DPNeT: Differentially Private Network Traffic Synthesis with Generative Adversarial Networks

Liyue Fan, Akarsh Pokkunuru

DPNeT: Differentially Private Network Traffic Synthesis with Generative Adversarial Networks

Liyue Fan and Akarsh Pokkunuru

University of North Carolina at Charlotte, Charlotte NC 28223, USA
{liyue.fan, apokkunu}@uncc.edu

Abstract. High quality network traffic data can be shared to enable knowledge discovery and advance cyber defense research. However, due to its sensitive nature, ensuring safe sharing of such data has always been a challenging problem. Current approaches for sharing networking data present several limitations to balance privacy (e.g., information leakage) and utility (e.g., availability and usefulness). To overcome those limitations, we develop DPNeT, a network traffic synthesis solution that generates high-quality network flows and satisfies (ϵ,δ)-differential privacy. We adopt generative adversarial networks (GANs) to capture the characteristics of real network flows and a similarity-preserving embedding model for mixed-type attributes. Furthermore, we propose new techniques to improve the outcome of differentially private learning and provide the privacy analysis of the overall solution. Through a comprehensive evaluation with large-scale network flow data, we demonstrate that our solution is capable of producing realistic network flows.

Keywords: Differential Privacy · Generative Adversarial Networks · Network Flow Generation

1 Introduction

Sharing fine-grained network traffic data has enabled numerous research studies for knowledge discovery and cyber security applications, such as in anomaly detection [16] and cyber attack classification [29, 4]. However, network traffic data is highly sensitive, e.g., with Internet protocol (IP) addresses and port numbers, etc., which may be used by adversaries to infer private information, e.g., a specific website visited by the user. In the worst case, home and commercial networks may be attacked [5, 27]. Therefore, it is imperative to protect the privacy of individuals and organizations in the published network data. In order to enhance the privacy for sharing network traffic data, many anonymization techniques have been proposed, [30, 3, 20] to name a few. However, it has been shown that inference attacks may still be launched against anonymized network traces [6, 13]. Moreover, it is challenging to quantify the quality of the anonymized data.

Recently, generative adversarial networks (GANs) [10] have been adopted to generate realistic network traffic data, e.g., for sequence of packet sizes [25], network flows [23], and traffic morphing [14]. However, it has been shown that

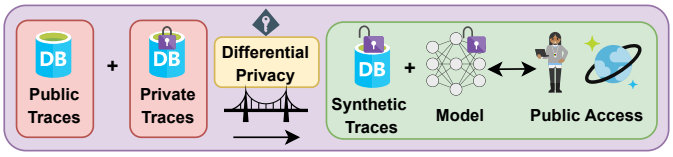


Fig. 1: DPNeT Overview.

deep learning models are subject to various attacks, e.g., inferring membership in the training set [26] and reconstructing training data [9]. Similarly, GAN models do not provide guarantees on what the generated data may reveal about real, sensitive training data. In fact, [12] successfully devised membership inference attacks against target GANs in both white-box and black-box access settings.

To provide rigorous privacy in network traffic synthesis, we propose to adopt differential privacy [7] when training GAN models with sensitive data. Our solution, dubbed DPNeT, builds on recent advances in training generative models (i.e., Wasserstein GAN [11]), and is able to produce *realistic* synthetic data that exhibits high *similarity* to the training data. As shown in Figure 1, the trained models as well as the synthetic traces can be widely shared for research and educational purposes. The specific contributions of our work are as follows:

1. We are the first to develop a differentially private solution based on GANs for synthesizing flow-level network data. The privacy of the training examples is protected via differentially private optimization [1]. To ensure the quality of the synthetic data, we adopt the state-of-the-art methodology for training generative models, as well as an advanced embedding model to preserve similarities between mixed-type feature values.

2. To address the challenges in private learning, such as noisy or non-convergence, we propose two improvement techniques: decaying the clipping bound over epochs and privately selecting the best models across all training epochs. Our empirical analysis shows that decaying the clipping bound outperforms the standard option, i.e., no decay. Furthermore, we show that private model selection significantly improves the quality of the synthetic network flows, compared to the model obtained at the last epoch.

3. We provide privacy analysis results for both GAN training and model selection. In short, we implement Moments Accountant to account for the privacy loss during GAN model training; we further analyze the sensitivity and the privacy guarantees for selecting up to K models. Overall, we show that the DPNeT solution achieves (ϵ,δ)-differential privacy.

4. We conduct an extensive evaluation with large-scale network flows. We quantify the similarity and realism of the synthetic network flows using distributional measures and domain knowledge tests, respectively. Our results examine the impact of privacy and demonstrate the effectiveness of our proposed improvements.

The rest of the paper is organized as follows: we discuss related work in Section 2 and describe fundamental concepts such as differential privacy and

GANs in Section 3. In Section 4, we provide a full description of DPNeT as well as privacy analysis results. In Section 5, we present and discuss empirical evaluation results. Finally, in Section 6 we conclude the paper with brief discussions on future work.

2 Related Work

Network Trace Anonymization. Anonymization techniques for network traffic data have been extensively studied in the last decades. For instance, IP addresses can be obfuscated with prefix-preserving pseudonyms [30] or bucketization [21]. Other features, such as timestamps and ports, can be shifted or suppressed [15, 20]. However, it has been shown that classic anonymization methods are prone to inference attacks [3, 6, 13]. A recent study [19] proposed to create multiple views of the dataset based on the assumed adversarial knowledge, which is not suitable for our setting, i.e., sharing data widely. [17] proposed a differential privacy based solution for analyzing network traces; however, the sharing of private flows has not been discussed.

