 Received 19-07-2024; Accepted 27-09-2024; Published 03-10-2024

Development of an Intrusion Detection System Leveraging Deep Learning Model Classification

Osa Edosa edosa.osa@uniben.edu
Department of Electrical/Electronic Engineering,
Faculty of Engineering, University of Benin,
P.M.B. 1154, Benin City, Nigeria

Ibhaze Augustus E. eibhaze@unilag.edu.ng
Department of Electrical and Electronic Engineering,
Faculty of Engineering, University of Lagos,
Lagos 100213, Nigeria

Ekoko Erumena C. erumena.ekoko@uniben.edu
Department of Electrical/Electronic Engineering,
Faculty of Engineering, University of Benin,
P.M.B. 1154, Benin City, Nigeria

Orukpe Patience E. patience.orukpe@uniben.edu
Department of Electrical/Electronic Engineering,
Faculty of Engineering, University of Benin,
P.M.B. 1154, Benin City, Nigeria

Corresponding Author: Osa Edosa

Abstract

The implementation of Deep Learning in development of models to act as interventions for addressing the continuously evolving spate of cybersecurity issues has become a noteworthy paradigm. This occurs since cyber attacks could be modeled or represented in terms of data records which can serve as bases for developing intrusion detection systems. This paper proposes an intrusion detection system that leverages deep learning techniques for attack classification. Two deep learning models were developed, a Deep Neural Network (DNN) with ReLU activation as well as Tabular model using the fastai deep learning library. The NSL-KDD benchmark dataset was imported and preprocessed for each model development. After evaluating the models, the fastai model with accuracy of 84 percent surpassed the other DNN with accuracy of 79 percent on NSL-KDD test data.

Keywords: Cybersecurity, NSL-KDD, Deep Learning, Keras, fastai, Intrusion.

Citation: Osa Edosa et al. Development of an Intrusion Detection System Leveraging Deep Learning Model Classification. Advances in Knowledge-Based Systems, Data Science, and Cybersecurity. Research 2024;1(1):2. Pages: 38-48

1. INTRODUCTION

The cyberspace is a constantly evolving environment where a continuous effort to maintain the confidentiality, integrity and availability of information is being exerted by cyber security specialists. This is due to the continuous innovations and crafts by attackers on various infrastructure in the cyberspace, be they software or hardware. It is therefore imperative for efficient measures to be incorporated for mitigating the activities of attackers in computer networks. Intrusion detection systems (IDSs) serve as major frontline defence mechanisms against malicious use of the internet as well as other networks [1]. An intrusion detection system represents an integration of techniques implemented for detecting malicious traffic on computer networks. Traditional IDSs such as the Signature-based types can perform well against already known attacks since a dictionary of such attacks is preconfigured in their databases. However, such systems are deficient against new or zero-day attacks. Zero-day attacks are more suitably addressed using Anomaly-based IDSs which investigate underlying behaviour of traffic and compare such behaviour with already established proper network behaviour. When an anomaly or strange occurrence is perceived by such systems, an alert is triggered. Deep learning which indicates a secondary subset of artificial intelligence is being utilized for the development of relevant and efficient Anomaly-based intrusion detection systems and various authors have therefore proposed systems in this regard. Deep learning is a subset of machine learning that is modeled in likeness to the neuronic operations of the human brain. This subset performs better than classical machine learning algorithms in the case of large dataset analysis. Since benchmark datasets employed in cybersecurity research for intrusion detection are relatively large, deep learning approaches are preferred to classical machine learning since they have deeper levels of data abstraction. Furthermore, deep learning algorithms generalize to unseen data better and thus can detect novel attacks with higher quality [2]. Authors in [3], used Long Short Term Memory (LSTM) technique with the NSL-KDD dataset and utilized the accuracy on the test data (KDD Test+) as the primary metric for assessing their model performances. The proposed LSTM model yielded accuracy scores of 74.77% and 68.78% for the binary and multiclass classification scenarios, respectively. In [4], an IDS using Deep Learning for Software Defined Networks (SDN) was implemented using the NSL-KDD benchmark dataset. A subset of six features deemed to be of importance to SDNs was manually selected for evaluation. Experimental results showed that an accuracy score of 75.75% was obtained by the DNN-IDS. Authors in [5], proposed an IDS using Gain Ratio (GR) measure for feature extraction and Artificial Neural Networks (MLPs) for classification. The UNSW-NB15 dataset was employed for model evaluation with thirty attributes of the UNSW-NB15 dataset being selected. The experimental results demonstrated that the developed model (GR-ANN-MLP) achieved a test precision of 79.80% and test accuracy of 76.96% respectively. In this work, the forty-one features present in the NSL-KDD dataset have been employed to ensure sufficient accuracy of the training and testing by the developed model.

