BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES
DOI:10.24425/bpasts.2026.1701

Recursive Aggregation of Liquid Neural Networks for Wind Power Prediction in Poland

W. JACHUŁA* and M. WYDRA

Lublin University of Technology, Faculty of Mathematics and Applied Computer Science
Nadbystrzycka 38 St., 20-618 Lublin, Poland

Abstract. This study presents an application of Liquid Neural Networks (LNNs) for medium-term wind power forecasting in Poland, integrating meteorological variables derived from Numerical Weather Prediction (NWP) models. This paper proposes a neural network consisting of two LNN architectures. Each LNN models a different dynamic range for wind generation prediction in Poland. One- and two-layer LNN architectures were analyzed for their predictive accuracy in wind generation forecasting. The input dataset included wind speed, direction, gust intensity, atmospheric pressure, temperature, and directional change, collected from the Polish Transmission System Operator (TSO) and NWP data. The study further introduced an adaptive prediction fusion framework based on the Recursive Least Squares (RLS) algorithm, which dynamically weights model outputs to minimize error. The proposed method for incorporating weights into LNN outputs is characterized by lower prediction error, resulting in greater accuracy in wind generation predictions in Poland. A comparative analysis of deep learning techniques with exogenous inputs confirmed that the proposed RLS-LNN-based fusion model achieved the highest accuracy on wind data for Poland developed by the authors over the last 5 years.

Keywords: recursive aggregation, wind power prediction, machine learning, liquid neural networks, recursive least squares

1. INTRODUCTION

Renewable energy sources are playing an increasingly important role in each country's energy mix nowadays [1]. Wind energy is one of the key renewable energy sources in Poland, alongside photovoltaics (PV), and is increasingly important in the national energy transition. To accurately predict wind power output in Poland, it is necessary to develop methods that model complex processes with or without exogenous inputs. Recent advances in computer science have improved traditional wind energy forecasting models by employing machine learning (ML) and Artificial Intelligence (AI) techniques [2], [3], thereby increasing forecast accuracy and reliability [4].

The evolution of wind power forecasting has transitioned from traditional statistical models to sophisticated Machine Learning (ML) and Artificial Intelligence (AI) frameworks. For complex time-series tasks, these approaches are generally categorized by how they handle temporal dependencies and non-stationarity.

Recurrent Architectures (RNNs & LSTMs): traditional Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have been the standard for modeling sequential data. These models process information in discrete time steps, using gated mechanisms to retain long-term dependencies. However, they often struggle with irregularly sampled data and the high volatility of wind regimes.

*e-mail: w.jachula@pollub.pl

Global & Probabilistic Models: Recent advancements have introduced "Global Forecasting" models such as DeepAR [5] and DeepVAR [6]. DeepAR uses autoregressive recurrent networks to produce probabilistic forecasts, enabling uncertainty estimation in energy production. Similarly, models like N-HiTS [7] (Neural Hierarchical Interpolation for Time Series) employ hierarchical structures to capture multi-rate periodicities, which is essential for wind generation that varies across different time scales.

One such method is the Liquid Neural Network (LNN). This is a new type of neural network distinguished by its ability to adapt dynamically to changing input data in real time [8]. The liquid neural network layers substantially improve robustness to a wide range of perturbations, including subtle forms of tampering that integrate almost imperceptibly with the background, thereby achieving superior performance compared to conventional approaches [9].

LNNs are inspired by biological neurons and more closely reflect how the human brain operates; they exhibit nonlinear signal-flow dynamics, making them flexible, economical, and effective in environments with high variability [10].

The versatility of LNNs is evidenced by their successful adoption in diverse fields, including medical diagnostics (brain tumor identification and respiratory sound classification), cybersecurity (forgery detection), and human-centric systems (emotion recognition and building energy demand prediction). These applications underscore the model's capacity to process

high-dimensional, non-stationary data where traditional models often fail.

Liquid Neural Networks (LNNs) are a modern method for modeling complex time series and nonlinear dynamics, capable of adapting to changing conditions in real time [11]. Unlike traditional Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM), LNNs are characterized by continuous dynamics described by differential equations, which makes them particularly effective for non-stationary time series that are challenging to predict [12]. In cases of complex data dynamics, it is essential to capture not only long-term dependencies but also local variability [13]. LNNs, which use neurons described by nonlinear differential equations, can represent these changes continuously, so that the system's state depends not only on previous data but also on current conditions and environmental changes [14]. Each neuron in the LNN describes its state using a system of Ordinary Differential Equations (ODEs), which enables smooth temporal evolution and a more accurate representation of the non-stationary nature of phenomena [15]. This feature is crucial for predicting wind energy production, which exhibits irregular patterns and depends on multiple input parameters, and exceptionally detailed technical data for individual wind farm locations [16].

Based on this knowledge, the authors present results from wind power forecasting using the LNN method to improve the precision of wind generation in Poland. The data were obtained from the Polish Transmission System Operator (TSO) and the Numerical Weather Prediction (NWP) system. In the research, data on generated energy, wind direction, atmospheric pressure, and temperature at all turbine coordinates were used, which significantly impacted prediction accuracy.

Although advanced models like DeepAR, DeepVAR, and N-HiTS have been explored for time-series forecasting, they often lack the continuous-time adaptability required for the high non-stationarity of wind power in regional grids. The primary novelty of this study lies in the introduction of a Recursive Aggregation framework that dynamically fuses different Liquid Neural Network (LNN) architectures using a Recursive Least Squares (RLS) algorithm. Unlike static ensemble methods, our approach offers three distinct contributions:

- Adaptive Dynamic Weighting: we propose a mechanism that "switches" importance between optimized single- and multi-layer LNNs to minimize instantaneous error during rapid weather transitions.
- Depth-Performance Analysis: This study provides the first systematic evaluation of how increasing LNN depth (from one to two layers) impacts forecasting stability and bias reduction specifically for the Polish power system.
- Regional Integration: We demonstrate superior predictive accuracy by integrating high-resolution Numerical Weather Prediction (NWP) variables with the continuous-time ODE dynamics of LNNs for a medium-term 60-hour horizon using real-world Polish TSO data.

The remainder of this paper is organized as follows: the second part describes the model's fundamental principles. It outlines the preparatory work conducted before the experiment and the model architecture. The third part summarizes the research results. The fourth part describes the conclusions.

2. METHOD

2.1. Data used in research

In the presented methodology, the authors developed a comprehensive database encompassing various wind turbine types and their corresponding production curves across Poland. The dataset was compiled using information provided by both turbine manufacturers and energy investors. Based on data obtained from the Polish Wind Energy Association, the precise geographic coordinates of individual turbines were identified and subsequently validated using publicly available satellite imagery sourced from Google Maps.

Data preprocessing was executed in several systematic stages to ensure the numerical stability of the model. First, missing data points were imputed with column-wise mean values. Subsequently, both the input feature vector (X) and the target variable (y) were normalized using the StandardScaler algorithm from the Scikit-learn library. This scaling is essential for the convergence of the Ordinary Differential Equation (ODE) solvers within the LTC layers. Finally, the dataset was partitioned chronologically into training (73%), validation (13.5%), and testing (13.5%) subsets, with random shuffling disabled (shuffle=False) to strictly preserve the temporal causality of the time-series data.

The resulting dataset was integrated with a Numerical Weather Prediction (NWP) model, which is updated every 6 hours and provides 60-hour forecasts, as illustrated in Fig. 1. The database stored a 10-minute temporal resolution for each turbine location.

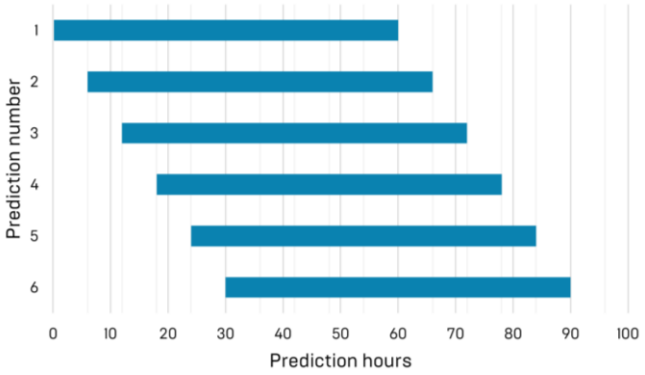

Fig.1. Diagram illustrating how the prediction is made

Figures from Fig. 2a to Fig. 2d present the characteristics of total wind power in Poland as a function of average measurement parameters.