Network Traffic Synthesis with GANs. With rapid advancements in deep learning and generative models, a few research studies have proposed to incorporate generative adversarial networks (GANs) [10] for generating network traffic data. For instance, Shahid et. al [25] proposes to generate synthetic packet sizes for IoT applications. Unfortunately, the synthesis of other network traffic features was not addressed. Two recent works aim to synthesize network traffic flows for data sharing (Ring et. al [23]) and bypass internet censorship (Li et. al [14]). They are most relevant to this paper, especially Ring et. al [23]. However, GAN models do not provide privacy guarantees on what the generated data may reveal about real, sensitive training data. For instance, [12] successfully devised membership inference attacks against target GANs in both white-box and black-box access settings.

Differential Privacy and Machine Learning. The vulnerabilities of deep learning, e.g., membership inference [26, 12], demonstrate great needs for strong privacy protection for the underlying training data. To this end, differential privacy has been applied to learning deep models [1] to combat such inference attacks. Recent studies adopt the framework in [1] to train GAN models [2, 28] and we have conducted a survey study on those approach in [8]. Based on our survey results, most of the existing studies focus on generating image data, e.g., MNIST, where GAN models can learn input data distributions accurately; and very few studies attempt to publish mixed-type data, with features as challenging as IP addresses. Furthermore, all studies report difficulties and utility loss encountered with differentially private learning, e.g., the generator and discriminator may converge to a noisy equilibrium or do not converge. Our solution aims to address the shortcomings of existing approaches, and the proposed improvements can be applied to general differentially private learning tasks.

3 Preliminaries

3.1 Differential Privacy

The privacy model adopted in our work is differential privacy [7]. By definition, a randomized algorithm A is (ϵ, δ)-differentially private if for any two databases D, D' differ in at most a single record and any subset $S \in Range(A)$:

$$\Pr[A(D) \in S] \leq \exp(\epsilon) \Pr[A(D^{'}) \in S] + \delta \tag{1}$$

Here $\epsilon > 0$ is known as the privacy budget, which bounds the difference between output probabilities of two neighboring databases $D, D^{'}$. In addition, $\delta \in [0, 1]$ accounts for the probability of *bad events* that might lead to a privacy breach. An advantage of DP is its resistance to post-processing [7], i.e., any computation performed on the output of a DP mechanism would not incur additional privacy cost. Other benefits of DP include the lightweight computation compared to crypto-based mechanisms and ease of control over the information leakage with the help of ϵ, δ parameters. Typically, smaller ϵ and δ values indicate stronger privacy protection, and vice versa. There exists a trade-off between preserving privacy and maintaining data utility.

In particular, we are interested in applying DP to deep learning, in order to protect the privacy of training examples. As shown in [1], it can be achieved by sanitizing the gradients during neural network optimization, which ultimately limits the overall influence of any training example on the model. A privacy accountant, i.e., Moments Accountant [1], has been proposed to account for differential privacy across training epochs, which provides stronger estimates of privacy loss compared to other composition theorems [7].

A key question to address in applying (ϵ,δ)-DP is the choice of privacy parameters. It is often seen that $\epsilon > 1$ in private deep learning, e.g., $\epsilon = 8$ as in [1] and up to $\epsilon = 96$ in some studies surveyed in [8]. In this study, we would like to provide privacy protection in deep learning applications without incurring significant utility loss. As a result, we consider $\epsilon \leq 20$. As for δ, to provide individual privacy in case of bad events, we set $\delta = \frac{1}{|D|}$ as recommended in [7].

3.2 Generative Adversarial Networks

Generative adversarial networks (GANs) [10] have become the state-of-the-art method to learn generative models, and has demonstrated superior performance in producing synthetic data that have similar characteristics as real data. A survey analysis for differentially private GANs can be found here [8]. GANs consist of two components, i.e., a generator G and a discriminator D. The problem is formulated as a minimax two-player game with the following objective [10]:

$$\min_G \max_D \ \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]. \tag{2}$$

The generator G learns to capture the original data distribution p_{data} by mapping a latent distribution p_z. Specifically, G takes as input a random noise

z and generates synthetic data. On the other hand, the discriminator D learns to discriminate between samples drawn from p_{data}, i.e., x, and those generated by G, i.e., $G(z)$. D takes a sample as input and returns a score representing whether it is real or synthetic. By generating samples that appear to come from the original data distribution, the goal of the generator is to fool the discriminator. The generator and discriminator are trained simultaneously through an adversarial process: the more the generator improves the quality of synthetic data, the harder it is for the discriminator to distinguish between original and synthetic samples. Wasserstein GAN [11] minimizes the earth-mover distance (i.e., Wasserstein-1 distance) between p_z and p_{data}, and allows more stable and faster training by penalizing the norm of gradient of the discriminator. For those considerations, we will adopt Wasserstein GAN in DPNeT to generate synthetic network flows.

4 DPNeT Solution

Overview. DPNeT for network flow synthesis entails three main steps:

1 Embedding the input network flow records to numerical vectors;
2 Training GAN models using the input vectors and generating synthetic vectors;
3 Decoding the synthetic data to produce network flow records.

To generate private synthetic network flows, we utilize two separate deep learning architectures: an advanced *embedding* model, and the generator and discriminator of *GANs*. Step 1 and 3 rely on the embedding network which transforms network flow records to numerical vectors while preserving data semantics. Step 2 learns to mimic the input data distribution with GAN models, in a differentially private setting.

4.1 DPNeT: Embedding

Network flows are mixed-type data, consisting of both numerical features and categorical features (e.g., IP address, port number, packet size, and flag). To learn from the input data, categorical feature values are often converted into numerical values, e.g., via one-hot encoding. Ring et. al [22] have shown that the standard encoding methods do not capture the relationships between mixed-type features. They modified the Word2Vec model [18] to learn "word" embeddings for network flow features. We adopt a similar architecture for the embedding model in DPNeT.