However, there is the need for continuous development of Deep learning based IDSs with improved accuracies and detection rate, hence this paper. The contributions made by this paper include:

1. Development of a Deep Neural Network with an input layer, two hidden layers and an output layer. The input and hidden layers were implemented with the Rectified Linear Unit (ReLU) activation function, while the output layer was implemented with Sigmoid activation function since a binary classification task was considered.

2. Adam optimizer was used with the learning rate while the Binary cross entropy loss function was implemented to evaluate how well the developed DNN algorithm modeled the dataset.
3. Utilization of the NSL-KDD dataset with the developed deep neural network for intrusion detection.
4. Grouping of different attacks present in the NSL-KDD dataset into one label named ‘attack’.
5. BinaryEncoder function was employed for encoding categorical variables and StandardScaler function was employed for standardizing the data.
6. Development of a Deep learning model using fastai.tabular library.
7. The two deep learning models were compared for performance accuracy.

2. MATERIALS AND METHODS

The materials and methods for developing the proposed Deep learning models for intrusion detection are described in this section. FIGURE 1 shows the common architecture for the developed IDS.

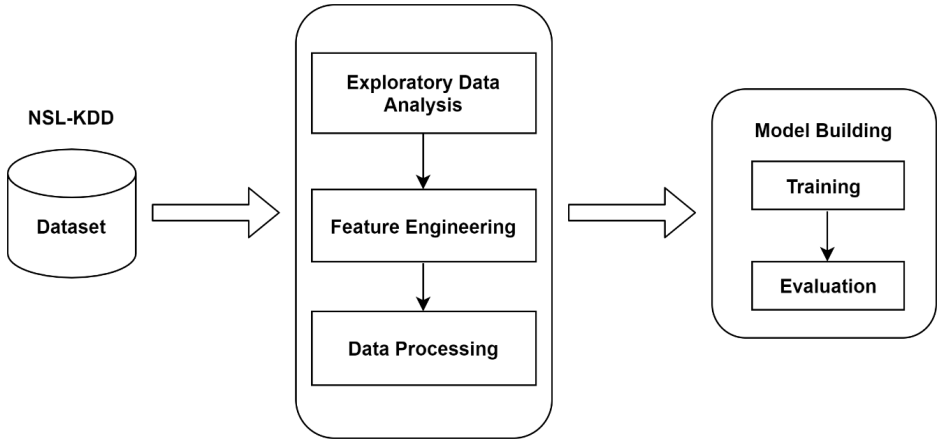

Figure 1: Architecture for Proposed Intrusion Detection System

2.1 Importation of Dataset

As shown in FIGURE 1, the initial step consisted of importing the NSL-KDD dataset from the official website of the University of New Brunswick, Canada. This was followed by exploratory data analysis and feature engineering. The NSL-KDD dataset consists of a training portion of 125973 samples and a test portion 22544 samples.

2.2 Exploratory Data analysis

The imported training data was first described as shown in TABLE 1. Furthermore, no missing or duplicate values were discovered in the data. However, some outliers were discovered in the src_bytes and dst_bytes columns as shown in FIGURE 2.

2.3 Feature Engineering for DNN

The target variable was highly skewed and imbalanced as shown in FIGURE 3, hence there was a need to class all attacks as one and encode in binary form so as to help reduce the data skew.

All attacks were encoded as '1' and normal or benign samples were encoded as '0', resulting in a distribution of '0'=67343 and '1'=58630 samples respectively.