The analysis in Fig. 2a indicates that the highest wind power generation occurs at wind speeds between 4.5 and 9.5 m/s. In turn, Fig. 2b shows that the maximum energy production occurs when the wind direction is northward. Fig. 2c shows that the optimal pressure range for efficient wind energy conversion lies between 0.98 and 1.02 Pa. Moreover, as shown in Fig. 2d, the highest power generation occurs at temperatures between 0°C and 10°C. These results clearly indicate the significant

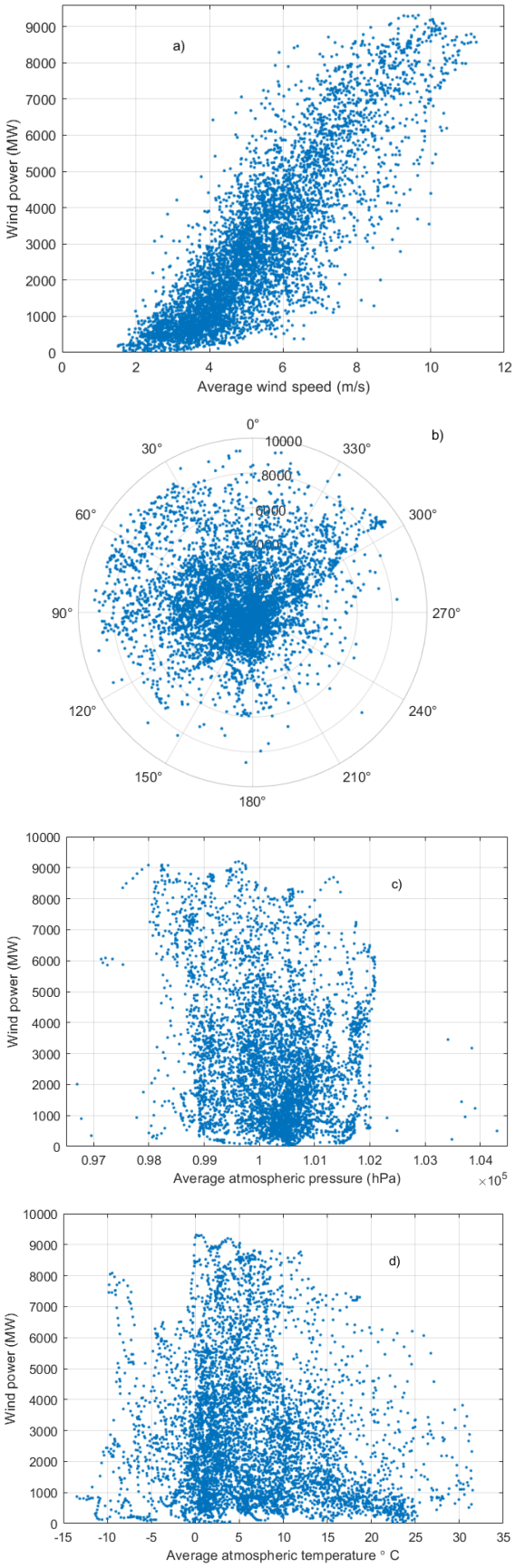

Fig.2. Characteristics of total wind power in Poland: a) average wind speed (m/s); b) average wind direction (°); c) average atmospheric pressure (Pa); d) average atmospheric temperature (°C) [17]

influence of environmental parameters – such as wind speed and direction, atmospheric pressure, and air temperature – on the efficiency of wind energy systems.

2.2. Liquid Neural Network for time-series predictions

LNNs are a class of neural models in which the dynamics of individual neurons are described by systems of ODEs, enabling the modeling of time-varying processes in a continuous, adaptive manner. Instead of treating neuronal activation as a static function of the input, as in classical neural networks, LNNs model it as a temporal trajectory governed by a nonlinear differential equation. The basic model of a neuron in an LNN is based on a biologically inspired formalism in which the state of a neuron h(t) at time t changes according to Eq. (1) [18]:

$$\tau(t)\frac{dh(t)}{dt} = -h(t) + \sigma(Wx(t) + Uh(t) + b) \qquad (1)$$

where: $\tau(t)$ – time delay constant, dynamically adjusting the rate of state changes; $h(t) \in R^D$ – neuron state vector; $\sigma(\cdot)$ – nonlinear activation function (tanh); W, U, b – weight parameters and biases learned during training; $x(t)$ – input signal in time t.

The characteristic feature of LNN is the use of time-varying dynamic parameters τ, which allows the neuron to adapt to local signal conditions. This distinguishes LNN from traditional recurrent neural networks (RNNs and LSTMs), which process data at discrete time steps.
The solution of the above differential equation (or the ODE system for many neurons) is computed numerically using methods such as Runge-Kutta. Model training is enabled by the ODE solver's use of backpropagation, as in the Neural ODE framework, which allows differentiation along a dynamic trajectory [19].

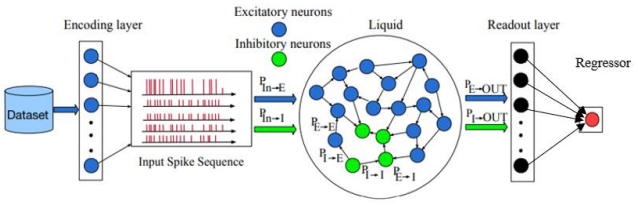

Fig.3. The schema of the LSM learning model with a regressor layer [20]

A standard learning model based on Liquid State Machine (LSM) consists of three main components: the encoding layer, the liquid, and the readout layer. As shown in Fig. 3, the input data is first converted into a spike sequence by the encoding layer using a selected coding scheme, such as rate, time, or phase coding. The encoding neurons, composed exclusively of excitatory neurons, are then randomly and sparsely connected to neurons within the liquid. The liquid – a network of randomly interconnected spiking neurons – projects low-dimensional input data into a high-dimensional, linearly separable liquid state. Both excitatory and inhibitory neurons in the liquid are modeled using the leaky integrate-and-fire (LIF) model [21]. The liquid features four types of connections: excitatory-to-excitatory (E → E), excitatory-to-

inhibitory (E → I), inhibitory-to-excitatory (I → E), and inhibitory-to-inhibitory (I → I) [21].
To simplify comprehension of the proposed contribution, the LNN's dynamic behavior is modeled as a system of Ordinary Differential Equations (ODEs) defined in Eq. (1), allowing continuous temporal adaptation to Poland's wind regime. Unlike generic LSM frameworks described in literature, our implementation replaces the standard classifier with a dedicated regression layer tailored for high-dimensional power output data.
In this study, the data processing pipeline has been modified as shown in Fig. 3, with the classifier layer replaced by a regression layer.
To optimize the neural network parameters, the Optuna framework was employed for automated hyperparameter tuning in Python [22]. In this process, two neural network architectures were defined and optimized independently. For the single-layer model, the search space for the number of neurons was constrained to the range of 3–112. Optuna iteratively sampled values within this interval and identified the optimal configuration, yielding a number of neurons equal to 29 for this architecture.
For the two-layer model, the optimization procedure was designed hierarchically. The number of neurons in the first layer was selected within a range from 1 to the upper limit of n_neurons (determined adaptively by Optuna). In contrast, the second layer's neurons were chosen within a subsequent range from the first layer's number of neurons, up to 112. The optimization process yielded an optimal configuration with 28 neurons. After determining these hyperparameters, both models were trained on the curated dataset, yielding optimal network architectures tailored to the characteristics of wind power generation data. In both cases, the optimization objective was to minimize the Root Mean Squared Error (RMSE), ensuring that the selected model configurations achieved the highest possible predictive accuracy for the observed power output.
During training and subsequent predictions, the Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and Absolute Error (AE) were calculated.

