Review

Not peer-reviewed version

Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications

Ibomoiye Domor Mienye * , Theo G. Swart , George Obaido

Posted Date: 12 August 2024

doi: 10.20944/preprints202408.0748.v1

Keywords: Deep learning; GRU; LSTM; Machine learning; NLP; RNN

 doi:10.20944/preprints202408.0748.v1

Article

Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications

Ibomoiye Domor Mienye [1,*,†], Theo G. Swart [1,†] and George Obaido [2,†]

[1] Institute for Intelligent Systems, University of Johannesburg, Johannesburg 2006, South Africa; ibomoiyem@uj.ac.za
[2] Center for Human-Compatible Artificial Intelligence (CHAI), Berkeley Institute for Data Science (BIDS), University of California, Berkeley, Berkeley, California, 94720, USA; gobaido@berkeley.edu
* Correspondence: ibomoiyem@uj.ac.za
† These authors contributed equally to this work.

Abstract: Recurrent Neural Networks (RNNs) have significantly advanced the field of machine learning by enabling the effective processing of sequential data. This paper provides a comprehensive review of RNNs and their applications, highlighting advancements in architectures such as Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), Bidirectional LSTM (BiLSTM), and stacked LSTM. The study examines the application of RNNs in different domains, including natural language processing (NLP), speech recognition, financial time series forecasting, bioinformatics, autonomous vehicles, and anomaly detection. Additionally, the study discusses recent innovations, such as the integration of attention mechanisms and the development of hybrid models that combine RNNs with convolutional neural networks (CNNs) and transformer architectures. This review aims to provide machine learning researchers and practitioners with a comprehensive overview of the current state and future directions of RNN research.

Keywords: deep learning; GRU; LSTM; machine learning; NLP; RNN

1. Introduction

Deep learning (DL) has reshaped the field of artificial intelligence (AI), driving advancements in a wide array of applications, from image recognition and natural language processing (NLP) to autonomous driving and medical diagnostics [1–3]. This rapid growth is fueled by the increasing availability of big data, advancements in computing power, and the development of sophisticated algorithms [4–6]. As DL models continue to evolve, they are increasingly being deployed in critical sectors, demonstrating their ability to outperform traditional machine learning (ML) techniques in handling complex tasks.

Recurrent neural networks (RNNs) are a class of deep learning models that are fundamentally designed to handle sequential data [7,8]. Unlike feedforward neural networks, RNNs possess the unique feature of maintaining a memory of previous inputs by using their internal state (memory) to process sequences of inputs [9]. This makes them ideally suited for applications such as natural language processing, speech recognition, and time series forecasting, where context and the order of data points are crucial.

The inception of RNNs dates back to the 1980s, with foundational work by Rumelhart et al. [10], which introduced the concept of backpropagation through time (BPTT). However, RNNs struggled with practical applications due to the vanishing gradient problem, where gradients either grow or shrink exponentially during backpropagation [11]. Meanwhile, the introduction of Long Short-Term Memory (LSTM) networks by Hochreiter and Schmidhuber [12] was a turning point for RNNs, allowing for the learning of dependencies over much longer periods. Additionally, gated recurrent units (GRUs), proposed by Cho et al. [13], offered a simplified alternative to LSTMs while maintaining comparable performance.

Over the years, these RNN architectures have been applied in different fields, achieving excellent performance [14–16]. Despite their advancements and adoption in various fields, RNNs have continued to evolve. Specifically, the increasing complexity of data and tasks in recent years has driven continuous innovations in RNN architectures and variants. These developments have expanded the application of

 doi:10.20944/preprints202408.0748.v1

RNNs from simple sequence prediction to complex tasks such as multimodal learning and real-time decision-making systems.

Recent studies and reviews have highlighted the significant progress made in the field of RNNs. For example, Lipton et al. [17] provided an overview of the theoretical foundations and applications of RNNs, while Yu et al. [18] focused on the LSTM cell and the different variants. Additionally, Tarwani et al. [19] reviewed the application and role of RNNs in natural language processing. However, many of these reviews do not fully capture the latest advancements and applications, given the rapid pace of innovation in this field. Additionally, there remains a gap in the literature that comprehensively covers the latest advancements in RNN architectures and their applications across a broader range of fields. Therefore, this paper aims to fill that gap by providing a comprehensive review of RNNs, assessing their theoretical advancements and practical implementations, and cutting-edge applications, thus helping to shape future research in neural networks.

The rest of this paper is organized as follows: Section 2 reviews related works. Section 3 covers the fundamentals of RNNs, including basic architecture and components. Section 4 explores advanced RNN variants, such as LSTM and GRU. Section 5 highlights innovations in RNN architectures and training methodologies. Section 6 discusses various applications of RNNs in the literature. Section 7 addresses challenges and future research directions. Finally, Section 8 concludes the study.

2. Related Works

Several reviews have been conducted on RNNs and their applications, each contributing to the understanding and development of the field. For instance, Dutta et al. [20] provided a comprehensive overview of the theoretical foundations of RNNs and their applications in sequence learning. Their review highlighted the challenges associated with training RNNs, particularly the vanishing gradient problem, and discussed the advancements in LSTM and GRU architectures. However, the review primarily focused on the theoretical aspects and applications of RNNs and did not extensively cover the latest innovations and practical applications in emerging fields such as bioinformatics and autonomous systems. Quradaa et al. [21] presented a start-of-the-art review of RNNs, covering the core architectures with a focus on applications in code clones.

Similarly, the review by Tarwani et al. [19] provided an in-depth analysis of RNNs in the context of NLP. While this review offered valuable insights into the advancements in NLP, it lacked a broader perspective on other application domains and recent architectural innovations. Another significant review by Goodfellow et al. [22] focused on the fundamentals of deep learning, including RNNs, and discussed their applications across various domains. This review provided a solid foundation but did not delve deeply into the specific advancements in RNN architectures and their specialized applications.

Greff et al. [23] conducted an extensive study comparing various LSTM variants to determine their effectiveness in different applications. While this review provided a thorough comparison, it primarily focused on LSTM architectures and did not address other RNN variants or the latest hybrid models. In a similar research, Al-Selwi et al. [24] reviewed LSTM applications in the literature, covering articles in the 2018-2023 time period. Zaremba et al. [25] reviewed the use of RNNs in language modeling, highlighting significant achievements and the ongoing challenges in this field. Their work offered valuable insights into the application of RNNs in NLP but was limited to language modeling and did not explore other potential applications. Bai et al. [26] provided a critical review of RNNs and their variants, comparing them with other sequence modeling techniques like convolutional neural networks (CNNs) and attention-based models. Che et al. [27] focused on the application of RNNs in healthcare, particularly for electronic health records (EHR) analysis and disease prediction. This review highlighted the potential of RNNs in medical applications.

Furthermore, more recent studies have explored various new aspects and applications of RNNs. For example, Chung et al. [28] explored the advancements in RNN architectures over the past decade, focusing on improvements in training techniques, optimization methods, and new architectural

 doi:10.20944/preprints202408.0748.v1

innovations. This review provided an extensive survey of recent developments and their implications for future research. Badawy et al. [29] provided a comprehensive overview of the use of RNNs in healthcare, particularly for predictive analytics and patient monitoring. They discussed the integration of RNNs with other ML techniques and the challenges in deploying these models in clinical settings.

Ismaeel et al. [30] examined the application of RNNs in smart city technologies, including traffic prediction, energy management, and urban planning. Their review discussed the potential and limitations of RNNs in these areas and suggested avenues for future research. Meanwhile, Mers et al. [31] reviewed the applications of RNNs in pavement performance forecasting and conducted a comprehensive performance comparison of the various RNN models, including simple RNN, LSTM, GRU, and hybrid LSTM-fully connected neural network (LSTM-FCNN).

Chen et al. [32] focused on the use of RNNs in environmental monitoring and climate modeling, discussing their effectiveness in predicting environmental changes and managing natural resources. They also highlighted the challenges in modeling complex environmental systems with RNNs. Linardos et al. [33] investigated the advancements in RNNs for natural disaster prediction and management, highlighting the successes and challenges in using RNNs for early warning systems, disaster response, and recovery planning. Zhang et al. [34] discussed RNN applications in robotics, particularly focusing on path planning, motion control, and human-robot interaction. They discussed the integration of RNNs with other DL techniques in robotics. The different related studies are Tabulated in Table 1, including their main contributions.

Table 1. Summary of Related Reviews on RNNs

Reference	Year	Description
Zaremba et al. [25]	2014	Insights into RNNs in language modeling.
Chung et al. [28]	2014	Survey of advancements in RNN training, optimization, and architectures.
Goodfellow et al. [22]	2016	Review on deep learning, including RNNs.
Greff et al. [23]	2016	Extensive comparison of LSTM variants.
Tarwani et al. [19]	2017	In-depth analysis of RNNs in NLP.
Chen et al. [32]	2018	Effectiveness of RNNs in environmental monitoring and climate modeling.
Bai et al. [26]	2018	Comparison of RNNs with other sequence modeling techniques like CNNs and attention mechanism.
Che et al. [27]	2018	Potential of RNNs in medical applications.
Zhang et al. [34]	2020	RNN applications in robotics, including path planning, motion control, and human-robot interaction.
Dutta et al. [20]	2022	Overview of RNNs, challenges in training, and advancements in LSTM and GRU for sequence learning.
Linardos et al. [33]	2022	RNNs for early warning systems, disaster response, and recovery planning in natural disaster prediction.
Badawy et al. [29]	2023	Integration of RNNs with other ML techniques for predictive analytics and patient monitoring in healthcare.
Ismaeel et al. [30]	2023	Application of RNNs in smart city technologies, including traffic prediction, energy management, and urban planning.
Mers et al. [31]	2023	Performance comparison of various RNN models in pavement performance forecasting.
Quradaa et al. [21]	2024	Start-of-the-art review of RNNs, covering core architectures with a focus on applications in code clones.
Al-Selwi et al. [24]	2024	Review of LSTM applications from 2018-2023.

 doi:10.20944/preprints202408.0748.v1

This research addresses the limitations in the existing literature by providing a more comprehensive review that includes the latest developments in RNN architectures, such as hybrid models and neural architecture search, as well as their applications across a wider range of domains. Additionally, this review contributes to a more holistic understanding of the current state and future directions of RNN research by integrating discussions on scalability, robustness, and interoperability.

3. Fundamentals of RNNs

3.1. Basic Architecture and Working Principle of Standard RNNs

RNNs are designed to process sequential data by maintaining a hidden state that captures information about previous inputs [35]. The basic architecture consists of an input layer, a hidden layer, and an output layer. Unlike feedforward neural networks, RNNs have recurrent connections, as shown in Figure 1, allowing information to cycle within the network. At each time step t, the RNN takes an input vector $\mathbf{x}_t$ and updates its hidden state $\mathbf{h}_t$ using the following equation:

$$\mathbf{h}_t = \sigma_h(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h), \tag{1}$$

where $\mathbf{W}_{xh}$ is the weight matrix between the input and hidden layer, $\mathbf{W}_{hh}$ is the weight matrix for the recurrent connection, $\mathbf{b}_h$ is the bias vector, and σ_h is the activation function, typically the hyperbolic tangent function (tanh) or the rectified linear unit [36]. The output at each time step t is given by:

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y), \tag{2}$$

where $\mathbf{W}_{hy}$ is the weight matrix between the hidden and output layers, $\mathbf{b}_y$ is the bias vector, and σ_y is the activation function for the output layer.

