IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 June 2024, accepted 9 August 2024, date of publication 16 August 2024, date of current version 28 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3445329

RESEARCH ARTICLE

Chrono Initialized LSTM Networks With Layer Normalization

ANTONIO TOLIC[1], BILJANA MILEVA BOSHKOSKA[1,2], AND SANDRO SKANSI[3]
[1]Faculty of Information Studies, 8000 Novo Mesto, Slovenia
[2]Jožef Stefan Institute, 1000 Ljubljana, Slovenia
[3]Faculty of Croatian Studies, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: Antonio Tolic (antonio.tolic@student.fis.unm.si)

ABSTRACT Recurrent Neural Networks (RNNs), including the distinguished Long Short-Term Memory Networks (LSTMs), have been shown to be effective in a wide range of sequential data problems. However, their ability to model very long-term dependencies still poses challenges, indicating that this remains an active area of research. We propose a new methodology for gradient propagation in LSTM networks that addresses the limitations of traditional approaches. This methodology employs Chrono Initialization (CI) and Layer Normalization (LN) techniques for LSTM networks. CI ensures that the gradients are neither too small nor too large, preventing gradient vanishing or exploding, while LN enhances the stability of hidden state dynamics by aligning the data distribution, mitigating gradient-related issues, and facilitating faster learning. The proposed approach consistently outperformed baseline models in our comparative analyses, achieving average accuracy improvements across classification tasks, with improvements of up to 5% and notable performance increases exceeding 30% in certain scenarios. Significant reductions in Mean Squared Error (MSE) were persistently achieved across regression tasks, averaging 35% lower to several orders of magnitude lower MSE, especially when the baseline models underperformed. Moreover, the method significantly improved performance in sequence generation tasks, consistently yielding much lower negative log-likelihoods relative to the reference counterparts, often by several orders of magnitude. Faster convergence to the optimal minimum was also repeatedly confirmed, even when final results were not significantly divergent. Overall, this new methodology improves the performance of LSTM networks and has been evaluated across various sequential learning tasks. A formal analysis provides a deeper understanding of the mechanisms behind these improvements.

INDEX TERMS Chrono initialization, gradient propagation, layer normalization, long short-term memory networks.

I. INTRODUCTION

Recurrent Neural Networks (RNNs) extend feed-forward neural networks by incorporating an internal memory mechanism, making them particularly effective for sequential problems. At each timestep t, the simplest RNN architecture updates its hidden state $\boldsymbol{h}_t$ using the input $\boldsymbol{x}_t$ according to the rule

$$\boldsymbol{h}_t = \phi(\boldsymbol{W}_{\mathrm{h}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\mathrm{h}}), \tag{1}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the model parameters with their respective layer denotations, and ϕ is the nonlinear activation function.

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

Efficient gradient propagation is essential for training RNNs but remains a challenge due to issues such as vanishing or exploding gradients, which can impede the model's ability to capture long-term dependencies. To address these issues, the Long Short-Term Memory (LSTM) architecture was introduced. The LSTM, a specialized RNN variant, includes memory cells and gating mechanisms that allow selective retention and updating of information over time. This architecture, proposed by Sepp Hochreiter and Jürgen Schmidhuber [1], mitigates the vanishing gradient problem and improves the handling of long-term dependencies. However, it is not immune to the challenges of vanishing and additionally, exploding gradients, which can hinder its performance. Over the years, numerous studies have explored

various components and extensions of the LSTM, as seen in works such as [2], [3], [4], [5], and [6]. This paper revisits the widely used LSTM variant [7], which includes the forget gate, a component whose critical importance was only recognized after many years [2], [8]. We aim to demonstrate the mathematical and empirical improvements in LSTM networks using two techniques: Chrono Initialization (CI) and Layer Normalization (LN). These methods are integrated into the LSTM architecture to enhance performance and efficacy. By incorporating both the CI and LN techniques, a combination not previously explored, we aim to further improve the LSTM architecture. CI [9] positively impacts memory retention and gradient propagation in LSTM networks by alleviating the vanishing gradient problem, enabling better capture of long-term dependencies. CI has already been applied within the LSTM, but only on the forget and input gate biases, as discussed in [9], and it was also utilized in the paper [10], where the issue of hindered gradient propagation was initially identified, albeit without presenting a successful solution. LN addresses internal covariate shift by normalizing inputs to each network layer, ensuring stable training, smoother gradient flow, and faster convergence. We formally analyze the functionality of the augmented networks to provide theoretical insights into the improvements achieved by incorporating CI and LN. Additionally, we conducted a series of experiments on various benchmark datasets, demonstrating the significant advantages of using CI and LN in the LSTM. Our results show consistent improvements in evaluation metrics and convergence speed, validating the efficacy of these techniques. Overall, this study contributes to the advancement of LSTM networks by providing mathematical and empirical evidence of the benefits of CI and LN. These methods can be seamlessly integrated into LSTM architectures, enhancing their capabilities in modeling sequential data across various domains such as natural language processing, speech recognition, time series analysis, and more.

In this research, we start with a survey of related works, followed by a fundamental analysis of LSTM networks. We then delve into the concept of gradient flow in CI-LSTM networks, examining its crucial role in efficient neural network training and potential improvements. Next, we highlight the benefits of LN, a technique that stabilizes and speeds up deep neural network training by addressing internal covariate shift issues. We then introduce a hybrid model combining CI-LSTM with LN, demonstrating how this approach enhances training dynamics by leveraging CI-LSTM's ability to capture sequential patterns and the gradient propagation improvements from LN. Our experimental section provides practical applications of these concepts with real-world examples and results. Additionally, we discuss the conclusions of an ablation analysis and outline potential directions for future work. Finally, our conclusion summarizes the key insights, underscoring the significance of these enhancements.

II. RELATED WORK

One of the main challenges of RNNs is the issue of vanishing and exploding gradients. This problem has a crucial impact on the ability of RNNs to effectively learn and maintain dependencies from previous time steps during training, as detailed in [14]. An LSTM is a specific type of RNN designed to address these challenges. Research has proposed a variety of simpler yet competitive architectures, such as [10], [15], and [16], along with more intricate designs like [6], [17], [18], designed to optimize sequential data processing. Furthermore, numerous research studies have also directly addressed the issues of the vanishing and exploding gradient problems in existing LSTM models, with the hope of improvement, as documented, for example, in [3], [5], [8], [9], [10], [11]. Our research was influenced by the papers [9], and [10]. Although LSTMs have been extensively studied over the past few decades, and their in-depth analysis and research have led to many significant discoveries in the field of sequential data processing, our mathematical exploration revealed the potential for their advancement by utilizing the initialization method introduced in [9]. However, newly encountered challenges led us into the field of data normalization. Normalization has become crucial for accelerating the training of deep networks and improving training stability. There are several normalization techniques. We found the concept of LN particularly intriguing as a possible solution to tackle the newly encountered challenges. While our paper is mathematically related to [12] and [13], the way we've incorporated LN into the LSTM differs slightly from the approach presented in the referenced paper.

III. LSTM OVERVIEW

When the first LSTM architecture was initially introduced in 1997, it marked a significant breakthrough in the realm of RNNs. As mentioned, traditional RNNs often encountered challenges with the vanishing or exploding gradient problems, impeding their ability to retain and propagate information across lengthy sequences. This limitation posed a considerable obstacle for tasks that required memory retention and a deep contextual understanding. The LSTM architecture [7] incorporates a set of equations that govern the flow of information within the network. Specifically, the forget gate equation employs a sigmoid activation function and plays a crucial role in deciding which information should be forgotten or discarded from the memory cell. This selective forgetting mechanism allows the network to prioritize and remember specific information while filtering out less important details. The forget gate equation can be expressed as (σ represents the sigmoid activation function)

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{\mathrm{f}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\mathrm{f}}). \tag{2}$$

Similarly, the input gate equation, also employing a sigmoid activation function, controls the flow of new input information into the memory cell. This gate selectively determines the relevance and significance of the input,

ensuring that only valuable information is stored while preventing irrelevant or noisy inputs from interfering with the stored memory. The input gate equation can be expressed as

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{\mathrm{i}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\mathrm{i}}). \tag{3}$$

Moreover, the output gate equation, using a sigmoid activation function likewise, controls the flow of information from the memory cell to the subsequent layers or the final output of the network. It determines the amount of information to be revealed based on the current state of the memory cell. The output gate equation can be expressed as

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{\mathrm{o}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\mathrm{o}}). \tag{4}$$

To update the content of the memory cell, the LSTM utilize the memory cell equation, which combines the forget gate, input gate, and a hyperbolic tangent activation function. This equation facilitates the selective update of the memory cell, allowing it to store relevant information while discarding irrelevant details ($\odot$ is the symbol for the element-wise or pointwise product, also known as the Hadamard product), as can be observed from the following expressions

$$\begin{aligned} \widetilde{\boldsymbol{c}_t} &= \tanh(\boldsymbol{W}_{\widetilde{\mathrm{c}}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\widetilde{\mathrm{c}}}), \\ \boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \widetilde{\boldsymbol{c}_t}, \\ \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t). \end{aligned} \tag{5}$$

The linear layer is typically utilized for outcome prediction or insight extraction through a linear transformation applied to the hidden state $\boldsymbol{h}_t$, in the following manner

$$\boldsymbol{y}_t = \boldsymbol{W}_{\mathrm{y}} \cdot \boldsymbol{h}_t + \boldsymbol{b}_{\mathrm{y}}. \tag{6}$$

In the provided equations of the LSTM, $\boldsymbol{W}$ and $\boldsymbol{b}$ are model parameters with their respective layer denotations. These equations define the LSTM architecture, which enables the capture and retention of long-term dependencies, thereby addressing the challenges faced by traditional RNNs. By effectively managing the flow of information, the LSTM preserves crucial contextual information over extended sequences, resulting in more accurate predictions and a deeper understanding of complex patterns.

A. GATE INITIALIZATION

One crucial component of an LSTM cell is the gating mechanism, which allows the network to selectively update and access information over time. Gate initialization is an important aspect of the LSTM training, as it sets the initial values for the gate activations. Proper initialization of these gates is crucial for ensuring effective information flow and preventing vanishing or exploding gradients during training. One common approach to gate initialization is using a technique called Xavier initialization [19]. It sets the initial weights of neural network layers to have a mean close to zero and a variance adjusted based on the number of input and output neurons, helping prevent the vanishing or exploding gradient problem. This technique reduces the likelihood of gradients becoming too small or too large and facilitates the training of deep networks. Regarding the initialization of biases for gates, the common practice is to initialize them with 0 or set them to some small random values. However, in [8], it is suggested to initialize the forget gate bias to a value of 1 or 2 to enable gradient flow. Furthermore, in 2018, a new initialization scheme was proposed for the LSTM, named chrono initialization [9]. CI initializes the forget gate bias $\boldsymbol{b}_{\mathrm{f}}$ by sampling from a log uniform distribution ranging from 1 to $T_{max} - 1$

$$\boldsymbol{b}_{\mathrm{f}} \sim \log(\mathcal{U}[1, T_{max} - 1]), \tag{7}$$

and sets $\boldsymbol{b}_{\mathrm{i}} = -\boldsymbol{b}_{\mathrm{f}}$, where T_{max} represents the expected range of long-term dependencies, essentially the input sequence length. It is important to notice that these initializations represent just the starting values, and as the training progresses, the gate biases will adjust independently. Emphasis should also be placed on the broader context of CI theory, which leads to the visual simplification of certain expressions for clarity, as seen, for instance, in [10]. This approach is applied throughout the paper, leading to some expressions resurfacing in equations to highlight specific auxiliary facts.