2.3. Liquid Neural Network model

The software was implemented in Python, using Keras as the primary high-level framework. The architecture leveraged the ncps (Neural Circuit Policies) extension to implement Liquid Time-Constant (LTC) layers and to define inter-neuron connectivity via the wiring module. To manage the optimization process, the Optuna framework was configured with a MySQL database backend using sqlalchemy, enabling the persistent storage and systematic management of high-dimensional hyperparameter search results.
The model was prepared using the Keras and ncps libraries [23], both of which are suitable for processing sequential data (e.g., time series). The Keras library is a high-level framework for creating and training neural networks. In contrast, ncps is an extension of Keras that enables the creation and training of LNNs, which are neural networks based on continuous dynamical systems (ODEs). The used extension contains two main parameters:

- wirings – definitions of the topology of connections between neurons and
- Liquid Time-Constant (LTC) layer, the central computing unit of Liquid Neural Networks.

The initial weight-assignment phase used the Glorot/Xavier algorithm. Beyond this, randomness was intentionally eliminated from the pipeline to ensure reproducibility: the data splitting process was strictly sequential without shuffling (shuffle=False), and the hidden layer utilized a wirings.FullyConnected topology, which establishes a deterministic and static connectivity matrix between neurons.
The LNN training protocol focused on minimizing the Mean Squared Error (MSE) loss function using the Adam optimizer with a learning rate of 0.01. The process was configured for a maximum of 400 epochs; however, an EarlyStopping mechanism with a patience of 100 epochs was integrated to prevent overfitting by reverting to the best-performing weights on the validation set. Hyperparameter tuning was automated using the Optuna framework, which explored the neuron count space (3-112) for both single- and two-layer architectures. Training was performed with a batch size of 1, allowing the continuous-time dynamics of the LTC layers to adapt to the signal at each time step.
A full mesh of connections (FullyConnected) is created between neurons in the LTC layer. The number of neurons is the best result from the study using Optuna. In this case, the number of neurons is 29, and the number of output neurons is 1, indicating a single output layer (here adjusted to the number of output parameters).
As shown in Fig. 4, three types of neurons are formed by creating connections: a) sensory neurons – they receive input data and transform it into a form understandable by subsequent layers; b) inter neurons – they implement the main computational dynamics in the network by creating internal hidden states, as well as processing information and learning complex temporal dependencies; c) motor neurons – receive the processed information from interneurons and generate the final model output.

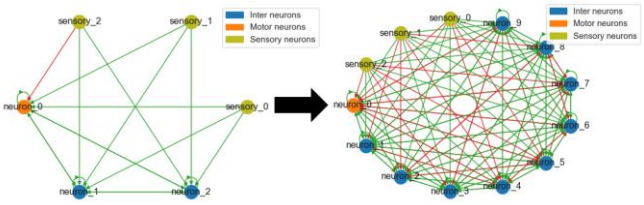

Fig.4. Liquid Neural Network simplified architecture

Next, we define a sequential model with a Liquid Time-Constant (LTC) layer that receives the input data. LTC is a recursive RNN layer that uses differential equations to model the state dynamics. Here, we also configure it to return the entire output sequence rather than just the last time step, enabling prediction of entire time-varying trajectories.
After entering the data and settings, the model is compiled with the Adam optimizer and the MSE loss function, a common loss function in regression. The final step is to train the model. This is where the datasets used for training, the sampling size, validation data, and training duration are defined – in this work, training was set to 400 epochs.

In summary, the model learns to predict sequential data based on previous steps. An LTC layer can efficiently model complex nonlinear dynamics, particularly where traditional recurrent neural networks (RNNs) may be limited.

2.4. Aggregation of predictions

The code developed by the authors implements adaptive prediction fusion based on the Recursive Least Squares (RLS) algorithm combined with mean-median weight estimation. The main goal of this approach is to adaptively combine predictions from two models to minimize prediction error relative to the actual values. The RLS algorithm uses a forgetting factor $\lambda = 0.9$ that controls the weight adaptation rate, and an initial covariance matrix:

$$P_0 = \delta I_2 \tag{2}$$

where:
$\delta = 1000$; I_2 – a 2×2 identity matrix; and initial weights $w_0 = [0.5; 0.5]$ that assume equal contributions from both predictions.

The algorithm processes the data row by row and updates the weights for each time sample in the row using standard RLS equations. The prediction vector is $\phi_t = [p_1(t); p_2(t)]$, and the real value is r_t.
The covariance matrix and Kalman gain are calculated from:

$$K_t = \frac{P_{t-1}\,\phi_t}{\lambda + \phi_t^T P_{t-1}\,\phi_{t'}} \tag{3}$$

The prediction is calculated as:

$$\hat{y}_t = w_{t-1}^T \phi_t \tag{4}$$

The prediction error is:

$$e_t = r_t - \hat{y}_t \tag{5}$$

The weights are updated according to:

$$w_t = w_{t-1} + K_t e_t \tag{6}$$

The covariance matrix is modified according to:

$$P_t = \frac{P_{t-1} - K_t\,\phi_t^T P_{t-1}}{\lambda} \tag{7}$$

This approach enables dynamic adaptation of weights based on the predictive performance of individual models, which is particularly important for data that changes over time. For each iteration, the RMSE and MSE metrics are calculated. These values are determined separately for each model's predictions, for adaptive RLS fusion, and for fusion with global weights defined as the mean and median. After adaptive weight updating, the global weights are determined from the formulas:

$$w_{mean} = \frac{1}{N} \sum_{k-1}^{N} w_k \tag{8}$$

and:

$$w_{median} = median(w_1, \ldots, w_N) \tag{9}$$

This method assesses predictive performance using fixed weights, establishing a reference point for adaptive evaluations. Lastly, time-series plots are produced to display the true values alongside all prediction variants. Additionally, the area under the RMSE and MSE curves is calculated using the trapezoidal method, enabling a quantitative assessment of the combined forecast performance over the analyzed horizon. As a result, the code enables a comprehensive evaluation of adaptive prediction fusion quality and a comparison of methods, thereby identifying the strategy that best minimizes error, consistent with the literature on adaptive model fusion in time series forecasting.
The diagram shown in Fig. 5 presents an adaptive Recursive Least Squares (RLS) algorithm used to estimate the weights of wind power forecasts generated by one-layer and two-layer Liquid Neural Network models based on numerical weather predictions. The weights (w1) and (w2) are recursively updated by minimizing the error between the real wind power generation at iteration (i+1) and the corresponding model predictions. The resulting weights reflect each model's current predictive performance and enable dynamic combination across successive forecasting horizons.

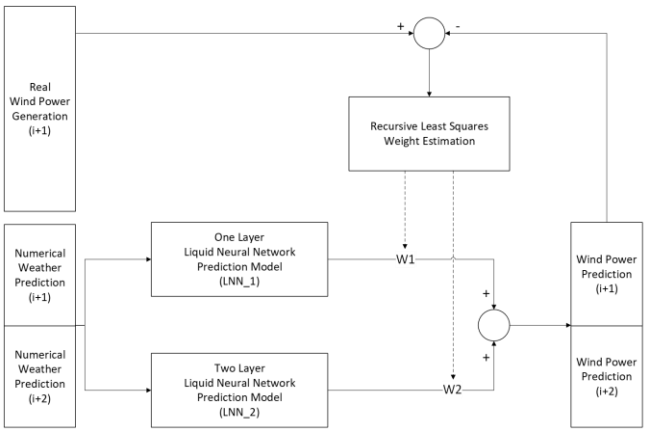

Fig.5. RLS algorithm for LNN weight calculation

where:
$i + 1$ – current iteration of prediction or real value; the nearest forecast horizon
$i + 2$ – the iteration following the current iteration of prediction or real value; next step of the forecast
w_1 – weight assigned to the forecast from the one-layer model (LNN_1)
w_2 – weight assigned to the forecast from the two-layer model (LNN_2)

3. RESULTS AND DISCUSSION

The performance of two Liquid Neural Network (LNN) architectures – a one-layer and a two-layer model – was evaluated for short-term wind power generation forecasting with a 60-hour prediction horizon and a 6-hour time step. Both models were trained and validated on historical meteorological and wind generation data, including average pressure,

temperature, wind speed, wind direction, gust intensity, and directional change. The dataset was divided sequentially into 73% for training, 13.5% for validation, and 13.5% for testing, yielding 793 hours of test data. All forecasts were point predictions. The adaptive RLS fusion demonstrated superior tracking of actual wind power peaks compared to static global weights. While the literature on adaptive model fusion suggests general stability, our results specifically confirm that a forgetting factor of $\lambda = 0.9$ is optimal for the rapid weather transitions observed in the Polish power system.

