**REVIEW**

# A Comprehensive Overview and Comparative Analysis on Deep Learning Models

**Farhad Mortezapour Shiri**[*]**, Thinagaran Perumal, Norwati Mustapha and Raihani Mohamed**

Faculty of Computer Science and Information Technology, University Putra Malaysia (UPM), Serdang, 43400, Malaysia

*Corresponding Author: Farhad Mortezapour Shiri. Email: gs63904@student.upm.edu.my

## ABSTRACT

Deep learning (DL) has emerged as a powerful subset of machine learning (ML) and artificial intelligence (AI), outperforming traditional ML methods, especially in handling unstructured and large datasets. Its impact spans across various domains, including speech recognition, healthcare, autonomous vehicles, cybersecurity, predictive analytics, and more. However, the complexity and dynamic nature of real-world problems present challenges in designing effective deep learning models. Consequently, several deep learning models have been developed to address different problems and applications. In this article, we conduct a comprehensive survey of various deep learning models, including Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Temporal Convolutional Networks (TCN), Transformer, Kolmogorov-Arnold Networks (KAN), Generative Models, Deep Reinforcement Learning (DRL), and Deep Transfer Learning. We examine the structure, applications, benefits, and limitations of each model. Furthermore, we perform an analysis using three publicly available datasets: IMDB, ARAS, and Fruit-360. We compared the performance of six renowned deep learning models: CNN, RNN, Long Short-Term Memory (LSTM), Bidirectional LSTM, Gated Recurrent Unit (GRU), and Bidirectional GRU alongside two newer models, TCN and Transformer, using the IMDB and ARAS datasets. Additionally, we evaluated the performance of eight CNN-based models, including VGG (Visual Geometry Group), Inception, ResNet (Residual Network), InceptionResNet, Xception (Extreme Inception), MobileNet, DenseNet (Dense Convolutional Network), and NASNet (Neural Architecture Search Network), for image classification tasks using the Fruit-360 dataset.

## KEYWORDS

Deep learning; Convolutional Neural Network (CNN); Long Short-Term Memory (LSTM); Gated Recurrent Unit (GRU); Temporal Convolutional Network (TCN); Transformer; Kolmogorov-Arnold Networks (KAN); Deep Reinforcement Learning (DRL); Deep Transfer Learning (DTL)

## 1 Introduction

Artificial intelligence (AI) aims to emulate human-level intelligence in machines. In computer science, AI refers to the study of "intelligent agents," which are objects capable of perceiving their environment and taking actions to maximize their chances of achieving specific goals [1]. Machine learning (ML) is a field that focuses on the development and application of methods capable of learning

from datasets [2]. ML finds extensive use in various domains, such as speech recognition, computer vision, text analysis, video games, medical sciences, and cybersecurity.

In recent years, deep learning (DL) techniques, a subset of machine learning (ML), have outperformed traditional ML approaches across numerous tasks, driven by several critical advancements [3]. The proliferation of large datasets has been pivotal in enabling models to learn intricate patterns and relationships, thereby significantly enhancing their performance [4]. Concurrently, advancements in hardware acceleration technologies, notably Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs) [5] have markedly reduced model training times by facilitating rapid computations and parallel processing capabilities. These advancements have substantially accelerated the training process. Moreover, enhancements in algorithmic techniques for optimization and training have further augmented the speed and efficiency of deep learning models, leading to quicker convergence and superior generalization capabilities [4]. Deep learning techniques have demonstrated remarkable success across a wide range of applications, including computer vision (CV), natural language processing (NLP), and speech recognition. These applications underscore the transformative impact of DL in various domains, where it continues to set new performance benchmarks [6,7].

Deep learning models draw inspiration from the structure and functionality of the human nervous system and brain. These models employ input, hidden, and output layers to organize processing units. Within each layer, the nodes or units are interconnected with those in the layer below, and each connection is assigned a weight value. The units sum the inputs after multiplying them by their corresponding weights [8]. Fig. 1 illustrates the relationship between AI, ML, and DL, highlighting that machine learning and deep learning are subfields of artificial intelligence.
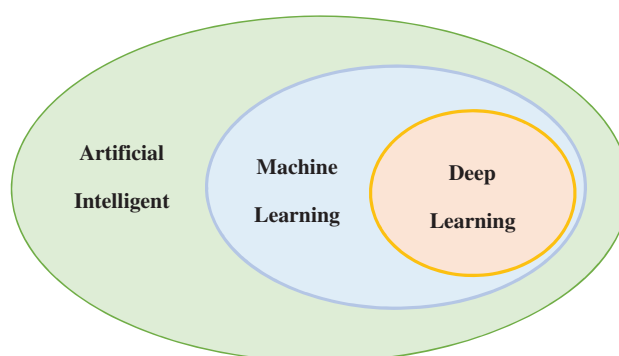


**Figure 1:** Relationship between artificial intelligence (AI), machine learning (ML), and deep learning (DL)

The objective of this research is to provide a comprehensive overview of various deep learning models and compare their performance across different applications. In Section 2, we introduce a fundamental definition of deep learning. Section 3 covers supervised deep learning models, including Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Temporal Convolutional Networks (TCN), and Kolmogorov-Arnold Networks (KAN). Section 4 reviews generative models such as Autoencoders, Generative Adversarial Networks (GANs), and Deep Belief Networks (DBNs). Section 5 presents a comprehensive survey of the Transformer architecture. Deep Reinforcement Learning (DRL) is discussed in Section 6, while Section 7 addresses Deep Transfer Learning (DTL). The principles of hybrid deep learning are explored in Section 8, followed by a discussion of deep learning applications in Section 9. Section 10 surveys the challenges in deep learning and potential alternative solutions. In Section 11, we conduct experiments and analyze

the performance of different deep learning models using three datasets. Research directions and future aspects are covered in Section 12. Finally, Section 13 concludes the paper.

## 2 Deep Learning

Deep learning (DL) involves the process of learning hierarchical representations of data by utilizing architectures with multiple hidden layers. With the advancement of high-performance computing facilities, deep learning techniques using deep neural networks have gained increasing popularity [9]. In a deep learning algorithm, data is passed through multiple layers, with each layer progressively extracting features and transmitting information to the subsequent layer. The initial layers extract low-level characteristics, which are then combined by later layers to form a comprehensive representation [6].

In traditional machine learning techniques, the classification task typically involves a sequential process that includes pre-processing, feature extraction, meticulous feature selection, learning, and classification. The effectiveness of machine learning methods heavily relies on accurate feature selection, as biased feature selection can lead to incorrect class classification. In contrast, deep learning models enable simultaneous learning and classification, eliminating the need for separate steps. This capability makes deep learning particularly advantageous for automating feature learning across diverse tasks [10]. Fig. 2 visually illustrates the distinction between deep learning and traditional machine learning in terms of feature extraction and learning.
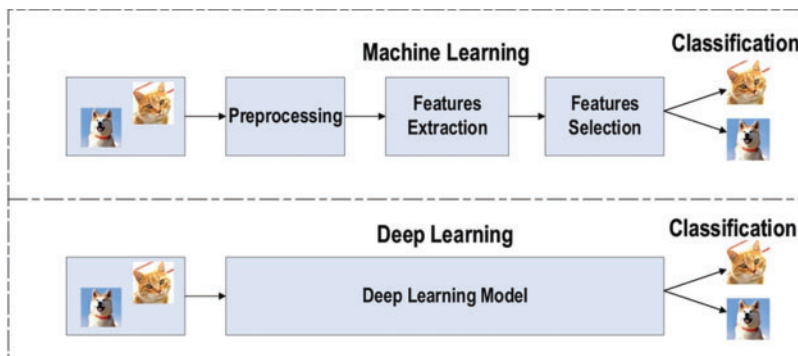


**Figure 2:** Visual illustration of the distinction between deep learning and traditional machine learning in terms of feature extraction and learning [10]

In the era of deep learning, a wide array of methods and architectures have been developed. These models can be broadly categorized into two main groups: discriminative (supervised) and generative (unsupervised) approaches. Among the discriminative models, two prominent groups are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Additionally, generative approaches encompass various models such as Generative Adversarial Networks (GANs) and Auto-Encoders (AEs) [11]. In the following sections, we provide a comprehensive survey of different types of deep learning models.

## 3 Supervised Deep Learning Models

In supervised learning and classification tasks, this family of deep learning algorithms is used to perform discriminative functions. These supervised deep architectures typically model the posterior distributions of classes based on observable data, enabling effective pattern classification. Common

supervised models include Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Temporal Convolutional Networks (TCN), Kolmogorov-Arnold Networks (KAN), and their variations. A brief overview of these methods are as follows.

### 3.1 Multi Layers Perceptron (MLP)

The Multi-Layer Perceptron (MLP) model is a type of feedforward Artificial Neural Network (ANN) that serves as a foundation architecture for deep learning or Deep Neural Networks (DNNs) [11]. It operates as a supervised learning approach. The MLP consists of three layers: the input layer, the output layer, and one or more hidden layers [12]. It is a fully connected network, meaning each neuron in one layer is connected to all neurons in the subsequent layer.

In an MLP, the input layer receives the input data and performs feature normalization. The hidden layers, which can vary in number, process the input signals. The output layer makes decisions or predictions based on the processed information [13]. Fig. 3a depicts a single-neuron perceptron model, where the activation function $\varphi$ (Eq. (1)) is a non-linear function used to map the summation function $(xw + b)$ to the output value $y$.

$$y = \varphi\,(xw + b) \tag{1}$$

In Eq. (1), the terms $x$, $w$, $b$, and $y$ represent the input vector, weighting vector, bias, and output value, respectively [14]. Fig. 3b illustrates the structure of the Multi-Layer Perceptron (MLP) model.
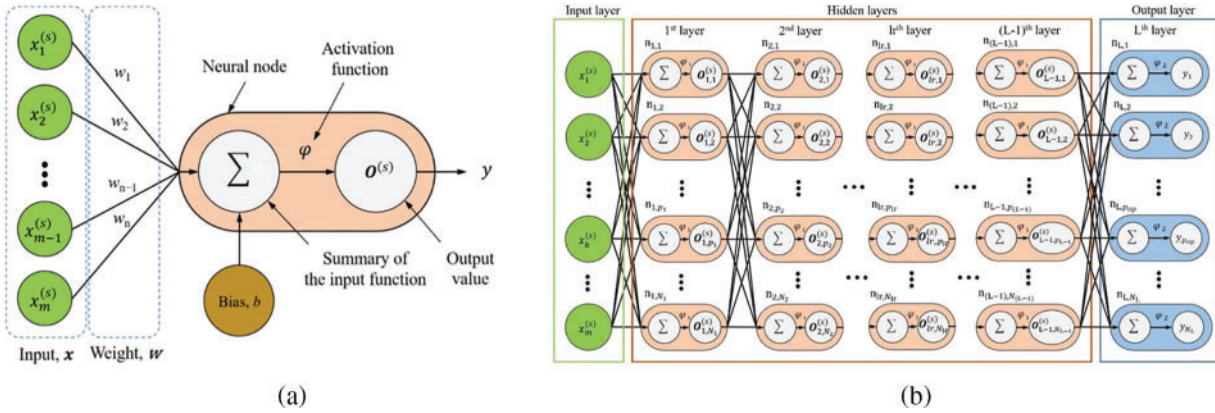


**Figure 3:** (a) Single-neuron perceptron model. (b) Structure of the MLP [14]

### 3.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a powerful class of deep learning models widely applied in various tasks, including object detection, speech recognition, computer vision, image classification, and bioinformatics [15]. They have also demonstrated success in time series prediction tasks [16]. CNNs are feedforward neural networks that leverage convolutional structures to extract features from data [17]. CNN has a two-stage architecture that combines a classifier and a feature extractor to provide automatic feature extraction and end-to-end training with the least amount of pre-processing necessary [18]. Unlike traditional methods, CNNs automatically learn and recognize features from the data without the need for manual feature extraction by humans [19]. The design of CNNs is inspired by visual perception [17]. The major components of CNNs include the convolutional layer, pooling layer, fully connected layer, and activation function [20,21]. Fig. 4 presents the pipeline

of the convolutional neural network, highlighting how each layer contributes to the efficient processing and successful progression of input data through the network.
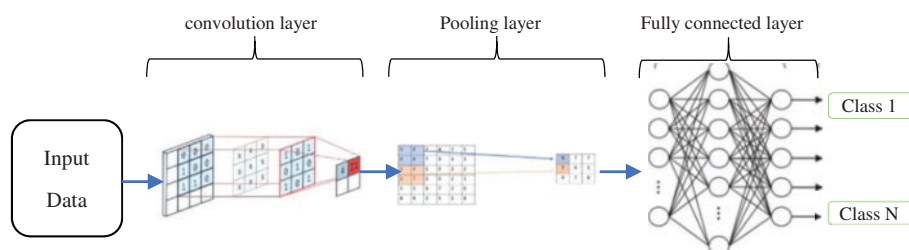


**Figure 4:** The pipeline of a Convolutional Neural Network

**Convolutional Layer:** The convolutional layer is a pivotal component of CNN. Through multiple convolutional layers, the convolution operation extracts distinct features from the input. In image classification, lower layers tend to capture basic features such as texture, lines, and edges, while higher layers extract more abstract features. The convolutional layer comprises learnable convolution kernels, which are weight matrices typically of equal length, width, and an odd number (e.g., $3 \times 3$, $5 \times 5$, or $7 \times 7$). These kernels are convolved with the input feature maps, sliding over the regions of the feature map and executing convolution operations [22]. Fig. 5 illustrates the schematic diagram of the convolution process.
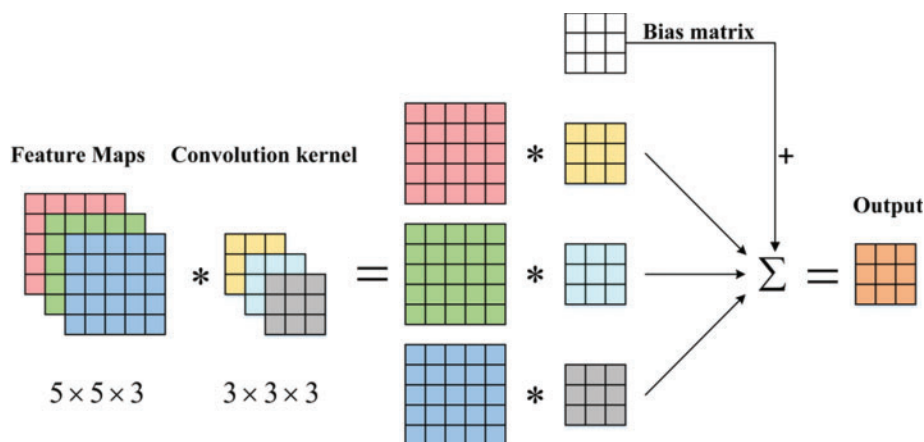


**Figure 5:** Schematic diagram of the convolution process [22]

**Pooling Layer:** Typically following the convolutional layer, the pooling layer reduces the number of connections in the network by performing down-sampling and dimensionality reduction on the input data [23]. Its primary purpose is to alleviate the computational burden and address overfitting issues [24]. Moreover, the pooling layer enables CNN to recognize objects even when their shapes are distorted or viewed from different angles, by incorporating various dimensions of an image through pooling [25]. The pooling operation produces output feature maps that are more robust against distortion and errors in individual neurons [26]. There are various pooling methods, including Max Pooling, Average Pooling, Spatial Pyramid Pooling, Mixed Pooling, Multi-Scale Order-Less, and Stochastic Pooling [27–30]. Fig. 6 depicts an example of Max Pooling, where a window slides across the input, and the contents of the window are processed by a pooling function [31].
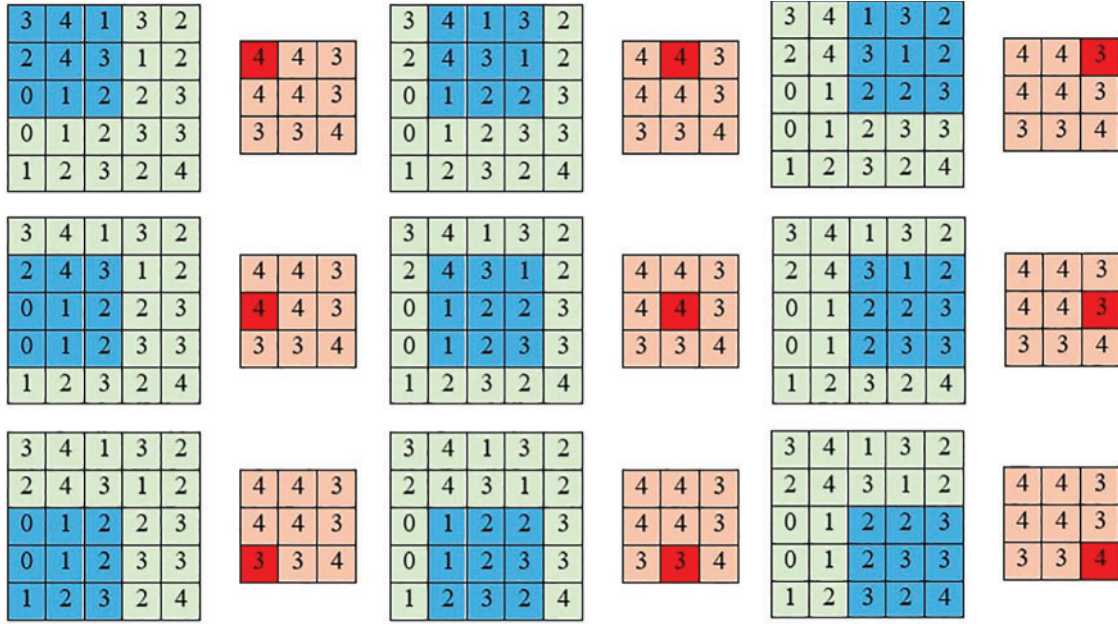
**Figure 6:** Computing the output values of a 3 × 3 max pooling operation on a 5 × 5 input

**Fully Connected (FC) Layer:** The FC layer is typically located at the end of a CNN architecture. In this layer, every neuron is connected to all neurons in the preceding layer, adhering to the principles of a conventional multi-layer perceptron neural network. The FC layer receives input from the last pooling or convolutional layer, which is a vector created by flattening the feature maps. The FC layer serves as the classifier in the CNN, enabling the network to make predictions [10].

**Activation Functions:** Activation functions are fundamental components in convolutional neural networks (CNNs), indispensable for introducing non-linearity into the network. This non-linearity is crucial for CNN's ability to model complex patterns and relationships within the data, allowing it to perform tasks beyond simple linear classification or regression. Without non-linear activation functions, a CNN would be limited to linear operations, significantly constraining its capacity to accurately represent the intricate, non-linear behaviors typical of many real-world phenomena [32].

Fig. 7 typically illustrates how these activation functions modulate input signals to produce output, highlighting the non-linear transformations applied to the input data across different regions of the function curve. In this figure, $x_i$ represents the input feature, while $w_{ij}$ denotes the weight associated with the connection between the input feature $x_i$ and neuron $j$. The figure shows that neuron $j$ receives $n$ features simultaneously. The output from neuron $j$ is labeled by $y_j$, and its internal state, or bias, is indicated by $b_j$. The activation function, depicted as $f(.)$, could be any one of several types such as the Rectified Linear Unit (ReLU), hyperbolic tangent (Tanh), Sigmoid function, or others [33,34].

These various activation functions are shown in Fig. 8, with emphasis on their distinct characteristics and profiles. These activation functions are essential for convolutional neural networks (CNNs) to be more effective in a variety of applications by allowing them to recognize intricate patterns and provide accurate predictions. Sigmoid and Tanh functions are frequently referred to as saturating nonlinearities due to the way they act when inputs are very large or small. As per the reference, the Sigmoid function approaches values of 0 or 1, whereas the Tanh function leans towards −1 or 1 [17].

Different alternative nonlinearities have been suggested for reducing problems associated with these saturating effects, including Rectified Linear Unit (ReLU) [35], Leaky ReLU [36], Parametric Rectified Linear Units (PReLU) [37], Randomized Leaky ReLU (RReLU) [38], S-shaped ReLU (SReLU) [39], and Exponential Linear Units (ELUs) [40], Gaussian Error Linear Units (GELUs) [41].
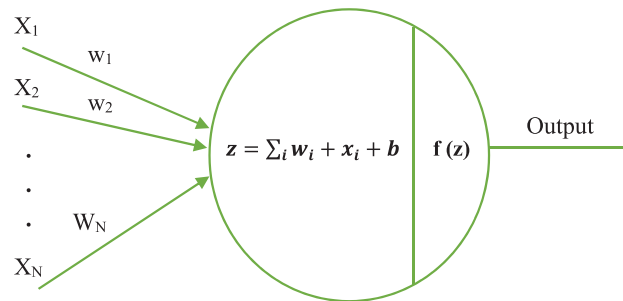


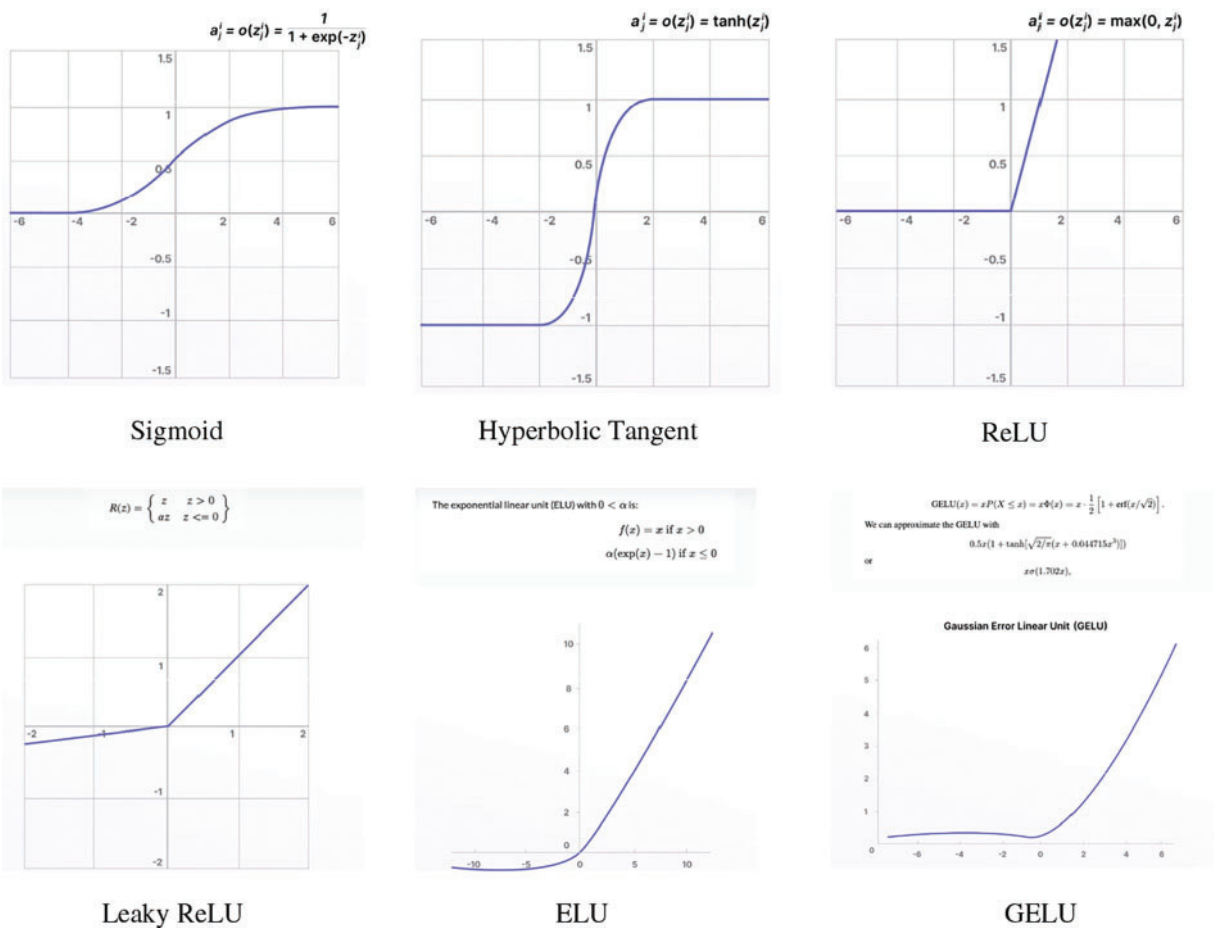**Figure 7:** The general structure of activation functions



**Figure 8:** Diagram of different activation functions

ReLU (Rectified Linear Unit) is one of the most often used activation functions in modern CNNs because of how well it solves the vanishing gradient issue during training. The definition of ReLU in mathematics is as Eq. (2), where the input to the neuron is represented by $x$ [34].

$$f(x) = \max\ (0,\ x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \tag{2}$$

This feature helps CNN learn complicated features more efficiently by effectively "turning off" any negative input values while maintaining positive values. It also keeps neurons from being saturated during training.

As an alternative, the definition of the Sigmoid function is represented by Eq. (3), where $x$ stands for the input of the neuron.

$$f(x) = \frac{1}{e^{-x}} \tag{3}$$

Although the sigmoid distinctive S-shape and capacity to condense real numbers into a range between 0 and 1 make it useful for binary classification, its propensity to saturate can hinder training by causing the vanishing gradient problem in deep neural networks.