IV. GRADIENT FLOW IN CI-LSTM NETWORKS

In this section, we will delve into the analysis of the gradient of the objective function J with respect to some arbitrary memory $\boldsymbol{c}_t$, specifically focusing on the gradient flow defined by the CI-LSTM architecture. The cell state is a crucial component of an LSTM, as it helps the network to retain and accumulate information over multiple time steps, allowing for the modeling of long-term dependencies in sequential data. Understanding the behavior and properties of the objective function J with respect to memory $\boldsymbol{c}_t$ is essential for gaining insights into how information flows and is updated within the LSTM network during the process of backpropagation. Analyzing the gradient can provide valuable information about the impact of different input sequences on the learning process, identify potential issues such as vanishing or exploding gradients, and guide the design of more efficient and effective LSTM models. To find $\frac{\partial J}{\partial \boldsymbol{c}_{t-1}}$, we express it as

$$\frac{\partial J}{\partial \boldsymbol{c}_{t-1}} = \frac{\partial J}{\partial \boldsymbol{h}_{t-1}} \frac{\partial \boldsymbol{h}_{t-1}}{\partial \boldsymbol{c}_{t-1}} + \frac{\partial J}{\partial \boldsymbol{c}_t} \frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{c}_{t-1}}. \tag{8}$$

Now let's explicitly write out $\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{c}_{t-1}}$ derivatives

$$\begin{aligned} \frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{c}_{t-1}} = {} & \tanh'(\boldsymbol{c}_{t-1}) \cdot \boldsymbol{c}_{t-1} \odot \boldsymbol{f}_t \cdot (1 - \boldsymbol{f}_t) \cdot \boldsymbol{W}_{\mathrm{f}} \cdot \boldsymbol{o}_{t-1} \\ & + \boldsymbol{f}_t \\ & + \tanh'(\boldsymbol{c}_{t-1}) \cdot \widetilde{\boldsymbol{c}_t} \odot \boldsymbol{i}_t \cdot (1 - \boldsymbol{i}_t) \cdot \boldsymbol{W}_{\mathrm{i}} \cdot \boldsymbol{o}_{t-1} \\ & + \tanh'(\boldsymbol{c}_{t-1}) \cdot \boldsymbol{i}_t \odot (1 - \widetilde{\boldsymbol{c}_t}^2) \cdot \boldsymbol{W}_{\widetilde{\mathrm{c}}} \cdot \boldsymbol{o}_{t-1}. \end{aligned} \tag{9}$$

If we initialize $\boldsymbol{b}_{\mathrm{f}}$ and $\boldsymbol{b}_{\mathrm{i}}$ using the CI technique and assume that the input and hidden layers are zero-centered over time, with T_{max} being a sufficiently large value representing the expected range of long-term dependencies, the preceding

expression (9), corresponding to this value (the same applies to later expressions analogously), simplifies to the following

$$\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{c}_{t-1}} \approx \boldsymbol{f}_t = \sigma(\log(T_{max} - 1)). \tag{10}$$

It has been indicated that T_{max} should be assumed to be a sufficiently large value. In other words, a higher T_{max} value implies that the sigmoid activation values of the forget gate tend closer to 1. However, this assumption is readily met, as evidenced by the graph in Figure 1, where even shorter sequences yield activations close to 1. Nonetheless, the emphasis should still be on longer sequences, as the general point of long-term dependencies may otherwise be diminished. Consequently, the equation (10) is approximated

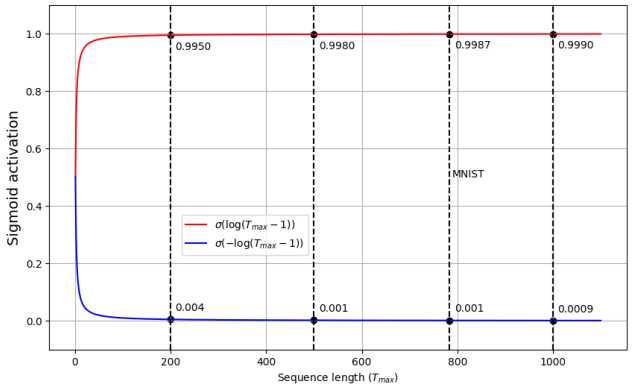


FIGURE 1. The graph depicts sigmoid activation with the bias of the pre-activation vector initialized using the CI technique under the assumptions that the input and hidden layers are zero-centered over time.

to 1. To substantiate the reduction from the equation (9) to the equation (10) in its entirety, it is essential to emphasize that the remaining terms in the equation (9) become negligible as $1 - \boldsymbol{f}_t = 1 - \sigma(\log(T_{max} - 1))$ and $\boldsymbol{i}_t = \sigma(-\log(T_{max} - 1))$ are approximated to 0, due to CI. Furthermore, in the paper [10], an example was demonstrated using the MNIST dataset, in which its images were transformed into sequences of $T_{max} = 784$ (28×28) individual pixels. Each pixel was regarded as a moment in time, and it was observed that $T_{max} - 1 = 783$ implies $\sigma(\log(783)) = 0.99872448979$. As a result, it was shown that the gradients of memory cells $\boldsymbol{c}_t$ are almost unaffected by the sequence length, as the values of the forget gate are close to 1. However, there is a potential disruption in the gradient flow within the calculation details. If we revisit the equation (8) and conduct a more detailed analysis of the expression $\frac{\partial J}{\partial \boldsymbol{c}_{t-1}}$, we will obtain the following

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{c}_{t-1}} = &\frac{\partial J}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} \cdot \boldsymbol{o}_{t-1} \odot \tanh'(\boldsymbol{c}_{t-1}) \\ &+ \frac{\partial J}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{c}_t}\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{h}_{t-1}} \cdot \boldsymbol{o}_{t-1} \odot \tanh'(\boldsymbol{c}_{t-1}) \\ &+ \frac{\partial J}{\partial \boldsymbol{c}_t} \cdot \boldsymbol{f}_t.\end{aligned} \tag{11}$$

The term $\frac{\partial J}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{c}_t}\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{h}_{t-1}}$ can be neglected because of the fact that the expression

$$\begin{aligned}\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{h}_{t-1}} = &\boldsymbol{f}_t \cdot (1 - \boldsymbol{f}_t) \cdot \frac{\partial(\boldsymbol{W}_\mathrm{f} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_\mathrm{f})}{\partial \boldsymbol{h}_{t-1}} \odot \boldsymbol{c}_{t-1} \\ &+ \boldsymbol{i}_t \cdot (1 - \boldsymbol{i}_t) \cdot \frac{\partial(\boldsymbol{W}_\mathrm{i} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_\mathrm{i})}{\partial \boldsymbol{h}_{t-1}} \odot \widetilde{\boldsymbol{c}}_t \\ &+ \boldsymbol{i}_t \odot (1 - \widetilde{\boldsymbol{c}}_t^2) \cdot \frac{\partial(\boldsymbol{W}_\mathrm{\tilde{c}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_\mathrm{\tilde{c}})}{\partial \boldsymbol{h}_{t-1}} \\ \approx &\, 0\end{aligned} \tag{12}$$

since both $1 - \boldsymbol{f}_t$ and $\boldsymbol{i}_t$ tend to be values close to 0, due to CI. Therefore, the term (12) nullifies the entire expression in (11) that it multiplies. Consequently, we obtain the next expression

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{c}_{t-1}} \approx &\frac{\partial J}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} \cdot \boldsymbol{o}_{t-1} \odot \tanh'(\boldsymbol{c}_{t-1}) \\ &+ \frac{\partial J}{\partial \boldsymbol{c}_t} \cdot \boldsymbol{f}_t.\end{aligned} \tag{13}$$

Now, if we revisit the assumption that the input and hidden layers are zero-centered over time with CI and a sufficient T_{max} as the expected range of long-term dependencies, we will recall that $\boldsymbol{f}_t \approx 1$. With $\boldsymbol{f}_t \approx 1$, it is evident that the expression

$$\frac{\partial J}{\partial \boldsymbol{h}_t}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} \cdot \boldsymbol{o}_{t-1} \odot \tanh'(\boldsymbol{c}_{t-1}), \tag{14}$$

within the equation (13) could theoretically present an obstacle or a hidden impediment. The question arises whether the expression (14) can be somehow eliminated from (13), for instance, by nullifying it (since the term (14) is likely not equal to 0 and can assume any real value), as this would lead to a Markov-like simplification of the partial derivatives, which has the potential to enhance the training dynamics of LSTM networks. To see this, let's revisit the expression (14) and express the expression $\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}}$.

$$\begin{aligned}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} = &\boldsymbol{o}_t \cdot (1 - \boldsymbol{o}_t) \cdot \frac{\partial(\boldsymbol{W}_\mathrm{o} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_\mathrm{o})}{\partial \boldsymbol{h}_{t-1}} \odot \tanh(\boldsymbol{c}_t) \\ &+ \boldsymbol{o}_t \odot \tanh'(\boldsymbol{c}_t) \cdot \frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{h}_{t-1}}.\end{aligned} \tag{15}$$

Here we can observe the following. The term $\boldsymbol{o}_t$ (the output gate) appears "very frequently" (observing the expressions (14) and (15) jointly, including the time propagation steps), in fact, it multiplies all the other terms. Therefore, if the $\boldsymbol{o}_t$ term were 0, then the expression (14) would be nullified. A theoretical approach to ensure that the output values of the output gate are close to 0 might involve initializing its bias to significantly negative figures. This strategy can be inspired by examining the practice utilized for the input gate and its bias, applying a similar methodology. For instance, if we were to initialize the bias of the output gate with large negative values by leveraging the principles of CI, then for sufficiently long sequences, the value of $\sigma(-\log(T_{max}-1))$ would be close to 0, as shown by the graph in Figure 1, and this is the desired outcome we are striving to

achieve. If we return to the example with the MNIST dataset and a sequence length of $T_{max} - 1 = 783$, taking into account the negativity of the initialization, the obtained result will be $\sigma(-log(783)) = 0.0012755102$. Therefore, with the same assumptions as before, the output gate would be centered around 0. This is crucial because the bias of the output gate is commonly initialized to 0, and assuming that the input and hidden layers are zero-centered over time, the output gate will be centered around 0.5 in this case. Hence, with the output gate centered around 0, the resulting expression will take the following form

$$\begin{aligned}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} = &\underbrace{\sigma(-\log(T_{max}-1))_t}_{\approx 0} \cdot \underbrace{(1-\sigma(-\log(T_{max}-1))_t)}_{\approx 1} \\ &\cdot \frac{\partial(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o)}{\partial \boldsymbol{h}_{t-1}} \odot \tanh(\boldsymbol{c}_t) \\ &+ \underbrace{\sigma(-\log(T_{max}-1))_t}_{\approx 0} \odot \tanh'(\boldsymbol{c}_t) \cdot \underbrace{\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{h}_{t-1}}}_{\approx 0} \\ &\approx 0. \end{aligned} \tag{16}$$

By substituting expression (16) into (13), wherein we also notice that there is one additional and very important term that assists in nullifying the expression, the result is as follows

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{c}_{t-1}} \approx &\frac{\partial J}{\partial \boldsymbol{h}_t} \cdot 0 \cdot \underbrace{\sigma(-\log(T_{max}-1))_{t-1}}_{\approx 0} \odot \tanh'(\boldsymbol{c}_{t-1}) \\ &+ \frac{\partial J}{\partial \boldsymbol{c}_t} \cdot 1. \end{aligned} \tag{17}$$

This leads to the expression (18), which showcases the promised improvement

$$\frac{\partial J}{\partial \boldsymbol{c}_{t-1}} \approx \frac{\partial J}{\partial \boldsymbol{c}_t} \cdot 1. \tag{18}$$

This approach brings another benefit when compared to the classical initialization approach. The idea was to nullify the expression (14) because it can pose an obstacle in gradient propagation. However, as mentioned earlier, this expression can take on any real number, thus potentially causing issues in the context of the exploding gradients problem. The partial derivative shown in (15) is calculated for a single time step, but as the number of steps increases, there will be consecutive multiplication of terms because the hidden state $\boldsymbol{h}$ is also present in the expression being partially differentiated. Therefore, having the value of the output gate close to 0 will more effectively neutralize large values during multiplication (small values multiply large ones) compared to the classical initialization, as can be observed in (16).

