Ingénierie des Systèmes d'Information
Vol. 30, No. 3, March, 2025, pp. 779-795
Journal homepage: http://iieta.org/journals/isi

A Survey of the Advances in the Applications of Deep Learning Algorithms Across Different Domains

Gabriel O. Sobola[1*], Samuel Daramola[1], Emmanuel Adetiba[1,2]

[1] Department of Electrical and Information Engineering, Covenant University, Ota 112104, Nigeria
[2] HRA, Institute for Systems Science, Durban University of Technology, Durban 4001, South Africa

Corresponding Author Email: gabriel.sobola@covenantuniversity.edu.ng

https://doi.org/10.18280/isi.300322

Received: 7 May 2024
Revised: 3 September 2024
Accepted: 13 February 2025
Available online: 31 March 2025

Keywords:
activation functions, artificial intelligence, convolutional neural network (CNN), deep learning, neural networks, neurons, recurrent neural network (RNN)

ABSTRACT

Deep learning has revolutionized the modern-day world starting with its application in computer vision such as image classification, face recognition, autonomous vehicle etc. it has been explored in various areas where human beings find it difficult to come up with solutions to the challenges at hand. By the word deep, it implies they are trained with millions, billions of parameters to achieve outstanding results. In this review paper, the fundamentals of deep learning have been discussed extensively starting with the classification, types of activation functions, different deep learning algorithms as well as their applications were also discussed. Recurrent neural network (RNNs) and its variant, convolution neural networks (CNNs) and various architectures, recursive neural networks (RvNNs), restricted Boltzmann machines (RBMs), deep belief networks (DBNs), generative adversarial networks (GANs) and other deep learning were discussed extensively. Some of the findings of researchers for some of these algorithms were highlighted. Based on various paper reviewed and thorough analysis carried out, it was observed that the exploration of deep learnings in this modern-day world has found applications in virtually all fields of life from medicine, academy, transportation, entertainments, particularly the exploration of CNNs, RNNs, and GANs.

1. INTRODUCTION

Artificial intelligence is a corner stone of the modern-day mega trends and technologies where it has facilitated the growth of various sectors from health to finance. It encompasses the deep learning which is a subset of machine learning [1]. The deep learning which is also called representative learning has been explored in various domains such as the natural language processing (NLP), Computer vision, speech recognition [2], etc.

The deep learning originated from human desire to develop a comprehensive system that could function as a human brain, hence the desire to understand the human cognitive system. Its historical development can be traced back to 300 BC during the era of Aristotle through whom Associationism (a theory that defined human mind as an organized set of conceptual elements) was introduced. He postulated four laws in relation to reminiscent, haven't been inspired Plato-these four laws are similarity, frequency, contrast, and contiguity. Similarity is the concept that the thought of one event is being triggered by a similar event's thought; frequency defines that the number of occurrences of two events is linked to their associations; contrast is the thought of one event being triggered by an opposite event's thought; and contiguity is the concept that there is an association in the mind of temporal or spatial events. In 1873, Alexander Bain introduced neural groupings, marking the earliest approach to neural networks. The McCulloch & Pitts (MCP) model, a linear predecessor of artificial neural networks, was introduced by McCulloch and Pitts in 1943. In 1949, Donald Hebb introduced the Hebbian Learning Rule, which serves as the foundation for modern neural networks; he is often called the father of neural networks, having postulated that "cells that fire together, wire together". Frank Rosenblatt introduced the first known perceptron in 1958. Other notable developments include: backpropagation by Paul Werbos in 1974; the Self-Organising Map and Neocogitron (the inspiration behind CNNs) by Teuvo Kohonen and Kunihiko Fukushima, respectively, in 1980; the Hopfield Network by John Hopfield in 1982; the Boltzmann Machine by Hinton and Sejnowski in 1985; the Harmonium (later known as the RBM) and RNN by Paul Smolensky and Michael I. Jordan, respectively, in 1986; LeNet—a deep learning-based model—by Yann LeCun in 1990; LSTM and bidirectional recurrent neural networks by Hochreiter & Schmidhuber and Schuster & Paliwal, respectively, in 1997; DBNs by Geoffrey Hinton in 2006; Deep Boltzmann Machines by Salakhutdinov & Hinton in 2009; and Dropout, a regularization technique, by Geoffrey Hinton in 2012. Other contributions include the integration of deep learning into ANN [3] and machine learning [4].

Various algorithms have been utilised for the achievements of various results in deep learnings. One of such algorithms is the CNNs which was originally designed and developed for image classification and accomplishments that seems to be

impossible with humans. It has been explored in autonomous vehicles, face recognition, intelligent medical treatment, self-service supermarket etc. In 1987, for speech recognition system, the time delay neural networks (TDNNs) which is a one-dimensional CNN was explored [5]. For CNNs, various models have been developed. These are the LeNets-5 for reading of bank checks and handwritten character recognition in 1996 [6], AlexNet in 2012 [7], ZFNet in 2013, GoogleNet, VGGNets in 2014 [8], ResNet for object detection and image classification in 2015 [9] SqueezeNet, DCGAN in 2016 [10, 11], ResNext, DenseNet, Xception, MobileNet v1 IN 2017 [12], ShuffleNet v2, MobileNet v2 in 2018 [13, 14], MobileNet v3 in 2019 [15], and GhostNet in 2020 [16]. Other available deep learning algorithms are the RNNs, BM, DBNs, LSTM, GRUs, SOM, RBF, that have also found applications in various works of life [17]. The RNNs have been explored on sequential data or time series data for various tasks such as weather forecast, text predictions, speech recognition.

Different classes of deep learning also exist [18]. These are the unsupervised deep learning, where only the input data is provided for trainings without the labeled data, supervised deep learning that makes provision for both the input data and target output in terms of labelled data. in such system, another output is generated whose efficiency is justified by comparing it with the target output, then the partially supervised learning where partial or weak supervision with the use of labelled data is provided for trainings. Then the final type is called the reinforcement deep learning that trains based on the experienced gathered by an agent where the agent is either rewarded or penalized. Besides these, deep learning can also be shallow or deep which depends on the number of hidden layers made available in the architecture. By shallow learning, it implies that there are lesser number of hidden layers while in deep learning, there are hundreds of hidden layers that are connected within the architecture. This paper addresses the state-of-the-art models and architectures developed using deep learning algorithms. Besides, the different types of deep learning as initially mentioned are highlighted to give readers great insight into the role they play in the deep learning domains and applications. Considering the aforementioned applications of deep learning, this review paper aims to give insight into the deep learning. Interested and enthusiastic researchers in the field of AI would find this paper of great help in terms of provisions of the fundamentals of technical words, terminologies, architectures, and applications in relations to deep learning.

The major contributions of this paper are as follows:

1. It serves as the material to give interested reader access to the information in relation to deep learning.
2. It presents insight into the different types of types of deep learning and their applications.
3. It gives comprehensive insight into the CNN architecture considering the fact that it is the first model that pave the way to deep learning.

2. FUNDAMENTALS OF DEEP LEARNING

2.1 Deep learning

Deep learning is a subset of machine learning that is explored in many applications. It is also called representation learning (RL). It has been explored in image processing, speech recognition, sentiment analysis, NLP, computer vision etc. Deep learning has been explored due to its universal learning approach, scalabilty, generalization, and robustness.

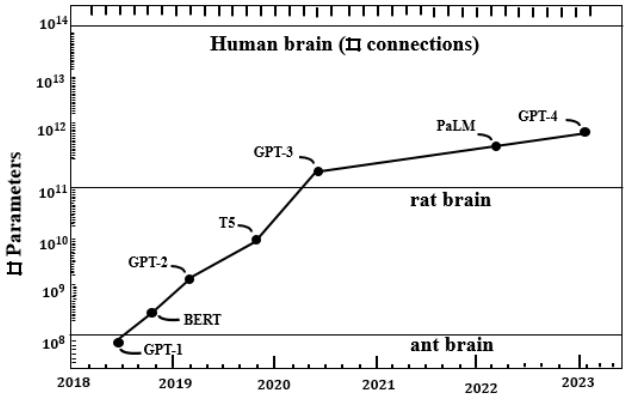

Figure 1. Comparison of the number of parameters in modern GenAIs, ant brain, rat brain and human brain

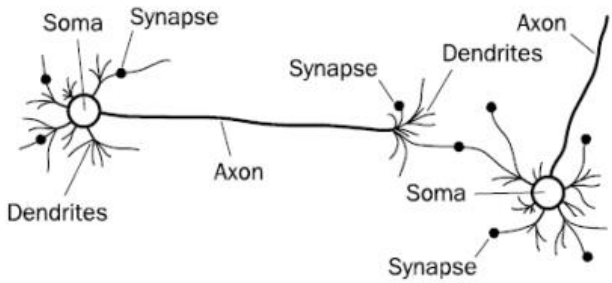

Figure 2. Simple biological neural network

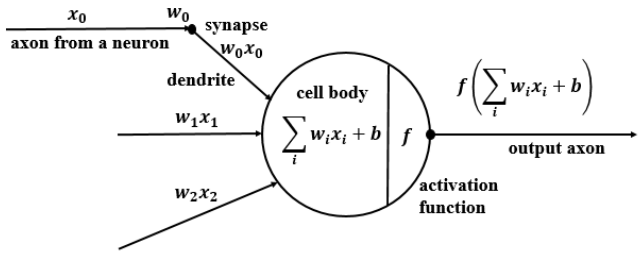

Figure 3. Artificial neuron

It is an integration of many artificial neural networks that learn from data to make predictions, recommendations, etc. Just like the biological human neurons that are billions in number that are wired for the processing of various tasks in human bodies, deep learning also has several billions of neurons that are combined and trained for various tasks as initially mentioned. Large language models (LLMs) like the ChatGPT were trained with 110 million parameters, GPT2 with 1.5 billion parameters, GPT3 with 175 billion parameters, and GPT4 is expected to have about 100 trillion parameters which is about 500 times more than that of GPT3 [19] This is further supported by the diagram in Figure 1. A closer look at the number of parameters used to train GPT4, one is expected to find difficulty in distinguishing the output of the model from human being. The human neurons and the artificial neural share similarity in their components. As illustrated in Figures 2 and 3, the cell body or soma is similar to neurons in human, dendrite to the input, axon to the output, and the synapse to the weight in artificial neurons.

The deep neural network became popular in the year 2011 with the use of new techniques, powerful computers, and huge dataset availability. In fact, the deep learning algorithm became popular due to the following three reasons: recent development and advances in deep learning, computer hardware affordability, and notable increase in the capabilities of its processing power [20]. One of the major challenges encountered in the traditional machine learning that involves the feature extraction was well catered for in the utilization of deep learning. It was also confirmed for most applications or problems where human beings might probably find it difficult to provide solutions, the deep learning could help out in such cases with better solutions provided. By the word deep learning, it means that the architecture is deep in the sense that there are many artificial neural networks that are aggregated in the system.