3.1. Performance of the one-layer LNN

The results for the one-layer LNN indicate moderate predictive capability but with substantial bias and variability in the forecasted power values.

As shown in Fig. 6, the predicted time series generally follow the overall trend of real wind generation but tend to underestimate peaks and occasionally overshoot low-generation intervals. The prediction curves are smoother than the real data, suggesting that the model struggles to capture short-term fluctuations and high-frequency dynamics inherent in wind generation.

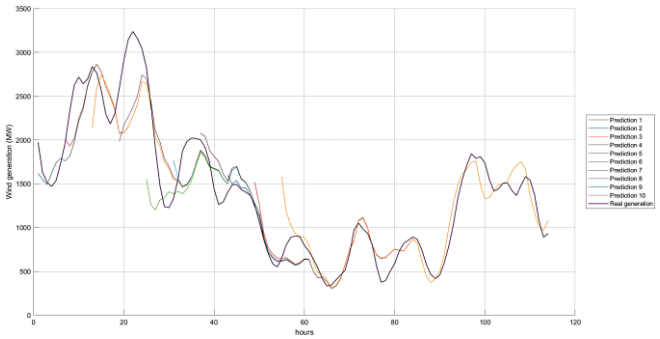

Fig.6. Predicted and real wind power generation for the one-layer LNN

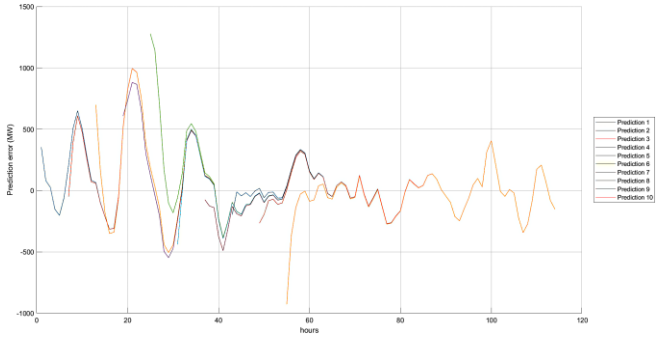

Fig.7. Temporal evolution of prediction errors for ten forecasts of the one-layer LNN

The temporal evolution of prediction errors in Fig. 7 demonstrates pronounced fluctuations, especially within the first 40 hours of the forecast horizon. This volatility highlights the model's limited temporal stability and its sensitivity to short-term variations in the wind regime.

The distribution of prediction errors (ΔP) presented in Fig. 8 is visibly skewed, with a mean deviation of approximately −473.5 MW and a standard deviation of 747.9 MW, confirming a systematic underestimation of wind power output. The histogram shows a wide spread, extending beyond ±2000 MW, indicating significant variability in accuracy over time.

The one-layer LNN exhibits noticeable temporal misalignment and smoothing as shown in Fig. 9.

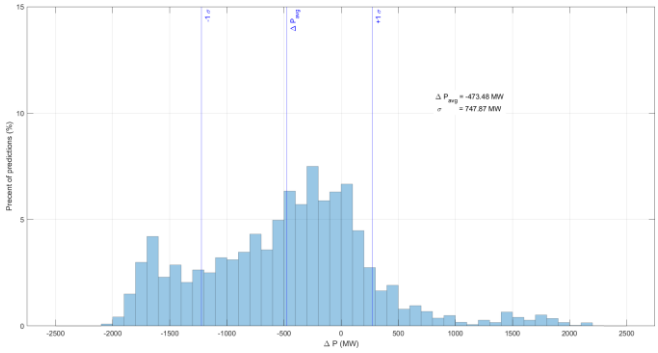

Fig.8. Histogram of prediction error (ΔP) for the one-layer LNN

Overall, the one-layer LNN exhibits the expected limitations of shallow recurrent architectures. While it captures general temporal profiles, it lacks the necessary dynamical complexity to model the chaotic, multiscale behavior of wind generation. The model's substantial error variance and systematic underestimation indicate an inadequate representation of spatiotemporal dependencies.

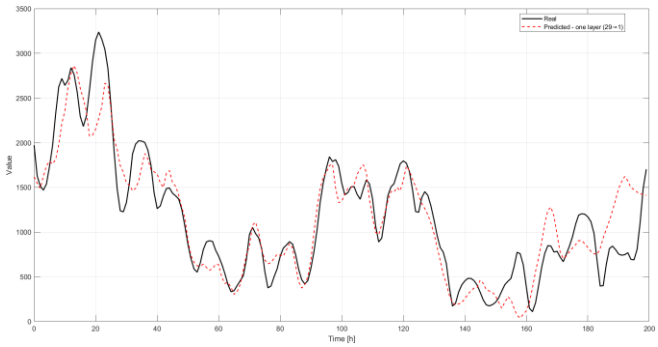

Fig.9. Real vs predicted wind generation for the one-layer LNN

3.2. Performance of the two-layer LNN

The two-layer LNN, incorporating an additional recurrent liquid layer, markedly improves forecasting accuracy. The comparison of predicted and actual generation in Fig. 10 shows a much closer alignment between the two, particularly in peak and trough regions. Unlike the single-layer model, the two-layer LNN more accurately captures amplitude and phase variations, reflecting superior temporal synchronization and higher dynamic fidelity.

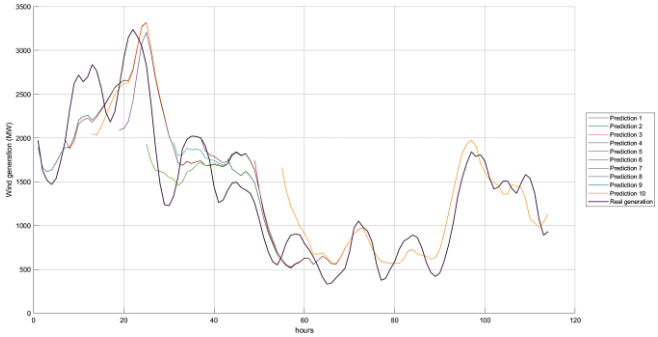

Fig.10. Predicted and real wind power generation for the two-layer LNN

The evolution of prediction errors over time, shown in Fig. 11, exhibits considerably lower amplitude and smoother oscillations than in the one-layer model. Most trajectories remain within ±500 MW, confirming improved forecast stability.

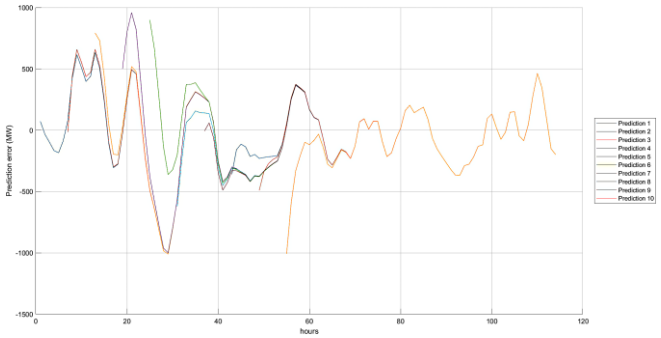

Fig.11. Temporal evolution of prediction errors for ten forecasts of the two-layer LNN

The error distribution for the two-layer LNN, shown in Fig. 12, is nearly symmetric around zero, with a mean deviation of −14.6 MW and a standard deviation of 475.5 MW. This represents a significant reduction in both bias and dispersion relative to the one-layer model.