Upon exploring the theoretical potential to arrange the concept for unhindered gradient propagation, it becomes evident that the concept itself is insufficient. The construction implementing unhindered gradient propagation imposes limitations on hidden training dynamics. To clarify, updating the hidden state can generate data with small values, which is potentially expected, but also with a very low variance. This phenomenon is further detailed later in the ablation study. When the data exhibit such low variance, this may suggest that they are almost constant, with minimal fluctuations. LSTM networks rely on the ability to detect patterns and features across a sequence of data. If all the data points are nearly the same, an LSTM may struggle to differentiate relevant information and learn features that can generalize across different situations. However, when addressing the newly emerged challenge, certain elegant techniques, such as LN, are emerging as a viable option. Applying LN to data with values close to zero and low variance assists in aligning the data distribution, thereby improving the network's capacity to adapt and learn pertinent features. More precisely, LN centers the data, bringing the mean value closer to zero. Additionally, it normalizes the data. This process maintains the relative order and distance between data points while also altering the variance, potentially ensuring that the variances of different features approach 1. This alignment of value ranges among various features can be advantageous for optimization and aids in stabilizing the learning process, even when the data initially exhibits low mean values and variances.

V. LAYER NORMALIZATION

LN is a technique introduced by [12] to address the issue of internal covariate shift in deep neural networks. It aims to normalize the inputs to each layer, ensuring a more stable distribution and facilitating smoother gradient flow during training, which helps in maintaining stability and improving the performance of the neural network.

Consider a mini-batch of m training examples and a layer with n neurons. Let $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ represent the input to the layer. LN can be mathematically defined as follows

$$\mu_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}, \quad i = 1, 2, \ldots, m, \tag{19}$$

where μ_i is the mean of the inputs across the neurons for the i-th training example. The variance σ_i^2 is calculated as

$$\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(x_{ij} - \mu_i)^2, \quad i = 1, 2, \ldots, m, \tag{20}$$

quantifying the spread of the inputs across the neurons for the i-th training example. Next, the inputs are normalized using the mean and variance

$$\hat{x}_{ij} = \gamma_j \cdot \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta_j, \tag{21}$$

where ϵ is a small constant added for numerical stability. The LN process also involves learning two additional parameters: a scale parameter $\gamma \in \mathbb{R}^n$ and a shift parameter $\beta \in \mathbb{R}^n$. These parameters allow the normalized inputs to be scaled and shifted, providing the network with the flexibility to adapt to different data distributions. LN has been shown [12] to improve the training of deep neural networks by reducing the

dependence of the network on the scale of the inputs. It helps alleviate the internal covariate shift problem and provides a more consistent and stable learning process, ensuring that a neural network can train more effectively and produce reliable results.

VI. CI-LSTM WITH LAYER NORMALIZATION

LN is a technique that is also commonly employed in LSTM networks to enhance their performance and stability. In [13] the authors explored different approaches to implementing LN in the LSTM and performed a comparative analysis of their effectiveness. In this paper, the focus is placed on the variant named Global Joined Norm (GJN) [13] framed as follows

$$\mathrm{LN}\begin{pmatrix} \boldsymbol{W}_{\mathrm{f}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \\ \boldsymbol{W}_{\mathrm{i}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \\ \boldsymbol{W}_{\tilde{\mathrm{c}}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \\ \boldsymbol{W}_{\mathrm{o}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \end{pmatrix}. \tag{22}$$

In this particular variant, which will be adapted afterwards, instead of applying LN separately to both the forward and recurrent inputs in the LSTM, it is applied jointly after combining them. The same learned normalization parameters are used for both inputs, resulting in one global normalized distribution. It is worth noting that the bias term is removed, as its impact is nullified by the normalization process. While this form of normalization is of interest to us, we will not rigorously adhere to the previously mentioned theory. In other words, we will restore the bias term but eliminate the shift parameter $\beta \in \mathbb{R}^n$. Having both the shift parameter $\beta \in \mathbb{R}^n$ and bias is redundant, therefore, the shift parameter $\beta \in \mathbb{R}^n$ will be ignored. LN will then be applied jointly to the combined forward and recurrent inputs, after which the bias will be incorporated through addition. Therefore, the bias is not included in LN. The LSTM gate biases are now initialized as

$$\begin{aligned} \boldsymbol{b}_{\mathrm{f}} &\sim \log(\mathcal{U}[1, T_{max} - 1]), \\ \boldsymbol{b}_{\mathrm{i}} &= -\boldsymbol{b}_{\mathrm{f}}, \\ \boldsymbol{b}_{\tilde{\mathrm{c}}} &= 0, \\ \boldsymbol{b}_{\mathrm{o}} &\sim -\log(\mathcal{U}[1, T_{max} - 1]), \end{aligned} \tag{23}$$

and after applying LN in the described manner, the resulting equations are

$$\mathrm{LN}\begin{pmatrix} \boldsymbol{W}_{\mathrm{f}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \\ \boldsymbol{W}_{\mathrm{i}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \\ \boldsymbol{W}_{\tilde{\mathrm{c}}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \\ \boldsymbol{W}_{\mathrm{o}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \end{pmatrix} + \begin{pmatrix} \boldsymbol{b}_{\mathrm{f}} \\ \boldsymbol{b}_{\mathrm{i}} \\ \boldsymbol{b}_{\tilde{\mathrm{c}}} \\ \boldsymbol{b}_{\mathrm{o}} \end{pmatrix}. \tag{24}$$

Upon conducting a detailed examination of the model's operation, especially in the context of LN, applying LN to the output of a time step in the LSTM (which serves as the input to the linear layer) proved to be crucial for stabilization. This adjustment significantly impacts the model's performance, and it aligns with the reasons for applying LN mentioned earlier. In accordance with the highlighted, the outcome is

$$\boldsymbol{y}_t = \boldsymbol{W}_{\mathrm{y}} \cdot \mathrm{LN}(\boldsymbol{h}_t) + \boldsymbol{b}_{\mathrm{y}}. \tag{25}$$

Observe that the hidden state is not layer normalized; it is only layer normalized in the context of the output (the input to the linear layer). Finally, the LSTM architecture that utilizes the techniques of CI and LN is abbreviated as CILN-LSTM.

A. GRADIENT FLOW IN CILN-LSTM NETWORKS

Introducing LN into the LSTM architecture can generally affect gradient propagation, but in our specific context, we can conclude that LN will have no effect. The output values of the gate activations, along with CI, still control the gradient flow and play a pivotal role in gradient propagation management. For instance, the equation (16) will be formulated thusly

$$\begin{aligned} \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} &= \underbrace{\sigma(-\log(T_{max} - 1))_t}_{\approx 0} \cdot \underbrace{(1 - \sigma(-\log(T_{max} - 1))_t)}_{\approx 1} \\ &\quad \cdot \frac{\partial(\mathrm{LN}(\boldsymbol{W}_{\mathrm{o}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t]) + \boldsymbol{b}_{\mathrm{o}})}{\partial \boldsymbol{h}_{t-1}} \odot \tanh(\boldsymbol{c}_t) \\ &\quad + \underbrace{\sigma(-\log(T_{max} - 1))_t}_{\approx 0} \odot \tanh'(\boldsymbol{c}_t) \cdot \underbrace{\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{h}_{t-1}}}_{\approx 0} \\ &\approx 0. \end{aligned} \tag{26}$$

Analogously, this same principle applies to the remaining equations as well.

In theory, without LN, the input values of the activation functions are typically assumed to be zero-centered, as seen, for example, in [10]. This assumption implies that if it doesn't hold, there's a likelihood that these input values could interfere with chrono initialized bias values, thus affecting gradient propagation. With LN, we formalize this assumption.

VII. EMPIRICAL VALIDATION THROUGH EXPERIMENTAL EXPLORATION

In this section, we conducted empirical validation to compare the performance of three different LSTM-based models: LSTM, Chrono Initialized LSTM (CI-LSTM), and Chrono Initialized LSTM with Layer Normalization (CILN-LSTM). In this context, it was logical to also explore the use of LSTM with Layer Normalization but without Chrono Initialization (LN-LSTM). However, most experiments did not conclude successfully, thereby reducing the need for an in-depth analysis of this variant. Our objective was to ascertain and evaluate the convergence efficiency of the models towards the optimal minimum, while also evaluating their ability to efficiently capture long-term dependencies by observing performance metrics. We employed well-established benchmark datasets as the foundation of our study, and strategically introduced a range of modifications to heighten the complexity and rigor of our experimental methodology. All models were trained using the Adam optimizer with a learning rate of 0.001, where the number of passes over the entire training dataset (epochs) was fixed at 100. Additional hyperparameterization is specified within the experiment

itself. In each experiment, a defined set of hyperparameters is applied to all the observed models to ensure fairness and enable meaningful comparisons. To ensure robustness and reliability, each experiment is conducted independently three times, and the average outcomes are calculated. The final model is selected based on the best validation loss, ensuring the most reliable performance on unseen data.

A. PERMUTED MNIST CLASSIFICATION

The MNIST dataset is a widely recognized and frequently used benchmark in the field of machine learning. It plays a crucial role in the development and testing of algorithms designed to recognize handwritten digits. Despite its popularity and utility, we chose not to employ the MNIST dataset for evaluating the performance of the models discussed. Instead, our focus was directed towards a more complex variant known as the Permuted MNIST (pMNIST) dataset [20]. The pMNIST dataset is a transformation of the MNIST dataset, primarily used for evaluating the robustness and generalization capabilities of machine learning models, especially neural networks. The pixels in every image of the MNIST dataset have been permuted in the same random order, which makes the problem of the classification of handwritten digits even harder. The labels are kept. Furthermore, each image in the pMNIST dataset is transformed into a sequence of 784 (28×28) individual pixels that are presented to the network sequentially, one pixel at a time. The dataset is divided into a training set of 60,000 images and a test set of 10,000 images. For the validation set, 10% of the training data was used. The models were tested, considering hidden layer units (L) of 128 and 256, employing a mini-batch size of 200. An L2 regularization factor was set to 1e-4, and the gradient norm was clipped at a value of 5. The performance of these models was assessed primarily using accuracy as a key performance metric.

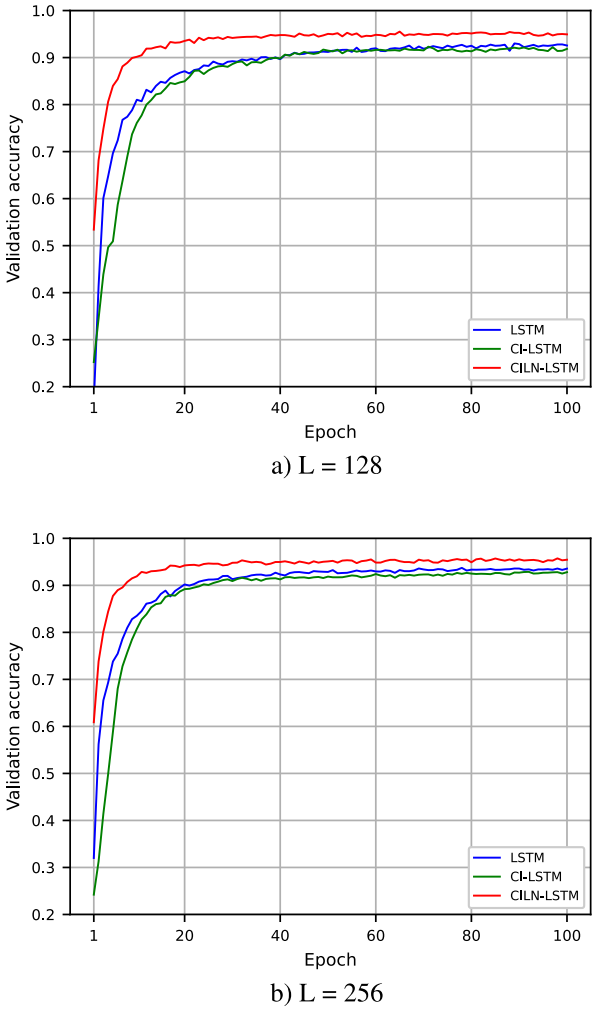


FIGURE 2. The mean accuracies obtained by evaluating the models on the pMNIST validation set across three independent training runs, using hidden layer sizes of 128 and 256 units.

