2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)
21-22 February, 2024, Mazandaran University of Science and Technology, Babol, Iran

Deep Learning: A Overview of Theory and Architectures

Hossein Shahinzadeh
Department of Electrical Engineering, Amirkabir University of Technology
Tehran, Iran
h.s.shahinzadeh@ieee.org

Arezou Mahmoudi
Faculty of Electrical and Computer Engineering, University of Tabriz
Tabriz, Iran
ar-mahmoodi@tabrizu.ac.ir

Amirhosein Asilian
Department of Electrical Engineering, Najafabad Branch, Islamic Azad University,
Najafabad, Iran
ah.asilian@sel.iaun.ac.ir

Hamidreza Sadrarhami
Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University,
Najafabad, Iran
sadrarhami@ieee.org

Mohammadreza Hemmati
Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University,
Najafabad, Iran
Mr.hemati@sco.iaun.ac.ir

Yaghoub Saberi
Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University,
Najafabad, Iran
y.saberi@sco.iaun.ac.ir

2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP) | 979-8-3503-8394-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/AISP61396.2024.10475265

***Abstract*— Deep learning (DL) , a dynamic subset of machine learning inspired by the human brain, has evolved into a transformative force, showcasing remarkable capabilities across diverse domains. Often referred to as the "Artificial Neural Network," DL involves neural networks with three or more layers. The integration of DL with the progression of Big Data has facilitated the deployment of intricate neural networks, enabling autonomous analysis of features and correlations within extensive datasets, whether structured or unstructured. Noteworthy is the heightened performance exhibited by DL algorithms when confronted with substantial volumes of data. This paper offers a comprehensive exploration of DL from multifaceted viewpoints, incorporating recent advancements in the field. Beyond elucidating the conceptual and theoretical foundations, the paper systematically addresses challenges, highlights advantages, and proposes solutions intrinsic to DL. Furthermore, it delves into future works in DL, identifying evolving trends and promising areas of exploration such as medical diagnostics, sports training, and energy-efficient approaches. The overarching goal of this paper is to contribute to the continued evolution and widespread application of DL across diverse sectors. By encapsulating the holistic landscape of DL, the research presented herein strives to provide a comprehensive resource for researchers, practitioners, and enthusiasts seeking insights into the current state and future directions of this transformative field.**

Keywords*— *Deep learning, Neural networks, Artificial intelligence, Machine learning, Supervised learning, Intelligent systems, Pattern recognition, Computer architecture, Object detection, Computer vision.

I. INTRODUCTION

In recent years, Deep learning (DL) has significantly influenced numerous domains, including autonomous operation, robotics, data science, autonomous vehicles, and medical diagnostics. One notable benefit of DL, in comparison to machine learning algorithms, is its capacity to leverage vast quantities of data, resulting in continued performance improvements [1]. DL, as a term, pertains to the training of large and sophisticated neural networks. The nomenclature and architecture of these networks are inspired by the human brain, aiming to imitate the communication between biological neurons [2-4].

Artificial Neural Networks (ANNs) are utilized in machine learning and DL applications as multi-layer, fully-connected neural networks. ANNs consist of an initial layer for input, one or multiple intermediate layers that are called hidden layers, and an output layer [5]. The general structure of an ANN can be visualized in Fig. 1 [6]. Each layer consists of some nodes, which establish connections with other nodes, incorporating weight and bias values. Algorithms are employed to analyze relationships within a given dataset, providing optimal outcomes when the input data undergoes changes [7]. Firstly, the input layer is initialized and the weights are allocated. The weights assigned to each variable are indicative of their respective degrees of significance, with higher weights signifying a greater impact on the output in comparison to the remaining inputs. Subsequently, all inputs are multiplied by their respective weights and the bias value [8]. An activation function is then applied to the resulting output, determining the predicted value. If the output of a node surpasses a predefined threshold, the node is activated, transmitting data to the subsequent layer of the network. This process, in which data is passed from one layer to the next, characterizes a feedforward neural network. ANNs are trained by employing a large volume of data, allowing the functionality to be suitably adjusted to yield the desired output [9]. After making recognizing patterns and trends and guessing predicted values, an appropriate loss function is applied to quantify the discrepancy between the predicted and actual values. Additionally, a technique known as backpropagation is employed, utilizing algorithms like gradient descent to identify errors. Backpropagation iteratively modifies the weights and bias values of the function, moving backward through the layers to train the model over a specified number of iterations [10-11].

DL and ANNs are two intertwined concepts that have revolutionized the field of artificial intelligence (AI) [12]. DL has emerged as a powerful technique for training complex models capable of extracting intricate patterns and making accurate predictions from big amounts of data. ANNs serve as the foundation for DL algorithms, enabling the development of highly sophisticated models [13]. DL, a subset of machine learning, centers on training Deep Neural Networks (DNNs) with multiple layers to learn hierarchical representations of data. This enables them to capture intricate patterns and relationships within the data. DL models employ an iterative process to automatically learn features from raw data. DL's effectiveness in areas like speech recognition, natural language processing, pattern recognition, computer vision, and image processing is due to its capability of extracting sophisticated representations from raw data [14]. ANNs are composed of interconnected nodes or artificial neurons arranged in layers. Each neuron processes incoming signals, applies an activation function, and produces an output. ANNs are highly adaptable and can learn from data through a process known as training. During training, the

weight and bias values of the neurons are adjusted to optimize the performance of ANN [15]. They exhibit the ability to generalize from training examples and make predictions or decisions on unseen data. DL mainly focuses on the training of DNNs, while machine learning includes a broader range of algorithms and techniques [16]. However, DL stands out for its ability to learn hierarchical representations of data through multiple layers so that it can get insight from large-scale data, complex patterns, and high-dimensional representations. ANNs form the backbone of DL with their multilayer architecture that allows them the creation of deep structures capable of learning hierarchical representations [17]. The presence of concealed layers in ANNs facilitates the process of extracting progressively more abstract characteristics. This hierarchical representation learning is a fundamental aspect of DL, facilitating the modeling of complex correlations and improving accuracy [18]. DL employs various techniques within ANNs to enhance performance. Some of the types of neural networks used for various tasks are Convolutional Neural Networks (CNNs), which process images and videos, Recurrent Neural Networks (RNNs), which analyze sequential data, and Generative Adversarial Networks (GANs), which create realistic data [19]. These techniques leverage the power of ANNs to capture intricate patterns and structures in data, enabling advanced applications such as image processing, speech synthesis, and natural language understanding [20]. This manuscript is structured in the following manner: Initially, a comprehensive examination of DL methodologies is conducted, where various approaches and their respective applications are thoroughly explored. Subsequently, an analysis of the fundamental elements constituting DNNs is presented, shedding light on their functional mechanisms. This is followed by a detailed review of diverse DL architectures, delving into their design aspects, performance characteristics, and areas of application. The ensuing section offers a comparative analysis of the most prevalent methods in the field. The paper culminates with a contemplative overview of the impact these technologies wield in the broader spectrum of data analytics and AI. It also provides foresight into the future trajectories and potential challenges within the realm of DL.

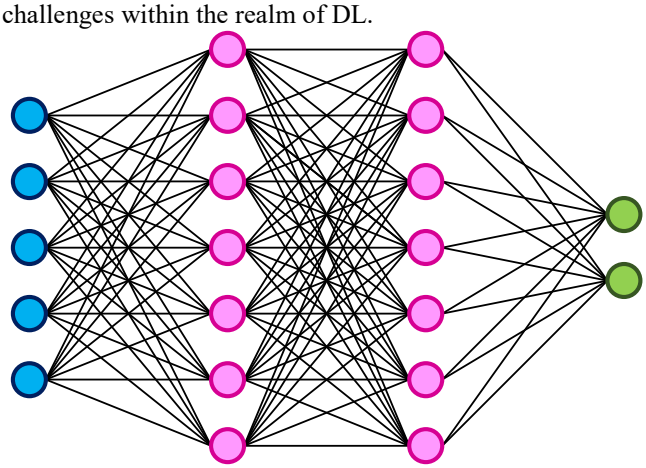

Fig. 1. Structure of an ANN including two hidden layers [6].

II. Deep Learning Approaches

DL encompasses a range of approaches that govern how the learning process unfolds. These approaches can be categorized into distinct types, including supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning. Each approach has its own fundamental concepts and principles, offering unique insights into the learning process. The following section delves into each of these approaches, exploring their fundamental concepts and principles.

A. Supervised Learning

Supervised learning is an approach to learning that relies on data with known labels. It involves training models to generate desired outputs based on a labeled dataset. This dataset consists of input data paired with corresponding correct and incorrect outputs, enabling the model to refine its predictions over time [21]. Specifically, for each input data point x, there exists a corresponding set of output values y that can be mapped as ($x \rightarrow y$). The algorithm assesses its performance using a loss function and iteratively adjusts its parameters until the error is minimized to an acceptable level [22]. DNNs, CNNs, and RNNs are examples of neural network architectures that utilize this supervised learning technique.

