Chapter

Data Processing Using Artificial Neural Networks

Wesam Salah Alaloul and Abdul Hannan Qureshi

Abstract

The artificial neural network (ANN) is a machine learning (ML) methodology that evolved and developed from the scheme of imitating the human brain. Artificial intelligence (AI) pyramid illustrates the evolution of ML approach to ANN and leading to deep learning (DL). Nowadays, researchers are very much attracted to DL processes due to its ability to overcome the selectivity-invariance problem. In this chapter, ANN has been explained by discussing the network topology and development parameters (number of nodes, number of hidden layers, learning rules and activated function). The basic concept of node and neutron has been explained, with the help of diagrams, leading to the ANN model and its operation. All the topics have been discussed in such a scheme to give the reader the basic concept and clarity in a sequential way from ANN perceptron model to deep learning models and underlying types.

Keywords: ANN, artificial neural network, node, network training, gradient descent, deep learning

1. Introduction

Artificial Intelligence (AI) is the knowledge domain that targets the development of computer systems to solve problems by giving them cognitive powers for performing tasks that usually require human intelligence. Hence, simulation of human intelligence, with computer programing and technologies, is the main objective of AI. Whereas, machine learning is one of the branches of AI, in which computer systems are programmed based on the data and type of input. Machine learning (ML) gives the capability to AI for solving problems based on available data. Likewise, artificial neural network (ANN) is an evolved method of ML algorithms, developed on a concept of imitating the human brain [1–3].

A single neuron is considered as a cell, processing electrochemical signals or nerve impulses, and the human brain is a complicated network of neurons that transfers information, with the help of various interlinked neurons. ANN models are considered as most popular among AI models because of their architecture, which is the collection of neurons linked with other neurons in various layers. ANN is non-linear and complex systems of neurons and neuron is a mathematical unit [4].

Literature depicts that ML, ANN and deep learning (DL) falls under the pyramid of AI and shown in **Figure 1**. Under ANN, DL has gained much importance among researchers. DL is a complex network set of ANN with various layers of processing, which improves the results by developing high levels of insight. DL methodologies

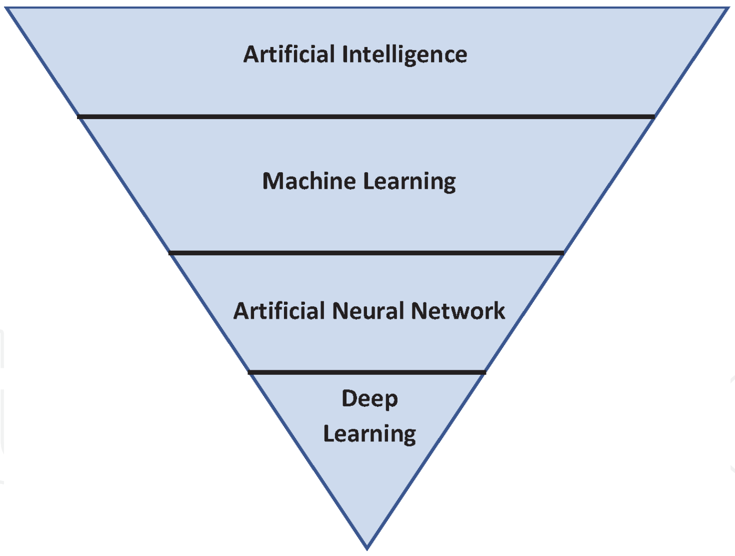


Figure 1.
AI pyramid.

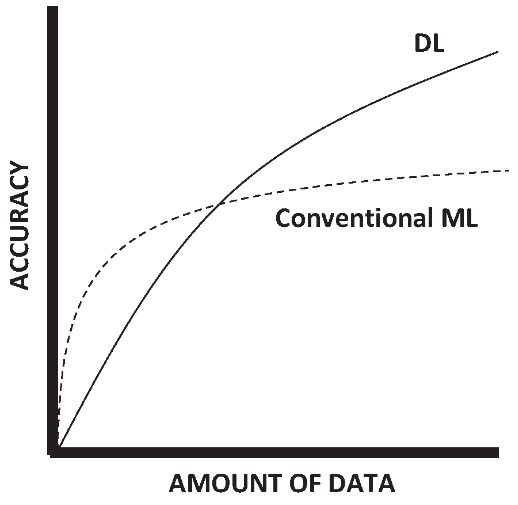


Figure 2.
Comparison between DL and conventional ML.

are popular due to their computational powers and handling of large data sets, and this makes them more attractive than conventional methods.

Past studies illustrated the comparison between DL and conventional ML methods for effective outputs, with the help of graphical representation, as shown in **Figure 2**. **Figure 2** illustrates the behaviour of curves, for DL and conventional ML, by comparing the accuracy of results (outputs) against the amount of data (input). The graph shows that the result accuracy of conventional ML methods is better for limited data, but it decreases as the amount of data is increased. Instead, the result accuracy of DL improves for large data sets, due to the presence of a vast neural network than conventional ML, hence, making DL more famous. DL is usually used for complicated tasks, such as image classification, image recognition, and handwriting identification [1, 3].

2. History of ANN

The origins of all the work on ANN are in neurobiological studies that date back to about a century ago. A brief overview of evolution in ANN and significant milestones are shown in the timeline, as shown in **Figures 3** and **4**.

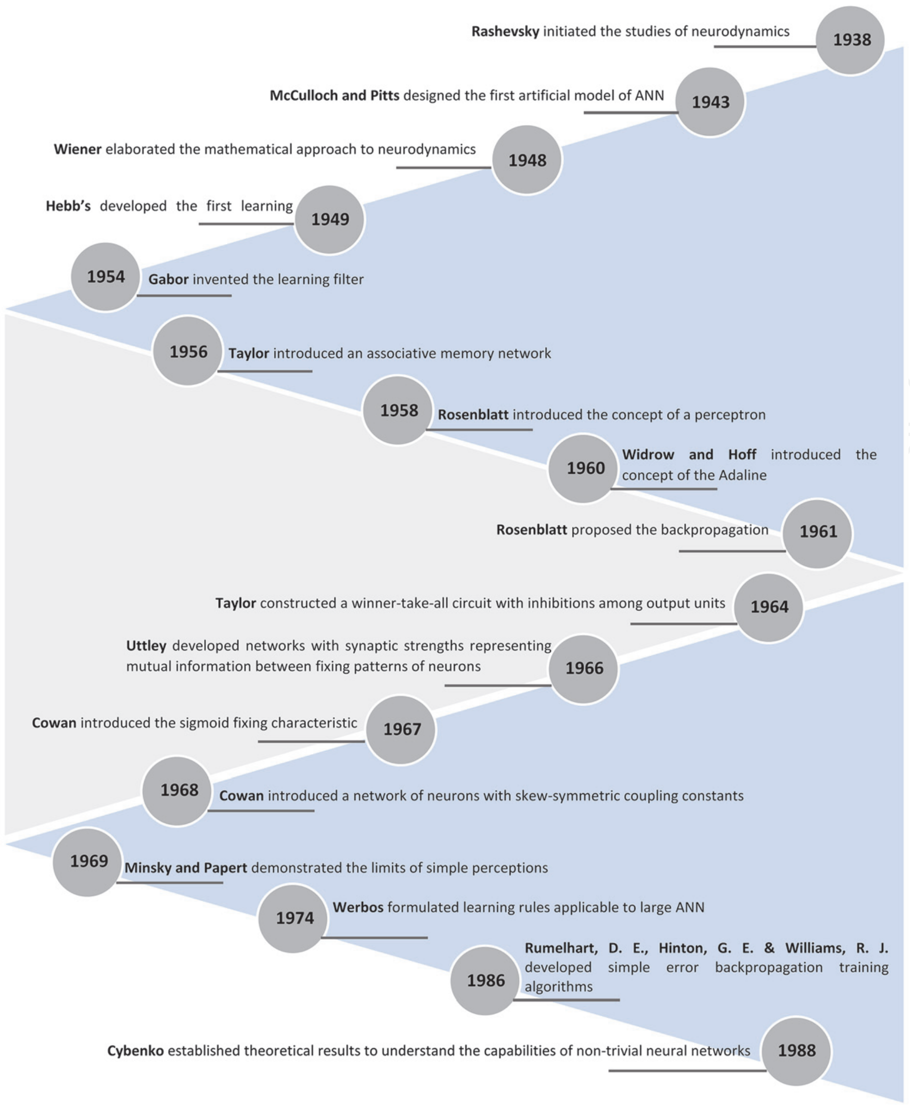

1938
Rashevsky initiated the studies of neurodynamics
1943
McCulloch and Pitts designed the first artificial model of ANN
1948
Wiener elaborated the mathematical approach to neurodynamics
1949
Hebb's developed the first learning
1954
Gabor invented the learning filter
1956
Taylor introduced an associative memory network
1958
Rosenblatt introduced the concept of a perceptron
1960
Widrow and Hoff introduced the concept of the Adaline
1961
Rosenblatt proposed the backpropagation
1964
Taylor constructed a winner-take-all circuit with inhibitions among output units
1966
Uttley developed networks with synaptic strengths representing mutual information between fixing patterns of neurons
1967
Cowan introduced the sigmoid fixing characteristic
1968
Cowan introduced a network of neurons with skew-symmetric coupling constants
1969
Minsky and Papert demonstrated the limits of simple perceptions
1974
Werbos formulated learning rules applicable to large ANN
1986
Rumelhart, D. E., Hinton, G. E. & Williams, R. J. developed simple error backpropagation training algorithms
1988
Cybenko established theoretical results to understand the capabilities of non-trivial neural networks

Figure 3.
ANN evolution timeline (1938–1988).