Convolutional Neural Networks (CNNs) are extensively used in various fields, including natural language processing, image segmentation, image analysis, video analysis, and more. Several CNN variations have been developed, such as AlexNet [42], VGG (Visual Geometry Group) [43], Inception [44,45], ResNet (Residual Networks) [46,47], WideResNet [48], FractalNet [49], SqueezeNet [50], InceptionResNet [51], Xception (Extreme Inception) [52], MobileNet [53,54], DenseNet (Dense Convolutional Network) [55], SENet (Squeeze-and-Excitation Network) [56], Efficientnet [57,58] among others. These variants are applied in different application areas based on their learning capabilities and performance.

### 3.3 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are a class of deep learning models that possess internal memory, enabling them to capture sequential dependencies. Unlike traditional neural networks that treat inputs as independent entities, RNNs consider the temporal order of inputs, making them suitable for tasks involving sequential information [59]. By employing a loop, RNNs apply the same operation to each element in a series, with the current computation depending on both the current input and the previous computations [60].

The ability of RNNs to utilize contextual information is particularly valuable in tasks such as natural language processing, video classification, and speech recognition. For example, in language modeling, understanding the preceding words in a sentence is crucial for predicting the next word. RNNs excel at capturing such dependencies due to their recurrent nature [61–63].

However, a limitation of simple RNN is their short-term memory, which restricts their ability to retain information over long sequences [64]. To overcome this, more advanced RNN variants have been developed, including Long Short-Term Memory (LSTM) [65], bidirectional LSTM [66], Gated Recurrent Unit (GRU) [67], bidirectional GRU [68], Bayesian RNN [69], and others.

Fig. 9 depicts a simple recurrent neural network, where the internal memory ($h_t$) is computed using Eq. (4) [70].

$$h_t = g\ (Wx_t + Uh_t + b) \tag{4}$$

In this equation, $g()$ represents the activation function (typically Tanh), $U$ and $W$ are adjustable weight matrices for the hidden state ($h$), $b$ is the bias, and $x$ denotes the input vector.
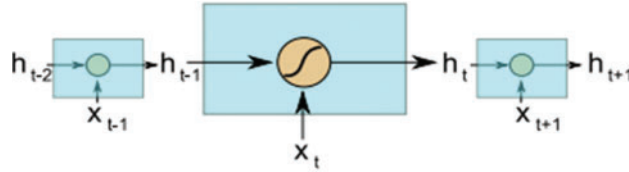


**Figure 9:** Simple RNN internal operation [70]

RNNs have proven to be powerful models for processing sequential data, leveraging their ability to capture dependencies over time. The various types of RNN models, such as LSTM, bidirectional LSTM, GRU, and bidirectional GRU, have been developed to address specific challenges in different applications.

### 3.3.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is an advanced variant of Recurrent Neural Networks (RNN) that addresses the issue of capturing long-term dependencies. LSTM was initially introduced by [65] in 1997 and further improved by [71] in 2013, gaining significant popularity in the deep learning community. Compared to standard RNN, LSTM models have proven to be more effective at retaining and utilizing information over longer sequences.

In an LSTM network, the current input at a specific time step and the output from the previous time step are fed into the LSTM unit, which then generates an output that is passed to the next time step. The final hidden layer of the last time step, sometimes along with all hidden layers, is commonly employed for classification purposes [72]. The overall architecture of an LSTM network is depicted in Fig. 10a. LSTM consists of three gates: input gate, forget gate, and output gate. Each gate performs a specific function in controlling the flow of information. The input gate decides how to update the internal state based on the current input and the previous internal state. The forget gate determines how much of the previous internal state should be forgotten. Finally, the output gate regulates the influence of the internal state on the system [60,73].
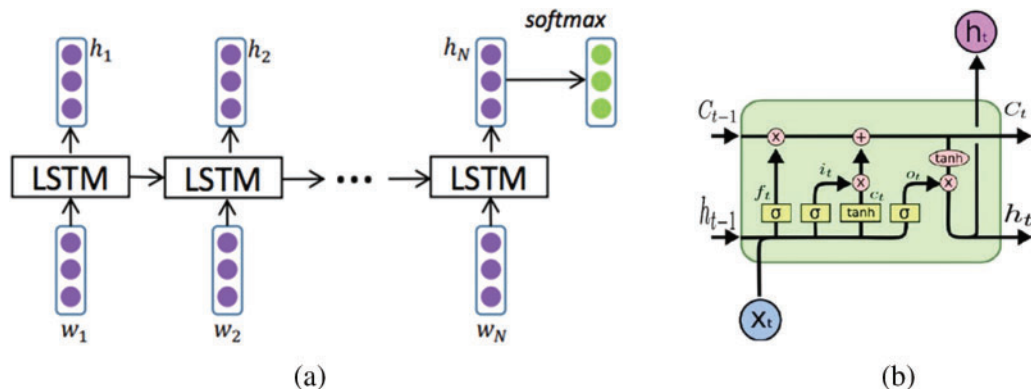


**Figure 10:** (a) The high-level architecture of LSTM. (b) The inner structure of LSTM unit [60]

Fig. 10b illustrates the update mechanism within the inner structure of an LSTM. The update for the LSTM unit is expressed by Eq. (5).

$$\begin{cases} h^{(t)} = g_o^{(t)} f_h\left(s^{(t)}\right) \\ s^{(t-1)} = g_f^{(t)} s^{(t-1)} + g_i^{(t)} f_s\left(wh^{(t-1)}\right) + uX^{(t)} + b \\ g_i^{(t)} = sigmoid\left(w_i h^{(t-1)} + u_i X^{(t)} + b_i\right) \\ g_f^{(t)} = sigmoid\left(w_f h^{(t-1)} + u_f X^{(t)} + b_f\right) \\ g_o^{(t)} = sigmoid\left(w_o h^{(t-1)} + u_o X^{(t)} + b_o\right) \end{cases} \tag{5}$$

where $f_h$ and $f_s$ represent the activation functions of the system state and internal state, typically utilizing the hyperbolic tangent function. The gating operation, denoted as g, is a feedforward neural network with a sigmoid activation function, ensuring output values within the range of [0, 1], which are interpreted as a set of weights. The subscripts $i$, $o$, and $f$ correspond to the input gate, output gate, and forget gate, respectively.

While standard LSTM has demonstrated promising performance in various tasks, it may struggle to comprehend input structures that are more complex than a sequential format. To address this limitation, a tree-structured LSTM network, known as S-LSTM, was proposed by [74]. S-LSTM consists of memory blocks comprising an input gate, two forget gates, a cell gate, and an output gate. While S-LSTM exhibits superior performance in challenging sequential modeling problems, it comes with higher computational complexity compared to standard LSTM [75].

### 3.3.2 Bidirectional LSTM

Bidirectional Long Short-Term Memory (Bi-LSTM) is an extension of the LSTM architecture that addresses the limitation of standard LSTM models by considering both past and future context in sequence modeling tasks. While traditional LSTM models process input data only in the forward direction, Bi-LSTM overcomes this limitation by training the model in two directions: forward and backward [76,77].

A Bi-LSTM consists of two parallel LSTM layers: one processes the input sequence in the forward direction, while the other processes it in the backward direction. The forward LSTM layer reads the input data from left to right, as indicated by the green arrow in Fig. 11. Simultaneously, the backward LSTM layer reads the input data from right to left, as represented by the red arrow [78]. This bidirectional processing enables the model to capture information from both past and future contexts, allowing for a more comprehensive understanding of temporal dependencies within the sequence.

During the training phase, the forward and backward LSTM layers independently extract features and update their internal states based on the input sequence. The output of each LSTM layer at each time step is a prediction score. These prediction scores are then combined using a weighted sum to generate the final output result [78]. By incorporating information from both directions, Bi-LSTM models can capture a broader context and improve the model's ability to model temporal dependencies in sequential data.

Bi-LSTM has been widely applied in various sequence modeling tasks such as natural language processing, speech recognition, and sentiment analysis. It has shown promising results in capturing complex patterns and dependencies in sequential data, making it a popular choice for tasks that require an understanding of both past and future context.
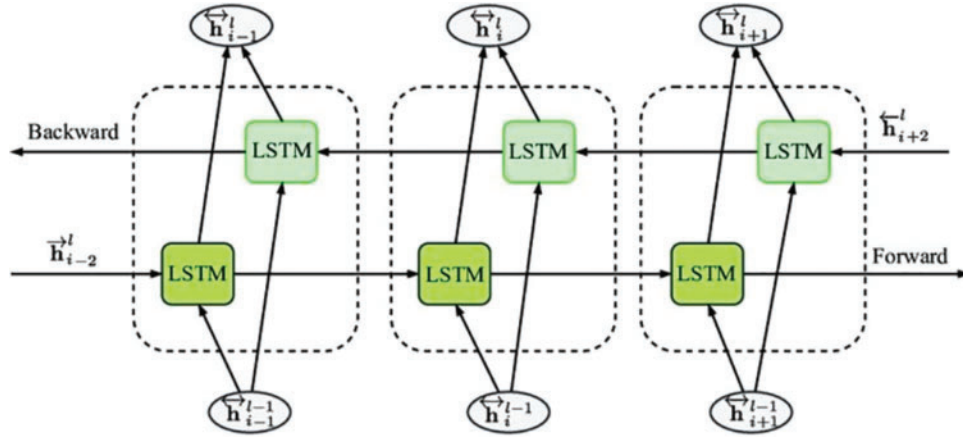
**Figure 11:** The architecture of a bidirectional LSTM model [76]

### 3.3.3 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is another variant of the RNN architecture that addresses the short-term memory issue and offers a simpler structure compared to LSTM [59]. GRU combines the input gate and forget gate of LSTM into a single update gate, resulting in a more streamlined design. Unlike LSTM, GRU does not include a separate cell state. A GRU unit consists of three main components: an update gate, a reset gate, and the current memory content. These gates enable the GRU to selectively update and utilize information from previous time steps, allowing it to capture long-term dependencies in sequences [79]. Fig. 12 illustrates the structure of a GRU unit [80].
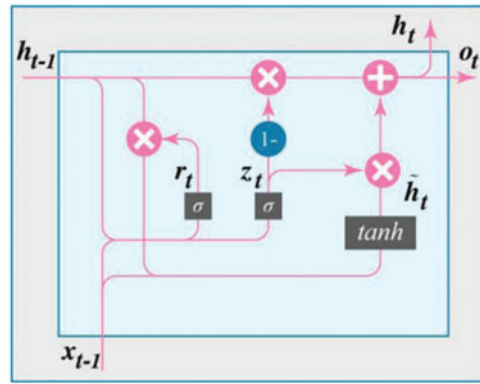


**Figure 12:** The structure of a GRU unit [80]

The update gate (Eq. (6)) determines how much of the past information should be retained and combined with the current input at a specific time step. It is computed based on the concatenation of the previous hidden state $h_{t-1}$ and the current input $x_t$, followed by a linear transformation and a sigmoid activation function.

$$z_t = \sigma \ (W_z \ [h_{t-1}, \ x_t] + b_z) \tag{6}$$

The reset gate (Eq. (7)) decides how much of the past information should be forgotten. It is computed in a similar manner to the update gate using the concatenation of the previous hidden state

and the current input.

$$r_t = \sigma \ (W_r \ [h_{t-1}, \ x_t] + b_r) \tag{7}$$

The current memory content (Eq. (8)) is calculated based on the reset gate and the concatenation of the transformed previous hidden state and the current input. The result is passed through a hyperbolic tangent activation function to produce the candidate activation.

$$\tilde{h}_t = tanh \ (W_h \ [r_t h_{t-1}, \ x_t]) \tag{8}$$

Finally, the final memory state $h_t$ is determined by a combination of the previous hidden state and the candidate activation (Eq. (9)). The update gate determines the balance between the previous hidden state and the candidate activation. Additionally, an output gate $o_t$ can be introduced to control the information flow from the current memory content to the output (Eq. (10)). The output gate is computed using the current memory state $h_t$ and is typically followed by an activation function, such as the sigmoid function.

$$h_t = (1 - z_t) \ h_{t-1} + z_t \tilde{h}_t \tag{9}$$

$$o_t = \sigma_o \ (W_o h_t + b_o) \tag{10}$$

where the weight matrix of the output layer is $W_o$ and the bias vector of the output layer is $b_o$.

GRU offers a simpler alternative to LSTM with fewer tensor operations, allowing for faster training. However, the choice between GRU and LSTM depends on the specific use case and problem at hand. Both architectures have their advantages and disadvantages, and their performance may vary depending on the nature of the task [59].

### 3.3.4 Bidirectional GRU

The Bidirectional Gated Recurrent Unit (Bi-GRU) [81] improves upon the conventional GRU architecture through the integration of contexts from the past and future in sequential modeling tasks. In contrast to the conventional GRU, which exclusively processes input sequences forward, the Bi-GRU manages sequences in both forward and backward directions. In order to do this, two parallel GRU layers are used, one of which processes the input data forward and the other in reverse [82]. Fig. 13 shows the Bi-GRU's structural layout.
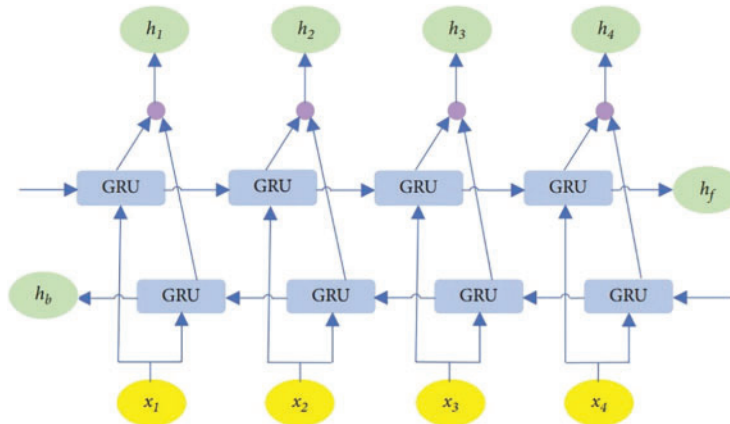


**Figure 13:** The structure of a Bi-GRU model [83]

### 3.4 Temporal Convolutional Networks (TCN)

Temporal Convolutional Networks (TCN) represent a significant advancement in neural network architectures, specifically tailored for handling sequential data, particularly time series. Originating as an extension of the one-dimensional Convolutional Neural Network (CNN), TCN was first introduced by [84] in 2017 for the task of action segmentation in video data, and its application was further generalized to other types of sequential data by [85] in 2018. TCN retains the powerful feature extraction capabilities inherent to CNN while being highly efficient in processing and analyzing time series data.

The purpose of training a TCN is to forecast the next $l$ values of the input time series. Assume that we have a sequence of inputs $x_0$, $x_1$, ..., $x_l$. We would like to predict, at each time step, some corresponding output $y_0$, $y_1$, ..., $y_l$, whose values are equal to the inputs shifted forward $l$ time steps. The primary limitation is that it can only use the inputs that have already been observed: $x_0$, $x_1$, ..., $x_t$, when forecasting the output $y_t$ for a given time step $t$ [86]. TCN is characterized by two fundamental properties: (1) The convolutions within the network are causal, ensuring that the output at any given time step depends solely on the current and past inputs, without any influence from future inputs. (2) Similar to Recurrent Neural Networks (RNNs), TCN can process sequences of arbitrary length and produce output sequences of identical length. The three primary components of a typical TCN are residual connections, dilated convolution, and causal convolution [85,87,88]. Fig. 14 illustrates the schematic architecture of a TCN model.
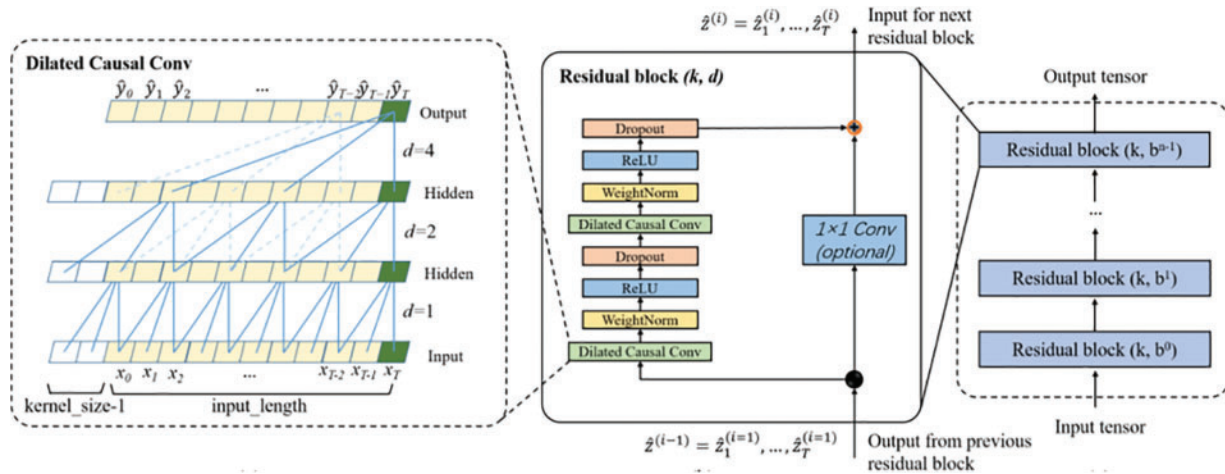


**Figure 14:** Schematic diagram of the TCN model architecture [89]

**Causal Convolution:**

The TCN architecture is built upon two foundational principles. To adhere to the first principle, the initial layer of a TCN is a one-dimensional fully convolutional network, wherein each hidden layer maintains the same length as the input layer, achieved through zero-padding. This padding ensures that each successive layer remains the same length as the preceding one. To satisfy the second principle, TCN employs causal convolutions. A causal convolution is a specialized one-dimensional convolutional network where only elements from time $t$ and earlier are convolved to produce the output at time $t$. Fig. 15 demonstrates the structure of a causal convolutional network.
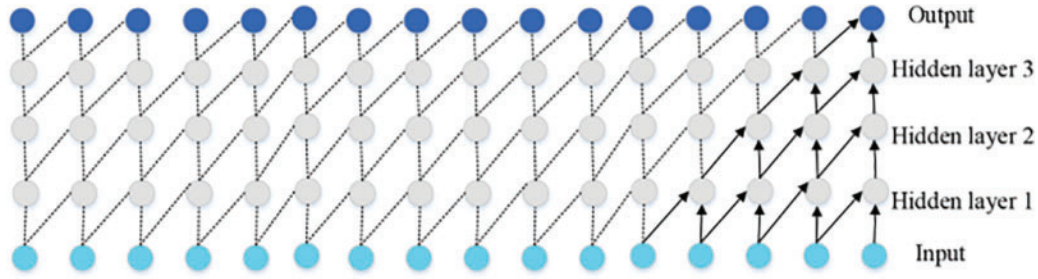
**Figure 15:** The structure of the causal convolutional network [87]

**Dilated Convolution:**

TCN aims to effectively capture long-range dependencies in sequential data. A simple causal convolution can only consider a history that scales linearly with the depth of the network. This limitation would necessitate the use of large filters or an exceptionally deep network structure, which could hinder performance, particularly for tasks requiring a longer history.

The depth of the network could lead to issues such as vanishing gradients, ultimately degrading network performance or causing it to plateau. To address these challenges, TCN employs dilated convolutions [90], which exponentially expand the receptive field, allowing the network to process large time series efficiently without a proportional increase in computational complexity. The architecture of a dilated convolutional network is depicted in Fig. 16.
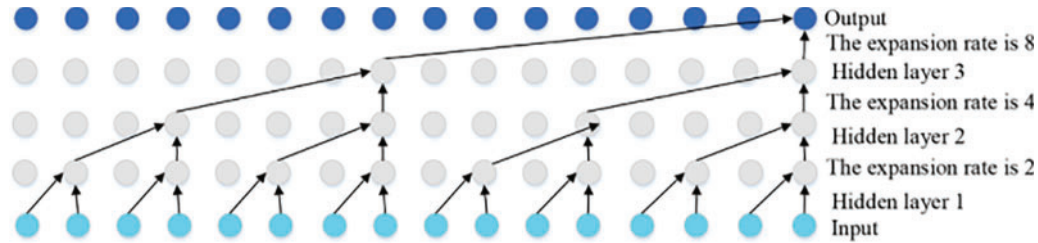


**Figure 16:** Dilated convolutional structure [87]

By inserting gaps between the weights of the convolutional kernel, dilated convolutions effectively increase the network's receptive field while maintaining computational efficiency. The mathematical formulation of a dilated convolution is given by Eq. (11).

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{11}$$

where $d$ is the dilation rate, $k$ is the size of the filter, and $s - d \cdot i$ accounts for the direction of the past. Dilation is the same as adding a fixed step in between each pair of neighboring filter taps. When $d = 1$, dilated convolution becomes a regular convolution. As $d$ increases, the output at the higher layers reflects a broader range of inputs, improving performance on long-range dependencies in time series.

**Residual Connections:**

To construct a more expressive TCN model, it is essential to use small filter sizes and stack multiple layers. However, stacking dilated and causal convolutional layers increases the depth of the

network, potentially leading to problems such as gradient decay or vanishing gradients during training. To mitigate these issues, TCN incorporates residual connections into the output layer. Residual connections facilitate the flow of data across layers by adding a shortcut path, allowing the network to learn residual functions, which are modifications to the identity mapping, rather than learning a full transformation. This approach has been shown to be highly effective in very deep networks.

A residual block [46] has a branch that lead to a set of transformations F, whose outputs are appended to block's input x, as shown in Eq. (12).

$$o = Activation \ (x + F(x)) \tag{12}$$

This method enables the network to focus on learning residual functions rather than the entire mapping. The TCN residual block typically consists of two layers of dilated causal convolutions followed by a non-linear activation function, such as Rectified Linear Unit (ReLU). The convolutional filters within the TCN are normalized using weight normalization [91], and dropout [92] is applied to each dilated convolution layer for regularization, where an entire channel is zeroed out at each training step. In contrast to a conventional ResNet, where the input is directly added to the output of the residual function, TCN adjusts for differing input-output widths by performing an additional $1 \times 1$ convolution to ensure that the element-wise addition $\oplus$ operates on tensors of matching dimensions.

### 3.5 Kolmogorov-Arnold Network (KAN)

Kolmogorov-Arnold Networks (KANs) represent a promising alternative to traditional Multi-Layer Perceptrons (MLPs) by leveraging the Kolmogorov-Arnold theorem, a sophisticated mathematical framework that enhances the capacity of neural networks to process complex data structures. KANs were first introduced in 2024 by [93], with the goal of incorporating advanced mathematical theories into deep learning architectures to improve their performance on intricate tasks. While MLPs are inspired by the universal approximation theorem, KANs are motivated by the Kolmogorov-Arnold representation theorem [94], which states that any multivariate continuous function $f$ over a bounded domain can be expressed as a finite composition of simpler one-dimensional continuous functions:

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left( \sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{13}$$

where $\phi_{q,p}$ is a mapping $[0, 1] \to \mathbb{R}$ and $\Phi_q$ is a mapping $\mathbb{R} \to \mathbb{R}$.

KAN maintain a fully connected structure like MLP, but with a key distinction: while MLP assign fixed activation functions to nodes (neurons), KAN assign learnable activation functions to edges (weights). Consequently, KAN do not employ traditional linear weight matrices; instead, each weight parameter is replaced by a learnable one-dimensional function parameterized as a spline. Unlike MLP, which apply non-linear activation functions at each node, KAN nodes only sum the incoming data, relying on the rich, learnable spline functions to introduce non-linearity. Although this approach might initially seem computationally expensive, KAN often result in significantly smaller computation graphs compared to MLP. Fig. 17 illustrates the structure of a KAN.
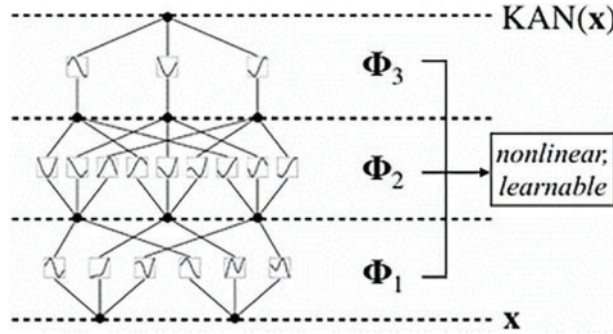
**Figure 17:** The structure of Kolmogorov-Arnold Network (KAN) [93]

The Kolmogorov-Arnold Network (KAN) can be expressed specifically as follows:

$$KAN(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_1 \circ \Phi_1)(x) \tag{14}$$

The transformation of each layer, $\Phi_l$, operates on the input $x_l$ to generate $x_{l+1}$, the input for the following layer, as follows:

$$x_{l+1} = \Phi_l(x_l) = \begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \ldots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \ldots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \ldots & \phi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix} x_l \tag{15}$$

where each activation function $\varnothing_{l,j,i}$ is a spline, offering a rich, flexible response surface to inputs from the model:

$$Spline(x) = \sum_i c_i B_i(x), c_i \text{ are trainable coefficients} \tag{16}$$

Several variants of KANs have emerged to tackle specific challenges in various applications:

➢ **Convolutional KAN (CKAN) [95]:** CKAN is a pioneering alternative to standard CNN, which have significantly advanced the field of computer vision. Convolutional KAN integrate the non-linear activation functions of KAN into the convolutional layers, leading to a substantial reduction in the number of parameters and offering a novel approach to optimizing neural network architectures.

➢ **Temporal KAN (TKAN) [96]:** Temporal Kolmogorov-Arnold Networks combines the principles of KAN and Long Short-Term Memory (LSTM) networks to create an advanced architecture for time series analysis. Comprising layers of Recurrent Kolmogorov-Arnold Networks (RKANs) with embedded memory management, TKAN excels in multi-step time series forecasting. The TKAN architecture offers tremendous promise for improvement in domains needing one-step-ahead forecasting by solving the shortcomings of existing models in handling complicated sequential patterns [97,98].

