Understanding Privacy and Quality Tradeoffs in Synthetic Network Data

Andrew Chu*
University of Chicago

Kyle MacMillan*
University of Chicago

Paul Schmitt
California Polytechnic State University

Nick Feamster
University of Chicago

Abstract

The limited availability of high-quality computer networking data, and the privacy risks of sharing what does exist, has prompted development of ML-based methods for generating *synthetic* network data that mimics real communication between networked devices. The viability of these models hinges on both the quality of their output and how well they preserve private information encoded in their training data. Prior work has sought to address this by training models with differential privacy (DP). However, how this choice affects the actual privacy of the training data, and subsequently the quality of the generated output, is not well understood.

In this work, we analyze the relationship between privacy and quality in generative network data models. Using the success of membership inference attacks (MIAs) as the metric for privacy, we observe that whether DP mitigates MIAs depends heavily on model architecture and representation of network data used for training. In particular, we empirically find that some approaches to generating synthetic network data train models that heavily skew towards either overgeneralizing or undergeneralizing to their training data, resulting in poor or inconsistent MIA performance. In these cases, using DP does not yield substantive improvements in vulnerability to MIAs. As for the quality of generated data, we find that DP synthetic network data can retain statistical similarity to real data even under strict privacy budgets, and that downstream models (*e.g.,* classifiers, regressors) trained on this data tend to achieve at least as good accuracy as models trained on non-DP data. These results suggest that DP, depending on the model, offers protection against MIAs without degrading the utility of the generated output, and in some cases, improves utility.

Keywords

Membership inference attack; Machine learning for networking

1 Introduction

The security of modern computer networks depends on having access to ample network traffic data. Indeed, network operators rely on flow and packet level traces to train machine learning for a variety of monitoring tasks: detecting unusual network events (e.g. failures, new devices, and attacks) [71], classifying application type [43], and identifying the websites visited by users [69]. But access to network traffic data is limited. The bulk of network data useful for these tasks is held by private companies that are reluctant to share traffic captured on their commercial network. For these companies, sharing either packet or flow level data risks disclosing private or proprietary information. Raw packet-level data can reveal IP addresses and ASNs, while flow-level data might leak company network topology or peering arrangements. Although some network datasets are publicly available, they do not generalize [32]. So, absent regular data sharing, network operators must rely exclusively on internal datasets, which tend to be too small to be useful for most monitoring tasks.

One potential solution is to share *synthetic* network traffic instead of the original traces. Synthetic traffic is created using either simulation [4, 5, 10, 26, 61, 67], or machine learning (ML)-based methods [9, 33, 63, 73] (the focus of this paper) which train generative network data models on real network data to produce synthetic counterparts. These data (ideally) preserve properties in the original traffic that companies want to share, while obfuscating sensitive information in the original traffic.

To illustrate the usefulness of synthetic data, consider a company that wants to develop a generalizable ML model for networking. The company may need access to a diversity of data to ensure that the model truly generalizes. Unfortunately for the company, privacy laws restrict the cross-border flow of sensitive data [1, 19–21, 65]. The goal of synthetic data is to satisfy both of these constraints. Instead of sharing the original traffic, the company can locally train generative network data models on their raw traffic, then use these models to create synthetic network data which *can* be exported. This is a real use case described by Rockfish Data [13].

The viability of synthetic traffic will thus depend on quality, privacy, and the relationship between these two metrics. For quality, the key question is how well synthetic traffic can substitute for the original traffic: when a network operator trains a model on synthetic traffic, how does that model's performance compare to one trained on the original traffic? The question of privacy is more flexible. Although a number of model privacy metrics exist, measuring a model's vulnerability to membership inference attacks (MIAs) is a standard approach. We can thus gauge the privacy-preserving properties of generative models by testing them against MIAs. From here, the question of how the quality of the traffic varies with how well the model preserves privacy arises naturally. Thus, the goal of this paper is to illuminate the relationship between privacy-preservation and utility in synthetic traffic. Table 1 summarizes our main findings.

*Equal contribution.

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–19

https://doi.org/XXXXXXX.XXXXXXX

Table 1: Main results and their location in the paper.

White-box binary classifier-based MIAs are effective against some generative network data models, but are inconclusive against others (§3.4).
Adding DP reduces MIA success against a token-based model, but not a diffusion or adversarial network-based model (§3.5).
DP reduces the fidelity of synthetic network data at differing degrees. Low fidelity does not always imply low utility (§4.1).
Downstream models trained on DP synthetic traffic can perform at least as well as those trained on non-DP traffic (§4.2).
DP can incidentally have a regularizing effect on all evaluated models, preventing overfitting and improving downstream fidelity/utility (§4.3).

Our paper proceeds as follows. Section 2 contextualizes our work with prior research on MIAs and network traffic generation. We then study the vulnerability of the most prominent generative models for networking data (at the time of writing) to an MIA. Our MIA is particularly effective (precision and recall ≥ 0.90, false positive rate ≤ 0.05) against some, but not all models (Section 3.4).

We then show how well DP can defend against an MIA in Section 3.5. While we find that adding DP can limit MIA efficacy, particularities observed in the non-DP case persist across thresholds of added DP noise, implying careful consideration in the development of future generative models to enable integration with DP.

In Section 4, we evaluate how the quality of the synthetic traffic varies with different DP privacy budgets along two dimensions: *fidelity* and *utility*. At a high level, fidelity is a general-purpose metric that measures how similar the distribution of fields in the synthetic traffic is to the fields in the original traffic. Utility is the opposite, evaluating the quality of the traffic on a single, specific task. In our analysis, we train two classifiers and one regressor on both synthetic traffic and the original traffic, and compare model performance. We find that, although making the generative models differentially private can decrease the fidelity of synthetic traffic, that decrease does not usually lead to a subsequent decrease in utility. In each of our downstream models, differentially private traffic either led to increased accuracy or no decrease. This result suggests that differential privacy, rather than degrading the synthetic traffic, can operate to augment the traffic.

Finally, we discuss and give suggestions for how differential privacy can be added to synthetic traffic to mitigate model vulnerability to MIAs without compromising downstream performance. Preserving privacy without degrading the utility of synthetic traffic is key to unlocking the potential of synthetic network traffic.

2 Background and Related Work

In this section, we sketch out the form and goal of membership inference attacks, as well as how differential privacy can serve as a defense. We then present an overview of existing generative ML-for-networking models, efforts to make them privacy-preserving, and existing analyses of potential privacy leakage.

2.1 Membership Inference Attacks

The goal of membership inference attacks (MIAs) is to predict if a specific data point exists (is a member) in an ML model's training set. MIAs exploit a model's tendency to memorize qualities of the training data by comparing model behavior on member data versus non-member data, and have been successful against models in various domains [6, 25, 58, 72]. In general, MIAs are either black-box (only the input and output to the target model can be used to infer membership status) or white-box (the adversary has access to the target model's architecture and state during use).

Shokri *et al.* [59] were first to present a technique called *shadow training* for membership inference. In this attack, an adversary generates their own ground truth of synthetic member and non-member data with similar distribution to data used to train the target model. The synthetic data are then used to train a binary classifier that adjudicates membership status. Besides shadow training, metric-based MIAs that rely on model performance metrics extracted from pre-trained models for a given input have also seen success (*e.g.,* loss [72], entropy [62], and token-based metrics [44, 45, 58]). This approach can be more efficient than shadow training because the target metrics can be extracted from the target model, and thus do not require the shadow training pipeline.

2.2 Differential Privacy

Differential privacy (DP) is a framework for preserving privacy via addition of calibrated noise to data-related model computations [18]. Let $\mathcal{X}$ be the universe where records for two databases x and y are drawn from, $\mathcal{M}$ be a randomized algorithm with domain $\mathbb{N}^{|\mathcal{X}|}$, and $\|x - y\|_1$ be a measure of how many records differ between x and y (*i.e.,* the ℓ_1 distance). Then, $\mathcal{M}$ is considered (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all $x, y \in \mathbb{N}$:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta, \tag{1}$$

where ε and δ are parameters that dictate the *privacy budget*, or robustness of the privacy guarantees. Specifically, smaller values for either parameter are more stringent. As ε approaches zero, the quantity of noise being added to satisfy the definition grows as a result of $\exp(\varepsilon)$. As δ approaches zero, the right side inequality threshold for satisfying the expression decreases. This definition is achieved by adding noise (drawn from either a Gaussian or Laplace distribution and dictated by ε) to x prior to its consumption by $\mathcal{M}$.

In practice, the updates and outputs of a differentially private model aren't the product of any one data point, but are based on noise-obfuscated versions of the intermediate values. Thus, a model's learned state is not significantly impacted by any one single data point. Tying the model output to multiple data points mitigates the underlying risk of membership inference. What's more, careful calibration of the added noise allows the underlying model utility and performance to remain relatively unaffected.

2.3 Generative Models for Network Data

Generative models for network data are ML models that learn from raw or summary-level data of communication between networked devices, and output synthetic data of the same granularity that mimics the patterns and characteristics of the original communication. In general, generative models for network data can be categorized into two distinct types: *traffic-attribute generators* and *raw packet generators*. Traffic attribute generators produce *traffic attributes* – sequences of one or more derived data: packet header fields (*e.g.,* addresses, ports, flags), flow statistics (*e.g.,* throughput, duration), or metadata (*e.g.,* event types, web page views) in tabular

format. Existing traffic attribute generators rely on either generative adversarial networks (GANs) [41, 73], diffusion models [60], transformers [11], or a combination of diffusion and transformers [76], to learn the patterns and range of values sequences of each attribute take over the span of a flow or session (*i.e.,* time series). Synthetic network data produced by these models have high statistical similarity to ground truth original data, and can improve the performance of downstream ML models trained for traffic inference tasks (*e.g.,* classification). One shortcoming of these generators is that, because they learn patterns piecewise per attribute, they are unable to synthesize data for stateful protocols (*e.g.,* TCP) or traffic comprised of interleaved flows, which rely on interactions occurring between multiple attributes and/or sources.

Raw packet generators, on the other hand, use simulation or machine learning (diffusion, transformers) to create synthetic network traffic in the form of verbose, content level PCAPs. Prior work on raw packet generation use diffusion [33], transformer [53] and state-space model [9]-based approaches to learn the fine-grained intra- and inter-packet and flow relationships that exist in raw networked device communication. These models' synthetic data report improvements over traffic attribute generators in both statistical similarity, and performance of downstream ML models trained on these data, often at the cost of higher training complexity and overhead. Unlike traffic-attribute generators, raw packet generators can synthesize data for stateful protocols because they learn the relationships between actual contents of packets.

2.3.1 Differential Privacy in Generative Models for Network Data. A major motivation for ML-based synthetic network data generation is enabling privacy-preserving data sharing. Rather than sharing real network traffic and risk privacy-leakage, network operators could train generative models to produce synthetic traffic that captures only non-private signals in the traffic [41]. Unfortunately, ML models tend to overfit to training data, memorizing and directly mimicking the inputs in their output. To solve this problem, two traffic attribute generators (DoppelGANger [41] and NetShare [73]) incorporate DP during the phase of optimizing the model towards its objective function. There, both models replace the default optimization algorithm (*e.g.,* stochastic gradient descent [SGD] [35], Adam [36]) with a DP variant (*e.g.,* DP-SGD [2]) that clips and adds noise to gradients according to a given DP privacy budget during backpropagation. This enforces the privacy guarantees described in Section 2.2, where any single training sample does not disproportionately influence the model state. Unfortunately, neither work thoroughly examines how well DP improves privacy. NetShare presents results for only a single privacy budget ($\varepsilon = 24.24$), which may arguably provide little practical utility for protecting the privacy of an individual sample [17, 22, 46]. While DoppelGANger evaluates some more conventional settings (*e.g.,* $\varepsilon \leq 1$), it similarly includes results for large privacy budgets (*e.g.,* $\varepsilon \in \{10^6, 10^8\}$). Both models also do not validate whether their DP network data can be used in downstream tasks (*e.g.,* training ML-models). No existing raw packet generator (i.e., NetDiffusion [33], NetSSM [9]) attempts to integrate any form of privacy protection in their pipelines. Thus, more work is needed to understand both how DP should be applied to generative network data models, and how it affects the quality of these models' resulting output. To this end, we evaluate in this work four privacy budgets ($\varepsilon \in \{0.1, 0.5, 2.0, 4.0\}$) that aim to balance recommendations from academic literature and real-world use [14, 46].

