CHAPTER 6

ENHANCING ZERO-DAY ATTACK PREDICTION A HYBRID GAME THEORY APPROACH WITH NEURAL NETWORKS

6.1 Introduction

The other approaches that have been used in the zero-day attacks are limited in some way to predict the attacks in advance. Bayesian and probabilistic models provide the result of the static forecasting, yet are always insufficient to tackle the problem of the resource asymmetry, dependencies between different nodes, and the multi-layer abstraction of a cloud network. To address these issues, this chapter presents a Hybrid Prediction Framework that uses an ANN-based Autoencoder to reduce the size of behavior features, a Modified Bi-LSTM network that is introduced to recognize sequential dependencies and evolving patterns of attack and a Hybrid Game-Theoretic Model that identifies strategic attacker-defender interactions. The autoencoder is useful in extracting behavior features of high dimensions meaningfully, whereas the Bi-LSTM determines the sequence dependencies and evolving patterns of attacks. The game-theoretic part of it represents dynamic competition of resources and uncertainty, based on tools such as Nash Equilibrium to compute the best defense strategy approximations. The legacy models are different in the sense that such a hybrid structure takes into consideration the contextual vulnerability, asymmetry of nodes at the node level, and opponent dynamic strategy in a cloud-aware system.

The addition is the incorporation of the neural sequence learning and strategic reasoning in the representation of the familiar and unfamiliar attack vectors across cloud infrastructure. Major contributions include: an interdisciplinary, scalable, zero-day attack path prediction structure, a new sequence and game-theoretic combination of sequence and game-theoretic learning on compressed features and experimental findings on the higher accuracy of predictions and lower false positives. The chapter provides a foundation to a predictive system that goes beyond the initial identification of zero-day attacks but can come up with informed and real-time decisions on resilience in cloud security.

In chapter 6, the author aims to discuss the topic of zero-day attack prediction, which starts with the introduction to frame the very importance of the concept in the context of cybersecurity. In Section 6.2, the proposed methodology is then described. Section 6.3 talks about the experimental set up and findings which include a simulated experiment, performance measures, result analysis and discussions, comparison with existing literature, and performance advances. Lastly, the chapter is concluded with a summary in Section 6.4, summing up the most important lessons and contributions of the zero-day attack prediction methodology to the process of improving cybersecurity defenses against new threats.

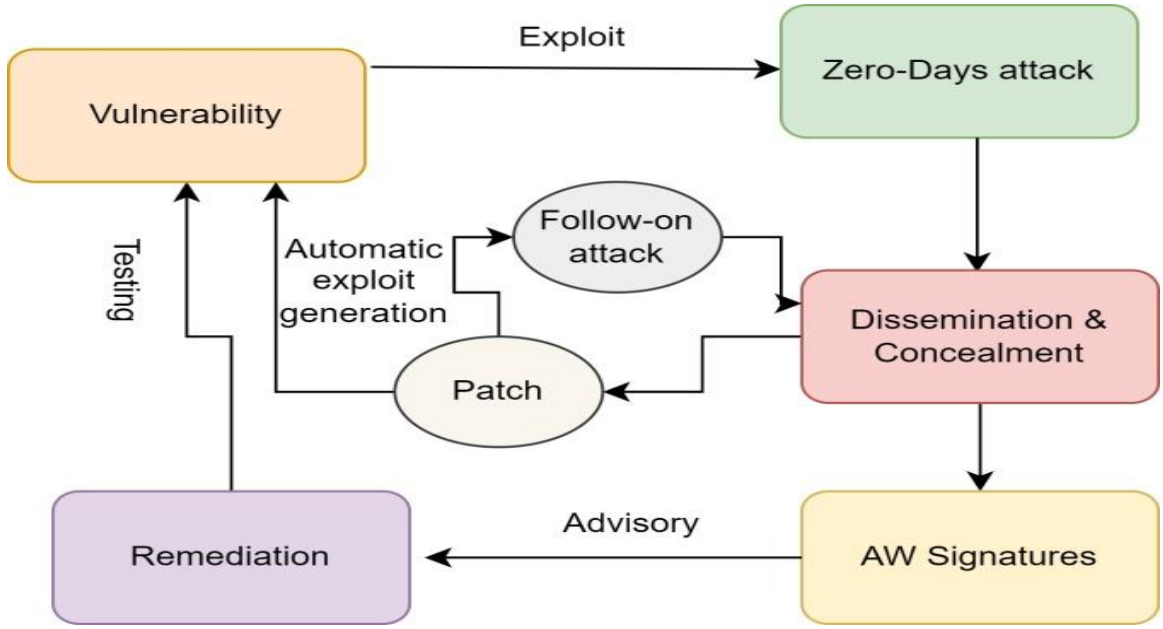

Figure 6.1 Zero-Day Vulnerability Lifecycle

The Figure 5.1 illustrates the lifecycle of zero-day exploitation on to a zero-day attack. The attack spreads and is obfused, and this leads to generation of AW (anti-malware) signature and remediation advisories. During this process there may be patches that are made and attackers may be having continuous following-on attacks or may use automatic exploit generation and the attack-defend cycle can continue until the remediation is most powerful and complete, leaving the attack.

6.2 Proposed Methodology

Zero-day adversarial samples are generated with the help of conventional test data. The classifier does not recognize them, and it represent a more severe network threat, as per

a survey of the most commonly used approach, the literature lacks analytical studies on zero-day adversarial instances on the attack and defense techniques based on trials in a variety of settings. The research will be used to implement the game theory to practical hostile situations with a focus on attack and defense strategies based on the Modified Bi-LSTM and Hybrid Game Theory using ANN Auto Encoder. In order to do this, experiments with gaming theory and adaptive gaming model are applied. Nash equilibrium method is used and the normal defense mechanism is an adversarial training method. It explores the winning percentages of zero-day adversarial scenarios, mean distortions, and original sample recognition on a number of adaptive game models in diverse settings. The research indicates that real-time adjustments of the target model of adaptive game models leave them much more resistant to adversarial samples and more likely to be attacked by all nodes as a security measure. The learning process of this type of method is used to replicate multi-attacker information sharing.

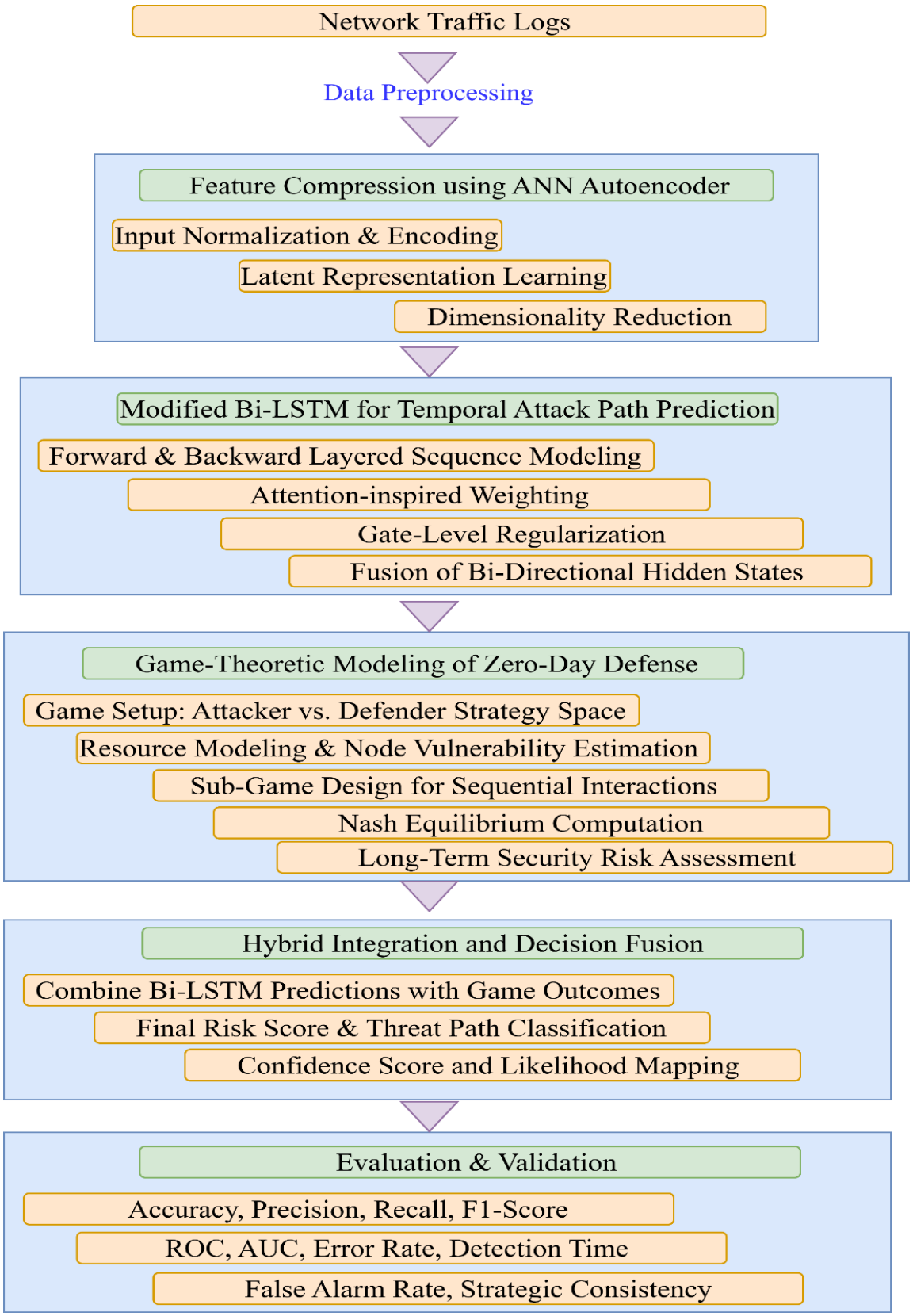

Figure 6.2 Framework for Zero-day Attack Prediction in Phase 2

The hybrid model of zero-day attacks attack prediction as illustrated in figure 6.2 is a combination of three significant modules. An ANN Autoencoder initially learns the raw network traffic and system properties to dimensionality reduction, noise elimination and the production of latent features to model the concept of time, which is evident in reconstruction losses. These truncated sequences are then run through a modified Bi-LSTM that is trained forward and backward on the temporal dependencies of the patterns of the attacks, and is regularized at the gate level with an emphasis placed on highlighting significant time steps; loss and accuracy on training/validation are used as measures. And finally, game-theoretic module uses Bi-LSTM outputs and implements the Nash Equilibrium strategies, schemes to calculate the adaptive defense allocation, and uses the node-level threat probabilities and Security Risk Index as performance indicators. The workflow that ensues and is named preprocessing Autoencoder Bi-LSTM predicted attack paths game-theoretic evaluation produces probabilities and confidence scores of zero-day attacks and strategic defense proposals compared to baseline ML models (DT, SVM, GNB, LR) as presented in Tables 6.2-6.4 and Figures 6.11-6.15.

6.2.1 Zero-Day Attack Prediction

The proposed research deals with the most serious problem of zero-day attacks, combining the game theory with the field of machine learning, with the attack and defense strategies based on Modified Bi-LSTM and Game theory and ANN Auto Encoder. Zero-day adversarial samples which are generated using traditional test data are highly dangerous because this is not known by the classifier and contribute to the vulnerability of the network. As opposed to the existing literature with no analytical studies on zero-day adversarial cases and attack and countermeasures, this research will fill the gap by using game theory in real-life adversarial situations. The research utilizes the experiment that uses the gaming theory and an adaptive gaming model, with the Nash equilibrium method and adversarial training method, being the default defense strategy. The experiment assesses the rate of success in zero-day adversarial condition, mean distortions and original sample recognition in different adaptive models of games in different environments. It is interesting to note that the research indicates the usefulness of real-time adjustments of the target model parameters, which improves the resistance of the adaptive game models to adversarial samples and minimizes the probability of successful attacks by malicious nodes.