➢ **Multivariate Time Series KAN (MT-KAN) [99]:** MT-KAN is specifically designed to handle multivariate time series data. The primary objective of MT-KAN is to enhance forecasting accuracy by modeling the intricate interactions between multiple variables. MT-KAN utilizes spline-parametrized univariate functions to capture temporal relationships while incorporating methods to model cross-variable interactions.

➢ **Fractional KAN (fKAN) [100]:** fKAN is an enhancement of the KAN architecture that integrates the unique properties of fractional-orthogonal Jacobi functions into the network's basis functions. This method guarantees effective learning and improved accuracy by utilizing the special mathematical characteristics of fractional Jacobi functions, such as straightforward derivative equations, non-polynomial behavior, and activity for positive and negative input values.

➢ **Wavelet KAN (Wav-KAN) [101]:** The purpose of this innovative neural network design is to improve interpretability and performance by incorporating wavelet functions into the Kolmogorov-Arnold Networks (KAN) framework. Wav-KAN is an excellent way to capture complicated data patterns by utilizing wavelets' multiresolution analysis capabilities. It offers a reliable solution to the drawbacks of both recently suggested KANs and classic Multi-Layer Perceptrons (MLPs).

➢ **Graph KAN [102]:** This innovative model applies KAN principles to graph-structured data, replacing the MLP and activation functions typically used in Graph Neural Networks (GNNs) with KAN. This substitution enables more effective feature extraction from graph-like data structures.

## 4 Generative (Unsupervised) Deep Learning Models

Supervised machine learning is widely used in artificial intelligence (AI), while unsupervised learning remains an active area of research with numerous unresolved questions. However, recent advancements in deep learning and generative modeling have injected new possibilities into unsupervised learning. A rapidly evolving domain within computer vision research is generative models (GMs). These models leverage training data originating from an unknown data-generating distribution to produce novel samples that adhere to the same distribution. The ultimate goal of generative models is to generate data samples that closely resemble real data distribution [103].

Various generative models have been developed and applied in different contexts, such as Auto-Encoder [104], Generative Adversarial Network (GAN) [105], Restricted Boltzmann Machine (RBM) [106], and Deep Belief Network (DBN) [107].

### 4.1 Autoencoder

The concept of an autoencoder originated as a neural network designed to reconstruct its input data. Its fundamental objective is to learn a meaningful representation of the data in an unsupervised manner, which can have various applications, including clustering [104].

An autoencoder is a neural network that aims to replicate its input at its output. It consists of an internal hidden layer that defines a code representing the input data. The autoencoder network is comprised of two main components: an encoder function, denoted as $z = f(x)$, and a decoder function that generates a reconstruction, denoted as $r = g(z)$ [108]. The function $f(x)$ transforms a data point $x$ from the data space to the feature space, while the function $g(z)$ transforms $z$ from the feature space back to the data space to reconstruct the original data point $x$. In modern autoencoders, these functions $z = f(x)$ and $r = g(z)$ are considered as stochastic functions, represented as $p_{encoder}(z|x)$ and $p_{dencoder}(r|z)$, respectively, where $r$ denotes the reconstruction of $x$ [109]. Fig. 18 illustrates an autoencoder model.

Autoencoder models find utility in various unsupervised learning tasks, such as generative modeling [110], dimensionality reduction [111], feature extraction [112], anomaly or outlier detection [113], and denoising [114].
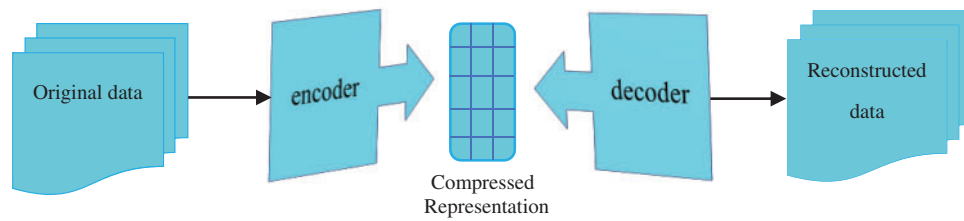
**Figure 18:** The structure of autoencoders

In general, autoencoder models can be categorized into two major groups: Regularized Autoencoders, which are valuable for learning representations for subsequent classification tasks, and Variational Autoencoders [115], which can function as generative models. Examples of regularized autoencoder models include Sparse Autoencoder (SAE) [116], Contractive Autoencoder (CAE) [117], and Denoising Autoencoder (DAE) [118].

Variational Autoencoder (VAE) is a generative model that employs probabilistic distributions, such as the mean and variance of a Gaussian distribution, for data generation [104]. VAE provide a principled framework for learning deep latent-variable models and their associated inference models. The VAE consists of two coupled but independently parameterized models: the encoder or recognition model and the decoder or generative model. During "expectation maximization" learning iterations, the generative model receives an approximate posterior estimation of its latent random variables from the recognition model, which it uses to update its parameters. Conversely, the generative model acts as a scaffold for the recognition model, enabling it to learn meaningful representations of the data, such as potential class labels. In terms of Bayes' rule, the recognition model is roughly the inverse of the generative model [119].

### 4.2 Generative Adversarial Network (GAN)

A notable neural network architecture for generative modeling, capable of producing realistic and novel samples on demand, is the Generative Adversarial Network (GAN), initially proposed by Goodfellow in 2014 [105]. A GAN consists of two key components: a generative model and a discriminative model. The generative model aims to generate data that resemble real ones, while the discriminative model aims to differentiate between real and synthetic data. Both models are typically implemented using multilayer perceptrons [120]. Fig. 19 depicts the framework of a GAN, where a two-player adversarial game is played between a generator (G) and a discriminator (D). The generator's updating gradients are determined by the discriminator through an adaptive objective [121].

As previously mentioned, GANs operate based on principles derived from neural networks, utilizing a training set as input to generate new data that resembles the training set. In the case of GANs trained on image data, they can generate new images exhibiting human-like characteristics.

The following outlines the step-by-step operation of a GAN [122]:

1. The generator, created by a discriminative network, generates content based on the real data distribution.
2. The system undergoes training to increase the discriminator's ability to distinguish between synthesized and real candidates, allowing the generator to better fool the discriminator.
3. The discriminator initially trains using a dataset as the training data.
4. Training sample datasets are repeatedly presented until the desired accuracy is achieved.

5. The generator is trained to process random input and generate candidates that deceive the discriminator.
6. Backpropagation is employed to update both the discriminator and the generator, with the former improving its ability to identify real images and the latter becoming more adept at producing realistic synthetic images.
7. Convolutional Neural Networks (CNNs) are commonly used as discriminators, while deconvolutional neural networks are utilized as generative networks.

Generative Adversarial Networks (GANs) have introduced numerous applications across various domains, including image blending [123], 3D object generation [124], face aging [125], medicine [126,127], steganography [128], image manipulation [129], text transfer [130], language and speech synthesis [131], traffic control [132], and video generation [133].
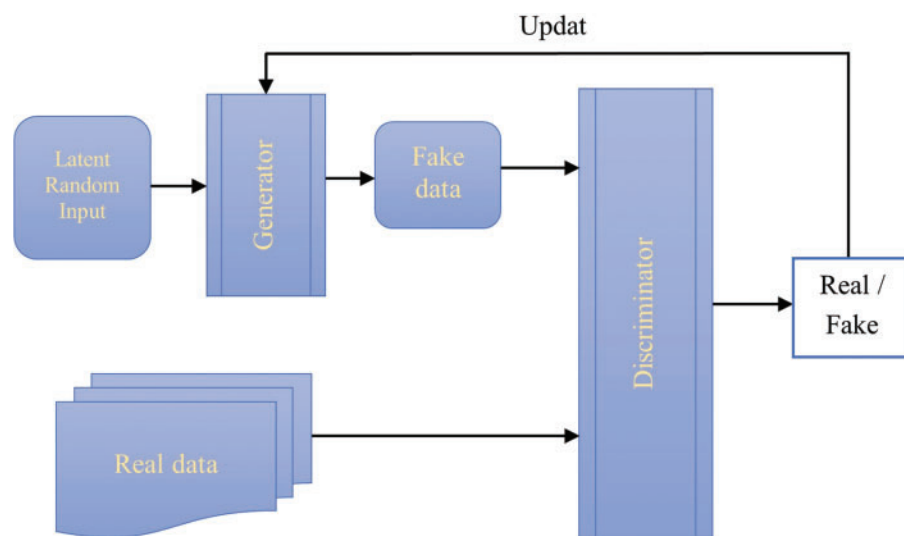


**Figure 19:** The framework of a GAN

Furthermore, several models have been developed based on the Generative Adversarial Network (GAN) framework to address specific tasks. These models include Laplacian GAN (Lap-GAN) [134], Coupled GAN (Co-GAN) [120], Markovian GAN [135], Unrolled GAN [136], Wasserstein GAN (WGAN) [137], and Boundary Equilibrium GAN (BEGAN) [138], CycleGAN [139], DiscoGAN [140], Relativistic GAN [141], StyleGAN [142], Evolutionary GAN (E-GAN) [121], Bayesian Conditional GAN [143], Graph Embedding GAN (GE-GAN) [132].

### 4.3 Deep Belief Network (DBN)

The Deep Belief Network (DBN) is a type of deep generative model utilized primarily in unsupervised learning to uncover patterns within large datasets. Consisting of multiple layers of hidden units, DBNs are adept at identifying intricate patterns and extracting features from data. Unlike discriminative models, DBNs exhibit a higher resistance to overfitting, making them well-suited for feature extraction from unlabeled data [144].

The stack of Restricted Boltzmann Machines (RBMs), which operate in an unsupervised learning framework, is a fundamental part of DBN. Every RBM in a DBN is made up of a hidden layer that contains latent representations and a visible layer that represents observable data features [145]. RBMs

are trained layer by layer: first, each RBM is trained independently, and then all of the RBMs are fine-tuned together as a whole within the DBN.

During the forward pass, the activations represent the probability of an output given a weighted input. In the backward pass, the activations estimate the probability of inputs given the weighted outputs. Through iterative training of RBMs within a DBN, these processes converge to form joint probability distributions of activations and inputs, allowing the network to effectively capture the underlying data structure [146,147]. Fig. 20 illustrates the schematic structure of a Deep Belief Network (DBN).
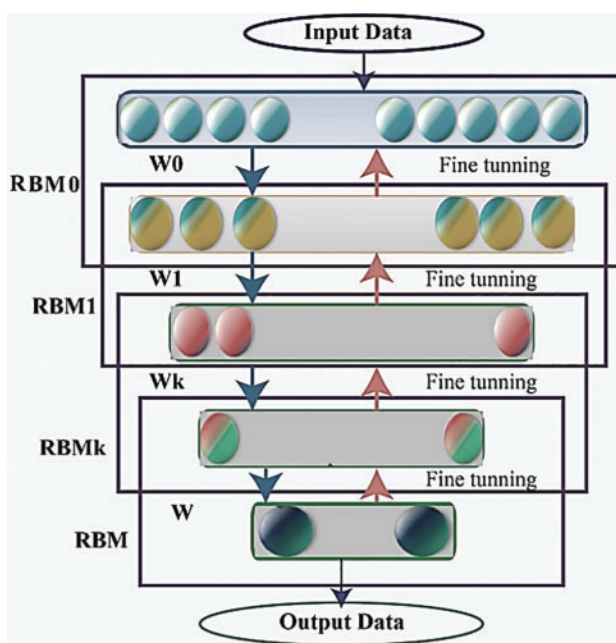


**Figure 20:** Structure of a DBN model [145]

## 5 Transformer Architecture

The Transformer architecture was originally introduced by Vaswani et al. [148] in 2017 for machine translation and has since become a foundational model in deep learning, especially for natural language processing (NLP). The Transformer functions as a self-attention encoder-decoder structure. The encoder consists of a stack of identical layers, and each layer consists of two sublayers. A multi-head self-attention mechanism is the first layer, while the other layer is a position-wise fully connected feed-forward network. Also, A normalizing layer [149] and residual connections [46] connect the multi-headed self-attention module's inputs and output. After that, a decoder uses the representation that the encoder produced to create an output sequence. A stack of identical layers makes up the decoder as well. The decoder adds a third sub-layer to each encoder layer in addition to the primary two, and this sub-layer handles multi-head attention over the encoder stack's output. Like the encoder, residual connections and a normalizing layer are used surrounding each of the sub-layers. The encoder and decoder's overall Transformer design is depicted in Fig. 21, left and right halves, respectively [150,151].
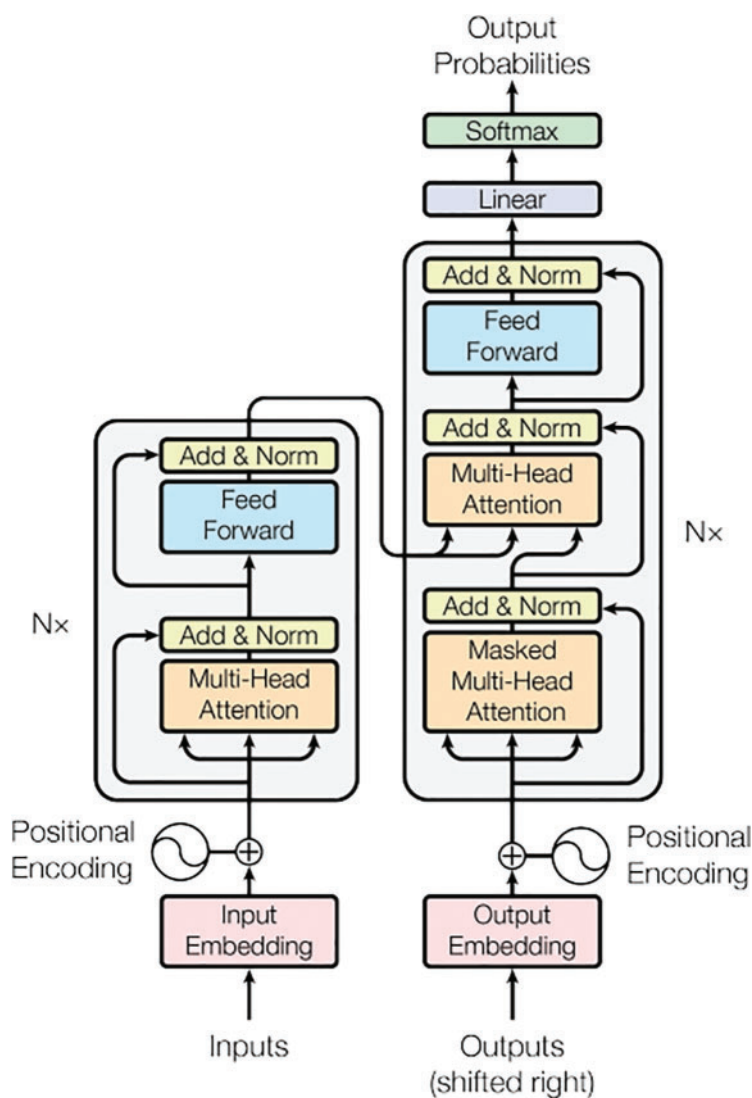
**Figure 21:** The architecture of the Transformer model [148]

Traditional RNN-based Seq2Seq models could be replaced with attention layers. Using various projection matrices, the query, key, and value vectors in the self-attention layer are all produced from the same sequence [152]. RNN training takes a very long period because it is sequential and iterative. Transformer training, on the other hand, is parallel and enables all features to be learned concurrently, significantly improving computational efficiency and cutting down on the amount of time needed for model training [153].

**Multi-Head Attention:** In the Transformer model, a multi-headed self-attention mechanism is employed to enhance the model's ability to capture dependencies between elements in a sequence. The core principle of the attention mechanism is that every token in the sequence can aggregate information from other tokens, allowing the model to understand contextual relationships more effectively. This is achieved by mapping a query, a set of key-value pairs, and an output (each represented as vectors) to

form an attention function. The output is computed as a weighted sum of the values, where the weights are determined by the compatibility function between the query and its corresponding key [148].

Multi-head attention is equivalent to the blended of $n$ distinct scaled dot-product attention (self-attention). It can effectively process the three vectors $Q$, $K$, and $V$, in parallel to obtain the final result by combining and calculating. The formula is visible in Eq. (17).

$$\begin{cases} MultiHead\ (Q,\ K,\ V) = Concat\ (head_1, \ldots,\ head_2)\ W^O \\ where\ head_i = Attention\ (QW_i^Q,\ KW_i^K,\ VW_i^V) \end{cases} \tag{17}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_V}$, and $W^O \in \mathbb{R}^{hd_V \times d_{model}}$.

The main component of the Transformer, scaled dot-product attention (self-attention), uses the weight of each sensor event in the input vector, which is represented by

$$Attention\ (Q,\ K,\ V) = softmax\ \left( \frac{QK^T}{\sqrt{d_k}} \right) V \tag{18}$$

The initial step in scaled dot-product attention is to convert the input data into an embedding vector and the three vectors of query vector $(Q)$, key vector $(K)$, and value vector $(V)$ are then extracted from the embedding vectors. Next, a score is determined for every vector: score is equal to $Q \cdot K$. Score normalization (dividing by $\sqrt{d_k}$) is used for gradient stability. Next, the score is processed using the softmax activation function. The weighted score $v$ for every input vector is obtained by taking the softmax dot product value $v$. The final result is produced after summing. Scaled dot-product attention and multi-head attention are displayed in Fig. 22 [154].
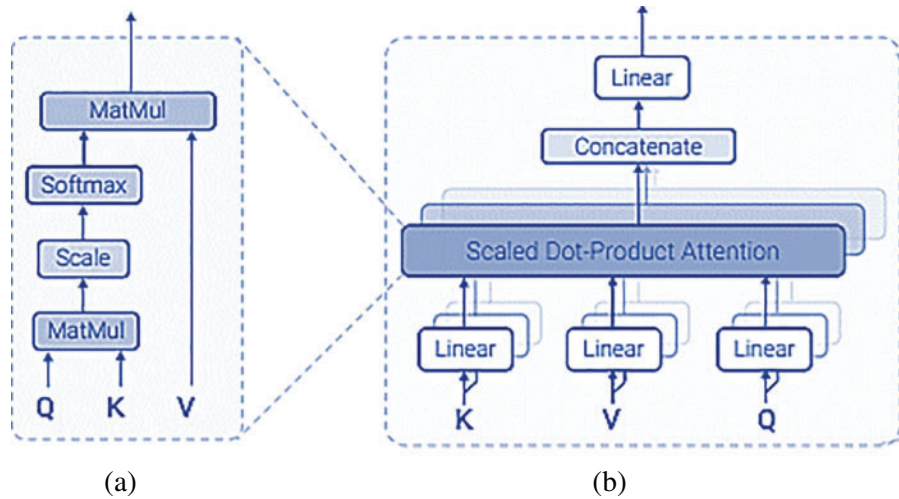


**Figure 22:** (a) Scaled dot-product attention, (b) multi-head attention

**Position-Wise Feed-Forward Networks:** Each encoder and decoder layer have a fully connected feed-forward network in addition to attention sub-layers. This feed-forward network is applied to each position independently and in the same way. This is made up of two linear transformations connected by a ReLU activation.

$$FFN\ (x) = max\ (0,\ xW_1 + b_1)\ W_2 + b_2 \tag{19}$$

**Positional Encoding:** Since the Transformer model does not rely on recurrence or convolution, it requires a way to capture the relative or absolute positions of tokens within a sequence to effectively utilize the sequence's order. To address this, positional encoding is introduced at the input level of both the encoder and decoder stacks. These positional encodings are added to the input embeddings, as they share the same dimensionality, $d_{model}$. This combination enables the model to incorporate positional information, allowing it to better understand the sequential nature of the data [148].

Positional encodings in Transformer architecture were achieved by using sine and cosine functions of various frequencies:

$$\begin{cases} PE_{(pos,\,2i)} = sin \,(pos/10000^{2i/d_{model}}) \\ PE_{(pos,\,2i+1)} = cos \,(pos/10000^{2i/d_{model}}) \end{cases} \tag{20}$$

where $pos$ is the position and $i$ is the dimension. Every dimension of the positional encoding has a sinusoidal relationship. The wavelengths range from $2\pi$ $to$ $10000 \cdot 2\pi$ in a geometric development. This function was selected because it would make it simple for the model to learn how to attend to relative positions, since for any fixed offset $k$, $PE_{pos+k}$ can be expressed as a linear function of $PE_{pos}$.

### *Transformer Variants*

The Transformer architecture has proven to be highly versatile, with numerous variants developed to address specific challenges across different domains. Typically, Transformers are pre-trained on large datasets using unsupervised methods to learn general representations, which are then fine-tuned on specific tasks using supervised learning. This hybrid approach leverages the strengths of both learning paradigms. Some notable Transformer variants include:

➢ **Bidirectional Encoder Representations from Transformers (BERT) [**155**]:** A multi-layer bidirectional Transformer encoder for unsupervised pre-training in natural language understanding (NLU) tasks.

➢ **Generative Pre-Training Transformer (GPT) [**156,157**]:** A type of Transformer model developed by OpenAI that excels in natural language processing (NLP) tasks through unsupervised pre-training followed by supervised fine-tuning.

➢ **Transformer-XL [**158**]:** It is proposed for language modeling to permit learning reliance beyond a set length without compromising temporal coherence. Transformer-XL (Transformer-Extra Long) comprises a unique relative positional encoding method and a segment-level recurrence mechanism. This approach not only makes it possible to record longer-term dependencies, but also fixes the issue of context fragmentation.

➢ **XLNet [**159**]:** It is a generalized autoregressive (AR) pretraining technique that combines the benefits of autoencoding (AE) and autoregressive (AR) techniques with a permutation language modeling aim. XLNet's neural architecture, which integrates Transformer-XL and the two-stream attention mechanism, is built to function effortlessly with the autoregressive (AR) objective.

➢ **Fast Transformer [**160**]:** It introduces multi-query attention as an alternative to multi-head attention. This approach reduces memory bandwidth requirements, leading to increased processing speed.

➢ **Multimodal Transformer (MulT) [**161**]:** It is designed for analyzing human multimodal language. At the heart of MulT is the crossmodal attention mechanism, which provides a latent crossmodal adaptation that fuses multimodal information by directly attending to low-level features in other modalities.

➢ **Vision Transformer (ViT) [**162**]:** An innovative approach based on Transformer structure for visual tasks like image classification.

➢ **Pyramid Vision Transformer (PVT) [**163**]:** An Transformer framework for complex prediction tasks like semantic segmentation and object recognition.

➢ **Swin Transformer [**164**]:** A hierarchical Transformer that uses shifted windows to construct its representation. A wide variety of vision tasks, including semantic segmentation, object detection, and image classification, may be performed with Swin Transformer.

➢ **Tokens-to-Token Vision Transformer (T2T-ViT) [**165**]:** A vision Transformer that can be trained from scratch on ImageNet. T2T-ViT overcomes ViT's drawbacks by accurately modeling the structural information of images and enhancing feature richness.

➢ **Transformer in Transformer (TNT) [**166**]:** A vision Transformer for visual recognition. Both local and global representations are extracted by the TNT architecture through the use of an inner Transformer and an outer Transformer.

➢ **PyramidTNT [**167**]:** A improved TNT model which used pyramid architecture, and convolutional stem in order to greatly enhance the original TNT model.

➢ **Switch Transformers [**168**]:** It is suggested as a straightforward and computationally effective method of increasing a Transformer model's parameter count.

➢ **ConvNeXt [**169**]:** A redesigned Transformer architecture that makes use of the Transformer attention mechanism and incorporates convolutional layers into the encoder and decoder modules to extract spatially localized data.

➢ **Evolutionary Algorithm Transformer (EATFormer) [**170**]:** An improved vision Transformer influenced by an evolutionary algorithm.

## 6  Deep Reinforcement Learning

Reinforcement learning (RL) is a machine learning approach that deals with sequential decision-making, aiming to map situations to actions in a way that maximizes the associated reward. Unlike supervised learning, where explicit instructions are given after each system action, in the RL framework, the learner, known as an agent, is not provided with explicit guidance on which actions to take at each timestep $t$. The RL agent must explore through trial and error to determine which actions yield the highest rewards [171]. Furthermore, unlike supervised learning, where the correct output is obtained and the model is updated based on the loss or error, RL uses gradients without a differentiable loss function to teach a model to explore randomly and learn to make optimal decisions [172]. Fig. 23 depicts the agent-environment interaction in reinforcement learning (RL). The standard theoretical framework for RL is based on a Markov Decision Process (MDP), which extends the concept of a Markov process and is used to model decision-making based on states, actions, and rewards [173].

Deep reinforcement learning combines the decision-making capabilities of reinforcement learning with the perception function of deep learning. It is considered a form of "real AI" as it aligns more closely with human thinking. Fig. 24 illustrates the basic structure of deep reinforcement learning, where deep learning processes sensory inputs from the environment and provides the current state data. The reinforcement learning process then links the current state to the appropriate action and evaluates values based on anticipated rewards [174,175].

