TIPSO-GAN: Malicious Network Traffic Detection Using a Novel Optimized Generative Adversarial Network

Ernest Akpaku[†*], Jinfu Chen[†*✉], Joshua Ofoeda[‡]
[†]School of Computer Science and Communication Engineering, Jiangsu University
Jiangsu Key Laboratory of Security Technology for Industrial Cyberspace,
301 Xuefu Road, Zhenjiang 212013, Jiangsu, China
[‡]Department of Information Technology, University of Professional Studies, Accra, Ghana
ernestakpaku1@gmail.com, jinfuchen@ujs.edu.cn, joshua.ofoeda@upsamail.edu.gh

Abstract*—Detecting advanced cyber threats, particularly zero-day vulnerabilities, poses significant challenges in network security. This paper presents TIPSO-GAN, an optimized Generative Adversarial Network (GAN) for detecting malicious traffic. TIPSO-GAN addresses common GAN-based intrusion detection system (IDS) issues, such as training instability and mode collapse, by framing GAN training as a swarm optimization problem, harnessing collective intelligence for complex optimization. To enhance Particle Swarm Optimization (PSO), TIPSO-GAN employs three strategies: (1) adaptive inertia weights for a balance of exploration and exploitation, (2) a diversity preservation strategy to prevent premature convergence, and (3) a feedback loop to reinitialize stagnant particles. TIPSO-GAN integrates transfer learning with a Temporal-Decaying Multi-Head Self-Attention mechanism to prioritize recent features, aiding in unseen malicious traffic detection. A combination of reconstruction loss and focal loss in the objective function further ensures realistic normal samples while focusing on challenging malicious samples. Across CIC-IDS2018, CICAPT-IIoT2024, and CIC-DDoS2019, TIPSO-GAN achieves 99.1±0.1, 98.9±0.1, and 98.7±0.1 F1, outperforming the strongest baseline by 0.2–1.0 F1 and exceeding transformer IDS models. On CICAPT-IIoT2024, it reaches 0.999±0.002 macro PR-AUC, ahead of the next best method (0.960±0.005). Under strict zero-day evaluations, TIPSO-GAN attains 92.3 F1 in LOFO tests and 79–83 F1 in cross-dataset experiments while maintaining recall above 0.80. Despite PSO-enhanced training, TIPSO-GAN maintains 0.42 ms latency, ∼2400 flows/s throughput, and a 2.1 GB footprint, with stable performance up to 10^8 flows. Our code is accessible at https://doi.org/10.5281/zenodo.17759516.

I. Introduction

The adoption of internet-based technologies such as cloud computing, information systems, web servers, and IoTs has proliferated in recent years. This heightened reliance on technology in our daily routines has rendered us more susceptible to unknown cyberattacks. Identifying previously unknown vulnerabilities in networks is challenging as they lack established signatures and exhibit novel behaviours not yet recognized by traditional security systems [1]. Widely used deep learning methods such as LSTM, CNN, DNN, and RNN traffic identification methods are broadly trained with labelled traffic data. This limits their ability to learn general features from unlabelled data. As such, they still suffer from low accuracy, particularly when challenged with unknown malicious traffic samples [2]. Recent transformer-based architectures, such as FlowTransformer [3], FT-Transformer [4], MalDetectFormer [5], and TransTraffic [6], have shown strong performance in modeling long-range dependencies in network flows. However, these methods are primarily optimized for static accuracy and rely on large labeled datasets. Generative adversarial networks (GANs) developed by Goodfellow et al. [7] have proven effective for creating models that can mimic previously unknown behaviours in real-world situations [8] [9]. GAN is made up of a generator which is a neural network that generates new data samples from a random noise input; and a discriminator that aims to distinguish between real data samples and the fake samples produced by the generator. The two networks are trained simultaneously using the MiniMax objective function which is based on the principles of the game theory. Compared with other existing generative models, GANs provide a concise and efficient framework for learning generative models.

Although GANs have been successful in image generation and manipulation [10], game development, text-to-image generation, medical imaging, and many other tasks [11], they suffer from mode collapse resulting in training instability. The training aims to reach a Nash equilibrium where neither the generator nor the discriminator can improve their strategy given the other's strategy. However, finding this equilibrium in practice is challenging due to the non-convex and high-dimensional optimization landscape, leading to unstable training and mode collapse [12]. Optimizing the network's weights to minimize the loss function is challenging due to the non-convex nature of the loss landscape in GANs. This makes it difficult to find the optimal set of weights that results in high-quality, diverse generated samples.

Many recent efforts on GANs have focused on overcoming

* These authors contributed equally and should be regarded as co-first authors.
✉ Corresponding author: Jinfu Chen.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.243241
www.ndss-symposium.org

these optimization difficulties by developing various adversarial training objectives [13]. The gradients of the objective function of the network parameters guide the training. However, training GANs with the original Jensen–Shannon divergence (JSD) leads to vanishing gradients leading to unstable training dynamics [14]. To address these issues, researchers have introduced alternative adversarial training objectives such as the least squares [15], absolute deviation, energy-based, spectral normalization [14], Kullback–Leibler (KL) divergence [16], Wasserstein Distance, and the Wasserstein Distance with a penalty term [17]. However, these objectives have not fully addressed the training and mode collapse challenges completely.

In this study, we propose a novel GAN framework named TIPSO-GAN, leveraging the improved architectural design of GAN and enhanced Particle Swarm Optimization (PSO) to enhance the training dynamics. Unlike conventional GANs that engage in fixed adversarial competition, our PSO-optimized GAN harnesses the collective intelligence of a swarm to navigate the complex optimization landscape. By treating the generator's parameters as particles within the swarm, we evolve the generator through iterative updates guided by both individual and social experiences within the swarm, effectively transforming the adversarial training into a PSO-driven optimization challenge. In our implementation, we note that the fast convergence speed of PSO can quickly lead GANs to find a good set of parameters, which can result in stable and efficient training, especially in the initial phases of training. However, this speed could limit the algorithm from exploring the entire solution space sufficiently, which could be crucial for GANs that require a thorough search to avoid getting stuck in suboptimal configurations. Also, despite its global search capability, the standard PSO might not be as effective in fine-tuning the parameters near the end of the optimization process, potentially leaving room for improvement in the generator's performance.

Linearly decreasing the inertia weight and improving the weight factor have been proposed to address these problems in clustering algorithms. However, it may not be suitable for complex problems where the optimization surface has many local optima or varying degrees of ruggedness. Also, it uses a fixed predetermined schedule for reducing inertia weights which lacks flexibility to adapt to different stages of the optimization process or diverse problem landscapes. For example, in a linearly decreasing inertia weight schedule, the inertia weight starts at an initial value and decreases uniformly to a final value by the end of the optimization process. To address these challenges, we enhanced the standard PSO algorithm to incorporate adaptive inertia weights, diversity preservation strategies, and a feedback loop mechanism that encourages exploration in the later stages of optimization. These modifications ensure PSO-driven GAN optimization to avoid local optima and achieve more robust and diverse generator parameters.

In this paper, several novel strategies are proposed to address the challenges discussed for better malicious network traffic detection. First, an adaptive Sigmoidal inertia weight adjustment strategy is introduced into the PSO algorithm and compared with the popular linearly decreasing approach, and others. The best performing is used to effectively explore GAN's parameter search space to find an optimal solution. Second, the traditional GAN is modified through novel architectural design a new loss reconstruction function, and PSO-based training. Last, a pretraining model named DeePred is introduced and applied to improve the performance of an unknown malicious traffic detector. Leveraging, on the strengths of the proposed methods, this study makes the following contributions:

1) We propose TIPSO-GAN, a framework that frames GAN training as a swarm optimization problem, leveraging an enhanced PSO algorithm in a two-staged adversarial training. We introduce adaptive inertia weights, a diversity preservation mechanism, and reinitialization of stagnant particles to enhance the PSO to dynamically balance exploration and exploitation, prevent premature convergence, and avoid stagnation in the optimization process, thereby enhancing the training stability and convergence of TIPSO-GAN. These enhancements ensure better exploration and exploitation of the search space during training, enabling better convergence speed, more stable training, and effective mitigation of mode collapse.
2) A novel reconstruction loss function and a pre-training network DeePred, are introduced to improve the adversarial traffic generation ability of the generator and the detection ability of the discriminator in TIPSO-GAN. DeePred is integrated with a novel Multihead Self Attention Mechanism (MHSA) that utilizes a temporal decaying parameter to provide larger weights to features that are more recent and are indicative of malicious behaviour. A focal loss is introduced into the discriminator to address class imbalance by focusing on hard-to-classify malicious samples.
3) Extensive experiments on three datasets demonstrate that TIPSO-GAN outperforms conventional GAN and its variants, state-of-the-art GAN-based-IDS variants and four other existing deep learning IDS models in terms of accuracy, precision, recall, and F1-score. TIPSO-GAN also exhibits improved training stability and resistance to mode collapse and class imbalance, making it highly suitable for real-world network intrusion detection systems.

The remaining sections of the study are organized as follows: Section II discusses related studies. Section III presents detailed descriptions of the proposed method. In Sections V and VI, we describe the experimental setup and evaluation, and results respectfully. The conclusion and future research are discussed in section VII.

II. Related Studies

A. Generative Adversarial Networks in Cybersecurity

In recent years, GANs have been implemented for network security, particularly for the detection of malicious traffic. The

ability of GANs to generate realistic synthetic data has been leveraged to enhance the performance of intrusion detection systems (IDS) by simulating a wide range of potential attacks [13]. One of the pioneering works in this area is the paper by Das [18], which demonstrates the application of GANs in reproducing both categorical and continuous synthetic network traffic features. Building upon this foundation, researchers have explored the integration of GANs with deep neural networks for more effective anomaly detection in network traffic. The work by Liu and Liu [19] presents a federated learning approach using GANs, which allows for localized training and centralized model updates, addressing concerns of privacy and model tampering in large-scale networks. Further advancements have been made in the detection of adversarial traffic flows by employing GAN-based attack algorithms like NIDSGAN [20], which develops a GAN-based attack algorithm capable of generating highly realistic adversarial traffic flows that can evade machine learning-based NIDS. Recent work has also explored conditional GANs for generating specific attack types and advanced architectures for improving synthetic traffic quality.

However, these approaches predominantly focus on generating synthetic data to augment training datasets rather than directly addressing the challenge of detecting previously unseen attack patterns. For instance, Das [18] generates synthetic network flows but requires pre-existing attack labels, while NIDSGAN [20] creates adversarial examples primarily for testing existing detectors rather than improving zero-day detection capabilities. Moreover, existing GAN-based IDS methods struggle with the unique characteristics of network traffic data. Unlike image generation where GANs excel, network traffic exhibits complex temporal correlations, multi-modal feature distributions, and extreme class imbalance between normal and malicious samples. Liu and Liu [19] reported that their federated GAN approach achieved only 78% detection accuracy on unknown attack variants, significantly lower than performance on known attacks. Similarly, recent studies show that conventional GAN architectures fail to preserve the statistical properties of network protocols when generating synthetic traffic, leading to unrealistic packet sequences that can be easily distinguished by modern firewalls. Despite these promising initial results, current GAN applications in network security remain limited by training instability and inability to generalize beyond their training distributions, constraining their effectiveness in dynamic threat environments.

B. GAN Training Optimization and Stability

As noted in [21], GANs are known to suffer from training instability, which can lead to issues such as mode collapse and training oscillations. These challenges can affect the quality and diversity of the generated samples, which are crucial for accurately simulating and detecting a wide range of network attacks. Another limitation highlighted in the literature is the complexity of the generator network in GANs [13]. When the generator's probability distribution is highly complex, it becomes difficult to estimate the maximum likelihood [22]. GANs address this by using neural networks to define the probability distribution [23], which introduces its own set of challenges, such as the need for careful architecture design and tuning to ensure that the generated data accurately represents the underlying distribution of normal network traffic. DCGAN [24] introduced architectural constraints that improved the stability of GAN training by replacing pooling layers with strided convolutions and fully connected layers with global average pooling in the discriminator to reduce checkerboard artifacts and simplify the architecture. However, these modifications did not eliminate the issues of training instability and mode collapse. Subsequent research focused on reconstructing the loss function to address these training problems. The Wasserstein GAN (WGAN) [14] introduced a new loss function based on the Wasserstein distance, which provides a more stable training objective. An extended WGAN called WGAN-GP [17] added a gradient penalty term to the loss function, though this introduced additional computational overhead. The LSGAN [15] proposed using least squares loss, but suffered from slower convergence rates on complex datasets. Beyond loss function modifications, advanced architectural innovations have attempted to address training instability. Techniques such as spectral normalization and progressive training have shown improvements in specific domains, yet these methods primarily target image generation tasks. When applied to network traffic data, these stabilization techniques show limited effectiveness due to the high-dimensional, non-stationary nature of network features. Recent empirical studies report that even with these advanced techniques, GANs applied to cybersecurity datasets still experience mode collapse rates of 20-35%, particularly when attempting to generate minority class samples representing sophisticated attacks [13]. The fundamental challenge remains that existing stabilization methods rely on assumptions about data structure and training dynamics that do not hold for the adversarial, multi-modal distributions characteristic of network security data.

C. Swarm Intelligence for Neural Network Optimization

In recent times, swarm intelligence algorithms [25] such as PSOs [26] have achieved considerable success across a wide range of computational tasks, including modelling, optimization, and design of GANs [27]. PSO operates by maintaining a population of particles, each representing a potential solution in the search space, where particles update their velocities and positions based on their personal best experiences and the global best solution found by the swarm. This population-based approach enables PSO to explore multiple regions of the search space simultaneously, making it particularly effective for complex optimization problems with multiple local optima. Recent applications of PSO in neural network optimization have demonstrated its effectiveness in avoiding local minima through population-based search strategies [28]. Unlike gradient-based methods that rely on local derivative information, PSO maintains diversity through swarm dynamics and can escape suboptimal regions by leveraging collective intelligence. In the context of GAN training, PSO offers the-

oretical advantages over traditional Adam or SGD optimizers by treating the generator parameters as particle positions in the high-dimensional parameter space, enabling global search capabilities that are particularly valuable given the non-convex, multi-modal nature of GAN loss landscapes [29]. However, conventional PSO suffers from premature convergence and insufficient exploration in high-dimensional parameter spaces typical of GAN architectures. The standard PSO algorithm can become trapped in local optima when the swarm loses diversity, particularly in complex optimization landscapes with numerous suboptimal peaks. Variants such as linearly decreasing inertia weight PSO [28] have been proposed to balance exploration and exploitation phases by gradually reducing the influence of previous velocities over time. However, this linear scheduling approach lacks the flexibility to adapt to different stages of the optimization process or diverse problem landscapes, making it less effective in complex problem scenarios where the optimization surface has many local optima or varying degrees of ruggedness. The challenge becomes particularly acute in GAN optimization, where the objective function changes continuously due to the adversarial game between generator and discriminator. The fitness landscape exhibits non-stationarity as discriminator updates alter the generator's optimization surface, requiring PSO variants that can dynamically adjust exploration-exploitation balance based on real-time feedback. Traditional PSO parameters such as fixed inertia weights and acceleration coefficients prove inadequate in this environment, as they cannot respond to the evolving loss topology that characterizes adversarial training dynamics [28]. These problems could critically compromise PSO's efficacy in high-dimensional, non-convex optimization problems characterized by extensive local minima networks, thereby potentially leading to suboptimal convergence trajectories and diminished algorithmic convergence velocities [30].

