Olufisayo S Ekundayo\textsuperscript{1} and Absalom El-Shamir Ezugwu\textsuperscript{1}

\textsuperscript{1}Unit for Data Science and Computing, North West University

May 07, 2024

Abstract

Deep learning stands at the forefront of contemporary machine learning techniques and is well-known for its outstanding predictive accuracy, adaptability to data variability, and remarkable ability to generalize across diverse domains. These attributes have spurred rapid progress and the emergence of novel iterations within the discipline. Yet, this swift evolution often obscures the foundational breakthroughs, with even trailblazing researchers at risk of fading into obscurity despite their seminal contributions. This study aims to provide a historical narrative of deep learning, tracing its origins from the cybernetic era to its current state-of-the-art status. We critically examine the contributions of individual pioneer scholars who have profoundly influenced the development of deep neural networks under the taxonomy of supervised, unsupervised, and reinforcement learning. Furthermore, the study also discusses the trending deep neural network architectures, explaining their operational principles, confronting associated challenges, exploring real-world applications, and outlining potential future trajectories that could offer a starting point for aspiring researchers in the field.
Deep Learning: Historical Overview from Inception to Actualization, Models, Applications and Future Trends

Olufisayo S. Ekundayo\textsuperscript{a}, Absalom E. Ezugwu\textsuperscript{a}.

\textsuperscript{a}Unit for Data Science and Computing, North West University, 11 Huffman Street, Potchefstroom, 2520, North West, South Africa

Abstract

Deep learning stands at the forefront of contemporary machine learning techniques and is well-known for its outstanding predictive accuracy, adaptability to data variability, and remarkable ability to generalize across diverse domains. These attributes have spurred rapid progress and the emergence of novel iterations within the discipline. Yet, this swift evolution often obscures the foundational breakthroughs, with even trailblazing researchers at risk of fading into obscurity despite their seminal contributions. This study aims to provide a historical narrative of deep learning, tracing its origins from the cybernetic era to its current state-of-the-art status. We critically examine the contributions of individual pioneer scholars who have profoundly influenced the development of deep neural networks under the taxonomy of supervised, unsupervised, and reinforcement learning. Furthermore, the study also discusses the trending deep neural network architectures, explaining their operational principles, confronting associated challenges, exploring real-world applications, and outlining potential future trajectories that could offer a starting point for aspiring researchers in the field.

Keywords: Deep Learning, History, Supervised Learning, Unsupervised Learning, Reinforcement Learning

\textsuperscript{Email addresses: 55261884@mymwu.ac.za (Olufisayo S. Ekundayo), absalom.ezugwu@nwu.ac.za (Absalom E. Ezugwu)}
1. Introduction

Deep learning represents a descriptive algorithm within machine learning, characterized by its layered structure in which algorithms progressively learn. The input layer receives the input and passes through numerous hidden layers before generating predictions via the output layer. Throughout the 21st century, deep learning has received extensive acceptance across various disciplines. Its crucial role in enhancing intelligence within AI systems is undeniable, fueled by its ongoing advances in architectural designs and the broader scope of applications.

The inception of deep learning can be traced back to the novel work of McCulloch and Pitts in the 1940s (Carpenter, 1989; Skansi, 2018; Kamath et al., 2019). Drawing from cybernetics principles, McCulloch and Pitts proposed a neuron-based model that mimicked the functionality of animal brains. Initially developed in 1943, their model featured an input layer, a logical layer representing an activation function, and an output layer. The model served as the foundational architecture for artificial neural networks (Abraham, 2002). However, the linearly defined logical function of McCulloch and Pitts’ neurons limited the model’s applicability to linear problems. A notable variation of this model is Rosenblatt’s perceptron, which, like McCulloch and Pitts’ neuron, operates as a single-layer perceptron with non-learnable randomized weights, primarily designed for binary classification Kussul et al. (2001). The era of cybernetic artificial neural models spanned from 1940 to 1960 before experiencing a prolonged break due to limitations identified in the perceptron model Block (1970).

The concept of connectionism reignited interest in neural network projects marked by the introduction of the multilayer perceptron or feedforward model by Ivakhnenko et al. (1967). This model consists of three layers: the input layer, the hidden layer, and the output layer. The feedforward network operates as a unidirectional network without stochastic gradient descent. The introduction of stochastic gradient descent (Amari, 1972), a crucial tool in backpropagation for feedforward networks (Werbos, 2005), coupled with the flexibility in the number of hidden layers, fueled the proliferation of deep neural networks. However, this breakthrough was limited due to challenges such as dependency on large volumes of data, computational complexity, resource consumption, and prolonged computation time, which again led research in the field to another recess. Additionally, the emergence of the Support Vector Machine (SVM) by Cortes (1995) posed a formidable alternative,
demonstrating greater efficiency with reduced computational complexity and minimal data requirements. For over a decade, SVM dominated the field of machine learning until the resurgence of deep learning, driven by relentless research efforts in the field (Schmidhuber, 2015).

The resurgence of deep learning can be attributed to several factors, including modifications to its architecture and advancements in hardware technology, the availability of large datasets and increased awareness of big data. The invention of fast computing devices such as GPUs (Graphics Processing Units) and High-Performance Computing (HPC) systems, along with larger memory capacities, has significantly contributed to the acceleration of deep learning algorithms. Additionally, the development of optimization techniques has improved performance and mitigated the complexity of neural networks. According to Dean (2022), the success of deep learning models across various domains is consequential to progressive hardware designs and the availability of open-source software, which have encouraged numerous researchers to explore and utilize deep learning effectively.

The evolution of deep learning networks and their variations has been driven by the need to tailor models for specific tasks. For example, Convolutional Neural Networks (CNNs) are specifically designed for tasks in Computer Vision, Recurrent Neural Networks (RNNs) excel in processing sequential data, Generative Adversarial Networks (GANs) are used in gamification, and Deep Reinforcement Learning (DRL) models are applied in Autonomous Driving and Robotics.

The applications of deep learning extend beyond Computer Vision Lecun et al. (1998), encompassing domains such as Natural Language Processing (Li, 2017; Yang et al., 2019b), Speech Recognition (Kumar et al., 2018), Autonomous Driving (Elallid et al., 2022), and Recommender Systems Guidotti et al. (2021). The successes of deep learning in the 21st century can be attributed to its nonlinear nature, rendering it suitable for addressing a wide range of real-world problems. However, attributing the deep learning model to 21st-century discovery would misguide the research community. Deep learning had persevered through several phases of rejection, abandonment, and modification for many decades (Laith and Jinglan, 2021), but survived through the relentless works of some researchers who saw the prospect of the model and gave it a chance despite all odds (Schmidhuber, 2022). Moreover, given the widespread adoption of deep learning across various disciplines where machine learning is applicable, the architectural design is tailored to the specific machine learning problem at hand. For instance, while most
CNNs are designed for addressing supervised learning problems, Autoencoder and Adversarial networks are devised for unsupervised learning tasks, and Graph Neural Networks are specifically designed for semi-supervised learning (Sarker, 2021).

Without controversy, deep learning is a state-of-the-art machine learning model. Its acceptability has cut across different fields of study due to its high predictive accuracy and robustness to unforeseen challenges Fradkov (2020). Numerous studies in the literature have endeavored to explore deep learning comprehensively in recent years. For instance, Schmidhuber (2015); Yann et al. (2015); Wang et al. (2017); Chhabra and Goyal (2023) have contributed to the discussion on deep learning. Yao et al. (2019) provided a review of feature learning techniques in machine learning, encompassing manifold techniques and deep learning approaches. Nash et al. (2018) conducted a review study on the application of deep learning for the detection, planning, and modeling of deteriorating materials, while Ball et al. (2017) reviewed deep learning applications in remote sensing. Lara-Benítez et al. (2021) conducted an experimental review of various deep learning models using time series data, aiming to propose the most suitable models for time series environments.

On the other hand, Voulodimos et al. (2018); Sengupta et al. (2020); Laith and Jinglan (2021); Dong et al. (2021); Chai et al. (2021) focused solely on recently proposed deep learning models and their application to computer vision tasks. Ganatra and Patel (2018) discussed deep learning models and available development tools, while Cao et al. (2020) reviewed geometric deep learning techniques, focusing on graph network algorithms and their applications. Alaskar and Saba (2021) conducted a comparative study of deep learning and machine learning, delineating their strengths and weaknesses. Furthermore, Sarker (2021) categorized deep learning into supervised, unsupervised, and hybrid models. Abdel-Jaber et al. (2022) provided an overview of deep learning architectures adapted to healthcare data, outlining their design, advantages, disadvantages, and limitations. Zhou et al. (2020) presented an overview of deep learning activities in developing computer-aided devices for medical imaging analysis. Lastly, the recent work of (Ahmed et al., 2023) conducted a comprehensive study on recent deep model techniques, discussing their limitations and advantages in detail. Table 1 contains the summary of some review studies on deep learning.

The aforementioned studies primarily focused on deep network architecture within specific domains, lacking comprehensive discussions on the evo-
olution of deep learning, except for the work of (Schmidhuber, 2022). Additionally, many historical depictions of deep learning have been criticized for containing distorted information and failing to formally acknowledge the contributions of pioneer researchers in the field, thereby undermining the credibility of their findings (Schmidhuber, 2022).

This study aims to provide a detailed and comprehensive historical review of deep learning in the literature, covering over 70 years from the 20th century to the present. Our objective is to offer interested scholars profound insights into the development of deep learning and the technical intricacies surrounding its evolution from its inception to its current state. We will meticulously examine the contributions of individual pioneer scholars who have played significant roles in shaping the evolution of deep networks. Additionally, we will provide a thorough description of trending deep neural network models and their operational modalities. Furthermore, we will explore challenges that have historically undermined the performance of deep neural networks and discuss various approaches employed to mitigate these challenges over time. Similar to the approach taken by Samanta et al. (2023), this study will delve into diverse and prominent areas of deep learning applications. Moreover, we will engage in a comprehensive discussion on the future trends of deep models, aiming to guide scholars interested in enhancing deep learning performance.
Figure 1: McCulloch-Pitts Neuron Model
<table>
<thead>
<tr>
<th>Authors</th>
<th>Year</th>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yann et al. (2015)</td>
<td>2015</td>
<td>Thoroughly gave account of supervised and unsupervised deep learning models, and their applications in different appropriate areas.</td>
</tr>
<tr>
<td>Zhong et al. (2016)</td>
<td>2016</td>
<td>Captured some feature learning algorithms with respect to manifold learning and deep learning models</td>
</tr>
<tr>
<td>Wang et al. (2017)</td>
<td>2017</td>
<td>Overview of deep learning models, model description and applications</td>
</tr>
<tr>
<td>Ball et al. (2017)</td>
<td>2017</td>
<td>Deep learning methods and application to remote sensing</td>
</tr>
<tr>
<td>Zhou et al. (2020)</td>
<td>2017</td>
<td>Deep learning methods application to medical image analysis</td>
</tr>
<tr>
<td>Voulodimos et al. (2018)</td>
<td>2018</td>
<td>The review encompassed deep learning models' application to computer vision.</td>
</tr>
<tr>
<td>Ganatra and Patel (2018)</td>
<td>2018</td>
<td>Discussed some deep learning models and various available tools for its development.</td>
</tr>
<tr>
<td>Cao et al. (2020)</td>
<td>2020</td>
<td>Information about geometric deep learning techniques especially the graph network algorithms and their respective applications.</td>
</tr>
<tr>
<td>Hassaballah and Awad (2020)</td>
<td>2021</td>
<td>Deep learning models in computer vision environment</td>
</tr>
<tr>
<td>Dong et al. (2021)</td>
<td>2021</td>
<td>Information about deep learning model and applications</td>
</tr>
<tr>
<td>Chai et al. (2021)</td>
<td>2021</td>
<td>Captured some recent deep learning model and their application to computer vision.</td>
</tr>
<tr>
<td>Lara-Benítez et al. (2021)</td>
<td>2021</td>
<td>Performance of deep learning models in time series environment was revealed through experimental study.</td>
</tr>
<tr>
<td>Sarker (2021)</td>
<td>2021</td>
<td>Detail information on the category of deep learning into Supervised, Unsupervised, Hybrid learning.</td>
</tr>
<tr>
<td>Sengupta et al. (2020)</td>
<td>2021</td>
<td>Deep learning models in computer vision applications.</td>
</tr>
<tr>
<td>Alakbar and Sabo (2021)</td>
<td>2021</td>
<td>Compared different machine learning models to establish their pros and cons.</td>
</tr>
<tr>
<td>Abd-El-Jaber et al. (2022)</td>
<td>2022</td>
<td>Present details of deep learning model employed in the health sector, carefully illustrating their pros and cons.</td>
</tr>
<tr>
<td>Schmidthuber (2022)</td>
<td>2022</td>
<td>Historical evolution with credit assigned to pioneer researchers with innovation in the development of deep learning. Logical grouping of deep learning models into supervised and unsupervised classes.</td>
</tr>
<tr>
<td>Chhabra and Goyal (2023)</td>
<td>2023</td>
<td>Presented deep neural networks that cover basic building components, especially in the common convolutional neural network.</td>
</tr>
<tr>
<td>Talaei et al. (2023)</td>
<td>2023</td>
<td>Discussed deep models in Supervised, Unsupervised and Hybrid models.</td>
</tr>
<tr>
<td>Our Study</td>
<td>2024</td>
<td>Covers the historical evolution of deep learning, a description of some prominent models in deep learning taxonomy (supervised, unsupervised deep reinforcement learning), and recent advancements in the field.</td>
</tr>
</tbody>
</table>
2. Early Foundation of Neural Network

2.1. Early Beginning of Neural Networks

Artificial Neural Networks (ANNs) are modeled after the complex connections between billions of neurons in the human brain. This is emphasized by the works of (Magdi et al., 2014; Lin, 2014). Since artificial neural networks got their inspiration from the neuron connectivity of the brain, it is pertinent to provide a succinct analogy illustrating how the functionality of the human brain mirrors the creative processes observed in ANNs.