One of the most renowned deep reinforcement learning models is the Deep Q-learning Network (DQN) [176], which directly learns policies from high-dimensional inputs using Convolutional Neural Network (CNN). Other common models in deep reinforcement learning include Double DQN [177], Dueling DQN [178], and Monte Carlo Tree Search (MCTS) [179].
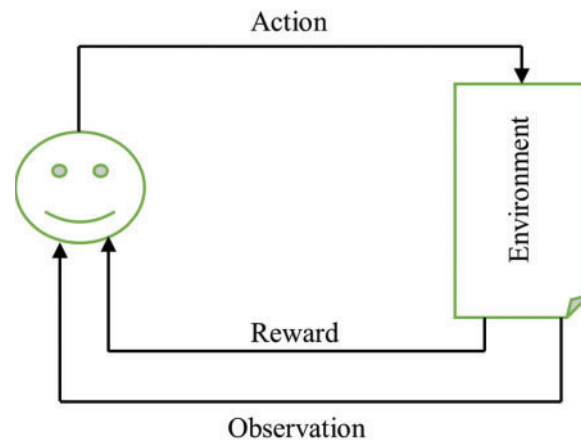
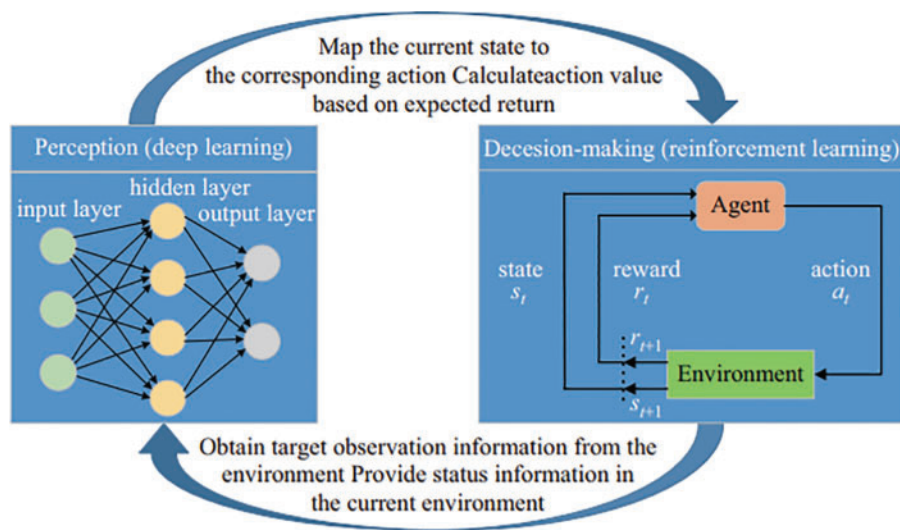**Figure 23:** Agent-environment interaction in RL



**Figure 24:** Basic structure of Deep Reinforcement Learning (DRL) [174]

Deep Reinforcement Learning (DRL) models find applications in various domains, such as video game playing [180,181], robotic manipulation [182,183], image segmentation [184,185], video analysis [186,187], energy management [188,189], and more.

## 7  Deep Transfer Learning

Deep neural networks have significantly improved performance across various machine learning tasks and applications. However, achieving these remarkable performance gains often requires large amounts of labeled data for supervised learning, as it relies on capturing the latent patterns within the data [190]. Unfortunately, in certain specialized domains, the availability of sufficient training data is a major challenge. Constructing a large-scale, high-quality annotated dataset is costly and time-consuming [191].

To address the issue of limited training data, transfer learning (TL) has emerged as a crucial tool in machine learning. The concept of transfer learning finds its roots in educational psychology, where the theory of generalization suggests that transferring knowledge from one context to another is facilitated by generalizing experiences. To achieve successful transfer, there needs to be a connection between the two learning tasks. For example, someone who has learned to play the violin is likely to learn the piano more quickly due to the shared characteristics between musical instruments [192]. Fig. 25 depicts the learning process of transfer learning. Deep transfer learning (DTL) makes use of the learning experience to reduce the time and effort needed to train large networks as well as the time and effort needed to create the weights for an entire network from scratch [193].
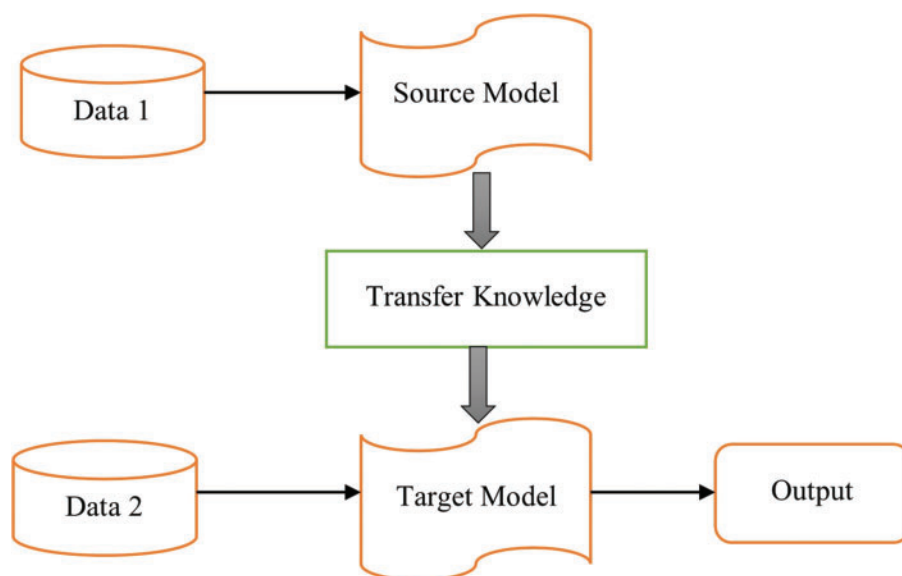


**Figure 25:** Learning process of transfer learning

With the growing popularity of deep neural networks in various fields, numerous deep transfer learning techniques have been proposed. Deep transfer learning can be categorized into four main types based on the techniques employed [191]: instances-based deep transfer learning, mapping-based (feature-based) deep transfer learning, network-based (model-based) deep transfer learning, and adversarial-based deep transfer learning.

Instances-based deep transfer learning involves selecting a subset of instances from the source domain and assigning appropriate weight values to these selected instances to supplement the training set in the target domain. Algorithms such as TaskTrAdaBoost [194] and TrAdaBoost.R2 [195] are well-known approaches based on this strategy.

Mapping-based deep transfer learning focuses on mapping instances from both the source and target domains into a new data space, where instances from the two domains exhibit similarity and are suitable for training a unified deep neural network. Successful methods based on this approach include Extend MMD (Maximum Mean Discrepancy) [196], and MK-MMD (Multiple Kernel variant of MMD) [197].

Network-based (model-based) deep transfer learning involves reusing a segment of a pre-trained network from the source domain, including its architecture and connection parameters, and applying it to a deep neural network in the target domain. These model-based approaches are highly effective

for domain adaptation between source and target data by adjusting the network (model), making them the most widely adopted strategies in deep transfer learning (DTL). Remarkably, these methods can even adapt target data that is significantly different from the source data [198].

Network-based (model-based) approaches in deep transfer learning typically involve pre-training, freezing, fine-tuning, and adding new layers. Pre-trained models consist of layers from a deep learning network (DL model) that have been trained using source data. Two key methods for training a model with target data are freezing and fine-tuning. These methods involve using some or all layers of a pre-defined model. When layers are frozen, they retain fixed parameters/weights from the pre-trained model. In contrast, fine-tuning involves initializing parameters and weights with pre-trained values instead of starting with random values, either for the entire network or specific layers [198].

A recent advancement in model-based deep transfer learning is Progressive Neural Networks (PNNs). This strategy involves the freezing of a pre-trained model and integrating new layers specifically for training on target data [199]. The concept behind progressive learning is grounded in the idea that acquiring a new skill necessitates leveraging existing knowledge. This mirrors the way humans learn new abilities. For instance, a child learns to run by employing all the skills acquired during crawling and walking. PNN constructs a new model for each task it encounters. Each freshly generated model is interconnected with all others, aiming to learn a new task by applying the knowledge accumulated from preceding models.

Adversarial-based methods focus on gathering transferable features from both the source and target data by leveraging logical relationships or rules acquired in the source domain. Alternatively, they may utilize techniques inspired by generative adversarial networks (GANs) [200].

These deep transfer learning techniques have proven to be effective in overcoming the challenge of limited training data, enabling knowledge transfer across domains, and facilitating improved performance in various applications such as image classification [201,202], speech recognition [203,204], video analysis [205,206], signal processing [207,208], and other.

In transfer learning, several popular pre-trained deep learning models are frequently used, including Xception [52], MobileNet [53], DenseNet [55], EfficientNet [57], NasNet [209], and among others. These models are initially trained on large-scale datasets like ImageNet, and their learned weights are then transferred to a target domain. The architectures of these networks reflect a broader trend in deep learning design, transitioning from manually crafted by human experts to automatically optimized patterns. This evolution focuses on striking a balance between model accuracy and computational complexity [210].

## 8 Hybrid Deep Learning Models

Hybrid deep learning architectures, which integrate elements from various deep learning models, demonstrate significant potential in enhancing performance. By combining different fundamental generative or discriminative models, the following three categories of hybrid deep learning models can be particularly effective for addressing real-world problems:

- Combination of various supervised models to extract more relevant and robust features, such as CNN+LSTM or CNN+GRU. By leveraging the strengths of different architectures, these hybrid models effectively capture both spatial and temporal dependencies within the data.
- Integrating various types of generative models, such as combining Autoencoders (AE) with Generative Adversarial Networks (GANs), to harness their strengths and enhance performance across a range of tasks.

● Integrating the capabilities of generative models with supervised models to leverage the strengths of both approaches can significantly enhance performance on various tasks. This hybrid strategy improves feature learning, data augmentation, and model robustness. Examples of such combinations include DBN+MLP, GAN+CNN, AE+CNN, and so on.

## 9  Application of Deep Learning

In recent years, deep learning has demonstrated remarkable effectiveness across a wide range of applications, tackling various challenges in fields including healthcare, computer vision, speech recognition, natural language processing (NLP), e-learning, smart environments, and more. Fig. 26 highlights several potential real-world application areas of deep learning.
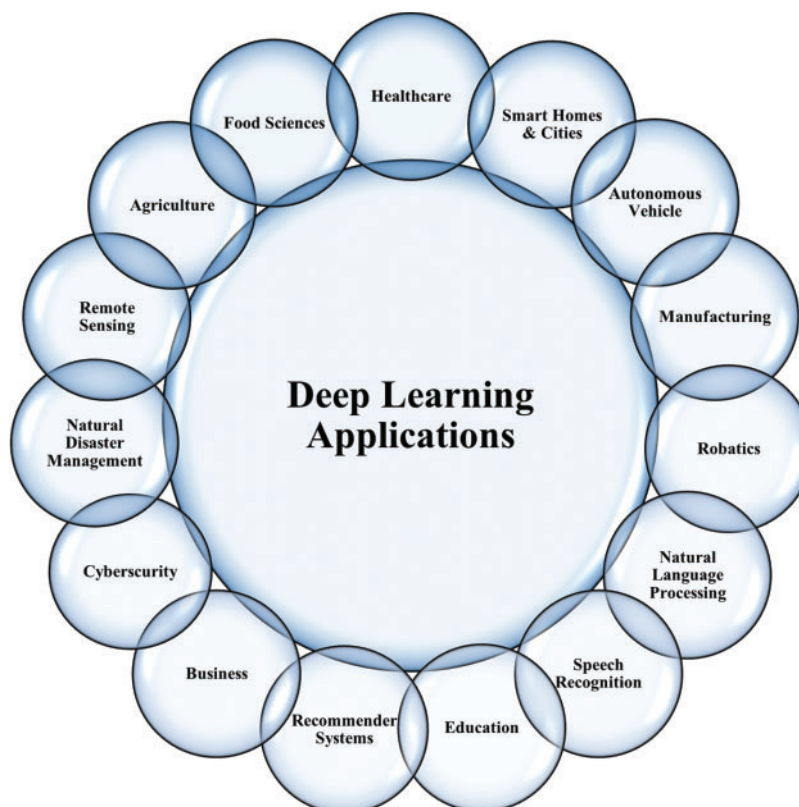


**Figure 26:** Numerous possible domains for deep learning applications in the real world

Five useful categories have been established for these applications: classification, detection, localization, segmentation, and regression [10]. A concept called classification divides a collection of facts into classes. Detection typically involves recognizing objects and their boundaries within images, videos, or other data types. Localization refers to the process of identifying and determining the position of specific objects or features within an image or other types of data. Segmentation involves dividing an image or dataset into distinct regions or segments, with each segment representing a particular object or feature of interest. Regression is used to model and analyze the relationships between a dependent variable and one or more independent variables. It predicts continuous outcomes based on input features.

However, each real-world application area has its own specific goals and requires particular tasks and deep learning techniques. Table 1 provides a summary of various deep learning tasks and methods applied across multiple real-world application domains.

**Table 1:** A summary of the practical applications of deep learning models in real-world domains

| Application setting | Tasks | Models | Reference |
|---|---|---|---|
| Smart homes & smart cities | Human activity recognition | CNN+LSTM | [211] |
| | Smart energy management | Reinforcement learning | [212] |
| | Traffic management | GRU based | [213] |
| | Waste management | CNN based | [214] |
| | Smart parking system | Stacked GRU+LSTM | [215] |
| Education | Student engagement detection | DenseNet self-attention | [216] |
| | Student affective states recognition | ConvNeXt+GRU | [82] |
| | Automatic attendance system | CNN+LSTM | [217] |
| | Automated exam control | CNN based (VGG) | [218] |
| Healthcare | Medical image analysis | Vision Transformer | [219] |
| | Early disease detection | InceptionV3 | [220] |
| | Remote patient monitoring | CNN based | [221] |
| | Analyze genomic data | Transfer learning based | [222] |
| natural language processing (NLP) | Question answering systems | BERT based | [223] |
| | Sentiment analysis | Transformer based | [224] |
| | Text summarization | Attentional LSTM | [225] |
| Speech recognition | Speech emotion recognition | LSTM+CNN | [226] |
| | Automatic speech translation | Deep transfer learning | [203] |
| Agriculture | Plant disease detection | ViT+CNN | [227] |
| | Precision agriculture | GRU+CNN | [228] |
| | Smart irrigation system | Autoencoders, GAN | [229] |
| | Soil quality prediction | CNN | [230] |
| Natural disaster management | Earthquake prediction | CNN+RNN | [231] |
| | Flood forecasting | Attention GRU | [232] |
| | Tsunami prediction | LSTM based | [233] |
| Remote sensing | Land cover classification | Extended ViT | [234] |
| | Investigation wildfire area | CNN based | [235] |
| | Deforestation detection | Transformer based | [236] |
| Cybersecurity | Intrusion detection | CNN+ Bi-LSTM | [237] |
| | Malware detection | LSTM based | [238] |
| | Phishing detection | LSTM+CNN | [239] |
| | Credit card fraud detection | Deep autoencoder | [240] |
| | Biometric authentication | CNN+LSTM | [241] |
| Recommender systems | Context-aware recommendation | RNN based | [242] |
| | Sequential recommendation | LSTM based | [243] |
| | Multimodal recommendation | CNN based | [244] |

(Continued)

**Table 1 (continued)**

| Application setting | Tasks | Models | Reference |
|---|---|---|---|
| Business | Purchase behavior prediction | RNN based | [245] |
| | Loan default prediction | CNN based | [246] |
| | Stock trend prediction | Bi-LSTM | [247] |
| Autonomous vehicles | Object detection | Swin Transformer +CNN | [248] |
| | Pedestrian detection | Deep CNN | [249] |
| | Localization and mapping | CNN-GRU | [250] |
| | Lane detection & path planning | CNN based | [251] |
| Manufacturing | Defect detection | Transformer based | [252] |
| | Predictive maintenance | LSTM, GRU, CNN | [253] |
| | Process optimization | Reinforcement learning | [254] |
| | Supply chain optimization | LSTM | [255] |
| Robotics | Robotic grasping | Reinforcement learning | [256] |
| | Tracking and motion planning | Reinforcement learning | [257] |
| | Human-robot interaction | RNN based | [258] |

## 10 Deep Learning Challenges

While deep learning models have achieved remarkable success across various domains, they also come with significant challenges. Below are some of the most critical challenges, followed by potential solutions to address them.

### 10.1 Insufficient Data

Deep learning models require large amounts of data to perform well. The performance of these models typically improves as the volume of data increases. However, in many cases, sufficient data may not be available, making it difficult to train deep learning models effectively [10].

Three possible approaches may be used to appropriately handle the insufficient data problem. The first method is Transfer Learning (TL), which is used to DL models by reusing pre-trained model pieces in new models. We thoroughly reviewed the transfer learning strategy in Section 7.

Data augmentation is the second method of gathering additional data. The goal of data augmentation is to improve the trained models' capacity for generalization. Generalization is necessary for networks to overcome small datasets or datasets with unequal class distributions, and it is especially crucial for real-world data [259]. There are several strategies for augmenting data, and each one is contingent upon the characteristics of the datasets [260]. A few of these techniques are geometric transformations [261], Mixup augmentation [262], Random oversampling [263], Feature space augmentation [264], generative data augmentation [265], and many more.

The third approach considers using simulated data to increase the training set's volume. If you have a good understanding of the physical process, you can sometimes build simulators from it. Consequently, the outcome will include simulating as much data as necessary [10,266].

### 10.2 Imbalanced Data

In real-world situations, particularly in those that deep learning models address, the issue of class imbalance is common. If the majority of instances in the data set belong to one class and the remaining

instances belong to the other class, then there is a class imbalance in a binary classification scenario. In multi-class, multi-label, multi-instance learning as well as in regression difficulties and other situations, class imbalances are present and are actually reinforced [267].

It has been determined that there are three main approaches to addressing imbalanced data: data-level techniques, algorithm-level techniques, and hybrid techniques. The focus of data-level techniques is to add or remove samples from training sets in order to balance the data distributions. These techniques balance the data distributions by adding new samples to the minority class (oversampling) or removing samples from the majority class (undersampling) [268,269]. A variety of oversampling techniques, including Synthetic Minority Over-sampling Technique (SMOTE) [270], Borderline-SMOTE [271], Adaptive Synthetic (ADASYN) [272], SVM (Support Vector Machine)-SMOTE [273], Majority Weighted Minority Oversampling Technique (MWMOTE) [274], Sampling With the Majority (SWIM) [275], Reverse-SMOTE (R-SMOTE) [276], Constrained Oversampling (CO) [277], SMOTE Based on Furthest Neighbor Algorithm (SOMTEFUNA) [278], and many more can be used to solve imbalanced data problems. Also, there are several techniques for undersampling, including EasyEnsemble [279], BalanceCascade [279], Inverse Random Undersampling [280], MLP-based Undersampling Technique (MLPUS) [281], and others.

Algorithm-level approaches modify existing learning algorithms to mitigate the bias towards the majority class. These techniques require specialized knowledge of both the application domain and the learning algorithm to diagnose why a classifier fails under imbalanced class distributions [268]. Two of the most commonly used methods in this context are Cost-Sensitive Learning [282,283] and One-Class Learning [284].

The third approach consists of hybrid methods, which combine algorithm-level techniques with data-level methods in the appropriate way. Hybridization is required to address issues with algorithm and data-level approaches and improve classification accuracy [285].

### 10.3 Overfitting

Overfitting occurs when a deep learning model learns the systematic and noise components of the training data to the point that it adversely affects the model's performance on new data. In fact, overfitting occurs as a result of noise, the small size of the training set, and the complexity of the classifiers. Overfitted models tend to memorize all the data, including the inevitable noise in the training set, rather than understanding the underlying patterns in the data [24]. Overfitting is addressed with methods including dropout [92], weight decay [286], batch normalization [287,288], regularization [289], data augmentation, and others, although determining the ideal balance is still difficult.

### 10.4 Vanishing and Exploding Gradient

In deep neural networks, the computation of gradients is propagated layer by layer, leading to a phenomenon known as the vanishing or exploding gradient problem. As gradients are backpropagated through the network, they can exponentially diminish or grow, respectively, causing significant issues in training. When gradients vanish, the weights of the network are adjusted so minimally that the model's learning process becomes exceedingly slow, potentially stalling altogether. Conversely, exploding gradients can cause weights to be updated excessively, leading to instability and divergence during training. This problem is particularly pronounced with non-linear activation functions such as sigmoid and tanh, which compress the output into a narrow range, further exacerbating the issue by limiting the gradient's magnitude. Consequently, the model struggles to learn effectively, especially in deep networks where gradients must pass through many layers [8].

To mitigate the vanishing and exploding gradient problem, several strategies have been developed. One effective approach is to use the Rectified Linear Unit (ReLU) activation function, which does not saturate and therefore helps to maintain the gradient flow throughout the network [290]. Proper weight initialization techniques, such as Xavier initialization [291] can also reduce the likelihood of gradient issues by ensuring that initial weights are set in a way that prevents gradients from becoming too small or too large [292]. Another solution is batch normalization, which normalizes the inputs of each layer to maintain a stable distribution of activations throughout training. By doing so, batch normalization helps to alleviate the vanishing gradient problem and can accelerate convergence by reducing internal covariate shifts. Overall, addressing the vanishing and exploding gradient problem is crucial for training deep neural networks effectively, enabling them to learn complex patterns without succumbing to instability or inefficiency [288].

### 10.5 Catastrophic Forgetting

Catastrophic forgetting is a critical challenge in the pursuit of artificial general intelligence within neural networks. It occurs when a model, after being trained on a new task, loses its ability to perform previously learned tasks. This phenomenon is particularly problematic in scenarios where a model is expected to learn sequentially across multiple tasks without forgetting earlier ones, such as in lifelong learning or continual learning applications. The root cause of catastrophic forgetting lies like neural networks, which update their weights based on new training data. When trained on a new task, the model adjusts its parameters to optimize performance on that task, often at the expense of previously acquired knowledge. As a result, the model may exhibit excellent performance on the most recent task but perform poorly on earlier ones, effectively "forgetting" them [293].

Several strategies have been proposed to address catastrophic forgetting. One such approach is Elastic Weight Consolidation (EWC) [294], which penalizes changes to the weights that are important for previous tasks, thereby preserving learned knowledge while allowing the model to adapt to new tasks. Incremental Moment Matching (IMM) [295] is another technique that merges models trained on different tasks into a single model, balancing the performance across all tasks. The iCaRL (incremental Classifier and Representation Learning) [296] method combines classification with representation learning, enabling the model to learn new classes without forgetting previously learned ones. Additionally, the Hard Attention to the Task (HAT) [293] approach employs task-specific masks that prevent interference between tasks, reducing the likelihood of forgetting.

### 10.6 Underspecifcation

Underspecification is an emerging challenge in the deployment of machine learning (ML) models, particularly deep learning (DL) models, in real-world applications. It refers to the phenomenon where an ML pipeline can produce a multitude of models that all perform well on the validation set but exhibit unpredictable behavior in deployment. This issue arises because the pipeline's design does not fully specify which model characteristics are critical for generalization in real-world scenarios. The underspecification problem is often linked to the high degrees of freedom inherent in ML pipelines. Factors such as random seed initialization, hyperparameter selection, and the stochastic nature of training can lead to the creation of models with similar validation performance but divergent behaviors in production. These differences can manifest as inconsistent predictions when the model is exposed to new data or deployed in environments different from the training conditions [297].

Addressing underspecification requires rigorous testing and validation beyond standard metrics. Stress tests, as proposed by D'Amour et al. [297], are designed to evaluate a model's robustness under various real-world conditions, identifying potential failure points that may not be apparent

during standard validation. These tests simulate different deployment scenarios, such as varying input distributions or environmental changes, to assess how the model's predictions might vary. Moreover, some researches have been conducted to analyze and mitigate underspecification across different ML tasks [298,299].

## 11 Analysis of Deep Learning Models

This section details the methodology used in this study, which focuses on applying and evaluating various deep learning models for classification tasks across three distinct datasets. For our experimental analysis, we utilized three publicly available datasets: IMDB [300], ARAS [301], and Fruit-360 [302]. The objective is to conduct a comparative analysis of the performance of these deep learning models.

The IMDB dataset, which stands for Internet Movie Database, provides a collection of movie reviews categorized as positive or negative sentiments. ARAS is a dataset comprising annotated sensor events for human activity recognition tasks. Fruit-360 is a dataset consisting of images of various fruit types for classification purposes.

We began by evaluating eight different models: CNN, RNN, LSTM, Bidirectional LSTM, GRU, Bidirectional GRU, TCN, and Transformer on the IMDB and ARAS datasets. Our analysis aimed to compare the performance of these deep learning models across diverse datasets. The CNN model (Convolutional Neural Network) is particularly effective in capturing spatial dependencies, making it suitable for tasks involving structured data. RNN (Recurrent Neural Network) is well-suited for sequential data analysis, while LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models are designed to capture long-term dependencies in sequential data. The Bidirectional LSTM and Bidirectional GRU models provide an additional advantage by processing information in both forward and backward directions.

Additionally, we evaluated eight different CNN-based models: VGG, Inception, ResNet, InceptionResNet, Xception, MobileNet, DenseNet, and NASNet for the classification of fruit images using the Fruit-360 dataset. Given that image data is not sequential or time-dependent, recurrent models were not suitable for this task. CNN-based models are particularly effective for image analysis because of their ability to capture spatial dependencies. Moreover, the faster training time of CNN models is due to their parallel processing capabilities, which allow for efficient computation on GPU (Graphics Processing Unit), thereby accelerating the training process.

To evaluate the performance of these models, we employed assessment metrics such as accuracy, precision, recall, and F1-measure. Accuracy measures the overall correctness of the model's predictions, while precision evaluates the proportion of correctly predicted positive instances. Recall assesses the model's ability to correctly identify positive instances, and F1-measure provides a balanced measure of precision and recall.

$$Accuracy = \frac{Tp + Tn}{Tp + Tn + Fp + Fn} \tag{21}$$

$$Precision = \frac{Tp}{Tp + Fp} \tag{22}$$

$$Recall = \frac{Tp}{Tp + Fn} \tag{23}$$

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{24}$$

where $Tp$ = True Positive, $Tn$ = True Negative, $Fp$ = False Positive, and $Fn$ = False Negative.