B. Semi-supervised Learning

Learn from datasets that contain a mix of labeled and unlabeled data can be viable by semi-supervised learning. It trains models using a limited amount of labeled data along with a large amount of unlabeled data, resulting in improved accuracy compared to unsupervised learning, but without requiring the same level of time and resources as supervised learning. This approach is sometimes utilized in RNNs and GANs to enhance their performance in certain scenarios [23].

C. Unsupervised Learning

Unsupervised learning is a learning approach where a model explores patterns in a dataset without relying on labeled data and minimal human intervention [24]. It differs from supervised learning methods like classification or regression, as it only has access to the input data and must uncover meaningful patterns. This technique is employed in GANs, Restricted Boltzmann Machines (RBMs), and Autoencoders to facilitate learning without explicit labels.

D. Reinforcement Learning

Reinforcement learning is a technique that relies on rewards and punishments. It follows an iterative process where an algorithm aims to maximize a value by considering the rewards associated with accurate actions. In an environment, an agent generates predictions for specific situations, and if the prediction is close to the actual value (resulting in a low loss), the agent is rewarded and progresses to the next task. In contrast, if the prediction is incorrect, the agent is penalized, leading to an increase in the loss. Through these positive and negative feedback signals, the agent begins to learn to perform better in the given environment [25].

III. Artificial Neural Networks

The input layer in ANN is the first layer that receives the raw data as input values. The hidden layer refers to any layer located between the input and output layers. The training, validating, and testing processes normally happen at this stage. The output layer is responsible for producing the predicted values [26].

A. Logistic Regression

Logistic regression is a commonly used machine learning algorithm that is specifically intended for solving binary classification problems in supervised learning. Its objective is to estimate the output value $\hat{y}$ based on input features $X=\{x_1, x_2,..., x_n\}$. For instance, when dealing with image data, such as a picture of a cat, the goal is to predict the probability of the presence of a cat in that image [27].

$$\hat{y} = P(y|X) \tag{1}$$

A potential approach for predicting $\hat{y}$ involves utilizing a linear function represented by Eq. (2):

$$\hat{y} = wX + b \tag{2}$$

Nevertheless, using this function alone may not yield desirable results since the predicted values are expected to be within the range of 0 and 1. Consequently, employing a

Sigmoid activation function becomes necessary to transform the linear regression equation and ensure that the output value falls between 0 and 1.

$$\hat{y} = \sigma(wX + b) \tag{3}$$

In logistic regression, it is essential to choose appropriate values for the parameters w and b so that the predicted value of $\hat{y}$ becomes a reasonable approximation of the true value of y [28].

B. Activation Function

The activation function plays a prominent role in determining the output values within neural network layers. If the activation function is omitted from the algorithm, a linear function will be obtained. Consequently, regardless of the number of hidden layers incorporated into the model, the activation remains linear, which is not effective in the solution of complex problems. To overcome this limitation, several non-linear activation functions have been developed, which are described below [29].

a) Sigmoid

The *sigmoid* activation function is utilized in the output layer and is particularly suitable for binary classification tasks as it restricts values to the range of [0,1]. The equation and corresponding diagram (Fig. 2) of the sigmoid activation function are presented below [30].

$$a = \left(1/(1+e^{-z})\right) \tag{4}$$

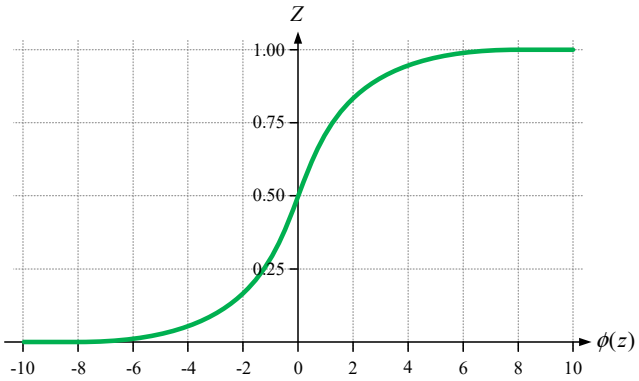

Fig. 2. The diagram of the *sigmoid* activation function for values ranging from [0,1].

b) Hyperbolic Tangent (tanh)

The *tanh* (hyperbolic tangent) function is similar to the *sigmoid* activation function but with a shifted range from [-1, 1]. It offers more stable gradient values compared to the *sigmoid* function, addressing the issue of vanishing gradients [31].

$$a = \left((e^{z} - e^{-z})/(e^{z} + e^{-z})\right) \tag{5}$$

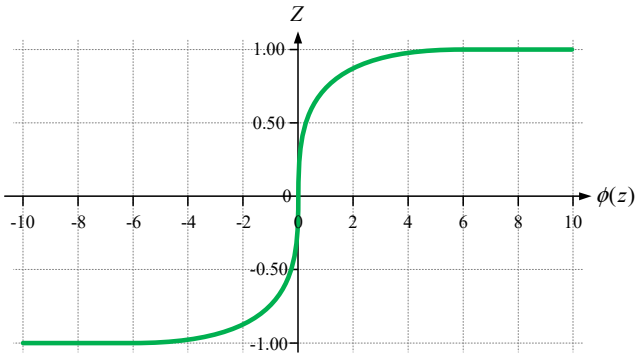

Fig. 3. The *tanh* diagram represents the range of values from [-1,1] for the *tanh* activation function.

c) Rectified Linear Unit (ReLU)

The *ReLU* activation function is widely used in neural networks. This function delivers the maximum value between zero and the input value, as described in Eq. (6). For negative inputs, the output is zero, while for positive inputs, the output equals the input value. The *ReLU* function has a derivative of one for positive inputs, which can speed up the training process of DNNs compared to other activation functions. Additionally, since *ReLU* has a constant value, computing error terms during training do not require extra time [32]. Figure 4 shows the diagram of the *ReLU* activation function.

$$a = \max(0, z) \tag{6}$$

d) Leaky Rectified Linear Unit (Leaky ReLU)

The Leaky Rectified Linear Unit (*Leaky ReLU*), which is depicted in Fig. 5, is an activation function that resembles *ReLU* but has a small slope for negative values. It is commonly employed in tasks that require sparse gradients, such as training GANs [33].

$$a = \max(0{,}01z, z) \tag{7}$$

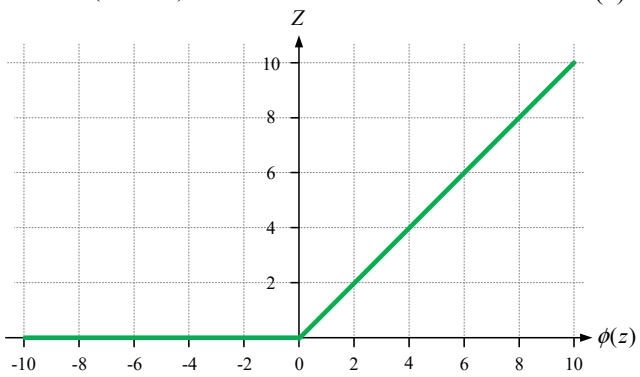

Fig. 4. The diagram of the *ReLU* activation function.

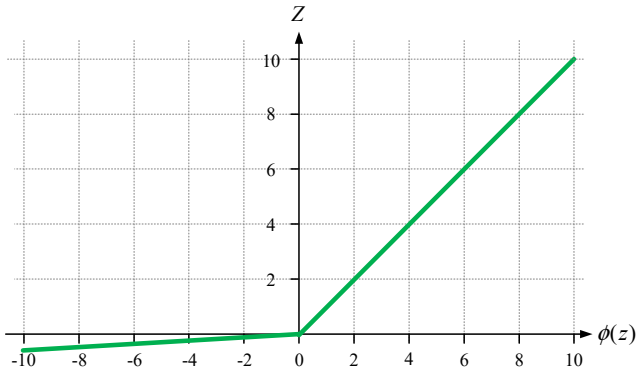

Fig. 5. *Leaky ReLU* activation function.

C. Loss Function

To assess the performance, the prediction error should be evaluated. The training process of a DL model sends effective input data to the corresponding nodes, adjusts the weight and bias values, makes predictions, and compares the forecasted values with the actual values to evaluate the performance of the prediction. Subsequently, the loss is computed, which is treated as a performance factor. As the loss has higher values, it implies a poorer overall performance. Loss functions are divided into two main categories: classification loss and regression loss. Classification loss is employed when the objective is to predict the output from distinct categorical values, such as recognizing handwritten digits ranging from (0–9). On the other hand, regression loss is utilized for problems involving the prediction of continuous values, like weather conditions or electricity prices based on specific features [34].

