iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.68644

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Demystifying Neural Network: A Case Study on Intrusion Detection System

Vyom Patel
SGVP International School

Abstract *Considering the historical context of Deep Neural Networks (DNNs), it can be inferred that DNN has had a promising history that has now evolved with various philosophical viewpoints. Modeling DNN has now become effortless with the generation and availability of large amount of data. Moreover, with the development of improved hardware and software infrastructure requirements, DNN models have now grown in size. DNN models have the capability of addressing complex application problems with improved accuracy over time. The art and science of DNN is based on the foundation of neural network. Thus, this paperr aims at discussing the fundamentals of neural networks and how they work. The paper includes a brief discussion on functionality of neural network, role of activation functions, backpropagation algorithm, loss function calculations, and optimizers for neural networks. Further, the paperr also discusses generalization in neural network and parameters in neural network architecture. The paper also includes a case study on intrusion detection model building using neural network. The demystifying neural network is the first stride towards understanding DNN.*

I. BIOLOGICAL MOTIVATION AND CONNECTIONS

The neural network architecture is inspired from how a human brain works. Neural network can be considered as a naive implementation of human brain. A human brain can be considered as a cluster of neurons that are connected in an inter- connected network. A neuron is a fundamental unit of human brain. A small unit of brain can consist of 10,000 neurons with approximately 6000 connections among the neurons [8]. Thus, a human brain consists of large number of neurons that are connected with each other in a humongous structure. The neurons in brain transmit electrical charge for communicating information to connecting neurons. Apart from transmitting information, one of the important functionality of neural connections is that the connection between two neurons can be a strong connection or a weak connection. A strong connection indicates that more information can flow between the two neurons and a weak connection indicates that less information flows between the connecting neurons. A connection that frequently shares information through continuous electric discharge will gradually become a strong connection [1]. Thus, this natural formation of human brain can be used to understand and design neural network models that address the problems in an analogous way as human brain. Thus, a neuron in neural network structure is programmed to receive data from other neurons, models the data based on its computational intelligence, and forwards the output to the other neurons connected in the network.

A schematic of human brain neuron structure is depicted in Fig. 1. In human brain, input to neurons is fed through antenna-like structure termed as dendrites. The connections between neurons is strengthen or weakened based on its usage. This implies that if the incoming connection is used often for transferring information to other connecting neurons then it is termed as a strong connection. Moreover, strength of each connection also reveals contribution of input neurons to its output [1]. Further, the strengths of each input connections are modeled and summed together in the cell body [1]. The summed connection is then weighted and transformed in to a electric signal. The electric signal passes through the cell's axon and transmits the data contained in the signal to other neurons.

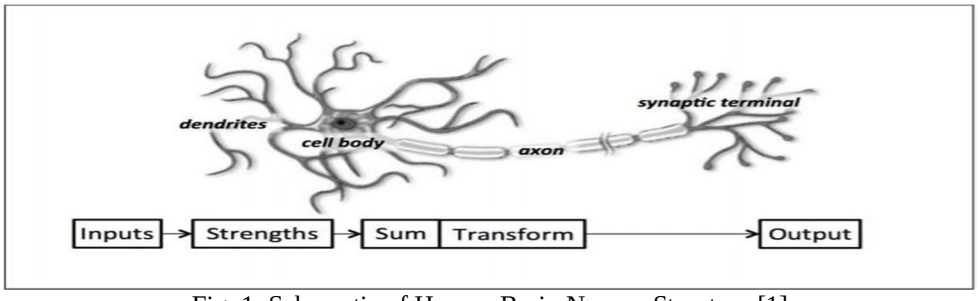


Fig. 1: Schematic of Human Brain Neuron Structure [1]

The functional understanding of neurons in human brain can be used as a basis to model neuron in neural network architecture. A schematic of neuron for neural network is presented in Fig. 2. Here, the input signals that travel along on the axon are represented by x_0, x_1, and x_2. The synaptic strength is represented as weights w_0, w_1, and w_2. The input signals interact multiplicatively with the dendrites as
w_0x_0. The weights w in neural network architecture are learnable parameters that are modeled and have the capability to control the strength of neuron as positive weight or negative weight. In neural network architecture, connecting edges of the neurons can be considered as dendrites. The edges of the neurons carry the signals to the cell body, where all the input signals are summed along with the bias b. The summed input signals are transformed and modeled using the activation function , that contributes in evaluating the signal strength. Thus, every neuron in neural network architecture interacts multiplicatively, by computing the dot product of input signals and its weights. Further, it sums the dot product of all the input signals and their weights along with bias, and applies activation function to obtain an output signal.