2.3.2 Privacy Leakage in Generative Models for Networking Data. At the time of writing, little work has been done to study potential privacy leakages in generative models for networking data. The closest comparison to our work is the concurrent work by Tran *et al.* [66], which also evaluates the privacy preservation of generative network data models, evaluating data extraction and membership inference attacks against these models and giving discussion on the tradeoffs between privacy and utility of synthesized data when applying different privacy protection techniques. Our paper differs from this work in several ways. In the attack setup, the concurrent work evaluates metric-based MIAs, whereas we evaluate the binary classifier-based approach. Most notably, the two efforts starkly differ when evaluating the effect of DP on MIA performance. The concurrent work adds DP noise directly to the training data, prior to its ingestion by any of the evaluated models. Further, this noise is not added uniformly, but only to specific "sensitive network properties," of certain extracted packet headers. In contrast, we directly integrate DP when training our evaluated models using a standardized method that uniformly adds noise to all values of input data. Finally, while Tran *et al.* report the results of an extensive breadth of privacy preservation techniques, they do not provide a comprehensive analysis (particularly for DP) towards understanding the specific qualities of networking data and/or tailored-models that cause the observed outcomes, or how added privacy practically affects the generated data. We provide analysis of how the attributes of networking data interact with qualities of network data models specifically, and how this impacts both the downstream fidelity and utility of the generated data.

Another related study is conducted by Jin *et al.* [34], which presents evidence of privacy leakage in generative models for networking data (dubbed SynNetGens in their paper) via a novel, networking-domain specific "source-level MIA" called TraceBleed. Our paper has two primary differences from this work. First, the goal of TraceBleed is to discern if the traffic of any given *user* is present in a model's training dataset, differing from the conventional MIA (such as evaluated in this work) where the goal is to determine if any specific *sample* is present in a model's training data. Following from this difference, all results and discussion in the TraceBleed paper on the privacy of generative network data models and DP, are in context of source-level membership inference, not conventional membership inference. The results of our two works thus present findings for orthogonal problems.

2.4 Evaluating the Quality of Synthetic Traffic

Regardless of whether synthetic traffic is used for privacy-preserving data sharing or data augmentation (i.e. supplementing a small dataset), it is only useful if it preserves important qualities in the original data. Thus, we want to evaluate the quality of synthetic traffic in terms of its *fidelity* to the original traffic, or by its *utility* in downstream tasks. Fidelity metrics measure the statistical similarity of synthetic data to the original. For example, how does the distribution of packet lengths in the original compare to the distribution in the synthetic? Utility-based evaluations measure how well the

synthetic traffic can substitute for the original traffic in specific downstream tasks, such as training a classifier or regressor.

Evaluation for either component has trade-offs. Utility-based metrics suffer from being task-dependent, while fidelity metrics are task-agnostic. As we will see, synthetic traffic that is useful for one task (*e.g.,* , classification) may be much worse for another task (*e.g.,* regression). However, fidelity metrics have the awkward property that "perfect" synthetic data fidelity is tantamount to being completely identical to the original. This fact poses difficulties for both data sharing and data augmentation. Perfect fidelity traffic risks privacy leakage in the former case, while being of little use in the latter. Additionally, fidelity and utility evaluations aren't always in agreement. As we will see, synthetic traffic can have high fidelity and low utility, and vice versa. Thus, a comprehensive evaluation of synthetic traffic will include both fidelity and utility measurements. To summarize our three methods of evaluation:

Privacy. In general, a generative model preserves privacy if private information in the training data cannot be unintentionally learned from the model or the model's outputs (*i.e.,* synthetic traffic). There are many ways to evaluate a generative model's privacy. In this work, we use MIA success rate and ε as metrics for privacy.

Fidelity. Fidelity is the distance between the distribution of values in synthetic and real data. Fidelity is task-agnostic, and does not depend on how the synthetic data is used.

Utility. Utility captures how well synthetic data replaces real data in downstream tasks. In this work, we use the performance of regressors and classifiers trained on either data as metrics for utility.

3 Membership Inference

In this section, we evaluate the vulnerability of generative models for network data to MIAs. Specifically, we evaluate one traffic-attribute generator (NetShare) and two raw packet generators (NetDiffusion and NetSSM) against a white-box, binary classifier-based MIA and report the results. We then examine the benefit of applying differential privacy (DP) to all models, and re-evaluate the same MIA construction on the DP model versions, exploring any improvements in resilience to membership inference. Interestingly, we observe that MIA performance in these generative network data models is highly variable across both non-DP and DP settings.

3.1 Motivation

MIAs are used to determine whether generative networking models leak non-public information about the training set. Identifying potential privacy leakages is necessary for network operators to ensure regulatory compliance. In particular, data is sometimes subject to export control regulations. To comply, local data controllers train a model and share synthetic data across geographic borders. If the generative models are public, an adversary could use an MIA to learn which traces were used to train the model. Likewise, generative model creators themselves could use MIAs to audit model compliance to ensure that it doesn't leak what data the model was trained on. Beyond compliance, protecting against MIAs is necessary to protect business secrets. Jin *et al.* [34] discuss one scenario where one network operator can use an MIA to determine whether a rival network operator runs a specific application in house. In addition, the choice of training data can greatly impact model performance [24, 40], so protecting that choice can be necessary to protect one's competitive edge.

3.2 Threat Model

We consider an adversary that can query a pre-trained, generative network data model that produces synthetic network data (*e.g.,* time series data for traffic attributes, raw PCAPs). The adversary knows the composition of the targeted models inputs and outputs (*i.e.,* modality, any pre-processing steps necessary prior to feeding them to the model), the model's underlying architecture and training pipeline (*i.e.,* training algorithm/loss function, hyperparameters), and the general population or distribution from which the target model's training dataset was sampled. Importantly, the adversary has *white-box* access to the model, and can access both the target model's output (*i.e.,* decoded output/prediction [if applicable]) and intermediate state (*i.e.,* raw, final layer prediction vectors/logits). The adversary uses this information to obtain a sample data record (set/sequence of traffic attributes, raw packets), and queries the target model with this record to determine its membership status. The MIA is successful if the adversary can correctly determine the status of the data record as a member or non-member of the target model's training dataset. The above assumptions are both common and realistic, with several existing works for membership inference adopting a similar framing [25, 39, 47, 51]. It is important to note that in this threat model, the adversary hopes to determine if any specific *network trace* was used to train a generative network model, not any traffic from a specific *user* (*i.e.,* TraceBleed [34]).

3.3 Setup

We first describe the different models and datasets used to evaluate membership inference in generative network data models. We then describe the mechanism of our MIA, with respect to this setup.

3.3.1 Models. We evaluate the open-sourced implementations of NetShare [48], NetDiffusion [49], and NetSSM [50]. We choose these models as they are the best performing open-sourced methods. All models used for our non-DP evaluation are trained "as is," meaning we follow the instructions for training and generation provided in their respective documentations, and make no changes to the repo code found in the main branches at the time of writing. We make slight code changes for our DP evaluation (described in Section 3.5), though these changes still do not alter the core model architecture or training objective. In total, we train 1,000 disjoint, generative network models to conduct the shadow training procedure for a binary classifier-based MIA. Specifically, 180 models are used to evaluate the default model training scenario (no DP), while the remaining 720 are used to evaluate the benefits of adding four different thresholds of ε-dictated noise. We briefly describe each model type that we evaluate membership inference on below:

NetShare. NetShare [73] is a generative adversarial network (GAN)-based, traffic attribute generator that trains on and generates a pre-defined set of continuous valued flow and packet header-level (up to the network/OSI layer 3) features in tabular form. NetShare treats each feature or "column" of its tabular input as an independent time series, and optimizes towards minimizing the distance

between the values of its predicted time series and the ground truth. Notably, the original work presents results for the utility of synthetic network data generated from a NetShare model trained with DP as measured by statistical similarity to ground-truth traffic attribute values. Unfortunately, this evaluation uses a high privacy budget ($\varepsilon = 24.24$) that may not provide practical privacy preservation utility [22] [18, pages 25 and 257], and does not examine if either non-DP or DP model variants are vulnerable to MIAs.

NetDiffusion. NetDiffusion [49] is a raw packet generator that trains low-rank adaptation (LoRA) "adapter" matrices [29] on image representations of raw packet traces for use with the Stable Diffusion latent diffusion text-to-image model [54]. Specifically, image representations of packet traces are converted from the nPrint format [28], where each comprising packet is one-hot encoded to an array of length 1,088, and each array index represents a possible assignment of a bit in the IPv4 (480 bits), ICMP (64 bits), TCP (480 bits), and UDP (64 bits) headers to $\{-1, 0, 1\}$. Here, -1 denotes absence/non-applicability of a bit, while 0 and 1 denote a bit set to either value, respectively. Each packet-representative array is then mapped to the three RGB color channels, and the mappings for all packets in a trace are stacked to form a $1{,}088 \times 1{,}024$ image, where 1,024 is the maximum number of packets learnable by NetDiffusion. NetDiffusion trains on these images by adding noise and minimizing the mean squared error between its estimate of the added noise and the true added noise. During generation, noise is diffused towards the signals learned during training to an output image, which is then parsed to the nPrint format and raw PCAP. Notably, the NetDiffusion pipeline is not fully generative as it applies heuristic-based corrections to incomplete and/or malformed packets during the final image-to-PCAP conversion.

NetSSM. NetSSM [9] is a raw packet generator built on the Mamba state-space model (SSM) architecture [12, 23], that trains on tokenized representations of raw packet traces. Packets of raw traces are tokenized by performing a one-to-one mapping of each comprising byte to a token of its decimal value. The tokenized representations that comprise each trace are then concatenated to a single sequence, delimited by a special `<|pkt|>` token. NetSSM treats the task of synthesizing traces as a self-supervised sequence generation problem, minimizing the cross-entropy loss between each predicted next token in a sequence and the ground truth during training. In comparison with NetDiffusion, NetSSM can learn from and generate PCAPs of substantially longer length (in packets) due to its linear scaling, and does not require post-generation correction to ensure protocol compliance.

3.3.2 Datasets. We train models on two datasets, each comprised of packet capture (PCAP) files and representative of different traffic workloads and network topologies. We split each dataset to its comprising application traffic classes to be aligned with the standard training approach for the network data generators we evaluate (generating traffic for a single application or workload). This results in six sub-datasets that we train network data generators on, and evaluate for susceptibility to membership inference. We briefly describe each dataset below:

CIC-IDS2017 (CICIDS). The CIC-IDS2017 dataset [56] comprises over 50 GB of various attack traffic (124,391 PCAPs) collected from an experimental testbed of 14 victim hosts using three operating systems (Windows, Ubuntu, macOS) and two attacker hosts across five days. We use a version of the dataset modified for the objective of passive OS detection that truncates each PCAP to 10 packets [28]. The truncated packet dataset is also well suited to helping with the training complexity of our evaluation; training on shorter traces has lower overhead, and in our context, learning from longer traces will not provide substantial value. Specifically, all three of our evaluated models are not dependent on the length of input to effectively learn, due to a lower complexity abstraction of network data being used during training. NetShare learns from and generates only 12 pre-defined features, NetDiffusion three RGB color values mapping to the nPrint representation, and NetSSM 256 tokens mapping to the value of bytes. Thus, longer trace samples are desirable for generating more *longitudinally* realistic content (*e.g.,* behaviors of different workloads), but do not help the models any better understand what their output should "look like" (*i.e.,* tabular features for NetShare, raw packets for NetDiffusion and NetSSM) any better than shorter traces. Finally, we split this dataset into three sub-datasets, each corresponding to a specific operating system and with a subsampled size of 10,000 PCAPs.

