IF848 Detecção de Intrusão
Tabular Multi-Generator Generative Adversarial Network for Data Augmentation

1st Filip Fernandes
Centro de Informática - UFPE
Recife, Brazil
fapf@cin.ufpe.br

2nd Diana Rocha
Centro de Informática - UFPE
Recife, Brazil
drs4@cin.ufpe.br

3rd Antonio Bento
Centro de Informática - UFPE
Recife, Brazil
absn3@cin.ufpe.br

***Abstract*—O aumento exponencial de dispositivos de Internet das Coisas (IoT) tem trazido consigo desafios significativos em relação à segurança, dada sua disseminação abrangente, recursos limitados de proteção e suscetibilidade a ataques maliciosos. A implementação de tecnologias de detecção de intrusão torna-se crucial para salvaguardar as redes, no entanto, depara-se com o desafio intrínseco do desequilíbrio entre o tráfego regular e anômalo, o que compromete o desempenho de algoritmos baseados em aprendizado de máquina. Este estudo propõe uma análise crítica da Tabular Multi-Generator GAN, inicialmente apresentada como uma solução para essa problemática. O TMG-IDS emprega técnicas de data augmentation fundamentadas em Redes Adversariais Generativas (GAN) para equilibrar a distribuição de amostras. A análise concentra-se em três aspectos do modelo TMG-GAN: a estrutura multigeradora, que permite a síntese simultânea de diferentes tipos de ataques; a inclusão de um classificador para aprimorar o processo de treinamento; e a incorporação da similaridade cosseno como função de perda do gerador, com o intuito de aprimorar a qualidade dos dados sintéticos e reduzir a sobreposição entre classes de ataques gerados. Por meio de uma revisão detalhada e experimentação com o conjunto de dados KDD Cup 99, visando comparação com métricas de outros métodos propostos, pretende-se avaliar a eficácia do TMG-IDS e discutir possíveis limitações e direções futuras para a pesquisa nessa área.**

I. Introdução

Embora diversos métodos avançados de detecção de intrusão tenham sido propostos por trabalhos relacionados, persistem algumas limitações significativas. Primeiramente, a coleta de amostras de ataques eficazes na rede é desafiadora e consome consideráveis recursos humanos e materiais. Isso resulta em um número significativamente menor de amostras de ataques em comparação com as amostras normais no experimento, levando ao problema do desequilíbrio de classes no conjunto de dados. Diante de um número muito reduzido de amostras, é difícil alcançar uma classificação eficaz utilizando métodos tradicionais de processamento de dados desequilibrados e métodos de aprendizado profundo.

Para lidar com esses desafios, várias abordagens têm sido propostas, empregando tecnicas como o oversampling, implementado em modelos como SMOTE e ADASYN, que utilizam da inerpolação linear de amostras para sintetizar novas amostras da classe minoritária, o que pode ocasionar em problemas como introdução de ruído e sobreposição de classes. Há também o undersampling, com aplicação de data pruning, que descarta amostras da classe majoritária para equilibrar o conjunto de dados, podendo levar à perda de amostras valiosas.

Nesse contexto, surge o modelo Tabular Multi-Generator Generative Adversarial Network (TMG-GAN) [1] como uma solução promissora para abordar o desafio do desequilíbrio de dados na detecção de intrusão em redes. As principais contribuições do método proposto são:

- Estrutura multigeradora que permite a síntese simultânea de diferentes tipos de ataques, gerando um conjunto de dados mais abrangente e realista.
- Classificador, auxiliando na diferenciação entre dados reais e sintéticos durante o treinamento, agindo como filtro de ruídos e aprimorando a qualidade dos dados gerados.
- Extrator de features com emprego de similaridade cosseno como função de perda, minimizando a sobreposição entre classes de ataques gerados e aumentando a fidelidade dos dados sintéticos.
- Ampla aplicabilidade e possibilidade de integração com outros métodos, visto que não se limita ao algoritmo em si.

O presente artigo se dedica a realizar uma análise crítica do TMG-GAN, avaliando a efetividade do modelo e o impacto de suas inovações. Exploraremos os pressupostos teóricos do modelo e investigaremos seu desempenho em um novo cenário. Para tanto, conduziremos experimentos com o dataset KDD Cup 99, complementando os resultados originais e fornecendo insights sobre a generalização do modelo para diferentes conjuntos de dados. Os objetivos da análise incluem avaliar a efetividade do TMG-GAN na detecção de tipos de ataques com diferentes taxas de amostragem, analisar o impacto da estrutura multigeradora na qualidade dos dados sintéticos e na possibilidade de mode collapse e comparar o desempenho do TMG-GAN com outros métodos de detecção de intrusão.