TABLE 1. The mean accuracies obtained by evaluating the models on the pMNIST test set during three independent training runs are listed in the table. The best accuracies from the experiments are presented in bold.

Model	L = 128	L = 256
LSTM	92.27 %	92.94 %
CI-LSTM	92.19 %	92.53 %
CILN-LSTM	**94.68 %**	**95.36 %**

The CILN-LSTM model has exhibited improved performance compared to the other models across multiple aspects, thereby establishing its efficacy in predictive tasks. Notably, when the hidden layer comprised either 128 or 256 units, as shown in Figure 2, the CILN-LSTM model outperformed its counterparts in terms of validation accuracy. This suggests a more robust generalization capability when dealing with unseen data. Additionally, it is important to emphasize that the CILN-LSTM model demonstrated a faster convergence to the optimal minimum on the validation dataset, as evidenced in Figure 2, indicating its efficiency in optimization. This characteristic is particularly valuable in complex scenarios where computational resources and time are limited. The rapid convergence reduces the computational overhead and accelerates the training process, making the model highly suitable for practical applications where efficiency is crucial. Moreover, the generalization potential of the CILN-LSTM model is further supported by its empirical performance on the test set, as detailed in Table 1. The consistent performance across both validation and test datasets highlights the model's ability to maintain high accuracy and reliability when applied to new data.

B. FASHION MNIST

The Fashion MNIST dataset [21], gathered from Zalando's collection of article images, has emerged as a significant benchmark in the realm of machine learning. Serving as an alternative to the traditional MNIST dataset, Fashion MNIST presents numerous advantages and complexities that are essential for contemporary machine learning models. Unlike the original MNIST dataset, which has been criticized for its

simplistic nature in today's context, Fashion MNIST offers a more intricate and diverse set of images, thus providing a tougher challenge for classification tasks. This increased complexity better mirrors real-world image classification and allows for thorough examination of algorithm robustness and generalization. Models trained on Fashion MNIST handle diverse textures, shapes, and subtle details, making it suitable for evaluating generalization from training to unseen data. This diversity helps identify model weaknesses, guiding improvements. Furthermore, this dataset maintains the same image size, data format, and split structure for training, validation, and testing as the original MNIST. However, instead of handwritten digits, it comprises grayscale images with dimensions of 28 × 28 pixels, each depicting various types of clothing items. These images are annotated with labels indicating the correct garment category. Furthermore, each image in the dataset is transformed into a sequence of 784 (28 × 28) individual pixels that are presented to the network sequentially, one pixel at a time. The dataset is divided into a training set of 60,000 images and a test set of 10,000 images. For the validation set, 10% of the training data was used. The models were evaluated, considering hidden layer units (L) of 128 and 256, employing a mini-batch size of 200. An L2 regularization factor was set to 1e-4, and the gradient norm was clipped at a value of 5. The performance of these models was assessed primarily using accuracy as a key performance metric.

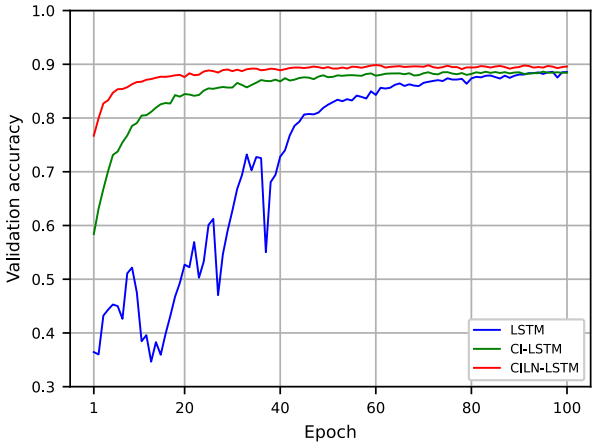
TABLE 2. The mean accuracies obtained by evaluating the models on the Fashion MNIST test set during three independent training runs are listed in the table. The best accuracies from the experiments are presented in bold.

Model	L = 128	L = 256
LSTM	88.71 %	89.83 %
CI-LSTM	88.67 %	88.76 %
CILN-LSTM	**89.58 %**	**90.08 %**

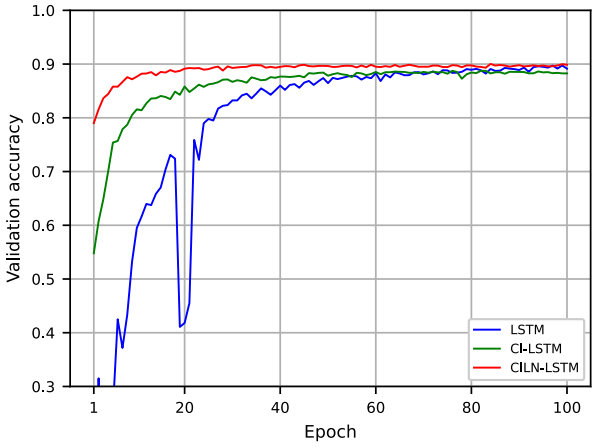
The CILN-LSTM model has shown improved performance compared to the other models in terms of validation accuracy, and it has also achieved faster convergence towards the optimal minimum compared to its counterparts on the validation dataset, as can be seen cumulatively in Figure 3. Furthermore, when examining Figure 3, it is evident that the LSTM model encountered significant challenges during its learning process. It is relevant to highlight that the LSTM model encountered significant challenges during its learning process. Evaluation on the test dataset also favors the CILN-LSTM model, as shown in Table 2. It should be mentioned that in experiments involving 256 units, the LSTM model showed an improvement in its performance when considering differences in results on the test dataset compared to the other models, as can also be observed in Table 2.

C. TIME SERIES ANALYSIS

Time series analysis is a specialized analytical technique employed to investigate and derive insights from a sequence of data points collected over a defined time span [22],



a) L = 128



b) L = 256

FIGURE 3. The mean accuracies obtained by evaluating the models on the Fashion MNIST validation set across three independent training runs, while employing hidden layer units (L) of both 128 and 256.

[23]. In this particular experiment, our objective was to assess the models' capability to efficiently detect and analyze patterns in time series datasets. To do so, we chose two comprehensive time series datasets for evaluation. These datasets could provide a rough estimate of the models' behavior in forecasting trends and patterns in similar time series data. Through this analysis, we aimed to enhance our understanding of how these models perform in real-world scenarios, thereby contributing to more accurate forecasting and decision-making processes. Here are the abbreviations and their respective explanations:

- **HG=F** (Copper Futures): This dataset represents the trading activity of standardized agreements to buy or sell copper at a predetermined price and date. These financial instruments are traded on commodities exchanges and are used by investors to hedge against price fluctuations or to speculate on future price movements of copper. Copper futures contracts are an important

indicator of economic health, as copper is widely used in construction, manufacturing, and electronics.

- **GSPTSE** (S&P/TSX Composite Index): This dataset represents the S&P/TSX Composite Index, a capitalization weighted equity index designed to monitor the performance of the largest companies listed on Canada's primary stock exchange, the Toronto Stock Exchange (TSX). It operates as the Canadian equivalent of the S&P 500 index in the United States, providing a crucial gauge for assessing the Canadian stock market's status and trends.

The datasets used in this study comprised data accumulated over the past four years up to the time of writing this paper, with observations recorded daily. The primary objective was to forecast closing prices by leveraging historical data. To achieve this, sequences of length 30 were established, with the focus on predicting the subsequent element in the sequence. The evaluation of model performance was conducted using Mean Squared Error (MSE) as the primary metric. After the datasets were formed, the training set consisted of 770 samples and the test set included 170 samples. For the validation set, 15% of the training data was utilized. The models were tested with hidden layer units (L) set at 16 and 32, along with a mini-batch size of 32. An L2 regularization factor was set to 1e-4, and the gradient norm was clipped at a value of 5. This approach facilitated a comprehensive evaluation of the overall effectiveness and suitability for forecasting scenarios, thus emphasizing their practical utility.

TABLE 3. The mean MSE results obtained by evaluating the models on the HG=F test set during three independent training runs are listed in the table. The best MSE results from the experiments are presented in bold.

Model	L = 16	L = 32
LSTM	0.00089	0.00076
CI-LSTM	0.00113	0.00104
CILN-LSTM	**0.00054**	**0.00048**

The CILN-LSTM model accomplished a lower MSE and faster convergence to the optimal minimum in contrast to the LSTM and CI-LSTM models on the HG=F validation dataset, as shown in Figure 4 and supported by HG=F test dataset evaluations in Table 3. It reached its target performance more promptly when observing the validation dataset and performed better in a shorter timeframe. This distinction underscores the CILN-LSTM model's efficiency and effectiveness in achieving enhanced results in a relatively shorter timeframe.

Continuing the experimentation with a different dataset, we obtained similar results suggesting that the CILN-LSTM model achieves a lower MSE and faster convergence to the optimal minimum compared to the LSTM and CI-LSTM models on the GSPTSE validation dataset, as shown in Figure 5, whereas the evaluation on the GSPTSE test dataset

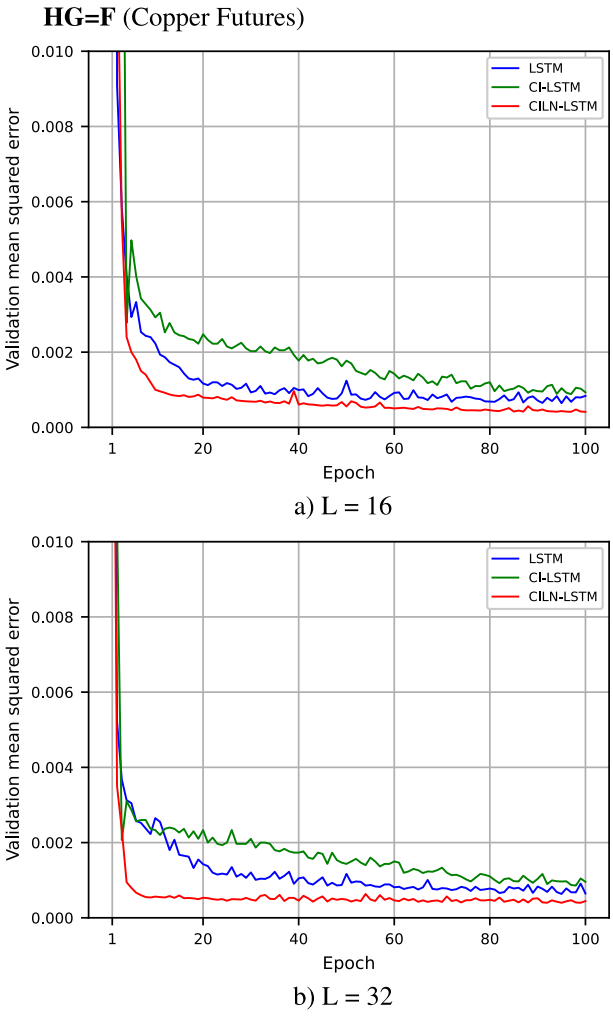


FIGURE 4. The mean MSE results obtained by evaluating the models on the HG=F validation set across three independent training runs, while employing hidden layer units (L) of both 16 and 32.

TABLE 4. The mean MSE results obtained by evaluating the models on the GSPTSE test set during three independent training runs are listed in the table. The best MSE results of the experiments are presented in bold.

Model	L = 16	L = 32
LSTM	0.00094	0.00082
CI-LSTM	0.00116	0.00109
CILN-LSTM	**0.00051**	**0.00045**

shows significant advantages favoring the CILN-LSTM model, as confirmed by the results in Table 4. Overall, the CILN-LSTM model performed better than the other models in both time series experiments.

