A novel uncertainty-aware liquid neural network for noise-resilient time series forecasting and classification

A Novel Uncertainty-Aware Liquid Neural Network for Noise-Resilient Time Series Forecasting and Classification

Muhammed Halil Akpinar[1], Orhan Atila[2], Abdulkadir Sengur[2], Massimo Salvi[3], U. R. Acharya [4,5]

[1] Department of Electronics and Automation, Vocational School of Technical Sciences, Istanbul University-Cerrahpasa, Istanbul, Turkey
[2] Electrical-Electronics Engineering Department, Technology Faculty, Firat University, Elazig, Turkey
[3] PolitoBIOMed Lab, Biolab, Department of Electronics and Telecommunications, Politecnico di Torino
[4] School of Mathematics, Physics and Computing, University of Southern Queensland, Springfield, Australia
[5] Centre for Health Research, University of Southern Queensland, Springfield, QLD 4300, Australia

Abstract

While Liquid Neural Networks (LNN) are promising for modeling dynamic systems, there is no internal mechanism that quantifies the uncertainty of a prediction. This can produce overly confident outputs, especially when operating in noisy or uncertain environments. One potential issue that might be highlighted with LNNs is that their highly flexible connectivity leads to overfitting on the training data. This is targeted by the present work, which introduces the uncertainty-aware LNN framework, the UA-LNN, by considering Monte Carlo dropout for quantifying the uncertainty of LNNs. The proposed UA-LNN enhances the stochasticity of both training and inference, hence allowing for more reliable predictions by modeling output uncertainty. We applied the UA-LNN in the two tasks of time series forecasting and multi-class classification, where we showed its performance on a wide range of datasets and under different noise conditions. The proposed UA-LNN has shown the best results, outperforming the benchmarks of standard LNN, Long Short-Term Memory (LSTM) and Multilayer Perceptron (MLP) models in terms of R^2, RMSE, and MAE consistently. Additionally, for performance metrics such as accuracy, precision, recall, and F1 score, the results showed improvement over LSTM and MLP models in multi-classification tasks. More importantly, under heavy noise, the UA-LNN maintained superior performance, while demonstrating enhanced classification capabilities across many datasets with challenging tasks, such as arrhythmia detection and cancer classification.

Keywords: Liquid neural network, Uncertainty, Monte Carlo dropout, prediction, classification.

1. Introduction

The Liquid Neural Network (LNN) is a specialized class of recurrent neural network (RNN) that can dynamically modify its inner structure at runtime based on input data, making it particularly suitable for controlling dynamic or time-dependent systems [1]. In contrast with neural networks of any other traditional architecture, LNNs have dynamic connectivity, and hence they may change their inner state in response to real-time signals [2]. This attribute renders LNN most suitable for application areas like

time-series prediction, signal analysis, and dynamic control systems involving sequence data [3]. These characteristics make LNN useful in robotics, natural language processing, and brain-inspired computing tasks that require the modeling of complex nonlinear temporal dependencies [4]. These methodologies are particularly valuable in critical applications like medical diagnosis, autonomous driving, and financial forecasting, where uncertainty quantification provides immense meaning to decision-making processes [5].

To date, numerous diverse research studies have been conducted using LNN for prediction and classification purposes. Udumula et al. [6] discussed a real-time object detection approach using LNN and echo state network architecture. The model presented here showed better performance when compared to all previous neural network models in object detection algorithms of autonomous vehicles, as it has the ability to include dynamical environments. It is also faster in detection and more accurate compared to the previous neural network models. Karn et al. [7] proposed the architecture of a Generalized LNN, beyond conventional sequential processing. Their contributions involved the redefinition of task management in LNN, an implementation of the Runge-Kutta DOPRI 5 method for equilibrium operations, and verification based on various case studies. The efficacy of the framework was shown for the prediction of damped sinusoidal trajectories, very common in oscillatory systems. It was further used in the estimation of the output of nonlinear RLC circuits, proving that it is robust enough to handle the complexities of an electronic system. Studies have been conducted using the GLNN for classifying retinal diseases from OCT images, showing promising potential in healthcare applications.

Gajjar et al. [8] developed the Liquid Time Constant Network (LTCN) model as an alternative to Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) with a view to enhance stock price forecasting. The Liquid Time Constant Network (LTCN) model improved the forecasting accuracy along with computational efficiency by exploiting the bidirectionality property in time series forecasting models. The developed LTCN model is compared with traditional models like RNN, Gated Recurrent Units (GRU), LSTM [9], and their bidirectional versions: BiLSTM [10]. The authors demonstrated that the developed LTCN achieved superior prediction performance and computational efficiency compared to other models.

An alternative model for intrusion detection in IoT-H networks was proposed by Kramarczyk et al. [11]. In this work, the authors proposed the Liquid NeorIoTic, which is an intrusion detection system for IoT-H networks based on LNN. They adopted the new CICIoT2023 dataset and trained and tested an LNN able to identify and classify many cyber-attacks such as Distributed Denial of Service (DdoS), spoofing, and reconnaissance. A new methodology was accordingly proposed for the conversion of all network traffic features into cyclic waveforms so that a time series representation could be availed in the LNN model. Experimental results showed that LNN can reduce as much as 40% neurons and as much as 64% epochs from the best performing ANN and RNN.

Hasani et al. [12] proposed a closed-form continuous-time neural network, which was made up of Liquid Time-Constant (LTC) layers and played a very important role in the development. This approach approximates neurons and synapse interaction in a continuous time frame, beating the regular RNN in many aspects-most especially time series and irregular data. It was tested on a variety of datasets for a range of tasks, whether classification or prediction. Indeed, the experimental results showed that the model outperformed the numeric solver 1 to 5 times in speed with equivalent performances. Pendyala et al. [13] have also proposed an explainable prediction model by making use of the combination of LTCN and symbolic regression to predict link quality. This model, given as an alternative to the classical RNN and LSTM methods, was tested at 28 GHz in a scenario with moving people and obstacles, and the results showed that LTCN outperformed LSTM by 13 times. Chahine et al. [14] have proposed a navigation mechanism based on LNN for vision-based flight control and target orientation tasks that adapt to changes in their environment and, while accomplishing a certain task, focus their attention only on data relevant for the completion of that particular task. The proposed approach was evaluated through flight-to-target, distance, challenging conditions, dynamic target tracking, and efficiency tests. These tests demonstrated superior performance of neural networks in handling environmental changes, challenging conditions, and distribution shifts, consistently outperforming conventional models.

Wahab et al. [15] presented a novel deep-learning method for pain detection using facial expressions. An LNN-based image extraction technique selected key frames from video files. A hybrid method using DenseNet 201 and MobileNet V3 models was applied in the feature extraction phase. In another work, the Lucey et al. [16] pain score was used to monitor pain intensity, and the model was trained using the LightGBM model, fine-tuned with a random search algorithm. Experiments using the Denver Spontaneous Facial Movement Intensity (DISFA) dataset achieved % classification accuracy of 97.8% with the proposed model. Furthermore, it was demonstrated that this model, which requires few parameters and computational resources, can be applied even to devices with limited capabilities.

LNN has demonstrated significant advantages in modeling time-varying and dynamic systems. However, despite the progress in this domain, LNNs still face notable challenges. A significant limitation of traditional LNNs is their inability to accurately estimate uncertainty in predictions, which can result in overconfident outputs.

The top fluid connectivity in LNNs, while effectively handling dynamic inputs, tends to promote memorization instead of generalization. This makes them susceptible to overfitting, especially if the training datasets are not big enough. However, their adaptive structure may lead to unstable training processes, resulting in divergence from optimal solutions. A majority of the open challenges were directed toward methods with which the robustness and reliability of LNNs could be improved, mainly in applications where accurate uncertainty quantification is desired or required. This work proposes an uncertainty-aware LNN, a variant designed to improve prediction uncertainty estimation. In this paper, we use one of the approaches that introduces stochasticity during the training and testing of the UA-LNN: MC dropout. Let the internal signal be the projection of the input signal at each time step using

the input weight matrix. Given previous states and input transformations, the neuron states are updated in a recurrent fashion with dropout applied to the recurrent weight matrix, which amounts to dynamic changes in network structure. During training, multiple MC samples are generated to create an ensemble of predictions that capture model stochasticity. For each sample, the loss is computed as a squared difference between model output and the actual target value. Then, gradients are computed to perform the iterative update via stochastic gradient descent. After training, the final prediction and uncertainty estimates for any test example are obtained by performing multiple samplings using MC dropout during inference. The variability in those samples reflects the model confidence in those predictions, hence serving as a measure of uncertainty. The contribution of this work is as follows:

1. The proposed UA-LNN model improves the reliability of prediction through uncertainty quantification, particularly in scenarios with noisy or incomplete data.
2. In the suggested framework, there is a fluid architecture that updates its internal parameters continuously in response to coming data. This adaptability enables the capture the dynamics associated with nonlinear time-varying interactions, which makes UA-LNN particularly suitable for time-series forecasting in dynamic contexts.
3. UA-LNN incorporates uncertainty awareness, which gives information on how sure the model is regarding a given prediction, thus enhancing the interpretability for end users.
4. UA-LNN demonstrates robust performance across various level of noise in both prediction and classification tasks.
5. We conduct a comprehensive comparison of the UA-LNN with standard LNN, LSTM, and MLP models across various datasets and noise levels, illustrating its enhanced performance and adaptability.