The suggested approach builds a learning mechanism to replicate the multi-attacker information sharing, which will be added to a more robust approach to the prediction and defense of the zero-day attacks in the real world.

Pseudocode for Proposed Zero-Day Attack Prediction
Initialization Initialize Modified Bi-LSTM model Initialize Game Theory components Initialize ANN Auto Encoder model # Data Preparation Load historical data for training Preprocess data for model input # Training Phase Train Modified Bi-LSTM model on pre-processed data Train ANN Auto Encoder model for anomaly detection # Game Theory Setup Define game parameters and strategies Initialize Nash equilibrium technique # Adaptive Game Model Implement an adaptive game model combining Modified Bi-LSTM and Game Theory Adjust model parameters in real-time based on Nash equilibrium # Adversarial Training Implement adversarial training using ANN Auto Encoder for defense mechanism # Testing Phase Generate zero-day adversarial samples Evaluate success rates, average distortions, and original sample recognition # Real-Time Adjustment Continuously adjust target model parameters based on the adaptive game model Monitor and update Nash equilibrium strategy # Security Metrics Calculate security metrics such as resistance to adversarial samples and likelihood of successful attacks # Conclusion End

6.2.2 Dataset Description

In this research, two datasets were utilized to train and test the proposed zero-day attack prediction model, namely Dataset D1 -PATH Dataset and Dataset D2 -Celosia Zero-Day Attack Dataset. Preprocessing of the two datasets is done to fit the feature extraction and modeling pipeline.

Dataset 1: This data, generated by CloudSim, is a simulation of virtual cloud network systems and the logs on the attack paths, such as multi stage exploit sequences and cloudlets and virtual machines interaction and is specific to zero-day vulnerabilities. The preprocessing is defined as the normalization of features, measurement of graph-based properties of attacks and arranging the data to be presented to the EBPNN such that the model will be developed to memorize the realistic patterns of attack propagation. This knowledge is largely used in chapter 5 to calculate the attack path and in chapter 6 to strategically predict on the basis of a game-theoretic computation.

Dataset 2: The dataset that was downloaded in Kaggle (Celosia Zero-Day Attack Detection Demo) contains 8.14 GB of actual traffic and system behavior logs of network traffic. This data was then made ready to feed deep learning models with the selection of the CSV files that contained the features of the flow, system activity and the presence of anomalies (indicated) by hand through curation and pre-processing that included feature cleaning and feature normalization and filtering of features using Boruta and Chi-square tests. Phase 2 and Phase 3 make use of this data in the zero-day attack forecasting and detection assessment.

These datasets are further undergone uniform preprocessing to make them consistent between stages, including terms normalization and encoding, selection of discriminating attributes, and train-test split (70:30) with SMOTE to address minority zero-day samples. The information is then ready to be inputted into EBPNN, Bi-LSTM and ResNet50. The proposed framework will consist of a methodology that will enable the proposed framework to be robust, generalizable as well as capable of analyzing the behavior of zero-day attacks efficiently based on simulated and real world data.

6.2.3 Autoencoder for Feature Compression

The Autoencoder-based Artificial Neural Network (ANN) is used to reduce the dimensionality and compress the features in the model presented. Supervised neuronal networks are autoencoders, which were trained to produce input. It can be said to be comprised of two parts, an encoder that stenches the input to a latent space in low dimension and a decoder that attempts to re-create the original input in this low-dimensional representation. The high-dimensional system log or network traffic data is used to train the autoencoder, which is again trained on an encoding of low-dimensional features which the model encompasses the majority of the useful information and removes the noises and redundancy. Two purposes that are achieved by the compressed feature set are to minimize the computational complexity and provide a pointer to the underlying patterns in the data that can be utilized to detect anomalies including a zero-day attack. The Modified Bi-LSTM further advances the result of the encoder to produce the knowledge of the temporal patterns and generate attack paths. This way only the most informative features will be held and as a matter of fact the hybrid system can be left with efficient and effective downstream analysis.

Algorithm 6.2 Auto encoder Training
Input: Benign_data, ANN_architecture, regularisation_value, num_epochs
Output: Trained Auto encoder
1: training = 75% i∈ benign_data
2: testing = benign_data ∩ training
3:autoencoder←build_autoencoder (ANN_architecture, regularisation_value)
4: batch_size ← 1024
5: autoencoder. Train(batch_size, num_epochs, training, testing)
6: return auto encoder

Algorithm 6.3 Autoencoder Detection Accuracy Evaluation
Evaluation Input: Trained Autoencoder, attack, thresholds
Output: Detection accuracies
1: detection_accuracies ← {}
2: predictions ← model. Predict(attack)
3: for th∈ thresholds do
4:accuracy←(mse(predictions, attack)>th)/len(attack)
5:detection_accuracies.add(threshold, accuracy)
6: end for
7: return detection_accuracies

6.2.4 Modified Bi-LSTM for Attack Path Prediction

In the current research, the Bidirectional Long Short-Term Memory (Bi-LSTM) network will be used to predict potential zero-day attack chain with training on the auto compressions features produced by the autoencoder. Bi-LSTM is an extension of the original Long Short-Term Memory (LSTM) network which is designed to recognize long-term patterns of sequential data. Unlike LSTM which processes input sequence in one (forward) direction, the Bi-LSTM processes input sequence on both forward and backward directions hence taking into account previous and future states.

In Bi-LSTM two parallel layers of LSTM are used:

- The forward LSTM was used to read a sequence in a single direction.
- Mechanism Backward LSTM is backward.

The Memory unit of an LSTM, like that of an RNN, receives data at some point in time, denoted by Xt, and produces some value, ht, at some later point in time, also denoted by t. The convoluted unit compares the result of the previous cell state, Ct, with the present cell state, Ct, and the prior unit cell state, Ct-l, to update the output of the preceding cell during training and during switching characters.

Similar to an RNN, the Memory unit of an LSTM also takes data at a particular time denoted by Xt and generates some value denoted as ht at a later time also denoted by t. The convoluted unit compares the output of the previous cell state, Ct, with the current cell state, Ct, and the previous unit cell state Ct-l in order to update the output of the previous cell in training and also in switching characters.

Forget gate: $f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$ ------ (6.1)

Input gate: $i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$ ------- (6.2)

Output gate: $o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$ ------ (6.3)

Candidate cell gate: $c_t = \tanh(w_c x_t + U_c h_{t-1} + b_c)$ ------- (6.4)

The value indices U_f, U_i, U_o,, and uc are used to relate the last cell activation state to three inputs and the hidden state. Weighted vectors W_f, W_i, W_o, and Wc on the other hand, pass the nonlinear activation information to the entry and input data channel, and the four sequences b_f, b_i, b_o, and b_c are the possible sequences of education. The major variable was

the nonlinear activity value, tanh, and the scale parameter commonly was considered to be the gate perceptron, denoted by g.

The little representations that are taught by the autoencoder are normally high-level abstractions. The low level information of all time steps however, is not always as valuable as predicting the courses of attack. One possible solution to this is that attention-driven mechanism is included in the Bi-LSTM layers. The mechanism uses dynamic weights on importance of each timestep in the sequence based on its value to the task.

Autoencer latent representation gives compressed features that are used as a latent representation in which redundant or irrelevant information is removed. Features may, however, retain temporal correlation, more in succession series of network activity or attack activity. Bi-LSTM automatically discovers temporal correlations as it can capture: What has occurred before and after an occurrence, as well as how both are temporal to create an attack pattern.

Let $x = [x_1, x_2, \ldots . x_T]$ be the compressed input feature sequence from the autoencoder, where □ is the sequence length. The Bi-LSTM generates a sequence of hidden states:

$x = [x_1, x_2, \ldots . x_T]$ is the sequence. x_T pressed input characteristic sequence of the autoencoder. The Bi LSTM gives the result of a series of hidden states:

$$\vec{h}_t = LSTM_{fwd}(x_t) \text{------- (6.5)}$$

$$\overleftarrow{h}_t = LSTM_{bwd}(x_t) \text{------- (6.6)}$$

$$h_t = Concat(\vec{h}_t, \overleftarrow{h}_t) \text{ ------- (6.7)}$$

The last representation h_t represents bidirectional context. It can optionally perform a soft attention mechanism α t on each of the timesteps:

$$\alpha_t = \frac{Exp(Wh_t)}{\sum_{t=1}^{T} Exp(Wh_t)} \quad \text{Final Output} = \sum_{t=1}^{T} \alpha_t h_t \text{------- (6.8)}$$

This allows the model to focus on the most important time steps (such as malicious behavior or resource misuse).Mathematically, given the production of a hidden state □□ by Bi-LSTM, a score is calculated as:

$e_t = \tanh(W_a h_t + b_a)$------- (6.9)

$\alpha_t = \frac{\exp(e_t)}{\sum_{t=1}^{T} Exp(e_t)}$------- (6.10)

W_aAnd b_a are learnable parameters, 2 - tis the attention weight at timestep t,$h_t$is the concatenated hidden state $[\vec{h}_t, \overleftarrow{h}_t]$.

The final representation is then obtained as context vector:

$c = \sum_{t=1}^{T} \alpha_t \, h_t$ ------- (6.11)

This is done by ensuring that the model pays more attention to important compressed structures such as patterns that are suggestive of abnormal activity or sideways movement in the network.

Standard Bi-LSTM is prone to overfit especially when the trained on sparsely or imbalanced attack data. To this end, we introduce the concept of regularizing at the gate-level, that is, dropout is not only put between layers but rather between the LSTM gates (input, forget, and output).

And this is achieved through the modification of the LSTM equations to:

$\mathrm{f_t} = \sigma_g(\mathrm{W_f x_t} + \mathrm{U_f h_{t-1}} + \mathrm{b_f}) \cdot Dropout(r_f)$ ------ (6.12)

$\mathrm{i_t} = \sigma_g(\mathrm{W_i x_t} + \mathrm{U_i h_{t-1}} + \mathrm{b_i}) \cdot Dropout(r_i)$ ------- (6.13)

$\mathrm{o_t} = \sigma_g(\mathrm{W_o x_t} + \mathrm{U_o h_{t-1}} + \mathrm{b_o}) \cdot Dropout(r_o)$------ (6.14)

$r_f, r_i, r_o$are dropout masks on both gates. The following solution comes in handy: Co-adaptation of gate activations is avoided, Robustness in adversarial examples can be learned at a lower overfitting rate, and Long-term dependencies are learned at a lower overfitting rate. It is useful particularly when the condensed features are on high dimensionality which some of the pattern patterns in the gate can take over the learning process unless suppressed.

B-LSTM generates two hidden states at every timestep, $\vec{h}_t$and$\overleftarrow{h}_t$. after a second-order processing of the input sequence. A normal Bi-LSTM directly adds them but in our fusion layer we have a context-weighting module that decides selectively on whether to place more weight on the more informative direction at each timestep.

Instead of naive concatenation, we compute:

$\beta_t = \sigma(W_f) \cdot [\vec{h}_t, \overleftarrow{h}_t] \cdot b_f$ ------ (6.15)

$h'_t = \beta_t \cdot \vec{h}_t + (1 - \beta_t) \cdot \overleftarrow{h}_t$ ------ (6.16)

Where learned gating value, β t, is used to determine the contribution of each directional state, $\overleftarrow{\square}_{\square}$ new is the fused hidden state at temporal point,t. This allows the model to learn dynamically what context is dominant (past or future) depending on the kind of data, i.e., a few attacks, the past activity would be more predictive, some others, the pattern changes in the future would be determinative.