Specifically, we consider 8 features of each flow, namely Source IP and Port, Destination IP and Port, Protocol, Packets, Bytes, and Duration. As a result, we are able to compute the similarities between Source IP and Source Port, or between Packets and Duration. For each flow, 13 bi-grams are constructed as the training data for the embedding model. Each bi-gram contains an input word, i.e., one of the 8 features, and a context word (expected output), i.e., among

a set of features identified by domain experts for each input feature [23]. The embedding network is a simple neural network with a single hidden layer, which contains 20 neurons in our solution. After the embedding network is trained, it is utilized to produce numerical feature vectors for GANs training set (Step 1 in Overview), with the weights at the hidden layer as feature vectors of the words (i.e., network flow attributes). Given a synthetic feature vector generated by GANs, we retrieve the most similar word in the vocabulary based on cosine similarity in the embedding space (Step 3 in Overview), and output the synthetic network flow records.

In DPNeT, we train the embedding model using a *public* dataset that is disjoint from the sensitive training set for the GAN models. Our consideration is three-fold: (1) utilizing a public dataset enables accurate learning of feature similarities. For instance, we can fine-tune the parameters iteratively without incurring any privacy loss. (2) Thanks to the accuracy, the training set of the embedding model needs not be very large. Additionally, it is easier to sanitize the vocabulary on a smaller dataset, e.g., ensuring certain IP addresses or protocols are not present. (3) The embedding network should learn to encode general network flows independent of GAN training data, e.g., specific network traffic patterns that only occur in the sensitive training data. This also produces an embedding model that can be deployed for other applications.

By design, the vocabulary of the embedding model may not include all words in the GAN training set. We propose to impute the sensitive training data, i.e., replacing the feature values that do not appear in the embedding model's vocabulary with the mode of the feature. Our empirical evaluation confirms the feasibility of using disjoint training sets for the embedding model and GAN models.

4.2 DPNeT: GAN Training

The center of DPNeT is to learn GAN models with differential privacy. The embeddings obtained in Step 1 are concatenated with the remaining features in the network flows, e.g., class, and are provided to the GAN models as training data. To achieve differential privacy, we adopt the deep learning framework proposed in [1] and present the pseudocode in Algorithm 1.

We train our GAN models on N examples for E epochs, on a randomly selected mini-batch of size B with a sampling probability $q = B/N$. For each training example in a batch, the discriminator computes the gradients $g_j(x_k)$ w.r.t the model parameters and the DP mechanism sanitizes the gradients with clipping and perturbation. The clipping of gradient norm is upper bounded by a hyper-parameter γ. Additionally, the variance of the additive Gaussian noise is controlled through σ. The choice of σ and γ values is essential to achieve a balance between data quality and privacy protection. Finally, the discriminator parameters are updated after the completion of each batch. The generator learns to produce network flows through iterative updates using a traditional gradient based optimization approach. The discriminator is trained for E_D epochs per generator epoch, which helps alleviate mode collapse issues.

At the end of each epoch, we save the intermediate generator model G_i, which will be utilized for model selection purposes. Optionally, the clipping bound γ may decay according to a few proposed decay functions; we refer readers to Section 4.3. The privacy loss will be estimated by the Moments Accountant. After E epochs, the algorithm outputs the final generator model G and total privacy spent ϵ_1.

Algorithm 1 DPNeT Training Procedure

Input: training examples $x_1 \cdots x_N$, batch size B, noise σ, $\delta = \frac{1}{N}$, learning rate α, number of epochs E, number of discriminator epochs E_D, clipping bound γ

```
for i = 1 ··· E do
  if i > 1 then
    For each generator parameter θ compute:
    g_i ← Adam ( ▽_θ (1/B) Σ_{i=1}^{θ} −D(G(Z_i)) )
    Update generator: θ_{G_(i+1)} ← θ_{G_(i)} + α g_i
  end if
  for j = 1 ··· E_D do
    Sample L with sampling probability q = B/N
    for each x_k ∈ L do
      Compute gradients: g_j(x_k) ← ▽_θ W(θ_j, x_k)
      Clipping: g_j(x_k) ← g_j(x_k) / max(1, ‖g_j(x_k)‖ / γ)
    end for
    Perturbation: g_j ← (1/|L|) ( Σ_{k=1} g_j(x_k) + N(0, (σ γ)^2 I) )
    θ_{D_(j+1)} ← θ_{D(j)} + α g_j
  end for
  Save the generator as G_i
  Optionally, decay the clipping bound γ (see Sec. 4.3)
  Estimate privacy ε_1 using Moments Accountant
end for
```

Output: Generator G and total privacy spent ϵ_1

Implementation. In DPNeT, we adopt the MLP architecture for both the generator (5 hidden layers) and discriminator (4 hidden layers) as suggested in [23]. Every hidden layer has 1024 units. The choice of hidden layer activation is *ReLU* and *leaky ReLU* with a negative slope of 0.2 for the generator and discriminator, respectively. We use a linear activation for the output layers in both networks.

Privacy Analysis. As the generator update is solely based on the discriminator (i.e., post-processing of DP outputs), it is sufficient to apply clipping and perturbation on training the discriminator [8]. Using Moments Accountant [1], we can bound the moments of a mechanism's privacy loss and prove the (ϵ, δ)-differential

privacy guarantee. In practice, we implement the Moments Accountant to estimate Algorithm 1's privacy loss ϵ_1 with the following: δ, the batch size, the total number of examples, the noise multiplier, and the number of training steps performed. Hence, *Algorithm 1 achieves (ϵ_1, δ)-differential privacy.* The intermediate models, i.e., G_i where $i < E$, incur less privacy loss due to small numbers of training steps required. We plot in Figure 2a an empirical analysis of the privacy estimates over the training epochs.

4.3 Improvements in DPNeT

While Algorithm 1 incorporates the state-of-the-art techniques (i.e., GANs) for generating synthetic data, the application of differential privacy may introduce new challenges, due to clipping the gradient norm and adding perturbation noise. As a result, the quality of the generated data may be affected by privacy. Below, we describe the proposed improvements in the DPNeT solution, with the goal of overcoming the limitations of differentially private deep learning.