By conducting a comprehensive analysis using these metrics, we can gain insights into the strengths and weaknesses of each deep learning model. This comparative evaluation enables us to identify the most effective model for specific datasets and applications, ultimately advancing the field of deep learning and its practical applications.

All experiments were conducted on a GeForce RTX 3050 GPU (Graphics Processing Unit) with 4 Gigabyte of RAM (Random Access Memory).

### 11.1 Methodology and Experiments on IMDB Dataset

The IMDB dataset is a widely used dataset for sentiment analysis tasks. It consists of movie reviews along with their corresponding binary sentiment polarity labels. The dataset contains a total of 50,000 reviews, evenly split into 25,000 training samples and 25,000 testing samples. There is an equal distribution of positive and negative labels, with 25,000 instances of each sentiment. To reduce the correlation between reviews for a given movie, only 30 reviews are included in the dataset [300]. Positive reviews often contain words like "great," "well," and "love," while negative reviews frequently use words like "bad" and "can't." However, certain words such as "one," "character," and "well" appear frequently in both positive and negative reviews, although their usage may differ in terms of frequency between the two sentiment classes [72].

In our analysis, we employed eight different deep learning models including CNN, RNN, LSTM, Bidirectional LSTM, GRU, Bidirectional GRU, TCN, and Transformer for sentiment classification using the IMDB dataset. Fig. 27 presents a structural overview of the deep learning model intended for analyzing the performance of eight different models on the IMDB dataset.
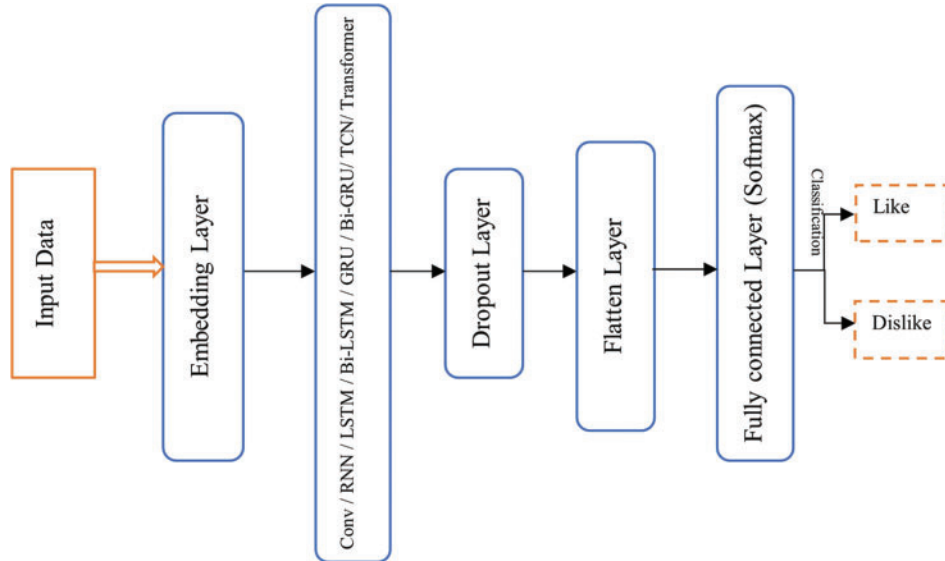


**Figure 27:** The structural for analysis of different deep learning models on IMDB dataset

In this architecture, text data is first passed through an embedding layer, which transforms the high-dimensional, sparse input into dense, lower-dimensional vectors of real numbers. This allows the

model to capture semantic relationships within the data. In the second layer, one of eight models: CNN, RNN, LSTM, Bi-LSTM, GRU, Bi-GRU, TCN, or Transformer is employed for feature extraction and data training. This layer is crucial for capturing patterns and dependencies in the data. Following this, a dropout layer is included to address the issue of overfitting by randomly deactivating a portion of the neurons during training, which helps improve the model's generalization. Subsequently, the multi-dimensional vector turns into a one-dimensional vector using a flatten layer, enabling it to work with fully connected layers. Finally, the output is passed through a fully connected (Dense) layer, which uses a Softmax function for classification, converting the model's predictions into probabilities for each class.

Building a neural network with high accuracy necessitates careful attention to hyperparameter selection, as these adjustments significantly influence the network's performance. For example, setting the number of training iterations too high can lead to overfitting, where the model performs well on the training data but poorly on unseen data. Another critical hyperparameter is the learning rate, which affects the rate of convergence during training. If the learning rate is too high, the network may converge too quickly, potentially overshooting the global minimum of the loss function. Conversely, if the learning rate is too low, the convergence process may become excessively slow, prolonging training. Therefore, finding the optimal balance of hyperparameters is essential for maximizing the network's performance and ensuring effective learning.

In the experiment phase, consistent parameters were applied across all models to ensure a standardized comparison. The parameters were set as follows: epochs = 30, batch size = 64, dropout = 0.2, with the loss function set to "Binary Crossentropy," and the optimizer function set to Stochastic Gradient Descent (SGD) with a learning rate of 0.2. For the CNN model, 100 filters were used with a kernel size of 3, along with the Rectified Linear Unit (ReLU) activation function. The RNN, LSTM, Bi-LSTM, GRU, and Bi-GRU models each employed 64 units. The TCN model was configured with 16 filters, a kernel size of 5, and dilation rates of [1, 2, 4, 8]. The Transformer model was set up with 2 attention heads, a hidden layer size of 64 in the feed-forward network, and the ReLU activation function. These parameter settings and architectural choices were designed to allow for a standardized comparison of the deep learning models on the IMDB dataset. This standardization facilitates an accurate analysis of each model's performance, enabling a comparison of their accuracy and loss values.

Table 2 shows the result of different deep learning models on IMDB review dataset based on various metrics including Accuracy, Precision, Recall, F1-Score, and Time of training.

To compare the performance of these models, we utilized accuracy, validation-accuracy, loss, and validation-loss diagrams. These diagrams provide insights into how well the models are learning from the data and help in evaluating their effectiveness for sentiment classification tasks.

Fig. 28 shows the accuracy and validation-accuracy diagrams where the accuracy, provides a visual representation of how the different deep learning models perform in terms of accuracy during the training process and validation-accuracy shows the trend of accuracy values on the testing set across multiple epochs for each model.

Fig. 29 illustrates the loss and validation-loss diagram where the loss diagram is a visual representation of loss values during the training process for six different models, and the validation-loss diagram depicts the variation in loss values on the testing set during the evaluation process for the different models. The loss function measures the discrepancy between the predicted sentiment labels and the actual labels.

**Table 2:** Result of different deep learning models on the IMDB dataset

| Model | Accuracy % | Precision % | Recall % | F1-Score % | Time (h:m:s) |
|---|---|---|---|---|---|
| CNN | 85.90 | 85.89 | 85.88 | 85.89 | 0:02:57 |
| RNN | 59.03 | 59.03 | 59.02 | 59.03 | 0:12:23 |
| LSTM | 87.53 | 87.53 | 87.54 | 87.54 | 0:09:09 |
| Bi-LSTM | 87.45 | 87.46 | 87.47 | 87.46 | 0:10:43 |
| GRU | 87.55 | 87.56 | 87.57 | 87.56 | 0:05:10 |
| BI-GRU | 87.97 | 87.92 | 87.99 | 87.95 | 0:09:54 |
| TCN | 84.42 | 84.40 | 84.42 | 84.41 | 0:07:38 |
| Transformer | **88.03** | **88.04** | **88.01** | **88.03** | 0:03:44 |
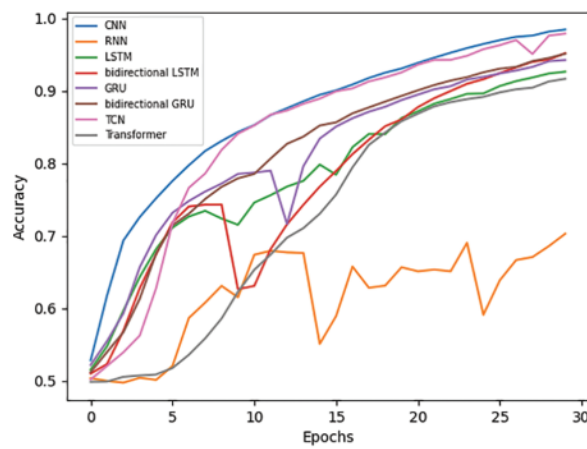


(a) Accuracy Diagram
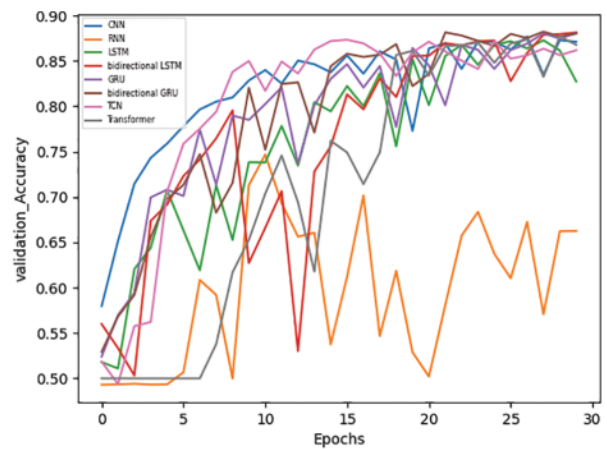
(b) Validation-Accuracy Diagram

**Figure 28:** Accuracy and validation-accuracy of deep learning models on IMDB dataset
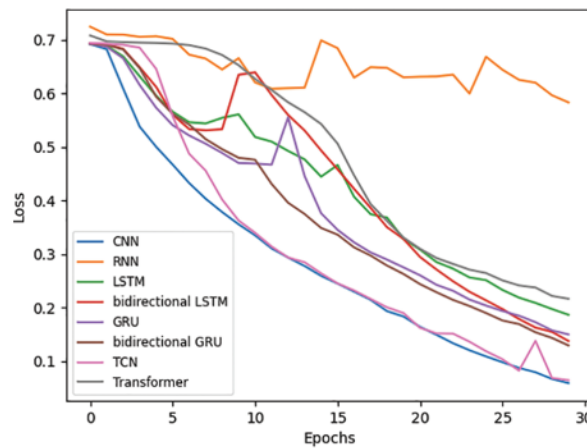


(a) Loss Diagram
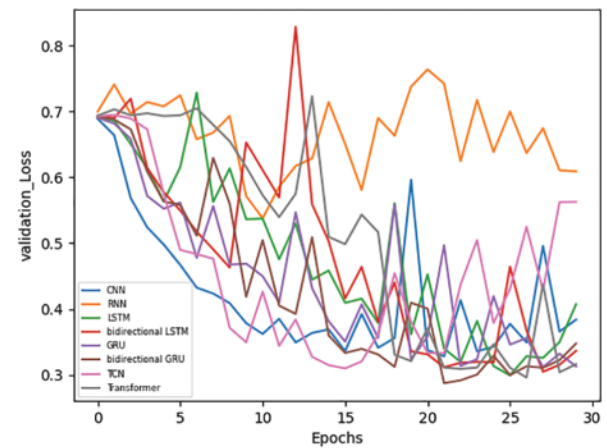
(b) Validation-Loss Diagram

**Figure 29:** Loss and validation-loss diagrams of deep learning models on IMDB dataset

Furthermore, the confusion matrices for the various deep learning models are displayed in Fig. 30. These matrices provide a detailed breakdown of each model's performance, highlighting how well the models classify different classes. By closely examining these confusion matrices, we can gain insights into the precision of the models and identify patterns of misclassification for each class. This analysis helps in understanding the strengths and weaknesses of the models' predictions.
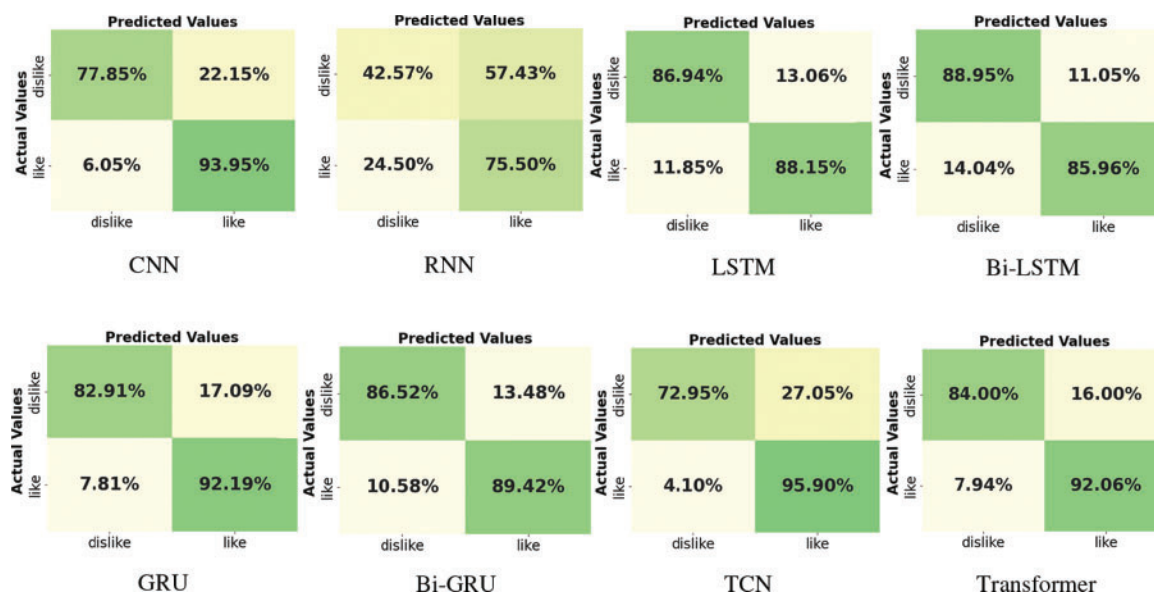


**Figure 30:** Confusion matrix for different deep learning models on IMDB dataset

Additionally, Fig. 31 displays the ROC-AUC (Receiver Operating Characteristic-Area Under Curve) diagrams for eight different deep learning models. These diagrams offer valuable insights into the classification performance of the models, aiding in the assessment of their effectiveness. By analyzing the ROC-AUC curves, we can make informed decisions regarding model selection and threshold adjustments, ensuring a more accurate and effective classification approach.
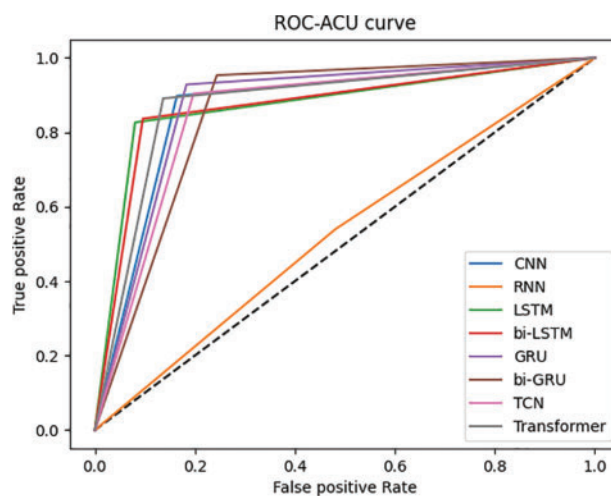


**Figure 31:** ROC-AUC diagrams for different deep learning models

Based on the results provided, it can be concluded that the Transformer and Bi-GRU models achieved the best performance on the IMDB review dataset for sentiment analysis. Both models demonstrated high accuracy in classifying the sentiment of movie reviews. However, it is worth noting that the training time of the Transformer model was significantly less than that of the Bi-GRU model. This suggests that the Transformer model was faster to train compared to the Bi-GRU model while still achieving excellent performance. Additionally, the GRU model also exhibited good accuracy in sentiment classification and required less training time than the Bi-GRU model. Overall, the results suggest that the Transformer, and GRU models are effective deep learning models for sentiment analysis on the IMDB review dataset, with varying trade-offs between performance and training time.

### 11.2 Methodology and Experiments on ARAS Dataset

Based on the provided information, the ARAS dataset [301] is a valuable resource for recognizing human activities in smart environments. It consists of data streams collected from two houses over a period of 60 days, with 20 binary sensors installed to monitor resident activity. The dataset includes information on 27 different activities performed by two residents, and the sensor events are recorded on a per-second basis.

Eight distinct deep learning models were used in our investigation to recognize human activities: CNN, RNN, LSTM, Bidirectional LSTM, GRU, Bidirectional GRU, TCN, and Transformer. A structural overview of the deep learning model designed to analyze the performance of eight different models on the ARAS dataset is shown in Fig. 32.
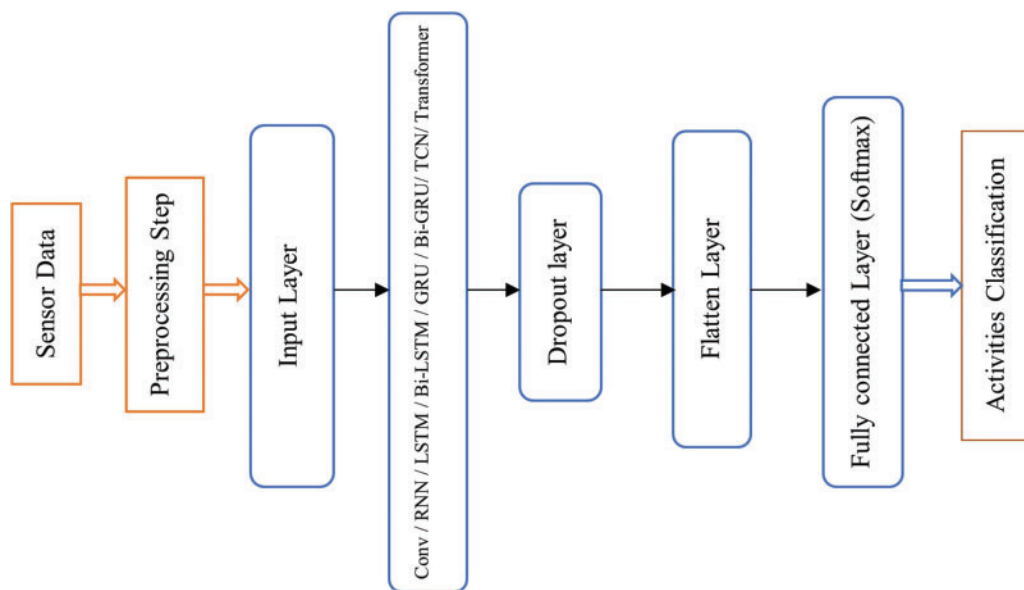


**Figure 32:** The structural for analysis of different deep learning models on the ARAS dataset

The first phase involves preprocessing the sensor data to ensure it is in a suitable and standardized format for deep learning models. The initial task in this phase is data cleaning, where any recorded instances where all sensor events are zero, and the resident is inside the house, are removed from the dataset. Next, a time-based static sliding window technique is applied for segmenting sensor events. This method groups sequences of sensor events into intervals of equal duration. Optimizing the time interval is crucial for effective segmentation; after evaluating intervals ranging from 30 to 360 s, a 90-s

interval was determined to be optimal for the ARAS dataset. The segmentation task aids in decreasing training time and increasing accuracy for the deep learning models.

After preprocessing, the data is passed through an input layer. In the second layer, one of eight models: CNN, RNN, LSTM, Bi-LSTM, GRU, Bi-GRU, TCN, or Transformer is employed for feature extraction and training. This layer plays a vital role in capturing patterns and dependencies within the data. To mitigate overfitting, a dropout layer follows, which randomly deactivates a portion of the neurons during training, thereby improving the model's generalization. Subsequently, a flatten layer is used to convert the multi-dimensional vector into a one-dimensional vector, making it compatible with fully connected layers. Finally, the output passes through a fully connected (Dense) layer, which uses a Softmax function for classification, transforming the model's predictions into probability distributions across the classes.

In the experimental phase, we split the data from the first resident of house B, allocating 70% for training and 30% for testing, using a random split. Additionally, 20% of the training data was set aside for validation. The models were trained with a fixed set of parameters: 30 epochs, a batch size of 64, a dropout rate of 0.2, the "Categorical Crossentropy" loss function, and the Adam optimizer. For the CNN model, we used 100 filters with a kernel size of 3 and the rectified linear unit (ReLU) activation function. The RNN, LSTM, Bi-LSTM, GRU, and Bi-GRU models were configured with 64 units each. The TCN model was set with 16 filters, a kernel size of 5, and dilation rates of [1, 2, 4, 8]. The Transformer model utilized 2 attention heads, a hidden layer size of 64 in the feedforward network, and the ReLU activation function.

Table 3 illustrates the results of experiments on ARAS dataset with various metrices including Accuracy, Precision, Recall, F1-Score, and Time of training.

**Table 3:** Result of different deep learning models on the ARAS dataset

| Model | Accuracy % | Precision % | Recall % | F1-Score % | Time (h:m:s) |
|---|---|---|---|---|---|
| CNN | 93.14 | 95.59 | 92.43 | 93.98 | 0:01:18 |
| RNN | 93.17 | 96.19 | 91.67 | 93.88 | 0:04:09 |
| LSTM | 93.29 | 95.56 | 92.82 | 93.81 | 0:03:23 |
| Bi-LSTM | 93.33 | **96.66** | 92.12 | 94.15 | 0:04:01 |
| GRU | 93.65 | 96.08 | 91.78 | 94.31 | 0:03:15 |
| BI-GRU | 93.90 | 95.87 | 92.61 | 94.49 | 0:03:56 |
| TCN | 94.04 | 95.37 | 93.48 | 94.42 | 0:04:06 |
| Transformer | **94.56** | 95.61 | **94.06** | **94.83** | 0:03:14 |

Also, Fig. 33 presents the accuracy diagram and validation-accuracy diagram for the deep learning models, while Fig. 34 shows the loss diagram and validation-loss diagram for deep learning models.