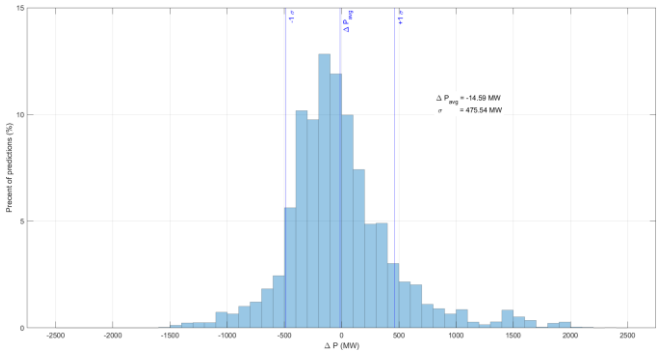

Fig.12. Temporal evolution of prediction errors for ten forecasts of the two-layer LNN

In contrast to the one-layer model, the two-layer model (Fig. 13) reproduces both high- and low-frequency components of the wind generation pattern with substantially greater precision. Overall, the two-layer architecture not only minimizes forecast bias but also enhances temporal consistency and noise resilience, demonstrating the advantages of a deeper liquid-state representation.

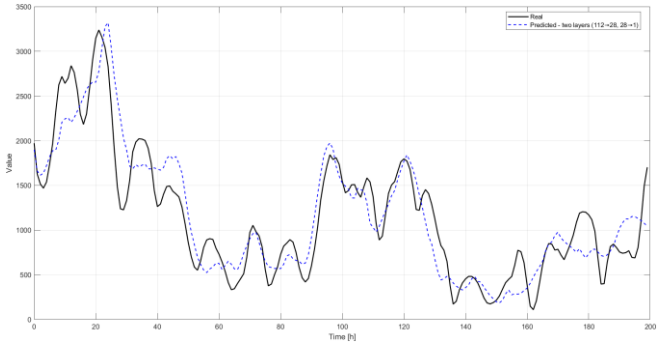

Fig.13. Real vs predicted wind generation for the two-layer LNN

The LNN model exhibits inherent robustness against input vector disturbances due to its continuous-time ODE-based hidden states. Unlike discrete-time RNNs, LTC layers naturally act as dynamic low-pass filters, smoothing transient noise and sudden perturbations in meteorological inputs. The inclusion of adaptive time constants (τ) allows the neurons to dynamically adjust their sensitivity to local signal fluctuations, thereby maintaining predictive stability during high-volatility events such as sudden wind gusts or rapid pressure changes.

3.3. Comparative analysis

To assess the effect of the number of training epochs on model performance, an epoch-wise analysis of both the training and generalization losses was performed. Figures 14 and 15 present the evolution of the loss values, expressed as log(MSE), over 400 training epochs for the one-layer and two-layer LNN architectures.

The training loss curves show that both models learn effectively throughout training. A rapid decrease in error is observed during the initial epochs, followed by a slower convergence phase. The two-layer architecture generally achieves lower training loss than the one-layer architecture, indicating its higher representation capacity and improved ability to approximate the training data.

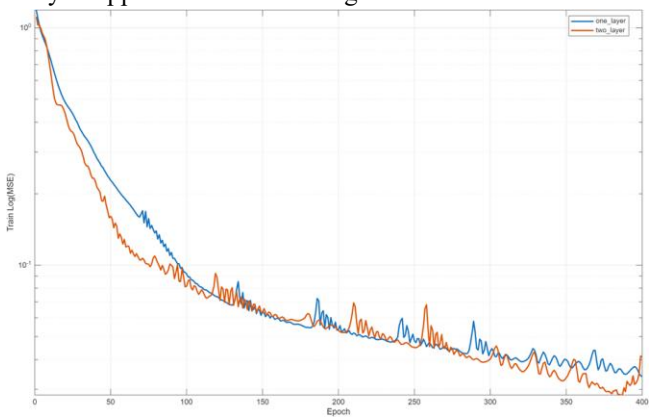

Fig.14. Training loss, expressed as log(MSE), as a function of the training epoch for the one-layer and two-layer LNN architectures. Both models show a substantial reduction in training error during the initial epochs, followed by gradual convergence. The two-layer architecture achieves lower training loss than the one-layer architecture

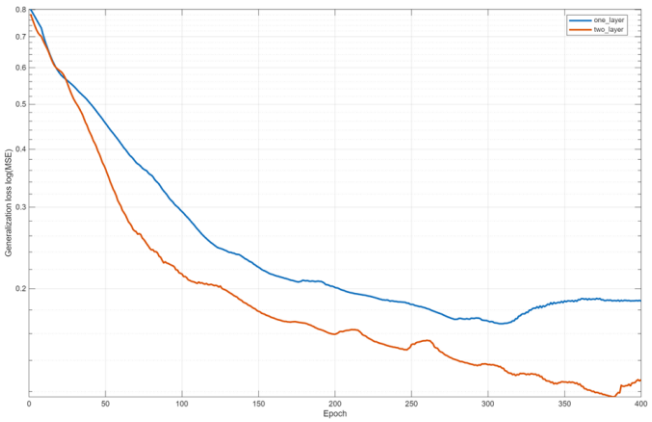

Fig.15. Generalization loss, expressed as log(MSE), as a function of the training epoch for the one-layer and two-layer LNN architectures. The one-layer model reaches its minimum generalization error around 280–310 epochs, after which a slight increase is observed, whereas the two-layer model continues to improve and achieves lower final generalization loss

However, the generalization loss reveals important differences between the two architectures. For the one-layer LNN, the generalization error decreases until approximately 280–310 epochs, then slightly increases, suggesting that additional training beyond this point may lead to mild overfitting. In contrast, the two-layer LNN shows a more consistent decrease in generalization error throughout training and achieves its best generalization performance near the final epochs. This indicates that the two-layer architecture benefits from an extended training horizon and exhibits better generalization.

These results confirm that the number of epochs is an important hyperparameter affecting generalization performance. Using 400 epochs was sufficient to observe convergence behavior for both architectures and compare their generalization trends. Nevertheless, the results also indicate that the optimal stopping point is architecture-dependent. For the one-layer model, early stopping based on the minimum generalization loss could further improve performance, whereas the two-layer model benefits from training for nearly the full 400 epochs. Consequently, the two-layer architecture was found to provide superior generalization performance under the analyzed training regime.

Figure 16 compares forecast error distributions for the one-layer and two-layer LNN models, both fitted with a Gaussian curve.

The errors for the one-layer LNN model are shown as blue points, with the corresponding Gaussian fit plotted as a blue curve. This distribution exhibits a moderately dispersed pattern, with a maximum error frequency of around 7%, is reasonably symmetric around the mean, and shows only minor deviations in the left tail. This indicates that extreme forecasting errors are relatively infrequent.

The errors for the two-layer LNN model are displayed as red points, with the corresponding Gaussian fit drawn as a red curve. This distribution shows a sharply pronounced central peak, with the highest frequency exceeding 12%, suggesting that many forecasts closely match the actual values. However, it also exhibits heavier tails, indicating occasional extreme deviations and a leptokurtic structure.

Overall, the Fig. 16 allows a direct visual comparison of the two Gaussian fits, highlighting differences in dispersion, peak height, and tail behavior.

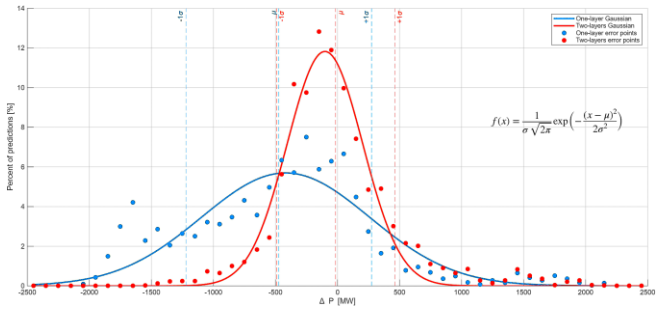

Fig.16. Distribution of forecast errors for one-layer and two-layer LNN models, with fitted Gaussian curves

TABLE 1. Gaussian fit parameters for the forecast error distributions of the one-layer and two-layer LNN models

Model	μ	σ	$\mu - \sigma$	$\mu + \sigma$
one-layer LNN	-473.5	747.8	-1221.4	274.4
two-layer LNN	-14.6	475.5	-490.1	460.9

Table 1 presents the parameters of the Gaussian fits for the forecast error distributions of the one-layer and two-layer LNN models. The mean (μ) indicates the average forecast error, with the one-layer model showing a larger negative bias of -473.5 MW, compared to -14.6 MW for the two-layer model, suggesting the latter achieves better overall accuracy. The standard deviation (σ) quantifies the spread of errors. The one-layer model exhibits a larger dispersion of 747.9 MW, whereas the two-layer model has a smaller dispersion of 475.5 MW, indicating more consistent forecasts. The ranges defined by $\mu - \sigma$ and $\mu + \sigma$ further illustrate these differences: the one-layer model spans from -1221 MW to 274 MW, whereas the two-layer model extends from -490.1 MW to 460.9 MW. Overall, the table shows that the two-layer LNN model produces errors closer to zero and less dispersed, in agreement with the visual comparison of the Gaussian fits shown in the Fig. 16.