2. Material and Methods

2.1. Liquid Neural Networks (LNN)

LNNs represent a particular class of RNNs designed to model dynamic and temporally varying systems through real-time adaptation of their internal structure [17]. Unlike other traditional RNNs, LNNs use dynamic neuron states controlled using differential equations, which allow continuous updating of the neuron states based on time-varying inputs. The temporal evolution of the neuron states can be mathematically expressed as:

$$\dot{s}(t) = f(s(t), u(t), W) \tag{1}$$

where $s(t) \in \mathrm{R}^N$ represents the neuron states at time t and $u(t) = W_{in} \times x(t)$ is the transformed input, with W being the weight matrices. The function $f(\cdot)$, often represented by a non-linear activation function such as tanh, governs the transitions of the neuron states based on the current state and input. The state update at each time step, considering both recurrent dynamics and input, is given by:

$$s(t + 1) = \sigma(W_r \times s(t) + u(t)) \quad (2)$$

Here, $W_r \in R^{N \times N}$ is the recurrent weight matrix, capturing the temporal dependencies within the input signal. The activation function $\sigma(\cdot)$, such as tanh, introduces non-linearity to the system. The predicted output $\hat{y}(t)$ is computed as:

$$\hat{y}(t) = W_{out} \cdot s(t) \quad (3)$$

where $W_{out} \in R^{1 \times N}$ is the output weight matrix that maps the neuron states to the final prediction. Training the network involves minimizing the Mean Squared Error between the predicted output $\hat{y}(t)$ and the actual signal $x(t + 1)$ at the next time step:

$$L(t) = (\hat{y}(t) - x(t + 1))^2 \quad (4)$$

Backpropagation is employed to update the weights. The gradient of the loss concerning the output is given by:

$$\frac{\partial L}{\partial \hat{y}(t)} = 2.(\hat{y}(t) - \mathrm{x(t + 1)}) \quad (5)$$

This gradient is used to update the weights W_{out} and W_{in} through gradient descent. Gradient clipping is applied to prevent the gradients from becoming too large, ensuring training stability. The network adjusts its weights over several epochs to minimize the total loss, improving its ability to predict the sine wave signal.

2.2. Uncertainty-Aware Liquid Neural Networks (UA-LNN)

The proposed UA-LNN extends the standard LNN by incorporating MC dropout [18], which serves three key purposes in our framework:

1. During training, MC dropout randomly deactivates neurons with probability p (set to 0.1 in our implementation) to prevent co-adaptation of features and reduce overfitting.
2. During inference, instead of disabling dropout as in traditional networks, we maintain it active and perform multiple forward passes (20 in our implementation) to generate an ensemble of predictions.
3. The variance across these multiple predictions provides a measure of model uncertainty - higher variance indicates less confidence in the prediction, while lower variance suggests higher confidence.

2.2.1 Forward Pass and Monte Carlo Sampling

At every time step t, the input signal x(t) is convolved with the input weight matrix W_{in}, yielding a transformed input signal u_t:

$$u_t = W_{in} . \mathrm{x(t)} \quad (6)$$

The state of the neurons, s_t is updated recurrently based on the previous state and the transformed input. Dropout is applied to the recurrent weight matrix W_r creating a stochastic recurrent matrix W_{rd}. The neuron state update is then performed as:

$$s_{t+1} = \sigma(W_r \,.\, s_t + \, u_t) \tag{7}$$

where $\sigma(.)$ is an element-wise activation function such as sigmoid or ReLU. This equation models the recurrent dynamics of the network, incorporating both the previous state s_t and input u_t. This dropout mechanism generates multiple versions of the recurrent weight matrix in training to prevent overfitting and improve generalization.

During training, multiple MC samples are generated at each time step, from which the network produces different outputs:

$$\hat{y}_{me} = W_{out} . s_{t+1} \tag{8}$$

where $\hat{y}_{me}$ is the predicted output for a single MC sample. This process yields an ensemble of predictions, capturing the model's stochasticity induced by dropout.

2.2.2 Loss Computation and Weight Update

The loss for each MC sample is computed as the squared difference between the predicted output and the actual target value $y(t+1)$:

$$L_{me} = \, -(\hat{y}_{me} - y(t+1))^2 \tag{9}$$

The total loss at each time step is the average of losses across all MC samples. Gradients are computed with respect to the output weights W_{out} and the input weights W_{in}. The gradient for the output weight matrix is given by:

$$\Delta W_{out} = \, -\sum_{me} (\hat{y}_{me} - y(t+1)) . s_{t+1}^T \tag{10}$$

The gradient related to the input weights W_{in} is calculated by backpropagating the loss through the network. First, the gradient for the neuron state s_{t+1} is computed as:

$$s_{grad} = \, W_{out}^T . (\hat{y}_{me} - y(t+1)) \tag{11}$$

This gradient is then used to update the input weights:

$$\Delta W_{in} = \sum_{me} (s_{grad} \,.\, x(t)^T) \tag{12}$$

Both gradients ΔW_{out} and ΔW_{in} are averaged over all Monte Carlo samples. The weights are then updated using stochastic gradient descent with a learning rate η:

$$W_{out} = \, W_{out} - \, \eta . \Delta W_{out} \tag{13}$$

$$W_{in} = \, W_{in} - \, \eta . \Delta W_{in} \tag{14}$$

This update is performed iteratively for each time step across all epochs, refining the network's parameters.

Figure 1 shows the general flowchart of the proposed UA-LNN algorithm. The process begins by initializing the model at the "Start" point. For each epoch, the model progresses through a series of time steps. At each time step, multiple MC dropout samples are generated to account for model uncertainty. Each MC sample is then processed through UA-LNN with its corresponding input. Once all MC samples for a time step are processed, the model checks if more time steps remain in the current epoch. After processing all time steps, the model updates its weights based on the accumulated information. The loss for the epoch is then computed, taking into account the uncertainty estimates derived from the Monte Carlo samples.

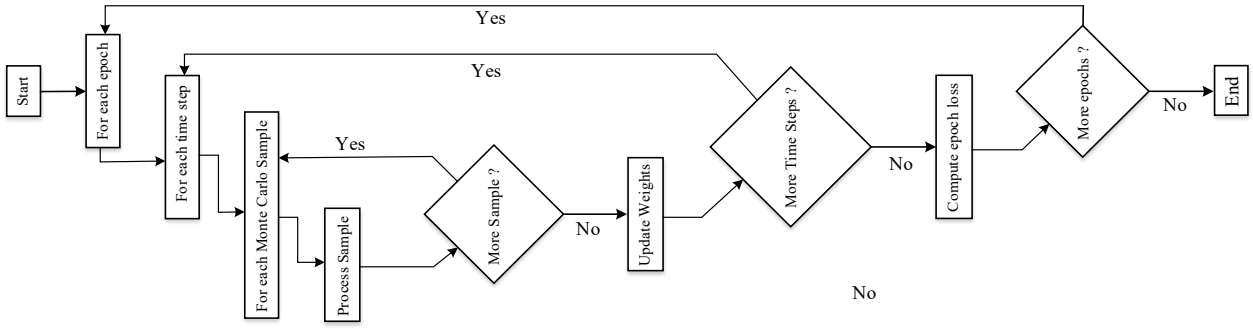


Figure 1. Illustration of the proposed method.

This process continues for the predefined number of epochs, repeating the steps until no more epochs remain. At this point, the training concludes at the "End." This iterative approach helps improve the model's robustness by capturing uncertainty during training through the MC sampling method.

2.2.3 Inference and Uncertainty Quantification

Predictions are generated by sampling multiple times from the model with MC Dropout during inference. For each MC sample, the predictions are calculated as:

$$\hat{y}_{pred}^{(me)} = -W_{out} \cdot s_{t+1}^{(me)} \tag{15}$$

Where $s_{t+1}^{(me)}$ is the updated state for the corresponding MC sample. This ensemble of predictions provides both the final predicted values and a measure of uncertainty since the variability across the MC samples reflects the model's uncertainty in its predictions.

3. Experimental Works and Results

3.1 Simulation data experiments

Initially, we conducted a time series prediction experiment using a simulated simulated complex sinusoidal waveform. The signal was defined over a time vector ranging from 0 to 6π. The simulated time-series signal is given as follows:

$$y(t) = 0.5\sin(t) \times \cos(2t) + 0.3\sin(3t) \times \sin(4t) + 0.2 \cdot \sin\left(5t + \frac{\pi}{4}\right) + \cdots \\ 0.4 \cdot cos(0.5t) \times sin(2.5t) + 0.1 \cdot cos(6t + \frac{\pi}{2}) + n(t) \quad (16)$$

where *n(t)* represents a random variable with an amplitude of 0.1. We employed a single recurrent hidden layer with 100 neurons for both UA-LNN and LNN models. The learning rate was set to 0.0001, and models were trained for 100 epochs. We tested both ReLU and sigmoid activation functions. We set the dropout rate for UA-LNN to 0.1 during training and used 20 Monte Carlo samples during inference. Before the results were presented, an analysis was conducted to evaluate how the UA-LNN performed with different dropout rates. For this purpose, the signal provided in Eq. 16 was used, and the dropout rates were varied in the range of [0.1:0.1:1]. The corresponding R^2 values were calculated and are shown in Figure 2, which presents the R^2 values for each dropout rate.

