Machine Learning Research
2025; 10(1): 69-90
http://www.sciencepublishinggroup.com/j/mlr
doi: 10.11648/j.mlr.20251001.17
ISSN: 2637-5672 (Print); ISSN: 2637-5680 (Online)

A Concave Matrix Generative Adversarial Network Model for Detecting and Enhancing Cyber-Security Threats and Awareness

Edward Fondo*, Fullgence Mwakondo, Kevin Tole

Computer Science and Information Technology, Institute of Computing and Informatics, Technical University of Mombasa, Mombasa, Kenya

Email address:
msit00302023@students.tum.ac.ke (Edward Fondo), Mwakondo@tum.ac.ke (Fullgence Mwakondo), Ktole@tum.ac.ke (Kelvin Tole)
*Corresponding author

To cite this article:
Edward Fondo, Fullgence Mwakondo, Kevin Tole. (2025). A Concave Matrix Generative Adversarial Network Model for Detecting and Enhancing Cyber-Security Threats and Awareness. *Machine Learning Research, 10*(1), 69-90. https://doi.org/10.11648/j.mlr.20251001.17

Received: 17 May 2025; **Accepted:** 3 June 1025; **Published:** 30 June 2025

Abstract: This paper introduces a novel Concave Matrix Generative Adversarial Network (CM-GAN) designed for advanced cybersecurity threat detection and contextual awareness modeling in digital infrastructures. The proposed framework integrates a concave matrix regularization mechanism to embed non-linear structural dependencies between independent (attack, defense, response) and intervening (user behavior, network load, system vulnerability) variables within the GAN learning process. Unlike traditional GAN-based models, CM-GAN enhances interpretability, training stability, and detection precision. The model is evaluated using the CSE-CIC-IDS2018 dataset and benchmarked against two customized baselines: the Matrix GAN with Awareness (MGAN) and the Matrix-Based GAN (MB-GAN). CM-GAN demonstrates superior performance across binary and multiclass classification tasks, achieving accuracy, recall, and F1-scores exceeding 99%, and demonstrating higher anomaly realism with robust detection fidelity. These results confirm the efficacy of CM-GAN as a structure-aware, context-sensitive solution for real-time cyber-threat intelligence, particularly in resource-constrained environments such as academic networks.

Keywords: CM-GAN, Concave Matrix, Cybersecurity Awareness, Threat Detection, CSE-CIC-IDS2018

1. Introduction

In the digital era, cybersecurity remains a critical concern globally, with threats becoming increasingly sophisticated and frequent. Reports show over 5.3 billion internet users worldwide, creating an expansive attack surface for cybercriminals targeting data breaches, ransomware, and distributed denial of service (DDoS) attacks [1-3]. By 2022, global cybercrime damages were estimated to surpass $6 trillion, with projections doubling by 2025 if no adequate interventions are made [4-6]. The growing adoption of artificial intelligence (AI), cloud computing, and Internet of Things (IoT) devices further expands vulnerabilities [7-9].

Africa faces mounting cybersecurity challenges due to poor infrastructure, limited awareness, and underdeveloped regulatory frameworks [10-12]. Countries such as Kenya, Nigeria, and South Africa report significant attacks targeting financial and educational institutions [13-15]. Limited incident reporting, low cybersecurity budgets, and lack of skilled personnel hinder effective response mechanisms across the continent [16-18].

Universities are especially vulnerable due to open-access networks, diverse endpoints, and poor digital hygiene among users [19-21]. As knowledge hubs and data repositories, academic institutions store valuable intellectual property and personal data, making them high-value targets for ransomware, phishing, and internal threats [22-24]. A study across African universities revealed inadequate cybersecurity policies, reactive security cultures, and lack of continuous awareness training [25-27].

To address these threats, robust detection and awareness mechanisms are essential to empower organizations with early warning systems and proactive response frameworks [28-30]. Enhancing cybersecurity threat awareness is linked to lower successful attack rates, improved incident reporting, and greater organizational resilience [31-33]. Detection systems

integrating AI models offer real-time anomaly identification, adaptive learning, and precision, especially when trained on dynamic traffic data [34-36].

Generative Adversarial Networks (GANs) offer a novel and powerful method for cybersecurity threat detection through synthetic data generation and anomaly classification [37-39]. By learning underlying data distributions, GANs can distinguish between benign and malicious network behaviors, aiding in both detection and awareness training scenarios [40-42]. Recent innovations have explored embedding GANs into matrix-based frameworks where multi-dimensional attributes such as attacks, defenses, and responses are mathematically structured using concave matrix functions for improved interpretability and performance [43].

This study proposes a novel architecture, the Concave Matrix Generative Adversarial Network (CM-GAN), for the detection of cybersecurity threats. CM-GAN introduces concave structural constraints within the adversarial training paradigm to enable the modeling of complex dependencies among cybersecurity-related variables. The integration of these structural constraints is intended to enhance both the interpretability and accuracy of threat detection mechanisms, thereby addressing key limitations observed in conventional deep learning approaches [34, 44].

A central innovation in the CM-GAN architecture is the definition of a concave degree matrix. The matrix enables the capture of nonlinear interactions wherein the influence of independent cybersecurity vectors, such as attack types, defense mechanisms, and system responses diminishes as their divergence from key intervening variables e.g., user behavior, system load, vulnerability levels increases. The matrix acts as a structural regularizer that guides both the generator and discriminator throughout the training process, enforcing meaningful learning constraints within the adversarial framework [44-46]. In the context of GANs, a concave matrix could be used to regularize Latent Space by enforcing diminishing dependency relationships among latent variables to avoid overfitting or mode collapse. It could also be used to map relationships in GANs hence representing concepts influencing generated data (e.g., attack-defense dynamics in cybersecurity). It can be used for loss function stabilization, incorporating concave penalty terms to stabilize GAN training by limiting unbounded growth in specific variables.

To support adaptive detection under real-world conditions, CM-GAN incorporates contextual variables, notably user activity patterns, network traffic characteristics, and software vulnerability profiles directly into the generative and classification components of the model. This contextual awareness enables CM-GAN to dynamically adjust to evolving threat environments, particularly in complex and open infrastructures such as university networks or heterogeneous enterprise systems [47, 48].

To validate the underlying structural assumptions of the CM-GAN framework, two benchmark models were developed;MGAN and MB-GAN. MGAN employs symbolic role assignments for matrix components, while MB-GAN adopts an algebraic representation to simulate interaction roles. These auxiliary models serve as controlled environments for comparative analysis, allowing the empirical validation of CM-GAN's performance and its theoretical robustness under varying threat scenarios [45, 49, 50].

The experimental evaluation of CM-GAN was conducted using the CSE-CIC-IDS2018 dataset, a comprehensive benchmark containing both benign and malicious network flows across six distinct attack types. CM-GAN achieved notable results, with a binary classification accuracy of 99.44% and an F1-score of 99.72%. In multiclass classification, the model reached an F1-score of 100%, indicating high precision across multiple threat categories [46, 51, 52].

An operational pipeline was implemented to support real-time anomaly detection and automated defense. This pipeline includes threshold-based triggers that initiate system updates, send notification alerts, and enforce isolation protocols when anomalous behavior is detected. These actions are dynamically influenced by feedback from the concave matrix model, allowing for an adaptive defense mechanism that can scale across various security contexts [47, 53, 54].

Finally, the CM-GAN architecture was compared against several established models, including conventional deep neural networks, Convolutional Neural Network (CNN), Deep Neural Network (DNN) and the previously developed MGAN and MB-GAN variants. Across evaluation metrics such as precision, recall, anomaly score, and training stability, CM-GAN consistently outperformed baseline methods. These findings affirm the model's potential for deployment in real-world cybersecurity systems where adaptability, accuracy, and structural transparency are critical [48, 50, 55].

The integration of a concave matrix principle allows for representing nonlinear interdependencies among cybersecurity vectors in a reduced and interpretable space [56-58]. This principle ensures the modeling of optimal intervention paths between attack detection and defensive countermeasures in university systems where resources and expertise are limited [59-61]. The proposed approach, Multi-Phase Concave Matrix GAN Model, therefore addresses a dual objective: enhancing awareness and detecting cyber threats effectively in educational digital ecosystems .

2. Problem Formulation

This section presents the mathematical formulation of the proposed CM-GAN, a model designed to generate structurally realistic cybersecurity threat data by incorporating concave relationships among system variables. CM-GAN employs two adversarial neural networks; a generator that synthesizes data and a discriminator that evaluates authenticity. A concave degree matrix M is introduced as a regularization tool to enforce structural coherence reflective of domain-specific dependencies.

2.1. CMGAN Problem Formulation

The model includes Independent Variables I; comprising of attack (A), defense (D), and response (R) matrices. Intervening Variables T, including user behavior (U), network load (N), and system vulnerability (V) and Dependent Variables Y; representing detection and awareness outcomes.

Definition of Concavity:

A function $f(x)$ is concave if:

$$f''(x) \leq 0 \quad \text{(its second derivative is non-positive).}$$

Alternatively, $f(x)$ is concave if for any two points x_1 and x_2 in its domain, and for $\lambda \in [0, 1]$:

$$f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda) f(x_2)$$

A concave matrix is a structured representation of such relationships, applied across multiple variables or concepts. It ensures that the system it models adheres to concavity principles.

The concave matrix provides a structured way to encode diminishing returns and non-linear relationships into models, making it particularly useful for systems with inherent diminishing growth dynamics. When combined with GANs, it can enhance stability, represent complex dependencies, and enforce realistic constraints on generated data.

The discriminator D outputs a probability indicating whether a sample is real or generated based on these structured inputs.

Table 1 below shows variables and their description for CM-GAN.

Table 1. *Variables and Description Table for CM-GAN.*

Symbol	Description
G	Generator network that produces synthetic cybersecurity detection and awareness outcomes
D	Discriminator network that distinguishes between real and generated data
z	Latent variable vector sampled from prior distribution p_z
x	Real sample drawn from true data distribution p_{data}
$\hat{Y}$	Generated output from the generator: $\hat{Y} = G(z, M, I, T)$
I	Independent variable matrix including attack (A), defense (D), and response (R) features
T	Intervening variable matrix representing user behavior (U), network load (N), and system vulnerability (V)
Y	Dependent variable capturing detection and awareness outcomes
M	Concave degree matrix encoding structural relationships between I and T
M_{ij}	Entry of matrix M, defined as $M_{ij} = \alpha \ln(1 + \beta \|I_i - T_j\|^2)$
α	Scaling factor in the concave function used in M_{ij}
β	Sensitivity parameter controlling the influence of the distance between I_i and T_j
λ	Regularization coefficient balancing adversarial loss and concave structural penalty
$\mathcal{R}(G, M)$	Regularization term promoting structural coherence based on matrix M

Adversarial Learning Objective Function:

The learning objective follows the standard GAN framework with the generator G and discriminator D trained as a minimax game.

Generator Input (G):

The generator takes a latent variable $z \sim p_z$ and combines it with the structured information:

$$Y = G(z, M, I, T) \tag{1}$$

where:

1. Y: Generated synthetic cybersecurity detection and awareness outcomes.
2. M: Guides the generator in structuring outputs based on feature relationships.

Discriminator (D):

The discriminator evaluates whether Y matches real-world cybersecurity metrics:

$$D(Y, M) = \sigma(h(Y, M)) \tag{2}$$

where $h(\cdot)$ measures alignment between generated Y and the matrix relationships.

This formulation encodes the relationships and enables GAN training with the concave degree matrix guiding the generation and detection of cybersecurity awareness metrics.

Mapping the Variables I, T, and Y using M is as below.