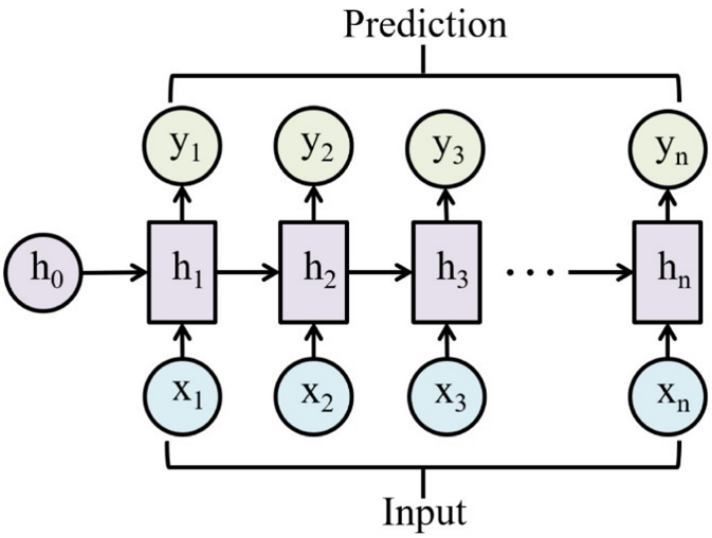

Figure 1. Basic RNN architecture

3.2. Activation Functions

The core of RNN operations involves the recurrent computation of the hidden state, which integrates the current input with the previous hidden state [37]. This recurrent computation allows RNNs to exhibit dynamic temporal behavior. The choice of activation function σ_h plays a crucial role in the behavior of the network, introducing non-linearity that enables the network to learn and represent complex patterns in the data [38,39]. One commonly used activation function in RNNs is the Hyperbolic Tangent (tanh). The tanh function squashes the input values to the range $[-1, 1]$, making it zero-centered and suitable for modeling sequences with both positive and negative values [40]. The tanh is represented mathematically as:

 doi:10.20944/preprints202408.0748.v1

$$\sigma_h(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{3}$$

Another widely used activation function is the Rectified Linear Unit (ReLU). The ReLU function outputs the input directly if it is positive; otherwise, it outputs zero [41]. This simplicity helps mitigate the vanishing gradient problem to some extent by allowing gradients to flow through the network more effectively. Furthermore, Leaky ReLU is a variant of ReLU designed to address the "dying ReLU" problem, where neurons can become inactive and stop learning [42]. Leaky ReLU allows a small, non-zero gradient when the input is negative, thus keeping the neurons active during the training process. Additionally, the Exponential Linear Unit (ELU) is another variant designed to bring the mean activation closer to zero, which speeds up learning by reducing bias shifts [43]. ELU tends to improve learning characteristics over ReLU by allowing the activations to take on negative values when the input is negative. These activation functions are represented as follows:

$$\sigma_h(z) = \max(0, z) \tag{4}$$

$$\sigma_h(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha z & \text{otherwise} \end{cases} \tag{5}$$

$$\sigma_h(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha(e^z - 1) & \text{otherwise} \end{cases} \tag{6}$$

where α is a small constant, typically 0.01. Meanwhile, the Sigmoid function squashes the input values to the range $[0, 1]$. It is similar to tanh but outputs values in a different range, making it useful for problems where the output needs to be interpreted as probabilities [44,45]. Similarly, the Softmax function is commonly used in the output layer of classification networks to convert raw scores into probabilities [46]. It is particularly useful in multi-class classification problems. The sigmoid and softmax functions are represented mathematically as:

$$\sigma_h(z) = \frac{1}{1 + e^{-z}} \tag{7}$$

$$\sigma_h(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{8}$$

where z_i is the i-th element of the input vector z. Each of these activation functions has its advantages and is chosen based on the specific requirements of the task at hand. Meanwhile, the hidden state update in RNNs can be seen as a function $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$, which captures the dependencies between the input sequence and the recurrent connections. The choice of σ_h significantly affects how well the network learns these dependencies and generalizes to new data.

3.3. The Vanishing Gradient Problem

The vanishing gradient problem is a major challenge in training RNNs. During the training process, the BPTT algorithm is used to compute the gradients of the loss function with respect to the weights [47]. However, as the gradients are propagated backwards in time, they can diminish exponentially, making it difficult for the network to learn long-term dependencies. Mathematically, the hidden state at time step t can be expanded as:

$$\mathbf{h}_t = \sigma_h(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\sigma_h(\mathbf{W}_{xh}\mathbf{x}_{t-1} + \mathbf{W}_{hh}\mathbf{h}_{t-2} + \mathbf{b}_h) + \mathbf{b}_h). \tag{9}$$

When calculating the gradients, we encounter terms involving the product of many Jacobian matrices:

 doi:10.20944/preprints202408.0748.v1

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-n}} = \prod_{k=t-n}^{t-1} \mathbf{J}_k, \tag{10}$$

where $\mathbf{J}_k$ is the Jacobian matrix of the hidden state at time step k. If the eigenvalues of $\mathbf{J}_k$ are less than 1, the product of these matrices will tend to zero as n increases, leading to vanishing gradients [48,49]. To mitigate this problem, various RNN variants have been developed, such as LSTM networks and GRUs. These architectures introduce gating mechanisms that control the flow of information and gradients through the network, allowing for better learning of long-term dependencies.

3.4. Bidirectional RNNs

Bidirectional RNNs (BiRNNs) enhance the architecture by processing the sequence in both forward and backward directions. This allows the network to have access to future context as well as past context, improving its performance on tasks where understanding both the preceding and succeeding elements is crucial [37,50]. In BiRNNs, two hidden states are maintained: one for the forward pass ($\overrightarrow{\mathbf{h}}_t$) and one for the backward pass ($\overleftarrow{\mathbf{h}}_t$):

$$\overrightarrow{\mathbf{h}}_t = \sigma_h(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\overrightarrow{\mathbf{h}}_{t-1} + \mathbf{b}_h), \tag{11}$$

$$\overleftarrow{\mathbf{h}}_t = \sigma_h(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\overleftarrow{\mathbf{h}}_{t+1} + \mathbf{b}_h). \tag{12}$$

The output $\mathbf{y}_t$ is then computed by concatenating the forward and backward hidden states:

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_{hy}[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t] + \mathbf{b}_y), \tag{13}$$

where $[;]$ denotes concatenation. Furthermore, BiRNNs are effective for tasks such as named entity recognition, machine translation, and speech recognition, where context from both directions improves the model's performance [51,52]. Through accessing information from both the past and future, BiRNNs can provide a more comprehensive understanding of the input sequence. For instance, in language modeling, understanding the surrounding words can significantly enhance the accuracy of predicting the next word [53,54]. In machine translation, knowing the entire sentence allows the network to translate words more accurately, considering the entire context. Additionally, BiRNNs are also used in various time series applications, such as stock price prediction and medical diagnosis, where understanding the temporal dependencies in both directions is beneficial [55]. However, BiRNNs require more computational resources than unidirectional RNNs due to the need to process the sequence twice (forward and backwards) [50,56].

3.5. Deep RNNs

Deep RNNs extend the basic architecture by stacking multiple RNN layers on top of each other, which allows the network to learn more complex representations [57]. Each layer's hidden state serves as the input to the subsequent layer, enhancing the model's capacity to capture hierarchical features. For a deep RNN with L layers, the hidden states at layer l and time step t are updated as follows:

$$\mathbf{h}_t^{(l)} = \sigma_h(\mathbf{W}_{xh}^{(l)}\mathbf{h}_t^{(l-1)} + \mathbf{W}_{hh}^{(l)}\mathbf{h}_{t-1}^{(l)} + \mathbf{b}_h^{(l)}), \tag{14}$$

where $\mathbf{h}_t^{(0)} = \mathbf{x}_t$ represents the input at the first layer. The output at the topmost layer is then computed using the same procedure as in basic RNNs:

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_{hy}\mathbf{h}_t^{(L)} + \mathbf{b}_y). \tag{15}$$

Deep RNNs can model more complex sequences and capture longer dependencies than shallow RNNs [58]. However, they are also more prone to the vanishing gradient problem, which can be mitigated by using advanced variants like LSTMs or GRUs. Deep RNNs have been successfully applied

 doi:10.20944/preprints202408.0748.v1

in various domains, including natural language processing, speech recognition, and video analysis. In NLP, deep RNNs can model complex linguistic structures and capture long-range dependencies, improving tasks such as machine translation and text generation. However, training deep RNNs can be challenging due to the increased complexity and the risk of overfitting [59,60]. Techniques such as dropout, layer normalization, and residual connections are often employed to improve the training process and generalization of deep RNNs [61–63]. Dropout helps prevent overfitting by randomly setting a fraction of the units to zero during training, which encourages the network to learn more robust features, while batch normalization helps stabilize and accelerate training by normalizing the inputs to each layer [64]. Residual connections, which add shortcut connections that bypass one or more layers, help mitigate the vanishing gradient problem in very deep networks [65].

4. Advanced Variants of RNNs

4.1. Long Short-Term Memory Networks

LSTMs were introduced by Hochreiter and Schmidhuber [12] to address the vanishing gradient problem inherent in basic RNNs. The key innovation in LSTMs is the use of gating mechanisms, shown in Figure 2, to control the flow of information through the network. This allows LSTMs to maintain and update their internal state over long periods, making them effective for tasks requiring the modeling of long-term dependencies. Each LSTM cell contains three gates: the input gate, forget gate, and output gate, which regulate the cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$ [66]. These gates determine how much of the input to consider, how much of the previous state to forget, and how much of the cell state to output. The LSTM update equations are as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i), \tag{16}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f), \tag{17}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o), \tag{18}$$

$$\mathbf{g}_t = \tanh(\mathbf{W}_{xg}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g), \tag{19}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t, \tag{20}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{21}$$

where $\mathbf{i}_t$ is the input gate, $\mathbf{f}_t$ is the forget gate, $\mathbf{o}_t$ is the output gate, $\mathbf{g}_t$ is the cell input, $\mathbf{c}_t$ is the cell state, $\mathbf{h}_t$ is the hidden state, σ represents the sigmoid function, tanh is the hyperbolic tangent function, and $\odot$ denotes element-wise multiplication [66]. Furthermore, the input gate $\mathbf{i}_t$ controls how much of the new input $\mathbf{x}_t$ is written to the cell state $\mathbf{c}_t$. The forget gate $\mathbf{f}_t$ decides how much of the previous cell state $\mathbf{c}_{t-1}$ should be retained. The output gate $\mathbf{o}_t$ determines how much of the cell state $\mathbf{c}_t$ is used to compute the hidden state $\mathbf{h}_t$. The cell input $\mathbf{g}_t$ is a candidate value that is added to the cell state after being modulated by the input gate. The use of these gating mechanisms allows LSTMs to selectively remember or forget information, enabling them to handle long-term dependencies more effectively than traditional RNNs. Other LSTM variants include Bidirectional LSTMs (BiLSTM) and Stacked LSTMs.

 doi:10.20944/preprints202408.0748.v1

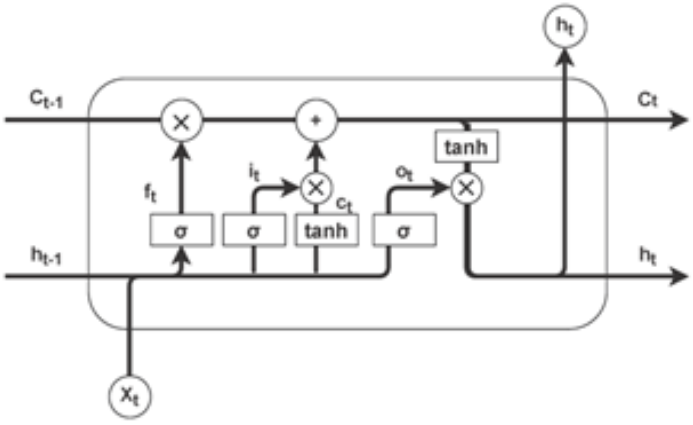

Figure 2. Architecture of the LSTM network. [36]

4.2. Bidirectional LSTMs

Bidirectional LSTMs, shown in Figure 3, extend the standard LSTM architecture by processing the sequence in both forward and backward directions, similar to BiRNNs [67]. This approach allows the network to capture context from both the past and the future, enhancing its ability to understand dependencies in the sequence more comprehensively. In BiLSTMs, two separate hidden states are maintained for each time step: one for the forward pass ($\overrightarrow{\mathbf{h}}_t$) and one for the backward pass ($\overleftarrow{\mathbf{h}}_t$). These hidden states are computed as described in Equations 11 and 12.

