Eng. Sci., 2025, **37**, 1785

Engineered Science
DOI: https://dx.doi.org/10.30919/es1785

Blending Additive and Multiplicative Gates: New FlexGate- Long Short-Term Memory Architecture

Surya Prakash,* Utkal Mehta and Bibhya Sharma

Abstract

In many real-world applications, sequential data exhibit additive and multiplicative dependencies between features and their temporal context. Traditional Long Short-Term Memory (LSTM) networks update their state through additive interactions modulated by multiplicative gates, which can limit flexibility when stronger feature interactions are present. Conversely, architecture such as the Multiplicative-integration Recurrent Neural Network (Mi-RNN) relies purely on multiplicative fusion, often sacrificing stability and interpretability. We introduce FlexGate-LSTM, a recurrent architecture that adaptively blends additive and multiplicative operations inside each gate. A learnable parameter vector in each gate continuously tunes the trade-off between the two interaction modes, allowing the network to select the most suitable integration strategy for the current task or time step. The proposed FlexGate-LSTM is evaluated in five controlled synthetic scenarios: additive, multiplicative, conditional, noisy, and non-stationary, where it matches or exceeds the performance of both vanilla LSTM and Mi-RNN baselines. To demonstrate real-world usefulness, we further test the model on the ETTh1 electricity-transformer temperature dataset. The investigation shows that the FlexGate-LSTM performs competitively in specialized cases and significantly outperforms other architectures when the data exhibits mixed or uncertain dynamics. In addition, the analysis of the learnt parameters provides insight into the internal adaptation strategies of the model, improving the interpretability.

Keywords: Recurrent neural networks (RNN); Long short-term memory (LSTM); Multiplicative integration; Additive-multiplicative fusion; Adaptive gating; Hybrid neural networks.
Received: 03 June 2025; Revised: 24 September 2025; Accepted: 01 October 2025
Article Type: Original research.

1. Introduction

Recurrent neural networks (RNNs) have become foundational tools in modeling sequential data across diverse domains, including natural language processing,[1] time-series forecasting,[2] speech recognition,[3] data analytics,[4] and biological sequence analysis.[5] Among them, the Long Short-Term Memory (LSTM) network remains one of the most widely adopted architectures due to its ability to learn long-term dependencies through its gating mechanisms and additive memory paths.[6] The LSTM design mitigates the vanishing gradient problem by introducing a memory cell that accumulates information additively over time, modulated by multiplicative gates that learn when to store, forget or expose information.

Despite its success, the standard LSTM architecture assumes that the interaction between the current input and the hidden state occurs predominantly through additive transformations (*e.g.*, affine combinations), which are then gated multiplicative. This methodology improves the stability of the training and facilitates a more efficient propagation of gradients between network layers. Nonetheless, its efficacy diminishes in contexts where feature interactions exhibit nonlinear or input-dependent multiplicative dynamics. These limitations are particularly pronounced in domains such as dynamical systems modeling, financial time series forecasting, and tasks involving symbolic reasoning.

Most recently, a multimode feature fusion method[7] was introduced by combining concatenation, addition, and multiplication. This hybrid model has enhanced accuracy in LSTM. Conversely, architectures such as the Multiplicative Integration RNN[8] (Mi-RNN) and its extensions[9] (*e.g.*, MI-LSTM) have explored direct multiplicative fusion of input and hidden state vectors, enabling more expressive interactions, but often at the cost of training stability or interpretability. This balance between expressiveness and stability inspires a new approach to recurrent modeling—one that unifies the advantages of additive and multiplicative architectures while

School of Information Technology, Engineering, Mathematics and Physics, The University of the South Pacific, Suva, Fiji
**E-mail*: *surya.prakash@usp.ac.fj* (S. Prakash),
utkal.mehta@usp.ac.fj (U. Mehta),
bibhya.sharma@usp.ac.fj (B. Sharma)

maintaining simplicity and interpretability. In some cases, the classification was improved using the multiplicative integration of ResNet in LSTM, for medical imaging, for example, skin lesions[10] and prostate cancer.[11] Even multiplicative LSTM[12] (mLSTM) can be a promising tool for an accurate flood risk map. In mobile communication,[13] an mLSTM-based resource estimation strategy was seen as a powerful solution to the handover problem.

Although prior research has advanced both the expressiveness and stability of recurrent networks through various forms of gating and integration, there remains a need for models that can adaptively blend additive and multiplicative interactions based on the demands of the data. Our proposed FlexGate-LSTM architecture addresses this gap by introducing a learnable parameter within each gate, enabling dynamic control over the integration mode at a fine-grained level. To the best of our knowledge, this is the first approach to offer such flexibility and interpretability within the gating structure of an LSTM, validated across a range of synthetic benchmarks designed to probe both pure and mixed dynamics.

The design of recurrent neural networks (RNNs)[14] has long focused on overcoming the challenges posed by learning long-term dependencies, particularly the issues of vanishing and exploding gradients. Early RNNs relied on purely additive updates, combining input and recurrent contributions through simple affine transformations. Although computationally efficient, such models struggled to propagate gradients effectively over long sequences. This limitation led to the development of gated architectures such as LSTM[15] and Gated Recurrent Unit (GRU),[16] which introduced multiplicative gates to regulate information flow over time.

1.1 LSTM and GRU architectures

The LSTM architecture introduced three key gates: input, forget, and output, to selectively control when to write, erase, and read from a memory cell. Crucially, it maintains an additive cell state that enables consistent gradient flow, while the gates modulate these updates via multiplicative interactions. This balance between additive memory and multiplicative control has proven effective across a wide range of applications, including machine translation,[17] speech recognition,[18] route prediction in transport,[19-20] and financial forecasting.[21] The GRU simplifies the LSTM by combining the input and forget gates, while still maintaining a similar additive-multiplicative dynamic.

Despite their success, LSTMs and GRUs share a core limitation: the interactions between input and hidden state within the gates are purely *additive before the gating operation*. This restricts the expressiveness of the model in scenarios where input-state interactions are inherently nonlinear or conditional.

1.2 Multiplicative architectures and Mi-RNN

To enhance modeling power, Multiplicative Integration RNNs (Mi-RNNs) were introduced by Wu *et al*.,[22] where the traditional additive interaction $Wx + Uh$ is replaced by an element-wise Hadamard product $Wx \odot Uh$. Here, the variables are defined as follows.