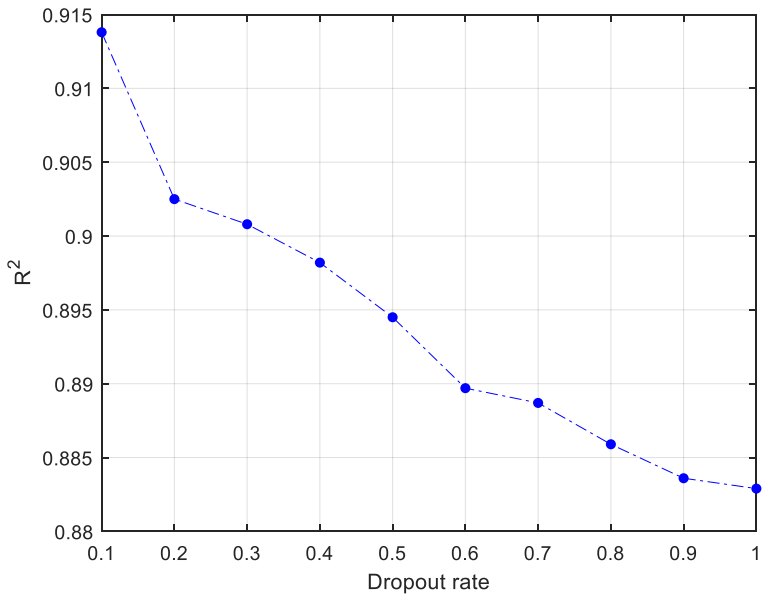


Figure 2. Various dropout rates and corresponding R2 values

As seen in Figure 2, the highest R^2 value 0.9138 is observed at a dropout rate of 0.1, highlighting the model's best performance without dropout. Up to a dropout rate of 0.2, the decline in R^2 values is minimal, suggesting that the model maintains robust predictive capability within this range. However, beyond 0.2, the R^2 values exhibit a steady decrease, implying that excessive dropout introduces too much regularization, thereby reducing the model's ability to effectively capture underlying patterns in the data. Based on these results, a dropout rate between 0 and 0.2 appears optimal, offering a balance

between regularization and performance, whereas higher dropout rates may only be suitable for datasets prone to overfitting, albeit at the cost of reduced R^2 values.

Figure 3 shows the input signal and the UA-LNN prediction using the ReLU activation function. The UA-LNN closely tracks the input signal, accurately modeling its key features, including amplitude and phase variations.

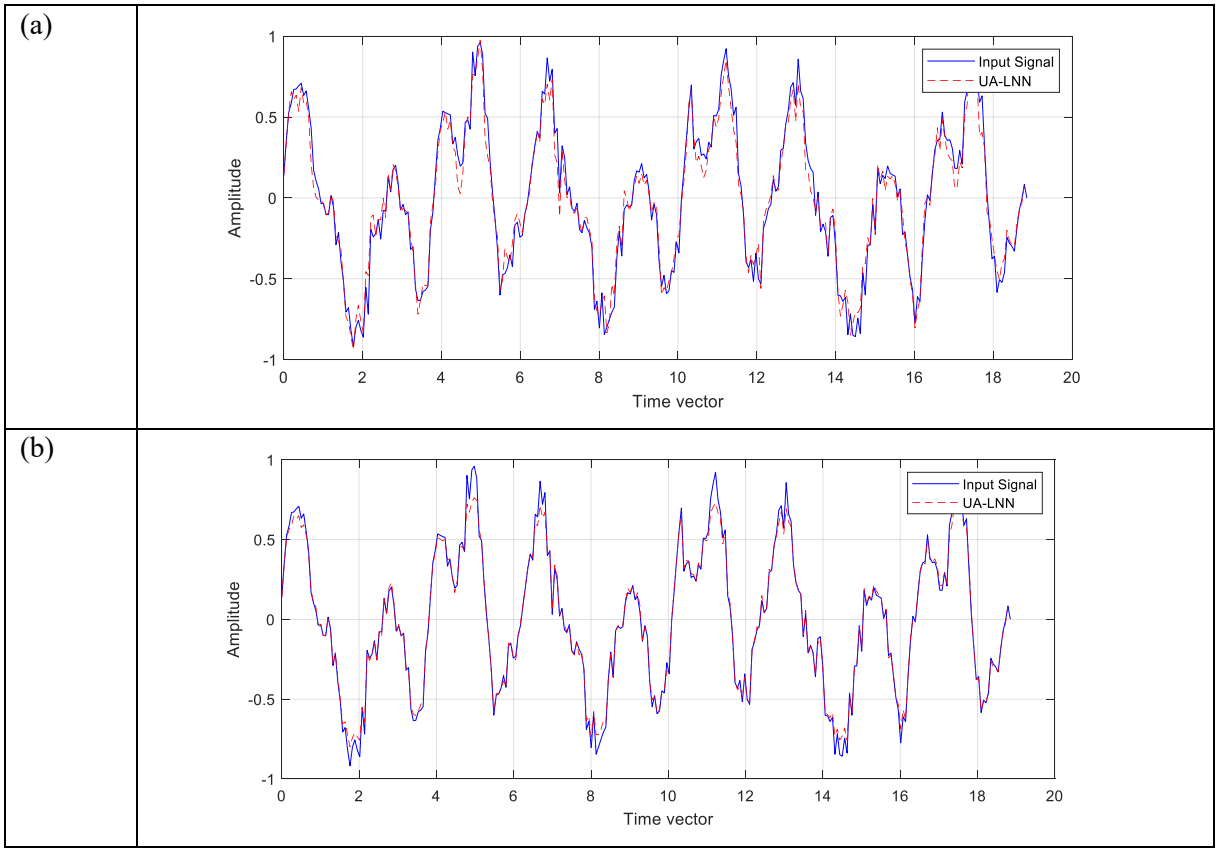
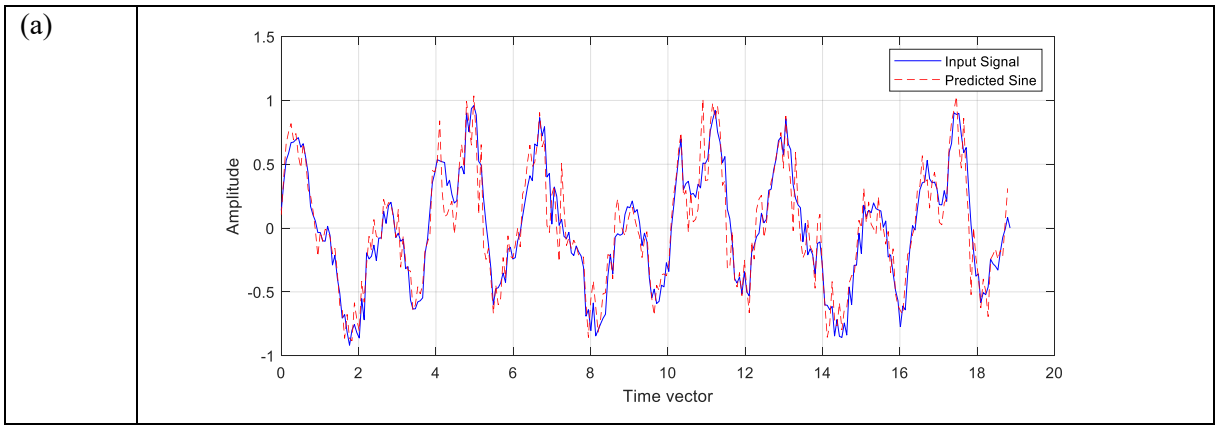


Figure 3. Input and predicted signal using UA-LNN with (a) ReLU activation function, (b) sigmoid activation function.

We conducted the same experiment using an LNN model to compare the performance of UA-LNN with standard LNN. Figure 4 illustrates the LNN prediction using the ReLU and sigmoid activation functions.



(b)

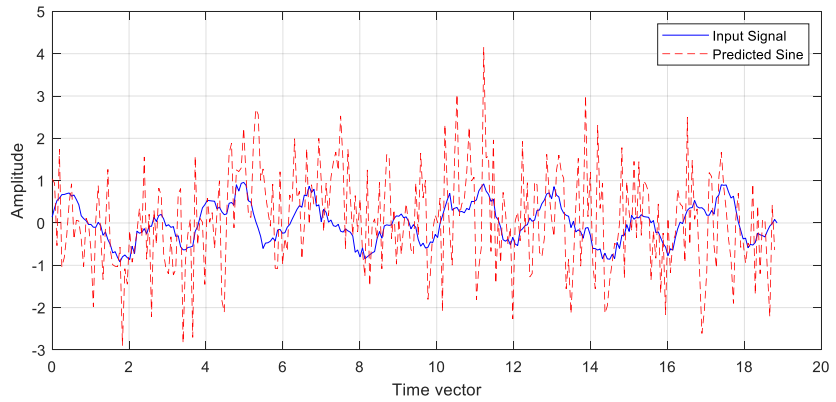


Figure 4. Input and predicted signal using traditional LNN with (a) ReLU activation function, (b) sigmoid activation function.

The models' performances were also evaluated using root mean square error (RMSE) [20], R^2 [21], and Mean Absolute Error (MAE) metrics [22]. Table 1 shows the quantitative evaluation of the proposed and LNN methods against various parameter settings. The results demonstrate that UA-LNN consistently outperforms standard LNN across all metrics, particularly when using the ReLU activation function. For instance, with ReLU activation and 1000 epochs, UA-LNN achieves an R^2 of 0.9138, compared to 0.9031 for LNN, indicating better prediction accuracy. Notably, UA-LNN maintains relatively stable performance across different configurations, while LNN shows more variability, especially with sigmoid activation. The UA-LNN with ReLU activation and 1000 epochs achieve the best overall performance, with the highest R^2 (0.9138) and lowest RMSE (0.1330) and MAE (0.1066) values.

