NetDiffusion: Network Data Augmentation Through Protocol-Constrained Traffic Generation

XI JIANG, University of Chicago, USA
SHINAN LIU, University of Chicago, USA
AARON GEMBER-JACOBSON, Colgate University, USA
ARJUN NITIN BHAGOJI, University of Chicago, USA
PAUL SCHMITT, University of Hawaii, Manoa / Invisv, USA
FRANCESCO BRONZINO, Univ Lyon, EnsL, UCBL, CNRS, LIP, France
NICK FEAMSTER, University of Chicago, USA

Datasets of labeled network traces are essential for a multitude of machine learning (ML) tasks in networking, yet their availability is hindered by privacy and maintenance concerns, such as data staleness. To overcome this limitation, synthetic network traces can often augment existing datasets. Unfortunately, current synthetic trace generation methods, which typically produce only aggregated flow statistics or a few selected packet attributes, do not always suffice, especially when model training relies on having features that are only available from packet traces. This shortfall manifests in both insufficient statistical resemblance to real traces and suboptimal performance on ML tasks when employed for data augmentation. In this paper, we apply diffusion models to generate high-resolution synthetic network traffic traces. We present *NetDiffusion*[1], a tool that uses a finely-tuned, controlled variant of a Stable Diffusion model to generate synthetic network traffic that is high fidelity and conforms to protocol specifications. Our evaluation demonstrates that packet captures generated from NetDiffusion can achieve higher statistical similarity to real data and improved ML model performance than current state-of-the-art approaches (e.g., GAN-based approaches). Furthermore, our synthetic traces are compatible with common network analysis tools and support a myriad of network tasks, suggesting that NetDiffusion can serve a broader spectrum of network analysis and testing tasks, extending beyond ML-centric applications.

CCS Concepts: • **Networks** → **Network simulations**; • **Computing methodologies** → **Neural networks**.

Additional Key Words and Phrases: Network traffic, synthesis, diffusion model

ACM Reference Format:
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 (March 2024), 32 pages. https://doi.org/10.1145/3639037

[1]We open source our sample datasets, pipeline, and results in https://github.com/noise-lab/NetDiffusion_Generator.

Authors' addresses: Xi Jiang, xijiang9@uchicago.edu, University of Chicago, Chicago, Illinois, USA; Shinan Liu, shinanliu@uchicago.edu, University of Chicago, Chicago, Illinois, USA; Aaron Gember-Jacobson, agemberjacobson@colgate.edu, Colgate University, Hamilton, New York, USA; Arjun Nitin Bhagoji, abhagoji@uchicago.edu, University of Chicago, Chicago, Illinois, USA; Paul Schmitt, pschmitt@hawaii.edu, University of Hawaii, Manoa / Invisv, Honolulu, Hawaii, USA; Francesco Bronzino, francesco.bronzino@ens-lyon.fr, Univ Lyon, EnsL, UCBL, CNRS, LIP, Lyon, France; Nick Feamster, feamster@uchicago.edu, University of Chicago, Chicago, Illinois, USA.

ACM 2476-1249/2024/3-ART11
https://doi.org/10.1145/3639037

1 INTRODUCTION

Modern networks are increasingly reliant on machine learning (ML) techniques for a wide range of management tasks, ranging from security to performance optimization. A central impediment when training network-focused ML models is the scarcity of labeled network datasets, as their collection and sharing are often associated with high costs and privacy concerns, particularly when data is collected from real-world networks [9, 42, 87, 88, 121]. Unfortunately, existing public datasets rarely receive updates, making them static and unable to reflect evolving network behaviors [68, 74, 105]. These limitations hinder the ability to train robust ML models that accurately reflect evolving real-world network conditions.

These challenges can be addressed through the creation of new synthetic network traces based on existing datasets. This approach aims to preserve the inherent characteristics of network traffic while introducing variations, thereby enhancing dataset size and diversity [78, 96, 115, 122, 135, 140, 142, 143]. Unfortunately, current state-of-the-art synthetic trace generation methods, particularly those based on Generative Adversarial Networks (GANs)-based methods [78, 104, 139, 140, 142], are not always sufficient for producing high-quality synthetic network traffic. Specifically, these approaches tend to focus on a limited set of attributes or statistics, as early machine learning for network tasks often relied on basic flow statistics for classification [17, 33, 44, 65, 76, 77, 100]. With recent ML advancements utilizing detailed raw network traffic to achieve enhanced classification accuracy [10, 23, 39, 83, 84, 103, 114, 128, 136, 141, 149], there is a clear need for synthetic traffic generation that includes the intricate, potentially unforeseen patterns present in full network traces. Existing traffic generation methods face two main issues: (1) a lack of statistical similarity with real data due to the limited attributes in existing methods, making the synthetic data highly sensitive to variations, and (2) unsatisfactory classification accuracy when synthetic statistical attributes are used to augment existing datasets. Moreover, their simplistic attribute focus and disregard for transport and network layer protocol behaviors prevent their use with traditional networking tools such as tcpreplay [43] or Wireshark [16].

Fortunately, the general increase in available computational power and the breakthroughs in high-resolution image generation techniques, particularly diffusion models [101, 107, 120], offer a promising avenue to overcome these challenges. Specifically, we harness the capabilities of text-to-image diffusion models, which execute conditioned generation based on descriptive text prompts. These models are adept at creating detailed, accurate visual representations from textual descriptions. By translating the intricate characteristics of network traffic into an appropriate image format, we can tap into the unique advantages offered by these models. In contrast to GANs, diffusion models are able to capture both broad patterns and detailed dependencies. This inherent generative quality makes them an ideal choice for producing network traces with high statistical resemblance to real traffic and full packet header values. By incorporating conditioning techniques, diffusion models can generate structured data that conforms to specific network properties, which ensures the desired sequential inter-packet characteristics and rough protocol dependencies. Moreover, the gradient dynamics of the training process in diffusion models is much more stable than GANs. We discuss the technical details and benefits of diffusion models in depth in Section 3. These attributes collectively position diffusion models as a compelling choice for advancing the state-of-the-art for synthetic network trace generation, addressing the extant limitations of current methodologies.

In this paper, we introduce NetDiffusion, an approach to synthetic raw network traffic generation for producing packet headers leveraging fine-tuned, controlled stable diffusion models. Our contributions are as follows:

(1) **Generation of synthetic network traces with high resemblance to real traffic:** Using stable-diffusion techniques, we propose a two-fold strategy: (1) a conversion process

for transforming raw packet captures to image representations (and vice versa), and (2) fine-tuning a text-to-image diffusion model based on packet capture-converted images for generating synthetic packet captures. To improve resemblance to real network traffic, we employ controlled generation techniques to maintain fidelity to the protocol and header field value distributions observed in real data and, post generation, use domain knowledge-based heuristics to finely check and adjust the generated fields, ensuring their semantic correctness in terms of compliance with transport and network layer protocol rules.

(2) **Improved classification accuracy in ML scenarios with synthetic network traffic augmentation:** We conduct a case study evaluation on a curated traffic classification dataset. By integrating NetDiffusion-generated network traffic into the real dataset at varying proportions during training and testing, we observe a general increase in accuracy compared to the state-of-the-art generation method [142]. This improvement is attributed to our synthetic data's significantly high statistical resemblance to the real dataset. Additionally, our method shows promise in addressing class imbalance issues, enhancing the accuracy of ML models in such cases.

(3) **Extended applicability of synthetic network traffic for network analysis and testing beyond ML tasks:** NetDiffusion-generated network traffic can be converted into raw packet captures suitable for traditional network analysis and testing tasks. We validate this compatibility through tests with tools such as Wireshark and Scapy [106], as well as tcpreplay for retransmission. More importantly, we show that critical statistical features for various network operations can be effectively extracted from the generated network traffic.