D. Novelty of this study

As summarized in Table I, although numerous GAN-based intrusion detection approaches exist, they seldom address key challenges comprehensively. Despite recent advances [31], [32], most still suffer from training instability and high mode collapse when applied to heterogeneous network traffic data. Stabilization techniques such as spectral normalization and progressive training, designed for vision tasks, remain ineffective for sparse cybersecurity datasets. Furthermore, the use of PSO in GANs for intrusion detection is largely unexplored. While PSO-GAN has shown promise in other domains [33], [34], limited works [35] lack adaptive inertia weighting and diversity-preserving mechanisms essential for robust adversarial learning. We also note that transformer-based IDS models (TranAD, TransfficFormer, FlowTransformer) improve stability but do not incorporate PSO-guided GAN training or zero-day–oriented transfer learning.

Another gap is the absence of transfer learning. Existing GAN-based IDSs are typically trained from scratch, neglecting pre-trained models that enhance convergence and generalization. Although hybrid approaches such as [36] employ zero-shot learning, they fail to integrate transfer learning for both data synthesis and knowledge reuse. Few studies unify known and evolving attack detection in a coordinated framework; even recent multi-level or hybrid models [36], [37] omit PSO optimization and transfer learning within a single training pipeline.

TABLE I: Selected Related Studies. Training Stability (TS), Mode Collapse (MC), Class Imbalance (CI), Zero-Day Detection (ZD)

Study	Year	Method	TS	MC	CI	ZD	TL	PSO	MPT
Das [18]	2022	FGAN	X	X	X	✓	X	X	X
Zolbayar et al. [20]	2022	NIDSGAN	X	X	✓	X	X	X	X
Nguyen et al. [31]	2023	GAN-SCADA	✓	✓	✓	X	X	X	X
Seo et al. [38]	2019	GIDS	✓	✓	✓	✓	X	X	X
Rahman et al. [39]	2024	SYN-GAN	X	X	✓	X	X	X	X
Aldhaheri & Alhuzali [40]	2023	SGAN-IDS	X	X	X	✓	X	X	X
Torres et al. [32]	2023	GAN-AE	✓	✓	✓	✓	X	X	X
Poongodi & Hamdi [37]	2023	Multi-GAN-IoV	✓	✓	✓	X	X	X	✓
Mari et al. [41]	2023	GAN-Adversarial	✓	X	X	✓	X	X	X
Zhang & Zhao [33]	2021	PSO-GAN	✓	✓	X	X	X	✓	X
Liu et al. [34]	2022	Parameterized PSO-GAN	✓	✓	X	X	X	✓	X
Sundaram et al. [35]	2023	PSO-GRU-GAN-5G	✓	✓	✓	✓	X	✓	X
Roopak et al. [42]	2024	Unsupervised Ensemble	✓	X	✓	✓	X	X	X
Sarhan et al. [43]	2023	Zero-Shot ML	X	X	✓	✓	X	X	X
Touré et al. [36]	2024	Hybrid Learning	✓	X	✓	✓	X	X	✓
Huang et al. [44]	2021	T-GAN	✓	X	✓	X	X	X	X
Tuli et al. [45]	2022	TranAD (Transformer)	✓	✓	✓	X	X	X	X
Du et al. [46]	2025	TransfficFormer	✓	✓	✓	X	X	X	X
Manocchio et al. [3]	2024	FlowTransformer	✓	✓	✓	✓	X	X	X
TIPSO-GAN (Proposed)	2025	Enhanced PSO-GAN	✓	✓	✓	✓	✓	✓	✓

III. Proposed Method

A. The TIPSO-GAN Framework

To detect unknown network attacks with high accuracy, we propose a two-stage adversarial-based intrusion detection model named TIPSO-GAN, as illustrated in Figure 1. The core of the framework is TIPSO-GAN, which combines the benefits of PSOGAN and the pre-training model (DeePred) to enhance the detection of both known and unknown attacks. In this framework, the discriminator of PSOGAN and the pre-training model DeePred are crucial for transferring knowledge to the final TIPSO-GAN. The Generator G_T in TIPSO-GAN is initialized by transferring the structure and parameters from the Generator G of PSOGAN. PSOGAN is trained exclusively on normal traffic data. During training, the generator G learns to produce modelled samples that approximate the normal traffic distribution. Once the PSOGAN has completed its training, its discriminator D is discarded, and the trained generator G is transferred to initialize the Generator G_T in TIPSO-GAN. This ensures that the Generator G_T starts with a good understanding of normal traffic, enabling it to generate more accurate representations of normal traffic during training.

To strengthen the detection capability of the Discriminator D_T in TIPSO-GAN, we transfer the structure and parameters of the pre-training model DeePred into Discriminator D_T. The DeePred model has a structure similar to the discriminator in PSOGAN but is trained using a dataset containing both normal and known malicious traffic. This gives DeePred the ability to effectively distinguish between normal and malicious samples (i.e., 2-class classification). By transferring the trained parameters from DeePred, the Discriminator D_T of TIPSO-GAN inherits a robust classification capability, enhancing its ability to detect unknown malicious traffic. After the initial transfer, TIPSO-GAN undergoes unsupervised training, using

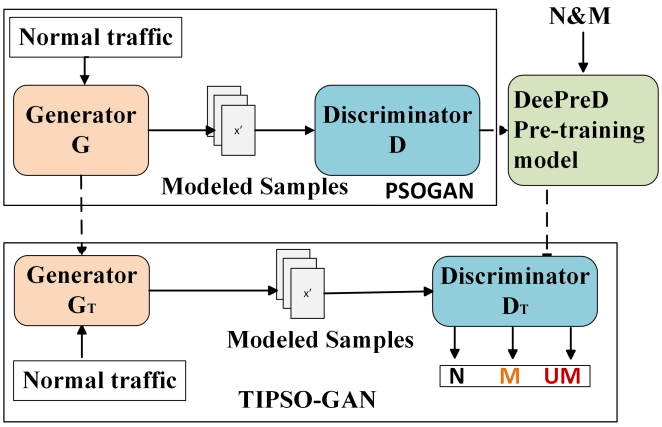


Fig. 1: **Detection roadmap of TIPSO-GAN.**

Note. N: Normal traffic, M: Known Malicious traffic, UM: Unknown Malicious traffic.

Algorithm 1 Algorithm of the Proposed TIPSO-GAN Model

Initialize: DN: normal traffic; DE: train set; VE: val set; TB: test set; B: batch size; $Epochs$
G: generator trained from PSOGAN using DN
$DeePred$: discriminator pre-trained with DE, VE
Initialize $GT \leftarrow G$, $DT \leftarrow DeePred$
Training:
for each epoch in $Epochs$ **do**
    Generate fake samples with GT (size B)
    Sample real flows from DN (size B)
    Train discriminator DT on both batches
    if validation accuracy drops for n=3 rounds **then**
        Output DT from round n+1 and break
    else if accuracy on normal $< T$ **then**
        Output DT from round $n-1$ and break
    end if
    Train generator GT
end for
Output: final discriminator DT as classifier C

a dataset consisting only of normal traffic samples. The goal is for the Generator G_T to continue learning the normal traffic distribution while the Discriminator D_T refines its ability to identify traffic samples that deviate from the normal (i.e., potential unknown attacks). A validation set containing both normal and unknown malicious samples is used to monitor the performance of TIPSO-GAN during training. The adversarial game between the generator and discriminator continues until the detection accuracy of normal traffic reaches a certain threshold or the overall accuracy starts to decline. Once training is complete, the Discriminator D_T of TIPSO-GAN is output as the final anomaly detector.

B. Modified PSO for GAN Optimization

We propose an improved PSO variant that integrates an adaptive inertia weight and a diversity preservation mechanism. Our enhanced PSO algorithm introduces an adaptive inertia weight w that dynamically adjusts based on the swarm's performance. Unlike traditional PSO algorithms that use a fixed or linearly decreasing inertia weight, our method adapts the inertia weight non-linearly based on real-time feedback, providing greater flexibility in balancing exploration and exploitation. In the early stages of optimization, a higher inertia weight is used to facilitate global exploration, enabling particles to traverse the search space extensively. As the algorithm progresses, the inertia weight decreases, honing the particles' search towards local refinement and convergence to optimal solutions. Algorithm 2 shows the working principle of the proposed PSO. The adaptive inertia weight is updated using a non-linear function, governed by a sigmoid function, to provide rapid adaptation when significant improvements are detected and a gradual approach when progress plateaus. The inertia weight $w(t)$ at iteration t is calculated as:

$$w(t) = w_{\min} + (w_{\max} - w_{\min}) \times \frac{1}{1 + e^{-k(\Delta f(t) - f_{\text{threshold}})}} \quad (1)$$

where: $w_{\max}$ and $w_{\min}$ are the maximum and minimum inertia weights, respectively. $\Delta f(t)$ represents the change in the global best fitness over a certain number of iterations, defined as $\Delta f(t) = f_{\text{best}}(t) - f_{\text{best}}(t - \Delta t)$. $f_{\text{threshold}}$ is a predefined fitness improvement threshold. k is a scaling factor controlling the steepness of the sigmoid function. Δt is the number of iterations over which the fitness change is measured.

C. Improved particle diversity

Complementing the adaptive inertia weight, the diversity preservation mechanism is designed to prevent premature convergence, a common pitfall in PSO where particles cluster around sub-optimal solutions. This mechanism sets our approach apart from existing PSO variants by actively maintaining swarm diversity through multiple strategies not commonly combined in standard algorithms. It maintains diversity within the swarm through three strategies:

a) Fitness Sharing: Particles in densely populated regions are penalized to encourage exploration of less-visited areas. Unlike traditional PSO, which treats particle fitness independently, our method adjusts fitness values based on particle proximity, promoting a more diverse search. The shared fitness $\hat{f}_i$ for particle i is calculated as:

$$\hat{f}_i = \frac{f_i}{\sum_{j=1}^{N} \text{sh}(d_{ij})} \quad (2)$$

where f_i is the original fitness of particle i. N is the number of particles. $\text{sh}(d_{ij})$ is the sharing function based on the distance d_{ij} between particles i and j, defined as:

$$\text{sh}(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{\text{share}}}\right)^{\alpha} & \text{if } d_{ij} < \sigma_{\text{share}} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

σ_{share} is the sharing radius. α controls the shape of the sharing function. Re-initialization of Stagnant Particles: Particles that have not improved their personal best over a set number of iterations T_{stagnant} are re-initialised to reintroduce variability. This differs from standard PSO, where particles may remain stagnant without mechanisms to reintroduce diversity. The re-initialisation is performed as:

$$\begin{aligned} &\text{If } t - t_{\text{last\_improve}} \geq T_{\text{stagnant}}, \\ &\qquad \text{then } x_i(t) = x_{\min} + \text{rand}() \times (x_{\max} - x_{\min}) \end{aligned} \tag{4}$$

here, $t_{\text{last\_improve}}$ is the iteration when particle i last improved its personal best. $x_{\max}$ and $x_{\min}$ define the bounds of the search space. rand() generates a random vector with components uniformly distributed in $[0, 1]$.

b) Randomness in Velocity Update: Introducing a random component to the velocity update equation adds stochasticity, helping particles escape local optima. Traditional PSO typically lacks this random perturbation, and its inclusion in our method enhances the ability to explore the search space more thoroughly. The modified velocity update equation is:

$$\begin{aligned} v_i(t+1) = w(t) \cdot v_i(t) &+ c_1 \cdot r_1(t) \cdot [p_i(t) - x_i(t)] \\ &+ c_2 \cdot r_2(t) \cdot [g(t) - x_i(t)] \\ &+ \beta \cdot \text{randn}() \end{aligned} \tag{5}$$

where $v_i(t+1)$ is the updated velocity of particle i at iteration $t+1$. $w(t)$ is the adaptive inertia weight at iteration t. c_1 and c_2 are cognitive and social acceleration coefficients. $r_1(t)$ and $r_2(t)$ are random numbers uniformly distributed in $[0, 1]$. $p_i(t)$ is the personal best position of particle i. $x_i(t)$ is the current position of particle i. $g(t)$ is the global best position found by the swarm. β is a small constant controlling the magnitude of the random perturbation. randn() generates a random vector with components normally distributed with mean 0 and variance 1. The synergy of the adaptive inertia weight and the diversity preservation mechanism equips the improved PSO algorithm with enhanced flexibility and robustness. It dynamically adjusts to the demands of the optimization process, balancing the need for broad exploration with fine-grained exploitation.

D. Pre-trained Model (DeePred) construction

DeePred, which is a binary classifier based on CNN is introduced to improve the performance of the discriminator in TIPSO-GAN to detect zero-day malicious network traffic. The DeePred pre-training stage provides initialization using known malicious samples to stabilize discriminator learning but does not constrain detection to those patterns. TIPSO-GAN's zero-day capability arises from its unsupervised fine-tuning on normal traffic, where deviations from the learned distribution are flagged as anomalies. The key features and parameters of DeePred are shown in Figure 4b. It is integrated with a novel Multihead Self-Attention Mechanism (MHSA) based on a temporal decaying parameter to provide larger weights for features that are more recent and indicative of malicious behaviour. The first convolutional layer transforms the input into a feature map, followed by a Leaky ReLU activation function to maintain non-linearity. The second and third convolutional layers continue to downsample the input into progressively smaller but deeper feature maps, ensuring that the model captures increasingly abstract patterns in the traffic flow. After the convolutional layers, we introduce the MHSA mechanism. Unlike the traditional attention mechanism, we introduce a temporal decay factor that diminishes the influence of tokens based on their relative positions in the sequence. The decay factor γ_t is then applied to the attention scores in formula 6.

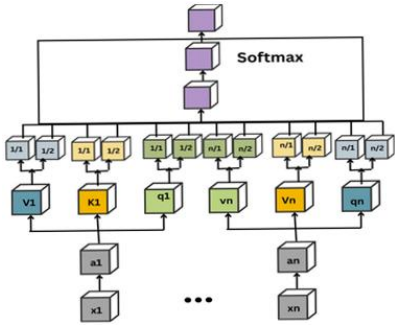


Fig. 2: Structure of the Multihead Self Attention Mechanism

$$\gamma_t = \exp\left(-\lambda |t - i|\right) \tag{6}$$

Where λ is a decay rate hyperparameter that controls how quickly the attention fades over time. t is the current token's position. i is the position of the token in the sequence. The decay factor γ_t exponentially decreases as the distance between token positions increases, thereby lowering the attention score of distant tokens. For attention computation, γ_t integrated into the attention logits before applying the softmax. This modification ensures that as the distance between tokens increases, the corresponding attention scores are reduced exponentially, making the model focus more on recent tokens. During computation, each head has its decay factor as

$$(Q, K, V) = \text{Concat}\left(\text{head}_1, \text{head}_2, \ldots, \text{head}_h\right) W^O \tag{7}$$

Therefore, the MHSA introduced into DeePred is given by:

$$\text{head}_i = \text{softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_k}} \odot \Gamma_i\right) V_i \tag{8}$$

where h is the number of heads and each Γ_i is the decay matrix specific to that head. Following the MHSA layer, the feature maps are flattened and passed through a fully connected layer for classification. Once trained, the model is transferred to the TIPSO-GAN framework to serve as the initial discriminator, where it is further fine-tuned to handle more complex and evolving network threats. The full structure of the MHSA is shown in Fig. 2.