TABLE I
MODELOS RELACIONADOS

Modelo	Método	Vantagens	Desvantagens
SMOTE	Oversampling	Redução do risco de overfitting Não há perda de informações Implementação simples	Introdução de ruídos Maior risco de overlapping Imprático para dados de alta dimensão
ADASYN		Considera distribuição de densidade de amostras Redução de viés introduzido por desequilibrio Ajuste adaptativo da fronteira de decisão da classificação	Maior risco de overlapping Amostras sintéticas muito similares às majoritarias Potencial geração de falsos positivos
MAGNETO	GAN	Aprimoramento na qualidade das amostras geradas Extração otimizada de features Aplicabilidade flexível	Suscetível a mode collapse Elevado custo computacional Complexidade
SNGAN		Aprimoramento na qualidade das amostras geradas Convergência mais rápida Maior generalização Estabilidade no treinamento	Dependência de arquitetura Elevado custo computacional Potencial perda de informação

II. TRABALHOS RELACIONADOS

Os métodos para lidar com o desbalanceamento de dados na detecção de intrusões podem ser categorizados em três abordagens principais: nível de dados, nível de algoritmo e métodos de aprendizado integrados.

A abordagem de nível de dados busca ajustar a distribuição de dados desbalanceados, geralmente por meio de oversampling [2] ou undersampling [3]. A técnica Synthetic Minority Over-sampling Technique (SMOTE) [4] é comumente utilizada para gerar novas amostras da classe minoritária, enquanto algoritmos como Adaptive Synthetic Sampling (ADASYN) [**?**] também são empregados para lidar com o desbalanceamento de forma adaptativa. No entanto, essas abordagens podem introduzir ruído nos dados sintéticos e podem ser computacionalmente intensivas.

No nível do algoritmo, o aprendizado sensível ao custo [6] é frequentemente empregado. Essa técnica atribui custos diferenciados aos erros de classificação das classes minoritárias e majoritárias, incentivando o classificador a dar mais importância às classes minoritárias. Embora eficaz, esse método pode ser sensível à escolha dos pesos dos custos e requer ajuste fino para cada conjunto de dados específico.

Os métodos de aprendizado integrados combinam várias técnicas para mitigar o desbalanceamento de dados. Um exemplo é a combinação de SMOTE com Random Forests ou Redes Neurais Profundas (DNNs), visando aproveitar as vantagens de cada método. No entanto, essa abordagem pode aumentar a complexidade do modelo e exigir um ajuste cuidadoso dos parâmetros.

As GANs [7] têm sido cada vez mais exploradas na geração de dados para lidar com o desbalanceamento em IDS. As Conditional GANs (CGANs), por exemplo, como a Auxiliary Classifier GAN (ACGAN) [8], mostraram-se eficazes na geração de dados sintéticos condicionados a classes específicas, o que pode ser útil para equilibrar a distribuição de classes em conjuntos de dados desbalanceados.

Além disso, as Spectrally Normalized GANs (SNGANs) [9] foram propostas como uma abordagem para melhorar a estabilidade do treinamento de GANs, mitigando problemas como o desaparecimento do gradiente e o mode collapse. Essa técnica ajuda a manter a diversidade das amostras geradas, o que é crucial para evitar o mode collapse, onde a GAN produz amostras altamente semelhantes ou repetitivas.

MAGNETO [10], uma estrutura de detecção de intrusões multiestágios, combina Convolutional Neural Networks (CNNs) [11] com GANs. Utiliza ACGAN para gerar novas amostras de dados, o que ajuda a superar o desbalanceamento de classes. No entanto, o treinamento de GANs pode ser desafiador devido à sua instabilidade, ao mode collapse e à necessidade de hiperparâmetros cuidadosamente ajustados.

III. SISTEMA PROPOSTO

O modelo proposto compreende quatro elementos-chave: gerador, discriminador, classificador e extrator de características. O gerador sintetiza dados artificiais, simulando os dados reais, enquanto o discriminador distingue entre os dados reais e os gerados. O classificador identifica a categoria dos dados (real ou gerado), enquanto o extrator de características captura informações relevantes antes da classificação.

A abordagem multigeradora permite a manipulação de diversos tipos de dados. O classificador atua como um filtro de ruído, categorizando amostras pseudo-geradas. O extrator de alta dimensão calcula a similaridade cosseno entre os dados originais, gerados e de diferentes tipos, o que é essencial para atualizar os parâmetros do gerador correspondente.

Símbolos:

- N: Número total de classes de dados
- X e $\tilde{X}$: Conjunto de amostras originais e geradas, respectivamente
- G_k: k-ésimo gerador
- D e C: Discriminador e classificador, respectivamente
- m: Número total de amostras no lote
- z: Ruído Gaussiano
- x_k e $\tilde{x}_k$: Amostras reais e geradas da k-ésima classe, respectivamente
- $F(x)$: Características de alta dimensão extraídas da amostra x na última camada oculta do classificador
- Pr e Pg: Distribuição original da amostra e distribuição da amostra gerada
- $O(F_1, F_2)$: Valor calculado da similaridade do cosseno