Video Conferencing (VCA). The video conferencing dataset is sourced from the "real-world data" collected by Sharma *et al.* [57] and consists of call traffic for 915 Meet, Teams, or Webex calls, each 15-20s in duration. The calls are conducted on Raspberry Pi (RPi) computers located in 15 households spanning different neighborhoods, speed tiers, and service providers in a major North American city. Each RPi is directly connected to a home router, and streams a prerecorded video over a virtual camera interface. To create more trace samples for evaluation, we split each PCAP in this dataset to its corresponding flows based on 5-tuple (*i.e.,* source /destination IP address and ports, transport protocol), yielding 182,259 PCAPs. For the reasons noted in the description of the CICIDS dataset, we similarly truncate each PCAP to 10 packets. Finally, we divide these captures into three sub-datasets, each corresponding to a respective VCA and with a subsampled size of 10,000 PCAPs.

3.3.3 Attack. We evaluate the vulnerability of each generative network data model (when trained both with and without DP) to a white-box, binary classifier-based MIA.

Design. We choose the white-box access scenario and binary classifier-based attack methodology for several reasons. First, the ML-for-networking domain has experienced an increasing number of efforts publicly releasing their model code, datasets and/or pre-trained checkpoints [15, 30, 31, 42, 70]. In fact, to the best of our knowledge, *all* publicly accessible (*i.e.,* queryable either via remote API or when locally cloned) generative network models have their full source code released [48–50]. Thus, the white-box access scenario best suits the data available in the current landscape of ML-for-networking models. Second, we evaluate a binary classifier-based MIA because it most comprehensively evaluates susceptibility to membership inference, and because the technique is model-agnostic, allowing for comparison across models. The metric-based MIA approach would require evaluating multiple attacks tailored to fit the different architecture and/or learning objectives of each model. Prior work has demonstrated the effectiveness of this MIA approach against both the GAN and DDPM architectures used by NetShare

and NetDiffusion [51]. While we did not find efforts evaluating MIAs against SSMs, prior works have used shadow training-based attacks against other token-based generative models [7, 68, 75].

The largest downside of the binary classifier-based MIA approach is its high overhead complexity. The total combined wall time needed to train and extract data from our 1,000 shadow models to build our attack model training dataset exceeded one week. While this is expected and consistent with the nature of the approach, metric-based MIAs tailored to different generative network data models may provide similar attack performance with lower overhead. We leave exploring this direction to future work.

Implementation. Our MIA follows the shadow training process. We first partition each sub-dataset into a 50:50 split of 5,000 PCAPs each. Using the first partition, we randomly select 2,500 PCAPs to represent the target model's training (member) data, and use the remaining 2,500 PCAPs to represent held out non-member data. We randomly sample from the second partition 10 times to create 10 independent shadow model training (member) datasets and 10 held out validation (non-member) datasets, also both of size 2,500 PCAPs. Here, we follow the experimental setup described by Shokri *et al.* where no overlap exists between the target (*i.e.,* first partition) and shadow training datasets, but different shadow model training datasets may overlap with each other [59].

We then use the trained shadow models to generate attack model training datasets (corresponding to each sub-dataset and network data generator pair). Each dataset consists of the (`model_attribute`, `label`) pairs from all 10 shadow models for a subclass, where the `model_attribute` differs for each of our evaluated models, but corresponds to the item most analog to the prediction vector described in the original work [59]. We extract `model_attributes` by inputting each sample from its respective shadow training and held out non-member sets to a model, grabbing the corresponding `model _attribute` item, and labeling it as 1 if the sample originates from the training set (member), or 0 if from the validation set (non-member). This results in 50,000 (`model_attribute`, `label`) samples for each sub-dataset and network data generator pair. We split these at a 70:30 ratio to create attack model train/test sets with 35,000 and 15,000 samples, respectively. Finally, we train binary classifier attack models on each of the corresponding datasets, and evaluate these models' performance on the corresponding original hold out sets from the first partition, for each evaluated subclass.

3.4 Evaluation on Non-Differentially Private Models

We first evaluate MIAs on generative network data models trained without DP. We create 180 shadow model training datasets (18 model/subclass pairs $\times$ 10 samples), and train a corresponding 180 models. Next, we train 18 attack models for each model/subclass pair. Table 2 presents an overview of attack efficacy for each of the four models we evaluate. We find MIA performance to vary across the different network data generators we evaluate. We provide further background on the `model_attribute` and attack model classifier for each model, and interpretation of the MIA results below.

3.4.1 NetShare. We use the model pipeline found in the NetShare repository, first converting PCAPs of each shadow dataset to their tabular representation of a fixed set of 15 features about each packet in a capture. Specifically, these features are a packet's: source and destination IP address and port, transport protocol, timestamp, total packet length, and IP header version, header length, TOS/DSCP field, identification, flags, offset, TTL, and checksum. Differing from the other models we evaluate, NetShare trains only on a single PCAP. As such, we concatenate all packets comprising all traces of a shadow training dataset to a single PCAP, convert it to its tabular representation of features, and train NetShare models on these captures using the default settings set in the repository. After training, we build the attack model training datasets by using the GAN discriminator scores output for each generated sample as the `model_attribute`, labeling scores from shadow models conditioned on their original training data as members, and scores generated from shadow models conditioned on samples held out from the training dataset as non-members.

We train one-class support vector machine-based attack models using the `sklearn` [52] implementation with default parameters, for each of the six attack model training datasets. Each classifier is trained for 1,000 steps, and we normalize each training dataset independently using `sklearn StandardScaler` prior to its use for training. We then evaluate each classifier on its corresponding original held out split from the first partition of our dataset, as described in Section 3.4. We find that all attack models correctly classify true members at a high rate (precision ≥ 0.88) and do not frequently misclassify non-members as members (FPR ≤ 0.20).

3.4.2 NetDiffusion. We follow the data pre-processing procedure detailed in the NetDiffusion repository, converting PCAP files first to the nPrint format [28], then to PNG images, and finally to (image hash, prompt embedding) pairs. We then train NetDiffusion models on these input pairs using the default settings set in the repository. Finally, we build the attack model training datasets by using shadow models' predicted noise on images from their training and held out sets as the `model_attribute`, labeling them accordingly.

We train fully connected, two-layer (size 256) neural networks with batch normalization and ReLU activation functions as our attack models, for each of the six attack model datasets. Each model is trained for 10 epochs using the Adam optimizer with learning rate of $1e{-}5$ and weight decay of $1e{-}7$. We find that our attack classifiers perform poorly across all six subclasses, never exceeding precision better than random guessing. This outcome is surprising, given prior efforts that show diffusion-based models are vulnerable to MIAs, and that they are more vulnerable than GAN-based models [51, 55]. We thus explore this outcome in further detail to determine if NetDiffusion, or network data in particular, comprise particular qualities that result in poor MIA performance.

We begin by examining the images used to fine-tune NetDiffusion, as the image-based representations of network data differ considerably from conventional images. Images from standard datasets (*e.g.,* MNIST [38], CIFAR-10 [37]) contain well-defined qualities (*e.g.,* foreground/background objects, edges) that can be isolated during the denoising process. These qualities typically slightly vary even for images of the same class, providing diverse signals for a model to learn from [3, 8, 64]. In contrast, NetDiffusion's network data images (examples shown in Figure 1) do not have these qualities. The majority of these images' content consists of a fixed

Table 2: Performance of a binary classifier-MIA against non-differentially private generative models for networking data.

Dataset \ Model	NetShare [73]			NetDiffusion [33]			NetSSM [9]		
	Precision	Recall	FPR	Precision	Recall	FPR	Precision	Recall	FPR
CICIDS Mac	0.88	0.87	0.20	0.49	0.67	0.68	0.92	0.92	0.03
CICIDS Ubuntu	0.99	0.98	0.00	0.50	0.40	0.40	0.93	0.93	0.05
CICIDS Windows	0.93	0.93	0.09	0.50	0.47	0.48	0.90	0.90	0.05
VCA Meet	0.92	0.91	0.01	0.50	0.67	0.66	0.97	0.97	0.02
VCA Teams	0.89	0.87	0.01	0.50	0.33	0.33	0.98	0.98	0.01
VCA Webex	0.93	0.93	0.08	0.50	0.68	0.68	0.98	0.98	0.01

color representing -1 (absence/non-applicability) in the nPrint format. Consequently, we want to determine if learnable signals exist in the images used to fine-tune NetDiffusion. To do so, we calculate the entropies for 6,000 randomly sampled NetDiffusion training images (1,000 per subclass) and compare these values against the entropies for 1,000 MNIST and CIFAR-10 dataset images. Across all subclasses, the average entropy is 0.03 bits, while the entropy for the MNIST/CIFAR-10 datasets are 1.61 and 6.96 bits, respectively. This confirms there are minimal signals representative of the original traffic for NetDiffusion to learn from, and it instead learns to reproduce the fixed color/content common to both member and non-member images. As the fixed content is identical in either group, our attack models fail to differentiate between the two. One plausible explanation for this inconsistency is NetDiffusion's post-hoc correction of generated traces for ensuring protocol compliance.

(a) CICIDS Mac **(b)** CICIDS Windows **(c)** VCA Meet

Figure 1: Sample training images for NetDiffusion shadow models.

3.4.3 NetSSM. We use the data pre-processing pipeline described in the NetSSM repository, converting the packets in PCAP files to sequences of raw bytes in decimal representation (*i.e.,* a value $[0-255]$) saved as strings. Each string is then converted to its tokenized form using a custom tokenizer. We then train NetSSM models on these tokenized representations using the default settings. We build the attack model training dataset by running a single forward pass of our shadow NetSSM models on both the tokenized sequences from their training and validation sets, extracting the resulting unnormalized final scores (logits) for each token for each model as the `model_attribute`, and labeling them accordingly.

We train neural network-based attack models with the same configuration as described for NetDiffusion for each of the six attack model datasets. The resulting classifiers can successfully discern membership status with high precision, recall, and low false positive rate, across all subclasses we evaluate.

Takeaway: NetShare and NetSSM show clear vulnerability to membership inference. MIAs evaluated on NetDiffusion yield poor, but inconclusive performance, potentially due to its end-to-end pipeline (training data representation, post-processing).

3.5 Does Differential Privacy Help?

We next evaluate the vulnerability of generative network data models trained with DP to MIAs. Here, we use each shadow training dataset four times to train DP-model variants with four different privacy budgets. Specifically, we use $\varepsilon \in \{0.1, 0.5, 2.0, 4.0\}$ as suggested by Nanayakkara *et al.* [46] that aim to balance recommendations from academic DP literature and those used in the real-world. We fix $\delta = 1e{-}5$ for all values of ε. Prior to training, we make small modifications to the original code for NetShare, NetDiffusion, and NetSSM to wrap model logic in a `PrivacyEngine` from the Opacus Python library [74], a standard method for integrating DP in existing model pipelines. Specifically, Opacus provides an efficient implementation of DP optimizers (*e.g.,* SGD, Adam) in which the gradient for each individual training sample is computed, clipped to bound its sensitivity on model updates, and has DP noise added before being backpropagated through the model for training. For NetShare specifically, we follow the same "as is" training approach as described in Section 3.3.1, and create DP models by finetuning from the final non-DP checkpoint for 10 epochs using DP. Neither the original NetDiffusion nor NetSSM works include code or evaluation of DP variants. The original paper describes this as allowing NetShare to "achieve a better privacy-fidelity tradeoff than the naive approach of training the GAN from scratch with DP." In total, we train 720 (ε, δ)-DP shadow models (18 model/subclass pairs $\times$ 10 samples $\times$ 4 privacy budgets) using the same setup for each of our evaluated models as described in the previous section. We then train 72 attack models for each distinct model/subclass/privacy budget triplet, again using the same `model_attribute` definition and attack model architecture for each model as defined earlier. Our expectation is that MIA performance will be inversely proportional to the magnitude of DP noise added (*i.e.,* privacy budget ε) during model training. Figure 2 shows an overview of the aggregate, mean MIA performance across both the CICIDS and VCA datasets, for all privacy budgets of our evaluated models. Appendix Table 1 contains the detailed results of this evaluation. Similar to the non-DP case, we find MIA performance to vary across models, and privacy budgets. We describe our findings for each model below.