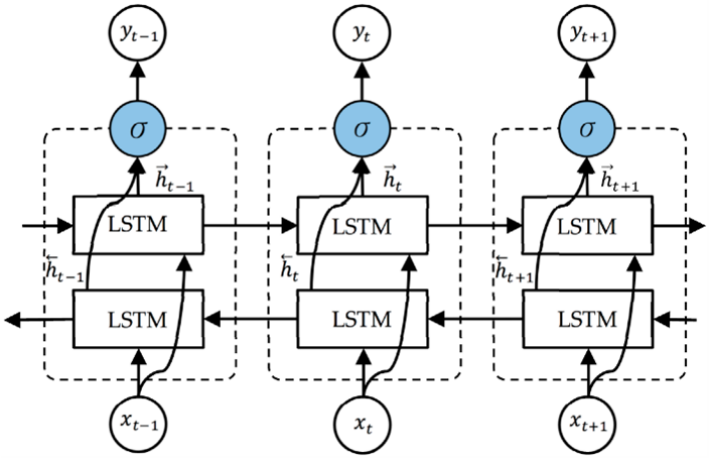

Figure 3. Architecture of BiLSTM network [36]

4.2.1. Stacked LSTMs

Stacked LSTMs involve stacking multiple LSTM layers, where the output of one LSTM layer serves as the input to the next, as shown in Figure 4. This deep architecture allows the network to capture more complex patterns and dependencies in the data by learning hierarchical representations at different levels of abstraction. For a stacked LSTM with L layers, the hidden states at layer l and time step t are updated as described in Equation 14 and 15.

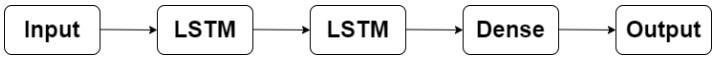

Figure 4. A stacked LSTM [36]

Stacking LSTM layers allows the network to learn increasingly complex features and representations. The lower layers can capture local patterns and short-term dependencies, while the higher layers can capture more abstract features and long-term dependencies [18]. This hierarchical learning

 doi:10.20944/preprints202408.0748.v1

is advantageous for tasks such as language modeling, where different levels of syntactic and semantic information need to be captured, or for video analysis, where temporal dependencies at different time scales must be understood. While stacked LSTMs offer improved modeling capabilities, they also come with increased computational complexity and a higher risk of overfitting.

4.3. Gated Recurrent Units

Gated Recurrent Units are another variant designed to address the vanishing gradient problem while simplifying the LSTM architecture. Introduced by Cho et al. [13], GRUs combine the forget and input gates into a single update gate and merge the cell state and hidden state, reducing the number of gates and parameters, thus simplifying the model and making it computationally more efficient. The GRU architecture consists of two gates: the update gate $\mathbf{z}_t$ and the reset gate $\mathbf{r}_t$ [68]. Figure 5 shows the GRU architecture. The gates control the flow of information to ensure that relevant information is retained and irrelevant information is discarded. The update equations for GRUs are:

$$\mathbf{z}_t = \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z), \tag{22}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r), \tag{23}$$

$$\mathbf{h}'_t = \tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{r}_t \odot (\mathbf{W}_{hh}\mathbf{h}_{t-1}) + \mathbf{b}_h), \tag{24}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \mathbf{h}'_t, \tag{25}$$

where $\mathbf{z}_t$ is the update gate, $\mathbf{r}_t$ is the reset gate, and $\mathbf{h}'_t$ is the candidate hidden state. The update gate $\mathbf{z}_t$ determines how much of the previous hidden state $\mathbf{h}_{t-1}$ should be carried forward to the current hidden state $\mathbf{h}_t$, while the reset gate $\mathbf{r}_t$ controls how much of the previous hidden state to forget [66]. The candidate hidden state $\mathbf{h}'_t$ represents the new content to be added to the current hidden state, modulated by the reset gate. Furthermore, the simplified architecture of GRUs allows them to be computationally more efficient than LSTMs while still addressing the vanishing gradient problem. This efficiency makes GRUs to be well-suited for tasks where computational resources are limited or when training needs to be faster. GRUs have been successfully applied in various sequence modeling tasks. Their ability to capture long-term dependencies with fewer parameters makes them a popular choice in many applications. Additionally, studies have shown that GRUs can achieve performance comparable to LSTMs [69–71], making them an attractive alternative for many use cases.

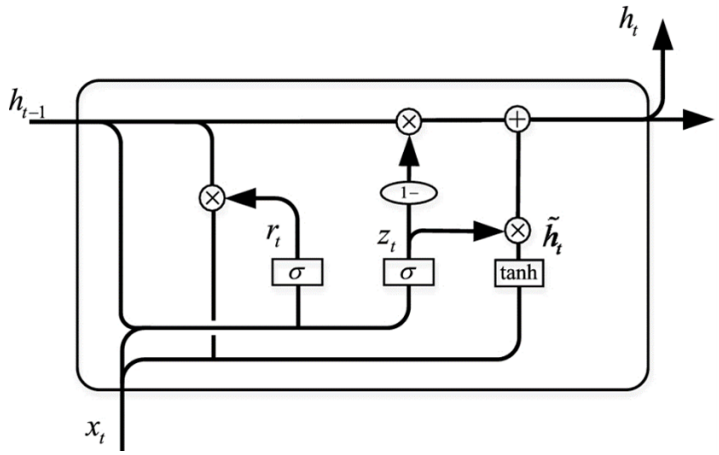

Figure 5. Architecture of the GRU network

4.3.1. Comparison with LSTMs

GRUs have fewer parameters compared to LSTMs due to the absence of a separate cell state and combined gating mechanisms [72]. This often leads to faster training times and comparable performance to LSTMs on many tasks. However, despite their advantages, the choice between GRUs

 doi:10.20944/preprints202408.0748.v1

and LSTMs often depends on the specific task and dataset. Some tasks may benefit more from the additional complexity and gating mechanisms of LSTMs, while others may perform equally well with the simpler GRU architecture.

4.4. Other Notable Variants

While LSTMs and GRUs are the most widely used RNN variants, other architectures like Peephole LSTMs, Echo State Networks, and Independently Recurrent Neural Networks offer unique advantages for specific applications.

4.4.1. Peephole LSTMs

Peephole LSTMs, introduced by Gers and Schmidhuber [73], enhance standard LSTMs by allowing the gates to have access to the cell state through peephole connections. This additional connection enables the LSTM to better regulate the gating mechanisms based on the current cell state, improving timing decisions in applications such as speech recognition and financial time series prediction [74]. In the following equations, the input gate ($\mathbf{i}_t$), forget gate ($\mathbf{f}_t$), and output gate ($\mathbf{o}_t$) are enhanced with peephole connections:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i), \tag{26}$$

where $\mathbf{i}_t$ is the input gate, $\mathbf{W}_{ci}$ is the peephole weight connecting the cell state $\mathbf{c}_{t-1}$ to the input gate.

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f), \tag{27}$$

where $\mathbf{f}_t$ is the forget gate, $\mathbf{W}_{cf}$ is the peephole weight connecting the cell state $\mathbf{c}_{t-1}$ to the forget gate.

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o), \tag{28}$$

where $\mathbf{o}_t$ is the output gate, $\mathbf{W}_{co}$ is the peephole weight connecting the cell state $\mathbf{c}_t$ to the output gate.

4.4.2. Echo State Networks

Echo State Networks (ESNs), proposed by Jaeger [75], is a type of RNN where the hidden layer is fixed and randomly connected, and only the output layer is trained. This architecture simplifies the training process, which makes ESNs suitable for real-time signal processing and adaptive control systems. The state update and output computation are achieved as follows:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{W}_{res}\mathbf{h}_{t-1}), \tag{29}$$

where $\mathbf{h}_t$ is the hidden state, $\mathbf{W}_{in}$ is the input weight matrix, and $\mathbf{W}_{res}$ is the fixed, randomly initialized reservoir weight matrix. Assuming $\mathbf{W}_{out}$ is the trained output weight matrix, the output of the network can be represented as:

$$\mathbf{y}_t = \mathbf{W}_{out}\mathbf{h}_t, \tag{30}$$

4.4.3. Independently Recurrent Neural Network

Most recently, Independently Recurrent Neural Networks (IndRNNs), proposed by Li et al. [76], use independent recurrent units to address the gradient vanishing and exploding problems, making it easier to train very deep RNNs. This architecture is useful for long sequence tasks such as video sequence analysis and long text generation. Assuming $\mathbf{h}_t$ is the hidden state, $\mathbf{W}_{xh}$ is the input weight matrix, and $\mathbf{u}$ is a vector of recurrent weights. The state update equation for IndRNN is:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{u} \odot \mathbf{h}_{t-1}), \tag{31}$$

 doi:10.20944/preprints202408.0748.v1

5. Innovations in RNN Architectures and Training Methodologies

In recent years, there has been significant innovations in RNN architectures and training methodologies aimed at enhancing performance and addressing existing limitations.

5.1. Hybrid Architectures

Combining RNNs with other neural network architectures has led to hybrid models that leverage the strengths of each component. For example, integrating Convolutional Neural Networks (CNNs) with RNNs has proven effective in video analysis, where CNNs handle spatial features while RNNs capture temporal dynamics [77]. This approach allows the model to process both spatial and temporal information, enhancing its ability to recognize patterns and make predictions. Furthermore, incorporating attention mechanisms into RNNs has also improved their ability to model long-range dependencies. Attention mechanisms enable the network to focus on relevant parts of the input sequence, which is useful in tasks such as machine translation and text summarization. The attention mechanism can be described as follows:

$$\mathbf{a}_t = \text{softmax}(\mathbf{u}_t), \tag{32}$$

$$\mathbf{c}_t = \sum_{i=1}^{T} \mathbf{a}_{t,i} \mathbf{h}_i, \tag{33}$$

where $\mathbf{a}_t$ is the attention weight, $\mathbf{u}_t$ is the score function, and $\mathbf{c}_t$ is the context vector.

5.2. Neural Architecture Search

Neural architecture search (NAS) has automated the design of RNN architectures, enabling the discovery of more efficient and powerful models [78,79]. NAS techniques, such as those pioneered by Zoph and Le [80], explore various combinations of layers, activation functions, and hyperparameters to find optimal configurations that outperform manually designed architectures. The NAS process can be formulated as an optimization problem:

$$\mathcal{A}^* = \arg\max_{\mathcal{A} \in \mathcal{S}} \text{Accuracy}(\mathcal{A}), \tag{34}$$

where $\mathcal{A}$ represents an architecture, $\mathcal{S}$ is the search space, and $\mathcal{A}^*$ is the optimal architecture.

5.3. Advanced Optimization Techniques

Advanced optimization techniques have been developed to improve the training efficiency and stability of RNNs. Gradient clipping is a technique used to prevent the gradients from becoming too large, which can destabilize training [81,82].

$$\mathbf{g} \leftarrow \frac{\mathbf{g}}{\max(1, \frac{\|\mathbf{g}\|}{\tau})}, \tag{35}$$

where $\mathbf{g}$ is the gradient and τ is the threshold value. Furthermore, adaptive learning rates, such as those used in the Adam optimizer, adjust the learning rate during training to accelerate convergence and improve performance [83]. The Adam optimizer updates the parameters using:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t, \tag{36}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2, \tag{37}$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}, \tag{38}$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}, \tag{39}$$

 doi:10.20944/preprints202408.0748.v1

where $\mathbf{m}_t$ and $\mathbf{v}_t$ are the first and second moment estimates, β_1 and β_2 are the decay rates, α is the learning rate, and ϵ is a small constant [83]. Also, second-order optimization methods, such as the Hessian-Free optimizer, have also been explored to improve the convergence speed and stability of training deep networks.

5.4. RNNs with Attention Mechanisms

Integrating attention mechanisms into RNNs allows these networks to selectively focus on important parts of the input sequence, addressing the limitations of traditional RNNs in modeling long-term dependencies [84–86]. This hybrid approach combines the strengths of RNNs and attention mechanisms, enhancing their capability to handle complex sequence tasks. Attention-enhanced RNNs have shown significant improvements in tasks such as speech recognition and text summarization. For example, Bahdanau et al. [87] demonstrated the use of attention mechanisms in neural machine translation, which allowed RNNs to focus on relevant words in the source sentence, improving translation accuracy. Similarly, Luong et al. [88] proposed global and local attention mechanisms, further enhancing the performance of RNNs in various sequence-to-sequence tasks.

