A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU

FARHAD MORTEZAPOUR SHIRI[1], University Putra Malaysia (UPM), Malaysia

THINAGARAN PERUMAL, University Putra Malaysia (UPM), Malaysia

NORWATI MUSTAPHA, University Putra Malaysia (UPM), Malaysia

RAIHANI MOHAMED, University Putra Malaysia (UPM), Malaysia

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 Networks (CNNs), Recurrent Neural Networks (RNNs), 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 compare the performance of six renowned deep learning models: CNN, Simple RNN, Long Short-Term Memory (LSTM), Bidirectional LSTM, Gated Recurrent Unit (GRU), and Bidirectional GRU.

Keywords: Deep learning, Machine Learning, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Generative Models, Autoencoder (AE), Generative adversarial network (GAN), Deep Reinforcement Learning (DRL), Deep Transfer Learning.

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.

Deep learning (DL) is a subset of machine learning that excels at processing unstructured data. Currently, deep learning methods outperform traditional machine learning approaches [3]. 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 [4]. Figure 1 illustrates the relationship between AI, ML, and DL, highlighting that machine learning and deep learning are subfields of artificial intelligence.

The objective of this research is to provide an overview of various deep learning models and compare their performance across different applications. Section 2 discusses the different deep learning models, including Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Generative Models, Deep Reinforcement Learning (DRL), and Deep Transfer Learning. In Section 3, we conduct experiments and analyze six deep learning models, namely Convolutional Neural Networks (CNN), Simple Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Bidirectional LSTM, Gated Recurrent Unit (GRU),

[1] Morteza.pour@student.upm.edu.my

and Bidirectional GRU, using three datasets. Finally, Section 4 concludes the paper.

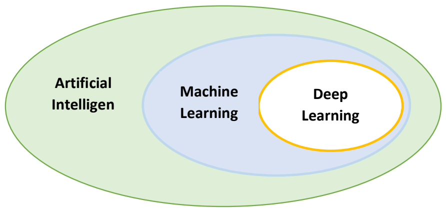

Figure 1. Relationship between Artificial Intelligence, Machine Learning, and Deep Learning.

2. Deep learning Models

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 [5]. 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 [3].

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 [6]. Figure 2 visually illustrates the distinction between deep learning and traditional machine learning in terms of feature extraction and learning.

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) [7]. Additionally, generative approaches encompass various models such as Generative Adversarial Networks (GANs) and Auto-Encoders (AEs). In the following sections, we provide a comprehensive survey of different types of deep learning models.

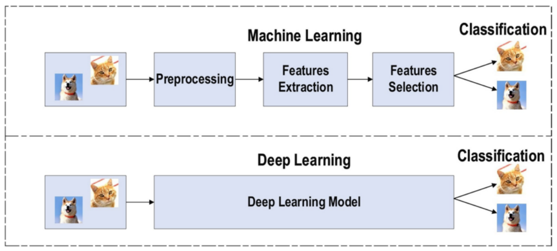

Figure 2. Visual illustration of the distinction between deep learning and traditional machine learning in terms of feature extraction and learning [6].

2.1. 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) [7]. 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. 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 [8]. Figure 3 depicts a single-neuron perceptron model, where the activation function φ (Equation 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 Equation 1, the terms x, w, b, and y represent the input vector, weighting vector, bias, and output value, respectively [9]. Figure 4 illustrates the structure of the multi layers perceptron (MLP) model.

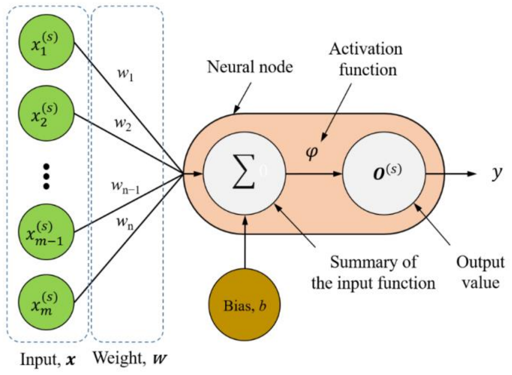

Figure 3. Single-neuron perceptron model [9].

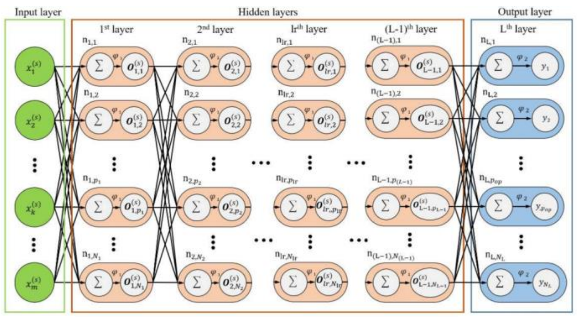
Figure 4. Structure of the multilayer perceptron (MLP) [9].

2.2. 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 [10]. They have also demonstrated success in time series prediction tasks [11]. CNNs are feedforward neural networks that leverage convolutional structures to extract features from data [12]. Unlike traditional methods, CNNs automatically learn and recognize features from the data without the need for manual feature extraction by humans [13]. The design of CNNs is inspired by visual perception [12]. The major components of CNNs include the convolutional layer, pooling layer, and fully connected layer [14]. Figure 5 presents a typical CNN architecture for image classification tasks [15].

Convolutional Layer: The convolutional layer is a pivotal component of CNNs. 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 [16]. The convolutional layer comprises learnable convolution kernels, which are weight matrices typically of equal length, width, and an odd number (e.g., 3x3, 5x5, or 7x7). These kernels are convolved with the input feature maps, sliding over the regions of the feature map and executing convolution operations [16]. Figure 6 illustrates the schematic diagram of the convolution process.

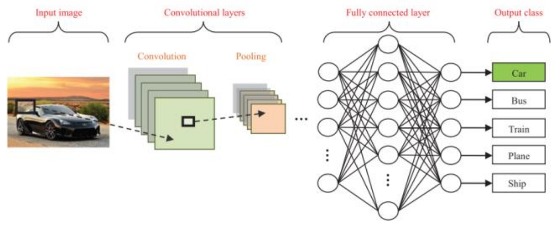

Figure 5. CNN pipeline for image classification [15].