- $x \in R^d$: Input vector at the current timestep.
- $h \in R^h$: Hidden state vector from the previous timestep.
- $W \in R^{h \times d}$: Weight matrix applied to the input vector.
- $U \in R^{h \times h}$: Weight matrix applied to the previous hidden state.
- $\odot$: Element-wise multiplication operation (Hadamard product).

This multiplicative fusion allows the input to directly modulate the hidden state, enabling more expressive, input-dependent transitions. The approach was further extended to gate architectures such as *MI-LSTM* and *MI-GRU*, where each gate computation includes additive and multiplicative components. Wu *et al*. also introduced learnable blending parameters to allow the network to balance these two interaction modes, demonstrating improved stability and performance in character-level language modeling tasks.[22]

However, while MI-LSTM blends additive and multiplicative paths, it typically applies a *fixed blend* to each gate without learning gate-specific or context-specific trade-offs. Furthermore, the interpretability of the learned integration weights was not deeply explored, leaving open questions about how these components influence learning dynamics in varied tasks.

1.3 mLSTM and input-dependent transitions

Another influential variant is the *Multiplicative LSTM* (mLSTM),[9] proposed by Krause *et al*., which factorizes the hidden-to-hidden transition via a multiplicative intermediate state. This allows the transition matrix to vary as a function of input, effectively creating *input-dependent dynamics*. The mLSTM[23] achieved state-of-the-art performance in character-level language modeling and was later scaled up for sentiment analysis. Despite its strength, mLSTM introduces significant parameter overhead and lacks the architectural simplicity of LSTM or MI-LSTM.

In contrast to these expressive models, some researchers have pursued simplification. The Recurrent Additive Network (RAN)[24] removes the nonlinearity and candidate state, relying entirely on gate-modulated additive updates. While remarkably effective on language modeling tasks, RAN assumes that gated summation is sufficient and lacks the flexibility to represent multiplicative effects when needed.

To this end, we propose FlexGate-LSTM, a flexible and

lightweight extension of the standard LSTM structure. In this structure, we introduce a learnable trade-off parameter α within each gating mechanism. This parameter adaptively controls the contribution of additive versus multiplicative interactions in calculating the pre-activation of each gate, allowing the model to learn the most appropriate integration strategy for a given task or temporal context.

Unlike fixed architectural decisions in traditional RNN variants, FlexGate-LSTM enables per-gate adaptability, making it particularly effective in environments characterized by mixed dynamics, where different portions of the sequence may favor different types of interactions. For example, in sensor data, temperature may evolve smoothly over time (favoring additive integration), while vibration signals may require abrupt modulation in response to external stimuli (favoring multiplicative gating).

To evaluate the efficacy and robustness of the proposed architecture, we conduct extensive experiments across five synthetic benchmark scenarios, each designed to highlight different temporal and structural characteristics: additive, multiplicative, conditional, noisy, and nonstationary. We compare FlexGate-LSTM against three established baselines: the standard LSTM, the Mi-RNN, and a basic hybrid variant. Results demonstrate that FlexGate-LSTM performs competitively in specialized cases and significantly outperforms other models in settings where the underlying dynamics are uncertain or mixed. Additionally, we analyze the learned values of α to provide insight into how the model self-tunes its internal behavior—offering a level of interpretability not typically accessible in standard RNN designs.

In summary, our contributions in this paper are given below.

- A novel gated recurrent architecture is proposed with a learnable blending of additive and multiplicative interactions.
- The structure, namely, FlexGate-LSTM, provides an interpretability analysis of the learnt gating behaviour via the trade-off parameter.
- The FlexGate-LSTM offers a flexible and general-purpose extension to LSTM that maintains performance while increasing modeling capacity.
- The FlexGate-LSTM presents a lightweight, interpretable, and versatile framework for resilient sequence modeling, particularly in scenarios where conventional assumptions regarding feature interactions are invalid.

The remainder of this paper is organized as follows. Section 2 discusses the motivation for developing the new architecture, followed by Section 3 which summarizes the related works on recurrent gating mechanisms. The proposed FlexGate-LSTM architecture is discussed in Section 4. Section 5 presents the experimental setup and evaluation. Section 6 analyzes the results and interprets the learned gate parameters. Finally, Section 7 concludes with implications and future directions.

2. FlexGate-LSTM architecture

The FlexGate-LSTM architecture builds upon the standard Long Short-Term Memory (LSTM) network by introducing a learnable blending mechanism that adaptively combines additive and multiplicative interactions within each gate. This section details the modifications to the standard LSTM equations and introduces the role of the trade-off parameter α in achieving flexible, context-sensitive gating.

2.1 Standard LSTM recap

A standard LSTM cell consists of the following equations with reference to Fig. 1.

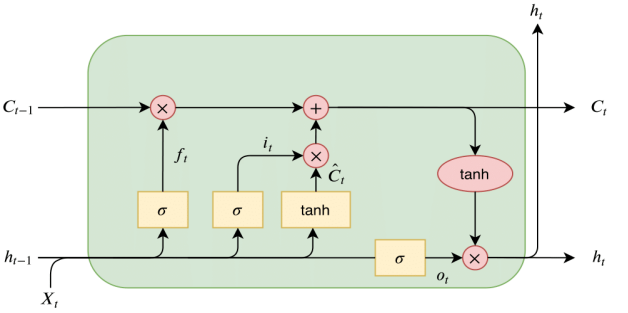

Fig. 1 The structure of the Long Short-Term Memory (LSTM) neural network.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \text{ [input gate]} \quad (1)$$

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \text{ [forget gate]} \quad (2)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \text{ [output gate]} \quad (3)$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \text{ [candidate cell state]} \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$

$$h_t = o_t \odot \text{Tanh} c_t \quad (6)$$

where the initial values are $c_0 = 0$ and $h_0 = 0$, and the operator, $\odot$ denotes the Hadamard product (element-wise product). The subscript, t indexes the time step. Let the superscripts d and h refer to the number of input features and hidden units, respectively. These equations follow the standard LSTM formulation.[6,15]

Table 1 shows all the definitions of the variables with their suitable ranges.

As referred to in Eq. (1-6), the LSTM uses multiplicative gates, and the internal operation of each gate relies on an additive combination of input and recurrent transformations.

2.2 Adaptive blending with FlexGate-LSTM

FlexGate-LSTM modifies the gate computations to introduce a learnable trade-off between additive and multiplicative interactions. For each gate, $g \in \{i, f, o\}$, the pre-activation is computed as:

Table 1: Definitions of variables.