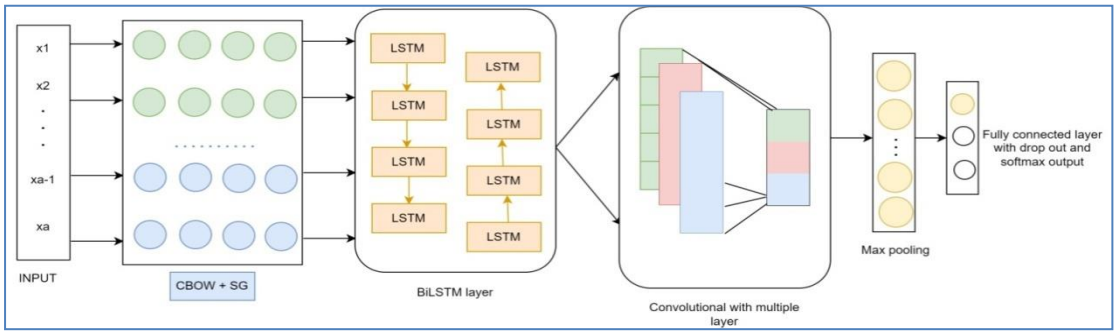

Figure 6.3 Modified Bi-LSTM Architecture

Figure 6.3 is an example of a hybrid deep learning model that possesses both semantic and sequence features of classification. The CBOW + Skip-Gram (SG) will be applied to retain the word semantics, and then a BiLSTM layer will be applied to retain context and time connections. It proceeds to run through a chain of convolutional layers having hierarchical feature computing, max-pooling and a fully connected layer including drop out and softmax in making a prediction. The model is the combination of sequential and spatial representations to attain a good performance.

Algorithm 6.2 Modified Bi-LSTM for Zero-Day Attack Detection
Input: Sequence of feature vectors extracted from input data
Output: Predicted class probabilities (e.g., normal or zero-day attack)

Step 1: **Initialize Parameters**
Set model hyperparameters (hidden size, learning rate, batch size, number of layers, dropout rate, etc.).

Step 2: **Embedding or Feature Representation**
Convert each input instance into a feature vector or embedding (e.g., word vector, traffic pattern).

Step 3: **Forward LSTM Pass**
Process the input sequence x1,x2,...,xT in the forward direction using LSTM cells:

- For each time step ttt, compute hidden state $\vec{h}_t$based on input xtand previous hidden state $\vec{h}_{t-1}$.

Step 4: **Backward LSTM Pass**
Process the sequence in reverse: xT,xT−1,...,x1:

- For each time step ttt, compute hidden state $\vec{h}_t$ based on xtx_txt and$\vec{h}_{t+1}$.

Step 5: **Concatenate Hidden States**
At each time step, concatenate forward and backward hidden states:

- ht=[$\vec{h}_t, \overleftarrow{h}_t$]

Step 6: **Apply Non-linear Activation and Pooling**
Pass the concatenated representation through a non-linear function (e.g., tanh or ReLU), followed by max or average pooling to obtain a fixed-size vector.

Step 7: **Classification Layer**
Feed the pooled output into a fully connected softmax layer to compute class probabilities:

- p=Softmax(Wh+b)

Step 8: **Loss Calculation**
Compute the loss using a suitable function (e.g., cross-entropy for classification tasks).

Step 9: **Backpropagation and Optimization**
Update model parameters using backpropagation through time (BPTT) and an optimizer (e.g., Adam or SGD).

Step 10: **Prediction**
Use the trained model to predict the probability of zero-day attack for unseen input sequences.

6.2.5 Hybrid Game-Theoretic Modeling for Zero-Day Defense

The game theory enables us to develop a formal model of attacker (e.g., zero-day exploiters) and defender (e.g., security software or administrators) behavior as rational decision makers. To become as close to the real cyberattacks and defense processes as possible, we incorporate a hybrid dynamic game model with resource allocation, sub-game decomposition, and long-term reward maximization in our proposed model.

6.2.5.1 Game Setup

Such an arrangement is the representation of a two player sequential non-cooperative game between a resource-constrained Attacker (A) and Defender (D). Some options open to the attacker include exploiting, lateral movement, or delaying whereas those open to the defender include patching, isolating nodes or tracking the activity. The players utilize the strategies π A and π d which are functions of the current state of the system s t to the best actions which maximize the respective rewards of the players. The game evolves under a transition function which is probabilistic.

$s_{t+1} = T(s_t, d_t, a_t)$ ------- (6.17)

Equation (5.17) which reflects the effect of actions by each side on the subsequent state s_{t+1}. The payoff function for the attacker $U_A(a, d)$is the effect of the attack decreased by the effort cost, whereas the defender's payoff $U_D(a, d)$is the damage prevented decreased by the cost of the resources, and it reflects the strategic trade-offs of active cyber defense.

Equation (5.17) that indicate the impact of every party and the consequential statess_{t+1}. $U_A(a, d)$is the cost of the attack to the attacker, and $U_D(a, d)$is the damage avoided by the defender less the cost of the resources, and the strategic trade-offs of active cyber defense.

Where T is a probabilistic transition function that depends on both players' actions.

Attacker $U_A(a, d) = Impact_{attack} - Cost_{effort}$ ------- (6.18)

Utility obtained by the attacker to launch an attack is Equation (5.18). It is expressed as the attack impact (e.g. system compromise, information stealing) and subtracted by the effort or cost of its execution (e.g. utilization of resources, chance of being caught). This value would be maximized by the attacker and it would be a trade-off between high impact action and low cost or risk.

Defender $U_D(a,d) = Damage\ prevented - Resource\ Cost$ ------- (6.19)

Equation (5.19) is an indicator of the reward of the action of the defender. It is harm evaded (e.g. loss of data or service evaded) lower than the price of assets (e.g. cognizance utilization, workforce, or financial investment) in establishing the defenses. The defender will strive to optimize this incentive by blocking threats in the most efficient manner that will not cause excessive expense against unnecessary defense.

Combined, these functions define the strategy trade-off between offensive and defensive choices in the zero-day attack-defense situation.

6.2.5.2 Resource Modeling

The attacker as well as the defender in this game-theoretic problem is both constrained by limitations of limited resources, and these factors are the major determinants in making long-run strategic decisions. At any particular time t, the available resources that the attacker has $R_A(t)$are the scarce energy, stealth, or calculating ability required to launch and sustain attacks. In line with this, resource budget$R_D(t)$, shows the resources (sparse security resources such as surveillance agents, patching capability, or personnel) available to detection and response by the defender. The amounts of these resources are dynamically changing and directly influence the choice of action and its possibility and effectiveness of every participant, which provides a realistic limit to decision-making.

$\sum_{i=1}^{n} r_{A_i}(t) \le R_A(t),\ \sum_{j=1}^{m} r_{D_j}(t) \le R_D(t)$------- (6.20)

Where r_{A_i}, r_{D_j} are resources to be utilized in each action.

$R_A(t+1) = R_A(t)\text{-}r_{A_i}(t) + \delta_A, R_D(t+1) = R_D(t)\text{-}r_{D_j}(t) + \delta_D$------- (6.21)

Where with δ is periodic regeneration of resources. This model reflects real capacities of recurring adventures or deployment of defense.

6.2.5.3 Attack-Defense Dynamics

The Attack-Defense Dynamics module represents the evolving states of the interactions between a defender and an attacker in a zero-day threat scenario through simulation of the effect that the actions of each have on the system state at various times. The system changes the state s, t of a system to a new state s_{t+1} with the defender d_t and the attacker at changing the state.

$P(s_{t+1}|s_t, a_t, d_t) = f(s_t, a_t, d_t)$------- (6.22)

The acquisition of this attack-defense activity f may be gained by previous experience in attack-defense or simulated experience in the attack, like the real-world constraints such as unknown outcomes and incomplete information.

$$R_A(t) = \sum_k I_{success}(a_k, d_k) \cdot \gamma^t \cdot V_{damage}$$------- (6.23)

Here, the model of the attacker rewards is $R_A(t)$ (all successful attacks)$I_{success}(a_k, d_k)$, damage, which is the measured damage value. This motivates the attacker to trade between the short term effects and long term sustainability.

$$R_D(t) = \sum_k I_{Detect}(d_k, a_k) \cdot \gamma^t \cdot V_{saved} - \lambda \cdot Cost_{defence}$$------- (6.24)

With γ representing the discount factor and λ representing the penalty coefficient. On the other hand, the reward of the defender $R_D(t)$is given as the sum of the detected attacks$I_{Detect}(d_k, a_k)$which is discounted in a similar fashion by gamma and it is multiplied byV_{saved}, which is a measure of the loss which is averted. Based on this, the defense expense, scaled by a penalty coefficient λ, is subtracted to indicate the amount of resources consumed or the cost of operations of defense measures. The model can consider the trade-offs over time (by means of γ), uncertainty over the consequences (by means of f), and the cost-benefit analysis of defensive resource allocation (by means of λ), and thus is a powerful framework of simulating real-time strategic behavior in cybersecurity.

Algorithm 6.5 Computation of Attack Defense Games

Input:

- Initial number of resources: n(0)
- Time duration: T
- List of newly added resources at each time step: $n_{new}(t)$for t = 1 to T
- List of expired resources at each time step: $n_{\exp(t)}$for t = 1 to T
- List of randomly eliminated resources at each time step: $n_{dis}(t)$for t = 1 to T

Algorithm:

1. Initialize an empty list for attack defense games: n = []
2. Set the initial number of attack defense games as the initial number of resources: n[0] = n(0)
3. For each time step t from 1 to T, do: 3.1. Calculate the total number of attack defense games at time step t using Equation (1): $n(t+1) = n(t) + n_{new}(t) - n_{dis}(t) - n_{\exp(t)}$
   3.2. Add n(t) to the list of attack defense games: n.append(n(t))
4. Return the list of total attack defense games: n

Output:

List of total attack defense games at each time step: n(t) for t = 1 to T

6.2.5.4 Sub-Game Design

The complexity present in the models of zero-day attack-defense games is dealt with by sub-game design methodology. This is done in place of treating the entire series of attacks as a monster game but as sub-games with each sub-game relating to one of the steps in attack life cycle. The module-based decomposition is handy in facilitating an easy analysis and qualitative strategic reaction. The confrontation in Sub-game 1 belongs to the first phase, i.e., the attacker would determine the way he would attack, and the defender would determine where he would plug first. Sub-game 2 is similar to middle-level behavior, in which the attacker is trying to move horizontally through the network, and the defender sets internal detection to monitor the horizontal movement. The last sub-game is 3: the attacker is attempting to steal the valuable information, the defender makes the containment actions to avoid the loss of data.

The sub-games $G_i = (S_i, A_i, D_i, U_{A_i}, U_{D_i})$are defined in terms of the state spaceS_i, the attacker action sets A_i and the defender action setsD_i and the utility functions U_{A_i} as well asU_{D_i}. These sub-games may be independently (i.e. parallel) or hierarchically solved, in the latter case, the result or strategy of a particular sub-game G_i defines the configuration of the next G_{i+1}. The recursive form **structure**$G = \sum_{i=1}^{k} G_i$can be used to implement an active and dynamic defense mechanism and lets the system predict the future attacks, based on previous behavior and context-dependent information.

6.2.5.5 Nash Equilibrium

In a game, Nash Equilibrium (NE) is the strategic equilibrium point, where it is neither possible to obtain anything or achieve through unilateral strategy change on the part of the attacker or the defender, should the strategy of the other party remain unchanged.

A Nash Equilibrium (NE) is a combination of strategies (π_A^*, π_D^*) in which neither can a player achieve a higher utility by acting contrary to the strategies chosen by the other player:

$U_A(\pi_A^*, \pi_D^*) \geq U_A(\pi_A, \pi_D^*) \quad \forall \pi_A$------- (6.25)

$U_D(\pi_D^*, \pi_A^*) \geq U_D(\pi_A^*, \pi_D) \quad \forall \pi_D$------- (6.26)

The strategy pair (π_A^*, π_D^*) is a NE formally in the sense that the utility of the attacker U_A is maximized at π_A^*given π_D^*, and the utility of the defender $U_D$is maximized at π_A^*given

π_D^*,. This is to guarantee the optimality of both sides no one-way deviation gives a better result.

