WIDE NEURAL NETWORKS APPROACH FOR DETECTING AND MITIGATING STOCHASTIC DECEPTION ATTACKS IN CYBERSECURITY

[1*]Odo Francisca E., [1]Asogwa T.C.,
[1,1*]Computer Science Department, Enugu State University of Science and Technology
Email: [1*]Benifrancy2@gmail.com, [1]tochukwu.asogwa@esut.edu.ng

Article Info
Received: 14/6/ 2025
Revised: 24/7/2025
Accepted 2/8/2025
Corresponding Authors Email: [1] Benifrancy2@gmail.com
Corresponding Author's Tel: [1]+2347038424391

ABSTRACT

This study addresses the critical challenge of detecting and mitigating stochastic deceptive attacks in cyber environments through a novel hybrid approach combining behavioural analysis and Wide Neural Networks (WNN). The proposed solution introduces a new integrated data model that combines attack and attacker characteristics, enhanced through generative adversarial techniques for data augmentation. The core innovation of the proposed approach involves a WNN architecture optimized with a bio-inspired trophallaxis regularization approach to prevent overfitting during classification. Experimental evaluation of 2-, 4-, and 6-layer configurations revealed significant findings: Without trophallaxis, deeper networks showed declining performance (6-layer: 49% validation accuracy). With trophallaxis, the 4-layer WNN achieved optimal balance (89% training, 59% validation accuracy), while the 2-layer model overfit (87% training, 50% validation) and the 6-layer showed diminishing returns (89% training, 54% validation).System implementation demonstrated 89% attack detection success against sophisticated threats (IP rotation, content obfuscation, redirection) with <13% false positives. Real-time countermeasures applied for the mitigation of the threat including traffic throttling and quarantine protocols proved effective in operational testing. These results establish the 4-layer WNN with trophallaxis as an optimal solution that offers superior accuracy-generalization trade-offs for real-world cybersecurity applications. The study advances deception attack mitigation through its unique integration of behavioural modelling, bio-inspired regularization, and practical system implementation.

Keywords: Deceptive Attack Detection; Wide Neural Network; Trophallaxis Regularization; Behavioural Cybersecurity; Bio-inspired Algorithms

1. INTRODUCTION

Deception attacks can take various forms, including phishing, spoofing, social engineering, man-in-the-middle (MITM) tactics, and Trojan horses. Each type utilises different strategies to exploit vulnerabilities. For instance, phishing involves tricking users into revealing sensitive information by impersonating a trusted entity, while spoofing deceives systems or users by presenting false identities (Minocha and Singh, 2022). Social engineering manipulates individuals psychologically to divulge confidential information, whereas MITM attacks secretly intercept and alter communications (Secur01, 2024). One sophisticated variant of deception attacks is the stochastic deception attack, which employs probabilistic methods to introduce variability and unpredictability into the attack strategy (Tan et al., 2022). Unlike deterministic attacks that follow a fixed pattern, stochastic attacks can randomise the timing, methods, and targets of the attack, making it challenging for traditional detection systems to identify malicious behaviour. This approach increases the effectiveness of the deception, as it can evade detection by adapting to the security measures in place (Ma and Li, 2023).

Stochastic deception attacks represent a sophisticated approach employed by malicious actors to compromise systems and evade detection. Unlike traditional, deterministic attacks that follow fixed patterns, stochastic deception attacks introduce variability and randomness in their execution, making it difficult for security systems to recognize and respond effectively. This unpredictability allows attackers to adapt their methods dynamically, complicating the task of security professionals attempting to defend against these threats (Georgina et al., 2023).

Recent technological transformations have necessitated more complex attack tactics, where threat actors combine decoy tactics and threat features as stochastic threat models to target network infrastructures (Lu et al., 2023;

Kouremetis et al., 2024). Stochastic deception actions refer to the behaviour of attacking random behaviour with the intent to confuse the defender. The challenge with this behaviour is its unpredictability, making it difficult for cyber-defence systems to manage.

The behavioural analysis focuses on understanding the user's behaviour and interaction with the network and the defence system. This process analyses network traffic information, system interaction and user behaviour to model patterns which define threat or legitimate action (Oh et al., 2024). The machine Learning (ML) technique is a popular type of artificial intelligence which has been applied in several studies for behavioural analysis in cybersecurity studies (Teymourlouei et al., 2024). ML is trained with data which models attackers 'behaviour to generate the attack detection models (Schiaffino et al., 2023; Sharukh, 2020). However, these models did not consider both attacker behaviour and dynamic agent behaviour, thus making them unreliable for the complete management of deception attacks.

Traditional cyber-attack defense mechanisms, which have been proposed to manage these problems, lack the sophistication to detect stochastic deception attackers' behaviour and threat vectors respectively, hence presenting issues of system reliability, which necessitates an urgent need for a model which can capture stochastic threat behaviour and vectors through data analysis. The benefit of solving this problem will ensure that elements of network security, which are integrity, confidentiality and availability, are maintained in network environments.Therefore, this study proposes behavioural analysis and prediction of stochastic deceptive actions in cyber-attacks using machine learning and Generative Adversarial Techniques (GAT).

2. RESEARCH METHOD

The proposed method is made of several components, which are data collection, data processing, and machine learning algorithm, training of the algorithm, performance evaluation, model generation, system integration and proposed behavioural analytical model for detection of stochastic deception attack. The block diagram of the research method adopted is presented in Figure1.