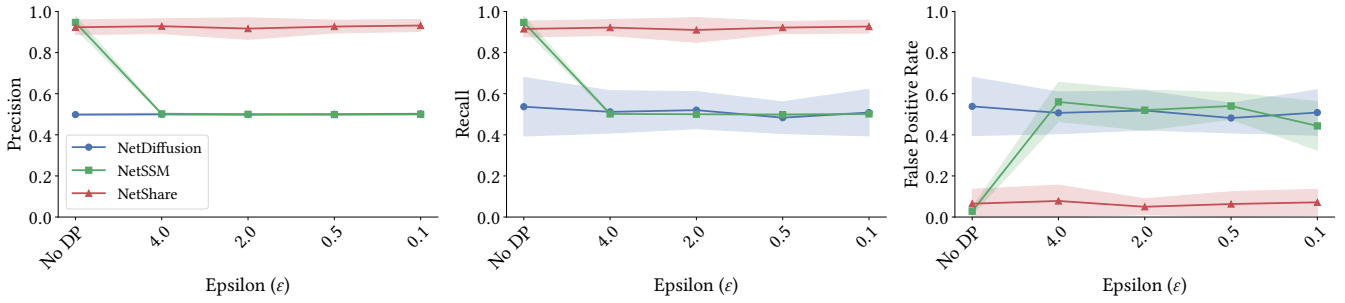


Figure 2: Aggregate mean MIA performance for models trained under different differential privacy budgets (ε) across both the CICIDS and VCA datasets. Smaller values of ε denote more added DP noise. Colored bands represent ±1 standard deviation from the mean.

3.5.1 NetShare. MIA performance on the NetShare models trained with DP strays from our expectation. Specifically, all three performance metrics across subclasses remain nearly identical to the non-DP case. We attribute this to *mode collapse* in NetShare, a common failure mode in GAN-based models where a model fails to learn equally from training data and instead produces repetitive output corresponding to the few "modes" which produce the highest discriminator scores. In NetShare's input features, a number of features (*e.g.,* IP Version, Protocol, TTL) may have a single, or few discrete values. Other features that typically have a larger range (*e.g.,* IP addresses, ports) may instead also have low variance, due to the small network topologies of traffic in our datasets (and network datasets generally). Thus, we hypothesize that NetShare's generator learns to produce these common, low variance features comprising the majority of its feature space to maximize discriminator confidence, while typically higher variance, continuous-valued features (*e.g.,* timestamp, checksum) become increasingly ignored. Resultingly, NetShare's generated data aligns with the "canonical shape" of common, low variance features found in network data.

We validate this hypothesis by comparing differences in variance of normalized feature values between the ground truth and DP synthetic data. We randomly select 1,000 samples for each subclass from the ground truth and DP synthetic data respectively, and in each sample, compute the variance of each feature. Across all subclasses, we find the median feature variance of the synthetic data to be 99.46% less than the ground truth, and the median maximum feature variance per sample to be 98.89% less than the ground truth. This confirms mode collapse, where the generator near exclusively produces the low variance features observed during training, and the discriminator overfits to these values, learning patterns that maximize its score. As our attack models learn from these scores, this behavior is preserved, resulting in highly similar MIA performance as compared to the non-DP case.

3.5.2 NetDiffusion. We observe MIA performance similar to the non-DP case when evaluating the DP models, with attack precision never significantly exceeding random chance, and recall/FPR fluctuating across subclasses. As verified in the previous section, the imaged-based representations of PCAPs used to train NetDiffusion have substantially lower entropy than typical images used to train diffusion models. Thus, any additional perturbations to these data, *e.g.,* DP's manipulation of model gradients which from the start contain weak signals for generation, will have little effect. We confirm this hypothesis by computing the average prediction error (MSE between predicted and actual noise added to the image) between the predicted noise maps for the same input image, but in two cases: (1) during DP training after gradients have been clipped or had noise added, and (2) during normal, non-DP training. We find that the average prediction error for both (1) and (2) are nearly the same, indicating that there is no strong membership signal that exists in the predicted noise.

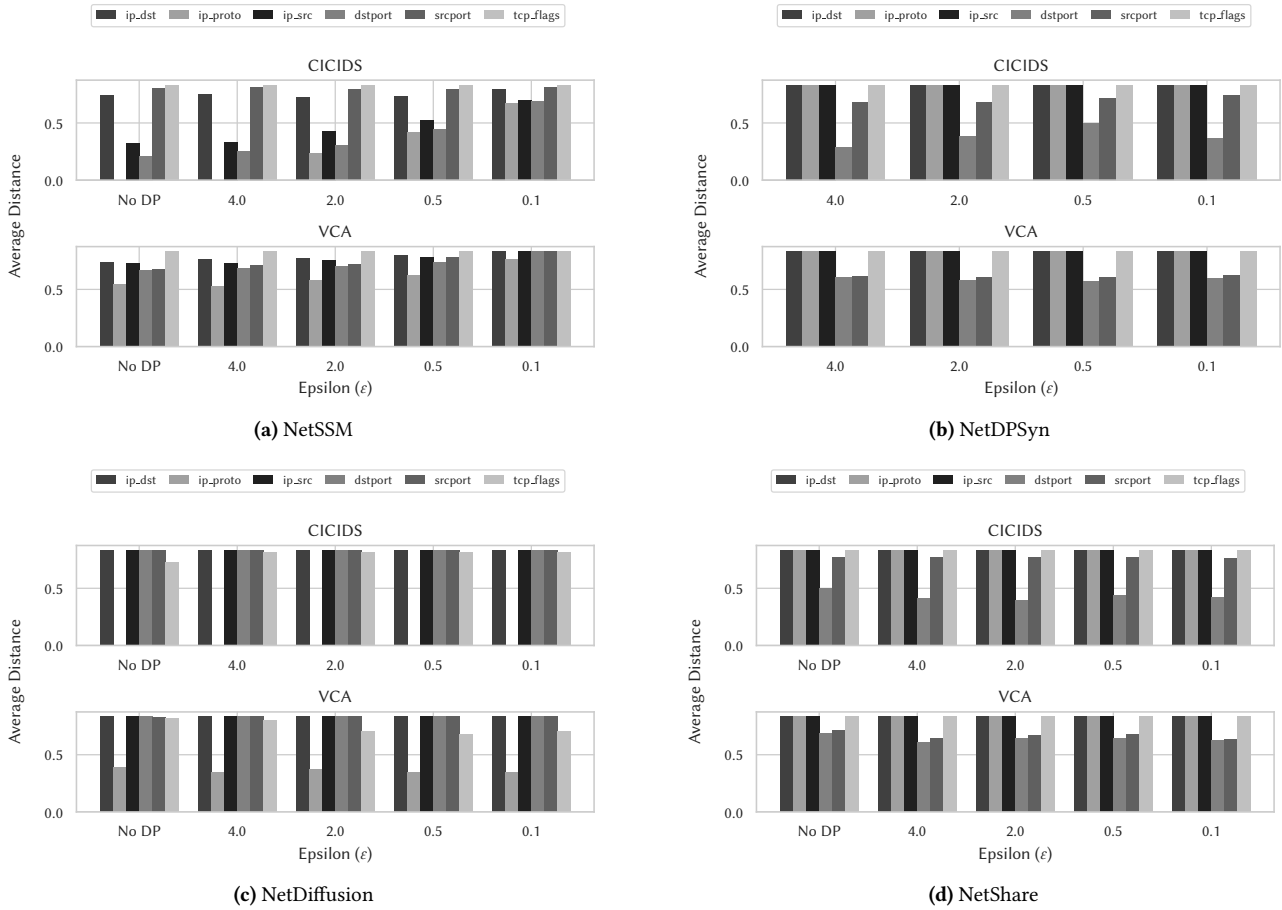
3.5.3 NetSSM. DP noise appears to successfully mitigate our MIA models trained on NetSSM (`model_attribute`, `label`) pairs, across all four privacy budgets. Precision and recall drop to no higher than 0.51 across values of ε, and false positive rates further jump from single-percentages to a median of 0.53. We again analyze how the DP optimizer interferes with the training paradigm of NetSSM, to explain this drop in performance. In contrast to NetShare or NetDiffusion whose loss functions directly minimize unbounded, distance between predicted and actual values, NetSSM's cross-entropy loss minimizes on an indirect, *bounded* value: the difference between predicted and actual probabilities for token selection. As such, any clipping or noise added to training sample gradients during NetSSM's optimization step can directly compete with the true "signal" originally present in the gradient which the model is intended to learn. When this noisy gradient is converted to a bounded probability, even small differences in the gradient may be enough to shift the most probable token from the desired result to another that is completely irrelevant.

Takeaway: DP defends against MIAs only when the model shows standard overfitting that DP noise can address.

Standard overfitting (NetSSM): training with DP results in significant reduction in MIA efficacy, with the DP optimizer reducing memorization of token sequences.

Structural overfitting (NetShare): DP has little effect on MIA success rate, even with strict privacy budgets ($\varepsilon = 0.1$), as the DP noise cannot resolve mode collapse.

Undergeneralization (NetDiffusion): DP has little effect on MIA success rate, as the added noise cannot outweigh the impact of undergeneralization observed in the non-DP models.



(a) NetSSM

(b) NetDPSyn

(c) NetDiffusion

(d) NetShare

Figure 3: Average JSD between real and synthetic header fields, by model and dataset. A lower distance means the distributions are closer, and the synthetic data "looks" more like the original data.

4 Fidelity and Utility of Synthetic Data

In this section, we analyze how both the fidelity and utility of synthetic data varies across models and privacy budgets. We first present our results in terms of ε, and provide plausible explanations for these results in Section 4.3. We compare against both ε and MIA success because DP can potentially be used to defend against other types of MIAs. Thus, our findings on the relationship between fidelity/utility and ε can be extended to future work examining other MIAs. In addition to the generative models (NetSSM, NetDiffusion, NetShare) evaluated in the previous section, we include NetDPSyn [63], a non-ML-based approach for adding DP to network data by directly injecting noise to the data's marginals. Because NetDPSyn is not ML-based, it is not possible to evaluate a MIA against it, and thus, we did not include it in the previous section. Instead, we use it as a benchmark in this analysis, to distinguish traffic quality changes attributable to DP versus the generation process of the other ML-based methods we evaluate.

4.1 Fidelity

We first evaluate the fidelity of synthetic traffic using the Jensen-Shannon Divergence (JSD) for categorical values (*e.g.,* IP addresses and ports) and the Wasserstein's Distance (EMD) for numerical values (*e.g.,* packet length). Both metrics measure the distance between two distributions (*i.e.,* the synthetic and real traffic) and are measured extensively in prior work. We define similarity based on the distribution of values because (1) each header field can be one of many values, and (2) we care about the actual value of the field, rather than frequency counts of each value present. As each trace has ten packets, we compute the similarity for each field, for each packet, then average across all discrete and continuous values.

Figure 3 shows the JSD between the original and synthetic traffic for categorical header fields. Appendix Table 2 contains a more direct table showing the minimum and maximum average JSD across categorical features for different ε-thresholds. The lower the JSD, the more similar the distributions. Despite minor variations in per-field fidelity, NetDiffusion, NetShare, and NetDPSyn exhibit remarkable consistency across ε values, with the average JSD across each ε value within 0.05. We hypothesize that this minimal variation owes to how the noise practically affects the modeling dynamics of each generator. In both NetDiffusion and NetShare, the models use a loss function that minimizes the distance between the ground truth and the noise-added data. Although this noise is sufficient to satisfy the

privacy guarantees of ε-DP, the added noise does not mechanically hinder this loss function objective. As a result, the overall structure and characteristics of the traffic will persist across ε values, with only minor variations (*e.g.,* minor sub-millisecond differences) in the `timestamp`. NetDPSyn, meanwhile, does not learn any representation of data, but instead applies DP to its input in a way that minimally impacts the resulting data.

In contrast, the fidelity of NetSSM traffic is directly proportional to ε. As Figure 3a shows, the similarity of four of the header fields is lowest at $\varepsilon = 0$ (*i.e.,* no DP) and highest at $\varepsilon = 0.1$ (*i.e.,* the highest level of DP). This trend makes sense because of how noise must be added in NetSSM. Noise added during NetSSM training has a large effect on its cross-entropy objective function. Here, slight perturbations are not minor additions of floats to output, but instead the predicted probabilities used towards token selection. Thus, as ε grows, the actual semantic relationships between tokens in its input sequence can be learned, instead of spurious signals.