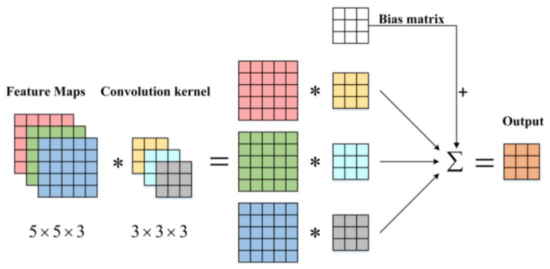

Figure 6. Schematic diagram of the convolution process [16].

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 [17]. Its primary purpose is to alleviate the computational burden and address overfitting issues [18]. Moreover, the pooling layer enables CNNs to recognize objects even when their shapes are distorted or viewed from different angles, by incorporating various dimensions of an image through pooling [19]. The pooling operation produces output feature maps that are more robust against distortion and errors in individual neurons [20]. There are various pooling methods, including Max Pooling, Average Pooling, Spatial Pyramid Pooling, Mixed Pooling, Multi-Scale Order-Less, and Stochastic Pooling [21-24]. Figure 7 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 [25].

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 [6].

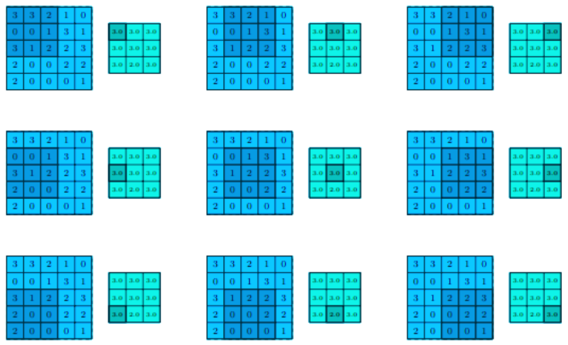
Figure 7. Computing the output values of a 3 × 3 max pooling operation on a 5 × 5 input [25].

2.3. 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 [26]. 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 [27].

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[28-30].

However, a limitation of simple RNNs is their short-term memory, which restricts their ability to retain information over long sequences [31]. To overcome this, more advanced RNN variants have been developed, including Long Short-Term Memory (LSTM) [32], bidirectional LSTM [33], Gated Recurrent Unit (GRU) [34], bidirectional GRU [35], Bayesian RNN [36], and others.

Figure 8 depicts a simple recurrent neural network, where the internal memory (h_t) is computed using Equation (2) [37]:

$$h_t = g(Wx_t + Uh_t + b) \qquad (2)$$

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

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.

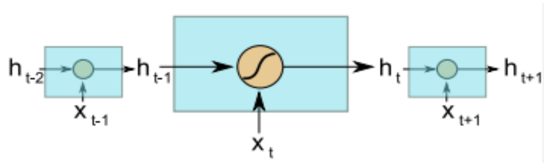

Figure 8. Simple RNN internal operation [37].

2.3.1. 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 [32] in 1997 and further improved by [38] in 2013, gaining significant popularity in the deep learning community. Compared to standard RNNs, LSTM models have proven to be more effective at retaining and utilizing information over longer sequences [39].

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 [40]. The overall architecture of an LSTM network is depicted in Figure 9.

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 [27]. Figure 10 illustrates the update mechanism within the inner structure of an LSTM.

The update equations for the LSTM unit are expressed by Equation 3:

$$\begin{aligned}
&h^{(t)} = g_0^{(t)} f_h\left(s^{(t)}\right) \\
&s^{(t-1)} = g_f^{(t)} s^{(t-1)} + g_i^{(t)} f_s\left(wh^{(t-1)}\right) + uX^{(t)} + b \\
&g_i^{(t)} = sigmoid\ (w_i h^{(t-1)} + u_i X^{(t)} + b_i \qquad (3) \\
&g_f^{(t)} = sigmoid\ (w_f h^{(t-1)} + u_f X^{(t)} + b_f \\
&g_o^{(t)} = sigmoid\ (w_o h^{(t-1)} + u_o X^{(t)} + b_o
\end{aligned}$$

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.

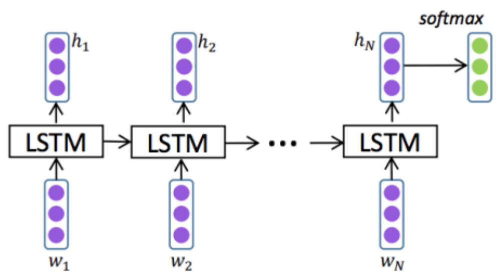

Figure 9. The high- level architecture of LSTM model

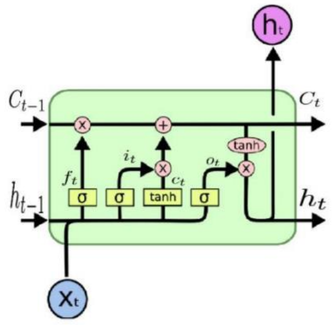

Figure 10. The inner architecture of a standard LSTM module [27].

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 [41]. 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 [42].

2.3.2. 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 [43]. 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 Figure 11. Simultaneously, the backward LSTM layer reads the input data from right to left, as represented by the red arrow [44]. This bidirectional processing enables the model to capture information from both past and future context, 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 [44]. 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.

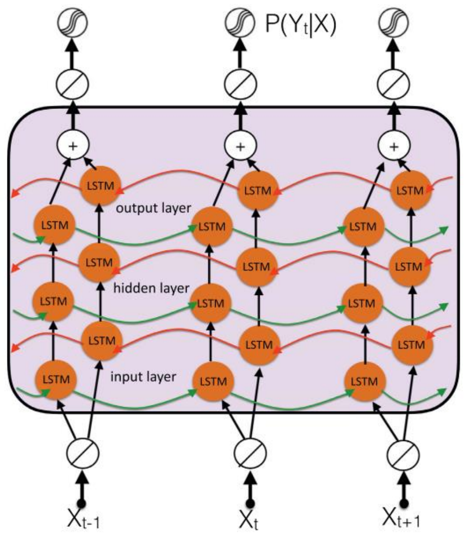

Figure 11. The architecture of a Bidirectional LSTM model [44].

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 understanding of both past and future context.

2.3.3.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 [26]. 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 [45]. Figure 12 illustrates the structure of a GRU unit [46].