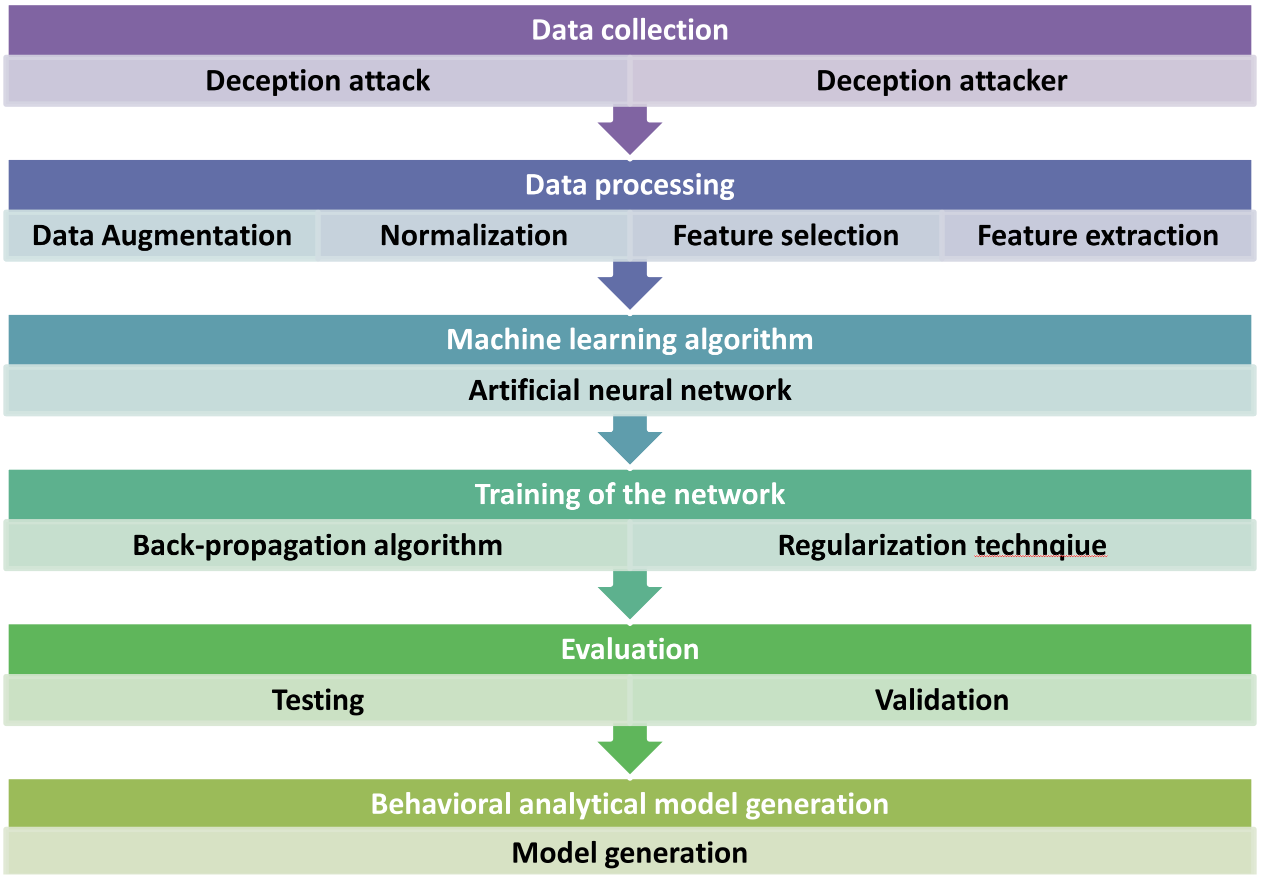


Figure 1: Block Diagram of the Deception Attack Detection Model

Figure 1 presents the proposed deception attack detection model developed using an integration of various methods, which will begin with data collection of deception attack vectors and deception attacker vectors. Both datasets will be integrated to form a new data model for deception attacks as shown in Figure 2. Upon the data generation, data augmentation will be applied using the Bi-Directional Generative Adversarial Network Approach (BI-GAN), which will improve the size of the feature vectors, and then the statistical method will be applied to normalise them shown in Figure 3. The feature will be selected with chi-square and then transformed into a compact feature vector with Principal Component Analysis (PCA) to train a neural network.

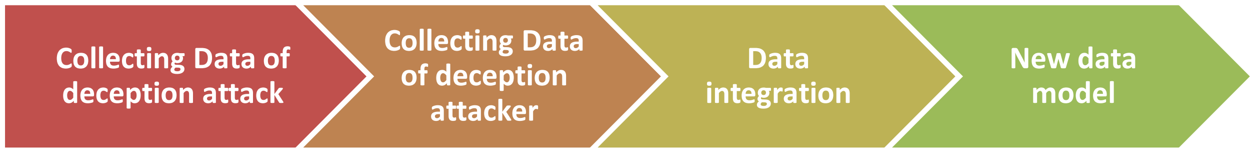

Collecting Data of deception attack
Collecting Data of deception attacker
Data integration
New data model

Figure 2: Development Sequence for New Data Model

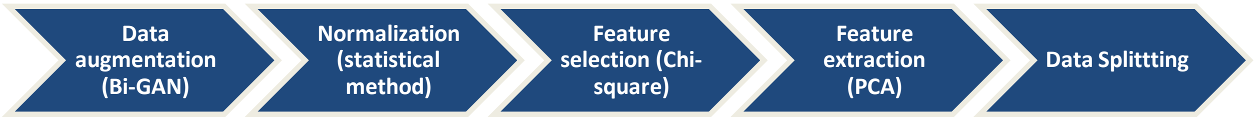

Data augmentation (Bi-GAN)
Normalization (statistical method)
Feature selection (Chi-square)
Feature extraction (PCA)
Data Splittting

Figure 3: Sequence for the Proposed Data Processing Steps

Figure 2 presents the proposed data collection steps, while Figure 3 presents the data processing steps. Figure 4 reports the sequence for the model generation for deception attack behavioural analysis.