Literature depicts that, in the 1980s, very few researchers were working on deep NNs, and it gained popularity in the early 1990s. Since then, a large number of research articles have been published on applications of ANN and this journey is on-going. The few significant milestones, after 1990, regarding ANN evolution is shown in **Figure 4** [5–10].

3. Basic architecture of ANN

The architecture of ANN is stimulated by the framework of biological neurons, like in the human brain. The human brain is the composition of a vast number of the interlinked neurons forming a network. A neuron is like a cell, and each neuron executes a simple task, i.e., response to an input signal. Likewise, the ANN is a framework of interlinked nodes, similar to neurons, forming a network model. Hence in ANN, several artificial neurons are interlinked and become a robust computer-based tool that can handle large amounts of data to execute enormously simultaneous calculations using input data. ANN operations are not based on explicit rules and outputs are generated by trial and error procedures through sequential computations. The ANN is also classified as 'connectionism' because the given data is not conceded from neuron to neuron, but it is encoded in the complicated interconnected network of neurons, unlike the traditional computers [2, 11, 12].

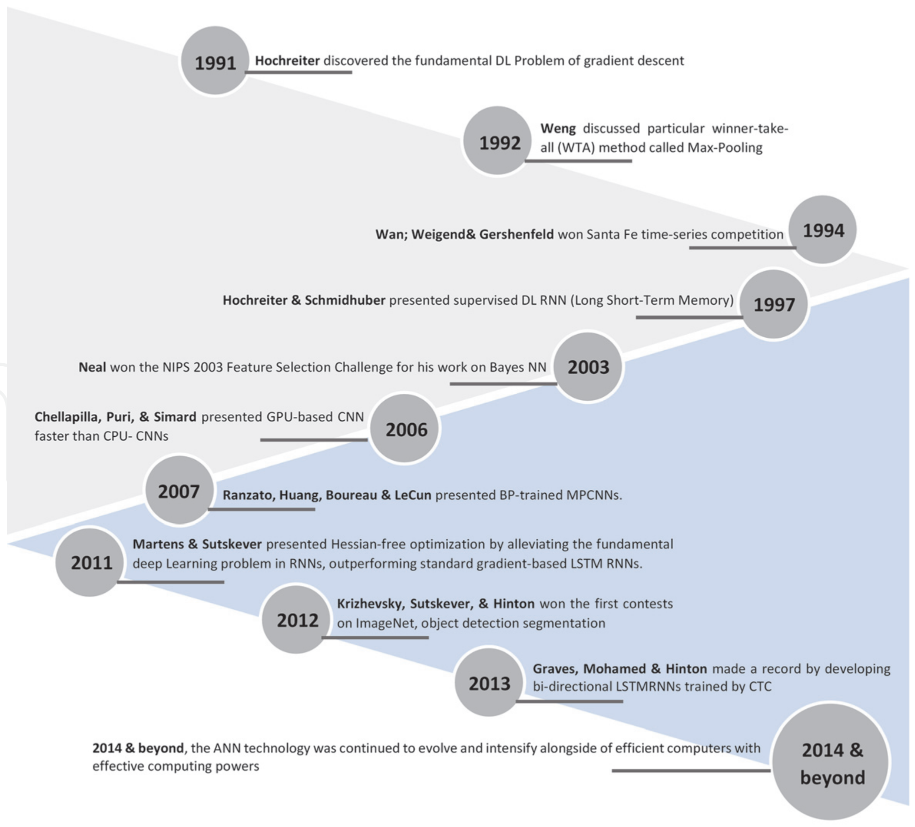

1991 Hochreiter discovered the fundamental DL Problem of gradient descent
1992 Weng discussed particular winner-take-all (WTA) method called Max-Pooling
Wan; Weigend& Gershenfeld won Santa Fe time-series competition 1994
Hochreiter & Schmidhuber presented supervised DL RNN (Long Short-Term Memory) 1997
Neal won the NIPS 2003 Feature Selection Challenge for his work on Bayes NN 2003
Chellapilla, Puri, & Simard presented GPU-based CNN faster than CPU- CNNs 2006
2007 Ranzato, Huang, Boureau & LeCun presented BP-trained MPCNNs.
2011 Martens & Sutskever presented Hessian-free optimization by alleviating the fundamental deep Learning problem in RNNs, outperforming standard gradient-based LSTM RNNs.
2012 Krizhevsky, Sutskever, & Hinton won the first contests on ImageNet, object detection segmentation
2013 Graves, Mohamed & Hinton made a record by developing bi-directional LSTMRNNs trained by CTC
2014 & beyond, the ANN technology was continued to evolve and intensify alongside of efficient computers with effective computing powers 2014 & beyond

Figure 4.
ANN evolution timeline (after 1988).

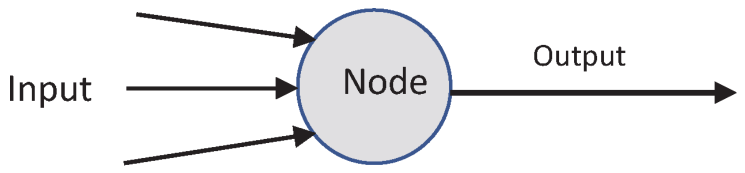


Figure 5.
Basic node model.

To comprehend the basic structure of ANN, firstly, the understanding of 'node' is necessary. The generic model for a node is shown in **Figure 5**.

Each node receives various inputs through connections and transfers it to adjacent nodes. **Figure 6** represents the general model of ANN, which is stimulated by a biological neuron.

The nodes are arranged and organised into linear arrays known as layers. **Figure 6** shows that there are three layers in ANN called the input layer, the output layer and the hidden layer.

In the input layer $X_1, X_2, X_3, \ldots X_n$ signifies several inputs to the network. Whereas, $W_1, W_2, W_3, \ldots W_n$ are known as connection weights, which shows the strength of a particular node. In ANN, weights are considered as the most significant factors as these are numerical parameters that determine the effect of neurons to each other and also impact the output, by converting the input.

In the ANN, the processing part is performed in the hidden layer. The hidden layer executes two operational functions, i.e., summation function and transfer function, also known as an activation function. The summation function is the first step, and in this part, each input (X_i) to ANN is multiplied by its respective weight (W_i) and then, the products $W_i.X_i$ is cumulated into the summation function $\xi = \Sigma W_i.X_i$. 'B' is a bias value; this parameter is used to regulate the output of the neuron in association with the weighted sum of the inputs. This process is denoted as Eq. (1):

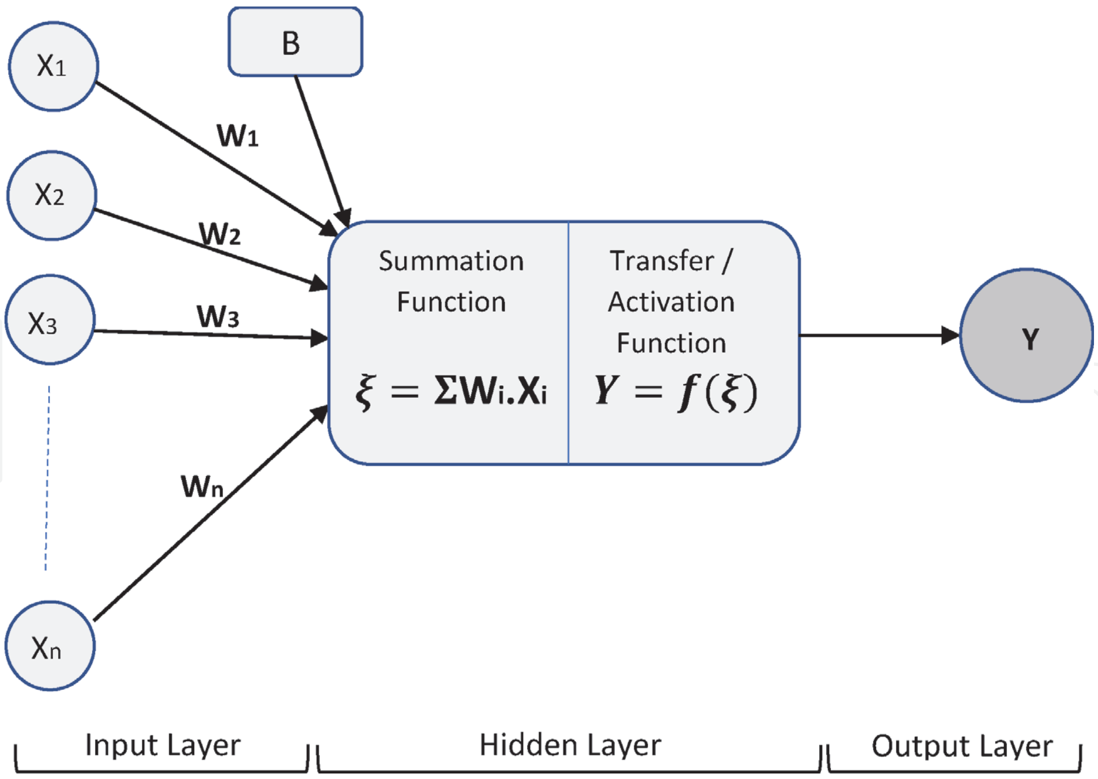


Figure 6.
Generic ANN model.

$$Output = \mathbf{\Sigma}(Weights \times Inputs) + Bias \tag{1}$$

The activation function is the second step; which converts the input signal, received from the summation function module and transformed it to an output of a node for an ANN model [1–3, 12, 13].