2.4 Data Preprocessing for DNN

At this stage, categorical variables in the data such as 'protocol_type', 'service' and 'flag' were encoded using the BinaryEncoder function. The 'attack' and 'level' columns were also dropped from the data so as to prevent leakage that could bias the prediction process. Since theoutliers were few, the StandardScaler function was further implemented for standardizing the data [6], to ensure that feature distributions have mean of 0 and standard deviation of 1 as represented by Equation 1, where z=standardized score of data point, x=given datapoint, μ=mean and σ=standard deviation.

$$z = \frac{(x - \mu)}{\sigma} \tag{1}$$

2.5 Model Implementation for DNN

Neural Networks are composed of interconnected neurons such that each neuron receives input data, performs some computation, and thereafter produces an output. The inputs are multiplied by weights, summed up, and then passed through an activation function. In addition to the weights by which inputs are multiplied, the Neural Networks also incorporate bias terms. A bias term represents an additional and independent learnable parameter [7]. The developed model contained four layers; one input, two dense hidden and an output layer respectively. The input and hidden layers were implemented with the Rectified Linear Unit (ReLU) activation (Equation 2) for non-linearity introduction into the model.

$$ReLU(z) = \max(0, z) \tag{2}$$

The output layer was implemented using the Sigmoid activation function (Equation 3) since binary classification was the task being considered. The sigmoid function maps input values to the range (0, 1), making it suitable for binary classification tasks [8]. The function maps any input to a value between 0 and 1, making it useful for binary classification and logistic regression problems. The

Table 1: Description of NSL-KDD Dataset

S/N	FEATURES	DATA TYPE
0	duration	int64
1	protocol_type	object
2	service	object
3	flag	object
4	src_bytes	int64
5	dst_bytes	int64
6	land	int64
7	wrong_fragment	int64
8	urgent	int64
9	hot	int64
10	num_failed_logins	int64
11	logged_in	int64
12	num_compromised	int64
13	root_shell	int64
14	su_attempted	int64
15	num_root	int64
16	num_file_creations	int64
17	num_shells	int64
18	num_access_files	int64
19	num_outbound_cmds	int64
20	is_host_login	int64
21	is_guest_login	int64
22	count	int64
23	srv_count	int64
24	serror_rate	float64
25	srv_serror_rate	float64
26	rerror_rate	float64
27	srv_rerror_rate	float64
28	same_srv_rate	float64
29	diff_srv_rate	float64
30	srv_diff_host_rate	float64
31	dst_host_count	int64
32	dst_host_srv_count	int64
33	dst_host_same_srv_rate	float64
34	dst_host_diff_srv_rate	float64
35	dst_host_same_src_port_rate	float64
36	dst_host_srv_diff_host_rate	float64
37	dst_host_serror_rate	float64
38	dst_host_srv_serror_rate	float64
39	dst_host_rerror_rate	float64
40	dst_host_srv_rerror_rate	float64
41	attack	object
42	level	object