2.2 Artificial neural networks

This is a machine learning algorithm or model that follows the patterns of human brain neurons to train data in generating target output. It is a three-layered classifier [21] that consists of the input layer, hidden layer, and the output layer. The input, hidden, and output layer each has one or more neurons. Each neuron in the input layer can have a single output (perceptron), or more than one output, in the hidden layer each neuron has more than one output, while it is always expected of each neuron in the output layer to have a single output. The neurons are linked to each other via weights. The output of a preceding neuron is the input to the succeeding one. A neural network with an input layer with more than one neuron and an output layer with a single neuron is called a perceptron or a linear classifier [22]. A typical ANN consists of an input, hidden, and an output layers containing a number of neurons. The strength of the output of a neuron is measured by the weight, and to ensure the neuron keep firing at all time, bias is added. The final output is passed through an activation function to transformed the neuron output obtained.

2.3 Types of activation function

1. Linear (Identity) function

This is a type of activation function in which the output of the neuron. i.e. the sum of the weighted input is the output of the activation function. That is to say that the effect of a linear activation is of no effect. It can be represented in Eq. (1) as:

$$f(z) = z \tag{1}$$

where, z=sum of the weighted inputs.

2. Step function

This is type of activation function in which the output is either 0 or 1; or -1 or 1. That is, the output is always between two state values. It is of two types:

Binary step function. In this type, the output of the activation function is either 0 or 1. This is based on the threshold value set as illustrated in the piecewise relationship of Eq. (2):

$$f(z) = \begin{cases} 1, & if\ z \geq T \\ 0, & z < T \end{cases} \tag{2}$$

where, z=sum of the weighted inputs; T=threshold value set.

Bipolar step function. In bipolar, the activation is found such that its output is either a -1 or +1. This is illustrated in Eq. (3).

$$f(z) = \begin{cases} 1, & if\ z \geq T \\ -1, & z < T \end{cases} \tag{3}$$

3. Sigmoid function

These are S shaped function that makes the value of the activation function to vary between 0 and 1 for binary sigmoid function or between -1 and 1 for bipolar sigmoid function. They are of two types:

Binary sigmoid function. This is also called the logistic function. The activation function for the binary sigmoid function is represented by Eq. (4).

$$f(z) = \frac{1}{1 + e^{-\sigma z}} \tag{4}$$

where, σ=stepness parameter; z=sum of the weighted inputs.

Bipolar sigmoid function. This is a type of sigmoid function that finds application as activation function when the desired output range is between -1 and 1. It is presented in Eq. (5)

$$f(z) = \frac{1 - e^{-\sigma z}}{1 + e^{-\sigma z}} \tag{5}$$

4. Softmax activation function

This is a type of activation function that describes multiple sigmoid function. The output of this function varies between 0 and 1, and the sum of probabilities is equal to one. It is mostly utilised for multi-class classification where the probability of each output class is a fraction of the sum of probabilities of all other output classes, and the class with the highest probability is often taken as the target class. It represented in Eq. (6).

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad for\ j = 1, \ldots, K \tag{6}$$

5. Rectified linear unit (ReLU)

This was developed to overcome the varnishing gradient problems encountered by binary and bipolar sigmoid function. It invented by Nair, and Hinton. It is the most widely used activation function for deep learning. It rectified the output value z, to z if z is greater than or equal to 0 and 0 if z is less than 0. The piecewise representation is illustrated in Eq. (7).

$$f(z) = \max(0, z) = \begin{cases} z_i, & z_i \geq 0 \\ 0, & z_i < 0 \end{cases} \tag{7}$$

2.4 Perceptron

This is the simplest architecture representing the artificial neural network. It consists of the input layer, output layer, and hidden layer. The number of neurons in the input layer could be one or more than one. It has a single output with a single neuron. In a Perceptron with a single neuron at the input layer as shown in Figure 4, the output of this neuron is connected directly to the neuron at the output layer. In a Perceptron with more than one neuron at the input layer, each neuron output is connected to the input of the neuron at the output layer. Hence, a summation of these sets of neurons is carried out at the output layer. This is illustrated in Figure 5, while Figure 6 shows a typical ANN system consisting of networks of large numbers

of neurons. In Figure 5, each neuron in the input layer is linked to the single neuron in the output via a weight, then each input neuron, x_i is multiplied by their respective weight, w_i, and at the output neuron, the sum of all these weighted inputs is obtained and added to the bias value, b, this sum gives the initial output at the output layer. The activation function, f is applied on the output to obtain the final output y as given in Eq. (8) as:

$$y = f(b + \sum_{i=1}^{N} x_i w_i) \quad (8)$$

During training, the bias' value can be adjusted thereby shifting the activation function, to obtain a better accuracy of the model.

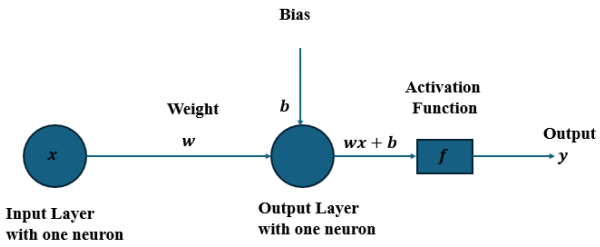

Figure 4. A perceptron with a single neuron at both input and output layer

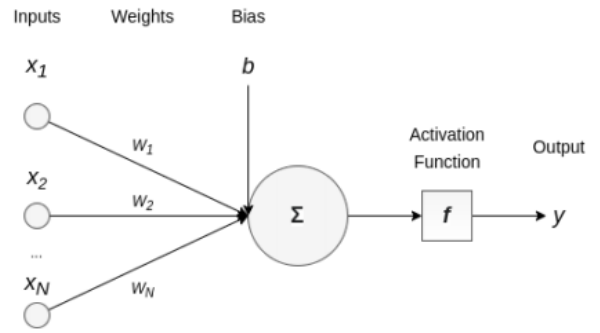

Figure 5. Artificial neural network (A perceptron) [23]

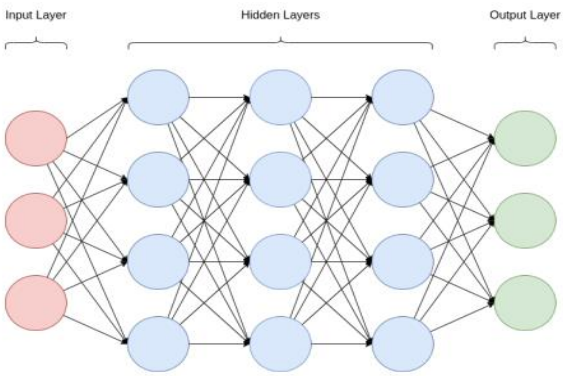

Figure 6. Artificial neural network [23]

2.5 Calculation of the number of neurons and parameters in a typical ANN

2.5.1 Number of neuron

The total number of neurons is computed as:

Total number of neurons at the input layer+Total number of neurons at the hidden layer+Total number of neurons at the output layer

For Figure 6, the total number of neurons is computed as: 3+((1*4)*3)+3=3+12+3=18 neurons.

2.5.2 Number of parameter

The parameters of an ANN refers to the total of number of weights and biases utilised for the development of the ANN architecture.

Considering the ANN of Figure 6, the total number of weights and biases are calculated as follows:

Computation of weights. Hidden Layer:

There are three hidden layers, each hidden layer has 3 weights entering it from each of the three neurons at the input layer.

First Hidden Layer

Each neuron at the first hidden layer has a total of 3 inputs coming = 3 weights from each of the 3 neurons in the input layer, for the 4 neurons at this first hidden layer, Total weights at the first hidden layer = 3 * 4 = 12 weights

Second Hidden Layer

Each neuron at the second hidden layer has a total of 4 inputs coming = 4 weights from each of the 4 neurons in the first hidden layer, for the 4 neurons at this second hidden layer, Total weights at the second hidden layer = 4 * 4 = 16 weights

Third Hidden Layer

Each neuron at the third hidden layer has a total of 4 inputs coming = 4 weights from each of the 4 neurons in the second hidden layer, for the 4 neurons at this third hidden layer, Total weights at the third hidden layer = 4 * 4 = 16 weights

nth Hidden Layer

Therefore, Each neuron at the nth hidden layer which in this case has n=3, has a total of x inputs coming = x weights from each of the xth neurons in the (n-1) hidden layer, for the x neurons at this $n = 3$ hidden layer, Total weights at the nth hidden layer = 4 * 4 = 16 weights.

Therefore, the total weights at the nth hidden layer of any ANN architecture are computed as total number of neurons at the (n-1)th hidden layer * total number of neurons at the nth hidden layer

Output Layer

Each neuron at the output layer has a total of 4 inputs coming = 4 weights from each of the 4 neurons in the third hidden or last hidden layer, for the 3 neurons at this output layer, Total weights at the output layer = 4 * 3 = 16 weights.

Therefore, the total weights at the output layer of any ANN architecture is computed as total number of neurons at the last hidden layer *total number of neurons at the output layer.

∴The total number of weights of ANN of Figure 6 is computed as: 12+16+16+12=56 weights

The formular to compute the total number of weights is given as (total number of neurons at input layer * total number of neurons at the first hidden layer) +(total number of neurons at the first hidden layer *total number of neurons at the second hidden layer) + (total number of neurons at the (n-1)th hidden layer * total number of neurons at the nth hidden layer) +(total number of neurons at the last or nth hidden layer *total number of neurons at the output layer)

Computation of bias. The bias is computed as follows:

First Hidden Layer

For Figure 6, a bias is added to each neuron at the first hidden layer after the sum of the weighted inputs at each of the

neurons, hence, a total of 4 biases is needed at this layer.

Therefore, in a typical ANN architecture, the total number of biases at the first hidden layer is computed as the total of number of neurons at the first hidden layer.

Second Hidden Layer

A bias is also added to each neuron at the second hidden layer after the sum of the weighted inputs at each of the neurons, hence, a total of 4 biases is needed at this layer.

Therefore, in a typical ANN architecture, the total number of biases at the second hidden layer is computed as the total of number of neurons at the second hidden layer.

Third Hidden Layer:

A bias is also added to each neuron at the third hidden layer after the sum of the weighted inputs at each of the neurons, hence, a total of 4 biases is needed at this layer.

Therefore, in a typical ANN architecture, the total number of biases at the third hidden layer is computed as the total of number of neurons at the third hidden layer.

nth Hidden Layer:

yth bias is also added to each neuron at the nth hidden layer after the sum of the weighted inputs at each of the neurons, hence, a total of *y* biases is needed at this layer. Where, *y* is the total number of biases = n the total number of neurons at this layer.

Therefore, in a typical ANN architecture, the total number of biases at the *n*th hidden layer is computed as the total of number of neurons at the nth hidden layer.

Output Layer

A bias is also added to each neuron at the output layer after the sum of the weighted inputs at each of the neurons, hence, a total of 3 biases is needed at this layer.

Therefore, in a typical ANN architecture, the total number of biases at the output layer is computed as given the total of number of neurons at the output layer.

∴The total number of biases of ANN of Figure 6 is computed as: 4+4+4+3=15 biases

The formular to compute the total number of biases is given as given as the total of number of neurons at the first hidden layer.+ The total of number of neurons at the second hidden layer.+ The total of number of neurons at the nth hidden layer.+ The total of number of neurons at the output layer.

Hence, the total parameters in a typical ANN architecture is computed by summing the total number of neurons due to the weight and biases.