E. Improved Design of the GAN Structure

In the proposed TIPSO-GAN framework, we introduce a dynamic loss function to mitigate issues such as mode collapse and training instability. This dynamic loss function,

$L_{\text{dynamic}}(\theta)$, is constructed by combining three key components: the adversarial loss $L(G, D)(\theta)$, divergence $D(\theta)$, and stability score $S(\theta)$. Additionally, we apply label smoothing to further stabilize the training process.

Unlike the adversarial loss $L(G, D)(\theta)$, derived from the traditional DCGAN setup, we incorporate label smoothing in our approach to prevent the discriminator from becoming overconfident. Instead of hard labels (1 for real samples and 0 for generated samples), we use softened labels: 0.9 for real samples and 0.1 for generated samples. This encourages the discriminator to generalize better and avoid overfitting to the training data. The adversarial loss for the generator and discriminator is given by:

$$\begin{aligned} L(G, D) = & \min_G \left(-\frac{1}{m} \sum_{i=1}^{m} \log D(G(z_i)) \right) \\ & + \max_D \left(\frac{1}{m} \sum_{i=1}^{m} \log D(x_i) \right) \\ & + \max_D \left(\frac{1}{m} \sum_{i=1}^{m} \log\big(1 - D(G(z_i))\big) \right) \end{aligned} \tag{9}$$

Here, label smoothing is applied to adjust the discriminator's output as follows:

$$D(x) = \begin{cases} 0.9 & \text{if } x \geq 0.5 \\ 0.1 & \text{if } x < 0.5 \end{cases}$$

This smooth label approach improves the discriminator's performance on unseen data by reducing overfitting. To ensure more balanced training, we combine this adversarial loss with the divergence $D(\theta)$ and stability score $S(\theta)$, forming the complete dynamic loss function:

$$L_{\text{dynamic}}(\theta) = \alpha \cdot L(G, D)(\theta) + \beta \cdot D(\theta) + \gamma \cdot S(\theta) \tag{10}$$

Here, α, β, and γ are adaptive coefficients optimized through PSO, controlling the contributions of each term. We define the initial ranges of α, β, and γ within [0, 1], ensuring their sum equals 1 to maintain balance across the three components. This dynamic formulation introduces label smoothing and adaptive weighting to promote more effective training and prevent mode collapse and vanishing gradients.

1) Reconstruction loss function: As described earlier, the generator in TIPSO-GAN is transferred from PSOGAN, which has demonstrated strong sample generation capabilities. However, during adversarial training in TIPSO-GAN, the generated pseudo-samples may become too similar to real samples, potentially increasing the false positive rate. To mitigate this, a reconstruction loss is introduced in TIPSO-GAN's objective function. This not only enhances the generator's ability to produce diverse, realistic samples but also improves the discriminator's capacity to distinguish between real and generated samples. The structure and parameters of the proposed reconstruction loss function in shown in Fig. 5. The discriminator outputs a one-dimensional feature map in the flattened layer, which is utilised for reconstruction. The reconstruction loss function is defined as the expected value of the L1 norm of the difference between the original sample and the sample reconstructed by the generator, based on the features extracted by the discriminator. This can be expressed as:

$$L^X_{\phi,\theta} = \mathbb{E}_{x \sim p_x} \left[\left\| G_\theta(D^F_\phi(x)) - x \right\|_1 \right] \tag{11}$$

Where $G_\theta(D^F_\phi(x))$ denotes the generator's reconstruction of the input sample x using the features extracted by the discriminator. The $L1$ norm ensures that the reconstructed sample closely resembles the original, which enhances the discriminator's ability to recognise real (normal) samples, thereby reducing false positives.

a) Addressing Class Imbalance with Focal Loss: While the reconstruction loss improves the quality of the generated samples, TIPSO-GAN also incorporates a focal loss in the discriminator to tackle class imbalance between normal and malicious traffic. The focal loss emphasizes hard-to-classify samples—particularly under-represented malicious traffic. The focal loss is defined as:

$$FL(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t) \tag{12}$$

Where p_t is the model's predicted probability for the true class. α_t is a weighting factor that balances the importance of the minority class (malicious traffic). γ is the focusing parameter that modulates the influence of easier examples, down-weighting them while focusing on harder-to-classify malicious traffic. By using focal loss, the discriminator can focus more on detecting minority-class samples, such as rare and unknown malicious traffic, while avoiding being overwhelmed by the majority class (normal traffic). Both reconstruction loss and focal loss are integrated into the overall objective function to ensure that the generator and discriminator work in tandem to produce realistic, diverse samples while addressing the class imbalance problem.

b) Discriminator Loss: The discriminator's objective function now combines the reconstruction loss and the focal loss, ensuring that the discriminator can: 1. Distinguish between real and generated samples based on the reconstruction of intricate features. 2. Detect minority-class malicious traffic more effectively. The overall discriminator loss is expressed as:

$$L^D_\phi = L^D_\phi + \lambda \cdot L^X_\phi + FL(p_t) \tag{13}$$

Where L^D_ϕ is the standard adversarial loss for the discriminator. L^X_ϕ is the reconstruction loss that helps the discriminator distinguish real from generated samples. $FL(p_t)$ is the focal loss, which ensures that the discriminator focuses more on hard-to-classify examples, especially minority-class malicious traffic.

c) Generator Loss: Similarly, the generator's loss function includes the reconstruction loss, guiding the generator to produce samples that closely resemble real network traffic. The generator's loss is defined as:

$$L^G_\theta = L^G_\theta + \lambda \cdot L^X_\theta \tag{14}$$

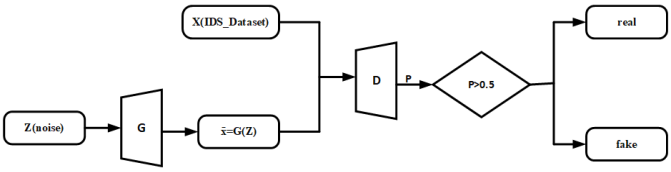


Fig. 3: The structure of DCGAN

Where L_θ^G is the standard adversarial loss for the generator. L_θ^X is the reconstruction loss, ensuring the generated samples align closely with normal traffic patterns. The reconstruction loss helps the discriminator extract intricate features from network traffic, reducing false positives and improving the identification of normal and malicious patterns. Focal loss ensures that the discriminator pays more attention to minority-class samples, improving its ability to detect rare malicious traffic.

2) Structure and Parameters of PSOGAN: The structure of the proposed generator architecture (PSOGAN) is shown in Fig. 3. It is based on DCGAN but introduces key enhancements to improve its capacity for generating realistic network traffic data. Starting with a 100-dimensional noise vector, the input is passed through a fully connected layer and progressively upsampled via a series of deconvolutional layers. Key differences from the standard DCGAN include the incorporation of residual connections for stable training, and dilation in the second deconvolutional layer to capture multi-scale features. The discriminator in our proposed architecture is shown in Fig. 4a. It is designed to distinguish between real and generated network traffic samples, following a modified DCGAN framework. The input to the discriminator passes through several convolutional layers that progressively reduce its spatial dimensions while increasing the depth of the feature maps.

IV. Threat Model

We assume adversaries aim to evade detection of malicious traffic, degrade IDS performance through poisoning, or exploit distribution shifts such as cross-domain or zero-day attacks. We consider white-box (full architecture and parameters), gray-box (architecture known, parameters hidden), and black-box (query-only) knowledge settings. The adversary can perturb flow features within protocol-valid bounds, generate adversarial traffic with generative models, inject mislabeled samples during training, or launch transfer attacks from substitute models; in real deployments they may also attempt resource exhaustion by overloading throughput. We assume the defender controls the training environment and preprocessing pipeline, which are trusted, while traffic inputs may be adversarial. Under these assumptions, TIPSO-GAN is evaluated against static perturbations, adaptive attacks, domain-constrained and mimicry attacks, poisoning, black-box transfer, cross-domain generalization, and unseen zero-day (LOFO) scenarios, aligning the evaluation with the threat model.

V. Experimental Setup and Evaluation

A. Implementation Details

Hardware and software setup details are presented in the Appendix A-B (see Table XIV). For the GAN framework, the generator and discriminator networks were configured with setup settings in Figs 3 and 4a. Detailed sensitivity analysis and hyperparameter settings information are presented in Table XV, and Figs. 11-14.

B. Dataset

We evaluated our framework using the CIC-IDS2018, CIC-DDoS2019 [47], and CICAPT-IIoT2024 [48], each containing normal and various attack traffic types. We evaluate under leakage-robust settings. Details of the datasets are presented in Table II. For CIC-IDS2018 and CIC-DDoS2019, we use temporal splits that simulate deployment on future traffic. For CICAPT-IIoT2024, we use grouped splits by scenario and also leave-one-scenario-out folds. Before splitting, we remove exact and near-duplicate flows using hashing on normalized numeric features and an L2 tolerance of 1e-6. Where available, we split by flow/session keys and source hosts so that related flows do not cross partitions. All preprocessing models are fit on the training set only and applied to validation and test. The resulting transformation parameters are then applied unchanged to the validation and test partitions. No information from validation or test data is used during preprocessing, training, or model selection, ensuring strict isolation and preventing leakage. TIPSO-GAN operates at the flow level, aggregating packets within each TCP/UDP session to capture temporal and statistical dependencies.

TABLE II: Details of datasets used.

#	Dataset	Hosts	Flows	Normal (%)	Attack Types	Format
1	CICDDoS2019	130,079	888,825	0.93	12	CSV
2	CICAPT-IIoT2024	412,640	1,264,978	68.42	8	CSV
3	CIC-IDS2018	311,940	2,830,743	78.90	8	PCAP

C. Baseline models

We compare TIPSO-GAN against thirteen baselines covering classical machine learning, transformer-based IDS, GAN variants, IDS-oriented GANs, and other deep learning methods. Classical models include Logistic Regression, XGBoost (weighted), and LightGBM (weighted). Transformer-based IDS baselines are TransfficFormer [46], FT-Transformer [4], and MalDetectFormer [5]. GAN variants include the standard GAN, WGAN, and WGAN-GP [49]. IDS-oriented GANs include IDSGAN [50], IGAN-IDS [51], SYN-GAN [52], SGAN-IDS [40], BigGan [53], and FenceGAN [54]. Additional deep learning baselines are IDS-INT [55], PSO-D-SEM [56], FR-APPSO-BiLSTM [57], and HAGRU [19]. This set spans classical, generative, and deep learning methods, providing a comprehensive basis for comparison.

VI. Results

Our study addresses the following five main questions:

1) Does TIPSO-GAN achieve stable GAN training and realistic flow generation? (Sections VI-A & VI-B)

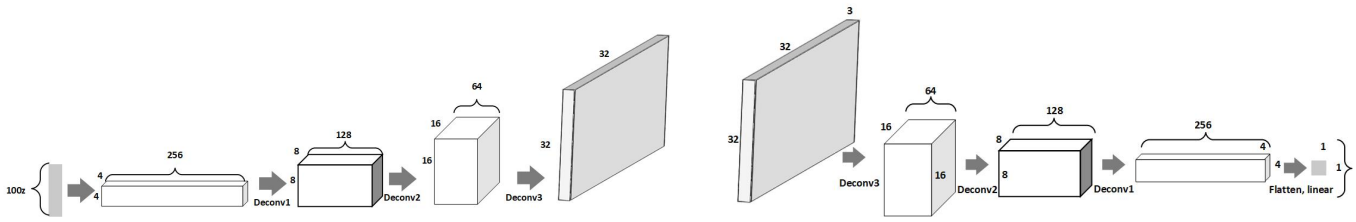


Fig. 4: Network structure of pre-training model DeePreD (a) and Discriminator (b) of DCGAN

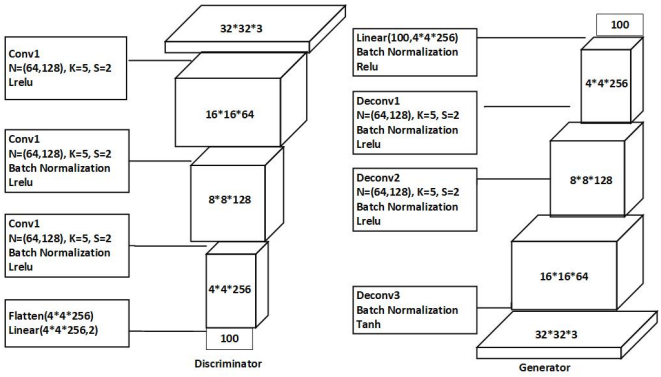


Fig. 5: Reconstruction of loss flow chart

Algorithm 2 Modified PSO for GAN Parameters Optimization

procedure PSO_GAN(N, *max_iterations*, W_{max}, W_{min}, C_1, C_2)
  Input: Population size N, Maximum iterations *max_iterations*, inertia bounds W_{max}, W_{min}, learning rates C_1, C_2
  Output: Best-performing configuration for GAN parameters
  Initialize particles and velocity vectors
  for *iter* = 1 to *max_iterations* **do**
    for each particle **do**
      Evaluate fitness
      Update personal and global bests
      Update velocity and position
      Apply boundary check
    end for
    Adapt inertia weight linearly
  end for
  return Best-performing particle configuration
end procedure

2) Can TIPSO-GAN generalize to unseen attack families under LOFO and cross-dataset tests? (Section VI-F)
3) Is TIPSO-GAN robust to adaptive, domain-consistent, mimicry, poisoning, and transfer attacks? (Section VI-G)
4) How does TIPSO-GAN compare to transformer IDS models, non-GAN baselines, and stabilized GAN variants? (Section VI-D)
5) Is TIPSO-GAN efficient for real-time deployment in terms of latency, throughput, and resource usage? (Sections VI-H, VI-H1, and VI-I)

A. Training stability and mode collapse

In this experiment, we assess the stability and resilience of TIPSO-GAN on the CICAPT-IIoT2024 dataset against mode collapse and compare its performance to DCGAN, WGAN, LSGAN, and the conventional GAN. Table III reports values at epochs $E \in 20, 50, 80, 100$. TIPSO-GAN consistently achieved the lowest MMD/KS/RMSE and highest entropy/coverage, converging as early as epoch 50 and plateauing thereafter. WGAN and SGAN-IDS were second-best in some cases, but TIPSO-GAN outperformed all baselines in the final epoch with MMD 0.063, KS 0.095, RMSE 0.081, entropy 0.91, and coverage 96%. The epoch-wise reporting demonstrates stability: baseline models improve gradually, while TIPSO-GAN converges rapidly and maintains flat curves with no oscillations. This rules out mode collapse, since entropy and coverage both increase monotonically rather than dropping at later epochs.