Generally speaking, each ANN has three main components, i.e., node character, network topology and the learning rules. The node character controls the processing of signals by determining the associated number of inputs and outputs, the associated weight for each input and output and the activation function, for each node. Learning rules establish the initiation and adjustment of weights. Whereas, the network topology defines the ways the nodes will be connected and organised (details are discussed in Section 3.2). The operation of the ANN model is computing the output of all the neurons, which is an entirely deterministic calculation [1, 2].

3.1 The activation function

An activation function is a mathematical function. In simple words, it receives the output of the summation function as an input and converts that into the final output of a node with the help of ANN processing.

There are different types of activation functions, but non-linear functions are more popular than the linear function. A linear function is just a polynomial of one degree, and it is considered as single-layer ANN model has less power and limited complexity to process complicated data. Therefore, non-linear activation functions are mostly included in designing of ANN models for solving complex problems and this unique quality makes ANN true universal function approximators.

The activation function uses the value $\xi = \Sigma W_i.X_i$ as an input for processing and controlling the input X_i for activation of the neuron. The most commonly known activation functions [1, 12–15] are shown in **Table 1**.

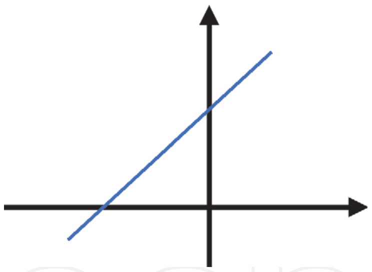
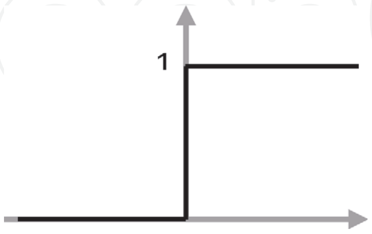
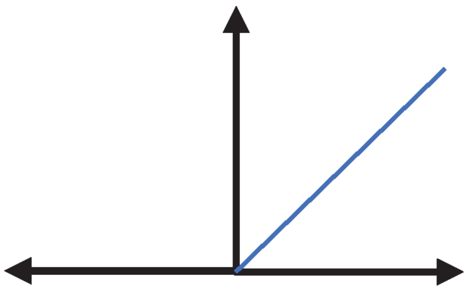
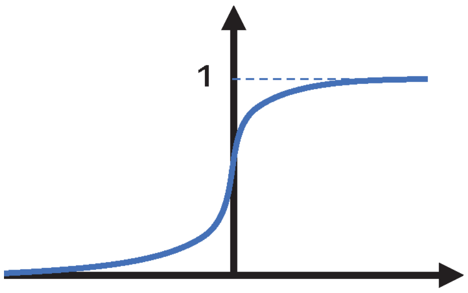
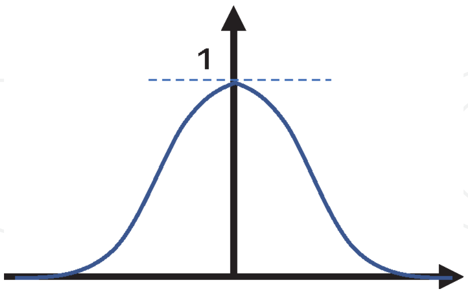
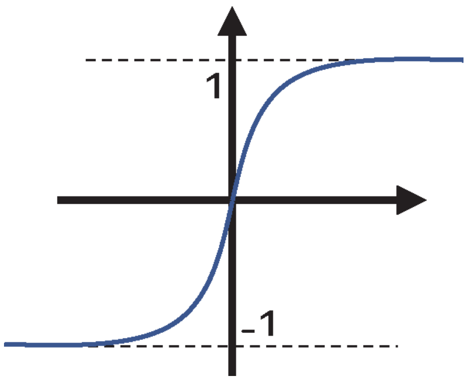
Transfer functions	**Graphical presentation**	**Numerical equation**	**Remarks**
Linear		$Y = f(\xi) = \xi$	Output = Input. Range $(-\infty, +\infty)$
Unit step		$f(\xi) = 0$ if $\xi < 0$ $f(\xi) = 1$ If $\xi \geq 0$	Useful for binary schemes. Range (0,1)
Rectified linear unit (ReLU)		$f(\xi) = 0$ if $\xi < 0$ $f(\xi) = \xi$ If $\xi \geq 0$	Most popular activation function since 2015. Range $(0, \infty)$
Sigmoid		$f(\xi) = \frac{1}{1+e^{\xi}}$	Commonly used function. Range (0, 1)
Gaussian		$f(\xi) = e^{-(\xi)^2}$	Named after the mathematician Carl Friedrich Gauss Range (0,1]
Hyperbolic tangent		$f(\xi) = \frac{2}{1+e^{-2\xi}}$ - 1	Alternative to sigmoid function. Range $(-1, 1)$

Table 1.
Activation functions.

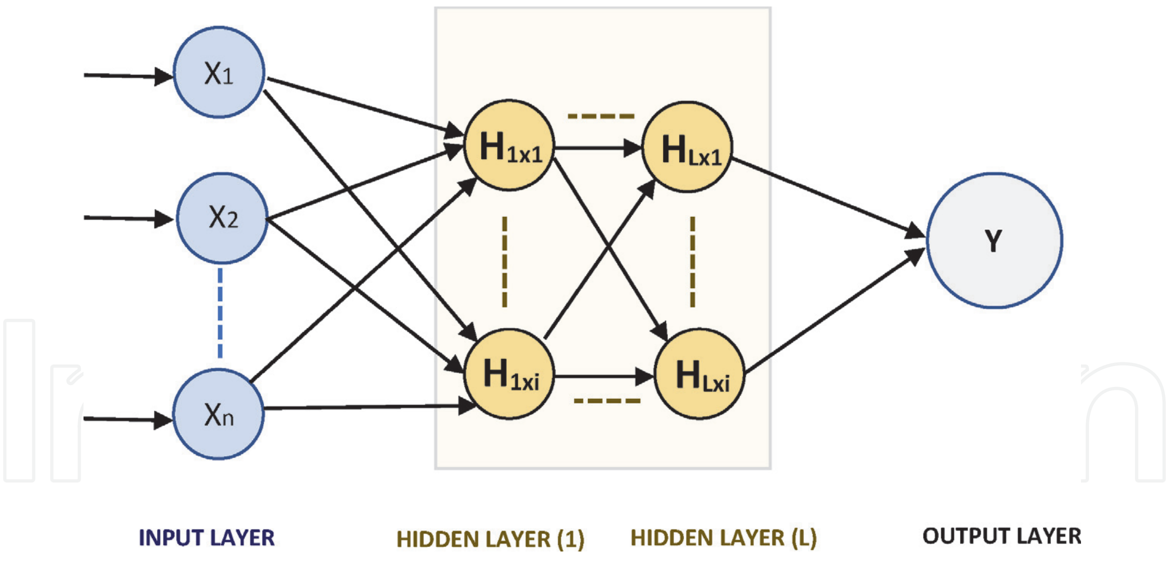


Figure 7.
Conceptual model for ANN topology.

3.2 Network topology

The nodes are arranged and organised into linear arrays known as layers. The interconnecting network model, between the nodes of ANN, with each other, is called the topology (or architecture). ANN is composed of input layers, hidden layers and output layers, as already discussed in **Figure 6**. Also, the hidden layers can be from none to numerous, based on the model-complexity. Each layer is a combination of many nodes, and these nodes, based on some properties, can be grouped in layers. A single-layer ANN, with a single output, is known as Perceptron. A conceptual model for layers and ANN topology is shown in **Figure 7**. **Figure 7** shows n number of data entries in the input layer as X_1, X_2, … . X_n. Also, it can be seen that there is L number of hidden layers in the ANN model. Whereas, there are i number of nodes in each hidden layer. The notations 1×1, $1 \times i$, $L \times 1$ and $L \times i$, on each node giving its information, expressing 'L' as (hidden) layer number, i.e., from 1 to L and 'i' as node number, i.e., from 1 to i. Y is the output for the mentioned ANN model.

Designing of network topology is based on following factors; (1) the number of nodes in each layer, (2) the number of layers in the network and (3) the connected path among the nodes [1, 2, 12].

3.2.1 Perceptron and multi-layer architectures

A single-layered ANN, with a single output, is known as the perceptron. The perceptron mostly uses the step function, in which, if the computed sum of the inputs transcends a threshold point, the output is 1; otherwise, it is 0.

Multi-layer perceptrons (MLPs) are the most commonly used architecture for ANN. Composition of MLPs contains layers of neurons with an input layer, an output layer, and the hidden layer (at least one). The layers of the perceptron are interlinked with each other by developing a multi-layered architecture, and this makes the model essentially complex for the ANN processing. The MLP terminology is originated from perceptron neural networks, but its problem-solving capabilities makes it unique [1, 14].

3.3 Connection types between nodes

The connections between nodes of ANN are classified into two categories: (1) the feedforward network, and (2) the feedback network or recurrent network.

