# Can Federated Network Traffic Classification Protect Private Dataset?

Huiwen Zhang and Feng Ye
Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI, USA
Emails: hzhang2279@wisc.edu, feng.ye@wisc.edu

***Abstract*—Network Traffic Classifiers (NTC) are critical for analyzing and categorizing traffic to ensure network performance, security, and quality of service. Federated learning (FL) has been adopted as a privacy-preserving solution for training these classifiers by enabling collaborative model updates without sharing raw traffic data. However, FL can be vulnerable to white-box inference attacks, where adversaries with access to model parameters and gradients can reconstruct the traffic features. Application updates will introduce the changed traffic data, which has a different distribution from the traffic data before. In this paper, we propose a data reconstruction attack by exploiting such vulnerability in FL-based NTC design. Using only public dataset and a Generative Adversarial Network (GAN), the proposed attack can successfully reconstruct data pieces from private datasets. Various metrics for similarity test are applied in the evaluation to validate the reconstructed dataset. These findings highlight the need for robust defenses to protect data privacy in network traffic data sharing for NTC development.**

## I. Introduction

With the advancement of communication technologies, the number of network users has been steadily increasing, leading to challenges in balancing network traffic and bandwidth. Network Traffic Classifier (NTC) has emerged as an effective method to enhance network management and improve the quality of service (QoS) [1–3]. It allows Internet Service Providers (ISPs) to prioritize specific types of traffic, enabling them to offer varying levels of QoS to their users. Deep learning has been proven to be a powerful tool for improving the accuracy of traffic classification. Collecting and sharing network traffic data is crucial to such NTC development. Especially since the characteristics of network traffic data can change significantly when applications are updated, such changes can lead to a mismatch between the traffic patterns represented in the training data and the real-world traffic encountered post-update [2, 4]. Therefore, NTC needs to be updated frequently with new data from users.

However, privacy concerns arise when developing NTCs that use private data. Recent advancements in distributed machine learning, particularly federated learning (FL), provide a promising solution for users to participate in updating NTC without sharing raw datasets. By eliminating the need to upload user data to a central server, its goal is to preserve user privacy [5]. However, despite its privacy-preserving goals, FL is not impregnable due to the collaborative nature of the framework. For example, one of the most concerning vulnerabilities in FL is its susceptibility to adversarial attacks, including data reconstruction attacks, poisoning attacks, and inference attacks [6].

In this work, we explores if federated NTC can truly protect private dataset by proposing a data reconstruction attack from a local client. The main focus is on the white-box inference attack, a scenario where an adversary has complete knowledge of the federated learning model, including its parameters and architecture. The attacker can take on different roles:

- **As the central server:** The attacker can observe the updates of the clients and manipulate the global model by controlling the aggregation process.
- **As a participating client:** The attacker knows the global model parameters and can intentionally upload malicious updates to influence the global model.

In both cases, the attacker can leverage their understanding of the model to disrupt or compromise the federated learning process in an active way, either by directly modifying the global parameters or by injecting incorrect updates to degrade model performance. This level of access allows the attacker to exploit the collaborative learning process to infer sensitive information about private client data [7]. To demonstrate the vulnerability of FL to white-box inference attacks, we implement a generative adversarial network (GAN)-based approach to reconstruct data pieces only exist in the private dataset. In this approach, the adversary utilizes a GAN to exploit data features by iteratively refining synthetic samples to match the observed gradients. Ultimately, the synthetic samples share similar features to private data pieces.

We validate the success of the attack through extensive experiments on real-world network traffic datasets collected at different time and from different environment. By analyzing feature maps and multiple metrics for similarity test, we show that the proposed reconstruction attack can successfully reveal a large portion of the private dataset, albeit being used only in the FL process. Furthermore, we examine the impact of introducing maliciously generated data into the federated learning process, evaluating how it can degrade the performance of the global model and compromise its reliability. It leads to a possible method for detecting a malicious client.

The remainder of this paper is organized as follows. Sec. II provides an overview of the preliminary concepts. Sec. III describes the threat model. Sec. IV outlines the proposed data reconstruction attack. Sec. V presents the evaluation results. And Sec. VI concludes the paper and discusses potential directions for future work.