$x_t \in R^d$	Input vector to the LSTM unit
$f_t \in (0,1)^h$	Forget gate's activation vector
$i_t \in (0,1)^h$	Input/update gate's activation vector
$o_t \in (0,1)^h$	Output gate's activation vector
$h_t \in (-1,1)^h$	Hidden state (output) vector
$\tilde{c_t} \in (-1,1)^h$	Cell input activation vector
$c_t \in R^h$	Cell state vector
$W \in R^{h \times d}$	Input weight matrix
$U \in R^{h \times h}$	Recurrent weight matrix
$b \in R^h$	Bias vector
σ_g	Sigmoid activation function
σ_c	Hyperbolic tangent function
σ_h	Hyperbolic tangent function
α	Learnable blending parameter per gate
$\odot$	Hadamard (element-wise) product
d	Dimension of input vector
H	Dimension of hidden state vector.

$$z_t^g = \alpha_g \odot \left(W_g x_t \odot U_g h_{\{t-1\}}\right) + \left(1 - \alpha_g\right) \odot \left(W_g x_t + U_g h_{\{t-1\}}\right) + b_g \quad (7)$$

where:

- $\alpha_g \in [0,1]^{\{d\}}$ is a learnable parameter vector (per gate, per dimension),
- $\odot$ denotes element-wise multiplication,
- $+$ denotes element-wise addition, and
- d is the dimensionality of the hidden state.

Eq. (7) generalizes prior multiplicative-integration approaches[9,22] by introducing a learnable blending parameter per gate

Each gate activation is then:

$$g_t = \sigma\left(z_t^g\right) \quad (8)$$

The candidate cell is:

$$\tilde{c}_t = \tanh(z_t^c) \quad (9)$$

where

$$z_t^c = (\alpha_c \odot (W_c x_t \odot U_c h_{t-1}) + (1 - \alpha_c) \odot (W_c x_t \odot U_c h_{t-1}) + b_c) \quad (10)$$

This formulation extends the candidate state update beyond the standard additive form[15] to incorporate adaptive multiplicative blending.[22]

The cell and hidden state updates follow standard LSTM dynamics:

$$c_t = f_t \odot c_{\{t-1\}} + i_t \odot c_t \quad (11)$$

$$h_t = o_t \odot tanh(c_t) \quad (12)$$

The α parameter introduced in Eq. (7) provides an interpretable mechanism to examine how each gate blends additive and multiplicative contributions. Values of α close to 1 indicate a stronger preference for multiplicative interactions, while values near 0 lean towards purely additive integration. This adaptivity enables FlexGate-LSTM to dynamically tailor its gating behavior according to the specific characteristics of each task or sequence segment, offering significant flexibility in capturing diverse temporal dependencies.

In Eq. (7), let each gate continuously interpolate between an affine (additive map) $W_g x_t + U_g h_{\{t-1\}}$ and a bilinear (multiplicative) map $W_g x_t \odot U_g h_{\{t-1\}}$. Additive paths capture smooth, saturating trends that accumulate over time, while multiplicative paths capture input-dependent scaling, switching or gating effects. Because α is learned *per gate and per hidden unit*, the cell can represent a *mixture of interaction orders* within a single time step—something neither vanilla LSTM (order~1 only) nor Mi-RNN (order~2 only) can do.

Regarding parameter complexity, each gate in FlexGate-LSTM incorporates an additional learnable α vector of dimensionality d, where d is the hidden state size. Since there are three primary gates involved: forget, input, and output. This results in a total of $3d$ extra parameters compared to the standard LSTM architecture. This incremental addition represents a minimal overhead, constituting a negligible fraction of the total parameter count in typical LSTM models, thus preserving computational efficiency while significantly enhancing expressiveness and interpretability. Table 2 provides a summary of expressions [Eq. (7-12)] representing the proposed FlexGate-LSTM.

The advantages of our proposed approach, FlexGate-LSTM can be summarised as follows:

- Flexibility: Allows dynamic adjustment of gating mechanisms during training.
- Interpretability: Learnable α values reveal integration preferences.
- Compatibility: Easily replaces LSTM and Mi-RNN in existing architectures, making it versatile.

Table 2: Summary of the mathematical expressions that constitute FlexGate-LSTM. Interaction indicates whether the computation is purely additive (A), purely multiplicative (M), or a learnable blend of the two (A|M).

Eq. #	Expression	Role in the cell	Interaction
1	$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$	Input gate (baseline LSTM)	A
2	$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$	Forget gate (baseline LSTM)	A
3	$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$	Output gate (baseline LSTM)	A
4	$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$	Candidate cell state	A
5	$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$	Cell-state update (baseline)	A+M
6	$h_t = o_t \odot \tanh(c_t)$	Hidden-state output (baseline)	M
7	$z_t^g = \alpha_g \odot (W_g x_t \odot U_g h_{\{t-1\}}) + (1 - \alpha_g) \odot (W_g x_t + U_g h_{\{t-1\}}) + b_g$	Blended pre-activation for gate $g \in \{i, f, o\}$	A+M (learnable)
8	$g_t = \sigma(z_t^g)$	FlexGate gate activation	A+M
9	$\tilde{c}_t = tanh(z_t^c)$	Blended candidate state	A+M (learnable)
10	$z_t^c = \alpha_c \odot (W_c x_t \odot U_c h_{\{t-1\}}) + (1 - \alpha_c) \odot (W_c x_t + U_c h_{\{t-1\}}) + b_c$	Blended pre-activation for candidate state	A+M (learnable)
11	$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$	Cell-state update (unchanged)	A+M
12	$h_t = o_t \odot \tanh(c_t)$	Hidden-state output (unchanged)	M

3. Experimental evaluation

3.1 Using synthetic data

To systematically evaluate how recurrent architectures adapt to different temporal dynamics, we define five synthetic sequence modeling scenarios. Each scenario highlights a specific type of dependency: additive, multiplicative, conditional, noisy, and nonstationary, that tests the flexibility and expressiveness of gated models like FlexGate-LSTM.

Each scenario is constructed using normalized Gaussian inputs with randomized weights w_t, and targets are generated with known ground-truth logic. These controlled environments enable fair comparisons across models and facilitate interpretability in learned gating behavior.

Each dataset consists of 10,000 sequences of length 30, split 70:15:15 for training, validation, and test.