When the play is a zero-sum game, that is, winning by one side causes the other to lose, NE can be determined by maximizing the expected utility of the defender, in this case the defender can play the best game possible to take the least bad guaranteed utility. Mathematically this can be expressed as:

$$\max_{\pi_D} \min_{\pi_A} U_D(a, d) \text{------- (6.27)}$$

This is the case in general-sum games where the payoffs are not inherently antithetical (i.e. both lose or both win otherwise), and best-response or algorithms based on Q-learning are repeatedly optimized to obtain an approximation of NE. The role of NE in computer security game modeling is extremely significant because it attempts to model rational and stable strategies in uncertain games whereby either the attacker or the defender can predict the other without having full information regarding the other party action which is usually the case in the scenario of a zero day attack.

Algorithm 6.6 Nash Computation

Inputs:

T: Time step
Q: Number of attack nodes
K_j: Number of resources used by attacker node j
P_(j,n)(t): Probability of the n-th resource used by attacker node j at time t, where n = 1,...,K_j
w_n(t): Weight associated with the n-th attack node at time t, where n = 1,...,Q

Based on these inputs, we can compute the following:

Algorithm:

1. Initialize S(t) = 0.
2. For each attack node n = 1 to Q, do steps 3-4.
3. Compute P_n(t) using Equation (9): P_n(t) = 1 - ∏(1 - P_(n,m)(t)), for m = 1 to K_n.
4. Compute S(t) using Equation (11): S(t) = S(t) + P_n(t) * w_n(t), for n = 1 to Q.
5. Output the value of S(t) as the outcome of the security measure.

Output:

S(t): Outcome of the security measure at time t, representing the threat level. A higher value of S(t) indicates a higher chance of threat.

6.2.5.6 Long-Term Security Assessment

In this section, a paradigm of the long-term evaluation is presented as a comparison and contrast of the efficiency of the defender and attacker strategies with the use of the long-term time scale. It seeks to determine the strength of a particular system regarding zero-day attacks through the calculation and comparison of expected cumulative rewards of both players in strategic equilibrium.

The value function $V_a(s)$approximates the cumulative expected reward of the attacker beginning at initial state $s_0 = s$ and over time T. It adds:Rewards at every time step $R_A(t)$,,a discount factor $\gamma^t \in (0,1)$ to discount future rewards in comparison to immediate ones (an urgency model or model of uncertainty).

$$V_A(s) = \mathrm{E}[\sum_{t=0}^{T} \gamma^t R_A(t) | s_0 = s] \text{------- (6.28)}$$

On the same note, the value of the defender $V_D(s)$is a computation of the expected benefits accrued by detection or mitigation actions:

$$V_D(s) = \mathrm{E}[\sum_{t=0}^{T} \gamma^t R_D(t) | s_0 = s] \text{------- (6.29)}$$

These functions take into account the performance and strategic sustainability of decisions of every player in the long run. A system is said to be resilient when cumulative reward of defender is more than the attacker:

$$V_D(s) > V_A(s) \text{ ------- (6.30)}$$

This disparity suggests that eventually, defensive mechanisms are useful in counteracting or exceeding the advantages of the attacker, resulting in ensuring reduced risk and stability of the system.

In order to measure the relative strength of the attacker, a Security Risk Index (SRI) is presented:

$$SRI = \frac{V_A(s)}{V_A(s)+V_D(s)} \text{ ------- (6.31)}$$

- In case, SRI=0, the defender wins and strong protection is being established.
- In case SRI=1, the attacker acquires a substantial benefit, the system is not safe.

This normalized score is capable of providing an effective measure of the exposure of a system and may result in adaptive policy decisions, such as increasing the strength of certain assets, or dynamically re-allocating resources to monitoring.

The evidence-based strategic approach to cybersecurity is available in long-term modeling setting that quantifies the risk in the system by value functions and the Security Risk Index (SRI). In the above framework, the SRI will be computed at Phase 2, the Game-Theoretic module will provide an approximation of how the attacker-defender interactions take place, and the approximations are based on predicted zero-day attack paths generated by Bi-LSTM. The SRI assigns a risk score of 0-1 to every node or system component where a high score implies that it is more vulnerable to an attack in the form of a zero-day attack. An example of this is that SRI values are placed on nodes with high probabilities of being in the path of any predicted attack or where there are high probabilities of anomaly which are critical areas that require mitigation.

This threat analysis can contribute to the aspect of preemptive defense in which the framework can be composed of many defensive mechanisms and perform a comparative analysis of the policies before its application. The system will be able to cope dynamically with the behavior of the attackers by introducing the adaptive learning, which adapts the system to take risk-related decisions, both at the architecture and operational levels. Thus, the SRI simplifies the allocation procedure, the priorities of the defense procedure, and mitigation measures to achieve maximum resilience of the system against a new and unexpected zero-day pattern of attacks.

6.2.6 Hybrid Integration Strategy

The proposed framework combines ANN-based Autoencoder, Modified Bi-LSTM and Hybrid Game-Theoretic Modeling into a highly integrated framework to generate accurate zero-day attack path prediction and adaptive defense strategy. All the components address one aspect compression feature of learning, time, and strategic decision-making and the other two assist each other to make them much more robust and interpretable.

6.2.6.1 Feature Compression using Autoencoder → Input to Bi-LSTM

The proposed structure uses ANN-based Autoencoder to perform unsupervised feature denoising and compression and feed the Bi-LSTM. The input features, including node interconnectivity, vulnerability scores, history of the past attacks, and traffic statistics

are usually of high dimensions and the output of certain redundant features. These traits are obtained based on Dataset D1 (PATH Dataset) and Dataset D2 (Celosia) and have been indicated in Tables 4.3 respectively.

- The in and out numbers of connections in a node are the values of node interconnectivity which are 1-25 in D1 and 0-20 in D2.
- Vulnerability scores Vulnerability scores are the scores on the normal scale (between 0 and 1) of the susceptibility based on the CVSS-like measurements.
- An archive of the old attacks is recorded as frequencies or presence variables of exploit patterns used in the past (range: 0-5 in D1; 0-3 in D2).
- Traffic statistics include the number of packets, flow period and mean packet size both sets of data on the normal scale [0,1].

Autoencoder is a model that learns a representation of latent features that are utilized to diminish the dimensionality and keep the discriminative information. The packed features are the most important characteristics of zero-day attack behavior, and thereby, the Bi-LSTM can effectively learn temporal dependencies, and not limited to irrelevant and noisy features. Empirically, the latent feature dimension will be set to 15 since the feature importance analysis and stability during the training were examined and shown to be the most appropriate in Chapters 6 and 7.

$z = f_e(X), \quad \hat{X} = f_{de}(X)$ ------- (6.32)

In this case, X denotes original input feature matrix, $Z \in \mathbb{R}^k$is compressed feature vector (k is much less thandim (X)), f_e and $f_{de}$are are encoder and decoder functions. Reconstruction loss $L_{rec} = ||X - \hat{X}||^2$ minimum information loss. Critical dynamics and attack-relevant features are represented by the compressed latent representationz, which is subsequently used as input to the Modified Bi-LSTM to model the occurrence of the attack over time and network layers sequentially.

6.2.6.2 Bi-LSTM for Sequential Prediction → Input to Game-Theoretic Logic

The compressed sequences $\{z_t\}$ are used in the Modified Bi-LSTM which learns the temporal relationships in the attack patterns across nodes or time slices:

Forward pass $\vec{h}_t = LSTM(z_t, \vec{h}_{t-1})$ ------- (6.33)

Backward pass $\overleftarrow{h}_t = LSTM(z_t, \overleftarrow{h}_{t-1})$------- (6.34)

Fusion $h_t = \varphi(\vec{h}_t, \overleftarrow{h}_t)$where φ includes context-aware attention weights.

The h_t output of the Bi-LSTM obtains deep contextual knowledge that is required to make predictions and counter zero-day attacks. It summarizes the likelihood of a node being in an attack chain when there is temporal dependence among nodes, pattern behaviour like long-term transitive influence of adjacent compromised nodes and latent attack patterns. The game-theoretic module is then used to exploit this attack context-aware model to provide predictions of real time node vulnerability scores in order to allocate strategic defense resources. Further, h t is also applicable to manage dynamic payoff matrix updating with incremental predicted levels of attack and the defender can therefore adaptively respond to the changing attack patterns and to enable proactive and adaptive defense control.

6.2.6.3 Mutual Reinforcement between ANN Autoencoder and Game Theory

The last phase of the hybrid integration is the self-reinforcement between ANN Autoencoder and Game-Theoretic Model which creates a closed-loop adaptive system.

Within the scope of this hybrid, the closed-loop cycle of Autoencoder, Bi-LSTM, and Game-Theoretic module can provide the opportunity to adjust dynamically to weaknesses and learn to play strategically through the provided learning. An example, the Vulnerability Estimation Loop, uses the hidden state of the Bi-LSTM to provide an estimate of expected payoffs with regard to both attacking and defending each node. These payoff values are then directly used as an extra feature channel into the Autoencoder as it is retrained to learn feature representations, which are responsive to high-risk or high-value decision situations. Next comes Strategy Feedback Loop which feeds back after the equilibrium defense strategies, e.g., isolating, tracking or deprioritizing particular nodes, into the encoder; which then learns patterns of attacker behavior and biases in strategic behavior. Besides that one, an Anomaly Loop identifies ambiguous nodes or time series that have conflicting payoff or cannot be classified by conflicting strategic indications. Such warning-labeled examples are subsequently used to selectively train the encoder and Bi-

LSTM to optimally respond to stealthy, rare or evolving zero-day attack vectors on the accuracy and timeliness of adversarial edge cases retraining.

6.3 Experimental Setup and Results

The following section describes the simulated experiments carried out in this phase.

6.3.1 Simulated Experiment

The expected method is calculated based on two aspects through numerical simulations. The first factor provides network analysis with threshold calculation and root mean squared error of zero-day attack and the second aspect is the rate of traffic, root mean squared error, and bandwidth calculation.

6.3.1.1 Analysis Algorithms

After the data has been preprocessed, this step involves desirable machine learning algorithms to run the data that has been prepared. The goal is to evaluate and compare the predictive capabilities of them as well as to identify the best features that will result in the correct prediction of the zero-day attacks.

1) Support Vector Machine- SVM is a supervised machine learning method. By far the most used classification method is svm. SVM constructs a band to show the distinction between two classes. It is able to produce a hyperplane or a collection of hyperplane in three dimensions. This hyperplane is very amenable to classification and regression. Svm classifications are made on the basis of the properties of the instances and even the classification of objects not covered by data can be done. In order to have separation, a hyperplane identifies the nearest training location to each of the classes.

Let D = {(xi,yi)}n i=1 be a classification dataset, with n points in a d-dimensional space. Additionally, let us assume that there are only two class labels, Yi ∈ {+1,−1}, denoting the positive and negative classes.

A hyperplane in d dimensions is given as the set of every one point x ∈ R_d that satisfy the equation h(x)= 0, where h(x) is the hyperplane function, defined as follows:

$h(x) = w^t x + b$ --------- (6.35)

Where w denotes a d-dimensional weight vector and b is a scalar value denoted by the term bias. This research has a hyperplane for points that lie on it.

$h(x) = w^t x + b = 0$ ---------- (6.36)

The hyperplane is thus defined as the set of all points such that□ □ □ + □ = *0*. To see the role played by b, presumptuous that w1 ≠ 0, and setting xi = 0 for all I> 1, this research can obtain the offset where the hyperplane intersects the first axis, as, by Eq. (6.36)

$w_1 x_1 = -b or x_1 = -b/w1$---------- (6.37)

Observe that the support vectors now comprise all points that are on the margin, which have zero slack (ξi = 0) and all points with positive slack (ξi> 0).