For Figure 6, it is given as: 56 weights + 15 biases = 71 parameters.

3. DEEP LEARNING ALGORITHMS

3.1 Deep learning architecture

A typical deep learning architectures consists of the input layer, hidden layer(s) and output layer. There can be shallow architectures or deep architectures. The number of hidden neurons determines if it is shallow or deep type. In shallow architecture, there are one or a smaller number of hidden layers, in deep architecture, there are quite a number of hidden layers. These are illustrated in Figures 7 and 8.

State-of-the-art transformer architectures have been developed using the deep learning architecture. Here the input data which is a sequential data is fed into an encoder which is nothing but a stack of BiLSTMs or LSTMs or even MLPs. The output of such system has a decoder which does opposite work of the encoder to output the target sequential data. The decoder is also a stack of LSTMs or BiLSTMs networks. This type of modelling is called a sequence-to-sequence modelling because the inputs of such model is fed with sequential data and a sequential data is also obtained at the output. Techniques such as the attention mechanism that works in similar fashion to filter or kernel in the CNN has been utilised along-side the positional encoding, embedding to achieve the enhanced performance of the transformer network. Notable example of transformer based models are the LLMs like the GPT, speech recognition like the Whisper, Jasper, speech synthesis like the SeamlessM4T, Translatotron and Translatotron 2 [24-27].

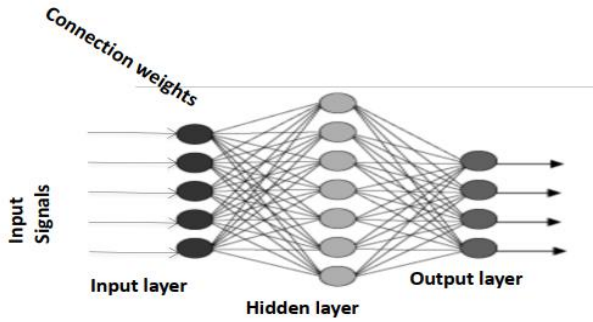

Figure 7. Shallow architecture

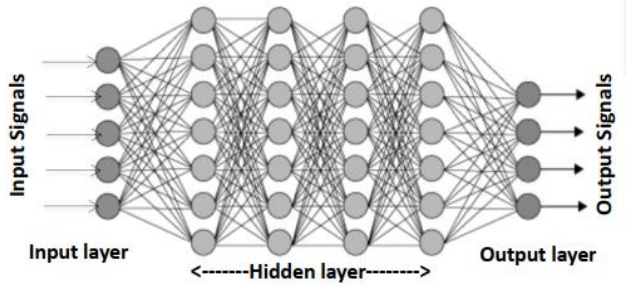

Figure 8. Deep learning architecture

3.2 Classification of deep learning techniques

According to the hierarchical block diagram of Figure 9, Deep Learning can be classified into three main groups: these are the supervised deep learning (discriminative learning), unsupervised deep learning (generative learning), and hybrid deep learning [20].

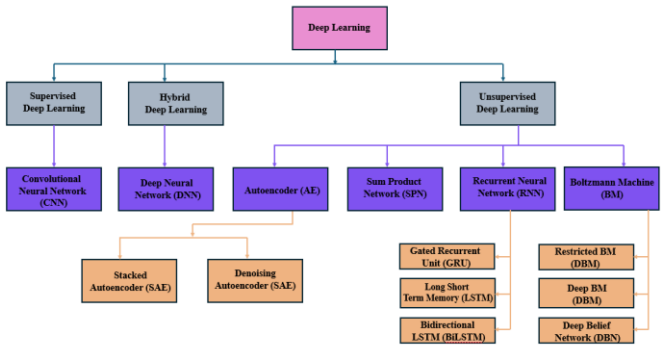

Figure 9. Classification of deep learning techniques [28]

As seen in Figure 9, an example of the Supervised Deep Learning algorithm is the popular CNN utilised for machine

learning when the inputs are images; Deep Neural Network is an example of a Hybrid Deep Learning algorithm; also, it is further given in the figure that unsupervised deep learning algorithms are the Autoencoder (AE) [29], sum product network (SPN), RNN, and Boltzmann machine (BM).

Apart from the deep supervised learning or supervised deep learning, deep unsupervised learning, others types of deep learnings are: deep semi-supervised learning, and deep reinforcement learning or reinforcement learning.

Deep supervised learning: In this type of deep learning, supervision in terms of label data is made available for the trainings alongside the input features. It is a type of learning in which the trainings is carried out towards already known outputs and the validity of the obtained output is compared with the already known output. Examples of such learning are the RNNs, CNNs, deep neural networks (DNN). Gate recurrent units (GRUs), and long short-term memory (LSTM) which are RNN variants are also parts of the algorithms in deep supervised learning [18].

Deep semi-supervised learning: This is also called the partially supervised learning. In this type of learning, the trainings are based on semi-labelled datasets. Examples of algorithms that fall in this category are the GANs, deep reinforcement learning (DRL), RNNs and its variants (GRUs and LSTM).

Deep unsupervised learning: In this type of learning, the trainings are carried out without the provision of the labelled output data. that is only the input features are made available for the trainings. Examples of algorithms here are the auto-encoders, RBMs, GANs, RNN (GRUs and LSTM).

Deep reinforcement learning: In this type of learning, the model learns from experience gathered in the course of trainings. The outcome of what is learned can be inform of reward or penalty. Carrying out this form of learning is much more difficult because of the absence of straight forward loss function. This finds application in the development of games, robotic system etc. The motivations for utilizing this type of learning are to assist in identifying the type of action that generates the highest reward over longer period, to discover the situation that demands actions, giving out of reward function to a learning agent, and for figuring out the best approach to reach large rewards.

3.3 Deep learning algorithms

3.3.1 RvNN

These are deep learning architectures that are used to make predictions for hierarchical structures, and to capture dependencies within recursively structure data. Unlike the CNN that uses convolution operation of the input and kernel in extracting features, and RNN that processes sequential data by traversing backward into the deeper layer of the network (LSTM) or forward and backward direction (BiLSTM), it uses recursive operation on the inputs or child nodes to form the parent nodes representation. The same set of weights are applied in a recursive way over the structure input to generate structure predictions. They are developed to process randomly shaped objects like trees, graphs, or molecular structures in chemistry. Hence their suitability for tasks that involve hierarchical and nested relationships.

It works using the data structure algorithm (DSA) where input is processed recursively, there by merging information from child nodes to parent nodes. This algorithm uses a Recursive Auto-Associative Memory (RAAM) for its development. It is utilised in several areas such as the NLP for sentiment analysis where the information available on the children's nodes is used to assign vectors to each word or sub-phrases.

On comparison with a RNN, both are utilised for sequential data. The RvNN form the present node (parent node) through recursive operations of the previous inputs or child nodes, while RNN also traverses backward but not in a recursive manner but is able to retain information of the previous inputs or time steps or states. Hence their choice in sequential events like the time-series predictions, NLP, speech recognition etc. These are illustrated in Figures 10 and 11 for RvNNs and Figure 12 for RNNs.

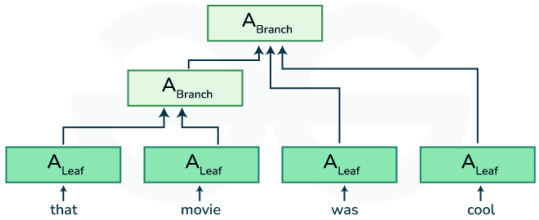

Figure 10. Tree structure of RvNNs [26]

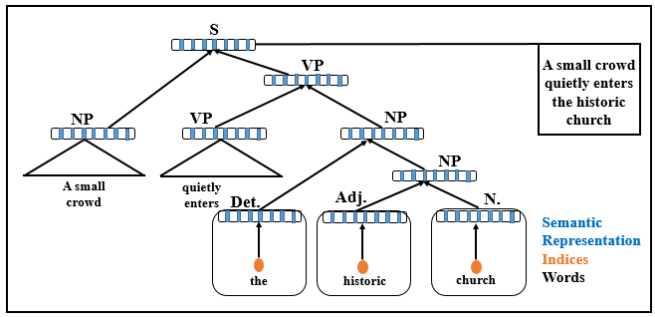

Figure 11. RvNNs for parsing NLS [18]

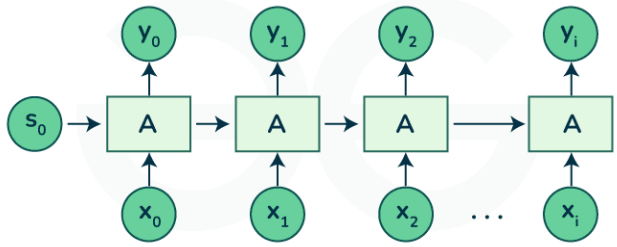

Figure 12. Structure of a RNN [26]

Training. The training of RvNNs involves learning the transformation matrices. Different algorithms are utilised for training. The most popular ones is the Gradient Descent Optimisation algorithm with Back propagation through structure (BTS) learning system. Here, the model learns the weight matrices for each child, and these weight matrices are shared across different recursions for the succeeding children at the same position. The network is trained to regenerate the input layer pattern at the output layer.

Advantages of RvNNs for NLP.

1. The structure of the tree can manage hierarchical data e.g. parsing problems.

2. There is reduction in the depth of the network. This is due to the implementation of logarithm where complexity is $O(\log_n)$

Disadvantages of RvNNs for NLP.

1. The implementation of tree structures introduces bias to the model as the data might not necessarily follow a tree hierarchy structure.
2. Ambiguity and slowness of sentence parsing.
3. Manual parsing of sentence into short components is tedious and time-consuming.

Application of RvNNs.

1. It is implemented for tasks that involves nested like structures such as molecular structure analysis or natural language parsing.
2. It is also utilised in utilised in image segmentation.

Review of Related Work on RvNNs. In the work carried out in [30], the logical deduction in the application of RvNNs for sentence parsing was carried out. Success have been recorded for sentence meaning using RvNNs. Two models comprising tree-structured neural tensor networks (TreeRNTNs) and plain TreeRNNs were evaluated. The models were trained using the SICK challenge dataset and evaluated for recursive structures, relational reasoning, and quantification. The findings of the experiments show that the two models generalize well for the three evaluations, which means they can give logical conclusion in NLP. A max-margin structure prediction architecture which is based on RvNNs was utilised in [31] for the predictions of image and sentence outputs. Using the Stanford background dataset, the algorithms developed for the image outputs achieved State of the Art (SOTA) performance (accuracy) of 78.1% for annotation and segmentation. The duplicated building block in deep neural network was simplified using the dynamic recursive neural network (DRNN) developed [32]. The DRNN was able to achieve recursive outputs using fewer blocks compared to other well-known methods. To further reduce the computation of the algorithm, a gate structure that determines the loop times for each block was added. The gradient problems encountered in RvNNs was solved using the Loop Variable Batch Normalisation (LVBN).