3.1.1 Additive scenario

Objective: Evaluate a model's ability to capture smooth, purely additive dependencies over time.

$$y = \sum_{t=1}^{T} w_t^T x_t \tag{13}$$

Each input sequence $\{x_1, x_2, \ldots, x_T\}$ consists of vectors $x_t \in R^d$, and the target is the direct weighted sum of the inputs across all time steps, as represented by Eq. (13) This formulation tests whether the model can accumulate information linearly over time—an ability crucial for tasks such as cumulative sensor readings or running totals. As shown in Fig. 2, this scenario favors architectures with strong additive memory (*e.g.*, standard LSTM), providing a baseline before introducing bounded or multiplicative dynamics.

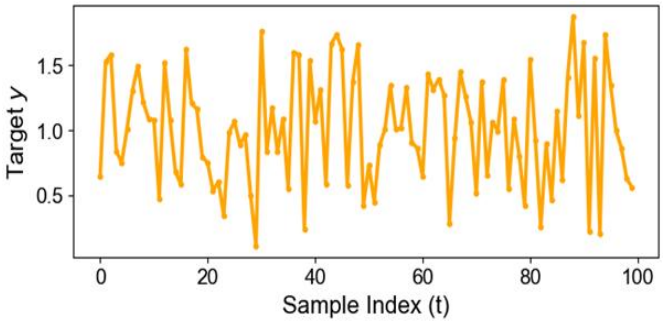

Fig. 2: Additive Scenario. (First 100 samples)

3.1.2 Multiplicative scenario

Objective: Capture multiplicative relationships between inputs across time.

$$y = \prod_{t=1}^{T} (w_t^T x_t) \tag{14}$$

The target is a product of weighted inputs, as represented by Eq. (14), introducing nonlinear dependencies. This favors architectures that natively support multiplicative integration. An example graph is presented in Fig. 3.

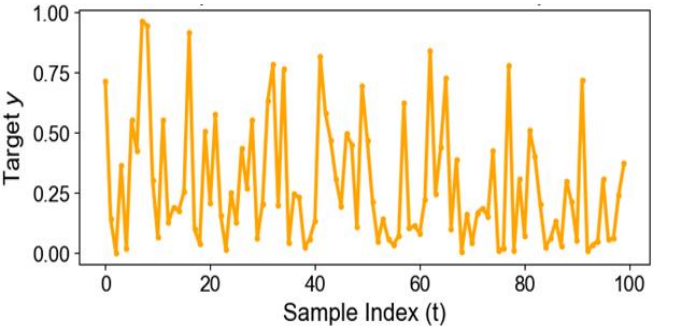

Fig. 3: Multiplicative scenario. (First 100 samples)

3.1.3 Conditional scenario

Objective: Evaluate context-aware decision making based on early input values.

$$y = \begin{cases} \sum_{t-1}^{T} w_t^T x_t: & \text{if} x_1^{(1)} > 0.5 \\ \prod_{t=1}^{T} w_t^T x_t & otherwise \end{cases} \quad (15)$$

The form of integration (additive or multiplicative) is selected based on an early input value as represented by Eq. (15) provided in Fig. 4. This tests whether the model can conditionally route based on context.

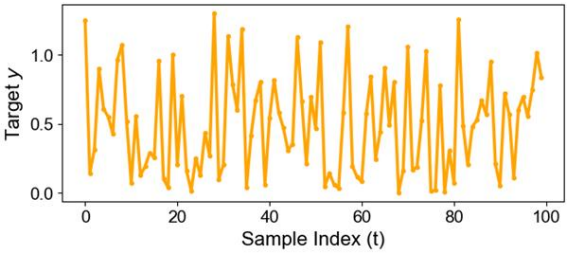

Fig. 4: Conditional Scenario. (First 100 samples)

3.1.4 Noisy scenario

Objective: Evaluate the robustness of the model to irrelevant or corrupted features.Then the output becomes

$$y = \sum_{t \in relevant} w_t^T x_t + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \quad (16)$$

The target output under a noisy scenario, as represented by Eq. (16) is seen from Fig. 5. In this case, only a subset of the sequence contributes to the target, while the rest introduces noise. Models must gate irrelevant signals and focus on key positions.

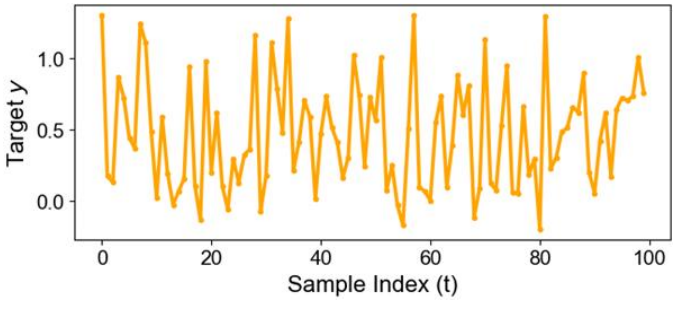

Fig. 5: Noisy Scenario. (First 100 samples)

3.1.5 Nonstationary scenario

Objective: To test adaptability to changes in data-generating distribution over time.

$$y = \begin{cases} \sum_{t-1}^{T} w_t^T x_t: & \text{if } t < \frac{T}{2} \\ \prod_{t=1}^{T} w_t^T x_t: & otherwise \end{cases} \quad (17)$$

The underlying pattern of the target changes midway through the sequence, simulating a regime shift, as represented by Eq. (17) (see Fig. 6). Models must recognize and adapt to the change.

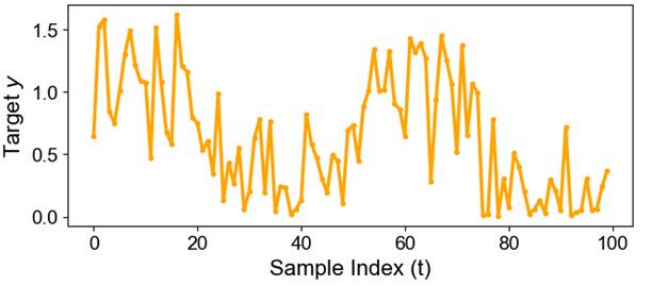

Fig. 6: Nonstationary Scenario. (First 100 samples)

3.2 Real-World dataset evaluation

To assess the practical utility of FlexGate-LSTM beyond synthetic settings, we evaluated its performance on the widely used real-world benchmark dataset: ETTh1 (Electricity Transformer Temperature). This dataset consists of hourly observations of power grid variables and transformer temperatures, making it ideal for multivariate time series forecasting. ETTh1 (Electricity Transformer Temperature - Hourly) is part of the ETDataset[25] series introduced in.

The dataset is published at https://github.com/zhouhaoyi/ETDataset.