Mean Absolute Error: The mean absolute error (MAE) is a loss function that computes the average magnitude of the differences between the actual and predicted values. It disregards the sign of these differences to prevent interference between positive and negative values [35]. The MAE is defined as:

$$MAE = (1/N)\sum_{i=1}^{N}\left|y_{i,pred} - y_{i,true}\right| \quad (8)$$

Mean Squared Error: The mean squared error (MSE) is a loss function that calculates the average of the squared differences between the actual and predicted values. It provides a measure of how far the predictions are from the true values [36]. When the model makes no errors, the MSE is equal to 0.

$$MSE = (1/N)\sum_{i=1}^{N}\left|y_{i,pred}^2 - y_{i,true}^2\right| \quad (9)$$

Cross-Entropy: The cross-entropy function is a frequently employed loss function in tasks related to classification. The utilization of this method serves to measure the disparity among various probability distributions. When training a binary classifier, the appropriate choice is the binary cross-entropy function [37]. The equation for binary cross-entropy is given by Eq. (10):

$$L(y_i, p) = -(1/N)\sum_{i=1}^{N} y_i log\left(p(y_i)\right) + (1-y_i) log\left(1-p(y_i)\right) \quad (10)$$

The given equation involves the use of variables such as N, y_i, and $p_{o,c}$. Here, N denotes the count of training examples, y_i is a binary indicator that signifies whether the class label c is the accurate classification for observation o, and $p_{o,c}$ represents the anticipated probability of observation o for class c. Moreover, when the classification problems deal with more than two classes, the categorical cross-entropy function is employed. The categorical cross-entropy is a loss function that is employed in situations where a given instance can solely pertain to one among multiple feasible categories, and the model is required to ascertain the suitable category [38].

$$L(y_i, \hat{y}_i) = \sum_{i=1}^{C} y_i log\left(\hat{y}_i\right) \quad (11)$$

In the abovementioned equation, $\hat{y}_i$ corresponds to the predicted probability of the observation belonging to class c.

D. Gradient Descent

Gradient descent is a widely employed optimization algorithm utilized for training DL models and neural networks. The cost function, which assesses the performance of parameters w and b across the entire training set, is evaluated during each iteration within gradient descent to update these parameters [39]. This process continues until the gradient approaches zero. Throughout this iterative process, these parameters are adjusted to minimize the error as much as possible. Figure 6 illustrates how the parameters are updated at each iteration by shifting in the opposite direction of the gradient of loss function $L(w,b)$. In the backpropagation step, the downhill slope is tuned by computing the cost function derivatives considering w and b. This process will be continued until the minimum possible point reaches [40].

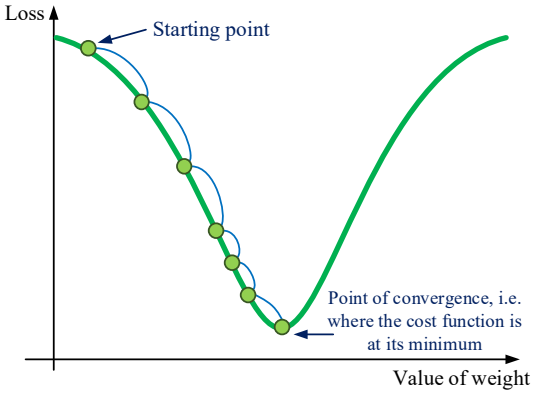

Fig. 6. The process of Gradient Descent, where the values gradually decrease over time.

The equation corresponding to the aforementioned description is presented below:

$$w = w - \propto \left(dJ(w,b)/dw\right) \ , \ b = b - \propto \left(dJ(w,b)/dw\right) \quad (12)$$

In the above, the learning rate α has a vital role in step size determining during each iteration of the training process. It directly influences the convergence speed and computational efficiency. Selecting an appropriate learning rate is important to ensure effective convergence. If the learning rate α is too small, the convergence will be slow, leading to longer training times and increased computational cost. On the other hand, if the learning rate α is too large, the model may fail to converge and overshoot the minimum [41]. Therefore, choosing an optimal learning rate is of the highest importance when training DNNs.

E. Forward Propagation

During the forward propagation step in a neural network, the input data is passed through the network in a sequential manner. Each hidden layer receives the data, applies the activation function to interpret it, and passes the transformed output to the next layer [42]. This process continues until the data reaches the output layer, where a prediction is made. Subsequently, the loss is calculated based on the prediction. The equations illustrating the forward propagation in a neural network are provided below.

$$Z^{[l]} = W^{\{l\}} A^{[l-1]} + b^{[l]} \quad (13)$$

$$A^{[l]} = g^{[l]}(Z^{[l]}) \quad (14)$$

The parameters $W^{\{l\}}$ and $b^{[l]}$ represent the specific weights and biases associated with layer l. The activation value $A^{[l]}$ corresponds to the output obtained from layer l, while $g^{[l]}$ denotes the activation function applied within that layer [43].

F. Backward Propagation

After computing the cost function for the training samples, the next step is to analyze how adjusting the weight and bias values and the way they impact the overall behavior of the neural network. This process, known as backward propagation, involves determining the cost function gradient with respect to other parameters in the model [44]. For each network weight or bias value, the partial derivative $\partial C/\partial w$ of the cost function L will be determined. This allows for gradual updates to the weights and biases, leading to a reduction in the cost function over multiple training iterations. To compute these derivatives, the first step is to calculate the cost function derivative with respect to the activation function a at layer l (as shown in Eq. (15)). Subsequently, the derivatives of w, b, and the activation of the previous layer are computed. The equations for backward propagation in layer l of a neural network are presented below.

$$dA^{[l]} = dL(a,y)/dA^{[l]} \quad (15)$$

$$dZ^{[l]} = dA^{[l]} \times g'^{[l]}(Z^{[l]}) \quad (16)$$

$$dW^{[l]} = \left(dZ^{[l]} A^{[l-1]} \cdot T\right)/m \quad (17)$$

$$db^{[l]} = \left(sum(dZ^{[l]})\right)/m \quad (18)$$

$$dA^{[l-1]} = \left(W^{[l]} \cdot T \times dZ^{[l]}\right) \quad (19)$$

Here, $L(a, y)$ is the cost function, $d_A{}^{[l]}$ denotes the activation value of layer l, $g'_A{}^{[l]}$ stands for the calculation of the original value $Z^{[l]}$ of the activation function g, $d_W{}^{[l]}$ and $d_b{}^{[l]}$ show the derivative values of the parameters w and b, and $d_A{}^{[l-1]}$ signifies the derivative value of the activation in the previous layer [45].

IV. Deep Learning Architectures

A. RNN

A RNN is a widely adopted neural network architecture that is particularly well-suited for handling sequential data. Unlike conventional neural networks, RNNs offer several advantages, primarily due to their ability to handle inputs and

outputs that vary in dimensions [46]. This flexibility allows RNNs to process data with varying lengths and structures, making them highly suitable for tasks such as natural language processing and speech recognition. One significant advantage of RNNs lies in their capacity to capture and share learned features across different positions within sequential data, such as text. This property enables RNNs to consider the context and dependencies between elements in the sequence, making them effective in tasks that require understanding and generating sequential information. By retaining memory of previous inputs, RNNs can retain important context and make informed predictions at each step of the sequence. Furthermore, the ability of RNNs to model temporal dependencies makes them well-suited for tasks such as time series analysis, where past observations are crucial for predicting future values. RNNs can capture patterns and trends in the temporal data, allowing them to make accurate predictions or generate new sequences based on the learned patterns. Several applications that involve sequence data are outlined below:

- *Speech recognition* (*sequence to sequence*): X represents a sequence of audio waveforms, while Y represents a corresponding sequence of textual data.
- *Music generation* (*one to sequence*): X can be either an empty input or an integer value, and Y indicates a sequence of waveforms that constitute the generated music.
- *Sentiment classification* (*sequence to one*): X shows a sequence of textual data, and Y denotes an integer rating ranging from 1 to 5, indicating the sentiment expressed.
- *DNA sequence analysis* (*sequence to sequence*): X refers to a sequence of DNA symbols, and Y represents the corresponding labels or annotations associated with the DNA sequence.
- *Machine translation* (*sequence to sequence*): X denotes a sequence of text in one language, while Y represents a sequence of text in another language, indicating the translation from one language to another [47].