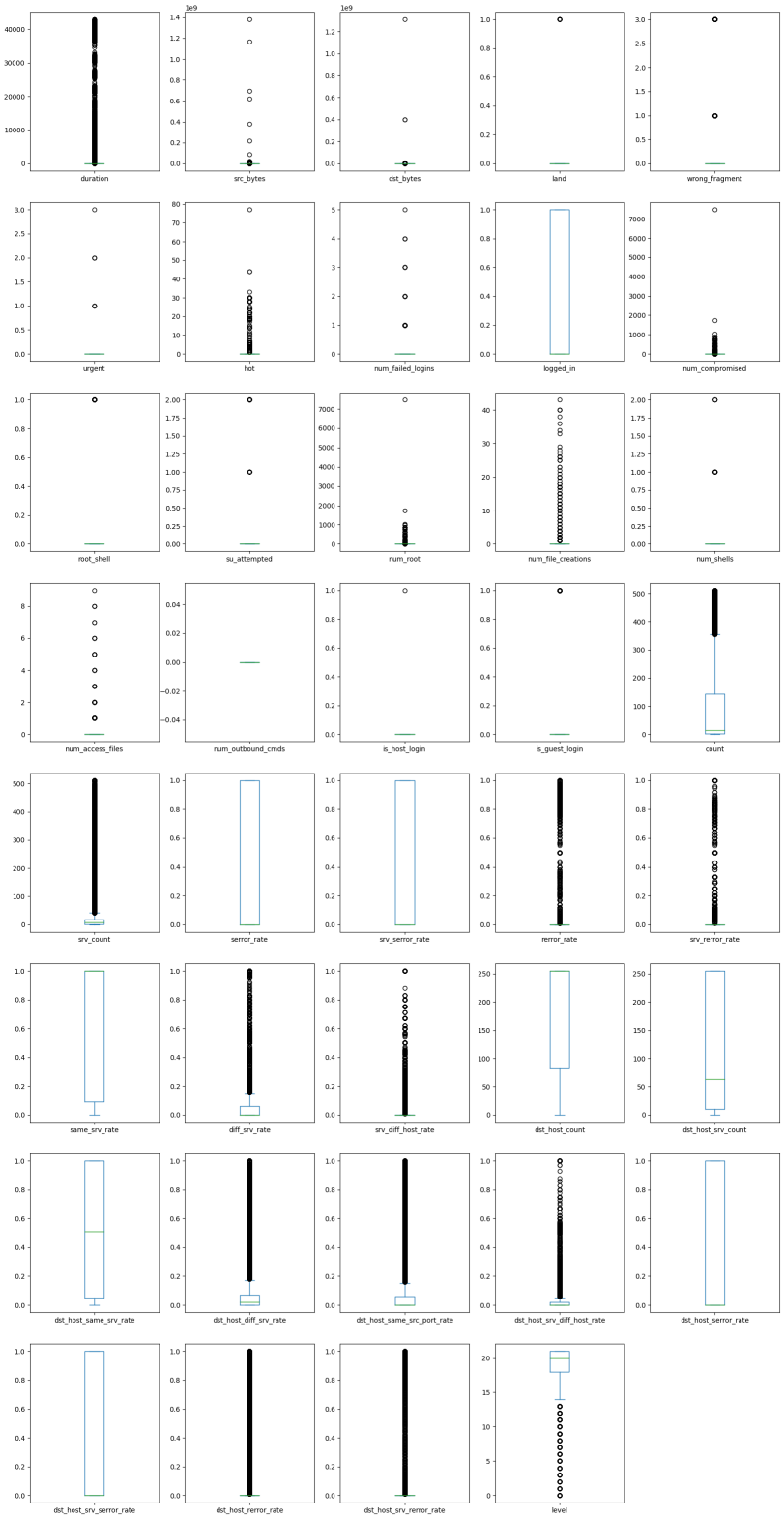

Figure 2: Outlier distribution

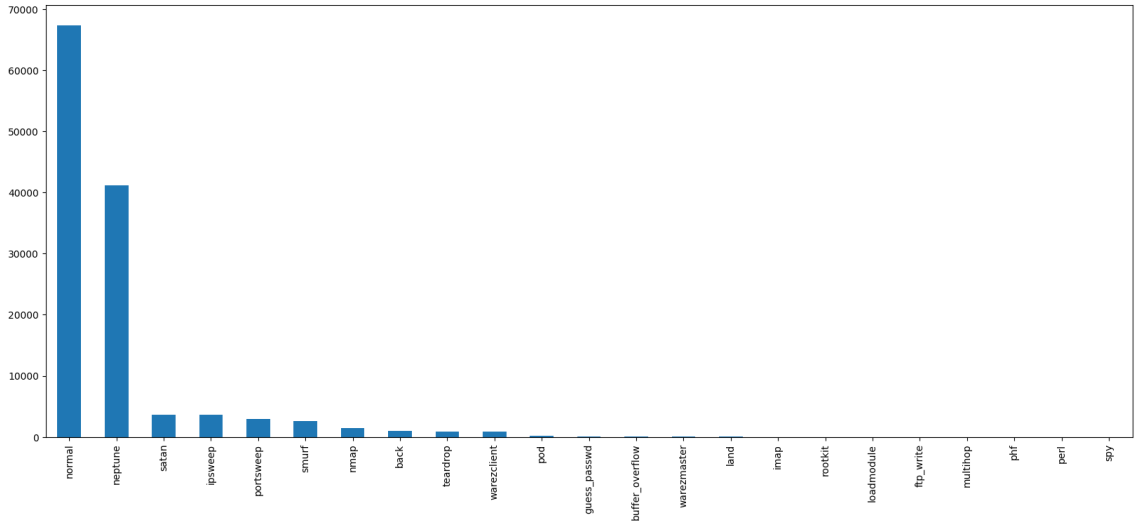

Figure 3: High Data Skew

range of the function is (0,1), and the domain is (-infinity, +infinity) [9].