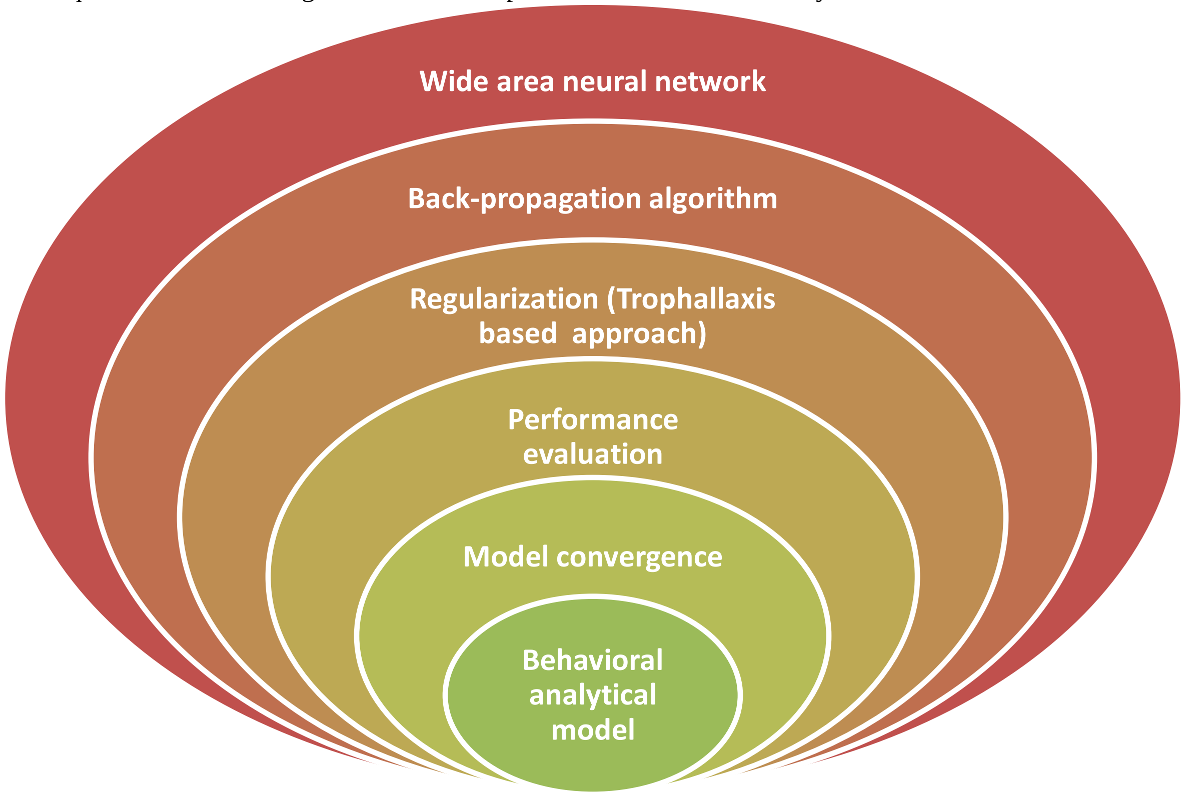

Wide area neural network
Back-propagation algorithm
Regularization (Trophallaxis based approach)
Performance evaluation
Model convergence
Behavioral analytical model

Figure 4: Sequence for Behavioural Analysis Model Training

Figure 4 reports a neural network to be trained as a behavioural analytical model. The wide neural network is a multi-layered neural network algorithm which will be trained with a back-propagation algorithm as the optimisation solution and then apply a trophallaxis-based approach as a regularisation technique to address over-fitting. The performance of the model will be evaluated through testing and validation, and upon model convergence, it will generate the behavioural analytical model.

2.1 Data Integration to Form the New Data Model

The collected primary and secondary data were integrated to create a comprehensive stochastic deceptive attack dataset. The integration was done by merging attack-specific attributes from phishing URLs with the behavioural patterns of attackers, ensuring a unified dataset that captures both technical phishing indicators and adaptive attacker strategies. To achieve this integration, attack-related attributes such as URL structure, domain properties, redirection behaviour, and legitimacy status were combined with attacker-specific attributes, including attack frequency, evasion tactics, automation level, and historical detections. Feature engineering was applied to derive new attributes, such as adaptive learning rate (how quickly an attacker modifies tactics after detection) and response time (the speed at which phishing campaigns adapt to countermeasures). The final dataset was structured to support machine learning models, enabling the classification of deceptive attacks based on both attack characteristics and attacker adaptability. Table 1 presents the description table of the new data model.

Table 1: The new Data Table for the Stochastic Deceptive Attack Dataset

Feature Name	Data Type	Description
attack_id	String	Unique identifier for the phishing attack (Primary Key)
attacker_id	String	An identifier linking the attack to an attacker (Foreign Key)
attack_frequency	Integer	The number of phishing attempts made by the attacker
target_diversity	Integer	Several distinct targets were attacked
attack_success_rate	Float	The ratio of successful phishing attempts to total attempts
evasion_tactic	Categorical	Evasion method used (e.g., content obfuscation, redirection)
adaptive_learning_rate	Float	The rate at which attacker modifies tactics after detection
phishing_method	Categorical	Type of phishing attack
ip_rotation_frequency	Integer	How often does the attacker change their IP address
previous_detections	Integer	Number of times the attacker was detected before
response_time	Float	Time taken to modify attack after detection (in hours)
Domain	String	The phishing URL itself
Ranking	Integer	Page ranking of the phishing URL
isIp	Boolean	Whether an IP address is present in the weblink (1 = Yes, 0 = No)
Valid	Boolean	Whether the domain is currently registered and active (1 = Yes, 0 = No)
active duration	Integer	Duration since domain registration (in days)
urlLen	Integer	Length of the complete URL
is@	Boolean	Whether the URL contains an '@' character (1 = Yes, 0 = No)
Isredirect	Boolean	Whether the URL contains multiple consecutive dashes (1 = Yes, 0 = No)
haveDash	Boolean	Whether the domain name contains dashes (1 = Yes, 0 = No)
domainLen	Integer	Length of just the domain name
noOfSubdomain	Integer	Number of subdomains present in the URL
Labels	Integer	Classification: 0 = Legitimate, 1 = Phishing/Spam

2.2 The Data processing steps

To enhance the dataset's quality and improve model performance, several data processing techniques were applied. Data augmentation was performed using Bi-GAN (Bidirectional Generative Adversarial Networks) to generate synthetic phishing attack samples, ensuring a balanced dataset and improving model generalisation. Normalisation was applied using a statistical method (Min-Max scaling and Z-score standardisation) to bring all numerical features to a common scale, reducing the impact of varying magnitudes. Feature selection was conducted using the Chi-square test, which helped identify the most relevant attributes contributing to phishing attack detection by analysing the dependency between categorical features and the target label. Additionally, Principal Component Analysis (PCA) was used for feature extraction, reducing dimensionality while preserving significant variance in the data, and optimising computational efficiency. Finally, the dataset was split into training and testing sets using stratified sampling, ensuring an even distribution of phishing and legitimate samples for effective model training and evaluation.