5.5. RNNs Integrated with Transformer Models

Transformers, introduced by Vaswani et al. [89] in 2017, employ self-attention mechanisms and have proven to be highly effective in capturing long-range dependencies. Unlike RNNs, transformers process sequences in parallel, which can lead to better performance on long sequences. The self-attention mechanism is defined as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{40}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices, respectively, and d_k is the dimension of the keys. Considering both Transformer and RNN architecture have limitations, studies have integrated both methods to obtain robust models, as shown in recent literature [**? ?**]. Therefore researchers can develop more powerful and efficient models for a wide range of applications by leveraging the sequential processing capabilities of RNNs and the parallel, attention-based mechanisms of transformers. This integrated approach addresses the limitations of each architecture and enhances the overall performance in sequence modeling tasks.

6. Applications of RNNs in Peer-Reviewed Literature

RNNs and their variants have been extensively studied and applied across various domains in the peer-reviewed literature. This section provides a comprehensive review of these applications.

6.1. Natural Language Processing

RNNs have transformed the field of NLP by enabling more sophisticated and context-aware models. Several studies have demonstrated the effectiveness of RNNs in various NLP tasks:

6.1.1. Text Generation

RNNs have been used extensively for text-generation tasks. Souri et al. [90] demonstrated the use of RNNs to generate coherent and contextually relevant Arabic text. Their model was trained on a large corpus of text data, allowing it to learn the probability distribution of word sequences, which proved effective in generating human-like text. Meanwhile, several researchers have proposed novel approaches to enhance the performance of RNNs in text generation. For instance, Islam [91] introduced a sequence-to-sequence framework that improved the generation quality using LSTM. This method allowed the network to handle longer sequences and maintain coherence over extended text.

Gajendran et al. [92] demonstrated the effectiveness of RNNs in generating character-level text. Their work showed that BiLSTMs could capture a wide range of patterns, from character-level depen-

 doi:10.20944/preprints202408.0748.v1

dencies to higher-level syntactic structures, making them versatile for different text generation tasks, including the generation of code, literature, and poetry. More recently, advancements in RNN-based text generation have focused on improving the diversity and coherence of generated text. Hu et al. [93] proposed the use of Variational Autoencoders (VAEs) combined with RNNs to enhance the creativity of text generation. Their approach enabled the generation of diverse and contextually rich sentences by learning a latent space representation of the text.

Meanwhile, Holtzman et al. [94] introduced the concept of "controlled text generation" using RNNs, which allowed users to influence the style and content of the generated text. This method provided more flexibility and control over the text generation process, making it useful for applications such as creative writing and personalized content generation. Additionally, with the advent of more sophisticated models like Transformers, RNN-based text generation has evolved to incorporate attention mechanisms.

Yin et al. [95] proposed an approach combining RNN with attention mechanism, which allows the model to focus on relevant parts of the input sequence during the generation process. This significantly improved the quality and coherence of the generated text by dynamically adjusting the focus of the model based on the context. Hussein and Savas [96] employed LSTM for text generation. Similarly, Baskaran et al. [97] employed LSTM for text generation, achieving excellent performance. These studies showed that LSTM networks are capable of generating texts that are contextually relevant and linguistically accurate.

Furthermore, studies have continued to explore and enhance the capabilities of RNNs in text generation. Keskar et al. [98] introduced a large-scale language model known as Conditional Transformer Language (CTRL), which can be conditioned on specific control codes to generate text in various styles and domains. This work highlights the growing trend of combining RNNs with transformer architectures to leverage their strengths in sequence modeling and text generation. Additionally, Guo [99] explored the integration of reinforcement learning with RNNs for text generation. The approach aimed to optimize the generation process by rewarding the model for producing high-quality, contextually appropriate text, thereby improving both the coherence and relevance of the generated content.

6.1.2. Sentiment Analysis

In sentiment analysis, RNNs have been shown to outperform traditional models by capturing the context and details of sentiment expressed in text. Yadav et al. [100] used LSTM-based models to analyze customer reviews and social media posts, achieving notable improvements in accuracy over conventional methods. Building on this, Abimbola et al. [101] proposed a hybrid LSTM-CNN model for document-level sentiment classification, which first captures the sentiment of individual sentences and then aggregates them to determine the overall sentiment of the document. This hierarchical approach allows for a more detailed understanding of sentiment, especially in long and complex texts. Zulqarnain et al. [102] utilized the concept of attention mechanisms and GRU to enhance sentiment analysis. By allowing the model to focus on specific parts of the input text that are most indicative of sentiment, attention mechanisms significantly improved the interpretability and performance of sentiment analysis models. This advancement enabled the models to highlight which words or phrases contribute most to the sentiment prediction.

Additionally, several studies have explored the integration of RNNs with CNNs to leverage the strengths of both architectures. For instance, Pujari et al. [103] combined CNNs and RNNs to capture both local features and long-range dependencies in text, resulting in a hybrid model that achieved state-of-the-art performance in sentiment classification tasks. Meanwhile Wankhade et al. [104] employed the fusion of CNN and BiLSTM with attention mechanism, leading to enhanced sentiment classification. Furthermore, Sangeetha and Kumaran [105] utilized BiLSTMs to enhance the sentiment analysis capability by processing text in both forward and backward directions. This approach captures the context from both past and future words, providing a more comprehensive understanding of the sentiment expressed in the text.

 doi:10.20944/preprints202408.0748.v1

In addition to these architectural innovations, there has been a focus on improving the robustness of RNN-based sentiment analysis models. For example, He and McAuley [106] developed an adversarial training framework that enhances the model's ability to handle noisy and adversarial text inputs, thereby improving its generalization to real-world data. Also, the use of transfer learning and pre-trained language models, such as BERT and GPT, has been increasingly popular in sentiment analysis [107–109]. These models, fine-tuned for sentiment classification tasks, have demonstrated exceptional performance by leveraging large-scale pre-training on diverse text corpora and then adapting to specific sentiment analysis datasets.

6.1.3. Machine Translation

To address the challenge of translating long sentences, Wu et al. [110] introduced the concept of deep RNNs with multiple layers in both the encoder and decoder. Their model, known as Google Neural Machine Translation (GNMT), improved translation accuracy and fluency by capturing more complex patterns and dependencies within the text. GNMT became a major achievement in neural machine translation, setting a new benchmark for translation systems. Sennrich et al. [111] presented a method for incorporating subword units into RNN-based translation models. This approach, known as Byte-Pair Encoding (BPE), enabled the translation models to handle rare and out-of-vocabulary words more effectively by splitting them into smaller, more frequent subword units. This method improved the robustness and generalization of the translation models.

With the advent of transformer models, Vaswani et al. [89] revolutionized the field of machine translation by introducing a fully attention-based architecture that eliminated the need for recurrence entirely. Transformers demonstrated superior performance in translation tasks by allowing for parallel processing of sequences and capturing long-range dependencies more efficiently. Despite this shift, RNN-based models with attention mechanisms continued to be relevant, particularly in scenarios where computational resources were limited or sequential processing was preferred. For example, Kang et al. [112] combined RNN with attention mechanism to obtain a bilingual attention-based machine translation model. While Zulqarnain et al. [102] utilized GRY in a multi-stage feature attention mechanism model.

Several studies have also combined RNNs with transformer models to utilise the strengths of both architectures. For instance, Yang et al. [113] proposed a hybrid model that integrates RNNs into the transformer architecture to enhance its ability to capture sequential dependencies while maintaining the efficiency of parallel processing. This hybrid approach achieved state-of-the-art performance in several translation benchmarks. Meanwhile, more recent studies have explored the integration of pre-trained language models like BERT and GPT into machine translation systems. Song et al. [114] demonstrated that incorporating BERT into the encoder of a translation model enhanced its understanding of the source language, leading to more accurate and fluent translations. Table 2 summarizes the discussed applications of RNNs in natural language processing.

 doi:10.20944/preprints202408.0748.v1

Table 2. Summary of Applications of RNNs in Natural Language Processing

Application Domain	Reference	Year	Methods and Application
Text Generation	Souri et al. [90]	2018	RNNs for generating coherent and contextually relevant Arabic text.
	Holtzman et al. [94]	2019	Controlled text generation using RNNs for style and content control.
	Hu et al. [93]	2020	VAEs combined with RNNs to enhance creativity in text generation.
	Gajendran et al. [92]	2020	Character-level text generation using BiLSTMs for various tasks.
	Hussein and Savas [96]	2024	LSTM for text generation.
	Baskaran et al. [97]	2024	LSTM for text generation, achieving excellent performance.
	Islam [91]	2019	Sequence-to-sequence framework using LSTM for improved text generation quality.
	Yin et al. [95]	2018	Attention mechanisms with RNNs for improved text generation quality.
	Guo [99]	2015	Integration of reinforcement learning with RNNs for text generation.
	Keskar et al. [98]	2019	Conditional Transformer Language (CTRL) for generating text in various styles.
Sentiment Analysis	He and McAuley [106]	2016	Adversarial training framework for robustness in sentiment analysis.
	Pujari et al. [103]	2024	Hybrid CNN-RNN model for sentiment classification.
	Wankhade et al. [104]	2024	Fusion of CNN and BiLSTM with attention mechanism for sentiment classification.
	Sangeetha and Kumaran [105]	2023	BiLSTMs for sentiment analysis by processing text in both directions.
	Yadav et al. [100]	2023	LSTM-based models for sentiment analysis in customer reviews and social media posts.
	Zulqarnain et al. [102]	2024	Attention mechanisms and GRU for enhanced sentiment analysis.
	Samir et al. [107]	2021	Use of pre-trained models like BERT for sentiment analysis.
	Prottasha et al. [108]	2022	Transfer learning with BERT and GPT for sentiment analysis.
	Abimbola et al. [101]	2024	Hybrid LSTM-CNN model for document-level sentiment classification.
	Mujahid et al. [109]	2023	Analyzing sentiment with pre-trained models fine-tuned for specific tasks.
Machine Translation	Sennrich et al. [111]	2015	Byte-Pair Encoding (BPE) for handling rare words in translation models.
	Wu et al. [110]	2016	Google Neural Machine Translation (GNMT) with deep RNNs for improved accuracy.
	Vaswani et al. [89]	2017	Fully attention-based transformer models for superior translation performance.
	Yang et al. [113]	2017	Hybrid model integrating RNNs into the transformer architecture.
	Song et al. [114]	2019	Incorporating BERT into translation models for enhanced understanding and fluency.
	Kang et al. [112]	2023	Bilingual attention-based machine translation model combining RNN with attention.
	Zulqarnain et al. [102]	2024	Multi-stage feature attention mechanism model using GRU.

 doi:10.20944/preprints202408.0748.v1

6.2. Speech Recognition

RNNs have also made significant contributions to the field of speech recognition, leading to more accurate and efficient systems. Hinton et al. [115] explored the use of deep neural networks, including RNNs, for speech-to-text systems. Their research showed that RNNs could capture the temporal dependencies in speech signals, leading to significant improvements in transcription accuracy compared to previous methods.

Hannun et al. [116] introduced DeepSpeech, a state-of-the-art speech recognition system based on RNNs. DeepSpeech employed a deep LSTM network trained on a vast amount of labeled speech data, theyreby improving transcription accuracy. This system was designed to handle noisy environments and diverse accents, making it robust for various real-world applications. Similarly, Amodei et al. [117] presented DeepSpeech2, which extended the capabilities of the original DeepSpeech model by incorporating bidirectional RNNs and a more extensive dataset. DeepSpeech2 achieved notable performance improvements, demonstrating that RNNs could effectively handle variations in speech patterns and accents.

Meanwhile, Chiu et al. [118] proposed the use of RNN-transducer (RNN-T) models for end-to-end speech recognition. RNN-T models integrate both acoustic and language models into a single RNN framework, allowing for more efficient and accurate transcription. This integration reduced the complexity and latency of real-time speech recognition systems, making them more practical for deployment in real-world applications. Furthermore, Zhang et al. [119] proposed the use of convolutional recurrent neural networks (CRNNs) for speech recognition. CRNNs combine the strengths of CNNs for feature extraction and RNNs for sequence modeling, resulting in a hybrid architecture that is robust in both accuracy and computational efficiency. Specifically, this model was effective in handling long audio sequences and varying speech rates.