2 MOTIVATION

The use of publicly available network datasets has significantly aided advancements in applying ML to networks, as well as network analysis and testing methodologies. For example, models trained on network datasets have been helpful in tackling challenges like anomaly detection, traffic classification, and network optimization, which in turn enhances network security and performance [9, 21, 31, 61–63, 67, 81, 83, 92, 131]. Additionally, these datasets are valuable for network analysts, aiding in understanding network behaviors, identifying performance issues, and evaluating the performance of network security tools like firewalls and intrusion detection systems [94, 102, 134].

2.1 Network Data Scarcity

Well-known network datasets such as CAIDA [7], MAWI [30], UNSW-NB15 [93], and KDD [6, 130] have been essential for numerous research projects in network science. However, the lack of updated datasets often hinders further progress. Those with the means to capture large-scale traffic, typically network operators and organizations with specialized hardware and network, are often hesitant to share their data due to the risk of exposing sensitive or personally identifiable information (PII). Even when entities are amenable to sharing, the task of providing consistent updates and ensuring reliable labels for sanitized data is daunting. Labeling network data is inherently challenging because of its dynamic nature, such the continuous evolution of network behaviors and threats. Notably, the CAIDA, UNSW-NB15, KDD Cup 99 [6], and NSL-KDD [130] datasets were last updated in 2020, 2015, 1999, and 2009 respectively, revealing notable gaps in data recency which render them less reflective of evolving network dynamics. Even frequently updated datasets like MAWI [30] are not exempt from issues, with instances of missing data from hardware failures and substantial duplicate traces. While not an exhaustive list of datasets, the issues highlighted are common across the board, accentuating the need for newer data to fuel ongoing network research and analysis.

2.2 Data Augmentation Using Synthetic Data

Data augmentation through synthetic data has proven effective in many fields. In computer vision, for instance, synthetic images have improved model performance, especially when there's a shortage of labeled data [29, 82, 90, 118, 129]. The success of synthetic data augmentation is largely attributed to generative methods which have showcased remarkable versatility in a variety of domains: In medical imaging, GANs have been harnessed to augment datasets, significantly enhancing the performance of diagnostic models [27, 48, 50]. In the domain of natural language processing, Variational Autoencoders (VAEs) have been utilized to create synthetic text data, aiding in tasks such as sentiment analysis and language translation [32, 132, 137]. In audio processing, the advent of WaveGAN has facilitated the expansion of sound datasets, proving indispensable for applications like speech recognition and sound event detection [13, 45, 46].

Translating these successes to the networking domain, certain endeavors have emerged, attempting to augment network datasets through the generation of synthetic network data [78, 104, 139, 140, 142]. The overarching goal of these generative methods is to produce synthetic traffic that exhibits high levels of statistical similarity to real-world network traffic. Distinguishing them from simulation-based approaches, these generative techniques introduce subtle variations in the synthetic data, diverging slightly from the real data. This aspect is crucial; it simulates potential, unseen variations and the dynamic nature of real-world network environments. By doing so, while the synthetic traffic largely mirrors the general patterns of the real data, it also aids in enhancing the generalizability of various applications, such as anomaly detection systems and ML models. For instance, in anomaly detection, these nuanced variations in synthetic traffic allow models to adapt better to unpredictable or novel network behaviors, such as small variations in the IP addresses or TCP flags, thereby improving their effectiveness and robustness in real-world scenarios. A notable state-of-the-art attempt in this regard is NetShare [142] which utilizes GANs to produce IPFIX [35]/NetFlow [34]-style statistics on network traffic. For simplicity, we refer to this general type of aggregated statistucal attributes as NetFlow for the rest of the paper. Yet, its focus on statistical properties might miss important network patterns essential for high ML accuracy. At the same time, the limited number of attributes that it focuses prohibits the generation of comprehensive, raw network traffic such as packet captures, which are essential for additional non-ML tasks such as network analysis and testing. In this paper, we do not consider other non-generative methods [20, 24, 53, 75] like TRex [8] and NS-3 [53] because, while useful for specific tasks, they lack the flexibility needed for broader dataset augmentation. They often rely on predefined templates or rules, which may not capture the evolving nature of network traffic or the complex interactions between various network protocols and applications.

2.3 Inadequate Performance from Existing Methods

We provide a short case study on NetShare, which is the current state-of-the-art network data generation method that produces NetFlow attributes, *i.e.*, derived statistics from raw network traffic flows. Following the method in the original paper, we test the accuracy of a variety of ML models–Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM)–under three scenarios: (1) training and testing on real NetFlow data, (2) training on NetShare synthetic data and testing on real data, and (3) vice versa. In this paper, we focus on a traffic classification task using a curated dataset, detailed in Section 4. The classification task is divided into *micro* and *macro* levels. On the micro level, the goal is to classify network traffic flows into specific applications, encompassing 10 distinct classes such as YouTube and Amazon. On the macro level, the aim is to classify network traffic flows into broader service categories, spanning 3 classes, like streaming and web browsing.

2.3.1 Unsatisafactory ML Accuracy. (1) NetShare Limitations: The results from Table 1 showcases a clear drop in accuracy when models are trained or tested on synthetic NetFlow data generated by NetShare compared to when only real NetFlow data is used, irrespective of the classification level. This points to a likely shortfall of NetShare in preserving critical distinguishing feature values inherent in the real dataset, which adversely affects the models' ability to accurately classify network traffic. (2) NetFlow vs. Raw Traffic: Following the NetShare evaluation, we compared the accuracy achieved with real NetFlow data to that when models train and test on raw network traffic in pcap format. This comparison underscores the information loss encountered when using NetFlow data and the potential classification performance gain from leveraging raw network traffic: The highest accuracy is achieved with raw network traffic, with SVM arriving at near-perfect accuracy. Conversely, a noticeable decrease in accuracy is observed with real NetFlow data, suggesting that its limited feature set adversely affects classification accuracy.

These observations lead to two main insights: First, the need for synthetic data generation methods to effectively preserve critical distinguishing feature values to maintain classification accuracy; Second, the advantage of using raw network traffic over NetFlow data due to its richer information content. The push towards generating raw network traffic to retain the fine-grained details and statistical properties of real network traffic, appears to be a crucial step to overcome the limitations observed with NetShare generated data and NetFlow data.

Training/Testing	Data Format (Generation Method)	Highest Accuracy (Model)	
		Macro-level	Micro-level
Real/Real	Network traffic (pcap) (N/A)	1.00 (SVM)	0.978 (SVM)
	NetFlow (N/A)	0.86 (RF)	0.648 (DT)
Synthetic/Real	NetFlow (NetShare)	0.396 (RF)	0.140 (SVM)
Real/Synthetic	NetFlow (NetShare)	0.503(SVM)	0.102 (RF)

Table 1. Comparison of model accuracy using real, synthetic NetFlow data, and raw network traffic. Results highlight a decrease in accuracy with NetShare's synthetic data and a boost when using raw traffic. Only the top-performing model is displayed.