3.3.2 RNN

These are deep learning architectures that are utilised for modeling sequential or temporal data. They find application in NLP, speech recognition, language translation etc. Unlike the traditional neural networks where outputs and inputs are independent of each other, in RNNs, the present output is a function of the previous input elements. It is also true that the present input also relies on the future events. In RNNs, the architecture is such that the system is able to traverse backward or forward into the layers to update the present output result, as illustrated in Figure 13. In RNNs, the same weights are shared across each layer or nodes of the networks, which are mostly adjusted during the gradient descent and backpropagation process in the quest to minimize the errors. The backpropagation through time (BPTT) is employed in RNNs.

Advantages of RNNs.

1. They are integrated with CNNs for best performance.
2. They have the ability to remember previous events.

Disadvantages of RNNs.

1. There are the issues of the exploding and varnishing gradients that occur during optimization.
2. Processing of very long sequence is difficult when ReLU or tanh activation function is utilised.
3. Difficulty in training.

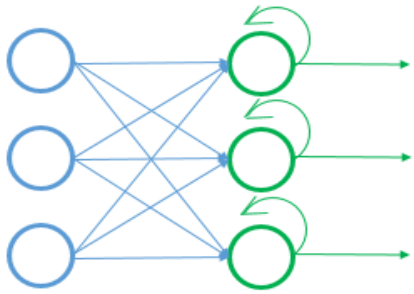

Figure 13. Recurrent nature of a RNN [28]

Types of RNNs:

Based on the numbers of inputs and outputs.

One-to-One. This is a RNNs with one input and one output. It is also known as Vanilla Neural Network. This is illustrated in Figure 14.

One-to-Many. As the name implies, it has one input and many output nodes. It is widely used in image captioning. It is shown in Figure 15.

Many-to-One. This type of RNNs has one input and more than one outputs. It is utilised in sentimental analysis [33]. It is shown in Figure 16.

Many-to-Many. As shown in Figure 17, it has multiple inputs nodes and multiple output nodes. It is used in language translation.

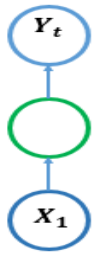

Figure 14. One-to-one RNNs [28]

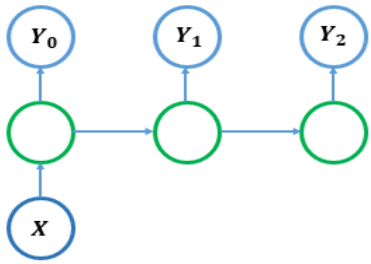

Figure 15. One-to-many RNNs [28]

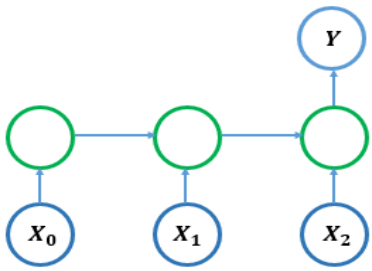

Figure 16. Many-to-one RNNs [28]

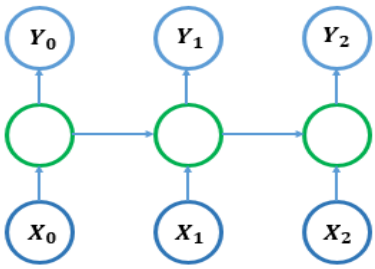

Figure 17. Many-to-many RNNs [28]

Based on the adjustment to the traditional RNNs layers' deep architectures:

Bidirectional RNNs (BRNNs).

These are variants of RNNs that are able to traverse forward and backward into the networks to make predictions of the present events. For instance, in the NLP, given the word string: "You get to go home now", the RNNs could make prediction of "home" by having the knowledge of "now" or any other words prior to "home". Through this architecture, it gives better predictions of any needed present word. Other applications of BRNNs in NLP are Sentimental analysis, Part of Speech Tagging, Machine Translation, and Name Entity Recognition. It has also been utilised in Speech Recognition

Advantages of BRNNs.

1. It allows variable length sequence easier handling.
2. It allows bidirectional processing.
3. It helps to capture better information.
4. BRNNs enhanced the accuracy of the prediction since they capture information both from the future and past.
5. BRNNs are resilience to irrelevant information and noise presents in the data by traversing forward and backward through the network.
6. BRNNs are able to handle the long-term dependency issues experienced in the conventional RNN.

Disadvantages of BRNNs.

1. The forward and backward traversing of the algorithm makes to be computationally complex.
2. The complexity of the system increases the training time of the algorithm when implemented for modelling.
3. Difficulty in the interpreting of the model due to its forward and backward movement through the network.
4. They are mostly prone to overfitting condition due to the huge number of data utilised for training the model.

LSTM.

Traditional RNNs have small memory which denotes their inability to traverse deeper backward into the architecture to make predictions of the present output. This problem was overcome via the LSTM memory architecture. This architecture was designed by Juergen Schmidhuber and Sepp Hochreiter to particularly overcome the varnishing gradient problem encountered in RNNs. It overcomes the problem of long dependency encountered in the network, where the prediction of the present output cannot be made possible if the recent set of events cannot be found. To overcome the long dependency, the LSTM was developed with cells that contain the input, output, and forget gate as shown in Figure 18. They are used in music composition, speech recognition, pharmaceutical development etc.

As indicated in Figure 18, the LSTM network consists of a cell represented by the block, with three gates which are the forget, input, and output gates, which are equivalent of the reset, write, and read for the present cell [34].

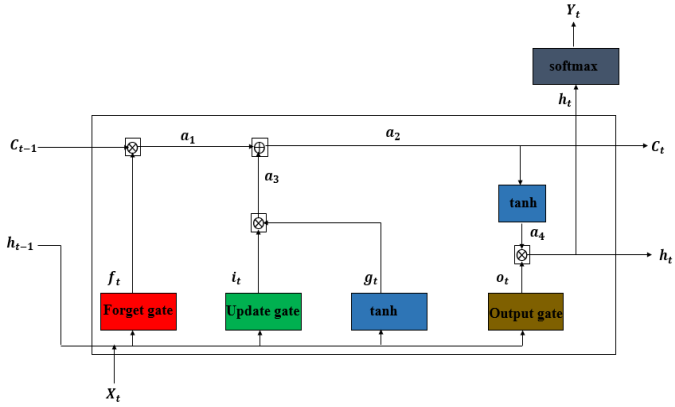

Figure 18. LSTM network [35]

Forget gate. The first gate from the left is the forget gate which has a sigmoid activation function, σ determines which part of the cell information to be forgotten, and which is to be saved. The sigmoid function acts on the present input, x_t and the previous time step, h_{t-1} represented by the bottom arrow entering the cell as indicated in Eq. (9) [34, 35].

$$f_t = \sigma(w_f . x_t + w_f . h_{t-1} + b_f) \tag{9}$$

The top arrow entering the cell represents the previous cell state, C_{t-1}, which is multiplied (elementwise multiplication) by the output of the sigmoid function in Eq. (9) (as indicated in Eq. (10) to determine what part of the previous cell state is to stored and discarded. Since the value of the sigmoid function is between 0 and 1. A 0 indicates to forget, and a 1 means to forget nothing in the previous cell state.

$$a_1 = C_{t-1} \otimes f_t \tag{10}$$

Input Gate. This is the second gate from the left. It has two parts, the update part, i_t and the generation of new values, g_t part for the cell state. These are represented by the sigmoid, σ and tanh functions as shown in Eq. (11) and Eq. (12) respectively [34].

$$i_t = \sigma(w_i . x_t + w_i . h_{t-1} + b_i) \tag{11}$$

$$g_t = tanh(w_g . x_t + w_g . h_{t-1} + b_g) \tag{12}$$

The update part updates the value of the present cell and the generate new values to be added to the cell, using the tanh sigmoid function. These two are elementwise multiplied to obtain output a_3 as indicated in Figure 18. This output a_3 is the input gate of the network. This is illustrated in Eq. (13) as:

$$a_3 = g_t \otimes i_t \tag{13}$$

Then the present or new value (state) of the cell (becomes previous in the next cell) is the sum of the forget gate and input gate as illustrated in Eq. (14) [34] as:

$$a_2 = a_1 + a_3 = [C_{t-1} \otimes f_t] + [g_t \otimes i_t] = C_t \tag{14}$$

Output Gate. This is used to compute the part of the current cell state that is utilised as the previous timestep of the next state. It is indicated as h_t. In the next cell, it becomes h_{t-1}. It is obtained by finding the elementwise multiplication of the output of the Tanh and sigmoid activation functions. The Tanh

activation function acts on the present cell state, C_t as given in Eq. (15) and its output is given as a_4 while the sigmoid function acts on the present input, x_t and previous timestep, h_{t-1} as illustrated in Eq. (16), where it is given as o_t. The elementwise multiplication of Eq. (15) and Eq. (16) are illustrated in Eq. (17) and given as h_t, which is the present output of the cell and it becomes the previous output or timestep in the next cell [34].

$$a_4 = \tanh(C_t) \tag{15}$$

$$o_t = \sigma(w_o.x_t + w_o.h_{t-1} + b_o) \tag{16}$$

$$h_t = a_4 \otimes o_t = \tanh(C_t) \otimes \sigma(w_o.x_t + w_o.h_{t-1} + b_o) \tag{17}$$

Advantages of LSTM

1. They are able to solve the long dependency issues associated with the Vanilla RNN.
2. They have the ability to learn sequential data.
3. LSTMs can make predictions by traversing backward into the network due to the memory.

Disadvantages of LSTM

1. LSTMs have complex systems due to their architecture.
2. There is the need for high computing power.
3. They sometimes forget very important information in the previous state of the network.
4. They are sometimes difficult to comprehend.
5. LSTMs require huge volume of data to learn from for better performance.
6. They require more training time due to the complexity of the system.

Gated Recurrent Units. This architecture is similar to the LSTM in that it was designed to overcome the problem of long dependencies due to memory issues in the network. It uses hidden states to control the flow of information instead of the cell utilised in the LSTM. It also uses two gates, that is an update gate, and a reset gate to control the volume and type of information to be kept. Prediction of previous information deeper into the network using RNN, becomes an issue due to exploding gradient and varnishing gradients. It has been applied in the detection of stress in Electro Encephalogram (EEG) signals [36].

Exploding Gradient

This arises during backpropagation when trying to improve the performance of the model where the gradient exponentially explode and prevents the convergent of the model. Then weights and biases' update tend to become unstable. One solution to this is called gradient clipping where the gradient vectors are clipped if greater than the threshold set.

Varnishing Gradient

This occurs when the gradient exponentially decay such that the performance of the model cannot be updated due to zero value of the gradient. Then the update made on the weights and biases in the network become so small. To solve this problem, the Gated Recurrent Unit (GRU) or LSTM is used so that very long dependence can be captured. While GRU is faster than LSTM with low memory, the LSTM is more accurate when dealing with longer datasets. Other approaches to solving the varnishing gradient problem is the use of ReLU activation function, and batch normalization

Application of RNNs

1. NLP
2. Speech Recognition
3. Machine Translation
4. Time Series Forecasting
5. Face Detection
6. Handwriting recognition

3.3.3 RvNN versus RNN

Both RvNN and RNN are used to process sequential data. Difference ocurs due to how they are structures. RvNNs process sequential data in a tree like fashion, while RNN is utilised to capture dependencies over time. The differences between the two are captured in the Table 1.