The human brain comprises fundamental units called neurons responsible for receiving and processing signals like sensations, emotions, movements, thoughts, and memories. Neurons have three primary observable components: dendrites, the cell body known as soma, and the axon. Dendrites resemble tree branches and help transmit signals to the cell body. The soma processes these signals before transmitting them to the axon, which outputs the processed signal to other neurons’ dendrites via connected synapses. The structure of a neuron is presented in Figure 1, the simple connection illustrated continues among billions of neurons before the responsible signal is eventually fired. Although the actual processes within the human brain are considerably more complex than what has been outlined here, interested readers may refer to sources like (Posner and Raichle, 1998; Müller and Knight, 2006; Bassett and Gazzaniga, 2011). This study provides a concise overview, focusing on the essential parts of neural structure and functions.

In 1920, Ising and Lenz developed a neural network architecture that resembles the human brain, with a connection from one node to another and a feedback connection leading back to the starting node. This architecture forms the foundation for all recurrent networks. Similarly, in 1943, McCulloch proposed a neural model with a linear logical function inspired by the cybernetic idea of developing a human-like system. This ideology later became the motivation for recent deep learning models.

2.2. McCulloch And Pitts Model

Neurologist McCulloch and logician Pitts joined forces to create the McCulloch-Pitts (M-P) model, an artificial neuron design that mimics the behavior of living neurons. The M-P model receives input signals in the 0,1 format from its inputs, just like biological neurons. These inputs are then combined and processed through a threshold operation to produce a singular output. The pictorial representation of the M-P model is available in Figure 2. M-P model...
is mathematically represented in the form of functions, assuming the possible set of input $x \in \{0,1\}$ the summation of input $x$ is given as $g(x)$ that is: $X \in (x_1, x_2, \ldots, x_n)$ then $g(x_n) = \sum_{i=1}^{n} x_n$. Therefore, the single output of the model:

$$y = \begin{cases} 1 & \text{if } g(x) \geq \theta \\ 0 & \text{if } g(x) < \theta \end{cases}$$

where $\theta$ is the model threshold.

The M-P model’s threshold parameter can be substituted with Boolean operators like OR, AND, or NOT, which makes it suitable for linearly separable Boolean functions. However, it could not effectively address the XOR Boolean function. As a result, its practical application in real-world problems is limited due to certain constraints. However, the M-P model is an essential tool primarily used to introduce fundamental concepts of neural networks. Apart from its educational value, the M-P model has been integrated into other neural network models to improve their functionality or create hybridized versions. In a study conducted by Zhang and Zhang (1999), they introduced a geometrically represented M-P model for visualization purposes, which was subsequently applied to the design of feedforward neural networks. Additionally, it was utilized to demonstrate the three-layer mapping capabilities of a feedforward model.

The habituation learning proposed by Bi et al. (2017) is M-P model-based.
Li et al. (2021) employed the M-P model to achieve their proposed orientation detection mechanism in detecting two-dimensional tasks. Ince (2004) used M-P neuron interaction to mitigate associative memory challenges in the Hopfield network. M-P model principle formed the basis for some hardware implementation (Flak, 2012; Nawrocki et al., 2014) for classification tasks. The cybernetic era conceived the innovation of the deep neural network we are celebrating today.

2.3. The Perceptron and its Contribution

In 1958, Rosenblatt (1958) introduced the concept of connectionism to neural networks. He proposed a pattern classifier that consisted of linear networks combined with an output threshold. The perceptron introduced by Rosenblatt can process a set of inputs. However, unlike the M-P model, these inputs are not restricted to boolean values only but can also include real values, making them more versatile and applicable. The Rosenblatt model has a unique feature of quantifying the importance of inputs. This is achieved through the linear combination of inputs and their corresponding weights. Furthermore, the perceptron introduced bias as a learning mechanism, allowing the model to adapt to adjust input weights. The linear combination of inputs, weights, and bias determines the output of the perceptron. If the result is positive, the output is +1; otherwise, it is 0. The diagrammatic representation of Rosenblatt’s perceptron model is presented in Figure 3.

Mathematically; assuming the set of input given as $x = \{x_1, x_2, \ldots, x_n\}$ and their respective weight as $w = \{w_1, w_2, \ldots, w_n\}$ the linear combination of the inputs $x$ and their weights $w$ and bias parameters $x_0$ and $w_0$:

$$g = \sum_{i=1}^{n} x_i w_i$$

Therefore $y = \begin{cases} 1 & \text{if } g(x) > 0 \\ 0 & \text{if } g(x) < 0 \end{cases}$

In addition to linear networks, (Rosenblatt, 1958) also explored the idea of multiple perceptron layers. However, his model was non-learning, as learning was only possible at the output layer. He suggested back-propagation as a possible learning technique that could be adopted into a neural network, but he was unable to implement it (Rosenblatt, 1969; Marvin and Seymour, 1969; Van Der Malsburg, 1986). analyzed the perceptron model and found that it was limited in its ability to handle non-linear functions, leading him to
Figure 3: Rosenblatt Perceptron Model

\[ g(x) = \sum_{i=1}^{n} x_i \]

- If \( y_{sum} \geq \theta \), output 1
- If \( y_{sum} < \theta \), output 0
suggest it should be abolished. This caused the first winter break in the field of artificial neural networks.

Rosenblatt’s perceptron model was known for its affinity for linearly separable tasks, making it suitable for binary classification, pattern recognition, fault diagnosis, control systems, signal processing, and educational purposes. Yoav and Robert E. (1999) proposed a linear classifier by combining Rosenblatt’s perceptron model with Helmbold and Warmuth’s leave-one-out method. They reported that their linear classifier performed comparably to Vapnik’s maximal-margin classifier on handwritten digit recognition tasks but with a simpler implementation.

Mendez Lucero et al. (2022) revisited the Rosenblatt perceptron model and modified it by replacing the linear combination function with analytic sinusoids, allowing for the formation of analytic signal representations in the function to be learned. They claimed that this modified Rosenblatt perceptron outperformed single hidden layer multilayer perceptrons in terms of computation speed, implementation simplicity, nonlinear boolean function learning, and image classification tasks.

Prior to the recent popularity of Deep Neural Networks, a shallow network was proposed by Adrien-Marie Lengendre in 1805 (Farebrother, 2001). The shallow network is similar to the architecture of modern linear neural networks in terms of the algorithm used, error function, and adaptive parameters (weights). It contains two layers: the input layer and the output layer. The input and the output layers were connected with connecting weights, and output is determined by the sum of the product of inputs and the connecting weights. The training of the networks embraced a supervised learning procedure such that the set of inputs with their expected outputs were provided, and the network weights were modified or adjusted to minimize the error between the network output and the expected outputs. The shallow learning method is often referred to as the least square error method or linear regression method. The least square method was accredited to Carl Friedrich Gauss and Adrien-Marie Legendre between the end of the 18th and the early 19th century (Stigler, 1981; Celmins, 1998; Singh, 2010), the notion of the least square method was conceived by Gauss, while the formal publication and application of the method was attributed to Legendre. The least square method has been a very useful tool in the field of machine learning for prediction and forecasting purposes (Hansen, 2008; Sulaimon, 2015; Dengen et al., 2018).
2.4. Limitation and Challenges in Early Neural Network Research

The early neural networks were designed linearly, making them suitable only for solving simple linear problems. When used in a more complex environment, especially where the problem is non-linear, their performance was poor. Introducing more network layers increased their complexity and resulted in high resource consumption. Moreover, gradient vanishing was a significant challenge for neural network research. This occurred due to the breakthroughs in gradient-based learning using backpropagation. It was also discovered that the deeper the neural network, the better its performance. Unfortunately, no computing devices could handle the required computation for back-propagating the derivative information in terms of speed and space.

3. The Resurgence of Neural Networks

The result of the perceptron analysis released by Marvin and Seymour (1969) caused a decrease in funding for Artificial Neural Networks, as supporters believed it to be a project with little worth. Nevertheless, certain researchers refused to give up on their work and instead decided to finance their research. This determination and perseverance are truly admirable and deserve recognition.

The field of neural networks saw a reawakening with the introduction of connectionism and evolutionary learning procedures. The M-P model and Rosenblatt Perceptron were models that could learn linear relationships from binary input and output. Although the Rosenblatt model was a single-layer machine learning model, it was never considered a deep model. Ivakhneko and Lapa’s work, titled "The Group Method of Data Handling" and its variant, "multilayer Group Method of Data Handling," achieved an architecture with hidden layers that could learn complex relationships between input and output variables. Another variant of their work, called "polynomial Group Method of Data Handling," introduced a polynomial activation function in the hidden layer of the network and used it to learn a nonlinear relationship between the input and output variables (Ivakhnenko, 1968; Ivakhnenko and Ivakhnenko, 1995).

Ivakhnenko and Lapa (1965) were the first to propose deep learning multilayer neural networks. They successfully introduced learning into a multilayer neural network with many hidden layers. Embracing supervised machine learning techniques, they employed least square analysis to train the model layer after layer. They also introduced some other optimization algorithms
to improve system performance. Amari and his student Saito Amari (1967) later enhanced the work of Ivakhnenko and Lapa with an end-to-end learning technique using Stochastic Gradient Descent (SGD). They used the learning method to train a five-layer Multilayer Neural network with two modifiable layers to achieve pattern classification in a non-linear environment. The general architecture of a Multilayer or feedforward NN is presented in Figure 4.

The second factor that contributes to the resurgence of deep neural networks is the introduction of learning algorithms to train multilayer neural networks. Several evolutionary search procedures have been considered in the literature. The Evolutionary Artificial Neural Network (EANN) could occur in any of the following or their combination: evolution of weight, evolution of architecture or evolution of learning rules Mühlenbein (1990); Whitley

---

Figure 4: General Architecture of a Multilayer or Feedforward NN
et al. (1990); Lindgren et al. (1993); Yoon et al. (1994); Belew et al. (1996); Salah and Al-Salqan (2005). Among the various evolutionary search algorithms for learning in ANN, gradient descent has proved to be more efficient in providing optimal solutions (Yoon et al., 1994).

3.1. Breakthroughs in back-propagation algorithm

According to Schmidhuber (2022), backpropagation is regarded as the most efficient method for implementing chain-rule in deep neural networks. On this note, since the core of backpropagation is chain-rule, it suffices to give some accounts of its discovery and evolution. The credit of chain-rule discovery was attributed to Godfree Wilhem for his work titled “Chain-rule for backward credit assignment” in 1676 (Leibniz, 1989; Leibniz and Heer, 1958). The idea has been employed in the modern neural network to resolve the problem of how the output of the output layer of a neural network could change with little changes in the weight parameter of the hidden layer(s). Cauchy and Hadamard Lemaréchal (2012) employed Chain-rule to develop a model which they called Gradient descent model, and the variant- Stochastic Gradient Descent was later proposed by Robinson and Moron in 1951 (Robbins and Monro, 1951).

Stochastic Gradient Descent and its variants have recently become a popular approach for training deep neural networks. In his work ”Gradient theory of optimal flight path,” Kelley (1960) derived the basis for backpropagation, which is now relevant in machine learning and deep neural networks. Dreyfus (1990) also presented work similar to the back-propagation described by (Kelley, 1960). During this time, derivative information was back-propagated from one layer to the next through Jacobian matrix computation (Schmidhuber, 2022). However, it was observed that Kelley’s implementation of back-propagation could not account for direct links across many layers and could not take network sparsity potential efficiency gain into account (Schmidhuber, 2022). In 1970, Linnainmaa successfully implemented the generally known and widely accepted back-propagation. Later, Werbos (1988) utilized Linnainmaa’s version of back-propagation to train neural networks as an extension to his thesis.