The RMSE decreased from approximately 785 MW to 430 MW (Fig. 17a), the MAE from 690 MW to 340 MW (Fig. 17b), and the MSE from $7.8.10^5$ to $2.3.10^5$ (Fig. 17c). Similarly, the MAPE was reduced from 167% to 61% (Fig. 17d).

The three-dimensional RMSE surface plots (Fig. 18) further elucidate the models' dynamic behavior. In the one-layer case, RMSE values exhibit substantial irregularity across prediction iterations, reflecting instability and sensitivity to initialization. Conversely, the two-layer LNN produces a smoother, more uniform surface, with reduced gradient magnitude and fewer localized peaks.

This indicates greater robustness and generalization across different neural architectures and prediction steps.

The improvement achieved by the two-layer LNN can be attributed to its enhanced representational capacity and hierarchical state dynamics. The additional liquid layer expands the system's state space, enabling it to encode multiscale temporal correlations that the one-layer model cannot capture. The substantial reduction in mean bias (from −473.5 MW to −14.6 MW) and variance demonstrates that the deeper model better learns balanced temporal dependencies between inputs and outputs. In practice, this advancement yields a more reliable and precise forecasting tool for wind power operators. By producing predictions that are both accurate and dynamically stable, the two-layer LNN provides significant operational benefits for energy dispatch and grid management. Nonetheless, minor residual discrepancies during rapid power transitions suggest that further improvements could be achieved by integrating probabilistic or ensemble-based approaches to account for atmospheric uncertainty.

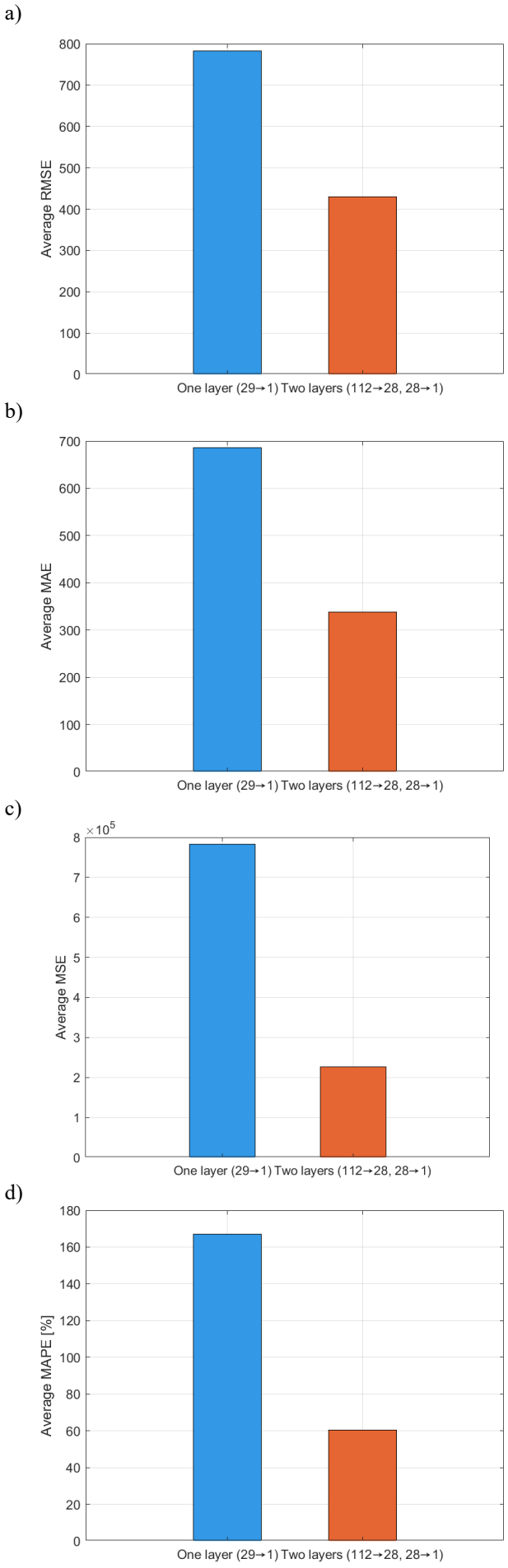

Fig.17. Comparison of average: a) RMSE, b) MAE, c) MSE, d) MAPE

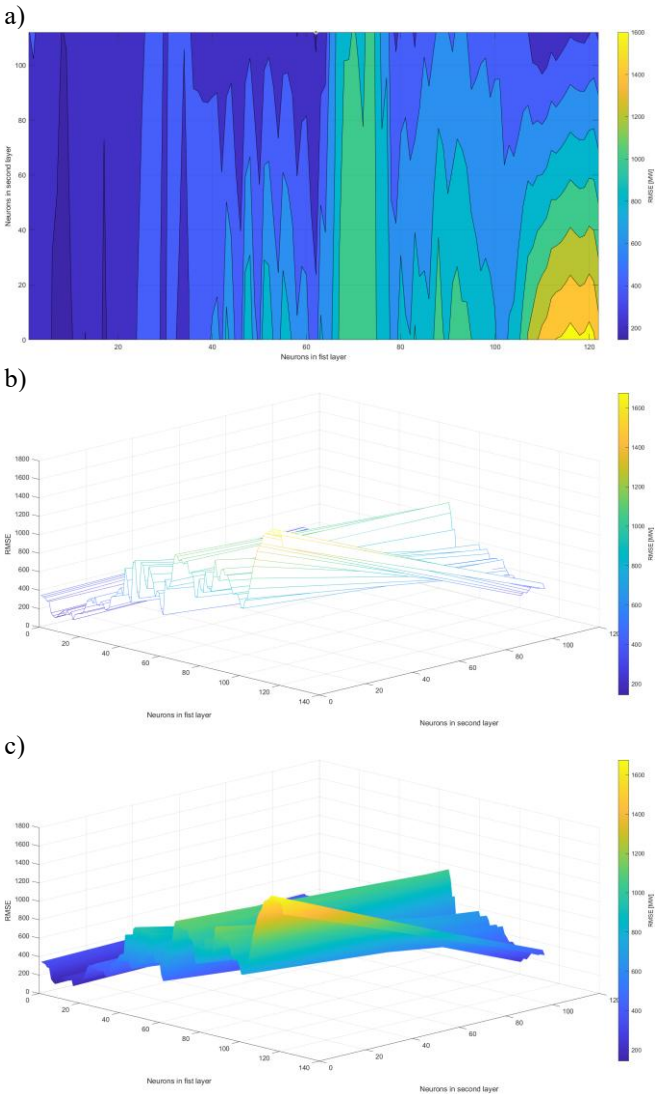

Fig.18. RMSE plots comparing one-layer and two-layer LNNs: a) contour, b) mesh, c) surface

The significance of structural complexity in the one-layer Liquid Neural Network (LNN) was systematically evaluated during hyperparameter tuning using the Optuna framework. For this architecture, the number of neurons was optimized over a range of 3 to 112, and the process identified 29 neurons as the optimal configuration for minimizing Root Mean Squared Error (RMSE). While this level of complexity is sufficient to capture the general temporal profiles and basic patterns of wind generation, it lacks the dynamical capacity required to model the chaotic, multiscale variations of the Polish power grid. Quantitatively, despite optimization, the single-layer model exhibited substantial error variance, with a mean deviation of approximately −473.5 MW and a standard deviation of 747.9 MW. These results indicate that within the one-layer framework, increasing structural complexity (neuron count) is secondary to architectural depth; even an optimized single-layer configuration cannot match the stability and bias reduction achieved by the two-layer architecture.