2.3.2 Limited Applicability to Non-ML tasks. Besides the performance of synthetic network data in ML tasks, another important metric to validate its usefulness is its applicability to traditional network analysis and testing tasks, such as packet-wise analysis and replay. This is important because, unlike other forms of data such as images where the quality of the data can be inferred relatively easily by visual resemblance to real images, it is hard for network experts to manually examine raw network traffic to verify its quality. NetFlow data, encapsulating aggregated or derived statistics from raw network traffic flows, lacks the detailed information crucial for these tasks. For instance, network analysts often use tools like Wireshark to investigate network traffic details like packet headers and sequences to diagnose issues or assess performance, such as tracing latency causes or detecting unauthorized access. However, the high-level statistical nature of NetFlow data omits such fine-grained details, rendering it inadequate for such in-depth analysis. Furthermore, synthetic NetFlow data cannot be retransmitted or replayed over network interfaces using tools like tcpreplay. Retransmitting network traffic is pivotal for various network testing and validation scenarios, such as evaluating the performance of network security tools under realistic traffic conditions or stress-testing network infrastructure. The absence of packet-level details in NetFlow data precludes its use for retransmission tasks. Moreover, certain network analysis tasks require deriving additional attributes directly from raw network traffic. For example, estimating the window size or investigating the distribution of packet sizes across a network necessitates access to raw traffic data. These computations are crucial for understanding network behavior and optimizing network configurations.

Converting high-level statistical NetFlow data back to raw network traffic, such as packet captures, is inherently challenging due to the loss of detailed attributes like header values and flags, thereby limiting the utility of synthetic NetFlow data for a myriad of non-ML network tasks. This challenge

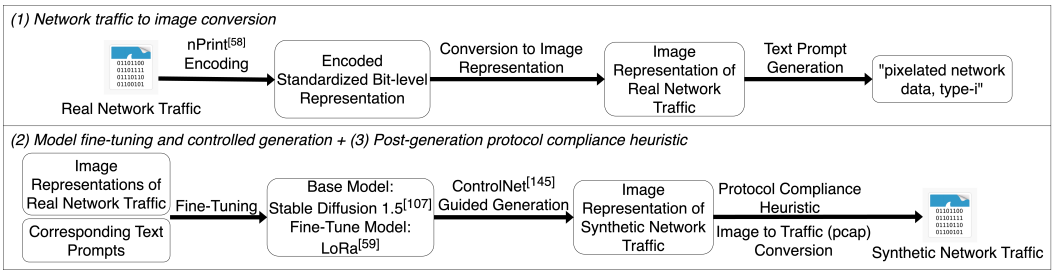


Fig. 1. Generation Framework Overview.

underscores the necessity of generating raw network traffic. At the same time, existing network data generation methods often neglect to ensure that synthetic data adhere to critical transport and network layer protocol rules found in real network traffic, which are crucial for traditional network analysis and testing tools. For example, a protocol constraint is that packet length frames must conform to specific sizes as per protocol standards. Generating synthetic data with incorrect frame sizes could lead to misinterpretations in data analysis or malfunctions in network testing scenarios. Other protocol constraints, like correct sequence numbers in TCP transmissions, valid checksums, and appropriate flag settings, are also crucial as they affect how network devices and analysis tools interpret and process the data. Hence, adhering to protocol rules is essential for the accuracy and reliability of network analysis and testing tasks, and serves as a gauge for the quality of synthetic network data generated.

3 METHOD

In this section, we introduce *NetDiffusion*, a framework that harnesses controlled text-to-image diffusion models [107] to generate synthetic raw network traffic that complies with transport and network layer protocol rules. We find this approach not only elevates classification accuracy when utilized for data augmentation in ML scenarios but also facilitates a broad range of network analysis and testing tasks (see Section 4), overcoming the limitations described in Section 2.3. We first provide an overview of our method which has the 3 components shown in Figure 1, before providing details of each component.

How do diffusion models work? Diffusion models synthesize data by modeling data generation as the process of noise removal from noisy data (referred to as the reverse process) [55, 125]. The noise removal is performed by a complex ML model, usually a neural network, that has been trained to predict the noise that was sequentially added to real data (the forward process). Running the forward and reverse processes in the latent space of a model has been found to generate better quality data [107]. Mathematically, consider an initial noise vector ($\mathbf{z}$) in the latent space. The goal of diffusion models is to transform $\mathbf{z}$ into an data point ($\mathbf{x}$) drawn from the desired distribution. The idea is to define a differential equation that controls the transformation from $\mathbf{z}$ to $\mathbf{x}$ over a series of discrete time steps. The rationale behind this is that by breaking down the generation process into a series of incremental diffusion steps, the model can capture intricate dependencies and details in the data manifold. An essential component of this approach is score-based generative modeling, where the gradient of the data log-likelihood with respect to the data (often termed the "score" function) is estimated. Modeling the score function is preferable because directly modeling the probability distribution poses challenges, especially in obtaining the correct normalization constant [124, 125]. Diffusion models have been adopted most effectively in the context of text-to-image synthesis, where images matching a given text prompt are to be generated. The text prompt serves as a conditioning variable to guide the reverse process towards generating an image that semantically aligns with the text prompt. By iteratively applying the score function, conditioned

on the text prompt, the model steers the data from a simple prior distribution (like Gaussian noise) to the desired complex image distribution. In our case, the text prompts characterize the *specific classes/types of network traffic* that we aim to generate.

We extend these principles on the operation of diffusion models to network traffic data via our NetDiffusion framework. Our approach is structured around three primary components:

(1) Converting raw network traffic in packet capture (pcap) format into image-based representations (§3.1).
(2) Fine-tuning a Stable Diffusion model to enable controlled text-to-traffic generation with high distributional similarity across header fields to real-world network traffic (§3.2).
(3) Domain knowledge-based post-processing heuristics for detailed modification of generated network traffic to ensure high level of protocol rule compliance (§3.3).

3.1 Network Traffic to Image Conversion

In this subsection, we explain the motivation and the process of representing network traffic as images, an important step in our method.

3.1.1 Advantages of Using Image Representation of Network Traffic. Network traffic data, with intricate inter-packet dependencies and vast range of attributes, presents a complex landscape that introduces specific challenges when it comes to accurate representation and efficient learning. Network traffic data exhibits *high dimensionality*, particularly when using standardized representations such as nPrint [58]. For instance, between the IP and TCP headers alone, there is an abundance of fields (*e.g.*, IP addresses, ports, sequence and acknowledgment numbers, flags, etc.). nPrint uses a bit-level and standardized representation to have a consistent format for each packet by accounting for all potential header fields (even if not present in the original packet). For instance, while a TCP packet won't have UDP header bits, the nPrint still includes placeholders for these bits. While this ensures a uniform input structure for ML models, the attribute count per packet often exceeds a thousand. This high dimensionality introduces computational bottlenecks for generative models.

Additionally, each network traffic trace, considered as a single network flow/session, inherently contains *sequential dependencies* between packets. For example, in TCP, packets need to follow a particular sequence to ensure the integrity and reliability of data transmission. The order of packets, dictated by sequence and acknowledgment numbers, is crucial to reconstruct the transmitted data accurately at the receiver's end. These dependencies are also beneficial to improve ML classification accuracy as they may be unique to different classes of network traffic. Traditional tabular formats fall short in preserving these sequential relations due to their static nature, which could lead to the misrepresentation of the underlying network behavior [40, 150, 151].