3.1.1. Backpropagation description and optimization algorithms

Backpropagation is the artificial neural network training process for minimizing network prediction error or misclassification by adjusting the networks’ connection weights between neurons by propagating the error back
into the network, and the process continues until a satisfactory minimum error is achieved (loss function). The whole process enables neural networks like feedforward neural networks, convolutional neural networks, and other supervised-based neural networks to learn representative features and complex patterns from input data.

Backpropagation involves two major steps: forward propagation and backward propagation. During forward propagation, data is passed from one layer of neurons to the next, starting from the input and ending at the output layer. Each neuron in the network computes its output activation by applying an activation function to its weighted sum. This allows the data to be propagated through the network until the final prediction is made. After a forward pass, a backward pass is done to compute the network error. The error is calculated using a loss function as the difference between the predicted output and the expected output. The error is then propagated back through the network, starting from the output layer and working its way back to the input layer. Using the chain rule, the contribution of each neuron to the network error is calculated by computing the gradient of the loss function with respect to the weight. This helps to correct the error and optimize the network. In Figure 5, the forward pass is represented with the orange arrow in a forward direction, and the green arrow indicates a backward pass with the arrowhead facing the left direction.

Different optimization algorithms employed for neuron weight updating in deep learning include Gradient Descent, Stochastic Gradient Descent (SGD), Mini-Batch Gradient Descent, SGD with momentum, Nesterov Accelerated Gradient, Adaptive Gradient, AdaDelta, RMSprop and ADAM. The optimization algorithms are summarised in Table 2.
<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Author</th>
<th>Feature</th>
<th>Strength</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mini-Batch Gradient Descent</td>
<td>Leon Bottou (1998)</td>
<td>Efficient memory usage and less variance.</td>
<td>Efficient memory usage and less variance.</td>
<td>High tendency to be trapped at local minima. Convergence may be affected by the choice of learning rates.</td>
</tr>
<tr>
<td>Momentum</td>
<td>Yoshua Bengio</td>
<td>SGD + Momentum</td>
<td>Very high rate of convergence. Tendency to decrease parameters high variance and oscillations</td>
<td>Manual selection of hyper parameter.</td>
</tr>
<tr>
<td>Adaptive Gradient (AdaGrad)</td>
<td>John Duchi, Elad Hazan, and Yoram Singer (2011)</td>
<td>Dynamism in the application of learning rate to each parameter at each iteration.</td>
<td>Automatic update of parameters. Possibility of slow training because of constant decrease in learning rate.</td>
<td>Incur high computational complexity due to the implementation of second-order derivatives.</td>
</tr>
<tr>
<td>AdaDelta</td>
<td>Matthew D. Zeiker 2012</td>
<td>Build upon AdaGrad. However, perform monotonic decrease of learning rate.</td>
<td>Eliminate decaying learning rate.</td>
<td>Incur high computational complexity just like in AdaGrad</td>
</tr>
<tr>
<td>RMSprop</td>
<td>Geoffrey Hinton (2012)</td>
<td>Built upon AdaGrad. However, perform monotonic decrease of learning rate.</td>
<td>Automatic update of parameters. Possibility of slow training due to constant decrease in learning rate. Quick convergence with less parameter tuning.</td>
<td>Manual definition of initial learning rate value. The decaying learning rate may become too small to accommodate further learning.</td>
</tr>
</tbody>
</table>
Amari (1972) was also the first scholar to introduce learning into recurrent neural networks when he improved the architecture proposed by Elman and Zipser (1925). He introduced learning to the network such that it could be adaptive, associating input patterns to output patterns when connecting weights are modified or adjusted. However, the network could only respond to static input and, therefore, unsuitable for sequence processing. Many learning techniques have been considered to achieve a more efficient and robust feedforward neural network. These evolutionary search procedures include backpropagation, heuristic search, stochastic search, simulated annealing, evolutionary programming, evolution strategies and genetic algorithm.

3.2. Neural networks in the 1980s: A period of progress

In 1979, Fukushima (1979) proposed a neural network called neocognitron that mimics the vision mechanism of animals for pattern recognition. This model was motivated by an experiment conducted by (Hubel and Wiesel, 1962). The neocognitron consists of cascaded cells, mainly S-cells and C-cells. The S-cells were used for local feature extraction, which was then integrated and classified in the higher layer. The C-cells were tolerant to any shift or deformation experienced in the local features, making the network robust against possible deformations. Fukushima’s work introduced the concept of convolution and downsampling, which are major techniques used in
the present CNN. Additionally, the ReLu (Rectified Linear Units) activation function was proposed in 1969, which has become part of the training algorithm for deep neural networks.

John Hopfield proposed a Hopfield neural network, Hopfield (1982) characterized by a recurrent architecture featuring fully interconnected neurons with self-connections. This framework endowed the network with the capacity for information storage and retrieval via feedback mechanisms. Employing binary inputs (0 or 1) and connection weights limited to +1 or -1 values, the network was governed by an energy function dictating its stability, with attractor states representing stored memories utilized for pattern recognition tasks. Leveraging the Hebbian learning rule, the Hopfield network established associative links between input and output patterns and proficiently restored patterns from noisy inputs through content-addressable memory mechanisms.

The Boltzmann machine (Hinton and Sejnowski, 1983) emerged as a noteworthy addition to the neural network domain, renowned for its adeptness in learning representative patterns from input distributions, capitalizing on statistical principles. Another significant milestone of the 1980s was the introduction of the backpropagation algorithm, which involved the effective application of the chain rule. Backpropagation swiftly garnered widespread acceptance as the preeminent training paradigm for a diverse array of Artificial Neural Networks (ANNs) and deep learning architectures, persisting as a foundational element of contemporary neural network methodologies.

The Radial Basis Function Network (RBFN), a notable advancement in neural network research during the 1980s, was conceptualized by (Broomhead and Lowe, 1988). This architecture comprises three fundamental layers: the input layer, the hidden layer, and the output layer. The input layer serves as the ingress point for the network, receiving data features. Meanwhile, the neurons within the hidden layer embody radial basis functions endowed with the capacity to convert input features into a higher-dimensional feature space. The output layer, in turn, synthesizes network outputs based on the transformed features. In tasks such as regression, a solitary node may suffice within the output layer, whereas classification tasks may necessitate multiple nodes. Of particular significance is the RBFN’s adeptness in modelling intricate relationships between input and output variables, facilitated by its nonlinear mapping capabilities. This attribute is the reason for its proficiency in both universal and function approximations. Moreover, its capability to effectuate transformations of input features into a high-dimensional feature space...
space contributes to its application in pattern recognition and classification tasks.

The Cascade Correlation network, proposed by Fahlman and Lebiere (1989), could be seen as the last innovation of neural networks during the 1980s. Distinguished by its unique capability to expand its structure during the learning process, this network exhibits remarkable suitability for modelling complex relationships within data. Comprising an input layer, hidden layer(s), and an output layer, the Cascade Correlation network commences its training with a rudimentary architecture, typically consisting of input and output layers. Initially employing backpropagation upon the addition of the first hidden layer, subsequent iterations involve the incremental inclusion of neurons to the hidden layer. Each newly added neuron undergoes training via cascade correlation learning, a variant of backpropagation tailored for this purpose. The iterative augmentation of the network’s structure continues until a predefined threshold is reached. Of notable significance is the network’s adaptive capacity to accommodate complex data environments without pre-knowledge of the optimal architecture. This dynamic adaptability enables the Cascade Correlation network to effectively model intricate patterns and relationships within the data, thus enhancing its utility across diverse domains of application. The summary of the contributions made in deep learning in the 1980s is presented in Table 3.
<table>
<thead>
<tr>
<th>Neural Network Model</th>
<th>Author &amp; year</th>
<th>Contributions</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hopfield Neural Network</td>
<td>John Hopfield 1982</td>
<td>Associative Memory.</td>
<td>Pattern Recognition and optimization</td>
</tr>
<tr>
<td>Boltzmann Machine</td>
<td>Geoffrey Hinton and Terry Sejnowski 1985</td>
<td>Statistical tools</td>
<td>Learning representative feature, dimensionality reduction, pretrained network in an unsupervised manner for a classifier</td>
</tr>
<tr>
<td>Back propagation</td>
<td>David Rumelhart, Geoffrey Hinton, and Ronald Williams 1986</td>
<td>Learning Model for Neural Network</td>
<td>Training Neural network models to achieve: Pattern recognition, classification, Natural Language Processing, Image detection and segmentation, Medical image analysis</td>
</tr>
<tr>
<td>Radial Basis Functions</td>
<td>Bernard Widrow and Marcin Hoff</td>
<td>Radial basis function as activation function for FFN</td>
<td>Pattern Recognition and Function approximation tasks, classification, control systems, time series prediction, regression</td>
</tr>
<tr>
<td>Cascade Correlation Network</td>
<td>Scott Fahlman and Christian Lebiere 1989</td>
<td>Adaptive learning and dynamic growth of network architecture</td>
<td>Efficient implementation of Multilayer Neural Network, introduction of constructive learning, Pattern recognition and pattern modelling</td>
</tr>
</tbody>
</table>
4. Convolutional Neural Networks (CNNs)

4.1. Introduction to CNNs and their architecture

CNN is a type of feedforward neural network that was inspired by a virtual cortex-based neural model proposed by (Fukushima, 1979). This model was designed to mimic human vision for pattern recognition and gave rise to the convolution and downsampling methods used in CNN. The availability of the ReLu activation function, combined with convolution, downsampling, and backpropagation techniques, prepared the ground for the development of CNN architectures Akilan and Wu (2020). Although (LeCun, 1989) is often referred to as the father of CNN, other scholars had previously introduced the concept of a convolution-like network under different names (Schmidhuber, 2022). For example, Waibel et al. (1989) proposed a neural network that combined convolution and weight sharing for speech processing, which he called a TDNN (Time Delay Neural Network) Sugiyama et al. (1991). Later, Yamaguchi et al. (1990) improved the TDNN’s performance by introducing a technique called Max-pooling (down-sampling).

Since CNN’s introduction, many variations of the network have been proposed by different individuals and teams. Therefore, it is necessary to present the general architecture of CNN to identify similarities and differences between network variants. The Cascade Correlation network proposed by Fahlman and Lebiere (1989) could be seen as the last innovation of neural networks during the 1980s. This network is unique in its ability to expand its structure during the learning process, making it particularly suitable for modeling complex relationships within data. The network comprises an input layer, a hidden layer(s), and an output layer. It starts with a rudimentary architecture consisting of input and output layers and uses backpropagation to add the first hidden layer. Subsequent iterations involve the incremental addition of neurons to the hidden layer. Each newly added neuron undergoes training via cascade correlation learning, a variant of backpropagation designed for this purpose. The iterative augmentation of the network’s structure continues until a predefined threshold is reached. Of notable significance is the network’s adaptive capacity to accommodate complex data environments without pre-knowledge of the optimal architecture. This dynamic adaptability enables the Cascade Correlation network to effectively model complex patterns and relationships within the data, enhancing its usefulness across diverse domains of application.
4.2. Convolution Neural Network Architecture

CNN is a non-linear model with the self-sufficient capability of extracting features from a pattern-based object such as images and videos (Dara and Tumma, 2018). The components or major processes involved in CNN include the following: convolution, downsampling, activation function, network optimization, and learning algorithms. One or more of these functions are employed in any of the layers of the network. Like ANN, CNN contains some layers, including the input layer, which is an entrance to the network. The convolution layer applies a filter to the data, the pooling layer minimizes computation through down-sampling techniques, and the output layer provides the decision of the network. A Simple diagrammatic representation that shows how CNN works is presented in Figure 6.

**Input Layer.** The CNN input layer is the gateway through which data are passed into the network. The number of neurons in this layer depends solely on the number of features in the data. For image data, features are considered to be the image pixels. Input layers are not usually considered as part of the depth of a CNN because there is no computation taking place in the layer Jogin et al. (2018). Although CNN is self-sufficient in extracting features from its input, observation shows that preprocessed input often provides better performance of CNN networks (Pitaloka et al., 2017; Zheng et al., 2018b; Tang et al., 2020; Beeravolu et al., 2021). Therefore, it is advisable to preprocess input data to minimize or eliminate possible aliases, noise, and redundancy from the data.