Algorithm 6.6 Support Vector Machine (SVM)

Input:

- Dataset D = {(xi, yi)} for i = 1 to n, where xi is a d-dimensional feature vector and yi is the corresponding class label (+1 or -1)
- Parameters: None

Procedure SVM():

1. Initialize the weight vector w and bias term b to zero or small random values.
2. Define the hyperplane function h(x) as h(x) = w^T x + b.
3. Define the margin as the distance between the hyperplane and the closest training instances from each class.
4. Perform optimization to find the optimal hyperplane that maximizes the margin.
   a. Construct the optimization problem as a convex quadratic programming task:
      - Minimize: 1/2 ||w||^2
      - Subject to: yi(w^T xi + b) ≥ 1 for all i = 1 to n

   b. Solve the optimization problem using methods like Sequential Minimal Optimization (SMO) or other efficient algorithms.
      - Update w and b to find the optimal values that satisfy the constraints.
5. Identify the support vectors, which are the data points that lie on or near the margin.
      - Support vectors have non-zero slack variables (ξi> 0) in the optimization problem.
6. Output the trained hyperplane parameters: weight vector w and bias term b.

Output:

Hyperplane parameters: weight vector w and bias term b

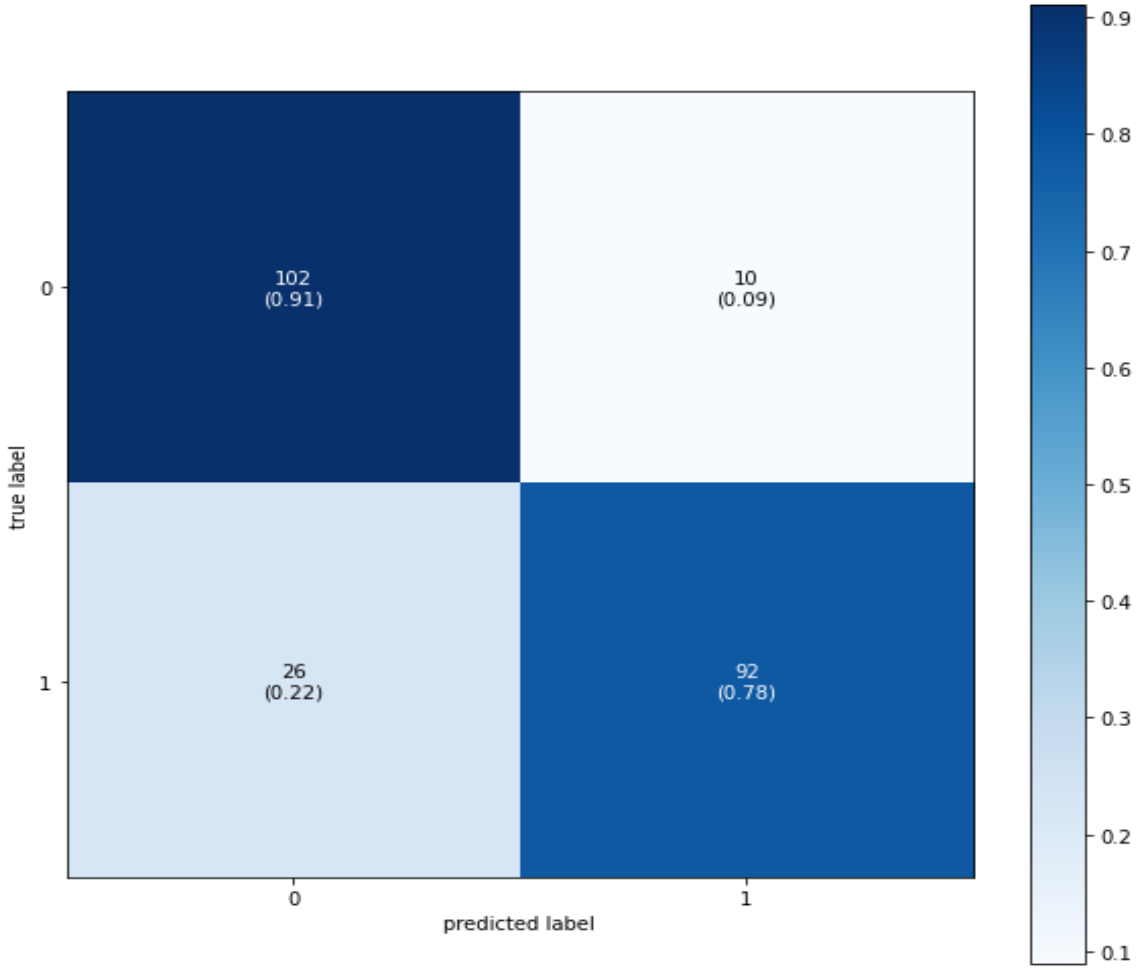

Figure 6.4 Confusion Matrix of SVM

2) Decision Tree

One technique of importance in categorization is a DT. It is a process of guided learning. A decision tree is used when a response variable is categorical. Decision tree model is a tree like structure where the process of categorizing input characteristics is illustrated. Input variables can be in the form of graphs, text and discrete or continuous data.

The Steps for Decision Tree Algorithm

- Step 1: The input features constitute a tree the nodes of which are defined.
- Step 2: Choose a feature to predict the output of the input feature of zero-day attack that has the largest information gain.
- Step 3: The amount of information that is maximized is computed at every node in the tree at each characteristic.

- Step 4: Repeat step 2 to generate a sub-tree of the current node that makes use of the characters not used by the last node.

This algorithm works based on the properties of a data of a class. It relies on Decision Tree Classification so as to construct a set of rule sets so as to provide an unconditional classification of differentiable data. This is the main strength of this Classification Algorithm it is able to recognize and see well defined characteristics of a data set.

The biggest problem with it is that it cannot determine related attributes values within the dataset. Some values of data could not be placed in any of the decision statements. This therefore means that such information may be overlooked in research. Hence, it is not suitable in complex data that combines two attributes in the same variable referred to as multi-variate dataset properties.

Rule-Based Decision Mechanism in Decision Tree

The formed results of a decision tree algorithm are based on If-Else rules:
IF(Variable(Index) >Threshold limit)
Accept the Positive Result and increment it
Else
Accept the negative result and increment it

The result of the Decision Tree algorithm is identified by a set of IF-ELSE decision rules, which are formed during the phase of tree building and various in relation to the feature thresholds which are learned with the assistance of the training data. According to these regulations, one can categorize sample of zero-day attacks.

The decision process of each node can be indicated in rule-based decision mechanism in decision tree table. A looping statement can deal with process repetition, especially with a statement in the research field as it may easily deal with multi-dimensional prediction characteristics.

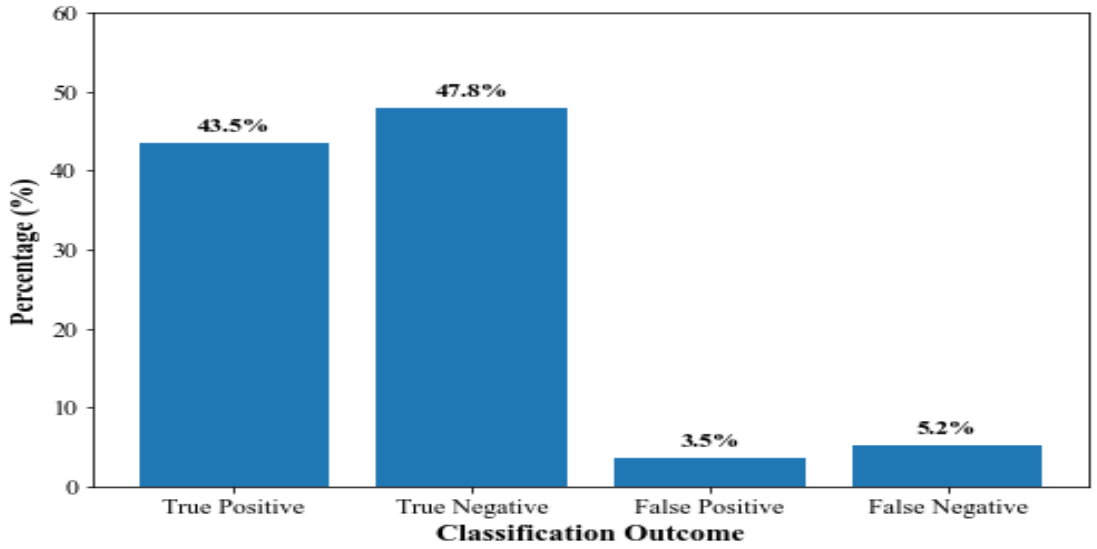

Figure 6.5 Classification of DT

3) G-Naive Bayes (NB) Classifier

An NB classifier is a basic probabilistic classifier, which is based on the assumptions of Bayes and independent Bayesian statistics. The presence or the lack of a particular quality of a class is, supposedly, independent of any other quality. This classifier can be learnt, in a supervised learning environment, in a short time depending on the structure of the probability model. The Bayes classifier is able to estimate the parameters of the classification variables which include the means and variances using fewer training data.

The LR is the recognition of relationships in terms of interrelated data values. The regression process involves the finding of overlapping data values in order to generate a new data set. One such method is the random forest tree that aims at determining the relationship between the characteristics of data and the outcomes. The method is a meta-estimator which can be extended to formulate trees of data without making any judgment concerning separable data.

$$p\left(\frac{c}{f1}\ldots fn\right) = \frac{p(c)p(f1\ldots fn/c)}{p(c)(f1\ldots fn)} \text{ ---------- (6.37)}$$

It can be simplified as

$$posterior = \frac{priorxlikelihood}{evidance} \text{ ---------- (6.38)}$$

The above statement is the Bayes theorem, and classification is performed using the preceding formula.

Algorithm 6.7 Naive Bayes Classifier

Input:

- Training dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)}, where xi is a feature vector and yi is the corresponding class label.
- Parameters: None

Procedure Naïve Bayes():

1. Calculate the prior probabilities for each class label:
   - Count the number of instances in each class label.
   - Calculate the probability of each class label: p(C).
2. For each feature fi in the feature set F: a. Calculate the likelihood probabilities for each class label:
   - Calculate the class-conditional probability: p(fi|C).
   - If the feature fi is continuous, assume a probability distribution (e.g., Gaussian) and estimate its parameters (mean and variance) using the instances in each class label.
   - If the feature fi is categorical, calculate the probability of each category value given each class label.
3. For a new instance x: a. Calculate the posterior probability for each class label:
   - Calculate the product of the prior probability and the likelihood probabilities: p(C|x) = p(C) * ∏ p(fi|C) for all fi in x.
4. Classify the instance x based on the highest posterior probability:
   - Assign the class label with the highest posterior probability as the predicted class label for x.

Output:

Trained Naive Bayes classifier model.

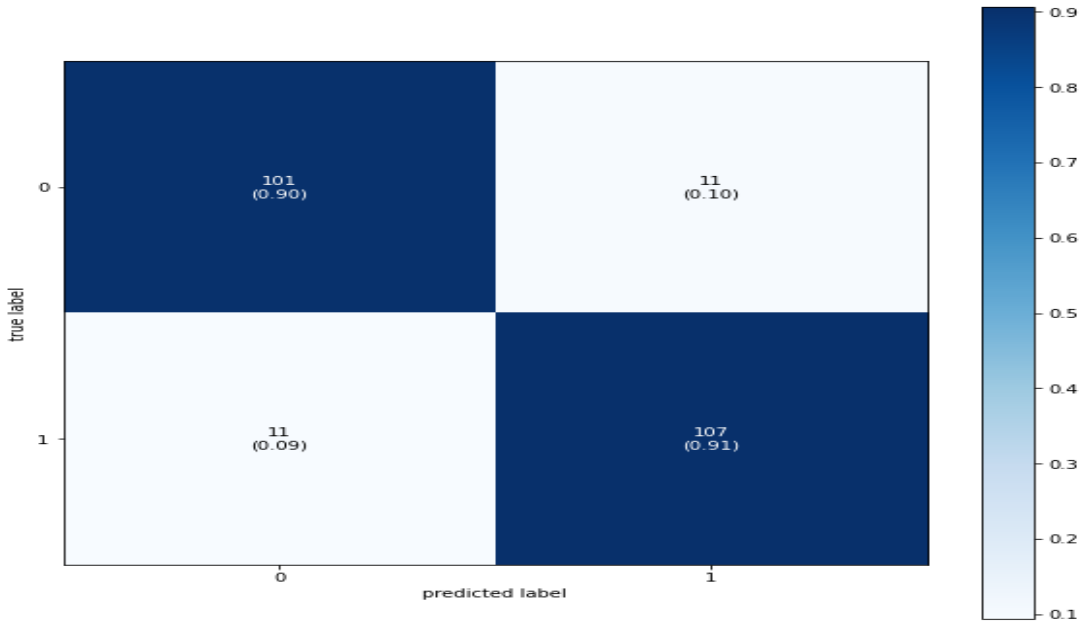

Figure 6.6 Confusion Matrix of G-Naive Bayes

4) Logistic Regression Algorithm

The LR is the recognition of relationships in terms of interrelated data values. The regression process involves the finding of overlapping data values in order to generate a new data set. One such method is the random forest tree that aims at determining the relationship between the characteristics of data and the outcomes. The method is a meta-estimator which can be extended to formulate trees of data without making any judgment concerning separable data.