Table 1. A quantitative evaluation of the obtained predictions against the variation of the activation function, number of neurons, and number of epochs.

Method	Activation function	Number of neurons	Number of epochs	R^2	RMSE	MAE
UA-LNN	ReLU	100	100	0.8848	0.1537	0.1207
	ReLU	100	1000	0.9138	0.1330	0.1066
	Sigmoid	100	100	0.8832	0.1548	0.1195
	Sigmoid	100	1000	0.8859	0.1533	0.1188
LNN	ReLU	100	100	0.8211	0.1916	0.1519
	ReLU	100	1000	0.9031	0.1410	0.1086
	Sigmoid	100	100	0.1481	0.4181	0.3201
	Sigmoid	20	1000	0.8263	0.1914	0.1449

To further evaluate the models, we tested them on a multi-input scenario. As shown in Figure 5, we used six simulated input and output: sin (t), cos (t), sin (2 t), cos (2 t), sin (3 t), and cos (3 t). The output was the sum of these signals, with low-amplitude random noise added.

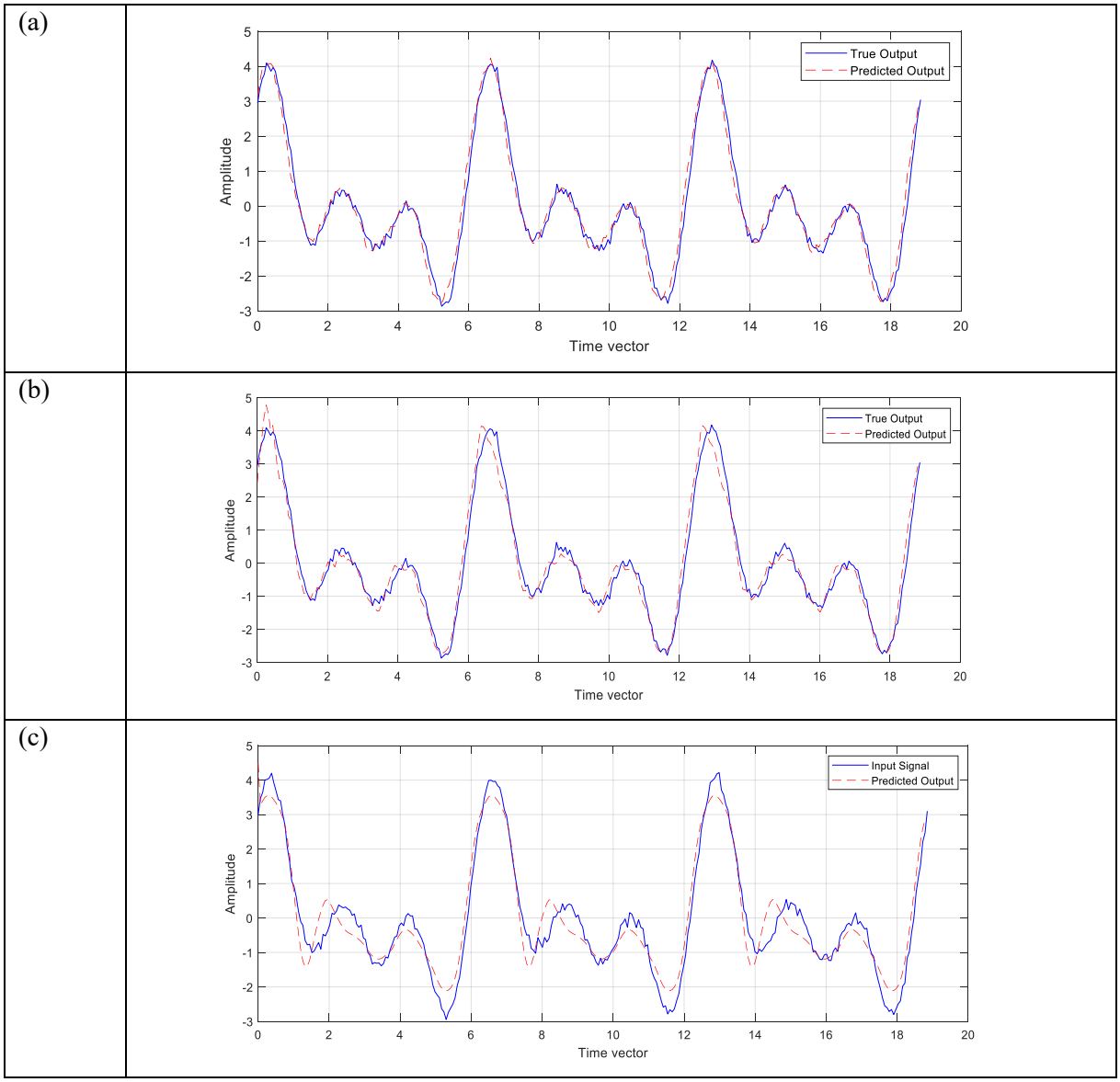


Figure 5. Performance on the multi-input scenario. (a) UA-LNN with ReLU activation function, (b) LNN with ReLU activation function, (c) LNN with sigmoid activation function.

The quantitative evaluation between UA-LNN and LNN is shown in Table 2. In the multi-input scenario, UA-LNN again demonstrates superior performance, achieving the highest R^2 value of 0.9925 and the lowest RMSE of 0.1494. This underscores UA-LNN's capability to handle complex, multi-dimensional input data effectively.

Table 2. Quantitative evaluation of the obtained predictions for the multi-input case.

Method	Activation function	Number of neurons	Number of epochs	R^2	RMSE	MAE
UA-LNN	ReLU	100	50	0.9925	0.1494	0.1191
LNN	ReLU	100	50	0.9825	0.2291	0.1703
LNN	Sigmoid	20	50	0.9391	0.4292	0.3536

3.2 Time series forecasting

We evaluated the proposed UA-LNN method on various time series forecasting datasets, including the Jena Climate Dataset [20], electric power consumption datasets [21], and stock market prices datasets [22-24]. The Jena Climate dataset is a comprehensive collection of meteorological data recorded from a weather station at the Max Planck Institute for Biogeochemistry in Jena, Germany [20]. It includes approximately 420,551 rows and 15 columns, capturing measurements such as temperature, pressure, humidity, wind speed, and others, taken at ten-minute intervals over several years. This dataset is widely used for time series analysis and predictive modeling tasks, particularly for exploring patterns in climate data or building forecasting models. The Electric Power Consumption dataset provides hourly energy consumption data for various regions in the United States. This dataset includes energy usage measured in megawatts over extended periods, offering granular insights into power consumption trends. With over 2.1 million rows, the dataset encompasses a range of features, such as timestamps and regional consumption values, making it suitable for time series forecasting, anomaly detection, and energy trend analysis. The S&P 500 Stock Prices dataset includes historical daily data for companies in the S&P 500 index, such as open, high, low, and close prices, adjusted close, and trading volumes, making it ideal for equity market analysis and trend prediction [22]. The Financial Markets dataset provides comprehensive data on stock prices, indexes, and other financial indicators, enabling global market trend analysis and portfolio optimization [23]. Additionally, the MarketWatch News Sentiment Scores for NASDAQ-100 dataset combines sentiment polarity scores from financial news with market data for NASDAQ-100 companies, facilitating the study of sentiment-driven market dynamics and the development of sentiment-aware trading strategies [24]. Z-score normalization was employed to preprocess the data for time series prediction. This method standardizes the data by subtracting the mean and dividing it by the standard deviation of each feature, ensuring that the transformed data has a mean of zero and a standard deviation of one.

3.2.1 Performance Comparison

Table 3 compares UA-LNN's performance against various state-of-the-art models across different datasets. For each dataset, we list the performance metrics of different methods, with our proposed UA-LNN presented last for easy comparison.

Table 3. Performance comparison of various models, including UA-LNN, on time series forecasting datasets. Best performances are highlighted in bold.

Dataset	Model	R^2	RMSE	MAE	Neurons	Epochs
Climate (Temperature) [20]	CNN-RNN [25]	0.9870	0.1168	0.1260	-	-
	LSTM [25]	0.9640	0.1168	0.0735	-	-
	ARIMA [25]	0.8310	0.2092	0.3038	-	-
	SVM [25]	0.8320	0.1674	0.3297	-	-
	MLP [25]	0.8420	0.2348	0.2758	-	-
	UA-LNN (Proposed)	**0.9955**	**0.0522**	**0.0394**	100	50
Electricity (PJM) [21]	ResNet-Stacked LSTM [26]	-	**0.0380**	**0.0270**	-	-
	UA-LNN (Proposed)	**0.9685**	0.2000	0.1475	100	50
Electricity (AEP) [21]	RCNN-ESN [27]	-	0.2670	0.1978	-	-
	UA-LNN (Proposed)	**0.9658**	**0.1937**	**0.1355**	10	200
S&P500 [22]	ICE2DE-MDL [28]	0.9900	0.1170	0.0660	-	-
	UA-LNN (Proposed)	**0.9934**	**0.0460**	**0.0334**	10	500
NIKKEI [23]	ICE2DE-MDL [28]	0.9190	0.1640	0.1230	-	-
	UA-LNN (Proposed)	**0.9719**	**0.0469**	**0.0354**	10	500
NASDAQ100 [24]	ICE2DE-MDL [28]	0.9660	0.2440	0.1430	-	-
	A-LNN (Proposed)	**0.9907**	**0.0455**	**0.0317**	10	500

The results demonstrate that UA-LNN consistently outperforms most compared models across various datasets. For the Climate (Temperature) dataset, UA-LNN achieves the highest R^2 (0.9955) and lowest RMSE (0.0522) and MAE (0.0394) among all compared models, including CNN-RNN and LSTM. In the case of Electricity (PJM), the ResNet-Stacked LSTM model shows lower error metrics compared to UA-LNN, suggesting superior performance on this specific dataset. However, for Electricity (AEP), UA-LNN performs better than the RCNN-ESN model, with lower RMSE and MAE values. Across all three stock market datasets (S&P500, NIKKEI, NASDAQ100), UA-LNN consistently outperforms the ICE2DE-MDL method, showing higher R^2 values and lower error metrics.