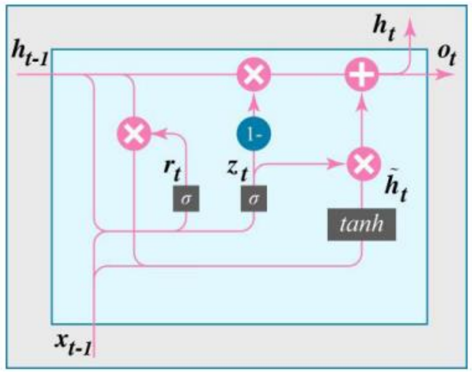

Figure 12. The structure of a GRU unit [46].

The update gate (equation 4) 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.

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{4}$$

The reset gate (equation 5) 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.

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{5}$$

The current memory content (equation 6) 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.

$$\tilde{h}_t = tanh(W_h[r_t h_{t-1}, x_t]) \tag{6}$$

Finally, the final memory state h_t is determined by a combination of the previous hidden state and the candidate activation (equation 7). 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 (equation 8). 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.

$$h_t = (1 - z_t)h_{t-1} + z_t \tilde{h}_t \tag{7}$$

$$o_t = \sigma_o(W_o h_t + b_o) \tag{8}$$

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 [26].

2.4.Generative 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 [47].

Various generative models have been developed and applied in different contexts, such as Auto-Encoder [48], Generative Adversarial Network (GAN) [49], Restricted Boltzmann Machine (RBM) [50], Deep Belief Network (DBN) [51], and Self-Organizing Map (SOM) [52].

2.4.1. 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 [48].

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 $r = g(z)$ [53]. 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 [54]. Figure 13 illustrates an autoencoder model. Autoencoder models find utility in various unsupervised learning tasks, such as generative modeling [55], dimensionality reduction [56], feature extraction [57], anomaly or outlier detection [58], and denoising [59].

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 [60], which can function as generative models. Examples of regularized autoencoder models include Sparse Autoencoder (SAE) [61], Contractive Autoencoder (CAE) [62], and Denoising Autoencoder (DAE) [63].

Variational Autoencoder (VAE) is a generative model that employs probabilistic distributions, such as the mean and variance of a Gaussian distribution, for data generation [48]. VAEs 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 [64].

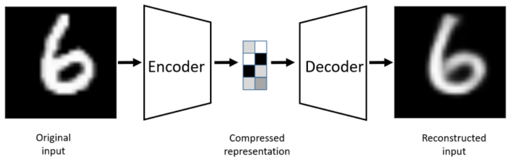

Figure 13. The structure of autoencoders [48].

2.4.2. 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 Ian Goodfellow [49]. A GAN consists of two key components: a generative model and a discriminative model. The generative model aims to generate images that resemble real ones, while the discriminative model aims to differentiate between real and synthetic images. Both models are typically implemented using multilayer perceptrons [65]. Figure 14 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 [66].

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 [67]:

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.

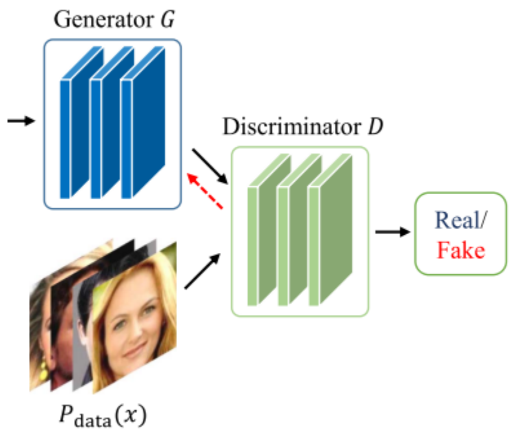

Figure 14. The framework of a GRU [66].

Generative Adversarial Networks (GANs) have introduced numerous applications across various domains, including image blending [68], 3D object generation [69], face aging [70], medicine [71, 72], steganography [73], image manipulation [74], text transfer [75], language and speech synthesis [76], traffic control [77], and video generation [78].

Furthermore, several models have been developed based on the Generative Adversarial Network (GAN) framework to address specific tasks. These models include Coupled GAN (Co-GAN) [65], Markovian GAN [79], Evolutionary GAN (E-GAN) [66], Unrolled GAN [80], Bayesian Conditional GAN [81], Relativistic GAN [82], Laplacian GAN (Lap-GAN) [83], Graph Embedding GAN (GE-GAN) [77], Wasserstein GAN (WGAN) [84], and Boundary Equilibrium GAN (BEGAN) [85].

2.5. 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 [86]. 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 [87]. Figure 15 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 [88].

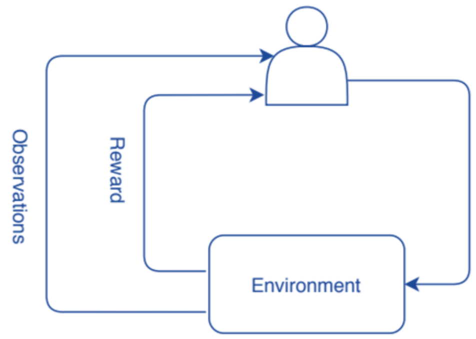

Figure 15. Agent-Environment interaction in RL [88].

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. Figure 16 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 [89].

One of the most renowned deep reinforcements learning models is the Deep Q-Learning Network (DQN) [90], which directly learns policies from high-dimensional inputs using convolutional neural networks (CNNs). Other common models in deep reinforcement learning include Double DQN [91], Dueling DQN [92], and Monte Carlo tree search (MCTS) [93].

Deep reinforcement learning (DRL) models find applications in various domains, such as video game playing [94], robotic manipulation [95], image segmentation [96], video analysis [97], energy management [98], and more.

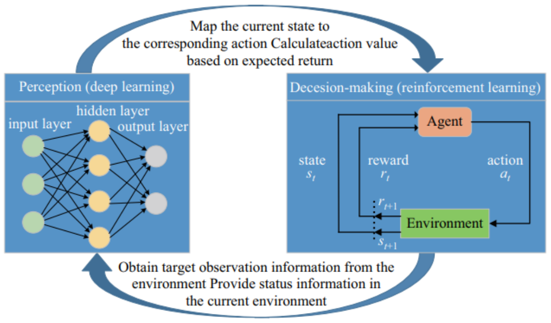
Map the current state to the corresponding action Calculateaction value based on expected return
Perception (deep learning)
hidden layer
input layer
output layer
Decesion-making (reinforcement learning)
Agent
state s_t
reward r_t
action a_t
r_{t+1}
Environment
s_{t+1}
Obtain target observation information from the environment Provide status information in the current environment

Figure 16. Basic structure of DRL [89].