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

In Equation 3, x denotes the input data values. The cost function employed for the model was the Binary Cross-Entropy (BCE) used for binary classification problems, where only two outcomes are possible. It estimates the probability of an incorrect classification. The Binary cross-entropy loss function actually calculates the average cross entropy for all data examples as represented by Equation 3.

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i.\log\left(p\left(y_i\right)\right) + \left(1 - y_i\right).\log\left(1 - p\left(y_i\right)\right) \tag{4}$$

In Equation 4, N represents the number of data points, y_i represents the actual class, $p\left(y_i\right)$ is the predicted probability of a data point being 'attack' and $1-p\left(y_i\right)$ is the predicted probability of a data point being 'normal or no attack'. Adam optimizer was implemented for modifying the attributes of the neural network and the model was then run through 500 epochs.

2.6 Implementation of Deep Learning Tabular Model

After carrying out the following preprocessing steps: encoding categorical variables, standardizing numerical features using the TabularPandas class, the train_dataloader function was used to load the training data and a learning rate of 0.001 was obtained for developing the Tabular model. Thereafter fitting was carried out. The model was run through 10 epochs after which there was no significant improvement in performance and training was stopped. The test_dataloader function was used with the test data for final evaluation of the developed model on unseen data.

3. RESULTS

3.1 First Deep Learning Model (DNN)

The developed model was implemented on a single Jupyter notebook with Python 3.10.12 using Keras 2.12.0 and TensorFlow 2.12.0 in the Google Colaboratory environment, which is an open-source platform for machine Learning-based experiments [10]. The developed deep neural network was trained based on the NSL-KDD train+ dataset. TABLE 2 represents a summary of the training results for loss and accuracy measured over 1.

Table 2: Training Results for Deep Neural Network Model

EPOCH	LOSS	ACCURACY
1	0.6928	0.5350
2	0.6918	0.5346
3	0.6891	0.5346
4	0.6819	0.5346
5	0.6639	0.6851
...	...	...
496	0.0076	0.9976
497	0.0075	0.9976
498	0.0074	0.9976
499	0.0075	0.9976
500	0.0074	0.9976

The model was finally evaluated for its predictive ability for previously unseen or zero-day attacks using the NSL-KDD test dataset. This portion of the NSL-KDD dataset contains seventeen attacks that are not present in the training portion used for training the model. FIGURE 4 displays the confusion matrix for the classification results while TABLE 3 describes the performance metrices for final model evaluation as percentage values.

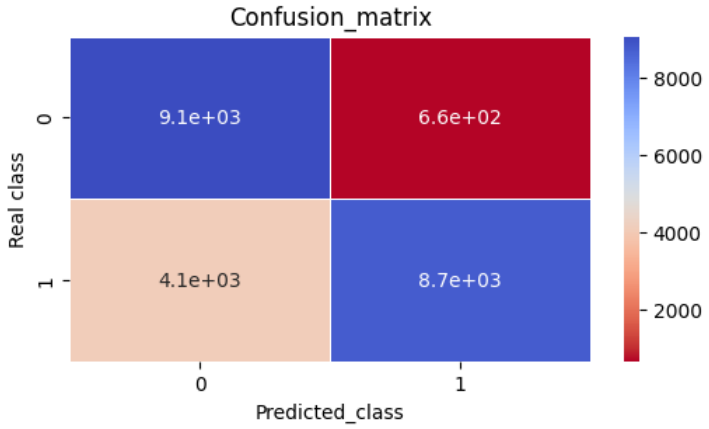

Figure 4: Confusion Matrix for Classification Results

Table 3: Binary Classification Test Results for Deep Neural Network Model

CLASS	PRECISION	RECALL	F1-SCORE	ACCURACY
NORMAL (0)	69	93	79	79
ATTACK (1)	93	68	79	79

3.2 Second Deep Learning Model (fastai Tabular)

The model was also developed in the Google Colaboratory environment. TABLE 4 below represents a summary of the training results for loss and accuracy measured over 1. TABLE 5 presents the test results for attack when the model was evaluated via the NSL-KDD test data.