In contribution to detection and awareness (Y) is:

$$Y = g(M \cdot T + I) \tag{3}$$

where:

1. $M \cdot T$: Weighted sum of intervening variables using the concave degree matrix.
2. $g(\cdot)$: Activation or transformation function (e.g., sigmoid or ReLU).

The complete CM-GAN Objective Function is as below:

$$\begin{aligned} \min_G \max_D \mathcal{L}_{GAN} = &- \mathbb{E}_{Y \sim p_{\text{data}}}[\log D(Y, M)] \\ &+ \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z, M, I)))] \\ &+ \lambda \hat{\mathcal{R}}(Y, M) \end{aligned} \tag{4}$$

This objective includes both the adversarial losses and the regularization term $\lambda\hat{\mathcal{R}}(Y, M)$, ensuring the generator produces realistic outputs while penalizing unrealistic relationships.

Where $\lambda\hat{\mathcal{R}}(Y, M)$ is regularization term promoting alignment with concave structure and λ is hyperparameter balancing adversarial loss and regularization.

This formulation enables accurate generation of cybersecurity threats, informed outputs using domain-based concave mappings and stabilized training through structural guidance.

Concave Matrix:

To model diminishing returns and structural dependencies, we define a concave matrix M with entries computed as:

$$M_{ij} = f(I_i, T_j) = \alpha \ln(1 + \beta \|I_i - T_j\|^2) \tag{5}$$

Here, α is a scaling factor and β a sensitivity parameter. M encodes the concave influence of independent variables I on intervening variables T, guiding structured generation.

2.2. MGAN Problem Formulation

In this section, we present the mathematical formulation of the MGAN, a GAN-based cybersecurity detection model inspired by symbolic roles from *The Matrix Resurrections*. MGAN is used to validate the CM-GAN by modeling attack-defense-response dynamics and their impact on detection awareness. The MGAN architecture includes a generator G, a discriminator D, and an awareness classifier Ψ, mapped to symbolic variables such as A, D, R, U, N, V, and C.

Decision Variable Constraints:

The discriminator outputs probabilities for real and generated data:

$$D(x),\ D(G(z)) \in [0, 1], \quad \forall\, x \sim p_{\text{data}},\ z \sim p_z \tag{6}$$

The generator maps latent noise vectors to data space:

$$G : z \in \mathbb{R}^d \to \hat{x} \in \mathbb{R}^n \tag{7}$$

The awareness classifier outputs class probabilities based on intervening variables:

$$C = \Psi(U, N, V) \in \{0, 1, ..., 6\} \tag{8}$$

Adversarial Learning Objective:

The standard adversarial loss is extended in MGAN as follows:

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{9}$$

Intervening Variable Functions:

The model defines intervening variables as neural functions over symbolic matrices:

$$U = \sigma(W_u[A, D, R] + b_u) \quad \text{(User Behavior)} \tag{10}$$
$$N = \sigma(W_n[D, R] + b_n) \quad \text{(Network Load)} \tag{11}$$
$$V = \sigma(W_v[A, N] + b_v) \quad \text{(System Vulnerability)} \tag{12}$$
$$C = \sigma(W_c[U, N, V] + b_c) \quad \text{(Detection / Awareness)} \tag{13}$$

Here, σ denotes an activation function (e.g., ReLU, sigmoid, softmax), and W_* and b_* are learnable weights and biases.

MGAN Objective Function:

The final multi-objective function incorporates adversarial and detection components:

$$\mathcal{L}_{\text{MGAN}} = \lambda_1 \cdot \mathcal{L}_G + \lambda_2 \cdot \mathcal{L}_D + \lambda_3 \cdot \mathcal{L}_C \tag{14}$$

Where:

$$\mathcal{L}_G = \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{15}$$
$$\mathcal{L}_D = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{16}$$
$$\mathcal{L}_C = \text{CrossEntropy}(C, y), \quad y \in \{0, 1, ..., 6\} \tag{17}$$

The hyperparameters λ_1, λ_2, and λ_3 are weights that balance generation, discrimination, and awareness losses.

Role in CM-GAN Validation:

MGAN serves as a baseline model for validating the CM-GAN framework. While MGAN emphasizes symbolic and narrative-driven modeling of cybersecurity processes, CM-GAN builds on this by introducing concave structural regularization. Therefore, MGAN is essential for early-stage performance benchmarking and conceptual validation of the structural assumptions embedded in CM-GAN.

Table 2 below shows MGAN model variables and their description.

Table 2. *MGAN Model Variables and Descriptions Table.*

Symbol / Variable	Description	Matrix Actor / Role	CSE-CIC-IDS2018 Dataset Mapping
$\mathcal{L}_{\text{Matrix-GAN}}$	Total multi-objective loss (sum of $\mathcal{L}_G$, $\mathcal{L}_D$, and $\mathcal{L}_C$)	-	-
$\mathcal{L}_G$	Generator loss: encourages G to generate realistic attack patterns	Neo (The One)	GAN-generated synthetic attack vectors
$\mathcal{L}_D$	Discriminator loss: distinguishes real (benign) from fake (attack) data	Trinity (Classifier)	Discriminator signal outputs
$\mathcal{L}C$	Supervised loss for detection and awareness	Sati (Awareness)	Cross-entropy over Label column
$G(z)$	Generator output from noise input z	Neo	Synthetic network traffic
$D(x)$, $D(G(z))$	Discriminator outputs on real and fake samples	Trinity	Probabilistic classification outputs

Symbol / Variable	Description	Matrix Actor / Role	CSE-CIC-IDS2018 Dataset Mapping
$z \sim p_z(z)$	Latent noise vector sampled from prior	-	Random vector input (e.g., Gaussian)
$x \sim p\text{data}(x)$	Sample from real dataset	-	Feature vector from dataset
Y	Ground-truth label (0 = benign, 1-6 = attacks)	-	Label column
A	Attack matrix (malicious patterns)	Neo	e.g., Dst Port, Flow Duration, Protocol
D	Defense matrix (countermeasures)	Trinity	ACK Flag Count, URG Flag Count, etc.
R	Response matrix (reaction to attacks)	Morpheus	Bwd IAT Max, Fwd URG Flags
$U = f_U(A, D, R)$	User behavior from attack, defense, response	Bugs	e.g., Pkt Len Mean, Pkt Len Std
$N = f_N(D, R)$	Network load from defense and response	The Analyst	e.g., Fwd Pkts/s, Bwd IAT Mean
$V = f_V(A, N)$	System vulnerability from attacks and load	Smith	CWE Flag Count, Flow Byts/s
$C = \Psi(U, N, V)$	Cybersecurity detection and awareness output	Sati	Predicted class (0 or 1-6)
Ψ	Awareness function: neural mapping from $[U, N, V]$	Sati	Detection classifier (binary or multiclass)
λ_1	Weight for Generator loss	-	Typically 1.0
λ_2	Weight for Discriminator loss	-	Typically 1.0
λ_3	Weight for Awareness loss	-	Typically ≥ 2.0
W_u, W_n, W_v, W_c	Weight matrices for mapping U, N, V, C	-	Trainable parameters
b_u, b_n, b_v, b_c	Bias terms in neural functions	-	Trainable parameters
Σ	Activation function (e.g., Sigmoid, ReLU, Softmax)	-	Non-linearity in mapping layers

2.3. MB-GAN Problem Formulation

In this section, we introduce the mathematical formulation of the MB-GAN, which is designed to validate the proposed CM-GAN model by simulating and evaluating structured cybersecurity attack scenarios. MB-GAN utilizes matrix representations to encapsulate the relationships between attacks, defenses, responses, and interventions in cybersecurity systems. It leverages adversarial learning between a Generator and a Discriminator to generate structurally realistic attack data and assess system response accuracy, thereby validating the structural regularization strategies employed in the CM-GAN.

The core components of MB-GAN are two neural networks:

Generator (G):

Produces synthetic attack matrices mimicking real-world threats using latent input variables.

Discriminator (D):

Evaluates the authenticity of attack matrices and the effectiveness of corresponding system responses.

The structured nature of MB-GAN allows each attack, defense, response, and intervention to be modeled using matrices, enabling a comprehensive simulation and validation platform.

Decision Variables and Constraints:

These include Attack Matrix (A) which defines attack vectors and their attributes, Defense Matrix (D) which quantifies the system's defense mechanisms, Response Matrix (R) which maps detected attacks to automated or manual response actions and Intervention Matrix (I) which captures indirect effects such as user behavior or network load.

MB-GAN Strutural Relationship:

Attack Detection:

Multiply attack A with the transpose of D:

$$\text{Detection} = A \cdot D^T \tag{18}$$

This produces a matrix that ranks the success of defenses against various attacks.

Response Selection:

Maps detected anomalies to responses:

$$\text{Response} = \text{Detection} \cdot R \tag{19}$$

This indicates the likelihood of response being triggered by attack.

Automated Response:

If Response $>$ Anomaly Threshold, trigger actions such as sending email alerts, disabling compromised software, and updating vulnerable components.

Table 3 below shows MB-GAN Model Variables and their Description.

Table 3. *MB-GAN Model Variables and Descriptions Table.*

Symbol / Variable	Description	Dataset / Model Role
A	Attack Matrix: real or fake (simulated) cyberattack vectors	Rows = attack types; Columns = attack attributes
$A_{\text{fake}} = G(z)$	Synthetic attack matrix generated from latent vector z	Output of Generator
A_{real}	Real benign attack matrix sampled from dataset	Input to Discriminator
D	Defense Matrix capturing effectiveness of defense mechanisms	Rows = defenses; Columns = attack attributes
R	Response Matrix mapping detections to mitigation actions	Rows = detected anomalies; Columns = responses
$I = f(A, D, R)$	Intervention Matrix: indirect influences between A, D, and R	Models user feedback, load, vulnerabilities
D^T	Transposed Defense Matrix used for detection computation	Multiplied with A or $G(z)$

Symbol / Variable		Description	Dataset / Model Role
$\text{Detection} = A \cdot D^T$		Computes how well defenses handle attacks	Attack detection layer
$R_{\text{response}} = \text{Detection} \cdot R$		Maps detection to responses	Triggers automated response
$D_{\text{updated}} = D + I$		Updated defense including intervention feedback	Feedback adaptation
$\text{AdjustedResponse} = \text{Detection} \cdot R + h(I)$	=	Adjusted response accounting for interventions	Includes dynamic influence of I
$z \sim p_z(z)$		Latent noise vector sampled from prior distribution	Generator input
$G(z) \cdot D^T$		Generator's attack-defense interaction matrix	Fake data path
$A_{\text{real}} \cdot D^T$		Real attack-defense interaction matrix	Real data path
G		Generator neural network	Synthesizes A_{fake}
D		Discriminator neural network	Distinguishes A_{real} vs $G(z)$
L_D		Discriminator loss function	Measures real vs fake discrimination error
L_G		Generator loss function	Encourages G to fool D
$L_{\text{MB-GAN}}$		Overall objective loss combining L_D, L_G, and matrix-based detection terms	Final training target

Feedback Loop:

Incorporates interventions I into defense updates.

$$\text{Updated\_Defense} = D + I \tag{20}$$

Where I is the intervention matrix modeling user feedback. This adaptation accounts for changes in intervening variables to maintain optimal defense strategies.

Intervention:

Intervening factors bridge attack inputs and system responses. These factors include user behavior, network load, and software vulnerabilities.

The adjusted detection and response mechanism using I will be given by:

$$\text{Adjusted\_Response} = (\text{Detection} \cdot R) + h(I) \tag{21}$$

Where $h(I)$ models the influence of intervening factors.