Recently, Dong et al. [120] introduced the Speech-Transformer, a model that leverages the self-attention mechanism to process audio sequences in parallel, improving both speed and accuracy. This model demonstrated that transformer-based architectures could effectively handle the sequential nature of speech data, providing a competitive alternative to traditional RNN-based models. Bhaskar and Thasleema [121] developed a speech recognition model using LSTM. The model achieved visual speech recognition using facial expressions. Other studies that explored the use of different RNN variants in speech recognition include [122–125]

6.3. Financial Time Series Forecasting

RNNs have been extensively used in time series prediction due to their ability to model temporal dependencies and trends in sequential data. Fischer and Krauss [126] conducted a comprehensive study using deep RNNs to predict stock returns. Their results indicated that RNNs could outperform traditional ML models, such as support vector machines and random forests, in financial forecasting tasks. The study demonstrated that deep RNNs could learn intricate patterns in stock price movements, contributing to better forecasting accuracy.

With the advancement of deep learning techniques, Nelson et al. [127] proposed a model combining CNNs and RNNs for stock price prediction. The CNN component extracted local features from historical price data, while the RNN component captured the temporal dependencies. This hybrid model showed significant improvements in prediction performance, suggesting that integrating different neural network architectures could enhance financial forecasting. Also, attention mechanisms have been integrated into RNNs to improve financial forecasting.

Luo et al. [128] used an attention-based CNN-BiLSTM model that focused on relevant time steps in the input sequence, enhancing the model's ability to capture important patterns in financial data. This approach allowed for more accurate predictions of stock prices and market trends by dynamically weighting the significance of past observations. Furthermore, Bao et al. [129] employed a novel deep learning framework combining LSTMs with stacked autoencoders for financial time series forecasting. Their model utilized stacked autoencoders to reduce the dimensionality of input data and LSTMs

 doi:10.20944/preprints202408.0748.v1

to model temporal dependencies. This method improved the model's ability to predict future stock prices by effectively capturing both feature representations and sequential patterns.

Feng et al. [130] explored the use of transfer learning for financial forecasting. They proposed a model that pre-trained an RNN on a large corpus of financial data and fine-tuned it on specific stock datasets. This approach employed the knowledge gained from broader market data to improve predictions on individual stocks, which demonstrates the potential of transfer learning in financial forecasting. Meanwhile, the application of reinforcement learning in conjunction with RNNs has gained attention in financial forecasting. Rundo [131] combined RL with LSTMs to develop a trading strategy that maximizes returns. Their model learned optimal trading actions through interactions with the market environment, resulting in a robust and adaptive financial forecasting system.

Table 3 provides a summary of the RNN applications in both speech recognition and financial time series forecasting.

Table 3. Summary of RNNs in Speech Recognition and Financial Time Series Forecasting

Application Domain	Reference	Year	Methods and Application
Speech Recognition	Hinton et al. [115]	2012	Deep neural networks, including RNNs, for speech-to-text systems.
	Hannun et al. [116]	2014	DeepSpeech: LSTM-based speech recognition system.
	Amodei et al. [117]	2016	DeepSpeech2: Enhanced LSTM-based speech recognition with bidirectional RNNs.
	Zhang et al. [119]	2017	Convolutional recurrent neural networks (CRNNs) for robust speech recognition.
	Chiu et al. [118]	2018	RNN-transducer (RNN-T) models for end-to-end speech recognition.
	Dong et al. [120]	2018	Speech-Transformer: Leveraging self-attention for better processing of audio sequences.
	Bhaskar and Thasleema [121]	2023	LSTM for visual speech recognition using facial expressions.
	Daouad et al. [122]	2023	Various RNN variants for automatic speech recognition.
	Nasr et al. [124]	2023	End-to-end speech recognition using RNNs.
	Kumar et al. [125]	2023	Performance evaluation of RNNs in speech recognition tasks.
	Dhanjal et al. [123]	2024	Comprehensive study on different RNN models for speech recognition.
Financial Time Series Forecasting	Nelson et al. [127]	2017	Hybrid CNN-RNN model for stock price prediction.
	Bao et al. [129]	2017	Combining LSTMs with stacked autoencoders for financial time series forecasting.
	Fischer and Krauss [126]	2018	Deep RNNs for predicting stock returns, outperforming traditional ML models.
	Feng et al. [130]	2019	Transfer learning with RNNs for stock prediction.
	Rundo [131]	2019	Combining reinforcement learning with LSTMs for trading strategy development.
	Luo et al. [128]	2024	Attention-based CNN-BiLSTM model for improved financial forecasting.

6.4. Bioinformatics

In bioinformatics, RNNs have been used to analyze biological sequences such as DNA, RNA, and proteins. Li et al. [132] employed RNNs for gene prediction and protein structure prediction, demonstrating the ability of RNNs to capture dependencies within biological sequences, providing insights into genetic information and biological processes. Zhang et al. [133] used bidirectional LSTM

 doi:10.20944/preprints202408.0748.v1

for predicting DNA-binding protein sequences. Their model, called DeepSite, leveraged the sequential nature of biological data, achieving higher accuracy in identifying binding sites compared to traditional methods. This application demonstrated the potential of RNNs to enhance our understanding of protein-DNA interactions.

In the field of proteomics, RNNs have been used for protein structure prediction and function annotation. Xu et al. [134] developed an RNN-based model to predict protein secondary structures, showing that RNNs could effectively capture the sequential dependencies in amino acid sequences. This application provided significant advancements in protein structure prediction, which is essential for drug discovery and disease research.

More recently, researchers have explored the integration of RNNs with other neural network architectures for bioinformatics applications. For example, Yadav et al. [135] combined BiLSTM with CNNs to analyze protein sequences. Their model extracted local features and captured long-range dependencies with BiLSTMs, resulting in improved performance in protein classification tasks. Additionally, the use of ensemble deep learning has enhanced the performance of RNNs in bioinformatics. Aybey et al. [136] introduced an ensemble model for predicting protein-protein interactions using RNN, GRU, and CNN. The model improves the accuracy of interaction predictions. This approach highlighted the potential of ensemble deep learning to enhance the interpretability and performance of RNNs in bioinformatics.

6.5. Autonomous Vehicles

RNNs play an important role in autonomous vehicles by processing sequential data from sensors to make driving decisions. Li et al. [137] used RNNs for path planning, object detection, and trajectory prediction, enabling autonomous vehicles to navigate complex environments and make real-time decisions. Following this foundational work, researchers have continued to explore and enhance the use of RNNs in autonomous driving. For instance, Lee et al. [138] developed a deep learning framework that integrates LSTM with CNN for end-to-end driving. Their model utilized CNN to extract spatial features from camera images and LSTM to capture temporal dependencies, which improved the accuracy and robustness of driving decisions in dynamic environments.

Codevilla et al. [139] introduced a conditional imitation learning approach that combined RNNs with imitation learning for autonomous driving. The model learned from human driving demonstrations and used RNNs to predict future actions based on past observations. This approach allowed the vehicle to adapt to various driving conditions and make safer decisions in complex scenarios. Additionally, researchers have explored the use of LSTMs for trajectory prediction in autonomous vehicles. Altché and de La Fortelle [140] proposed an LSTM-based model that predicts the future trajectories of surrounding vehicles. This model leveraged the sequential nature of traffic data to anticipate the movements of other road users, enabling more accurate and proactive path planning for autonomous vehicles. Meanwhile, attention mechanisms have been integrated into RNN models to enhance their performance in autonomous driving tasks. Li et al. [141] introduced an attention-based LSTM model that focuses on relevant parts of the data, improving the detection and tracking of video objects.

Researchers have also explored the use of RNNs for behavior prediction in autonomous driving. Li et al. [142] proposed a model that combines RNNs with CNN to predict the intentions of other drivers. Their approach used sequential data to learn the behavioral patterns of surrounding vehicles, enabling the autonomous vehicle to anticipate potential hazards and respond accordingly. In addition, researchers have investigated the use of RNNs for decision-making in autonomous vehicles. Liu and Diao [143] introduced a deep reinforcement learning framework that incorporates GRU for decision-making in complex traffic scenarios. Their model used RNNs to process sequential observations and make real-time decisions, achieving state-of-the-art performance in various driving tasks.

 doi:10.20944/preprints202408.0748.v1

6.6. Anomaly Detection

RNNs are used in anomaly detection across different fields, such as cybersecurity, industrial monitoring, and healthcare. Altindal et al. [144] demonstrated the use of LSTM networks for anomaly detection in time series data, showing that RNNs could effectively model normal behavior patterns and identify deviations indicative of anomalies. Similarly, Matar et al. [145] proposed a model for anomaly detection in multivariate time series. Their model utilized BiLSTMs to learn temporal dependencies and to detect deviations from normal patterns. This approach was effective in industrial applications where monitoring the health of machinery and predicting failures is critical. In cybersecurity, Kumaresan et al. [146] applied RNNs to detect anomalies in network traffic. Their model analyzed sequential data to identify unusual patterns that could indicate security breaches or malicious activities. The use of RNNs allowed for real-time detection and response to potential threats, enhancing the security of network systems.

Furthermore, Li et al. [147] explored the application of RNNs for anomaly detection in manufacturing processes. They developed a model combining RNNs with Transfer learning to capture both temporal dependencies and feature representations. This method improved the detection of anomalies in complex industrial processes, contributing to the optimization of production efficiency and quality control. In healthcare, researchers have utilized RNNs for detecting anomalies in physiological signals. For instance, Mini et al. [148] employed RNNs to detect abnormal patterns in electrocardiogram (ECG) signals. Their model accurately identified deviations indicative of cardiac arrhythmias, demonstrating the potential of RNNs to assist in early diagnosis and monitoring of heart conditions.

Moreover, advances in unsupervised learning have further enhanced the capabilities of RNNs in anomaly detection. Zhou and Paffenroth [149] introduced a robust deep autoencoder model that leverages RNNs for unsupervised anomaly detection. This approach effectively captured the underlying structure of the data, identifying anomalies without requiring labeled training data. Ren et al. [150] proposed an attention-based RNN model that focuses on relevant time steps in the data, improving the accuracy and interpretability of anomaly detection. This approach allowed for more precise identification of anomalies by dynamically weighting the importance of different parts of the sequence.

Additionally, hybrid models combining RNNs with other neural network architectures have also been employed in anomaly detection. Munir et al. [151] developed a hybrid model that integrates CNNs and RNNs to detect anomalies in multivariate time series data. The CNN component extracted local features, while the RNN component captured temporal dependencies, resulting in improved performance in various anomaly detection tasks. The summary of RNN applications in bioinformatics, autonomous vehicles, and anomaly detection is shown in Table 4.

 doi:10.20944/preprints202408.0748.v1

Table 4. Summary of RNNs in Bioinformatics, Autonomous Vehicles, and Anomaly Detection

Application Domain	Reference	Year	Methods and Application
Bioinformatics	Li et al. [132]	2019	RNNs for gene prediction and protein structure prediction.
	Yadav et al. [135]	2019	Combining BiLSTM with CNNs for protein sequence analysis.
	Zhang et al. [133]	2020	DeepSite: Bidirectional LSTM for predicting DNA-binding protein sequences.
	Xu et al. [134]	2021	RNN-based model for predicting protein secondary structures.
	Aybey et al. [136]	2023	Ensemble model for predicting protein-protein interactions using RNN, GRU, and CNN.
Autonomous Vehicles	Altché and de La Fortelle [140]	2017	LSTM-based model for predicting the future trajectories of surrounding vehicles.
	Codevilla et al. [139]	2018	Conditional imitation learning combining RNNs with imitation learning for autonomous driving.
	Li et al. [137]	2020	RNNs for path planning, object detection, and trajectory prediction in autonomous vehicles.
	Lee et al. [138]	2020	Integrating LSTM with CNN for end-to-end driving.
	Li et al. [142]	2024	Combining RNNs with CNN to predict the intentions of other drivers.
	Li et al. [141]	2024	Attention-based LSTM for improving the detection and tracking of video objects.
	Liu and Diao [143]	2024	Deep reinforcement learning framework with GRU for decision-making in traffic scenarios.
Anomaly Detection	Zhou and Paffenroth [149]	2017	Unsupervised anomaly detection using robust deep autoencoder models with RNNs.
	Munir et al. [151]	2018	Hybrid model integrating CNNs and RNNs for anomaly detection in multivariate time series data.
	Ren et al. [150]	2019	Attention-based RNN model for improving accuracy and interpretability in anomaly detection.
	Li et al. [147]	2023	Combining RNNs with Transfer learning for anomaly detection in manufacturing processes.
	Mini et al. [148]	2023	RNNs for detecting abnormal patterns in ECG signals in healthcare.
	Matar et al. [145]	2023	BiLSTM for anomaly detection in multivariate time series.
	Kumaresan et al. [146]	2024	RNNs for detecting anomalies in network traffic in cybersecurity.
	Altindal et al. [144]	2024	LSTM networks for anomaly detection in time series data.