2.6. 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 [99]. 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 [100].

To address the issue of limited training data, transfer learning 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. In order 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 [101]. Figure 17 depicts the 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 [100]: instances-based deep transfer learning, mapping-based deep transfer learning, network-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 [102] and TrAdaBoost.R2 [103] 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 [104] and MK-MMD [105].

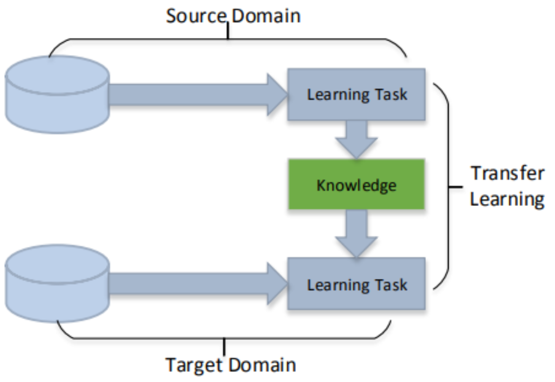

Figure 17. Learning process of transfer learning [100].

Network-based deep transfer learning revolves around reusing a portion of a network that has already been trained on the source domain, along with its network structure and connection parameters, and transferring it to a deep neural network employed in the target domain.

Adversarial-based deep transfer learning aims to identify transferable representations using generative adversarial networks (GANs) as models that are applicable to both the source and target domains.

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 machine learning applications.

3. Analysis of deep learning models

In our experimental analysis, we utilized three publicly available datasets: IMDB [106], ARAS [107], and Fruit-360 [108]. The purpose was to perform a comparative analysis of various deep learning models. Specifically, we examined six different models: CNN, Simple RNN, LSTM, Bidirectional LSTM, GRU, and Bidirectional GRU. By evaluating these models on the testing data, we aimed to assess their performance using multiple evaluation metrics such as accuracy, precision, recall, and F1-measure.

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 sensors events for human activity recognition tasks. Fruit-360 is a dataset consisting of images of various fruit types for classification purposes.

Our analysis focused on comparing the performance of the different deep learning models across these diverse datasets. The CNN model, known as Convolutional Neural Network, is particularly effective in image analysis tasks due to its ability to capture spatial dependencies. Simple RNN (Recurrent Neural Network) is suitable for sequential data analysis, while LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models excel in capturing long-term dependencies in sequential data. Bidirectional LSTM and Bidirectional GRU models offer the advantage of processing information in both forward and backward directions.

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.

$$Accuracy = \frac{Tp + Tn}{Tp + Tn + Fp + Fn} \tag{9}$$

$$Precision = \frac{Tp}{Tp + Fp} \tag{10}$$

$$Recall = \frac{Tp}{Tp + Fn} \tag{11}$$

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{12}$$

Where Tp = True Positive, Tn = True Negative, Fp = False Positive, and Fn = False Negative [39].

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.

3.1. Analysis of models 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 correlation between reviews for a given movie, only 30 reviews are included in the dataset [106]. 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 [40].

In our analysis, we employed six different deep learning models for sentiment classification using the IMDB dataset. 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.

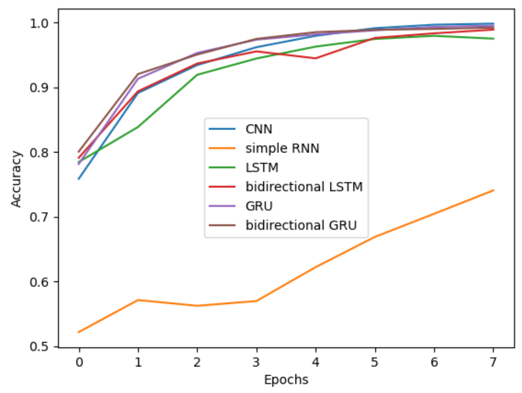

Figure 18. Accuracy of deep learning models on IMDB dataset

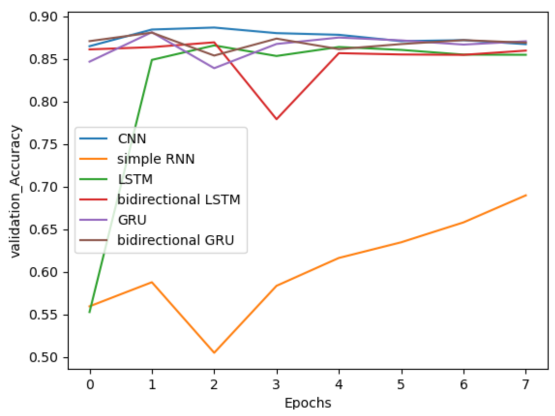

Figure 19. validation- accuracy of deep learning models on IMDB dataset

Figure 18, the accuracy diagram, provides a visual representation of how the different deep learning models perform in terms of accuracy during the training process. Also, figure 19, the validation-accuracy shows the trend of accuracy values on the validation set across multiple epochs for each model.

Figure 20 illustrates the loss diagram as a visual representation of loss values during the training process for six different models. Figure 21, 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.

In these experiments, the parameters used for all models were set to epochs = 8 and batch-size = 64. The CNN model utilized 200 filters with a kernel size of 3 and the rectified linear unit (ReLU) activation function. For the simple RNN, LSTM, and GRU models, 128 units were employed, along with a 20% dropout rate and recurrent dropout. On the other hand, the Bi-LSTM and Bi-GRU models used 64 units. These parameter settings and architectural choices allow for a standardized comparison of the deep learning models on the IMDB dataset, facilitating the analysis of their performance and comparison of their accuracy and loss values.