3.2.2 Ablation Study

We conducted an ablation study to evaluate the impact of key hyperparameters on UA-LNN's performance. We varied the number of neurons between 10 and 100 and the number of epochs from 50 to 500. The ReLU activation function was used consistently across all experiments.

The study revealed that the optimal configuration depends on the specific dataset. For instance, the Climate dataset achieved its highest accuracy ($R^2 = 0.9955$) with 100 neurons and 50 epochs, while stock market datasets like NIKKEI and NASDAQ100 performed well with fewer neurons (10) but more epochs (500). The Electricity datasets showed different requirements: PJM needed more neurons (100)

over fewer epochs (50), while AEP achieved similar results with fewer neurons (10) but more epochs (200).

This variation in optimal configurations emphasizes the dataset-specific nature of hyperparameter tuning. The ReLU activation function performed consistently well across all datasets, effectively capturing complex time series patterns. However, future work could explore the impact of other activation functions, such as tanh or sigmoid.

The ablation study highlights the importance of balancing computational efficiency with model accuracy when configuring UA-LNN for different time series forecasting tasks. While UA-LNN generally outperformed traditional models like LSTM and CNN-RNN, some specialized hybrid models (e.g., ResNet-Stacked LSTM for Electricity PJM) showed superior performance on specific datasets, suggesting areas for potential further improvement of the UA-LNN architecture.

3.3 Multi-class classification

3.3.1 Performance Comparison

We evaluated the performance of UA-LNN on six well-known classification datasets: Iris, Wine, Breast Cancer, Ionosphere, Arrhythmia, and Ovarian Cancer [29-35]. These datasets represent a range of classification tasks, from simple multi-class problems to more complex binary classifications in medical domains. The Iris dataset contains 150 samples from three species of iris flowers, namely Setosa, Versicolor, and Virginica. It is often used for multi-class classification purposes [30]. The wine dataset was developed from the chemical analysis of wines derived from three different cultivars. 178 samples are produced as a result of a chemical analysis of wines, with each sample described by 13 continuous attributes alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavonoids, nonflavonoid phenols, proanthocyanidins, color intensity, hue, OD280/OD315 of diluted wines, and Proline, respectively [31]. The breast cancer dataset consists of 569 samples of breast tumor cells [32]. The features describe properties of cell nuclei present in an image of the tissue, such as radius and texture. One wants to classify tumors as benign or malignant; hence, this is a binary classification. The Ionosphere dataset contains 351 radar signals from the ionosphere. Each example has 34 continuous features. It describes signals as "good" or "bad"; thus, this is a binary classification [33]. The arrhythmia dataset contains 452 samples from 279 attributes, originally from features extracted from ECG signals coupled with patient characteristics [34]. This data is used for multi-class classification purposes to identify between normal and several types of arrhythmias. The ovarian cancer dataset is a mass spectrometry dataset containing ovarian cancer and control samples with spectral intensity features [35]. The task is binary classification to identify whether the sample is cancerous or non-cancerous.

We used consistent parameters for both UA-LNN and LNN: 100 neurons, a learning rate of 0.001, and 50 epochs. We employed hold-out validation, using 80% of each dataset for training and 20% for testing. Performance was evaluated using five key metrics: accuracy, precision, recall, specificity, and F1-score

[36]. Table 4 presents a detailed comparison of UA-LNN and LNN across the six datasets. UA-LNN consistently outperformed LNN across most datasets and metrics. The most significant improvements were observed in complex datasets like Arrhythmia and Ovarian Cancer. For the Wine dataset, UA-LNN showed notable improvements across all metrics, with a 2.85% increase in accuracy and a 3.30% increase in F1-score compared to LNN. In the Breast Cancer dataset, UA-LNN demonstrated substantial improvements, particularly in specificity (18.34% increase) and overall accuracy (7.35% increase). The Ionosphere dataset was the only case where LNN slightly outperformed UA-LNN, suggesting that the additional complexity of UA-LNN may not always be beneficial for certain datasets.

Table 4. Summary of performance comparisons of the proposed UA-LNN and LNN on classification problems using various public databases. Best performances are highlighted in bold.

Dataset	Model	Accuracy	Precision	Recall	Specificity	F1-score
Iris [30]	LNN	0.9000	0.8889	0.9167	0.9583	0.8857
	UA-LNN	**0.9333**	**0.9556**	**0.9259**	**0.9608**	**0.9345**
Wine [31]	LNN	0.9429	0.9372	0.9372	0.9629	0.9336
	UA-LNN	**0.9714**	**0.9524**	**0.9848**	**0.9885**	**0.9666**
Breast cancer [32]	LNN	0.8633	0.8791	0.9091	0.7843	0.8939
	UA-LNN	**0.9368**	**0.9810**	**0.9196**	**0.9677**	**0.9493**
Ionosphere [33]	LNN	**0.9571**	**0.9565**	0.9778	**0.9200**	**0.9670**
	UA-LNN	0.9286	0.9000	**1.0000**	0.8000	0.9474
Arrhythmia [34]	NN	0.7558	0.8605	0.7115	0.8235	0.7789
	UA-LNN	**0.8023**	**0.8723**	**0.7885**	**0.8235**	**0.8283**
Ovarian cancer [35]	NN	0.9070	0.9600	0.8889	0.9375	0.9231
	UA-LNN	**0.9767**	**1.0000**	**0.9630**	**1.0000**	**0.9811**

To better understand the data distribution and clustering patterns across our datasets, we employed t-SNE (t-Distributed Stochastic Neighbor Embedding) visualization. Figure 6 presents the t-SNE plots for all six datasets used in our experiments.

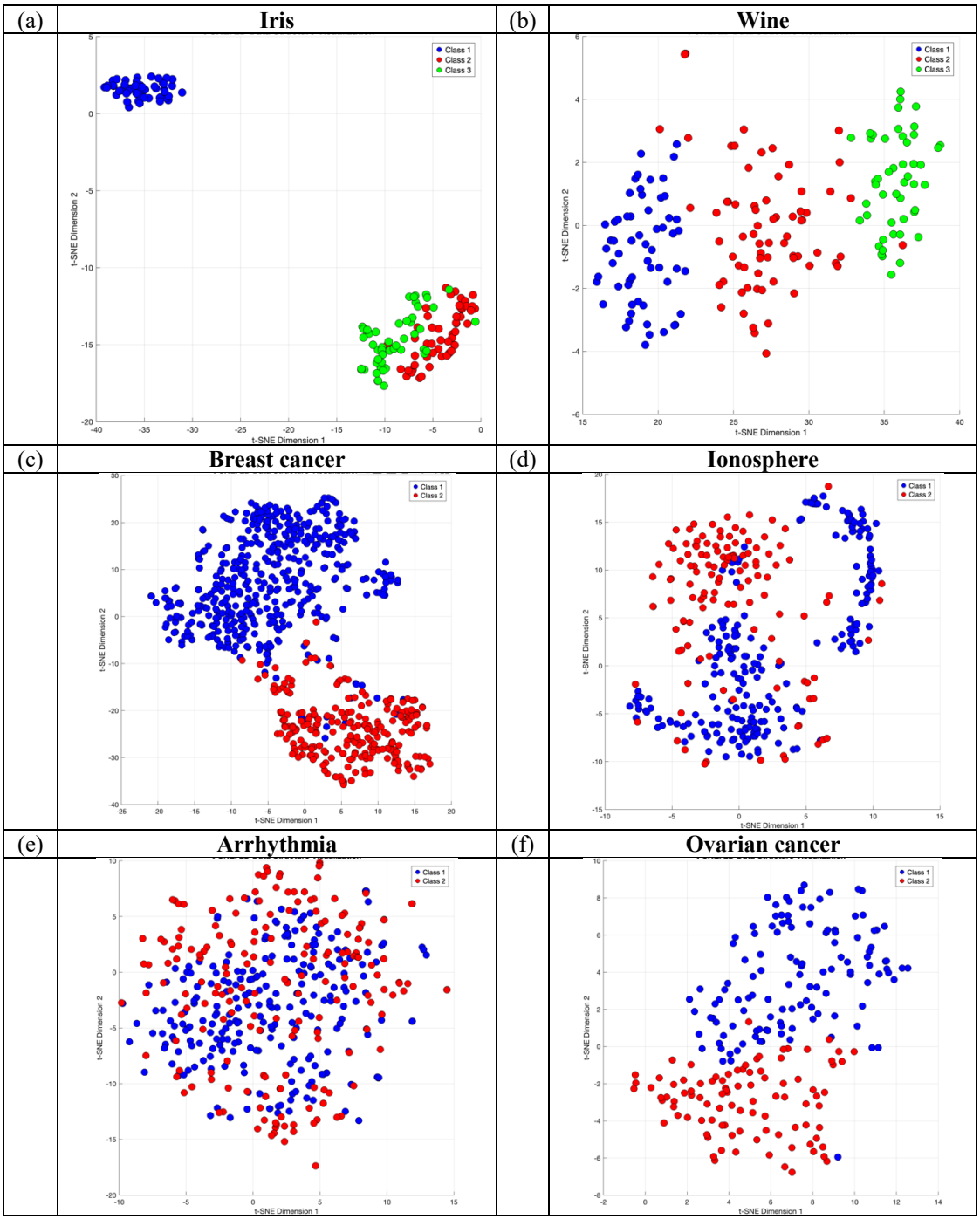


Figure 6. t-SNE visualization of the six classification datasets used in this study: (a) Iris dataset, (b) Wine dataset, (c) Breast cancer, (d) Ionosphere, (e) Arrhythmia dataset, and (f) Ovarian cancer dataset. The different colors represent distinct classes within each dataset, and the spatial proximity indicates similarity between data points in the reduced dimensional space.