7. Challenges and Future Research Directions

Despite significant advancements, several unresolved problems are encountered when applying RNNs. Addressing these issues is crucial for further improving the performance and usage of RNNs.

7.1. Scalability and Efficiency

Training RNNs on large datasets with long sequences remains computationally intensive and time-consuming [152–154]. Although techniques like gradient checkpointing and hardware accelerators have provided improvements, the sequential nature of RNNs continues to limit their scalability compared to parallelizable architectures like Transformers [155]. Future research could focus on developing more efficient training algorithms and exploring asynchronous and parallel training methods to distribute the computational load more effectively. Additionally, hybrid architectures that combine RNNs with other models, such as integrating RNNs with attention mechanisms or

 doi:10.20944/preprints202408.0748.v1

convolutional layers, could provide new solutions. These hybrid models have the potential to reduce training times and improve scalability while maintaining the performance advantages of RNNs [89].

7.2. Interpretability and Explainability

RNNs are often perceived as "black-box" models due to their complex internal dynamics, making it challenging to interpret their decisions [156,157]. Although attention mechanisms and post-hoc explanation techniques like Local Interpretable Model-agnostic Explanations (LIME) and SHapley Addictive exPlanations (SHAP) have been proposed to improve interpretability, these methods can still be improved to further provide more comprehensive explanations [158]. Therefore, future research should aim to develop inherently interpretable RNN architectures and hierarchical models that offer structured insights into the model's decision-making process. Additionally, integrating domain knowledge into RNN models can help align their behavior with human reasoning, enhancing both interpretability and performance in specialized applications.

7.3. Bias and Fairness

RNNs can inadvertently learn and propagate biases present in the training data, leading to unfair predictions. While various bias detection and mitigation techniques have been developed, such as fairness-aware algorithms and adversarial training, these methods need further refinement to ensure fairness across diverse applications and datasets [159–161]. Research should continue to focus on developing robust bias detection techniques and fair training algorithms that explicitly incorporate fairness constraints. Additionally, transparency and accountability frameworks, including external audits and impact assessments, are essential for ensuring that RNNs are developed and deployed responsibly.

7.4. Data Dependency and Quality

RNNs require large amounts of high-quality, labeled sequential data for effective training [162]. In many real-world scenarios, such data may be scarce, noisy, or incomplete. Although data augmentation, transfer learning, and semi-supervised learning techniques have been explored, these methods require further refinement to handle diverse data challenges more effectively. Future research should focus on enhancing these techniques to improve the robustness of RNNs when trained on limited or imperfect data. Additionally, developing new methods for utilizing unlabeled data and integrating domain-specific knowledge can further improve the performance of RNNs in data-scarce environments.

7.5. Overfitting and Generalization

RNNs, particularly deep architectures, are prone to overfitting, especially when trained on small datasets [163]. Ensuring that RNN models generalize well to unseen data without overfitting remains a significant challenge. While regularization techniques like dropout and L2 regularization are commonly used, more robust methods for improving generalization are needed. Future research can explore advanced regularization techniques, such as adversarial training and ensemble methods, to enhance the generalization capabilities of RNNs. Additionally, applying data augmentation and transfer learning can help RNN models learn more robust features, improving their ability to generalize to new data.

8. Conclusion

RNNs have demonstrated a remarkable ability to model sequential data, making them indispensable in numerous ML applications such as natural language processing, speech recognition, time series prediction, bioinformatics, and autonomous systems. This paper provided a comprehensive overview of RNNs and their variants, covering fundamental architectures like basic RNNs, LSTM networks, and GRUs, as well as advanced variants, including Bidirectional RNNs, Peephole LSTMs, ESNs, and IndRNNs. The study provided a detailed and comprehensive review of RNNs, their architectures,

 doi:10.20944/preprints202408.0748.v1

applications, and challenges. The paper will be a valuable resource for researchers and practitioners in the field of machine learning, helping to guide future developments and applications of RNNs.

Author Contributions: Conceptualization, I.D.M., T.G.S., and G.O.; methodology, I.D.M.; validation, I.D.M., T.G.S., and G.O.; investigation, I.D.M., T.G.S., and G.O.; resources, X.X.; writing—original draft preparation, I.D.M. and G.O..; writing—review and editing, I.D.M., T.G.S., and G.O.; visualization, I.D.M; supervision, T.G.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ANN	Artificial Neural Network
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
DL	Deep Learning
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
ML	Machine Learning
NAS	Neural Architecture Search
NLP	Natural Language Processing
RNN	Recurrent Neural Network
RL	Reinforcement Learning
SHAP	SHapley Additive exPlanations
TPU	Tensor Processing Unit
VAE	Variational Autoencoder

References

1. O'Halloran, T.; Obaido, G.; Otegbade, B.; Mienye, I.D. A deep learning approach for Maize Lethal Necrosis and Maize Streak Virus disease detection. *Machine Learning with Applications* **2024**, *16*, 100556.
2. Peng, Y.; He, L.; Hu, D.; Liu, Y.; Yang, L.; Shang, S. Decoupling Deep Learning for Enhanced Image Recognition Interpretability. *ACM Transactions on Multimedia Computing, Communications and Applications* **2024**.
3. Khan, W.; Daud, A.; Khan, K.; Muhammad, S.; Haq, R. Exploring the frontiers of deep learning and natural language processing: A comprehensive overview of key challenges and emerging trends. *Natural Language Processing Journal* **2023**, p. 100026.
4. Al-Jumaili, A.H.A.; Muniyandi, R.C.; Hasan, M.K.; Paw, J.K.S.; Singh, M.J. Big data analytics using cloud computing based frameworks for power management systems: Status, constraints, and future recommendations. *Sensors* **2023**, *23*, 2952.
5. Gill, S.S.; Wu, H.; Patros, P.; Ottaviani, C.; Arora, P.; Pujol, V.C.; Haunschild, D.; Parlikad, A.K.; Cetinkaya, O.; Lutfiyya, H.; et al. Modern computing: Vision and challenges. *Telematics and Informatics Reports* **2024**, p. 100116.
6. Mienye, I.D.; Jere, N. A Survey of Decision Trees: Concepts, Algorithms, and Applications. *IEEE Access* **2024**.
7. Alhajeri, M.S.; Ren, Y.M.; Ou, F.; Abdullah, F.; Christofides, P.D. Model predictive control of nonlinear processes using transfer learning-based recurrent neural networks. *Chemical Engineering Research and Design* **2024**, *205*, 1–12.
8. Shahinzadeh, H.; Mahmoudi, A.; Asilian, A.; Sadrarhami, H.; Hemmati, M.; Saberi, Y. Deep Learning: A Overview of Theory and Architectures. In Proceedings of the 2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP). IEEE, 2024, pp. 1–11.

 doi:10.20944/preprints202408.0748.v1

9. Baruah, R.D.; Organero, M.M. Explicit Context Integrated Recurrent Neural Network for applications in smart environments. *Expert Systems with Applications* **2024**, p. 124752.
10. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *nature* **1986**, *323*, 533–536.
11. Lalapura, V.S.; Amudha, J.; Satheesh, H.S. Recurrent neural networks for edge intelligence: a survey. *ACM Computing Surveys (CSUR)* **2021**, *54*, 1–38.
12. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation* **1997**, *9*, 1735–1780.
13. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* **2014**.
14. Liu, F.; Li, J.; Wang, L. PI-LSTM: Physics-informed long short-term memory network for structural response modeling. *Engineering Structures* **2023**, *292*, 116500.
15. Ni, Q.; Ji, J.; Feng, K.; Zhang, Y.; Lin, D.; Zheng, J. Data-driven bearing health management using a novel multi-scale fused feature and gated recurrent unit. *Reliability Engineering & System Safety* **2024**, *242*, 109753.
16. Niu, Z.; Zhong, G.; Yue, G.; Wang, L.N.; Yu, H.; Ling, X.; Dong, J. Recurrent attention unit: A new gated recurrent unit for long-term memory of important parts in sequential data. *Neurocomputing* **2023**, *517*, 1–9.
17. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* **2015**.
18. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation* **2019**, *31*, 1235–1270.
19. Tarwani, K.M.; Edem, S. Survey on recurrent neural network in natural language processing. *Int. J. Eng. Trends Technol* **2017**, *48*, 301–304.
20. Dutta, K.K.; Poornima, S.; Sharma, R.; Nair, D.; Ploeger, P.G. Applications of recurrent neural network: Overview and case studies. In *Recurrent Neural Networks*; CRC press, 2022; pp. 23–41.
21. Quradaa, F.H.; Shahzad, S.; Almoqbily, R.S. A systematic literature review on the applications of recurrent neural networks in code clone research. *Plos one* **2024**, *19*, e0296858.
22. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep learning*; MIT press, 2016.
23. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* **2016**, *28*, 2222–2232.
24. Al-Selwi, S.M.; Hassan, M.F.; Abdulkadir, S.J.; Muneer, A.; Sumiea, E.H.; Alqushaibi, A.; Ragab, M.G. RNN-LSTM: From applications to modeling techniques and beyond—Systematic review. *Journal of King Saud University-Computer and Information Sciences* **2024**, p. 102068.
25. Zaremba, W.; Sutskever, I.; Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* **2014**.
26. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* **2018**.
27. Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* **2018**, *8*, 6085.
28. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* **2014**.
29. Badawy, M.; Ramadan, N.; Hefny, H.A. Healthcare predictive analytics using machine learning and deep learning techniques: a survey. *Journal of Electrical Systems and Information Technology* **2023**, *10*, 40.
30. Ismaeel, A.G.; Janardhanan, K.; Sankar, M.; Natarajan, Y.; Mahmood, S.N.; Alani, S.; Shather, A.H. Traffic pattern classification in smart cities using deep recurrent neural network. *Sustainability* **2023**, *15*, 14522.
31. Mers, M.; Yang, Z.; Hsieh, Y.A.; Tsai, Y. Recurrent neural networks for pavement performance forecasting: review and model performance comparison. *Transportation Research Record* **2023**, *2677*, 610–624.
32. Chen, Y.; Cheng, Q.; Cheng, Y.; Yang, H.; Yu, H. Applications of recurrent neural networks in environmental factor forecasting: a review. *Neural computation* **2018**, *30*, 2855–2881.
33. Linardos, V.; Drakaki, M.; Tzionas, P.; Karnavas, Y.L. Machine learning in disaster management: recent developments in methods and applications. *Machine Learning and Knowledge Extraction* **2022**, *4*.
34. Zhang, J.; Liu, H.; Chang, Q.; Wang, L.; Gao, R.X. Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly. *CIRP annals* **2020**, *69*, 9–12.