O treinamento do TMG-GAN segue a abordagem adversária comum em GANs. O gerador (G) e o discriminador (D) competem entre si para melhorar continuamente. O objetivo de G é maximizar a similaridade entre a distribuição dos dados gerados ($pg(z)$) e a distribuição dos dados reais ($pr(x)$). O objetivo de D é classificar corretamente a qual distribuição (real ou gerada) pertencem os dados de entrada. Segue a função usada para calcular G e D:

$$\min_G \max_D L(D,G) = E_{x\sim P_\mathrm{r}}[\log(D(x))] + E_{\tilde{x}\sim P_\mathrm{g}}[\log(1-D(\tilde{x}))]$$

onde x representa dados reais, $\tilde{x}$ representa dados de amostra pseudo-aleatória gerados pelo gerador G. Onde P_r é a distribuição de dados reais e P_g é a distribuição de dados gerativos definida implicitamente por $\tilde{x} = G(z)$, $z \sim P(z)$, para o qual z é amostrado a partir de uma distribuição de ruído simples P, como uniforme, normal ou gaussiana.

A última camada oculta do classificador (C) no TMG-GAN funciona como um extrator (F) de características de alta dimensão. Uma vez que C é totalmente treinado, essas características ajudam a diferenciar melhor as classes dos dados. Em outras palavras, a distribuição de amostras de classes diferentes pode ser identificada com mais eficácia.

Com base nisso, o objetivo é que a distribuição das amostras geradas para uma determinada classe se aproxime da distribuição original daquela classe e se afaste das distribuições de outras classes.

Para alcançar isso, é usada a seguinte equação:

$$O(F(\tilde{x}_k), F(x_k)) = \frac{|F(\tilde{x}_k) \cdot F(x_k)|}{\|F(\tilde{x}_k)\| \, \|F(x_k)\|}, \quad k \in \{1, \ldots, N\}$$

onde:

x_k é a amostra original correspondente;

$x_k \sim k$ é a amostra gerada pelo gerador $G_k(z)$;

$F(G_k(z))$ e $F(x_k)$ são as características de alta dimensão extraídas de ambas as amostras pelo extrator F;

$O(F_1, F_2)$ representa o valor da similaridade do cosseno calculada entre as duas amostras.

Similarmente ao caso anterior, podemos calcular a similaridade do cosseno entre a amostra gerada $\tilde{x}_k = G_k(z)$ e amostras geradas de outros tipos $\tilde{x}_j = G_j(z)$, onde j varia entre 1 e N, mas j é diferente de k. A equação usada para esse cálculo é:

$$O_k(F(\tilde{x}_k), F(\tilde{x}_j)) = \frac{1}{(N-1)} \sum_j \left[\frac{|F(\tilde{x}_k) \cdot F(\tilde{x}_j)|}{\|F(\tilde{x}_k)\| \, \|F(\tilde{x}_j)\|} \right]$$

onde:

$O_k(F(\tilde{x}_k), F(\tilde{x}_j))$ representa a similaridade do cosseno calculada;

N é o número total de classes;

j percorre todas as classes de 1 a N, exceto a classe k da amostra gerada $\tilde{x}_k$;

$F(\tilde{x}_k)$ e $F(\tilde{x}_j)$ são as características de alta dimensão extraídas, respectivamente, das amostras $\tilde{x}_k$ e $\tilde{x}_j$ pelo extrator F;

$\|F(\tilde{x}_k)\|$ e $\|F(\tilde{x}_j)\|$ representam as normas das características extraídas, respectivamente.

Como queremos que a distribuição da amostra gerada $\tilde{x}_k$ se aproxime da amostra original x_k, quanto maior a similaridade do cosseno $O_k(F(\tilde{x}_k), F(x_k))$, melhor. Por outro lado, queremos que a distribuição de $\tilde{x}_k$ não coincida com a distribuição de amostras geradas de outros tipos $\tilde{x}_j$, logo, quanto menor a similaridade do cosseno $O_k(F(\tilde{x}_k), F(\tilde{x}_j))$, melhor.

Baseado nisso, a função de perda baseada em similaridade do cosseno usada para atualizar o gerador G_k é definida como:

$$O_k = O_k(F(\tilde{x}_k), F(\tilde{x}_j)) - O_k(F(\tilde{x}_k), F(x_k)) \quad , \quad \{k, j\} \in \{1, \ldots, N\} \text{ e } j \neq k$$

onde:

O_k representa a perda final usada para treinar o gerador G_k;

k e j percorrem todas as classes de 1 a N, sendo j diferente de k.

Na TMG-GAN, é usado um conjunto de geradores G_1 a G_N para construir conjuntamente uma função de mapeamento que transforma ruído aleatório (z) em dados reais (X). Esse ruído segue uma distribuição Gaussiana específica ($p_z(z)$), e cada gerador G_k a utiliza para criar amostras e gerar uma distribuição própria ($p_g(x_{\sim k})$). A distribuição dos dados originais é representada por $p_r(x)$.

Similar ao GAN original, o TMG-GAN usa um jogo de min-max entre o gerador (G) e o discriminador (D). Os geradores G_1 a G_N são otimizados para minimizar uma perda, enquanto o discriminador D é otimizado para maximizar outra. Porém, diferentemente do GAN original, cada gerador no TMG-GAN é incentivado a focar apenas na geração de uma classe específica de amostras, evitando sobreposição com outras classes.