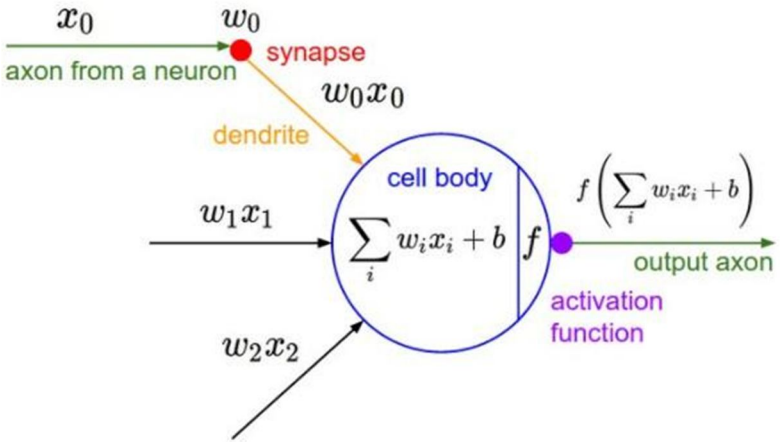


Fig. 2: A Schematic of Neuron in Neural Network Architecture

II. FUNDAMENTALS AND FUNCTIONALITY OF NEURAL NETWORKS

Neural Network has a layered architecture consisting of neurons. A neural network contains three types of layers namely, an input layer, an output layer, and one or more hidden layers. Each layer can have any number of neurons [6]. Neural network is just like an acyclic graph where neurons in the same layer are not connected with each other [6]. A schematic of neural network structure is shown in Fig. 3. Thus, as depicted in the Fig 3, each and every node in one layer is connected to each and every node in the neighbouring layer through edges. Type of layers in neural network architecture can be describes as.

Input Layer: Input layer consists of input nodes that takes raw data as input and transfers the data to the network. No computation is performed on the input nodes, they just pass the information to hidden layers [10].

Hidden Layer: Hidden layer consists of hidden nodes that takes input from the input nodes and performs computation and forwards the information to the next hidden layer or output layer [10].

Output Layer: Output layer consists of output nodes. This layer is responsible for performing computations and delivering the output of neural network learning process [10].

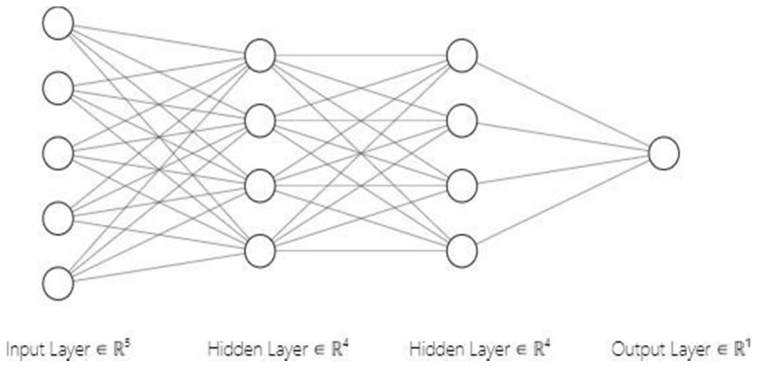


Fig. 3: A Sample Neural Network Structure

A. *Functionality of Single Neuron*

For understanding the functionality of single neuron, consider an example as shown in Fig 4. Here, X1, X2, and X3 are input values that are connected to a hidden node H through edges. The edges represent the weight values of X1, X2, and X3 input nodes as W1, W2, and W3, respectively. The single neuron computation consists of a bias B, which is a trainable constant value that is added to the calculation of hidden neuron to have a bit of adjustability. Y is the final output value, which is obtained by applying activation function f(.) and the computation of intermediate hidden node is performed using equation 1.