Since recent strides in synthetic data generation have centered around image generation [25, 37, 101, 107, 144], we seek to leverage these methods to generate high-fidelity synthetic network data with low computational complexity. Our reasons for adapting these models are: (1) *Maturity of Image Generative Models*: The advancements in the domain of image generative models, such as diffusion models, offer a robust foundation to produce detailed synthetic network traffic. These models have been optimized over years to understand and reproduce intricate patterns in high-resolution images [37, 101, 107]. (2) *Spatial Hierarchies and Connectivity*: Images inherently capture spatial hierarchies, which is crucial for representing intricate inter-packet and intra-packet dependencies in network traffic. Pixels in images naturally form patterns and structures. Deep learning models, especially convolutional neural networks (CNNs), are adept at exploiting these structures to capture both local and global dependencies. Unlike traditional tabular formats where data points might be perceived as independent entities, images inherently emphasize the significance of a packet concerning its neighboring packets, preserving crucial contextual information [28, 73, 85, 116]. A

straightforward example demonstrating the effectiveness of appropriate image representations in capturing network dependencies is the use of edge detection in discerning packet protocol distributions within a flow, detailed later in Section 3.2.3. (3) *Visualization and Interpretability*: Image representations offer a intuitive way to discern packet flows, anomalies, and patterns in network traffic. (4) *Research and Tools Availability*: The extensive research and tools available in computer vision mean that scalability and optimization are already mature, providing a significant advantage when handling high-dimensional data like network traffic [107, 148, 152].

3.1.2 Conversion Process. To arrive at image representations of network traffic, we first encode packet captures (pcaps) using nPrint [58], which converts network traffic into standardized bits where each bit corresponds to a packet header field bit as shown in Figure 1. This binary representation is simple yet effective, where the presence or absence of a bit in the packet header is denoted as 1 or 0 respectively, and a missing header bit is represented as -1. This encoding scheme ensures a standardized representation irrespective of the protocol in use. The payload content is not encoded since it is often encrypted. However, the size of the packet payloads can be inferred from other encoded header fields such as the IP Total Length fields.

Following this encoding, a sequence of packets in a pcap is converted into a matrix, which is then interpeted as an image. The colors green, red, and gray represent a set bit (1), an unset bit (0), and a vacant bit (-1), respectively. This color coding provides a visually intuitive representation of the network traffic. We then group the packets in groups of 1024, representing the packet headers of the first 1024 packets in a flow. Through this process, any network traffic in pcap format is transformed into an image with dimensions of 1088 pixels in width and 1024 pixels in height, with each row of pixels representing a packet in the network traffic flow as shown in Figure 2. Any image in this format can be converted back to pcaps in a straightforward manner. This representation not only preserves the complexity of the data but also retains the essential sequential relationships among packets, laying a robust foundation for the ensuing steps in the NetDiffusion pipeline. We opt for a grouping of 1024 packets, aligning with conventional practices in training and fine-tuning text-to-image diffusion models, where image resolutions are commonly capped at 1024 by 1024 pixels. This choice is primarily driven by computational constraints encountered at higher data resolutions. Nonetheless, this aspect presents an opportunity for future exploration, as we discuss in Section 6, where the potential to handle larger packet groups could be considered.

3.2 Fine-Tuning Diffusion Model and Controlled Generation

Given a real, labeled network traffic dataset, we first transform the network traffic flows into their corresponding image representations as previously described. Leveraging these image-based representations, we then fine-tune a generative model, specifically a diffusion model, to produce synthetic network traffic.

3.2.1 Advantages of Text-to-Image Diffusion Models for Network Traffic Generation. The decision to employ diffusion models for image-based network traffic generation over other generative approaches, such as GANs, is anchored on several compelling advantages:

(1) *High-Fidelity Generation:* Diffusion models excel in capturing and replicating intricate data distributions with remarkable fidelity [56, 99, 112]. This attribute is pivotal, given the complex and nuanced patterns inherent in real network traffic. The ability of diffusion models to closely mimic these patterns ensures that the synthetic traces they produce have high resemblance to real traces.
(2) *High-Resolution Image Handling:* Diffusion models, through techniques like latent diffusion, are adept at generating and managing high-resolution images [95, 101, 107]. This capability is pivotal for our framework, where the image representation of network traffic demands

high resolution for accuracy and detail retention. While diffusion models can be tailed to handle tabular data directly, this may forgo the distinct benefits of image representations, such as capturing spatial and sequential intricacies, as previously discussed.

(3) *Conditional Generation:* By allowing for conditional generation based on textual prompts, text-to-image diffusion models can be instructed to generate network traffic that mirrors specific classes or types, offering an unparalleled fusion of precision and versatility. The model learns the relationship between text and image during its training phase by adjusting its reverse diffusion trajectory based on the given textual prompt [49, 110, 145]. This ensures that the final generated image aligns with desired image distribution. This becomes invaluable when the need arises to produce specific classes of network traffic or to conform to protocol rules and other essential network characteristics.

(4) *Transparency and Training Stability:* The nature of diffusion models ensures a transparent generation process, leading to reproducible results. Such transparency are critical for producing network traces that meet specific patterns or constraints, as it shows good interpretability. Moreover, unlike the often unpredictable training dynamics of GANs due to their adversarial nature [11, 36, 91], diffusion models exhibit stable training behavior and well-behaved gradient dynamics [55, 123, 125]. This stability not only ensures consistent and anomaly-free output but also streamlines the optimization process.

In totality, these advantages make diffusion models a robust and versatile choice for the generation of synthetic network traces, effectively addressing the challenges and constraints observed with other generative techniques. At the same time, a key goal of our study is to promote the generation of highly granular network traffic, with our preference for diffusion models partly influenced by their swift recent advancements. Nevertheless, investigating alternative methods suited for generating long sequential time-series data, such as transformer-based architectures, could be a promising approach. While this is not the focus of this paper, we discuss potential future directions in Section 6.

3.2.2 Fine-Tuning a Stable Diffusion Base Model Using LoRa. Training generative models, especially those as sophisticated as diffusion models, from scratch can be resource-intensive and time-consuming. This is particularly true when considering existing base models like Stable Diffusion 1.5 [107] which have been pre-trained on the LAION-5B dataset [111] containing over 5.85B CLIP-filtered image-text pairs. While the out-of-the-box Stable Diffusion model is undeniably potent, we cannot directly use it to synthesize network traffic because it's designed to cover a broad spectrum of patterns and intricacies, causing it to lack the depth needed in specific generation tasks. For instance, generating images based on task-specific prompts might yield results that, although thematically aligned, lack the precision and high-fidelity one might expect. In our case, a 'Netflix Network Traffic' prompt might yield a generic image like a highway scene within a Netflix player. This is particularly evident when the textual description provided has multiple potential visual interpretations, causing the model to produce images that may be blurry or off-target. By fine-tuning Stable Diffusion on specific network datasets, we can address these limitations. Fine-tuning augments the model's expressiveness, enabling it to better associate with specific patterns or embeddings of the network traffic domain.

As a result, in this framework, we build upon Stable Diffusion 1.5 and fine-tune this model on our specific network datasets as shown in Figure 1, making it aptly suited for generating synthetic network traffic that mirrors the complexities and nuances of real-world network traffic. To facilitate this fine-tuning, we employ Low-Rank Adaptation (LoRa) [59], which is a training technique tailored to swiftly fine-tune diffusion models, particularly in text-to-image diffusion models. Its crux lies in enabling the diffusion model to learn new concepts or styles effectively, while maintaining a

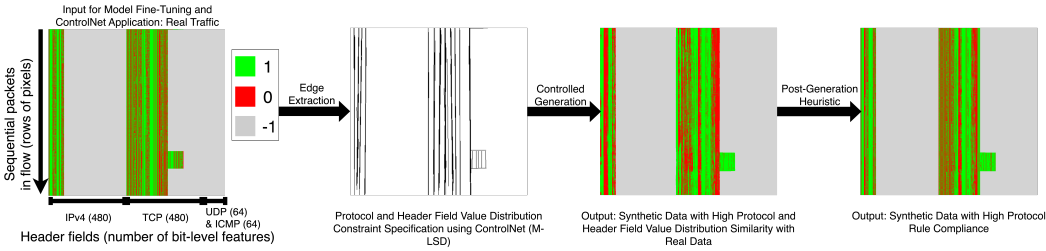


Fig. 2. Synthetic Amazon network traffic outputs: (1) Using ControlNet, we detect regions present in the original traffic and ensure protocol and header field value distribution conformance by generating within specified regions. (2) We apply post-generation heuristic to refine field details for protocol conformance.

manageable model file size. This is beneficial given the traditionally large sizes of models like Stable Diffusion, which can be cumbersome for storage and deployment. With LoRa, the resultant models are compact, striking a balance between file size and training capability. This compactness doesn't sacrifice the model's ability but rather applies minute yet effective changes to the base/foundational model, ensuring that the core knowledge remains intact while adapting to new data.