D. THE ADDING PROBLEM

The adding problem task, introduced as a benchmark in neural network research [1], is a synthetic task specifically crafted to evaluate the capabilities of neural networks in handling complex sequence processing tasks. This challenge requires the model to focus on and accurately process

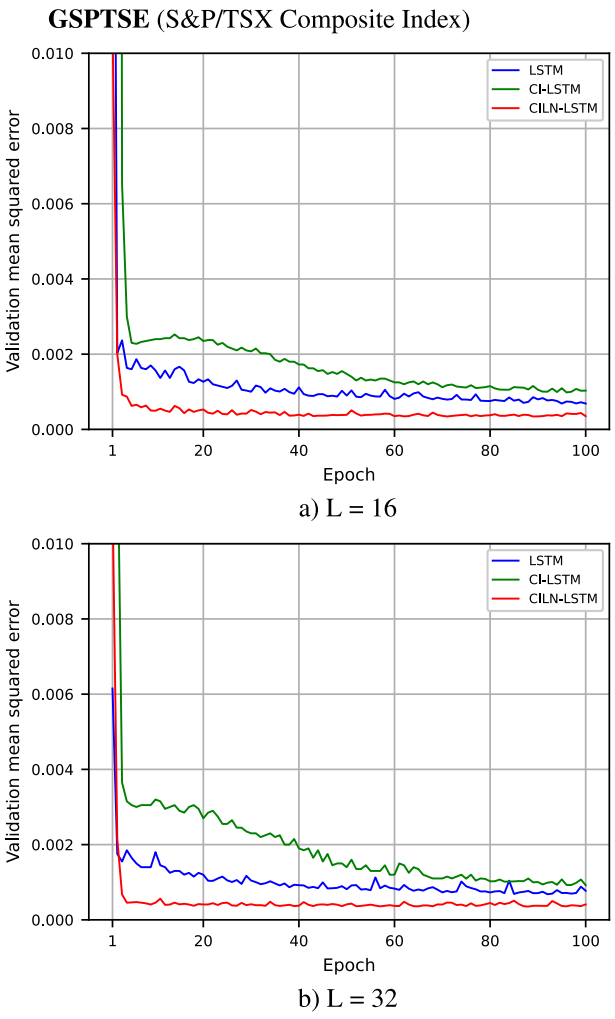


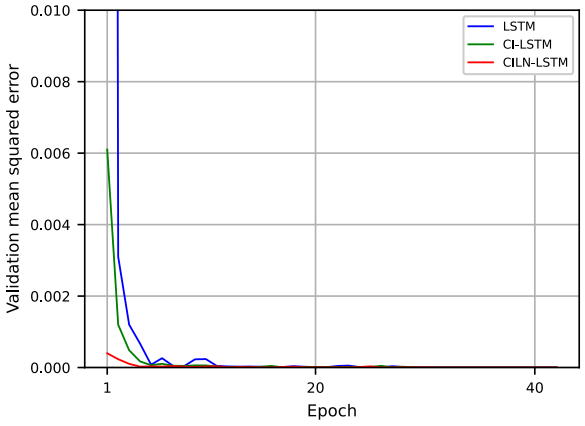
FIGURE 5. The mean MSE results obtained by evaluating the models on the GSPTSE validation set across three independent training runs, while employing hidden layer units (L) of both 16 and 32.

specific, crucial pieces of information embedded within long data sequences. It is meticulously structured to examine various aspects of a neural network's functionality, including its ability to maintain attention over important data points spread across a sequence, identify the relevant information necessary for solving a given problem, and perform precise manipulations on this information to reach a correct solution. Specifically, the inputs to the network consist of a series of K pairs (x_k, y_k), with $0 \leq k \leq K$. The first element x_k is a real-valued number between 0 and 1, and y_k is the corresponding marker that can take exactly two values: 0 and 1. Within the entire sequence, a distinctive feature is that only two markers are deliberately set to the binary value of 1, effectively acting as key indicators, while the remaining elements are set to 0. The goal of the network is to compute the sum of the two numbers from the first sequence that have corresponding marker values equal to 1 in the second sequence. This requirement places a demand on the network to not only recognize and locate the relevant numbers but also to execute the addition operation with precision. We conducted experiments with sequence lengths (T) of 100, 200, and 500 to assess model performance. The training set size was 105000 instances, the validation set size was 15000 instances and the test set was 30000 instances. All models were evaluated using a fixed number of hidden layer units (L) set at 128, with a mini-batch size of 50. It was established through a series of evaluations that additional hyperparameter tuning did not contribute to the effectiveness of the models. The performance of these models was assessed primarily using MSE as a key performance metric.

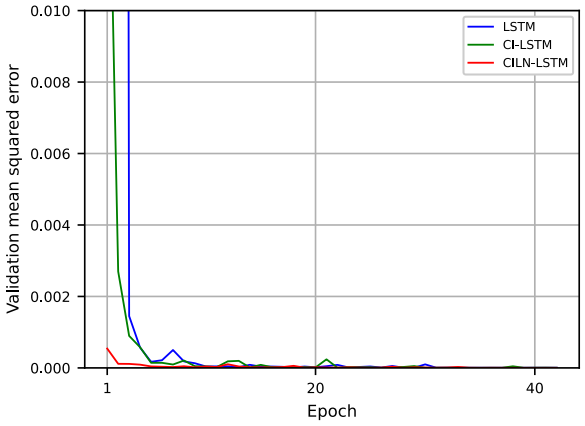
Visually observing the graph in Figure 6, where the obtained MSE on the validation dataset for the given models is displayed, it's evident that the CILN-LSTM model attains the highest level of performance, particularly when considering the speed of convergence towards the optimal minimum. However, regardless of the performance of the CILN-LSTM model, the other models also performed excellently on the task as well, but with a bit more time required. We need to emphasize that despite the training being carried out for a total of 100 epochs, not all of them are necessary to display, as the significant learning differences are visible only in the initial stages. Therefore, the graph is deliberately simplified to highlight the epochs that illustrate the most substantial changes in performance, allowing for a clearer comparison of the learning efficiency between the models over the crucial early epochs. Furthermore, a table displaying the test dataset evaluation results is omitted for simplicity, as all models achieved results around 10^{-6} with minimal fluctuations, regardless of input sequence length.

E. THE COPYING PROBLEM

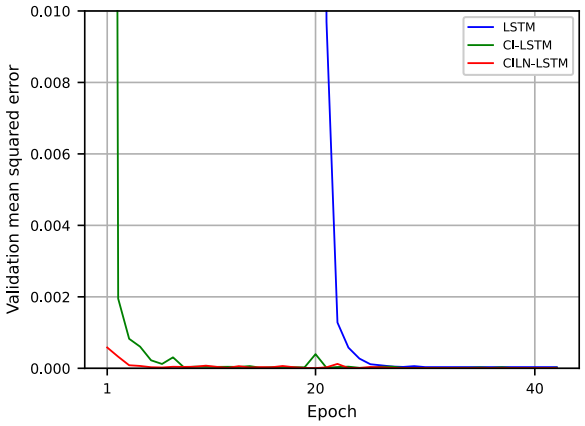
The copying memory task, introduced in [1], assesses how RNNs manage long-term memory and recall information from the distant past. It highlights the challenge of training neural networks to copy and reproduce sequences of varying lengths, serving as a benchmark for a recurrent model's ability to learn and handle complex long-term dependencies [24]. The same setup as in [3] was followed, with a brief overview provided here. $A = \{a_i\}_{i=1}^{K}$ is a set of K symbols and $S, T \in \mathbb{N}$ are randomly chosen. The input encompasses a $T + 2S$ length sequence of categories, where the initial S entries are to be remembered, all the while being sampled uniformly and independently and with replacement from $\{a_i\}_{i=1}^{K}$. The subsequent $T - 1$ inputs are set to a_{K+1} and they indicate a placeholder or empty category. Furthermore, the next input a_{K+2} suggests that the network should predict the initial S entries of the input, and it may be viewed as a delimiter. The rest of the S inputs are set to a_{K+1}. The anticipated output sequence consists of the $T + S$ repeated entries of a_{K+1}, succeeded by the initial S categories of the input sequence retained exactly in the same order. The experiments systematically explored model performance across different sequence lengths (T) at values of 100, 200, and 500. The training set size was 100000 instances, the validation set size was 10000 instances, and the test set was 40000 instances. All models were tested, considering hidden layer units (L)



a) T = 100

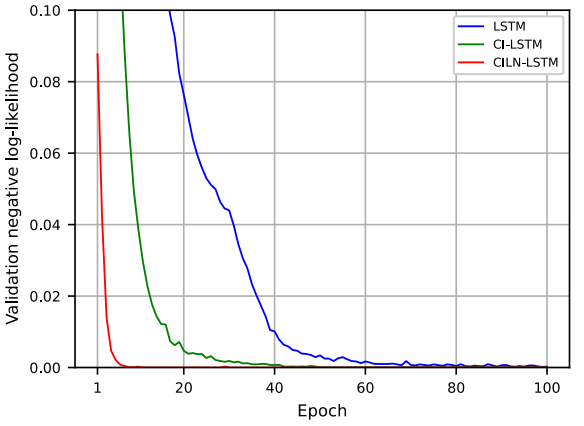


b) T = 200

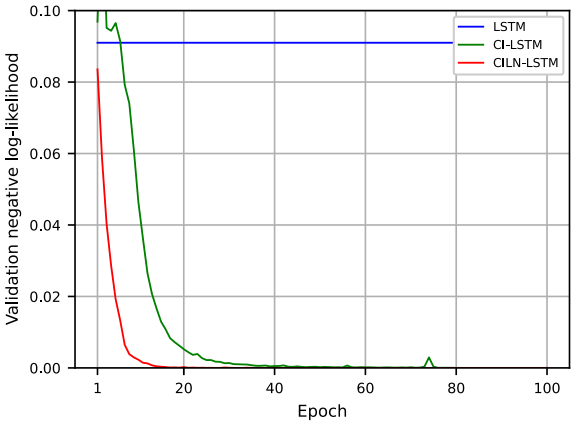


b) T = 500

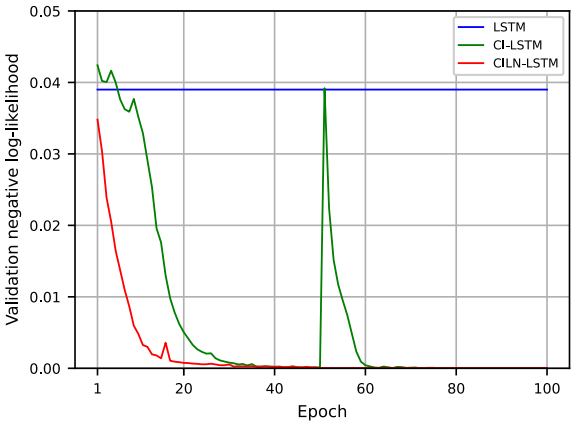
FIGURE 6. The mean MSE results obtained by evaluating the models on the validation set across three independent training runs for the adding task, while considering sequences of lengths (T) 100, 200, and 500. The displayed 40 epochs are representative.



a) T = 100



b) T = 200



b) T = 500

FIGURE 7. The mean negative log-likelihood results obtained by evaluating the models on the validation set across three independent training runs for the copying memory task, while considering sequences of lengths (T) 100, 200, and 500.

128, with a mini-batch size of 50. It was established through a series of evaluations that additional hyperparameter tuning did not contribute to the effectiveness of the models. The performance of these models was assessed primarily using the negative log-likelihood as a key performance metric.

Upon visual inspection of the graph in Figure 7, it's evident that the CILN-LSTM model outperforms the other models, showing faster convergence and generally reaching a more favorable optimum on the validation dataset. It should be highlighted that the conventionally initialized LSTM

TABLE 5. The mean negative log-likelihood results obtained by evaluating the models on the test set during three independent training runs are listed in the table. The best negative log-likelihood results from the copy memory experiments are presented in bold.