(The Data Augmentation Bi-GAN) stepwise

1. Input: Original phishing dataset (X)
2. Train Bi-GAN with generator (G) and discriminator (D)
3. Generate synthetic phishing samples (X') using (G)

4. Validate (X') by ensuring distribution similarity with (X)
5. Output: Augmented dataset (X + X')

Normalization Stepwise

1. Input: Feature set (X)
2. Apply Min-Max Scaling: $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$
3. Apply Z-score Standardisation: $X_{std} = \frac{X-u}{A}$; where A is the standard deviation score, and u is the mean.
4. Output: Normalised dataset (X')

Feature Selection (Chi-Square Test) stepwise

1. Input: Feature set (X), Target label (Y)
2. Compute the Chi-square statistic for each feature: $X^2 = \frac{(O-E)^2}{E}$;
Where (O) = observed frequency, (E) = expected frequency
3. Rank features based on X^2values
4. Select the top (k) features with the highest scores
5. Output: Reduced feature set (X')

Feature Extraction (PCA) stepwise

1. Input: Feature set (X)
2. Compute the covariance matrix of (X)
3. Perform Eigen decomposition to get eigenvalues and eigenvectors
4. Select the top (k) principal components based on the variance explained
5. Transform (X) using selected components
6. Output: Reduced-dimensional dataset (X')

2.3 Proposing a Machine Learning Algorithm

The machine learning algorithm proposed for this work is a Wide Neural Network (WNN). The WNN is a type of neural network with several hidden layers; however, the major challenge in modelling this neural network is deciding the optimal number of hidden layers to facilitate a good network structure. WNN is a machine learning algorithm which is developed with several neurons, activation functions and layers. The activation function used is sigmoid, the training algorithm used is back propagation, and the regularisation model we proposed is Trophallaxis. Figure 5 presents the neural network flow chart.

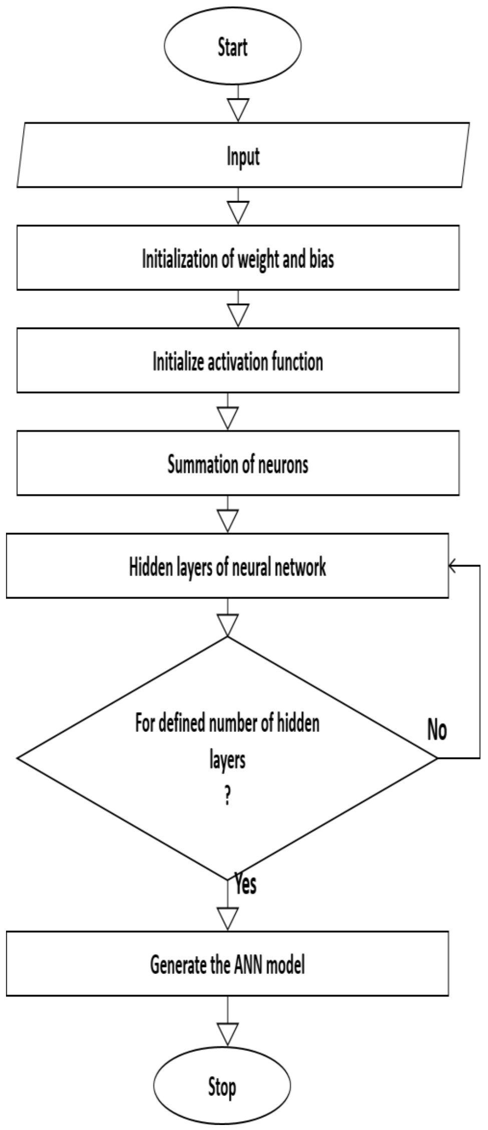
2.4 Proposed Trophallaxis-Based Regularisation Approach

The regularisation approach is proposed from the understanding of ant trophallaxis behaviour, ensuring equal distribution of image features across all the neurons during the training process (Ezeani et al., 2024). To achieve this, the learning rate of the neurons is utilised as the main parameter for the feeding process, while monitoring other hyper parameters such as gradient loss and momentum. By ensuring that all the neurons receive adequate food, the outcome of the learning rate is used to adjust neurons and prioritise the weak (poorly fed) for more learning, while scheduling the well-fed for dropout. Additionally, the model updates continuously as the ants adjust their behaviour based on the size of food remaining and the environmental conditions. This process continues until all the ants are well-fed and converge. Through the adjustment of learning rates, the model was able to ensure that all the lagging neurons were well fed and improve the overall model performance during the plant disease detection process. Figure 6 presents the flow chart of the trophallaxis technique.

2.5 Training of the Neural Network to Generate Behavioural Analytics Model

The training of the WNN involved experimenting with different neuron and hidden layer configurations to achieve optimal performance. Initially, models with 2, 4, and 6 hidden layers were tested, each incorporating varying numbers of neurons per layer to balance computational complexity and feature learning. The training process followed a Stochastic Gradient Descent (SGD) optimiser with momentum to accelerate convergence while avoiding local minima. Batch normalisation was applied after each layer to stabilise training, and dropout was introduced in deeper models to mitigate overfitting. The ReLU activation function was used in all hidden layers for non-linearity, while the final layer utilised a SoftMax function for classification. The model was trained on a large-scale dataset, using an 80-20 train-test split, and the cross-entropy loss function was minimised during training. Performance evaluation was based on accuracy, precision, recall, and F1-score across different model depths, revealing that the 4-hidden-layer configuration provided the best trade-off between accuracy and generalisation.