$$H = X1 * W1 + X2 * W2 + X3 * W3 + B \quad (1)$$

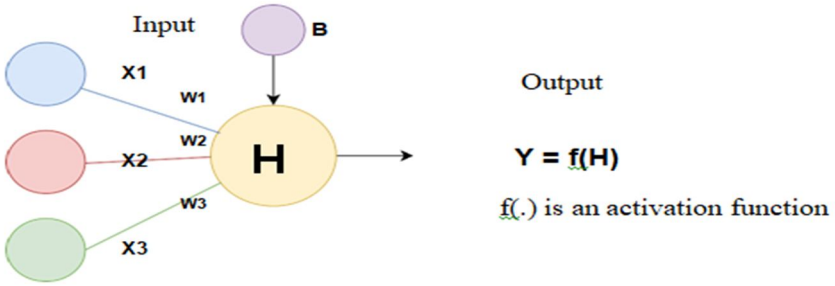


Fig. 4: Functionality of Single Neuron

A schematic of computation between two layers is shown in Fig. 5. Here, input layer consists of three neurons X1, X2, and X3 which are connected to intermediate hidden nodes H1, H2, and H3. The output layer consists of three nodes Y1, Y2, and Y3 whose values are derived by applying the activation function f(.). The intermediate node calculation is performed by multiplying weight matrix with input vector. Also, bias vector is added to the intermediate node values. Thus, the entire calculation to obtain the out vector can be represented using equation 2.

$$Y = f(W * X + B) \quad (2)$$

Here, Y is the vector of output values, f(.) is the activation function, W is the weight matrix, X is the vector of input values, and B is the bias vector. Moreover, the size of the weight matrix can be determined by number of nodes between the input layer and hidden layer.

Consider an entire neural network architecture as depicted in Fig 3, with four layers that is an input layer, an output layer and two hidden layers. The input layer consists of five input nodes from where raw data is fed in to the neural network for learning. The output layer consists of one node, this is where we get the target value that depicts what exactly the neural network is trying to predict based on the learning. All layers between the input and output layer are defined as hidden layers. Here, in Fig. 3, we have two hidden layers with four neurons in each layer. The intermediate node values of hidden layer is computed by performing matrix multiplication using the input values and weights from the first layer and adding bias values. Activation function is applied on the intermediate node values to obtain final output value at the output layer. The output value for the neural network structure in Fig 3 can be obtained using equation 3.

$$Y = f(W3 * f(W2 * f(W1 * X + B1) + B2) + B3) \quad (3)$$

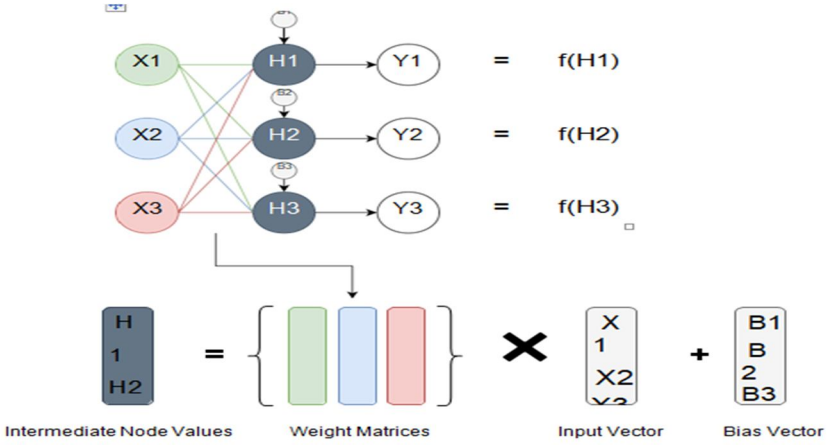


Fig. 5: Schematic of Computation between two Layers

Here, Y is the output vector, f(.) is the activation function, W1 is the weight matrix between input layer and first hidden layer, W2 is the weight matrix between first hidden layer and second hidden layer, W3 is the weight matrix between second hidden layer and output layer. B1, B2, and B3 are the bias values for the neural network architecture. Thus, a neural network can be described as a series of matrix multiplications and activation functions. Input vector is multiplied with sequence of weight matrices and activation function is applied to derive the output vector, which is the predicted value by neural network for a given input.