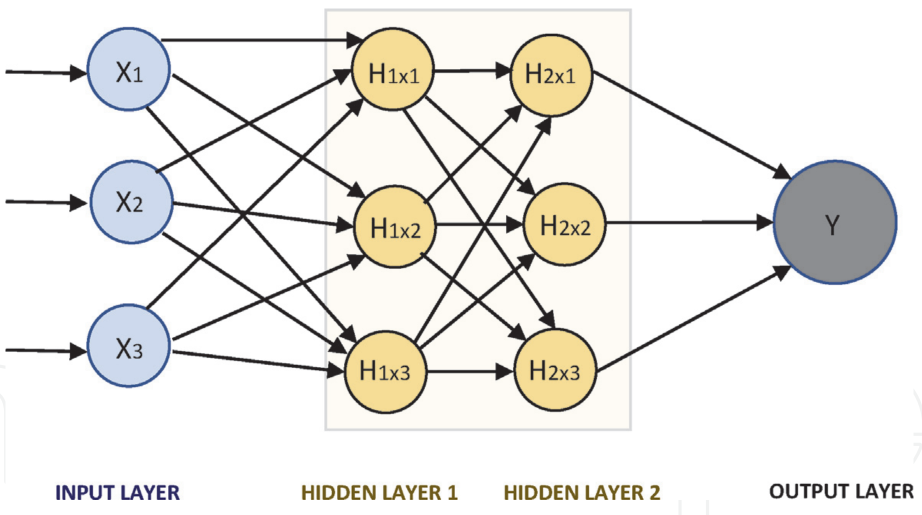


Figure 8.
Feedforward network connection.

3.3.1 Feedforward networks

Feedforward network is a one-way connection having no loop backwards. They are static in nature as their signal travels in one way only. **Figure 8** is a model example of feedforward networks.

3.3.2 Feedback networks

In feedback network, nodes have backward connected loops, and in these connections, the output of the nodes can be the input to the same level or previous nodes. Unlike the feedforward network, the feedback networks are dynamic. In feedback networks, signals are transmitted in forward as well as in backward directions [16]. Feedback process occurs when the output (partial or full) is channelled back into the input of a network as part of a repeated cause-and-effect process [17]. In the feedback network, a single input generates a series of outputs cycles until it reaches an equilibrium point. Equilibrium point refers to minimum error, i.e., for each predicted output if the error is enormous then, the output is routed back, and parameters (weights and biases) are modified until the error becomes minimum [18]. **Figure 9** shows the ANN model for feedback network

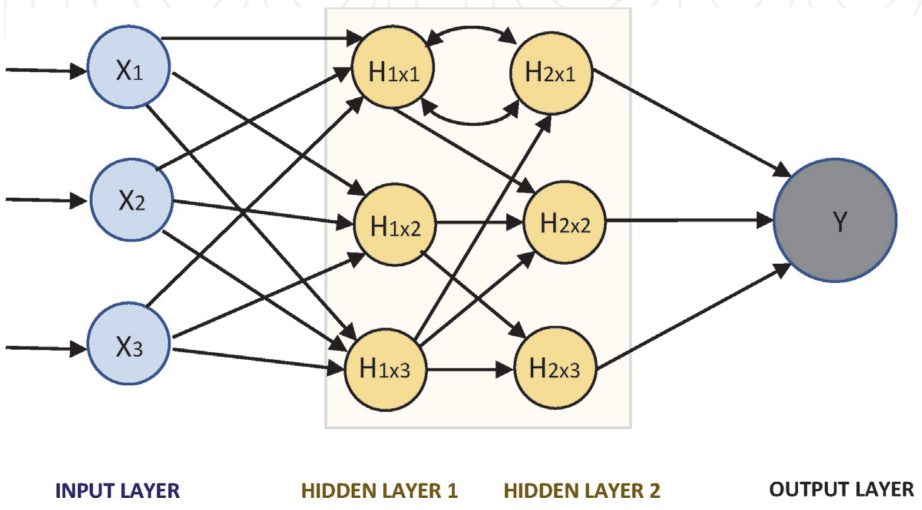


Figure 9.
Feedback network connection.

connections. It can be observed that node H2x1 is sending the information back to node H1x1 and the cycle goes on until the output will reach an equilibrium state, i.e., with minimum error. In a feedback network, there exists at least one interconnected path that drives it back to the starting neuron. It may cause a delay in specific time units, and this interconnected path is called a cycle [1, 2, 12]. This process will be better understood, after going through the next section.

4. Training of ANN (learning process)

The training of the ANN is accomplished through a learning process. While in the training process, weights are modified for attaining required results. In the training process, some sample data is processed to the network and weights are modified to attain better approximation of the desired output.

The learning process is mostly classified into two categories: (1) supervised learning, and (2) unsupervised learning.

4.1 Supervised learning

In supervised learning, a training set is presented to the model. The training set constitutes of input examples and corresponding target outputs. The inputs are noted for the response of the network, and the weights between with networks are adjusted for error reduction, for the attainment of the desired output. The network follows successive iterations during this process until the computed result converges to the correct one. Construction of the training set requires special consideration. A training set is considered an ideal one, and it should be giving a better representation of the underlying model. Otherwise, a reliable model with desirable results cannot be achieved with an unrepresentative training set.

In the supervised learning process, the networks are trained first before its operation in a model for predictive outputs. Significantly, when the network starts computing the intended outputs with the series of inputs, with fixed weights, then the ANN model can be set for the required operation. Few of the well-known algorithms with a supervised learning method are the Adaline (used for binary data), the Perceptron (used for continuous data), and the Madaline (developed from the Adaline).

4.1.1 Reinforcement learning

Reinforcement learning is a particular case scenario of supervised learning. It is, when the external environment only checks for the information for acceptance and rejection, instead of indicating the correct output. In this process, the well-performing and the most active neuron connections for the input are strengthened over successive iterations. Few of the renown algorithms of reinforcement learning are the Boltzmann machine, the learning vector quantisation, and Hopfield networks.

Supervised ANN models have many applications for image classification, plant control, forecasting, prediction, robotics, ECG signals classification and many more [19–21].

4.2 Unsupervised learning

Unsupervised learning does not follow a training set or a targeted output approach. Instead, it trails the input data pattern of the underlying model. In this process, the ANN model adjusts its weights, against the supplied inputs, thus producing outputs similar to inputs. The model, without any outer support, recognises the patterns and differences in the inputs. In this process, the clusters are formed, each cluster consists of a group of several weights, in such a way that related input path results in a similar output. If any new pattern is detected during the iteration process, it is classified as a new cluster.

Autoencoders, Hebbian Learning, Deep Belief Nets, Self-Organising Map, Generative Adversarial Networks, and Algebraic Reconstruction Technique (ART) are the few most renown algorithms for unsupervised learning. Unsupervised ANN models are used in diagnosing diseases, image segmentation and many more. Unsupervised algorithms have become very useful and powerful tools in segmentation of magnetic resonance images for detection of anomalies in the body systems [1, 2, 4, 12, 14, 22–24].

5. Mapping by ANNs

The primary reason for ANN popularity is due to approximated data output. There are five main steps for the approximation function in the ANN model, as given below.

5.1 Data pre-processing

In data pre-processing, the appropriate predictors are selected as inputs before processing to a network for mapping. There are three general processes in data pre-processing, mentioned as follows:

a. *Standardising*: The input values are rescaled to a uniformed scale.

b. *Normalising*: It normalises a vector to have unity variance and zero mean value.

c. *Principal component analysis*: This process replaces the groups of related variables by new unrelated variables by detecting linear dependencies between them.

5.2 Selection of network architecture

A network architecture comprises several hidden neurons, the number of hidden layers, the flow of data, the way neurons are interconnected, and specific transfer functions. Recurrent neural networks, multi-layer perceptron (MLP), probabilistic neural networks, radial basis function networks, generalised regression neural networks and time-delay neural networks are the few of the renown architectures.

5.3 Network training

About function mapping, the training process is known as the calibration of the network through input and out pairs. During the training process, ANN might suffer from the overfitting and underfitting. The overall performance of the network decreases because of these two mentioned factors. This unfitting of the network, during the training process, can be managed by increasing the number of epochs, but it may result in network overfitting if the number of epochs is more significant than a required number. Epoch is defined as a process of providing one pass or iteration of input through the network and modifying the weights. The optimal number of epochs can be determined by the comparison of training error and model testing procedure.

5.4 Simulation

Simulation is the ultimate goal of applying ANN networks. It is the representation of predicted output data for an ANN model.

5.5 Post-processing

There are three types of sets in which sample data is distributed: (i) the training set, (ii) the validation set, and (iii) the testing set. The training set is used to train the ANN model; it is a set of sample data that is used to modify or adjust the weights in the ANN to produce the desired outcome. The validation set is used to inform the ANN when training is to be terminated (when the minimum error point is achieved). The test set provides an entirely independent way of examining the precision of the ANN. The test set is a set of sample data that is used for the evaluation of the ANN model. A rule of thumb for this random split regarding percentage is 70, 15, 15%, respectively [3, 12, 14].

The post-processing comprises of all the tests, which are applied on a specific network for the validation of results, also, to analyse, describe, and to improve its final performance. The comparison of results is achieved by using three different statistics. The first one is the root-mean-square error (RMSE), and it is described in Eq. (2):

$$RMSE = \sqrt{\frac{\sum_i^n (obs_i - est_i)^2}{n}} \tag{2}$$

The second statistical factor is percentage volume error (%VE), which is the measuring of the absolute relative bias error of estimated values. It is formulated as Eq. (3):

$$\%VE = \frac{\sum_{i=1}^n \left(\frac{obs_i - est_i}{obs_i}\right)}{n} \tag{3}$$

whereas, est_i = ith estimated variable, obs_i = ith observed data, and n = number of observed values.

The third statistical factor is the correlation, and it is used in the measuring of the linear correlation coefficient between the predicted and observed data.

In case of unsatisfactory results in the post-processing, modification can be made in the following: (1) weights and biases, (2) number of hidden neurons, (3) transfer functions, and (4) number of hidden layers [4, 25].