If the network is learning text, $X^{<t>}$ represents the t^{th} word in a sentence and the m^{th} training example. Once the activation for the t^{th} word is calculated, the neural network proceeds to the next word in the sentence. Unlike a standard neural network, the output value $y^{<t>}$, in this case, depends not only on $X^{<t>}$ but also on the activation from the previous time step, $a^{<t-1>}$. As a result, the RNN transmits its activation to the following time step at each iteration. At time zero, $a^{<0>}$ is an artificially created activation typically initialized as a vector of all zeros to initiate the process. The RNN sequentially examines the data from left to right, and the parameters utilized for each time step are shared. This implies that when the RNN predicts $y^{<t>}$, it incorporates information not only from $X^{<t>}$ but also from $\{X^{<t>}, X^{<t-1>},..., X^{<1>}\}$. The equations for the basic RNN are provided below [48].

$$a^{\langle t\rangle} = g\left(W_{aa}\left[a^{\langle t-1\rangle}, W_{ax}{}^{\langle t\rangle}\right] + b_a\right) \tag{20}$$

$$\tilde{y}^{\langle t\rangle} = g\left(W_{ya}a^{\langle t\rangle} + b_y\right) \tag{21}$$

The activation value at time step t is denoted as $a^{<t>}$, g denotes the activation function used for the specific time step, $\tilde{y}^{<t>}$ shows the predicted value at time step t, W_{ax}, W_{aa}, and W_{ya} are weight parameters in the network, and b_a and b_y are bias values. Backpropagation in RNN is referred to as propagation through time, as the activation (a) is passed from one sequence element to another in a backward manner [49]. However, RNNs encounter a problem during the training process known as vanishing/exploding gradients. In DNNs, when computing the gradients during backpropagation, the gradient can explode if the weight is too large or become close to zero if the weight is too small. Consequently, the hidden layer near the input layer takes a considerable amount of time to learn meaningful information. To tackle this problem, two alterations to the RNN architecture were suggested: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) [50].

B. LSTM

In 1997, Hochreiter and Schmidhuber introduced a groundbreaking advancement to the basic RNN algorithm known as LSTM [51]. This modification was aimed at addressing the challenge of vanishing gradients, which hindered the effectiveness of traditional RNNs in capturing long-range connections and retaining temporal dependencies. LSTM significantly improves upon the hidden layer of the RNN, enabling it to capture and preserve important information across longer sequences. By addressing the vanishing gradient problem more effectively, LSTM overcomes the limitations of traditional RNNs and enhances their ability to model and understand sequential data. One key component of LSTM is the memory cell state (c), which plays a vital role in retaining relevant information from earlier sequences. As the LSTM progresses through the sequence, it can retain valuable knowledge about various aspects, such as determining whether the subject of a sentence is singular or plural. This memory retention capability enables LSTM to capture and utilize context-specific information, contributing to more accurate predictions and better understanding of sequential patterns. Each LSTM cell consists of three gates: the update gate, the forget gate, and the output gate. These gates regulate the flow of information within the neural network and control what information is stored and transmitted [52]. The update gate determines which new information should be included in the memory cell state, the forget gate decides what information should be discarded from the memory cell state, and the output gate regulates the information that gets propagated to the next step or output of the LSTM. By incorporating these gates, LSTM effectively manages and selectively processes information at each step of the sequence, allowing it to capture long-term dependencies and overcome the challenges posed by the vanishing gradient problem. The ability of LSTM to retain and utilize essential information over extended sequences has made it a powerful tool in various applications, including natural language processing, speech recognition, and machine translation. They play crucial roles in the LSTM equations, which are as follows:

$$\tilde{c}^{\langle t\rangle} = tanh\left(W_c\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_c\right) \tag{22}$$

$$\Gamma_u = \sigma\left(W_u\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_u\right) \tag{23}$$

$$\Gamma_f = \sigma\left(W_f\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_f\right) \tag{24}$$

$$\Gamma_o = \sigma\left(W_o\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_o\right) \tag{25}$$

$$c^{\langle t\rangle} = \Gamma_u \times \tilde{c}^{\langle t\rangle} + \Gamma_f \times c^{\langle t-1\rangle} \tag{26}$$

$$a^{\langle t\rangle} = \Gamma_o \times tanh\, c^{\langle t\rangle} \tag{27}$$

The update gate (Γ_u), the forget gate (Γ_f), and the output gate (Γ_o) are vectors that control various operations in the LSTM. The candidate value ($\tilde{c}^{<t>}$) represents a potential value to be added to the current memory cell state ($c^{<t>}$). The weight and bias values are parameters, while the time step value is denoted by $<t>$. The sigmoid function (σ) and *tanh* are activation functions, and the element-wise multiplication is denoted by $\times$. The candidate value ($\tilde{c}^{<t>}$) is a tensor with values ranging from [-1, 1]. The update gate (Γ_u), forget gate (Γ_f), and output gate (Γ_o) are calculated using the sigmoid activation function, ensuring that their values are always between [0, 1]. The update gate is responsible for ascertaining the degree to which the candidate value ($\tilde{c}^{<t>}$) ought to be incorporated into the cell state ($c^{<t>}$) for the subsequent time step. The forget gate is responsible for determining the extent to which the preceding cell state ($c^{<t-1>}$) ought to be preserved [53]. In the context of LSTM networks, it can be observed

that when the forget gate unit attains a value proximate to 0, the LSTM model will proceed to discard the previously stored state in the corresponding unit of the antecedent state. In contrast, when a unit within the forget gate exhibits a value proximal to 1, the LSTM model predominantly preserves the associated value within the stored state. Ultimately, the output gate serves to ascertain which information is transmitted as the output of the given time step. LSTM cell blocks retain relevant information from previous states, making them capable of handling long-term correlations [54].

C. GRU

The GRU is a modified variant of the LSTM model that offers a streamlined and simplified architecture [55]. Introduced as an alternative to LSTM, GRU retains the ability to capture long-term dependencies and address the vanishing gradient problem while reducing the complexity of the model. One notable distinction between GRU and LSTM is the absence of a separate memory cell state. Instead, GRU utilizes a hidden state that combines the roles of both the hidden state and the memory cell state in LSTM. This simplification leads to a more compact architecture with fewer parameters and computations, making GRU particularly attractive in scenarios where computational resources are limited. Similar to LSTM, GRU also incorporates gating mechanisms to control the flow of information within the network. It consists of two gates: the update gate and the reset gate. The update gate determines how much of the previous hidden state should be retained and how much of the new information should be integrated, while the reset gate controls how much of the past information should be neglected. These gates enable GRU to selectively update and forget information, allowing it to capture relevant context and maintain a balance between retaining important information and discarding less relevant details. The simplified design of GRU makes it easier to train and computationally efficient compared to LSTM. It has been observed that GRU can achieve performance comparable to LSTM in various tasks, such as language modeling, machine translation, and speech recognition, while requiring fewer resources. This makes GRU a particularly appealing choice in scenarios where efficiency and simplicity are prioritized without compromising on the ability to capture long-term dependencies in sequential data. In recent years, GRU has gained popularity and has been widely adopted in many DL applications. Its simplicity, effectiveness, and efficiency make it a valuable tool for modeling sequential data, and it continues to be an active area of research and development in the field of RNNs. In contrast to LSTM, the memory cell $c^{<t>}$ in GRU is equivalent to the activation $a^{<t>}$. Additionally, GRU does not include the output gate and the forget gate found in LSTM. Instead, GRU incorporates another gate called Γ_r within its cell to calculate the candidate value c. This gate determines the relevance of the previous memory cell $c^{<t-1>}$ to the current cell $c^{<t>}$.

$$\tilde{c}^{\langle t\rangle} = tanh\left(W_c\left[\Gamma_r \times c^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_c\right) \tag{28}$$

$$\Gamma_u = \sigma\left(W_u\left[c^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_u\right) \tag{29}$$

$$\Gamma_r = \sigma\left(W_r\left[c^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_r\right) \tag{30}$$

$$c^{\langle t\rangle} = \Gamma_u \times \tilde{c}^{\langle t\rangle} + \left(1-\Gamma_f\right)\times c^{\langle t-1\rangle} \tag{31}$$

$$a^{\langle t\rangle} = c^{\langle t\rangle} \tag{32}$$

The update gate (Γ_u) in the GRU cell determines whether to replace the current memory cell $c^{<t>}$ with the candidate value $\tilde{c}^{<t>}$. In particular, if $\tilde{c}^{<t>} = 1$, then Γ_u equals 1, resulting in $c^{<t>} = \tilde{c}^{<t>}$. This approach differs from LSTM, as GRU does not employ a separate gate (Γ_f) to compute $c^{<t-1>}$. Instead, the value of $c^{<t-1>}$ is obtained by subtracting Γ_u from 1 (as shown in Eq. (31)). In summary, GRU is a simplified variant of the more complex LSTM model, both of which address the problem of vanishing gradients [56]. GRU offers advantages such as utilizing two gates and faster computation, enabling the construction of larger networks. However, LSTM is considered more robust and flexible due to its utilization of three gates.