III. ACTIVATION FUNCTIONS

Activation functions are an integral part of neural network that are used for determin- ing and regulating the output of neurons in neural networks [22]. Activation functions also play a vital role in controlling accuracy and computational efficiency of neural network model. They have the capability to scale the neural network structure and also, affect the convergence ability of neural network [2]. Activation functions are mathematical equations that are applied on nodes in neural network model to derive the output and instill non-linearity [2]. The acti- vation function is applied on every node present in the network to dictate whether a given node should be activated or not. Moreover, activation functions are also used to normalize the output of every node in a range of [0,1] or [-1,1] [2]. Thus, activation functions are expected to be computationally efficient as they are applied on thousands of nodes that are present in the neural network model. In neural network modeling, numeric raw data is given as input to the nodes of input layer. Every node in the input layer has a weight that is multiplied with numeric input of a given neuron and is further given as input to the hidden layer. Here, activation function act as a mathematical gate between the input and output of a given node as shown in Fig. 6 [23]. It is similar to a step function that activates or deactivates a node based on a rule or threshold defined for a given mathematical computation. Moreover, activation function may also be used for mapping the input data into output data that is required for making predictions in neural network. There are three types of activation functions namely, binary step, linear, non-linear activation functions [22]. Generally, non-linear activation are applied in neural network modeling. This is because non-linear activation function can assist the neural network to learn and process complex data and subsequently provide precise prediction for a given input data. The common non-linear activation functions are summarized in Table 1 along with their advantages and disadvantages.

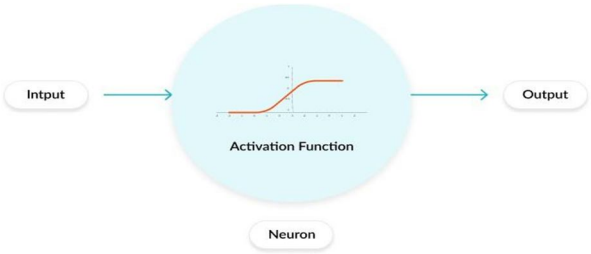


Fig. 6: Application of Activation Function on a Single Node [23]

Table 1: Non-Linear Activation Functions [22]

Name	Equation	Advantages	Disadvantages
Sigmoid	$\sigma(x) = \frac{1}{1+ex.^{-1}(-x)}$ p	Smooth Gradient · Output Values Bound between 0 and 1 Derives clear predictions	· Suffers from vanishing gradient problem It is not centered zero Computationally expensive
TanH	$tanh(x) = 2\,\sigma(2x) - 1$	It is centered zero · Output values bound between -1 and 1 Derives clear predictions	· Suffers from vanishing gradient problem Computationally expensive
ReLU	$f(x) = max(0, x)$	Allows network to converge uickly · Replaces negative weight values to zero Allows back-propagation	· Suffers from dying ReLU prob- lem
Leaky ReLU	$f(x) = max(0.1 * x, x)$	· Addresses the dying ReLU prob- lem · Allows network to converge uickly	· It does not provide consistent result values for negative weight values.
Parametric ReLU	$f(x) = max(\alpha * x, \alpha)$	· It has the capability to learn neg- ative slope Fixes dying ReLU problem Allow back-propagation	It might perform different for dif- erent application problems
Softmax	$exp(xi)$ $softmax(x)_i = . j exp(xj))$	· Can be applied for multi-class classification · Output values bound between 0 and 1 Usually applied on output layer	· It does not work for linearly sep- arable data It does not work for null rejection

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

IV. PARAMETERS AND HYPERPARAMETERS IN NEURAL NETWORKS

Neural network model parameters are configuration variables that are learned during the training process and are estimated from raw data given as input to the model [1]. In neural networks, connection weights are considered as model parameters that are trained and optimized for predicting the output. Feature of model parameters can be summarized as follows.

Model parameters contribute in making predictions for given input based on the learning.

The values of parameters helps in defining the model performance for the given application problem.

The values of parameters are estimated and learned from the dataset used for training neural network model.

Values of model parameters can not be assigned manually, instead values of parameters are gradually learned and optimized during the training process.

– The values of model parameters are often saved as a part of trained model.

Neural network hyperparameters are configuration variables that are assigned manually for training model and raw data is not required for estimating the value of hyperparameters [1]. In neural network, number of hidden layers, number of neurons in hidden layers, number of epochs, batch size, optimizers, learning rate, to name a few are some of the hyperparameters that are defined manually for enhancing the performance of neural networks [1]. Feature of model hyperparameters can be summarized as follows.