O classificador (C) identifica a classe das amostras de entrada e sua função de perda é usada para atualizar tanto o gerador (G_1 a G_N) quanto o discriminador (D). A função objetivo do TMG-GAN, baseada nessas perdas, é mostrada na equação (5).

$$\min_{G_0 \sim \mathcal{N}} \max_D = \frac{1}{N} \sum_{k=1}^{N} \left[E_{x_k \sim P_r}[D(x_k)] - E_{\tilde{x}_k \sim P_g}[D(\tilde{x}_k)] \right] - \left[E_{x_k \sim P_r}[C(x_k)] + E_{\tilde{x}_k \sim P_g}[C(\tilde{x}_k)] \right] + O_k$$

Na equação (5), o termo $\left[E_{x_k \sim p_r}[D(x_k)] - E_{\tilde{x}_k \sim p_g}[D(\tilde{x}_k)] \right]$ corresponde à função objetivo original do WGAN [12]. $C(x_k)$ representa a perda de classificação usada para atualizar o discriminador D, e $C(\tilde{x}_k)$ representa a perda de classificação usada para atualizar o gerador G_k. O_k representa o valor da similaridade do cosseno entre a amostra gerada ($\tilde{x}_k$) e outras amostras.

O classificador C usa cross-entropy loss como função objetivo, mostrada na equação (6). Nessa equação, y_{ik} e p_{ik} representam a classe verdadeira e a probabilidade prevista de a amostra i pertencer à classe k, respectivamente.

$$\mathcal{L}_C = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{N}[y_{ik}\log(p_{ik})]$$

Como podemos ver nas equações (7) e (8), as fórmulas de otimização para D e G_1 a G_N são derivadas da função objetivo principal (equação 5).

$$\max_{D} = \frac{1}{N}\sum_{k=1}^{N}\left[E_{x_k\sim P_r}[D(x_k)] - E_{\tilde{x}_k\sim P_g}[D(\tilde{x}_k)]\right] - [E_{x_k\sim P_r}[C(x_k)]]$$

$$\min_{G_0\sim\mathcal{N}} = \frac{1}{N}\sum_{k=1}^{N}\left[-E_{\tilde{x}_k\sim P_g}[D(\tilde{x}_k)] + E_{\tilde{x}_k\sim P_g}[C(\tilde{x}_k)] + O_k\right]$$

Além disso, o TMG-GAN se inspira na normalização de peso do SNGAN. Ao normalizar a matriz de pesos W, o gradiente é controlado para ser menor ou igual a 1, melhorando a estabilidade do treinamento e evitando o problema do mode collapse, comum em modelos GAN originais. A normalização da matriz de peso é mostrada na equação (9).

$$\bar{W} := \frac{W}{\sigma(W)}$$

Essa normalização garante que a norma espectral da matriz W seja sempre igual a 1, o que, por sua vez, limita o gradiente a um valor menor ou igual a 1. Isso contribui para solucionar problemas de instabilidade no treinamento e mode collapse observados em GANs originais.

O algoritmo 1 apresenta o pseudocódigo para o treinamento do TMG-GAN.

Etapas do treinamento:

- Amostragem de dados reais: Na linha 3, amostras de várias classes são coletadas a partir dos dados reais.
- Geração de amostras pseudo: Na linha 5, o gerador G cria amostras pseudo (artificiais) para várias classes.
- Treinamento do discriminador: O discriminador D é treinado usando amostras reais e pseudo para atualizar seus parâmetros (linha 6).
- Atualização do gerador: Na linha 10, um novo lote de amostras pseudo é gerado e utilizado, em conjunto com a função de perda baseada em similaridade do cosseno (Equação 4), para treinar o gerador G.
- Critério de parada: O treinamento termina quando o número máximo de iterações (t) é atingido (linha final).

Após o processo de treinamento, os geradores G1 a GN do TMG-GAN demonstram habilidade em modelar implicitamente a distribuição dos dados reais, permitindo a geração de pseudoamostras que mimetizam a distribuição das amostras reais positivas.

Algorithm 1 TMG-GAN Training

Require: Training sample set $X_1 \sim X_N$; training times t_d of discriminator D and training times t_g of generator $G_1 \sim G_N$ in each round of alternate training; round t of iterative training.

Ensure: Model parameters $(\theta_D, \theta_G, \theta_C)$.