D. CNNs

A CNN is a specialized type of neural network designed for analyzing image inputs, recognizing objects within images, and capturing spatial and temporal correlations effectively [57]. One of the key advantages of CNNs over traditional feed-forward neural networks is their ability to leverage associated filters to better understand the complex structures present in an image. By applying filters to different regions of an input image, CNNs can learn and assign adjustable weight and bias values specific to those regions, enabling them to identify objects and patterns accurately. This localized processing allows CNNs to capture intricate details and extract meaningful features from images. The architecture of a CNN offers several benefits. Firstly, CNNs have a reduced number of parameters compared to fully-connected networks, making them computationally efficient and easier to train. Additionally, CNNs utilize weight sharing, which means the same set of weights is reused across different regions of an image, enhancing generalization and reducing the risk of overfitting. This weight sharing property enables CNNs to efficiently learn and recognize patterns in various parts of an image. The layers in a CNN can be categorized into three primary types: convolutional layers, pooling layers, and fully-connected layers. Convolutional layers perform the filtering operation by applying filters to the input image, capturing important features and generating feature maps. Pooling layers downsample the feature maps, reducing their spatial dimensions while preserving the most salient information. Finally, fully-connected layers connect all the neurons from the previous layer to the subsequent layer, enabling high-level feature extraction and classification. During the convolution step, filters are applied to the input image, and a bias term is added to each filter. This process allows the network to detect and emphasize specific features present in the image, such as edges, textures, or shapes. By adjusting the weights and biases of the filters through the learning process, CNNs can effectively learn to recognize complex objects and patterns in images. If the input size is $n \times n \times c$ and the filter size is $f \times f \times c$, with a total of n_f filters, the resulting output size would be $(n-f+1)\times(n-f+1)\times n_f$. The CNN employs a ReLU transformation to characterize nonlinearity after each convolution process. The pooling layers in a CNN extract essential features from the image and reduce the parameter size. These parameters are then flattened and fed into fully-connected layers. The fully-connected layers perform similar tasks as in standard ANNs, aiming to produce class scores based on the activated features for classification purposes [58].

E. GANs

A GAN is a powerful category of DNNs designed to generate new data that resembles the training dataset. GANs consist of two interconnected networks, namely the generator and the discriminator, that work in a competitive manner to improve the overall performance of the network [59]. The generator network is responsible for generating new samples that mimic the patterns and characteristics present in the training data. It takes random input, often referred to as the latent space or noise vector, and transforms it into a data sample that resembles the original dataset. The generator learns to produce increasingly realistic samples by adjusting its parameters through the training process. On the other hand, the discriminator network acts as a binary classifier,

distinguishing between the generated samples from the generator and the real samples from the training dataset. It learns to differentiate between real and fake samples by optimizing its parameters. The discriminator's objective is to correctly identify the generated samples as fake while accurately classifying the real samples. The generator and discriminator networks are trained together in an adversarial manner. The generator aims to generate samples that are so realistic that the discriminator cannot distinguish them from real data. Conversely, the discriminator aims to become more proficient at distinguishing between real and generated samples. This adversarial training process creates a competition between the generator and the discriminator, leading to a continual improvement in both networks. As the generator learns to produce more authentic samples, the discriminator becomes more adept at detecting subtle differences, resulting in a feedback loop that drives the quality of generated data towards that of the real dataset. GANs have demonstrated remarkable success in various domains, including image synthesis, text generation, and music composition. They have been used to generate realistic images, create unique artwork, enhance data augmentation techniques, and even generate deepfake videos. GANs have opened up new possibilities for generating novel data and have become an exciting area of research in the field of DL.

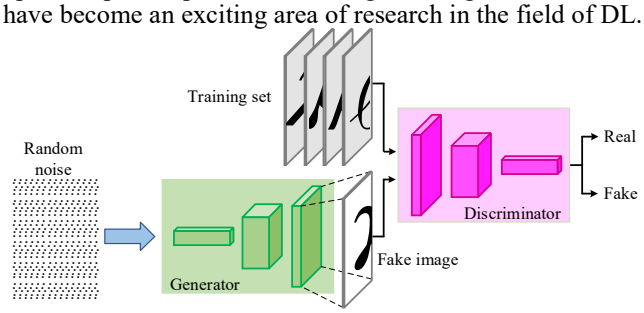

Fig. 7. The architecture of GANs.

As shown in Fig. 7, the generator component of the system takes a random noise with a normal distribution as input and generates data, typically in the form of an image. Simultaneously, both the generated fake samples and the actual training set of real data are randomly presented to the discriminator network. It is important to note that the discriminator has no prior knowledge of whether a given input originated from the generator or the training set. Consequently, the primary objective of the discriminator is to distinguish between real and manipulated data. Once the discriminator makes its prediction, a suitable loss function, such as binary cross-entropy, is used to determine the loss. Then, during the backpropagation process, the weights of the discriminator are updated to improve its performance. It is worth mentioning that the generator and discriminator are two distinct neural networks that require individual training, but they also interact and influence each other [60]. As a result, they cannot be trained in complete isolation from one another. Over the course of time, the output becomes increasingly realistic, and it becomes more proficient at deceiving the discriminator. Finally, the outputs become so authentic and sensible that the discriminator is unable to differentiate them from genuine samples.

V. Analysis of Deep Learning Architectures: Advantages, Challenges, and Solutions

DL architectures are fundamental to the progress in the field of AI, with each architecture offering unique advantages and limitations. This comparative analysis aims to dissect the characteristics of six prominent DL architectures: ANNs, CNNs, RNNs, LSTMs, GRUs, and GANs, providing insights into their optimal application domains.

A. ANNs:

ANNs are the cornerstone of DL, boasting flexibility in handling complex patterns and noisy datasets. Their robustness is attributed to their ability to learn non-linear relationships within data, which makes them highly versatile and applicable across a myriad of applications. Despite this, ANNs suffer from a lack of interpretability, often acting as "black boxes" which hinders transparency in decision-making processes. Additionally, they are prone to overfitting, especially when data is scarce or overly complex, which can impair their generalization capabilities [61].

* *Advantages:* ANNs exhibit broad applicability, showcasing their capability to model intricate and non-linear relationships within data. Their strength lies in the ability to generalize patterns from training data, enabling accurate predictions or classifications on new and unseen data. This versatility makes ANNs indispensable for various AI tasks across different domains.

* *Challenges:* The primary challenge associated with ANNs is their lack of transparency, often referred to as the "black box" problem. This means that understanding the decision-making process within the network can be challenging, limiting interpretability. Additionally, ANNs are prone to overfitting, where the model becomes too specialized in the training data, leading to poor performance on new, unseen data.

* *Solutions:* To address the interpretability challenge, attention mechanisms or Explainable AI (XAI) methods can be integrated. These techniques provide insights into how the model arrives at specific decisions, making it more transparent and understandable. To combat overfitting, regularization techniques such as dropout, early stopping, or pruning can be implemented. These methods introduce constraints during training to prevent the model from fitting noise in the data, thereby enhancing its generalization capabilities [62].

B. CNNs:

CNNs represent a leap forward in image and video recognition tasks, largely due to their proficiency in capturing and processing spatial hierarchical structures inherent in grid-like data. This attribute makes them especially powerful for tasks such as image classification and object detection [63]. However, their specialization also confines their use primarily to areas dealing with image data, and their performance comes at the cost of high computational demands, which can be a limiting factor for resource-constrained environments.

* *Advantages:* CNNs excel in processing data with a grid-like topology, particularly images. This is attributed to their proficiency in capturing spatial hierarchies and features through convolutional filters. The hierarchical feature extraction allows CNNs to recognize complex patterns and representations in visual data, making them highly effective for image-related tasks.

* *Challenges:* CNNs can be computationally intensive and demand substantial resources, posing challenges for deployment in low-resource settings. The complex architecture and the need for large-scale computations during training can limit their application in scenarios with constrained computational power.

* *Solutions:* To mitigate computational challenges, network pruning and weight sharing techniques can be employed. These methods reduce the size of the network by eliminating redundant connections or parameters, thereby lowering computational requirements. Efficient architectures like SqueezeNet or MobileNets are designed to provide comparable performance with reduced computational demands [64]. Additionally, transfer learning can be leveraged by utilizing pre-trained models on large datasets, reducing the need for extensive training on limited resources.

C. RNNs:

RNNs stand out in their ability to handle sequential data, providing a sense of memory that is essential for understanding context in time-series information or language. This makes them suitable for language modeling, speech recognition, and other sequence-dependent tasks. The main challenge in training RNNs arises from the vanishing gradient problem, which complicates the learning of long-term dependencies and can hinder their performance on longer sequences [65].

* Advantages: RNNs specialize in handling sequential data, showcasing their versatility in language processing, time series analysis, and other sequential tasks. The inherent memory mechanism within RNNs allows them to capture dependencies over time, making them suitable for tasks where context and sequence information are crucial.

* Challenges: RNNs encounter challenges such as the vanishing gradient problem, particularly when training deep networks for long sequences. The vanishing gradient problem hampers the network's ability to learn dependencies over extended periods, affecting its performance on tasks with long-term dependencies.