Clipping Bound Decay. The clipping bound γ is an important factor in Algorithm 1. Although not influential on the privacy accountant, the act of clipping changes the gradient estimation. Specifically, when γ is too small, the average gradient may point to a different direction than the true gradient; on the other hand, increasing γ would require higher noise to be added, as the noise distribution is based on $\sigma\gamma$. Our idea is to decay the value of γ over the course of training, such that the model is able to learn quickly initially (i.e., with higher γ values) and accurately in the final stages (i.e., with lower γ values).

To that end, we propose three types of decay functions to control the speed of decay and investigate the impact on private learning. Our approach is based on commonly used decay functions, namely linear, exponential, and logarithmic:

$$\text{Linear decay}: \ \gamma(i) = C - i\left(\frac{C - C'}{E - 1}\right) \tag{3}$$

$$\text{Exponential decay}: \ \gamma(i) = C\ d^{(i-1)} \tag{4}$$

$$\text{Logarithmic decay}: \ \gamma(i) = \frac{C}{1 + \log(i)} \tag{5}$$

$$\text{No decay}: \ \gamma(i) = C \tag{6}$$

In the above functions, i indicates the current epoch where $i \in [1, E]$; d is the exponential decay rate; C is the initial value for the clipping bound and C' is the final value. The design ensures that all decay functions start with $\gamma(1) = C$, including the option of no decay. We choose the other parameters such that at the final epoch, the three decay functions arrive at similar γ values, i.e., around C'. In our empirical evaluation, we show for each decay option how γ values change throughout the training process (see Figure 4).

Private Model Selection. We also develop an effective strategy to privately select the best generative models among all models obtained throughout the training process. The rationale is that as DP introduces noise in training, the discriminator and the generator may converge towards a noisy equilibrium or do not converge. As a result, the generator obtained at the last epoch may not be optimal in quality. Therefore, in our approach, we consider models saved throughout the training epochs with the goal of choose the best among them. Furthermore, we need to ensure differential privacy in the model selection process as the private training data is utilized to evaluate the goodness of each model. Our approach is inspired by [2], where a classification accuracy score was utilized to select the best model. However the goal of DPNeT is to produce network flows that can be broadly used, including but not limited to classification applications. Thus, a generic quality measure for the generated data would be much more beneficial. We achieve this with the L1-distance between histograms of real data and synthetic data. We choose histograms in our algorithm for the low sensitivity.

Algorithm 2 Private Model Selection

Input: ground truth flows X_{real}, synthetic flows X_{gen}, number of features n, number of epochs E, number of models to select K, privacy budget ϵ_m for each selection

$\ell_{dist} = \{\}$

for $i = 1 \cdots E$ **do**

$X^i_{gen} \leftarrow N$ synthetic flows by model obtained at epoch i

Compute the following score based on L1 distance between histograms:

$\ell_i = \frac{1}{n} \sum_{j=1}^{n} ||\text{HIST}(\text{X}_{\text{real}}[\text{j}]) - \text{HIST}(\text{X}^{\text{i}}_{\text{gen}}[\text{j}])||_1$

$\ell_{dist}.insert(\ell_i)$

end for

for $t = 1 \cdots K$ **do**

Add noise every score: $\ell_{noisy} = \ell_{dist} + \text{Laplace}(0, \frac{1}{\epsilon_{\text{m}}})$

Sort ℓ_{noisy} and pick the epoch i_t with the smallest noisy score

$\ell_{dist}.remove(\ell_{i_t})$

end for

Output: K epochs $\{i_t | t = 1 \cdots K\}$ with the best models

As shown in Algorithm 2, we aim to choose K best models among all models obtained over E epochs, where the trained generator model can be saved at the end of each epoch. For each epoch i, we compute L1-based score to measure the similarity between the training data distribution and the synthetic data distribution, averaged across all features. Note that `HIST` in Algorithm 2 generates histograms for a given dataset (e.g., X_{real} or X^i_{gen}) and a certain feature (i.e., indexed by j). All scores are perturbed with noise drawn from a Laplace distribution with 0 mean and $\frac{1}{\epsilon_m}$ scale. The model that corresponds to the lowest noisy score is picked in each iteration until all K models have been selected. The selected models (which may not include the final epoch model) will be used to generate synthetic data. We further propose a *mixture* strategy to combine

synthetic flows from K models, which is demonstrated to be superior in our evaluation.

Privacy Analysis. Recall that intermediate generator models, i.e., G_i where $i \leq E$, are saved as part of Algorithm 1. Using G_i to generate synthetic data X^i_{gen} for Algorithm 2 does not incur additional privacy loss, thanks to the post-processing property of DP. The privacy result for Algorithm 2 is as follows:

1 the global sensitivity of ℓ_i is 1, for any i,
2 each iteration of model selection is ϵ_m-differentially private,
3 Algorithm 2 is $K\epsilon_m$-differentially private.

Suppose two neighboring datasets X_{real} and X'_{real}, where $X_{real} = X'_{real} \bigcup \{x^*\}$ and x^* is one flow record. The global sensitivity of ℓ_i can be analyzed with the Minkowski's inequality and the fact that by removing x^* the histograms of $X_{real}[j]$ for any attribute j can change by at most 1:

$$||\ell_i - \ell'_i||_1 \leq \frac{1}{n} \sum_{j=1}^{n} ||\mathrm{HIST}(\mathrm{X_{real}[j]}) - \mathrm{HIST}(\mathrm{X'_{real}[j]})||_1 \leq 1. \tag{7}$$

Given $\Delta\ell_i = 1$ $(\forall i)$, we can prove that each model selection with Laplace perturbation is ϵ_m-differentially private[1]. It follows that selecting K models with Algorithm 2 satisfies $K\epsilon_m$-differential privacy. Finally, we state the overall privacy guarantee of DPNeT.