Model hyperparameters contribute in assessing and optimizing parameters of neural network.

Model hyperparameters are assigned manually or are derived using optimization algorithms such as grid search and Bayesian optimization.

Model hyperparameters can also be tuned to achieve better predictive capability of neural network model.

– Heuristics algorithms can also be used to find the values of hyperparameters.

A comparison between parameters and hyperparameters of neural network is pre- sented in Table 2.

V. NEURAL NETWORK MODEL GENERALIZATION

Generalization of a neural network model can be described as model's ability to precisely classify and predict unknown data given as input to the trained model [19]. If the model is not trained enough then it might not perform well on training as well as test data. Whereas, if the model is trained more then it might perform well with training data but not with test data [19]. Thus, a model needs to be generalized well to achieve better performance with training as well as test data. Thus, to summarize the use cases for considering generalization in neural network.

Under-fitting: A neural network model that lacks in learning from data and does not perform well with training data as well as test data, such a condition can be described as under-fitting [30].

Over-fitting: A neural network model that learns too well on training data and performs well with training data but fails to perform for test data, such a condition can be described as over-fitting [30].

Good-fit: A neural network model that sufficiently learns well on training and even performs well for the test data, such a model can be described as a good fit model as it generalizes well on unseen data based on the learning from training data [30].

The problem of over-fitting can be addressed by tuning network parameters or changing network structure. Tuning the network parameters refers to optimizing the values of parameters using hyperparameters such as optimization algorithms [19]. Whereas, changing the network structure refers to reducing the complexity of neural network by reducing weights or number of neurons in the neural network architecture. The techniques applied for tuning the network parameters or changing the network strcuture are referred to as regularization techniques. The commonly used regularization techniques are summarized as follows.

Table 2: Comparison between Parameters and Hyperparameters

Criteria	Parameters	Hyperparameters
Purpose	Parameters are required for prediction and classification	Hyperparameters are required for optimiz- ing and updating the parameter values
Estimation	Parameters are estimated by using opti- mization algorithms	Hyperparameters are estimated by apply- ing hyperparameter tuning techniques
Assignment	Parameters are not assigned manually, in- stead they are learned during the training process	Hyperparameters are assigned manually by the practitioner
Decisive Capa- bility	Final parameter values obtained after com- pletion of training process help in estimat- ing model performance	Hyperparameter values helps in evaluating training efficiency and estimating parame- ter values based on optimization process

Dropout Probability: It can be applied on the input layer or hidden layers of neural network model. The goal of dropout probability is to randomly deactivate neurons connected in neural network [13]. Dropout probability helps in addressing issues in neural network such as over-fitting and co-adaptation. There are various variants of dropout such as Gaussian dropout, adaptive dropout that can be used for achieving generalization in neural networks [13].

Noise: It is a regularization technique that is used to insert noise in training data. Here, noise is induced by adding or multiplying noise in the hidden nodes for a given neural network [8]. Thus, incorporating uncertainties in training data, training performance as well as predictive capability of neural network can be enhanced. Commonly, Gaussian noise is used to add noise in the training data [8].

Early Stopping: It is a regularization technique that prompts neural network model to stop during training phase, if the model has learned all features from input data. Early stopping technique monitors the validation loss for stopping the training phase. It stops the model training if no improvement is found in the validation loss. Thus, early stopping ensure that model gets enough time for learning from data and not learn from noise [8].

Batch Normalization: It is a regularization technique that normalizes the data which is given as input to neural network model. It is applied on activations of previous layers or on input data. Batch normalization enables every layer of neural network model to learn independently [8].

Weight Decay: Weight decay is applied in neural network using L1 and L2 reg- ularization [32]. It reduces the complexity of the network by penalizing weights. The weights are reduced to a small magnitude using regularization parameter [32]. Thus, inputs to the model are mapped with the output values by keeping the magnitude of weights small. Thus, in this way complexity of the network can be reduced and generalization can be achieved for a given neural network model [32].

VI. INTRUSION DETECTION USING NEURAL NETWORK: A CASE STUDY

There has been research in the field of Intrusion Detection System (IDS) for attack detection and classification. However, with increase in the network traffic, complex- ity in nature and type of attacks has evolved [26]. Therefore, there have been efforts to design efficient IDS using Deep Learning (DL) techniques. The functional modules for designing neural network based-IDS is shown in Fig. 8. The main functional mod- ules include, data pre-processing, feature engineering, and building neural network model [26].