3.4. Performance of the Recursive Aggregation Method

Analysis of the experimental results enables evaluation of the effectiveness of the proposed prediction methods and the Recursive Least Squares (RLS)–based fusion algorithm, compared with baseline models and static weighting methods (mean and median). Key conclusions were drawn from the root-mean-square error (RMSE) curve and from direct observation of the predicted time series. Visualization of the RMSE values as a function of prediction number (Fig. 19) reveals significant differences in the stability of the analyzed models. The single-layer model tends to significantly degrade its accuracy over time, as evidenced by a sharp increase in error beyond the 80th prediction.

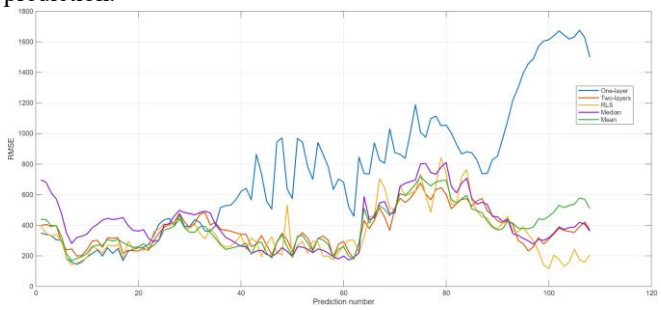

Fig.19. Comparison of Root Mean Square Error (RMSE) for baseline models and various prediction fusion strategies

This phenomenon suggests that the static model cannot adapt to the evolving dynamics of the real signal. In contrast, the two-layer model and the fusion methods maintain the error at a relatively constant, lower level. The RLS algorithm effectively minimizes the impact of the less accurate single-layer model by dynamically selecting weights that favor the model with the lower instantaneous error. It is worth noting that the RLS method, despite periodic oscillations arising from the adaptation process (e.g., around the 50th and 70th predictions), quickly returns to a stable state, confirming the effectiveness of the forgetting factor $\lambda = 0.9$ in trend tracking.

Figure 20 compares the predictions with the actual values over a 200-hour time horizon. Observation of the waveforms confirms that the RLS adaptive fusion best reproduces the dynamics of the actual signal, especially at extreme points (local maxima and minima). The single-layer model (dashed red line) exhibits the greatest deviations, especially in the final phase of the analyzed period, which corresponds to the high RMSE values shown in the first graph.

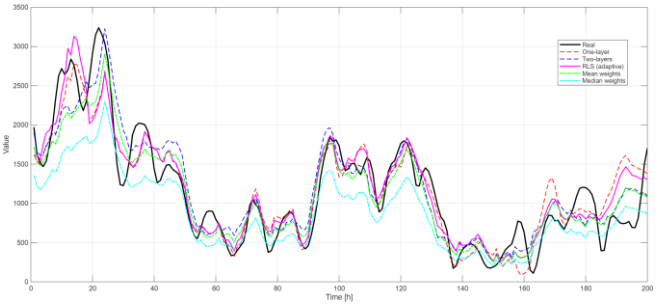

Fig.20. Time-series characteristics of baseline model predictions and fusion algorithms compared against absolute values over a 200-hour horizon

Methods based on global weights (mean and median) serve a stabilizing role, but they tend to underestimate signal amplitude, particularly with median weights. Because the median is a robust estimator for outliers, it filters out too much of the dynamic range, leading to a systematic underestimation of peak values. The adaptive nature of RLS avoids this problem because the weights are updated at each time step, allowing "switching" between models based on which better describes the current state of the process.

Finally, the area under the RMSE curve using the trapezoidal method and its average were calculated individually for each technique. The results in Tab. 2 indicate that the RLS and two-layer methods differ significantly from the other methods, achieving the lowest errors.

TABLE 2. Mean RMSE area values

Model	Mean RMSE area value
one-layer	660.5
two-layer	328.4
median	363.4
mean	336.9
RLS	308.7

The models compared in Table 3 are categorized into established global forecasting benchmarks and the specific architectures proposed in this research. The first group includes DeepAR, a probabilistic forecasting framework based on autoregressive recurrent networks, DeepVAR, which leverages deep neural networks for multivariate time-series risk assessment, and N-HiTS, which utilizes neural hierarchical interpolation to capture multi-rate periodicities. The second group consists of the study's primary architectures: the one-layer and two-layer LNNs, which were independently optimized to evaluate the impact of network depth on non-stationary wind data. Finally, the aggregation strategies include Mean and Median weights, serving as static ensemble benchmarks, and the proposed RLS (Recursive Least Squares) framework. The RLS model represents the study's target methodology, using an adaptive weighting mechanism to dynamically prioritize the most accurate base model in real time.

The deep learning approaches (N-HiTS, DeepVAR, DeepAR) and aggregation models (one-layer, two-layer, median, mean, RLS) exhibit substantial variability in both error magnitude and dispersion. Among the methods, DeepVAR and DeepAR achieve positive MAE and MAPE values with relatively moderate standard deviations, indicating more stable performance. In contrast, N-HiTS, one-layer, and two-layer models yield negative error metrics, reflecting systematic bias in their forecasts. Ensemble-type strategies (median, mean, RLS) exhibit comparable error metrics, with RLS achieving the lowest MAE. Overall, the results highlight considerable heterogeneity in predictive accuracy and robustness across the evaluated models.

TABLE 3. Comparison of prediction analysis for prediction techniques: N-HiTS, DeepVAR, DeepAR [17], and one-layer, two-layer, median, mean, RLS

Model	MAE	Standard deviation of MAE	MAPE	Standard deviation of MAPE
N-HiTS	-499.0 MW	578.0 MW	-125.3%	181.0%
DeepVAR	51.4 MW	283.5 MW	30.5%	-1.0%
DeepAR	97.5 MW	251.2 MW	16.2%	25.5%
one-layer	-473.5 MW	747.9 MW	-156.5%	268.0%
two-layer	-14.6 MW	475.5 MW	-38.4%	146.2%
median	379.2 MW	440.6 MW	57.7%	107.9%
mean	363.7 MW	349.7 MW	70.3%	138.8%
RLS	291.3 MW	311.4 MW	55.2%	162.0%

To provide a comprehensive evaluation, the models in Table 3 are categorized by their underlying architectural paradigms. DeepAR and DeepVAR are state-of-the-art recurrent frameworks; DeepAR uses autoregressive RNNs for univariate probabilistic forecasting, while DeepVAR extends this approach to multivariate datasets. These serve as the primary benchmarks for comparing LNNs against traditional discrete-time recurrent networks. In contrast, N-HiTS is included as a non-recurrent baseline that employs neural hierarchical interpolation to handle multi-scale temporal patterns. The remaining models comprise the study's proposed one- and two-layer LNNs, static ensemble methods (Mean and Median), and the target RLS adaptive fusion framework. This diverse selection ensures that the continuous-time dynamics of LNNs are benchmarked against both modern recurrent and non-recurrent time-series architectures.

4. CONCLUSIONS

This study evaluated the performance of one-layer and two-layer Liquid Neural Networks (LNNs) for short-term wind power forecasting using meteorological predictors over a 60-hour horizon. The comparison reveals that network depth substantially enhances predictive accuracy, stability, and bias reduction.

The one-layer LNN captured the general temporal pattern of wind generation but exhibited significant bias and large variance, with a mean underestimation of approximately −473 MW and an MAE exceeding 1400 MW. Its limited dynamic capacity led to overly smooth forecasts and poor responsiveness to rapid changes in wind conditions. As illustrated in the RMSE analysis, the one-layer model exhibits a critical lack of long-term stability, with error values increasing sharply beyond the 80th prediction step and peaking at over 1600 MW. This divergence is further confirmed by the time-series plot, which shows that the one-layer prediction fails to closely follow the actual power trajectory over the final 40 hours of the simulation.