Theorem 1. DPNeT satisfies (ϵ,δ)-differential privacy, where $\epsilon = \epsilon_1 + K\epsilon_m$ and $\delta = 1/N$.

Proof. Algorithm 1 satisfies (ϵ_1, δ)-DP where ϵ_1 can be estimated by Moments Accountant and $\delta = 1/N$. Algorithm 2 satisfies ($K\epsilon_m$, 0)-DP. By composition, the overall DPNeT solution satisfies ($\epsilon_1 + K\epsilon_m$, δ)-DP.

5 Evaluation Results

In this section, we present our methodology for empirical evaluation and discuss results obtained.

Dataset. We utilize a large-scale network flow dataset CIDDS-001 [24] in our evaluation. CIDDS-001 was captured within an emulated small business environment, which contains four weeks of flow-based network traffic. This dataset is publicly available[2], with around 32 *million* labelled network flows consisting of both anomalous and normal behaviors. For each flow, we adopt 11 relevant features, namely: *Date_first_seen, Duration, Protocol, Source_IP, Source_Port, Destination_IP, Destination_Port, Packets, Bytes, Flags, Class*, and preprocess

[1] Proof omitted for brevity; it is similar to the proof of Report Noisy Max [7].
[2] http://www.hs-coburg.de/cidds

them as suggested [23]. We then randomly subsample 2% of the entire dataset, which is then partitioned into two disjoint portions consisting of $N = 287435$ examples each. One partition is treated as *public* and utilized for training the embedding model, while the other partition remains *private* and used for training GAN models. Note that, since the two partitions are disjoint, some feature values in the GAN training data may not be present in the embedding model, e.g., unseen values. In those cases, we impute those feature values with the mode of their respective columns.

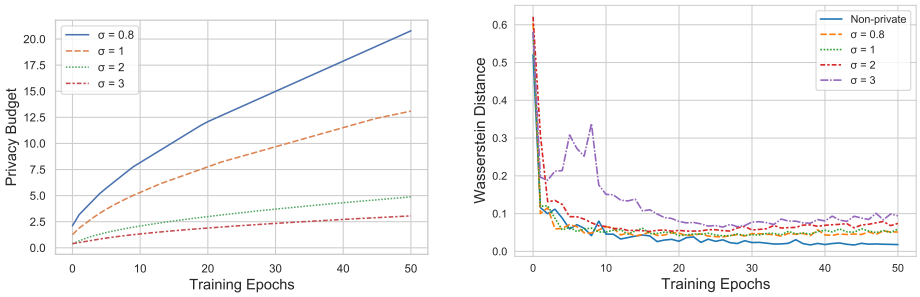
Metrics. For all our experiments, we utilize the following metrics to evaluate the quality of the synthetic data, in *similarity* and *realism*. To evaluate similarity, we measure the distance (e.g., Euclidean) between the probability distributions of the training data and the generated data for each attribute. To evaluate realism, we adopt 7 domain knowledge tests to check the consistency between attributes within each flow. For instance, one of them (Test 3) checks: "if the flow describes normal user behavior and the source port or destination port is 80 (HTTP) or 443 (HTTPS), the transport protocol must be TCP". The rationale as well as the detailed design of those domain knowledge tests can be found in [23]. For each test, we report the percentages of flows in the data that pass the test.

Hyper-parameters. The embedding model is trained for 500 epochs to accurately encode feature values. To achieve a trade-off between privacy and quality, the differentially private GAN models are trained for 50 epochs and we report the average results obtained in 3 separate runs. Other hyper-parameters for training the differentially private GAN models are: the training data size $N = 287435$, privacy parameter $\delta = \frac{1}{N}$, batch size $B = 2048$, initial clip norm $\gamma = 0.03$, and learning rate $\alpha = 0.0005$. For every generator iteration, discriminator is trained for $E_D = 5$ iterations. The privacy budget for training differentially private GANs is accounted by the Moment Accountant technique [1] as we vary the noise multiplier σ. For private model selection, we allocate $\epsilon_2 = 0.25$ for selecting top-5 models, i.e., spending 0.05 privacy budget for each round of Report Noisy Min. We compare our results with the non-private baseline.

5.1 Impact of Privacy

We study the effect of privacy on the quality of generated data. As we vary the noise multiplier σ in Algorithm 1, we apply the Moment Accountant to track the privacy budget ϵ_1 spent over the training epochs. We report the intermediate results in Figure 2a and ϵ_1 after 50 epochs in Table 1. Note that small ϵ values indicate stronger privacy protection. When setting $\sigma = 3$, we achieve $\epsilon_1 = 3.06$ for training the GAN model for 50 epochs; reducing the noise parameter degrades the privacy protection, e.g., $\epsilon_1 = 20.79$ when $\sigma = 0.8$. We also examine the impact of privacy on training quality. Figure 2b plots the loss in Wasserstein distance at the end of each epoch. Table 1 reports the final loss after 50 epochs, i.e., W_{dist}. We observe that in comparison to the non-private model, different privacy inflicts instability in the training process; stronger privacy leads to a higher level of instability, e.g., when $\sigma = 0.8$. Due to the added noise, a

differentially private model may converge to a noisy equilibrium. As privacy is relaxed, the performance of differentially private models gradually approaches that of the non-private model.



(a) Privacy budget spent on training GAN models.

(b) Wasserstein distance obtained during training.

Fig. 2: Impact of Noise (σ) on GAN Model Training (best viewed in color).

Table 1: Impact of Privacy and Domain Knowledge Test Accuracy (in %) on Real and Synthetic Data: ϵ_1 - privacy budget spent on training GAN models; ϵ - total privacy budget including training GAN models and model selection; W_{dist} - Wasserstein distance obtained at the last training epoch.