Table 1. Difference between RNNs and RvNNs

S/N	Features	RNN	RvNN
1	Architecture	There is a tree-like, or hierarchical structure	There is a chain-like or sequential structure
2	Memory	It captures information via the sequential memory	Memory is limited
3	Data Preprocessing	It processes time-series and sequential data	It processes hierarchical data
4	Training Complexity	It uses backpropagation through time	For training, it uses specific tree traversal algorithms
5	Connections	Based on sequential order	Based on hierarchical structure.
6	Application Areas	Speech recognition, Speech Synthesis, Language Modelling	Image Parsing, Syntactic parsing, and other NLP applications

3.3.4 CNNs

These are networks with multiple layers that are mostly used for object detection and image processing. They are also called ConvNets. The first CNN architecture named LeNet was developed by Yann LeCun in 1988. It was used for character recognition like the digits, zip codes etc. it has been applied in brain tumor detection [37], brain stroke detection [38], cyber bullying detection [39]. Other application areas include detection of anomalies, identification of satellite images, forecasting of time series, medical image processing etc. [18, 40].

Architecture of CNN. The CNN has three layers, which are the input, hidden, and the output layer. The hidden layer consists of the convolution layer, pooling and fully connected layer.

The Convolution Layer. Within the convolution layer, the input is convolved with the filter or kernel in the hidden layer. The kernel slides over the input layer convolving with it to obtain the reduced dimensional feature map. Feature extractions do mostly occur during the stage of the CNN. The movement of the sliding operation of the filter over the input is controlled by the parameter called stride of the CNNs. It has been observed that input parameters of the matrix towards the edge are not actively engaged in the convolution process. To address these issues, padding of the input matrix is normally carried. This ensures that every input values within the matrix is actively engaged in the convolution process.

Convolution Operation within the Convolution Layer.

Consider an image with 5 × 5 dimensions. When fed into the CNN, the convolution layer with the help of a filter or kernel acts on the input image so as to find local patterns and features from the input image. If a filter or kernel of dimension 3 × 3 acts on the image, the following convolution operation occurs where the filter slides over the image to extract relevant information:

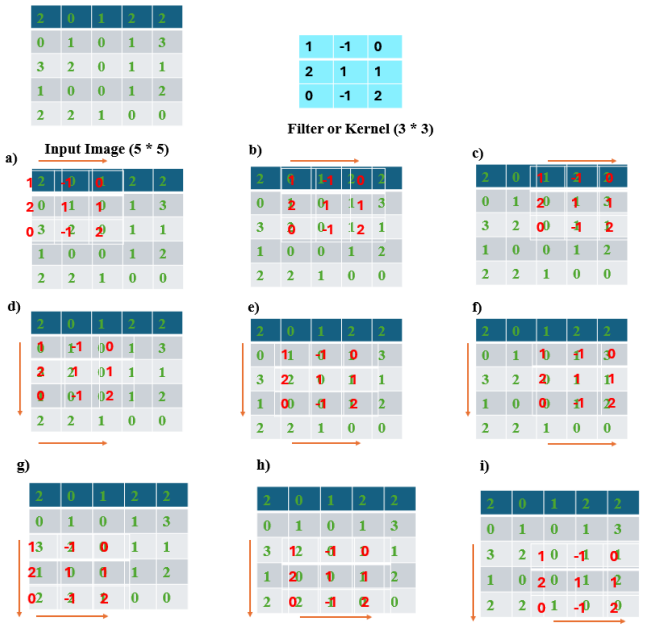

Figure 19. Sliding of the kernel through the image and convolution operation with stride = 1

As can be seen in Figure 19, the convolution operation starts at a), and as the filter slides through the images, various convolution operations are carried out as indicated in b) to i) after which the a 3 × 3 feature map is obtained. The convolution operation at each position of the slide as given as follows:

$$\begin{vmatrix}2&0&1\\0&1&0\\3&2&0\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (2*1)+(0*-1)+(1*0)+(0*2)+(1*1)+(0*1)+(3*0)+(2*-1)+(0*2) = (2+0+0)+(0+1+0)+(0+-2+0) = (2+1-2) = 1$$

$$\begin{vmatrix}0&1&2\\1&0&1\\2&0&1\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (0*1)+(1*-1)+(2*0)+(1*2)+(0*1)+(1*1)+(2*0)+(0*-1)+(1*2) = (0+-1+0)+(2+0+1)+(0+0+2) = (-1+3+2) = 4$$

$$\begin{vmatrix}1&2&2\\0&1&3\\0&1&1\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (1*1)+(2*-1)+(2*0)+(0*2)+(1*1)+(3*1)+(0*0)+(1*-1)+(1*2) = (1+-2+0)+(0+1+3)+(0+-1+2) = (-1+4+1) = 4$$

$$\begin{vmatrix}1&1&0\\3&2&0\\1&0&0\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (0*1)+(1*-1)+(0*0)+(3*2)+(2*1)+(0*1)+(1*0)+(0*-1)+(0*2) = (0+-1+0)+(6+2+0)+(0+0+0) = (-1+8+0) = 7$$

$$\begin{vmatrix}1&0&1\\2&0&1\\0&0&1\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (1*1)+(0*-1)+(1*0)+(2*2)+(0*1)+(1*1)+(0*0)+(0*-1)+(1*2) = (1+0+0)+(4+0+1)+(0+0+2) = (1+5+2) = 8$$

$$\begin{vmatrix}1&1&3\\0&1&1\\0&1&2\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (1*1)+(1*-1)+(3*0)+(0*2)+(1*1)+(1*1)+(0*0)+(1*-1)+(2*2) = (0+-1+0)+(0+1+1)+(0+-1+4) = (-1+2+3) = 4$$

$$\begin{vmatrix}3&2&0\\1&0&0\\2&2&1\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (3*1)+(2*-1)+(0*0)+(1*2)+(0*1)+(0*1)+(2*0)+(2*-1)+(1*2) = (3+-2+0)+(2+0+0)+(0+-2+2) = (1+2+0) = 3$$

$$\begin{vmatrix}2&0&1\\0&0&1\\0&1&0\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (2*1)+(0*-1)+(1*0)+(0*2)+(0*1)+(1*1)+(0*0)+(1*-1)+(0*2) = (2+0+0)+(0+0+1)+(0+-1+0) = (2+1+-1) = 2$$

$$\begin{vmatrix}0&1&1\\0&1&2\\1&0&0\end{vmatrix} * \begin{vmatrix}1&-1&0\\2&1&1\\0&-1&2\end{vmatrix} = (0*1)+(1*-1)+(1*0)+(0*2)+(1*1)+(2*1)+(1*0)+(0*-1)+(0*2) = (0+-1+0)+(0+1+2)+(0+0+0) = (-1+3+0) = 2$$

The results of Figure 20 is a 3 × 3 matrix called the feature map, which is given in Figure 21.

Figure 20. Feature map of above convolution (stride = 1)

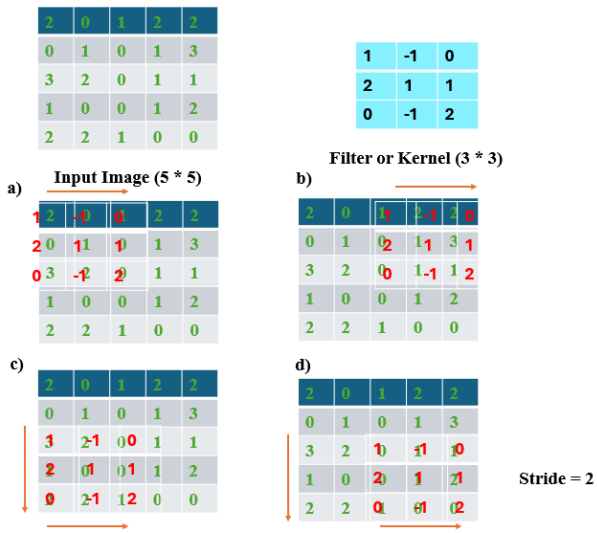

Figure 21. Sliding of the kernel through the image and convolution operation with stride = 2

As given in the convolution above, the matrix operation is an array multiplication carried out at each position. It should also be noted that for the convolution operation carried out, the strides, which is the speed of the filter movement along the image is taken to be 1.

Techniques Utilised CNN Convolution Operation

Stride. This refers to the speed with which a filter translates through an image. A stride of 1 depicts the filter travels one pixel at a time, 2 pixels per time with a stride of two. e.g.

If a stride of length 2 is to be utilised, the following

convolution operations shown in Figure 21 will be obtained.

As shown in Figure 21, the higher the stride, the higher the speed or motion through the input image by the filter or kernel and the lower the dimension. This also reduces the computational complexity of the model due to a reduced dimension. However, the use of high value for the stride has the effect of information loss. The result of using a stride of two for a 5 × 5 image and 3 × 3 kernel is a 2 × 2 feature map which is given as follows and the feature map output is shown in Figure 22.

$$\begin{vmatrix}2 & 0 & 1\\ 0 & 1 & 0\\ 3 & 2 & 0\end{vmatrix} * \begin{vmatrix}1 & -1 & 0\\ 2 & 1 & 1\\ 0 & -1 & 2\end{vmatrix} = (2*1)+(0*-1)+(1*0)+(0*2)+(1*1)+(0*1)+(3*0)+(2*-1)+(0*2) = (2+0+0)+(0+1+0)+(0+-2+0) = (2+1-2) = 1$$

$$\begin{vmatrix}1 & 2 & 2\\ 0 & 1 & 3\\ 0 & 1 & 1\end{vmatrix} * \begin{vmatrix}1 & -1 & 0\\ 2 & 1 & 1\\ 0 & -1 & 2\end{vmatrix} = (1*1)+(2*-1)+(2*0)+(0*2)+(1*1)+(3*1)+(0*0)+(1*-1)+(1*2) = (1+-2+0)+(0+1+3)+(0+-1+2) = (-1+4+1) = 4$$

$$\begin{vmatrix}3 & 2 & 0\\ 1 & 0 & 0\\ 2 & 2 & 1\end{vmatrix} * \begin{vmatrix}1 & -1 & 0\\ 2 & 1 & 1\\ 0 & -1 & 2\end{vmatrix} = (3*1)+(2*-1)+(0*0)+(1*2)+(0*1)+(0*1)+(2*0)+(2*-1)+(1*2) = (3+-2+0)+(2+0+0)+(0+-2+2) = (1+2+0) = 3$$

$$\begin{vmatrix}0 & 1 & 1\\ 0 & 1 & 2\\ 1 & 0 & 0\end{vmatrix} * \begin{vmatrix}1 & -1 & 0\\ 2 & 1 & 1\\ 0 & -1 & 2\end{vmatrix} = (0*1)+(1*-1)+(1*0)+(0*2)+(1*1)+(2*1)+(1*0)+(0*-1)+(0*2) = (0+-1+0)+(0+1+2)+(0+0+0) = (-1+3+0) = 2$$

1	4
3	2

Figure 22. Feature map of the above convolution for stride of 2

Padding. During the convolution operations, it has been observed that only border regions of the input image undergo less filtering operation. That is, the pixels around the border or edges are convolved the least amount of time compare to the pixels within the body of the image matrix (border effect)-this leads to loss of information at the borders. Hence, the padding technique is used to extend the dimension by adding zeros so that the initial border regions are now within the body of the new matrix pixel obtained. This is illustrated in Figure 23 for the initial 5 × 5 input image matrix of Figure 23.