The relationship between numerical feature fidelity and ε is similar. Figure 5 shows the Wasserstein Distance between the original and synthetic traffic. For NetDiffusion, non-DP synthetic traffic has consistently higher fidelity than DP traffic. But for NetSSM and NetShare, the fidelity of non-DP synthetic traffic is commensurate with traffic at $\varepsilon = 0.5$. Again, the takeaway is that adding DP does not necessarily degrade the fidelity of the synthetic traffic.

Finally, comparing fidelity across datasets, we see that high fidelity for one dataset does not imply high fidelity in another. This observation is most clear with NetSSM. As the TTL plots in Figure 5 show, NetSSM has an identical distribution to the original CICIDS traffic, but varies more for the VCA dataset. The same is true for the categorical features in Figure 3a.

Takeaway: For all models, DP does not necessarily lead to lower fidelity synthetic traffic. While non-DP NetDiffusion has higher fidelity than DP-NetDiffusion, DP-NetSSM and NetShare can have commensurate fidelity with the non-DP versions.

4.2 Utility

Evaluating the fidelity of synthetic data only tells half the story, as data is only valuable if it has utility. One common use of synthetic network data is towards training downstream models for classification or prediction tasks. In this section, we compare the performance of three models, two classifiers and one regressor, trained on synthetic traffic with those trained on the original traffic. Our results show that the utility of synthetic traffic can vary depending on the downstream task. Indeed, synthetic traffic with high fidelity and good performance on one task can perform comparatively worse on other tasks. Thus, whether synthetic traffic is viable is closely tethered to the intended application.

4.2.1 Classification Tasks. In our first classification task, we train a `sklearn` Random Forest (RF) classifier with default configuration to predict the source operating system based on a 10 packet trace. The second task is similar, but with the goal of predicting the type of video-conferencing application (VCA) instead of OS. We pick two tasks of variable difficulty to better characterize the effect of DP. The OS detection task is easy because an RF classifier trained on the original traffic achieves perfect accuracy. Indeed, distinguishing between Windows and the other two classes is possible looking only at the TTL field, while the Mac traffic only originates from a single (unique) IP source address.

OS Classifier. We observe that adding DP to the synthetic traffic either improves or leaves unchanged the accuracy of every classifier. Figure 4 (left) presents the accuracy of a classifier trained on synthetic traffic by model and by ε. Classifier F1 scores follow the same trend (Appendix Figure 9). Classifiers trained on NetDiffusion traffic perform consistently poorly across different ε, achieving 0.33 accuracy. Examining the generated traffic reveals that the TTL field is inconsistent even within the same class. As for the distribution of source IP addresses, NetDiffusion includes additional addresses even in the non-DP traffic, making it difficult to distinguish Mac traffic from the others.

Classifiers trained on NetSSM traffic are also consistent across different ε, but perform well. This is attributable to NetSSM's traffic closely mimicking the original traffic's TTL and unique IP source address both with no DP noise, and when $\varepsilon = 4$. For other values of ε, the perfect accuracy is possibly explained by NetSSM's higher fidelity for the IP source address and TTL.

For classifiers trained on NetShare traffic, the DP-traffic trained classifier has greater accuracy than the non-DP-traffic trained classifier. That said, this result owes to the fact that the model overfit to the training data. DP can act as a regularizer, correcting such overfitting [16]. Thus, although using DP will not, in general, improve downstream utility, it can correct overfitting for a given model or dataset. We explore this notion in more detail in Section 4.3.

Finally, we look at classifiers trained on NetDPSyn traffic. Though there is no non-DP NetDPSyn to compare to—that would simply be the original traffic—we can see how classifier accuracy varies with ε. Notably, the performance of the trained classifier does not vary meaningfully across ε, achieving between $75 - 77\%$ accuracy at each level. Looking at the distribution of important features, we see that the NetDPSyn preserves the TTL values of the original traffic and thus perfectly classifies Windows traffic based on the TTL. And although Mac and Ubuntu have distinct sets of IP source addresses, the classifier is unable to distinguish the two classes.

Comparing fidelity scores to classification accuracy, we observe that lower fidelity does not always imply better downstream performance. For example, consider the fidelity of IP source addresses of Mac traffic in NetSSM generated traffic. It is perfect or close to perfect for non-DP and $\varepsilon = 4.0$ (*i.e.,* it only contains the unique address present in the original traffic). But for other values of ε, the fidelity is much worse, ranging from 0.23 to 0.67 at $\varepsilon = 0.1$. The upshot: there are on average 35 different source IPs at $\varepsilon = 0.1$, compared to only a single IP address in the original traffic. Nevertheless, each classifier trained on NetSSM generated traffic is perfect. Thus, even if we know ex ante which features will be important for downstream tasks, looking only at the fidelity may be insufficient to predict the utility of the traffic. With this said, we note that fidelity is not uncorrelated with utility. Looking at Figure 5 (top), we can see that the fidelity of the TTL field in NetDiffusion traffic is far worse than the others. Likewise, the fidelity of the IP source address field in NetSSM traffic is better than the others.

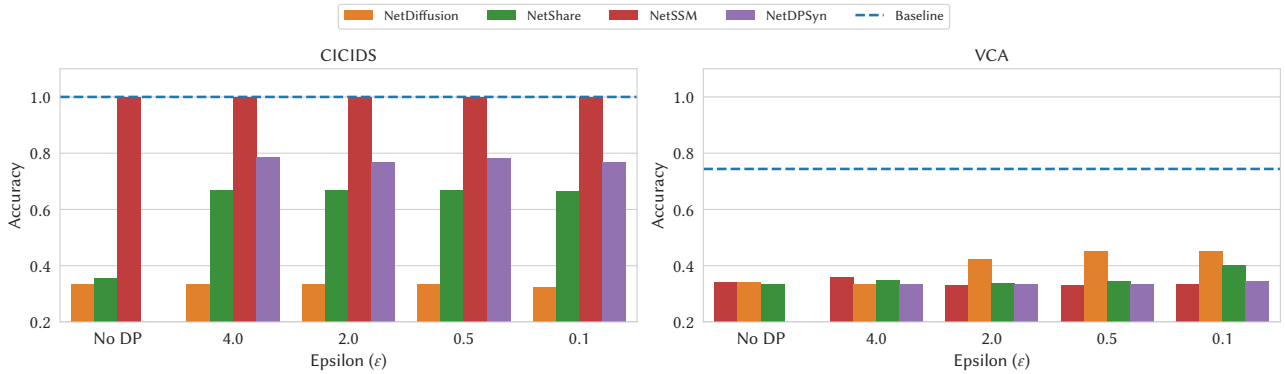


Figure 4: Accuracy of random forest classifiers trained on the referenced model, against epsilon (DP) value.

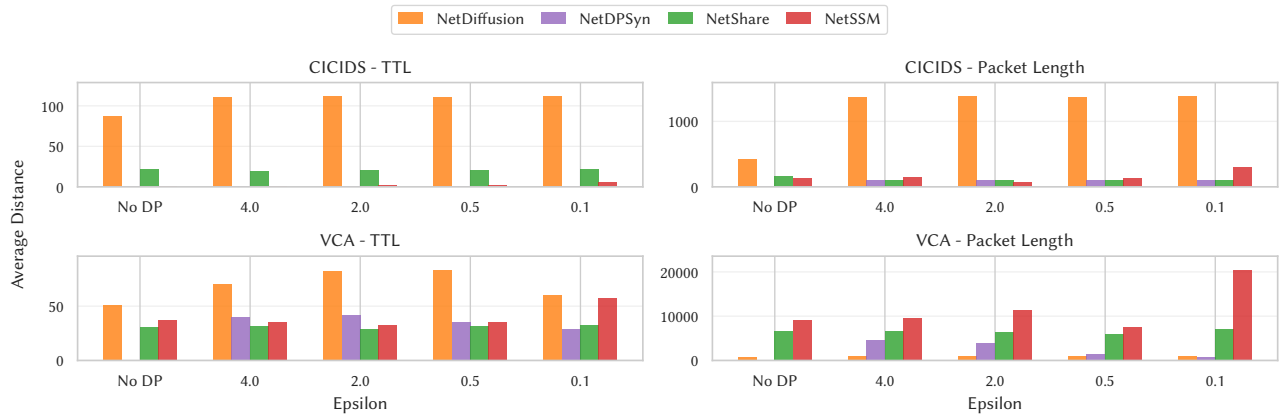


Figure 5: Wasserstein Distance between original traffic and synthetic traffic. A lower average distance means the distributions are closer, and the synthetic traffic has higher fidelity.

VCA Classifier. Like the CICIDS dataset, the VCA dataset is composed of 10-packet long network traces from three different applications. We again train an RF classifier on the synthetic data and test using the original traffic. Looking at Figure 4, the baseline model trained on the original traffic achieves 76% accuracy. Unlike the OS classifier, there are no features in the VCA data that uniquely identify the class of traffic. The baseline classifier relies primarily on the TTL fields and the length of the packet to predict to which class the traffic belongs.

The difficulty of the task is reflected in the quality of the classifiers trained on the synthetic traffic, and no single model dominates across tasks. NetSSM, NetShare, and NetDPSyn trained classifiers achieve at or just above random, while NetDiffusion accuracy peaks at 0.46 at $\varepsilon = 0.5$. One explanation for NetDiffusion's superior performance is that it has the best packet length fidelity at $\varepsilon = 0.5$. This in turn is likely because of NetDiffusion's post-processing feature, which ensures that the generated packet length is always less than 1500 bytes and that the packet length is no less than the sum of the header and transport length.

Beyond the fact that the VCA classification task is harder than the OS task (as measured by baseline classifier accuracy), another explanation for the reduced accuracy is that, as Figure 5 (bottom) shows, fidelity of numerical header fields is lower for the VCA dataset than the CICIDS dataset. The fidelity of packet lengths is especially poor because the size of video conferencing packets has much higher variance than the traffic in the CICIDS dataset. Table 3 in the Appendix shows that the distribution of packet lengths in the VCA dataset is more spread out and has a much longer tail.

Takeaways: ① The utility of synthetic traffic for classification tasks can vary depending on the type of traffic the generator aims to emulate and the task difficulty and ② low fidelity synthetic traffic does not imply poor downstream utility. Synthetic traffic for all models trains VCA classifiers with poor accuracy, but NetSSM-generated traffic trains OS classifiers with perfect accuracy despite poor fidelity on important features in the synthetic traffic. Similarly, NetDiffusion and NetShare-generated traffic have the lowest and second-lowest fidelity across datasets, respectively, but train VCA classifiers that perform better than classifiers trained on NetSSM and NetDPSyn traffic.

4.2.2 Regression Task. While classification tasks are common in networking, regression-based analyses (*e.g.,* for load-balancing and queue size management) are also widespread. Thus, evaluation of synthetic network data utility should include non-classification tasks.

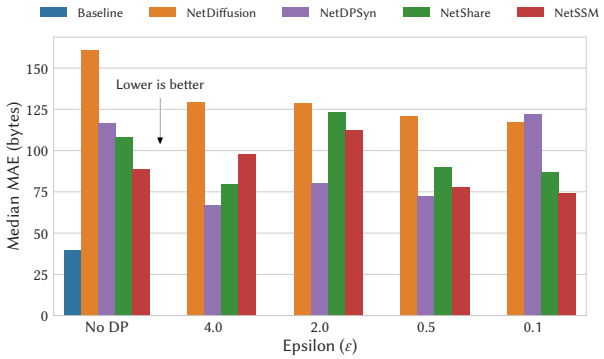


Figure 6: Median MAE for regressors trained to predict next packet length using DP/non-DP synthetic data.

One shortcoming of typical generative approaches is that they do not capture timing information well. In other words, the value in the timestamp field of synthetic traffic has little meaning. With this in mind, we choose a regression task that does not rely on that timing information, but relies only on the ordering of packets. We train a Darts N-BEATS neural network [27] to predict the next packet length based on the previous ten packets. To test, the model is given a trace as input and predicts the next packet length based on the previous ten packets. We then calculate the mean average prediction error (MAE) for each trace, and repeat on 100 traces.