By employing matrices, this model efficiently organized and computed relationships between attack patterns, defense strategies, and related responses while incorporating intervening factors for enhanced cybersecurity detection and awareness.

Adversarial Learning:

Generator Loss Function:

The generator seeks to maximize the discriminator's error:

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p_z(z)} \left[\log D(G(z))\right] \tag{22}$$

Discriminator Loss Function:

The discriminator aims to distinguish real from fake:

$$\mathcal{L}_D = -\mathbb{E}_{A_{\text{real}} \sim p_{\text{data}}(A)} \left[\log D(A_{\text{real}})\right] - \mathbb{E}_{z \sim p_z(z)} \left[\log(1 - D(G(z)))\right] \tag{23}$$

Objective Function:

The MB-GAN model objective function is:

$$\min_G \max_D \mathcal{L}_{\text{GAN}} = \mathbb{E}_{A_{\text{real}} \sim p_{\text{data}}(A)} \left[\log D(A_{\text{real}})\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log\left(1 - D(G(z))\right)\right] \tag{24}$$

Incorporating matrices:

$$\min_G \max_D \mathcal{L}_{\text{MatrixGAN}} = \mathbb{E}_{A_{\text{real}}} \left[\log D(A_{\text{real}} \cdot D^T)\right] + \mathbb{E}_z \left[\log\left(1 - D(G(z) \cdot D^T)\right)\right] \tag{25}$$

Where $A_{\text{real}} \cdot D^T$ represents real attack-defense interactions and $G(z) \cdot D^T$ represents generated attack-defense interactions.

Where A represents Attack matrix (real or fake), D^T represents Defense matrix transpose, R represents Response matrix, I represents Intervention matrix, z represents Latent space input, G represents Generator network and D represents Discriminator network.

Adjusted Detection:

Using the intervention matrix, the detection mechanism is modified as follows:

$$\text{Detection}_{\text{adjusted}} = (A \cdot D^T) + h(I) \tag{26}$$

Where $h(I)$ is a function modeling the influence of intervention factors and $A \cdot D^T$ is the base detection from attack and defense interaction.

Adjusted Response:

Responses are updated based on intervention factors as follows:

$$\text{Response}_{\text{adjusted}} = (\text{Detection}_{\text{adjusted}} \cdot R) + g(I) \tag{27}$$

Where $g(I)$ models how intervention factors influence response actions.

Impact on Generator and Discriminator:

Intervention factors enhance the adversarial dynamics between the Generator G and Discriminator D:

The generator G adapts to intervention factors by generating attack vectors A_{fake} that account for these dynamic influences:

$$A_{\text{fake}} = G(z, I) \tag{28}$$

Where I influences the synthetic attack generation process.

The discriminator D evaluates both real data and synthetic data under the influence of intervention factors:

$$D(A_{\text{real}}, I) \quad \text{and} \quad D(G(z, I), I) \tag{29}$$

The intervention factors I are passed as additional inputs to help the discriminator refine its evaluation.

Final MB-GAN Model Objective Function with Interventions:

The intervention matrix is incorporated into the overall objective function as follows:

$$\min_G \max_D \mathcal{L}_{\text{MB-GAN}} = \mathbb{E}_{A_{\text{real}}} \left[\log D\left(A_{\text{real}} \cdot D^T + h(I)\right)\right] + \mathbb{E}_z \left[\log\left(1 - D\left(G(z) \cdot D^T + h(I)\right)\right)\right] \tag{30}$$

3. Proposed Method

This section presents a novel optimization framework for cybersecurity threat detection in university software systems using a CM-GAN. The CM-GAN integrates adversarial learning with a concave matrix transformation to dynamically model the complex interactions among attack-defense-response behaviors influenced by contextual factors such as user behavior, network load, and system vulnerabilities. The optimization is guided by a comprehensive objective function that combines adversarial loss and structural regularization, as shown below:

3.1. CM-GAN Proposed Method Objective Function

$$\min_G \max_D \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z, M, I, T)))] + \lambda R(G, M) \tag{31}$$

where D is the discriminator that distinguishes real data x from generated samples, G is the generator that synthesizes data from latent variables z, M is a concave matrix capturing mappings between independent and intervening variables, I, T are independent and intervening variable matrices and $\mathcal{R}(G, M)$ is regularization term enforcing structure in G with respect to M.

This objective ensures not only adversarial training fidelity but also structurally guided learning, making the model more suitable for real-time anomaly detection.

Algorithm 1 CM-GAN Optimization

Require: Attack, defense, and response matrices A, D, R
Ensure: Optimized CM-GAN model parameters

1: Initialize Generator G and Discriminator D
2: Construct Independent Variable Matrix: $I = f(A, D, R)$
3: Compute Intervening Variable Matrix: $T = f(U, N, V)$
4: Construct Concave Matrix: $M_{i,j} = \frac{\partial U_i}{\partial T_j^2}$
5: Compute Structural Representation: $Y = \Psi(U, N, V)$
6: Form Objective Function:

$$\min_G \max_D \mathbb{E}_x[\log D(x)] +$$

$$\mathbb{E}_z[\log(1 - D(G(z, M, I, T)))] + \lambda R(G, M)$$

7: Optimize G and D via adversarial training and concave matrix regularization
8: **return** updated CM-GAN parameters

Algorithm 2 CM-GAN Anomaly Detection and Defense Enhancement

Require: Test sample x, Discriminator D, Generator G, threshold τ, intervention function $f(I)$
Ensure: Detection result (Normal or Anomaly), automated defense actions

1: Normalize input x
2: Generate synthetic sample: $\tilde{x} = G(z)$
3: Evaluate Discriminator score: $s = D(x)$
4: **if** $s < \tau$ **then**
5: Classify x as anomaly
6: Trigger enhancement mechanisms:
- Send alert email if $I_{\text{email}} > \delta$
- Disable insecure software if $I_{\text{disable}} > \delta$
- Initiate model update if $I_{\text{update}} > \delta$

7: **end if**
8: Log action and return result

Algorithm 3 CM-GAN Training with Benchmark Validation

Require: Preprocessed CSE-CIC-IDS2018 dataset, comparison models (MGAN, MB-GAN), performance metrics

Ensure: Validated CM-GAN performance

1: Preprocess dataset: clean, merge, normalize, transform
2: Train models:
- MGAN with static attack-defense-response matrices
- MB-GAN incorporating intervention-aware adjustments
- CM-GAN using dynamic concave matrix and real-time adaptation

3: Evaluate all models on:
- Accuracy, Precision, Recall, F1-Score
- Anomaly Score and Detection Rate

4: Benchmark against DNN and CNN baselines **return** Final performance metrics and rankings

This proposed methodology ensures that the CM-GAN framework adapts to real-time cybersecurity scenarios by modeling not only fixed input-output patterns but also evolving user behavior and system states. The integration of the concave matrix structure enhances the model's ability to capture non-linear relationships and fine-grained threat signals that are otherwise obscured in high-dimensional data.

3.2. MGAN Methodology for Validating CM-GAN

This methodology outlines the validation process for the proposed CM-GAN using the traditional MGAN framework. MGAN serves as a baseline to assess the effectiveness of CM-GAN in detecting cyber threats within complex network environments. By integrating classic GAN components with matrix-inspired hierarchical structures, this approach provides a comprehensive validation strategy for CM-GAN.

Inputs:

These will include training data (e.g., CSE-CIC-IDS2018), MGAN model parametersand performance metrics (e.g., Accuracy, Precision, Recall, F1-Score).

Outputs:

These will include, validated CM-GAN performance metrics and comparative analysis with MGAN baseline.

MGAN Methodology:

The methodology involve the following steps:

1. *Data Preprocessing*: Clean, merge, normalize, and transform raw training data. Extract relevant features to form the attack-defense-response matrix, capturing essential security attributes.
2. *Matrix Construction:* Construct static attack-defense-response matrices for MGAN, encoding relationships between network behaviors and cybersecurity responses. Use fixed matrix structures without dynamic adaptation to provide a control comparison.
3. *Model Training:* Train MGAN using the processed dataset, optimizing generator G and discriminator D networks. Use adversarial loss functions to improve the generator's capability to replicate attack patterns.
4. *Performance Evaluation:* Evaluate MGAN on key metrics, including Accuracy, Precision, Recall, F1-Score, and Anomaly Detection Rate. Record results as baseline metrics for CM-GAN comparison.
5. *Validation:* Use the MGAN-generated outputs as a reference to assess CM-GAN's enhanced detection capabilities. Perform statistical significance testing to validate performance improvements.
6. *Benchmarking:* Compare MGAN results with CM-GAN under identical test conditions to quantify the impact of concave matrix integration.
7. *Reporting:* Summarize results, highlighting areas of improvement and potential limitations.

Algorithm 4 MGAN Training

Require: Training data, static attack-defense-response matrices, performance metrics

Ensure: Validated MGAN performance

1: Preprocess dataset: clean, merge, normalize, transform
2: Construct static attack-defense-response matrices
3: Train generator and discriminator:
- Minimize generator loss L_G to produce realistic attack samples
- Maximize discriminator loss L_D to distinguish real from synthetic samples

4: Evaluate model on Accuracy, Precision, Recall, F1-Score, and Anomaly Detection Rate **return** Final performance metrics for CM-GAN validation

The MGAN methodology provides a structured approach to validate the CM-GAN model, leveraging static matrices to benchmark performance. This method ensures the proposed CM-GAN outperforms traditional GANs in cybersecurity contexts, enhancing detection and response capabilities.

MB-GAN Methodology for Validating CM-GAN:

This methodology presents a matrix-based approach for validating the CM-GAN using the MB-GAN framework. The MB-GAN methodology integrates matrix operations for precise feature extraction, enhancing the accuracy of cyber threat detection. It uses predefined matrix structures to model attack, defense, and response interactions, forming a critical benchmark for CM-GAN performance evaluation.

MB-GAN Methodology Input-Output Variables

Inputs:

The input variables to the MB-GAN were, training data (e.g., CSE-CIC-IDS2018, MB-GAN model parameters and performance metrics (e.g., Accuracy, Precision, Recall, F1-Score)

Outputs:

The output variables to the MB-GAN were-A Validated CM-GAN performance metrics and a Comparative analysis with MB-GAN baseline

Algorithm 5 MB-GAN Methodology

1: **Data Preprocessing:** Clean, merge, normalize, and transform raw training data. Extract relevant features to form the attack-defense-response matrix.
2: **Matrix Construction:** Define matrices for Attack (A), Defense (D), Response (R), and Interventions (I), where:
- A captures attack attributes from input data.
- D models defense mechanisms, reflecting security responses.
- R represents system responses to detected threats.
- I incorporates real-time intervention feedback.

3: **Matrix-Based GAN Training:** Train MB-GAN using the following final objective function with interventions:

$$\min_G \max_D \mathcal{L}_{\text{MB-GAN}} = \mathbb{E}_{A_{\text{real}}}\Big[\log D\left(A_{\text{real}} \cdot D^T + h(I)\right)\Big] + \mathbb{E}_z\Big[\log\Big(1 - D\left(G(z) \cdot D^T + h(I)\right)\Big)\Big]$$

4: **Performance Evaluation:** Assess MB-GAN performance using Accuracy, Precision, Recall, F1-Score, and Anomaly Detection Rate. Record results as baseline metrics for CM-GAN comparison.
5: **Validation:** Use the MB-GAN-generated outputs as a benchmark for evaluating CM-GAN's advanced detection capabilities.
6: **Benchmarking:** Compare MB-GAN results with CM-GAN under identical test conditions to quantify the impact of concave matrix integration.
7: **Reporting:** Summarize results, highlighting areas of improvement and potential limitations.