Privacy	ϵ_1/ϵ	W_{dist}	Domain Knowledge Test - Accuracy in %							
			Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	**Avg.**
Real data	∞/∞	0	100	100	100	100	100	100	100	**100**
Non-private	∞/∞	0.048	99.56	99.94	99.95	99.65	99.82	99.59	99.98	**99.64**
$\sigma = 0.8$	20.79/21.04	0.061	96.79	99.19	99.63	94.53	99.91	77.33	91.85	**94.18**
$\sigma = 1$	13.1/13.35	0.064	96.92	98.79	99.47	94.17	99.97	65.37	90.42	**92.16**
$\sigma = 2$	4.88/5.13	0.084	91.85	98.06	99.47	90.12	100.0	65.92	84.72	**90.02**
$\sigma = 3$	3.06/3.31	0.124	94.7	98.37	99.51	91.15	100.0	3.37	89.51	**82.37**

We conduct the domain knowledge tests on real/synthetic data and Table 1 reports the accuracy results, i.e., the percentage of flows that pass each test. We observe that both private and non-private GAN models are able to produce realistic network flows, e.g., highly accurate for Test 3 and Test 5. Test 6 appears to be the most challenging to capture by the private models. The content of Test 6 is: "if the flow represents a netbios message (destination port is 137 or 138), the source IP addresses must be internal (192.168.XXX.XXX) and the destination IP address must be an internal broadcast (192.168.XXX.255)." As it examines a specific type of flows and involves multiple features, even introducing a small

Table 2: Impact of Privacy and Data Quality: Euclidean distance between synthetic data distribution and training data distribution is reported.

Feature	Non-private	$\sigma = 0.8$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$
Day	0.009	0.032	0.043	0.047	0.082
Time	0.008	0.007	0.006	0.006	0.008
Duration	0.74	0.73	0.725	0.715	0.713
Protocol	0.004	0.003	0.002	0.007	0.012
Src IP	0.067	0.131	0.113	0.099	0.104
Src Pt	0.01	0.019	0.016	0.037	0.039
Dst IP	0.063	0.118	0.118	0.098	0.09
Dst Pt	0.01	0.017	0.016	0.024	0.036
Packets	0.016	0.044	0.055	0.068	0.074
Bytes	0.018	0.047	0.055	0.072	0.083
Flags	0.006	0.074	0.098	0.103	0.139
Class	0.003	0.004	0.006	0.005	0.011
Avg.	**0.079**	**0.103**	**0.104**	**0.107**	**0.116**

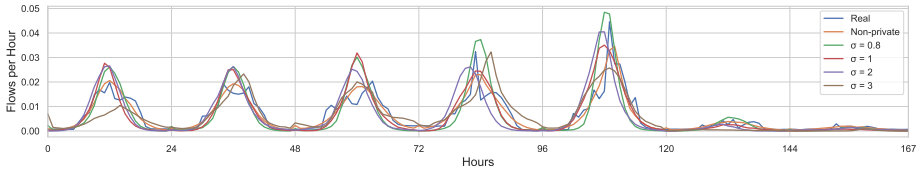
amount of noise ($\sigma = 0.8$) inflicts a drop in performance. We also report the average accuracy among all 7 tests. It can be seen that the accuracy degrades gradually from real data, to synthetic data generated by the non-private model, to synthetic data generated by differentially private models.

We report in Table 2 the similarity between real and synthetic data via the Euclidean distance computed for the probability distributions of each feature. The average similarity among all features is reported in the last row of the table. As can be seen, synthetic data generated by the non-private model is most similar to the real data; stronger privacy (i.e., higher σ values) would inflict a higher distance from the real distributions.

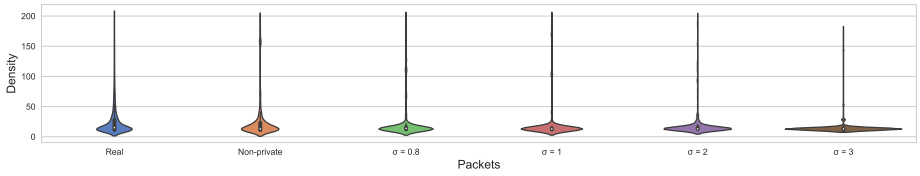
Qualitative Evaluation. To better understand the quality of the synthetic data, we plot the distributions of real and synthetic network flows in Figure 3. Figure 3a shows the percentage of flows per hour in each dataset. We can observe that the non-private model can accurately capture variations in the real data distribution, and the private models are able to capture high-level trends while missing local variations. Similarly we observe in Figure 3b that the distribution of Packet feature is well preserved by the non-private model. As the noise increases, higher distortions in the probability distribution can be observed.

5.2 Impact of Clipping Bound Decay

Here we study the effect of decaying the gradient norm bound on the quality of the generated data. We set $\sigma = 1$ in the following experiments and train GAN models with different decay functions proposed in Section 4.3. The resulting clipping bounds are depicted in Figure 4. It can be seen that the clipping bounds reduce much more quickly when exponential decay and logarithmic decay are adopted, although all three decay functions reach similar values at the



(a) Distribution of flows over time.



(b) Distribution of "Packet" feature.

Fig. 3: Distributions of Real and Synthetic Flows (best viewed in color).

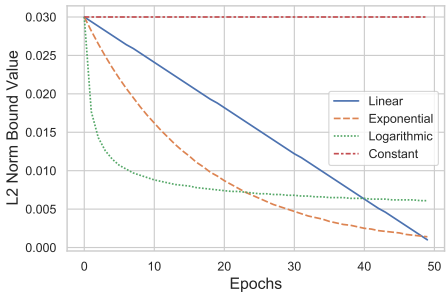


Fig. 4: Clipping Bound Decay: we set $C = 0.03$, $C' = 0.001$, and $d = 0.94$ such that all clipping bounds initialize at C and decay accordingly over 50 epochs, with the exception of Constant.