In our fine-tuning process, we start by sampling classes of real network traffic from our dataset that we aim to generate synthetically. These traffic samples are then transformed into their image representations. For each of these images, we craft an unique encoded text prompt (*e.g.*, 'pixelated network data, type-0' for Netflix traffic) that succinctly describes its class type. Consequently, this results in a number of text prompt categories corresponding to the variety of network traffic types within the dataset. For example, in the subsequent evaluation task (Section 4), we utilize a dataset comprising flows from 10 distinct applications to fine-tune our NetDiffusion pipeline. This results in 10 unique categories of text prompts, each corresponding to different flow applications, e.g., 'pixelated network data, type-5' for Zoom flows and 'pixelated network data, type-6' for Google Meet flows. The choice of our encoded prompt, though seemingly simplistic, achieves two main objectives. It offers a specific vocabulary that reduces ambiguity and ensures the model hones in on the network traffic's nuances. Additionally, it minimizes interference from the base model's original word embeddings, optimizing the generative process. Experimentally, we found that this specific prompt structure provides a balance between specificity and simplicity to prevent overfitting and misinterpretations, leading to better results. Subsequently, these image-text pairs are fed into the fine-tuning process, where the base Stable Diffusion model, augmented with LoRa, learns to generate network traffic images conditioned on our prompts. By merging the power of Stable Diffusion models with the adaptability of LoRa, we create a potent mechanism to generate high-fidelity, synthetic network traffic images, tailored to our specific requirements.

3.2.3 Controlled Prompt-based Generation Via ControlNet. Upon fine-tuning the generation model, the next phase involves generating the desired class of synthetic network traffic. This is achieved by supplying the appropriate text prompts to the diffusion models to produce the image representations of the traffic. Diffusion models operate by simulating a reverse process from a simple noise distribution to the data distribution, which enables them to capture and replicate the intricate patterns inherent in real-world data. The noise is progressively reduced over several steps, allowing the model to gradually refine the generated image until it closely resembles genuine network traffic patterns. We show an example synthetic network traffic in image representation for Amazon traffic in Figure 2. This prompt-based generation process facilitates the creation of a synthetic nPrint-encoded network traffic dataset tailored to specific class distribution requirements. For instance, to curate a dataset with a certain class distribution and size, one would provide the corresponding quantity of text prompts per class and activate the generation process accordingly.

However, a challenge arises from the inherent flexibility of general diffusion models. While they are designed to foster creativity in the generated output, it can lead to anomalies in the context of network traffic generation. For example, generated traffic might incorrectly populate packet header fields, leading to protocol distribution discrepancies between synthetic and real traffic. Such deviations can compromise ML accuracy and make it arduous to ensure strict adherence to critical protocol rules as we describe later. To ensure that the generated traffic closely aligns with the prevalent protocol and header field value distributions observed in real traffic, certain constraints are introduced during the generation process. If, for instance, the actual Amazon network traffic primarily consists of TCP packets, the generation process should prioritize populating header fields associated with TCP packets. This approach ensures that the generated traffic image predominantly features green (set bit) or red (unset bit) pixels corresponding to TCP packet headers, while other pixels remain gray (vacant bit). Moreover, consistent bit characteristics within headers, such as consistently unset bits, should be mirrored in the synthetic output.

Leveraging the controllable nature of diffusion models, we incorporate ControlNet [146] into the generation process. ControlNet is a commonly used neural network architecture designed to add spatial conditioning controls to large, pre-trained text-to-image diffusion models. It capitalizes on the robust encoding layers of these models, which are pre-trained with vast datasets, to learn a diverse set of conditional controls. With "zero convolutions", the architecture gradually grows its parameters from an initialized state, ensuring no adverse noise affects the fine-tuning. ControlNet can work with a range of conditioning controls, from edges and depth to segmentation and human poses. It offers flexibility in training, demonstrating robustness with both small and extensive datasets. In our specific use case of ControlNet, we leverage M-LSD straight line detection for detection the boundaries between fields that are suppose to be populated and those that are not, as shown in Figure 2. Other edge detection methods such as Canny edge detection produce similar results. Such line or edge detection methods are effective because they align with the inherent columnar consistency present in packet traces.

Incorporating ControlNet allows the synthetic generation process to more closely emulate the protocol and header field value distributions observed in real network traffic. This minimizes deviations and ensures that the generated packets largely adhere to the expected protocol type and header field values, enhancing the quality and reliability of the synthetic network traffic dataset. And while ControlNet offers coarse-grained control, determining which image regions to populate, the diffusion model provides fine-grained control, specifying individual pixel values which further contributes to high resemblance to real traffic. In Appendix A, we carry out detailed ablation studies showcasing the necessity of applying ControlNet during the generation process, as well as the generation improvement from fine-tuning a base text-to-image diffusion model. Despite the integration of ControlNet, the inherent variability and generative nature of diffusion models can lead to differences in generated instances. For a detailed analysis of these differences, refer to Appendix B. On the other hand, while our current approach utilizes pretrained ControlNet models as an initial step for automating constraint enforcement, they fall short in supporting more specific constraints, such enforcing and confining highly detailed constraints within smaller header field columns. Future efforts will focus on training ControlNet models dedicated to network traffic synthesis from scratch. This direction, as described later in Section 6, aims to achieve finer-grained control over the synthetic traffic generation process.

3.3 Improving Transport and Network Layer Protocol Compliance

Utilizing the controlled diffusion model, we generate encoded network traffic that adeptly resembles the protocol and header field value distributions inherent in real-world data. Our encoded format not only captures every feature observed in real network traffic but also minimizes the

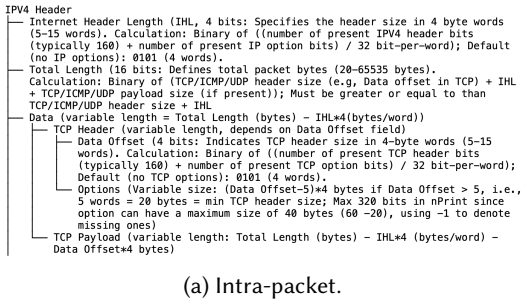
statistical disparity between real and synthetic feature values, ensuring models can recognize and act on underlying patterns. Yet, the domain of network dataset augmentation presents unique challenges. While the synthetic data's quality in ML applications is crucial, its utility extends beyond. The data's relevance in traditional network analysis and testing tasks – often requiring raw network traffic – becomes equally significant. Despite the guidance provided by ControlNet during the generation process, converting our synthetic encoded traffic back to raw formats, like pcaps, isn't straightforward. This complexity arises from the multitude of detailed transport and network layer protocol rules at both inter and intra-packet levels. Properly formatted traffic must strictly adhere to these rules. We now explore these complexities and their implications.