TABLE I: Chosen metrics for similarity test.

| Metric | Application in AI Domain | Computation |
|---|---|---|
| Mean Squared Error (MSE) | Regression tasks, model evaluation | $\text{MSE}(A,B) = \frac{1}{n}\sum_{i=1}^{n}(A_i - B_i)^2$. |
| Jensen-Shannon Divergence (JS) | Probability distribution comparison | $\text{JS}(A,B) = \frac{1}{2}[\text{KL}(A \parallel M) + \text{KL}(B \parallel M)]$. |
| Cosine Similarity (CS) | Vector similarity [8] | $\text{CS}(A,B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \in [-1,1]$, where 1 indicates maximum similarity. |
| Jaccard Similarity [9] | Vector comparison [10] | $J(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert} \in [0,1]$, where 1 indicates complete similarity. |
| Jaro-Winkler (JW)Similarity [11] | Text similarity [12] | $J_w = J + l \cdot p \cdot (1-J)$, where $l$ is the length of the common prefix (up to 4), $p$ is the scaling factor (default 0.1), and $J = \frac{1}{3}\left(\frac{m}{\lvert A \rvert} + \frac{m}{\lvert B \rvert} + \frac{m-t}{m}\right)$ is the Jaro similarity, where $m$ is the number of matching characters, $\lvert A \rvert$ and $\lvert B \rvert$ are the lengths of the strings $A$ and $B$, and $t$ is the number of transpositions. $J_w \in [0,1]$, where 1 indicates maximum similarity. |
| Levenshtein (Lev) Distance [13] | Text alignment [14] | Similarity Score $= 1 - \frac{\text{Levenshtein Distance}(A,B)}{\max(\lvert A \rvert, \lvert B \rvert)} \in [0,1]$, where 1 represents identical strings and higher values indicate greater similarity. Levenshtein Distance is the minimum edits (insertions, deletions, or substitutions) needed to transform $A$ into $B$. |

## II. Preliminaries

### A. AI-based Network Traffic Classification

NTC refers to the process of identifying the types, sources, or destinations of data packets transmitted across a network. It plays a vital role in network security and management, enabling Internet Service Providers (ISPs) to optimize resource allocation and improve overall network efficiency. With the increasing prevalence of encrypted traffic and the exponential growth in data volume, machine learning-based NTC models have become a key focus due to their efficiency and accuracy [15]. Recently, FL has been applied to NTC, gaining attention for its privacy-preserving and distributed computing capabilities. In the FL framework, clients train models locally using their network traffic data, and only the model parameters are aggregated by the server, ensuring data privacy. In this study, a Multi-Layer Perceptron (MLP) is utilized as the NTC model because of its simplicity and effectiveness. We also separate two distinct datasets, one serving as public data and the other as private data. The public dataset consists of network traffic shared publicly yet mostly out-dated. Some of the data in this dataset is used as a pre-training resource. The remaining portion of the dataset is accessible to all clients as their local data in the FL framework. Each client consists of a private dataset consisting of up-to-date network traffic data, even if they belong to the same applications in the public dataset. Note that the characteristics of network traffic often differ significantly due to frequent updates of applications.

### B. GAN-based Data Synthesis

Generative Adversarial Network (GAN) is a powerful tool for generating data that is close to the real data. It consists of two components: a Generator ($G$) and a Discriminator ($D$). $G$ attempts to create data that resembles the training data. $D$ tries to distinguish between real data from the training set and fake data generated by the $G$. The $G$ aims to minimize the $D$'s ability to distinguish between real and synthetic data, while the $D$ attempts to maximize its accuracy, creating a dynamic optimization problem. In federated learning, GAN can exploit the iterative exchange of model parameters between clients and the server to achieve an attack [16]. A malicious participant can use the shared model parameters to infer sensitive information about the client data.

### C. White-box Attack

In federated learning, a white box attack refers to an attack in which the attacker has complete knowledge of the model, including its parameters and structure [7]. The primary goal of the attacker is to exploit the gradients to infer sensitive information about the private data. This is possible because the gradients exchanged during training encode information about the data. In this work, we mainly focused on the scenario where an attacker pretends to be a participating client.

### D. Similarity Test

Similarity test is a method used to detect and analyze the effectiveness of data reconstruction attacks. It determines whether an attack is successful or the extent of the impact by evaluating the similarity of generated samples compared with the original data. Six widely used metrics for similarity tests are listed in Table I. To ensure consistency and clarity, we define two pieces of data $A$ and $B$ as the inputs for all chosen metrics for similarity test.