* Solutions: Mitigating the vanishing gradient problem can be achieved through gradient clipping and proper weight initialization. These techniques ensure that the gradients during backpropagation do not vanish or explode, facilitating more stable and effective training. Alternatively, more advanced architectures like LSTMs or GRUs are designed to address the vanishing gradient problem by allowing networks to selectively retain and update information, making them more adept at capturing long-term dependencies in sequential data [66].

D. LSTMs:

Designed as an evolution of RNNs, LSTMs address the issue of long-term dependencies by incorporating mechanisms that allow for better retention of information over extended sequences. This has broadened their applicability to a range of complex tasks that require understanding over longer periods [67]. The downside of LSTMs is their computational intensity, which translates to higher resource consumption, and the potential for overfitting when the available training data is limited.

* *Advantages:* LSTMs excel in managing long-range dependencies within sequential data. This ability is crucial for complex tasks such as machine translation, where context from distant parts of a sentence is essential, or in event prediction in time series data where historical information significantly impacts future events. LSTMs overcome the vanishing gradient problem associated with traditional RNNs, making them particularly effective for tasks requiring memory of past inputs.

* *Challenges:* LSTMs come with computational expenses, making them resource-intensive during both training and inference. Additionally, they are susceptible to overfitting, especially when trained on smaller datasets, which can compromise their generalization performance on new, unseen data.

* *Solutions:* To mitigate overfitting, regularization techniques tailored for LSTMs, such as dropout applied to the LSTM layers, can be employed. This prevents the network from relying too heavily on specific patterns present in the training data. For computational efficiency, stacked LSTMs or bidirectional LSTMs can be utilized. Stacked LSTMs involve using multiple LSTM layers on top of each other, enabling the model to learn hierarchical representations [68]. Bidirectional LSTMs process the input sequence from both forward and backward directions, capturing dependencies in both directions, often enhancing performance with a reduced number of parameters.

E. GRUs:

GRUs simplify the LSTM architecture while maintaining a comparable level of performance on many tasks. Their streamlined design allows for more efficient training, making them a compelling choice when computational resources or data are limited [69]. However, the simplification in GRUs may come at the cost of reduced capability in handling very complex tasks that require sophisticated memory management, where LSTMs might still hold an edge.

* *Advantages:* GRUs provide a simpler alternative to LSTMs with fewer parameters. This simplicity leads to faster training times and reduced computational complexity while maintaining competitive performance on various tasks. GRUs are particularly advantageous for scenarios where computational efficiency is crucial, and the task doesn't require nuanced long-term memory management.

* *Challenges:* GRUs may not perform as well as LSTMs on extremely complex sequence modeling tasks that demand precise memory handling. The simplified architecture, while advantageous for efficiency, might limit their capacity to capture intricate long-term dependencies within sequential data.

* *Solutions:* In situations where GRUs might struggle with complex tasks, hybrid models can be explored. Combining GRUs with other architectures, such as LSTMs or attention mechanisms, can enhance their ability to handle nuanced memory requirements. Alternatively, modifying the GRU architecture by introducing additional gates or mechanisms tailored to the specific task at hand can be a strategy to improve performance on complex sequence modeling tasks [70].

F. GANs:

GANs have revolutionized the generation of synthetic data by pitting two neural networks against each other: a generator that creates data and a discriminator that evaluates its authenticity. This adversarial process enables the production of highly realistic synthetic datasets [71]. Training GANs, however, is a delicate process that requires careful balancing of the generator and discriminator to prevent issues such as mode collapse, where the generator produces limited varieties of outputs.

* *Advantages:* GANs are renowned for their ability to generate highly realistic synthetic data. This capability is invaluable for tasks like data augmentation, where increasing the diversity of a dataset enhances the robustness of machine learning models, as well as for creative applications such as art creation and image synthesis.

* *Challenges:* Training GANs is notoriously challenging due to the need to balance two competing networks—the generator and the discriminator. Mode collapse, where the generator produces limited diversity in generated samples, is a common issue that can hinder the overall quality and diversity of the generated data.

* *Solutions:* To stabilize GAN training, Wasserstein loss can be employed as an alternative to traditional adversarial loss functions. Batch normalization helps in providing stability during training, and architectural innovations like progressive growing of GANs, where the generator and discriminator are grown gradually, can contribute to more stable training dynamics. Additionally, close monitoring of the training process and tuning hyperparameters can help strike the right balance between the generator and discriminator, mitigating mode collapse and ensuring the production of diverse and realistic synthetic data [72].

Table 1 provides a concise overview of the strengths, weaknesses, and ideal applications for each DL architecture, aiding in the comparative analysis of their characteristics.

TABLE I. COMPARATIVE ANALYSIS OF DEEP LEARNING ARCHITECTURES [73-76]

Architecture	Strengths	Weaknesses	Ideal Applications
ANNs	- Versatile and robust to noise	- Lack transparency, risk of overfitting	- Pattern recognition, predictive analytics
CNNs	- Excellent at spatial hierarchies, parameter efficiency	- Limited to grid data, computationally intense	- Image recognition, video analysis
RNNs	- Great for sequential data, context awareness	- Vanishing gradient problem, computationally complex	- Natural language processing, time-series analysis
LSTMs	- Learns long-term dependencies, flexible	- High training time and complexity, prone to overfitting	- Complex sequential tasks, predictive typing
GRUs	- Efficient, comparable performance to LSTMs	- Limited control, relatively new	- Sequence modeling in limited computational resources, real-time applications
GANs	- Generates high-quality, realistic data	- Difficult to train, risk of mode collapse	- Image/video generation, data augmentation

VI. FUTURE WORKS IN DEEP LEARNING

The recent developments in DL have opened up exciting avenues for future research, showcasing the potential of these algorithms in diverse fields. One promising area is the application of DL in medical diagnostics, where models leveraging ophthalmic data exhibit promise for screening and diagnosing systemic diseases [77]. Despite advancements, challenges in specificity and generalizability persist, paving the way for future research to refine these models for more robust clinical applications. Sports training is another domain witnessing the integration of DL, offering insights into the impact of training loads on athletes' biological systems [78]. This avenue presents opportunities to optimize training regimens and enhance physiological outcomes, suggesting a direction for further investigation into personalized and data-driven sports science. Innovations like Frequency Domain Diffractive Deep Neural Networks (D2NNs) signal a shift toward exploiting frequency domain information for improved data recovery in free-space optical communication systems [79]. Future works in this area may focus on refining and scaling these architectures for practical applications. The integration of fuzzy logic with DL to enhance algorithm performance is gaining traction. Future research could delve deeper into refining the synergy between DL and fuzzy systems, aiming to strike a balance between the robustness of DL and the nuanced decision-making capabilities of fuzzy logic [80].

Beyond this, the adaptation of Vision Transformers (ViTs) for biometric recognition presents exciting possibilities in various modalities, emphasizing the need for future research to explore the potential and limitations of ViTs in diverse biometric applications [81]. As energy efficiency becomes a critical concern, the development of lightweight DL accelerators for IoT devices holds promise. Future research could focus on addressing the challenges of energy consumption in DL, ensuring real-time object detection on edge devices with minimal environmental impact [82]. The emergence of adaptive frameworks like DeepAdapter for cross-platform web applications highlights the need for future research to optimize the performance of DL services in terms of latency, energy efficiency, and throughput. This paves the way for more accessible and practical DL applications in diverse web-based contexts [83]. In the realm of spectral snapshot compressive imaging and deep class-incremental learning, future works may aim to further enhance the speed, accuracy, and adaptability of these technologies [84-85]. Additionally, the ongoing focus on energy-efficient approaches throughout the DL lifecycle underscores the importance of sustainable AI practices, encouraging research into eco-friendly optimizations at every stage of the DL process [86]. These developments underscore the dynamic nature of DL research, pushing the boundaries of what is possible. As the field continues to evolve, addressing ethical considerations and ensuring accessibility will be essential to harness the full potential of these advanced technologies across various sectors.

VII. CONCLUSION

In the culmination of this research endeavor, a meticulous exploration into the foundational principles and architectures of DL has been conducted, transcending a mere overview. The research extends beyond theoretical underpinnings to systematically address challenges, highlight advantages, and propose intrinsic solutions within the realm of DL. Delving into diverse learning approaches, including supervised, unsupervised, semi-supervised, and reinforcement learning, the paper offers comprehensive insights. In addition to shedding light on the conceptual and theoretical foundations, the research meticulously provides a detailed examination of the key components crucial for training a DNNs. This includes in-depth discussions on loss functions, activation functions, and the gradient descent algorithm. The exploration further extends to elucidate the intricacies of forward and backward propagation steps, emphasizing their paramount importance in the training process. A nuanced understanding of various supervised DL techniques, such as ANNs, CNNs, and RNNs, is presented, offering practical considerations and distinctions in their applications. A thorough evaluation is then conducted on prominent types of RNNs, specifically LSTM networks and GRUs. Following this comprehensive analysis, the architecture of GANs is dissected, providing valuable insights into the intrinsic learning process within this network. The holistic exploration presented in this research extends beyond theoretical foundations to encompass practical considerations across diverse architectures and learning methodologies. In conclusion, this exhaustive examination contributes to the broader understanding and future development of DL. By identifying evolving trends and promising areas of exploration, such as medical diagnostics, sports training, and energy-efficient approaches, this research aims to pave the way for continued advancements and applications in various domains.