Since we performed preprocessing tasks like data cleaning and segmentation, the data is nearly normalized and balanced, leading to consistent and closely grouped results across all models. However, the results indicate that the Transformer and TCN models outperformed the others on the ARAS dataset. This outcome aligns with the dataset's nature, which comprises spatial and temporal sequences of sensor events. Among the models, the Transformer exhibited the highest performance in terms of accuracy, recall, and F1-Score, while the Bi-LSTM model excelled in the precision metric. Moreover, the Transformer model demonstrated a notable advantage in training time, second only to the CNN

model, underscoring its efficiency in processing and learning from time-series data. Additionally, when examining the accuracy and loss curves, it is evident that the Transformer, TCN, and CNN models stabilized earlier than the others. Overall, the Transformer model proved to be the most effective for working with the ARAS dataset, striking a balance between accuracy, training time, and consistency throughout the training phases, making it the optimal choice for recognizing human activities based on sensor data.
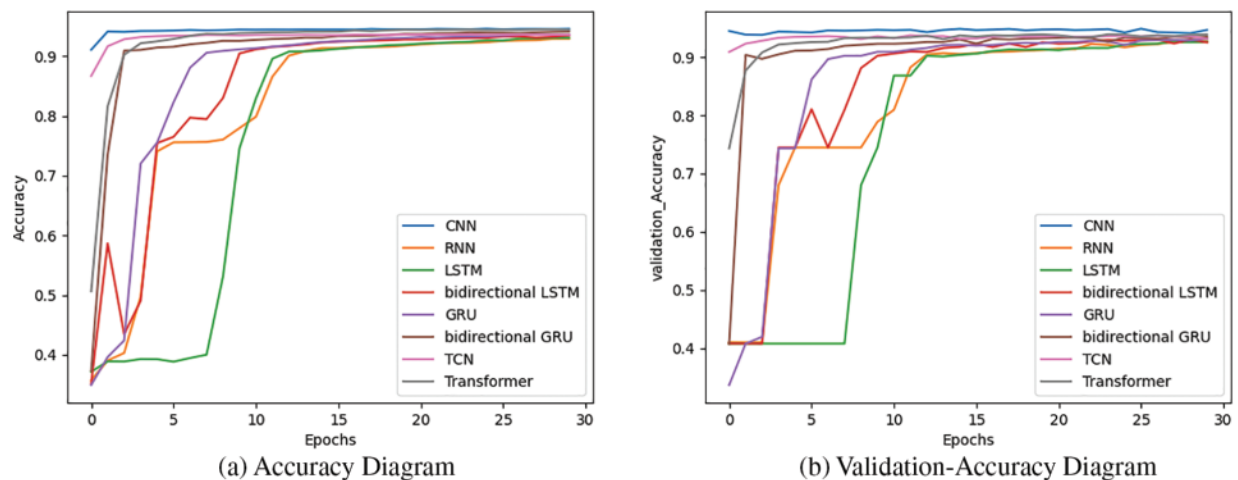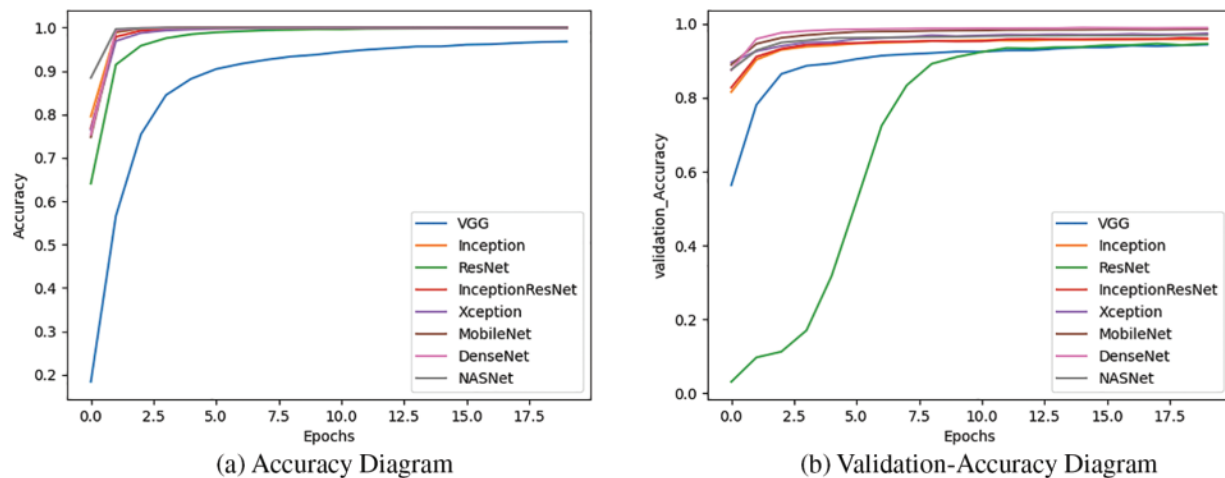


(a) Accuracy Diagram

(b) Validation-Accuracy Diagram

**Figure 33:** Accuracy and validation-accuracy diagrams of deep learning models on ARAS dataset



(a) Loss Diagram

(b) Validation-Loss Diagram
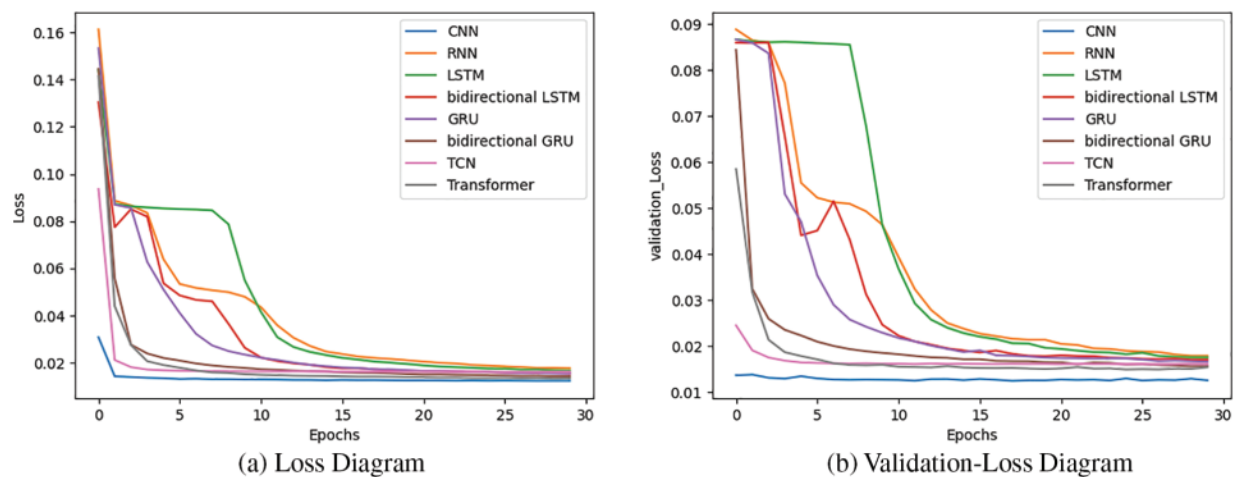
**Figure 34:** Loss and validation-loss diagrams of deep learning models on ARAS dataset

### 11.3 Methodology and Experiments on the Fruit-360 Dataset

Since images are not sequential or time-dependent, recurrent models were less effective for these tasks. CNN-based models, on the other hand, are highly valuable for image analysis due to their ability to capture spatial relationships. Consequently, the analysis of deep learning models on the Fruit-360 dataset for image classification focused on eight CNN variants: VGG, Inception, ResNet, InceptionResNet, Xception, MobileNet, DenseNet, and NASNet. These models use deep transfer

learning technique for training image data and improving classification accuracy. Fig. 35 provides a structural overview of the deep learning models used to evaluate the performance of these eight variants on the Fruit-360 dataset.
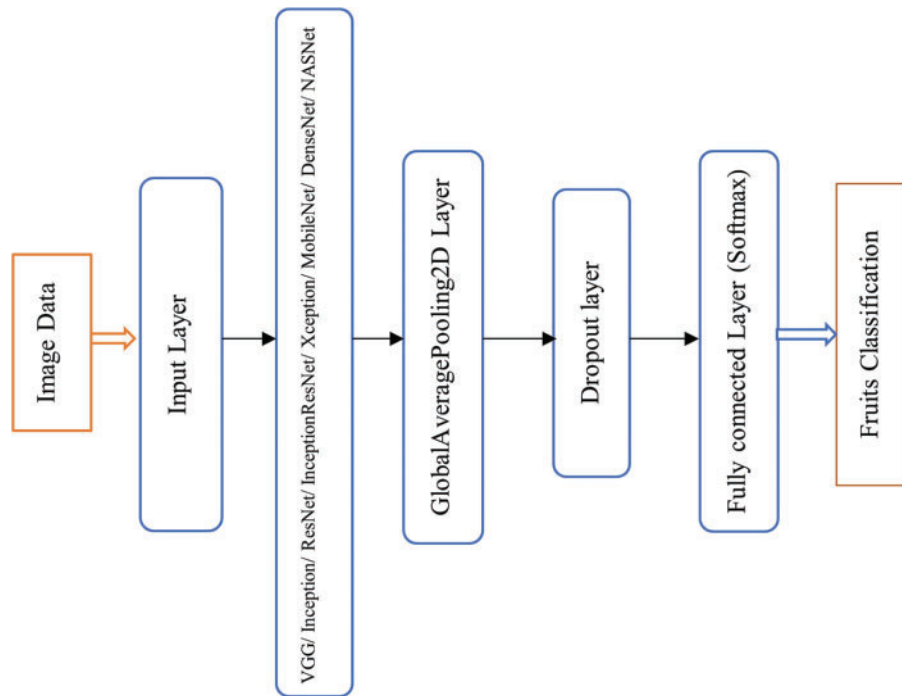


**Figure 35:** The structural for analysis of different CNN-based models on Fruit-360 dataset

First, the fruit images are passed through an input layer. In the second layer, one of eight models (VGG, Inception, ResNet, InceptionResNet, Xception, MobileNet, DenseNet, or NASNet) is employed for feature extraction and training. Next, a Global Average Pooling 2D (GAP) layer is applied, which significantly reduces the spatial dimensions of the data by collapsing each feature map into a single value. To combat overfitting, a dropout layer is then introduced, randomly deactivating a portion of the neurons during training, which enhances the model's ability to generalize. Finally, the output is passed through a fully connected (Dense) layer, where a Softmax function is used to classify the fruit images.

The dataset comprises 55,244 images of 81 different fruit classes, each with a resolution of 100 × 100 pixels. For the experiments, a subset of 60 fruit classes was selected, containing 28,484 images for training and 9558 images for testing. Non-fruit items such as chestnuts and ginger root were removed from the dataset.

All models were trained with a consistent set of parameters: 20 epochs, a batch size of 512, a dropout rate of 0.2, the "Categorical Crossentropy" loss function, and the Adam optimizer. Additionally, all models utilized the "ImageNet" dataset for pre-training.

Table 4 presents the experimental results for various models on the Fruit-360 dataset, including VGG16, InceptionV3, ResNet50, InceptionResNetV2, Xception, MobileNet, DenseNet121, and NASNetLarge. The table includes metrics such as Accuracy, Precision, Recall, F1-Score, and Time of training.

**Table 4:** Result of different deep learning models on the Fruit-360 dataset

| Model | Accuracy % | Precision % | Recall % | F1-Score % | Time (h:m:s) |
|---|---|---|---|---|---|
| VGG | 94.39 | **99.79** | 80.65 | 89.20 | 2:17:32 |
| Inception | 95.86 | 96.65 | 95.14 | 95.89 | 0:23:34 |
| ResNet | 94.59 | 95.30 | 93.64 | 94.46 | 1:12:56 |
| InceptionResNet | 96.05 | 97.01 | 95.36 | 96.18 | 0:54:18 |
| Xception | 97.38 | 98.28 | 96.61 | 97.44 | 1:01:11 |
| MobileNet | 98.54 | 98.88 | 98.28 | 98.58 | 0:17:22 |
| DenseNet | **98.94** | 99.12 | **98.75** | **98.94** | 1:10:30 |
| NASNet | 96.99 | 97.69 | 96.56 | 97.12 | 3:50:05 |

Furthermore, the accuracy, validation-accuracy, loss, and validation-loss diagrams were used to compare the performance of various models. When assessing the models' performance for tasks involving the categorization of fruit photos, these graphs offer valuable insights into how effectively the models are learning from the data. Fig. 36 shows the accuracy and validation-accuracy diagram of the deep learning models, while Fig. 37 illustrates the loss diagram and validation-loss diagram of the deep learning models.



(a) Accuracy Diagram

(b) Validation-Accuracy Diagram

**Figure 36:** Accuracy and validation-accuracy diagrams of different CNN-based deep learning models on Friut-360 dataset

Based on the results, it can be concluded that the DenseNet and MobileNet models achieved the best performance for fruit image classification on the Fruit-360 dataset. Both models demonstrated high accuracy in classifying fruit images. Notably, MobileNet had a significantly shorter training time compared to DenseNet, indicating that it was faster to train while still delivering performance close to that of DenseNet. Additionally, the Xception model also showed good accuracy and required less training time than DenseNet. Overall, the MobileNet model stands out as a favorable choice due to its balance between accuracy and training efficiency.
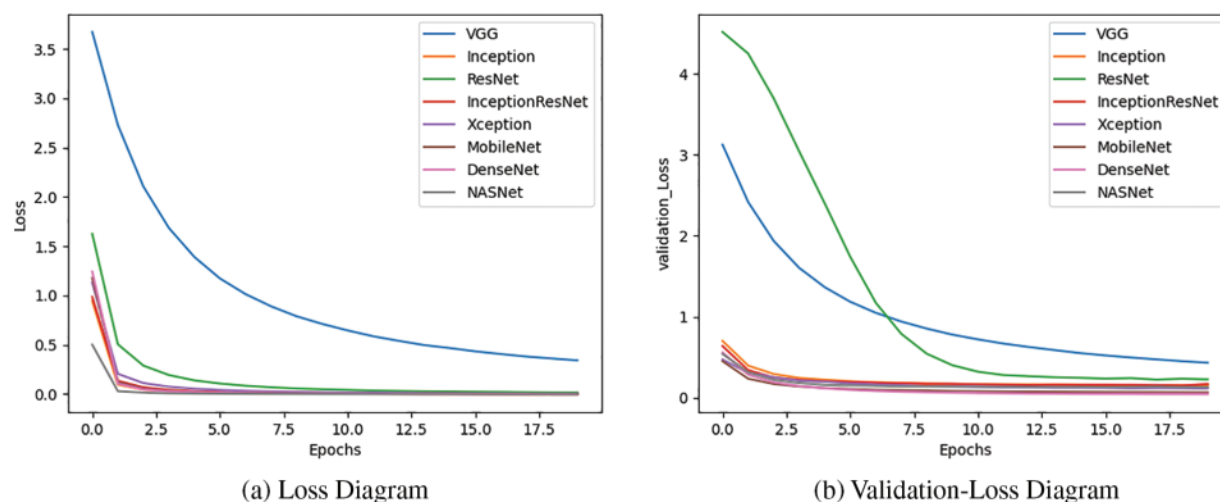
**Figure 37:** Loss and validation-loss diagrams of different CNN-based deep learning models on Friut-360 dataset

## 12 Research Directions and Future Aspects

In the preceding sections, we explored a range of deep learning topics, highlighting both the advantages and limitations of various deep learning models. Additionally, we examined the application of several models across different domains. Despite the benefits demonstrated, our research has identified certain gaps, indicating that further advancements are necessary. This section outlines potential future research directions based on our analysis.

- *Generative ( Unsupervised) Models:* Generative models, a key category of deep learning models discussed in Section 4, hold significant promise for future research. These models enable the creation of new data representations through exploratory analysis and can identify high-order correlations or features in data. Unlike supervised learning, unsupervised models can derive insights from data without the need for labeled examples, making them valuable for various applications. Several generative models, including Autoencoders, Generative Adversarial Networks (GANs), Deep Belief Networks (DBNs), and Self-Organizing Maps (SOMs), have been developed and employed across diverse contexts. A promising research avenue involves analyzing these models in various settings and developing new methods or variations that enhance data modeling or representation for specific real-world applications. The rising interest in GANs is particularly noteworthy, as they excel in leveraging unlabeled image data for deep representation learning and training highly non-linear mappings between latent and data spaces. The GAN framework offers the flexibility to formulate new theories and methods tailored to emerging deep learning applications, positioning it as a pivotal area for future exploration.

- *Hybrid/Ensemble Modeling*: Hybrid deep learning architectures have shown great potential in enhancing model performance by combining components from multiple models. For instance, the integration of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can capture both temporal and spatial dependencies in data, leveraging the strengths of each model. Hybrid models also benefit from combining generative and supervised learning, offering superior performance and improved uncertainty handling in high-risk scenarios. Developing effective hybrid models, whether supervised or unsupervised, presents a significant research opportunity to address a wide range of real-world problems, including semi-supervised

learning tasks and model uncertainty. This approach moves beyond conventional, isolated models, emphasizing the need for sophisticated methods that can handle the complexity of various data types and applications.

- *Hyperparameter Optimization for Efficient Deep Learning*: As deep learning models have evolved, the number of parameters, computational latency, and resource requirements have increased substantially [152]. Selecting the appropriate hyperparameters is critical to building a neural network with high accuracy. Key hyperparameters include learning rate, loss function, batch size, number of training iterations, and dropout rate, among others. The challenge lies in finding an optimal balance of these parameters, as they significantly influence network performance. However, iterating through all possible combinations of hyperparameters is computationally expensive. To address this, metaheuristic optimization techniques, such as Genetic Algorithm (GA) [303], Particle Swarm Optimization (PSO) [304], and others, can be employed to explore the search space more efficiently than exhaustive methods. Future research should focus on optimizing hyperparameters tailored to specific data types and contexts. For example, the learning rate plays a crucial role in training, where a rate too high may cause the model to converge prematurely, while a rate too low can lead to slow convergence and prolonged training times. Adaptive learning rate techniques, such as including Adaptive Moment Estimation (Adam) [305], Stochastic Gradient Descent (SGD) [306], adaptive gradient algorithm (ADAGRAD) [307], and Nesterov-accelerated Adaptive Moment Estimation (Nadam) [308], and more recent innovations like Evolved Sign Momentum (Lion) [309], offer promising avenues for improving network performance and minimizing loss functions. Future research could further explore these optimizers, focusing on their comparative effectiveness in enhancing model performance through iterative weight and bias adjustments.
- *Federated Learning*: Federated learning is an emerging deep learning paradigm that enables collaborative model training across multiple organizations or teams without the need to share raw data. This approach is particularly relevant in contexts where data privacy is paramount. However, federated learning introduces new challenges, especially with the advent of data fusion technologies that combine data from multiple sources with varying formats. As data diversity and volume continue to grow, optimizing data and model utilization in federated learning becomes increasingly important. Addressing challenges such as safeguarding user privacy, developing universal models, and ensuring the stability of data fusion outcomes will be crucial for the future application of federated learning across multiple domains [310].
- *Quantum Deep Learning:* Quantum computing and deep learning have both seen significant advancements over the past few decades. Quantum computing, which leverages the principles of quantum mechanics to store and process information, has the potential to outperform classical supercomputers on certain tasks, making it a powerful tool for complex problem-solving. The intersection of quantum computing and deep learning has led to the emergence of quantum deep learning and quantum-inspired deep learning algorithms. Future research directions in this area include investigating and developing quantum deep learning models, such as Quantum Convolutional Neural Network (Quantum CNN) [311], Quantum Recurrent Neural Network (Quantum RNN) [312], Quantum Generative Adversarial Network (Quantum GAN) [313], and others. Additionally, exploring the application of these models across various domains and creating novel quantum deep learning architectures represents a cutting-edge frontier in the field [314,315].

In conclusion, the research directions outlined above underscore the dynamic and evolving nature of deep learning. By addressing these challenges and exploring new avenues, the field can continue to

advance, driving innovation and enabling the development of more powerful and efficient models for a wide range of applications.

## 13 Conclusion

This article provides an extensive overview of deep learning technology and its applications in machine learning and artificial intelligence. The article covers various aspects of deep learning, including neural networks, MLP models, and different types of deep learning models such as CNN, RNN, TCN, Transformer, generative models, DRL, and transfer learning. The classification of deep learning models allows for a better understanding of their specific applications and characteristics. The RNN models, including LSTM, Bi-LSTM, GRU, and Bi-GRU, are particularly suited for time series data due to their ability to capture temporal dependencies. On the other hand, CNN-based models excel in image data analysis by effectively capturing spatial features.

The experiments conducted on three public datasets, namely IMDB, ARAS, and Fruit-360, further reinforce the suitability of specific deep learning models for different data types. The results demonstrate that the CNN-based models such as DenseNet and MobileNet perform exceptionally well in image classification tasks. The RNN models, such as LSTM and GRU, show strong performance in time series analysis. However, the Transformer model outperforms classical RNN-based models, particularly in text analysis, due to its use of the attention mechanism.

Overall, this article highlights the diverse applications and effectiveness of deep learning models in various domains. It emphasizes the importance of selecting the appropriate deep learning model based on the nature of the data and the task at hand. The insights gained from the experiments contribute to a better understanding of the strengths and weaknesses of different deep learning models, facilitating informed decision-making in practical applications.

**Author Contributions:** The authors confirm contribution to the paper as follows: Study conception and design: Farhad Mortezapour Shiri, Thinagaran Perumal; data collection: Farhad Mortezapour Shiri; analysis and interpretation of results: Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, Raihani Mohamed; draft manuscript preparation: Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, Raihani Mohamed. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** The code used and/or analyzed during this research are available from the corresponding author upon reasonable request. Data used in this study can be accessed via the following links: IMDB dataset: https://ai.stanford.edu/~amaas/data/sentiment/, 6/19/2011; ARAS dataset: http://aras.cmpe.boun.edu.tr/download.php, 7/22/2013; Fruit-360 dataset: https://data.mendeley.com/datasets/rp73yg93n8/1, 10/20/2018 (accessed on 22 October 2024).

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1]    P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *4th Int. Conf. Comput. Commun. Ctrl. Autom. (ICCUBEA)*, Pune, India, IEEE, Aug. 16–18, 2018, pp. 1–6. doi: 10.1109/ICCUBEA.2018.8697857.

[2]    C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electron. Mark.*, vol. 31, no. 3, pp. 685–695, 2021. doi: 10.1007/s12525-021-00475-2.

[3]    W. Han *et al.*, "A survey of machine learning and deep learning in remote sensing of geological environment: Challenges, advances, and opportunities," *ISPRS J. Photogramm. Remote. Sens.*, vol. 202, pp. 87–113, 2023. doi: 10.1016/j.isprsjprs.2023.05.032.

[4]    S. Zhang *et al.*, "Deep learning in human activity recognition with wearable sensors: A review on advances," *Sensors*, vol. 22, no. 4, Feb. 14, 2022. doi: 10.3390/s22041476.

[5]    S. Li, Y. Tao, E. Tang, T. Xie, and R. Chen, "A survey of field programmable gate array (FPGA)-based graph convolutional neural network accelerators: Challenges and opportunities," *PeerJ Comput. Sci.*, vol. 8, no. 9, 2022, Art. no. e1166. doi: 10.7717/peerj-cs.1166.

[6]    A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: An overview," in *Adv. Mach. Learn. Technol. App.: AMLTA 2020*, 2021, pp. 599–608.

[7]    J. Liu and Y. Jin, "A comprehensive survey of robust deep learning in computer vision," *J. Autom. Intell.*, vol. 2, no. 4, pp. 175–195, 2023. doi: 10.1016/j.jai.2023.10.002.

[8]    A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019. doi: 10.1109/ACCESS.2019.2912200.

[9]    M. A. Wani, F. A. Bhat, S. Afzal, and A. I. Khan, *Advances in Deep Learning*. Singapore: Springer, 2020.

[10]   L. Alzubaidi *et al.*, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, pp. 1–74, 2021. doi: 10.1186/s40537-021-00444-8.

[11]   I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Comput. Sci.*, vol. 2, no. 6, p. 420, 2021. doi: 10.1007/s42979-021-00815-1.

[12]   M. N. Hasan, T. Ahmed, M. Ashik, M. J. Hasan, T. Azmin and J. Uddin, "An analysis of COVID-19 pandemic outbreak on economy using neural network and random forest," *J. Inf. Syst. Telecommun.*, vol. 2, no. 42, 2023, Art. no. 163. doi: 10.52547/jist.34246.11.42.163.

[13]   N. B. Gaikwad, V. Tiwari, A. Keskar, and N. Shivaprakash, "Efficient FPGA implementation of multilayer perceptron for real-time human activity classification," *IEEE Access*, vol. 7, pp. 26696–26706, 2019. doi: 10.1109/ACCESS.2019.2900084.

[14]   K. -C. Ke and M. -S. Huang, "Quality prediction for injection molding by using a multilayer perceptron neural network," *Polymers*, vol. 12, no. 8, 2020, Art. no. 1812. doi: 10.3390/polym12081812.

[15]   A. Tasdelen and B. Sen, "A hybrid CNN-LSTM model for pre-miRNA classification," *Sci. Rep.*, vol. 11, no. 1, pp. 1–9, 2021. doi: 10.1038/s41598-021-93656-0.

[16]   L. Qin, N. Yu, and D. Zhao, "Applying the convolutional neural network deep learning technology to behavioural recognition in intelligent video," *Tehnički Vjesnik*, vol. 25, no. 2, pp. 528–535, 2018. doi: 10.17559/TV-20171229024444.

[17]   Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022. doi: 10.1109/TNNLS.2021.3084827.

[18]   B. P. Babu and S. J. Narayanan, "One-vs-all convolutional neural networks for synthetic aperture radar target recognition," *Cybern Inf. Technol.*, vol. 22, pp. 179–197, 2022. doi: 10.2478/cait-2022-0035.

[19]   S. Mekruksavanich and A. Jitpattanakul, "Deep convolutional neural network with rnns for complex activity recognition using wrist-worn wearable sensor data," *Electronics*, vol. 10, no. 14, 2021, Art. no. 1685. doi: 10.3390/electronics10141685.

[20]  W. Lu, J. Li, J. Wang, and L. Qin, "A CNN-BiLSTM-AM method for stock price prediction," *Neural Comput. Appl.*, vol. 33, pp. 4741–4753, 2021. doi: 10.1007/s00521-020-05532-z.

[21]  W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, 2017. doi: 10.1162/neco_a_00990.

[22]  L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang and Y. Miao, "Review of image classification algorithms based on convolutional neural networks," *Remote Sens.*, vol. 13, no. 22, 2021, Art. no. 4712. doi: 10.3390/rs13224712.

[23]  J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, 2018. doi: 10.1016/j.patcog.2017.10.013.

[24]  S. Salman and X. Liu, "Overfitting mechanism and avoidance in deep neural networks," 2019, *arXiv:1901.06566*.

[25]  A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," in *2020 Int. Conf. Emerg. Tren. Inf. Technol. Engr. (ic-ETITE)*, IEEE, 2020, pp. 1–5.

[26]  W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017. doi: 10.1016/j.neucom.2016.12.038.

[27]  K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015. doi: 10.1109/TPAMI.2015.2389824.

[28]  D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Rough. Sets. Knwl. Technol. 9th Int. Conf., RSKT*, Shanghai, China, Springer, Oct. 24–26, 2014, pp. 364–375. doi: 10.1007/978-3-319-11740-9_34.

[29]  Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Comput. Vis. (ECCV): 13th Europ. Conf.*, Zurich, Switzerland, Springer, Sep. 6–12, 2014, pp. 392–407.

[30]  M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," 2013, *arXiv:1301.3557*.

[31]  V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016, *arXiv:1603.07285*.

[32]  M. Krichen, "Convolutional neural networks: A survey," *Computers*, vol. 12, no. 8, 2023, Art. no. 151. doi: 10.3390/computers12080151.

[33]  S. Kılıçarslan, K. Adem, and M. Çelik, "An overview of the activation functions used in deep learning algorithms," *J. New Results Sci.*, vol. 10, no. 3, pp. 75–88, 2021. doi: 10.54187/jnrs.1011739.

[34]  C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018, *arXiv:1811.03378*.

[35]  K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *Int. Jt. Conf. Neural. Netw. (IJCNN)*, Killarney, Ireland, 2015, pp. 1–8.

[36]  A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Atlanta, GA, USA, 2013, vol. 30, no. 1.

[37]  K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[38]  B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*.

[39]  X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep learning with s-shaped rectified linear activation units," *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, 2016. doi: 10.1609/aaai.v30i1.10287.

[40]  D. -A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*.

[41]  D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[42]  A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Adv Neural Inf. Process. Syst.*, vol. 25, pp. 1097–1105, 2012.

[43]  K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[44] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2015, pp. 1–9.

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2016, pp. 2818–2826.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2016, pp. 770–778.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Comput. Vis. (ECCV): 14th Europ. Conf.*, Amsterdam, The Netherlands, Springer, Oct. 11–14, 2016, pp. 630–645.