50th epoch. In contrast, the no decay option adopts a constant clipping bound throughout all training epochs.

Table 3 and Table 4 report the distributional similarity and the accuracy of domain knowledge tests, respectively, using synthetically generated data. As expected, having a constant clipping bound (i.e., no decay), would force more noise to be added and thus reduce the quality of the learned model, i.e., higher Euclidean distance and lower domain knowledge test accuracy. It can be seen that linear decay is competitive in distributional similarity, and performs the best in domain knowledge tests. We believe that it is beneficial to gradually reduce the clipping bound over training epochs to achieve a trade-off between learning and privacy. Thus, we propose to adopt linear decay in the DPNeT solution.

Table 3: Impact of Decay Functions and Data Quality: Euclidean distance between synthetic data distribution and training data distribution is reported.

Feature	No Decay	Exponential	Logarithmic	Linear
Duration	0.742	**0.717**	0.731	0.725
Protocol	0.003	0.004	0.01	**0.002**
Src IP	0.116	**0.103**	0.118	0.113
Src Pt	**0.016**	0.019	0.022	**0.016**
Dst IP	0.121	0.120	0.127	**0.118**
Dst Pt	0.02	**0.016**	**0.016**	**0.016**
Packets	0.056	**0.052**	0.056	0.055
Bytes	0.06	0.057	0.056	**0.055**
Flags	0.113	**0.095**	0.097	0.098
Class	0.007	0.007	**0.005**	0.006
Avg.	0.125	**0.119**	0.124	0.120

Table 4: Impact of Decay Functions and Domain Knowledge Test Accuracy (in %).

Domain Test	No Decay	Exponential	Logarithmic	Linear
Test 1	96.73	96.39	**97.09**	96.92
Test 2	98.72	**99.07**	98.97	98.79
Test 3	99.53	**99.59**	99.36	99.47
Test 4	92.55	93.01	**94.17**	**94.17**
Test 5	**99.98**	99.71	99.95	99.97
Test 6	45.05	50.03	53.56	**65.37**
Test 7	89.4	91.29	**91.36**	90.42
Avg.	88.85	89.87	90.64	**92.16**

5.3 Private Model Selection

Here we examine the effect of model selection using Algorithm 2. For each run of differentially private GAN training, we choose the best $K = 5$ epoch models and allocate $\epsilon_m = 0.05$ privacy budget for each selected model. We name the selected models Nosiy Best, Noisy 2nd, etc. $N = 287435$ synthetic flows are generated from each model; in addition, we create a *mixture* of size N by randomly subsampling from each model. The model obtained at the 50th epoch is also included as a baseline, since it may not be selected by the algorithm.

Table 5 and Table 6 report the distributional similarity and the accuracy of domain knowledge tests, respectively. Since the algorithm utilizes an L1-based score, we report the L1 distance between the probability distributions of real data and synthetic data in Table 5. We observe that our private model selection approach is highly beneficial: the average L1 distance monotonically increases from Noisy Best to Noisy 5th. The last epoch shows a much higher average L1 distance to the real data distributions, confirming that the final model may not be optimal. We also observe that the private model selection is affected by perturbation, i.e., Noisy 5th shows the best quality in Flags and Class. A

very important observation is that the mixture strategy is superior to all other models, exhibiting the highest similarity to real data in many features.

Table 5: Impact of Model Selection on Data Quality: L1 distance between synthetic data distribution and training data distribution is reported.

Feature	Noisy Best	Noisy 2nd	Noisy 3rd	Noisy 4th	Noisy 5th	Last Epoch	Mixture
Duration	1.484	1.532	1.511	1.519	1.513	1.608	**1.408**
Protocol	0.008	0.008	0.016	0.012	0.016	0.015	**0.003**
Src IP	0.867	0.847	0.854	0.849	0.882	1.031	**0.708**
Src Pt	0.830	0.857	0.920	0.923	0.895	1.014	**0.723**
Dst IP	0.872	0.853	0.905	0.874	0.895	1.068	**0.708**
Dst Pt	0.849	0.862	0.912	0.880	0.924	1.008	**0.717**
Packets	0.212	0.276	0.251	0.257	0.232	0.277	**0.202**
Bytes	0.713	0.759	0.731	0.768	0.786	0.922	**0.636**
Flags	0.267	0.251	0.219	0.225	**0.215**	0.219	0.219
Class	0.025	0.011	0.009	0.025	**0.007**	0.024	0.009
Avg.	0.613	0.626	0.633	0.633	0.637	0.718	**0.533**

Table 6: Domain Knowledge Test Accuracy (in %) for Models Selected with Algorithm 2.

Domain Test	Noisy Best	Noisy 2nd	Noisy 3rd	Noisy 4th	Noisy 5th	Last Epoch	Mixture
Test 1	97.68	94.20	**98.34**	96.57	97.46	97.46	96.92
Test 2	98.86	98.87	**99.23**	98.25	98.76	98.28	98.79
Test 3	99.37	99.41	**99.77**	99.28	99.52	99.39	99.47
Test 4	95.08	94.10	93.31	91.86	**96.06**	94.02	94.17
Test 5	**100.00**	100.00	100.00	100.00	99.55	100.00	99.97
Test 6	45.26	**66.79**	61.78	60.52	18.32	35.03	65.37
Test 7	90.02	90.56	**90.67**	90.66	90.18	90.55	90.42
Avg.	89.47	91.99	91.87	91.02	85.69	87.82	**92.16**

Similarly in Table 6, Noisy 5th and Last Epoch show lower average test accuracy, compared to other models. The mixture shows the highest average test accuracy among all models. Both tables demonstrate the advantage of our model selection approach and creating a diverse set of synthetic data using *mixture*.