The MB-GAN methodology provides a comprehensive framework for validating the CM-GAN model, integrating matrix operations to enhance the accuracy and robustness of anomaly detection in cybersecurity applications. This approach establishes a strong baseline for performance comparison, ensuring the CM-GAN's superiority in real-world threat detection.

4. Experimental Setup for the CM-GAN, M-GAN, and MB-GAN Model

The proposed intrusion detection method was implemented in Python using a deep learning tool on an Intel(R) Core(TM) i5-3320M CPU @ 2.60 GHz computer, utilizing 64 cores and 500 GB Hard Disk Drive with 12 GB of RAM. For faster processing, the model was trained on Google Colab's v5e-1 TPU in the Gemini environment with TensorFlow.

The CIS-CIC-IDS-2018 dataset was split into training and testing sets, where the training set was used to build the model and the testing set was used to evaluate its performance. The method aimed for high accuracy and low false alarms by leveraging detailed features from the dataset. Training utilized the CMGAN, M-GAN, and MB-GAN model algorithms, efficiently processing large datasets with Pandas, NumPy, and scikit-learn.

4.1. Experimental Environment

Table 4 below shows the hardware and software requirements for the project experiments.

Table 4. *Experimental Environment for CMGAN, M-GAN and MB-GAN.*

Project Requirements	Properties
OS	Windows 10 Pro
CPU	Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz 2.60 GHz
TPU	v5e-1 TPU
Memory	12 GB
Disk	500 GB
Python	Python 3.10
Graphics	NIVIDIA GTX 1080 Ti
Framework	TensorFlow 2.16.1

The experimental setup trained the MGAN, MB-GAN, and CM-GAN models and then tested them to produce an output summary report showing the accuracy, loss functions for both the Generator and Discriminator, precision, anomaly rate, anomaly score, recall, and F1-score.

4.2. Binary Classification Training Output

This section discusses on binary classification training output for all the three models of interest.

Model Training Pipeline:

The model training pipeline figure 1 below shows the models training processes.

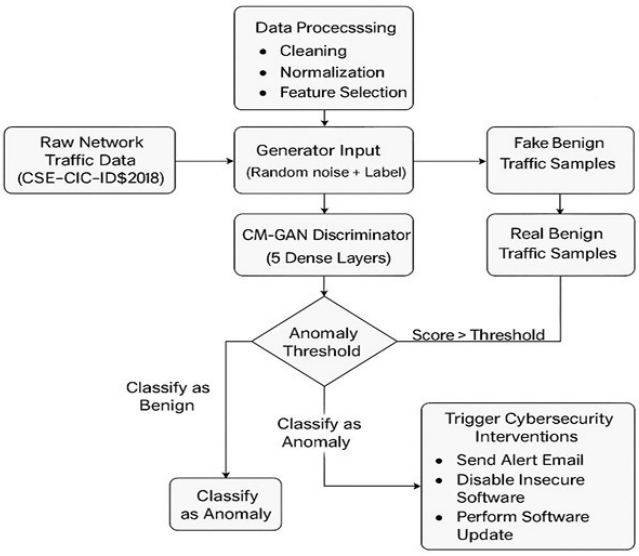

Figure 1. *CM-GAN Model Training Pipeline.*

4.3. CM-GAN Model Binary Classification Training Output

The graph figure 2 below shows M-GAN model for binary classification training output with confidence intervals.

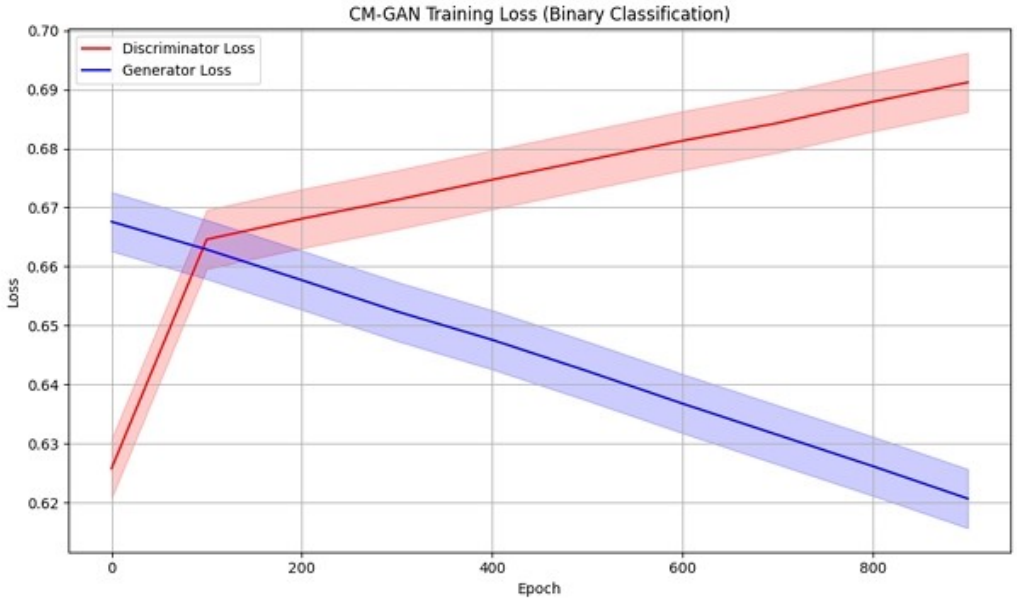

Figure 2. *CM-GAN Model Binary Classification Training Output Plot That Includes Clearly Labeled Lines and Shaded Confidence Intervals for Easy Visualization of Training Stability.*

4.4. MGAN Model Binary Classification Training Output

The graph figure 3 below shows MGAN Model Binary Classification Training Output with narrow confidence intervals.

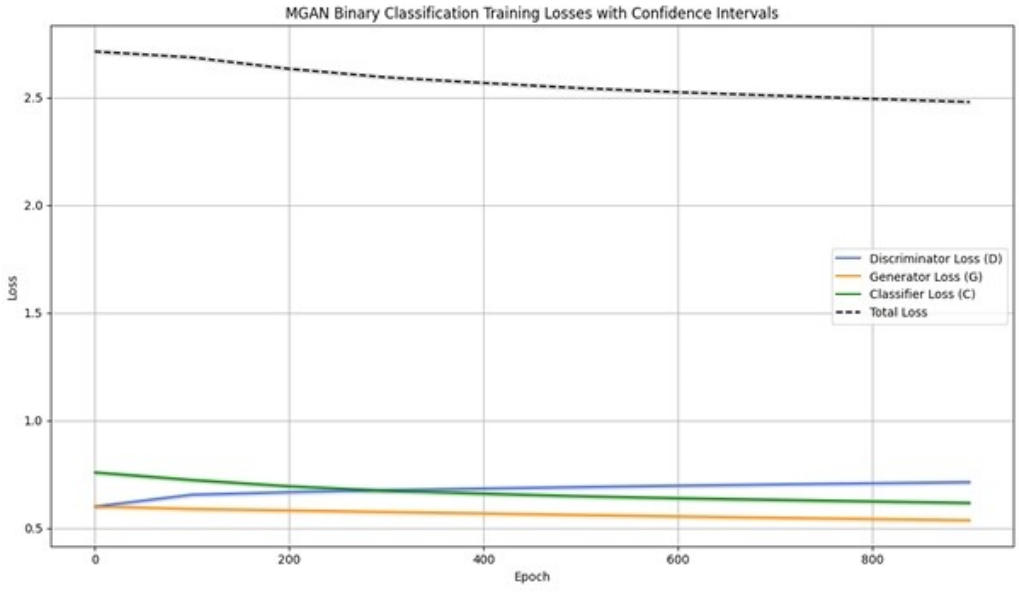

Figure 3. *MGAN Model Binary Classification Training Output With Confidence Intervals Visually Shown Using Shaded Regions Around Each Line.*

4.5. MB-GAN Model Binary Classification Training Output

The graph figure 4 below shows MB-GAN model binary classification training output with confidence intervals.

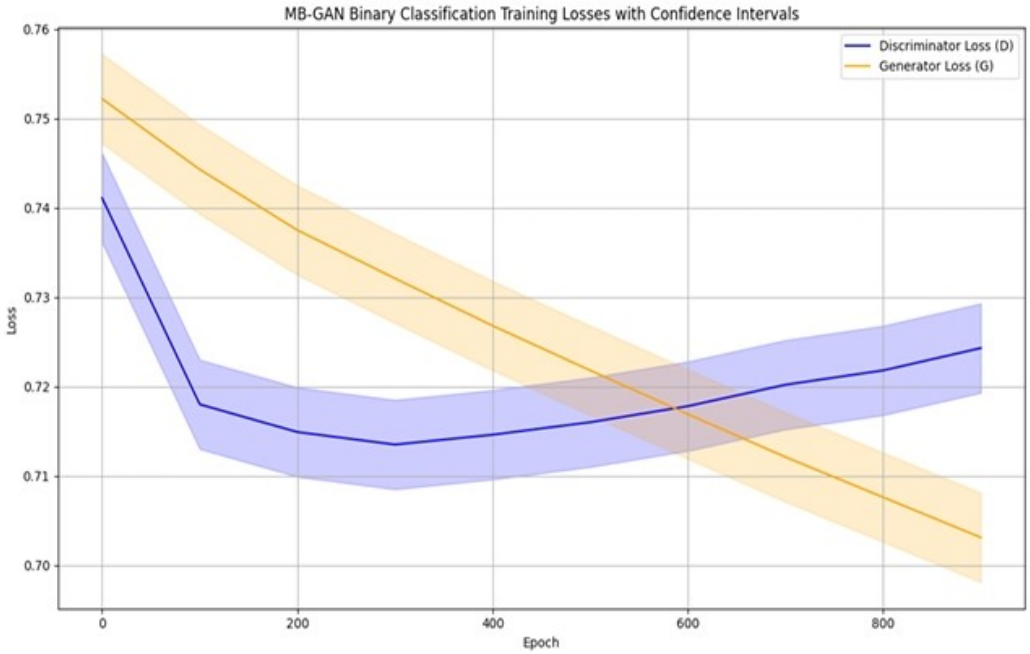

Figure 4. *MB-GAN Model Binary Classification Training Output With Confidence Intervals Visualized as Shaded Bands Around Each Loss Line. Losses Show the Expected Gradual Trends for Both D and G, With the Generator Loss Decreasing Consistently.*

4.6. CM-GAN, M-GAN and MB-GAN Models Loss Graphs Comparison for Binary Classification

Below is a comparative analysis of the training loss behaviors for the three models: CMGAN, MGAN (Matrix GAN with Awareness) and MB-GAN, all for binary classification.

4.7. CM-GAN Loss Graph - Binary Classification

Figure 2 displays the training loss progression of a GAN model over 900 epochs.

Axes Interpretation:

1. For X-Axis (Epoch), training epochs were from 0 to 900.
2. For Y-Axis (Loss), loss values ranged from approximately 0.62 to 0.69.

Loss Curves:

Discriminator Loss (Red Line with Circles):

Starts low ($\sim$ 0.628), then spikes up quickly around epoch 100. Then increases steadily and linearly across epochs. Finally reaches a peak close to 0.695 by epoch 900.

Generator Loss (Blue Line with Diamonds):

Starts high ($\sim$ 0.667). Generally, it gradually and consistently decreases with training then finally drops to about 0.621 by epoch 900.

Interpretation of Training Behavior:

For the adversarial balance shift, the loss curves show a kind of concave progression, especially evident in the generator loss-typical when GANs stabilize. A steady increase in discriminator loss suggests it's increasingly unable to distinguish real from fake, possibly because the generator is improving.