The t-SNE visualizations reveal clustering patterns that align with UA-LNN's classification performance across datasets. The Iris and Wine datasets (Fig. 6a-b) show relatively well-defined clusters with minimal overlap, corresponding to high accuracies of 0.9333 and 0.9714, respectively. Among medical datasets, Breast Cancer (Fig. 6c) and Ovarian Cancer (Fig. 6f) display distinct cluster separation, explaining UA-LNN's strong performance (0.9368 and 0.9767 accuracy). The Ionosphere dataset (Fig.

6d) shows more complex boundaries, while Arrhythmia (Fig. 6e) exhibits the most significant class overlap, resulting in more modest performance (0.8023 accuracy). These visualizations align with the quantitative results and demonstrate how UA-LNN's uncertainty-aware mechanism helps navigate varying levels of class separation and overlap. The model performs best with well-separated clusters (Wine, Ovarian Cancer) while maintaining robust performance even with more challenging data distributions (Arrhythmia, Ionosphere).

3.3.2. Ablation Study

To assess the impact of key components in UA-LNN, we conducted an ablation study focusing on neuron count, learning rate, and the number of epochs across all datasets. We tested configurations with 50, 100, and 200 neurons for neuron count. Our findings revealed that performance generally improved with an increase in neurons but with diminishing returns beyond 100. Simpler datasets like Iris and Wine showed optimal performance around 100 neurons, providing a balance between accuracy and computational efficiency. In contrast, the more complex datasets, such as Arrhythmia, exhibited significant gains with increased numbers of neurons up to 200.

We explored the impact of learning rate by testing different values including 0.001, 0.01, and 0.0001. A learning rate of 0.001 performed best across all datasets. Higher learning rates, such as 0.01, frequently resulted in unstable training, particularly evident in the Breast Cancer dataset. Lower learning rates, like 0.0001, resulted in slower convergence without significant benefits to performance, confirming that 0.001 provides an optimal balance for UA-LNN in these classification problems. Lastly, we also investigated the effect of training duration using 20, 50, and 100 epochs. Whereas indeed overall performance improves with more epochs, this gain generally leveled off after 50 epochs in most datasets. The Arrhythmia and Ovarian Cancer datasets were exceptions, showing slight improvements in both accuracy and F1-score when trained for 100 epochs. This suggest that complex data benefit from longer training, while simpler datasets achieve optimal performance with fewer epochs.

3.3.3 Robustness to noise

To evaluate the robustness of UA-LNN under varying noise conditions, we conducted additional experiments by adding noise to the datasets at different levels: 5 dB, 10 dB, 15 dB, 20 dB, and 25 dB Peak Signal-to-Noise Ratio (PSNR). We employed five-fold cross-validation and used average performance metrics for evaluation.

Table 5 presents the performance metrics of UA-LNN across the six classification datasets under varying noise levels. The results demonstrate UA-LNN's robustness to noise across all datasets. As PSNR increases from 5 to 25 dB, all performance metrics (accuracy, precision, recall, specificity, and F1-score) show consistent improvement across all datasets. This trend is particularly evident in the Iris dataset,

where accuracy increases from 0.8400 at 5 dB to 0.9067 at 25 dB, showcasing enhanced model performance under less noisy conditions.

The Wine and Ovarian Cancer datasets exhibit particularly high precision and specificity values, with performance remaining stable even at lower PSNR levels. For instance, the Wine dataset maintains an accuracy above 0.90 even at 5 dB PSNR, demonstrating the model's resilience to noise in certain datasets.

More complex datasets, such as Arrhythmia, show the most significant improvement as noise decreases. In this case, accuracy increases from 0.7442 at 5 dB to 0.8279 at 25 dB, highlighting the model's ability to handle complex data patterns more effectively as noise levels decrease.

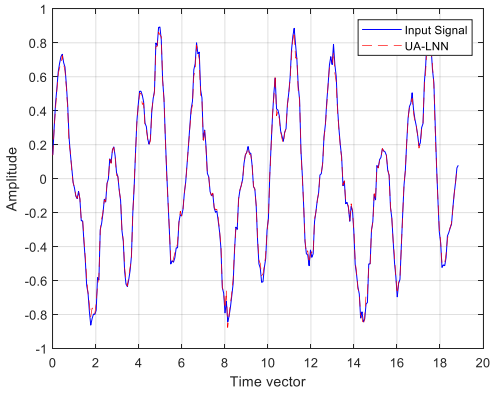
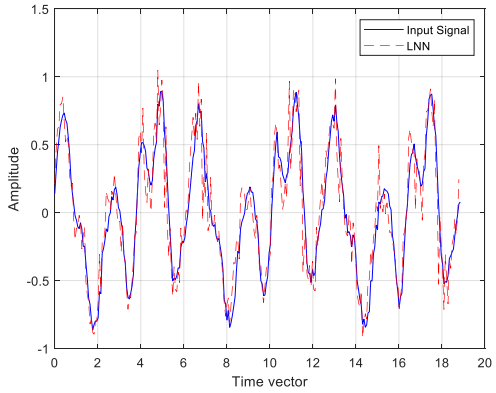
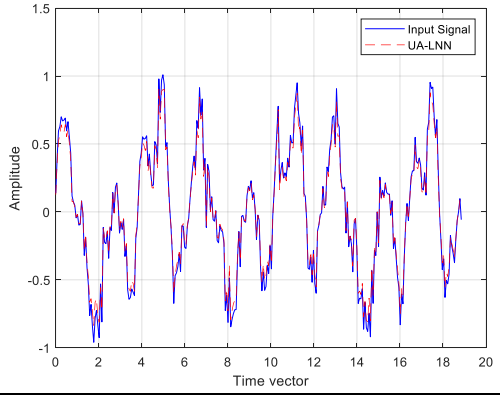
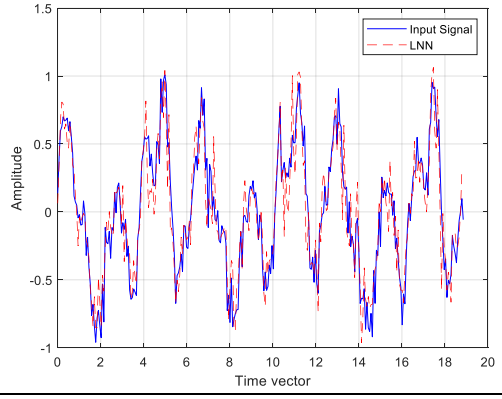
Notably, UA-LNN maintains a balance between precision and recall across all noise levels, as reflected in the consistently high F1-scores. This balanced performance is crucial for real-world applications where both false positives and false negatives can have significant consequences.

Table 5. The performance of UALNN on classification datasets under varying noise levels.

Dataset	PSNR (dB)	Accuracy	Precision	Recall	Specificity	F1-score
Iris [30]	5	0.8400	0.8884	0.8406	0.9175	0.8222
	10	0.8667	0.9216	0.8519	0.9216	0.8603
	15	0.8733	0.8911	0.8723	0.9354	0.8680
	20	0.8800	0.9052	0.8721	0.9386	0.8686
	25	0.9067	0.9291	0.8999	0.9513	0.8975
Wine [31]	5	0.9044	0.8983	0.9134	0.9543	0.8983
	10	0.9665	0.9662	0.9622	0.9831	0.9617
	15	0.9719	0.9640	0.9766	0.9877	0.9681
	20	0.9719	0.9658	0.9773	0.9872	0.9694
	25	0.9775	0.9717	0.9771	0.9895	0.9729
Breast cancer [32]	5	0.8741	0.9328	0.8754	0.8829	0.9015
	10	0.8998	0.9499	0.8949	0.9081	0.9206
	15	0.9056	0.9227	0.9354	0.8486	0.9287
	20	0.9184	0.9455	0.9287	0.9040	0.9367
	25	0.9199	0.9388	0.9415	0.8827	0.9391
Ionosphere [33]	5	0.8404	0.8122	0.9787	0.5917	0.8871
	10	0.8889	0.8557	0.9957	0.6979	0.9200
	15	0.8918	0.8588	0.9953	0.7042	0.9217
	20	0.9089	0.8764	10.000	0.7436	0.9339
	25	0.9116	0.8820	0.9962	0.7478	0.9353
Arrhythmia [34]	5	0.7442	0.8590	0.6324	0.8863	0.7250
	10	0.7791	0.8426	0.7176	0.8504	0.7725
	15	0.8209	0.8873	0.7655	0.8854	0.8203

	20	0.8279	0.8904	0.7784	0.8848	0.8296
	25	0.8279	0.8757	0.7932	0.8642	0.8311
Ovarian cancer [35]	5	0.9536	0.9726	0.9396	0.9682	0.9550
	10	0.9628	0.9714	0.9565	0.9700	0.9636
	15	0.9721	0.9810	0.9652	0.9800	0.9727
	20	0.9721	0.9810	0.9652	0.9800	0.9727
	25	0.9767	0.9818	0.9739	0.9800	0.9778

To visualize the robustness of UA-LNN under noisy conditions, we conducted comparative experiments with varying levels of noise. Figure 7 presents the signal prediction performance of UA-LNN versus standard LNN under different PSNR levels.