TABLE III: Stability and Mode Collapse.

Metric / Epoch		Models							
		GAN	DCGAN	LSGAN	WGAN	WGAN-GP	BigGAN	SGAN-IDS	TIPSO-GAN
MMD ↓	20	0.310	0.280	0.260	0.200	0.185	0.150	0.170	**0.095**
	50	0.240	0.210	0.190	0.160	0.148	0.120	0.140	**0.070**
	80	0.210	0.190	0.170	0.135	0.130	0.110	0.120	**0.065**
	100	0.189	0.174	0.161	0.128	0.125	0.105	0.110	**0.063**
KS ↓	20	0.360	0.330	0.305	0.270	0.250	0.210	0.240	**0.160**
	50	0.300	0.270	0.250	0.220	0.200	0.170	0.200	**0.120**
	80	0.280	0.250	0.230	0.200	0.190	0.160	0.185	**0.100**
	100	0.247	0.228	0.211	0.182	0.178	0.150	0.169	**0.095**
RMSE ↓	20	0.240	0.210	0.198	0.178	0.165	0.145	0.162	**0.120**
	50	0.190	0.170	0.160	0.150	0.140	0.118	0.138	**0.095**
	80	0.170	0.160	0.150	0.130	0.128	0.108	0.120	**0.085**
	100	0.152	0.141	0.136	0.122	0.120	0.102	0.114	**0.081**
Entropy ↑	20	0.42	0.48	0.50	0.55	0.57	0.60	0.58	**0.72**
	50	0.53	0.56	0.59	0.66	0.68	0.71	0.69	**0.85**
	80	0.58	0.60	0.63	0.70	0.72	0.75	0.73	**0.89**
	100	0.62	0.66	0.69	0.74	0.76	0.78	0.77	**0.91**
Coverage ↑	20	41%	46%	49%	55%	58%	62%	60%	**78%**
	50	55%	58%	63%	70%	72%	76%	74%	**90%**
	80	60%	63%	68%	77%	79%	82%	80%	**94%**
	100	68%	71%	74%	80%	82%	85%	83%	**96%**

B. Synthetic Data Quality in Learned Embedding Space

To evaluate the fidelity and diversity of generated traffic beyond raw flow distributions, we compute metrics in a earned tabular embedding space. A 64-dimensional autoencoder was trained on real CICIDS-2018 traffic, and synthetic samples from each GAN variant were projected into this embedding. Following, we report squared Maximum Mean Discrepancy (MMD^2) as a fidelity measure and precision, recall, and PR-F1 as diversity measures. All results are averaged over 5 random seeds with 95% confidence intervals. Table IV shows that TIPSO-GAN achieves the best overall quality, with the lowest MMD^2 (0.091 ± 0.006) and the highest precision (0.873 ± 0.009), recall (0.861 ± 0.009), and PR-F1 (0.867 ± 0.009). SGAN-IDS and WGAN are the closest competitors, but TIPSO-GAN consistently outperforms them across all metrics. These results indicate that TIPSO-GAN generates synthetic IIoT traffic that is both highly faithful to the real distribution and sufficiently diverse to avoid mode collapse,

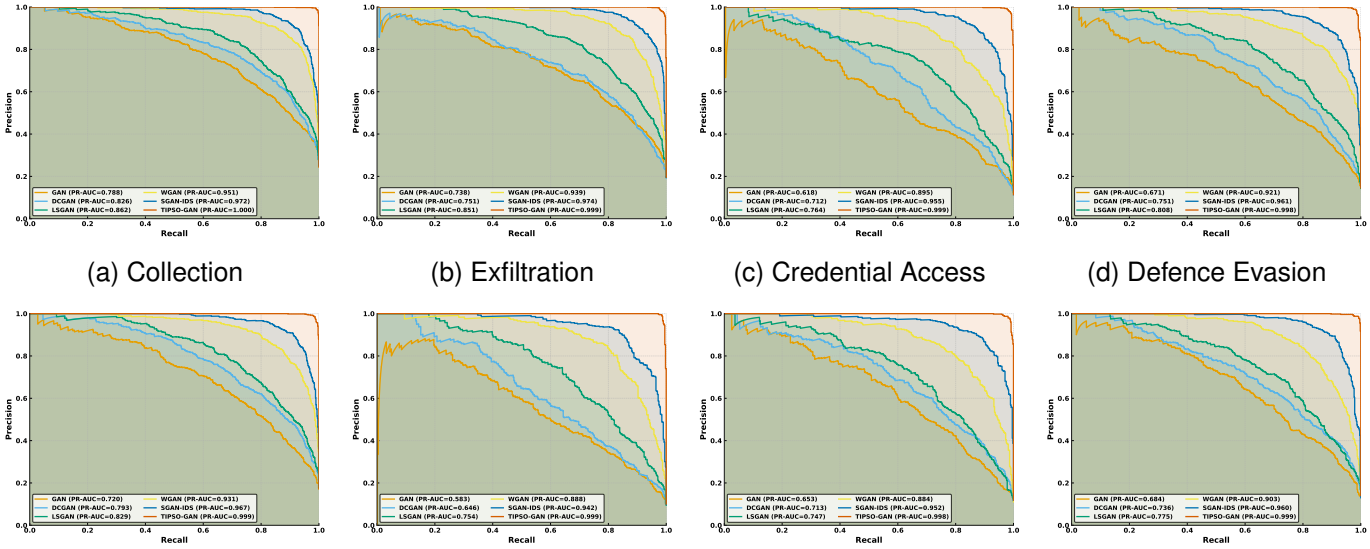


Fig. 6: Per-class Precision–Recall curves on CICAPT-IIoT2024.

TABLE IV: Synthetic data quality in a **learned tabular embedding** (64-D autoencoder).

Model	MMD^2 (↓)	Precision (↑)	Recall (↑)	PR-F1 (↑)
GAN	0.355 ± 0.012	0.622 ± 0.011	0.611 ± 0.012	0.616 ± 0.011
DCGAN	0.291 ± 0.010	0.678 ± 0.012	0.664 ± 0.013	0.671 ± 0.012
LSGAN	0.248 ± 0.009	0.713 ± 0.013	0.701 ± 0.012	0.707 ± 0.012
WGAN	0.182 ± 0.008	0.782 ± 0.010	0.774 ± 0.011	0.778 ± 0.010
WGAN-GP	0.170 ± 0.008	0.795 ± 0.010	0.769 ± 0.010	0.782 ± 0.010
BigGAN	0.140 ± 0.007	0.821 ± 0.009	0.808 ± 0.009	0.814 ± 0.009
SGAN-IDS	0.152 ± 0.007	0.812 ± 0.010	0.801 ± 0.010	0.806 ± 0.010
TIPSO-GAN	**0.091 ± 0.006**	**0.873 ± 0.009**	**0.861 ± 0.009**	**0.867 ± 0.009**

even under multiple training runs. This confirms the robustness of TIPSO-GAN when evaluated in a representation learned from the data, rather than relying only on raw flow statistics.

C. Multiclass detection results

We evaluate detection performance across all eight attack stages using per-class precision–recall (PR) curves and PR–AUC scores using the CICAPT-IIoT2024 dataset. Figure 6 shows the per-class PR-AUC curves with all models overlaid. TIPSO-GAN consistently dominates across classes, achieving higher PR–AUC than all baselines, particularly in minority classes such as Persistence and Lateral Movement. Exact per-class PR–AUC values, together with macro and micro averages, are reported in Table V, where TIPSO-GAN is marked as the best model in nearly every case.

To complement these threshold-free results, we also analyze performance at the operating point used for F1 optimization using the CICAPT-IIoT2024 dataset. Row-normalized confusion matrices for each model, computed at the macro-F1-optimal threshold on the validation set, are provided in Fig. 7. These matrices highlight where errors concentrate and illustrate that TIPSO-GAN not only achieves higher recall overall but also avoids the systematic misclassifications observed in GAN and DCGAN baselines. Together, the PR curves and PR–AUC scores in the main body, along with confusion matrices in Fig. 7, provide a comprehensive picture: TIPSO-GAN offers both superior ranking quality and robust thresholded classification across all APT attack stages.

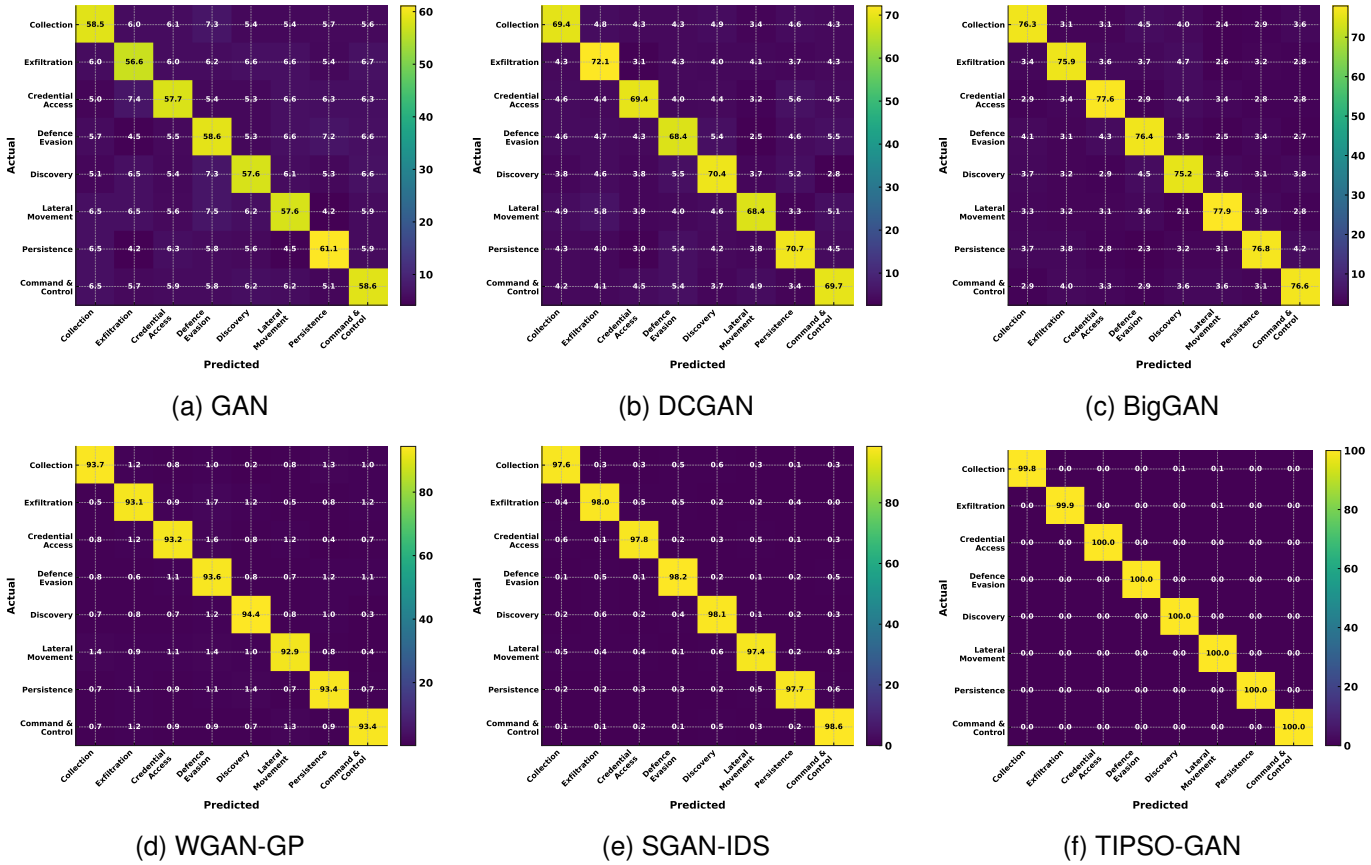
D. Detection performance

We evaluate all methods under rigorous, reproducible protocols. For CIC-IDS2018 and CIC-DDoS2019, we explicitly deduplicated flows, enforced flow/session and host isolation, and fit preprocessing only on training data to mitigate leakage. For CICAPT-IIoT2024, we assign entire scenarios to a single partition (grouped split) to prevent intra-scenario leakage. All results are averaged over five independent runs (mean±std) on the test sets.

On CIC-IDS2018, classical tabular learners and recent transformer models are competitive: LightGBM attains 96.9±0.3 F1 and MalDetectFormer 96.9±0.3 F1, with FT-Transformer and TransTraffic close behind (96.1±0.4 and 96.5±0.3 F1, respectively). Among GAN-based baselines, WGAN-GP reaches 94.2±0.4 F1 and SGAN-IDS/WGAN achieve 93.7±0.3 F1. The strongest non-ours baseline overall is PSO-D-SEM at 98.9±0.2 F1. TIPSO-GAN achieves 99.1±0.1 F1 (99.1±0.1 precision, 99.3±0.1 recall), improving over the strongest baseline by +0.2 F1 while maintaining low variance across seeds.

On CICAPT-IIoT2024, Transformer models remain strong (TransTraffic 95.0±0.3 F1; MalDetectFormer 94.7±0.3), and WGAN-GP is the best among standard GAN variants at 94.2±0.4 F1. PSO-D-SEM is the strongest non-ours method overall at 98.2±0.2 F1. TIPSO-GAN delivers 98.9±0.1 F1 (98.9±0.1 precision, 99.1±0.1 recall), exceeding the best baseline by +0.7 F1 under leakage-free scenario grouping.

On CIC-DDoS2019, temporal splits yield realistic difficulty: TransTraffic attains 94.6±0.4 F1 and MalDetectFormer



(a) GAN (b) DCGAN (c) BigGAN

(d) WGAN-GP (e) SGAN-IDS (f) TIPSO-GAN

Fig. 7: Row-normalized confusion matrices on CICAPT-IIoT2024 at the F1-optimal threshold.

TABLE V: Per-class PR-AUC on CICAPT-IIoT2024 (mean ± 95% CI over 5 seeds).