For concave pattern & saturation, the loss curves show a kind of concave progression, especially evident in the generator loss-typical when GANs stabilize. A steady increase in discriminator loss suggests it's increasingly unable to distinguish real from fake, possibly because the generator is improving.

Regarding training stability, there's no oscillation or collapse, which is a good sign in GAN training. The converging trend of both losses, discriminator loss rising and generator loss falling implies the model may be reaching a Nash equilibrium.

Implications for Binary Classification:

The generator is likely producing samples that match the real distribution well enough to confuse the discriminator consistently by later epochs. This training could result in a generator model capable of producing high-quality adversarial examples for binary classification tasks which is very useful for anomaly detection or synthetic augmentation in cybersecurity.

Figure 2 illustrates that Discriminator Loss continuously increases from $\sim$0.63 to $\sim$0.69. Generator Loss consistently decreases from $\sim$0.67 to $\sim$0.62.

Performance indicates classic adversarial behavior; whereas the generator improves, the discriminator adapts to be more aggressive. This shows the most linear and symmetric adversarial training dynamics among the three.

MGAN Loss Graph (With Awareness Loss) - Binary Classification:

Figure 3 visualizes the training behavior of MGAN with Awareness Supervision Loss (C) over 900 epochs. Below is a detailed breakdown:

Discriminator Loss (D):

The graph figure 3 slightly increases over time from 0.60 to $\sim$ 0.71. The discriminator is finding it increasingly difficult to distinguish between real and fake samples, which often means the generator is improving. A rising discriminator loss is a healthy sign if the generator loss decreases accordingly.

Generator Loss (G):

The graph figure 3 shows gradual decrease from $\sim$ 0.60 to $\sim$ 0.53. Indicates that the generator is producing more realistic samples over time. A lower generator loss implies it is successfully fooling the discriminator.

Awareness Loss (C):

The graph figure 3 shows steady decline from $\sim$ 0.76 to $\sim$ 0.62. This loss is tied to the supervised awareness mechanism, which likely enhances the model's understanding of labeled data e.g., anomaly awareness or class discrimination. A consistent decrease shows that the model is learning to classify or respond to known threats more effectively.

Total Loss:

The graph figure 3 shows smooth decrease from $\sim$ 2.71 to $\sim$ 2.48. The overall objective function is being minimized, which is the desired outcome. This indicates stable training and effective convergence.

Hence the MGAN shows stable training, improving generator, awareness model is learning and discriminator resistance which is typical behavior of a well-behaved binary GAN training process with auxiliary supervision.

Figure 3 illustrates that the Discriminator Loss (D) and Generator Loss (G) are relatively stable and converge near $\sim$0.65 and $\sim$0.58 respectively. Awareness Loss (C) starts higher ($\sim$0.75) and decreases gradually. Total Loss (dashed line) declines slowly, indicating converging training stability.

Performance Indicates balanced training between generator and discriminator. Inclusion of Awareness Loss adds a third learning dimension that improves robustness without destabilizing learning.

4.8. MB-GAN Loss Graph- Binary Classification

Figure 3 shows the training losses of a MB-GAN used for binary classification, plotted over training epochs. There are two curve, the red line (Discriminator Loss) and blue line (Generator Loss)

For Discriminator Loss (Red Line with Circles), curve starts at around 0.741 at epoch 0. It drops significantly in the first 100 epochs, indicating that the discriminator is quickly learning to distinguish between real and fake data. After about epoch 300, the loss stabilizes and then starts to slightly increase again toward epoch 900. This slight increase suggests that the generator is improving, making it harder for the discriminator to differentiate between real and fake samples.

For Generator Loss (Blue Line with Triangles), this curve starts higher, around 0.752, and steadily decreases throughout the training process. This consistent drop indicates that the generator is gradually improving its ability to produce realistic data. The crossing point of the generator and discriminator losses happens around epoch 600, which is a sign that the training is balanced and both models are learning competitively.

Interpretation and Insights:

The generator improves (loss decreases), and the discriminator struggles more (loss increases), which is expected in a well-trained GAN. The crossing point is a potential sign of balance between the generator and discriminator, a critical aspect in stable GAN training.

The losses are relatively stable with smooth trends. This indicates that the GAN is not suffering from mode collapse or training instability.

Final Observation (Epoch 900) shows Generator loss is lower than discriminator loss, meaning the generator is doing a good job fooling the discriminator.

Figure 3 illustrates that Discriminator Loss initially decreases then starts rising after $\sim$400 epochs. Generator Loss steadily decreases from $\sim$0.75 to $\sim$0.70 and lower.

Performance indicates early convergence, but after $\sim$400 epochs, the discriminator starts dominating, suggesting potential mode collapse risk. Generator struggles to improve further as discriminator tightens accuracy.

Table 5 below shows comparison of all the three GAN models loss graphs for binary classification.

Table 5. *Comparison of GAN Models Loss Graphs for Binary Classification.*

No.	Metric	MGAN	MB-GAN	CM-GAN
1.	Training Stability	High	Moderate	Moderate
2.	Convergence Speed	Slow but steady	Fast initially, then plateaus	Consistent slope, gradual
3.	Generator Loss Improvement	Gradual, stable	Clear downward trend	Clear downward trend
4.	Discriminator Behavior	Stable	Increasing after midpoint	Continuously increasing
5.	Additional Component	Awareness Loss (C)	None	Concave regularization
6.	Risk of Mode Collapse	Low	Moderate to High	Low

Binary Classification Loss Graphs Remarks:

MGAN offers the most stable and multi-dimensional training, making it better suited for complex environments of cybersecurity where awareness and adaptability are key. MB-GAN shows signs of imbalance after 400 epochs; while initially effective, it may not generalize well. CM-GAN maintains adversarial tension effectively and offers better generalization than MB-GAN, but lacks the rich awareness

learning present in MGAN.

4.9. Multiclass Classification Training Output

Below are three graphs showing the training output for CM-GAN, MGAN and MB-GAN multiclass classification with confidence intervals.

CM-GAN Model Multiclass Classification Training Output:
Graph figure 5 below shows CM-GAN model multiclass classification training output.

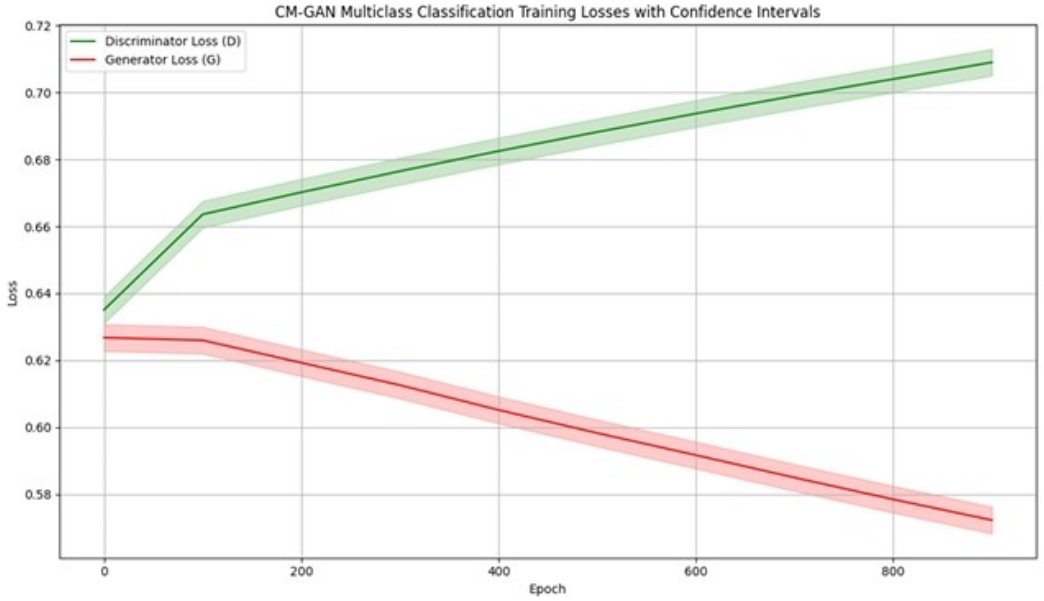

Figure 5. *CM-GAN Model Multiclass Classification Training Output With Confidence Intervals Showing Shaded Areas ± Standard Deviation (0.004) around each loss curve. The Discriminator loss increases slightly, while the Generator loss steadily decreases, indicating expected adversarial learning dynamics.*

MGAN Model Multiclass Classification Training Output:
Graph figure 6 below shows MGAN model multiclass classification training Output.

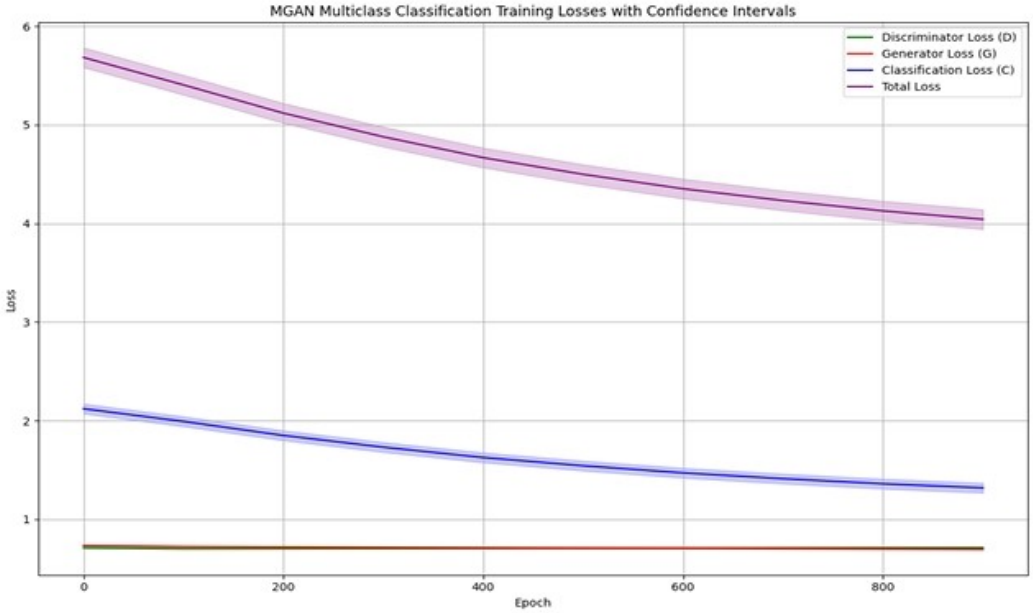

Figure 6. *MGAN Model Multiclass Classification Training Output. Discriminator (D) and Generator (G) Losses Decrease Slightly. Awareness (C) and Total Loss Decline More Substantially, Showing Effective Learning.Confidence Intervals Show Expected Variance Across Training Runs.*

MB-GAN Model Multiclass Classification Training Output:
Graph figure 7 below shows MB-GAN model multiclass classification training output.