0	0	0	0	0	0	0
0	2	0	1	2	2	0
0	0	1	0	1	3	0
0	3	2	0	1	1	0
0	1	0	0	1	2	0
0	2	2	1	0	0	0
0	0	0	0	0	0	0

Figure 23. The effect of pooling on a 5 × 5 input image

Pooling Layer. This is used for dimensionality reduction of the output activated feature map. For instance, if the size of the output of the convolution layer is 8 × 8, pooling can reduce this size to 2 × 2. In this case pooling is carried out on 4 × 4 sub-block of the convolution output matrix. If the size of the feature map is 2 × 2, pooling can reduce it to 1 dimensional space. It involves the application of filter that traverse over varying regions of the feature map to extract a single output value. The single output value can be an average or maximum representation. Hence, two types of pooling are known with CNNs. These are the average pooling and max pooling. In average pooling, the average of the selected matrix grouped is found while the max is selected in the case of the max pooling. Another benefits of pooling is increase in the receptive field of the CNN network. Pooling over a region or window within the feature map ensures the capturing of necessary patterns needed for trainings.

Average Pooling. The average value of the region of the filter or window is found to obtain a single value representation of that window. This is illustrated in Figure 24 for a 4 × 4 by matrix.

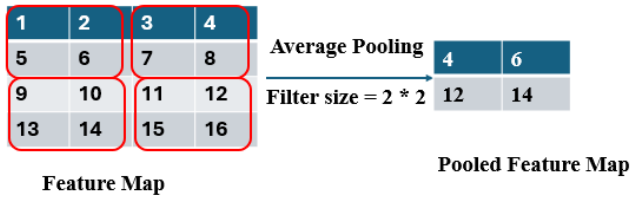

Figure 24. Average pooling in CNN

As shown in Figure 24, the average of each pooling window is found as a single representation of that window, and the resulting matrix is a feature map with a dimension of 2 × 2.

Max Pooling. This involves selecting the maximum value within each region or pool window within the matrix. This ensures that most important features are captured while potential noise is ignored. An example of max pooling is shown in Figure 25.

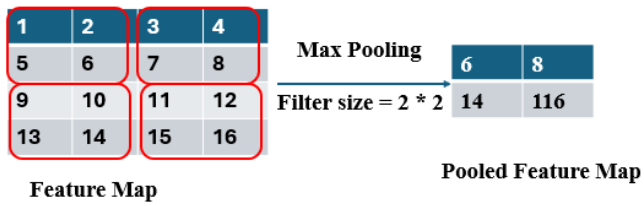

Figure 25. Max pooling in CNN

Fully Connected Layer. The output of the pooling layer, is flattened into a 1-dimensional space. If for instance, the output is a 4 × 4, this will be flattened to become 1 × 8 or simply put a vertical dimensional space with a size or length of 8. This will serve as input to the fully connected layer. By fully connected layer, it implies that every input is fully linked to the node of the next layer in the hidden layer of the network. Then classification takes place within this layer. This is illustrated in Figure 26.

As shown in Figure 26, the pooled feature map is first flattened and served as input to the fully connected layer with large number of hidden layers. The first hidden layer has 5 neurons and the last hidden layer also has 5 neurons. The output of the network is obtained at the output layer with 2 neurons.

Evolution of CNN Architectures. As initially stated, the first

CNN architecture named LeNet was developed by Yann LeCun in 1988. It was used for character recognition like the digits, zip codes etc. over the past 10 years several CNN architectures have been developed. Modifications have been made to the existing ones to obtain an improved CNN architecture. Some of the modifications carried out are regularization, structural reformulation, parameter optimizations etc. [18]. Presented next are some of the CNN architectures.

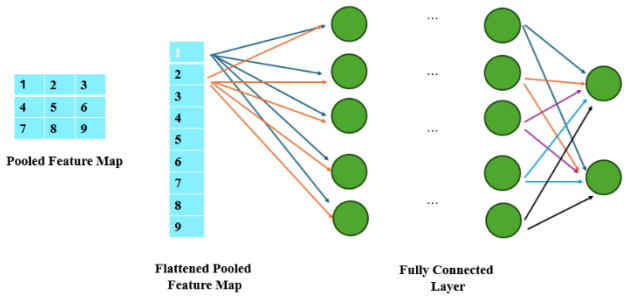

Figure 26. Fully connected layer in CNN

LeNet. The inputs to the LeNet are images and this CNN architecture consists of the following:

1. Convolutional layer with 6 filters each of size 5 * 5 followed by 2 * 2 max pooling layer.
2. Convolutional layer with 16 filters each of size 5 * 5 followed by 2 * 2 max pooling layer.
3. Fully connected layer with a sigmoid activation function and 120 units.
4. Fully connected layer with a sigmoid activation function and 84 units.
5. Output layer with a softmax activation function and 10 units.

AlexNet. In 2012, AlexNet was developed by Alex Krizhevsky et al where it achieved SOTA results on ImageNet dataset. The architecture consists of 5 convolutional layers unlike 2 present in LeNet followed by 2 fully connected layers, and 1 output layer. It is summarized as follows:

1. Convolutional layer with 96 filters each of size 11 * 11 followed by 2 * 2 max pooling layer.
2. Convolutional layer with 256 filters each of size 5 * 5 followed by 2 * 2 max pooling layer.
3. Convolutional layer with 384 filters each of size 3 * 3
4. Convolutional layer with 384 filters each of size 3 * 3
5. Convolutional layer with 256 filters each of size 3 * 3 followed by 2 * 2 max pooling layer
6. Fully connected layer with a ReLU activation function and 4096 units.
7. Fully connected layer with a ReLU activation function and 4096 units.
8. Output layer with a softmax activation function and 1000 units.

VGG. This was developed by Andrew Zisserman and Karen Simonyan in the year 2014. It is an improvement on the AlexNet considering the number of filters, dropout regularization technique, and max pooling layers utilised. It is summarized as:

1. Convolutional layer with 64 filters each of size 3 * 3 followed by 2 * 2 max pooling layer.
2. Convolutional layer with 128 filters each of size 3 * 3 followed by 2 * 2 max pooling layer.
3. Convolutional layer with 256 filters each of size 3 * 3 followed by 2 * 2 max pooling layer.
4. Convolutional layer with 512 filters each of size 3 * 3 followed by 2 * 2 max pooling layer.
5. Convolutional layer with 512 filters each of size 3 * 3 followed by 2 * 2 max pooling layer
6. Fully connected layer with a ReLU activation function, dropout regularization and 4096 units.
7. Fully connected layer with a ReLU activation function, dropout regularization and 4096 units.
8. Output layer with a softmax activation function and 1000 units.

ResNet. To tackle the varnishing gradient problem encountered during training, ResNet was introduced by Kaiming He et al. in 2015. The architecture can be summarized as follows:

1. Convolutional layer with 64 filters each of size 7 * 7 followed by 3 * 3 max pooling layer.
2. Multiple residual blocks with each containing:
   - Convolutional layer with 64 filters each of size 3 * 3
   - Convolutional layer with 64 filters each of size 3 * 3
   - Shortcut connection that merges the original input to the output of the block.
3. Multiple residual blocks with each containing:
   - Convolutional layer with 128 filters each of size 3 * 3 followed by 2 * 2 max pooling layer.
   - Convolutional layer with 128 filters each of size 3 * 3.
   - Convolutional layer with 128 filters each of size 3 * 3
   - Shortcut connection that merges the original input to the output of the block.
4. Multiple residual blocks with each containing:
   - Convolutional layer with 256 filters each of size 3 * 3 followed by 2 * 2 max pooling layer.
   - Convolutional layer with 256 filters each of size 3 * 3.
   - Convolutional layer with 256 filters each of size 3 * 3
   - Shortcut connection that merges the original input to the output of the block.
5. Multiple residual blocks with each containing:
   - Convolutional layer with 512 filters each of size 3 * 3 followed by 2 * 2 max pooling layer.
   - Convolutional layer with 512 filters each of size 3 * 3.
   - Convolutional layer with 512 filters each of size 3 * 3
   - Shortcut connection that merges the original input to the output of the block.
6. Fully connected layer with a softmax activation function, and 1000 units [18].

A block diagram of ResNet is displayed in Figure 27.

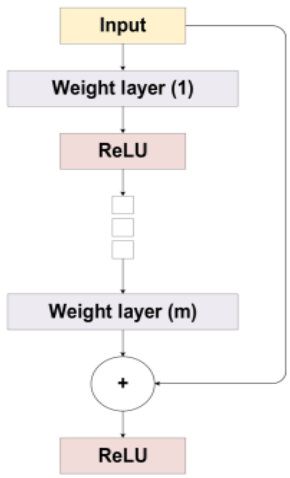

Figure 27. Block diagram of ResNet [18]

EfficientNet. This was invented by Quoc V. Le and Mingxing Tan from Google Research in 2019. The distinct feature of this architecture is its new scaling method. This ensures uniform scaling of all dimensions of width, depth, and resolution with the help of a compound coefficient. Variants of efficientNet are EfficientNet-B0, EfficientNet-B1 to B7, EfficientNet-Lite

Review of Related Work on CNNs.

CNN is one of the most significant networks in the field of deep learning [22]. It has been explored in NLP, and computer vision, hence its choice in academia and industry in the past few years. The review paper looked into the applications of CNN in diverse scenarios without its consideration from general perspective. In addition to those novel aspects and ideas in this area of CNN, the one-dimensional CNN, two-dimensional CNN, as well as multi-dimensional CNN were also discussed in this paper. The paper started with an introduction to CNN, then an overview, followed by introduction of classic and advanced CNN models being introduced. Thereafter, experimental analysis of CNN was carried out after which conclusions were drawn and several rules of thumbs were highlighted for selection of functions. Then the areas of applications of one-dimensional, two-dimensional, and multi-dimensional CNN were covered and the paper ended with some challenges faced with the CNN architecture.

CNNs overviewed was covered in the study [41]. Its applications in the area of image recognition tasks were also highlighted. The paper started with an introduction to the CNN architecture where the different phases of Machine learning and Artificial Intelligence such as the data collection, preprocessing of data, selection of model, trainings of the model, evaluation, model deployment, monitoring and model maintenance were discussed extensively. Then the various contributions of the paper such as a thorough review of recent developments in CNNs, discussion on the fundamentals of CNNs, comparison of different CNN architecture, recommendation for data scientists and developers, discussion on libraries for CNN, cost of CNN estimations, recent developments in CNN, efficiency and reliability enhancement of CNNs and finally, summary of the whole work was given where they stated that image recognition field has been revolutionized by the CNNs and they also find applications in other domains, and that CNNs have the ability and potential to bring about transformation from health care industry and finance to entertainment and transportation sectors due to increase in the volume of data and large computing resources made available for the trainings.