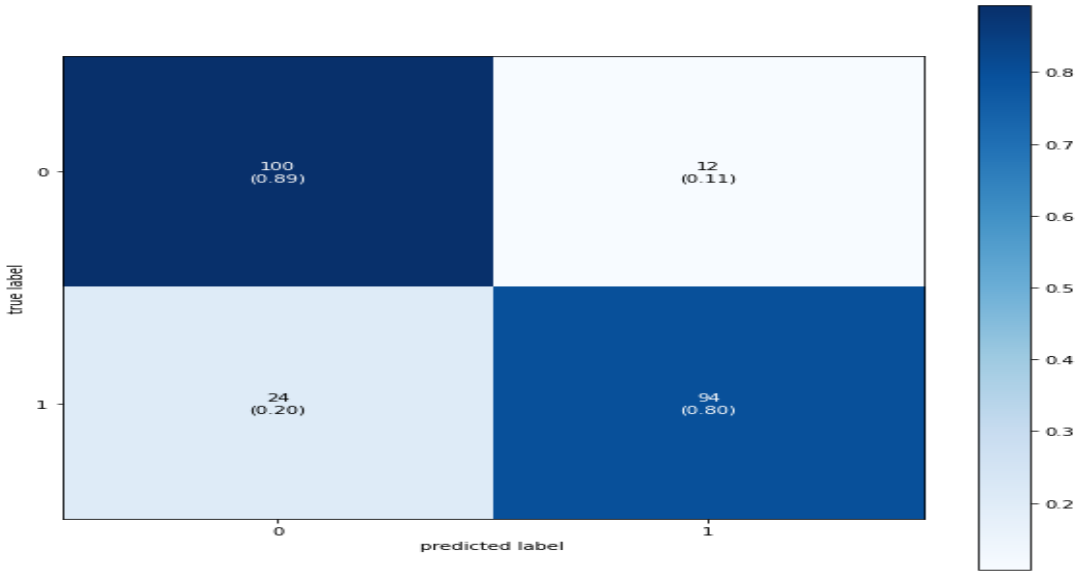

Figure 6.7 Confusion Matrix of LR

5) Latency Analysis in Zero-Day Attack Environment

The numerical analysis suggests that the expected model is able to handle a Zero-day attack environment with the lowest differences realized in offering alert messages latency. As it was evidenced by the acquired graph, the model that is expected to be used is capable of handling a network of zero-day attacks with effective variation between values as it is demonstrated in Table II. Based on these results, values are observed to analyze message latency of data sharing protocol with the following equation expressed below (6.40). In case of lesser message latency, the infected environs are not depicted by a large number of infected nodes. In cases where the network size is also large, message latency may also be increased. The general deviation of the current situation and the observed one is smaller; hence, the predicted model can be applied to sound communication when there is a Zero-day attack.

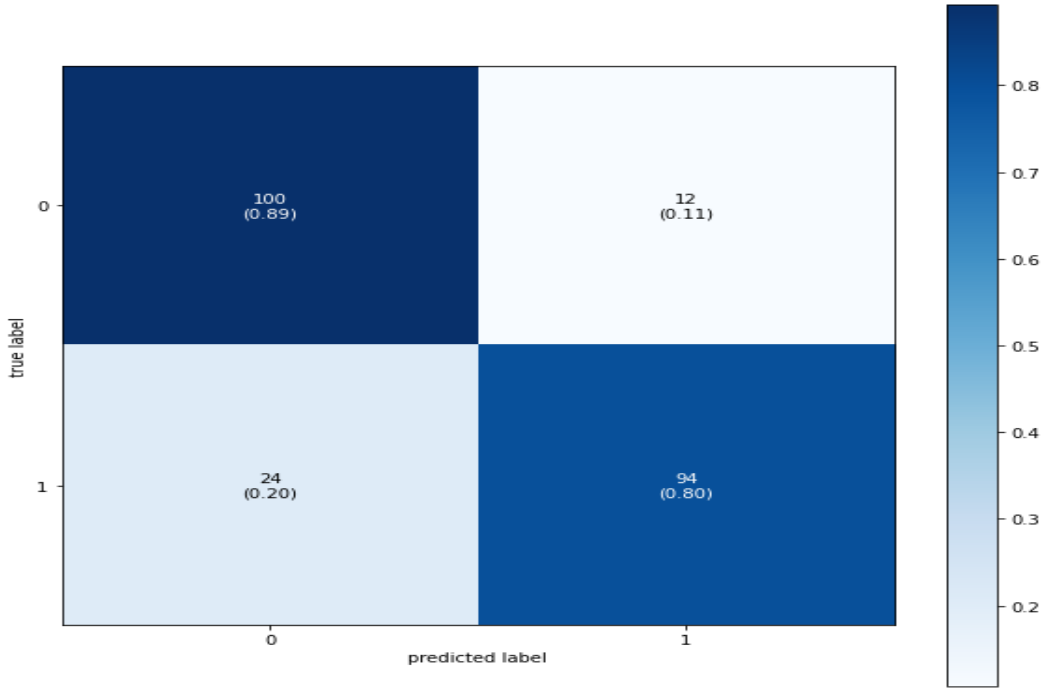

Figure 6.8 Confusion Matrix for Stacking Ensemble

Figure 5.8 indicates the TP, FP, TN, and FN values in the confusion matrix. The estimated classification of TP is 100, TN is 94, FP and FN are 12 and 24 respectively. At last the stacking ensemble classification has been integrated with DT, SVM, LR, GNB algorithms are embedded, and it give the best results.

6.3.1.2 Analysis Interpretation

By utilizing the CloudSim simulator, the stage will put to test by a numerical analysis the proposed mechanism of detecting and alerting the zero-day attack. The assumption is that a clustered network model with base station is taken into consideration and the network reliability is tested under various network sizes and conditions of infection. The nodes with a 0-day attack probability are isolated to reduce the spread of the node and reduce overhead. The value of 0.5 is used to distinguish between reliable and unreliable communication states whereby the value above the threshold value is a safe indicator of the network functioning.

Latency, traffic rate, error rate and overhead of communication are used to measure the impacts of the infected and uninfected nodes. The two communication protocols used are the protocol of alert messaging and data sharing protocol to test the message latency and

overhead in case of critical attack situation. The measures of reliability and cost demonstrate that the cost of network with a removal of infected nodes is low, the overhead is low and the reliability is better as compared to the cost of a network with infected nodes. Overall, this step will prove the effectiveness of the suggested model so that the stability of the communication could be guaranteed, and the load would be minimized in the conditions of the zero-day attacks. For certain message protocols, alerts are generated with Eq. (6.39):

$$L = 2T + 3T + 4T + 2T + T \text{------ (6.39)}$$

The final alert message produces latency after decision making is evaluated as in Eq. (5.40):

$$L = L_i + 4Q + Q + 3Q \text{ ---------- (6.40)}$$

Moreover, network with equal decision-making delay with Eq, (6.41):

$$L = L_i + 8Q \text{---------- (6.41)}$$

At present, in a linked state, communication among clustered nodes produces packet delay as in Eq. (5.42):

$$D(P,E) = \frac{P*E}{\beta} + D_{wired} \text{---------- (6.42)}$$

In wireless links, packet delay is provided as in Eq. (6.43):

$$D(P) = F + (x-1)t \text{ ---------- (6.43)}$$

Then, communication overhead for alerting messaging as in Eq. (6.44):

$$C_{overhead} = 2D(P) + 2D(P) + 3D(P,E) + D(P,E) + 4D(P,E) + L \text{ -------- (6.44)}$$

In the next stage, analysis is carried out in data sharing protocols, generating message latency is modeled as in Eq. (6.45):

$$L = 3T + 3T + 2T + 4T + 5T \text{-------- (6.45)}$$

Where final critical message production latency is provided as in Eq. (6.46):

$$L = L_i + 5Q \text{ ---------- (6.46)}$$

At last, the communication overhead for sharing data is computed as in Eq. (6.47):

$$C_{overhead} = 3D(P) + 5D(P) + 3D(P,E) + 2D(P,E) + 5D(P,E) + L \text{..------- (6.47)}$$

6.3.2 Comparison with Existing Methods for Zero-Day Attack Prediction

Traditional machine learning frameworks, such as Decision Trees (DT), Support Vector Machines (SVM), Gaussian Naive Bayes (GNB), and Logistic Regression (LR) were applied at the initial level in the traditional intrusion detection systems by Mishra et al. (2018) and Alazab et al. (2011). The strong points are that effectively work with known attacks in the training set and provide quick classification and easy interpretability. As an example, Decision Trees can simply classify common types of attacks with if-then rules, and is not able to deal with advanced or obfuscated zero-day attacks. SVM can be used effectively with balanced data but poorly with sparse data of high dimension and does not give sequential or context-dependent threats. GNB is effective in the case of independent features, but it involves the unrealistic condition of independence of the features, particularly in the context of network traffic sequences. Logistic Regression acquires linear relationships and cannot acquire time-related relationships and antagonistic actions. However, it not time-modelled, it have no way of generalizing to new (zero-day) attacks; it is not adaptive to evolving adversarial attacks.

Concerning our hybrid approach, it will involve an ANN-based Autoencoder to compress features, a Modified Bi-LSTM to learn sequence of attack path, and Hybrid Game Theory to make adaptive defense decisions. It can learn latent attack behaviour, synthesize latent adversarial dynamics symbolically, and predict on node-level vulnerabilities even in the face of uncertainty. As such, it offers a superior performance regarding the accuracy, new attack recall and resistance to new attack techniques. Thus, our solution overcomes the disadvantage of traditional classifiers and offers a solution that can be utilized in the future and one that is powerful in the detection of zero-day attacks.

6.3.3 Performance Metrics

. The research was tested with that prism of different classifications. The information of the multi classifications was also divided into two classification issues and one verses the rest methodology adopted.

a) TPd: type d of prediction, and type d of the reality.

b) Other category D are the reality, other category d are the predicted ones.

c) FPd: the type D is researched and the other types of category d are the reality.

d) FNd: the other class category D and reality category d.

The overall accuracy/ precision and recall score was discovered as on each category on a positive sample. The accuracy can be described as in equation (6.30):

a) $Accuracy = \frac{Numberofsamplescorrectlyclassified}{Numeberofsamplesforallcategories}$------- (6.48)

The accuracy of a given category could be seen as the prediction of the accuracy of the sample, which is presented in Equation (6.49):

b) $Precision_i = \frac{TP_d}{TP_d + FP_d}$--------- (6.49)

The recall of a certain category may be considered the amount whereby the duly forecasted sample of category d covers the sample of category d in the sample set as indicated in Equation (6.50),

c) $Recall_i = \frac{TP_d}{TP_d + FN_d}$---------- (6.51)

F measure is calculated by giving the Equation.

d) $F - Measure = 2.\frac{Precison.recall}{Nprecison+recall}$ -------- (6.52)

Table 1 demonstrates the results of training and testing with the given dataset, which relates to the model suggested in this research. The training and testing were performed on the chosen dataset, in accordance with Modified Bi-LSTM Deep Neural Networks.