3.3.1 Inter and Intra Packet Transport and Network Layer Protocol Rules. At its core, both transport and network layer protocol rules define the conventions and constraints that ensure seamless communication between devices in a network. These rules are crucial as they dictate the structure, formatting, and sequencing of packets, ensuring that data transmission occurs efficiently and reliably. While transport layer rules emphasize end-to-end communication and reliability, network layer protocols focus on packet routing and address assignment. Combined, these rules can be broadly divided into two categories:

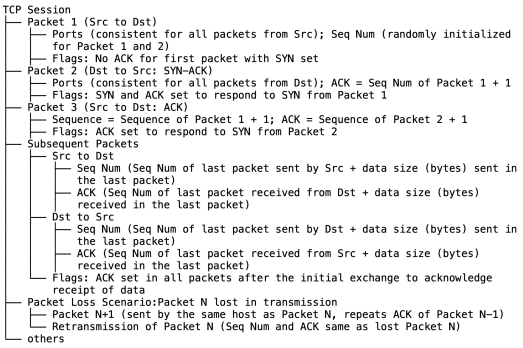
(1) *Inter-Packet Rules:* These rules dictate the relationships and sequencing between header fields within multiple packets in a network flow. For instance, in a typical TCP connection, packets need to be sequenced properly, starting with the handshake process involving SYN and SYN-ACK flags. The integrity of data transfer is ensured by aligning sequence numbers and acknowledgment numbers. Misalignment or incorrect sequencing can disrupt the connection or data transfer process.
(2) *Intra-Packet Rules:* These pertain to the structure and contents within individual packets. For example, many protocol headers have a checksum field computed based on the packet's contents to detect errors during transmission. It's crucial that the checksum is consistent with the packet's payload. Additionally, certain fields within a packet, such as port numbers in TCP and UDP headers, must adhere to specific formatting and value constraints to ensure the packet's validity and proper routing.

Ensuring compliance with these rules is vital. Properly formatted traffic is not only more efficient but also crucial for network applications devices, such as analysis tools, routers, and firewalls, which rely on well-structured packets to function correctly. The challenge with synthetic data generation, especially when optimized for ML accuracy, is that ML models primarily focus on patterns within features that contribute to classification or prediction accuracy. These models might overlook intricate protocol rules in favor of patterns that enhance classification performance. For example, a ML model might deem certain bit patterns as significant for classifying a particular type of traffic, even if those patterns violate protocol rules. While our use of ControlNet aids in approximating the general protocol and header field value distributions by ensuring correct field population, it does not fully capture the nuances of specific bit-level values. This disparity between ML optimization and protocol rule compliance accentuates the necessity for post-generation adjustments. Instead of entirely overhauling the ML generation process, which would require embedding a vast amount of rule-bound constraints derived from domain knowledge, post-generation adjustments offer a more manageable approach to refine the generated data for protocol compliance. Implementing such detailed control during generation remains a challenging endeavor, and we envision addressing these complexities in future work.

3.3.2 Post-Processing Heuristic for Protocol Rule Compliance. To maximize the encoded synthetic network traffic's compliance with transport and network layer protocol rules, we first discern a subset of critical header fields that mandate strict adherence to their formatting rules, *e.g.*, sequence



(a) Intra-packet.



(b) Inter-packet.

Fig. 3. Example TCP protocol rules/dependencies.

and acknowledgement numbers. In contrast, some fields can accommodate a degree of flexibility without compromising the integrity of the network traffic, such as TCP window size or TTL. The objective here is to limit the scope of fields requiring modification during post-processing, ensuring we retain as much of the original generative model's output as possible. By doing so, we minimize potential impacts on ML-driven tasks, while still ensuring the synthetic traffic's compatibility with network analysis tools. With the critical fields identified, we develop a systematic way to calculate their correct values based on other generated fields. This is achieved by constructing two dependency trees—one for intra-packet header field dependencies and another for inter-packet dependencies. These trees are built upon domain knowledge and are sourced from standard network protocol documentation [1–5, 16]. Although constructing them requires significant manual effort to extract critical protocol rules from these standards, this process is a one-time endeavor. Future applications and iterations of our tool will benefit from this foundational work without the need for repeated effort. We present example protocol rules and the associated dependency trees for TCP protocol in Figure 3. More comprehensive and detailed dependency trees can be found in the open-sourced repository[2].

Given a generated encoded network traffic, we begin the correction process by traversing the trees in an automated, bottom-up fashion. Initially, we satisfy intra-packet dependencies, ensuring that individual packets are internally consistent. Subsequently, we address inter-packet dependencies, guaranteeing that the packets in a flow relate correctly to one another. Certain fields necessitate uniformity across packets within the same network traffic trace—like IP addresses and ports. Others require specific initialization values, such as the IP identification number and TCP acknowledgment number. To determine the most appropriate values for these fields, we employ a majority voting system by selecting the most frequently appearing value within the generated traffic. Another notable challenge is timestamp assignment for individual packets, given its intricate time-series dependencies. Diffusion models, while adept at spatial dependencies, might struggle with long-range temporal patterns inherent in time-series. As a result, our current generation process isn't fully optimized for this. As an interim solution, we sample original time distributions from real traffic to produce similar timestamp distribution in the post-generated synthetic data. An example post-processing algorithm for realizing the intra-packet dependency adjustments for IP headers is in Algorithm 1. We recognize the potential for improvement and plan to address this in future research. Post these steps, the synthetic traffic should be in a state where it closely adheres to the essential protocol rules we've identified. This post-processing ensures that the encoded synthetic

[2]https://anonymous.4open.science/r/packet-capture-dependency-DB0C/README.md

Algorithm 1 Post-Processing Heuristic Example: Intra-packet Dependency Adjustments for IP headers

Input: Diffusion model generated synthetic traffic
Analyze and sort source IP address distribution (P_{src}) from a radomly sampled real flow ▷ Ensure that packet directions in the synthetic flow closely align with those in real flows.
Identify the top two IPs (IP_1, IP_2) with the highest occurrence from P_{src} ▷ Select the most frequent source IP addresses as primary nodes for transition probability modeling.
Construct synthetic IP address sequence ($IP_{sequence}$) using transition probabilities $T_{IP_1 \to IP_2}$ and $T_{IP_2 \to IP_1}$ ▷ Use transition probabilities to model the likelihood of switching between IP_1 and IP_2 in the synthetic sequence, reflecting packet flow dynamics in real traffic.
Correct packet directions within synthetic flow using $IP_{sequence}$
if version_field ≠ 'IPv4 (0100)' **then** set to 'IPv4 (0100)' ▷ Ensure Version field adheres to IPv4 standards for compatibility.
for bit in IPv4 headers: **if** bit = -1 **then** set bit to random (0,1) ▷ Replace undefined bits in IPv4 headers with random binary values to maintain structural integrity.
Identify IPv4 protocol $\mathcal{P}$ in the synthetic flow by maximizing $\frac{\sum \text{non-negative bits in } \mathcal{P}}{\text{total bits in } \mathcal{P}}$ ▷ Optimize protocol identification by maximizing the ratio of valid to total bits in IPv4 protocol.
for bit in non-IPv4 headers: **if** $bit \notin \mathcal{P}$ **then** set bit to -1 ▷ Discard irrelevant bits in non-IPv4 headers to focus on IPv4 protocol consistency.
for bit in IPv4 options: **if** bit ≠ -1 **then** set bit to -1 ▷ Render IPv4 options bits obsolete to reflect modern Internet protocols.
if IPv4 TTL = 0 **then** set TTL using majority voting across all packets ▷ Minimum IPv4 TTL guarantee to prevent packet expiration.
Set HL field to number of active bits in the IPv4 fields ▷ Adjust Header Length (HL) field to accurately represent the active bits in IPv4 headers.
Output: Post-processed synthetic traffic with IP header intra-packet dependency compliance

traffic can be seamlessly converted into raw network traffic formats (like pcap) and subsequently be utilized for a range of non-ML tasks.