**Convolution Layer.** The parameters at the first convolution layer include filter size, image height, image width and image channel. Grayscale images have one channel, while RGB (coloured) images have three channels. In addition, the output of the convolution process is part of the parameters to be provided at the first layer. Generally, the convolution layer is saddled with the responsibility of extracting features from input data. A convolution operation is performed on the data when the data is convolved with a kernel of size 3x3, 5x5 or 7x7, depending on the dimensionality of the data. The feature map, which is the output of the layer, is determined by the kernel number and the size of the input data. Activation functions like ReLu, Tanh, or Leaky ReLu are applied to the output layer of every convolution layer in the network to achieve nonlinearity in the network. Equation (2) formally represents forward pass convolution operation in CNN. Assuming f is the
input image and h is the kernel \( G(x,y) \), it shows the convolution of f with h filter.

\[
G[x, y] = (f * h)[x, y] = \sum_{j} \sum_{k} h[j, k] f[x - j, y - k]
\]  

(2)

**Pooling Layer.** At the pooling layer, the feature map generated from the convolution layer is down-sampled. The applied pooling type could be max pooling (Max(0,1)), average pooling or global pooling. The purpose of pooling is to cause a dimensionality reduction in the feature map and reduce the number of parameters to learn in the network, thus minimising system resource consumption and overfitting. In the process, the depth of the feature map is preserved. Pooling is done by convolving 2x2 filters with the CNN feature map, aiming to preserve important features in the feature map, (3) is the Mathematical expression for Max Pooling operation.

\[
\text{Max pooling} = \text{Max}(I * K)(i, j)
\]  

(3)

where I is the Image or Input and K is the Kernel

**Fully Connected Layer.** The fully connected layer ensures that the output feature map of either the convolution layer or the pooling layer that precedes the output layer is densely connected to the output neuron so that every feature vector will influence the result of the output vector. The output of the convolutional or Pooling layer is first flattened, that is, it is converted
to a one-dimensional vector, which becomes an input to a fully connected layer. The fully connected layer applied a linear function to the input before applying a non-linear function. This process may continue depending on the depth of the network. Each complete linear and nonlinear function application makes one fully connected layer (1FC). Equation (4) is the output of a neutron in the fully connected layer in each forward pass.

\[ \text{output} = \phi(W_{i,j} \cdot X_j + b_j) \]  

(4)

Where \( \phi \) is the activation function, \( W_{i,j} \) is the unique weight assigned to each input, \( X_j \) represent each and \( b_j \) represent the bias in the network.

**Output Layer.** The output Layer is a fully connected layer with a non-linear classifier and error function to generate the probability of the classes. The popularly used non-linear classifiers are the softmax function and the sigmoid function. However, in the case of network hybridization where a convolution neural network is used for feature extraction, some traditional machine learning classifiers such as the Support vector machine (SVM) (Kang et al., 2021; Liang et al., 2021; Latif et al., 2022), decision tree (Kumar et al., 2015) or forest tree (Oraibi et al., 2018; Guehairia et al., 2020; Yaqoob et al., 2021) can be used in place of softmax or sigmoid function.

The predicted value of the network is obtained at the output layer, which is then evaluated with a loss function depending on the task the network performs. The possible task in CNN include, Binary classification, Muticlass classification or Regression. This study assumes Binary Cross Entropy (BCE) loss function, formally represented in (5)

\[ \text{BCE}(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^{n} (y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \]  

(5)

where \( y \) is the expected output and \( \hat{y} \) is the predicted result of the network.

### 4.2.1 Activation Functions in Deep Learning

These are functions that help in non-linearity learning in deep learning the popularly used include: Sigmoid, Tanh, ReLu, Leaky ReLu, SeLu and Softmax. The graph of the stated activation functions is presented in Figure 7.
**Sigmoid Function.** This function may be applied at the convolution layer but mostly applied to the output layer for prediction purposes, suitable for a binary classification task. It is differentiable and applicable to real inputs within the range of 0 and 1 to ensure non-negative derivative and precisely possess one inflection point. Sigmoid function is mathematically represented in (6) where \( x \) is the input and \( e \) is the euler’s number and these are used in other functions’ discussion.

\[
\sigma(x) = \frac{1}{1 + e^{-x}}
\]  
(6)

**Hyperbolic Tangent Function (Tanh).** Tanh is similar to sigmoid but it rather centered at 0 and ranges from -1 to 1. It is applicable at the hidden layer or fully connected layer, it helps in data point centralization. Tanh mathematical expression is available in (7).

\[
\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]  
(7)

**Rectified Linear Unit.** ReLu activation function is appropriate at the convolution layer of CNN, other deep learning models also employ it to avoid vanishing gradient. ReLu has simple implementation with efficient computation, its mathematical expression is given in (8). The input range in ReLu is from 0 to \(-\infty\). However, ReLu is susceptible to dying ReLu challenge where some neurons may not be activated during training. Leaky ReLu a variant of ReLu resolves the problem of dying ReLu by returning small negative value for possible negative inputs. Leaky ReLu mathematical expression is presented in (9), where \( a \) is a small positive function and \( x \) range from \(-\infty\) to \(+\infty\). Another variants of ReLu activation function is ELU (Exponential Linear Unit), it resolve the issue of vanishing gradient, which is one of the problems of ReLu. The mathematical expression is given in (10), where \( a \) is the small positive value and the input value ranges from \(-\infty\) to \(+\infty\). ELu was further modified to SELU (Scaled Linear Unit) which introduced self normalization to deep network by enforcing zero mean and unit variance at the output of each neural network layer. it is mathematically expressed in (11), where \( \lambda, \alpha, \alpha > 0 \) or a small positive constant value.

\[
\text{ReLU} \ (x) = \max(0, x)
\]  
(8)
Leaky ReLu($x$) = \[
\begin{cases}
  x & \text{if } x > 0 \\
  ax, & \text{otherwise}
\end{cases}
\] (9)

ELU($x$) = \[
\begin{cases}
  x, & \text{if } x > 0 \\
  a(e^x - 1), & \text{otherwise}
\end{cases}
\] (10)

SeLu($x$) = \[
\begin{cases}
  x, & \text{if } x > 0 \\
  \lambda a(e^x - 1), & \text{if } x \leq 0
\end{cases}
\] (11)

**Softmax Activation Function.** This function is predominantly used in the output layer of neural networks for multi-class classification tasks. Its primary purpose is to convert raw scores, also known as logits, into probabilities that add up to 1. This transformation allows the network to provide probabilities for each class as its output. Mathematical expression is presented in (12) where $i, j$ are natural numbers.

\[
\text{Softmax}(x_i) = \frac{e^{x_j}}{\sum_{j=1}^{n} e^{x_j}}
\] (12)

4.3. **LeNet-5 and its impact on computer vision**

LeCun is one of the renowned researchers who has made pronounced contributions to deep learning. His contribution to the real-world application using CNN cannot be overemphasized. He is regarded as the father of CNN, as a compliment to his achievements.

LeCun (1989) successfully applied the convolution neural network to real-life problems when he used a backpropagation algorithm to train a combination of neural networks for recognising the zip code numbers provided by the postal service in the US. The motive behind the success was the generalization concept he had. He was convinced that the network’s ability to learn how to generalise could be improved if some constraints about the task’s domain were available. To substantiate his belief, He experimented by considering a linearly separable handwritten recognition problem using a single-layer neural network. He observed a poor performance in the generalisation of the network. However, the network provided a well-generalised model by performing a similar experiment with a shift-invariant feature detector on a constrained
Figure 7: A is the graphical representation of Sigmoid function, graph labelled B is Tanh, graph with tag C is ReLU, Graph with label D is Leaky ReLU, Graph with label E is SELU and F is Softmax graph.

Figure 8: LeNet Architectures
multilayer network. With the experiment output, he concluded that network learning generalisation ability could be enhanced by minimizing the free parameters in the network, and the model was tagged LeNet-1 (LeCun, 1989). Later, their model was adapted to an image-based handwritten recognition task in 1990 (Lecun et al., 1990). This time, they proposed a highly constrained backpropagation neural networks model, which was applied to images of handwritten numbers as input data with data preprocessing. The model generalised well with an error rate of 1% and a rejection rate of 9%. The availability of the MNIST dataset in 1994, which was later published in 2012 (Deng, 2012) encouraged LeCun et al. to introduce Lenet-4 Lecun et al. (1995), a variant of LeNet-1, with more learning parameters. Lenet-4 was trained on the MNIST dataset, and the following year AT&T lab proposed Lenet-5 for handwritten character recognition. The performance of the new model was compared with other handwritten character recognition models in the literature, and it was found that Lenet-5 provided optimal performance over others Lecun et al. (1995).

In 1998, Lecun et al. (1998) practically and successfully applied their neural network model to online handwritten character recognition and reading of large numbers of checks. Many recent neural network models are not similar to the LeNet model. However, the model brought about great inspiration to them and prepared a footprint to follow. Since LeNet-5 form the basic unit of all the recent and efficient convolution neural networks, it is necessary to discuss its architecture.

The architecture of LeNet-5 comprises five layers that can be learned as parameters. Among these layers, three are convolution layers and pooling layers that are arranged alternatively. The other layers consist of fully connected layers, and the final layer has a classifier activation function and an error function. The output layer classifies the images into their respective classes. The LeNet architectures are presented in Figure 8.

The input images to the model are grayscale and have a size of 32 x 32 pixels with one channel. These images are fed into the first convolution layer, where six filters of size 5 x 5 are convolved with the input to produce a feature map of size 28 x 28 x 6. Then, an average pooling operation is performed, which reduces the size of the feature map to 14 x 14 x 6 without affecting its depth.

The next convolution layer uses 16 filters with a size of 5 x 5 to convolve the output of the pooling layer and generate a feature map of size 10 x 10 x 16. The new feature map is then subsampled with average pooling, which
reduces the size to $5 \times 5 \times 16$. The final convolution layer reduces the feature map to $1 \times 1 \times 120$ by convolving the output of the pooling layer with 120 filters of size $5 \times 5$.

The resulting feature map is then flattened to a one-dimensional array of 120 values. All 120 vectors are fully connected to 84 neurons at the first dense layer, which is then connected to 10 neurons, the supposed classes to be predicted at the last layer that contains the classifier activation function and the error function.

Maraoui et al. (2021) developed a LeNet-5 based hardware for traffic sign recognition. The proposed model capitalised on the advantages of Field Programmable Gate Arrays (FPGAs) and deployed the traffic sign recognition on the PYNQ z1 platform. The LeNet-based hardware was created and its interface was by using a model generated from the training of traffic sign recognition on a GPU. The model accelerator was made to communicate with PS via the HLS tool, and for easy transferring of data, another interface was created between PS and PL, which was done after importing the LeNet-5 accelerator into Vivado IPI. They later implement the bitstream file into the PYNQ.

Njuguna et al. (2023) recently proposed the deployment of an optimised LeNet-5 model on Field Programmable Gate Arrays (FPGAs) using Brevitas and FINN frameworks for handwritten digit recognition. They reported that the network generalised well on a test set with an accuracy of 96.64% coupled with harnessing hardware resources of low power consumption for the deployment of the deep model.

Several notable variants of Convolutional Neural Networks (CNNs) have emerged, each distinguished by its unique architectural features and contributions to the field of computer vision. These include AlexNet, GoogLeNet, ResNet (Residual Neural Network), ZFNet, VGGNet, R-CNN (Region-based Convolutional Neural Network), and Fast R-CNN. These architectures have effectively employed convolutional and downsampling techniques in distinct ways, catering for various tasks within the realm of computer vision. The tasks to which these CNN variants have been applied extend beyond image detection, image segmentation, classification, feature extraction through transfer learning, and network fine-tuning. The following section will comprehensively explore the diverse applications of these networks. Also, a thorough examination of the complexities inherent in CovNet architectures is presented in Table 4.
<table>
<thead>
<tr>
<th>Network</th>
<th>Author &amp; Year</th>
<th>No of Parameter</th>
<th>Features</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>Krizhevsky &lt;i&gt;et al.&lt;/i&gt; (2012)</td>
<td>Approximately 60 million</td>
<td>Parallelism feature for efficient usage of GPU. Pooling implements spatial information, redundant information is removed. Subject to long training time.</td>
<td>Overfitting to a small dataset. Requires high-performing computing resources. Interpreting the learned representation is difficult.</td>
</tr>
<tr>
<td>ZFNet</td>
<td>Fergus (2014)</td>
<td>Approximately 57 million</td>
<td>Replaced Sigmoid function with ReLU activation function.</td>
<td>Subject to long training time. Interpretable representations are difficult.</td>
</tr>
<tr>
<td>VGGNet</td>
<td>Simonyan and Zisserman (2014)</td>
<td>Approximately 138 million (VGG-16) 144 million (VGG-19)</td>
<td>Large number of learnable parameters, large amount of computation, and more training time are required.</td>
<td>Interpretable representations are difficult.</td>
</tr>
<tr>
<td>ResNet</td>
<td>He &lt;i&gt;et al.&lt;/i&gt; (2016)</td>
<td>Depends on the depth of the network (25.6 parameters for ResNet-34, 50.6 for ResNet-50).</td>
<td>Implementation of skip connections that help to resolve the problem of vanishing gradients. Combination of multiple convolution layers as Residual blocks for learning residual features. Incorporation of identity mapping that maps the input with the output without any changes.</td>
<td>Extra carefulness and several experiments are required for optimal positioning and structure of skip connections.</td>
</tr>
</tbody>
</table>

Table 4: Brief description of deep CoVNet.
5. Recurrent Neural Networks

5.1. Introduction to RNNs and their sequential modeling capabilities

RNN is a specialized model designed for processing sequential data such as voice, natural language, video, text, and other spatiotemporal data. This model includes a memory mechanism enabling it to recall its input. The RNN architecture is designed in a way that allows a path from specific internal nodes to others, providing a possibility of getting back to the starting node. This feature is crucial for implementing memory, as proposed by (Elman and Zipser, 1988; Storck et al., 1995). (Elman and Zipser, 1925) were the first scholars to suggest the concept of RNNs, but their model could not learn because it settled in an equilibrium state after receiving input. This formed the basis for the development of learnable RNNs, as proposed by (Niss, 2005). Additionally, Warren and Pitts (1943) made significant contributions to non-learning RNNs.