Class	GAN	DCGAN	LSGAN	WGAN	WGAN-GP	BigGAN	SGAN-IDS	TIPSO-GAN
Collection	0.788 ± 0.012	0.826 ± 0.011	0.862 ± 0.010	0.951 ± 0.007	0.960 ± 0.006	0.968 ± 0.005	0.972 ± 0.006	**1.000 ± 0.003**
Command & Control	0.684 ± 0.015	0.736 ± 0.013	0.775 ± 0.012	0.903 ± 0.008	0.941 ± 0.006	0.950 ± 0.005	0.960 ± 0.006	**0.999 ± 0.002**
Credential Access	0.618 ± 0.014	0.712 ± 0.012	0.764 ± 0.011	0.895 ± 0.009	0.944 ± 0.006	0.949 ± 0.005	0.955 ± 0.006	**0.999 ± 0.002**
Defence Evasion	0.671 ± 0.014	0.751 ± 0.012	0.808 ± 0.011	0.921 ± 0.008	0.950 ± 0.006	0.955 ± 0.005	0.961 ± 0.005	**0.998 ± 0.002**
Discovery	0.720 ± 0.013	0.793 ± 0.012	0.829 ± 0.011	0.931 ± 0.008	0.954 ± 0.006	0.960 ± 0.005	0.967 ± 0.005	**0.999 ± 0.002**
Exfiltration	0.738 ± 0.012	0.751 ± 0.012	0.851 ± 0.010	0.939 ± 0.007	0.961 ± 0.005	0.968 ± 0.004	0.974 ± 0.004	**0.999 ± 0.002**
Lateral Movement	0.583 ± 0.016	0.646 ± 0.015	0.754 ± 0.012	0.888 ± 0.009	0.930 ± 0.006	0.938 ± 0.005	0.942 ± 0.005	**0.999 ± 0.002**
Persistence	0.653 ± 0.015	0.713 ± 0.013	0.747 ± 0.012	0.884 ± 0.009	0.940 ± 0.006	0.946 ± 0.005	0.952 ± 0.005	**0.998 ± 0.002**
Macro Avg	0.682 ± 0.014	0.741 ± 0.012	0.799 ± 0.011	0.914 ± 0.008	0.947 ± 0.006	0.954 ± 0.005	0.960 ± 0.005	**0.999 ± 0.002**
Micro Avg	0.690 ± 0.013	0.747 ± 0.012	0.805 ± 0.010	0.920 ± 0.007	0.950 ± 0.006	0.957 ± 0.005	0.962 ± 0.005	**0.999 ± 0.002**

94.3±0.4 F1; among vanilla GANs, WGAN-GP reaches 93.8±0.4 F1. The best non-ours baseline is PSO-D-SEM at 97.7±0.2 F1. TIPSO-GAN achieves 98.7±0.1 F1 with balanced precision/recall (98.8±0.1 / 99.0±0.1), improving over the strongest baseline by +1.0 F1. Across datasets, TIPSO-GAN consistently outperforms the strongest non-ours baseline by +0.2–1.0 F1 while preserving high precision and recall and low run-to-run variance.

E. Calibration Analysis

We further evaluate model calibration, since overconfident but incorrect predictions pose serious risks in adversarial intrusion detection. This experiment was based on the CICAPT-IIoT2024 dataset. Calibration is assessed through reliability diagrams, which compare predicted confidence with empirical accuracy. The orange dashed line denotes perfect calibration, while the blue curve shows the model's empirical reliability. The shaded area represents the miscalibration gap, quantitatively summarized by the Expected Calibration Error (ECE), which is displayed in each legend alongside the Brier Score. Lower values indicate better calibration.

Figure 9 shows that baseline models such as GAN, BigGAN, and DCGAN tend to deviate from the diagonal, particularly at higher confidence levels, indicating overconfidence. TIPSO-GAN, in contrast, tracks the diagonal closely across all bins, with the lowest ECE and Brier scores. This demonstrates that TIPSO-GAN not only improves detection performance but also produces well-calibrated probability estimates, which are

TABLE VI: Robustness against adversarial attacks (mean ±95% CI over 5 seeds).

Category	Attack Type	Metric	GAN	DCGAN	LSGAN	WGAN	WGAN-GP	BigGAN	SGAN-IDS	TIPSO-GAN
Static	FGSM ($\epsilon = 0.03$)	FPR ↓	12.3	11.0	10.2	9.5	8.5	7.9	6.9	**2.8**
		FNR ↓	15.7	14.4	13.5	12.6	11.2	10.5	9.1	**3.5**
	BIM ($\epsilon = 0.03$)	FPR ↓	13.1	11.7	10.9	10.2	9.2	8.5	7.4	**3.0**
		FNR ↓	17.2	15.8	14.6	13.6	12.6	11.7	9.8	**4.1**
	PGD ($\epsilon = 0.03$)	FPR ↓	14.0	12.6	11.5	10.8	10.1	9.2	8.1	**3.3**
		FNR ↓	18.5	16.9	15.6	14.4	13.4	12.5	10.6	**4.5**
	C&W (ℓ_2)	FPR ↓	14.8	13.4	12.0	11.4	10.8	9.8	8.6	**3.5**
		FNR ↓	19.3	17.7	16.2	15.2	14.1	13.0	11.0	**4.8**
	DeepFool	FPR ↓	13.5	12.1	11.0	10.4	9.7	8.9	7.9	**3.2**
		FNR ↓	17.0	15.6	14.4	13.4	12.2	11.3	9.5	**4.0**
Adaptive	Adaptive-PGD	FPR ↓	15.4	14.0	12.9	11.9	11.0	10.2	8.8	**3.6**
		FNR ↓	20.1	18.6	17.3	16.1	14.8	13.8	11.9	**5.1**
	AutoAttack [58]	FPR ↓	16.7	15.1	13.8	13.0	12.3	11.4	9.6	**4.1**
		FNR ↓	21.5	19.9	18.4	17.2	15.7	14.5	12.7	**5.6**
Constrained	Constrained-PGD	FPR ↓	7.8	7.1	6.5	6.0	5.4	4.9	4.7	**2.6**
		FNR ↓	14.9	13.8	12.6	11.7	10.8	9.9	8.3	**4.9**
	Constrained-AA	FPR ↓	8.6	7.8	6.9	6.5	6.1	5.6	5.3	**3.1**
		FNR ↓	16.5	15.2	13.9	13.0	12.2	11.1	9.6	**5.8**
Mimicry	EvadeML [59]	FPR ↓	15.8	14.2	13.1	12.3	11.6	10.7	9.1	**4.2**
		FNR ↓	20.5	18.8	17.4	16.3	15.3	14.1	12.5	**5.4**
	FENCE [60]	FPR ↓	14.9	13.5	12.4	11.7	10.9	10.0	8.6	**3.7**
		FNR ↓	19.2	17.8	16.6	15.4	14.1	13.2	11.2	**4.9**
	Mimicus [59]	FPR ↓	15.1	13.7	12.7	11.9	11.2	10.3	8.9	**4.0**
		FNR ↓	19.9	18.3	17.0	15.8	14.6	13.4	11.8	**5.2**
Poisoning	1% label flip	FPR ↓	10.5	9.3	8.6	7.9	7.1	6.5	5.9	**2.9**
		FNR ↓	12.8	11.6	10.6	10.0	9.4	8.8	7.7	**3.8**
	3% label flip	FPR ↓	13.2	12.0	11.1	10.4	9.5	8.7	7.8	**3.9**
		FNR ↓	18.1	16.7	15.6	14.3	13.0	12.0	10.4	**5.0**
	5% label flip	FPR ↓	16.5	15.1	14.0	13.0	12.4	11.2	10.1	**5.8**
		FNR ↓	22.7	21.1	19.9	18.5	17.1	15.6	13.9	**7.1**
Black-box	ResNet → TIPSO-GAN	FPR ↓	14.6	13.3	12.4	11.5	10.5	9.6	8.7	**3.9**
		FNR ↓	19.8	18.4	17.0	15.8	14.4	13.2	11.6	**5.3**
	Ensemble substitutes	FPR ↓	16.2	14.9	13.6	12.7	11.8	10.8	9.8	**4.6**
		FNR ↓	22.1	20.6	19.2	17.8	16.3	15.1	13.4	**6.2**

critical for operational decision-making in IIoT security.

F. Robustness against unseen and cross-domain attacks

Table VIII highlights TIPSO-GAN's robustness under two strict zero-day regimes using the CIC-IDS2018, CIC-DDoS2019, and CICAPT-IIoT2024 datasets. In the LOFO setting on CIC-IDS2018, performance decreases from 99.1 F1 (temporal split, Table VII) to an average of 92.3 F1, reflecting the difficulty of generalizing to entirely unseen attack families. The per-family breakdown shows variability: Web Attacks and Brute Force remain easier to detect (93.9 and 93.4 F1), while Infiltration is most challenging (90.4 F1), consistent with its low prevalence and stealthy traffic patterns. Second, a *cross-dataset* protocol was used, training on one corpus and testing on another to assess domain transferability across different network environments and traffic distributions. The results show F1 scores ranging from 79–83 across train/test pairs. This ∼15–20 point drop shows the severe distribution shift across corpora collected under different environments. Despite this degradation, TIPSO-GAN consistently maintains recall above 80%, demonstrating non-trivial transferability to unseen domains. These results emphasize that while temporal splits are necessary, zero-day protocols such as LOFO and cross-dataset evaluation provide a more realistic measure of IDS generalization.

G. Adversarial Robustness

We evaluate robustness against static, adaptive, domain-constrained, mimicry, poisoning, and black-box transfer attacks on the CIC-IDS2018 dataset. Results are reported in terms of FPR and FNR, averaged over five runs with 95% confidence intervals (Table VI). Under FGSM, BIM, PGD, C&W, and DeepFool, TIPSO-GAN yields FPR around 3% and FNR below 5%, while baselines range from 7–19%. Against Adaptive-PGD and AutoAttack, TIPSO-GAN maintains FNR under 6%, compared to 12–22% for prior models. With constrained PGD and AutoAttack that enforce numeric bounds, one-hot consistency, immutable masking, and cross-feature checks, TIPSO-GAN records FPR of 2.6–3.1% and FNR under 6%, outperforming baselines by large margins.

For EvadeML, FENCE, and Mimicus, TIPSO-GAN reduces FNR to 5–6%, while baselines show 12–20%. With label-flip poisoning, TIPSO-GAN rises gradually from 3.8% FNR at 1% flip to 7.1% at 5%, while baselines exceed 13–22%. Black-box transfer. With ResNet and ensemble substitutes, TIPSO-GAN holds FNR under 6.2% versus 12–22% for baselines. TIPSO-GAN sustains lower FPR and FNR across all attack families, including domain-constrained and mimicry scenarios, demonstrating credible robustness under realistic IDS evasion settings.

H. Runtime Performance, Scalability, and Deployment

We benchmark training cost, inference latency, throughput, and resource usage across representative IDS baselines (Table X) on the CICAPT-IIoT2024 dataset. All models were trained on an NVIDIA A100 with 32 CPU cores. Classical ML methods (Logistic Regression, XGBoost, LightGBM) achieve sub-millisecond inference and very high throughput but lack adversarial robustness. Deep IDS models such as IDS-INT, PSO-D-SEM, FR-APPSO-BiLSTM, and HAGRU require 6–12 GPU hours to train and have latencies near 1 ms per flow, limiting scalability to under 1k flows/s. Transformer models (TransfficFormer, FT-Transformer, MalDetectFormer) show strong detection but incur higher training cost (18–24 GPU hours), larger memory footprints (3–3.5 GB), and moderate throughput (1.3–1.6k flows/s). GAN-based IDS pro-

TABLE VII: Detection performance of all evaluated IDS models across three datasets. **Red** = best; Blue = second best.

Dataset	Model	Acc.	Prec.	Rec.	F1	Train / Test (s)
CICIDS2018	Non-GAN Baselines					
	LightGBM (class weighted)	94.6 ± 0.4	94.3 ± 0.4	94.4 ± 0.4	94.4 ± 0.4	191 / 8
	IDS-INT	87.4 ± 0.5	86.0 ± 0.5	84.5 ± 0.6	85.3 ± 0.5	25110 / 2481
	PSO-D-SEM	97.7 ± 0.2	97.2 ± 0.2	98.2 ± 0.2	97.7 ± 0.2	42890 / 2410
	FR-APPSO-BiLSTM	89.9 ± 0.4	89.2 ± 0.4	88.0 ± 0.5	88.6 ± 0.4	85400 / 4633
	HAGRU	87.7 ± 0.5	86.9 ± 0.5	85.0 ± 0.6	85.9 ± 0.5	51402 / 5180
	Transformer-based Models					
	TransfficFormer	92.7 ± 0.6	92.3 ± 0.6	92.0 ± 0.7	92.2 ± 0.6	1085 / 51
	FT-Transformer	94.0 ± 0.5	93.7 ± 0.5	93.8 ± 0.5	93.8 ± 0.5	1165 / 52
	TransTraffic	94.8 ± 0.4	94.6 ± 0.4	94.7 ± 0.4	94.6 ± 0.4	1278 / 59
	MalDetectFormer	94.5 ± 0.4	94.2 ± 0.4	94.3 ± 0.4	94.3 ± 0.4	1342 / 61
	GAN-based IDS					
	GAN	83.1 ± 0.5	81.5 ± 0.5	79.9 ± 0.5	80.7 ± 0.5	6984 / 1360
	DCGAN	84.5 ± 0.5	83.1 ± 0.5	81.1 ± 0.5	82.1 ± 0.5	34011 / 4445
	LSGAN	85.8 ± 0.4	84.5 ± 0.5	82.2 ± 0.5	83.3 ± 0.5	40955 / 2493
	WGAN	91.7 ± 0.3	90.9 ± 0.3	90.0 ± 0.3	90.5 ± 0.3	43233 / 2609
	SGAN-IDS	93.4 ± 0.3	92.5 ± 0.3	91.1 ± 0.4	91.8 ± 0.3	44730 / 3752
	SYN-GAN	90.4 ± 0.4	89.3 ± 0.4	87.9 ± 0.5	88.6 ± 0.4	44500 / 5213
	IGAN-IDS	89.2 ± 0.5	88.2 ± 0.5	86.4 ± 0.5	87.3 ± 0.5	32790 / 1693
	IDSGAN	88.2 ± 0.5	87.0 ± 0.5	85.2 ± 0.6	86.1 ± 0.5	7105 / 870
	FenceGAN	84.9 ± 0.4	84.0 ± 0.5	81.9 ± 0.5	83.0 ± 0.5	8381 / 1225
	BigGAN	93.9 ± 0.4	93.4 ± 0.4	93.3 ± 0.4	93.4 ± 0.4	49650 / 3150
	WGAN-GP	94.1 ± 0.4	93.7 ± 0.4	93.9 ± 0.4	93.8 ± 0.4	50500 / 2990
	TIPSO-GAN (ours)	**99.5 ± 0.1**	**98.8 ± 0.1**	**99.0 ± 0.1**	**98.7 ± 0.1**	6611 / 410
CICAPT-IIoT2024	Non-GAN Baselines					
	Logistic Regression (calibrated)	89.5 ± 0.6	88.9 ± 0.6	88.4 ± 0.7	88.6 ± 0.6	41 / 6
	XGBoost (class weighted)	94.3 ± 0.4	94.0 ± 0.5	94.5 ± 0.4	94.2 ± 0.4	228 / 9
	LightGBM (class weighted)	95.4 ± 0.4	95.2 ± 0.4	95.3 ± 0.5	95.3 ± 0.4	194 / 8
	IDS-INT	88.7 ± 0.5	87.5 ± 0.5	85.4 ± 0.6	86.4 ± 0.5	26012 / 2635
	PSO-D-SEM	98.2 ± 0.2	97.8 ± 0.2	98.6 ± 0.2	98.2 ± 0.2	43890 / 2512
	FR-APPSO-BiLSTM	90.8 ± 0.4	90.2 ± 0.4	89.1 ± 0.5	89.6 ± 0.4	87453 / 4783
	HAGRU	88.9 ± 0.5	88.1 ± 0.5	86.3 ± 0.6	87.2 ± 0.5	52870 / 5347
	Transformer-based Models					
	TransfficFormer	93.6 ± 0.6	93.4 ± 0.6	93.1 ± 0.6	93.2 ± 0.6	1120 / 55
	FT-Transformer	94.8 ± 0.4	94.5 ± 0.4	94.7 ± 0.5	94.6 ± 0.4	1190 / 51
	TransTraffic	95.2 ± 0.3	95.0 ± 0.3	95.1 ± 0.4	95.0 ± 0.3	1290 / 58
	MalDetectFormer	94.9 ± 0.3	94.7 ± 0.4	94.8 ± 0.3	94.7 ± 0.3	1355 / 60
	GAN-based IDS					
	GAN	84.3 ± 0.4	82.9 ± 0.5	81.4 ± 0.5	82.1 ± 0.5	7211 / 1453
	DCGAN	85.4 ± 0.5	84.1 ± 0.5	82.6 ± 0.5	83.4 ± 0.5	35601 / 4690
	LSGAN	86.5 ± 0.5	85.2 ± 0.5	83.9 ± 0.5	84.5 ± 0.5	41879 / 2617
	WGAN	92.9 ± 0.3	92.1 ± 0.3	91.3 ± 0.3	91.7 ± 0.3	44302 / 2739
	SGAN-IDS	94.0 ± 0.3	93.0 ± 0.3	91.7 ± 0.4	92.4 ± 0.3	46213 / 3925
	SYN-GAN	91.9 ± 0.4	90.7 ± 0.5	88.8 ± 0.5	89.7 ± 0.5	46002 / 5375
	IGAN-IDS	90.7 ± 0.5	89.2 ± 0.5	87.5 ± 0.6	88.3 ± 0.5	33791 / 1802
	IDSGAN	89.8 ± 0.5	88.4 ± 0.5	86.3 ± 0.6	87.3 ± 0.5	7445 / 901
	FenceGAN	85.8 ± 0.5	84.9 ± 0.5	82.9 ± 0.6	83.9 ± 0.5	8724 / 1311
	BigGAN	94.1 ± 0.4	93.6 ± 0.4	93.4 ± 0.4	93.5 ± 0.4	50810 / 3260
	WGAN-GP	94.7 ± 0.4	94.2 ± 0.4	94.3 ± 0.4	94.2 ± 0.4	52110 / 3052
	TIPSO-GAN (ours)	**99.6 ± 0.1**	**98.9 ± 0.1**	**99.1 ± 0.1**	**98.9 ± 0.1**	6733 / 429