[48] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

[49] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," 2016, *arXiv:1605.07648*.

[50] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.

[51] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017. doi: 10.1609/aaai.v31i1.11231.

[52] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2017, pp. 1251–1258.

[53] G. A. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[54] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. -C. Chen, "MobileNetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2018, pp. 4510–4520.

[55] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2017, pp. 4700–4708.

[56] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2018, pp. 7132–7141.

[57] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 6105–6114.

[58] M. Tan and Q. Le, "EfficientNetv2: Smaller models and faster training," in *Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 10096–10106.

[59] S. Abbaspour, F. Fotouhi, A. Sedaghatbaf, H. Fotouhi, M. Vahabi and M. Linden, "A comparative analysis of hybrid deep learning models for human activity recognition," *Sensors*, vol. 20, no. 19, 2020, Art. no. 5707. doi: 10.3390/s20195707.

[60] W. Fang, Y. Chen, and Q. Xue, "Survey on research of RNN-based spatio-temporal sequence prediction algorithms," *J. Big Data*, vol. 3, no. 3, pp. 97–110, 2021. doi: 10.32604/jbd.2021.016993.

[61] J. Xiao and Z. Zhou, "Research progress of RNN language model," in *2020 IEEE Int. Conf. Artif. Intell. Comput. App. (ICAICA)*, Dalian, China, IEEE, Jun. 27–29, 2020, pp. 1285–1288. doi: 10.1109/ICAICA50127.2020.9182390.

[62] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2015, pp. 4694–4702.

[63] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU," *J. Artif. Intell. Soft Comput. Res.*, vol. 9, no. 4, pp. 235–245, 2019. doi: 10.2478/jaiscr-2019-0006.

[64] H. Apaydin, H. Feizi, M. T. Sattari, M. S. Colak, S. Shamshirband and K. -W. Chau, "Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting," *Water*, vol. 12, no. 5, 2020, Art. no. 1500. doi: 10.3390/w12051500.

[65] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[66] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, 2008. doi: 10.1109/TPAMI.2008.137.

[67] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[68] J. Chen, D. Jiang, and Y. Zhang, "A hierarchical bidirectional GRU model with attention for EEG-based emotion classification," *IEEE Access*, vol. 7, pp. 118530–118540, 2019. doi: 10.1109/ACCESS.2019.2936817.

[69] M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian recurrent neural networks," 2017, *arXiv:1704.02798*.

[70] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger, "Rainfall-runoff modelling using long short-term memory (LSTM) networks," *Hydrol Earth Syst. Sci.*, vol. 22, no. 11, pp. 6005–6022, 2018. doi: 10.5194/hess-22-6005-2018.

[71] A. Graves, "Generating sequences with recurrent neural networks," 2013, *arXiv:1308.0850*.

[72] S. Minaee, E. Azimi, and A. Abdolrashidi, "Deep-sentiment: Sentiment analysis using ensemble of cnn and bi-lstm models," 2019, *arXiv:1904.04206*.

[73] D. Gaur and S. Kumar Dubey, "Development of activity recognition model using LSTM-RNN deep learning algorithm," *J. Inf. Organ. Sci.*, vol. 46, no. 2, pp. 277–291, 2022. doi: 10.31341/jios.46.2.1.

[74] X. Zhu, P. Sobihani, and H. Guo, "Long short-term memory over recursive structures," in *Int. Conf. Mach. Learn.*, PMLR, 2015, pp. 1604–1612.

[75] F. Gu, M. -H. Chung, M. Chignell, S. Valaee, B. Zhou and X. Liu, "A survey on deep learning for human activity recognition," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–34, 2021. doi: 10.1145/3472290.

[76] T. H. Aldhyani and H. Alkahtani, "A bidirectional long short-term memory model algorithm for predicting COVID-19 in gulf countries," *Life*, vol. 11, no. 11, 2021, Art. no. 1118. doi: 10.3390/life11111118.

[77] F. M. Shiri, E. Ahmadi, M. Rezaee, and T. Perumal, "Detection of student engagement in E-learning environments using EfficientNetV2-L together with RNN-based models," *J. Artif. Intell.*, vol. 6, no. 1, pp. 85–103, 2024. doi: 10.32604/jai.2024.048911.

[78] D. Liciotti, M. Bernardini, L. Romeo, and E. Frontoni, "A sequential deep learning application for recognising human activities in smart homes," *Neurocomputing*, vol. 396, pp. 501–513, 2020. doi: 10.1016/j.neucom.2018.10.104.

[79] A. Dutta, S. Kumar, and M. Basu, "A gated recurrent unit approach to bitcoin price prediction," *J. Risk Financial Manag.*, vol. 13, no. 2, 2020, Art. no. 23. doi: 10.3390/jrfm13020023.

[80] A. Gumaei, M. M. Hassan, A. Alelaiwi, and H. Alsalman, "A hybrid deep learning model for human activity recognition using multimodal body sensing data," *IEEE Access*, vol. 7, pp. 99152–99160, 2019. doi: 10.1109/ACCESS.2019.2927134.

[81] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[82] F. M. Shiri, T. Perumal, N. Mustapha, R. Mohamed, M. A. B. Ahmadon and S. Yamaguchi, "Recognition of student engagement and affective states using ConvNeXtlarge and ensemble GRU in E-learning," in *2024 12th Int. Conf. Inf. Edu. Technol. (ICIET)*, Yamaguchi, Japan, IEEE, Mar. 18–20, 2024, pp. 30–34. doi: 10.1109/ICIET60671.2024.10542707.

[83] C. Chai *et al.*, "A multifeature fusion short-term traffic flow prediction model based on deep learnings," *J. Adv. Transp.*, vol. 2022, no. 1, 2022, Art. no. 1702766. doi: 10.1155/2022/1702766.

[84] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 156–165.

[85] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.

[86] Y. He and J. Zhao, "Temporal convolutional networks for anomaly detection in time series," *J. Phys.: Conf. Ser.*, vol. 1213, no. 4, 2019, Art. no. 042050. doi: 10.1088/1742-6596/1213/4/042050.

[87] J. Zhu, L. Su, and Y. Li, "Wind power forecasting based on new hybrid model with TCN residual modification," *Energy AI*, vol. 10, 2022, Art. no. 100199. doi: 10.1016/j.egyai.2022.100199.

[88]  D. Li, F. Jiang, M. Chen, and T. Qian, "Multi-step-ahead wind speed forecasting based on a hybrid decomposition method and temporal convolutional networks," *Energy*, vol. 238, 2022, Art. no. 121981. doi: 10.1016/j.energy.2021.121981.

[89]  X. Zhang, F. Dong, G. Chen, and Z. Dai, "Advance prediction of coastal groundwater levels with temporal convolutional and long short-term memory networks," *Hydrol. Earth Syst. Sci.*, vol. 27, no. 1, pp. 83–96, 2023. doi: 10.5194/hess-27-83-2023.

[90]  F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.

[91]  T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 29, no. 29, pp. 901–909, 2016.

[92]  N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[93]  Z. Liu *et al.*, "KAN: Kolmogorov-Arnold Networks," 2024, *arXiv:2404.19756*.

[94]  J. Braun and M. Griebel, "On a constructive proof of Kolmogorov's superposition theorem," *Constr. Approx.*, vol. 30, pp. 653–675, 2009. doi: 10.1007/s00365-009-9054-2.

[95]  A. D. Bodner, A. S. Tepsich, J. N. Spolski, and S. Pourteau, "Convolutional Kolmogorov-Arnold Networks," 2024, *arXiv:2406.13155*.

[96]  R. Genet and H. Inzirillo, "TkAN: Temporal Kolmogorov-Arnold Networks," 2024, *arXiv:2405.07344*.

[97]  K. Pan, X. Zhang, and L. Chen, "Research on the training and application methods of a lightweight agricultural domain-specific large language model supporting Mandarin Chinese and Uyghur," *Appl. Sci.*, vol. 14, no. 13, 2024, Art. no. 5764. doi: 10.3390/app14135764.

[98]  R. Genet and H. Inzirillo, "A temporal Kolmogorov-Arnold transformer for time series forecasting," 2024, *arXiv:2406.02486*.

[99]  K. Xu, L. Chen, and S. Wang, "Kolmogorov-Arnold Networks for time series: Bridging predictive power and interpretability," 2024, *arXiv:2406.02496*.

[100]  A. A. Aghaei, "fKAN: Fractional Kolmogorov-Arnold Networks with trainable Jacobi basis functions," 2024, *arXiv:2406.07456*.

[101]  Z. Bozorgasl and H. Chen, "Wav-KAN: Wavelet Kolmogorov-Arnold Networks," 2024, *arXiv:2405.12832*.

[102]  F. Zhang and X. Zhang, "GraphKAN: Enhancing feature extraction with graph Kolmogorov Arnold networks," 2024, *arXiv:2406.13597*.

[103]  A. Jabbar, X. Li, and B. Omar, "A survey on generative adversarial networks: Variants, applications, and training," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–49, 2021. doi: 10.1145/3463475.

[104]  D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," in *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*. Cham: Springer, 2023, pp. 353–374.

[105]  I. Goodfellow *et al.*, "Generative adversarial nets," *Adv. Neural. Inf. Process. Syst.*, vol. 27, pp. 2672–2680, 2014.

[106]  N. Zhang, S. Ding, J. Zhang, and Y. Xue, "An overview on restricted Boltzmann machines," *Neurocomputing*, vol. 275, pp. 1186–1199, 2018. doi: 10.1016/j.neucom.2017.09.065.

[107]  G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, 2009, Art. no. 5947. doi: 10.4249/scholarpedia.5947.

[108]  I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge: MIT Press, 2016.

[109]  J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *2018 IEEE Int. Conf. Syst. Man. Cybern. (SMC)*, Miyazaki, Japan, IEEE, Oct. 7–10, 2018, pp. 415–419. doi: 10.1109/SMC.2018.00080.

[110]  A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2015, *arXiv:1511.05644*.

[111]  Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016. doi: 10.1016/j.neucom.2015.08.104.

[112] Y. N. Kunang, S. Nurmaini, D. Stiawan, and A. Zarkasi, "Automatic features extraction using autoencoder in intrusion detection system," in *2018 Int. Conf. Electr. engr. Comput. Sci. (ICECOS)*, Pangkal, Indonesia, IEEE, Oct. 2–4, 2018, pp. 219–224. doi: 10.1109/ICECOS.2018.8605181.

[113] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. 23rd ACM SIGKDD Int. Conf. Knwl. Discov. Data Mining*, 2017, pp. 665–674. doi: 10.1145/3097983.3098052.

[114] A. Creswell and A. A. Bharath, "Denoising adversarial autoencoders," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 968–984, 2018. doi: 10.1109/TNNLS.2018.2852738.

[115] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.

[116] A. Ng, "Sparse autoencoder," in *CS294A Lecture Notes*, 2011, vol. 72, no. 2011, pp. 1–19.

[117] S. Rifai *et al.*, "Higher order contractive auto-encoder," in *Mach. Learn. Knwl. Discov. DB.: Europ. Conf. ECML PKDD*, Athens, Greece, Springer, Sep. 5–9, 2011, pp. 645–660.

[118] P. Vincent, H. Larochelle, Y. Bengio, and P. -A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103. doi: 10.1145/1390156.1390294.

[119] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found Trends®. Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019. doi: 10.1561/2200000056.

[120] M. -Y. Liu and O. Tuzel, "Coupled generative adversarial networks," *Adv. Neural Inf. Process. Syst.*, vol. 29, pp. 469–477, 2016.

[121] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 921–934, 2019. doi: 10.1109/TEVC.2019.2895748.

[122] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *Int. J. Inf. Manag. Data Insights*, vol. 1, no. 1, 2021, Art. no. 100004. doi: 10.1016/j.jjimei.2020.100004.

[123] B. -C. Chen and A. Kae, "Toward realistic image compositing with adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2019, pp. 8415–8424.

[124] D. P. Jaiswal, S. Kumar, and Y. Badr, "Towards an artificial intelligence aided design approach: Application to anime faces with generative adversarial networks," *Proc. Comput. Sci.*, vol. 168, pp. 57–64, 2020. doi: 10.1016/j.procs.2020.02.257.

[125] Y. Liu, Q. Li, and Z. Sun, "Attribute-aware face aging with wavelet-based generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2019, pp. 11877–11886.

[126] J. Islam and Y. Zhang, "GAN-based synthetic brain PET image generation," *Brain Inform.*, vol. 7, pp. 1–12, 2020. doi: 10.1186/s40708-020-00104-2.

[127] H. Lan, A. D. N. Initiative, A. W. Toga, and F. Sepehrband, "SC-GAN: 3D self-attention conditional GAN with spectral normalization for multi-modal neuroimaging synthesis," *bioRxiv*, 2020. doi: 10.1101/2020.06.09.143297.

[128] K. A. Zhang, A. Cuesta-Infante, L. Xu, and K. Veeramachaneni, "SteganoGAN: High capacity image steganography with GANs," 2019, *arXiv:1901.03892*.

[129] S. Nam, Y. Kim, and S. J. Kim, "Text-adaptive generative adversarial networks: Manipulating images with natural language," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 42–51, 2018.

[130] L. Sixt, B. Wild, and T. Landgraf, "RenderGAN: Generating realistic labeled data," *Front Robot. AI.*, vol. 5, 2018, Art. no. 66. doi: 10.3389/frobt.2018.00066.

[131] K. Lin, D. Li, X. He, Z. Zhang, and M. -T. Sun, "Adversarial ranking for language generation," *Adv Neural Inf. Process. Syst.*, vol. 30, pp. 3158–3168, 2017.

[132] D. Xu, C. Wei, P. Peng, Q. Xuan, and H. Guo, "GE-GAN: A novel deep learning framework for road traffic state estimation," *Transp. Res. Part C Emerg.*, vol. 117, 2020, Art. no. 102635. doi: 10.1016/j.trc.2020.102635.

[133] A. Clark, J. Donahue, and K. Simonyan, "Adversarial video generation on complex datasets," 2019, *arXiv:1907.06571*.

[134] E. L. Denton, S. Chintala, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," *Adv Neural Inf. Process. Syst.*, vol. 28, pp. 1486–1494, 2015.

[135] C. Li and M. Wand, "Precomputed real-time texture synthesis with Markovian generative adversarial networks," in *Comput. Vis. (ECCV): 14th Europ. Conf.*, Amsterdam, Netherlands, Springer, Oct. 11–14, 2016, pp. 702–716. doi: 10.1007/978-3-319-46487-9_43.

[136] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," 2016, *arXiv:1611.02163*.

[137] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 214–223.

[138] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks," 2017, *arXiv:1703.10717*.

[139] J. -Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2223–2232.

[140] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 1857–1865.

[141] A. Jolicoeur-Martineau, "The relativistic discriminator: A key element missing from standard GAN," 2018, *arXiv:1807.00734*.

[142] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2019, pp. 4401–4410.

[143] G. Zhao, M. E. Meyerand, and R. M. Birn, "Bayesian conditional GAN for MRI brain image synthesis," 2005, *arXiv:2005.11875*.

[144] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu and Y. Liu, "Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–40, 2021. doi: 10.1145/3447744.

[145] N. Alqahtani *et al.*, "Deep belief networks (DBN) with IoT-based Alzheimer's disease detection and classification," *Appl. Sci.*, vol. 13, no. 13, 2023, Art. no. 7833. doi: 10.3390/app13137833.

[146] A. P. Kale, R. M. Wahul, A. D. Patange, R. Soman, and W. Ostachowicz, "Development of deep belief network for tool faults recognition," *Sensors*, vol. 23, no. 4, 2023, Art. no. 1872. doi: 10.3390/s23041872.

[147] E. Sansano, R. Montoliu, and O. Belmonte Fernandez, "A study of deep neural networks for human activity recognition," *Comput. Intell.*, vol. 36, no. 3, pp. 1113–1139, 2020. doi: 10.1111/coin.12318.

[148] A. Vaswani *et al.*, "Attention is all you need," *Adv Neural Inf. Process. Syst.*, vol. 30, pp. 5998–6008, 2017.

[149] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[150] K. Gavrilyuk, R. Sanford, M. Javan, and C. G. Snoek, "Actor-transformers for group activity recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2020, pp. 839–848.

[151] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1–28, 2022. doi: 10.24963/ijcai.2023/764.

[152] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *ACM Comput. Surv.*, vol. 55, no. 12, pp. 1–37, 2023. doi: 10.1145/3578938.

[153] Y. Liu and L. Wu, "Intrusion detection model based on improved transformer," *Appl. Sci.*, vol. 13, no. 10, 2023, Art. no. 6251. doi: 10.3390/app13106251.

[154] D. Chen, S. Yongchareon, E. M. K. Lai, J. Yu, Q. Z. Sheng and Y. Li, "Transformer with bidirectional GRU for nonintrusive, sensor-based activity recognition in a multiresident environment," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23716–23727, 2022. doi: 10.1109/JIOT.2022.3190307.

[155] J. Devlin, M. -W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[156] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. Accessed: Aug. 10, 2024. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[157] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019, Art. no. 9.

[158] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," 2019, *arXiv:1901.02860*.

[159] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 5754–5764, 2019.

[160] N. Shazeer, "Fast transformer decoding: One write-head is all you need," 2019, *arXiv:1911.02150*.

[161] Y. -H. H. Tsai, S. Bai, P. P. Liang, J. Z. Kolter, L. -P. Morency and R. Salakhutdinov, "Multimodal transformer for unaligned multimodal language sequences," in *Proc. Conf. Assoc. Comput. Linguist. Meet.*, NIH Public Access, 2019, vol. 2019, Art. no. 6558.

[162] A. Dosovitskiy *et al.*, "An image is worth $16 \times 16$ words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[163] W. Wang *et al.*, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2021, pp. 568–578.

[164] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10012–10022.

[165] L. Yuan *et al.*, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 558–567.

[166] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu and Y. Wang, "Transformer in transformer," *Adv. Neural. Inf. Process. Syst.*, vol. 34, pp. 15908–15919, 2021.

[167] K. Han, J. Guo, Y. Tang, and Y. Wang, "Pyramidtnt: Improved transformer-in-transformer baselines with pyramid architecture," 2022, *arXiv:2201.00978*.

[168] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, no. 120, pp. 1–39, 2022.

[169] Z. Liu, H. Mao, C. -Y. Wu, C. Feichtenhofer, T. Darrell and S. Xie, "A convnet for the, 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11976–11986.

[170] J. Zhang *et al.*, "EATFormer: Improving vision transformer inspired by evolutionary algorithm," *Int. J. Comput. Vis.*, pp. 1–28, 2024. doi: 10.1007/s11263-024-02034-6.

[171] N. Vithayathil Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, 2020, Art. no. 1363. doi: 10.3390/electronics9091363.

[172] N. Le, V. S. Rathour, K. Yamazaki, K. Luu, and M. Savvides, "Deep reinforcement learning in computer vision: A comprehensive survey," *Artif. Intell. Rev.*, pp. 1–87, 2022. doi: 10.1007/s10462-021-10061-9.

[173] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken: John Wiley & Sons, 2014.

[174] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE J. Power Energy Syst.*, vol. 6, no. 1, pp. 213–225, 2019. doi: 10.17775/CSEE-JPES.2019.00920.

[175] S. E. Li, "Deep reinforcement learning," in *Reinforcement Learning for Sequential Decision and Optimal Control*. Singapore: Springer, 2023, pp. 365–402.

[176] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. doi: 10.1038/nature14236.

[177] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, 2016. doi: 10.1609/aaai.v30i1.10295.

[178] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Int. Conf. Mach. Learn.*, PMLR, 2016, pp. 1995–2003.

[179] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *Comput. Gam.: 5th Int. Conf.*, Turin, Italy, 2007, pp. 72–83.

[180] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Trans. Games.*, vol. 12, no. 1, pp. 1–20, 2019. doi: 10.1109/TG.2019.2896986.

[181] K. Souchleris, G. K. Sidiropoulos, and G. A. Papakostas, "Reinforcement learning in game industry—Review, prospects and challenges," *Appl. Sci.*, vol. 13, no. 4, 2023, Art. no. 2443. doi: 10.3390/app13042443.

[182] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE Int. Conf. Robot. Autom. (ICRA)*, IEEE, 2017, pp. 3389–3396. doi: 10.1109/ICRA.2017.7989385.

[183] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, "A survey on deep reinforcement learning algorithms for robotic manipulation," *Sensors*, vol. 23, no. 7, 2023, Art. no. 3762. doi: 10.3390/s23073762.

[184] K. M. Lee, H. Myeong, and G. Song, "SeedNet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation," in *IEEE/CVF Conf. Comput. Vis. Pattern. Recognit. (CVPR)*, Salt Lake City, UT, USA, IEEE Computer Society, Jun. 18–23, 2018, pp. 1760–1768. doi: 10.1109/cvpr.2018.00189.

[185] H. Allioui *et al.*, "A multi-agent deep reinforcement learning approach for enhancement of COVID-19 CT image segmentation," *J. Pers. Med.*, vol. 12, no. 2, 2022, Art. no. 309. doi: 10.3390/jpm12020309.

[186] F. Sahba, "Deep reinforcement learning for object segmentation in video sequences," in *2016 Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vegas, NV, USA, IEEE, Dec. 15–17, 2016, pp. 857–860. doi: 10.1109/CSCI.2016.0166.

[187] H. Liu *et al.*, "Learning to identify critical states for reinforcement learning from videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2023, pp. 1955–1965.

[188] A. Shojaeighadikolaei, A. Ghasemi, A. G. Bardas, R. Ahmadi, and M. Hashemi, "Weather-aware data-driven microgrid energy management using deep reinforcement learning," in *2021 North. American. Power. Symp. (NAPS)*, College Station, TX, USA, IEEE, Nov. 14–16, 2021, pp. 1–6. doi: 10.1109/NAPS52732.2021.9654550.

[189] B. Zhang, W. Hu, A. M. Ghias, X. Xu, and Z. Chen, "Multi-agent deep reinforcement learning based distributed control architecture for interconnected multi-energy microgrid energy management and optimization," *Energy Conv. Manag.*, vol. 277, 2023, Art. no. 116647. doi: 10.1016/j.enconman.2022.116647.

[190] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 2208–2217.

[191] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu, "A survey on deep transfer learning," in *Artif. Neural NET. Mach. Learn. ICANN 2018: 27th Int. Conf. Artif. Neural NET*, Rhodes, Greece, Springer, Oct. 4–7, 2018, pp. 270–279. doi: 10.1007/978-3-030-01424-7_27.

[192] F. Zhuang *et al.*, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, 2020. doi: 10.1109/JPROC.2020.3004555.

[193] M. K. Rusia and D. K. Singh, "A color-texture-based deep neural network technique to detect face spoofing attacks," *Cybern. Inf. Technol.*, vol. 22, no. 3, pp. 127–145, 2022. doi: 10.2478/cait-2022-0032.

[194] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources," in *2010 IEEE Comput. Conf. Comput. socy. Vis. Pattern. Recognit.*, San Francisco, CA, USA, IEEE, Jun. 13–18, 2010, pp. 1855–1862. doi: 10.1109/CVPR.2010.5539857.

[195] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 863–870.

[196] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," 2014, *arXiv:1412.3474*.

[197] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Int. Conf. Mach. Learn.*, PMLR, 2015, pp. 97–105.

[198] M. Iman, H. R. Arabnia, and K. Rasheed, "A review of deep transfer learning and recent advancements," *Technol*, vol. 11, no. 2, 2023, Art. no. 40. doi: 10.3390/technologies11020040.

[199] A. A. Rusu *et al.*, "Progressive neural networks," 2016, *arXiv:1606.04671*.

[200] Y. Guo, J. Zhang, B. Sun, and Y. Wang, "Adversarial deep transfer learning in fault diagnosis: Progress, challenges, and future prospects," *Sensors*, vol. 23, no. 16, 2023, Art. no. 7263. doi: 10.3390/s23167263.

[201] Y. Gulzar, "Fruit image classification model based on MobileNetV2 with deep transfer learning technique," *Sustainability*, vol. 15, no. 3, 2023, Art. no. 1906. doi: 10.3390/su15031906.

[202] N. Kumar, M. Gupta, D. Gupta, and S. Tiwari, "Novel deep transfer learning model for COVID-19 patient detection using X-ray chest images," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, no. 1, pp. 469–478, 2023. doi: 10.1007/s12652-021-03306-6.

[203] H. Kheddar, Y. Himeur, S. Al-Maadeed, A. Amira, and F. Bensaali, "Deep transfer learning for automatic speech recognition: Towards better generalization," *Knowl.-Based Syst.*, vol. 277, 2023, Art. no. 110851. doi: 10.1016/j.knosys.2023.110851.

[204] L. Yuan, T. Wang, G. Ferraro, H. Suominen, and M. -A. Rizoiu, "Transfer learning for hate speech detection in social media," *J. Comput. Soc. Sci.*, vol. 6, no. 2, pp. 1081–1101, 2023. doi: 10.1007/s42001-023-00224-9.

[205] A. Ray, M. H. Kolekar, R. Balasubramanian, and A. Hafiane, "Transfer learning enhanced vision-based human activity recognition: A decade-long analysis," *Int. J. Inf. Manag. Data Insights*, vol. 3, no. 1, 2023, Art. no. 100142. doi: 10.1016/j.jjimei.2022.100142.

[206] T. Kujani and V. D. Kumar, "Head movements for behavior recognition from real time video based on deep learning ConvNet transfer learning," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, no. 6, pp. 7047–7061, 2023. doi: 10.1007/s12652-021-03558-2.