Amari (1972) introduced adaptiveness into the RNN model by varying the connection weights, which made it possible to associate input patterns with output patterns. Amari’s work was the first RNN to be published. However, the model could not learn sequence patterns. The main challenge of the first published RNN is how the subsequent layers would remember more information from the input data. To address this problem, Elman and Zipser (1988) proposed the implementation of short-time memory in RNN by creating a recursive connection in the hidden layers such that a hidden unit would be aware of its previous output and thus use the information to determine its subsequent action. Figure 9 contains the architectural diagram of RNN. Hoehfeld and Fahlman (1992) in his contribution, developed a cascaded correlation network called Recurrent Cascaded Correlation (RCC). This network started with a single layer, that is, the input and output layer with their connections, and used a quick propagation algorithm for the training of the network. At every training, if the error in the network was negligible, the network would be evaluated. Otherwise, a hidden layer would be added until a satisfactory minimum error was attained. Furthermore, both Williams and Zipser (1989); Schmidhuber (1992) addressed the issue of time lag to determine the content of short memory by reducing the complexity incurred through backpropagation in a recurrent network. Their goal was to minimise the time lag between successive inputs in a sequence data.

Recurrent networks can store recent input information using their feedback reconnection mechanism, as shown in Figure 9. This ability is very
necessary in using neural network models to process sequential data. Earlier models enhanced with short-time memory and trained with backpropagation were prone to either gradient explosion or gradient vanishing problem due to backpropagation of error Storck et al. (1995). Furthermore, the short time memory cannot tolerate a long time lag between the input signal and the associate teacher signals, which is often caused by the algorithm that searches for the right content to put in the short time memory. The approach of (Williams and Zipser, 1989; Elman and Zipser, 1988; Hoehfeld and Fahlman, 1992; Schmidhuber, 1992) contributed little or no effect in resolving the challenges associated with gradient explosion, gradient vanishing and time lag in determining the content of short memory in a recurrent neural network. Identifying this problem, Hochreiter and Schmidhuber (1997) proposed a Long Short Term Memory (LSTM) neural model with the ability to enforce constant, non-vanishing and non-exploding error flow.

5.2. Long Short-Term Memory networks and their significance

The LSTM version of RNN was proposed in the 1990s to resolve amnesia issues, gradient vanishing and gradient explosion in RNN (Storck et al., 1995; Hochreiter and Schmidhuber, 1997). LSTM is enhanced with a fast learning mechanism to distinguish between two or more input sequences occurring in
succession. This feature ensures LSTM robustness against short-time memory, gradient vanishing and explosion.

The architecture of LSTM consists of a series of LSTM cells (Input and output) and gates. Each unit cell contains three inputs (token at the current time, the previous hidden state and the previous cell state) and two outputs (updated hidden state and current cell state). Each gate contains an input gate: which allows relevant information for the current input, an output gate that updates and finalizes the next hidden state and a forget gate that eliminates irrelevant information by multiplying such information with zero. The gates of the cell dictate how information flows in and out of the cell, and hence, can either forget or retain information from the previous time stamp. With this capability, LSTM could ensure long time dependency on data. Just like RNNs, the LSTM cell passes its output to the next LSTM cell in the network because each cell also has a memory facility that influences the output of the current timestamp by using the information it stores from the previous time stamp (Yu et al., 2019).

The architecture presented is general for LSTM. However, there are many variants of implementation of LSTM in the literature with one or two modifications to the general model. Like Peephole LSTM (PLSTM) proposed by Gers et al. (2000) improvised a peephole connection such that the gate layer could peep at the cell state. The peephole connection enhanced the gate also to consider the previous internal state and not only the previous hidden state. Multilayered Attentional Peephole convolutional long short-term memory (MAPCoL; LSTM) (Rahman and Siddiqui, 2021), Couple forget input gate LSTM, Gated Recurrent Unit and many more are the variants of LSTM proposed in the literature (Guo et al., 2021; Dey and Salem, 2017; Kanai et al., 2017).

6. Unsupervised Learning and Deep Belief Networks

Unsupervised learning can be seen as the machine learning approach that deduces useful patterns from streams or pools of raw data without prior information provided. Unlike supervised learning, unsupervised models are capable of finding similarities and differences in unlabeled or uncategorised data in the absence of an external instructor. Unsupervised models are generative models, and common examples include K-means, Principal Component Analysis (PCA), and Gaussian Mixture Model (GMM), to mention a few.
Unsupervised learning techniques are suitable for tasks such as clustering, association, or dimensionality reduction.

Although the neural network is a discriminative model, however, some of its variants are modelled to perform generative tasks like text generation, image generation, music generation, video generation, speech synthesis, handwritten generation, drug discovery or protein design. The popular neural networks for unsupervised learning tasks include Helmholtz machines, Boltzmann machines, Restricted Boltzmann Machines (RBM), Autoencoder, GAN, and Deep Belief Network, to mention a few. For the rest of this section, we will thoroughly consider the recent Unsupervised neural network models that are widely used in the field (Autoencoder and Deep Belief Network). Should there be interest in the earlier models of unsupervised neural networks the work of (Wang et al., 2017) contained a comprehensive study with necessary information.

6.1. Autoencoders and their contributions

Autoencoder is a generative form of deep feedforward neural network used to find representative features in data. It applies to data like image, tabular, sequential, Spatiotemporal, and so on. Autoencoders have two important functional parts: encoders and decoders. The encoder compresses input data into a code, which forms a lower-dimension representation of the data in the hidden nodes. The code becomes an input into the decoder, which in turn transforms by the decoder to a replica of the input data (Ng et al., 2011; Lopez Pinaya et al., 2020). Both the encoder and the decoder are fully connected feedforward neural network layers.

Autoencoder applications are not limited to dimensionality reduction, image denoising, feature extraction, image compression, recommendation systems, and data generation. The function performed sometimes depends on the type of encoder used. The following are the types of autoencoder discussed in the study: sparse, denoising, contractive, undercomplete, convolutional and variational autoencoder.

Sparse Autoencoder: This type of autoencoder controls network errors by increasing or decreasing the number of hidden nodes. Changing the number of nodes will cause the hidden layer activation nodes to be penalised by the network. The number of neurons activated is proportional to the penalty imposed by the network, which is then applied to the loss function. This process is regarded as a regularisation technique, which is different from the popular regularisation methods that penalise the size of the weight of the
network. It rather penalises activation nodes number (number of nodes to be activated) (Ng et al., 2011; Makhzani and Frey, 2013; Xu et al., 2016; Sankaran et al., 2017).

Undercomplete Autoencoder: This type of autoencoder ensures that the hidden layer is smaller than the network input size to control overfitting. Unlike sparse autoencoder, it does not perform network regularization. This autoencoder is efficient for dimensional reduction because of its nonlinearity preservation in data (Lopez Pinaya et al., 2020; Davila Delgado and Oyedele, 2021).

Contractive autoencoder: This type of autoencoder aims to present a robust representative feature of the input data, which would not be affected by any small variation in the data. It is done in practice by ensuring that when similar inputs are fed into the network, the input hidden layer activations derivatives are kept small (Rifai et al., 2011; Diallo et al., 2021).

Denoising autoencoder: This autoencoder could be termed a regularised autoencoder. Like a sparse and contractive autoencoder, it is efficient for feature extraction in a classification task (Gondara, 2016). The denoising autoencoder first applies noise to the input data to make a corrupt copy of the data and then tries to reproduce the input without noise. The decoder output always differs from the encoder input (Vincent et al., 2010).

Convolutional autoencoder: It is used mainly in imaged-based environments. It replaces the feedforward network in both the encoder and decoder phases with a convolutional network and employs the necessary loss function (Chen et al., 2021c).

Variational Autoencoder (VAE): The aforementioned autoencoders are useful for extracting representative features from input data in the hidden state. While doing this, they relied much on self-adaptive regularisation. During dimensionality reduction, it is possible that the network either fails to capture the required information as the data representation or has no reconstruction loss. VAE is an autoencoder whose encodings distribution is regularised during the training to ensure that its latent space has good properties to generate some new data (Cheng et al., 2018; Chien and Wang, 2019).

Deep autoencoder: This autoencoder implements two similar deep belief networks having up to four or five layers at both the encoder and decoder ends (Zhou and Paffenroth, 2017).
6.2. Deep Belief Networks (DBN) and their impact on deep learning

Deep Belief Network proffers solutions to existing neural network challenges such as slow learning, a large volume of training data requirement, and local minimum trap due to poor feature selection, to name a few. DBN is a generative model using Restricted Boltzmann Machines (RBM) as its building blocks, and it is regarded as a stacked RBM of acyclic hidden layers (Hua et al., 2015). In DBN, representative features are extracted from the input data in hierarchies of layers. The lower layer identifies basic patterns, while the higher layer recognizes more abstractive patterns, and each layer targets different features from input data. Like many unsupervised models, DBN could learn complexity in input data without a teacher. Its capacity to learn complex data representations with good scalability and intrinsic ability to model non-linear relationships is the reason for its wide recognition and acceptance in the field (Keyvanrad and Homayounpour, 2014; Movahedi et al., 2018). DBN architecture contains the training phase and the fine-tuning phase.

The training phase of DBN is unsupervised learning, where the network learns the probability distribution of the input data via layer-by-layer RBM processes. The lower layer in the network passes its output to the next layer as input, and each layer learns unique features from the data using a greedy search learning algorithm. Thus, DBN learns complex features from the input data (Kamada and Ichimura, 2016). The fine-tuning phase is a supervised learning where parameters learned in the training face are adjusted with back-propagation and gradient descent. The network is evaluated using label data, and the network error is back-propagated to update the network parameters.

7. Reinforcement Learning

The history of reinforcement learning revolved around three threading concepts that existed independently in their early introduction to artificial intelligence. The concepts threaded independently in their approach to producing reinforcement learning until they are integrated to achieve their common goal—modern reinforcement learning. The first concept encompasses learning by trial-and-error that revived the reinforcement learning application to artificial intelligence in the early 1980s. The second concept approached optimal control problems with dynamic programming and value function. The last is the temporal difference methods (Sutton, 1984).
The trial-and-error was proposed by Edward Thorndike, a psychologist (Thorndike, 1898). Through cat puzzle experiments, he was able to propound the law of effect expressed as:

"Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur" (Thorndike, 1898).

The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond. (Thorndike, 1911). In summary, this means actions that lead to satisfaction have a high tendency to be reconsidered, and those that lead to discomfort are less likely to be reconsidered. The law of effect birthed the concept of trial and error in reinforcement learning.

Trial and error is selectional and associative. Selectional because it involves selecting from many options based on consequences, and associative because the choice of options is associated with specific situations. Apart from selection and association, the law of effect could also be seen as a combination of search and memory. This is because it involves searching for actions that are best from several alternatives in a particular situation, and memory keeps actions that worked best for remembrance so that they could be associated with the best situation (Sutton, 1984).