3.2.1 Experimental setup

We used the first 7 features (including temperature) and formulated a one-step-ahead forecasting task. Each sequence input consisted of 24 consecutive time steps, and the model predicted the temperature value at the next step.

All models—Standard LSTM, MiRNN, and FlexGate—LSTM were trained using a single-layer architecture with a hidden dimension of 16 and a batch size of 64. Models were optimized using the Adam optimizer with a learning rate of 0.001 over 50 epochs. The dataset was split 70% for training, 15% for validation, and 15% for testing.

3.3 Baselines

We compare FlexGate-LSTM against the following baselines:

- LSTM: Standard long short-term memory network.
- Mi-RNN: Multiplicative integration RNN.

4. Results and discussion

We now present results and discussion for the synthetic and real world data. In addition to reporting numerical performance, we compare our findings with those from related models in the literature, highlighting where FlexGate-LSTM aligns with or extends prior results.

4.1 Using synthetic data

All models were implemented using PyTorch and trained

under consistent hyperparameter settings to ensure a fair comparison. Each model comprises a single hidden layer with 16 units and is trained using the Adam optimizer with a learning rate of 0.005 for 100 epochs. The loss function used is the mean squared error (MSE), which aligns with the regression nature of the synthetic forecasting tasks. Inputs were processed as sequences of length 20 with 3 features per time step, and the models were trained and evaluated across 10 independent runs for each scenario. No dropout or regularization was applied, and batch processing was omitted due to the small synthetic dataset size. The reported results reflect the mean and standard deviation of the MSE across these 10 runs. All experiments were conducted using fixed random seeds to ensure reproducibility.

To evaluate the adaptability and generalization capability of the proposed *FlexGate-LSTM*, we conducted controlled experiments on five synthetic benchmark scenarios: *additive, multiplicative, conditional, noisy*, and *nonstationary*. These scenarios were designed to isolate specific types of temporal dependencies and perturbations commonly observed in real-world sequential data. Each model was trained and tested across 10 independent runs per scenario using identical train-test splits for a fair comparison.

Fig. 7 presents the mean and standard deviation of the mean squared error (MSE) for each model across the five synthetic scenarios. All models were trained with a hidden size of 16, learning rate of 0.005, and sequence length of 20, evaluated over 10 independent runs. The results show that FlexGate-LSTM consistently performs competitively with both baselines, often achieving the lowest MSE. For instance, in the additive scenario, FlexGate-LSTM obtained 0.8438 ± 0.0203, which lies between Standard LSTM (0.826 ± 0.0011) and MiRNN (0.8676 ± 0.0008), showing its ability to maintain stability while matching baseline accuracy. In the conditional task, FlexGate-LSTM achieved the lowest error at 0.0451 ± 0.0001, outperforming both MiRNN (0.0473 ± 0.0026) and LSTM (0.0788 ± 0.0035). Similarly, in the noisy scenario, FlexGate achieved 0.0596 ± 0.0015, slightly better than MiRNN (0.0601 ± 0.0026) and well below LSTM (0.0946 ± 0.0049). In the multiplicative scenario, MiRNN performed best (0.0100 ± 0.0090), as expected, but FlexGate (0.0133 ± 0.0097) remained close while clearly outperforming LSTM (0.0255 ± 0.0034). Finally, in the non-stationary setting, FlexGate-LSTM again obtained the lowest error (0.1788 ± 0.0020), ahead of MiRNN (0.1814 ± 0.0009) and LSTM (0.1881 ± 0.0075). These results demonstrate that FlexGate-LSTM adapts effectively to diverse temporal structures, emulating LSTM in additive regimes and MiRNN in multiplicative ones, while maintaining consistent advantages in conditional, noisy, and non-stationary settings.

The results indicate that FlexGate-LSTM achieves highly competitive performance across all scenarios. It outperforms the Standard LSTM and MiRNN in almost every case, particularly displaying a strong performance in the conditional and nonstationary tasks—highlighting its ability to adapt to complex, dynamic environments. It performs competively for scenarios of additive and multiplicative against LSTM and MiRNN respectively.

While MiRNN performs best in purely multiplicative settings, its performance degrades in scenarios requiring additive accumulation. FlexGate-LSTM, by contrast, flexibly adapts: it behaves like LSTM in tasks driven by additive dependencies and emulates MiRNN under multiplicative dynamics. This adaptability results in a consistently strong and stable performance profile across all scenarios. Overall, while MiRNN may offer marginal gains in strictly multiplicative settings, FlexGate-LSTM provides a more balanced and robust solution across diverse temporal challenges, validating its value as a general-purpose recurrent unit. This observation is consistent with Wu *et al*,[22] who demonstrated that multiplicative integration improves performance in tasks dominated by multiplicative dynamics. However, unlike MiRNN, FlexGate-LSTM retains strong performance in additive and hybrid scenarios, highlighting its adaptability.

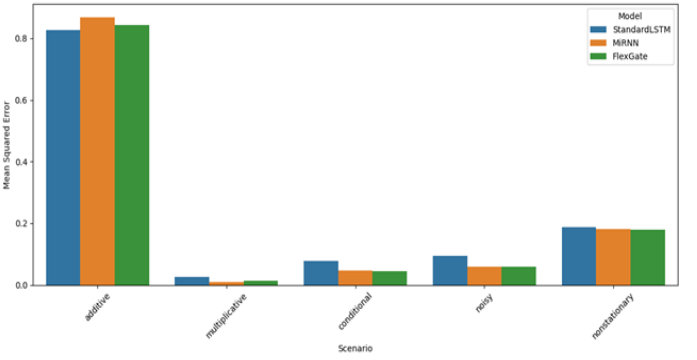

Fig. 7: Mean squared error (MSE) comparison of FlexGate-LSTM, MiRNN, and standard LSTM across five synthetic scenarios, averaged over 10 independent runs. Lower values indicate better performance.

LSTM performance

Although the Long-Short-Term Memory (LSTM) network is a well-established and widely successful model in sequence learning, its comparatively weaker performance in our experiments stems from the deliberate design of the benchmark scenarios. Each synthetic task was crafted to evaluate specific dynamics such as multiplicative feature interactions, conditional switching, or noise robustness—conditions under which purely additive architectures like standard LSTM may be at a structural disadvantage. This does not imply that LSTM is ineffective in general; indeed, it remains a competitive and robust baseline across a wide range of real-world applications. Rather, the scenarios presented here are designed to expose limitations in fixed gating structures and highlight the benefits of adaptively blended gating mechanisms as proposed in FlexGate-LSTM. Future studies may benefit from including additional control tasks or naturalistic data streams where LSTM is expected to perform at parity or better, further enriching the comparative evaluation.