Model	T = 100	T = 200	T = 500
LSTM	10^{-4}	10^{-2}	10^{-2}
CI-LSTM	10^{-4}	10^{-4}	10^{-4}
CILN-LSTM	$\mathbf{10^{-6}}$	$\mathbf{10^{-6}}$	10^{-4}

encounters significant challenges in solving the copy memory problem, particularly for sequences longer than roughly 100, as stated in prior studies [3], [10], [24]. However, the introduction of CI leads to convergence. Finally, as evident from the results presented in Table 5, CILN-LSTM most often delivered better results on the test data set, notably in scenarios where the sequence length was shorter, specifically when T was set to 100 and 200. When T was set to 500, the performance of CILN-LSTM aligns closely with that of CI-LSTM.

F. SYMMETRIC PATTERN LEARNING

This paper introduces a synthetic task to evaluate RNN models' ability to recognize, retain, and recall widely spaced symmetrical patterns. It assesses the models' memory capabilities in capturing unique patterns during sequence processing, focusing on storing information from both ends and identifying palindromic patterns. Palindromic sequences vary in length, capped at 10 characters in one version and limited to 50 characters in a more challenging version. Each element of the sequence contains any English alphabet letter except for x, which is used for padding to ensure uniform sequence lengths. The training set comprised 10,000 instances, the test set 2,000 instances, and 10% of the training data served as the validation set. The models were tested with hidden layer units of 64 and 128, using a mini-batch size of 32. An L2 regularization factor was set to 1e-5, and the gradient norm was clipped at a value of 5. The performance of the models was assessed using accuracy as a key performance metric.

TABLE 6. The mean accuracies obtained by evaluating the models on the test set during three independent training runs are listed in the table. The best accuracies from the symmetric pattern learning experiments are presented in bold. The maximum length of the palindrome was limited to 10.

Model	L = 64	L = 128
LSTM	97.23 %	97.79 %
CI-LSTM	97.28 %	97.83 %
CILN-LSTM	**99.19 %**	**99.41 %**

In this newly introduced task, where the maximum palindrome length is set to 10, it has once again been demonstrated that the CILN-LSTM model achieves significantly better performance compared to the other models. Visually observing the graph in Figure 8, it can easily be concluded that the CILN-LSTM model, in addition to faster convergence towards the optimal minimum, also achieves better accuracy on the validation dataset. The evaluation results on the test dataset presented in Table 6 confirm that the CILN-LSTM model outperformed its counterparts.

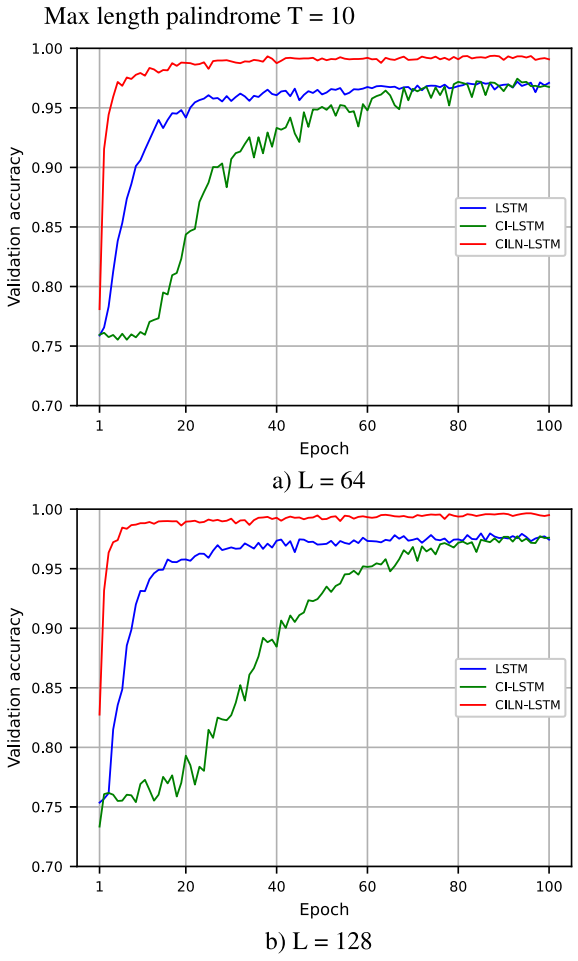


FIGURE 8. The mean accuracies obtained by evaluating the models on the validation set for the symmetric pattern learning task across three independent training runs, while employing hidden layer units (L) of both 64 and 128.

TABLE 7. The mean accuracies obtained by evaluating the models on the test set during three independent training runs are listed in the table. The best accuracies from the symmetric pattern learning experiments are presented in bold. The maximum length of the palindrome was limited to 50.

Model	L = 64	L = 128
LSTM	74.58 %	74.81 %
CI-LSTM	93.88 %	94.67 %
CILN-LSTM	**98.31 %**	**98.97 %**

In the more complex variation of the experiment, where the maximum palindrome length is set to 50, the CILN-LSTM model continues to exhibit better performance when compared to its counterparts. It consistently demonstrates a faster convergence towards the optimal minimum, resulting

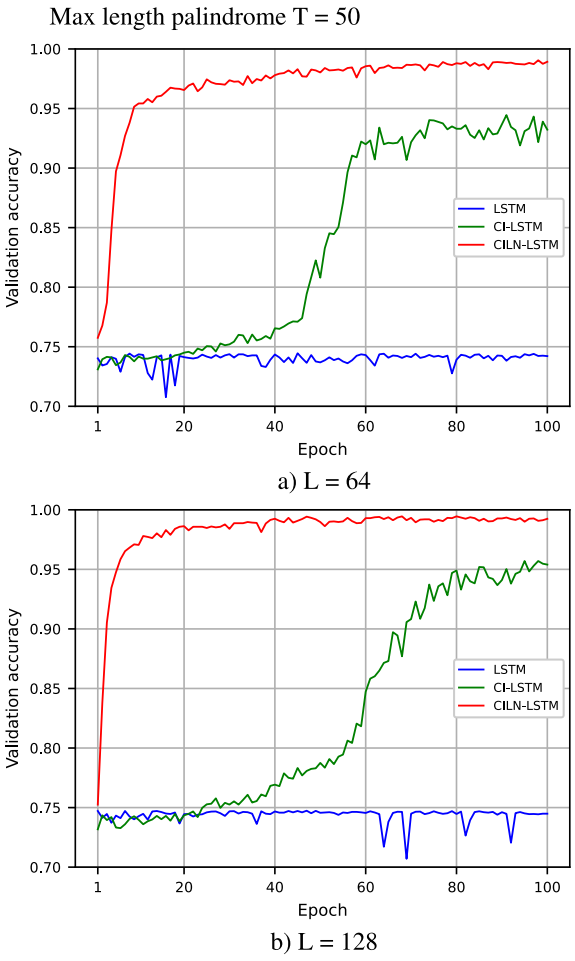


FIGURE 9. The mean accuracies obtained by evaluating the models on the validation set for the symmetric pattern learning task across three independent training runs, while employing hidden layer units (L) of both 64 and 128.

in significantly higher accuracy rates, which can be visually confirmed in Figure 9 obtained from the evaluation of the models on the validation dataset. The LSTM and CI-LSTM models, while having their strengths, faced significant challenges in the endeavor to uncover concealed patterns within the data. These difficulties reveal their limitations in this specific context, as they struggled to identify the underlying structures that the CILN-LSTM model could more readily detect. The evaluation results of the models on the test dataset, specifically highlighted in Table 7, significantly favor the CILN-LSTM model. These results highlight its ability to achieve higher accuracy and efficiency, showcasing its capacity to understand the complex patterns hidden within the test dataset. This underscores the adaptability and effectiveness of the CILN-LSTM model in handling various data complexities.

G. RESISTANT SEQUENTIAL FUSION

This paper introduces a task designed to evaluate models' abilities to handle irrelevant information, maintain relationships between distant elements in sequences, and accumulate information over extended time intervals. The models must selectively focus on relevant information while filtering out what is not important. Additionally, they need to manage long-range dependencies, ensuring that relationships between distant elements within a sequence are accurately maintained. This comprehensive evaluation helps in understanding the models' proficiency in processing complex sequences, making them more robust and effective in real-world applications where data often contains a mix of important and irrelevant information. The concept involves sequences of length n, where the elements are aggregated up until the k-th element, while disregarding the remaining $n - k$ elements. In the second variation of this experiment, we expand the mentioned concept by additionally summing elements up to the n-th element, starting from the $(n - k)$-th element. Essentially, the first variation involves summing the first k elements and disregarding the rest, whereas in the second variation, it encompasses summing the first k elements and the last k elements while excluding the elements in between. In the first variation of the experiment, sequence lengths were set at 50, whereas in the second variation, they were set at 100. In both cases, k was set to 20. The training set size was 10000 instances and the test set was 2000 instances. As the validation set, 10% of the training data was used. The models were tested, considering hidden layer units (L) of 64 and 128, employing a mini-batch size of 32. An L2 regularization factor was set to 1e-5, and the gradient norm was clipped at a value of 5. The performance of these models was assessed primarily using MSE as a key performance metric.

TABLE 8. The mean MSE results obtained by evaluating the models on the test set during three independent training runs are listed in the table. The best MSE results from the resistant sequential fusion experiments are presented in bold. The sequence length (T) was set to 50.

Model	L = 64	L = 128
LSTM	10^{0}	10^{0}
CI-LSTM	10^{-4}	10^{-4}
CILN-LSTM	10^{-4}	$\mathbf{10^{-5}}$

TABLE 9. The mean MSE results obtained by evaluating the models on the test set during three independent training runs are listed in the table. The best MSE results from the resistant sequential fusion experiments are presented in bold. The sequence length (T) was set to 100.

Model	L = 64	L = 128
LSTM	10^{-2}	10^{-2}
CI-LSTM	10^{-2}	10^{-2}
CILN-LSTM	$\mathbf{10^{-3}}$	$\mathbf{10^{-4}}$

Considering sequences of length 50, the CILN-LSTM and CI-LSTM models achieved satisfactory results on the validation dataset in terms of convergence. However, compared to the CI-LSTM, the CILN-LSTM exhibited faster convergence, as shown in Figure 10. Evaluation on the test dataset showed negligible differences between the CILN-LSTM and CI-LSTM models in cases of a hidden layer with 64 units. This

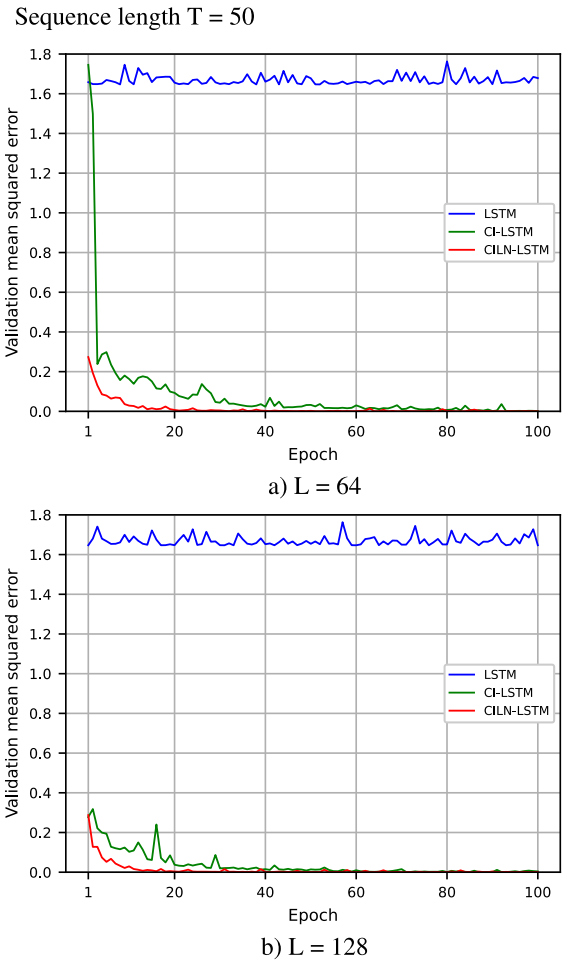


FIGURE 10. The mean MSE results obtained by evaluating the models on the validation set across three independent training runs for the resistant sequential fusion task, while employing hidden layer units (L) of both 64 and 128, are listed in the table.