Minsky was one of the first researchers to investigate the implementation of trial-and-error learning in his PhD thesis, where he discussed the computational model for reinforcement learning (Minsky, 1954). In the same year Farley and Clark (1954) proposed trial-and-error as a learning mechanism for the neural network machine they designed. The work of Minsky (1961) revived reinforcement learning in the engineering field, and many published works in the field began to employ reinforcement learning (Waltz and Fu, 1965; Fu, 1970; Mendel and McLaren, 1970) because of the innovation in (Minsky, 1961) work titled "Steps Towards Artificial Intelligence" where he illustrated convincing issues about reinforcement learning, especially the credit assignment problem.

Farley and Clark (1954), in contrast, explored trial-and-error learning as a form of supervised learning aimed at achieving generalization and pattern recognition. They transitioned their attention completely from reinforcement learning to supervised learning, leading to a misinterpretation of reinforce-
ment learning for supervised learning, as noted in the work of, (Widrow et al., 1960; Rosenblatt, 1969; Andreae, 1963). The confusion persisted and nearly stopped reinforcement learning in the 1960s and 1970s. The exception to the misconception was the work of (Michie, 1961, 1963). He proposed reinforcement learning using trial and error for learning tic-tac-toe games. Also, in 1968, Michie and Chamber (Michie and Chambers, 1968) developed GLEE and BOXES, GLEE is another reinforcement learning for playing tic-tac-toe, and BOXES was used to describe the challenge of balancing poles contrary to the supervised model approach (Widrow et al., 1960). Many researchers were influenced into the study of reinforcement learning due to the successful work of Michie, who continued to be an advocate of reinforcement learning and its application to artificial intelligence (Barto et al.; Sutton, 1984; Michie, 1986).

Widrow et al. (1973) improved on his work (Widrow et al., 1960) by creating reinforcement learning rules, which rather learn from success and failure within the system and not from training samples. This form of learning was tag learning from critics and not from teachers. TSetlin (1974) proposed learning automata, another research area that influenced the leading role of trial and error in achieving modern reinforcement learning. This method was a low-memory machine and only employed the selectional aspect of trial-and-error learning. Another selectional-based reinforcement learning was the early work of (Hayes-Roth, 1975). Holland (1995) proposed a complete reinforcement learning system called classifier systems. These systems include selection, associative, and value functions. The method was evolutionary and used a genetic algorithm as the main component. This type of system has been considered in the literature by many researchers (Goldberg and Holland, 1988; Wilson, 1994).

Another researcher who also has credit for resuscitating the treading of trial-and-error in reinforcement learning is (Klopf, 1972, 1975, 1982). Klopf arguably differentiated reinforcement learning from supervised learning; he reaffirmed the essence of trial and error as learning that deduced achievement from its environment with some control to drive it towards achieving the ultimate desire. Klopf’s observation was later complemented by the work of (Barto and Sutton, 1982; Barto, 1985, 1986).

The second historical concept of reinforcement learning is the optimal control problem. The term optimal control problem surfaced in the 1950s, and it was used to describe a model that controlled and minimised dynamic system’s behaviour measured per time. Bellman (1958) was one of the re-
searchers who addressed this problem with a functional equation called the Bellman equation, which he defined from a value function and dynamic system’s state concepts. Solving optimal control problems requires solving the Bellman equation, and the class of methods that solved the equation is called dynamic programming. In the same year, Bellman (1958) proposed Markovian Decision Processes (MDPs) for a stochastic discrete version of the optimal control problem. Building on the work of Bellman, Howard (1960) proposed a policy iteration method for MDPs. Despite the computational complexity of dynamic programming, which increases exponentially with the number of variable states, its feasible solution to the general optimal control problem is optimal compared to other existing methods. Since the 1950s, Dynamic Programming has evolved and extended to partially observable MDPs Lovejoy (1991). Its applications are provided in the work of (White, 1985, 1988, 1993), it has been employed as approximation methods presented in a survey by (Rust, 1996), used as asynchronous methods Bertsekas (1982). Some recent applications of dynamic programming are available in Bertsekas (2012).

The last aspect of the reinforcement learning history is temporal difference. It is less distinct compared to other aspects discussed. However, it has contributed a great part to reinforcement learning due to its unique-
Figure 11: The architecture of DRL
ness. Temporary-Difference (TD) could be traced to the learning attitude in animals, particularly the secondary and the primary reinforcers, where the secondary reinforcers served as a stimulus to the primary reinforcers. Therefore, the two pairs reflect similar properties. The first researcher to propose the implementation of TD in his learning method is Samuel (1959). He included TD in the program developed for playing checkers. Samuel (1959) got inspired by the statement of Shannon (1950), who suggested the possibility of a computer playing chess when programmed to use an evaluation function that could be updated online. Minsky (1961), who had earlier conceived the possibility of inculcating TD into reinforcement learning, thoroughly presented Samuel’s work and suggested its connection to the theory of secondary reinforcement (artificial and natural).

Implementation of TD hibernated for almost a decade until Klopf introduced its component in conjunction with trial-and-error learning to develop generalised reinforcement learning (Klopf, 1972), a learning ideology whereby every component perceived its inputs in reinforcement terms such that exciting inputs are viewed as reward and inhibiting inputs are viewed as punishment. Klopf’s reinforcement learning system is more of trial-and-error learning than temporal difference learning. To improve Klopf’s work, Sutton (1991) proposed a classical conditioning psychological model for temporal-difference learning. Many other researchers later used this model in their works (Moore et al., 1986; Sutton, 1988; Korf, 1988). The interpretation given to many neuroscience works during this era was temporal-difference learning-based.

Barto et al. incorporated TD into trial-and-error learning to create an actor-critic architecture model; the model was later applied to the pole balancing problem of Michie and Chamber. Sutton (1984) in his thesis extensively studied actor critic’s method. The method was also extended with a backpropagation neural network in Aderson’s thesis Barto (1985). Holland (1995) also followed the same trend of research by incorporating temporal-difference into a classifier system he developed. Modern reinforcement learning integrates trial-and-error, optimal control and temporal-difference an idea earlier suggested by Witten (1977) in his reinforcement system called Q-learning. Tesauro (1991) also integrated the three concepts in his game program called TD-Gammon, which generated many attentions in the field.
7.1. Deep Reinforcement Learning

One of the recent areas of attention in the field is deep reinforcement learning, a type of reinforcement learning that employs deep learning as a policy optimizer. Before this time, Farley and Clerk (Farley and Clark, 1954) had used reinforcement learning to train neural networks in a supervised model. Traditional reinforcement learning usually uses the Markov Decision Process (MDP) to model its environment, and the MDP model of the real world would contain high dimensional states, which is difficult or impossible for any traditional reinforcement learning algorithm to solve. Deep reinforcement learning harnessed deep learning’s high-dimensional application prowess to resolve MDP states’ challenges. The combination of reinforcement learning generality and deep learning high-dimensional affinity makes deep reinforcement learning a threading model in the field of Artificial Intelligence.

7.1.1. Deep Reinforcement Learning Model

Earlier, we presented that deep reinforcement learning is reinforcement learning that uses a deep neural network as its policy optimizer. Therefore, it suffices to present details of the reinforcement model and critically indicate the aspect of deep learning in the model. Before we proceed, reinforcement learning implies a process through which an agent is empowered with decision-making using the trial-and-error learning technique. This process is mathematically modelled with MDP, whereby an Agent at every position in a state \((s)\) will receive a reward for every action \((a)\) Decision taking that leads to a new state \((s')\) based on the environment dynamics \(p(s'\mid s, a)\). In the process, the agent strives to learn a policy for reward maximization as it moves from observation to action in the only available dynamics environment \(p(s'\mid s, a)\). This description of reinforcement learning is diagrammatically represented in Figure 8.

There are two major models of reinforcement learning: the model-based reinforcement learning model and the model-free reinforcement learning model, as depicted in Figure 10. The difference between the two models is that the model-based reinforcement learning model has complete information about its environment, including the required action for successive transitions from one state to another, the associated reward, and the probability attached. However, model-free reinforcement learning has little knowledge about its dynamic environment. It has no reward policy to assess its best policy; it can only estimate it from experience. Figure 11 is the diagrammatic representation of the two types of reinforcement learning models with the details.
of their components and the interaction with Deep Learning.

7.2. Breakthroughs in game playing

The evolution of deep learning allowed many applications to harness its purpose of feature approximation, which has sparked interest in its application for Q-function and policy learning, as well as the value of reinforcement learning algorithms. TD-Gammon was the pioneering computer program that was created using deep reinforcement learning techniques. The program utilized a neural network trained with Temporal-difference learning TD (\(\lambda\)) and was specifically designed for playing the backgammon game. With 192 input signals and without built-in knowledge, TD-Gammon could learn backgammon by itself, up to the intermediate level using TD (\(\lambda\)) (Tesauro, 1994).

Another interesting and impressive result of applying deep reinforcement learning was recorded at DeepMind. They developed a computer player for the Atari game by training a neural network with deep Q-network (a variant of deep reinforcement learning) using the game’s score as a reward. The computer program called Deep Q-network (DQN) was made to learn 49 games with little prior knowledge of the game, and the results showed that DQN had superior performance compared to all existing methods. Also, the program performance was compared to the performance of an expert game tester Mnih et al. (2015).

The rise in the application of deep reinforcement learning continued when it was used to develop a program called AlphaGo in 2015 Chang et al. (2016). AlphaGo was trained for the Go game, and the program won the Best Human Professional Player award. In 2017, an improved version of AlphaGo was proposed called AlphaZero to extend its ability to play other games like chess and shogi (Silver et al., 2018). AlphaZero also outperformed all competing computer programs for the stated games. In 2019, another variant, MuZero (Schrittwieser et al., 2020) was proposed as an improved version of AlphaZero. In 2019, researchers at Carnegie Mellon University developed a computer program to play poker. The deep reinforcement program called Pluribus defeated professionals of the game (Brown and Sandholm, 2019). Furthermore, OpenAI Five is another deep reinforcement learning game player developed for playing five-on-five Dota 2. It was reported that the deep reinforcement program outshines the former world champion (OpenAI et al., 2019).
7.3. Limitation of Deep Reinforcement Learning

Deep reinforcement learning systems often require learning from their environment. Therefore, the system needs a handful of interactions with the environment to learn the required policies. This kind of sample space is almost mission-impossible in the real world, where such training samples are expensive.

Deep reinforcement learning uses deep learning models as an approximation optimizer, and deep neural networks are susceptible to training instability and hyperparameter sensitivity. Gradient descent explosion and vanishing remain severe challenges in deep learning and will likely affect the model’s training reliability. One of the strengths of deep reinforcement learning is generability. However, this may become a challenge when there is a wide disparity between the training and test datasets. Reliable generability to achieve effective transfer learning that could be deployed into unstable conditions remains a research focus. Using deep reinforcement learning trained with trial-and-error in a critical system requires that its capability is supported with other methods or other algorithms because risks in such a system are too costly. Interpretability is one of the identified challenges of deep neural networks; this inherited challenge is transferred to deep reinforcement learning, so understanding and debugging the policy learned remains challenging. Applying deep reinforcement learning to large-scale challenges often results in high dimensional complexity and, thus, high computation cost or impractical. Deep reinforcement learning needs time and experiential data to produce the best outcome.

8. Recent Advancements and Future Directions

8.1. Transformer models and their impact on natural language processing

RNN, CNN, LSTM are renowned efficient deep learning networks for modelling sequence-based tasks, especially Natural Language Processing in the early days of deep learning evolution (Yao and Guan, 2018). The existing methods have some limitations, which are not limited to gradient explosion, gradient vanishing and amnesia (Gillioz et al., 2020; Jeon et al., 2021). The introduction of the transformer model has been able to improve the processing of sequential tasks beyond expectation. Its self-attention mechanism and other features have eliminated the amnesia challenge found in the existing models Gillioz et al. (2020).
In the Transformer Model (TM), gradient vanishing and gradient explosion are considered. The TM has implemented mechanisms such as layer normalization, residual connections, scaled dot-product attention, initialization schemes, and gradient clipping to handle vanishing and explosion problems effectively. As a result, the transformer model has emerged as the current state-of-the-art model for Natural Language Processing and similar sequential tasks.

The year 2017 marked the introduction of the transformer model by Vaswani et al. (2017), the authors proposed a self-attention model—a completely new deep architecture for processing sequential tasks without recurrence and convolution implementation. (Gillioz et al., 2020). Not long after its introduction, the research community in Natural Language Processing began to explore its application to some sequential tasks such as text generation (Varshney et al., 2020; Mishra et al., 2021), machine translation and language understanding (Banar et al., 2021).

Around the early stage of introduction, TM started evolving with several modifications to its original architecture, which include the modification of the self-attention layer with different configuration (Rao et al., 2021; Lai et al., 2023), the introduction of different positional strategies and overall architectural modification (Lu et al., 2021; Narang et al., 2021). These modifications engendered transformer variants in the field (Rao et al., 2021), and thus different applications for public use. For instance, the Google research group presented a Bidirectional Encoder Representations from Transformers (BERT), a transformer-based model pre-trained on a large corpus data and fine-tuned on a specific domain following unsupervised learning approach (Devlin et al., 2019; Chen et al., 2021a; Singh and Jaiswal, 2021).