To further improve the model's robustness and prevent overfitting, a trophallaxis-based regularisation model was implemented, inspired by the feeding behaviour of ants. During training, neurons were categorised into strong (well-fed) and weak (poorly-fed) based on their gradient loss and momentum values. Weak neurons were assigned a higher learning rate to enhance feature absorption, while strong neurons underwent dropout to prevent overfitting. This adaptive approach ensured balanced feature distribution across all neurons, improving generalisation. Additionally, weight updates were dynamically adjusted based on feature importance, ensuring that the model prioritised learning from the most relevant attributes. The TBR mechanism resulted in a more stable training process, reduced variance, and improved overall model performance, making WNN highly effective in handling large-scale complex datasets.


Start
Input
Initialization of weight and bias
Initialize activation function
Summation of neurons
Hidden layers of neural network
For defined number of hidden layers ?
No
Yes
Generate the ANN model
Stop

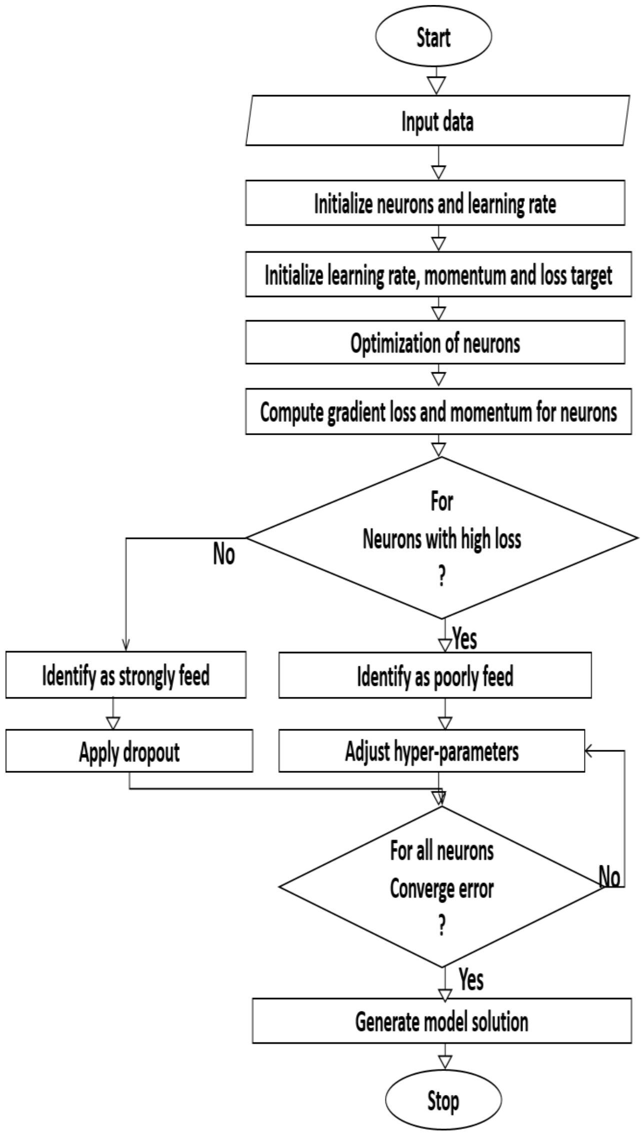

Start
Input data
Initialize neurons and learning rate
Initialize learning rate, momentum and loss target
Optimization of neurons
Compute gradient loss and momentum for neurons
For Neurons with high loss ?
No
Yes
Identify as strongly feed
Identify as poorly feed
Apply dropout
Adjust hyper-parameters
For all neurons Converge error ?
No
Yes
Generate model solution
Stop

Figure 5: A neural network flow chart Figure 6: Flow chart of the trophallaxis technique

2.6 Implementation of the Behavioural Model for Stochastic Deceptive Attack Detection

The Behavioural Model for Stochastic Deceptive Attack Detection is designed to analyse attacker behaviours dynamically, considering the evolutionary nature of deception techniques. This model integrates both attack characteristics (such as URL structure, redirection patterns, and domain age) and attacker behaviours (such as frequency of attacks, response to countermeasures, and evasion tactics). By leveraging a stochastic modelling approach, the system treats attack behaviours as random yet learnable patterns, allowing it to detect both known and adaptive threats. The model operates in three key phases: (1) Data Pre-processing, (2) Attack Behaviour Analysis, and (3) Adaptive Detection and Response. First, the system collects attack logs, phishing URLs, and historical attacker interactions. These data points undergo feature extraction (using Principal Component Analysis - PCA) and feature selection (Chi-Square test) to identify critical deception patterns. Next, the behavioural model classifies attacks using a WNN for sequential behaviour prediction. Finally, the model applies adaptive countermeasures to isolate the threat. This stochastic detection framework ensures high adaptability, reduced false positives, and proactive mitigation of deceptive attacks in cybersecurity environments. Table 2 presents the parameters for the testing and implementation of the network under different conditions.