## III. Threat Model

This work adopts the threat model described in [16], which relies on an active insider to achieve a white-box attack. In this scenario, the model is NTC, and each local client possesses both public and private network traffic data from some applications. Attacker $\mathcal{O}$ is an outsider at the beginning of the training process. It has access to the public dataset and can access the global model parameters. During the federated learning process, where each client updates the model using their private dataset, attacker $\mathcal{O}$ joins as a local client and launches a data reconstruction attack. The attacking objectives are twofold: first, to extract the private data distribution related to the updated application, and second, to influence the learning process in a way that coerces victims into revealing additional information about specific targeted classes. Specifically, the adversary operates as an insider within the privacy-preserving

collaborative learning framework while maintaining several critical characteristics:

- The adversary aims to infer meaningful information about private data from a certain label that has different features from the public data.
- The adversary participates in collaborative learning only when these labels correspond to data with features distinct from the public data.
- The adversary deploys a local GAN while adhering to the protocol specifications observed by legitimate participants - including proper parameter selection, gradient uploads/downloads.
- The adversary possesses knowledge of other participants' label information and knows which labels are associated with data that have features differing from the public data.

## IV. Proposed Data Reconstruction Attack

### A. Federated NTC Model with Private Dataset

Before federated learning begins, a warm-up model is pre-trained by the central server using a subset of a public dataset to ensure initial stability in the global model. The weights of the warm-up model are then used to initialize the global model. During federated learning, local clients agree in advance on the traffic classifier model and the labels of the data they possess. The warm-up dataset is not included in any client's training process. Some labels are shared among clients to mitigate the non-independent and identically distributed (non-i.i.d.) problem. Mitigation of the non-i.i.d. dataset is beyond the scope of this work. The labels for the data being replaced with private data do not overlap between clients. The data replacement process intends to mimic periodic data collection in NTC development. Without loss of generality, the global model used in this federated learning framework is an MLP designed for classification tasks. It consists of three main layers: an input layer with 500 units to match the feature dimensionality of the dataset, a hidden layer with 256 units incorporating Batch Normalization, LeakyReLU activation (with $\alpha = 0.01$), and a Dropout layer with a rate of 0.5, followed by another hidden layer with 32 units, also equipped with Batch Normalization, LeakyReLU activation, and Dropout. The output layer is a fully connected layer with 7 units, corresponding to the number of classes in the dataset. This model is trained using the Adam optimizer and Sparse Categorical Crossentropy loss to optimize classification accuracy. Other approaches can be used as well.

### B. Data Reconstruction Attack

For better illustration, the proposed data reconstruction attack is summarized in Algorithm 1. The attacker $\mathcal{O}$ is assumed to have access to the public dataset. It participates in the federated learning process as a local client. To uncover the private distribution of the target label from other clients, $\mathcal{O}$ first removes the target label data from the public dataset to create its own dataset. Acting as a legitimate client, $\mathcal{O}$ then participates in the federated learning process and launches the data-reconstructed attack based on a GAN. In particular, $\mathcal{O}$ initializes the weights of the discriminator ($D$) in its GAN with the weights of the global model, an overview of the data reconstruction attack and the general structure of the GAN are shown in Fig. 1.

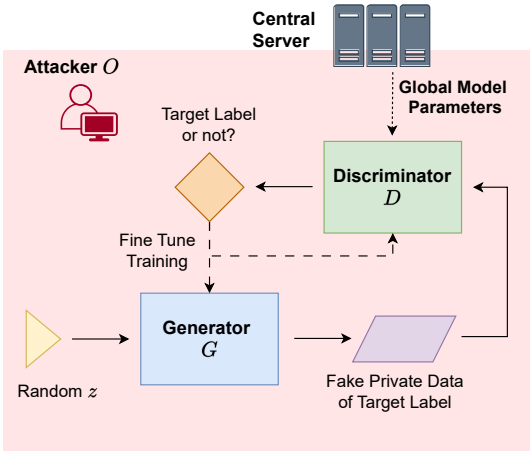



Fig. 1: The proposed data reconstruction attack.