1: $item \leftarrow t$
2: **while** $item = t$ **do**
3: **for** $p = 1$ to t_d **do**
4: Sampling samples of different classes: $(x_i^k)_{i=1}^m \sim \text{pr}(X_k)$
5: Sample $\{z^{(i)}\}_{i=1}^m \sim P_z$ a generated batch
6: Generate pseudo samples: $(\tilde{x}_i^k)_{i=1}^m = G_k(z^{(i)})$
7: $\nabla\theta_{(D,C)} = \frac{1}{mN}\sum_{i=1}^{m}\sum_{k=1}^{N}[D(x_i^k) - D(\tilde{x}_i^k) - C(x_i^k)]$
8: **end for**
9: **for** $q = 1$ to t_g **do**
10: Sample $\{z^{(i)}\}_{i=1}^m \sim P_z$ a generated batch
11: Generate pseudo samples: $(\tilde{x}_i^k)_{i=1}^m = G_k(z^{(i)})$
12: Calculate cosine similarity loss O_k using Eq. 4
13: $\nabla\theta_{(G_k)} = \frac{1}{mN}\sum_{i=1}^{m}\sum_{k=1}^{N}[-D(\tilde{x}_i^k) + C(\tilde{x}_i^k) + O_i^k]$
14: **end for**
15: $item \leftarrow item + 1$
16: **end while**
17: **return** $(\theta_D, \theta_G, \theta_C)$ =0

Uma característica vantajosa do TMG-GAN é sua capacidade de produzir amostras de ataque para múltiplas classes simultaneamente, empregando diversos geradores. Entretanto, é importante notar que, mesmo com essa funcionalidade, o processo de geração pode resultar em um pequeno número de amostras de ruído.

O artigo de referência define amostras ruidosas como aquelas capazes de induzir erros no classificador. Para filtrar amostras de alta qualidade dentre as geradas, utilizou-se o classificador treinado como um filtro de ruído.

Após o término do treinamento do TMG-GAN, o classificador C é treinado para classificar dados multiclasses com elevada precisão, tornando-se sensível à presença de amostras de ruído e, consequentemente, permitindo sua detecção e remoção. Após a geração de amostras por G_1 a G_N, representadas por $\tilde{X}_1$ a $\tilde{X}_N$ e rotuladas como $\tilde{Y}_1$ a $\tilde{Y}_N$, podemos filtrar aquelas que o classificador C tem dificuldade para classificar.

Cada amostra gerada $\tilde{x}_k$ é alimentada na rede C. Se $\tilde{x}_k$ pertence à classe gerada $\tilde{G}_k$ e o classificador C a classifica corretamente ($C(\tilde{x}_k) = \tilde{Y}_k$), então ela é considerada uma amostra fácil de classificar e mantida. Se $\tilde{x}_k$ pertence à classe gerada $\tilde{G}_k$, mas o classificador a classifica incorretamente ($C(\tilde{x}_k) \neq \tilde{Y}_k$), então ela é considerada uma amostra difícil de classificar e descartada.

Ao filtrar as amostras difíceis de classificar, o modelo almeja um conjunto final de amostras geradas com maior qualidade e confiabilidade para uso em tarefas posteriores. Uma questão a ser examinada é a possibilidade do classificador estar descartando amostras potencialmente valiosas,

que contribuiriam com a generalização e robustez do modelo, ao classificá-las incorretamente como ruidosas, especialmente considerando que o próprio classificador é treinado com um dataset desbalanceado.

Algorithm 2 Attack Sample Generation Based on TMG-GAN

Require: The number $n_{1:k}$ of various attack samples to be generated.

Ensure: The generated attack sample set $\tilde{X}_{1:N}$.

0: // Call TMG-GAN trained in algorithm 1 to generate pseudo sample data
0: Sample $\{z^{(i)}\}_{i=1}^{m} \sim P_z$ a generated batch
0: Generate pseudo sample: $(\tilde{x}_i^k)_{i=1}^{m} = G_k(z^{(i)})$
0: // Noise sample filtering
0: $t_1 = 0, t_2 = 0, \ldots, t_N = 0$
0: **while** $t_k = n_k$; $k = 1, 2, \ldots, N$ **do**
0: Using the trained classifier C to judge the label $C(\tilde{x}_k)$ of the pseudo sample $\tilde{x}_k$
0: **if** $\tilde{x}_k \in \tilde{G}_k$ and $C(\tilde{x}_k) = \tilde{Y}_k$ **then**
0: Add $\tilde{x}_k$ to $\tilde{X}_k$
0: $t_k = t_k + 1$
0: **end if**
0: **end while**
0: **Return** $\tilde{X}_{1:N}$ =0

IV. Possibilidades de otimização

Embora o artigo de apresentação da TMG-GAN tenha apresentado métricas de avaliação satisfatórias, demonstrando por meio de experimentos conduzidos nos datasets UNSW-NB15 e CICIDS2017 que o modelo supera o estado-da-arte em detecção de intrusão em ambas as classificações binária e multi-classes, possíveis aprimoramentos devem ser considerados, dentre eles, a incorporação de uma Deep Convolutional Generative Adversarial Network (DCGAN) [13], caracterizada pelo emprego de layers convolucionais em ambos o gerador e o discriminador.