We find that the regressor trained on differentially private synthetic traffic achieves at least as good performance (for some ε) as the regressor trained on non-DP traffic. Appendix Figure 6 shows how the median MAE (mean absolute error) in bytes varies with model and privacy budget. This trend is clearest with NetDiffusion, where the prediction accuracy gradually improves from 160 bytes mean absolute error to under 120 bytes. NetDiffusion is likely the worst because its packet length fidelity is worst among the models (see Figure 5). The trend in model performance for NetShare and NetSSM is, however, less consistent. For one, the model trained on $\varepsilon = 4.0$ NetShare traffic is best, while for NetSSM it is $\varepsilon = 0.1$. Although more work is necessary to understand ex ante which ε level is best, neither generative model produces uniformly worse synthetic traffic when DP is added.

Finally, although the models trained on synthetic traffic perform worse than the models trained on real traffic, the absolute error difference is only on the order of tens of bytes. That said, this error magnitude might still be intolerable given the median packet length in the CICIDS dataset is only 48 bytes (see Table 3). Thus, as generative models are developed, it is important to evaluate the synthetic traffic on both regression and classification tasks.

Takeaway: For all models, regressors trained on DP synthetic traffic can perform at least as well as those trained on non-DP synthetic traffic, for some ε. However, real traffic-trained regressors greatly outperform any synthetic traffic-trained regressor.

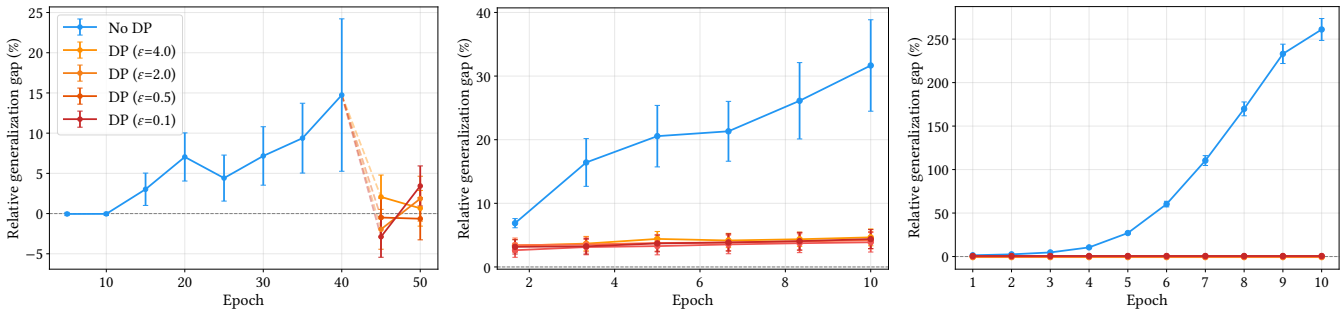
4.3 Analysis

The findings presented in this section thus far appear, at first glance, counterintuitive. How can models trained under greater noise (*i.e.,* DP) output data that is of higher quality? We previously discussed one possible explanation, overfitting, in the context of how it prevents membership inference (Section 3.5). In this section, we will re-examine overfitting but in the context of how it manifests in the fidelity and utility of generated data. Additionally, we use these results in conjunction with findings from previous sections to provide guidance to practitioners on epsilon selection and privacy preservation considerations for generative network data models in Appendix C.

We evaluate each model's tendency to overfit as measured by the *generalization gap*, defined as the absolute difference between a model's test and training loss, between previously observed (training) and unobserved (testing) data. Large differences between these values indicate overfitting, while values close to zero indicate that the model can generalize well to its universe of potential inputs. We compute the mean generalization gap for all three of our evaluated models. Specifically, for each shadow model (both non-DP and DP), we calculate at each epoch its mean respective loss on both its training data, and held out test data for the same class. For each train/test loss pair, we then calculate the relative generalization gap ($\frac{\texttt{test\_loss} - \texttt{train\_loss}}{\texttt{train\_loss} \cdot 100}$) for that epoch. We then find the mean relative generalization gap of all shadow models of the same class and DP threshold, at each epoch. This provides an overview of how each model's generalization gap changes during training.

Figure 7 visualizes the progression of the generalization gaps for each DP threshold in the VCA dataset, for all three models we evaluate. Additional visualizations for the CICIDS dataset can be found in Appendix Figure 10. Points on each line are the mean generalization gap of all combined classes in the dataset (*i.e.,* Meet, Teams, Webex). We show only the DP model generalization gaps at epochs 45 and 50 for NetShare, as these models are finetuned from the final non-DP epoch for 10 additional epochs (Section 3.5). Broadly, we observe that regardless of model type, shadow models trained in the normal context (no DP) overfit to their training data, while those trained with DP do not. However, the nature of this overfitting differs substantially across architectures. We explore these model-specific mechanisms below.

NetShare. The non-DP NetShare models display the least amount of overfitting across the three models with a mean relative generalization gap of 14.74% at its final checkpoint. While notable, this amount of memorization is comparatively low when viewed in the scope of NetDiffusion and NetSSM, and we interpret it as consistent with the mode collapse behavior we describe in Section 3.5.1. Specifically, as the model learns to produce only a small set of common modes, it will not directly memorize the values of all features, capping but not preventing overfitting. Finetuning with DP to create the DP shadow models results in a substantial reduction of the relative generalization gap, dropping to nearly zero after five epochs. These results align with the observations for both fidelity (Section 4.1) and downstream utility (Section 4.2), where DP NetShare models may yield higher fidelity and classifier performance. Some nuance exists in this process as it is not necessarily the case that DP improves the generator directly, but that the added noise disrupts mode collapse. However, this benefit is transient, as signs of overfitting return (albeit less aggressively) after 10 epochs of finetuning, suggesting that even under added noise, mode-collapse can re-establish itself in the model.



(a) NetShare **(b)** NetDiffusion **(c)** NetSSM

Figure 7: Models' relative generalization gap. Non-DP models have notable overfitting, while DP models have minimal or no overfitting.

NetDiffusion. Figure 7b depicts that non-DP NetDiffusion shadow models have a gradually increasing generalization gap as training progresses, settling at 31.66%. This may at first appear contradictory to our MIA findings, where NetDiffusion undergeneralizes due to the low entropy of its training data. Here, NetDiffusion does not overfit to an individual sample, but the shared average structure of the images as a whole. Because NetDiffusion's LoRA adapter matrices are low-dimensional, they quickly converge to reproduce this mean image, resulting in a large generalization gap despite the absence of member-specific memorization. Consistent with DP training behavior for all models, the generalization gap significantly decreases to be less than 5% for all ε-thresholds. DP noise prevents the LoRA weights from converging to the training set mean, introducing variance that improves the diversity of generated output. With respect to utility, the reduction in generalization gap under DP training is consistent with the results for both downstream utility tasks. For the classification task, this is most notable in the results for the VCA classifier, where no-DP and low amounts of DP noise ($\varepsilon = 4.0$) yield similarly low accuracy, but higher amounts of added noise increase performance. For regression, the regularization effect appears the strongest and most monotonic across ε-thresholds of any model. However, for all utility results, we must also keep in mind NetDiffusion's post-processing step which ensures protocol compliance (*e.g.,* valid packet lengths) in the final output. It is possible that a combination of added DP noise and post-generation corrections allow NetDiffusion to output more diverse data, and yield the observed results.

NetSSM. NetSSM models show the most clear separation of non-DP and DP training behavior with the final non-DP checkpoint having a relative generalization gap over 2.5× more than its testing loss, while DP models have a near zero gap over all epochs. This is consistent with our DP findings; as NetSSM optimizes its token-based, cross-entropy loss, even small gradient perturbations can shift the most probable token from the correct value to an entirely unrelated one, resulting in NetSSM being the only model that is reliably protected against MIAs by adding DP noise (Section 3.5.3). For fidelity, NetSSM was the sole model that displayed a more predictable degradation of fidelity (in the majority of fields) as more DP noise was added. Here, the memorization highly benefits the computed JSD distance metric, as NetSSM outputs many values that are very close to, or exact matches with the ground truth. For some utility tasks (*e.g.,* the "easy" OS classifier), this results in constant high accuracy, as it only relies on a single TTL field that NetSSM can memorize. For other tasks, the same sensitivity that provides privacy benefits makes utility under DP fundamentally unpredictable. The added DP noise may sometimes disrupt unimportant tokens (preserving utility), but may also corrupt features critical to the downstream task (degrading utility). This explains why regression performance varies inconsistently across ε rather than improving monotonically as with NetDiffusion: without a post-processing safety net to rescue corrupted output, each corrupted token propagates directly to the generated traffic.

Takeaway: DP itself does not *directly* improve the fidelity or utility of synthetic network data. Instead, DP can incidentally have a regularizing effect, preventing overfitting that occurs due to the nature of network data and how generative network models learn. As a result, the models capture not only the few dominant traffic signals, but also tail-case traffic that both is in the true distribution, and important for downstream tasks.

5 Conclusion

Generating and using synthetic network traffic is a promising alternative to using real network traffic. However, it is important to evaluate the synthetic traffic in terms of both privacy and quality, lest we risk compromising one over the other. This work offers an empirically grounded starting point for reasoning about these attributes. Specifically, we analyzed the privacy-preserving properties and quality of synthetic traffic generated by state-of-the-art models, both with and without DP. Using the success of an MIA as our metric for privacy, we found that DP worked as a defense in only one model (NetSSM). As for the effect of DP on traffic quality, we found that models trained on differentially-private synthetic traffic tended to perform at least as well as models trained on non-DP synthetic traffic. However, because different network workloads may have highly variable dynamics, and generative network models continue to evolve, we caution practitioners from generalizing our findings to their deployment scenarios, and encourage evaluating the tradeoffs of integrating DP or other privacy preserving technologies on a case-by-case basis.

Acknowledgments

We thank the anonymous reviewers and revision editor for their feedback and suggestions. We also thank Arjun Bhagoji for early discussion on this project. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] 13th National People's Congress. 2021. Personal Information Protection Law of the People's Republic of China. 30th Meeting of the Standing Committee of the Thirteenth National People's Congress. http://en.npc.gov.cn.cdurl.cn/2021-12/29/c_694559.htm Citing Chapter III: Rules for Cross-border Provision of Personal Information.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[3] Dmitry Baranchuk, Ivan Rubachev, Andrey Voynov, Valentin Khrulkov, and Artem Babenko. 2021. Label-efficient semantic segmentation with diffusion models. *arXiv preprint arXiv:2112.03126* (2021).

[4] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. 2012. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks* 56, 15 (2012), 3531–3547.

[5] Tobias Bühler, Roland Schmid, Sandro Lutz, and Laurent Vanbever. 2022. Generating representative, live network traffic out of millions of code repositories. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. 1–7.

[6] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*. 5253–5270.

[7] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*. 2633–2650.

[8] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. 2023. Diffusiondet: Diffusion model for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*. 19830–19843.

[9] Andrew Chu, Xi Jiang, Shinan Liu, Arjun Bhagoji, Francesco Bronzino, Paul Schmitt, and Nick Feamster. 2025. NetSSM: Multi-Flow and State-Aware Network Trace Generation using State-Space Models. *arXiv preprint arXiv:2503.22663* (2025).

[10] ciscotrex2023 2024. The CISCO TRex Tool. https://trex-tgn.cisco.com/. [Online; accessed 31-May-2024].

[11] Tianyu Cui, Xinjie Lin, Sijia Li, Miao Chen, Qilei Yin, Qi Li, and Ke Xu. 2025. TrafficLLM: Enhancing Large Language Models for Network Traffic Analysis with Generic Traffic Representation. arXiv:2504.04222 [cs.LG] https://arxiv.org/abs/2504.04222

[12] Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *International Conference on Machine Learning (ICML)*.

[13] Rockfish Data. 2025. Central Global Model Accuracy - Rockfish Documentation. https://web.archive.org/web/20250423193318/https://docs142.rockfish.ai/uc-demos/use-case-constraint-centralized-MLOps.html Last accessed 23 April 2025.

[14] Damien Desfontaines. 2021. A list of real-world uses of differential privacy. https://desfontain.es/blog/real-world-differential-privacy.html. Ted is writing things (personal blog).

[15] duowuyms. 2024. duowuyms/NetLLM. https://github.com/duowuyms/NetLLM Last accessed 21 November 2025.

[16] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. 2015. Generalization in adaptive data analysis and holdout reuse. *Advances in neural information processing systems* 28 (2015).