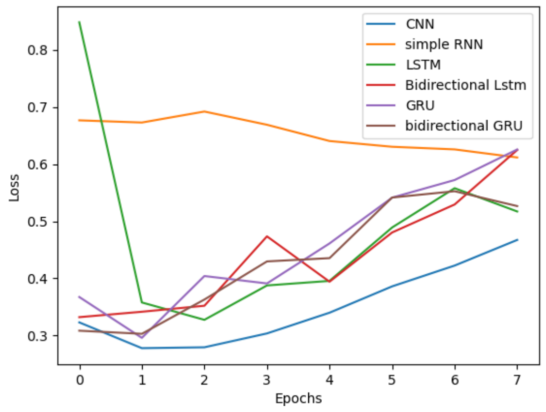

Figure 20. Loss of deep learning models on IMDB dataset

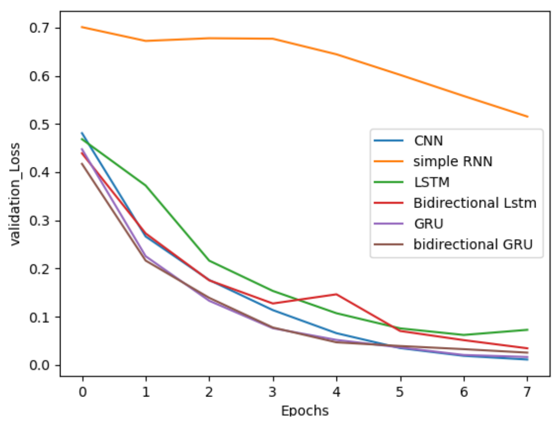

Figure 21. validation- loss of deep learning models on IMDB dataset

Also, table 1 shows the result of different deep learning models on IMDB review dataset base on various metrices including Accuracy, precision, Recall, F1-Score, and time of training on CPU.

Table 1. Result of different deep learning models on IMDB dataset

model	Accuracy	Precision	Recall	F1-Score	time
CNN	86.69	87.01	86.12	86.61	0:01:59
RNN	68.97	74.72	77.14	75.96	0:16:56
LSTM	85.46	85.70	84.32	85.04	1:45:44
Bi-LSTM	85.95	87.21	83.74	85.43	0:27:35
GRU	87.04	84.92	88.98	86.97	1:29:25
BI-GRU	86.88	90.53	83.59	86.69	0:15:03

Based on the results provided, it can be concluded that the GRU and CNN 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 CNN model was significantly less than that of the GRU model. This suggests that the CNN model was faster to train compared to the GRU model while still achieving excellent performance. Additionally, the Bi-GRU model also exhibited good accuracy in sentiment classification and required less training time than the GRU model. This indicates that the Bi-GRU model could be a favorable choice, offering a balance between accuracy and training efficiency. Overall, the results suggest that the GRU, CNN, and Bi-GRU models are effective deep learning models for sentiment analysis on the IMDB review dataset, with varying trade-offs between performance and training time.

3.2.Analysis of models on ARAS dataset

Based on the provided information, the ARAS dataset [107] is a valuable resource for automatic recognition of 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 the residents, and the sensor events are recorded on a per-second basis.

In order to analyze and classify the human activities in the ARAS dataset, six deep learning models were employed. Training data from 10 days of house B, comprising 846,000 samples, were used to train the models, while 4 days of data, amounting to 338,400 samples, were used for testing.

Figure 22 presents the accuracy diagram for the deep learning models, while figure 23 illustrates the validation accuracy diagram. Also, figure 24 shows the loss diagram and figure 25 provides a validation-loss diagram for deep learning models.

The models were trained with a fixed set of parameters: epochs=8 and batch size=256. The CNN model utilized 32 filters with a kernel size of 7, an activation function of 'Relu', and a pooling size of 2. The Simple RNN, LSTM, Bi-LSTM, GRU, and Bi-GRU models were configured with 256 units, along with 20% dropout and recurrent dropout.