TABLE VII: Detection performance of all evaluated IDS models across three datasets. **Red** = best; Blue = second best.

Dataset	Model	Acc.	Prec.	Rec.	F1	Train / Test (s)
CIC-DDoS2019	*Non-GAN Baselines*					
	Logistic Regression (calibrated)	88.9 ± 0.6	88.2 ± 0.6	87.6 ± 0.7	87.9 ± 0.6	38 / 6
	XGBoost (class weighted)	93.8 ± 0.5	93.3 ± 0.5	93.5 ± 0.4	93.4 ± 0.5	219 / 9
	LightGBM (class weighted)	94.6 ± 0.4	94.3 ± 0.4	94.4 ± 0.4	94.4 ± 0.4	191 / 8
	IDS-INT	87.4 ± 0.5	86.0 ± 0.5	84.5 ± 0.6	85.3 ± 0.5	25110 / 2481
	PSO-D-SEM	97.7 ± 0.2	97.2 ± 0.2	98.2 ± 0.2	97.7 ± 0.2	42890 / 2410
	FR-APPSO-BiLSTM	89.9 ± 0.4	89.2 ± 0.4	88.0 ± 0.5	88.6 ± 0.4	85400 / 4633
	HAGRU	87.7 ± 0.5	86.9 ± 0.5	85.0 ± 0.6	85.9 ± 0.5	51402 / 5180
	Transformer-based Models					
	TransfficFormer	92.7 ± 0.6	92.3 ± 0.6	92.0 ± 0.7	92.2 ± 0.6	1085 / 51
	FT-Transformer	94.0 ± 0.5	93.7 ± 0.5	93.8 ± 0.5	93.8 ± 0.5	1165 / 52
	TransTraffic	94.8 ± 0.4	94.6 ± 0.4	94.7 ± 0.4	94.6 ± 0.4	1278 / 59
	MalDetectFormer	94.5 ± 0.4	94.2 ± 0.4	94.3 ± 0.4	94.3 ± 0.4	1342 / 61
	GAN-based IDS					
	GAN	83.1 ± 0.5	81.5 ± 0.5	79.9 ± 0.5	80.7 ± 0.5	6984 / 1360
	DCGAN	84.5 ± 0.5	83.1 ± 0.5	81.1 ± 0.5	82.1 ± 0.5	34011 / 4445
	LSGAN	85.8 ± 0.4	84.5 ± 0.5	82.2 ± 0.5	83.3 ± 0.5	40955 / 2493
	WGAN	91.7 ± 0.3	90.9 ± 0.3	90.0 ± 0.3	90.5 ± 0.3	43233 / 2609
	SGAN-IDS	93.4 ± 0.3	92.5 ± 0.3	91.1 ± 0.4	91.8 ± 0.3	44730 / 3752
	SYN-GAN	90.4 ± 0.4	89.3 ± 0.4	87.9 ± 0.5	88.6 ± 0.4	44500 / 5213
	IGAN-IDS	89.2 ± 0.5	88.2 ± 0.5	86.4 ± 0.5	87.3 ± 0.5	32790 / 1693
	IDSGAN	88.2 ± 0.5	87.0 ± 0.5	85.2 ± 0.6	86.1 ± 0.5	7105 / 870
	FenceGAN	84.9 ± 0.4	84.0 ± 0.5	81.9 ± 0.5	83.0 ± 0.5	8381 / 1225
	BigGAN	93.9 ± 0.4	93.4 ± 0.4	93.3 ± 0.4	93.4 ± 0.4	49650 / 3150
	WGAN-GP	94.1 ± 0.4	93.7 ± 0.4	93.9 ± 0.4	93.8 ± 0.4	50500 / 2990
	TIPSO-GAN (ours)	**99.5 ± 0.1**	**98.8 ± 0.1**	**99.0 ± 0.1**	**98.7 ± 0.1**	6611 / 410

TABLE VIII: Generalization performance of TIPSO-GAN under strict zero-day evaluations. Left: leave-one-family-out (LOFO) on CIC-IDS2018. Right: cross-dataset evaluation where the model is trained on one corpus and tested on another. All values are mean ± std over five seeds.

LOFO on CIC-IDS2018 (per-family)					**Cross-Dataset Evaluation (Train → Test)**				
Held-out Family	Accuracy	Precision	Recall	F1	Train → Test	Accuracy	Precision	Recall	F1
DoS	91.5 ± 0.7	90.2 ± 0.8	92.7 ± 0.7	91.4 ± 0.7	CIC-IDS2018 → CIC-DDoS2019	82.4 ± 0.8	80.9 ± 0.9	83.1 ± 0.8	81.9 ± 0.8
Brute Force	93.8 ± 0.6	92.6 ± 0.6	94.2 ± 0.6	93.4 ± 0.6	CIC-IDS2018 → CICAPT-IIoT2024	80.6 ± 0.9	79.3 ± 1.0	81.0 ± 0.9	80.1 ± 0.9
Botnet	92.1 ± 0.7	91.0 ± 0.8	92.9 ± 0.7	91.9 ± 0.7	CIC-DDoS2019 → CIC-IDS2018	83.1 ± 0.7	81.5 ± 0.8	84.2 ± 0.8	82.8 ± 0.7
Infiltration	90.6 ± 0.8	89.2 ± 0.9	91.7 ± 0.8	90.4 ± 0.8	CIC-DDoS2019 → CICAPT-IIoT2024	81.2 ± 0.8	80.0 ± 0.9	82.1 ± 0.8	81.0 ± 0.8
Web Attacks	94.3 ± 0.6	93.0 ± 0.6	94.9 ± 0.6	93.9 ± 0.6	CICAPT-IIoT2024 → CIC-IDS2018	79.8 ± 0.9	78.5 ± 1.0	80.3 ± 0.9	79.4 ± 0.9
Heartbleed	93.0 ± 0.7	91.9 ± 0.7	93.8 ± 0.7	92.8 ± 0.7	CICAPT-IIoT2024 → CIC-DDoS2019	80.2 ± 0.8	78.9 ± 0.9	81.1 ± 0.9	80.0 ± 0.8
Mean across families	**92.6 ± 0.6**	**91.7 ± 0.7**	**93.4 ± 0.6**	**92.3 ± 0.6**	–				

TABLE IX: Computational complexity and resource usage of representative IDS models. E = epochs, N = samples, d = feature dimension, L = layers, P = swarm size, I = iterations. FLOPs are estimated per inference.

Model	Training Complexity	Inference Complexity	Memory (GB)	FLOPs/flow
Logistic Regression	$O(E \cdot N \cdot d)$	$O(d)$	< 0.5	10^5
XGBoost (weighted)	$O(E \cdot N \cdot d \log N)$	$O(d \log N)$	0.5	10^6
BiLSTM (FR-APPSO)	$O(E \cdot N \cdot d \cdot L)$	$O(d \cdot L)$	2.4	10^7
HAGRU	$O(E \cdot N \cdot d \cdot L)$	$O(d \cdot L)$	2.1	10^7
TransfficFormer	$O(E \cdot N \cdot d \cdot L)$	$O(d^2 \cdot L)$	3.0	10^8
MalDetectFormer	$O(E \cdot N \cdot d \cdot L)$	$O(d^2 \cdot L)$	3.5	10^8
WGAN-GP	$O(E \cdot N \cdot d \cdot L)$	$O(d \cdot L)$	2.4	10^7
FenceGAN	$O(E \cdot N \cdot d \cdot L)$	$O(d \cdot L)$	2.0	10^7
TIPSO-GAN	$O(E \cdot N \cdot d \cdot L + P \cdot I \cdot d)$	$O(d \cdot L + d^2)$	2.1	10^7

TABLE X: Runtime performance comparison.

Model	Train (GPU h)	Latency (ms)	Throughput	Mem (GB)	Power (W)
Logistic Regression	0.1	0.05	20,000	0.2	20
XGBoost (weighted)	0.3	0.20	5,000	0.5	40
LightGBM (weighted)	0.2	0.18	5,500	0.5	35
IDS-INT	6	0.80	1,250	1.2	70
PSO-D-SEM	10	1.10	900	2.0	100
FR-APPSO-BiLSTM	12	1.20	830	2.4	110
HAGRU	8	0.95	1,050	2.1	95
TransfficFormer	18	0.60	1,650	3.0	140
FT-Transformer	20	0.65	1,540	3.2	145
MalDetectFormer	24	0.72	1,390	3.5	160
GAN	4	0.35	2,850	1.5	120
WGAN	6	0.38	2,630	2.0	140
SGAN-IDS	8	0.51	1,960	1.8	140
WGAN-GP	9	0.48	2,080	2.4	150
TIPSO-GAN	14	0.42	2,400	2.1	150

TABLE XI: TIPSO-GAN runtime on increasing data sizes.

Data size (flows)	Thrpt (flows/s)	P50 (ms)	P90 (ms)	P99 (ms)	Mem (GB)	Power (W)
1×10^4	2420	0.17	0.27	0.40	2.0	148
5×10^4	2410	0.18	0.28	0.41	2.0	149
1×10^5	2400	0.18	0.29	0.42	2.1	150
5×10^5	2380	0.19	0.31	0.44	2.1	152
1×10^6	2360	0.20	0.32	0.46	2.2	153
5×10^6	2340	0.21	0.34	0.48	2.2	155
1×10^7	2320	0.22	0.35	0.50	2.3	156
5×10^7	2300	0.24	0.38	0.55	2.3	158
1×10^8	2280	0.25	0.40	0.60	2.4	160

vide a better trade-off: GAN and WGAN achieve 2.6–2.8k flows/s, while SGAN-IDS and WGAN-GP trade higher training cost for robustness. TIPSO-GAN requires 14 GPU hours

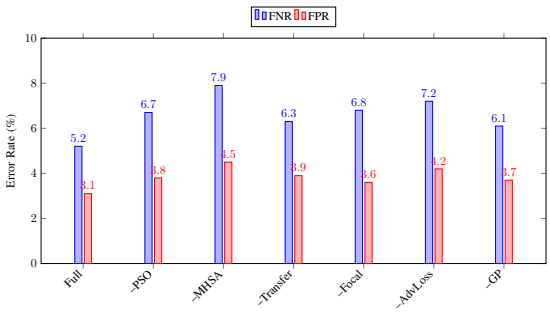


Fig. 8: Ablation of TIPSO-GAN components.

for training due to PSO optimization and multiple loss terms, but inference remains efficient at 0.42 ms per flow ($\approx$ 2400 flows/s). Peak memory (2.1 GB) and power draw (150 W) are comparable to WGAN-GP. At these rates, a single GPU can sustain 10–40 Gbps traffic, and throughput scales linearly across multiple GPUs. We observe that TIPSO-GAN maintains stable throughput and sub-millisecond latency as input size grows from 10^4 to 10^8 flows; detailed results are provided in Appendix Table XI. This confirms that TIPSO-GAN scales reliably to real-world high-volume traffic settings.

1) Runtime and Scalability Analysis: Table XI summarizes the runtime behavior of TIPSO-GAN as the dataset size increases from 1×10^4 to 1×10^8 network flows using the CICAPT-IIoT2024 dataset. The model maintains near-linear scalability, with throughput decreasing only marginally (2420 $\rightarrow$ 2280 flows/s) as data volume grows by four orders of magnitude. Median (P50) inference latency remains below 0.25 ms even at the largest scale, while memory usage stays within 2–2.4 GB. These results demonstrate that TIPSO-GAN achieves efficient inference with modest resource growth, confirming its suitability for large-scale real-time intrusion detection deployments.

I. Computational Complexity Analysis

Table IX reports complexity and resource usage across representative IDS models using the CICAPT-IIoT2024 dataset.

TABLE XII: Resource Utilization Across TIPSO-GAN Modules (CICAPT-IIoT2024)

Module	GPU Memory (GB)	FLOPs ($\times 10^7$)	Runtime / Epoch (s)	Latency (ms / flow)
Generator (G / G_T)	1.0	10.3	36	12.8
Discriminator (D / D_T)	0.9	9.8	34	12.4
DeePred (pre-trained)	1.2	10.7	39	13.1
PSO optimizer (train-time only)	0.3	1.2	11	–
Total (TIPSO-GAN)	**2.1**	**10^7**	**120**	**12.8**

Classical ML methods (Logistic Regression, XGBoost) are linear in d, with negligible memory and FLOPs, but provide limited robustness. RNN-based IDS (BiLSTM, HAGRU) scale as $O(d \cdot L)$ at inference, requiring 2–3 GB memory and 10^7 FLOPs. Transformer IDS (TransfficFormer, MalDetectFormer) incur quadratic $O(d^2 \cdot L)$ cost from attention, leading to the highest memory (3–3.5 GB) and 10^8 FLOPs. GAN baselines (WGAN-GP, FenceGAN) remain at $O(d \cdot L)$ with moderate memory (2–2.4 GB). TIPSO-GAN introduces $O(P \cdot I \cdot d)$ overhead during training from PSO, but maintains $O(d \cdot L + d^2)$ inference cost, with 2.1 GB memory and 10^7 FLOPs. This shows TIPSO-GAN is heavier to train but efficient at inference, close to GAN baselines and lighter than transformer IDS.