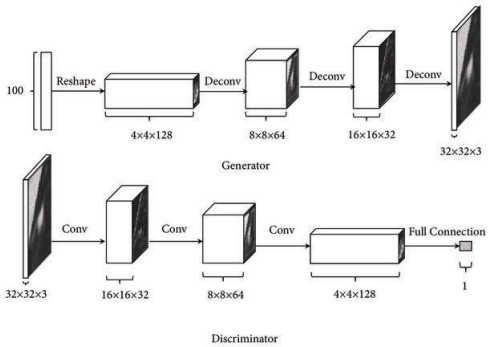

Fig. 1. Arquitetura da DCGAN

Essa arquitetura convolucional aplica filtros em regiões locais dos dados de entrada, permitindo que o modelo aprenda características em diferentes níveis de abstração. Devido à capacidade de capturar esses relacionamentos espaciais, as DCGANs se sobressaem na criação de novas amostras que se assemelham bastante aos dados de treinamento, o que se torna especialmente favorável se tratando de datasets precários. Além disso, a DCGAN inicializa os pesos usando uma distribuição Gaussiana com desvio padrão 0,02, promovendo mais estabilidade no treino e uma convergência mais eficiente.

Essas particularidades técnicas da DCGAN combinadas promovem sigificativa redução de riscos de mode collapse, um problema verificado durante nossos experimentos realizados na TMG-GAN. Embora a estrutura multi-geradora tenha sido arquitetada visando mitigar o mode collapse interclasses, parece ter sido negligenciado o possível viés dos geradores de classes que apresentam baixíssimas taxas de amostragem no conjunto de treino, resultando na criação de classes individuais incremenadas com amostras praticamente idênticas. Essa consideração é retificada pelos resultados dos experimentos apresentados pelo próprio artigo de referência, o qual descreve o significativo decaimento de desempenho do modelo quando lidando com classes minoritárias do conjunto de dados.

Sumarizando, os aprimoramentos ligados à incorporação da DCGAN ao modelo são:

- Melhora da estabilidade do treino
- Convergência mais eficiente
- Redução na propensão a mode collapse
- Geração de amostras com estruturas locais mais realistas

V. Metodologia

Diante da consideração de possíveis limitações do modelo e potenciais otimizações, faz-se necessária a experimentação do sistema com um novo dataset, a intuito de comparação e complementação aos resultados apresentados pelo artigo de referência, bem como de averiguar a aplicabilidade e generalização do modelo a outros conjuntos de dados, equiparar com trabalhos relacionados e validar a proposta de integração da DCGAN.

A. Dados usados pelo artigo de referência

O artigo de referência faz análises experimentais do modelo com os datasets UNSW-NB15 e CICIDS2017, dois reconhecidos datasets de detecção de intrusão. O conjunto de dados CICIDS2017 se constitui em 5 dias de coleta de dados, sendo que no primeiro dia foram coletados apenas dados benignos, e nos quatro dias seguintes foram incluídos ataques de Força Bruta em FTP e SSH, DoS, Heartbleed, Ataques Web, Infiltração, Botnet e DDoS. Cada registro no dataset possui 78 atributos de característica e 1 atributo de rótulo, incluindo classes normais e classes de ataques comuns.

Por outro lado, o dataset original UNSW-NB15 contempla 9 tipos de ataques. Cada registro é composto por 47 atributos de característica e 1 atributo de rótulo. O artigo de referência utiliza cinco tipos de dados para os experimentos, incluindo dados normais e quatro tipos de dados de ataque, sendo DoS e Reconhecimento ataques comuns no conjunto de dados e Shellcode e Worms ataques raros. As tabelas a seguir mostram

de forma detalhada a distribuição de ataques por dataset e as classes do UNSW-NB15 utilizadas na avaliação.

TABLE II
DATASET INFORMATION

Data sets	Label	Class	Training set	IR	OS
CICIDS2017	0	Benign	105222	-	105222
	1	DoS	21550	4.88	34609
	2	Port Scan	10809	9.73	23868
	3	Brute Force	5235	20.10	18294
	4	Web Attack	1476	71.19	14535
	5	Bot	857	121.78	13916
UNSW-NB15	0	Normal	56000	-	56000
	1	DoS	12264	4.57	22264
	2	Reconnaissance	10491	5.34	20491
	3	Shellcode	1133	49.43	11133
	4	Worms	130	430.77	10130

Note: "Label" represents the label assigned to each class of data; "Class" represents the class contained in the data set; "Training set" indicates the number of samples of each class in the training set; "IR" represents the ratio of normal samples to attack samples ; "OS" represents the number of instances in the training set after oversampling.

Fig. 2.

TABLE III
UNSW-NB15 TRAIN AND TEST DATA DISTRIBUTION

Class	Training Set	Percentage	Test Set	Percentage
Analysis	2,000	1.141%	677	0.822%
Backdoor	1,746	0.996%	583	0.708%
DoS	12,264	6.994%	4,089	4.966%
Exploits	33,393	19.045%	11,132	13.521%
Fuzzers	18,184	10.371%	6,062	7.363%
Generic	40,000	22.813%	18,871	22.921%
Normal	56,000	31.938%	37,000	44.940%
Reconnaissance	10,491	5.983%	3,496	4.246%
Shell Code	1,133	0.646%	378	0.459%
Worms	130	0.074%	44	0.053%
Total	175,341	100%	82,332	100%

Fig. 3.