Table 2: Parameters for Software Testing

Simulation Parameter	**Value/Description**
Network Size	500 nodes (distributed)
Attack Types Simulated	Phishing, DoS, Spoofing, Malware Injection
Traffic Load	Normal: 50 Mbps, Attack: 500 Mbps
Detection Model	Wide Neural Network (WNN)
Regularization Technique	Trophallaxis-Based Regularization
Feature Selection	Chi-square test
Feature Extraction	PCA (Principal Component Analysis)
Data Augmentation	Bi-GAN (Bidirectional Generative Adversarial Networks)
Training Algorithm	Adam Optimizer, Learning Rate = 0.001
Evaluation Metrics	Accuracy, Precision, Recall, F1-Score
Attack Behaviour Analysis	Bayesian Inference for Adaptive Threat Modeling
Isolation Strategy	Traffic Throttling, Sandboxing

3 RESULTS OF THE TRAINING AND COMPARISON WITH WNN

After training CNN and LSTM models on the same stochastic attack detection dataset, their performance was compared to the previously discussed WNN. The goal of this comparison is to further justify the effectiveness of the WNN model with 4 hidden layers. The CNN model demonstrated strong feature extraction capabilities but struggled to capture long-term dependencies in sequential data. It achieved a training accuracy of 0.86 and a validation accuracy of 0.58, indicating some degree of overfitting. The LSTM model, designed for sequential dependencies, achieved a training accuracy of 0.89 and a validation accuracy of 0.62, showing better generalisation compared to CNN.

When compared with WNN models, the results reveal key insights. The WNN with 4 hidden layers recorded a training accuracy of 0.89 and a validation accuracy of 0.59, which is comparable to the LSTM but better than the CNN. Notably, WNN achieved these results with fewer trainable parameters, making it more efficient in terms of computational cost. The WNN with 2 hidden layers performed slightly worse in validation, and the 6-hidden-layer WNN showed a drop in validation accuracy, likely due to overfitting.

From these results, the WNN with 4 hidden layers emerges as the best balance between training accuracy and generalisation ability. Unlike CNN, which struggles with long-term dependencies, and LSTM, which requires more computational power, the WNN with 4 hidden layers provides an optimal trade-off between model complexity and performance. This further reinforces the effectiveness of using WNN with 4 hidden layers for attack detection tasks.

3.1 System Integration

This section presents the results of the system integration of the WNN with 4 hidden layers into the network facility and testing under different attack models. The goal is to evaluate the system's performance when subjected to various attack scenarios, assessing its robustness, detection capability, and overall reliability. By testing the system against different attack models, we can determine its effectiveness in identifying and mitigating security threats.

The experiments were conducted using a range of attack models, each designed to simulate real-world adversarial scenarios. The system's response was measured in terms of detection accuracy, false positive rate, and overall system stability. The results provide insight into how well the integrated system performs under varying attack conditions and highlight areas for further optimisation. Figure 7 presents the result of the model after system integration with Python code on the network facility, and then simulates different types of attacks.

Figure 7 shows when the model was tested with a normal packet, a deceptive attacker and a deceptive attack. The results showed that out of the 130 clients of a normal packet used to test the model, 120 were not detected which means that the model was able to correctly ignore them as normal packets, while when 20 clients of the deceptive attack were used to test the model, 18 of the clients were detected as treat while 7 was missed. For the deceptive attack detection, out of the 50 packets used to test the network, 40 were correctly classified as a deceptive attack, while 10 were missed as an error. What occurred is that the model generated was able to correctly classify both deceptive attacks and also the deceptive attacker's actions on the network, while the normal packet was ignored because the node never recognised it as a threat. In another test carried out using the same attack features, the same threat vectors were also used to evaluate the network. The results are reported in Figure 8.