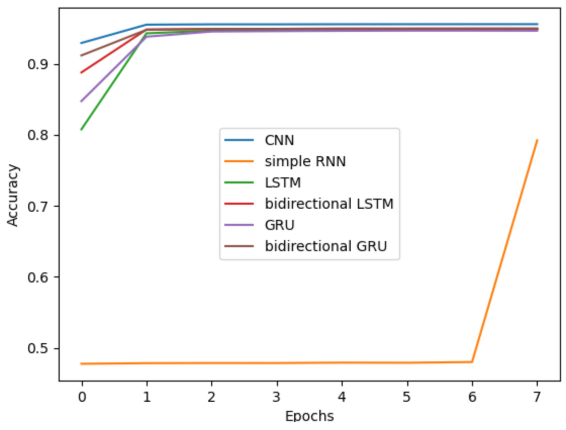

Figure 22. Accuracy of deep learning models on ARAS dataset

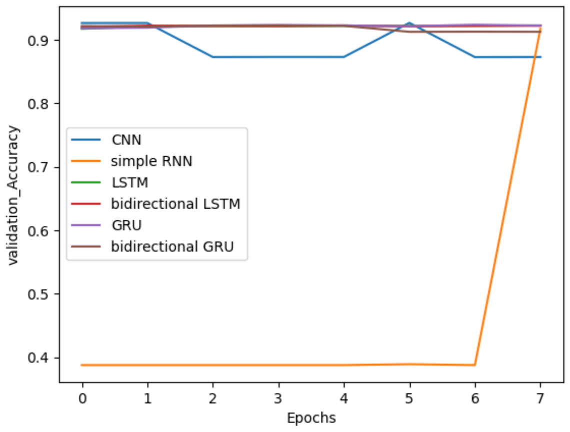

Figure 23. validation- accuracy of deep learning models on ARAS dataset

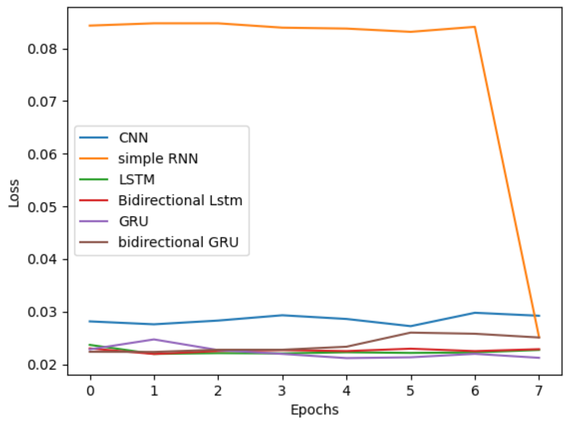

Figure 24. Loss diagram of deep learning models on ARAS dataset

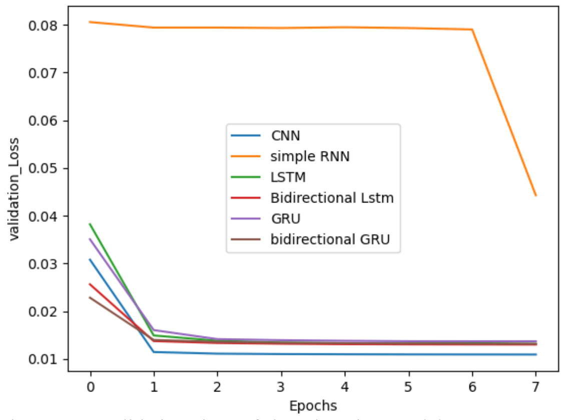

Figure 25. validation- loss of deep learning models on ARAS dataset

Also, table 2 illustrates the results of experiments on ARAS dataset with various metrices including Accuracy, precision, Recall, F1-Score, and time of training on CPU.

Table 2. Result of deep learning models on ARAS dataset

model	Accuracy	Precision	Recall	F1-Score	time
CNN	89.18	93.78	88.69	91.16	0:02:18
RNN	91.73	93.56	90.74	92.13	0:10:00
LSTM	92.27	93.97	91.75	92.85	1:03:27
Bi-LSTM	92.21	93.91	91.78	92.83	1:54:52
GRU	92.24	94.00	91.91	92.94	0:36:13
BI-GRU	91.26	92.99	90.74	91.85	0:48:37

Based on the provided results, it can be observed that the recurrent models, specifically LSTM, Bi-LSTM, and GRU, outperformed the convolutional model (CNN) when working with time-series data from the ARAS dataset. This finding is consistent with the nature of the dataset, which involves temporal sequences of sensor events. Among the recurrent models, GRU demonstrated the best performance in terms of accuracy or other evaluation metrics. Additionally, it had the advantage of having a lower training time compared to the other recurrent models, indicating its efficiency in processing, and learning from the time-series data. These results suggest that for the task of human activity recognition in smart environments using the ARAS dataset, recurrent models like LSTM, Bi-LSTM, and GRU are well-suited due to their ability to capture temporal dependencies. Among these models, GRU stands out as a particularly efficient option, achieving good performance while requiring less training time.

3.3.Analysis of models on Fruit-360 dataset

The analysis of deep learning models on the Fruit-360 dataset for image classification task involved three models: CNN, LSTM, and Bi-LSTM. The dataset consists of 55,244 images of 81 different fruit classes, each with a resolution of 100x100 pixels. For this experiment, a subset of 20 fruit classes was used, with 9,722 images for training and 3,261 images for testing.

Figure 26 shows the accuracy diagram of the three deep learning models, while figure 27 illustrates the validation-accuracy of these models. Also, figure 28 illustrates the loss diagram and figure 29 providing a validation-loss diagram of the three deep learning models.

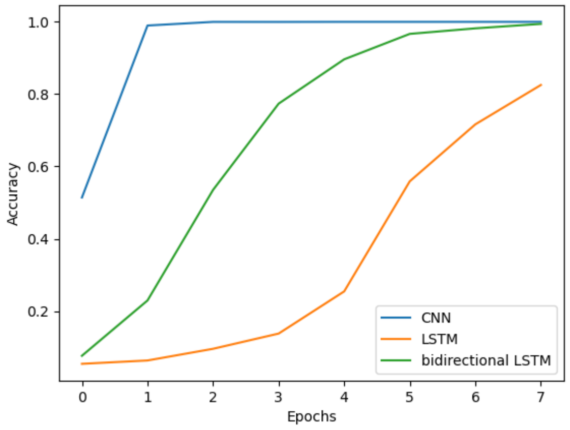

Figure 26. Accuracy of CNN, LSTM, Bi-LSTM models on Friut-360 dataset.

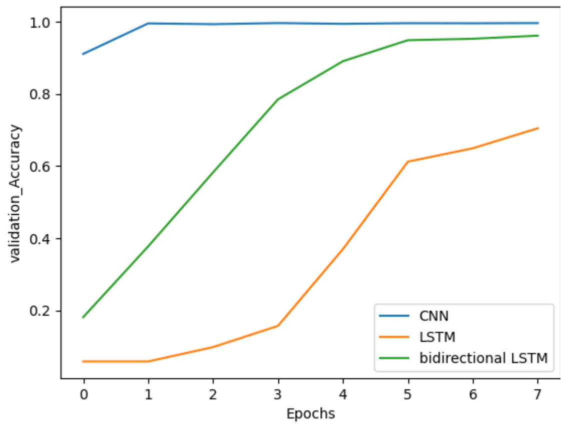

Figure 27. Validation- accuracy of CNN, LSTM, Bi-LSTM models on Friut-360 dataset.

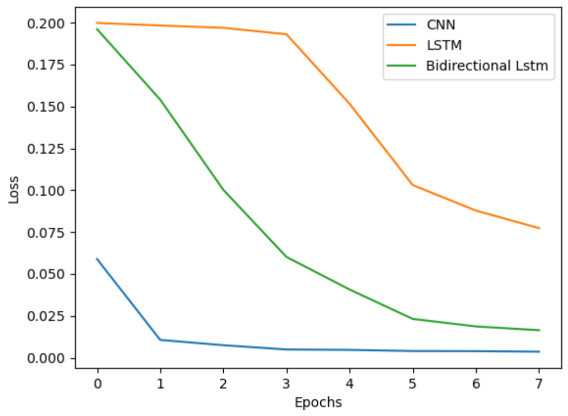

Figure 28. Loss diagram of CNN, LSTM, Bi-LSTM models on Friut-360 dataset.

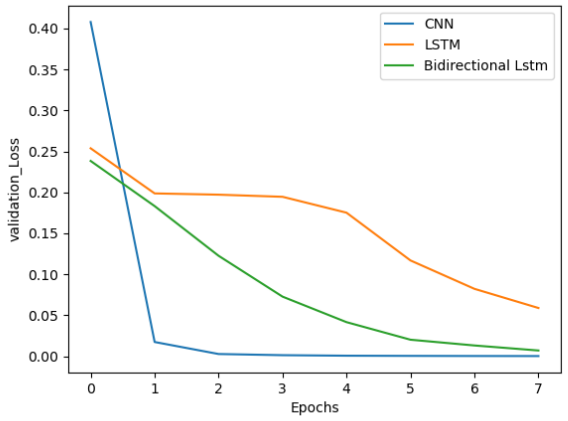

Figure 29. Validation- loss of CNN, LSTM, Bi-LSTM models on Friut-360 dataset.