4.2 Real-world evaluation

Table 3 summarizes the Mean Squared Error (MSE) for all models on the ETTh1 dataset. This is also depicted in Fig. 8. FlexGate-LSTM significantly outperforms both Standard LSTM and MiRNN, achieving the lowest test MSE. This indicates its superior ability to adaptively balance additive and multiplicative dynamics in a real-world transformer temperature forecasting task. Fig. 9 depicts the Forecast comparison using different methods for the ETTh1 data by overlaying forecasts from all models ETTh1 validation set for the first 100 steps. It is visually evident that FlexGate–LSTM tracks the amplitude of transformer-temperature swings more faithfully than the baselines. While Mi-RNN has been reported to excel in purely multiplicative tasks,[22] our results show that its performance degrades on ETTh1, where dynamics combine additive and multiplicative effects. In contrast, FlexGate-LSTM balances both, achieving significantly lower MSE than LSTM or Mi-RNN.

Table 3: ETTh1 forecasting performance. (lower MSE is better)

Model	MSE
Standard LSTM	0.8723
MiRNN	0.9404
FlexGate-LSTM	0.5944

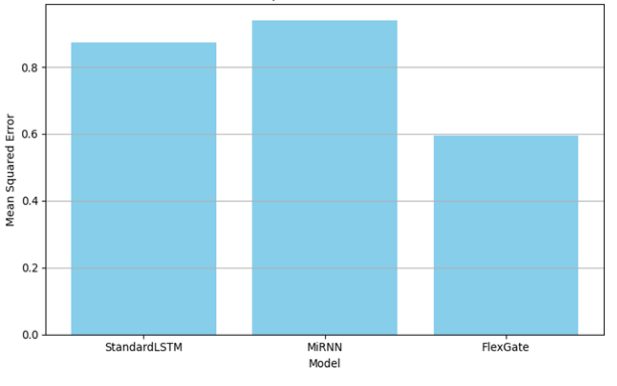

Fig. 8: MSE Comparison on ETTh1 Dataset.

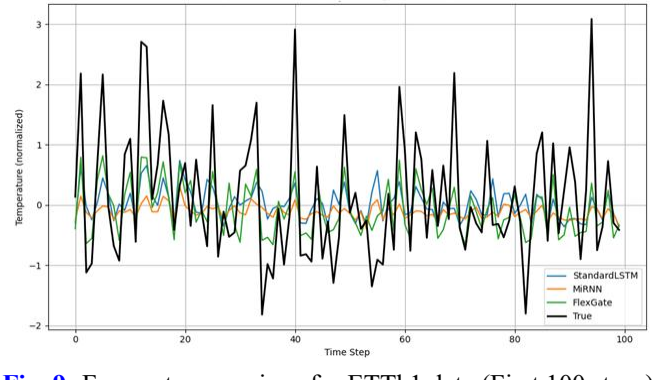

Fig. 9: Forecast comparison for ETTh1 data (First 100 steps).

4.3 Alpha parameter trends

Fig. 10 presents a grouped boxplot of the learned α values across all gates and scenarios in the synthetic benchmark. Each box summarizes the distribution of gate-specific α values across 10 independent training runs, revealing both intra- and inter-scenario behavior.

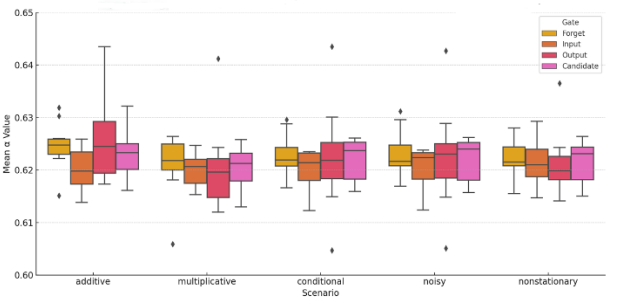

Fig. 10: Grouped boxplot of learned α values for each gate (Forget, Input, Output, Candidate) across five synthetic scenarios. Each box represents the distribution over 10 training runs.

Across all scenarios, α values remain in a stable and interpretable range between 0.61 and 0.64. The Forget and Output gates frequently exhibit slightly higher α values, suggesting a stronger reliance on multiplicative dynamics. In contrast, the Input and Candidate gates maintain slightly lower values. The Candidate gate shows $\alpha \approx 0.619$–0.623 across scenarios (Table 4), indicating a tendency toward more additive integration. This makes sense, since the Candidate state directly contributes to the cell state update, where additive accumulation enhances stability. By contrast, the Forget and Output gates modulate or expose information, roles where multiplicative blending provides finer control. These patterns confirm that FlexGate-LSTM adapts blending strategies not only per scenario but also according to each gate's functional role.

The observed distribution patterns support the hypothesis that the learnable α parameters in FlexGate-LSTM facilitate meaningful and interpretable adaptation across a wide range of sequential dynamics.

Also, for the real world dataset, We examined the learned α values per gate after training. As shown in Table 5, the model placed moderate weight on multiplicative interactions (values around 0.6) for all gates—indicating a balanced preference rather than extremes of either additive or multiplicative gating.

These values reinforce FlexGate-LSTM's ability to discover and adapt to underlying dynamics, making it a suitable model for real-world multivariate time series forecasting.

4.4 Scope and comparison to transformer-based models

Transformer-based models[25-26] have become increasingly popular for sequence modeling tasks, particularly in natural language processing and multivariate time series forecasting. However, our focus in this study is to explore the internal flexibility of recurrent architectures—specifically, how adaptively blending additive and multiplicative gates within

Table 4: Mean and standard deviation of learned α values for each gate across 10 runs and five synthetic scenarios.

Scenario	Forget Gate	Input Gate	Output Gate	Candidate Gate
Additive	0.6270± 0.0050	0.6209 ± 0.0046	0.6266 ± 0.0071	0.6233 ± 0.0058
Multiplicative	0.6227 ± 0.0058	0.6205 ± 0.0032	0.6202 ± 0.0076	0.6193 ± 0.0043
Conditional	0.6237 ± 0.0038	0.6195 ± 0.0046	0.6254 ± 0.0067	0.6225 ± 0.0041
Noisy	0.6234 ± 0.0043	0.6203 ± 0.0047	0.6229 ± 0.0066	0.6226 ± 0.0042
Non-stationary	0.6230 ± 0.0042	0.6214 ± 0.0049	0.6213 ± 0.0059	0.6214 ± 0.0042

Table 5: Average learned α values per gate on ETTh1 dataset.