6. Gradient descent

The term ‘gradient descent’ is a combination of two words the ‘gradient’, which means a slope and the ‘descent’, which means to incline. Therefore, with gradient descent, the slope of gradients is descended to find the lowest point with the smallest error. It is an iterative process until the correction of the error in the ANN learning model. It is defined as during the backpropagation in the ANN model, the process of iteration keeps updating biases and weights with the error times derivative of the activation function. The steepest descent step size is substituted by a similar size from the previous step.

A gradient is the derivative of the activation function, as shown in **Figure 10**.

The primary purpose of using gradient descent is to find the overall cost minimum at each step, with the lowest error. Also, at this point, model predictions are more reliable because of upright fit data. Evaluation of slope can be done with the help of **Figure 11**, and Eq. (4) can be derived.

$$\Delta x_i = -\alpha \frac{dy}{dx_i} \tag{4}$$

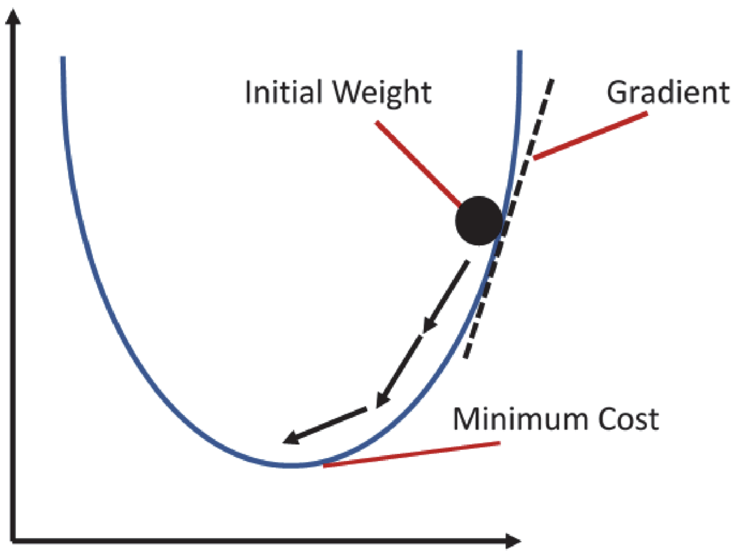


Figure 10.
Gradient descent.

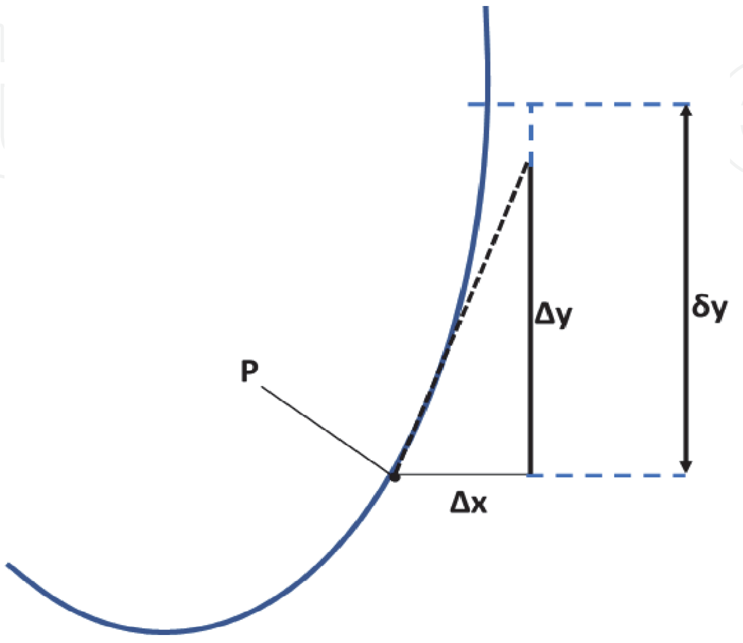


Figure 11.
Slope computation.

whereas, α = learning rate and dy/dx_i, also known as the partial derivative of y with respect to x_i. For gradient descent, this equation can be used for each variable when $\delta y < 0$ (δ is a partial derivative).

Gradient descent can be achieved either for the stochastic or full batch. In stochastic, gradient descent performs calculation for gradient by taking a single sample. Whereas, in full batch, the gradient is calculated for the full training dataset. One of the advantages of stochastic gradient descent is the fast calculation of gradients [1, 13, 23].

6.1 Training algorithm by delta rule

The biases and weights are the parameters of the network that are required to be adjusted before operating an ANN. These parameters can be modified by using either supervised or unsupervised approach for any ANN model. For training purpose, the supervised learning process is generally considered for determining biases and weights of an ANN network. The supervised training process of an ANN network could be attained by using delta rule. The delta rule is expressed as W_{ij} with the help Eqs. (5)–(7), as shown:

$$W_{ij}^{new(L)} = W_{ij}^{old(L)} + \alpha\left(-\frac{\partial_{e_p}}{\partial_{W_{ij}^{(L)}}}\right) \tag{5}$$

$$E = \frac{1}{n}\sum\nolimits_{p=1}^{n} e_p \tag{6}$$

$$e_p = \left(t_p - y_p\right) \tag{7}$$

whereas, n = the number of pairs of data, W = the weight of the link between the ith neuron to the jth neuron in the Lth layer, E = the average error of estimation, t_p = target output, y_p = simulated output, α = learning rate, the value of which is selected between 0 and 1 experimentally.

The backpropagation algorithm is mostly used for the application of delta rule for the training process of an ANN. The mathematical expression of delta rule is changed to computational relation because of the backpropagation algorithm, which can be applied through an iterative process. This process provides a way to the gradient for determining of the minimum error function, and it is efficiently calculated by using the chain rule of differentiation provided by the backpropagation algorithm. This characteristic makes this process to also be known as the generalised delta rule. In this algorithm, during each iteration, the network weights are shifted along with the negative of the gradient in the steepest descent direction of the performance function (epoch). For a certain weight in the Lth hidden layer, the chain rule gives Eq. (8):

$$\frac{\partial_{e_p}}{\partial_{W_{ij}}^{(L)}} = \frac{\partial_{e_p}}{\partial I_{pj}^{L}} \cdot \frac{\partial I_{pj}^{(L)}}{\partial w_{ij}^{(L)}} \tag{8}$$

This algorithm keeps the iterations continued until the expected output of network training is achieved. The basis for stopping the training process may be the minimum target value of performance function, the number of epochs and run time of the process; this is known as stopped training. The above mentioned equations lead to the following weight calculating Eqs. (9) and (10):

For the last layer

$$W_{ij}^{new} = W_{ij}^{old} + \alpha\delta_{pj}y_{pi}^{(L-1)} \quad (9)$$

For the hidden layer

$$W_{ij}^{new} = W_{ij}^{old} + \alpha\delta_{pj}^{(L)}y_{pi}^{(L-1)} \quad (10)$$

Following this procedure of training, based on the specific input vectors using the final derived weights and biases, the ANN model will be operated on sample data for initiation of simulation for the related outputs. The ANN training can be achieved either by batch training or incremental training. During the batch training process, the adjustment of biases and weights is attained after the presentation of all the inputs and targets. Whereas, during the incremental training, the adjustment of biases and weights is attained just after the presentation of individual input. In training, the process affects network performance. In the case of the low learning rate, the time required for learning the synaptic weights will be extremely long. On the other hand, if the set learning rate will be too high, this will tend the algorithm to oscillate, and the trained network performance will be reduced because the weight changes are too drastic. Therefore, the learning rate controls the convergence of the algorithm. These weight modifications can be applied after each pattern is completed, and these computed weight changes can be summed up to be applied to the network weights, as shown in Eq. (11):

$$\Delta w_{ij}^{l} = \sum_{p=1}^{n} \Delta w_{pij}^{L} \quad (11)$$

Usually, in dynamic networks, the inputs and targets are shown in sequence. In the adaptive learning process, the recent data, that is perceived before the time of simulation is considered as necessary as compared to all the data [4, 14, 26].

7. Deep learning

In the field of AI, deep learning (DL) has gained much popularity and trending for investigation domains. One of the foremost shortcomings of conventional machine learning is their inability to solve the selectivity-invariance problem, and because of this drawback, these methods have limited capability of data processing in their real state. Selectivity-invariance enables the model for the selection of those parameters that comprise of more information and disregard parameters with less information. This characteristic of DL, i.e., ability to overcome the selectivity-invariance dilemma, makes it more likeable among researchers and motivate them to the advancement of machine learning using the DL approach.

The architecture of DL is composed of various layers of trainable parameters, and this helps DL-based algorithms for excellent performance in machine learning and AI applications. DL algorithm is Deep Neural Networks (DNNs), and they usually use backpropagation optimised algorithms for end-to-end training. DNNs capability of selectivity-invariance extracts the compound features through successive layers of neurons equipped with differentiable, non-linear activation functions, and this provides a suitable platform for the backpropagation algorithm. A generic architectural model of DNNs is shown in **Figure 12**.