OpenAI research group employed a similar approach in 2018 and 2019 (OpenAI, 2018) when they proposed a Generative Pre-trained Transformer (GPT1, GPT2 and GPT3), where GPT3 is their latest version and a building block for ChatGPT (Roumeliotis and Tselikas, 2023). GPTs were created by a pre-trained transformer model on a large volume of data (documents) for language understanding, text creation and provision of meaningful answers to questions (Roumeliotis and Tselikas, 2023). Casey and Michal K. (2023) is a review study on GPT models. Also, (Sufi, 2024) reviewed the application of GPTs to research capturing data augmentation and synthetic data generation. Several efforts are ongoing to improve the performance of transformer models in terms of efficiency, robustness, interpretability and adaptability (OpenAI, 2018). Recently, Transformer models have been adapted
to other areas of research like; recommendation system (Nikzad Khasmakhi et al., 2020; Wei et al., 2023; Gheewala et al., 2024), reinforcement learning (Parisotto et al., 2020; Chen et al., 2021b; Melo, 2022) and computer vision (Aouayeb et al., 2021; Mehta et al., 2023), with the sole goal of resolving long term challenges in different environment (Han et al., 2023).

8.2. Explainable AI: The quest for interpretability in deep learning

Recently, Deep learning has proved its potency beyond any reasonable doubt in AI system developments. Its self and sufficient decision is undeniably one of the factors for its wide range of applications and general acceptance. However, deep learning processes and decision-making have been described as a black box model, absolutely hidden from the user (Castelvecchi, 2016; Hussain, 2019; Buhrmester et al., 2021). Being one of the driving forces in AI models, it is a quest in the field that AI models should be enhanced with the capability of providing details of its processes and decisions to its users. This quest is the reason for an aspect of AI termed Explainable AI (XAI) (Castelvecchi, 2020; Buhrmester et al., 2021).

Explainable AI refers to AI explaining its decision reasonably and understandably to its users (Castelvecchi, 2020). It could be otherwise considered a user interface in AI, where systems communicate with users and users prompt action in response. This feature aids AI systems in informing their users what the system is doing and reasons for considering certain decisions, ensuring trust. The following goals are the expectations of systems with explainable facilities:

(a) Interpretability: This ensures that AI models, decisions and its hidden logic are accessible, understandable or comprehensible to its users.

(b) Transparency: This feature creates awareness of the systems’ internal mechanism (the input decision processes, data and algorithms) to users.

(c) Trustworthiness: This fosters the system with trust, accountability, fairness and reliability.

(d) Actionability: This allows users’ involvement based on an AI system’s decisions, insight and recommendations. The involvement and the response of the user should be in line with the understanding of the logic regarding the system’s recommendation (Chakraborty et al., 2017) (Li et al., 2022a) (Cheng et al., 2021).
8.3. Techniques and Approaches to Explainable AI

Buhrmester et al. (2021) categorised explainable methods as either white or black box, ante-hoc or post-hoc, global or local. White box methods have access to a deep network and provide details of actions taking place in the network. Black-box, on the other hand, does not access the details of the network; however, its explanation is based on the outcome of the network. Methods described as ante-hoc belong to the category of white box because the model explanation begins from the input and scans through parameters one after the other to account for every action in the model. Post-hoc methods belong to the black box class as they explain the model considering only the aspect involved in generating the output.

Feature Importance. : Feature Importance Techniques: These techniques inspect the contribution of input features and allocate scores based on their importance in the model’s prediction. The techniques include; Local Interpretable Model-agnostic Explanation (Zhao et al., 2021; Schlegel et al., 2021), Permutation Importance (Shevskaya, 2021; Lucas et al., 2022), and SHapley Additive exPlanation (SHAP) (Park et al., 2020; Walia et al., 2022; Brusa et al., 2023). Another feature importance approach is constructing a decision trees model (Dikmen and Burns, 2022; Paudel et al., 2023) or constructing a rules-based model to explain the decision-making process of some complex models. The decision boundaries learned by the model are transparently represented with decision trees. At the same time, the rule-based sees the generation of conditional rules in the human-readable form to explain model predictions (Vilone et al., 2020; Gohel et al., 2021).

Local Explanation. One way to understand the relationship between the model predictions and the features is through partial dependence plots, individual condition expectation (ICE) plots or counterfactual explanations. Partial dependence plots help explain how each feature affects the model predictions. ICE plots help visualize the effect of individual instances on the model’s predicted outcome. Finally, counterfactual explanations involve generating another model using counter-input instances to help understand how the model responds to input changes. These techniques have been discussed in detail in (Lundberg et al., 2019; Guidotti et al., 2021; Qiu et al., 2021).

Model Specific Technique. : This technique employs layer-wise relevance propagation or attention mechanism. Layer-wise relevance propagation involves the decomposition of the model’s prediction by assigning importance
score to features in each layer of a neural network and also providing information about what each component has contributed to the model’s final predictions (Wenli et al., 2023). Attention mechanisms worked with the model that implements attention mechanism, and it visualises the learning weight of the model to figure out the aspect of the input sequence that the model considered for its predictions (Bendre et al., 2021; Ntrougkas et al., 2022; Gkartzonika et al., 2023).

Post Hoc Interpretable technique. comprises model Distillation and surrogate model techniques. Model distillation tries to mimic a complex model behaviour by training a simple and interpretable model to provide a more transparent option for decision-making (Jonsson Ewerbring, 2021; Li et al., 2022b). The surrogate model on the alternatives approximates the decision function of a black box model by fitting an interpretable model to its prediction for better interpretation and explanation (Tan and Kotthaus, 2022).

User Interaction and Explanation Interfaces. This involves the development of interactive and visualization tools that enable users to interact with model predictions and explanations. Likewise, Natural language Explanations could also aid in presenting model predictions’ explanations in a human-readable format so that even non-technical users could access it (Kadir et al., 2023; Rjoob et al., 2021).

Saliency Maps and Gradient-based Methods. This approach consists of two categories: gradient-based saliency map and SmoothGrad. The gradient-based saliency map aims to identify the regions of the inputs most influencing the model’s prediction. It does so by computing the gradient of the output with respect to the input features and highlighting them. Visual explanations can be provided through Gradient-based Class Activation Mapping (Grad-CAM) and Integrated Gradients, which assign relevance scores to input features. SmoothGrad, on the other hand, introduces noise to the input features and generates a smoother saliency map from the average of the gradients. This helps reduce noise and highlight more features.

8.4. Hybrid models and their potential in advancing deep learning

The known fact about machine learning and deep learning models is that no models have the optimal capacity for all tasks. Each model has its strengths and weaknesses. As a result, models can be hybridized to complement each other. Hybridization combines two or more machine learning
models to leverage their strengths and resolve their limitations. Deep learning architecture could be hybridized with some machine learning architectures. Also, hybridization could take place among different deep learning architectures.

**CNN and Shallow learners hybrid models.** Convolution Neural network is popularly known for its uniqueness in extracting representative features from data. Likewise, the literature has reported that the performance of some machine learning algorithms depends greatly on feature extraction techniques Ekundayo and Viriri (2021). Ekundayo and Viriri (2021) reported that CNN features in combination with some efficient machine learning classifiers like SVM, RF, and decision trees could yield an optimal performance either in classification or regression tasks than to employ any of the models separately. Lekha and Suchetha (2018) hybridized CNN with SVM to detect chronic diseases in real-time.

Sharifrazi et al. (2021) designed a hybrid of CNN, SVM and Sobel filters to detect COVID-19 from X-ray images automatically. Using accuracy, sensitivity and specificity as evaluation metrics, they reported that The hybrid model gave optimal performance of 99.02% accuracy, 100% sensitivity and 95.23% specificity. The results are far better than the performance of the individual classifier model. Le et al. (2021) in a similar manner, proposed a depth-wise separable CNN in combination with deep support machine classifier for both binary and multiclass of Covid-19 from X-ray images with a report of optimal performance of the hybrid model. Deep CovNet has been hybridized with support vector machines to achieve optimal results in different application areas. For instance, AlexNet Covnet was hybridized with SVM to recognise surface defects in the images of hot-rolled steel strips (Adel et al., 2022). Hameed et al. (2018) also employed the feature extraction strength of AlexNet and the classification capability of a modified SVM called Error Correcting Output Code SVM (ECOC-SVM) to classify skin images into five different classes.

Hybridization of Convolution neural and SVM is not limited to aforementioned, it has been explored in the literature in differs ways to different areas of applications; classification tasks (Abdullahi et al., 2017) (Li et al., 2019a) (Manohar et al., 2019; Shrivastava et al., 2019; Hasan et al., 2019; Truong et al., 2021; Latha et al., 2021), medical imaging (Dina A et al., 2019; Oliver et al., 2020; Priya et al., 2021; Cui et al., 2019; Kibriya et al., 2021; Janghel and Rathore, 2021), forecasting (Cao and Wang, 2019; Zhang and Li, 2020).
face detection and recognition (Nwosu et al., 2017; Hui and Yu-jie, 2018), Fault detection (Li et al., 2019b; Et-taleby et al., 2022).

Furthermore, other machine learning classifiers hybridized with CNN are random forest or forest trees and decision trees. Like SVM, a hybridized model of CNN and Random forest has been applied severally in literature; Hui and Yu-jie (2018) mitigated the complexity of convolutional Neural Networks with random forest for extracting representative features in facial expression recognition tasks. Zheng et al. (2018a) combined CNN with random forest for recognising emotion in speech, CNN was employed to extract the representative feature of emotion from speech data, while Random Forest was used to classify the extracted feature into basic emotion. A CNN and Random Forest hybrid was also used for speech emotion classification in (Yalamanchili et al., 2023). It was reported that CNN-RF performed better than the CNN model. Liu et al. (2020) enhanced CNN multiple classifiers with Random Forest to develop a non-apnea sleep arousal detection system. CNN was first used in the preliminary classification of the selected representative signal of PSG (Polysomnography) data, and the result was sent into a Random Forest, which performs ensemble voting for final classification. The hybrid model comprised of CNN and Random Forest was proposed by (Xi, 2022) for image classification. The model performance was not limited to a high accuracy rate but also generalised well with unseen data compared to either of its components.

Hybrid models of CNN and Random Forest or decision trees have been applied to many different areas for classification, recognition or prediction purposes, which include: Agriculture (Bhavya et al., 2023; Sarkar and Chakroborty, 2023), Medical imaging (Reddy and Parvathy, 2022), Character recognition (Altememe and Abbadi, 2023).

Deep Learning and Metaheuristics Algorithms. Metaheuristic algorithms are known for their efficiency in performing an optimal search with less computational time and resource management. These features can be of significant advantage in mitigating the complexity of deep neural networks. Combining metaheuristic algorithms with a deep learning model can facilitate achieving a global minimum within a limited time. In (Mittal and Hasija, 2020), this approach has been considered for the appropriate selection of convolutional neural network’s weight and bias in developing a model for early detection of lung cancer. In (Olaide and Ezugwu, 2023), the Ebola algorithm was hybridized with a Generative Adversarial Network for cancer detection in
histopathological images. Another interesting hybridization of deep learning models with optimization algorithms is the combination of the Immune-Ebola optimization algorithm with a convolution network. (Olaide and Ezugwu, 2022) optimized the CNN model by performing a reduction of the representative feature extracted by CNN and presenting discriminating features to the CNN classifier. In (Olaide and Ezugwu, 2021), metaheuristic algorithms were employed for deep learning (CNN) hyperparameter tuning. In their experiments, they hybridized a CNN network with five different metaheuristic algorithms (genetic algorithm, whale optimization algorithm, life choice-based optimization algorithm, multiverse optimizer, and satin bower optimization). They observed the performance of the hybrid models. Their findings revealed that the hybrids with the metaheuristic algorithms (Multiverse optimizer, satin bower optimization, and life choice-based optimization) performed better than the hybrid with both genetic and whale optimisation algorithms. They concluded that physics-based, biology-based, and human-based metaheuristics have higher potential for deep network hyperparameter tuning than evolutionary and swarm-based counterparts. Recently, metaheuristics have continued to evolve either as feature selection techniques for deep models or as deep architecture weight and bias optimizers (Chiroma et al., 2020; Elsayed Abd Elaziz et al., 2021; Saadi et al., 2022).