 doi:10.20944/preprints202408.0748.v1

35. Tsantekidis, A.; Passalis, N.; Tefas, A. Recurrent neural networks. In *Deep learning for robot perception and cognition*; Elsevier, 2022; pp. 101–115.
36. Mienye, I.D.; Jere, N. Deep Learning for Credit Card Fraud Detection: A Review of Algorithms, Challenges, and Solutions. *IEEE Access* **2024**, *12*, 96893–96910. https://doi.org/10.1109/ACCESS.2024.3426955.
37. Rezk, N.M.; Purnaprajna, M.; Nordström, T.; Ul-Abdin, Z. Recurrent neural networks: An embedded computing perspective. *IEEE Access* **2020**, *8*, 57967–57996.
38. Yu, Y.; Adu, K.; Tashi, N.; Anokye, P.; Wang, X.; Ayidzoe, M.A. Rmaf: Relu-memristor-like activation function for deep learning. *IEEE Access* **2020**, *8*, 72727–72741.
39. Mienye, I.D.; Ainah, P.K.; Emmanuel, I.D.; Esenogho, E. Sparse noise minimization in image classification using Genetic Algorithm and DenseNet. In Proceedings of the 2021 Conference on Information Communications Technology and Society (ICTAS). IEEE, 2021, pp. 103–108.
40. Ciaburro, G.; Venkateswaran, B. *Neural Networks with R: Smart models using CNN, RNN, deep learning, and artificial intelligence principles*; Packt Publishing Ltd, 2017.
41. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378* **2018**.
42. Szandała, T. Review and comparison of commonly used activation functions for deep neural networks. *Bio-inspired neurocomputing* **2021**, pp. 203–224.
43. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* **2015**.
44. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing* **2022**, *503*, 92–108.
45. Mienye, I.D.; Sun, Y.; Wang, Z. An improved ensemble learning approach for the prediction of heart disease risk. *Informatics in Medicine Unlocked* **2020**, *20*, 100402.
46. Martins, A.; Astudillo, R. From softmax to sparsemax: A sparse model of attention and multi-label classification. In Proceedings of the International conference on machine learning. PMLR, 2016, pp. 1614–1623.
47. Bianchi, F.M.; Maiorino, E.; Kampffmeyer, M.C.; Rizzi, A.; Jenssen, R.; Bianchi, F.M.; Maiorino, E.; Kampffmeyer, M.C.; Rizzi, A.; Jenssen, R. Properties and training in recurrent neural networks. *Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis* **2017**, pp. 9–21.
48. Mohajerin, N.; Waslander, S.L. State initialization for recurrent neural network modeling of time-series data. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017, pp. 2330–2337.
49. Forgione, M.; Muni, A.; Piga, D.; Gallieri, M. On the adaptation of recurrent neural networks for system identification. *Automatica* **2023**, *155*, 111092.
50. Fei, H.; Tan, F. Bidirectional grid long short-term memory (bigridlstm): A method to address context-sensitivity and vanishing gradient. *Algorithms* **2018**, *11*, 172.
51. Dong, X.; Chowdhury, S.; Qian, L.; Li, X.; Guan, Y.; Yang, J.; Yu, Q. Deep learning for named entity recognition on Chinese electronic medical records: Combining deep transfer learning with multitask bi-directional LSTM RNN. *PloS one* **2019**, *14*, e0216046.
52. Chorowski, J.K.; Bahdanau, D.; Serdyuk, D.; Cho, K.; Bengio, Y. Attention-based models for speech recognition. *Advances in neural information processing systems* **2015**, *28*.
53. Zhou, M.; Duan, N.; Liu, S.; Shum, H.Y. Progress in neural NLP: modeling, learning, and reasoning. *Engineering* **2020**, *6*, 275–290.
54. Naseem, U.; Razzak, I.; Khan, S.K.; Prasad, M. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing* **2021**, *20*, 1–35.
55. Adil, M.; Wu, J.Z.; Chakrabortty, R.K.; Alahmadi, A.; Ansari, M.F.; Ryan, M.J. Attention-based STL-BiLSTM network to forecast tourist arrival. *Processes* **2021**, *9*, 1759.
56. Min, S.; Park, S.; Kim, S.; Choi, H.S.; Lee, B.; Yoon, S. Pre-training of deep bidirectional protein sequence representations with structural information. *IEEE Access* **2021**, *9*, 123912–123926.
57. Jain, A.; Zamir, A.R.; Savarese, S.; Saxena, A. Structural-rnn: Deep learning on spatio-temporal graphs. In Proceedings of the Proceedings of the ieee conference on computer vision and pattern recognition, 2016, pp. 5308–5317.

 doi:10.20944/preprints202408.0748.v1

58. Pascanu, R.; Gulcehre, C.; Cho, K.; Bengio, Y. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026* **2013**.
59. Shi, H.; Xu, M.; Li, R. Deep learning for household load forecasting—A novel pooling deep RNN. *IEEE Transactions on Smart Grid* **2017**, *9*, 5271–5280.
60. Gal, Y.; Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. *Advances in neural information processing systems* **2016**, *29*.
61. Moradi, R.; Berangi, R.; Minaei, B. A survey of regularization strategies for deep models. *Artificial Intelligence Review* **2020**, *53*, 3947–3986.
62. Salehin, I.; Kang, D.K. A review on dropout regularization approaches for deep neural networks within the scholarly domain. *Electronics* **2023**, *12*, 3106.
63. Cai, S.; Shu, Y.; Chen, G.; Ooi, B.C.; Wang, W.; Zhang, M. Effective and efficient dropout for deep convolutional neural networks. *arXiv preprint arXiv:1904.03392* **2019**.
64. Garbin, C.; Zhu, X.; Marques, O. Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia tools and applications* **2020**, *79*, 12777–12815.
65. Borawar, L.; Kaur, R. ResNet: Solving vanishing gradient in deep networks. In Proceedings of the Proceedings of International Conference on Recent Trends in Computing: ICRTC 2022. Springer, 2023, pp. 235–247.
66. Mienye, I.D.; Sun, Y. A deep learning ensemble with data resampling for credit card fraud detection. *IEEE Access* **2023**, *11*, 30628–30638.
67. Kiperwasser, E.; Goldberg, Y. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* **2016**, *4*, 313–327.
68. Zhang, W.; Li, H.; Tang, L.; Gu, X.; Wang, L.; Wang, L. Displacement prediction of Jiuxianping landslide using gated recurrent unit (GRU) networks. *Acta Geotechnica* **2022**, *17*, 1367–1382.
69. Cahuantzi, R.; Chen, X.; Güttel, S. A comparison of LSTM and GRU networks for learning symbolic sequences. In Proceedings of the Science and Information Conference. Springer, 2023, pp. 771–785.
70. Shewalkar, A.; Nyavanandi, D.; Ludwig, S.A. Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU. *Journal of Artificial Intelligence and Soft Computing Research* **2019**, *9*, 235–245.
71. Vatanchi, S.M.; Etemadfard, H.; Maghrebi, M.F.; Shad, R. A comparative study on forecasting of long-term daily streamflow using ANN, ANFIS, BiLSTM and CNN-GRU-LSTM. *Water Resources Management* **2023**, *37*, 4769–4785.
72. Mateus, B.C.; Mendes, M.; Farinha, J.T.; Assis, R.; Cardoso, A.M. Comparing LSTM and GRU models to predict the condition of a pulp paper press. *Energies* **2021**, *14*, 6958.
73. Gers, F.A.; Schmidhuber, J. Recurrent nets that time and count. In Proceedings of the Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. IEEE, 2000, Vol. 3, pp. 189–194.
74. Gers, F.A.; Schraudolph, N.N.; Schmidhuber, J. Learning precise timing with LSTM recurrent networks. *Journal of machine learning research* **2002**, *3*, 115–143.
75. Jaeger, H. Adaptive nonlinear system identification with echo state networks. *Advances in neural information processing systems* **2002**, *15*.
76. Li, S.; Li, W.; Cook, C.; Zhu, C.; Gao, Y. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 5457–5466.
77. Yang, J.; Qu, J.; Mi, Q.; Li, Q. A CNN-LSTM model for tailings dam risk prediction. *IEEE Access* **2020**, *8*, 206491–206502.
78. Ren, P.; Xiao, Y.; Chang, X.; Huang, P.Y.; Li, Z.; Chen, X.; Wang, X. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)* **2021**, *54*, 1–34.
79. Mellor, J.; Turner, J.; Storkey, A.; Crowley, E.J. Neural architecture search without training. In Proceedings of the International conference on machine learning. PMLR, 2021, pp. 7588–7598.
80. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* **2016**.
81. Chen, X.; Wu, S.Z.; Hong, M. Understanding gradient clipping in private sgd: A geometric perspective. *Advances in Neural Information Processing Systems* **2020**, *33*, 13773–13782.

 doi:10.20944/preprints202408.0748.v1

82. Qian, J.; Wu, Y.; Zhuang, B.; Wang, S.; Xiao, J. Understanding gradient clipping in incremental gradient methods. In Proceedings of the International Conference on Artificial Intelligence and Statistics. PMLR, 2021, pp. 1504–1512.
83. Zhang, Z. Improved adam optimizer for deep neural networks. In Proceedings of the 2018 IEEE/ACM 26th international symposium on quality of service (IWQoS). Ieee, 2018, pp. 1–2.
84. de Santana Correia, A.; Colombini, E.L. Attention, please! A survey of neural attention models in deep learning. *Artificial Intelligence Review* **2022**, *55*, 6037–6124.
85. Lin, J.; Ma, J.; Zhu, J.; Cui, Y. Short-term load forecasting based on LSTM networks considering attention mechanism. *International Journal of Electrical Power & Energy Systems* **2022**, *137*, 107818.
86. Chaudhari, S.; Mithal, V.; Polatkan, G.; Ramanath, R. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)* **2021**, *12*, 1–32.
87. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* **2014**.
88. Luong, M.T.; Pham, H.; Manning, C.D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* **2015**.
89. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.
90. Souri, A.; El Maazouzi, Z.; Al Achhab, M.; El Mohajir, B.E. Arabic text generation using recurrent neural networks. In Proceedings of the Big Data, Cloud and Applications: Third International Conference, BDCA 2018, Kenitra, Morocco, April 4–5, 2018, Revised Selected Papers 3. Springer, 2018, pp. 523–533.
91. Islam, M.S.; Mousumi, S.S.S.; Abujar, S.; Hossain, S.A. Sequence-to-sequence Bangla sentence generation with LSTM recurrent neural networks. *Procedia Computer Science* **2019**, *152*, 51–58.
92. Gajendran, S.; Manjula, D.; Sugumaran, V. Character level and word level embedding with bidirectional LSTM–Dynamic recurrent neural network for biomedical named entity recognition from literature. *Journal of Biomedical Informatics* **2020**, *112*, 103609.
93. Hu, H.; Liao, M.; Mao, W.; Liu, W.; Zhang, C.; Jing, Y. Variational auto-encoder for text generation. In Proceedings of the 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC). IEEE, 2020, pp. 595–598.
94. Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; Choi, Y. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751* **2019**.
95. Yin, W.; Schütze, H. Attentive convolution: Equipping cnns with rnn-style attention mechanisms. *Transactions of the Association for Computational Linguistics* **2018**, *6*, 687–702.
96. Hussein, M.A.H.; Savaş, S. LSTM-Based Text Generation: A Study on Historical Datasets. *arXiv preprint arXiv:2403.07087* **2024**.
97. Baskaran, S.; Alagarsamy, S.; S, S.; Shivam, S. Text Generation using Long Short-Term Memory. In Proceedings of the 2024 Third International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), 2024, pp. 1–6. https://doi.org/10.1109/INCOS59338.2024.10527547.
98. Keskar, N.S.; McCann, B.; Varshney, L.R.; Xiong, C.; Socher, R. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858* **2019**.
99. Guo, H. Generating text with deep reinforcement learning. *arXiv preprint arXiv:1510.09202* **2015**.
100. Yadav, V.; Verma, P.; Katiyar, V. Long short term memory (LSTM) model for sentiment analysis in social data for e-commerce products reviews in Hindi languages. *International Journal of Information Technology* **2023**, *15*, 759–772.
101. Abimbola, B.; de La Cal Marin, E.; Tan, Q. Enhancing Legal Sentiment Analysis: A Convolutional Neural Network–Long Short-Term Memory Document-Level Model. *Machine Learning and Knowledge Extraction* **2024**, *6*, 877–897.
102. Zulqarnain, M.; Ghazali, R.; Aamir, M.; Hassim, Y.M.M. An efficient two-state GRU based on feature attention mechanism for sentiment analysis. *Multimedia Tools and Applications* **2024**, *83*, 3085–3110.
103. Pujari, P.; Padalia, A.; Shah, T.; Devadkar, K. Hybrid CNN and RNN for Twitter Sentiment Analysis. In Proceedings of the International Conference on Smart Computing and Communication. Springer, 2024, pp. 297–310.
104. Wankhade, M.; Annavarapu, C.S.R.; Abraham, A. CBMAFM: CNN-BiLSTM multi-attention fusion mechanism for sentiment classification. *Multimedia Tools and Applications* **2024**, *83*, 51755–51786.