**Algorithm 1** The proposed data reconstruction attack.

**Input:** Noise dimension for GAN, target label $T$
**Output:** Generated target data
*Initialization* :
Warm-up model training using a subset of the public data;
Set the global model to the warm-up model;
**Training Loop:**
**for** each round $r$ in total rounds $R$ **do**
  Evaluate the global model on testing data;
  Check for private data replacement conditions;
  **if** performance is consistent for $m$ epochs and private data replacement not completed **then**
    Replace a batch of client data with private samples;
  **end if**
  Local client training:
  **for** each client $i$ **do**
    Set client model weights to global model weights;
    Train client model on its local data;
    **if** replacement is complete **then**
      Attacker ($\mathcal{O}$) join the federated learning;
      Generate data of $T$ with the GAN;
      Append malicious data to $\mathcal{O}$'s dataset;
    **end if**
    Update client model weights;
  **end for**
  Aggregate client weights to update the global model;
**end for**
**if** Attack has been initiated **then**
  Evaluate generated data performance;
**end if**

The GAN structure used in this work is a conditional GAN (cGAN) [17]. The generator ($G$) is designed to synthesize data

for the target label using a 100-dimensional noise vector as input. It consists of dense layers with Batch Normalization and LeakyReLU activations, culminating in an output layer that matches the dimensionality of the real data (500 features). The discriminator mirrors the architecture of the global model. By training $G$ to produce data that $D$ classifies as the target label, the cGAN effectively generates synthetic data resembling the target class. After generating the synthetic data, $\mathcal{O}$ adds it to its local dataset, mimicking the behavior of a legitimate client with private data. By minimizing the loss of $G$ over several epochs, $\mathcal{O}$ refines the generated data to closely resemble the real private data to the extent that it can deceive the central classifier into categorizing it as the target private label.

## V. Evaluation Results

### A. Datasets and Testing Scenarios

In the evaluation, the public dataset consists of 7 classes of data ('Facebook', 'Netflix', 'SFTP', 'YouTube', 'Scp-Down', 'Skype' and 'Vimeo') randomly extracted from the ISCXVPN2016 dataset [18]. The private dataset consists of two classes ('Netflix' and 'YouTube') collected more recently in our lab [1, 2]. Note that the data from the same application on a later day may not be correctly classified using the warmed-up model due to frequent and possible updates of protocols, codecs, and algorithms in network applications. In the training process, data are normalized to 1. And the header (first 24 bytes) of the data is removed.

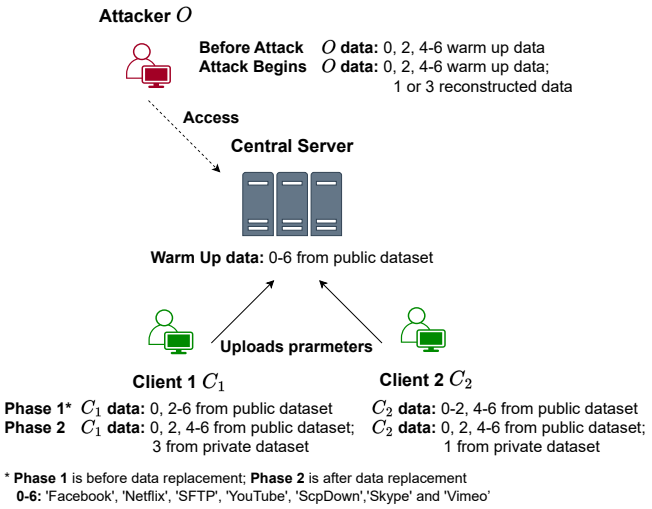



Fig. 2: The evaluated attacking scenario.

The evaluated attack scenario is illustrated in Fig. 2. Two clients, $C_1$ and $C_2$, participate in the federated learning process. The central server possesses 5% of the data from the public dataset to train the warm-up model. During the initialization phase, both clients have data from six classes in the public dataset, which excludes the warm-up data. The datasets of $C_1$ and $C_2$ are mutually exclusive. Specifically, $C_1$ holds data from labels 0, 2, 3, 4, 5, and 6, while $C_2$ holds data from labels 0, 1, 2, 4, 5, and 6. Once the central model converges, $C_1$ gradually replaces its data from label 3 with private data, and $C_2$ does the same with its data from label 1, mimicking the updating process with the new dataset from updated applications, e.g., on a montly basis. The attacker $\mathcal{O}$ is assumed to have access to the public dataset. When $\mathcal{O}$ launches the attack, it generates data for the target label. In this experiment, the target label is either 1 (Netflix) or 3 (YouTube) that are in the private dataset.