In contrast, the two-layer LNN demonstrated a marked improvement across all evaluation metrics. The MAE decreased to 295 MW, the RMSE to 430 MW, and the MAPE to 61%, representing more than a 60% reduction relative to the shallower model. The near-zero mean bias (−14.6 MW) indicates balanced predictions and improved generalization. The results suggest that the two-layer architecture maintains consistently lower, more stable RMSE across the entire horizon, thereby suppressing the error spikes observed in the single-layer architecture. The smoother RMSE surfaces and error contours further confirm the enhanced stability and robustness of the deeper architecture.

The study further explored the potential of adaptive fusion strategies to optimize these predictions. The implementation of the Recursive Least Squares (RLS) algorithm demonstrated the highest degree of alignment with real-world data, particularly during periods of high volatility. By dynamically updating model weights, the RLS approach mitigated the individual weaknesses of the base models, yielding a superior fit compared with static weighting methods such as global mean or median weights.

From an operational perspective, the improved performance of the two-layer LNN and its adaptive fusion variants suggests its suitability for real-world energy forecasting systems, where reliable short-term predictions are essential for grid balancing and market planning. Future work should explore probabilistic extensions and spatiotemporal modeling further to enhance forecast reliability under highly variable atmospheric conditions.

In conclusion, the results confirm that deeper LNN architectures, when enhanced by adaptive RLS fusion, provide a robust and precise tool for grid balancing and energy dispatch. Future research will focus on integrating probabilistic extensions to further account for atmospheric uncertainty.

5. REFERENCES

[1] Y. Wang, R. Zou, F. Liu, L. Zhang, and Q. Liu, "A review of wind speed and wind power forecasting with deep neural networks," *Appl. Energy*, vol. 304, p. 117766, Dec. 2021, doi: 10.1016/J.APENERGY.2021.117766.

[2] G. Li, C. Lin, and Y. Li, "Probabilistic Forecasting of Provincial Regional Wind Power Considering Spatio-Temporal Features," *Energies 2025, Vol. 18, Page 652*, vol. 18, no. 3, p. 652, Jan. 2025, doi: 10.3390/en18030652.

[3] J. Keisler and E. Le Naour, "WindDragon: Enhancing wind power forecasting with Automated Deep Learning," Feb. 2024, Accessed: Feb. 03, 2026. [Online]. Available: http://arxiv.org/abs/2402.14385

[4] W. Liao, J. Fang, L. Ye, B. Bak-Jensen, Z. Yang, and F. Porte-Agel, "Can we trust explainable artificial intelligence in wind power forecasting?," *Appl. Energy*, vol. 376, p. 124273, Dec. 2024, doi: 10.1016/J.APENERGY.2024.124273.

[5] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *Int. J. Forecast.*, vol. 36, no. 3, pp. 1181–1191, Jul. 2020, doi: 10.1016/J.IJFORECAST.2019.07.001.

[6] G. Fatouros, G. Makridis, D. Kotios, J. Soldatos, M. Filippakis, and D. Kyriazis, "DeepVaR: a framework for portfolio risk assessment leveraging probabilistic deep neural networks," *Digital Finance 2022 5:1*, vol. 5, no. 1, pp. 29–56, Apr. 2022, doi: 10.1007/S42521-022-00050-0.

[7] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. Mergenthaler-Canseco, and A. Dubrawski, "N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting," *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, vol. 37, pp. 6989–6997, Jan. 2022, doi: 10.1609/aaai.v37i6.25854.

[8] S. B. Shaheema, S. D. Suganya Devi, and N. B. Muppalaneni, "An explainable Liquid Neural Network combined with path aggregation residual network for an accurate brain tumor diagnosis," *Computers and Electrical Engineering*, vol. 122, p. 109999, Mar. 2025, doi: 10.1016/J.COMPELECENG.2024.109999.

[9] S. Narvare and K. B. Meena, "LiqU-Net: A liquid neural network-enhanced U-Net for efficient copy-move forgery detection and localization," *Appl. Soft Comput.*, vol. 185, p. 114057, Dec. 2025, doi: 10.1016/J.ASOC.2025.114057.

[10] N. Aslan, "A novel hybrid liquid neural network with time dependent neuromorphic and transformer encoder based reservoir dynamics for respiratory sound classification," *Eng. Appl. Artif. Intell.*, vol. 162, p. 112728, Dec. 2025, doi: 10.1016/J.ENGAPPAI.2025.112728.

[11] N. Osiecka-Drewniak, M. Piwowarczyk, and M. Gałązka, "Key factors in neural network performance for liquid crystal texture classification," *J. Mol. Liq.*, vol. 428, p. 127511, Jun. 2025, doi: 10.1016/J.MOLLIQ.2025.127511.

[12] G. Antonesi, T. Cioara, I. Anghel, I. Papias, V. Michalakopoulos, and E. Sarmas, "Hybrid transformer model with liquid neural networks and learnable encodings for buildings' energy forecasting," *Energy and AI*, vol. 20, p. 100489, May 2025, doi: 10.1016/J.EGYAI.2025.100489.

[13] G. Chen, Y. Liao, D. Zhang, W. Yang, Z. Mai, and C. Xu, "Multimodal Emotion Recognition via the Fusion of Mamba and Liquid Neural Networks with Cross-Modal Alignment," *Electronics 2025, Vol. 14, Page 3638*, vol. 14, no. 18, p. 3638, Sep. 2025, doi: 10.3390/ELECTRONICS14183638.

[14] M. H. Akpinar, O. Atila, A. Sengur, M. Salvi, and U. R. Acharya, "A novel uncertainty-aware liquid neural network for noise-resilient time series forecasting and classification," *Chaos Solitons Fractals*, vol. 193, p. 116130, Apr. 2025, doi: 10.1016/J.CHAOS.2025.116130.

[15] J. M. Worsham and J. K. Kalita, "A guide to neural ordinary differential equations: Machine learning for data-driven digital engineering," *Digital Engineering*, vol. 6, p. 100060, Sep. 2025, doi: 10.1016/J.DTE.2025.100060.

[16] Y. Wang, Q. Hu, L. Li, A. M. Foley, and D. Srinivasan, "Approaches to wind power curve modeling: A review and discussion," *Renewable and Sustainable Energy Reviews*, vol. 116, p. 109422, Dec. 2019, doi: 10.1016/J.RSER.2019.109422.

[17] W. Jachuła and M. Wydra, "Wind power prediction in Poland using temporal fusion transformers and numerical weather prediction," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 73, no. 5, Aug. 2025, doi: 10.24425/BPASTS.2025.155038.

[18] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Liquid Time-constant Networks," *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, vol. 9A, pp. 7657–7666, Jun. 2020, doi: 10.1609/aaai.v35i9.16936.

[19] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," *NIPs*, vol. 109, no. NeurIPS, pp. 31–60, Jun. 2018, Accessed: Apr. 16, 2025. [Online]. Available: https://arxiv.org/abs/1806.07366v5

[20] S. Tian, L. Qu, L. Wang, K. Hu, N. Li, and W. Xu, "A neural architecture search based framework for liquid state machine design," *Neurocomputing*, vol. 443, pp. 174–182, Jul. 2021, doi: 10.1016/j.neucom.2021.02.076.

[21] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front. Comput. Neurosci.*, vol. 9, no. AUGUST, p. 149773, Aug. 2015, doi: 10.3389/FNCOM.2015.00099/BIBTEX.

[22] T. Muthukumaran *et al.*, "Ionospheric TEC prediction in low-latitude Indian region during geomagnetic storm periods based on XGBoost with optuna framework and comparison with IRI-Plas 2020," *J. Atmos. Sol. Terr. Phys.*, vol. 275, p. 106606, Oct. 2025, doi: 10.1016/J.JASTP.2025.106606.

[23] L. Barillaro, "Deep Learning Platforms: Keras," *Encyclopedia of Bioinformatics and Computational Biology*, pp. 163–166, Jan. 2025, doi: 10.1016/B978-0-323-95502-7.00092-0.