The experimental setup included a fixed number of epochs (8) and a batch size of 256 for all models. In the CNN model, 200 filters with a kernel size of 3x3 and an activation function of 'relu' were employed.

Furthermore, table 3 shows results of experiments based on different metrices including Accuracy, precision, Recall, F1-Score, and time of training on Fruit-360 dataset for CNN, LSTM, and Bi-LSTM models.

Table 3. Result of deep learning models on Fruit360 dataset

model	Accuracy	Precision	Recall	F1-Score	time
CNN	99.69	99.75	98.25	98.99	0:06:28
LSTM	70.49	84.08	59.46	69.60	0:38:32
Bi-LSTM	96.19	96.65	95.64	96.14	1:31:13

Based on the results, it appears that the CNN model outperformed the LSTM and Bi-LSTM models in terms of both accuracy and training time. The CNN model's superior performance can be attributed to its ability to effectively capture spatial features in images through convolutional layers. Since image data is not inherently sequential or time-dependent, the recurrent models may not have been as well-suited for this particular task. Furthermore, the faster training time of the CNN model can be attributed to its parallel processing nature, which allows for efficient computation on GPUs and accelerates the training process. Overall, the results suggest that for image classification tasks, such as the Fruit-360 dataset, where the focus is on capturing spatial features, CNN models tend to be more effective and efficient compared to recurrent models like LSTM and Bi-LSTM.

4. Conclusion

In 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 model, and different types of deep learning models such as CNN, RNN, 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 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 model performs exceptionally well in image classification tasks, while the RNN models, such as LSTM and GRU, show strong performance in time series analysis. Additionally, the GRU model stands out for its faster training time compared to LSTM, attributed to its simplified architecture with two gates instead of three.

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.

Reference

[1] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, 2018: IEEE, pp. 1-6.

[2] D. K. Rathore and P. K. Mannepalli, "A Review of Machine Learning Techniques and Applications for Health Care," in *2021 International Conference on Advances in Technology, Management & Education (ICATME)*, 2021: IEEE, pp. 4-8.

[3] A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: an overview," *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020,* pp. 599-608, 2021.

[4] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access,* vol. 7, pp. 53040-53065, 2019.

[5] M. A. Wani, F. A. Bhat, S. Afzal, and A. I. Khan, *Advances in deep learning*. Springer, 2020.

[6] L. Alzubaidi *et al.*, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *Journal of big Data,* vol. 8, pp. 1-74, 2021.

[7] I. H. Sarker, "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science,* vol. 2, no. 6, p. 420, 2021.

[8] 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.

[9] K.-C. Ke and M.-S. Huang, "Quality prediction for injection molding by using a multilayer perceptron neural network," *Polymers,* vol. 12, no. 8, p. 1812, 2020.

[10] A. Tasdelen and B. Sen, "A hybrid CNN-LSTM model for pre-miRNA classification," *Scientific reports,* vol. 11, no. 1, pp. 1-9, 2021.

[11] 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.

[12] 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.

[13] 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, p. 1685, 2021.

[14] W. Lu, J. Li, J. Wang, and L. Qin, "A CNN-BiLSTM-AM method for stock price prediction," *Neural Computing and Applications,* vol. 33, pp. 4741-4753, 2021.

[15] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation,* vol. 29, no. 9, pp. 2352-2449, 2017.

[16] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of image classification algorithms based on convolutional neural networks," *Remote Sensing,* vol. 13, no. 22, p. 4712, 2021.

[17] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition,* vol. 77, pp. 354-377, 2018.

[18] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, 2019, vol. 1168: IOP Publishing, p. 022022.

[19] A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," in *2020 international conference on emerging trends in information technology and engineering (ic-ETITE)*, 2020: IEEE, pp. 1-5.

[20] 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.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence,* vol. 37, no. 9, pp. 1904-1916, 2015.

[22] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Rough Sets and Knowledge Technology: 9th International Conference, RSKT 2014, Shanghai, China, October 24-26, 2014, Proceedings 9*, 2014: Springer, pp. 364-375.

[23] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*, 2014: Springer, pp. 392-407.

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

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

[26] 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, doi: 10.3390/s20195707.

[27] W. Fang, Y. Chen, and Q. Xue, "Survey on research of RNN-based spatio-temporal sequence prediction algorithms," *Journal on Big Data,* vol. 3, no. 3, p. 97, 2021.

[28] J. Xiao and Z. Zhou, "Research progress of RNN language model," in *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2020: IEEE, pp. 1285-1288.

[29] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694-4702.

[30] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU," *Journal of Artificial Intelligence and Soft Computing Research,* vol. 9, no. 4, pp. 235-245, 2019.

[31] 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, p. 1500, 2020.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[33] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE transactions on pattern analysis and machine intelligence,* vol. 31, no. 5, pp. 855-868, 2008.

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

[35] 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.

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

[37] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger, "Rainfall–runoff modelling using Long Short-Term<? xmltex\break?> Memory (LSTM) networks," *Hydrology and Earth System Sciences,* vol. 22, no. 11, pp. 6005-6022, 2018.

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

[39] F. Shiri, T. Perumal, N. Mustapha, R. Mohamed, M. A. B. Ahmadon, and S. Yamaguchi, "A Survey on Multi-Resident Activity Recognition in Smart Environments," 2023.

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

[41] X. Zhu, P. Sobihani, and H. Guo, "Long short-term memory over recursive structures," in *International conference on machine learning*, 2015: PMLR, pp. 1604-1612.

[42] F. Gu, M.-H. Chung, M. Chignell, S. Valaee, B. Zhou, and X. Liu, "A survey on deep learning for human activity recognition," *ACM Computing Surveys (CSUR),* vol. 54, no. 8, pp. 1-34, 2021.

[43] 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, p. 1118, 2021.

[44] 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.

[45] A. Dutta, S. Kumar, and M. Basu, "A gated recurrent unit approach to bitcoin price prediction," *Journal of Risk and Financial Management,* vol. 13, no. 2, p. 23, 2020.