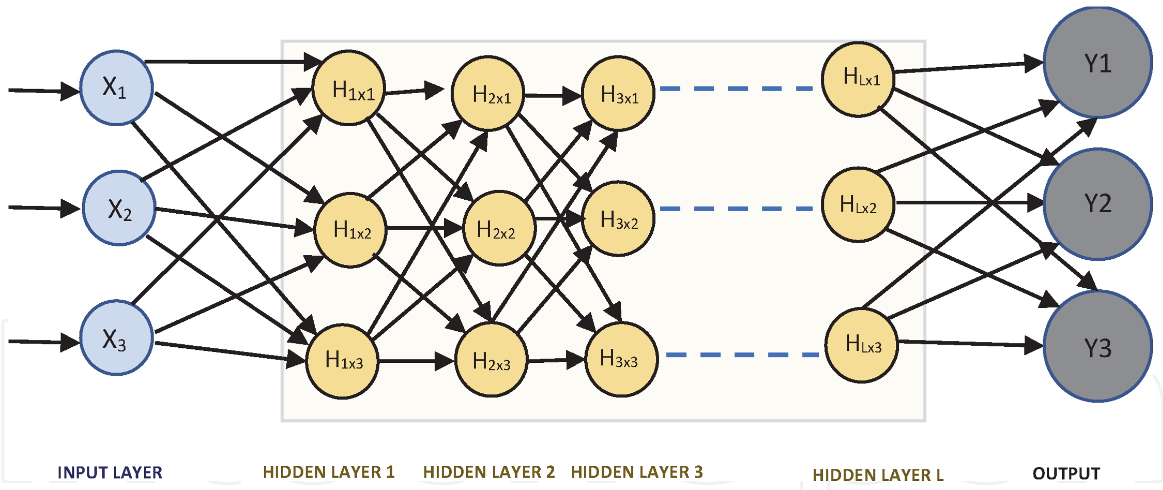


Figure 12.
DNNs generic model.

Figure 12 depicts a DNNs model with numerous hidden layers. The outer layer of DNN mostly uses the softmax module for the solution of most of the classification problems. The softmax formula is also known as normalised exponential, is given below in Eq. (12):

$$Y_i = \frac{exp\ (a_i)}{\sum_j exp\ (a_j)} \tag{12}$$

whereas, j is the set of output nodes, a_i is the net input to a particular output node, and Y_i is the value of output node between range (0, 1).

DNNs models with non-linear behaviour can go up to several abstractions of levels that helps in decision making by transforming original data into higher abstract levels. This process streamlines finding the solution for non-linear and complex functions. Basis of DL is automated learning of features that offer the facility of transfer learning and modularity. Unlike conventional machine learning, training of DL networks requires a large amount of data. Convolutional neural network (CNN) and recurrent neural network (RNN) are the renown deep networks [27, 28].

7.1 Convolutional neural network (CNN)

CNN is the popular DL methodology, based on the animal's visual cortex. CNNs are very much similar to ANN that can be observed as the acyclic graph in the form of a well-arranged collection of neurons. Although, in CNNs, the neurons in the hidden layers are only interconnected with a subset of neurons in the preceding layer, unlike regular ANN model. This rare type of interconnectivity enables CNN models to learn the discreet features on an object. CNN models are used for face recognition, scene labelling, image classification, document analysis and many more.

The police department of the Penang Island, Malaysia had installed more than 500 CCTV cameras around the Island and many of them were equipped with face recognition technology, which was developed by IBM. Their main objective was to control crime and capture the wanted criminals [29]. Likewise, in China Pharmaceutical University, to control the student attendance and class discipline the university management installed the facial recognition system in the campus, including the classrooms, labs, library and entrance gates. This overall improved the students' response towards academics [30]. Face recognition technology is based on deep CNN models. This process can be performed by using both supervised and unsupervised approaches but supervised methodologies are mostly preferred. Face recognition is performed by taking an input from video or image and detection is made by taking input to greyscale. The features in greyscale are applied one by one and compared with pixel values. The CNN models give high accuracy than past techniques by overcoming the problems, like light intensity and expressions, with the help of trained models using more training samples [31–33].

7.2 Recurrent neural network (RNN)

RNNs are used for the tasks that require consecutive sequential inputs for processing. Initially, training of RNNs was done by using backpropagation. RNNs approach utilises one factor of input, at a time, in sequence by keeping state vector in their hidden nodes, in which implicitly within nodes contains information of all the past value of factors of that sequence. RNNs are dynamic and fairly powerful systems, but during the training process the problem occurs as in gradients of backpropagation algorithm either would shrink or grow at every time step, ultimately they might disappear after many cycles. If we explore RNN, deep feedforward networks will be found having all layers sharing the same weight. RNN lags to the capability of storing information for a long time, and deficiency is known as long-term dependencies. To control this shortcoming, one approach has been introduced with explicit memory known as long short-term memory (LSTM). In this method, particular hidden nodes are used to store the information in the form of input data for a much higher time. LSTM is very much recognised for the better-quality performance in speech recognition systems [1, 27, 28].

Apple's Siri, Amazon's Alexa, Microsoft's Cortana, and Google's Assistant are the most popular voice recognizer tools and they are used for making a phone call, play reminders, alarms, provide driving directions and much more. The speech recognizers are developed on RNN networks, which are based on LSTM-RNN architecture. This gives the RNN models the ability to deal with long-distance patterns and makes them suitable for learning long-span relations. The models are trained end-to-end and output is attained [34, 35]. Other few applications of RNN models are keyphrase recognition, meteorological data updating, speech to text [35–38]. Massachusetts Institute of Technology (MIT) had performed an interesting simulated study on self-driving cars, and its framework was also being developed on the deep reinforced model [39].

8. Examples of ANN model using Python

8.1 Supervised ANN model

A simple ANN model was developed using Python. The model was designed by using supervised CNN methodology for image classification. Images were collected for training and validation purpose of the model for apples and oranges. For training purpose, 20 images were collected for each (apple and orange), making a total of 40 images. For validation purpose, 10 more images were collected for each, making a total of 20 images. The data for the supervised process, of the ANN model, was arranged in a specific way with a separate folder for each process, i.e., training and validation. In a folder named as 'Training', images of each fruit were placed separately in the folders having their name titles, i.e., 'Apple' and 'Orange', and same was done for 'Validation' folder. In the classification and prediction process, the model output was analysed, for the effectiveness of the results, against two parameters: (1) effect of increasing the number of epochs per run, and (2) the number of hidden layers.

8.1.1 Number of epochs per run

The effect of increasing the number of epochs on the model, for each run, is shown in **Table 2**. The effectiveness of the output is measured against the % accuracy, and % loss for different number epochs. The number of hidden layers for these tests were kept constant for each run.

Table 2 clearly shows that an increasing number of epochs refines the output by increasing the accuracy and decreasing the data loss. The model gave a correct prediction of the fruit classification in all the runs.

8.1.2 Number of hidden layers

The effect of increasing the number of hidden layers on the model, for each run, is shown in **Table 3**. The effectiveness of the output is measured against the % accuracy, and % loss for various number hidden layers. The number of epochs for these tests was kept constant for each run.

Table 3 clearly shows that an increasing number of hidden layers increases the model effectiveness by increasing the accuracy and decreasing the data loss. The model gave one wrong prediction, when there were 2 hidden layers. Whereas, by increasing the number of hidden layers, the model started to predict correctly.

8.1.3 Overall summary

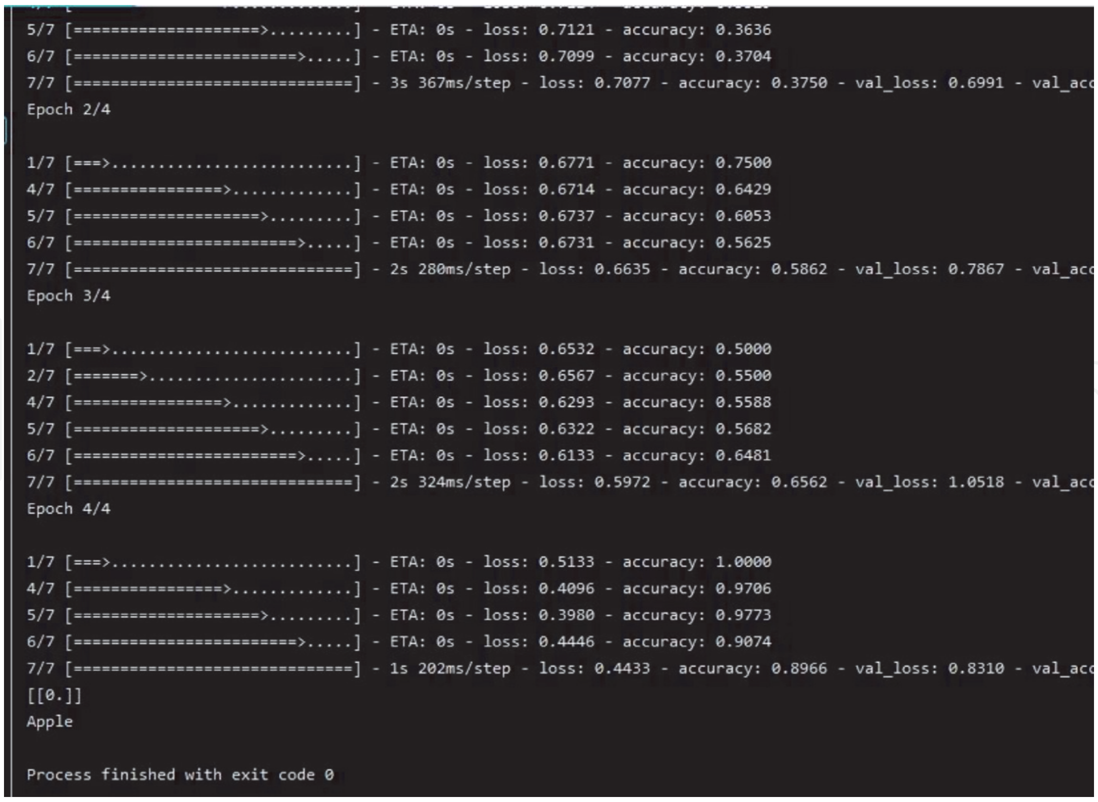
The output window from the model is shown in **Figure 13**. It can be seen that the model successfully predicted the correct output ('Apple'). The accuracy of the model was increasing with each epoch from almost 37 to 89% and data loss was also decreasing, consecutively. The program code for this model is given in Appendix A.