4 EVALUATION

To assess the effectiveness of our generative framework, we applied it to an exemplary real network traffic dataset, generating its synthetic counterpart as a case study. Our ML-oriented evaluation comprises two main analyses: a statistical comparison to gauge the fidelity of the synthetic data and a model accuracy assessment to determine its utility in enhancing ML outcomes. In the non-ML scenarios, we explore the broader implications of our synthetic data by applying it in diverse network analysis and testing scenarios. The choice of our baseline dataset, which we detail next, serves as a demonstration of our method's validity.

4.1 Dataset Overview and Synthetic Traffic Generation

Our dataset for real network traffic, summarized in Table 2, comprises pcap files that capture traffic from ten prominent applications in areas such as video streaming, video conferencing, and social media. Although we focus our study on a single dataset, it comprises varied network data collected from multiple scenarios, locations, and times from three different data sources [22, 62, 86], underpinning our methodology's applicability and generalizability. During preprocessing, we analyze DNS queries to identify relevant IP addresses for the specified services and applications, keep only packets associated with these IPs, and split the traffic into individual flows. We retain the application and service label for

Macro Services	Total Flows	Application Labels	Collection Date
Video Streaming [22]	9465	Netflix YouTube Amazon Twitch	2018-06-01
Video Conferencing [86]	6511	MS Teams Google Meet Zoom	2020-05-05
Social Media [62]	3610	Facebook Twitter Instagram	2022-02-08

Table 2. Summary of the real network traffic dataset: 10 applications across these three macro service types.

Data Format	Generation Method	All Generated Features			Ex. Common Feature: Protocol		
		Avg. JSD	Avg. TVD	Avg. HD	Avg. JSD	Avg. TVD	Avg. HD
NetFlow	Random Generation	0.67	0.80	0.76	0.82	0.99	0.95
	NetShare	0.16	0.16	0.18	0.04	0.03	0.04
Network Traffic (pcap)	Random Generation	0.82	0.99	0.95	0.83	1.00	1.00
	NetDiffusion	**0.04**	**0.04**	**0.05**	**0.02**	**0.03**	**0.02**

Table 3. Average normalized statistical differences between real and synthetic network data across (1) all generated fields and (2) example commonly generated field – IPv4 protocol: Jensen-Shannon Divergence (JSD), Total Variation Distance (TVD), and Hellinger Distance (HD).

each processed flow, which is later used for generating text prompts and assessing classification accuracy[3]. The comprehensive dataset contains nearly 20,000 flows. For feasibility and consistency in our evaluations, we randomly sampled 10% of this collection We also carried out evaluations on both larger and smaller subsets of the dataset and obtained comparable results. We adapted the fined-tuned diffusion model to this dataset, resulting in the generation of a synthetic dataset as outlined in the previous section. The volume of this synthetic dataset adjusts based on specific evaluation requirements, as we will detail further. The prompt-driven nature of diffusion model allows for the generation of synthetic network traffic in any desired quantity, providing flexibility for diverse analytical needs.

4.2 Statistical and ML Performance Analysis

4.2.1 Statistical Similarity Results. A primary measure of synthetic data quality is its statistical resemblance to the original data. This comparison is critical as the essence of synthetic data lies in its ability to represent the statistical properties of the real data without mirroring it exactly. Ensuring statistical similarity ensures that models trained on synthetic data generalize well to real-world scenarios. In our evaluation, we benchmarked our synthetic data against two baselines: the NetShare method, which produces synthetic NetFlow attributes and outperforms most of the othe GANs-based methods [142], and a naive random generation approach. The latter, by generating purely random values, acts as a worst-case scenario, illustrating the lower bounds of similarity and underscoring the value added by more sophisticated methods. While our diffusion model inherently captures a broader set of network attributes, for fairness in comparison, we examine similarity both at an aggregated level, encompassing all features, and at a more focused level, targeting only the common features between NetDiffusion and NetShare – using the Protocol attribute as an example.

We employ three distinct metrics to quantify statistical similarity: Jensen-Shannon Divergence (JSD), Total Variation Distance (TVD), and Hellinger Distance (HD). JSD gauges informational overlap between distributions, offering insights into shared patterns. TVD captures the maximum difference between two distributions, highlighting worst-case discrepancies. Meanwhile, HD, rooted in Euclidean distance, is especially sensitive to differences in the tails of distributions, shedding light on distinctions in rare events or outliers. Collectively, these metrics provide a holistic view of the statistical overlap between the real and synthetic datasets. Values for all three metrics range between 0 and 1, with values closer to 0 indicating superior statistical similarity and thus a closer resemblance to the original dataset.

The results in Table 3 offer a nuanced view into the challenges and successes of synthetic data generation for network traffic. At a foundational level, raw network traffic in pcap format is inherently more intricate than the NetFlow format. This complexity is evident in the stark contrast in statistical distances when generating synthetic data for these two formats using random methods. The higher statistical distance observed for the randomly generated raw network traffic underscores the inherent challenges in replicating its multifaceted nature. Yet, it's this very complexity that

[3]All traces used in this paper are sanitized and contain no personally identifiable information (PII).

highlights the prowess of NetDiffusion. Despite pcap being a more challenging format, NetDiffusion—specialized in generating raw network traffic—demonstrates a remarkable capability. Its statistical distances relative to real network traffic are notably lower than even NetShare's distances when NetShare is generating for the simpler NetFlow attributes. This finding underscores the efficacy of NetDiffusion in producing high-fidelity synthetic data for intricate formats like pcap.

In summary, while the inherent challenges in replicating the complex pcap format are evident, NetDiffusion's capability to produce synthetic data with high statistical similarity, even surpassing methods for simpler formats, validates its potential as a robust tool for data augmentation in the realm of raw network traffic.

4.2.2 ML Classification Results. To gauge the efficacy of our synthetic network traffic in ML-based data augmentation, we employ two classification tasks. The first task aims to categorize network flows at a granular level, aligning them with their corresponding applications (micro-level). The second task operates at a broader scale, classifying flows into their overarching services (macro-level). We conduct evaluation using three prominent models, including random forest (RF), decision tree (DT), and support vector machine (SVM). We adopt accuracy as the performance metric for these ML-driven augmentation evaluations due to its intuitive interpretability, allowing for a straightforward comparison between different augmentation techniques. Specifically, in scenarios involving classification tasks where classes are (or are made to be) approximately balanced, accuracy provides a clear picture of how well the model performs across all classes. Three distinct augmentation scenarios, utilizing synthetic data, are evaluated:

- *Complete Synthetic Data Usage:* Here, either the training or testing set is entirely composed of synthetic data, *e.g.*, training exclusively on synthetic data and testing on real data, and vice versa. This approach tests the robustness of synthetic data and its capacity to emulate real-world data intricacies. Using synthetic data in isolation ensures that models are not biased by any inherent patterns of the real dataset during training, allowing for an assessment of the synthetic data's standalone quality.
- *Mixed Data Proportions:* Synthetic data is interspersed with real data at varying proportions, *e.g.*, a 50-50 split between synthetic and real data during training. This strategy evaluates the synergy between real and synthetic data. Mixing allows models to benefit from the diversity of synthetic data while still grounding the learning process in real-world patterns, potentially improving generalization.
- *Class Imbalance Rectification:* Synthetic data is employed specifically to address and rectify class imbalances in the training set. For instance, underrepresented classes in the real dataset are augmented using synthetic data until a balance is achieved. This targeted augmentation ensures that the model is exposed to a balanced representation of all classes, mitigating biases and improving performance on minority classes. Addressing class imbalance is crucial as it prevents models from becoming skewed towards overrepresented classes, thereby enhancing their predictive accuracy across all classes.