PSNR	**UA-LNN**	**LNN**
20 dB		
R^2	0.9404	0.8765
10 dB		
R^2	0.8172	0.7592

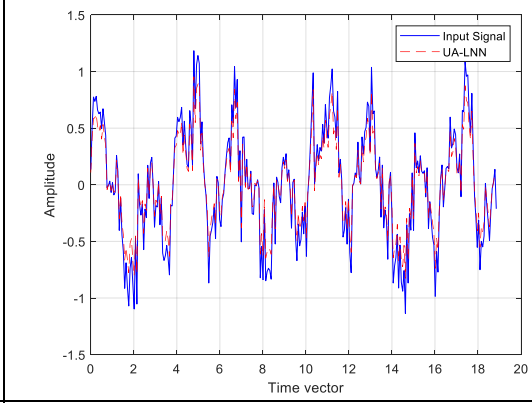
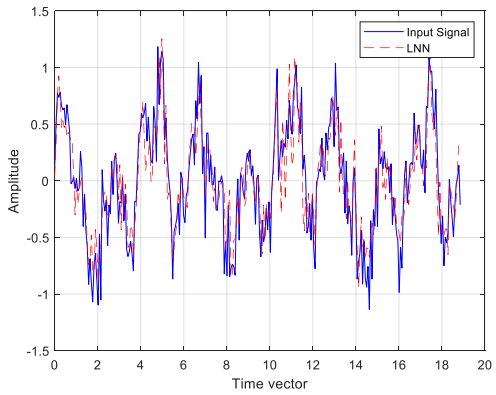
5 dB		
R^2	0.6009	0.5886

Figure 7. Signal Prediction Performance of UA-LNN and LNN under Various PSNR Levels

The UA-LNN demonstrates superior noise resilience across all tested PSNR levels. At 20 dB PSNR, UA-LNN achieves an R^2 value of 0.9404 compared to LNN's 0.8765. This performance advantage is maintained as noise increases, with UA-LNN showing R^2 values of 0.8172 and 0.6009 at 10 dB and 5 dB respectively, while LNN's performance drops more significantly to 0.7592 and 0.5886. These results quantitatively demonstrate UA-LNN's enhanced capability in maintaining prediction accuracy under increasing noise conditions, attributed to its uncertainty-aware mechanism that better adapts to signal variations.

4. Discussions

In this paper, we present a UA-LNN algorithm that overcomes some limitations of the standard LNN model. The key deficiencies of LNNs stem from their deterministic nature, making them unsuitable for applications involving high data uncertainty and variability. A crucial disadvantage of LNN is their inability to quantify prediction uncertainty, an important attribute for many practical applications where decisions depend on a confidence level. Moreover, LNN is prone to overfitting while training with sparse or noisy data due to its highly flexible nature; it will usually memorize the training data instead of learning general patterns.

4.1 Advantages of UA-LNN

The UA-LNN tackles these shortcomings through the implementation of MC dropout. This technique introduces stochasticity in training and inference phases, enabling the model to generate multiple predictions for each input and capture output uncertainty. By applying dropout to the recurrent weight matrix during training, UA-LNN creates diverse model realizations, enhancing generalization and mitigating overfitting. Furthermore, the weight update process, which incorporates an average over several MC samples with gradients, refines the learning process. This approach makes the model robust against input signal variability, resulting in more reliable predictions.

Integrating Monte Carlo dropout allows UA-LNN to quantify prediction uncertainty, offering valuable insights for decision-making processes that rely on confidence levels. By introducing stochasticity during training and inference, UA-LNN improves its ability to generalize from limited or noisy data, thereby reducing overfitting risk.

Experimental results demonstrate that UA-LNN consistently outperforms the standard LNN model across various configurations, yielding superior metrics such as R^2, RMSE, and MAE. UA-LNN maintains relatively consistent performance across different activation functions and epoch settings, with particularly strong results when using the ReLU activation function. The model effectively captures the dynamics of complex time-series signals, showing strong alignment with input signals in various experimental setups.

Additionally, our experiments on noise robustness highlight UA-LNN's effectiveness in handling noisy data. The model maintained high classification accuracy and overall performance despite significant noise levels, demonstrating its potential for real-world applications where data quality may fluctuate.

4.2 Comparative Performance

We conducted extensive experiments to evaluate the proposed UA-LNN method's efficiency in regression and classification tasks. The results consistently show that UA-LNN achieves better accuracy than standard LNN in most cases. We also performed classification experiments using Long Short-Term Memory (LSTM) and Multilayer Perceptron (MLP) models.

Table 6 compares performance metrics for the proposed UA-LNN model, LSTM, and MLP across various classification problems. The results indicate that UA-LNN consistently outperformed both LSTM and MLP models in terms of accuracy, precision, recall, specificity, and F1-score across most datasets. Notably, UA-LNN achieved the highest accuracy of 0.9767 on the Ovarian cancer dataset and maintained strong performance across others, including 0.9333 on the Iris dataset and 0.9714 on the Wine dataset. The LSTM model also showed competitive results, particularly in the Breast Cancer dataset, where it attained an accuracy of 0.9640. However, its overall performance was generally lower than that of UA-LNN. The MLP model exhibited variable performance, achieving the lowest accuracy of 0.7907 on the Arrhythmia dataset. Nevertheless, it performed well on other datasets, such as Wine and Breast Cancer, where it matched UA-LNN's accuracy.

Table 6. Summary of performance comparisons obtained using the proposed UA-LNN, LSTM, and MLP on the classification problem with various public databases. The best performance for each dataset is shown in **bold** font.

Dataset	Model	Accuracy	Precision	Recall	Specificity	F1-score
Iris	LSTM	0.9000	0.9333	0.9091	0.9444	0.9103
	MLP	0.8667	0.9216	0.8333	0.9216	0.8444
	UA-LNN	**0.9333**	**0.9556**	**0.9259**	**0.9608**	**0.9345**
Wine	LSTM	0.9429	0.9524	0.9487	0.9710	0.9466

	MLP	0.9714	**0.9744**	0.9744	0.9855	**0.9733**
	UA-LNN	**0.9714**	0.9524	**0.9848**	**0.9885**	0.9666
Breast cancer	LSTM	0.9640	0.9425	**1.0000**	0.9123	0.9704
	MLP	**0.9640**	0.9659	0.9770	0.9423	**0.9714**
	UA-LNN	0.9368	**0.9810**	0.9196	**0.9677**	0.9493
Ionosphere	LSTM	0.8857	**0.9362**	0.8980	**0.8571**	0.9167
	MLP	0.8714	0.8085	1.0000	0.7188	0.8941
	UA-LNN	**0.9286**	0.9000	**1.0000**	0.8000	**0.9474**
Arrhythmia	LSTM	**0.8488**	0.8033	**0.9800**	0.6667	**0.8829**
	MLP	0.7907	0.7500	0.9130	0.6500	0.8235
	UA-LNN	0.8023	**0.8723**	0.7885	**0.8235**	0.8283
Ovarian cancer	LSTM	0.9535	0.9310	**1.0000**	0.8750	0.9643
	MLP	0.9302	1.0000	0.9032	1.0000	0.9492
	UA-LNN	**0.9767**	**1.0000**	0.9630	**1.0000**	**0.9811**

The superior performance of UA-LNN compared to LSTM and MLP models can be attributed to several key architectural advantages in its design. At its core, UA-LNN's integration of MC dropout enables it to capture prediction uncertainty, leading to more robust generalization compared to the deterministic nature of standard neural networks. The dynamic state adaptation makes UA-LNN particularly effective at handling time-varying patterns and nonlinear relationships in the data. Furthermore, the stochastic nature of MC dropout in UA-LNN provides an implicit regularization effect that helps prevent overfitting, a common challenge in both LSTM and MLP architectures. The model maintains sufficient complexity to capture intricate patterns while using dropout to prevent mere memorization of training data. The improvement over the compared method is most pronounced in scenarios with noisy data (as demonstrated in the Arrhythmia and Ovarian cancer datasets), where UA-LNN's uncertainty-aware learning mechanism allows it to make more reliable predictions by accounting for the inherent variability in the data.

To further validate the performance improvements of UA-LNN over traditional architectures, we conducted paired t-tests comparing accuracy scores across all datasets. Table 7 presents the statistical significance of these comparisons.