Biological vision's early findings inspired the birth of CNN [42]. That CNNs have found immense applications in Computer vision tasks. The paper started with a brief introduction to CNN, the origin of CNN, its application in computer vision, and its validation in the field, its comparison to human neural and behavioural levels, its variants, the datasets available, different architecture of CNNs, trainings procedures, understanding of CNNs where approaches such as the empirical methods, mathematical analysis were discussed extensively. Then they ended by considering the limitations of CNN such as the way in which the networks are trained, assumption of weight sharings etc, and they provided future directions.

CNN was utilised to model a sign language recognition system in the study [43]. Deafness and voice impairment have caused a gap in communication between these impaired people and the outside world. The static America Sign Language dataset where through the HandDetector module, the signer's hand images were detected and captured using PC webcam was utilised for the training. The dataset consists of 44, 654 images, which is split into 20, 772, 8, 903, and 14, 979 for train, validation, and test sets respectively. The input images were first processed and then fed to the CNN which consisted of 3 convolution layers, and an output SoftMax layer, and is trained using an Adam Optimiser. Accuracies of 99.86%, 99.94%, and 94.68% were obtained for train, validation, and test respectively.

However, CNN was used to train a model that classified three types of maize leaf diseases that comprises of Common Rust, Leaf Blight, and Leaf Spot [44]. According to the work, the input to the CNN trained using Keras platform with Python TensorFlow had neither preprocessing nor feature extraction, and its size was 224 × 224 pixels. The model was able to achieve an accuracy of 98.56% which shows that farmers could save time and problem of incorrect detection mostly encountered using manual approach is taken care of.

3.3.5 GANs

In 2014, a paper published by researchers at the University of Montreal introduced the GANs. They are deep learning algorithms that are trained to make prediction, classify images etc through two components comprising the generator and discriminator. The generator generates fake output which the discriminator learns from and authenticity evaluation of the data to a real world is carried out. The generator receives the input data and undergo training to generate output which are fed to the discriminator. The discriminator then checks if the output is a representation of the real-world output. If it is a good representation, this is given as the output else, this is returned or fed back to the generator that further trains on the data until better results free from discrimination from the discriminator is generated.

3.3.6 Radial basis function networks

These are types of feedforward neural networks that use the radial basis function as their activation functions. They are utilised for time-series prediction, regression, and classification. They perform classification through measurements of the similarity between the input and the training set. The output layer has one node in each class or category. They have input vector that are fed to the input layer, and they also have a layer of radial basis function neurons. Then the weighted sum of the inputs is found. Gaussian transfer function are used for neurons in the hidden layer. The output of this function and the distance from the neuron's center are inversely proportional to each other. Linear combination of the neuron's parameters and the input of RBF gives the output of the network.

3.3.7 Multilayer perceptrons

This is a feedforward neural network with multiple layers of perceptrons. The number of inputs and outputs in MLPs are the same but there are multiple hidden layers. They are utilised in machine translation, image recognition, and speech recognition.

Working Principle. The data are feed to the input layer of the network. The signal passes in one direction via the graphical connection of layers of neurons. It then computes the weighted inputs that exists between the input and hidden layers. It then uses activation functions such as tanh, ReLU, sigmoid

functions to determine which node to fire. Thereafter, it trains the model to learn dependencies and understand the correlation that exist between the independent and target output from a training dataset.

3.3.8 Self organising maps

This was invented by Professor Teuvo Kohonen. It is a type of artificial neural network that reduces the dimension of data via the data visualisation. They are created to help humans have the ability to visualize high-dimensional data through self-organising artificial neural networks.

Working principle. It first carryout weights initialization for each node and make random selection of vectors from the training data. it then finds out weights that are most likely be the input vector through examination of every node. The winning node is referred to as the Best Matching Unit (BMU). It then discovers then neighbourhood of the BMU. The sample vector is awarded a winning weight. The shorter the distance between a node and a BMU, the more the changes in weight, and the longer the distance between a node (neighbour) and BMU, the less it learns.

3.3.9 RBMs

These are stochastically shallow two-layered neural networks that are developed by Goeffrey Hinton. It was able to tackle the problem of varnishing gradients known to most deep learning networks. They can learn from a probability distribution over a set of inputs which are not labeled, and the non-labeling seems to be beneficial for real world data like the videos, photos, sensor data that are mostly unlabeled.

Working Principles. Two phases of RBMs exists. These are forward pass and backward pass. It translates the accepted inputs into a set of numbers that encodes the inputs in the forward pass. It then combines each activation with their corresponding weight and one overall bias. The output is passed to the next hidden layer. Through the backward pass, it takes the set of numbers and translate them to produce the reconstructed inputs. It then combines each activation with individual weight and bias and pass the output to the visible layer for reconstruction. A high degree of accuracy is achieved for backpropagation of a well trained RBM network. The biases and weights assist the RBM to decode the interrelationships between the inputs and to decide the essential inputs in detecting patterns. Via different weights and biases, the model is trained until the difference between the inputs and the reconstructed inputs are as minimal as possible.

Review of Literatures on RBMs. RBM is a family of machine learning and statistical models and played a central role in the growth and development of the deep learning was discussed [45]. It is explored in tasks such as representation and classification learning. According to the authors, through the phase diagrams derived for various statistical ensembles of RBM, the functioning of RBM can be analysed and that the identification of compositional phase has been achieved where a small number of nodes or features are combined to form patterns that are complex. The paper started with an introduction to RBM, definition of the model and learning, then discussion on stochastic gradient descent, was given and overview of various RBM settings that involve the Gaussian-Gaussian RBM, Gaussian-Spherical, Gaussian-Softmax, Bernoulli-Gaussian RBM, Gaussian-Bernoulli RBM, Bernoulli-Bernoulli RBM were highlighted. Discussion on phase diagram of the Bernoulli-Bernoulli RBM, learning RBM and finally conclusions in the form of summary was given where they discussed about the learning quality, the number of hidden nodes which affects representation power of RBM, the landscape of learned RBMs, the landscape of free energy, and the link between the dataset and the learned features.

The study [46] looked into the survey on the RBM and DBN. The paper was also a tutorial that discussed about the BM, RBM, and DBN. The paper commenced by giving more insight into the graphical models that are probabilistic in nature, Gibbs sampling, Markov random field, Ising model, statistical physics, and Hopfield network. Thereafter, the structure of BM and RBM were introduced. The explanation on Gibbs sampling in RBM for generating variables, contrastive divergence, maximum likelihood estimation training of BM & RBM, as well as the conditional distributions of hidden and visible variables were given. Then discussion on the continuous and discrete distributions for the variables was also highlighted, then they explained on how conditional RBM is trained. Finally, explanation of DBM which is a stack of RBM models was given. According to the authors, the model can find application in statistics, data science, statistical physics, as well as neural computation.

3.3.10 DBNs

DBNs was also developed by Geoffrey Hinton as alternative to providing solution to backpropagation problems that involves the varnishing gradients. It has a similar structure to the MLP but diverse in training. These are generative models that have multiple layers of latent, stochastic variables. The latent variables that contain binary values are called hidden units. DBNs are integration or stack of Boltzmann machines. Communication exists between each RBM layer with both the previous and subsequent layers. The hidden layer of the first RBM which is trained for the reconstruction of inputs accurately, is the visible layer of the next RBM ahead of it. They are utilised in motion capture data, image recognition, video recognition. Just like an integration of perceptrons called MLP outperforms perceptrons, the stack or integration of RBMs outperformed a single RBM network. To complete the trainings of the DBN, labels are mostly introduced to the patterns in the network and fine-tuned in a supervised way.

Working Principles. Greedy learning algorithm which uses a layer-by-layer approach to learn the top-down generative weights, are used to train the DBNs. Steps of Gibbs sampling on the top two hidden layers are carried out. It then draws a sample from the visible units by utilizing a single pass ancestral training. It then learns that the latent variables in every layer can be inferred by a single, bottom-up pass.

3.3.11 Autoencoders

They are feedforward neural networks with identical inputs and output. It was developed by Geoffrey Hinton in 1980s. they are used to solve unsupervised learning problems. Other areas of application include image processing, pharmaceutical discovery, and popularity predictions. In autoencoders, input data are encoded as vectors, which are compressed, hidden and representation of the raw input data, and to achieve the low dimensionality reduction of the input raw data. it has been applied in the detection of intrusion and cyber attacks [29, 47].

Working Principles. It consists of encoder, code, and decoder which are three main components of the network. It receives the inputs and transforms it into a signal representation. It first encodes the image; this ensures that the size of the input reduction size. Finally, the decoding of the

image to generate the reconstruction images.

3.3.12 Choosing a deep network

In choosing the type of deep network, decision has to be made if patterns are to be found, classification model is to be built, or the type of deep learning such as the supervised or unsupervised etc is needed for the task. If image classification model is needed, then one can be thinking of exploring the CNNs, modeling of sequential data is best carried out by the RNNs, for extraction of patterns from a set of unlabeled data, autoencoder or RBM can be considered.

Factors to be considered in choosing a particular type of deep learning algorithms. The choice of a deep learning algorithm must align with the problem being addressed or the application being taking into consideration. Several factors must be taking care of when choosing a deep learning algorithm for specific tasks. They are as follows:

The type of input data. The input data fed into the model to be trained has a lot to say about the type of deep learning algorithm. For instance, for sequential data, the RNN or RvNN or variants of RNN can be utilised; for image data or data with certain patterns, CNN can be used for such tasks.

Learning tasks. The type of learning task also has an effect in determining the type of deep learning algorithm to be used. For instance, for classification and prediction problems, the FNN can be used; for tasks that needs knowledge of the long dependences in the data, the LSTMs can be implemented.

Availability of data. This is another key factor in the selection of the deep learning algorithm. Deep learning algorithms or models developed like the CNNs architectures such as the AlexNet, ResNet etc, RNN based models like the speech recognition models, machine translation models and state-of-the-art transformer-based models like the CHAT-GPT, speech synthesis, and speech recognition models such as Whisper, Jasaper, AudioPalm are trained with large volume of data. [cite whisper, Translatotron, Jasper, AudioPalm]. Whisper is trained with over 600, 000 hours of corpus and transcripts sourced over the internets.

Complexity and depth of the problem. Simple problem modelling can be carried-out using simple architecture such as the Perceptron; while complex problems like the image classification as in the case of brain tumor classification, and disease classification, can be achieved using the CNN architecture; speech recognition, text-to-speech modelling or any other speech processing tasks can be carried-out using RNNs, or transformer based RNNs where stack of LSTMs or BiLSTMs are utilised as encoders and decoders.

Table 2. Type of deep learning and their areas of application

S/N	Applications	Types of Deep Learning
1	Sentimental analysis, text processing, name entity recognition, parsing	Recursive Neural Tensor Network (RNTN), RNNs,
2	Language model that functions at character level	RNNs
3	Object recognition	RNTN
4	Image Recognition	CNNs, DBNs
5	Speech Recognition	RNNs
6	Classification	DBNs and MLP with ReLU
7	Time series	RNNs

Computational resources. The availability of the hardware resources like the high computing Graphic Processing Units (GPU) for tasks like the CNNs or RNNs that require high memory and space is another important factor to be considered in choosing the type of deep learning algorithms for a specific task.

A detailed illustration of the type of deep learning for various application is given in Table 2.