 doi:10.20944/preprints202408.0748.v1

105. Sangeetha, J.; Kumaran, U. A hybrid optimization algorithm using BiLSTM structure for sentiment analysis. *Measurement: Sensors* **2023**, *25*, 100619.
106. He, R.; McAuley, J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Proceedings of the proceedings of the 25th international conference on world wide web, 2016, pp. 507–517.
107. Samir, A.; Elkaffas, S.M.; Madbouly, M.M. Twitter sentiment analysis using BERT. In Proceedings of the 2021 31st international conference on computer theory and applications (ICCTA). IEEE, 2021, pp. 182–186.
108. Prottasha, N.J.; Sami, A.A.; Kowsher, M.; Murad, S.A.; Bairagi, A.K.; Masud, M.; Baz, M. Transfer learning for sentiment analysis using BERT based supervised fine-tuning. *Sensors* **2022**, *22*, 4157.
109. Mujahid, M.; Rustam, F.; Shafique, R.; Chunduri, V.; Villar, M.G.; Ballester, J.B.; Diez, I.d.l.T.; Ashraf, I. Analyzing sentiments regarding ChatGPT using novel BERT: A machine learning approach. *Information* **2023**, *14*, 474.
110. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* **2016**.
111. Sennrich, R.; Haddow, B.; Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* **2015**.
112. Kang, L.; He, S.; Wang, M.; Long, F.; Su, J. Bilingual attention based neural machine translation. *Applied Intelligence* **2023**, *53*, 4302–4315.
113. Yang, Z.; Dai, Z.; Salakhutdinov, R.; Cohen, W.W. Breaking the softmax bottleneck: A high-rank RNN language model. *arXiv preprint arXiv:1711.03953* **2017**.
114. Song, K.; Tan, X.; Qin, T.; Lu, J.; Liu, T.Y. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450* **2019**.
115. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* **2012**, *29*, 82–97.
116. Hannun, A.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* **2014**.
117. Amodei, D.; Ananthanarayanan, S.; Anubhai, R.; Bai, J.; Battenberg, E.; Case, C.; Casper, J.; Catanzaro, B.; Cheng, Q.; Chen, G.; et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In Proceedings of the International conference on machine learning. PMLR, 2016, pp. 173–182.
118. Chiu, C.C.; Sainath, T.N.; Wu, Y.; Prabhavalkar, R.; Nguyen, P.; Chen, Z.; Kannan, A.; Weiss, R.J.; Rao, K.; Gonina, E.; et al. State-of-the-art speech recognition with sequence-to-sequence models. In Proceedings of the 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2018, pp. 4774–4778.
119. Zhang, Y.; Chan, W.; Jaitly, N. Very deep convolutional networks for end-to-end speech recognition. In Proceedings of the 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2017, pp. 4845–4849.
120. Dong, L.; Xu, S.; Xu, B. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In Proceedings of the 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2018, pp. 5884–5888.
121. Bhaskar, S.; Thasleema, T. LSTM model for visual speech recognition through facial expressions. *Multimedia Tools and Applications* **2023**, *82*, 5455–5472.
122. Daouad, M.; Allah, F.A.; Dadi, E.W. An automatic speech recognition system for isolated Amazigh word using 1D & 2D CNN-LSTM architecture. *International Journal of Speech Technology* **2023**, *26*, 775–787.
123. Dhanjal, A.S.; Singh, W. A comprehensive survey on automatic speech recognition using neural networks. *Multimedia Tools and Applications* **2024**, *83*, 23367–23412.
124. Nasr, S.; Duwairi, R.; Quwaider, M. End-to-end speech recognition for arabic dialects. *Arabian Journal for Science and Engineering* **2023**, *48*, 10617–10633.
125. Kumar, D.; Aziz, S. Performance Evaluation of Recurrent Neural Networks-LSTM and GRU for Automatic Speech Recognition. In Proceedings of the 2023 International Conference on Computer, Electronics & Electrical Engineering & their Applications (IC2E3). IEEE, 2023, pp. 1–6.

 doi:10.20944/preprints202408.0748.v1

126. Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research* **2018**, *270*, 654–669.
127. Nelson, D.M.; Pereira, A.C.; De Oliveira, R.A. Stock market's price movement prediction with LSTM neural networks. In Proceedings of the 2017 International joint conference on neural networks (IJCNN). Ieee, 2017, pp. 1419–1426.
128. Luo, A.; Zhong, L.; Wang, J.; Wang, Y.; Li, S.; Tai, W. Short-term stock correlation forecasting based on CNN-BiLSTM enhanced by attention mechanism. *IEEE Access* **2024**.
129. Bao, W.; Yue, J.; Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one* **2017**, *12*, e0180944.
130. Feng, F.; Chen, H.; He, X.; Ding, J.; Sun, M.; Chua, T.S. Enhancing Stock Movement Prediction with Adversarial Training. In Proceedings of the IJCAI, 2019, Vol. 19, pp. 5843–5849.
131. Rundo, F. Deep LSTM with reinforcement learning layer for financial trend prediction in FX high frequency trading systems. *Applied Sciences* **2019**, *9*, 4460.
132. Li, Y.; Huang, C.; Ding, L.; Li, Z.; Pan, Y.; Gao, X. Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. *Methods* **2019**, *166*, 4–21.
133. Zhang, Y.; Qiao, S.; Ji, S.; Li, Y. DeepSite: bidirectional LSTM and CNN models for predicting DNA–protein binding. *International Journal of Machine Learning and Cybernetics* **2020**, *11*, 841–851.
134. Xu, J.; Mcpartlon, M.; Li, J. Improved protein structure prediction by deep learning irrespective of co-evolution information. *Nature Machine Intelligence* **2021**, *3*, 601–609.
135. Yadav, S.; Ekbal, A.; Saha, S.; Kumar, A.; Bhattacharyya, P. Feature assisted stacked attentive shortest dependency path based Bi-LSTM model for protein–protein interaction. *Knowledge-Based Systems* **2019**, *166*, 18–29.
136. Aybey, E.; Gümüş, Ö. SENSDeep: an ensemble deep learning method for protein–protein interaction sites prediction. *Interdisciplinary Sciences: Computational Life Sciences* **2023**, *15*, 55–87.
137. Li, Z.; Du, X.; Cao, Y. DAT-RNN: trajectory prediction with diverse attention. In Proceedings of the 2020 19th IEEE International conference on machine learning and applications (ICMLA). IEEE, 2020, pp. 1512–1518.
138. Lee, M.j.; Ha, Y.g. Autonomous Driving Control Using End-to-End Deep Learning. In Proceedings of the 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), 2020, pp. 470–473. https://doi.org/10.1109/BigComp48618.2020.00-23.
139. Codevilla, F.; Müller, M.; López, A.; Koltun, V.; Dosovitskiy, A. End-to-end driving via conditional imitation learning. In Proceedings of the 2018 IEEE international conference on robotics and automation (ICRA). IEEE, 2018, pp. 4693–4700.
140. Altché, F.; de La Fortelle, A. An LSTM network for highway trajectory prediction. In Proceedings of the 2017 IEEE 20th international conference on intelligent transportation systems (ITSC). IEEE, 2017, pp. 353–359.
141. Li, P.; Zhang, Y.; Yuan, L.; Xiao, H.; Lin, B.; Xu, X. Efficient long-short temporal attention network for unsupervised video object segmentation. *Pattern Recognition* **2024**, *146*, 110078.
142. Li, R.; Shu, X.; Li, C. Driving Behavior Prediction Based on Combined Neural Network Model. *IEEE Transactions on Computational Social Systems* **2024**, *11*, 4488–4496. https://doi.org/10.1109/TCSS.2024.3350199.
143. Liu, Y.; Diao, S. An automatic driving trajectory planning approach in complex traffic scenarios based on integrated driver style inference and deep reinforcement learning. *PLoS one* **2024**, *19*, e0297192.
144. Altindal, M.C.; Nivlet, P.; Tabib, M.; Rasheed, A.; Kristiansen, T.G.; Khosravanian, R. Anomaly detection in multivariate time series of drilling data. *Geoenergy Science and Engineering* **2024**, *237*, 212778.
145. Matar, M.; Xia, T.; Huguenard, K.; Huston, D.; Wshah, S. Multi-head attention based bi-lstm for anomaly detection in multivariate time-series of wsn. In Proceedings of the 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE, 2023, pp. 1–5.
146. Kumaresan, S.J.; Senthilkumar, C.; Kongkham, D.; Beenarani, B.; Nirmala, P. Investigating the Effectiveness of Recurrent Neural Networks for Network Anomaly Detection. In Proceedings of the 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE). IEEE, 2024, pp. 1–5.
147. Li, E.; Bedi, S.; Melek, W. Anomaly detection in three-axis CNC machines using LSTM networks and transfer learning. *The International Journal of Advanced Manufacturing Technology* **2023**, *127*, 5185–5198.

 doi:10.20944/preprints202408.0748.v1

148. Minic, A.; Jovanovic, L.; Bacanin, N.; Stoean, C.; Zivkovic, M.; Spalevic, P.; Petrovic, A.; Dobrojevic, M.; Stoean, R. Applying recurrent neural networks for anomaly detection in electrocardiogram sensor data. *Sensors* **2023**, *23*, 9878.
149. Zhou, C.; Paffenroth, R.C. Anomaly detection with robust deep autoencoders. In Proceedings of the Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 665–674.
150. Ren, H.; Xu, B.; Wang, Y.; Yi, C.; Huang, C.; Kou, X.; Xing, T.; Yang, M.; Tong, J.; Zhang, Q. Time-series anomaly detection service at microsoft. In Proceedings of the Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 3009–3017.
151. Munir, M.; Siddiqui, S.A.; Dengel, A.; Ahmed, S. DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *Ieee Access* **2018**, *7*, 1991–2005.
152. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting* **2021**, *37*, 388–427.
153. Ahmed, S.F.; Alam, M.S.B.; Hassan, M.; Rozbu, M.R.; Ishtiak, T.; Rafa, N.; Mofijur, M.; Shawkat Ali, A.; Gandomi, A.H. Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artificial Intelligence Review* **2023**, *56*, 13521–13617.
154. Li, X.; Qin, T.; Yang, J.; Liu, T.Y. LightRNN: Memory and computation-efficient recurrent neural networks. *Advances in Neural Information Processing Systems* **2016**, *29*.
155. Katharopoulos, A.; Vyas, A.; Pappas, N.; Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the International conference on machine learning. PMLR, 2020, pp. 5156–5165.
156. Shao, W.; Li, B.; Yu, W.; Xu, J.; Wang, H. When Is It Likely to Fail? Performance Monitor for Black-Box Trajectory Prediction Model. *IEEE Transactions on Automation Science and Engineering* **2024**.
157. Jacobs, W.R.; Kadirkamanathan, V.; Anderson, S.R. Interpretable deep learning for nonlinear system identification using frequency response functions with ensemble uncertainty quantification. *IEEE Access* **2024**.
158. Mamalakis, M.; Mamalakis, A.; Agartz, I.; Mørch-Johnsen, L.E.; Murray, G.; Suckling, J.; Lio, P. Solving the enigma: Deriving optimal explanations of deep networks. *arXiv preprint arXiv:2405.10008* **2024**.
159. Shah, M.; Sureja, N. A Comprehensive Review of Bias in Deep Learning Models: Methods, Impacts, and Future Directions. *Archives of Computational Methods in Engineering* **2024**, pp. 1–13.
160. Goethals, S.; Calders, T.; Martens, D. Beyond Accuracy-Fairness: Stop evaluating bias mitigation methods solely on between-group metrics. *arXiv preprint arXiv:2401.13391* **2024**.
161. Weerts, H.; Pfisterer, F.; Feurer, M.; Eggensperger, K.; Bergman, E.; Awad, N.; Vanschoren, J.; Pechenizkiy, M.; Bischl, B.; Hutter, F. Can fairness be automated? Guidelines and opportunities for fairness-aware AutoML. *Journal of Artificial Intelligence Research* **2024**, *79*, 639–677.
162. Bai, Y.; Geng, X.; Mangalam, K.; Bar, A.; Yuille, A.L.; Darrell, T.; Malik, J.; Efros, A.A. Sequential modeling enables scalable learning for large vision models. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 22861–22872.
163. Taye, M.M. Understanding of machine learning with deep learning: architectures, workflow, applications and future directions. *Computers* **2023**, *12*, 91.