Raw Network Traffic Data: Neural network requires network traffic data for learn- ing and deriving patterns of normal network traffic and anomalous network traffic. Moreover, raw traffic consists of network features that are extracted from network packets flowing through the network. The network packets can be captured using network tools such as sniffer that intercepts the network packets. Network packets consists of network header and network payload. The information extracted from network header and network payload can be combined as fed as input to the neural network for learning and deriving relationships between the features. Apart from capturing the network traffic, publicly available datasets can be used for carrying out the analysis for intrusion detection [27]. Thus, based on the learning, neural network can be used for predicting and classifying data as normal or anomalous. Data Pre-Processing: The collected raw network traffic needs to be pre-processed. This is because packet header and packet payload might consists of network data of varied types such as nominal, binary, categorical, discrete, and continuous. Thus, in order to process and learn, data needs to normalized and converted into numerical data. Therefore, data-preprocessing include scaling and standardization of feature values and conversion of categorical features in to numerical features. Feature Engineering: Neural networks have the characteristic property of perform- ing end-to-end learning. Moreover, they have the capability of learning features automatically during training process. The features of the dataset for a given application problem are given as input to neural network. Gradually, feature rep- resentations are learned during the training process, which are further used for prediction.

Neural Network Model: A layered architecture is designed for learning. It con- sists of input layer, hidden layers, and output layer. The performance of the model depends on various hyperparameters such as number of hidden layers, number of neurons in hidden layers, batch size, epochs, learning rate, and optimization algo- rithm. Model's performance is evaluated based on the values of hyperparameters. Thus, neural network model is built that can learn features from raw dataset and predict the output for unseen data based on learning.

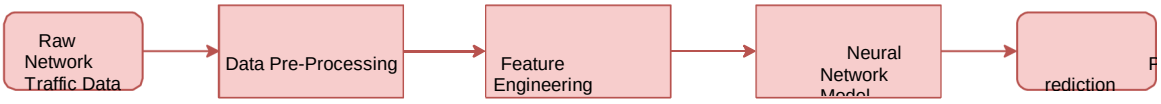


Fig. 8: Sample Module for Neural Network-based Intrusion Detection

Thus, neural network can be used for detecting intrusions and feature learning. However, one of the challenge for designing IDS is to handle large amount of data volume that is being generated by the network. Moreover, various variants of existing attacks and novel attacks are being introduced by the attacker. Therefore, it would be challenging for a neural network based IDS to precisely identify and predict attacks that are not known.