4. DISCUSION

Different architectures of deep learning have been developed by different researchers. For instance, for various image recognition, character recognition, reading of bank checks, various CNNs architectures have addressed these areas. Different architectures are AlexNet, ResNet, ResNext, SqueezeNet, MobileNet, LeNet [12-15]. For sentence parsing using the RvNNs, plain TreeRNNs and tree-structured neural tensor networks (TreeRNTNs) models were utilised [30] for evaluation. DRNN was modeled where its computational complexity was catered for using a gate structure, and the backpropagation problems were addressed via the Loop Variable Batch Normalisation (LVBN) [32]. According to the study [22], CNN is one of the most significant networks in the field of deep learning, and its application areas particulary in image processing were highlighted [41, 42]. Illustration of the different types of RBMs settings such as the Gaussian-Spherical, Bernoulli-Gaussian RBM, Gaussian-Gaussian RBM, Gaussian-Softmax [45] shows that more intense works are ongoing in this exploration of deep learning. In addition to this, the present-day world is ocuupied with the generation of video contents, image contents which have been made possible through the GANs. The GANs with the help of generator and discriminator is able to achieve real world output representation that have been utilised by individuals for their daily consumptions. The following review paper shows that researchers are findings or developing systems or models that achieve the purpose for which deep learnings were developed for. Findings show that CNNs are the most widely used deep learning architectures.

5. CONCLUSIONS

In this review work, the different types of deep learning have been explored. Deep learning which is a subset of machine learning via its deep architecture have achieved great feat in various works of life. Researchers and experts have developed various architectures to simulate real life scenarios. The paper discussed different algorithms of deep learning such as the RNNs, CNNs, GANs, RBMs, DBNs, LSTM, GRUs, etc, and the works of different researchers were also reported. It was discovered that deep learning has made immense contributions to human development. For instance, it has been explored in speech recognition where different speech models have been trained to interact with humans. Image classifications and recognitions models have been developed using the CNNs. Various large language models like the GPT, Dale-2 have made immense contributions for human interactive learnings, adaptive learnings, etc. To further ensure the deep learning model run faster, high computing Graphic Processing Unit (GPU) are available to run various deep learning model. The conclusion drawn from this paper is that CNNs, RNNs, GANs are some of the deep learning models

that have found great contributions in various fields of endeavour.

ACKNOWLEDGMENT

The authors wish to acknowledge the Covenant University Centre for Research, Innovations and Discovery (CUCRID) and Google for providing fund towards the publication of this study.

REFERENCES

[1] Alzubi, J., Nayyar, A., Kumar, A. (2018). Machine learning from theory to algorithms: An overview. In Journal of Physics: Conference Series, 1142: 012012. https://doi.org/10.1088/1742-6596/1142/1/012012

[2] Adeyinka, A.A., Adebiyi, M.O., Akande, N.O., Ogundokun, R.O., Kayode, A.A., Oladele, T.O. (2019). A deep convolutional encoder-decoder architecture for retinal blood vessels segmentation. In Computational Science and Its Applications–ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, pp. 180-189. https://doi.org/10.1007/978-3-030-24308-1_15

[3] Aizenberg, I., Aizenberg, N.N., Vandewalle, J.P. (2013). Multi-valued and universal binary neurons: Theory, learning and applications. Springer Science & Business Media.

[4] Dechter, R. (1986). Learning while searching in constraint-satisfaction problems. AAAI, 178-185.

[5] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.J. (2013). Phoneme recognition using time-delay neural networks. In Backpropagation, pp. 35-61.

[6] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11): 2278-2324. https://doi.org/10.1109/5.726791

[7] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.

[8] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. https://doi.org/10.48550/arXiv.1409.1556

[9] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778.

[10] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9.

[11] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Bengio, Y. (2014). Generative adversarial nets. Advances in Neural Information Processing Systems, 27.

[12] Howard, A.G. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861. https://doi.org/10.48550/arXiv.1704.04861

[13] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520.

[14] Ma, N., Zhang, X., Zheng, H.T., Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient CNN architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 116-131.

[15] Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Adam, H. (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1314-1324.

[16] Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C. (2020). Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1580-1589.

[17] Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. Mechanical Systems and Signal Processing, 115: 213-237. https://doi.org/10.1016/j.ymssp.2018.05.050

[18] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8: 1-74. https://doi.org/10.1186/s40537-021-00444-8

[19] Rudolph, J., Tan, S., Tan, S. (2023). ChatGPT: Bullshit spewer or the end of traditional assessments in higher education? Journal of Applied Learning and Teaching, 6(1): 342-363.

[20] Benavides, E., Fuertes, W., Sanchez, S., Sanchez, M. (2020). Classification of phishing attack solutions by employing deep learning techniques: A systematic literature review. Developments and Advances in Defense and Security: Proceedings of MICRADS 2019, pp. 51-64. https://doi.org/10.1007/978-981-13-9155-2_5

[21] Trivedi, A., Pant, N., Shah, P., Sonik, S., Agrawal, S. (2018). Speech to text and text to speech recognition systems—A review. IOSR Journal of Computer Engineering, 20(2): 36-43. https://doi.org/10.9790/0661-2002013643

[22] Li, Z., Liu, F., Yang, W., Peng, S., Zhou, J. (2021). A survey of convolutional neural networks: Analysis, applications, and prospects. IEEE Transactions on Neural Networks and Learning Systems, 33(12): 6999-7019. https://doi.org/10.1109/TNNLS.2021.3084827

[23] Medeiros, E.F. (2023). Deep learning for speech to text transcription for the Portuguese language. Universidade de Évora.

[24] Radford, A., Kim, J.W., Xu, T., Brockman, G., Mcleavey, C., Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. https://github.com/openai/.

[25] Jia, Y., Johnson, M., Macherey, W., Weiss, R.J., Cao, Y., Chiu, C.C., Ari, N., Laurenzo, S., Wu, Y. (2019). Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7180-7184. https://doi.org/10.1109/ICASSP.2019.8683343

[26] Li, J., Lavrukhin, V., Ginsburg, B., Leary, R., Kuchaiev, O., Cohen, J. M., Gadde, R.T. (2019). Jasper: An end-to-end convolutional neural acoustic model. arXiv preprint

arXiv:1904.03288. https://doi.org/10.48550/arXiv.1904.03288
[27] Barrault, L., Chung, Y.A., Meglioli, M.C., Dale, D., et al. (2023). SeamlessM4T-Massively Multilingual & multimodal machine translation. arXiv preprint arXiv:2308.11596. https://doi.org/10.48550/arXiv.2308.11596
[28] Selvaganapathy, S.G., Nivaashini, M., Natarajan, H.P. (2018). Deep belief network based detection and categorization of malicious URLs. Information Security Journal, 27(3): 145-161. https://doi.org/10.1080/19393555.2018.1456577
[29] Moraboena, S., Ketepalli, G., Ragam, P. (2020). A deep learning approach to network intrusion detection using deep autoencoder. Revue d'Intelligence Artificielle, 34(4): 457-463. https://doi.org/10.18280/ria.340410
[30] Bowman, S., Potts, C., Manning, C.D. (2015). Recursive neural networks can learn logical semantics. In Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, pp. 12-21.
[31] Socher, R., Lin, C.C., Manning, C., Ng, A.Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 129-136.
[32] Guo, Q., Yu, Z., Wu, Y., Liang, D., Qin, H., Yan, J. (2019). Dynamic recursive neural network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5147-5156.
[33] Khonde, S.R., Virnodkar, S.S., Nemade, S.B., Dudhedia, M.A., Kanawade, B., Gawande, S.H. (2024). Sentiment analysis and stock data prediction using financial news headlines approach. Revue d'Intelligence Artificielle, 38(3): 999-1008. https://doi.org/10.18280/ria.380325
[34] Safarzadeh, V.M., Jafarzadeh, P. (2020). Offline persian handwriting recognition with CNN and RNN-CTC. In 2020 25th International Computer Conference, Computer Society of Iran (CSICC), Tehran, Iran, pp. 1-10. https://doi.org/10.1109/CSICC49403.2020.9050073
[35] Limbu, S.H. (2020). Direct speech to speech translation using machine learning. http://www.teknat.uu.se/student.
[36] Mhaouch, A., Fradi, M., Gtifa, W., Ben Abdelali, A., Machhout, M. (2024). Deep learning based recurrent neural network model for stress detection in EEG signals. Revue d'Intelligence Artificielle, 38(3): 979–985. https://doi.org/10.18280/ria.380323
[37] Elango, P., Arthanareeswaran, A. (2024). BT detection using improved whale optimization and convolutional neural networks. Revue d'Intelligence Artificielle, 38(3): 815-823. https://doi.org/10.18280/ria.380308
[38] Qasim, A.N., Alani, S., Mahmood, S.N., Mohammed, S. S., Aziz, D.A., Ata, K.I.M. (2024). Enhancing brain stroke detection: A novel deep neural network with weighted binary cross entropy training. Revue d'Intelligence Artificielle, 38(3): 777-785. https://doi.org/10.18280/ria.380304
[39] Agbaje M., Afolabi, O. (2024). Neural network-based cyber-bullying and cyber-aggression detection using Twitter(X) text. Revue d'Intelligence Artificielle, 38(3): 837-846. https://doi.org/10.18280/ria.380310
[40] Napte, K., Mahajan, A. (2022). Deep learning based liver segmentation: A review. Revue d'Intelligence Artificielle, 36(6): 979-984. https://doi.org/10.18280/ria.360620
[41] Krichen, M. (2023). Convolutional neural networks: A survey. Computers, 12(8): 151. https://doi.org/10.3390/computers12080151
[42] Lindsay, G.W. (2021). Convolutional neural networks as a model of the visual system: Past, present, and future. J Cogn Neurosci, 33(10): 2017-2031. https://doi.org/10.1162/jocn_a_01544
[43] Orovwode, H., Oduntan, I.D., Abubakar, J. (2023). Development of a sign language recognition system using machine learning. In 2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Durban, South Africa, pp. 1-8. https://doi.org/10.1109/icABCD59051.2023.10220456
[44] Olayiwola, J.O., Adejoju, J.A. (2023). Maize (Corn) leaf disease detection system using convolutional neural network (CNN). In 23rd International Conference on Computational Science and Its Applications, Athens, Greece, pp. 309-321. https://doi.org/10.1007/978-3-031-36805-9_21
[45] Decelle, A., Furtlehner, C. (2021). Restricted Boltzmann machine: Recent advances and mean-field theory. Chinese Physics B, 30(4): 40202. https://doi.org/10.1088/1674-1056/abd160
[46] Ghojogh, B., Ghodsi, A., Karray, F., Crowley, M. (2021). Restricted boltzmann machine and deep belief network: Tutorial and survey. arXiv preprint arXiv:2107.12521. https://doi.org/10.48550/arXiv.2107.12521
[47] Kadhim, Q.K., Alwan, O.F., Khudhair, I.Y. (2024). Deep Learning methods to prevent various cyberattacks in cloud environment. Revue d'Intelligence Artificielle, 38(3): 893-900. https://doi.org/10.18280/ria.380316

NOMENCLATURE

w	weight of a neuron
b	bias of a neuron
x	input to a neural network
y	output of a neural network
f	activation function

Greek symbols

σ	steepness parameter