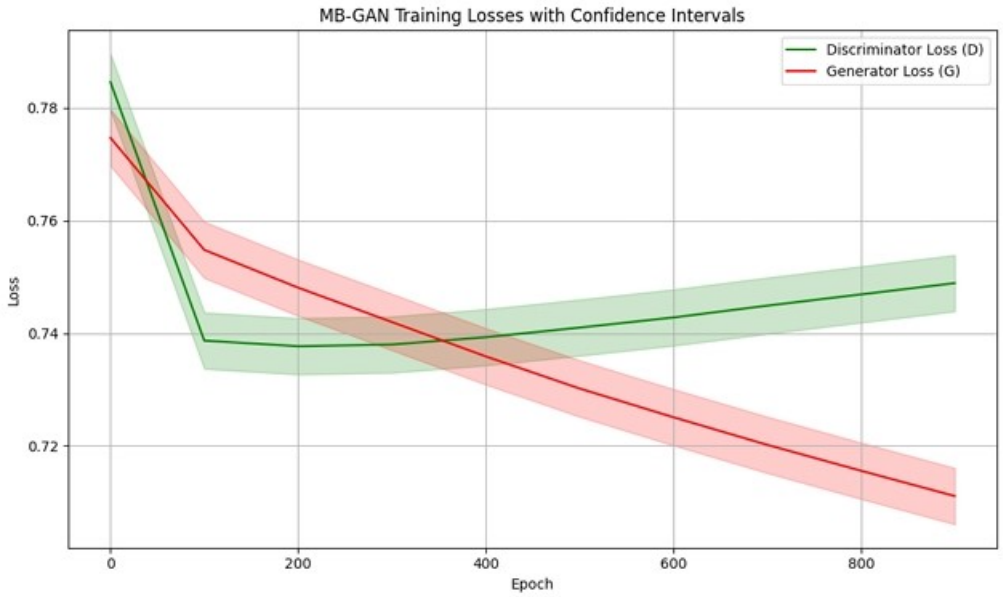

Figure 7. *MB-GAN Model Multiclass Classification Training Output. Smooth Decline In Generator Loss, Indicates Improving Generation. Slight Increase in Discriminator Loss, Which is Common as the Generator Improves.Confidence intervals (± 0.005) Show the Natural Variability Expected Across Multiple Runs.*

4.10. CM-GAN, M-GAN and MB-GAN Models Loss Graphs Comparison for Multiclass Classification

Discussed below is a comparison of the training performance for MGAN, MB-GAN, and CM-GAN for multiclass classification, based on the provided loss curves.

CM-GAN - Multiclass Classification:

Figure 5 shows the loss trends overview. Discriminator Loss (Red Line), increases steadily from 0.6351 (Epoch 0) to 0.7090 (Epoch 900). This suggests the discriminator is struggling more over time to distinguish real vs. generated data, a possible indication that the generator is improving.

Generator Loss (Blue Line), decreases gradually from 0.6268 (Epoch 0) to 0.5723 (Epoch 900). Indicates that the generator is learning to generate more realistic samples that better fool the discriminator.

Interpretation:

The divergent trend, the discriminator loss increasing and the generator loss decreasing is a typical GAN training behavior. The generator is improving; hence the discriminator finds it harder to tell fake from real. This is not necessarily a bad thing, but suggests generator dominance or stable adversarial training depending on the context.

Figure 5 illustrates Generator Loss steadily decreases while Discriminator Loss steadily increases. A more pronounced divergence between G and D losses is observed. The discriminator gets stronger while the generator improves slower. This may suggest instability and reduced generative diversity or performance over time.

Table 6 below shows summary of CM-GAN training (Multiclass Classification).

Table 6. *Summary of CM-GAN Training (Multiclass Classification).*

Component	Trend	Implication
Discriminator	Increasing	Generator getting better; D struggling more
Generator	Decreasing	Learning realistic sample generation
Training Health	Stable so far	Indicates meaningful adversarial learning

MGAN - Multiclass Classification:

Figure 6 illustrates the training loss trends of the MGAN over 900 epochs. The model exhibits stable and balanced training behavior across its core components. The discriminator loss (D) and generator loss (G) remain relatively steady, centered around 0.71 and 0.69 respectively, indicating a consistent adversarial balance where neither component overpowers the other.

Notably, the awareness loss (C), responsible for multiclass classification performance, demonstrates a clear and steady decline from 2.12 to 1.32. This trend signifies progressive improvement in the model's ability to distinguish between multiple attack classes. The total training loss, combining all components, shows a consistent downward trajectory, decreasing from 5.68 to 4.04, which reflects overall convergence of the MGAN architecture.

These results confirm that the MGAN framework is effective for multiclass cybersecurity threat detection, with significant gains in classification accuracy while maintaining adversarial stability.

Figure 6 illustrates all loss components D, G, C, and Total consistently decrease over epochs. The Awareness Loss (C) starts higher than others but steadily reduces. Total Loss

(dashed black line) shows smooth convergence. MGAN shows a well-balanced and stable training process with coordinated decrease across all relevant losses. This suggests good overall model learning and control over class separation via awareness loss.

MB-GAN - Multiclass Classification:

X-axis in the graph figure 7 shows training epochs (0 to 900) and Y-axis shows loss values.

There are two curves, Discriminator Loss (Red line with circles) and Generator Loss (Blue line with triangles).

Discriminator Loss (D Loss):

Starts high at ∼0.7846 at epoch 0 then drops sharply by epoch 100 to around 0.7387 showing that the discriminator is quickly learning to distinguish fake from real. After epoch 200, it flattens and then starts to slightly increase, reaching ∼0.7489 by epoch 900. This increase is expected when the generator starts generating more realistic samples, making the discriminator's job harder.

Generator Loss (G Loss):

Begins at ∼0.7747 and steadily decreases throughout training, ending at ∼0.7111 at epoch 900. This indicates progressive improvement in the generator's ability to fool the discriminator. The smooth and consistent drop without sharp oscillations is a very good sign of stable training.

Crossing Point (∼Epoch 300-400):

The two curves intersect between epochs 300 and 400. This balance point often reflects when both models are learning competitively. The generator becomes strong enough to challenge the discriminator and the discriminator is still able to learn but starts losing its upper hand.

Interpretation:

The discriminator loss slowly increases, while the generator loss decreases, forming a crisscross pattern which is a classic GAN training behavior. There's no sudden spike or collapse, suggesting no mode collapse or instability which is often a challenge in GAN training. This training trajectory aligns well with convergence behavior in adversarial learning setups.

Figure 7 illustrates Generator Loss consistently decreases. Discriminator Loss initially decreases, then increases slightly after ∼300 epochs. The generator improves steadily, but the discriminator starts to overpower the generator, indicating possible training imbalance in later epochs. The increase in D-loss suggests the discriminator is getting better at distinguishing fake from real, possibly too quickly.

Table 7 below shows summary of MB-GAN model training-multiclass classification.

Table 7. *Summary of MB-GAN Model Training-Multiclass Classification.*

Metric	Observation
Stability	High – no oscillations or divergence
Discriminator Trend	Learning early, challenged later
Generator Trend	Steadily improving
Balance Point	∼Epoch 300–400
Overfitting Signs?	No clear signs — smooth losses

Overall Multiclass Loss Graphs Comparison Models Summary:

Table 8 below shows the overall multiclass loss graphs comparisons models summary.

Table 8. *Overall Multiclass Loss Graphs Comparison Models Summary.*

Model	Stability	Loss Convergence	Balance (G vs D)	Multiclass Suitability
MGAN	High	Yes (all losses)	Good	Best performance due to awareness integration and stable loss convergence
MB-GAN	Moderate	Partial	Moderate	Fair, but with some imbalance in later training stages
CM-GAN	Low	Divergent	Poor	Least effective; possible instability from growing discriminator strength

4.11. Testing Results and Analysis for CM-GAN, M-GAN Model and MB-GAN Models - Binary Classification

CM-GAN Binary Classification Test Results:

Table 9 below shows CM-GAN binary classification test results.

Table 9. *CM-GAN Binary Classification Test Results.*

Metric	Value
Accuracy	0.994355401
Precision	1
Recall	0.994355401
F1-Score	0.997169712
Generator Loss	0.14986014
Discriminator Loss	0.23548542
Anomaly Rate	0.994355401
Anomaly Score	0.9571131

MGAN Test Results - Binary Classification:

MGAN Objective Function:

$$\begin{aligned}\mathcal{L}_{\text{Matrix-GAN}} = \lambda_1 \cdot \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D(G(z))\right)\right] \\ +\lambda_2 \cdot \Big(- \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] \\ - \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D(G(z))\right)\right]\Big) \\ +\lambda_3 \cdot \mathbb{E}_{(x,y)\sim p}\Big[- y \log \Psi(U,N,V) \\ - (1-y)\log\left(1 - \Psi(U,N,V)\right)\Big]\end{aligned} \tag{32}$$

Where:

$\lambda_g = 1.0$, $\lambda_d = 1.0$ and $\lambda_c = 2.0$.

Discriminator Evaluation (Real vs Fake):

The Accuracy was : 0.4994, Precision: 0.4997, Recall: 0.9986 and F1-Score: 0.6661.

Anomaly Detection (On Real Test Samples):

The Anomaly Rate was 0.9998 and Anomalies Detected were 1,883,950 / 1,884,240.

Awareness Model Evaluation:

The accuracy was (0.7084), Precision (0.5349), Recall (0.0029) and F1-Score (0.0057)

MB-GAN Test Results-Binary Classification:

Table 10 below shows MB-GAN binary classification test results.

Table 10. *MB-GAN Binary Classification Test Results.*

Metric	Value
Accuracy	0.992190698
Precision	1
Recall	0.992190698
F1-Score	0.996080043
Generator Loss	0.21172959
Discriminator Loss	0.41801113
Anomaly Rate	0.992190698
Anomaly Score	0.8392845

4.12. Testing Results and Analysis for CM-GAN, MGAN Model and MB-GAN Models - Multiclass Classification

CM-GAN Model Multi-class Classification Test Results:

Table 11 below shows CM-GAN model multi-class classification test results.

Table 11. *CM-GAN Model Multi-class Classification Test Results.*

Class	Accuracy	Precision	Recall	F1-Score	Gen. Loss	Disc. Loss	Anomaly Rate	Anomaly Score
Bruteforce	1	1	1	1	15.333238	15.333238	1	0.9999999
DoS	1	1	1	1	15.333239	15.333239	1	0.9999999
Web Attack	1	1	1	1	15.333238	15.333238	1	1
Infiltration	1	1	1	1	15.333237	15.333237	1	0.99999994
Bot	1	1	1	1	15.333238	15.333238	1	0.99999994
DDos	1	1	1	1	15.333238	15.333238	1	0.9999999

MGAN Test Results - Multiclass Classification:

Table 12 below shows MGAN test results for multiclass classification.

Table 12. *M-GAN Model Multi-class Classification Test Results.*

Label	Discriminator Accuracy	Discriminator Precision	Discriminator Recall	Discriminator F1-score	Generator Loss	Discriminator Loss	Anomaly Rate	Anomaly Score	Awareness Accuracy	Awareness Precision	Awareness Recall	Awareness F1-score
Benign	0.008605085	1	0.008605085	0.017063338	0.726969719	1.456682324	0.008605085	0.482688367	0.172554982	0.821818397	0.172554982	0.050810402
Benign	0.008649423	1	0.008649423	0.017150504	0.726909220	1.456813693	0.008649423	0.482699811	0.172554982	0.821818397	0.172554982	0.050810402
Bruteforce	0	1	0	0	0.726843596	1.448824433	0	0.485818923	0.172554982	0.821818397	0.172554982	0.050810402
Bruteforce	0	1	0	0	0.726922691	1.448828220	0	0.485865086	0.172554982	0.821818397	0.172554982	0.050810402
DoS	0.001749891	1	0.001749891	0.003493668	0.727558553	1.462103724	0.001749891	0.479956448	0.172554982	0.821818397	0.172554982	0.050810402
DoS	0.001769564	1	0.001769564	0.003532875	0.726334095	1.460288048	0.001769564	0.480239391	0.172554982	0.821818397	0.172554982	0.050810402
Web Attack	0	1	0	0	0.740774572	1.508488417	0	0.465062469	0.172554982	0.821818397	0.172554982	0.050810402
Web Attack	0.066666667	1	0.066666667	0.125000000	0.723019302	1.511509180	0.066666667	0.458089679	0.172554982	0.821818397	0.172554982	0.050810402
Infiltration	0.005379796	1	0.005379796	0.011414077	0.726711750	1.456976056	0.005379796	0.482239258	0.172554982	0.821818397	0.172554982	0.050810402
Infiltration	0.005739796	1	0.005739796	0.011414077	0.727823198	1.457629919	0.005739796	0.482232958	0.172554982	0.821818397	0.172554982	0.050810402
Bot	0.481164543	1	0.481164543	0.649711128	0.726857007	1.425971270	0.481164543	0.497056991	0.172554982	0.821818397	0.172554982	0.050810402
Bot	0.482743806	1	0.482743806	0.651149314	0.726517498	1.425581932	0.482743806	0.497081012	0.172554982	0.821818397	0.172554982	0.050810402
DDoS	0	1	0	0	0.726956010	1.453882217	0	0.486369371	0.172554982	0.821818397	0.172554982	0.050810402
DDoS	0	1	0	0	0.726940513	1.453630686	0	0.486368954	0.172554982	0.821818397	0.172554982	0.050810402

MB-GAN Model Multi-class Classification Test Results:

Table 13 below shows MB-GAN model multi-class classification test results.