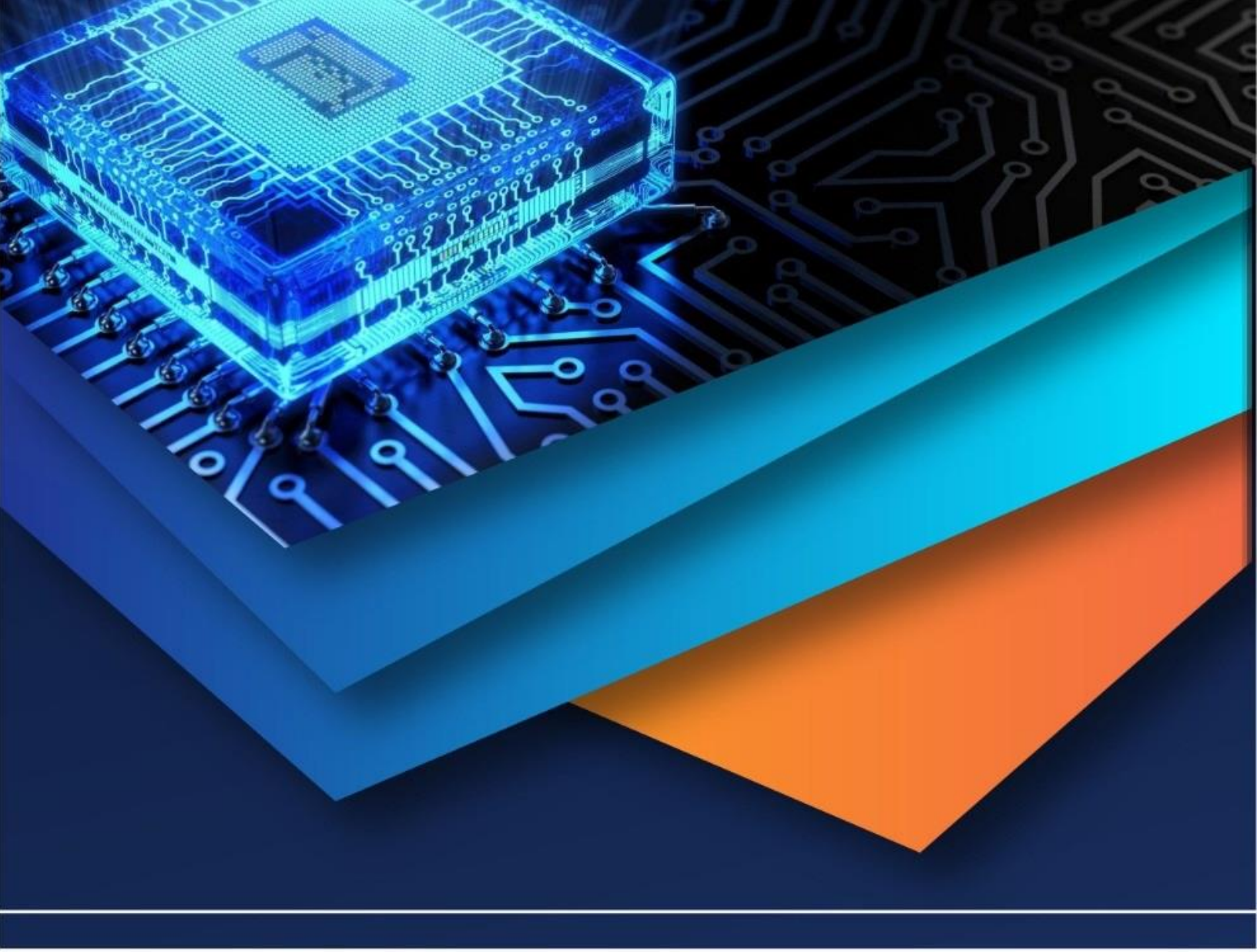
VII. CONCLUSION

Neural network techniques have been widely used for various application domains such as Intrusion Detection System (IDS). The characteristic properties of neural networks such as end-to-end learning and automated feature learning has increased their usage in various application domains. In this paper, we have discussed fun- damentals of neural networks for building models for learning and prediction. Thispaeprdiscusses functionality of neural network, activation functions, backprop- agation technique, loss functions, optimizers, parameters, and hyperparameters of neural networks. Moreover, we have also discussed need to generalize neural net- work models and the techniques used to achieve generalization. Further, we have also discuss a case study that describes functional components that can be considered while building IDS using neural networks.