[17] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. 2019. Differential privacy in practice: Expose your epsilons! *Journal of Privacy and Confidentiality* 9, 2 (2019).

[18] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science* 9, 3–4 (2014), 211–407.

[19] Government of Brazil. 2020. General Personal Data Protection Law. https://lgpd-brazil.info/chapter_05/index

[20] Government of Russia. 2014. Federal Law No. 242-FZ. https://wilmap.stanford.edu/entries/federal-law-no-242-fz

[21] Government of Vietnam. 2022. Decree No. 53/2022/ND-CP. https://www.trade.gov/market-intelligence/vietnam-cybersecurity-data-localization-requirements

[22] Andy Greenberg. 2017. How One of Apple's Key Privacy Safeguards Falls Short. https://www.wired.com/story/apple-differential-privacy-shortcomings/. Wired.

[23] Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752* (2023).

[24] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644* (2023).

[25] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2017. LOGAN: evaluating privacy leakage of generative models using generative adversarial networks. *arXiv preprint arXiv:1705.07663* (2017), 506–519.

[26] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. 2008. Network simulations with the ns-3 simulator. *SIGCOMM demonstration* 14, 14 (2008), 527.

[27] Julien Herzen, Francesco Lässig, Samuele Giuliano Piazzetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan Kocisz, Dennis Bader, Frédérick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and Gaël Grosch. 2022. Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research* 23, 124 (2022), 1–6. http://jmlr.org/papers/v23/21-1177.html

[28] Jordan Holland, Paul Schmitt, Nick Feamster, and Prateek Mittal. 2021. New Directions in Automated Traffic Analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 33663383. https://doi.org/10.1145/3460120.3484758

[29] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[30] IDP-code. 2024. IDP-code/TrafficFormer. https://github.com/IDP-code/TrafficFormer Last accessed 21 November 2025.

[31] InspiringGroup-Lab. 2024. InspiringGroup-Lab/Brain-on-Switch. https://github.com/InspiringGroup-Lab/Brain-on-Switch Last accessed 21 November 2025.

[32] A. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville. 2022. AI/ML and Network Security: The Emperor has no Clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA.

[33] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. 2024. NetDiffusion: Network Data Augmentation Through Protocol-Constrained Traffic Generation. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 1, Article 11 (Feb. 2024), 32 pages. https://doi.org/10.1145/3639037

[34] Minhao Jin, Hongyu He, and Maria Apostolaki. 2025. Assessing User Privacy Leakage in Synthetic Packet Traces: An Attack-Grounded Approach. *arXiv preprint arXiv:2508.11742* (2025).

[35] Jack Kiefer and Jacob Wolfowitz. 1952. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics* (1952), 462–466.

[36] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

[37] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[38] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[39] Klas Leino and Matt Fredrikson. 2020. Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In *29th USENIX security symposium (USENIX Security 20)*. 1605–1622.

[40] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems* 37 (2024), 14200–14282.

[41] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 464483. https://doi.org/10.1145/3419394.3423643

[42] linwhitehat. 2022. linwhitehat/ET-BERT. https://github.com/linwhitehat/ET-BERT Last accessed 21 November 2025.

[43] Kyle MacMillan, Jordan Holland, and Prateek Mittal. 2020. Evaluating snowflake as an indistinguishable censorship circumvention tool. *arXiv preprint arXiv:2008.03254* (2020).

[44] Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. 2024. LLM Dataset Inference: Did you train on my dataset?. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=Fr9d1UMc37

[45] Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. Membership Inference Attacks against Language Models via Neighbourhood Comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*. 11330–11343.

[46] Priyanka Nanayakkara, Mary Anne Smart, Rachel Cummings, Gabriel Kaptchuk, and Elissa M Redmiles. 2023. What are the chances? explaining the epsilon parameter in differential privacy. In *32nd USENIX Security Symposium (USENIX Security 23)*. 1613–1630.

[47] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Comprehensive privacy analysis of deep learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, Vol. 2018. 1–15.

[48] netsharecmu. 2023. netsharecmu/NetShare: (SIGCOMM '22) Practical GAN-based Synthetic IP Header Trace Generation using NetShare. https://github.com/netsharecmu/NetShare Last accessed 7 May 2025.

[49] noise lab. 2025. noise-lab/NetDiffusion_Generator. https://github.com/noise-lab/NetDiffusion_Generator Last accessed 7 May 2025.

[50] noise lab. 2025. noise-lab/netssm. https://github.com/noise-lab/netssm Last accessed 7 May 2025.

[51] Yan Pang, Tianhao Wang, Xuhui Kang, Mengdi Huai, and Yang Zhang. 2023. White-box membership inference attacks against diffusion models. *arXiv preprint arXiv:2308.06405* (2023).

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[53] Jian Qu, Xiaobo Ma, and Jianfeng Li. 2024. Trafficgpt: Breaking the token barrier for efficient long traffic analysis and generation. *arXiv preprint arXiv:2403.05822* (2024).

[54] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

[55] Ilana Sebag, Jean-Yves Franceschi, Alain Rakotomamonjy, Alexandre Allauzen, and Jamal Atif. 2025. On the MIA Vulnerability Gap Between Private GANs and Diffusion Models. *arXiv preprint arXiv:2509.03341* (2025).

[56] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1 (2018), 108–116.

[57] Taveesh Sharma, Tarun Mangla, Arpit Gupta, Junchen Jiang, and Nick Feamster. 2023. Estimating webrtc video qoe metrics without using application headers. In *Proceedings of the 2023 ACM on Internet Measurement Conference*. 485–500.

[58] Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2023. Detecting Pretraining Data from Large Language Models. arXiv:2310.16789 [cs.CL]

[59] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[60] Nirhoshan Sivaroopan, Dumindu Bandara, Chamara Madarasingha, Guillaume Jourjon, Anura P Jayasumana, and Kanchana Thilakarathna. 2024. Netdiffus: Network traffic generation by diffusion models through time-series imaging. *Computer Networks* 251 (2024), 110616.

[61] Joel Sommers, Hyungsuk Kim, and Paul Barford. 2004. Harpoon: a flow-level traffic generator for router and network tests. *ACM SIGMETRICS Performance Evaluation Review* 32, 1 (2004), 392–392.

[62] Liwei Song and Prateek Mittal. 2021. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX Security Symposium (USENIX Security 21)*. 2615–2632.

[63] Danyu Sun, Joann Qiongna Chen, Chen Gong, Tianhao Wang, and Zhou Li. 2024. Netdpsyn: synthesizing network traces under differential privacy. In *Proceedings of the 2024 ACM on Internet Measurement Conference*. 545–554.

[64] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. 2023. Emergent correspondence from image diffusion. *Advances in Neural Information Processing Systems* 36 (2023), 1363–1389.

[65] The European Parliament and the Council of the European Union. 2016. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union. http://data.europa.eu/eli/reg/2016/679/oj Citing Chapter V: Transfers of personal data to third countries or international organisations (Articles 45-46).

[66] Van Tran, Shinan Liu, Tian Li, and Nick Feamster. 2025. Quantifying the Privacy Implications of High-Fidelity Synthetic Network Traffic. *arXiv preprint arXiv:2511.20497* (2025).

[67] Kashi Venkatesh Vishwanath and Amin Vahdat. 2009. Swing: Realistic and responsive network traffic generation. *IEEE/ACM Transactions on Networking* 17, 3 (2009), 712–725.

[68] Jason Wang, Jeffrey Wang, Marvin Li, and Seth Neel. 2023. MoPe: Model Perturbation-based Privacy Attacks on Language Models. In *Socially Responsible Language Modelling Research*.

[69] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*. 143–157.

[70] wangtz19. 2024. wangtz19/NetMamba. https://github.com/wangtz19/NetMamba Last accessed 21 November 2025.

[71] Kun Yang, Samory Kpotufe, and Nick Feamster. 2020. Feature extraction for novelty detection in network traffic. *arXiv preprint arXiv:2006.16993* (2020).

[72] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE, 268–282.

[73] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. 2022. Practical GAN-based synthetic IP header trace generation using NetShare. In *Proceedings of the ACM SIGCOMM 2022 Conference* (Amsterdam, Netherlands) *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 458472. https://doi.org/10.1145/3544216.3544251

[74] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch. *arXiv preprint arXiv:2109.12298* (2021).

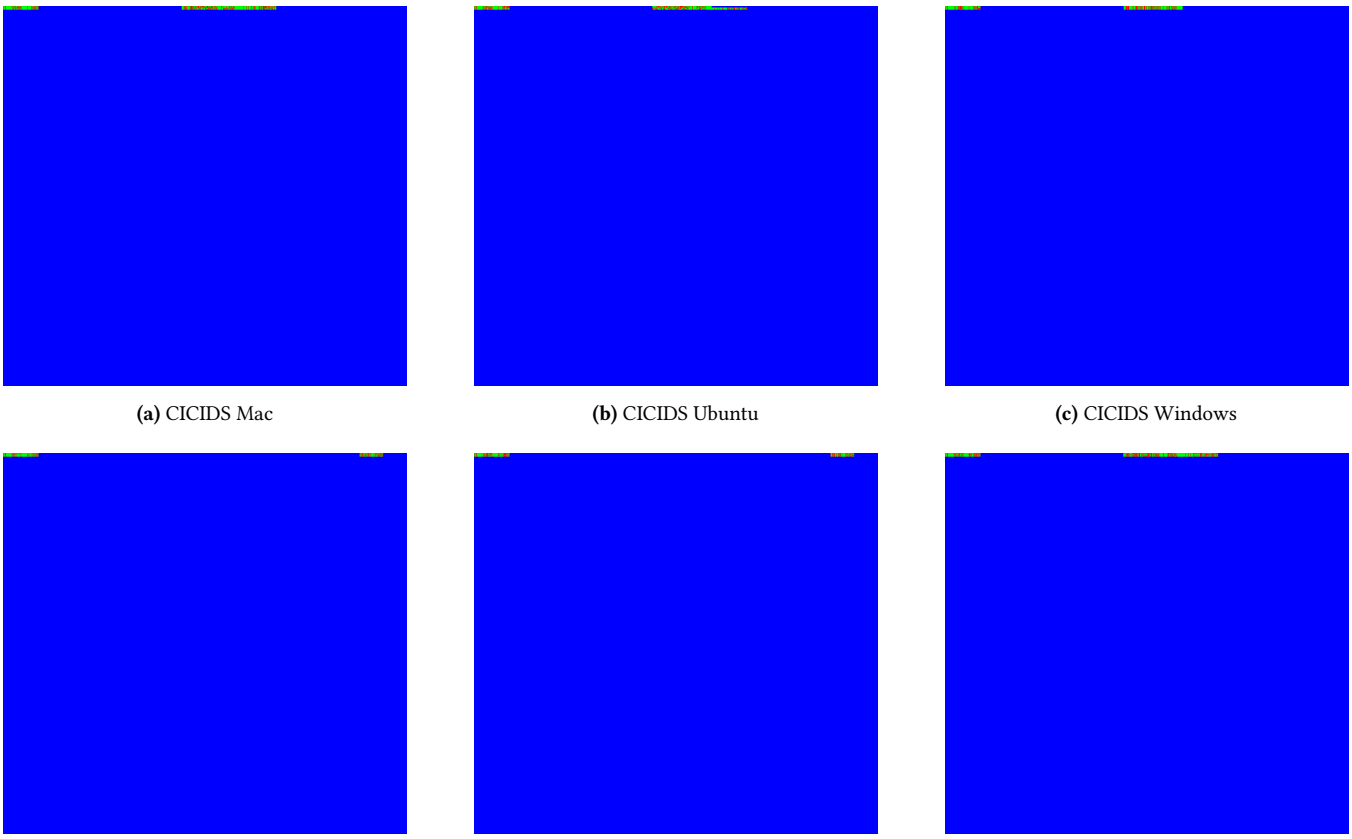
[75] Sheng Zhang, Hui Li, and Rongrong Ji. 2024. Code membership inference for detecting unauthorized data use in code pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 10593–10603.

[76] Shiyuan Zhang, Tong Li, Depeng Jin, and Yong Li. 2024. NetDiff: a service-guided hierarchical diffusion model for network flow trace generation. *Proceedings of the ACM on Networking* 2, CoNEXT3 (2024), 1–21.