Number of epochs	% Accuracy	% Loss	Prediction
4	74.14	56.41	Correct
8	81.25	43.44	Correct
12	100	27.77	Correct

Table 2.
Output summary for increasing number of epochs.

Number of hidden layers	% Accuracy	% Loss	Prediction
2	45.83	67	Incorrect
4	70	64.90	Correct
6	100	61.38	Correct

Table 3.
Output summary for increasing number of hidden layers.



```
5/7 [===================>.........] - ETA: 0s - loss: 0.7121 - accuracy: 0.3636
6/7 [=======================>.....] - ETA: 0s - loss: 0.7099 - accuracy: 0.3704
7/7 [=============================] - 3s 367ms/step - loss: 0.7077 - accuracy: 0.3750 - val_loss: 0.6991 - val_acc
Epoch 2/4

1/7 [===>.........................] - ETA: 0s - loss: 0.6771 - accuracy: 0.7500
4/7 [===============>.............] - ETA: 0s - loss: 0.6714 - accuracy: 0.6429
5/7 [===================>.........] - ETA: 0s - loss: 0.6737 - accuracy: 0.6053
6/7 [=======================>.....] - ETA: 0s - loss: 0.6731 - accuracy: 0.5625
7/7 [=============================] - 2s 280ms/step - loss: 0.6635 - accuracy: 0.5862 - val_loss: 0.7867 - val_acc
Epoch 3/4

1/7 [===>.........................] - ETA: 0s - loss: 0.6532 - accuracy: 0.5000
2/7 [=======>.....................] - ETA: 0s - loss: 0.6567 - accuracy: 0.5500
4/7 [===============>.............] - ETA: 0s - loss: 0.6293 - accuracy: 0.5588
5/7 [===================>.........] - ETA: 0s - loss: 0.6322 - accuracy: 0.5682
6/7 [=======================>.....] - ETA: 0s - loss: 0.6133 - accuracy: 0.6481
7/7 [=============================] - 2s 324ms/step - loss: 0.5972 - accuracy: 0.6562 - val_loss: 1.0518 - val_acc
Epoch 4/4

1/7 [===>.........................] - ETA: 0s - loss: 0.5133 - accuracy: 1.0000
4/7 [===============>.............] - ETA: 0s - loss: 0.4096 - accuracy: 0.9706
5/7 [===================>.........] - ETA: 0s - loss: 0.3980 - accuracy: 0.9773
6/7 [=======================>.....] - ETA: 0s - loss: 0.4446 - accuracy: 0.9074
7/7 [=============================] - 1s 202ms/step - loss: 0.4433 - accuracy: 0.8966 - val_loss: 0.8310 - val_acc
[[0.]]
Apple

Process finished with exit code 0
```

Figure 13.
Output summary of CNN model.

8.2 Unsupervised ANN model

A simple unsupervised ANN model was developed for the colour quantization of an image, using Python, and Self-Organising Maps (SOM) methodology was adopted. SOM is basically used for feature detection.

Two different images of houses were selected for colour quantization by the SOM model. Separate tests were conducted with each image keeping the same model conditions. In each test, the developed SOM model reduced the distinct colours of the image, and another image was developed. This technique helped the model to learn the colours in the image and then use the same colours to reconstruct that image. The pictorial views for each output are shown in **Figure 14**.

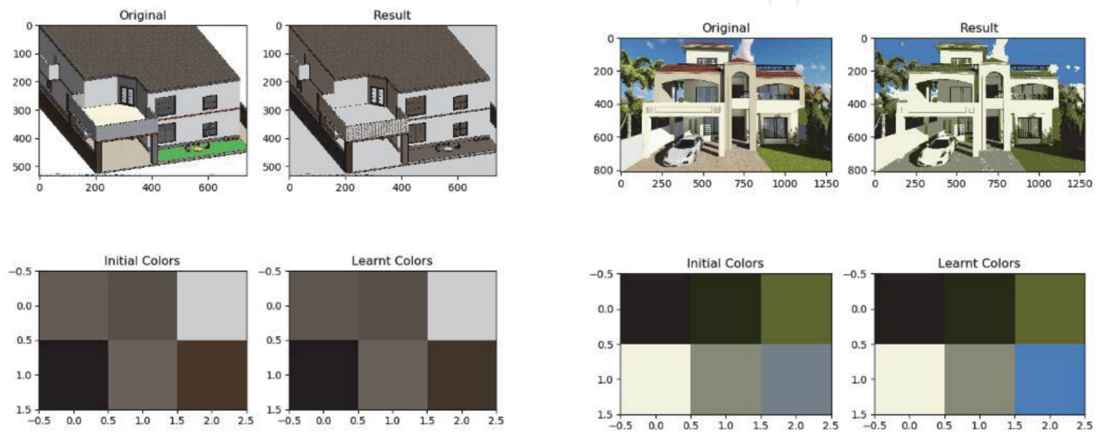
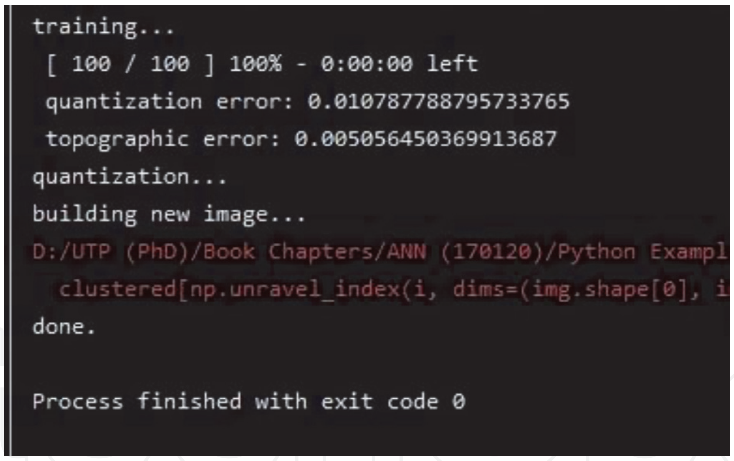


Figure 14.
Pictorial output of SOM model.



```
training...
 [ 100 / 100 ] 100% - 0:00:00 left
 quantization error: 0.010787788795733765
 topographic error: 0.005056450369913687
quantization...
building new image...
D:/UTP (PhD)/Book Chapters/ANN (170120)/Python Exampl
  clustered[np.unravel_index(i, dims=(img.shape[0], i
done.

Process finished with exit code 0
```

Figure 15.
Output window of SOM model.

8.2.1 Overall summary

It can be seen in the output results that for each test the model detected the distinct colours and using the same colours it reproduced that image. The output window from the model is shown in **Figure 15**. The program code for this model is given in Appendix B.

9. Conclusions

Operation of the ANN model is the simulation of the human brain, and they fall under the knowledge domain of AI. The popularity of ANN models were increased in the early 1990s, and many studies have been done since. The basic ANN model has three main layers, and the main process is performed in the middle layer known as the hidden layer. The output of the ANN model is very much dependent on the characteristics and function it carries under the hidden layer. Among the feedforward and feedback networks, the latter one propagates the error unless it became minimum for more effective results. The ANN models can perform supervised learning as well as unsupervised learning depending upon the task. The DL algorithms are very much popular among researchers because of effective outputs with large data. CNN and RNN are the two renown deep networks, and they have been used for various applications. Output accuracy of the ANN models is very much dependent on the number of hidden layers and the number of epochs.

In this era of automation, the AI plays an important role, and most of the daily use applications are based on the architecture of ANN models. This ANN technology, combined with other advanced and AI knowledge areas, is making life easier in almost every domain. This evolution of DNN models has led to the creation of Sophia the Robot (Hanson Robotics); the journey is on-going.

Acknowledgements

We will like to acknowledge UTP.

Conflict of interest

There is no conflict of interest.

Notes/Thanks/Other declarations

Special thanks to UTP

Appendix A

Program code for supervised CNN model is given below:

Step#1 *Opening Python*

Python was opened, and conda environment was selected.

Step#2 *Installing and Import Necessary Data Sources*

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
from keras.preprocessing import image
import tensorflow as tf
import numpy as np
```

Step#3 *CNN Convolutional Network Model*

```
#Input Layer
model = Sequential()

# Convolutional Layer 1
model.add(Conv2D(64 (3, 3), input_shape = (150, 150, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

# Convolutional Layer 2
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
```

The convolutional network helps to extract features from the image and digit 64 means to extract 64 features.

```
model.add(Flatten())

# Hidden Layer 1
model.add(Dense(units = 64, activation = 'relu'))

# Hidden Layer 2
model.add(Dense(units = 32, activation = 'relu'))
```

64 and 32 represents the number of neurons in these layers.

```
# Output Layer
model.add(Dense(units = 1, activation = 'sigmoid'))

# Compiling the CNN
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Sigmoid represents the activation function of this model.

***Step#4** Fitting CNN to the Images (Training and Validation)*

```
train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.2,
zoom_range = 0.2,
horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
training_set = train_datagen.flow_from_directory('Data/Train',

target_size = (150, 150),
batch_size = 12,
class_mode = 'binary')
test_set = test_datagen.flow_from_directory('Data/Validation',

target_size = (150, 150),
batch_size = 8,
class_mode = 'binary')
model.fit_generator(training_set,
steps_per_epoch = 10,
epochs = 4,
validation_data = test_set,
validation_steps = 10)
```

directories are for the path to the training folder and validation folder

***Step#5** Running the ANN Model*

```
test_image = image.load_img('Data/Apple.JPG', target_size = (150, 150))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = model.predict(test_image)
training_set.class_indices

if result[0][0] == 1:
  prediction = 'Orange'
else:
  prediction = 'Apple'

print (prediction)
```

Appendix B

Program code for unsupervised SOM model is given below:

***Step#1** Opening Python*

Python was opened, and conda environment was selected.