[207] A. Maity, A. Pathak, and G. Saha, "Transfer learning based heart valve disease classification from Phonocardiogram signal," *Biomed Signal Process. Control.*, vol. 85, 2023, Art. no. 104805. doi: 10.1016/j.bspc.2023.104805.

[208] K. Rezaee, S. Savarkar, X. Yu, and J. Zhang, "A hybrid deep transfer learning-based approach for Parkinson's disease classification in surface electromyography signals," *Biomed Signal Process. Control.*, vol. 71, 2022, Art. no. 103161. doi: 10.1016/j.bspc.2021.103161.

[209] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2018, pp. 8697–8710.

[210] Y. Zhang et al., "Deep learning in food category recognition," *Inf Fusion*, vol. 98, 2023, Art. no. 101859. doi: 10.1016/j.inffus.2023.101859.

[211] E. Ramanujam and T. Perumal, "MLMO-HSM: Multi-label multi-output hybrid sequential model for multi-resident smart home activity recognition," *J. Ambient Intell. Hum. Comput.*, vol. 14, no. 3, pp. 2313–2325, 2023. doi: 10.1007/s12652-022-04487-4.

[212] M. Ren, X. Liu, Z. Yang, J. Zhang, Y. Guo and Y. Jia, "A novel forecasting based scheduling method for household energy management system based on deep reinforcement learning," *Sustain. Cities Soc.*, vol. 76, 2022, Art. no. 103207. doi: 10.1016/j.scs.2021.103207.

[213] S. M. Abdullah et al., "Optimizing traffic flow in smart cities: Soft GRU-based recurrent neural networks for enhanced congestion prediction using deep learning," *Sustainability*, vol. 15, no. 7, 2023, Art. no. 5949. doi: 10.3390/su15075949.

[214] M. I. B. Ahmed et al., "Deep learning approach to recyclable products classification: Towards sustainable waste management," *Sustainability*, vol. 15, no. 14, 2023, Art. no. 11138. doi: 10.3390/su151411138.

[215] C. Zeng, C. Ma, K. Wang, and Z. Cui, "Parking occupancy prediction method based on multi factors and stacked GRU-LSTM," *IEEE Access*, vol. 10, pp. 47361–47370, 2022. doi: 10.1109/ACCESS.2022.3171330.

[216] N. K. Mehta, S. S. Prasad, S. Saurav, R. Saini, and S. Singh, "Three-dimensional DenseNet self-attention neural network for automatic detection of student's engagement," *Appl. Intell.*, vol. 52, no. 12, pp. 13803–13823, 2022. doi: 10.1007/s10489-022-03200-4.

[217] A. K. Shukla, A. Shukla, and R. Singh, "Automatic attendance system based on CNN-LSTM and face recognition," *Int. J. Inf. Technol.*, vol. 16, no. 3, pp. 1293–1301, 2024. doi: 10.1007/s41870-023-01495-1.

[218] B. Rajalakshmi, V. K. Dandu, S. L. Tallapalli, and H. Karanwal, "ACE: Automated exam control and e-proctoring system using deep face recognition," in *2023 Int. Conf. Circuit. Power. Comput. Technol. (ICCPCT)*, Kollam, India, IEEE, Aug. 10–11, 2023, pp. 301–306. doi: 10.1109/ICCPCT58313.2023.10245126.

[219] I. Pacal, "MaxCerVixT: A novel lightweight vision transformer-based approach for precise cervical cancer detection," *Knowl.-Based Syst.*, vol. 289, 2024, Art. no. 111482. doi: 10.1016/j.knosys.2024.111482.

[220] M. M. Rana *et al.*, "A robust and clinically applicable deep learning model for early detection of Alzheimer's," *IET Image Process.*, vol. 17, no. 14, pp. 3959–3975, 2023. doi: 10.1049/ipr2.12910.

[221] S. Vimal, Y. H. Robinson, S. Kadry, H. V. Long, and Y. Nam, "IoT based smart health monitoring with CNN using edge computing," *J. Internet Technol.*, vol. 22, no. 1, pp. 173–185, 2021. doi: 10.3966/160792642021012201017.

[222] T. S. Johnson *et al.*, "Diagnostic Evidence GAuge of Single cells (DEGAS): A flexible deep transfer learning framework for prioritizing cells in relation to disease," *Genome Med.*, vol. 14, no. 1, 2022, Art. no. 11. doi: 10.1186/s13073-022-01012-2.

[223] W. Zheng, S. Lu, Z. Cai, R. Wang, L. Wang and L. Yin, "PAL-BERT: An improved question answering model," *Comput. Model. Eng. Sci.*, pp. 1–10, 2023. doi: 10.32604/cmes.2023.046692.

[224] F. Wang *et al.*, "TEDT: Transformer-based encoding-decoding translation network for multimodal sentiment analysis," *Cogn. Comput.*, vol. 15, no. 1, pp. 289–303, 2023. doi: 10.1007/s12559-022-10073-9.

[225] M. Nafees Muneera and P. Sriramya, "An enhanced optimized abstractive text summarization traditional approach employing multi-layered attentional stacked LSTM with the attention RNN," in *Comput. Vis. Mach. Intell. Paradigm.*, Springer, 2023, pp. 303–318. doi: 10.1007/978-981-19-7169-3_28.

[226] M. A. Uddin, M. S. Uddin Chowdury, M. U. Khandaker, N. Tamam, and A. Sulieman, "The efficacy of deep learning-based mixed model for speech emotion recognition," *Comput. Mater. Contin.*, vol. 74, no. 1, pp. 1709–1722, 2023. doi: 10.32604/cmc.2023.031177.

[227] M. De Silva and D. Brown, "Multispectral plant disease detection with vision transformer-convolutional neural network hybrid approaches," *Sensors*, vol. 23, no. 20, 2023, Art. no. 8531. doi: 10.3390/s23208531.

[228] T. Akilan and K. Baalamurugan, "Automated weather forecasting and field monitoring using GRU-CNN model along with IoT to support precision agriculture," *Expert Syst. Appl.*, vol. 249, 2024, Art. no. 123468. doi: 10.1016/j.eswa.2024.123468.

[229] R. Benameur, A. Dahane, B. Kechar, and A. E. H. Benyamina, "An innovative smart and sustainable low-cost irrigation system for anomaly detection using deep learning," *Sensors*, vol. 24, no. 4, 2024, Art. no. 1162. doi: 10.3390/s24041162.

[230] M. Hosseinpour-Zarnaq, M. Omid, F. Sarmadian, and H. Ghasemi-Mobtaker, "A CNN model for predicting soil properties using VIS-NIR spectral data," *Environ. Earth. Sci.*, vol. 82, no. 16, 2023, Art. no. 382. doi: 10.1007/s12665-023-11073-0.

[231] M. Shakeel, K. Itoyama, K. Nishida, and K. Nakadai, "Detecting earthquakes: A novel deep learning-based approach for effective disaster response," *Appl. Intell.*, vol. 51, no. 11, pp. 8305–8315, 2021. doi: 10.1007/s10489-021-02285-7.

[232] Y. Zhang, Z. Zhou, J. Van Griensven Thé, S. X. Yang, and B. Gharabaghi, "Flood forecasting using hybrid LSTM and GRU models with lag time preprocessing," *Water*, vol. 15, no. 22, 2023, Art. no. 3982. doi: 10.3390/w15223982.

[233] H. Xu and H. Wu, "Accurate tsunami wave prediction using long short-term memory based neural networks," *Ocean Model.*, vol. 186, 2023, Art. no. 102259. doi: 10.1016/j.ocemod.2023.102259.

[234] J. Yao, B. Zhang, C. Li, D. Hong, and J. Chanussot, "Extended vision transformer (ExViT) for land use and land cover classification: A multimodal deep learning framework," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, pp. 1–15, 2023. doi: 10.1109/TGRS.2023.3284671.

[235] A. Y. Cho, S. -E. Park, D. -J. Kim, J. Kim, C. Li and J. Song, "Burned area mapping using Unitemporal Planetscope imagery with a deep learning based approach," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 16, pp. 242–253, 2022. doi: 10.1109/JSTARS.2022.3225070.

[236] M. Alshehri, A. Ouadou, and G. J. Scott, "Deep transformer-based network deforestation detection in the Brazilian amazon using Sentinel-2 imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 21, pp. 1–5, 2024. doi: 10.1109/LGRS.2024.3355104.

[237] V. Hnamte and J. Hussain, "DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system," *Telemat. Inform. Rep.*, vol. 10, 2023, Art. no. 100053. doi: 10.1016/j.teler.2023.100053.

[238] E. S. Alomari *et al.*, "Malware detection using deep learning and correlation-based feature selection," *Symmetry*, vol. 15, no. 1, 2023, Art. no. 123. doi: 10.3390/sym15010123.

[239] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem and M. H. Faheem, "A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN," *Electronics*, vol. 12, no. 1, 2023, Art. no. 232. doi: 10.3390/electronics12010232.

[240] H. Fanai and H. Abbasimehr, "A novel combined approach based on deep Autoencoder and deep classifiers for credit card fraud detection," *Expert. Syst. Appl.*, vol. 217, 2023, Art. no. 119562. doi: 10.1016/j.eswa.2023.119562.

[241] R. A. Joshi and N. Sambre, "Personalized CNN architecture for advanced multi-modal biometric authentication," in *2024 Int. Conf. Invent. Comput. Technol. (ICICT)*, Lalitpur, Nepal, IEEE, Apr. 24–26, 2024, pp. 890–894. doi: 10.1109/ICICT60155.2024.10544987.

[242] J. Sohafi-Bonab, M. H. Aghdam, and K. Majidzadeh, "DCARS: Deep context-aware recommendation system based on session latent context," *Appl. Soft Comput.*, vol. 143, 2023, Art. no. 110416. doi: 10.1016/j.asoc.2023.110416.

[243] J. Duan, P. -F. Zhang, R. Qiu, and Z. Huang, "Long short-term enhanced memory for sequential recommendation," *World Wide Web*, vol. 26, no. 2, pp. 561–583, 2023. doi: 10.1007/s11280-022-01056-9.

[244] P. Mondal, D. Chakder, S. Raj, S. Saha, and N. Onoe, "Graph convolutional neural network for multimodal movie recommendation," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, 2023, pp. 1633–1640. doi: 10.1145/3555776.3577853.

[245] Z. Liu, "Prediction model of E-commerce users' purchase behavior based on deep learning," *Front Bus. Econ. Manag.*, vol. 15, no. 2, pp. 147–149, 2024. doi: 10.54097/p22ags78.

[246] S. Deng, R. Li, Y. Jin, and H. He, "CNN-based feature cross and classifier for loan default prediction," in *2020 Int. Conf. Image. Video. Process. Artif. Intell.*, SPIE, 2020, vol. 11584, pp. 368–373.

[247] C. Han and X. Fu, "Challenge and opportunity: Deep learning-based stock price prediction by using Bi-directional LSTM model," *Front. Bus. Econ. Manag.*, vol. 8, no. 2, pp. 51–54, 2023. doi: 10.54097/fbem.v8i2.6616.

[248] Y. Cao, C. Li, Y. Peng, and H. Ru, "MCS-YOLO: A multiscale object detection method for autonomous driving road environment recognition," *IEEE Access*, vol. 11, pp. 22342–22354, 2023. doi: 10.1109/AC-CESS.2023.3252021.

[249] D. K. Jain, X. Zhao, G. González-Almagro, C. Gan, and K. Kotecha, "Multimodal pedestrian detection using metaheuristics with deep convolutional neural network in crowded scenes," *Inf. Fusion*, vol. 95, pp. 401–414, 2023. doi: 10.1016/j.inffus.2023.02.014.

[250] S. Sindhu and M. Saravanan, "An optimised extreme learning machine (OELM) for simultaneous localisation and mapping in autonomous vehicles," *Int. J. Syst. Syst. Eng.*, vol. 13, no. 2, pp. 140–159, 2023. doi: 10.1504/IJSSE.2023.131231.

[251] G. Singal, H. Singhal, R. Kushwaha, V. Veeramsetty, T. Badal and S. Lamba, "RoadWay: Lane detection for autonomous driving vehicles via deep learning," *Multimed. Tools Appl.*, vol. 82, no. 4, pp. 4965–4978, 2023. doi: 10.1007/s11042-022-12171-0.

[252] H. Shang, C. Sun, J. Liu, X. Chen, and R. Yan, "Defect-aware transformer network for intelligent visual surface defect detection," *Adv Eng. Inform.*, vol. 55, 2023, Art. no. 101882. doi: 10.1016/j.aei.2023.101882.

[253] T. Zonta, C. A. Da Costa, F. A. Zeiser, G. de Oliveira Ramos, R. Kunst and R. da Rosa Righi, "A predictive maintenance model for optimizing production schedule using deep neural networks," *J. Manuf. Syst.*, vol. 62, pp. 450–462, 2022. doi: 10.1016/j.jmsy.2021.12.013.

[254] Z. He, K. -P. Tran, S. Thomassey, X. Zeng, J. Xu and C. Yi, "A deep reinforcement learning based multi-criteria decision support system for optimizing textile chemical process," *Comput. Ind.*, vol. 125, 2021, Art. no. 103373. doi: 10.1016/j.compind.2020.103373.

[255] M. Pacella and G. Papadia, "Evaluation of deep learning with long short-term memory networks for time series forecasting in supply chain management," *Proc. CIRP*, vol. 99, pp. 604–609, 2021. doi: 10.1016/j.procir.2021.03.081.

[256] P. Shukla, H. Kumar, and G. C. Nandi, "Robotic grasp manipulation using evolutionary computing and deep reinforcement learning," *Intell Serv. Robot.*, vol. 14, no. 1, pp. 61–77, 2021. doi: 10.1007/s11370-020-00342-7.

[257]  K. Kamali, I. A. Bonev, and C. Desrosiers, "Real-time motion planning for robotic teleoperation using dynamic-goal deep reinforcement learning," in *2020 17th Conf. Comput. Robot. Vis. (CRV)*, IEEE, May 13–15, 2020, pp. 182–189. doi: 10.1109/CRV50864.2020.00032.

[258]  J. Zhang, H. Liu, Q. Chang, L. Wang, and R. X. Gao, "Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly," *CIRP Annals*, vol. 69, no. 1, pp. 9–12, 2020. doi: 10.1016/j.cirp.2020.04.077.

[259]  B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLoS One*, vol. 16, no. 7, 2021, Art. no. e0254841. doi: 10.1371/journal.pone.0254841.

[260]  C. Khosla and B. S. Saini, "Enhancing performance of deep learning models with different data augmentation techniques: A survey," in *2020 Int. Conf. Intell. Eng. Mgmt. (ICIEM)*, London, UK, IEEE, Jun. 17–19, 2020, pp. 79–85. doi: 10.1109/ICIEM48762.2020.9160048.

[261]  M. Paschali, W. Simson, A. G. Roy, R. Göbl, C. Wachinger and N. Navab, "Manifold exploring data augmentation with geometric transformations for increased performance and robustness," in *Inf. Process. Medical. Image.: 26th Int. Conf., IPMI*, Hong Kong, China, Springer, Jun. 2–7, 2019, pp. 517–529.

[262]  H. Guo, Y. Mao, and R. Zhang, "Augmenting data with mixup for sentence classification: An empirical study," 2019, *arXiv:1905.08941*.

[263]  O. O. Abayomi-Alli, R. Damaševičius, A. Qazi, M. Adedoyin-Olowe, and S. Misra, "Data augmentation and deep learning methods in sound classification: A systematic review," *Electronics*, vol. 11, no. 22, 2022, Art. no. 3795. doi: 10.3390/electronics11223795.

[264]  T. -H. Cheung and D. -Y. Yeung, "MODALS: Modality-agnostic automated data augmentation in the latent space," in *Int. Conf. Learn. Represent.*, 2020.

[265]  C. Shorten, T. M. Khoshgoftaar, and B. Furht, "Text data augmentation for deep learning," *J. Big Data*, vol. 8, no. 1, 2021, Art. no. 101. doi: 10.1186/s40537-021-00492-0.

[266]  F. Wang, H. Wang, H. Wang, G. Li, and G. Situ, "Learning from simulation: An end-to-end deep-learning approach for computational ghost imaging," *Opt. Express.*, vol. 27, no. 18, pp. 25560–25572, 2019. doi: 10.1364/OE.27.025560.

[267]  K. Ghosh, C. Bellinger, R. Corizzo, P. Branco, B. Krawczyk and N. Japkowicz, "The class imbalance problem in deep learning," *Mach. Learn.*, vol. 113, no. 7, pp. 4845–4901, 2024. doi: 10.1007/s10994-022-06268-8.

[268]  D. Singh, E. Merdivan, J. Kropf, and A. Holzinger, "Class imbalance in multi-resident activity recognition: An evaluative study on explainability of deep learning approaches," *Univ. Access. Inf. Soc.*, pp. 1–19, 2024. doi: 10.1007/s10209-024-01123-0.

[269]  A. S. Tarawneh, A. B. Hassanat, G. A. Altarawneh, and A. Almuhaimeed, "Stop oversampling for class imbalance learning: A review," *IEEE Access*, vol. 10, pp. 47643–47660, 2022. doi: 10.1109/AC-CESS.2022.3169512.

[270]  N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002. doi: 10.1613/jair.953.

[271]  H. Han, W. -Y. Wang, and B. -H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Int. Conf. Intell. Comput.*, Springer, 2005, pp. 878–887.

[272]  H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 Int. Jt. Conf. Neural. Netw.*, IEEE, 2008, pp. 1322–1328.

[273]  Y. Tang, Y. -Q. Zhang, N. V. Chawla, and S. Krasser, "SVMs modeling for highly imbalanced classification," *IEEE Trans. Syst. Man. Cybern. Part B (Cybern.)*, vol. 39, no. 1, pp. 281–288, 2008. doi: 10.1109/TSMCB.2008.2002909.

[274]  S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–Majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 405–425, 2012. doi: 10.1109/TKDE.2012.232.

[275]  C. Bellinger, S. Sharma, N. Japkowicz, and O. R. Zaïane, "Framework for extreme imbalance classification: SWIM—Sampling with the majority class," *Knowl. Inf. Syst.*, vol. 62, pp. 841–866, 2020. doi: 10.1007/s10115-019-01380-z.

[276] R. Das, S. K. Biswas, D. Devi, and B. Sarma, "An oversampling technique by integrating reverse nearest neighbor in SMOTE: Reverse-SMOTE," in *2020 Int. Conf. Smart. Electron. Commun. (ICOSEC)*, IEEE, 2020, pp. 1239–1244.

[277] C. Liu *et al.*, "Constrained oversampling: An oversampling approach to reduce noise generation in imbalanced datasets with class overlapping," *IEEE Access*, vol. 10, pp. 91452–91465, 2020. doi: 10.1109/ACCESS.2020.3018911.

[278] A. S. Tarawneh, A. B. Hassanat, K. Almohammadi, D. Chetverikov, and C. Bellinger, "SMOTEFUNA: Synthetic minority over-sampling technique based on furthest neighbour algorithm," *IEEE Access*, vol. 8, pp. 59069–59082, 2020. doi: 10.1109/ACCESS.2020.2983003.

[279] X. -Y. Liu, J. Wu, and Z. -H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst. Man. Cybern. Part B (Cybern.)*, vol. 39, no. 2, pp. 539–550, 2008. doi: 10.1109/TSMCB.2008.2007853.

[280] M. A. Tahir, J. Kittler, and F. Yan, "Inverse random under sampling for class imbalance problem and its application to multi-label classification," *Pattern Recognit.*, vol. 45, no. 10, pp. 3738–3750, 2012. doi: 10.1016/j.patcog.2012.03.014.

[281] V. Babar and R. Ade, "A novel approach for handling imbalanced data in medical diagnosis using undersampling technique," *Commun. Appl. Electron.*, vol. 5, no. 7, pp. 36–42, 2016. doi: 10.5120/cae2016652323.

[282] Z. H. Zhou and X. Y. Liu, "On multi-class cost-sensitive learning," *Comput. Intell.*, vol. 26, no. 3, pp. 232–257, 2010. doi: 10.1111/j.1467-8640.2010.00358.x.

[283] C. X. Ling and V. S. Sheng, "Cost-sensitive learning and the class imbalance problem," *Ency. Mach. Learn.*, vol. 2011, pp. 231–235, 2008.

[284] N. Seliya, A. Abdollah Zadeh, and T. M. Khoshgoftaar, "A literature review on one-class classification and its potential applications in big data," *J. Big Data*, vol. 8, pp. 1–31, 2021. doi: 10.1186/s40537-021-00514-x.

[285] V. S. Spelmen and R. Porkodi, "A review on handling imbalanced data," in *Int. Conf. Curr. Trend. Toward. Converg. Technol. (ICCTCT)*, Coimbatore, India, IEEE, Mar. 1–3, 2018, pp. 1–11. doi: 10.1109/ICCTCT.2018.8551020.

[286] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," 2018, *arXiv:1810.12281*.

[287] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *2016 IEEE Int. Conf. Acoust. Speech. Signal. Process. (ICASSP)*, Shanghai, China, IEEE, Mar. 20–25, 2016, pp. 2657–2661. doi: 10.1109/ICASSP.2016.7472159.

[288] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Int. Conf. Mach. Learn.*, PMLR, 2015, pp. 448–456.

[289] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," 2017, *arXiv:1701.06548*.

[290] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *IEEE Int. Conf. Acoust. Speech. Signal. Process.*, IEEE, 2013, pp. 8609–8613.

[291] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13 Int. Conf. Artif. Intell. Stats.*, PMLR, 2010, pp. 249–256.

[292] G. Srivastava, S. Vashisth, I. Dhall, and S. Saraswat, "Behavior analysis of a deep feedforward neural network by varying the weight initialization methods," in *Smart Innovations in Communication and Computational Sciences*. Singapore: Springer, 2021, pp. 167–175. doi: 10.1007/978-981-15-5345-5_15.

[293] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 4548–4557.

[294] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Natl. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017. doi: 10.1073/pnas.1611835114.

[295] S. -W. Lee, J. -H. Kim, J. Jun, J. -W. Ha, and B. -T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 4655–4665, 2017.

[296] S. -A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2017, pp. 2001–2010.

[297] A. D 'Amour *et al.*, "Underspecification presents challenges for credibility in modern machine learning," *J. Mach. Learn. Res.*, vol. 23, no. 226, pp. 1–61, 2022.

[298] D. Teney, M. Peyrard, and E. Abbasnejad, "Predicting is not understanding: Recognizing and addressing underspecification in machine learning," in *Europ. Conf. Comput. Vis.*, Springer, 2022, pp. 458–476.

[299] N. Chotisarn, W. Pimanmassuriya, and S. Gulyanon, "Deep learning visualization for underspecification analysis in product design matching model development," *IEEE Access*, vol. 9, pp. 108049–108061, 2021. doi: 10.1109/ACCESS.2021.3102174.

[300] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annual. Meet. Assoc. Comput. Linguist.: Hum. langu. Tech.*, Portland OR, USA, Jun. 19–2, 2011, pp. 142–150.

[301] H. Alemdar, H. Ertan, O. D. Incel, and C. Ersoy, "ARAS human activity datasets in multiple homes with multiple residents," in *2013 7th Int. Conf. Perv. Comput. Technol. Healthc. Workshop*, IEEE, 2013, pp. 232–235.

[302] H. Muresan and M. Oltean, "Fruit recognition from images using deep learning," *Acta U. Sapien. Inform.*, vol. 10, no. 1, pp. 26–42, 2018. doi: 10.2478/ausi-2018-0002.

[303] X. Xiao, M. Yan, S. Basodi, C. Ji, and Y. Pan, "Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm," 2020, *arXiv:2006.12703*.

[304] H. J. Escalante, M. Montes, and L. E. Sucar, "Particle swarm model selection," *J. Mach. Learn. Res.*, vol. 10, no. 2, pp. 405–440, 2009.

[305] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[306] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, 2nd ed. Berlin, Heidelberg: Springer, 2012, pp. 421–436.

[307] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 2121–2159, 2011.

[308] T. Dozat, "Incorporating nesterov momentum into adam," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR) Workshop Track*, San Juan, Puerto Rico, 2016, pp. 1–4.

[309] X. Chen *et al.*, "Symbolic discovery of optimization algorithms," *Adv Neural Inf. Process. Syst.*, vol. 36, pp. 49205–49233, 2024.

[310] L. Alzubaidi *et al.*, "A survey on deep learning tools dealing with data scarcity: Definitions, challenges, solutions, tips, and applications," *J. Big Data*, vol. 10, no. 1, 2023, Art. no. 46. doi: 10.1186/s40537-023-00727-2.

[311] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nat. Phys.*, vol. 15, no. 12, pp. 1273–1278, 2019. doi: 10.1038/s41567-019-0648-8.

[312] Y. Takaki, K. Mitarai, M. Negoro, K. Fujii, and M. Kitagawa, "Learning temporal data with a variational quantum recurrent neural network," *Phys. Rev. A*, vol. 103, no. 5, 2021, Art. no. 052414. doi: 10.1103/PhysRevA.103.052414.

[313] S. Lloyd and C. Weedbrook, "Quantum generative adversarial learning," *Phys. Rev. Lett.*, vol. 121, no. 4, 2018, Art. no. 040502. doi: 10.1103/PhysRevLett.121.040502.

[314] S. Garg and G. Ramakrishnan, "Advances in quantum deep learning: An overview," 2020, *arXiv:2005.04316*.

[315] F. Valdez and P. Melin, "A review on quantum computing and deep learning algorithms and their applications," *Soft Comput.*, vol. 27, no. 18, pp. 13217–13236, 2023. doi: 10.1007/s00500-022-07037-4.