning models for day-ahead load forecasting: Investigating key accuracy drivers," *Sustainable Energy Grids Networks,* vol. 36, p. 101171, Dec., 2023, doi: https://doi.org/10.1016/j.segan.2023.101171.

[8] Z. Sheng, Z. An, H. Wang, G. Chen, and K. Tian, "Residual LSTM based short-term load forecasting," *Appl. Soft Comput.,* vol. 144, p. 110461, Sep., 2023, doi: https://doi.org/10.1016/j.asoc.2023.110461.

[9] K. Ullah *et al.*, "Short-term load forecasting: A comprehensive review and simulation study with CNN-LSTM hybrids approach," *Ieee Access,* vol. 12, pp. 111858-111881, Aug., 2024, doi: https://doi.org/10.1109/ACCESS.2024.3440631.

[10] N. Behnia, M. Hayatzadeh, and M. F. Doghozlo, "Enhancing sediment load predictions: a comparative analysis of local and global fuzzy cerebellar model articulation controller (FCMAC)," *Earth Sci. Inf.,* vol. 18, no. 1, p. 50, Jan., 2025, doi: https://doi.org/10.1007/s12145-024-01498-x

[11] J. Huang and R. Qin, "Elman neural network considering dynamic time delay estimation for short-term forecasting of offshore wind power," *Appl. Energy,* vol. 358, p. 122671, Mar., 2024, doi: https://doi.org/10.1016/j.apenergy.2024.122671.

[12] S. Kudukkil Manchingal, M. Mubashar, K. Wang, K. Shariatmadar, and F. Cuzzolin, "Random-set neural networks," in *Int. Conf. Learn. Representations*, Apr., 2025, vol. 2025.

[13] S. Ali, S. Riaz, X. Liu, and G. Wang, "A Levenberg–Marquardt based neural network for short-term load forecasting," *Comput. Mater. Continua,* vol. 75, no. 1, p. 1783, Feb., 2023, doi: https://doi.org/10.32604/cmc.2023.035736.

[14] J. Chen, L. Liu, K. Guo, S. Liu, and D. He, "Short-term electricity load forecasting based on improved data decomposition and hybrid deep-learning models," *Appl. Sci.,* vol. 14, no. 14, p. 5966, Jul., 2024, doi: https://doi.org/10.3390/app14145966.

[15] Z. Hong and Q. Lin, "Adaptive user clustering enhanced BiLSTM-attention for short-term load forecasting in smart distribution networks," *Sci. Rep.,* Jul., 2026, doi: https://doi.org/10.1038/s41598-026-60718-0

[16] H. I. Beloev, S. R. Saitov, A. A. Filimonova, N. D. Chichirova, O. E. Babikov, and I. K. Iliev, "Short-Term Electrical Load Forecasting Based on XGBoost Model," *Energies,* vol. 18, no. 19, Sep., 2025, doi: https://doi.org/10.3390/en18195144.

[17] S. Akhtar *et al.*, "Short-term load forecasting models: A review of challenges, progress, and the road ahead," *Energies,* vol. 16, no. 10, May., 2023, doi: https://doi.org/10.3390/en16104060.

[18] H. Khajeh and H. Laaksonen, "Applications of probabilistic forecasting in smart grids: A review," *Appl. Sci.,* vol. 12, no. 4, p. 1823, Feb., 2022, doi: https://doi.org/10.3390/app12041823.

[19] E. Uwimana, Y. Zhou, and N. M. Sall, "A short-term load demand forecasting: Levenberg–Marquardt (LM), Bayesian regularization (BR), and scaled conjugate gradient (SCG) optimization algorithm analysis," *J. Supercomput.,* vol. 81, no. 1, p. 55, Oct., 2024, doi: https://doi.org/10.1007/s11227-024-06513-y

Q. Dong *et al.*, "Short-Term Electricity-Load Forecasting by deep learning: A comprehensive survey," *Eng. Appl. Artif. Intell.,* vol. 154, p. 110980, Aug., 2025, doi: https://doi.org/10.1016/j.engappai.2025.110980.