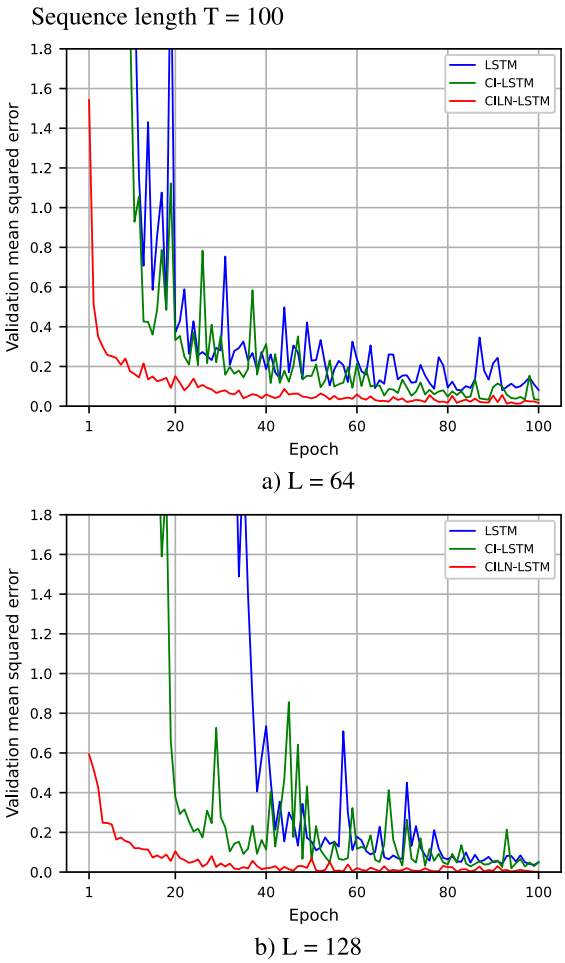


FIGURE 11. The mean MSE results obtained by evaluating the models on the validation set across three independent training runs for the resistant sequential fusion task, while employing hidden layer units (L) of both 64 and 128, are listed in the table.

indicates that both models are capable of handling the task with smaller network sizes effectively. However, increasing the units in the hidden layer to 128 led to test set results differing by an order of magnitude, as seen in Table 8, in favor of CILN-LSTM. This analysis also highlighted that LSTM models might encounter challenges when dealing with this type of task. Specifically, the LSTM models, while generally robust and versatile for many sequence processing tasks, showed limitations in handling long-range dependencies and effectively ignoring irrelevant information.

In the next variation of the experiment, Figure 11 clearly shows that CILN-LSTM once again achieved the best performance on the validation set. The evaluation results on the test set, as indicated in Table 9, strongly favor the CILN-LSTM model. A notable observation is the significant improvement in the performance of the CILN-LSTM model after increasing the number of units in the hidden layer. This enhancement underscores the model's capacity to leverage additional computational power effectively, leading to better learning and performance outcomes. In contrast, the other models in the experiment showed only minor enhancements, despite similar adjustments, and their performance remained within the same order of magnitude. Additionally, it is important to notice that both the LSTM and CI-LSTM models exhibited significantly more fluctuations during training compared to the CILN-LSTM model, as can be seen from Figure 11.

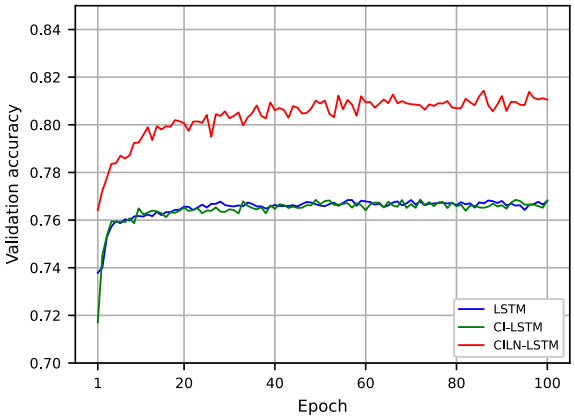
H. GENDER CLASSIFICATION BY NAMES

This experiment addresses the challenge of predicting an individual's gender based solely on their first name. The task of gender prediction from names is particularly complex due to the significant diversity in naming conventions across various cultures and languages. A name that is common and distinctly gendered in one culture may be rare or unisex in another. Moreover, the evolving nature of names over time adds another dimension of difficulty. Some names that were traditionally associated with a particular gender may now be used for multiple genders, reflecting changing social norms and trends, while others are inherently ambiguous and can be associated with multiple genders, adding another layer of complexity to this task. These cultural

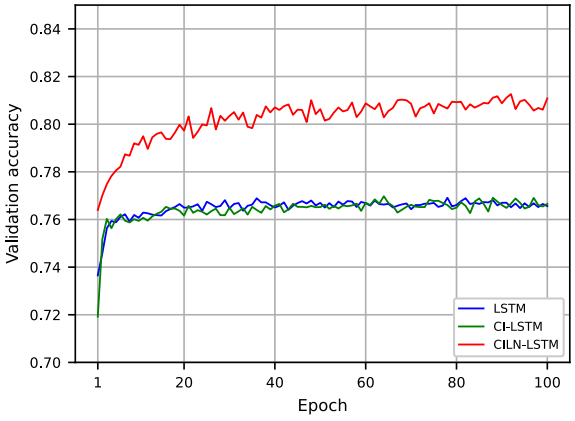
TABLE 10. The mean accuracies obtained by evaluating the models on the test set during three independent training runs are listed in the table. The best accuracies from the gender classification by names experiments are presented in bold.

Model	L = 32	L = 64
LSTM	77.26 %	78.42 %
CI-LSTM	77.19 %	77.28 %
CILN-LSTM	**81.14 %**	**81.33 %**

and linguistic variations necessitate the development of a robust and adaptable model capable of accurately predicting gender from names. The dataset contains names and their associated genders. Within it, the training set comprises 41,300 instances, the test set 10,320 instances, and 10% of the training data served as the validation set. To handle ambiguous and inconsistent entries, the data underwent a cleaning process before being utilized in the model training. This step was crucial to ensure the quality and reliability of the dataset, as inconsistent data could adversely affect the model's performance. Experimentation began with an embedding layer set to a dimension of 64, while the hidden units of the components were set to 32. In the second variant, these values were doubled. Training was conducted with a mini-batch size of 128. A dropout value of 0.5 was applied to the output of the recurrent layers. Additionally, An L2 regularization factor was set to 1e-2, and the gradient norm was clipped at a value of 5. The performance of the models was assessed using accuracy as the key performance metric.



a) L = 32



b) L = 64

FIGURE 12. The mean accuracies obtained by evaluating the models on the validation set across three independent training runs for the gender classification by names task, while employing hidden layer units (L) of both 64 and 128.

The experiment results indicate that the CILN-LSTM model performs better in gender classification from names compared to the LSTM and CI-LSTM models. As shown in Figure 12 and Table 10, the CILN-LSTM model achieved higher accuracy for both configurations of hidden layer units, and the convergence speed towards the optimal minimum favors the CILN-LSTM model. Notably, the LSTM and CI-LSTM models produced nearly identical results, demonstrating similar performance levels. Furthermore, increasing the number of hidden layer units did not significantly improve performance in any of the models. Additionally, the LSTM and CI-LSTM models quickly reached a plateau in validation accuracy after only a few training epochs, highlighting certain limitations in their learning capacity compared to the CILN-LSTM model. To address one of these potential limitations, specifically the risk of overfitting, we applied regularization techniques. While these techniques helped reduce overfitting, they did not lead to a significant improvement in the overall generalization of these models.

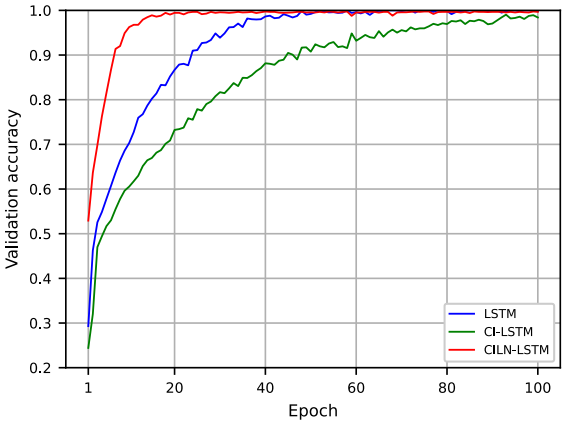
I. SMART HOME EVENT FORECASTING

In this experiment, we utilize data from the PEEVES dataset (Physical Event Verification in Smart Homes) [25] to train models aimed at predicting future events. The dataset is a comprehensive collection of data gathered from various sensors installed within a home environment. These sensors continuously record a wide range of activities, providing a rich source of information for analysis. The dataset encompasses events from 12 different event sources, captured through 48 sensors deployed in an office environment over twelve days. These sensors monitor a variety of activities, including door openings and closings, light switching, window shade movements, fridge door openings, coffee machine usage, PC power status changes, screen status changes, fan operation, radiator status changes, doorbell usage, window openings and closings, and smart camera status changes. Each sensor logs events with precise timestamps, offering detailed tracking of when specific events occur. The data is loaded and processed to generate event sequences, representing a series of activities over a given time period. We train the models to learn patterns in these event sequences, enabling them to predict future events based on past data. The prediction process begins with an initial event, referred to as the seed event. Based on this starting point, the models generate sequences of future events. These predicted sequences can then be evaluated by comparing them against actual recorded events. This comparison helps identify any discrepancies, which can indicate irregularities or potential false reports. After constructing the dataset, we divided

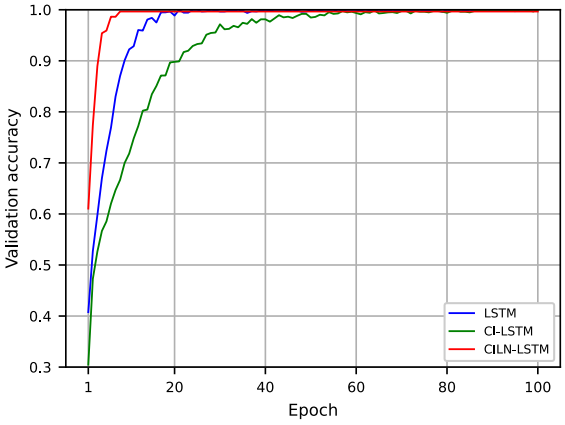
TABLE 11. The mean accuracies obtained by evaluating the models on the PEEVES test set during three independent training runs are listed in the table. The best accuracies from the experiments are presented in bold.

Model	L = 32	L = 64
LSTM	99.67 %	99.78 %
CI-LSTM	98.86 %	99.72 %
CILN-LSTM	**99.83 %**	**99.91%**

it into training, test, and validation sets. The training set comprised 5,870 entries, while the test set contained 1,580 entries. Additionally, 10% of the training data served as the validation set. We conducted our experiments beginning with an embedding layer set to a dimension of 64, while the hidden units of the components were initially set to 32. In the second variant of our experiments, these values were doubled. Training was conducted with a mini-batch size of 50, and a dropout value of 0.5 was applied to the output of the recurrent layers to prevent overfitting. Additionally, An L2 regularization factor was set to 1e-3, and the gradient norm was clipped at a value of 5 to stabilize the training process.



a) L = 32



b) L = 64

FIGURE 13. The mean accuracies obtained by evaluating the models on the PEEVES validation set across three independent training runs, while employing hidden layer units (L) of both 64 and 128.

Evaluation results on the test dataset, as shown in Table 11, indicate that the final accuracies of the models are similar, with the CI-LSTM model performing slightly worse compared to the others when fewer hidden layer units were used. However, a significant difference in favor of the CILN-LSTM model lies in the speed of convergence to the optimal minimum, as depicted in the graph in Figure 13, indicating its training efficiency. These results suggest that, although the final accuracies are similar, the CILN-LSTM model offers advantages in training speed, which can be very important in real-world smart home applications that require rapid adaptation and accurate predictions.