**CNN and RNN hybrid Models.** This hybridization model combines the feature extraction strength of CNN with the sequential processing prowess of RNN. Dutta et al. (2018) capitalised on the uniqueness of deep architectures (CNN and RNN) in processing large volumes of data for handwritten recognition tasks. The hybrid model engaged CNN with extracting representative features from the sequential data, which immuned the model against invariances challenges in handwritten image data. Shi et al. (2019) showed that 2D-CNN-RNN and 3D-CNN-RNN outperformed CNN and RNN models in the automatic analysis of electroencephalograms (EEG) for Parkinson’s disease detection. Likewise, LSTM was developed over a pre-trained DenseNet to resolve the issue of inter-slice correlation in MRI data. Ajao et al. (2018); Nasir et al. (2021); Goonathilake and Kumara (2020) hybridized CNN with RNN and CNN with LSTM for automatic detection of fake news from social media datasets. They reported that the hybridized models gave superior performance compared to individual models. Hybridization of CNN, RNN and LSTM has been considered in applications like emotion, gesture, speech recognition (Fan et al., 2016; Arshad et al., 2022), diagnosis of diseases in
medical imaging (Lane and Kahanda, 2021; Uly et al., 2023) (Datta and Rohilla, 2024), intrusion detection (Deore and Bhosale, 2022; Halbouni et al., 2022), Text classification and Character recognition (She and Zhang, 2018; Jain et al., 2017; Hsu et al., 2017)

**ConvNets with Attention Mechanisms.** Deep Convolutional Neural Networks (CNN), such as DenseNet, ResNet, and GoogLeNet, can benefit from adding an attention mechanism. This allows the deep network to focus on relevant features in the input data, making it more efficient and effective. Attention mechanisms can be integrated into various tasks, such as image classification, disease recognition, and classification from medical imagery. For example, GoogLeNet with attention mechanism was employed in classifying rice diseases from rice leaves, as shown in the study by (Yang et al., 2023).

**Capsule Networks (CapsNets) with Recurrent structures.** Capsule Networks (CapsNets) are known for their ability to capture hierarchical relationships between features. When combined with recurrent features of Recurrent Neural Networks (RNN), this hybridization can be useful in environments that require memory over time. Studies such as (Wu et al., 2021; Sezavar et al., 2024; Ali et al., 2024) have demonstrated the optimal performance of CapsNets with recurrent structures.

**Autoencoder Variants with Recurrent Connections.** A combination of autoencoder variants with recurrent neural network a suitable hybrid model for tasks like sequence generation (Jang et al., 2019), video reconstruction (D’Avino et al., 2017) and anomaly detection (Yin et al., 2022).

**Graph Neural Networks (GNNs) with Attention Mechanisms.** Combining a Graph Neural Network (GNN) with an attention mechanism can improve the GNN’s ability to focus on more relevant nodes and links within a graph (Zhou and Li, 2021). This hybrid model is suitable for tasks such as link, node, and graph classification (Yang et al., 2019a) (Sun et al., 2021). Additionally, if a transformer model, initially designed for sequence-sequence tasks, is combined with a convolutional or recurrent layer, it can be applied to spatial-related or sequential tasks.

**Transformer Models with Sequential Components.** Application of the transformer model can be extended if its model initially designed for sequence-sequence tasks is hybridized with either a convolutional layer or recurrent layer in a spatial-related environment or sequential tasks.

Deep learning is a powerful tool in artificial intelligence that has proven relevant in today’s world due to its outstanding performance and future potential. The diversity of deep learning models is an advantage for real-world applications as different sectors increasingly embrace artificial intelligence, and the use of deep learning continues to spread. This section will cover various sectors where deep learning is applicable:

*Education.* Deep learning has brought significant advancements to education, including teaching, research, and administration. Educators and school administrators can use deep learning techniques for pattern recognition to analyze student data, make informed decisions, and adjust curriculums and teaching methodologies. With the adaptive nature of deep learning, pedagogical systems can be developed to provide quick feedback and assistance to students facing challenges in their learning journey. Furthermore, deep learning’s text sequencing and generation capabilities can be used to create fair assessments and grading methods for students’ exams, quizzes, and other formal assessments, which can help reduce educators’ academic burdens. Additionally, deep learning’s Natural Language Processing capabilities can be used to create advisory systems like chatbots that can facilitate effective interactive communication with students, addressing their queries and concerns (Zhang, 2020). COVID-19 created a surge in Online teaching (Bhardwaj et al., 2021), and one of the advantages of physical contact is the ability of an educator to periodically assess the mind state of the students for possible adjustment to the teaching method. This dynamism in class can be implemented in online classes with deep learning facial expression analysis techniques (Ekundayo and Viriri, 2021).

*Security.* Security is a significant concern across the world. It is essential to protect lives, assets, and information. With the help of deep learning models, real-time analysis of video streams in surveillance systems is possible. It can identify suspicious activities or individuals in specific environments. Deep learning is highly effective in detecting anomalies, preventing and detecting malware, and improving data and system security (Zeng et al., 2020). It can efficiently detect phishing attempts and categorize document or email content as malicious or fraudulent. Deep learning’s pattern recognition capabilities are also important in developing biometric systems. It can analyze biometric data to forestall unauthorized access attempts.
Military. Deep learning is a valuable tool in the military sector for target detection and enemy recognition (Das et al., 2018). Many advanced countries currently rely on autonomous systems achieved through deep reinforcement learning. Military surveillance and spy systems are autonomous machines, such as unmanned aerial vehicles (Zhang et al., 2020b), drones, and underwater vehicles. Securing military information is crucial both for the army and the nation. Deep learning can help prevent any suspicious attack on the military network and prevent impostors from accessing helpful information by analyzing network traffic, security events, and system logs (Kara-can and Sevri, 2021). Deep learning can be used to develop intrusion and anomaly detection systems to prevent military networks from cyber attacks (Vinayakumar et al., 2019). Another feature of deep learning is its ability to process and analyze large amounts of language. It can help to analyze intercepted communication and intelligent reports and content from social media to deduce action-based information and identify threat communications. In multinational operations where language may hinder effective operation, deep learning language translation can greatly help (Ali et al., 2021).

Health. Deep learning demonstrates efficiency and robustness in processing and analyzing sequential data, which is particularly valuable in medical applications such as genomics. This involves the analysis of genetic data, including DNA sequences, single nucleotide polymorphisms, and protein expression profiles for predicting disease risk and treatment response. Additionally, deep learning’s detection, pattern recognition, and classification capabilities are adept at identifying and categorizing diseases in medical imaging. Moreover, deep learning contributes to molecular structure generation and drug target prediction in pharmaceutical research and development. By integrating computer vision techniques, deep learning enhances the precision and flexibility of surgical robots, thereby improving surgical outcomes (Mittal and Hasija, 2020) (Sapoval et al., 2022).

Agriculture. In recent years, agriculture has undergone a technological revolution, integrating deep learning into the analysis of satellite imagery and Internet of Things (IoT) sensor (Sishodia et al., 2020) data to assess crop environmental conditions, growth status, and overall health. This information empowers farmers to promptly identify crop deficiencies, such as nutrient deficiencies and pest and insect infestations. Furthermore, deep learning detection algorithms can accurately identify plant diseases from images, enabling early intervention and disease control measures. Deep learning also
plays a crucial role in developing autonomous weed detection and robotic weed removal, reducing the reliance on chemical herbicides and minimizing environmental pollution. In precision agriculture (Saranya et al., 2023), deep learning provides detailed insight into soil fertility, composition, and crop performance, aiding farmers in optimizing resource allocation and maximizing yields. Moreover, in livestock management, deep learning can analyze animal behaviour and environmental conditions to predict the well-being of animals, enabling proactive veterinary care and management practices (Zhang et al., 2020a) (Kamilaris and Prenafeta-Boldú, 2018).

*Transportation.* Self-driving automobiles are among the most well-known achievements in Artificial Intelligence (Muhammad et al., 2021). This accomplishment has been made possible through the use of deep learning models. By taking input from sensors, deep learning can process the input to learn its environment, detect surrounding objects, and be solely responsible for driving decisions in real-time. The deep learning detection algorithm is crucial in implementing driver-assisted systems (Lv et al., 2024) and pedestrian detection and safety. Deep learning applications in transportation tend to reduce accidents due to human errors.

*Meteorology.* Deep learning performance is determined by several factors, one of which is the availability of large amounts of data. When applying deep learning to a substantial volume of meteorological data, it can accurately predict short- and long-term weather patterns (Ren et al., 2021). To increase the accuracy of real-time weather prediction, the spatiotemporal nature of meteorological data, such as satellite data, remote sensor data, and radar data, can be modeled using RNN or LSTM (Miao et al., 2020). Similarly, Deep Learning can predict air quality indices and the concentration of pollutants by using meteorological variables, emission sources, and atmospheric dynamics. Thus, a deep learning model can be employed in air pollution control (Zhang et al., 2024).

10. Conclusion

10.1. Summary of Deep Learning Historical Review

The state-of-the-art model (Deep Learning) has significantly evolved and is recognised as the leading model with excellent efficiency in AI. The initial purpose of its creation emerged from introducing the concept of biological
neurons into machines. The early machines (MP and Rosenblatt machines) could solve linear problems and efficiently recognize some logical patterns except "XOR". The limitations of the early models were the reason for the first winter break experienced in the evolution of neural networks when there was little or no financial support for the project.

Breakthrough in backpropagation reintroduced neural networks because an efficient way of implementing stochastic gradient descent using chain rule is now available. With backpropagation as the learning technique, Deep neural networks emerged, such as Feedforward Neural Networks, Recurrent Neural Networks, Long short-term memory with high efficiency and more applications environment. However, another recess set is due to high computational complexity, unavailable system resources for implementing backpropagation, and significant data challenges.

Deep Learning finally came out of recess after introducing the Internet, through which large volumes of data can be easily generated. Also, high computing devices like GPU and extensive system resources are available. Since then, deep learning has kept evolving. LeNet considered the basic CovNet, has evolved into different versions of CovNets like ResNet, DenseNet, GoogLeNet, etc. These convolution Neural Networks are known for their efficiency in computer vision tasks. Likewise, sequence processing deep neural models also experience evolution in the same respect, from neocognitron to RNN to LSTM.

The evolution of deep learning has spread in conjunction with another aspect of learning called reinforcement learning. Deep learning is now the search mechanism employed in reinforcement learning, and these two learning paradigms are evolving greatly in their applications to different environments. Introducing deep learning into reinforcement learning increases the taxonomy of deep learning into supervised, unsupervised and reinforcement learning.

10.2. Implications and Future Directions for Deep Learning Research and Applications

The development of backpropagation marked a significant advance in the effectiveness of learning procedures for deep learning after exploring several options. The consensus in the field is that deeper networks perform better, but the ability of a deep model to generalize depends heavily on the availability of sufficient data. Deep models tend to overfit if there is a limited amount of data. In addition to data challenges, deeper networks have more parameters, creating parameter-tuning challenges. The creation of data and
the identification of effective learning parameters for deep models require further investigation.

One of the main challenges in deep networks is the fine-tuning of parameters. Additionally, as the depth of the network increases, it becomes more susceptible to vanishing and exploding gradients. This is particularly relevant to the Convolution Neural Network and its variants. Similarly, recurrence in RNN and LSTM models also presents a significant challenge. There is a need for better methods to implement memory in these models that will eliminate recurrence and improve computation time and resource management efficiency.

Deep reinforcement learning (DRL) has been gaining attention across various domains beyond its traditional games, autonomous driving, and robotics applications. However, some notable challenges still need to be addressed to enhance its performance. One major challenge is the time complexity associated with the model’s learning of environmental policies. Finding efficient methods to accomplish this within limited time constraints remains an unsolved problem. Moreover, deep learning plays a crucial role in DRL’s policy and rule learning process, making it susceptible to issues like hyperparameter sensitivity and problems with exploding and vanishing gradients. Addressing or mitigating these challenges to a manageable extent is a primary research focus in the field.

How best can the distinct strengths inherent in individual deep learning models be harnessed to develop hybridised models that can be adaptable and generalised across various environments? This is a significant question to be considered in the field. Similarly, the opaque nature of deep learning implementations limits users’ understanding and participation in decision-making processes. Thus, there is a need in the field for user-friendly and effective visualization techniques that provide insight into the inner workings of these "black box" models and encourage user engagement in decision-making.

**Conflict of Interest:** The authors declare that there is no conflict of interest regarding the publication of this paper.

**References**


Ali, J., Saleem, N., Bourouis, S., Alabdulkreem, E., Mannai, H.E., Dhabbi, S., 2024. Spatio-temporal features representation using recurrent cap-


Chen, K., Cosgro, B., Domfeh, O., Stern, A., Korkmaz, G., Kattampallil, N.A., 2021a. Leveraging google bert to detect and


67


87