REFERENCES

[1] Sharifani, K., & Amini, M. (2023). Machine Learning and Deep Learning: A Review of Methods and Applications. *World Information Technology and Engineering Journal*, *10*(07), 3897-3904.

[2] Zuo, C., Qian, J., Feng, S., Yin, W., Li, Y., Fan, P., ... & Chen, Q. (2022). Deep learning in optical metrology: a review. *Light: Science & Applications*, *11*(1), 39.

[3] Lakshmanna, K., Kaluri, R., Gundluru, N., Alzamil, Z. S., Rajput, D. S., Khan, A. A., ... & Alhussen, A. (2022). A review on deep learning techniques for IoT data. *Electronics*, *11*(10), 1604.

[4] Suganyadevi, S., Seethalakshmi, V., & Balasamy, K. (2022). A review on deep learning in medical image analysis. *International Journal of Multimedia Information Retrieval*, *11*(1), 19-38.

[5] Zanjani, S. M., Shahinzadeh, H., Moradi, J., Fayaz-dastgerdi, M. H., Yaïci, W., & Benbouzid, M. (2022, October). Short-term Load Forecasting using the Combined Method of Wavelet Transform and Neural Networks Tuned by the Gray Wolf Optimization Algorithm. In *2022 Global Energy Conference (GEC)* (pp. 294-299). IEEE.

[6] Moradi, J., Shahinzadeh, H., & Khandan, A. (2017). A cooperative dispatch model for the coordination of the wind and pumped-storage generating companies in the day-ahead electricity market. *International Journal of Renewable Energy Research (IJRER)*, *7*(4), 2057-2067.

[7] Joshi, M., Bhosale, S., & Vyawahare, V. A. (2023). A survey of fractional calculus applications in artificial neural networks. *Artificial Intelligence Review*, 1-54.

[8] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Umar, A. M., Linus, O. U., ... & Kiru, M. U. (2019). Comprehensive review of artificial neural network applications to pattern recognition. *IEEE access*, *7*, 158820-158846.

[9] Wang, X., Lin, X., & Dang, X. (2020). Supervised learning in spiking neural networks: A review of algorithms and evaluations. *Neural Networks*, *125*, 258-280.

[10] Yang, G. R., & Wang, X. J. (2020). Artificial neural networks for neuroscientists: a primer. *Neuron*, *107*(6), 1048-1070.

[11] Lobo, J. L., Del Ser, J., Bifet, A., & Kasabov, N. (2020). Spiking neural networks and online learning: An overview and perspectives. *Neural Networks*, *121*, 88-100.

[12] Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., & Campbell, J. P. (2020). Introduction to machine learning, neural networks, and deep learning. *Translational vision science & technology*, *9*(2), 14-14.

[13] Chen, M., Challita, U., Saad, W., Yin, C., & Debbah, M. (2019). Artificial neural networks-based machine learning for wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, *21*(4), 3039-3071.

[14] Silvestrini, S., & Lavagna, M. (2022). Deep learning and artificial neural networks for spacecraft dynamics, navigation and control. *Drones*, *6*(10), 270.

[15] Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

[16] Li, Y. (2022, January). Research and application of deep learning in image recognition. In *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)* (pp. 994-999). IEEE.

[17] Nguyen, H. T., Li, S., & Cheah, C. C. (2022). A layer-wise theoretical framework for deep learning of convolutional neural networks. *IEEE Access*, *10*, 14270-14287.

[18] Tsuneki, M. (2022). Deep learning models in medical image analysis. *Journal of Oral Biosciences*, *64*(3), 312-320.

[19] Malhotra, P., Gupta, S., Koundal, D., Zaguia, A., & Enbeyle, W. (2022). Deep neural networks for medical image segmentation. *Journal of Healthcare Engineering*, *2022*.

[20] Chakraborty, S., & Mali, K. (2023). An overview of biomedical image analysis from the deep learning perspective. *Research Anthology on Improving Medical Imaging Techniques for Analysis and Intervention*, 43-59.

[21] Gu, S., & Yang, Y. (2020). A deep learning algorithm for the max-cut problem based on pointer network structure with supervised learning and reinforcement learning strategies. *Mathematics*, *8*(2), 298.

[22] Wang, R., Lei, T., Cui, R., Zhang, B., Meng, H., & Nandi, A. K. (2022). Medical image segmentation using deep learning: A survey. *IET Image Processing*, *16*(5), 1243-1267.

[23] Ouali, Y., Hudelot, C., & Tami, M. (2020). An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*.

[24] Tao, X., Gong, X., Zhang, X., Yan, S., & Adak, C. (2022). Deep learning for unsupervised anomaly localization in industrial images: A survey. *IEEE Transactions on Instrumentation and Measurement*.

[25] Liu, R., Nageotte, F., Zanne, P., de Mathelin, M., & Dresp-Langley, B. (2021). Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review. *Robotics*, *10*(1), 22.

[26] Chu, Y., Fei, J., & Hou, S. (2019). Adaptive global sliding-mode control for dynamic systems using double hidden layer recurrent neural network structure. *IEEE transactions on neural networks and learning systems*, *31*(4), 1297-1309.

[27] Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and KNN models for the text classification. *Augmented Human Research*, *5*, 1-16.

[28] Farrell, M. H., Liang, T., & Misra, S. (2021). Deep neural networks for estimation and inference. *Econometrica*, *89*(1), 181-213.

[29] Feng, J., & Lu, S. (2019, June). Performance analysis of various activation functions in artificial neural networks. In *Journal of physics: conference series* (Vol. 1237, No. 2, p. 022030). IOP Publishing.

[30] Langer, S. (2021). Approximating smooth functions by deep neural networks with sigmoid activation function. *Journal of Multivariate Analysis*, *182*, 104696.

[31] Shakiba, F. M., & Zhou, M. (2020). Novel analog implementation of a hyperbolic tangent neuron in artificial neural networks. *IEEE Transactions on Industrial Electronics*, *68*(11), 10856-10867.

[32] Wang, S. H., & Chen, Y. (2020). Fruit category classification via an eight-layer convolutional neural network with parametric rectified linear unit and dropout technique. *Multimedia Tools and Applications*, *79*, 15117-15133.

[33] Heydari, A. A., Thompson, C. A., & Mehmood, A. (2019). Softadapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions. *arXiv preprint arXiv:1912.12355*.

[34] Karimi, D., & Salcudean, S. E. (2019). Reducing the hausdorff distance in medical image segmentation with convolutional neural networks. *IEEE Transactions on medical imaging*, *39*(2), 499-513.

[35] Qi, J., Du, J., Siniscalchi, S. M., Ma, X., & Lee, C. H. (2020). On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Letters*, *27*, 1485-1489.

[36] Sabir, Z., Botmart, T., Raja, M. A. Z., Sadat, R., Ali, M. R., Alsulami, A. A., & Alghamdi, A. (2022). Artificial neural network scheme to solve the nonlinear influenza disease model. *Biomedical Signal Processing and Control*, *75*, 103594.

[37] Wu, M. T. (2022). Confusion matrix and minimum cross-entropy metrics based motion recognition system in the classroom. *Scientific Reports*, *12*(1), 3095.

[38] Freire, P. J., Prilepsky, J. E., Osadchuk, Y., Turitsyn, S. K., & Aref, V. (2022). Deep Neural Network-Aided Soft-Demapping in Coherent Optical Systems: Regression Versus Classification. *IEEE Transactions on Communications*, *70*(12), 7973-7988.

[39] Xue, Y., Wang, Y., & Liang, J. (2022). A self-adaptive gradient descent search algorithm for fully-connected neural networks. *Neurocomputing*, *478*, 70-80.

[40] Vijayalakshmi, K., Vijayakumar, K., & Nandhakumar, K. (2022). Prediction of virtual energy storage capacity of the air-conditioner using a stochastic gradient descent based artificial neural network. *Electric Power Systems Research*, *208*, 107879.

[41] Cheridito, P., Jentzen, A., Riekert, A., & Rossmannek, F. (2022). A proof of convergence for gradient descent in the training of artificial neural networks for constant target functions. *Journal of Complexity*, *72*, 101646.

[42] Lu, F., Liang, Y., Wang, X., Gao, T., Chen, Q., Liu, Y., ... & Liu, Y. (2022). Prediction of amorphous forming ability based on artificial neural network and convolutional neural network. *Computational Materials Science*, *210*, 111464.