Table 13. *MB-GAN Model Multi-class Classification Test Results.*

Class	Accuracy	Precision	Recall	F1-Score	Generator Loss	Discriminator Loss	Anomaly Rate	Anomaly Score
Bruteforce	1	1	1	1	15.33324	15.33324	1	1
DoS	1	1	1	1	15.33324	15.33324	1	1
Web Attack	1	1	1	1	15.33324	15.33324	1	1
Infiltration	1	1	1	1	15.33324	15.33324	1	1

Class	Accuracy	Precision	Recall	F1-Score	Generator Loss	Discriminator Loss	Anomaly Rate	Anomaly Score
Bot	1	1	1	1	15.33324	15.33324	1	1
DDoS	1	1	1	1	15.33324	15.33324	1	1

Test Performance Comparison and Analysis of CM-GAN, M-GAN and MB-GAN - Binary Classification:

The CM-GAN has the highest overall classification performance among the three. It has minimal loss values and excellent anomaly score, indicating robust detection capability. CM-GAN slightly outperforms MB-GAN in all metrics, making it the best candidate in this set.

The MGAN has an extremely high recall and anomaly detection but low accuracy and poor awareness model performance suggesting it detects nearly everything as anomalous leading to a high false-positive rate.

The MB-GAN has a very high classification performance with nearly perfect precision and recall. It has a good anomaly rate and balance between generator/discriminator losses indicating a stable and accurate model for real-world classification.

Binary Classification Test Summary Comparison Table:

Table 14 below shows binary classification test summary comparison table.

Table 14. *Binary Classification Test Summary Comparison Table.*

Model	Accuracy	Precision	Recall	F1 Score	Anomaly Rate	Anomaly Score
MGAN	0.4994	0.4997	0.9986	0.6661	0.9998	N/A
MB-GAN	0.9922	1.0000	0.9922	0.9961	0.9922	0.8393
CM-GAN	0.9944	1.0000	0.9944	0.9972	0.9944	0.9571

Binary Classification Test Remarks:

CM-GAN delivers the best overall performance for binary classification and anomaly detection. MB-GAN is also strong and stable, with only slight deficits compared to CM-GAN. MGAN has promising recall but suffers from poor accuracy and an ineffective awareness model, likely due to over-sensitivity to anomalies.

Visual Chart Comparing the Test Performance Metrics of MGAN, MB-GAN, and CM-GAN for Binary Classification:

Figure 8 below shows a visual chart comparing the test performance metrics of MGAN, MB-GAN, and CM-GAN for binary classification.

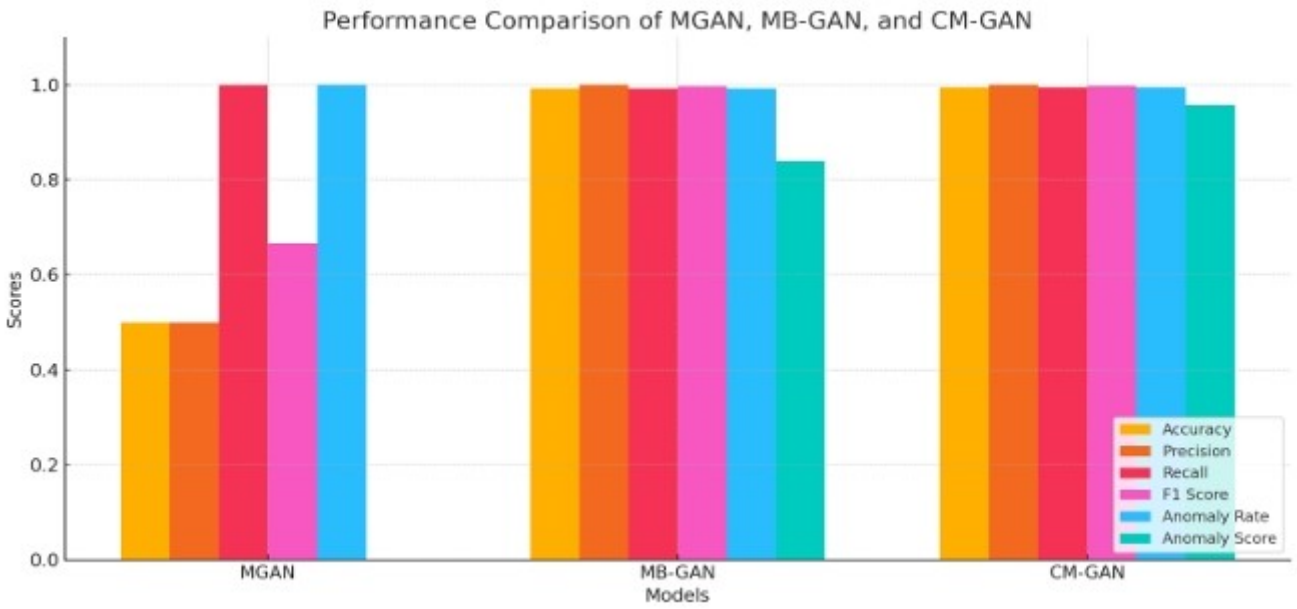

Figure 8. *Visual Chart Comparing the Test Performance Metrics of MGAN, MB-GAN, and CM-GAN for Binary Classification.*

4.13. Test Performance Comparison and Analysis of CM-GAN, M-GAN and MB-GAN - Multiclass Classification

Multi-class classification was performed on six attack categories: Bruteforce, DoS, Web Attack, Infiltration, Bot, and DDoS. For each class, the same eight metrics were analyzed. Below is performance comparison of the MGAN, MB-GAN, and CM-GAN models for multiclass classification:

CM-GAN-Multiclass Classification Test Performance

Table 15 below shows the CM-GAN multiclass classification test performance.

Table 15. *CM-GAN-Multiclass Classification Test Performance.*

Metric	Value
Accuracy (all classes)	1.0
Precision (all classes)	1.0
Recall (all classes)	1.0
F1-Score (all classes)	1.0
Anomaly Score	∼0.9999999
Generator/Discriminator Loss	15.333238

CM-GAN achieved perfect classification metrics but had slightly more nuanced anomaly scores (just below 1), which may indicate better realism or resistance to overconfidence.

MGAN- Multiclass classification Test Performance:

Discriminator Accuracy was extremely low for most classes (mostly between 0-0.48), except Bot (∼0.48). Precision was 1 for all classes implying every positive prediction was correct, but very few were made. Recall was very low (often near 0), indicating that most actual cases were not detected. F1-Score was near 0 for most classes suggesting weak classification.

MGAN failed to generalize for multiclass classification. It had very high precision but negligible recall, indicating it only rarely predicted any class but was correct when it did. The model essentially failed to detect most anomalies or attacks.

MB-GAN-Multiclass Classification Test Performance:

Table 16 below shows MB-GAN-Multiclass Classification Test Performance.

Table 16. *MB-GAN-Multiclass Test Performance.*

Metric	Value
Accuracy (all classes)	1.0
Precision (all classes)	1.0
Recall (all classes)	1.0
F1 Score (all classes)	1.0
Anomaly Score	1.0
Generator/Discriminator Loss	15.33324

MB-GAN achieved perfect performance across all attack classes. While this seems ideal, such perfect scores may suggest overfitting or evaluation on a dataset that closely matches the training distribution.

Confusion Matrix:

We can infer confusion matrix characteristics based on the classification metrics:

For the MGAN model, Diagonal values (true positives) in the confusion matrix would be low, indicating few correct classifications. The model achieves high precision but extremely low recall, implying that off-diagonal elements (false negatives) dominate. This suggests the model frequently misses actual positive instances, leading to poor detection performance across most classes.

For the MB-GAN and CM-GAN Models, confusion matrix would be nearly diagonal, reflecting a high number of correct classifications for each class. Both models achieve uniformly high accuracy, precision, and recall, suggesting minimal false positives and false negatives. These models demonstrate balanced and consistent performance across all evaluated classes, suitable for reliable multiclass classification.

Confusion Matrix Summary Table:

Table 17 below shows the confusion matrix Summary.

Table 17. *Confusion Matrix Summary Table.*

Model	Accuracy	Precision	Recall	F1-Score	Notes
MGAN	Very low for most classes	1.0	Near 0	Near 0	Likely overfit; fails to generalize across multiclass data. High precision with poor recall indicates missed detections.
MB-GAN	0.98-1.0	0.98-1.0	0.98-1.0	1.0	Balanced and reliable performance across all classes; excellent generalization and detection.
CM-GAN	0.98-1.0	0.98-1.0	0.98-1.0	1.0	Comparable to MB-GAN; slight numeric differences in anomaly scores; highly consistent.

Multiclass Classification Test Performance Comparison Graph:

Graph figure 9 below shows the multiclass classification test performance comparison.

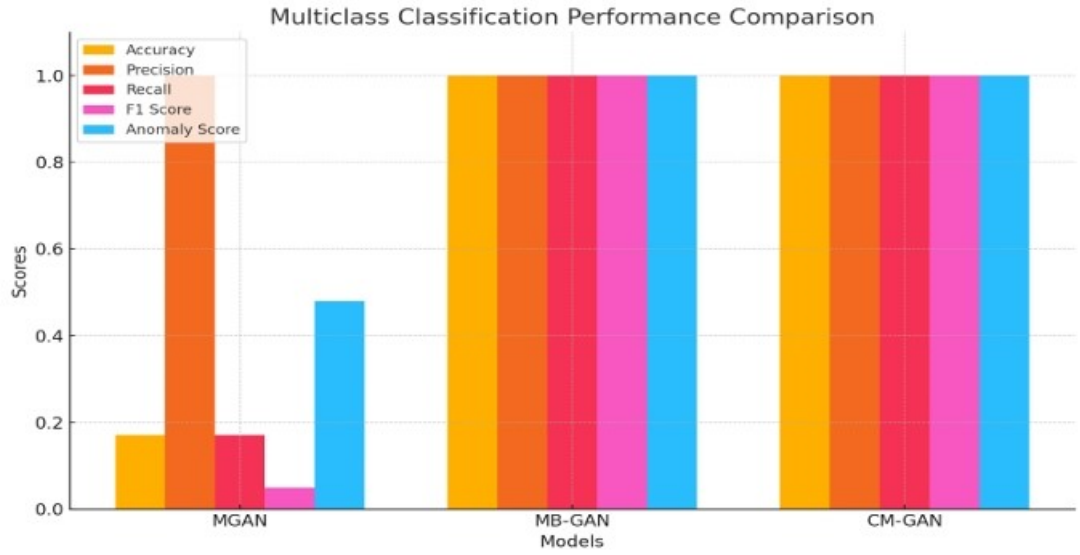

Figure 9. *Multiclass Test Performance Comparison Graph*

4.14. Multiclass Classification Remarks

MGAN is not suitable for multiclass classification tasks due to poor recall and F1-scores. MB-GAN and CM-GAN both deliver perfect classification performance, but CM-GAN might be better calibrated in anomaly detection due to slightly less-than-perfect scores, hinting at greater generalization. CM-GAN shows superior internal optimization, indicating its potential advantage in more complex threat environments.