Table 7. Statistical significance testing results comparing UA-LNN against LSTM and MLP. P-values are reported for accuracy scores across all datasets.

	UA-LNN vs LSTM	**UA-LNN vs MLP**
Iris	0.0281	0.0001
Wine	0.1739	0.6120
Breast cancer	0.0022	0.0087

Ionosphere	0.00064	0.000072
Arrhythmia	0.0079	0.0071
Ovarian cancer	0.0058	0.0001

The statistical analysis reveals significant performance improvements ($p < 0.05$) of UA-LNN over both LSTM and MLP for most datasets. Particularly strong statistical significance was observed for complex medical datasets such as the Breast Cancer ($p = 0.0022$ vs LSTM, $p = 0.0087$ vs MLP) and Ovarian Cancer datasets ($p = 0.0058$ vs LSTM, $p = 0.0001$ vs MLP). The only exception was the Wine dataset, where the performance differences were not statistically significant ($p > 0.05$). These results provide strong statistical evidence for the superior performance of UA-LNN across diverse classification tasks, particularly in challenging medical applications where reliable prediction is crucial.

4.3 Limitations and Future Works

Despite its advantages, UA-LNN has some limitations. Using Monte Carlo dropout introduces additional computational overhead due to multiple forward passes during training and inference, increasing training time by approximately 20-30%. This overhead becomes more pronounced when dealing with large-scale datasets or real-time applications where rapid inference is crucial.

Furthermore, while UA-LNN excels in many scenarios, its effectiveness may vary depending on the specific characteristics of the data and problem domain, which may limit its generalizability across all tasks. The quality of uncertainty estimation might be affected by factors such as data distribution, noise levels, and the complexity of the underlying patterns.

Future research could focus on optimizing UA-LNN's computational efficiency through techniques such as selective dropout or adaptive sampling strategies. Additionally, exploring alternative uncertainty quantification methods could help address the variability in performance across different domains. The development of task-specific uncertainty calibration methods could also enhance the model's adaptability to diverse applications.

5. Conclusions

This paper proposes an innovative approach called UA-LNN, which embeds MC dropout into the LNN structure for better uncertainty estimation. Our experimental results demonstrate that UA-LNN achieves significantly higher performance compared to benchmark LNN and other classic models including LSTM and MLP. Further, the ability of the UA-LNN to provide estimates of uncertainty along with its predictions is crucial in domains where knowledge of prediction reliability may be as important as the prediction itself, such as medical diagnostics, financial forecasting, and autonomous systems. Our results regarding performance stability across different configurations, especially with ReLU activation, demonstrate the robust capacity of UA-LNNs to capture the complex dynamics within time-series data effectively. UA-LNN showed very strong noise robustness by sustaining high classification accuracy

under high noise levels. This robustness further underlines its potential to be used in real-world scenarios where the quality of data might be compromised.

References

[1] Pascanu, R., Mikleus, P., & Ghaffari, A. (2013). Understanding LSTM Training Strategies. In Proceedings of the 30th International Conference on Machine Learning, 28, 275-283.

[2] Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to Forget: Continual Prediction with LSTM. Neural Computation, 12(10), 2451-2471.

[3] Kirk, S., Houghton, R. J., & McKenzie, P. (2018). Liquid Neural Networks for Sequential Data. Proceedings of the International Conference on Neural Networks, 1-6.

[4] Jaeger, H. (2001). The "Echo State" Approach to Analysing and Training Recurrent Neural Networks. GMD Report 152, German National Research Center for Information Technology.

[5] Hullermeier, E., & Waegeman, W. (2019). Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. Informatics, 6(1), 1-16.

[6] Udumula, V. P. C. R., Ancha, R., Ramayanam, M. K., Teja, P. V. S. M., & Rachpudi, V. (2024, April). An Enhanced Real-Time Object Detection Method using Liquid Neural Network and Echo State Network Architecture. In 2024 International Conference on Inventive Computation Technologies (ICICT) (pp. 669-677). IEEE.

[7] Karn, P. K., Ardekani, I., & Abdulla, W. H. (2024). Generalized Framework for Liquid Neural Network upon Sequential and Non-Sequential Tasks. Mathematics, 12(16), 2525.

[8] Gajjar, P., Saxena, A., Acharya, K., Shah, P., Bhatt, C., & Nguyen, T. T. (2024). Liquidt: stock market analysis using liquid time-constant neural networks. International Journal of Information Technology, 16(2), 909-920.

[9] Abumohsen, M., Owda, A. Y., & Owda, M. (2023). Electrical Load Forecasting Using LSTM, GRU, and RNN Algorithms. *Energies*, *16*(5), 2283. https://doi.org/10.3390/en16052283

[10] Deniz, E., Akpinar, M.H., Sengur, A. (2022). Bidirectional LSTM Based Harmonic Prediction. *VI International European Conference on Interdisciplinary Scientific Research*, August 26-27, 2022, Bucharest, Romania, pp. 70-79.

[11] Kramarczyk, J. M. (2024). Liquid NeurIoTic: Liquid Neural Network-Based Intrusion Detection System for Internet of Things in Healthcare Networks (Doctoral dissertation, The George Washington University).

[12] Hasani, R., Lechner, M., Amini, A., Liebenwein, L., Ray, A., Tschaikowski, M., ... & Rus, D. (2022). Closed-form continuous-time neural networks. Nature Machine Intelligence, 4(11), 992-1003.

[13] Pendyala, V. S., & Patil, M. (2024). Multi-Link Prediction for mmWave Wireless Communication Systems Using Liquid Time-Constant Networks, Long Short-Term Memory, and Interpretation Using Symbolic Regression. Electronics, 13(14), 2736.

[14] Chahine, M., Hasani, R., Kao, P., Ray, A., Shubert, R., Lechner, M., ... & Rus, D. (2023). Robust flight navigation out of distribution with liquid neural networks. Science Robotics, 8(77), eadc8892.

[15] Wahab Sait, A. R., & Dutta, A. K. (2024). Developing a Pain Identification Model Using a Deep Learning Technique. Journal of Disability Research, 3(3), 20240028.

[16] Lucey, P., Cohn, J. F., Prkachin, K. M., Solomon, P. E., Chew, S., & Matthews, I. (2012). Painful monitoring: Automatic pain monitoring using the UNBC-McMaster shoulder pain expression archive database. *Image and Vision Computing*, *30*(3), 197–205. https://doi.org/10.1016/j.imavis.2011.12.003

[17] Chahine, M., Hasani, R., Kao, P., Ray, A., Shubert, R., Lechner, M., Amini, A., & Rus, D. (2023). Robust flight navigation out of distribution with liquid neural networks. *Science Robotics*, *8*(77), eadc8892. https://doi.org/10.1126/scirobotics.adc8892

[18] Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of ICML* (pp. 1050–1059).

[19] Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE). *Geoscientific model development discussions*, *7*(1), 1525-1534. DOI: 10.5194/gmdd-7-1525-2014.

[20] https://www.kaggle.com/datasets/mnassrib/jena-climate

[21] https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption

[22] https://www.kaggle.com/datasets/rprkh15/sp500-stock-prices

[23] https://www.kaggle.com/datasets/regaipkurt/financial-markets

[24] https://www.kaggle.com/datasets/hserdaraltan/marketwatch-news-sentiment-scores-for-nasdaq100

[25] Utku, A., & Can, U. (2023). An efficient hybrid weather prediction model based on deep learning. International Journal of Environmental Science and Technology, 20(10), 11107-11120.

[26] Khan, Z. A., Ullah, A., Haq, I. U., Hamdy, M., Mauro, G. M., Muhammad, K., ... & Baik, S. W. (2022). Efficient short-term electricity load forecasting for effective energy management. Sustainable Energy Technologies and Assessments, 53, 102337.

[27] Alanazi, M. D., Saeed, A., Islam, M., Habib, S., Sherazi, H. I., Khan, S., & Shees, M. M. (2023). Enhancing Short-Term Electrical Load Forecasting for Sustainable Energy Management in Low-Carbon Buildings. Sustainability, 15(24), 16885.

[28] Akşehir, Z. D., & Kılıç, E. (2024). Multi-level perspectives in stock price forecasting: ICE2DE-MDL. PeerJ Computer Science, 10, e2125.

[29] Murphy, P. and A. Asuncion, UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2010, http://archive.ics.uci.edu/ml, accessed at June 2012.

[30] Shukla, A., Agarwal, A., Pant, H., & Mishra, P. (2020). Flower classification using supervised learning. Int. J. Eng. Res, 9(05), 757-762.

[31] https://www.kaggle.com/datasets/aarontanjaya/uci-wine-dataset

[32] https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

[33] https://www.kaggle.com/datasets/creepyghost/uci-ionosphere

[34] https://github.com/shreyas-muralidhara/UCI-Cardiac-Arrhythmia-Classification

[35] Conrads, Thomas P., Vincent A. Fusaro, Sally Ross, Don Johann, Vinodh Rajapakse, Ben A. Hitt, Seth M. Steinberg, et al. "High-Resolution Serum Proteomic Features for Ovarian Cancer Detection." Endocrine-Related Cancer 11 (2004): 163–78.

[36] Yacouby, R., & Axman, D. (2020). Probabilistic Extension of Precision, Recall, and F1 Score for More Thorough Evaluation of Classification Models. *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, 79–91. https://doi.org/10.18653/v1/2020.eval4nlp-1.9