B. Novo conjunto de dados escolhido

A intuito de comparação, escolhemos o dataset KDDcup99, criado pela DARPA em 1999, utilizando o tráfego de rede registrado no conjunto de dados de 1998. Ele foi pré-processado em 41 características por conexão de rede. Existem quatro categorias principais de ataques: DoS (negação de serviço), R2L (acesso não autorizado de uma máquina remota), U2R (acesso não autorizado à raiz) e Probe (sondagem).

Análises estatísticas do dataset concluiram que o conjunto de dados KDD apresenta dois problemas críticos que afetam profundamente o desempenho do sistema. O problema mais significativo é a grande quantidade de registros replicados. Foi descoberto que cerca de 78% e 75% dos registros são duplicados nos conjuntos de dados de treinamento e teste,

TABLE II
DISTRIBUIÇÃO DO CONJUNTO DE DADOS

Ataque	**Treino**	**%**	**Validação**	**%**
normal	61.509	61,931	13.137	61.729
neptune	36.274	36,523	7.805	36.672
satan	640	0,644	122	0.573
ipsweep	435	0,437	106	0.498
portsweep	281	0,282	76	0.357
pod	144	0,144	30	0.140
guess password	35	0,035	6	0.028
Total	**99318**	**100%**	**21282**	**100%**

Ataque	**Teste**	**%**
normal	13.184	61,949
neptune	7741	36,373
satan	144	0,676
ipsweep	110	0,516
portsweep	59	0,277
pod	32	0,150
guess password	12	0,056
Total	**99318**	**100%**

respectivamente, o que o torna adequado para nossa análise do desempenho da TMG-GAN em datasets desbalanceados. Utilizamos na pesquisa uma versão de 10% do KDDCup99, contendo dados normais e os ataques IP Sweep, Port Sweep(Port Scanning), Satan(Ransomware), Neptune(DoS), PoD(DoS) e brute force.

C. Métricas de avaliação

Os resultados dos experimentos foram analisados segundo as métricas de avaliação f1-score, recall e precisão, objetivando uma verificação minuciosa de eficácia e confiabilidade.

VI. RESULTADOS E DISCUSSÕES

A. Resultados reprodução do artigo de referência

Para avaliar a capacidade do TMG-IDS na detecção de tráfego normal e anormal, foram realizados experimentos em dois conjuntos de dados distintos: CICIDS2017 e UNSW-NB15. A classificação binária foi utilizada, considerando cinco tipos de tráfego anormal como amostras de ataque. As métricas de avaliação foram Precisão, Recall e F1-score, e os modelos comparados foram TMG-IDS, TMG-GAN, MAGENTO, SNGAN, TACGAN e DNN.

No conjunto de dados CICIDS2017, o TMG-IDS apresentou o melhor desempenho na detecção de dados normais e ataques, com uma média de 0.9973 em Precisão, Recall e F1-score. O TMG-GAN também obteve bons resultados, com melhorias de 2.12%, 2.13% e 2.13% em Precisão, Recall e F1-score, respectivamente, em comparação com o MAGENTO. O TMG-IDS também superou o DNN em 7.85%, 7.86% e 7.86% em Precisão, Recall e F1-score, respectivamente.

No conjunto de dados UNSW-NB15, os modelos geradores profundos SNGAN, MAGENTO e TACGAN apresentaram melhorias significativas. O TMG-IDS novamente se destacou, com aumentos de 8.18%, 3.46% e 7.20% em Precisão, Recall e F1-score, respectivamente, em relação ao DNN.

BINARY CLASSIFICATION RESULTS (CICIDS2017)

Data	DNN			ADASYN			SNGAN		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Normal	0.9139	0.9247	0.9193	0.9239	0.9485	0.9361	0.9266	0.9737	0.9496
Attack	0.9237	0.9126	0.9181	0.9470	0.9217	0.9342	0.9722	0.9227	0.9468
Macro-	0.9188	0.9187	0.9187	0.9355	0.9351	0.9351	0.9494	0.9482	0.9482
Data	MAGENTO			TACGAN-IDS			TMG-IDS		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Normal	0.9703	0.9821	0.9762	0.9940	0.9390	0.9657	0.9974	0.9973	0.9973
Attack	0.9818	0.9698	0.9758	0.9420	0.9943	0.9674	0.9973	0.9974	0.9973
Macro-	0.9761	0.9760	0.9760	0.9680	0.9666	0.9666	**0.9973**	**0.9973**	**0.9973**

BINARY CLASSIFICATION RESULTS (UNSW-NB15)

Data	DNN			ADASYN			SNGAN		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Normal	0.9935	0.8852	0.9362	0.9931	0.8768	0.9313	0.9960	0.9176	0.9552
Attack	0.6473	0.9731	0.7775	0.6306	0.9719	0.7649	0.7209	0.9831	0.8319
Macro-	0.8204	0.9292	0.8568	0.8119	0.9244	0.8481	0.8585	0.9504	0.8936
Data	MAGENTO			TACGAN-IDS			TMG-IDS		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Normal	0.9967	0.9381	0.9665	0.9957	0.9302	0.9619	0.9949	0.9503	0.9720
Attack	0.7750	0.9858	0.8677	0.7527	0.9816	0.8520	0.8096	0.9773	0.8856
Macro-	0.8858	0.9619	0.9170	0.8742	0.9559	0.9069	**0.9022**	**0.9638**	**0.9288**

Fig. 4.