VIII. ABLATION ANALYSIS

In our experimentation, we explored various negative bias values for the output gate to better understand their impact on the CILN-LSTM network's behavior. Intuitively, during the initial phase, the output gate should exhibit behavior similar to the input gate. This similarity in behavior is attributed to the fact that both gates have identical components in their construction, and their activation values are generally similar, typically hovering around 0. With (7) taken into consideration, it made sense to experiment with initializing the output gate bias using the CI technique. Furthermore, when examining the CILN-LSTM architecture and its utilization of sigmoid activation functions, LN is applied jointly to the combined forward and recurrent inputs, after which the bias is incorporated through addition. In this context, a scale parameter γ is learned. This could imply that if the input gate and output gate have biases that deviate significantly from each other, it may be expected that the scale parameter γ is more challenging to learn. In the end, through experimentation, we concluded that it was most effective to apply the CI technique to the output gate bias because it consistently produced the best results.

As mentioned earlier, in certain cases, it was also observed that the omission of LN from the hidden layers resulted in a significant decrease in the model's performance. The aforementioned phenomenon can be attributed to the fact that without LN, the variance of input components passing through the activation functions can become exceptionally low, especially the values that are forwarded to the linear layer. This substantial reduction in variance significantly impedes the learning process, making it much more challenging for the network to adapt and improve its performance during training. When LN was applied to the hidden layers, a change in input component variance occurred due to the alteration in the distribution. This, in turn, closely correlated with improved model performance, as visualized in Figure 14.

The figure clearly illustrates a positive trend, where the introduction of LN leads to higher model performance. This underscores the crucial role of LN in maintaining an appropriate level of variance in the input components, which, in turn, facilitates more effective learning and contributes to the overall enhanced performance of the model. In this context, we are not discussing how incremental increases

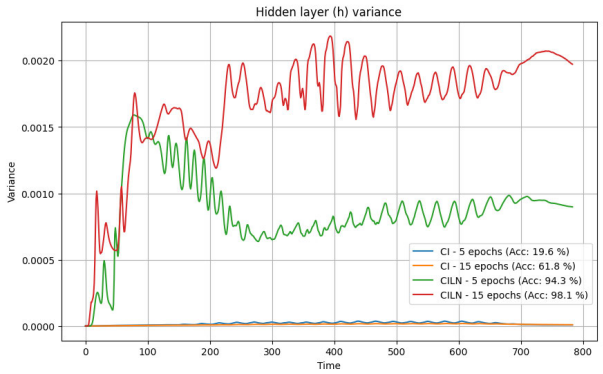


FIGURE 14. The data points for this plot were collected during the processing of the MNIST test dataset. Variance values of the hidden layer were recorded for each time step of every instance, and then the average variance value was calculated. The figure includes informative accuracy values on the test dataset, where these values were observed with respect to two different numbers of epochs, both with and without LN. The gate biases were initialized according to the CILN-LSTM architecture.

in variance automatically lead to incrementally improved model performance. Instead, we are focusing on the impact of LN on the variance of input components passing through activation functions and its connection to the learning process. In our case, the change in variance will manifest as an increase in variance, while enabling the network to adapt and learn relevant features. Furthermore, extending LN to all components within our formalized architecture yielded even more promising results. This comprehensive application of LN further elevated the overall performance of the neural network, highlighting its importance as a stabilizing and performance-boosting architectural component.

IX. FUTURE WORK

In future work, the goal is to explore additional approaches to enhance the performance and stability of the CILN-LSTM model through innovative methods in initialization and layer normalization. One of the main objectives will be to investigate alternative initialization methods that can provide better initial conditions for training, thereby further reducing the problems of exploding and vanishing gradients. The focus will be on methods that can ensure more stable initial values and enable faster and more efficient model convergence. For example, the potential of Uniform Gate Initialization (UGI) [5] will be investigated to enhance the model. UGI ensures that all timescales are covered, with units having very high forget activations capable of remembering information nearly indefinitely, while those with low activations focus solely on the incoming input. Notably, this technique does not require additional parameters. Additionally, advanced layer normalization techniques will be explored to further enhance training stability and the model's adaptability to various data distributions. Methods that refine the normalization process to better suit the dynamic nature of sequential data will be investigated. This includes evaluating the effectiveness of normalization techniques that can adapt more rapidly to shifts in data patterns. Exploring these advanced techniques will help understand their potential in reducing training times and improving model performance. Specific advanced layer normalization methods mentioned in [26] will also be examined to identify the most promising approaches. Furthermore, detailed experimental studies will be conducted to assess the impact of these new techniques on various tasks, including prediction, classification, and sequential learning tasks. Different datasets will be used to ensure comprehensive evaluation and identify the best approaches for specific problems. The objective is to contribute to the further development of the CILN-LSTM model, making it more robust and efficient for a wide range of applications. These findings are also expected to provide valuable guidelines for future researchers in the field of neural networks and deeplearning.

X. CONCLUSION

In this research, we compared three different models of RNNs: LSTM, CI-LSTM, and CILN-LSTM. Our goal was to examine their capabilities and performance in a specific task or application. Through our evaluation and analysis, we have drawn the following conclusions: The LSTM has gained prominence in the field of deep learning due to its notable performances in various tasks. Its ability to mitigate the vanishing gradient problem and effectively capture long-term dependencies has made it a popular choice for sequence modeling, machine translation, speech recognition, and sentiment analysis, among other applications. The LSTM's success in these domains has established it as a reliable and powerful tool in the realm of RNNs. It exhibited solid performance in handling sequential data, demonstrating its effectiveness in capturing long-term dependencies. It showed competitive results and provided a strong baseline for comparison. The CI-LSTM, a variation of LSTM in which the forget and input gate biases are initialized using the CI technique, demonstrated enhanced performance in tasks reliant on sequential data. On occasion, it achieves better predictive accuracy when compared to the LSTM model and successfully demonstrates enhanced modeling capabilities in various tasks. The CILN-LSTM, an enhanced version of the LSTM and CI-LSTM that incorporates the CI and LN techniques, demonstrated commendable performance. The new architecture utilizes the CI technique to initialize the biases not only for the forget and input gates but also for the output gate. In this context, CI addressed the challenges of both vanishing and exploding gradients, while LN technique enhanced the stability of the training process. Therefore, by utilizing these techniques, we successfully achieved an unhindered gradient flow. Moreover, the synergistic effect of these two methods resulted in enhanced stability in the training dynamics, notably augmenting the performance of the recurrent neural network, thereby implying significant enhancements in overall outcomes.

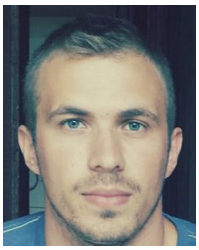
In summary, our findings indicate that the CILN-LSTM model exceeds the performance of both the LSTM and CI-LSTM in capturing long-term dependencies and effectively

utilizing sequential information. While the LSTM and CI-LSTM continue to be widely used, the CILN-LSTM model holds the potential to outperform both on a wide variety of tasks.

During this research, the programming language Python was utilized in conjunction with the Keras functional API and TensorFlow to construct and train the models. The experiments were conducted within the Google Colab environment.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[2] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.
[3] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1120–1128.
[4] Z. He, S. Gao, L. Xiao, D. Liu, H. He, and D. Barber, "Wider and deeper, cheaper and faster: Tensorized LSTMs for sequence learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.
[5] A. Gu, C. Gulcehre, T. L. Paine, M. Hoffman, and R. Pascanu, "Improving the gating mechanism of recurrent neural networks," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 3800–3809.
[6] D. Arpit, B. Kanuparthi, G. Kerg, N. R. Ke, L. Mitliagkas, and Y. Bengio, "*h*-detach: Modifying the LSTM gradient towards better optimization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–19.
[7] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, Oct. 2000.
[8] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proc. 32nd Int. Conf. Mach. Learn.*, Jun. 2015, pp. 2342–2350.
[9] C. Tallec and Y. Ollivier, "Can recurrent neural networks warp time?" in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.
[10] J. van der Westhuizen and J. Lasenby, "The unreasonable effectiveness of the forget gate," 2018, *arXiv:1804.04849*.
[11] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," 2012, *arXiv:1211.5063*.
[12] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[13] M. Zeineldeen, A. Zeyer, R. Schlüter, and H. Ney, "Layer-normalized LSTM for hybrid-hmm and end-to-end ASR," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 7679–7683.
[14] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[15] J. Heck and F. M. Salem, "Simplified minimal gated unit variations for recurrent neural networks," 2017, *arXiv:1701.03452*.
[16] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 2, pp. 92–102, Apr. 2018.
[17] H.-Y. S. Chien, J. S. Turek, N. Beckage, V. A. Vo, C. J. Honey, and T. L. Willke, "Slower is better: Revisiting the forgetting mechanism in LSTM for slower information decay," 2021, *arXiv:2105.05944*.
[18] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw. IJCNN, Neural Comput., New Challenges Perspect. New Millennium*, Jul. 2000, pp. 189–194.
[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[20] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," 2015, *arXiv:1504.00941*.
[21] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[22] K. Albeladi, B. Zafar, and A. Mueen, "Time series forecasting using LSTM and ARIMA," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 1, pp. 313–320, 2023.
[23] W. Chen, W. Hussain, F. Cauteruccio, and X. Zhang, "Deep learning for financial time series prediction: A state-of-the-art review of standalone and hybrid models," *Comput. Model. Eng. Sci.*, vol. 139, no. 1, pp. 187–224, 2024.
[24] L. Jing, C. Gulcehre, J. Peurifoy, Y. Shen, M. Tegmark, M. Soljacic, and Y. Bengio, "Gated orthogonal recurrent units: On learning to forget," *Neural Comput.*, vol. 31, no. 4, pp. 765–783, Apr. 2019.
[25] S. Birnbach, S. Eberz, and I. Martinovic, "Peeves: Physical event verification in smart homes," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1455–1467.
[26] J. Xu, X. Sun, Z. Zhang, G. Zhao, and J. Lin, "Understanding and improving layer normalization," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2019, pp. 4381–4391.



ANTONIO TOLIC was born in Zagreb, Croatia. He holds dual master's degrees in applied mathematics and computer engineering. He is currently pursuing the dual postgraduate degrees, focusing on the mathematical theories behind deep learning models and statistical methods for economic analysis and forecasting.

He has authored several articles in his fields of interest. In addition to his academic pursuits, he is also engaged in the industry, working at Ericsson. His research interests include exploring advanced deep learning models for various applications, such as natural language processing, computer vision, and generative modeling, alongside a notable interest in understanding economic trends and enhancing decision-making processes.

BILJANA MILEVA BOSHKOSKA received the Ph.D. degree from Jozef Stefan International Postgraduate School.

She is currently a Professor of computer science and information systems and the Head of the Laboratory, School of Graduate Professional Studies, Faculty of Information Studies, Slovenia. She teaches courses in *Decision Support Systems* and *Machine Learning*. Her work has also appeared in other prestigious academic outlets, such as *Computers in Industry*, *Journal of Decision Systems*, and *Computer Methods and Programs in Biomedicine*. She has published in top academic journals in the field of *Computer Science and Information Systems*, such as *Expert Systems with Applications*, *European Journal of Operational Research*, and *Knowledge-Based Systems*. Her research interests include decision support systems, data-driven decision models, and machine learning for obtaining analytical models on processes.

Dr. Boshkoska is an Editor of *Journal of Decision Support Systems Technology*.

SANDRO SKANSI was born in Zagreb, Croatia, in 1985. He received the M.A. degree in philosophy and Croatian culture from the University of Zagreb, in 2009, and a thesis in logic from the University of Zagreb, in 2013.

From 2017 to 2023, he was an Assistant Professor of logic with the University of Zagreb. Since 2023, he has been an Associate Professor of logic and artificial intelligence with the University of Zagreb. Prior to his academic post, he worked in software development, data science, and intelligence analysis for homeland security. His current research interests include cellular automata, complex systems and cybernetics, and artificial neural networks.

Mr. Skansi is a Life Member of the Association for the Advancement of Artificial Intelligence and the National Rifle Association.