J. Ablation study

We performed an ablation study to quantify the contribution of each component in TIPSO-GAN (Fig. 8) using the CICAPT-IIoT2024 dataset. Removing MHSA led to the sharpest degradation, increasing FNR from 5.2% to nearly 8%. Excluding adversarial loss or focal loss also caused notable increases in error, confirming their role in stabilizing training and improving recall on rare attack classes. Transfer learning and PSO contributed moderate but consistent gains, while gradient penalty and regularization further reduced both FPR and FNR by about one point. The ablation demonstrates that the full TIPSO-GAN design yields the lowest error rates and that multiple components work synergistically to achieve robustness. As shown in Table XII, TIPSO-GAN maintains modest computational demands despite its multi-component design. Training consumes approximately 2.1 GB GPU memory and 10^7 FLOPs per epoch, with an average runtime of about 120 s. The PSO optimizer adds only $\approx$12% additional FLOPs during training, while inference latency remains 12.8 ms per flow comparable to WGAN-GP. DeePred and the discriminator account for most of the memory footprint, confirming that TIPSO-GAN delivers robust detection efficiency with limited computational overhead suitable for real-time deployment.

VII. Conclusion and Future Work

This work introduced TIPSO-GAN, a novel intrusion detection framework that redefines GAN training as a swarm optimization problem. The results demonstrate that TIPSO-GAN consistently outperforms thirteen baselines, under temporal splits and stability under strict zero-day regimes. The framework also exhibits superior calibration, robustness against adaptive adversarial attacks, and resilience to class imbalance, confirming its suitability for deployment in real-world network defense systems. The current implementation of TIPSO-GAN

operates at the flow level, where each record aggregates multiple packets belonging to the same TCP or UDP session. This design captures temporal and statistical dependencies that are not observable in isolated packets and enables stable adversarial training with compact feature spaces. However, this also means TIPSO-GAN does not perform per-packet classification. Future work will investigate the integration of fine-grained packet-level embeddings with TIPSO-GAN's adversarial optimization to enable real-time packet-wise detection without sacrificing stability. Future study could also extend TIPSO-GAN to streaming detection environments, optimize its efficiency for resource-constrained IoT deployments, and investigate cross-domain transfer to improve adaptability across heterogeneous network infrastructures.

ACKNOWLEDGMENTS

This work was partly supported by the National Natural Science Foundation of China (NSFC) (Grant nos. 62172194, 62202206 and U1836116), the Natural Science Foundation of Jiangsu Province, China (Grant no. BK20220515), the China Postdoctoral Science Foundation, China (Grant no. 2021M691310), and Qinglan Project of Jiangsu Province, China.

REFERENCES

[1] I. Cvitić, D. Perakovic, B. B. Gupta, and K.-K. R. Choo, "Boosting-based ddos detection in internet of things systems," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 2109–2123, 2022.

[2] J. Yang, X. Jiang, G. Liang, S. Li, and Z. Ma, "Malicious traffic identification with self-supervised contrastive learning," *Sensors*, vol. 23, 8 2023.

[3] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "Flowtransformer: A transformer framework for flow-based network intrusion detection systems," *Expert Systems with Applications*, vol. 241, p. 122564, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742303066X

[4] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2021.

[5] S. Zhang, Y. Fan, H. Zhou, and B. Li, "Maldetectformer: leveraging sparse spatiotemporal information for effective malicious traffic detection," in *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. [Online]. Available: https://doi.org/10.1609/aaai.v39i21.34411

[6] C. Kang, J. Yoon, D. Choi, E. Park, S. Pack, and J. Han, "Transtraffic: Predicting network traffic using low resource data," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, 2022, pp. 786–788.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, 6 2014. [Online]. Available: http://arxiv.org/abs/1406.2661

[8] Z. Moti, S. Hashemi, H. Karimipour, A. Dehghantanha, A. N. Jahromi, L. Abdi, and F. Alavi, "Generative adversarial network to detect unseen internet of things malware," *Ad Hoc Networks*, vol. 122, p. 102591, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870521001281

[9] H. D. Menéndez, S. Bhattacharya, D. Clark, and E. T. Barr, "The arms race: Adversarial search defeats entropy used to detect malware," *Expert Systems with Applications*, vol. 118, pp. 246–260, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418306535

[10] W. Qian, H. Li, and H. Mu, "Circular lbp prior-based enhanced gan for image style transfer," *Int. J. Semant. Web Inf. Syst.*, vol. 18, no. 2, p. 1–15, Dec. 2022. [Online]. Available: https://doi.org/10.4018/IJSWIS.315601

[11] X. Zhang, J. Wang, and S. Zhu, "Dual generative adversarial networks based unknown encryption ransomware attack detection," *IEEE Access*, vol. 10, pp. 900–913, 2022.

[12] Z. Zhou, Y. Li, J. Li, K. Yu, G. Kou, M. Wang, and B. B. Gupta, "Gan-siamese network for cross-domain vehicle re-identification in intelligent transport systems," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2779–2790, 2023.

[13] A. Dunmore, J. Jang-Jaccard, F. Sabrina, and J. Kwak, "A comprehensive survey of generative adversarial networks (gans) in cybersecurity intrusion detection," *IEEE Access*, vol. 11, pp. 76 071–76 094, 2023.

[14] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 214–223. [Online]. Available: https://proceedings.mlr.press/v70/arjovsky17a.html

[15] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2813–2821, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:206771128

[16] G. Dlamini and M. Fahim, "Dgm: a data generative model to improve minority class presence in anomaly detection domain," *Neural Computing and Applications*, vol. 33, pp. 13 635–13 646, 10 2021.

[17] R. Chapaneri and S. Shah, "Enhanced detection of imbalanced malicious network traffic with regularized generative adversarial networks," *Journal of Network and Computer Applications*, vol. 202, 6 2022.

[18] S. Das, "Fgan: Federated generative adversarial networks for anomaly detection in network traffic," 2022.

[19] X. Liu and J. Liu, "Malicious traffic detection combined deep neural network with hierarchical attention mechanism," *Scientific Reports*, vol. 11, no. 1, p. 12363, 2021.

[20] B.-E. Zolbayar, R. Sheatsley, P. McDaniel, M. J. Weisman, S. Zhu, S. Zhu, and S. Krishnamurthy, "Generating practical adversarial network traffic flows using nidsgan," 2022.

[21] B. Babayigit and M. Abubaker, "Towards a generalized hybrid deep learning model with optimized hyperparameters for malicious traffic detection in the industrial internet of things," *Engineering Applications of Artificial Intelligence*, vol. 128, p. 107515, 2 2024.

[22] M. Fathallah, M. Sakr, and S. Eletriby, "Stabilizing and improving training of generative adversarial networks through identity blocks and modified loss function," *IEEE Access*, vol. 11, pp. 43 276–43 285, 2023.

[23] Q. Yuan, G. Gou, Y. Zhu, Y. Zhu, G. Xiong, and Y. Wang, "Mcre: A unified framework for handling malicious traffic with noise labels based on multidimensional constraint representation," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 133–147, 2024.

[24] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv e-prints*, p. arXiv:1511.06434, Nov. 2015.

[25] S. Das, A. Abraham, and A. Konar, "Swarm intelligence algorithms in bioinformatics," in *Computational Intelligence in Bioinformatics*. Springer, 2008, pp. 113–147.

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," 1995, pp. 1942–1948.

[27] E. K. Boahen, B. E. Bouya-Moko, and C. Wang, "Network anomaly detection in a controlled environment based on an enhanced psogsarfc," *Computers and Security*, vol. 104, 5 2021.

[28] S. Choudhary, S. Sugumaran, A. Belazi, and A. A. A. El-Latif, "Linearly decreasing inertia weight pso and improved weight factor-based clustering algorithm for wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2023.

[29] M. A. K. Raiaan, S. Sakib, N. M. Fahad, A. A. Mamun, M. A. Rahman, S. Shatabda, and M. S. H. Mukta, "A systematic review of hyperparameter optimization techniques in convolutional neural networks," *Decision Analytics Journal*, vol. 11, p. 100470, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772662224000742

[30] L. Zhang and L. Zhao, "High-quality face image generation using particle swarm optimization-based generative adversarial networks," *Future Generation Computer Systems*, vol. 122, pp. 98–104, 2021.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21001126
[31] H. N. Nguyen, T. Lan-Phan, and C.-J. Song, "Generative adversarial network-based network intrusion detection system for supervisory control and data acquisition system," in *2024 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Nov 2024, pp. 1–3.
[32] T. K. Boppana and P. Bagade, "Gan-ae: An unsupervised intrusion detection system for mqtt networks," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105805, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622007953
[33] L. Zhang and L. Zhao, "High-quality face image generation using particle swarm optimization-based generative adversarial networks," *Future Generation Computer Systems*, vol. 122, pp. 98–104, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21001126
[34] L. Tian, Z. Wang, W. Liu, Y. Cheng, F. E. Alsaadi, and X. Liu, "Empower parameterized generative adversarial networks using a novel particle swarm optimizer: algorithms and applications," *International Journal of Machine Learning and Cybernetics*, vol. 13, pp. 1145–1155, 4 2022.
[35] R. Shameli and S. Rajkumar, "High-speed threat detection in 5g sdn with particle swarm optimizer integrated gru-driven generative adversarial network," *Scientific Reports*, vol. 15, no. 1, p. 10025, 2025. [Online]. Available: https://doi.org/10.1038/s41598-025-95011-z
[36] A. Touré, Y. Imine, A. Semnont, T. Delot, and A. Gallais, "A framework for detecting zero-day exploits in network flows," *Computer Networks*, vol. 248, p. 110476, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624003086
[37] M. Poongodi and M. Hamdi, "Intrusion detection system using distributed multilevel discriminator in gan for iot system," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 11, p. e4815, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4815
[38] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug 2018, pp. 1–6.
[39] S. Rahman, S. Pal, S. Mittal, T. Chawla, and C. Karmakar, "Syn-gan: A robust intrusion detection system using gan-based synthetic data for iot security," *Internet of Things*, vol. 26, p. 101212, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660524001537
[40] S. Aldhaheri and A. Alhuzali, "Sgan-ids: Self-attention-based generative adversarial network against intrusion detection systems," *Sensors*, vol. 23, no. 18, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/18/7796
[41] A.-G. Mari, D. Zinca, and V. Dobrota, "Development of a machine-learning intrusion detection system and testing of its performance using a generative adversarial network," *Sensors*, vol. 23, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1315
[42] M. Roopak, S. Parkinson, G. Y. Tian, Y. Ran, S. Khan, and B. Chandrasekaran, "An unsupervised approach for the detection of zero-day distributed denial of service attacks in internet of things networks," *IET Networks*, vol. 13, no. 5-6, pp. 513–527, 2024. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ntw2.12134
[43] M. Sarhan, S. Layeghy, M. Gallagher, and M. Portmann, "From zero-shot machine learning to zero-day attack detection," *International Journal of Information Security*, vol. 22, no. 4, pp. 947–959, 2023. [Online]. Available: https://doi.org/10.1007/s10207-023-00676-0
[44] R. Huang, L. Ma, J. He, and X. Chu, "T-gan: A deep learning framework for prediction of temporal complex networks with adaptive graph convolution and attention mechanism," *Displays*, vol. 68, p. 102023, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141938221000366
[45] S. Tuli, G. Casale, and N. R. Jennings, "Tranad: deep transformer networks for anomaly detection in multivariate time series data," *Proc. VLDB Endow.*, vol. 15, no. 6, p. 1201–1214, Feb. 2022. [Online]. Available: https://doi.org/10.14778/3514061.3514067
[46] W. Du, J. Xue, X. Yang, W. Guo, D. Gu, and W. Han, "Transfficformer: A novel transformer-based framework to generate evasive malicious traffic," *Knowledge-Based Systems*, vol. 319, p. 113546, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705125005921
[47] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4707749
[48] E. Ghiasvand, S. Ray, S. Iqbal, S. Dadkhah, and A. A. Ghorbani, "Cicapt-iiot: A provenance-based apt attack dataset for iiot environment," 2024. [Online]. Available: https://arxiv.org/abs/2407.11278
[49] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," *arXiv e-prints*, p. arXiv:1704.00028, Mar. 2017.
[50] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for nbsp;attack generation against intrusion detection," in *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part III*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 79–91. [Online]. Available: https://doi.org/10.1007/978-3-031-05981-0_7
[51] S. Huang and K. Lei, "Igan-ids: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks," *Ad Hoc Networks*, vol. 105, p. 102177, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870519311035
[52] S. Rahman, S. Pal, S. Mittal, T. Chawla, and C. Karmakar, "Syn-gan: A robust intrusion detection system using gan-based synthetic data for iot security," *Internet of Things*, vol. 26, p. 101212, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660524001537
[53] J. Donahue and K. Simonyan, *Large scale adversarial representation learning*. Red Hook, NY, USA: Curran Associates Inc., 2019.
[54] L. Niu, Z. Li, and S. Li, "Mmd fence gan unsupervised anomaly detection model based on maximum mean discrepancy," *Int. J. Cogn. Inform. Nat. Intell.*, vol. 18, no. 1, p. 1–13, jun 2024. [Online]. Available: https://doi.org/10.4018/IJCINI.344813
[55] F. Ullah, S. Ullah, G. Srivastava, and J. C.-W. Lin, "Ids-int: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic," *Digital Communications and Networks*, 3 2023.
[56] F. Rustam and A. D. Jurcut, "Malicious traffic detection in multi-environment networks using novel s-date and pso-d-sem approaches," *Computers and Security*, vol. 136, 1 2024.
[57] H. Jiang, S. Ji, G. He, and X. Li, "Network traffic anomaly detection model based on feature reduction and bidirectional lstm neural network optimization," *Scientific Programming*, vol. 2023, pp. 1–18, 11 2023.
[58] S. Liu, F. Peng, and K. Tang, "Reliable robustness evaluation via automatically constructed attack ensembles," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. [Online]. Available: https://doi.org/10.1609/aaai.v37i7.26064
[59] C. Liu, C. Lou, M. Yu, S. Yiu, K. Chow, G. Li, J. Jiang, and W. Huang, "A novel adversarial example detection method for malicious pdfs using multiple mutated classifiers," *Forensic Science International: Digital Investigation*, vol. 38, p. 301124, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666281721000226
[60] A. Chernikova and A. Oprea, "Fence: Feasible evasion attacks on neural networks in constrained environments," *ACM Trans. Priv. Secur.*, vol. 25, no. 4, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3544746
[61] K. G. Shreeharsha, C. G. Korde, M. H. Vasantha, and Y. B. N. Kumar, "Training of generative adversarial networks using particle swarm optimization algorithm." Institute of Electrical and Electronics Engineers Inc., 2021, pp. 127–130.
[62] X.-L. Li, R. Serra, and O. Julien, "Effects of the Particle Swarm Optimization parameters for structural dynamic monitoring of cantilever beam," in *Surveillance, Vishno and AVE conferences*. Lyon, France: INSA-Lyon, Université de Lyon, Jul. 2019. [Online]. Available: https://hal.science/hal-02188562

APPENDIX A
ADDITIONAL EXPERIMENTAL RESULTS

A. Experiment on the improved particle swarm inertia weight

The performance of five inertia weight strategies was assessed using a hybrid sinc–exponential test function (Eq. 15-

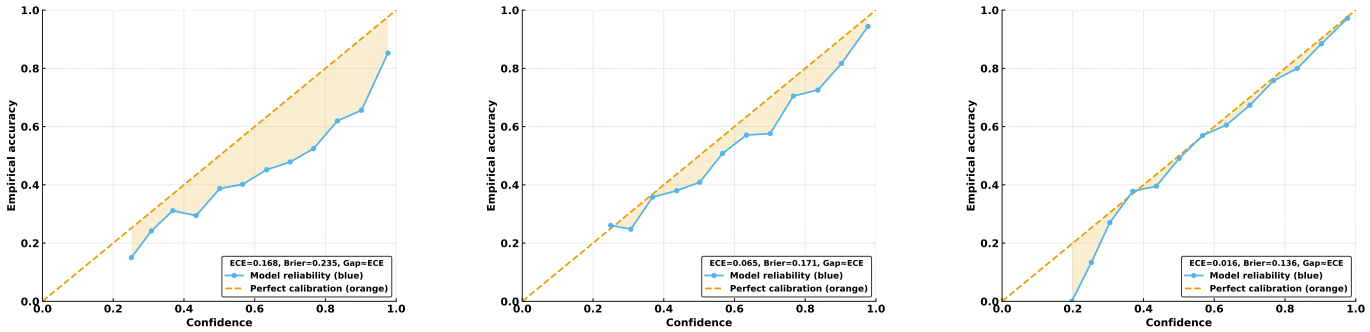


Fig. 9: Reliability diagrams on CICAPT-IIoT2024 with shaded miscalibration area. The orange dashed line denotes perfect calibration, and the blue curve shows the model's reliability. Legends report ECE and Brier scores, confirming that TIPSO-GAN achieves the best calibration.