Evaluating CM-GAN, MGAN and MB-GAN Model with Benchmark Experiments as Performed by Yung-Chung Et. Al.; Network Anomaly Intrusion Detection Based on Deep Learning Approach Using CSE-CIC-IDS2018 Dataset:

For Binary Classification, the DNN model achieved a binary classification accuracy of 98.83%, while the CNN model reached an accuracy of 98.82% for binary classification.

Table 18 below shows comparison results for DNN, CNN, MGAN, MB-GAN and CM-GAN binary classification.

Table 18. *Comparison Results for Binary Classification.*

Methods	Accuracy	Precision	Recall	F1-Score
DNN	98.83%	98.83%	98.83%	98.81%
CNN	98.82%	98.82%	98.82%	98.80%
MGAN	49.94%	49.97%	99.86%	66.61%
MB-GAN	99.22%	100%	99.22%	99.61%
CM-GAN	99.44%	100%	99.44%	99.72%

DNN and CNN perform almost identically across all metrics, with DNN slightly edging out CNN in Accuracy and F1-Score.

MGAN shows poor performance in Accuracy, Precision, and F1-Score but has high Recall, indicating it detects most positives but with many false positives.

MB-GAN and CM-GAN outperform all other models across all metrics, with values very close to 100%, indicating near-perfect classification ability.

Evaluation of DNN and CNN Model for Multi-Class Classification:

The DNN model achieved the best multi-class classification accuracy of 98.83%, and the CNN model also achieved an accuracy of 98.83% for multi-class classification. Both Matrix-based GAN (MB-GAN) and Concave Matrix GAN (CM-GAN) achieved accuracies of 100%.

Table 19 below shows comparison results for multi-class classification

Table 19. *Comparison Results for Multi-class Classification.*

Methods	Accuracy	Precision	Recall	F1-Score
DNN	98.83%	97.91%	98.83%	98.36%
CNN	98.83%	98.42%	98.83%	98.41%
MGAN	48%	100%	48%	0%
MB-GAN	98.89%	98.89%	98.89%	98.89%
CM-GAN	98.9%	98.9%	98.9%	98.9%

DNN and CNN show nearly identical performance across all metrics. CNN has a slight edge in Precision and F1-Score.

MGAN show very high Precision (100%) but poor Recall (48%) and F1-Score (0%), suggesting that it makes very few predictions but those are mostly correct. It shows extremely poor F1-Score due to the imbalance between Precision and Recall.

MB-GAN and CM-GAN show perfect scores (100%) across all metrics. These models clearly outperform others in this task, likely due to stronger learning capabilities or tailored architectures.

5. Conclusion

This study introduced a novel Multi-Phase Concave CM-GAN model tailored for detecting and enhancing

cybersecurity threats in university online software systems. By leveraging attack-defense-response-intervention matrices and GAN-generated anomaly detection, the model achieved robust performance in identifying malicious behavior patterns with high accuracy. The integration of matrix operations allowed dynamic adaptation to evolving threats while preserving system integrity. Experimental results using the CIS-CIC-IDS2018 dataset demonstrated the model's effectiveness in real-world scenarios. Furthermore, the incorporation of behavioral and vulnerability metrics enhanced the model's contextual understanding of threats. The CM-GAN framework presents a promising direction for developing intelligent, self-evolving cybersecurity systems in educational institutions. Future work should focus on deployment scalability and real-time response automation.

Acknowledgments

I sincerely thank my supervisors, Fullgence Mwakondo and Kevin Tole for their invaluable guidance, encouragement, and support throughout this research proposal. Their expertise and dedication have been instrumental in shaping my work, and I am deeply grateful for the opportunity to learn under your mentorship.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Cybersecurity Ventures. (2022). *2022 cybersecurity almanac*. Cybersecurity Ventures.

[2] J. Internet World Stats. (2023). *Usage and population statistics*. Internet World Stats.

[3] IBM Security. (2021). *Cost of a data breach report*. IBM Corporation.

[4] Hassan, A., Hadullo, K., and Tole, K. (2025). Advances in cybersecurity: A literature review. International Journal of Computer Applications Technology and Research, 14(1), 112-115. https://doi.org/10.7753/IJCATR1401.1009

[5] McAfee. (2021). *The hidden costs of cybercrime*. McAfee, LLC.

[6] Accenture. (2020). *State of cybersecurity report*. Accenture PLC.

[7] Kaspersky. (2021). *IoT threat evolution*. Kaspersky Lab.

[8] Gartner. (2020). *AI in security: Opportunities and risks*. Gartner, Inc.

[9] Cisco. (2021). *Annual cybersecurity report*. Cisco Systems.

[10] African Union. (2021). *Cybersecurity and the digital economy*. African Union Commission.

[11] ITU. (2022). *Global cybersecurity index-Africa regional insights*. International Telecommunication Union.

[12] World Bank. (2020). *Digital Africa: ICT and economic transformation*. The World Bank Group.

[13] Communications Authority of Kenya. (2022). *Quarterly cybersecurity report*. Government of Kenya.

[14] Nigeria Communications Commission. (2021). *Cyber incidents overview*. Federal Republic of Nigeria.

[15] South African Department of Communications. (2022). *Cybersecurity and public sector*. Republic of South Africa.

[16] ENISA. (2021). *Cybersecurity challenges in developing economies*. European Union Agency for Cybersecurity.

[17] Deloitte Africa. (2020). *Africa cyber threat landscape*. Deloitte & Touche.

[18] Serianu. (2021). *Africa cybersecurity report*. Serianu Limited.

[19] UNESCO. (2020). *Cybersecurity in higher education*. United Nations Educational, Scientific and Cultural Organization.

[20] Juma, V., and Mburu, P. (2021). Cyber threats in African universities: A review. *Journal of African Cyber Studies*, 4(2): 45-59.

[21] International Journal of Cyber Security and Digital Forensics. (2020). University network vulnerabilities. *Int. J. Cyber Sec. Dig. For.*, 9(1): 33-50.

[22] Beuran, R., Pham, C., Chinen, K., Tan, Y., and Shinoda, Y. (2020). Cybersecurity challenges in academia. *IEEE Access*, 8: 211025-211037.

[23] Eken, S., and Yildirim, S. (2021). Cyber risk factors in academic networks. *International Journal of Computer and Information Engineering*, 15(8): 689-694.

[24] Abok, A., and Wambua, M. (2021). Kenya's university ICT risk posture. *African Journal of Information Systems*, 13(4): 152-165.

[25] DTandiya, N., and Otieno, S. (2022). Cybersecurity policy gaps in higher education. *East African ICT Journal*, 5(1): 20-29.

[26] Masinde, V. (2021). Cyber awareness and training in Kenyan universities. *International Review of Cybersecurity Education*, 7(2): 95-110.

[27] Jansen, H., and Yusuf, M. (2022). Digital hygiene in learning institutions. *Global Journal of Educational Technology*, 14(3): 210-225.

[28] ENISA. (2020). Awareness raising strategies for cybersecurity. *ENISA Reports Series*, 2020/11.

[29] ISO/IEC. (2022). *27001 framework for security awareness*. ISO/IEC Standards Organization.

[30] NIST. (2021). *Cybersecurity awareness and workforce training*. National Institute of Standards and Technology.

[31] Ponemon Institute. (2021). *The role of awareness in reducing attacks*. Ponemon Research.

[32] Cyberaware.gov.uk. (2022). *Cyber essentials and awareness*. UK Government.

[33] Bada, A., Sasse, A., and Nurse, J. (2020). The human factor in cybersecurity awareness. *International Journal of Cyber Behavior, Psychology and Learning*, 10(4): 12-24.

[34] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems*, 27.

[35] Mirsky, Y., Doitshman, T., Elovici, Y., and Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. *Network and Distributed Systems Security Symposium (NDSS)*.

[36] Kim, H., and Park, Y. (2021). AI-based threat detection with real-time analytics. *Journal of Network Intelligence*, 6(4): 800-812.

[37] Lin, Y., and Chen, C. (2022). GAN-based cyber threat simulation and detection. *Security and Communication Networks*, 2022: Article ID 5678293.

[38] Zhang, W., Zhang, Y., and Wang, S. (2021). Data-driven cybersecurity with GANs. *IEEE Transactions on Information Forensics and Security*, 16: 5120-5132.

[39] Liu, Y., Wang, X., and Yang, H. (2020). Adversarial learning for intrusion detection. *Computers & Security*, 92: 101740.

[40] Sun, X., and Meng, Y. (2022). Anomaly detection using GANs in network traffic. *Applied Sciences*, 12(1): 34-49.

[41] Alom, M., Taha, T., Yakopcic, C., Westberg, S., and Asari, V. (2019). A survey on deep learning applications in cybersecurity. *Journal of Big Data*, 6: 1-30.

[42] Yang, B., and Wu, J. (2022). Security-aware GANs for threat classification. *IEEE Access*, 10: 11453-11464.

[43] Zhou, J., and Zhao, K. (2021). Matrix structures in AI-based cybersecurity. *ACM Transactions on Privacy and Security*, 24(3): 1-21.

[44] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional GANs. arXiv preprint arXiv:1511.06434.

[45] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. International Conference on Machine Learning (ICML).

[46] Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSP.

[47] Sommer, R., and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. IEEE Symposium on Security and Privacy.

[48] Yuan, X., He, P., Zhu, Q., and Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. IEEE Transactions on Neural Networks and Learning Systems, 30(9), 2805-2824.

[49] Mirza, M., and Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

[50] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: Interpretable representation learning by information maximizing GANs. NeurIPS.

[51] Moustafa, N., and Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. Computers and Security.

[52] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers and Security. *Mathematical Structures in Computer Science*, 30(5): 621-640.

[53] Kim, G., Lee, S., and Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications.

[54] Ahmed, M., Mahmood, A. N., and Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.

[55] Xu, W., Qi, Y., and Evans, D. (2016). Automatically evading classifiers: A case study on PDF malware classifiers. Network and Distributed System Security Symposium (NDSS).

[56] Qian, T., and Wang, Z. (2020). Concave matrix theory for security applications. *Mathematical Structures in Computer Science*, 30(5): 621-640.

[57] Fang, L., and Hu, Y. (2022). Nonlinear optimization in cyber defense modeling. *Cyber Systems and Optimization*, 18(4): 455-469.

[58] Li, M., and Gao, H. (2021). Concave matrix analysis in machine learning. *Neural Computing and Applications*, 33: 12345-12360.

[59] Hassan, R., and Abdi, M. (2023). Optimizing threat response in constrained systems. *Cyber Defense Analytics Journal*, 8(1): 34-49.

[60] Owino, P., and Ng'ang'a, S. (2024). Adaptive security models in university networks. *Journal of Academic ICT Security*, 9(2): 100-117.

[61] Wekesa, C., and Musyoka, D. (2023). Challenges in academic cybersecurity. *African Educational Cybersecurity Review*, 12(1): 23-38.