### B. Evaluation on the Data Reconstruction Attack

We first show the performance of the warm-up model. As illustrated in Fig. 3, the warm-up model can provide relatively high accuracy on the public dataset. Note that the warm-up model needs not provide the optimal performance as it is not useful for actual NTC. It is mainly becuase the warm-up model cannot classify the private dataset, albeit they carry the same labels included in the public dataset. This discrepancy clearly demonstrates the unique challenge in networking: updates to the application introduce changes in data distribution, rendering the warm-up model inadequate for handling the updated features. The sustainability of NTC performance has been discussed in our previous work [1, 2].

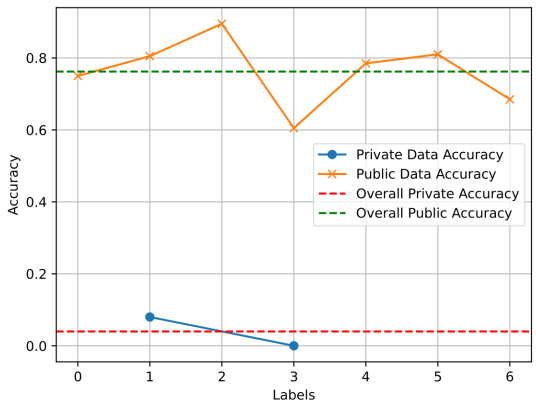
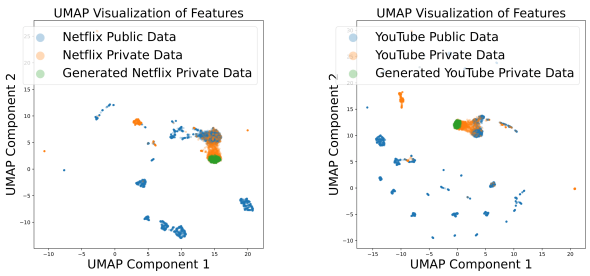



Fig. 3: Testing accuracy for private and public datasets.

Each reconstruction attack generates 500 pieces of data to exploit a total of 1,450 pieces of data in the original private dataset of Youtube and Netflix, respectively. We further evaluate the Uniform Manifold Approximation and Projection (UMAP) [19] feature maps of the public data, private data, and reconstructed data for YouTube and Netflix. UMAP is a feature-reduction technique for visualizing data. These feature maps are generated using two features UMAP, with the projection performed using the correlation metric. As shown in Fig. 4, it is evident that the features of the public data (blue points) and original private data (orange points) are separated, indicating that the public data has different features compared to the private data. The features of the reconstructed private data (green points) overlap significantly with those of the original private data, demonstrating that the reconstructed data effectively captures the feature characteristics of the private data. This overlap highlights the fidelity of the reconstruction




(a) Netflix data (b) YouTube data

Fig. 4: Feature Map of the Public Data, Private Data, and the Reconstructed Private Data

process in retaining the distinctive features of the private data, leading to the success of the internal reconstruction attack.

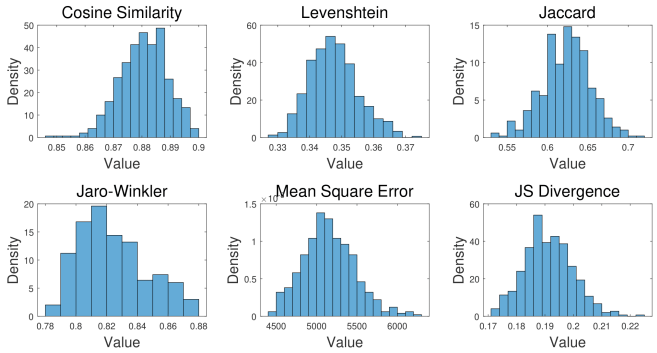



Fig. 5: The PDF of the similarity scores of the reconstructed Netflix private data among 6 metrics.