6.3.4 Analysis of Results and Discussions

This part talks of the experimental results of the proposed zero-day attack prediction and detection framework since the analysis of the key performance indicators of the two datasets is discussed. It also enables one to do a comparative analysis of the existing techniques to show the enhancement of the performance and show the usefulness of the proposed technique.

Table 6.1 Training and Testing Values with 10 Epochs

Epoch	Training Loss	Validation Loss	Training Accuracy	Testing Accuracy
1	0.1552	0.0614	0.9533	0.9849
2	0.0531	0.0493	0.9843	0.9840
3	0.0341	0.0474	0.9893	0.9849
4	0.0224	0.0455	0.9931	0.9846
5	0.0147	0.0589	0.9954	0.9827
6	0.0110	0.0521	0.9963	0.9850
7	0.0082	0.0510	0.9972	0.9859

8	0.0061	0.0533	0.9979	0.9869
9	0.0047	0.0608	0.9985	0.9852
10	0.0053	0.0533	0.9982	0.9872

Table 6.1 summarizes the neural network training process and it involves a number of epochs. During the training process, the model kept changing its parameters, thus streamlining training loss which continued to decrease a good indicator of learning. Also the loss on validation reduced exhibiting improved generalization to the unobserved data with minor discrepancy between successive epochs. Meanwhile, training and testing accuracies also increased with time with high values at the end of the epochs. The trends provide the testament to the fact that the model was capable of learning the discriminative trends, relying on the data and retaining the high levels of generalization that results in the high levels of accuracy on training and test data.

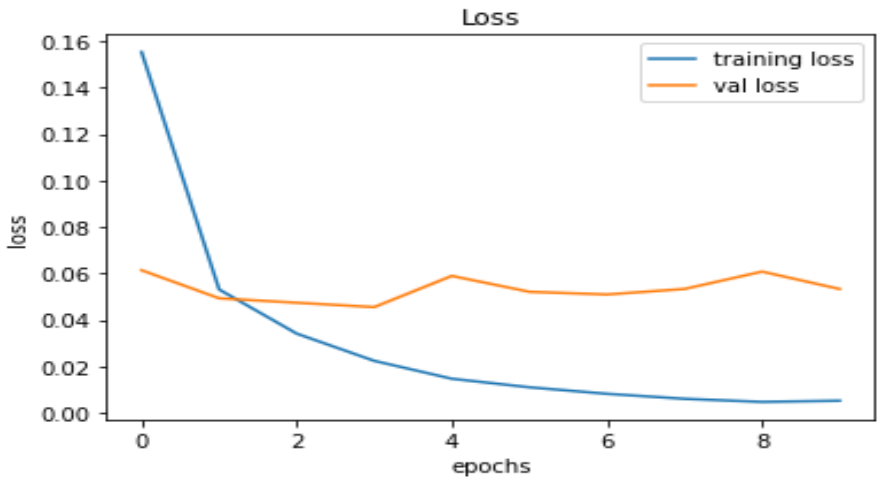

Figure 6.9 Training and Testing Loss

The proposed model is trained using values of the loss as illustrated in Figure 5.9. Where X-axis represents the number of Epochs and Y-axis represents the value of the loss.

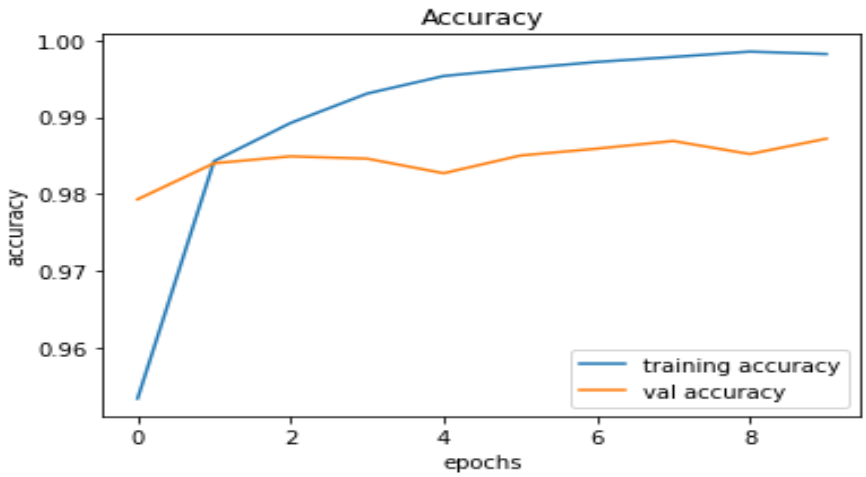

Figure 6.10 Training and Testing Accuracy

CNN-ResNet has been trained with 10 Epochs with Training and the testing accuracy is depicted in Figure 6.10 the X-axis is the Epoch number and Y-axis is the accuracy.

Table 6.2 Algorithm Comparison Chart

Dataset	Algorithm	Accuracy	Precision	Recall	F-measure
Dataset 1	DT	89	89	88	89
	SVM	84	80	86	85
	GNB	90	90	87	89
	LR	84	81	89	85
	Modified Bi-LSTM and Hybrid GT with an ANN-AE	95.4	91.3	91.5	90.4
Dataset 2	DT	85	82	83	84
	SVM	85	81	84	86
	GNB	90	84	86	85
	LR	84	81	89	85
	Modified Bi-LSTM and Hybrid GT with an ANN-AE	95	90.3	90.8	89.1

Comparison between two datasets performance shows that Modified Bi-LSTM using Hybrid Game Theory (GT) in combination with ANN-based Autoencoder (AE) is better than the regular machine learning algorithms, such as Decision Tree (DT), Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), and Logistic regression (LR) in table 54. It achieves the most accurate 95.4% on Dataset 1 which is way above the 90% other mark. The level of precision and recall indicates that the model is well balanced concerning both picking real positives and false alarm. On the same note, Dataset 2, it is a good generalizer with high accuracy of 95, good precision and recall.

Although GNB shows the constant performance in all the metrics, it only operates on the assumption of statistical feature independence without modelling an attack in terms of time and strategy. In the proposed framework, the temporal relationship between the sequence of attacks will be trained on the Bi-LSTM module, and the strategic relationships between the actions of the defender and attacker will be modelled by the Game Theory element. In addition to this, the Autoencoder removes redundancy and noise in features before classification hence enabling more discriminative learning to take place. The result of this balanced pipeline in the processing is the superior balanced precisionrecall trade-off as reflected by the superior F-measure hence the success of the proposed hybrid method in zero-day attacks prediction.

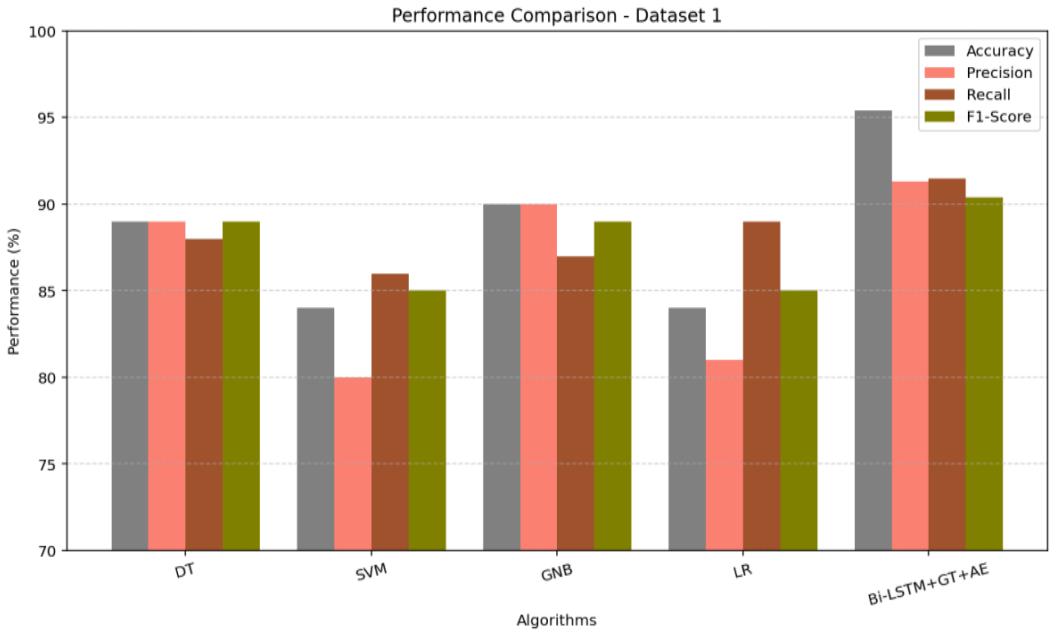

Figure 6.11 Performance Comparison Chart for Dataset 1

Benchmarking of the accuracy, precision, recall and F-measure of DT, SVM, GNB and the proposed algorithm on dataset 1 are outlined in Figure 6.11. The Performance values have been plotted in the y-axis and the algorithms in the x-axis in this chart.

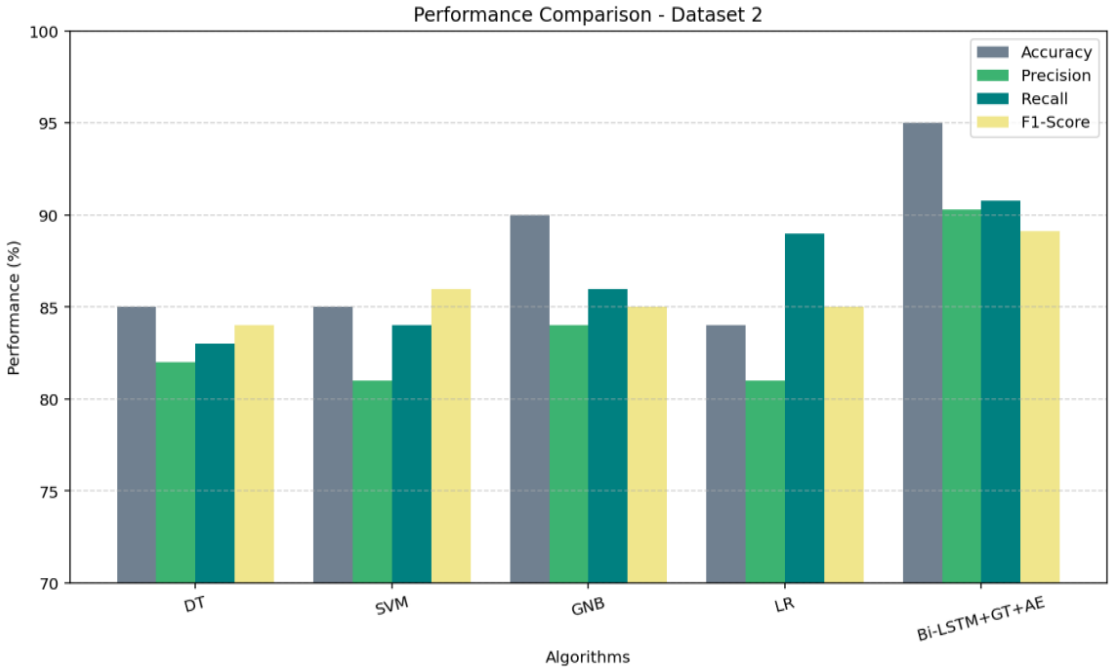

Figure 6.12 Performance Comparison Chart for Dataset 2

The accuracy, precision, recall and the F-measure of the DT, SVM and GNB and the proposed method are compared and laid down in figure 6.12 using dataset 2. The x-axis indicates the algorithms in this chart, and the y-axis indicates the performance values.