Gate	Forget	Input	Output	Candidate
α	0.62	0.60	0.63	0.61

LSTM cells affects performance and interpretability.

While Transformer models offer powerful long-range dependency modeling via self-attention, they operate under fundamentally different assumptions: full input visibility ateach timestep, parallel computation, and positional encoding, which diverge from the recurrent, step-wise processing employed by LSTMs and Mi-RNNs. Moreover, Transformer variants often require significantly larger models, longer training times, and tuning of additional hyperparameters such as attention heads, layers, and embedding dimensions. In contrast, FlexGate-LSTM is a lightweight, drop-in replacement for LSTM cells that preserves the sequential nature of computation—making it suitable for settings with limited data, real-time inference needs, or interpretability constraints.

Future work will explore how FlexGateLSTM style adaptive gating might be incorporated into Transformer architectures, but such comparisons fall outside the scope of this study, which aims to isolate the effects of gating strategy within the recurrent paradigm.

4.5 Limitations

While FlexGate-LSTM extends the expressive power of conventional LSTMs with minimal parameter overhead, several caveats remain that readers and practitioners should note.

- Element-wise coupling only. The current formulation blends element-wise additive and multiplicative terms $(W_g x_t \odot U_g h_{t-1})$. It therefore cannot model higher-order cross-feature interactions (*e.g*, quadratic terms $x_i h_j$ for $i \neq j$. Tasks that rely heavily on feature co-dependencies may still benefit more from factorised or tensor-train RNNs.
- Sensitivity to hidden-state dimensionality. FlexGate-LSTM adds $3d$ extra scalars (one α vector per gate), which is negligible at small d but scales linearly. In resource-constrained edge settings with large hidden sizes or many stacked layers, even this modest growth may increase memory footprint and on-device latency.
- Compatibility with attention and convolutional hybrids. FlexGate has not yet been validated inside encoder–decoder or attention-augmented architectures. Interaction effects between adaptive gating and multi-head attention mechanisms are an open research question.

5. Conclusion and future work

In this work, we introduced FlexGate-LSTM, a new recurrent architecture that adaptively blends additive and multiplicative interactions within each gating mechanism through a learnable parameter α. This design enables per-gate flexibility and yields interpretable insights into the integration strategies learned during training. Our extensive evaluation across five synthetic sequence modeling scenarios demonstrates that the FlexGate-LSTM outperforms baseline models, particularly in tasks involving mixed or uncertain temporal dynamics. Furthermore, we validated FlexGate-LSTM on the dataset of the real-world ETTh1 electricity transformer, where it achieved the lowest mean squared error among all models compared, confirming its practical applicability in forecasting tasks with complex dependencies. The ability of FlexGate-LSTM to interpolate between additive and multiplicative operations enables it to mimic the inductive biases of LSTM or MiRNN as needed, making it particularly effective in data regimes with uncertain or changing dynamics. Analysis of α parameters provides additional evidence that the model effectively adapts its gating behaviour in response to the varying requirements of the task.

For future scope, the proposed FlexGate-LSTM can be extended to other real-world time series and language modeling tasks to further validate its generalizability. It is also interesting to explore the gating mechanism to include context-aware or attention-based parameters. The immediate possible extension of this structure is the integration of FlexGate-LSTM within encoder-decoder architectures[27] or memory-augmented models and a recent fractional order activation function in the hidden state. Another promising step is to embed FlexGate–LSTM inside end-to-end decision pipelines for mission-critical mobility tasks, including Monte-Carlo-based emergency logistics[28] and hybrid Lyapunov–high-level path-planning architectures[29-33] and other applications.[34-37] These domains combine gradual trends with interaction-driven transitions, offering an ideal test-bed for the

proposed gating mechanism.

In conclusion, FlexGate-LSTM offers a promising direction for enhancing the adaptability and interpretability of gated recurrent networks, making it a useful component for a wide range of sequential modeling tasks.

Conflict of Interest

There are no conflicts to declare.

Supporting Information

Not applicable.

CRediT Statement

Surya Prakash: Writing – Original draft, Investigation, Formal analysis, Data curation; **Utkal Mehta:** Writing – Review & Editing; **Bibhya Sharma:** Writing – Review & Editing.

References

[1] L. Yao, Y. Guan, An improved LSTM structure for natural language processing, *IEEE International Conference of Safety Produce Informatization (IICSPI)*, Chongqing, China, December 10-12, 2018, 565-569, doi: 10.1109/IICSPI.2018.8690387.

[2] A. Staffini, A CNN–BiLSTM architecture for macroeconomic time series forecasting, *Engineering Proceedings*, 2023, **39**(1), 33, doi: 10.3390/engproc2023039033.

[3] J. Oruh, S. Viriri, A. Adegun, Long short-term memory recurrent neural network for automatic speech recognition, *IEEE Access*, 2022, **10**, 30069-30079, doi: 10.1109/ACCESS.2022.3159339.

[4] P. M. B. Abrasaldo, S. J. Zarrouk, A. W. Kempa-Liehr, A systematic review of data analytics applications in above-ground geothermal energy operations, *Renewable and Sustainable Energy Reviews*, 2024, **189**, 113998, doi: 10.1016/j.rser.2023.113998.

[5] V. I. Jurtz, A. R. Johansen, M. Nielsen, J. J. Almagro Armenteros, H. Nielsen, C. K. Sønderby, O. Winther, S. K. Sønderby, An introduction to deep learning on biological sequence data: examples and solutions, *Bioinformatics*, 2017, **33**(22), 3685-3690, doi: 10.1093/bioinformatics/btx531.

[6] Goodfellow I, Bengio Y, Courville A. Deep Learning, Cambridge, MA, USA, MIT Press, 2016.

[7] K. Peng, W. Zhou, L. Jiang, L. Xiong, W.-J. Yan, Multimodal fusion hybrid neural network approach for multi-class damage classification in high-speed rail track-bridge systems with multi-parameter, *Engineering Structures*, 2025, **328**, 119710, doi: 10.1016/j.engstruct.2025.119710.

[8] Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, *et al*. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, In: Moschitti A, Pang B, Daelemans W, *Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, 1724–34, doi: 10.3115/v1/D14-1179.