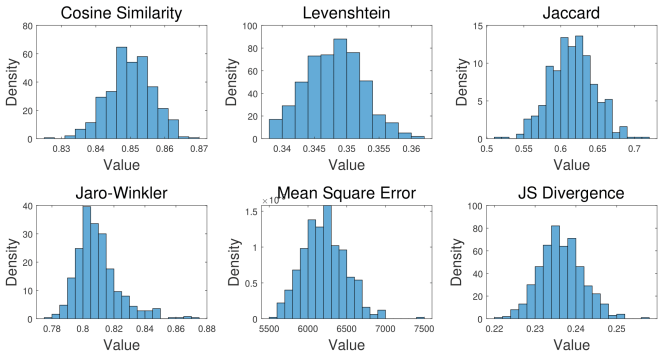
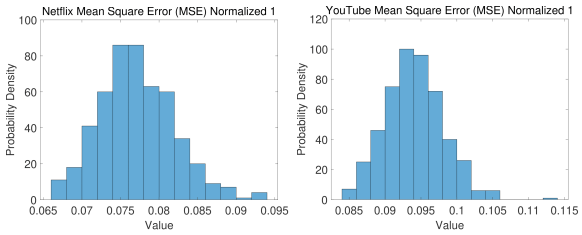



Fig. 6: The PDF of the similarity scores of the reconstructed YouTube private data among 6 metrics.

To better evaluate the reconstructed data, we compare the results with the original private dataset using the similarity metrics introduced in Sec. II-D. Before calculating the similarity scores, the data are normalized to integers ranging from 0 to 255, as three of the metrics are designed to work with strings. For each piece of reconstructed data, the similarity scores are computed for the entire private dataset with the same label. Fig. 5 and Fig. 6 present histograms of the Probability Density



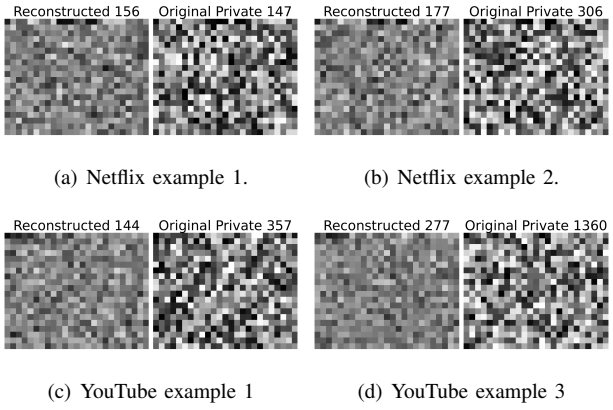

(a) Reconstructed Netflix data (b) Reconstructed YouTube data

Fig. 7: MSE between the original private data and the reconstructed private data (normalized to 1).

Function (PDF) for the highest similarity scores (except for MSE) of all the reconstructed Netflix and YouTube private data. The MSE metric is challenging to interpret for data scaled between 0 and 255. Instead, the MSE metric is applied with data normalized between 0 and 1. Similar histogram plots for the highest similarity scores using the MSE metric are shown in Fig. 7. To match the reconstructed data with the original one, high values of Cosine Similarity, Jaccard, Levenshtein, and Jaro-Winkler are desirable, while low values of MSE and JS divergence are preferred. From the results, we can see that, among the chosen similarity metrics, Cosine Similarity, MSE, JS divergence, and Jaro-Winkler can better measure the similarity between reconstructed and original data. Meanwhile, Levenshtein and Jaccard are less useful, as their character-based nature makes them highly sensitive to minor changes in the normalized 0 to 1 scale, leading to significant variations when rescaled to 0 to 255.

However, relying solely on the highest score of each similarity score cannot find the original data consistently. For example, for label 1 (Netflix) cross-referencing two metrics cosine similarity and MSE, only 5 pieces of original private data have the matching reconstructed data. Cross-referencing three metrics cosine similarity, MSE and JS divergence leaves to merely 4 pieces of original private data with the matching reconstructed data. Cross-referencing all metrics returns zero pieces of original private data that match with the reconstructed data. To address this issue, we choose to cross-reference 5% data pieces that present the highest similarity scores by each metric for each reconstructed data. The implementation of 5% is based on the empirical results for the chosen dataset. The setting depends on specific testing scenarios. To ensure robust matches, results from Cosine Similarity, MSE, JS divergence, and Jaro-Winkler are used for cross-referencing. By doing so, 476 (out of 500) reconstructed data pieces of label 1 (Netflix) and 459 (out of 500) reconstructed data pieces of label 3 (YouTube) can be mapped to just a few (1 to 3) original data pieces. It means that the data reconstruction attack can successfully extract the original data piece from the private dataset. Fig. 8 visualizes two randomly chosen reconstructed data samples of each label (Netflix and YouTube) and their best match in the private dataset.