***Step#2** Installing and Import Necessary Data Sources*

```
from minisom import MiniSom
import numpy as np
import matplotlib.pyplot as plt
```

Step#3 *Importing Image*

```
img = plt.imread('HouseTest2.jpg')
```

Image path is to be given here.

Step#4 *SOM Model*

```
# Reshaping the pixels matrix
pixels = np.reshape(img, (img.shape[0]*img.shape[1], 3)) / 255.

# SOM initialization
som = MiniSom(2, 3, 3, sigma=1.,
learning_rate=0.2, neighborhood_function='bubble')

# Setting Weights
som.random_weights_init(pixels)
starting_weights = som.get_weights().copy()
som.train_random(pixels, 100, verbose=True)
```

100 is the number of training iteration

```
# Quantization
qnt = som.quantization(pixels)

# Compilation
clustered = np.zeros(img.shape)

for i, q in enumerate(qnt):
  clustered[np.unravel_index(i, dims=(img.shape[0], img.shape[1]))] = q
print('done.')
```

Step#5 *Running and Plotting of ANN Model*

```
plt.figure(figsize=(7, 7))
plt.subplot(221)
plt.title('Original')
plt.imshow(img)
plt.subplot(222)
plt.title('Result')
plt.imshow(clustered)

plt.subplot(223)
plt.title('Initial Colors')
plt.imshow(starting_weights)
plt.subplot(224)
plt.title('Learnt Colors')
plt.imshow(som.get_weights())

plt.tight_layout()
plt.show()
```

Data Processing Using Artificial Neural Networks
DOI: http://dx.doi.org/10.5772/intechopen.91935

Author details

Wesam Salah Alaloul* and Abdul Hannan Qureshi
Department of Civil and Environmental Engineering, Universiti Teknologi PETRONAS, Perak, Malaysia

*Address all correspondence to: wesam.alaloul@utp.edu.my

References

[1] Ciaburro G, Venkateswaran B. Neural Networks with R: Smart Models Using CNN, RNN, Deep Learning, and Artificial Intelligence Principles. Birmingham: Packt Publishing; 2017

[2] Zou J, Han Y, So S-S. Overview of Artificial Neural Networks. Vol. 458. Totowa: Humana Press; 2008. pp. 14-22. DOI: 10.1007/978-1-60327-101-1_2

[3] Aggarwal CC. Neural Networks and Deep Learning. Cham: Springer International Publishing; 2018. DOI: 10.1007/978-3-319-94463-0

[4] Araghinejad S. Data-Driven Modeling: Using MATLAB® in Water Resources and Environmental Engineering. Vol. 67. Dordrecht: Springer Netherlands; 2014. DOI: 10.1007/978-94-007-7506-0

[5] Adeli H, Yeh C. Perceptron learning in engineering design. Computer-Aided Civil and Infrastructure Engineering. 1989;**4**:247-256

[6] Adeli H. Neural networks in civil engineering: 1989–2000. Computer-Aided Civil and Infrastructure Engineering. 2001;**16**:126-142. DOI: 10.1111/0885-9507.00219

[7] Mehrotra K, Mohan CK, Ranka S. Elements of Artificial Neural Networks. Cambridge: MIT Press; 1997

[8] Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science; 1985

[9] Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks. 2015;**61**:85-117. DOI: 10.1016/j.neunet.2014.09.003

[10] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation. 1997;**9**:1735-1780. DOI: 10.1162/neco.1997.9.8.1735

[11] Feldman JA, Fanty MA, Goddard NH. Computing with structured neural networks. IEEE Computer. 1988;**21**:91-103

[12] Profillidis VA, Botzoris GN. Artificial intelligence—Neural network methods. Modeling of Transport Demand. Elsevier; 2019:353-382. DOI: 10.1016/B978-0-12-811513-8.00008-X

[13] Taylor M. Make Your Own Neural Network: An In-Depth Visual Introduction for Beginners. 2017. Independently published

[14] Priddy KL, Keller PE. Artificial Neural Networks: An Introduction. Bellingham: SPIE Press; 2005

[15] Suk H-I. An Introduction to Neural Networks and Deep Learning. 1st ed. Cambridge: Academic Press; 2017. DOI: 10.1016/B978-0-12-810408-8.00002-X

[16] Poznyak TI, Chairez Oria I, Poznyak AS. Background on dynamic neural networks. Ozonation and Biodegradation in Environmental Engineering. 2019:57-74. DOI: 10.1016/b978-0-12-812847-3.00012-3

[17] Zamir AR, Wu TL, Sun L, Shen WB, Shi BE, Malik J, et al. Feedback networks. In: Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017); 2017. pp. 1808-1817. DOI: 10.1109/CVPR.2017.196

[18] Whittington JCR, Bogacz R. Theories of error back-propagation in the brain. Trends in Cognitive Sciences. 2019;**23**: 235-250. DOI: 10.1016/j.tics.2018.12.005

[19] Awodele O, Jegede O. Neural networks and its application in

engineering. In: Proceedings of the 2009 InSITE Conference; 2009. DOI: 10.28945/3317

[20] Rao Z, Alvarruiz F. Use of an artificial neural network to capture the domain knowledge of a conventional hydraulic simulation model. Journal of Hydroinformatics. 2007;**9**:15-24. DOI: 10.2166/hydro.2006.014

[21] Perez RR, Marques A, Mohammadi F. The application of supervised learning through feed-forward neural networks for ECG signal classification. In: Canadian Conference on Electrical and Computer Engineering; 2016. pp. 1-4. DOI: 10.1109/CCECE.2016.7726762

[22] Daniel G. Principles of Artificial Neural Networks. Singapore: World Scientific; 2013

[23] Gurney K. An Introduction to Neural Networks. Boca Raton: CRC Press; 1997

[24] Damilola S. A review of unsupervised artificial neural networks with applications. International Journal of Computers and Applications. 2019; **181**:22-26. DOI: 10.5120/ijca2019918425

[25] Coulibaly P, Anctil F, Bobée B. Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. Journal of Hydrology. 2000;**230**:244-257

[26] Liang P, Bose NK. Neural Network Fundamentals with Graphs, Algorithms and Applications. New York: MacGraw-Hill; 1996

[27] Chauhan NK, Singh K. A review on conventional machine learning vs deep learning. In: 2018 International Conference on Computing, Power and Communication Technologies (GUCON 2018); 2019. pp. 347-352. DOI: 10.1109/GUCON.2018.8675097

[28] Tavanaei A, Ghodrati M, Kheradpisheh SR, Masquelier T, Maida A. Deep learning in spiking neural networks. Neural Networks. 2019;**111**:47-63. DOI: 10.1016/j.neunet.2018.12.002

[29] This Malaysian State is Using Facial Recognition to Catch Criminals—Tech. Available from: https://sea.mashable.com/tech/1725/this-malaysian-state-is-using-facial-recognition-to-catch-criminals [Accessed: 27 February 2020]

[30] Facial Recognition System in University Classrooms Sparks Controversy—SHINE News. Available from: https://www.shine.cn/news/nation/1909041394/ [Accessed: 27 February 2020]

[31] Karahan Ş, Yildirim MK, Kirtaç K, Rende FŞ, Bütün G, Ekenel HK. How image degradations affect deep CNN-based face recognition? 2016 Int. Conf. Biometrics Spec. Interes. Gr., Vol. P-260. IEEE; 2016. pp. 1-5. DOI: 10.1109/BIOSIG.2016.7736924

[32] Khan S, Javed MH, Ahmed E, Shah SAA, Ali SU. Networks and implementation on smart glasses. In: 2019 International Conference on Information Science and Communication Technologies; 2019. pp. 1-6. DOI: 10.1109/CISCT.2019.8777442

[33] Bhandare A, Bhide M, Gokhale P, Chandavarkar R. Applications of convolutional neural networks. International Journal of Computer Science and Information Technologies. 2016;7:2206-2215

[34] Zhang D, Wang D. Relation classification: CNN or RNN? Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2016;**10102**:665-675. DOI: 10.1007/978-3-319-50496-4_60

[35] Graves A, Mohamed AR, Hinton G. Speech recognition with deep recurrent

neural networks. In: ICASSP, IEEE International Conference on Acoustics, Speech, and Signal Processing; 2013. pp. 6645-6649. DOI: 10.1109/ICASSP.2013.6638947

[36] Wang WJ, Liao YF, Chen SH. RNN-based prosodic modeling for mandarin speech and its application to speech-to-text conversion. Speech Communication. 2002;**36**:247-265. DOI: 10.1016/S0167-6393(01)00006-1

[37] Yin W, Kann K, Yu M, Schütze H. Comparative Study of CNN and RNN for Natural Language Processing. ArXiv Prepr ArXiv170201923 2017 (belongs to Cornell University, NY, USA)

[38] Tasyurek M, Celik M. RNN-GWR: A geographically weighted regression approach for frequently updated data. Neurocomputing. 2020. DOI: 10.1016/j.neucom.2020.02.058

[39] Fridman L, Terwilliger J, Jenik B. DeepTraffic: Crowdsourced Hyperparameter Tuning of Deep Reinforcement Learning Systems for Multi-Agent Dense Traffic Navigation. 2018