2.71289), with results in Table XIII.

$$f(x,y) = \frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}} + \exp\left(\frac{\cos(2\pi x)+\cos(2\pi y)}{2}\right) \quad (15)$$

The proposed adaptive sigmoidal strategy achieved the best mean value (1.2183), finding all 50 global optima with no local minima. Its dynamic inertia control balanced exploration and exploitation more effectively than the fixed (ω_0) or decreasing (ω_1–ω_3) schemes. Fixed and linearly decreasing weights produced lower averages (0.90–0.93) and frequent local optima, while quadratic and exponential variants (0.81–0.79) over-exploited early, reducing diversity. Figure 10 confirms that adaptive weighting preserved population diversity and avoided premature convergence.

TABLE XIII: The comparative performance on different inertia weights using the customized test function.

Author/ Method	**Average** Value	**# of Local** Optima	**# of Global** Optima
[26] Conventional	0.9028	10	40
[28] Linear	0.9305	8	42
[30] Quadratic	0.8052	9	48
[61] Constant	0.7864	6	44
[30] Exponential	0.8136	6	44
Adaptive Sigmoidal	**1.2183**	**0**	**50**

The results indicate the importance of maintaining diversity in the particle swarm, as it allows the algorithm to explore the complex landscape more thoroughly and avoid being trapped in suboptimal regions.

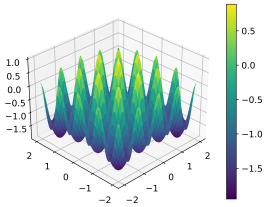


Fig. 10: The graph of test function $f(x)$.

TABLE XIV: Experimental Environment Summary.

Category	**Specification**
Processor	Intel Core i7 (8 cores, 3.6 GHz)
Memory	64 GB RAM
GPU	NVIDIA GTX 1080 Ti (11 GB VRAM)
Operating System	Windows 10 / Ubuntu 22.04
Python Version	3.10 (Anaconda)
Deep Learning Framework	TensorFlow 2.15.0
ML Utilities	Scikit-learn 1.3, NumPy, Pandas

B. Detailed experimental settings and parameters

TIPSO-GAN employs a two-phase optimization: PSO-based initialization followed by Adam fine-tuning. The PSO stage uses 100 particles ($c_1 = c_2 = 2.05$, $w_{\min} = 0.4$, $w_{\max} = 0.9$), optimizing the GAN loss for 1000 iterations with adaptive

inertia control [62]. The global best solution initializes the generator before Adam refinement. Training stops when validation accuracy falls below or exceeds 99.95% for three epochs, balancing convergence and overfitting. Hyperparameters (Table XV) follow prior work with validation tuning. Sensitivity tests varying MHSA heads (2–8), focal loss γ (0–3), and attention decay (0.90–0.98) confirmed stable accuracy and low false negatives (Fig. 11), demonstrating robustness and reproducibility across configurations.

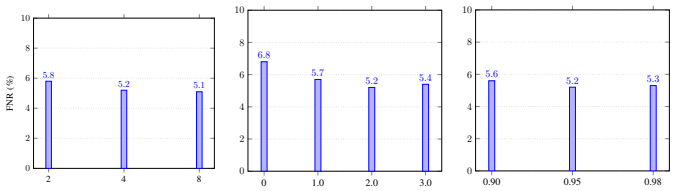


Fig. 11: sensitivity of tipso-gan to (left) mhsa heads, (middle) focal γ, and (right) attention decay. bars show mean fnr over 5 runs; lower is better.

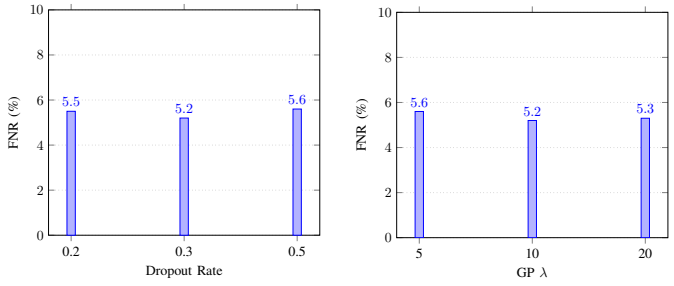


Fig. 12: Sensitivity of TIPSO-GAN to regularization and GAN stability: dropout rate (left) and gradient penalty λ (right). Mean FNR over 5 runs.

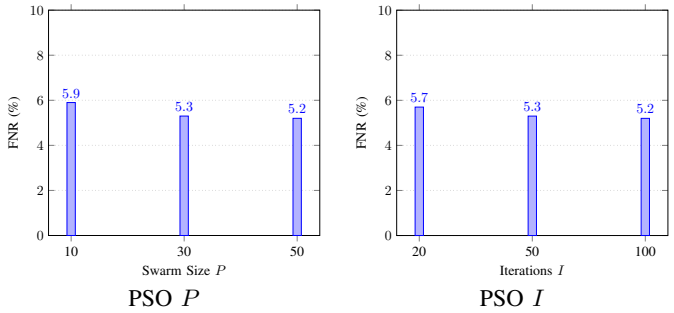


Fig. 13: Sensitivity of TIPSO-GAN to PSO parameters: swarm size P (left) and iterations I (right). Mean FNR over 5 runs.

TABLE XV: Parameter settings of TIPSO-GAN.

Component	Hyperparameter	Value
Optimization	Optimizer	Adam (β_1=0.5, β_2=0.999)
	Learning rate	2×10^{-4}
	Batch size	256
GAN Training	Discriminator updates per generator	5
	Gradient penalty λ	10
	Adversarial loss weight	1.0
DeePred (Pre-train)	Encoder depth	3 convolutional blocks
	Kernel size / stride	3×3 / 1×1
	Pooling	MaxPool (2)
	Activation	ReLU
	Optimizer	Adam (10^{-4})
	Epochs	50
DCGAN (PSOGAN base)	Conv layers (G/D)	4 / 4
	Latent dimension z	100
	Feature map base (G)	64
	Feature map base (D)	64
	Normalization	BatchNorm2d
	Activation	LeakyReLU (0.2)
TIPSO-GAN Generator (G_T)	Layers	4 ConvTranspose + BN + ReLU
	Output activation	Tanh
	Feature expansion	$128 \rightarrow 256 \rightarrow 512$
	Dropout rate	0.3
MHSA (Discriminator head)	Heads	4
	Hidden dim./head	32
	Attention decay factor	0.95
PSO (Training phase only)	Swarm size P	30
	Iterations I	50
	Inertia range $(w_{\max}, w_{\min})$	(0.9, 0.4)
	Cognitive/social coefficients (c_1, c_2)	(2.05, 2.05)
Regularization	Dropout rate	0.3
	Weight decay	10^{-4}

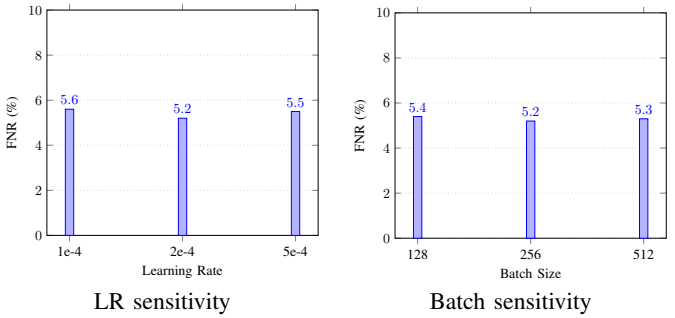


Fig. 14: Sensitivity of TIPSO-GAN to learning rate (left) and batch size (right). Values are mean FNR over 5 runs.

We evaluated the effect of additional training and regularization parameters on TIPSO-GAN. Figure 14 shows that varying the learning rate from 1×10^{-4} to 5×10^{-4} and the batch size from 128 to 512 changes the false negative rate (FNR) by less than one point. Figure 13 reports the PSO optimizer settings: swarm sizes between 30 and 50 and iterations of 50–100 achieve stable results, while smaller values slightly increase FNR. Figure 12 analyzes regularization and GAN stability; dropout rates of 0.2–0.5 and gradient penalty strengths $\lambda \in \{5, 10, 20\}$ yield only marginal shifts, with $\lambda = 10$ and dropout 0.3 giving the lowest error. Across all settings, TIPSO-GAN maintains low FNR, indicating robustness to hyperparameter choices and supporting reproducibility.

APPENDIX B
ARTIFACTS

A. Code Installation & Configuration

Installation Steps:

1) Create a Python 3.11 environment (Anaconda recommended).

TABLE XVI: Mapping of Paper Claims to Artifact Outputs.

Claim	Validated by Artifact
C1	run_repro_perf.py, run_compare_baselines.py → perf_summary_*.json, baselines_perf_*.json, confusion_matrix_*.json run_cost_profile.py → cost_metrics_*.json
C2	run_loss_curves.py → loss_history_*.csv, loss_curves_*.png
C3	run_adaptive_attacks.py → adaptive_attacks_report_*.json, adaptive_attacks_summary_*.csv, adaptive_attacks_*plots
C4	run_transfer.py → dee_transfer_report_*.json, transfer_*.png
C5	run_balance_eval.py → balance_grid_*.csv, preds_*.npy
A1	run_attention_ablation.py → attention_ablation_*.csv run_pso_ablation.py → pso_vs_static_*.csv
Cost	run_cost_profile.py → cost_metrics_*.json, cost_latency.png

Note: filenames with ⋆ stands for the dataset names used.

2) Run `pip install -r requirements.txt`.
3) Test setup: `python run_repro_perf.py`. Results appear under `artifacts/`.

B. Experiment Workflow

Each experiment is executed via a standalone Python script (`run_*.py`) that loads data, trains TIPSO-GAN, evaluates results, and saves outputs to `artifacts/`.

Workflow steps:

1) **Load data** using `cicids_loader.py` (default: `cicids2018.csv, cicddos2019.csv, cicaptiiot.csv`).
2) **Initialize model** components (PSO, generator, discriminator, DeePred) from `train.py`.
3) **Train & evaluate** the TIPSO-GAN framework with configurations in `config.py`.
4) **Store results** (metrics, confusion matrix, loss curves) in `artifacts/`.

To execute any experiment, open the project folder in **Visual Studio Code (VS Code)** and run the desired script (e.g., `run_repro_perf.py`) using the integrated terminal (right click on the script you want to execute and select "Run Python File in Terminal") or "Run Python File" option.

All scripts can alternatively be executed from the command line as: Example command:

```
python run_repro_perf.py
python run_ablation_pso.py etc.
```

Results are automatically saved under `artifacts/`. All scripts can be run independently or sequentially to reproduce results in the paper. VS Code or any terminal may be used; results appear automatically in `artifacts/`.

C. Major Claims

- (C1): TIPSO-GAN achieves superior detection metrics (Acc, Prec, Rec, F1) vs baselines. — (E1, E8) (Table VII).
- (C2): Stability and resilience of TIPSO-GAN against mode collapse. — (E2) Table III.
- (C3): DeePred transfer learning effects. — Fig. 8.

```python
def parse_args():
    p = argparse.ArgumentParser(
        description="Evaluate class balancing strategies across one or more datasets."
    )
    p.add_argument(
        "--data", "-d",
        nargs="+",
        default=["cicids2018.csv", "cicddos2019.csv", "cicaptiiot.csv"],
        help="One or more CIC-style CSV files, e.g. cicids2018.csv cicddos2019.csv cicaptiiot.csv",
    )
    return p.parse_args()
```

Fig. 15: Data configuration

TABLE XVII: Baseline Implementations and Code Repositories

Method	Repository URL
TIPSO-GAN	https://doi.org/10.5281/zenodo.17759516
DCGAN, WGAN-GP	https://github.com/Zeleni9/pytorch-wgan
FenceGAN	https://github.com/phuccuongngo99/Fence_GAN
BiGAN	https://github.com/ajbrock/BigGAN-PyTorch
SYN-GAN	https://github.com/dagrate/syngan
FT-Transformer	https://github.com/yandex-research/rtdl-revisiting-models

- (C4): PSO-based balancing improves recall. — Fig. 6.
- (C5): Model generalizes to unseen attack types. — Table VIII.
- (C6): PSO and attention modules provide additive gains. — added in Table X and Fig. 8.
- (C7): Computational cost comparable to baseline GANs. — Table X.
- (C8): TIPSO-GAN maintains robustness under adaptive attacks. — Table VI.

D. Evaluation

Table XVI summarizes all experiments (E1–E10), mapping each to its corresponding claim (C1–C8), expected outcome, and generated output files. The corresponding evaluation results reported in the paper are indicated in the major claims section (B-C).

E. Customization

Modify `config.py` to adjust epochs, dataset paths (see Fig 15 in each script), and verbosity. All scripts run with CPU by default; set TensorFlow GPU if available.

F. Code Availability

The implementation of the proposed method is available at https://doi.org/10.5281/zenodo.17759516. See Table XVII for full implementation codes of TIPSO-GAN and the baselines methods.