[43] Ren, Y. M., Alhajeri, M. S., Luo, J., Chen, S., Abdullah, F., Wu, Z., & Christofides, P. D. (2022). A tutorial review of neural network modeling approaches for model predictive control. *Computers & Chemical Engineering*, 107956.

[44] Rajabi, A. M., Khodaparast, M., & Mohammadi, M. (2022). Earthquake-induced landslide prediction using back-propagation type artificial neural network: case study in northern Iran. *Natural Hazards*, *110*(1), 679-694.

[45] Geetha, V., Aprameya, K. S., & Hinduja, D. M. (2020). Dental caries diagnosis in digital radiographs using back-propagation neural network. *Health Information Science and Systems*, *8*, 1-14.

[46] Bouarara, H. A. (2021). Recurrent neural network (RNN) to analyse mental behaviour in social media. *International Journal of Software Science and Computational Intelligence (IJSSCI)*, *13*(3), 1-11.

[47] Dhruv, P., & Naskar, S. (2020). Image classification using convolutional neural network (CNN) and recurrent neural network (RNN): A review. *Machine Learning and Information Processing: Proceedings of ICMLIP 2019*, 367-381.

[48] Zhu, J., Jiang, Q., Shen, Y., Qian, C., Xu, F., & Zhu, Q. (2022). Application of recurrent neural network to mechanical fault diagnosis: A review. *Journal of Mechanical Science and Technology*, *36*(2), 527-542.

[49] Song, H., Al Khafaf, N., Kamoona, A., Sajjadi, S. S., Amani, A. M., Jalili, M., ... & McTaggart, P. (2023). Multitasking recurrent neural network for photovoltaic power generation prediction. *Energy Reports*, *9*, 369-376.

[50] Shu, W., Cai, K., & Xiong, N. N. (2021). A short-term traffic flow prediction model based on an improved gate recurrent unit neural network. *IEEE Transactions on Intelligent Transportation Systems*, *23*(9), 16654-16665.

[51] Huang, R., Wei, C., Wang, B., Yang, J., Xu, X., Wu, S., & Huang, S. (2022). Well performance prediction based on Long Short-Term Memory (LSTM) neural network. *Journal of Petroleum Science and Engineering*, *208*, 109686.

[52] Ma, M., Liu, C., Wei, R., Liang, B., & Dai, J. (2022). Predicting machine's performance record using the stacked long short-term memory (LSTM) neural networks. *Journal of Applied Clinical Medical Physics*, *23*(3), e13558.

[53] Yadav, V., Verma, P., & Katiyar, V. (2023). 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*, *15*(2), 759-772.

[54] Setyanto, A., Laksito, A., Alarfaj, F., Alreshoodi, M., Oyong, I., Hayaty, M., ... & Kurniasari, L. (2022). Arabic language opinion mining based on long short-term memory (LSTM). *Applied Sciences*, *12*(9), 4140.

[55] Cho, M., Kim, C., Jung, K., & Jung, H. (2022). Water level prediction model applying a long short-term memory (lstm)–gated recurrent unit (gru) method for flood prediction. *Water*, *14*(14), 2221.

[56] Zhang, W., Li, H., Tang, L., Gu, X., Wang, L., & Wang, L. (2022). Displacement prediction of Jiuxianping landslide using gated recurrent unit (GRU) networks. *Acta Geotechnica*, *17*(4), 1367-1382.

[57] Zhang, Q., Xiao, J., Tian, C., Chun-Wei Lin, J., & Zhang, S. (2023). A robust deformed convolutional neural network (CNN) for image denoising. *CAAI Transactions on Intelligence Technology*, *8*(2), 331-342.

[58] Ahlawat, S., Choudhary, A., Nayyar, A., Singh, S., & Yoon, B. (2020). Improved handwritten digit recognition using convolutional neural networks (CNN). *Sensors*, *20*(12), 3344.

[59] Xun, S., Li, D., Zhu, H., Chen, M., Wang, J., Li, J., ... & Huang, P. (2022). Generative adversarial networks in medical image segmentation: A review. *Computers in biology and medicine*, *140*, 105063.

[60] Pan, T., Chen, J., Zhang, T., Liu, S., He, S., & Lv, H. (2022). Generative adversarial network in mechanical fault diagnosis under small sample: A systematic review on applications and future perspectives. *ISA transactions*, *128*, 1-10.

[61] Prieto, A., Prieto, B., Ortigosa, E. M., Ros, E., Pelayo, F., Ortega, J., & Rojas, I. (2016). Neural networks: An overview of early research, current frameworks and new challenges. *Neurocomputing*, *214*, 242-268.

[62] Abdolrasol, M. G., Hussain, S. S., Ustun, T. S., Sarker, M. R., Hannan, M. A., Mohamed, R., ... & Milad, A. (2021). Artificial neural networks based optimization techniques: A review. *Electronics*, *10*(21), 2689.

[63] McCann, M. T., Jin, K. H., & Unser, M. (2017). Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*, *34*(6), 85-95.

[64] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*.

[65] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.

[66] Tsantekidis, A., Passalis, N., & Tefas, A. (2022). Recurrent neural networks. In *Deep learning for robot perception and cognition* (pp. 101-115). Academic Press.

[67] Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, *53*, 5929-5955.

[68] Lindemann, B., Müller, T., Vietz, H., Jazdi, N., & Weyrich, M. (2021). A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, *99*, 650-655.

[69] Zulqarnain, M., Ghazali, R., Hassim, Y. M. M., & Aamir, M. (2021). An enhanced gated recurrent unit with auto-encoder for solving text classification problems. *Arabian Journal for Science and Engineering*, *46*(9), 8953-8967.

[70] BATUR DİNLER, Ö., & Aydin, N. (2020). An optimal feature parameter set based on gated recurrent unit recurrent neural networks for speech segment detection. *Applied Sciences*, *10*(4), 1273.

[71] Alqahtani, H., Kavakli-Thorne, M., & Kumar, G. (2021). Applications of generative adversarial networks (gans): An updated review. *Archives of Computational Methods in Engineering*, *28*, 525-552.

[72] Saxena, D., & Cao, J. (2021). Generative adversarial networks (GANs) challenges, solutions, and future directions. *ACM Computing Surveys (CSUR)*, *54*(3), 1-42.

[73] Mehrish, A., Majumder, N., Bharadwaj, R., Mihalcea, R., & Poria, S. (2023). A review of deep learning techniques for speech processing. *Information Fusion*, 101869.

[74] Hakim, M., Omran, A. A. B., Ahmed, A. N., Al-Waily, M., & Abdellatif, A. (2023). A systematic review of rolling bearing fault diagnoses based on deep learning and transfer learning: Taxonomy, overview, application, open challenges, weaknesses and recommendations. *Ain Shams Engineering Journal*, *14*(4), 101945.

[75] Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, *40*, 100379.

[76] Le, Q., Miralles-Pechuán, L., Kulkarni, S., Su, J., & Boydell, O. (2020). An overview of deep learning in industry. *Data Analytics and AI*, 65-98.

[77] Rana, Meghavi, and Megha Bhushan. "Machine learning and deep learning approach for medical image analysis: diagnosis to detection." *Multimedia Tools and Applications* 82.17 (2023): 26731-26769.

[78] Li, S., & Bai, Y. (2022). Deep learning and improved HMM training algorithm and its analysis in facial expression recognition of sports athletes. *Computational Intelligence and Neuroscience*, *2022*.

[79] Song, M., Li, R., & Wang, J. (2023). Only frequency domain diffractive deep neural networks. *Applied Optics*, *62*(4), 1082-1087.

[80] Sierra-Garcia, J. E., & Santos, M. (2022). Deep learning and fuzzy logic to implement a hybrid wind turbine pitch control. *Neural Computing and Applications*, *34*(13), 10503-10517.

[81] Garcia-Martin, Raul, and Raul Sanchez-Reillo. "Vision Transformers for Vein Biometric Recognition." *IEEE Access* 11 (2023): 22060-22080.

[82] Sankaran, K. S., & Kim, B. H. (2023). Deep learning based energy efficient optimal RMC-CNN model for secured data transmission and anomaly detection in industrial IOT. *Sustainable Energy Technologies and Assessments*, *56*, 102983.

[83] Krause, Thorsten, et al. "Adaptive Cross-Platform Learning for Teachers in Adult and Continuing Education." *International Conference on Artificial Intelligence in Education*. Cham: Springer International Publishing, 2022.

[84] Meng, Z., Yuan, X., & Jalali, S. (2023). Deep unfolding for snapshot compressive imaging. *International Journal of Computer Vision*, *131*(11), 2933-2958.

[85] Zhang, Xiaohan, et al. "Deep class-incremental learning from decentralized data." *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[86] Mehlin, V., Schacht, S., & Lanquillon, C. (2023). Towards energy-efficient Deep Learning: An overview of energy-efficient approaches along the Deep Learning Lifecycle. *arXiv preprint arXiv:2303.01980*.