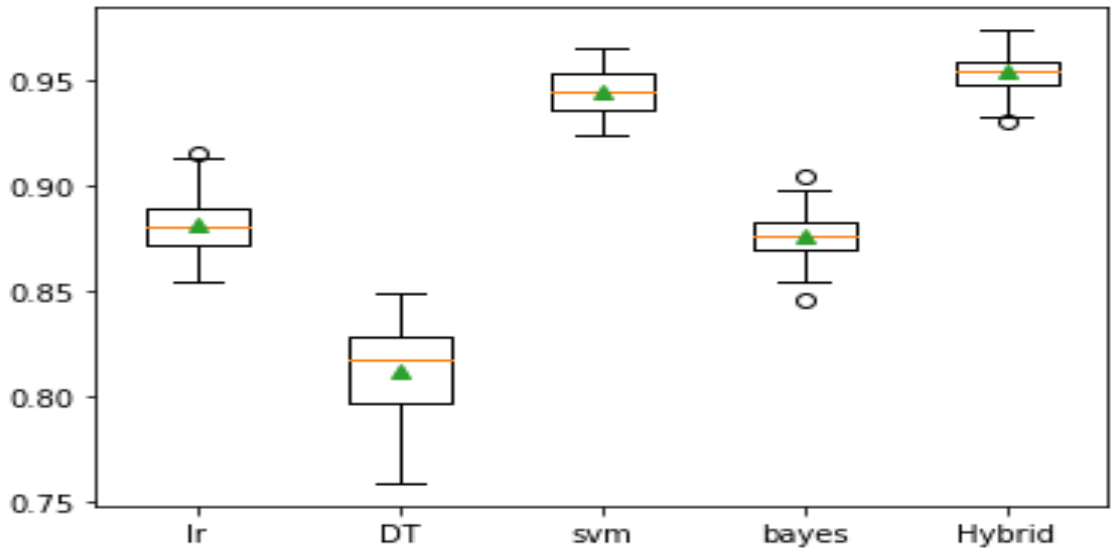

Figure 6.13 Performance Evaluation

Based on the research findings, it is seen that the proposed multi-phase zero-day attack model can be used to effectively detect the attack paths, probability of the attack, and detect the zero-day threat in a diverse network condition environment. The graph-based analysis, which is supplemented with machine learning, deep learning, and optimization strategies are the assurance that the framework is more precise in detection due to a higher F-measure and less false alarm rates compared to the baseline methods. Its results confirm the hypothesis that temporal attack behavior, strategic interaction, and maximizing feature representations can be predicted to enhance the reliability of prediction and the transferability of features across data sets. In total, the results prove the validity and applicability of the proposed framework to the research of the zero-day attacks in the dynamic and complex network environment.

6.3.5 Performance Improvement Table – Proposed Vs Existing Methods

This section represents the performance improvement of proposed model with existing methods.

Table 6.3 Dataset 1

Metric	DT → Proposed	SVM → Proposed	GNB → Proposed	LR → Proposed
Accuracy	↑ 6.4%	↑ 11.4%	↑ 5.4%	↑ 11.4%
Precision	↑ 2.3%	↑ 11.3%	↑ 1.3%	↑ 10.3%
Recall	↑ 3.5%	↑ 5.5%	↑ 4.5%	↑ 2.5%
F-measure	↑ 1.4%	↑ 5.4%	↑ 1.4%	↑ 5.4%

Based on Table 6.3, the proposed Hybrid Game Theory-ANN Autoencoder-Bi-LSTM (Modified) model performs superior to all the conventional models in all the measures of performance applied. It is clear that the values of accuracy and precision were better in Table 6.3 because the proposed model presents 11.4 and 11.3 accuracy and precision, respectively, compared to the traditional classifiers such as SVM and Logistic Regression. Furthermore, the proposed method is highly accurate and F-measure in comparison with the most powerful base model which is the Gaussian Naive Bayes (GNB) since according to the numerical values used in the table 6.3, the method has high accuracy and F-measure. These studies reveal that the model is more efficient in modeling the

dynamic and complex attack patterns and thereby specifies the suitability of the proposed model in the detection of zero-day attacks in the real-life cybersecurity setting.

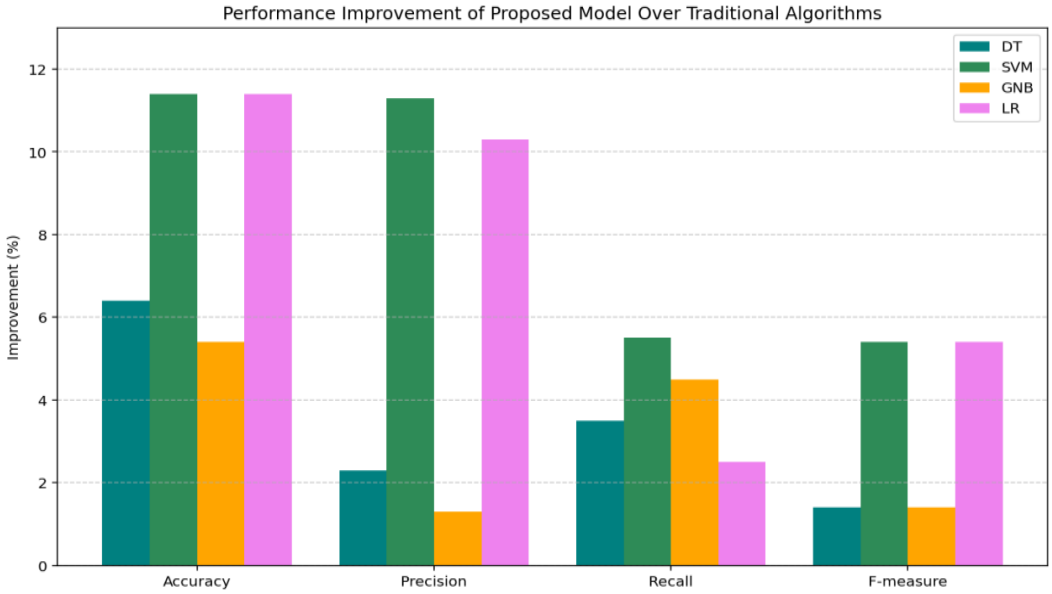

Figure 6.14 Performance Improvements on Proposed Model for Dataset 1

The percentage gain of the proposed model is presented in figure 6.14 based on the four most essential performance parameters Accuracy, Precision, Recall, and F-measure using four baseline machine learning models Decision Tree (DT), Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), and Logistic Regression (LR). The highest quality gains in precision and accuracy are in the proposed model, especially in comparison to the SVM and the LR (both approximately 11.5%). The improvements in F-measure and recall are not as significant yet exist, and SVM and LR have a clear advantage. It means that the offered methodology offers a better tradeoff between reliability and accuracy of classification.

Table 6.4 Dataset 2

Metric	**DT → Proposed**	**SVM → Proposed**	**GNB → Proposed**	**LR → Proposed**
Accuracy	↑ 10.0%	↑ 10.0%	↑ 5.0%	↑ 11.0%
Precision	↑ 8.3%	↑ 9.3%	↑ 6.3%	↑ 9.3%

Recall	↑ 7.8%	↑ 6.8%	↑ 4.8%	↑ 1.8%
F-measure	↑ 5.1%	↑ 3.1%	↑ 4.1%	↑ 4.1%

In Dataset 2, the proposed Modified bi-LSTM with Hybrid GT and ANN-AE also shows a significant performance improvement compared to all the other analogous algorithms. Above all, the accuracy is also enhanced by up to 11% and precision, by over 9% over SVM and LR. Another important aspect of the model is that it is also reliable in terms of recall and F-measure, which show that it detects better and no false negatives. Such gains also denote the strength of the model dealing with complex patterns of the zero-day attack environment.

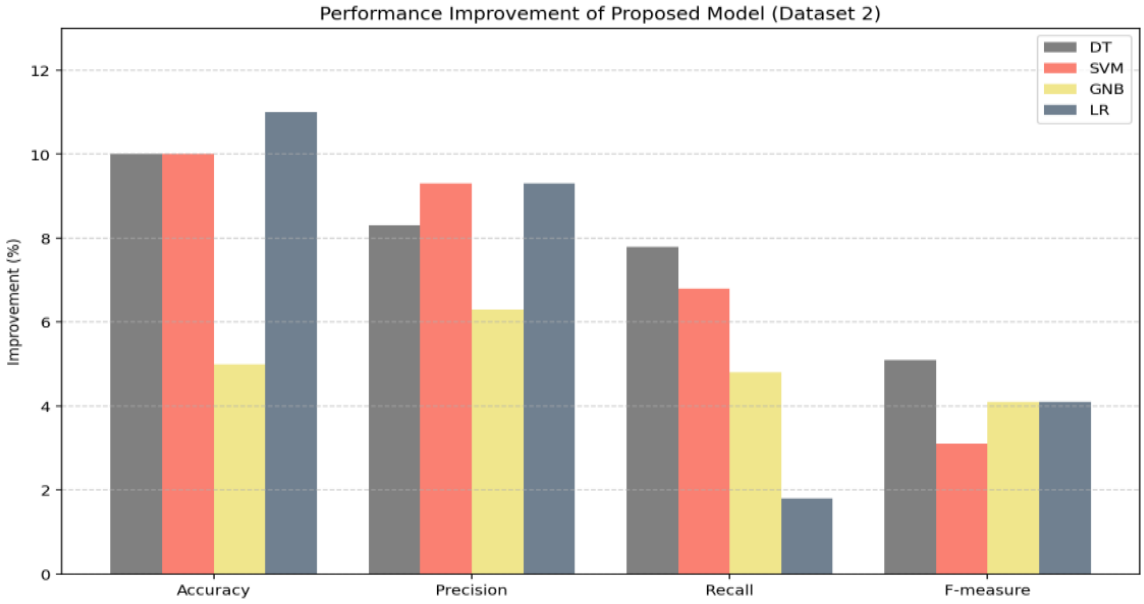

Figure 6.15 Performance Improvements on Proposed Model for Dataset 2

This figure 6.15 shows the better performance of the proposed model in relation to the conventional algorithms (DT, SVM, GNB, and LR) on Dataset 2. The proposed model ranks superior to all the four baselines on all the evaluation measures of Accuracy, Precision, Recall, and F-measure. The maximum accuracy increase (approximately 11 percent) and high precision improvement is observed in Logistic Regression (LR), whilst the recall and F-measure are high in Decision Tree (DT). The least improvement is shown

by the Gaussian Naive Bayes (GNB), which points to the fact that the suggested model is particularly better than the more robust old school classifiers.

6.3.6 Justification for Modified Bi-LSTM and Hybrid Game Theory with Autoencoders

The proposed hybrid framework is a combination of complementary strategies to address different aspects of prediction of a zero-day attack. ANN-based Autoencoder, Phase 2 (feature processing) is an input feature reduction method that attempts to reduce the high-dimensional input features and remove the redundancy before sequential learning. Phase 2 and Phase 3 apply the Modified Bi-LSTM to acquire forward and backward temporal relationships among the attack patterns so that are capable of predicting the varying patterns of zero-days precisely. Phase 2 includes Hybrid Game Theory module, to simulate the strategic correlation between attackers and defenders, and help to make risk-conscious predictions in the condition of uncertainty. These elements together with the traditional inertial classifier can overcome the limitations of the traditional inertial classifiers, as an ensemble of them learns small-scale representations, temporal strategies, and strategic actions, resulting in a higher prediction rate and allowing them to resist any new previously unknown zero-day attacks.

6.4 Chapter Summary

The framework that has been proposed in this chapter based on the idea of the Modified Bi-LSTM with Game Theory and Autoencoder-based feature compression suggested to predict the Zero-day attacks. The proposed model was found to have achieved the maximum accuracy of up to 95.4 percent that results in an improved prediction performance compared to the existing classifiers of the ML when compared to the previous models with the capacity to jointly model the temporal attacker behavior and the strategic interaction between an attacker and a defender rather than relying on the contrived patterns of the features. Despite the fact that has been improved, the stage primarily deals with sequential and strategic properties whereby it can hardly do much to affect deeper structural expressions of an attack behavior. Therefore, the second step extends the framework with deep feature learning via ResNet50 to add value to a greater spatial-behavioral reference and enhanced strength and extrapolation to more complicated zero-day attack styles.

Publications

- Akshaya S and Padmavathi G. Enhancing zero-day attack prediction a hybrid game theory approach with neural networks. International Journal of Intelligent Systems and Applications in Engineering. 2024; 12, 643-663. (Scopus)