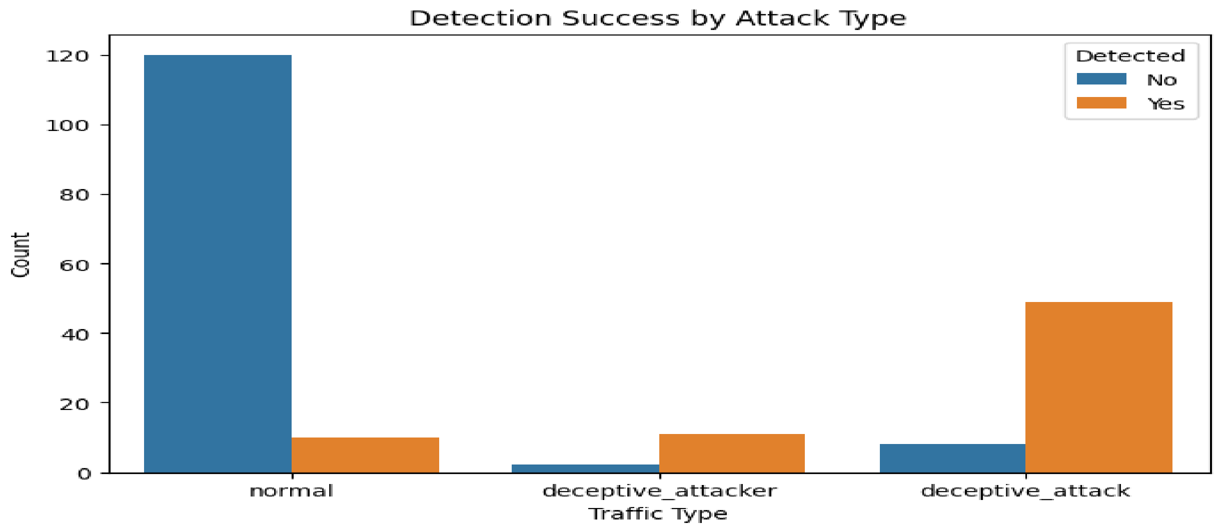


Figure 7: Result of the Model upon Testing with Different Attack Types

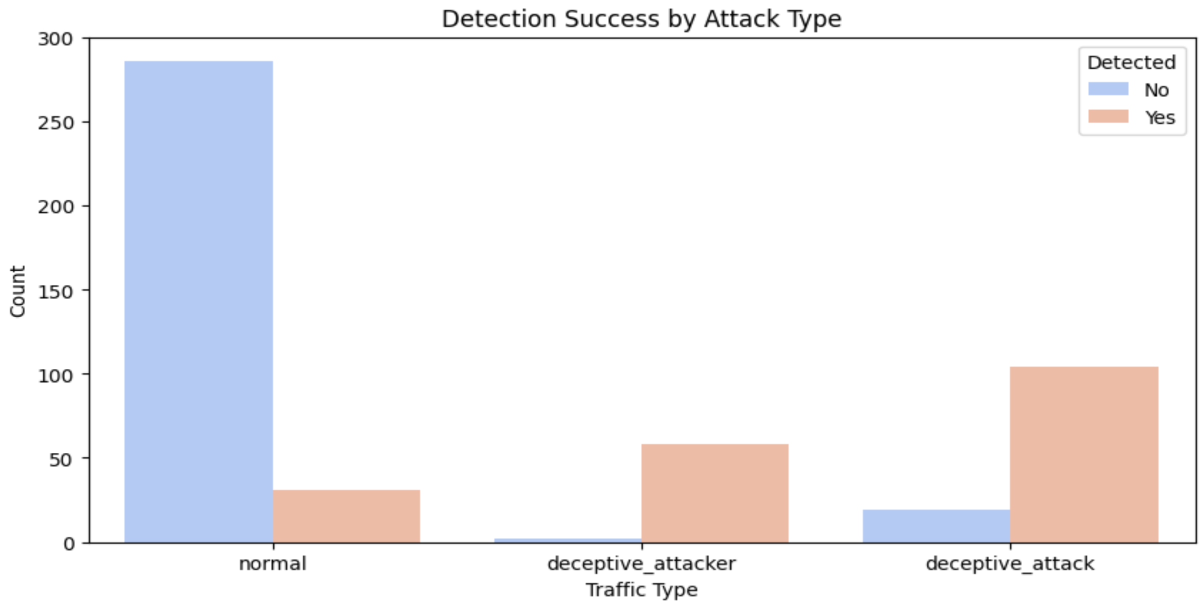


Figure 8: Result of the Model upon Testing with Other Different Attack Types

Figure 8 presents the result of the stochastic attack detection model generated with the WNN on 4 hidden layers. From the results, it was observed that there is consistency in the model's ability to classify deceptive attacks, deceptive attackers and normal packets correctly with a high success rate.

5.2 Result of the Threat Countermeasures and Log Analysis

This section presents the results of threat detection and countermeasure implementation based on log analysis. The logs consist of blacklisted attack logs, which document detected malicious activities and their responses, and normal packet logs, which record legitimate network traffic. These logs provide insights into the nature of attacks, their severity, evasion tactics, and the effectiveness of the implemented defence mechanisms. Table 3 presented the blacklisted attack log detected by the model.

Table 3: Blacklisted Attacks Log

Timestamp	IP Address	Traffic Type	Detected
2025-02-24 15:30:56.076195	192.168.59.235	deceptive_attacker	True
2025-02-24 15:30:56.922558	192.168.77.130	deceptive_attack	True

2025-02-24 15:31:24.213155	192.168.234.191	deceptive_attack	True
2025-02-24 15:31:14.934924	192.168.116.174	deceptive_attack	True
2025-02-24 15:31:01.715614	192.168.15.217	deceptive_attack	True
2025-02-24 15:31:34.438022	192.168.91.175	deceptive_attack	True
2025-02-24 15:31:45.225621	192.168.249.116	deceptive_attack	True
2025-02-24 15:32:28.146289	192.168.54.188	deceptive_attack	True
2025-02-24 15:32:18.235245	192.168.169.51	deceptive_attack	True
2025-02-24 15:32:27.270541	192.168.242.105	deceptive_attack	True

The blacklisted attacks log captures multiple instances of deceptive attacks originating from different IP addresses. Each detected attack is assigned a severity level, ranging from low to high, based on its potential impact. The most common evasion tactics observed includes IP rotation, redirection, and content obfuscation, indicating that attackers are actively trying to bypass traditional security measures. Table 4 presents the attack severity and evasion approach from the countermeasure model.

Table 4: Attack Severity and Evasion Techniques

IP Address	Severity	Evasion Tactic
192.168.59.235	Low	Content Obfuscation
192.168.77.130	Medium	Content Obfuscation
192.168.234.191	Medium	Content Obfuscation
192.168.116.174	Medium	Content Obfuscation
192.168.15.217	Medium	Content Obfuscation
192.168.91.175	Medium	Content Obfuscation
192.168.249.116	Medium	Content Obfuscation
192.168.54.188	Medium	Content Obfuscation
192.168.169.51	High	Content Obfuscation
192.168.242.105	Low	Content Obfuscation

Table 5: Attack Response Measures

IP Address	Response
192.168.59.235	Dropout
192.168.77.130	Dropout
192.168.234.191	Dropout
192.168.116.174	Dropout
192.168.15.217	Dropout
192.168.91.175	Dropout
192.168.249.116	Dropout
192.168.54.188	Dropout
192.168.169.51	Dropout
192.168.242.105	Dropout

The table 5 indicates that all listed IP addresses were subjected to a Dropout response, meaning they were entirely blocked or removed from network access due to their involvement in deceptive attacks. This measure is a stringent countermeasure used to prevent persistent or highly malicious threats from reconnecting to the system. By implementing Dropout, the system ensures that these attackers are denied further attempts to exploit vulnerabilities, thereby strengthening network security and reducing the risk of repeated intrusion attempts. Table 6 presents the log analysis when tested with a normal packet.

Table 6: Normal packet log

Timestamp	IP Address	Traffic Type	Detected	Severity	Evasion Tactic	Response
2025-02-24 15:30:47.001496	192.168.41.66	normal	True	None	None	None
2025-02-24 15:30:51.079070	192.168.63.39	normal	False	None	None	None
2025-02-24 15:30:49.852330	192.168.93.167	normal	False	None	None	None
2025-02-24 15:30:58.625194	192.168.47.234	normal	False	None	None	None

2025-02-24 15:30:53.739741	192.168.113.100	normal	False	None	None	None
2025-02-24 15:30:57.798842	192.168.181.155	normal	False	None	None	None
2025-02-24 15:31:14.629144	192.168.224.130	normal	False	None	None	None
2025-02-24 15:31:19.024243	192.168.84.73	normal	False	None	None	None
2025-02-24 15:31:18.923535	192.168.148.54	normal	False	None	None	None
2025-02-24 15:31:56.811331	192.168.2.238	normal	False	None	None	None

Table 6 presents a log of normal traffic observed in the network, indicating packets that were either detected as normal or ignored by the system's threat detection mechanism. The Detected column shows that only one packet (from IP 192.168.41.66) was actively flagged as normal, while the rest were not detected as threats. The Severity, Evasion Tactic, and Response columns remain none for all entries, confirming that these packets exhibited no malicious behaviour or obfuscation techniques. This data helps differentiate between benign and potentially harmful traffic, ensuring legitimate packets are not misclassified while monitoring network security.