A Membership Inference

Table 1: Full results for our MIA evaluation (Section 3) and fidelity/utility analysis (Section 4).

	Model / Dataset	NetShare					NetDiffusion					NetSSM				
		Privacy			Utility		Privacy			Utility		Privacy			Utility	
		Prec.	TPR	FPR	F1	MAE	Prec.	TPR	FPR	F1	MAE	Prec.	TPR	FPR	F1	MAE
No DP	CICIDS Mac	0.88	0.87	0.20	0.00	169.7	0.49	0.67	0.68	0.50	177.0	0.92	0.92	0.03	1.00	112.1
	CICIDS Ubuntu	0.99	0.98	0.00	0.51	120.9	0.50	0.40	0.40	0.00	201.1	0.93	0.93	0.05	1.00	110.2
	CICIDS Windows	0.93	0.93	0.09	0.13	34.0	0.50	0.47	0.48	0.00	104.0	0.90	0.90	0.05	1.00	38.6
	VCA Meet	0.92	0.91	0.01	0.50		0.50	0.67	0.66	0.03		0.97	0.97	0.02	0.05	
	VCA Teams	0.89	0.87	0.01	0.00		0.50	0.33	0.33	0.03		0.98	0.98	0.01	0.51	
	VCA Webex	0.93	0.93	0.08	0.00		0.50	0.68	0.68	0.50		0.98	0.98	0.01	0.01	
$\varepsilon = 0.1$	CICIDS Mac	0.91	0.90	0.17	0.00	136.3	0.50	0.30	0.31	0.49	197.0	0.51	0.51	0.45	1.00	120.0
	CICIDS Ubuntu	0.99	0.99	0.00	0.67	104.0	0.50	0.43	0.43	0.00	126.9	0.50	0.50	0.51	1.00	85.6
	CICIDS Windows	0.93	0.92	0.13	1.00	21.0	0.50	0.63	0.63	0.00	27.4	0.49	0.50	0.22	1.00	17.8
	VCA Meet	0.92	0.91	0.01	0.52		0.50	0.56	0.56	0.53		0.50	0.50	0.46	0.00	
	VCA Teams	0.90	0.90	0.03	0.00		0.50	0.61	0.61	0.00		0.50	0.50	0.61	0.00	
	VCA Webex	0.94	0.94	0.09	0.39		0.51	0.52	0.51	0.55		0.50	0.50	0.41	0.50	
$\varepsilon = 0.5$	CICIDS Mac	0.89	0.89	0.14	0.00	122.4	0.50	0.47	0.46	0.50	197.0	0.50	0.50	0.54	1.00	120.0
	CICIDS Ubuntu	0.99	0.98	0.00	0.67	120.6	0.50	0.45	0.45	0.00	126.9	0.50	0.50	0.53	1.00	85.6
	CICIDS Windows	0.92	0.92	0.08	1.00	49.0	0.50	0.38	0.39	0.00	34.6	0.49	0.49	0.42	1.00	28.9
	VCA Meet	0.91	0.90	0.01	0.13		0.50	0.54	0.54	0.53		0.50	0.50	0.59	0.00	
	VCA Teams	0.93	0.93	0.01	0.50		0.50	0.62	0.61	0.51		0.50	0.50	0.63	0.50	
	VCA Webex	0.92	0.91	0.14	0.00		0.50	0.44	0.44	0.17		0.50	0.50	0.53	0.02	
$\varepsilon = 2.0$	CICIDS Mac	0.81	0.79	0.09	0.00	194.1	0.50	0.69	0.70	0.00	237.6	0.50	0.50	0.54	1.00	101.1
	CICIDS Ubuntu	0.98	0.98	0.00	0.67	121.8	0.50	0.56	0.55	0.00	122.3	0.50	0.50	0.49	1.00	145.8
	CICIDS Windows	0.92	0.92	0.07	1.00	54.8	0.50	0.43	0.42	0.50	25.9	0.50	0.50	0.65	1.00	90.1
	VCA Meet	0.91	0.89	0.01	0.05		0.50	0.42	0.42	0.53		0.50	0.50	0.50	0.51	
	VCA Teams	0.94	0.94	0.03	0.50		0.49	0.52	0.53	0.49		0.50	0.50	0.34	0.09	
	VCA Webex	0.94	0.94	0.10	0.02		0.50	0.50	0.49	0.00		0.50	0.50	0.60	0.00	
$\varepsilon = 4.0$	CICIDS Mac	0.87	0.86	0.21	0.00	134.3	0.50	0.58	0.57	0.00	214.7	0.50	0.50	0.52	1.00	130.7
	CICIDS Ubuntu	0.99	0.99	0.00	0.67	80.9	0.49	0.47	0.49	0.00	141.7	0.50	0.50	0.42	1.00	125.1
	CICIDS Windows	0.92	0.91	0.14	1.00	22.9	0.51	0.41	0.40	0.50	31.2	0.51	0.51	0.56	1.00	37.1
	VCA Meet	0.92	0.91	0.01	0.14		0.50	0.45	0.44	0.00		0.50	0.50	0.54	0.13	
	VCA Teams	0.95	0.94	0.02	0.52		0.50	0.71	0.70	0.50		0.50	0.50	0.58	0.50	
	VCA Webex	0.92	0.92	0.09	0.00		0.50	0.45	0.44	0.00		0.50	0.50	0.74	0.15	



(a) CICIDS Mac **(b)** CICIDS Ubuntu **(c)** CICIDS Windows

(d) VCA Meet **(e)** VCA Teams **(f)** VCA Webex

Figure 8: Sample training images used to train NetDiffusion shadow models (Section 3.4).

B Fidelity and Utility

Table 2: Min and max average fidelity across categorical features for a given ε value.

Dataset	Model	Min. JSD	Max. JSD	Diff.
CICIDS	NetSSM	0.48	0.75	0.27
	NetDPSyn	0.72	0.76	0.04
	NetShare	0.75	0.77	0.02
	NetDiffusion	0.68	0.69	0.02
VCA	NetSSM	0.70	0.82	0.12
	NetDiffusion	0.73	0.75	0.03
	NetShare	0.76	0.79	0.02
	NetDPSyn	0.75	0.76	0.01

Table 3: Comparison of VCA and CICIDS Datasets

Dataset	Class	Median	Std Dev	Max
VCA	Teams	68.00	783.65	8572.00
	Webex	65.00	610.25	7152.00
	Meet	157.00	779.45	11412.00
CICIDS	Ubuntu	52.00	265.15	5844.00
	Mac	52.00	224.72	1500.00
	Windows	40.00	359.71	6206.00

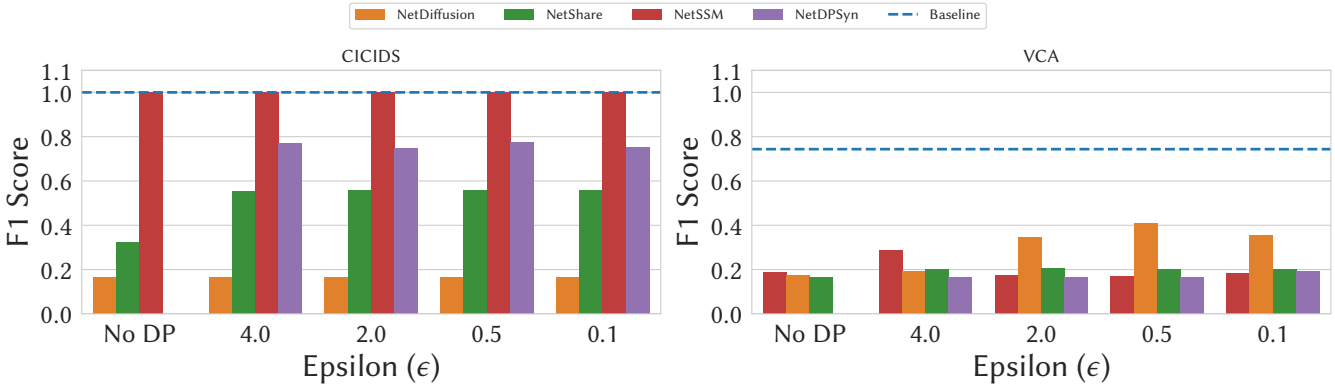
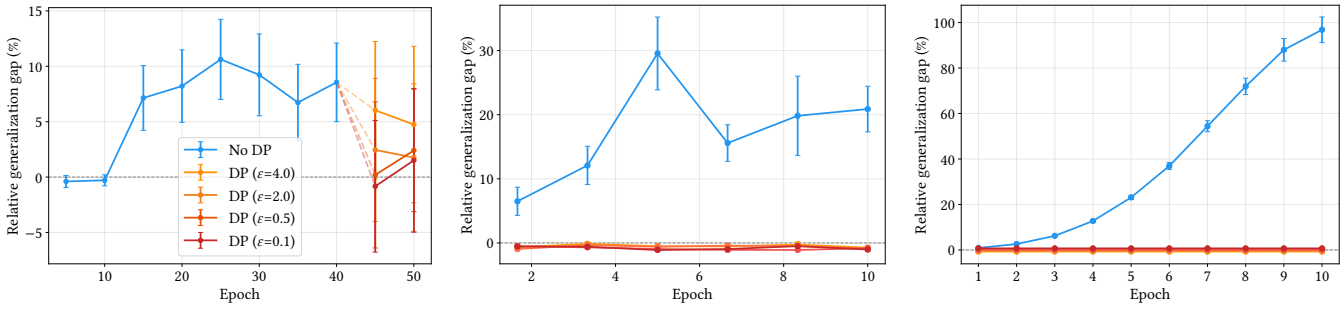


Figure 9: F1 Score of random forests classifiers trained on synthetic traffic from the referenced model, against epsilon (DP) value.



(a) NetShare **(b)** NetDiffusion **(c)** NetSSM

Figure 10: Relative generalization gap across training steps (NetShare, NetSSM) and generation timestamps (NetDiffusion) for the CICIDS dataset.

C Guidance for Practitioners

Practitioners who wish to deploy DP towards preserving the privacy of networking data should assess what qualities of the data require protection and how the data will be used in downstream tasks. Networking data differs from data in other domains due to the many granularities at which it can be collected, nuances in implementation it contains, and the contained qualities one can examine. For instance, collecting data only between devices on a home local area network may reduce concerns of publishing traffic attributes such as IP addresses, as revealing this information has no significant negative effect (these addresses are not publicly routable). To contrast, an internet service or content distribution network provider would likely not wish to release raw IP addresses as these may reveal insight into business operation (*e.g.,* peering agreements, internal quality of service prioritization). At the implementation or protocol level, implicit encryption qualities of some protocols such as HTTPS or DNS over TLS/HTTPS provide direct privacy protection, while others used for the same purpose (HTTP, normal DNS) do not have such guarantees.

We have shown in this work that DP is unreliable at mitigating MIAs in some generative models for network data, not due to DP itself, but domain-specific characteristics about network data and the generative models. Adding in consideration for the search space of all collection and deployment scenarios, it is unfortunately very difficult to suggest epsilon values for use in training models with DP, that broadly satisfy privacy guarantees. Practitioners who use generative models for network data should instead carefully assess what qualities of their network have the potential for privacy leakage given their source collection environment or deployment, with what granularity they plan to release information about their network, and who the potential adversaries that may want to leverage this information may be. As discussed in the first paragraph, some attributes may or may not be meaningful if leaked, and can be ignored for enforcing privacy guarantees. In other cases, practitioners may wish to consider releasing summary statistics of network data in lieu of raw records, or abstract out only the necessary values or fields required for downstream use. To connect with the results we present in Sections 4.2.1 and 4.2.2, our trained OS classifiers rely primarily on TTL (CICIDS subclasses) and source IP (VCA subclasses), while the regressors use only the packet length. Release of only a narrower set of these values and/or a few others could inform downstream models that still provide utility, while knowledge of them alone is likely insufficient for an adversary to make any actionable assumptions about a network. Finally, the nuclear approach may be to not publicly release the weights and checkpoints for a trained generative model for networking data, nor provide query access to it, and to simply publish the generated data (given it satisfies one's definition of privacy preservation).