TABLE III
BINARY CLASSIFICATION RESULTS (KDDCUP99)

Modelo	Precisão	Recall	F1- score
SMOTE	0.9976	0.9398	0.9678
DNN	0.9748	0.9998	0.9871
TMG-GAN	0.9599	0.9794	0.9695
TMG-GAN + DCGAN	0.9993	0.9891	0.9942

B. Resultados no dataset escolhido

A intuito de comparação, o dataset KDDCup99 foi utilizado nos modelos SMOTE, DNN, TMG-GAN original e na TMG-GAN com DCGAN implementada. Diferentemente dos resultados obtidos com os datasets anteriores, no conjunto de 10% do KDD a TMG-GAN não ultrapassou os modelos SMOTE e DNN em precisão, apresentando uma queda de 3,74% em relação à média macro da precisão alcançada na classificação binária com o dataset CICIDS-2017. Nos dados obtidos a TMG-GAN ultrapassa SMOTE apenas em recall e f1-score, por 3,96% e 0,17%, respectivamente, e não utrapassa a DNN em nenhuma das métricas. Também avaliamos o desempenho da TMG-GAN quando associada à DCGAN, e obtivemos dados que ultrapasaram todas as outras métricas dos modelos comparados, com exceção do recall da DNN, que sobressai em relação ao recall da TMG otimizada por 1,07%. Comparada com a TMG-GAN proposta pelo artigo de referência, a TMG-DCGAN mostrou um aumento em precisão, recall e f1-score de 3,94%, 0,97% e 2,4%, respectivamente.

VII. CONCLUSÕES E TRABALHOS FUTUROS

A queda no desempenho da TMG-GAN em relação aos modelos SMOTE e DNN ao mudar o dataset de CICDS-2017 para o KDDCup99 traz questionamentos sobre a capacidade de generalização do modelo proposto, bem como da sua eficácia e aplicabilidade em datasets com desequilíbrios mais significativos. No entanto, essas limitações podem ser ultrapassadas com a integração de uma DCGAN, como verificado nos resultados dos experimentos. Análises futuras podem ser realizadas a fim de verificar outras questões na funcionalidade do modelo, dentre elas a configuração do classificador, que descarta amostras errôneamente classificadas como amostras ruidosas, sem considerar que talvez elas possam contribuir para com a capacidade de generalização do modelo.

REFERÊNCIAS

[1] H. Ding, Y. Sun, N. Huang, Z. Shen and X. Cui, *TMG-GAN: Generative Adversarial Networks-Based Imbalanced Learning for Network Intrusion Detection*, IEEE Transactions on Information Forensics and Security, vol. 19, pp. 1156-1167, 2024.

[2] X. Tao et al., *SVDD-based weighted oversampling technique for imbalanced and overlapped dataset learning*, Inf. Sci., vol. 588, pp. 13-51, Apr. 2022.

[3] R. Zhang, Z. Zhang and D. Wang, *RFCL: A new under-sampling method of reducing the degree of imbalance and overlap*, Pattern Anal. Appl., vol. 24, no. 2, pp. 641-654, May 2021.

[4] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[5] Haibo He and Yunqian Ma. Imbalanced learning with synthetic minority over-sampling technique. In *International conference on intelligent computing*, pages 878–887. Springer, 2008.

[6] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, pages 973–978. Morgan Kaufmann Publishers Inc., 2001.

[7] I. Goodfellow et al., *Generative adversarial nets*, Proc. Adv. Neural Inf. Process. Syst., no. 27, pp. 1-9, 2014.

[8] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

[9] T. Miyato, T. Kataoka, M. Koyama and Y. Yoshida, *Spectral normalization for generative adversarial networks*, Proc. Int. Conf. Learn. Represent., pp. 1-10, 2018.

[10] G. Andresini, A. Appice, L. De Rose and D. Malerba, *GAN augmentation to deal with imbalance in imaging-based intrusion detection*, Future Gener. Comput. Syst., vol. 123, pp. 108-127, Oct. 2021.

[11] R. V. Mendonça et al., *Intrusion detection system based on fast hierarchical deep convolutional neural network*, IEEE Access, vol. 9, pp. 61024-61034, 2021.

[12] M. Arjovsky, S. Chintala and L. Bottou, *Wasserstein generative adversarial networks*, Proc. Int. Conf. Mach. Learn., pp. 214-223, 2017.

[13] Alec Radford, Luke Metz, and Soumith Chintala, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, 2016, arXiv:1511.06434 [cs.LG].