REFERENCES

[1] Aggarwal, C.C., et al.: Neural networks and deep learning. Springer (2018)
[2] Agostinelli, F., Hoffman, M., Sadowski, P., Baldi, P.: Learning activation functions to improve deep neural networks. arXiv preprint arXiv:1412.6830 (2014)
[3] Al-Sammarraie, N.A., Al-Mayali, Y.M.H., El-Ebiary, Y.A.B.: Classification and diagnosis using back propagation artificial neural networks (ann). In: 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), pp. 1–5. IEEE (2018)
[4] Anand, P., Rastogi, R., Chandra, S.: Generalized —loss function-based regression. In: Machine intelligence and signal analysis, pp. 395–409. Springer (2019)
[5] Cao, Y., Gu, Q.: Generalization bounds of stochastic gradient descent for wide and deep neural networks. In: Advances in Neural Information Processing Systems, pp. 10836–10846 (2019)
[6] Deng, L.: A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing 3 (2014)
[7] Du, S., Lee, J., Li, H., Wang, L., Zhai, X.: Gradient descent finds global minima of deep neural networks. In: International Conference on Machine Learning, pp. 1675–1685 (2019)
[8] Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
[9] Hameed, A.A., Karlik, B., Salman, M.S.: Back-propagation algorithm with variable adaptive momentum. Knowledge-Based Systems 114, 79–87 (2016)
[10] Hatcher, W.G., Yu, W.: A survey of deep learning: Platforms, applications and emerging research trends. IEEE Access 6, 24411–24432 (2018)
[11] Indrapriyadarsini, S., Mahboubi, S., Ninomiya, H., Asai, H.: An adaptive stochastic nesterov accelerated quasi newton method for training rnns. arXiv preprint arXiv:1909.03620 (2019)
[12] Janocha, K., Czarnecki, W.M.: On loss functions for deep neural networks in classification. arXiv preprint arXiv:1702.05659 (2017)
[13] Labach, A., Salehinejad, H., Valaee, S.: Survey of dropout methods for deep neural networks. arXiv preprint arXiv:1904.13310 (2019)
[14] Li, Y., Liang, Y.: Learning overparameterized neural networks via stochastic gradient descent on structured data. In: Advances in Neural Information Processing Systems, pp. 8157–8166 (2018)
[15] Liang, S., Sun, R., Li, Y., Srikant, R.: Understanding the loss surface of neural networks for binary classification. arXiv preprint arXiv:1803.00909 (2018)
[16] Lydia, A., Francis, S.: Adagrad—an optimizer for stochastic gradient descent. Int. J. Inf. Comput. Sci. 6(5) (2019)
[17] Martins, A., Astudillo, R.: From softmax to sparsemax: A sparse model of attention and multi-label classification. In: International Conference on Machine Learning, pp. 1614–1623 (2016)
[18] Narasimhan, H.: Learning with complex loss functions and constraints. In: International Conference on Artificial Intelligence and Statistics, pp. 1646–1654 (2018)
[19] Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., Srebro, N.: The role of over-parametrization in generalization of neural networks. In: International Conference on Learning Representations (2018)
[20] Okewu, E., Misra, S., Lius, F.S.: Parameter tuning using adaptive moment estimation in deep learning neural networks. In: International Conference on Computational Science and Its Applications, pp. 261–272. Springer (2020)
[21] Ren, Y., Zhao, P., Sheng, Y., Yao, D., Xu, Z.: Robust softmax regression for multi-class clas- sification with self-paced learning. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 2641–2647 (2017)
[22] Sharma, S.: Activation functions in neural networks. Towards Data Science 6 (2017)
[23] Sibi, P., Jones, S.A., Siddarth, P.: Analysis of different activation functions using back prop- agation neural networks. Journal of theoretical and applied information technology 47(3), 1264–1268 (2013)
[24] Siregar, S.P., Wanto, A.: Analysis of artificial neural network accuracy using backpropagation algorithm in predicting process (forecasting). IJISTECH (International Journal of Information System & Technology) 1(1), 34–42 (2017)

[25] Tan, H.H., Lim, K.H.: Review of second-order optimization techniques in artificial neural networks backpropagation. In: IOP Conference Series: Materials Science and Engineering, vol. 495, p. 012003. IOP Publishing (2019)
[26] Thakkar, A., Lohiya, R.: Attack classification using feature selection techniques: a comparative study. Journal of Ambient Intelligence and Humanized Computing pp. 1–18 (2020)
[27] Thakkar, A., Lohiya, R.: A review of the advancement in intrusion detection datasets. Procedia
[28] Computer Science 167, 636–645 (2020)
[29] Wang, L., Yang, Y., Min, R., Chakradhar, S.: Accelerating deep neural network training with inconsistent stochastic gradient descent. Neural Networks 93, 219–229 (2017)
[30] Wang, Q., Ma, Y., Zhao, K., Tian, Y.: A comprehensive survey of loss functions in machine learning. Annals of Data Science pp. 1–26 (2020)
[31] Xie, B., Liang, Y., Song, L.: Diversity leads to generalization in neural networks. In: Interna- tional Conference on Artificial Intelligence and Statistics (AISTATS) (2017)
[32] Zajmi, L., Ahmed, F.Y., Jaharadak, A.A.: Concepts, methods, and performances of particle swarm optimization, backpropagation, and neural networks. Applied Computational Intelli- gence and Soft Computing 2018 (2018)
[33] Zhang, G., Wang, C., Xu, B., Grosse, R.: Three mechanisms of weight decay regularization. arXiv preprint arXiv:1810.12281 (2018)
[34] Zhang, Z.: Improved adam optimizer for deep neural networks. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), pp. 1–2. IEEE (2018)



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY