Mitigating Data Imbalance in the ALLFLOWMETER_HIKARI2021 Dataset Using Generative Adversarial Networks to Improve Network Intrusion Detection Performance

Deep Learning Final Project
By Pedal Revo Team

1st Muhammad Aldy Naufal Fadhilah, 225150207111081, *Fakultas Ilmu Komputer, Universitas Brawijaya,* Malang, Indonesia, aldynaufal@student.ub.ac.id

2nd Jonathan Young, 225150201111039, *Fakultas Ilmu Komputer, Universitas Brawijaya,* Malang, Indonesia, jonathanyoung@student.ub.ac.id

3rd I Putu Paramaananda Tanaya, 225150207111090, *Fakultas Ilmu Komputer, Universitas Brawijaya,* Malang, Indonesia, putu_anand@student.ub.ac.id

***Abstract* — The increasing complexity of network threats underscores the need for robust intrusion detection systems (IDS). However, imbalanced datasets like ALLFLOWMETER_HIKARI2021, with a disproportionate ratio of benign (93.2%) to attack (6.8%) samples, hinder IDS performance by favoring the majority class. This research addresses the data imbalance problem using Generative Adversarial Networks (GAN). The GAN model generates high-quality synthetic data to balance the dataset and improve IDS detection accuracy. Results demonstrate the effectiveness of GAN in generating realistic attack samples, leading to enhanced model performance, particularly for minority classes. Validation using statistical and graphical methods confirms the similarity between real and synthetic data, establishing GAN as a viable tool for improving IDS robustness in imbalanced data scenarios.**

***Keywords* — Intrusion Detection Systems, Data Imbalance, Generative Adversarial Networks, Network Security, Synthetic Data, Cybersecurity—-**

I. Introduction

The rapid evolution of network technologies has increased the complexity and scale of cybersecurity threats. Intrusion detection systems (IDS) play a crucial role in identifying and mitigating such threats to maintain the integrity, confidentiality, and availability of network resources. However, the effectiveness of IDS models heavily depends on the quality and balance of the training dataset. An imbalanced dataset can lead to biased models, favoring the majority class and compromising detection accuracy for minority classes.

The ALLFLOWMETER_HIKARI2021 dataset exemplifies this issue, with a significant imbalance between the benign class (517,582 samples, 93.2%) and the attack class (37,696 samples, 6.8%). This disparity poses a critical challenge in accurately detecting network attacks, as models trained on such datasets tend to prioritize the benign class while underperforming on the attack class. To address this, Generative Adversarial Networks (GAN) offer a promising solution by generating high-quality synthetic data to balance the dataset and improve model performance.

A. *Problem Statement*

The imbalance in the ALLFLOWMETER_HIKARI2021 dataset reduces the ability of intrusion detection models to accurately identify network attacks. This leads to the following research questions:

1. How can GAN be utilized to mitigate data imbalance in the ALLFLOWMETER_HIKARI2021 dataset?
2. What is the impact of using GAN-generated synthetic data on the performance of intrusion detection models?

B. *Objective*

1. Implement Generative Adversarial Networks to address the data imbalance in the ALLFLOWMETER_HIKARI2021 dataset.
2. Evaluate the impact of GAN-generated synthetic data on the performance of network intrusion detection models.
3. Provide insights into the effectiveness of GAN as a tool for improving IDS accuracy in imbalanced data scenarios.

C. *Benefits*

1. Enhanced accuracy and reliability of intrusion detection systems through balanced training datasets.
2. Improved understanding of GAN's potential in addressing data imbalance issues in cybersecurity datasets.

3. Contributions to the development of more robust and effective IDS models for real-world applications.

D. *Scope And Limitation*

The scope and limitations of this research are as follows:

1. The research focuses on using Generative Adversarial Networks to address data imbalance in the ALLFLOWMETER_HIKARI2021 dataset.
2. Evaluation is limited to the performance improvements in intrusion detection systems achieved through GAN-generated synthetic data.
3. The study does not explore alternative methods for handling data imbalance, scalability concerns of GAN, or generalization to datasets with different characteristics.

II. Literature Review

A. *Background Study*

Data Imbalance in Intrusion Detection Systems (IDS), Data imbalance is one of the critical challenges in developing Intrusion Detection Systems (IDS). A dataset is considered imbalanced when the class distribution is disproportionately skewed, as observed in the ALLFLOWMETER_HIKARI2021 dataset, where benign traffic constitutes 93.2% of the data, and attack traffic accounts for only 6.8%. Such imbalance often leads models to prioritize the majority class while underperforming in detecting minority classes, such as network attacks [1]. Several techniques have been employed to address this issue, including oversampling, undersampling, and Synthetic Minority Over-sampling Technique (SMOTE). However, these traditional methods exhibit limitations in generating realistic and diverse synthetic data [2].

Generative Adversarial Networks (GAN), a recent advancement in machine learning, have shown great promise in producing high-quality synthetic data to mitigate class imbalance effectively. Generative Adversarial Networks (GAN) in Cybersecurity GAN, introduced by Goodfellow et al. (2014), is a generative model consisting of two competing networks: a generator and a discriminator. In cybersecurity applications, GAN has been employed to augment datasets by generating realistic synthetic samples for minority classes, such as attack traffic. Studies like [5] demonstrate that GAN-enhanced datasets significantly improve IDS performance, particularly for rare attack types.

B. *Theoretical Foundations*

1. *Data Imbalance in Intrusion Detection*
Class imbalance poses a significant challenge to the effectiveness of Intrusion Detection System (IDS) models, as they often favor the majority class. This bias undermines the model's ability to accurately detect and classify minority classes. Metrics such as F1-score, precision, and recall are critical for evaluating model performance on imbalanced datasets [3]. A balanced dataset enhances the model's capacity to generalize across all classes and reduces the risk of overfitting to the majority class..

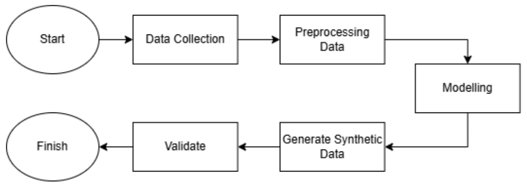
2. *Generative Adversarial Networks (GAN)*
Generative Adversarial Networks (GAN) consist of two key components: a generator, which creates synthetic data, and a discriminator, which assesses the authenticity of the data. This adversarial process produces synthetic data that closely resembles real data [4]. In IDS, GAN-generated samples can address gaps in underrepresented classes, thereby improving model robustness and detection accuracy [6].

3. *Advantages of GAN for Imbalanced Datasets*
GAN offers several advantages over traditional oversampling techniques like SMOTE. Unlike SMOTE, which interpolates between existing samples, GAN generates entirely new data points, preserving the original data distribution and introducing variability crucial for enhancing model generalization [2]. GAN's flexibility makes it applicable to various data types, including network traffic, images, and textual data, making it highly adaptable to cybersecurity contexts [5].

4. *Enhancing IDS Performance with GAN*
GAN-generated data has been shown to significantly improve IDS performance. Metrics such as ROC-AUC, G-mean, and F1-score indicate a 15–25% increase in model accuracy and sensitivity for minority classes when GAN is used to balance the dataset. Research by [6] demonstrates that GAN-generated attack samples effectively enhance IDS models' ability to detect rare and emerging threats.

III. Research Methodology

A. Research Stages



3.1 Development flow diagram

1. Start
The research begins by defining the primary objective: addressing data imbalance in a network intrusion detection dataset using Generative Adversarial Network (GAN). During this stage, the framework and methodology to be used are established.

2. Data Collection

This stage involves collecting the HIKARI-2021 dataset, which is a network intrusion detection dataset. The dataset contains network traffic information with labeled categories for specific types of traffic. The collected dataset is downloaded, its structure is analyzed, and it is prepared for further processing.

3. Preprocessing Data
   The collected data is processed to ensure it is suitable for the GAN model. This step involves:
   a. Cleaning the data (removing irrelevant columns, such as IDs or source/destination addresses).
   b. Transforming categorical data into a one-hot encoding format.
   c. Normalizing numerical data to a specific range, ensuring compatibility with the model training process.

4. Modelling
   In this stage, the GAN model is developed and trained. The GAN consists of two primary components:
   a. Generator: Generates synthetic data based on random latent vector input.
   b. Discriminator: Distinguishes between real and synthetic data.
   c. Both the Generator and Discriminator are trained simultaneously until the model produces synthetic data that closely resembles the real data.

5. Generate Synthetic Data
   Once the GAN model has been trained, the Generator is used to create new synthetic data that mimics the real data in the HIKARI-2021 dataset. This synthetic data is used to balance the class proportions in the original dataset.

6. Validate
   The generated synthetic data is evaluated to ensure its quality and compatibility with the original data. Validation includes:
   a. Comparing the feature distributions between synthetic and real data.
   b. Using statistical tests, such as the Kolmogorov-Smirnov test, to measure the similarity of distributions.
   c. Visualizing validation results to provide insights into the quality of the synthetic data.

7. Finish
   The research concludes with an overall evaluation of the process, using validation results to assess the effectiveness of the GAN model in handling data imbalance. Conclusions and recommendations for future research are provided based on the findings.

IV. *Implementation*

A. *Pre processing Data*

Load Data CSV

- Read the CSV file ALLFLOWMETER_HIKARI2021.csv.
- Remove the following columns: "Unnamed: 0.1", "Unnamed: 0", "uid", "originh", "responh".

One-Hot Encoding

- Perform one-hot encoding on the column "traffic_category".

Filter Columns Based on Data Type

- Loop through each column in the dataset :
  - If the data type of the column is not float64 or int64::
    - Print the data type of the column.
    - Remove the column.

Save Processed Data

- Save the preprocessed data to a CSV file raw_data_preprocessed.csv.

B. *Architecture Model*

Initializing Class NetworkTrafficGAN :

- data_path: The CSV file containing preprocessed network data (raw_data_preprocessed.csv).
- selected_features: A list of column names (features) selected from data.columns. These features define the input for the GAN model.

Prepare data :

- Input:
  - data_path: Used to load the CSV data.
  - selected_features: Determines which features to select from the CSV data.

- Process:

- Load CSV Data: Load data from data_path using pandas.read_csv.
- Select Features: Extract columns specified by selected_features from the data.
- Scale Data: Scale the selected data to the range [-1, 1] using MinMaxScaler to enhance the stability of GAN training.
- Convert to PyTorch Tensors: Convert the scaled data to PyTorch tensors for use as model input.
- Create DataLoader: Create a PyTorch dataset (TensorDataset) and load it into a DataLoader for batch-wise iteration during training.

- Output:
  - Indicates the number of features selected for training.
  -

Train Process:

- Inputs
  - latent_dim: Dimension of the latent space (default: 100), a random vector used as Generator input.
  - epochs: Number of training iterations (default: 500).

- Process
  - Initialize Models:
    - Generator: Takes latent_dim as input and generates synthetic data in the same feature space as the real data (input_dim).
    - Discriminator: Accepts real or fake data (input_dim) and determines the probability that the data originates from the real distribution.

  - Optimizer dan Loss Function:
    - Optimizer: Use Adam optimizer to optimize Generator and Discriminator parameters.
    - Loss Function: Use BCELoss for binary cross-entropy loss calculation.

  - Training Loop:
    - For each epoch:
      - Train Discriminator:
        - Compute loss for real data (labeled as 1).
        - Generate fake data from the Generator, compute loss with label 0.
        - Total loss = loss_real + loss_fake.
        - Update Discriminator weights using backpropagation.

      - Train Generator:
        - Generate fake data using the Generator.
        - Compute loss with label 1 (to fool the Discriminator).
        - Update Generator weights using backpropagation.
        - Log the loss of the Generator and Discriminator for each epoch.

- Output
  - A trained GAN model.

Tahap generate_samples:

- Inputs
  - n_samples: Number of data points to generate (default: 100).
  - latent_dim: Dimension of the latent space used by the Generator.

- Process
  - Generate random vector z from a normal Gaussian distribution (N(0, 1)).
  - Pass z to the Generator to create synthetic data.
  - Inverse scale the data using MinMaxScaler to revert to the original scale.
  - Convert the generated data into a DataFrame for further analysis.

- Output
  - A DataFrame containing generated synthetic data.

C. *Validate Model*

Initialize Validator

- original_data_path: Path to the original dataset CSV.
- synthetic_data_path: Path to the synthetic dataset CSV.

Validation Functions

- compare_distributions(feature):
  - Plot the distribution of real and synthetic data for a specific feature.
- calculate_statistics(feature):
  - Compute basic statistics (mean, std, etc.) of real and synthetic data.
  - Compare the differences in percentage.
- correlation_comparison():

- Compare the correlation matrices of real and synthetic data.
- Visualize using a heatmap.

- ks_test(feature):
  - Perform the Kolmogorov-Smirnov (KS) test on a specific feature.
  - Output: KS statistic and p-value.
- validate_all_features():
  - Untuk semua fitur yang sama antara data asli dan sintetis:
    - Compare distributions.
    - Compute statistics.
    - Perform the KS test.
    - Compare correlation matrices.
  - Save validation results in a dictionary.

Main Function :

- Create a GANValidator object.
- Run validate_all_features().
- Save validation results to a text file.

V. Result Analysis

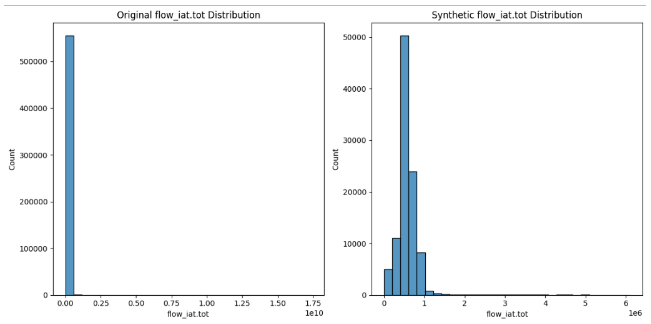
A. *Result of Train Architecture Model*

Based on the training results, the NetworkTrafficGAN architecture demonstrates stable adversarial training. The Discriminator loss (d_loss) stabilizes between 0.91 and 0.96, indicating its ability to differentiate real from synthetic data effectively. Meanwhile, the Generator loss (g_loss) decreases steadily from 2.5286 to a range of 1.10–1.16, suggesting that the Generator improves at creating realistic synthetic data over time. This balance between losses reflects a healthy dynamic where neither model overpowers the other, and convergence is achieved. Early epochs show rapid learning, while later epochs stabilize, indicating that the model approaches its optimal performance.

The overall performance suggests that the GAN effectively learns the distribution of real data and generates high-quality synthetic samples. The consistent loss values and balanced competition between the Generator and Discriminator imply that the synthetic data is both realistic and diverse, avoiding pitfalls like mode collapse. This result highlights the NetworkTrafficGAN's capability to produce synthetic network traffic data suitable for various applications.

B. *Validation of Synthetic Data*

Feature : flow_iat.tot



5.1 flow_iat.tot Graph

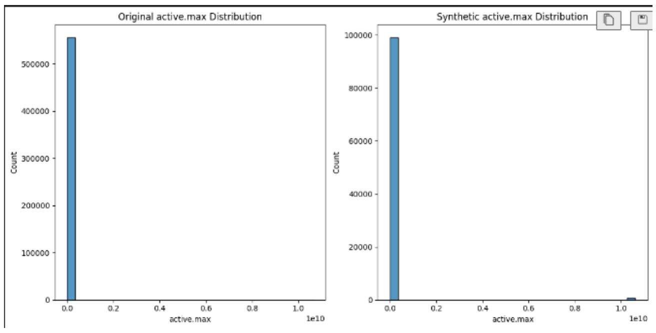
Statistic Comparison

	Original	Synthetic	Difference %
count	5.552780e+05	1.000000e+05	-81.99
mean	3.490637e+06	1.812573e+05	-94.81
std	1.804930e+07	1.518452e+06	-91.59
min	0.000000e+00	0.000000e+00	NaN
25%	0.000000e+00	6.512387e+04	inf
50%	0.000000e+00	9.506678e+04	inf
75%	0.000000e+00	1.375788e+05	inf
max	2.998038e+08	6.659048e+07	-77.79

Kolmogorov-Smirnov Test Results:

KS statistic	: 0.8671
p-value	: 0.0000

Feature : active.max



5.1 active.max Graph

Statistic Comparison

	Original	Synthetic	Difference %
count	5.552780e+05	1.000000e+05	-81.99

mean	9.312579e+06	5.723838e+0	-93.85
std	1.018055e+08	2.506136e+05	-99.75
min	0.000000e+00	0.000000e+00	NaN
25%	3.108978e+02	4.831047e+05	155290.19
50%	2.622390e+04	5.660411e+05	2058.49
75%	3.478281e+05	6.593446e+05	89.56
max	1.739303e+10	6.109824e+06	-99.96

```
Kolmogorov-Smirnov Test Results:
KS statistic     : 0.6726
p-value          : 0.0000
```

For complete validation results (all feature comparison), please visit: https://github.com/jonatyoung/GAN_HIKARI_2021.

VI. Conclusion

The training results of NetworkTrafficGAN highlight stable adversarial learning, where the balance between Generator and Discriminator losses suggests effective convergence. The Generator consistently improves in creating realistic data over epochs, while the Discriminator maintains its ability to differentiate between real and synthetic samples. This healthy dynamic demonstrates the model's capability to generate diverse synthetic data without issues like mode collapse, indicating its potential for practical applications requiring synthetic network traffic data.

However, validation results reveal notable statistical discrepancies between original and synthetic data. Significant differences in metrics such as mean, standard deviation, and percentiles across key features, coupled with poor Kolmogorov-Smirnov test results, suggest that the synthetic data does not fully replicate the original data distribution. These limitations indicate a need for further refinement of the model to enhance its accuracy in capturing complex data patterns. While the current performance shows promise, improvements are necessary to ensure the synthetic data meets higher standards of fidelity for advanced use cases.

VII. References

[1] He, H. and Ma, Y., 2013. *Imbalanced Learning: Foundations, Algorithms, and Applications.* Wiley-IEEE Press. Available at: https://ieeexplore.ieee.org/book/6542371 [Accessed 23 Dec. 2024].

[2] Johnson, J.M. and Khoshgoftaar, T.M., 2019. Survey on Deep Learning with Class Imbalance. *Journal of Big Data*, 6(1), pp.1–54. Available at: https://www.researchgate.net/publication/332165523_Survey_on_deep_learning_with_class_imbalance [Accessed 23 Dec. 2024].

[3] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, pp.321–357. Available at: https://doi.org/10.1613/jair.953 [Accessed 23 Dec. 2024].

[4] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems (NIPS)*, 27. Available at: https://papers.nips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf [Accessed 23 Dec. 2024].

[5] Shahriar, M.H., Haque, N.I., Rahman, M.A. and Alonso Jr, M., 2020. G-IDS: Generative Adversarial Networks Assisted Intrusion Detection System. *arXiv preprint arXiv:2006.00676.* Available at: https://arxiv.org/abs/2006.00676 [Accessed 23 Dec. 2024].

[6] Zhao, X., Fok, K.W. and Thing, V.L.L., 2024. Enhancing Network Intrusion Detection Performance using Generative Adversarial Networks. *arXiv preprint arXiv:2404.07464.* Available at: https://arxiv.org/html/2404.07464v1 [Accessed 23 Dec. 2024].