[9] Krause B, Lu L, Murray I, Renals S. Multiplicative LSTM for sequence modelling, *In: Workshop track of ICLR*, 2016, **3**, doi: 10.48550/arXiv.1609.07959.

[10] S. Padhy, S. Dash, N. Kumar, S. P. Singh, G. Kumar, P. Moral, Temporal integration of ResNet features with LSTM for enhanced skin lesion classification, *Results in Engineering*, 2025, **25**, 104201, doi: 10.1016/j.rineng.2025.104201.

[11] A. B. Gavade, R. B. Nerli, P. A. Gavade, M. Kumar, U. Mehta, Innovative prostate cancer classification: merging autoencoders, PCA, SHAP, and Machine learning techniques, *Proceedings of the First International Conference on Advanced Robotics, Control, and Artificial Intelligence*, Singapore, 2025, 375-390, doi: 10.1007/978-981-96-5373-7_32.

[12] A. Mohammadifar, H. Gholami, S. Golzari, Novel integrated modelling based on multiplicative long short-term memory (mLSTM) deep learning model and ensemble multi-criteria decision making (MCDM) models for mapping flood risk, *Journal of Environmental Management*, 2023, **345**, 118838, doi: 10.1016/j.jenvman.2023.118838.

[13] A. K. Bandani, S. Riyazuddien, P. Bidare Divakarachari, S. N. Patil, G. Arvind Kumar, Multiplicative long short-term memory-based software-defined networking for handover management in 5G network, *Signal, Image and Video Processing*, 2023, **17**, 2933-2941, doi: 10.1007/s11760-023-02514-1.

[14] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks*, 1994, **5**, 157-166, doi: 10.1109/72.279181.

[15] Hochreiter S, Schmidhuber J. Long Short-Term Memory, *Neural Computation*, 1997, **9**(8), 1735–80, doi: 10.1162/neco.1997.9.8.1735.

[16] G. Shen, Q. Tan, H. Zhang, P. Zeng, J. Xu, Deep learning with gated recurrent unit networks for financial sequence predictions, *Procedia Computer Science*, 2018, **131**, 895-903, doi: 10.1016/j.procs.2018.04.298.

[17] Sutskever I, Vinyals O, Le Q V, Sequence to sequence learning with neural networks, Cambridge, MA, USA, 2014, 3104–12, doi: 10.48550/arXiv.1409.3215.

[18] A. Graves, A.-R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, May 26-31, 2013, 6645-6649, doi: 10.1109/ICASSP.2013.6638947.

[19] S. Prakash, U. Mehta, B. Sharma, Artificial intelligence-driven multimodal route planning: addressing dynamic unavailability and disruptions, *IEEE Access*, 2024, **12**, 172088-172100.

[20] S. Prakash, B. Sharma, A novel approach for route prediction in multimodal transport networks: a Monte Carlo simulation and long short-term-based model, *Engineered Science*, 2024, **29**, doi: 10.30919/es1145.

[21] T. Fischer, C. Krauss, Deep learning with long short-term memory networks for financial market predictions, *European Journal of Operational Research*, 2018, **270**, 654-669, doi: 10.1016/j.ejor.2017.11.054.

[22] Wu Y, Zhang S, Zhang Y, Bengio Y, Salakhutdinov R. On Multiplicative Integration with Recurrent Neural Networks, 2016,

doi: 10.48550/arXiv.1606.06630.
[23] Radford A, Jozefowicz R, Sutskever I. Learning to Generate Reviews and Discovering Sentiment, 2017, doi: 10.48550/arXiv.1704.01444.
[24] J. Lee, K. Cho, T. Hofmann, Fully character-level neural machine translation without explicit segmentation, *Transactions of the Association for Computational Linguistics*, 2017, **5**, 365-378, doi: 10.1162/tacl_a_00067.
[25] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: beyond efficient transformer for long sequence time-series forecasting, *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, **35**(12), 11106-11115, doi: 10.1609/aaai.v35i12.17325.
[26] Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang YX, *et al*. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting, *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, doi: 10.5555/3454287.3454758.
[27] M. Kumar, U. Mehta, G. Cirrincione, Enhancing neural network classification using fractional-order activation functions, *AI Open*, 2024, **5**, 10-22, doi: 10.1016/j.aiopen.2023.12.003.
[28] Prakash S. Emergency relief goods transportation strategies – A Monte Carlo simulation approach, Australasia, 2019.
[29] S. Prakash, B. Sharma, An optimized hybrid approach for path planning: a combination of Lyapunov functions and high-level planning algorithms, *Advances in Data-Driven Computing and Intelligent Systems*, Singapore, 2024, 425-436, doi: 10.1007/978-981-99-9524-0_32.
[30] S. Prakash, P. Sharma, B. Sharma, Achieving safer, more efficient, and smoother pathfinding with LSTM obstacle prediction and Lyapunov control, *Data Science and Applications*, Singapore, 2025, 191-209, doi: 10.1007/978-981-96-2299-3_14.
[31] Prakash Surya, Sharma B. Pathfinding in Autonomous Robotics: Balancing Risk, Energy, and Environmental Impact in Diverse Path Networks. In: Sharma Harish and Chakravorty A and HS and KR, *Artificial Intelligence: Theory and Applications*, Singapore, 2025, 699–714, doi: 10.1007/978-981-96-1918-4_50.
[32] S. Prakash, B. Sharma, Navigating dynamic environments with LSTM, dijkstra, and Lyapunov: a unified approach for autonomous pathfinding and control, *Business Intelligence and Data Analytics*, Singapore, 2025, 477-491, doi: 10.1007/978-981-97-7717-4_34.
[33] S. Prakash, A. P. Sami, B. Sharma, Optimized pathfinding for autonomous robots: enhancing operational range and safety in varied terrains, *Artificial Intelligence: Theory and Applications*, Singapore, 2025, 17-31, doi: 10.1007/978-981-96-1687-9_2.
[34] Prakash S. Alternative approach to estimating crash costs for cost-benefit analysis using Monte Carlo simulation, In: ATRF Australasian Transport Research Forum, 2018.
[35] Prakash S, Jokhan A. An optimal cane delivery scheduling using the Monte Carlo method, In: ATRF 2016 - Australasian Transport Research Forum, 2016.
[36] Prakash S, Jokhan A. Monte Carlo for selecting risk response strategies, In: ATRF 2017 - Australasian Transport Research Forum, 2017.
[37] Prakash S, Mitchell, D. Estimating freight movements using Dijkstra's algorithm. In: ATRF 2018 - Australasian Transport Research Forum, 2018.