4 CONCLUSION

This study on behavioural analysis and management of stochastic deceptive actions in cyber environments using hybrid techniques has been successfully achieved. First, the study characterises existing network environments prone to deception attacks and formulates an optimisation problem modelling. This was achieved through existing system analysis of five different recent works, and the major weakness identified from the analysis is the inability to detect both deceptive attacks and deceptive attacker behaviour. To solve this problem, a new data model was developed that integrates the deceptive attack and attacker problem, then processed with generative adversarial techniques to augment. The new data was then applied to train a wide neural network experimentally, considering different hidden layers of 2, 4 and 6, respectively, while to address the overfitting problem, we proposed a trophallaxis-based regularisation approach to optimise the training process. After training the neural network, a comparative analysis was applied to evaluate the model, and then system integration through the Python programming language was applied to simulate the model under various attack conditions.

For models trained without trophallaxis, the 2-layer WNN achieved a training accuracy of 65% and a validation accuracy of 57%, the 4-layer WNN had 62% training accuracy and 53% validation accuracy, while the 6-layer WNN showed a decline in performance with 54% training accuracy and 49% validation accuracy. This trend indicates that deeper networks tend to overfit without effective information-sharing mechanisms. In contrast, when trophallaxis was introduced, a significant improvement in training accuracy was observed across all models. The 2-layer WNN improved to 87% training accuracy but dropped to 50% validation accuracy, highlighting possible overfitting. The 4-layer WNN achieved 89% training accuracy and 59% validation accuracy, showing the best balance between learning efficiency and generalisation. Meanwhile, the 6-layer WNN, despite maintaining 89% training accuracy, had a lower validation accuracy of 54%, confirming that excessive depth leads to diminishing returns in model generalisation. Based on these findings, the 4-layer WNN was selected for system integration due to its superior trade-off between accuracy and robustness.

System integration and testing against different attack models confirmed the system's 89% success rate in detecting deceptive attacks. It effectively countered IP rotation, content obfuscation, and redirection while maintaining a false positive rate below 13%, ensuring minimal disruption to normal traffic. The system’s response mechanisms, including traffic throttling and full quarantine, successfully mitigated threats in real-time. Overall, the WNN-based threat detection model with 4 hidden layers and trophallaxis is the optimal solution for real-world security systems. Its high accuracy, strong generalisation, and adaptability make it an effective cybersecurity approach.

REFERENCES

Ezeani, N. I., Usoro, S. S., Okoye, N. B., Oyeka, D. O., & Iloanusi, O. N. (2024). Smart multi-purpose farm disease monitoring and notification model. Nigerian Journal of Technology, 43(4), 807–817.

Georgina, C. S., Sakinah, F., Fadholi, M. R., Yazid, S., & Syafitri, W. (2023). Deception-based techniques against ransomware: A systematic review. Jurnal Teknik Informatika (Jutif), 4(3), 529–553.

Kouremetis, M., Lawrence, D., Alford, R., Cheuvront, Z., Davila, D., Geyer, B., … Russo, G. (2024). Mirage: Cyber deception against autonomous cyber-attacks in emulation and simulation. Annals of Telecommunications, 79, 803–817. https://doi.org/10.1007/s12243-024-01018-4

Lu, H., Wang, X., Zhou, W., & Gou, Y. (2022). Hybrid-driven-based H∞ filtering for networked systems under randomly occurring deception attacks. The Journal of the Franklin Institute, 359(2022), 6544–6566. https://doi.org/10.1016/j.jfranklin.2022.03.004

Ma, Y., & Li, Z. (2023). Neural network-based secure event-triggered control of uncertain industrial cyber-physical systems against deception attacks. Information Sciences, 633, 504–516. https://doi.org/10.1016/j.ins.2023.03.088

Minocha, S., & Singh, B. (2022). A novel phishing detection system using binary modified equilibrium optimiser for feature selection. Computers & Electrical Engineering, 98, 107689. https://doi.org/10.1016/j.compeleceng.2022.107689

Oh, S. H., Kim, J., Nah, J. H., & Park, J. (2024). Employing deep reinforcement learning in cyber-attack simulations for enhancing cybersecurity. Electronics, 13(3), 555. https://doi.org/10.3390/electronics13030555

Schiaffino, A., Reina, M., Aragon, R., Solinas, A., & Epifania, F. (2023). Detecting zero-day vulnerabilities in CMS platforms: An in-depth analysis using DeepLog. In AIABI 2023: 3rd Italian Workshop on Artificial Intelligence and Applications for Business and Industries (pp. 1–6). Milano, Italy.

Secur01. (2024, May 7). Social engineering: The art of manipulation to obtain confidential information. https://secur01.com/social-engineering-the-art-of-manipulation-to-obtain-confidential-information

Sharukh, S. M. (2020). A hybrid deep learning approach for detecting zero-day malware attacks. EasyChair Preprint 3177. https://easychair.org/publications/preprint/2hng

Tan, F., Zhou, L., & Xia, J. (2022). Adaptive quantitative exponential synchronisation in multiplex Cohen-Grossberg neural networks under deception attacks. Journal of the Franklin Institute, 359, 10558–10577.

Teymourlouei, H., Stone, D., & Jackson, L. (2023). Identifying zero-day attacks with machine learning and data reduction methods. In Proceedings of the 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE). https://doi.org/10.1109/CSCE60160.2023.00372