6 Conclusion and Future Work

In this paper, we have described DPNeT, a differentially private solution for generating high quality synthetic network flow data. Privacy of the sensitive training data is protected by training GAN models with differential privacy. We have also proposed novel approaches for clipping bound decay and private model selection.

We have demonstrated their effectiveness in improving the quality of synthetic data through comprehensive empirical evaluations. Our approaches may be applied to other differentially private deep learning tasks, e.g. classification.

We identify several directions for future work. Firstly, we observe that fine tuning of hyper parameters, such as learning rate and number of discriminator epochs, is essential to circumvent issues such as mode collapse and non-convergence. In the future, effective parameter tuning methods can be explored for differentially private solutions. Secondly, it is desirable to investigate the usefulness of the synthetic network flow data in domain specific applications, e.g., anomaly detection. Future work can study the performance of anomaly detection models trained with synthetic data. Thirdly, as our solution builds on an embedding model trained with public data, it is important to study the efficacy of the solution when public data may come from a different distribution.

Acknowledgements

The authors would like to thank the anonymous reviewers for their suggestions and comments. This work has been supported in part by NSF CNS-1949217, NSF CNS-1951430, and UNC Charlotte. The opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 308–318 (2016)
2. Beaulieu-Jones, B.K., Wu, Z.S., Williams, C., Lee, R., Bhavnani, S.P., Byrd, J.B., Greene, C.S.: Privacy-preserving generative deep neural networks support clinical data sharing. Circulation: Cardiovascular Quality and Outcomes **12**(7), e005122 (2019)
3. Brekne, T., Årnes, A., Øslebø, A.: Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In: International Workshop on Privacy Enhancing Technologies. pp. 179–196. Springer (2005)
4. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: Smoteboost: Improving prediction of the minority class in boosting. In: European conference on principles of data mining and knowledge discovery. pp. 107–119. Springer (2003)
5. Chen, Y., Trappe, W., Martin, R.P.: Detecting and localizing wireless spoofing attacks. In: 2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. pp. 193–202. IEEE (2007)
6. Coull, S.E., Wright, C.V., Monrose, F., Collins, M.P., Reiter, M.K., et al.: Playing devil's advocate: Inferring sensitive information from anonymized network traces. In: Ndss. vol. 7, pp. 35–47 (2007)
7. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science **9**(3-4), 211–407 (2014)

8. Fan, L.: A survey of differentially private generative adversarial networks. In: The AAAI Workshop on Privacy-Preserving Artificial Intelligence (2020)
9. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 1322–1333. ACM (2015)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
11. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in neural information processing systems. pp. 5767–5777 (2017)
12. Hayes, J., Melis, L., Danezis, G., De Cristofaro, E.: Logan: Membership inference attacks against generative models. Proceedings on Privacy Enhancing Technologies **2019**(1), 133–152 (2019)
13. King, J., Lakkaraju, K., Slagell, A.: A taxonomy and adversarial model for attacks against network log anonymization. In: Proceedings of the 2009 ACM symposium on Applied Computing. pp. 1286–1293 (2009)
14. Li, J., Zhou, L., Li, H., Yan, L., Zhu, H.: Dynamic traffic feature camouflaging via generative adversarial networks. In: 2019 IEEE Conference on Communications and Network Security (CNS). pp. 268–276. IEEE (2019)
15. Li, Y., Slagell, A., Luo, K., Yurcik, W.: Canine: A combined conversion and anonymization tool for processing netflows for security. In: International conference on telecommunication systems modeling and analysis. vol. 21 (2005)
16. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., et al.: Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In: Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00. vol. 2, pp. 12–26. IEEE (2000)
17. McSherry, F., Mahajan, R.: Differentially-private network trace analysis. ACM SIGCOMM Computer Communication Review **40**(4), 123–134 (2010)
18. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
19. Mohammady, M., Wang, L., Hong, Y., Louafi, H., Pourzandi, M., Debbabi, M.: Preserving both privacy and utility in network trace anonymization. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 459–474 (2018)
20. Pang, R., Allman, M., Paxson, V., Lee, J.: The devil and packet trace anonymization. ACM SIGCOMM Computer Communication Review **36**(1), 29–38 (2006)
21. Riboni, D., Villani, A., Vitali, D., Bettini, C., Mancini, L.V.: Obfuscation of sensitive data in network flows. In: 2012 Proceedings IEEE INFOCOM. pp. 2372–2380. IEEE (2012)
22. Ring, M., Dallmann, A., Landes, D., Hotho, A.: Ip2vec: Learning similarities between ip addresses. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 657–666. IEEE (2017)
23. Ring, M., Schlör, D., Landes, D., Hotho, A.: Flow-based network traffic generation using generative adversarial networks. Computers & Security **82**, 156–172 (2019)
24. Ring, M., Wunderlich, S., Grüdl, D., Landes, D., Hotho, A.: Flow-based benchmark data sets for intrusion detection. In: Proceedings of the 16th European conference on cyber warfare and security. pp. 361–369 (2017)

25. Shahid, M.R., Blanc, G., Jmila, H., Zhang, Z., Debar, H.: Generative deep learning for internet of things network traffic generation. In: 2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC). pp. 70–79 (2020). https://doi.org/10.1109/PRDC50213.2020.00018
26. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 3–18. IEEE (2017)
27. Son, S., Shmatikov, V.: The hitchhiker's guide to dns cache poisoning. In: International Conference on Security and Privacy in Communication Systems. pp. 466–483. Springer (2010)
28. Torkzadehmahani, R., Kairouz, P., Paten, B.: Dp-cgan: Differentially private synthetic data and label generation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2019)
29. Wright, C., Monrose, F., Masson, G.M.: Hmm profiles for network traffic classification. In: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security. pp. 9–15 (2004)
30. Xu, J., Fan, J., Ammar, M., Moon, S.B.: On the design and performance of prefix-preserving ip traffic trace anonymization. In: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement. pp. 263–266 (2001)