To detect the data reconstruction attack, the central server or a legitimate user can monitor the performance of the global

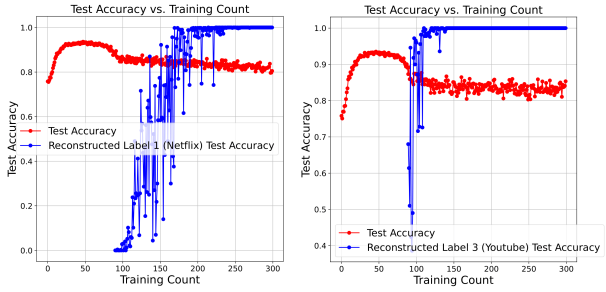



(a) Netflix example 1. (b) Netflix example 2.

(c) YouTube example 1 (d) YouTube example 3

Fig. 8: Visual demonstration of the reconstructed data (left) and its best matching original data (right).

model during federated learning. As illustrated in Fig.9, a noticeable decrease in performance can occur when the attack is launched in the learning process. The sudden change is due to the rough reconstruction in the first few iterations from the attacker $\mathcal{O}$. However, the performance is stable as the quality of data reconstruction increases.




(a) Netflix. (b) YouTube.

Fig. 9: Global model accuracy during attack vs. # epochs.

## VI. Conclusion and Future Works

In this work, we proposed a data reconstruction attack on FL-based NTC design. The proposed attack is intended to explore if the private dataset can be truly protected with FL only in NTC training. Evaluations were conducted using real-world network traffic data and multiple metrics for similarity tests. The results demonstrated that the data reconstruction attack can be successfully launched, revealing a large portion of the private dataset, albeit being used solely for local model updates in the FL process. We also discovered a sudden NTC performance decrease when the attack initiates. The performance change may serve as an alert to flag the existence of an internal attacker. In future work, we will explore more robust countermeasures and provide secure methods for data sharing with privacy preservation.

## Acknowledgment

This project was supported by the U.S. National Science Foundation under the grants NSF 2344341.

## References

[1] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for dl-based traffic classifier update," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 397–405.

[2] J. Zhang, F. Li, and F. Ye, "Sustaining the high performance of ai-based network traffic classification models," *IEEE/ACM Transactions on Networking*, vol. 31, no. 2, pp. 816–827, 2023.

[3] F. Li and F. Ye, "Adaptive and lightweight network traffic classification for edge devices," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 4, pp. 2003–2014, 2022.

[4] J. Ge, T. Li, and Y. Wu, *Concept Drift Detection for Network Traffic Classification*, 2023, pp. 91–108.

[5] H. Mun and Y. Lee, "Internet traffic classification with federated learning," *Electronics*, vol. 10, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/1/27

[6] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 7, pp. 8726–8746, 2024.

[7] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 739–753.

[8] X. Li, V. Metsis, H. Wang, and A. H. H. Ngu, "Tts-gan: A transformer-based time-series generative adversarial network," in *Artificial Intelligence in Medicine*, M. Michalowski, S. S. R. Abidi, and S. Abidi, Eds. Cham: Springer International Publishing, 2022, pp. 133–143.

[9] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bulletin de la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[10] T. Hu, C. Long, and C. Xiao, "A novel visual representation on text using diverse conditional gan for visual recognition," *IEEE Transactions on Image Processing*, vol. 30, pp. 3499–3512, 2021.

[11] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989.

[12] A. Dunn, J. Dagdelen, N. Walker, S. Lee, A. S. Rosen, K. A. Persson, G. Ceder, and A. Jain, "Structured information extraction from complex scientific text with fine-tuned large language models," *arXiv preprint arXiv:2212.05238*, 2022, submitted on 10 Dec 2022. [Online]. Available: https://arxiv.org/abs/2212.05238

[13] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707–710, 1966.

[14] A. Biswas and W. Talukdar, "Robustness of structured data extraction from in-plane rotated documents using multi-modal large language models (llm)," *arXiv preprint arXiv:2408.10925*, 2024, submitted on 13 Jun 2024. [Online]. Available: https://arxiv.org/abs/2408.10925

[15] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 2451–2455.

[16] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 603–618. [Online]. Available: https://doi.org/10.1145/3133956.3134012

[17] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014. [Online]. Available: https://arxiv.org/abs/1411.1784

[18] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, Rome, Italy, 2016, pp. 407–414.

[19] L. McInnes, J. Healy, N. Saul, and L. Grossberger, "Umap: Uniform manifold approximation and projection," *The Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.