Table 4: Training Results for fastai Tabular Model

EPOCH	LOSS	ACCURACY
0	0.225966	0.966579
1	0.102326	0.989005
2	0.047513	0.993133
3	0.034809	0.995158
4	0.027515	0.995396
5	0.020588	0.996229
6	0.017612	0.996587
7	0.019141	0.996467
8	0.016991	0.996706
9	0.016091	0.996626

Table 5: Test Results for fastai Tabular Model

CLASS	LOSS	ACCURACY
ATTACK	0.15813	0.841864

4. DISCUSSION

As seen in TABLE 2, the loss score had a steady decrease from the first epoch and achieved a steady crescendo that oscillated between 0.0075 and 0.0074 as the 500^{th} epoch was approached where training was stopped. Furthermore, the accuracy improved through the epoch runs and achieved a steady value of 99.76% as the 500^{th} epoch was approached. These steady values represent excellent training results for the developed model.

FIGURE 4 displays number of True Negative samples as 9053, number of False Positive samples as 658, number of False Negative samples as 4096 and number of True Positive samples as 8737. These outcomes resulted in the test scores for the performance metrices as displayed in Table 3 for binary classification. As shown, the model yielded test attack scores of 79% accuracy, 79%

F1-Score, 68% recall and 93% precision. These values represent an acceptable test performance for the developed model. TABLE 4 shows that around the tenth epoch of training, the Tabular model presented approximately similar results hence training was stopped at that epoch to yield an approximate loss value of 0.16 and accuracy of 99.66%. However, as shown in TABLE 5, the Tabular model achieved a loss value of 0.15813 and accuracy of 84.2% on the test data.

As observed above, the Tabular model outperforms the first DNN model in terms of accuracy for attack prediction when it comes to unseen data or zero-day attacks. It thus relatively presents a more favourable deep learning model for intrusion detection systems.

5. CONCLUSION

This work involved the development of an Intrusion Detection System Leveraging Deep Learning Model Classification. Two deep learning models were developed, a Deep Neural Network (DNN) with ReLU activation as well as Tabular model using the fastai deep learning library. The NSL-KDD dataset was employed and preprocessed suitably for realizing the models. Training performance metrices such as Loss and Accuracy were considered for both models, while test performance metrics such as Recall, F1-Score, Precision and Accuracy were considered for evaluating the first model. The Loss and Accuracy metrics were considered for testing the second model. Results show that both models presented an acceptable performance while detecting attacks based on the NSL-KDD dataset. However, the Tabular model performed better in terms of accuracy.

As a future scope of the work, suitable web interface or Application Programmable Interface (API) could be developed to port the models into the production environment.

References

[1] Osa E, Orukpe PE, Iruansi U. Machine Learning Based Intervention for Security in Internet of Health Things Systems. In: Proceedings of the 4th University of Benin annual research day; 2024.

[2] Rahman T. Intrusion Detection System Based on Deep Learning ,School of Science. Aalto University; 2022.

[3] Hsu CM, Hsieh HY, Prakosa SW, Azhari MZ, Leu JS. Using Long-Short-Term Memory Based Convolutional Neural Networks for Network Intrusion Detection. In: Int. wir. Internet Conference Springer Cham. 2018;86-94.

[4] Tang TA, Mhamdi L, McLernon D, Zaidi SA, Ghogho M. Deep Learning Approach for Network Intrusion Detection in Software Defined Networking. In: Int. Conf. Wireless Netw Mobile Commun, WINCOM. IEEE PUBLICATIONS; 2016;258-263.

[5] Mebawondu JO, Alowolodu OD, Mebawondu JO, Adetunmbi AO. Network Intrusion Detection System Using Supervised Learning Paradigm. Sci Afr. 2020;9:e00497.

[6] Available from: https://developers.google.com/machine-learning.

[7] Available from: http://pub.aimind.so/how-to-build-a-neural-network-from-scratch-a-step-by-step-guide.

[8] Available from: https://medium.com/@evertongomede/activation-functions-and-loss-functions-in-deep-learning-differences-and-overlaps.

[9] Available from: https://www.analyticsvidhya.com/blog/2023/01/why-is-sigmoid-function-important-in-artificial-neural-networks/Accessed.

[10] G'eron A. Hands-on Machine Learning With Scikit-Learn, Keras, and Tensorflow. 1005 Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc.; 2019;95472.