[46] 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.

[47] A. Jabbar, X. Li, and B. Omar, "A survey on generative adversarial networks: Variants, applications, and training," *ACM Computing Surveys (CSUR),* vol. 54, no. 8, pp. 1-49, 2021.

[48] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991,* 2020.

[49] I. Goodfellow *et al.*, "Generative adversarial networks," *Communications of the ACM,* vol. 63, no. 11, pp. 139-144, 2020.

[50] N. Zhang, S. Ding, J. Zhang, and Y. Xue, "An overview on restricted Boltzmann machines," *Neurocomputing,* vol. 275, pp. 1186-1199, 2018.

[51] G. E. Hinton, "Deep belief networks," *Scholarpedia,* vol. 4, no. 5, p. 5947, 2009.

[52] T. Kohonen, *Self-organizing maps*. Springer Science & Business Media, 2012.

[53] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[54] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *2018 IEEE international conference on systems, man, and cybernetics (SMC)*, 2018: IEEE, pp. 415-419.

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

[56] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing,* vol. 184, pp. 232-242, 2016.

[57] Y. N. Kunang, S. Nurmaini, D. Stiawan, and A. Zarkasi, "Automatic features extraction using autoencoder in intrusion detection system," in *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 2018: IEEE, pp. 219-224.

[58] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 665-674.

[59] A. Creswell and A. A. Bharath, "Denoising adversarial autoencoders," *IEEE transactions on neural networks and learning systems,* vol. 30, no. 4, pp. 968-984, 2018.

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

[61] A. Ng, "Sparse autoencoder," *CS294A Lecture notes,* vol. 72, no. 2011, pp. 1-19, 2011.

[62] S. Rifai *et al.*, "Higher order contractive auto-encoder," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece,*

September 5-9, 2011, Proceedings, Part II 22, 2011: Springer, pp. 645-660.

[63] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096-1103.

[64] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning,* vol. 12, no. 4, pp. 307-392, 2019.

[65] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," *Advances in neural information processing systems,* vol. 29, 2016.

[66] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Transactions on Evolutionary Computation,* vol. 23, no. 6, pp. 921-934, 2019.

[67] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights,* vol. 1, no. 1, p. 100004, 2021.

[68] B.-C. Chen and A. Kae, "Toward realistic image compositing with adversarial learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8415-8424.

[69] D. P. Jaiswal, S. Kumar, and Y. Badr, "Towards an artificial intelligence aided design approach: application to anime faces with generative adversarial networks," *Procedia Computer Science,* vol. 168, pp. 57-64, 2020.

[70] Y. Liu, Q. Li, and Z. Sun, "Attribute-aware face aging with wavelet-based generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11877-11886.

[71] J. Islam and Y. Zhang, "GAN-based synthetic brain PET image generation," *Brain informatics,* vol. 7, pp. 1-12, 2020.

[72] 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,* p. 2020.06. 09.143297, 2020.

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

[74] S. Nam, Y. Kim, and S. J. Kim, "Text-adaptive generative adversarial networks: manipulating images with natural language," *Advances in neural information processing systems,* vol. 31, 2018.

[75] L. Sixt, B. Wild, and T. Landgraf, "Rendergan: Generating realistic labeled data," *Frontiers in Robotics and AI,* vol. 5, p. 66, 2018.

[76] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," *Advances in neural information processing systems,* vol. 30, 2017.

[77] D. Xu, C. Wei, P. Peng, Q. Xuan, and H. Guo, "GE-GAN: A novel deep learning framework for road traffic state estimation," *Transportation Research Part C: Emerging Technologies,* vol. 117, p. 102635, 2020.

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

[79] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, 2016: Springer, pp. 702-716.

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

[81] G. Zhao, M. E. Meyerand, and R. M. Birn, "Bayesian conditional GAN for MRI brain image synthesis," *arXiv preprint arXiv:2005.11875,* 2020.

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

[83] E. L. Denton, S. Chintala, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," *Advances in neural information processing systems,* vol. 28, 2015.

[84] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, 2017: PMLR, pp. 214-223.

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

[86] N. Vithayathil Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics,* vol. 9, no. 9, p. 1363, 2020.

[87] N. Le, V. S. Rathour, K. Yamazaki, K. Luu, and M. Savvides, "Deep reinforcement learning in computer vision: a comprehensive survey," *Artificial Intelligence Review,* pp. 1-87, 2022.

[88] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[89] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE Journal of Power and Energy Systems,* vol. 6, no. 1, pp. 213-225, 2019.

[90] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature,* vol. 518, no. 7540, pp. 529-533, 2015.

[91] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2016, vol. 30, no. 1.

[92] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, 2016: PMLR, pp. 1995-2003.

[93] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *Computers and Games: 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers 5*, 2007: Springer, pp. 72-83.

[94] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Transactions on Games,* vol. 12, no. 1, pp. 1-20, 2019.

[95] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*, 2017: IEEE, pp. 3389-3396.

[96] K. M. Lee, H. Myeong, and G. Song, "SeedNet: Automatic Seed Generation with Deep Reinforcement Learning for Robust Interactive Segmentation," presented at the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00189.

[97] F. Sahba, "Deep reinforcement learning for object segmentation in video sequences," in *2016 International conference on computational science and computational intelligence (CSCI)*, 2016: IEEE, pp. 857-860.

[98] 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 Symposium (NAPS)*, 2021: IEEE, pp. 1-6.

[99] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *International conference on machine learning*, 2017: PMLR, pp. 2208-2217.

[100] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, 2018: Springer, pp. 270-279.

[101] F. Zhuang *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE,* vol. 109, no. 1, pp. 43-76, 2020.

[102] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources," in *2010 IEEE computer society conference on computer vision and pattern recognition*, 2010: IEEE, pp. 1855-1862.

[103] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, pp. 863-870.

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

[105] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*, 2015: PMLR, pp. 97-105.

[106] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142-150.

[107] H. Alemdar, H. Ertan, O. D. Incel, and C. Ersoy, "ARAS human activity datasets in multiple homes with multiple residents," in *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, 2013: IEEE, pp. 232-235.

[108] H. Mureşan and M. Oltean, "Fruit recognition from images using deep learning," *arXiv preprint arXiv:1712.00580,* 2017.