Result on Complete Synthetic Data Usage. The results presented in Table 4 reveal that our method, which specializes in generating raw network traffic data in pcap format, consistently outperforms the NetShare approach, which is tailored for the simpler NetFlow data format. Note that the classification accuracy tends to be higher for the macro-level task in all train/test scenarios. This is anticipated because the macro-level task categorizes broad traffic service types (like video streaming vs. video conferencing), while the micro-level task involves finer distinctions between specific applications (e.g., YouTube vs. Twitch). The increased specificity of the micro-level task generally yields lower accuracy compared to the broader macro-level classification.

Training/Testing	Data Format (Generation Method)	Post-Gen. Heuristic	Highest Accuracy (Model) Macro-level	Micro-level
Real/Real	Network traffic (N/A)	N/A	1.00 (SVM)	0.978 (SVM)
	Netflow (N/A)	N/A	0.86 (RF)	0.648 (DT)
Synthetic/Real	**Network traffic (NetDiffusion)**	**Not Used**	**0.738 (DT)**	**0.262 (DT)**
		Used	**0.676(DT)**	**0.222 (DT)**
	Netflow (NetShare)	N/A	0.396 (RF)	0.140 (SVM)
Real/Synthetic	**Network traffic (NetDiffusion)**	**Not Used**	**0.542 (SVM)**	**0.249 (SVM)**
		Used	**0.529 (SVM)**	**0.182 (SVM)**
	Netflow (NetShare)	N/A	0.503(SVM)	0.102 (RF)

Table 4. Across all complete synthetic data usage scenarios, NetDiffusion augmented dataset yields to higher classification accuracy. Only the top-performing model is displayed.

Rank	Real/Real	Synthetic/Real
1	3_tcp_wsize_13: 0.0115	1_tcp_opt_92: 0.0125
2	3_tcp_wsize_15: 0.0107	2_udp_cksum_14: 0.0098
3	1_udp_len_5: 0.0100	8_tcp_opt_78: 0.0093
4	1_tcp_opt_24: 0.0099	4_tcp_opt_93: 0.0085
5	1_ipv4_tl_5: 0.0090	9_udp_cksum_14: 0.0085
6	0_tcp_opt_6: 0.0088	5_tcp_opt_93: 0.0084
7	0_ipv4_dfbit_0: 0.0088	5_tcp_urp_8: 0.0082
8	1_tcp_opt_6: 0.0086	9_tcp_cksum_1: 0.0081
9	9_ipv4_proto_3: 0.0085	6_tcp_opt_78: 0.0079
10	1_tcp_opt_23: 0.0076	2_tcp_opt_93: 0.0079

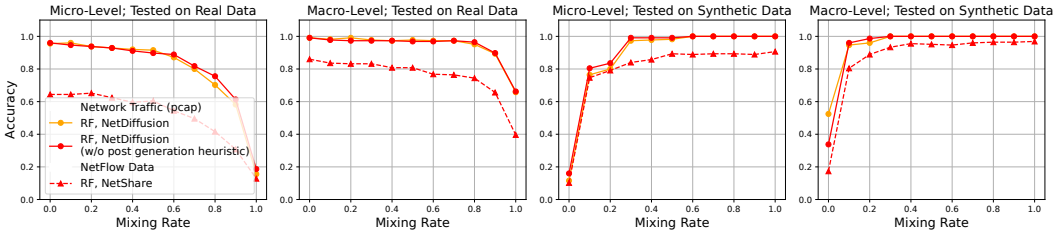
Table 5. Macro-level RF feature importance for complete NetDiffusion synthetic data usage; Green highlights denote shared header fields with the real/real scenario. Feature structure: packet_protocol_header_bit.

The rich feature space inherent in raw network traffic offers a plethora of learnable patterns that can bolster model accuracy. With a broader and more intricate feature set, there's more room for the model to identify and leverage intricate patterns, nuances, and correlations within the synthetic network traffic data to enhance its predictive prowess. At the same time, the higher statistical similarity between real and our synthetic datasets (as we observed previously) implies that the synthetic network traffic data mirrors real-world patterns more closely. This, in turn, means that features in the real dataset that are pivotal in distinguishing between network flows are likely to retain their discriminative power in the synthetic dataset. Such preservation of feature significance ensures that models trained on synthetic data can generalize more effectively to real-world scenarios. Supporting this is our feature importance analysis for the RF model in the macro-level classification task as shown in Table 5. When trained on NetDiffusion synthetic data and tested on real data, the RF model exhibited a propensity to prioritize features (on the header level) that are also critical when both training and testing are done on real data. This nuanced focus on specific feature subsets is indicative of the model's ability to discern and leverage patterns in the synthetic data that are reflective of real-world traffic. While we use the RF model for subsequent detailed analysis due to its relatively consistent performance and interpretability across scenarios, our later sections will delve into a broader spectrum of model performances.

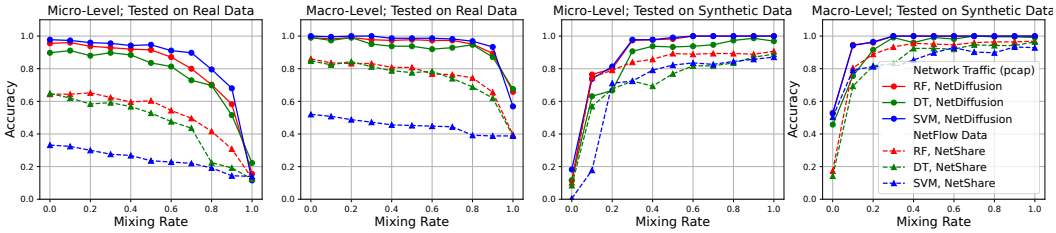
An additional layer to our analysis pertains to the post-generation heuristic for enhancing the synthetic network traffic's adherence to protocol rules while minimally altering the diffusion-generated outputs. The heuristic affects only ~8% of the synthetic traffic features which produces marginal influence on ML performance, with accuracy reductions ranging from 0.013 to 0.067 across all scenarios. This minor accuracy degradation aligns with expectations: Diffusion models are adept at achieving high statistical similarity to real data during fitting and generation, enhancing ML accuracy. However, the necessary post-generation heuristic adjustments for protocol compliance inevitably alter the output. Although we strive to minimize this interference, it does slightly impact accuracy, as it deviates from the model's original generation output.

Result on Mixed Data Proportions. We introduce the "mixing rate" to denote the percentage of real data replaced by synthetic data in the training set. This approach ensures the training set size remains constant across diverse mixing rates, enabling a clear evaluation of the interplay between the mixing rate and the resultant model accuracy. Introducing a controlled blend of synthetic data into real datasets can often enhance model robustness by potentially introducing diverse patterns, a practice that is considered standard in data augmentation.

Using the RF model as an example, Figure 4a shows that models trained with dataset augmented with NetDiffusion-generated traffic consistently achieve higher classification accuracy than those with NetShare-produced NetFlow attributes. When testing on real data, models trained entirely on



(a) Classification accuracy comparison using the RF model with mixed data proportions. Datasets augmented with NetDiffusion-generated traffic consistently outperform those using NetShare-produced NetFlow attributes.



(b) Comparative ML performance across different model choices using NetDiffusion-augmented datasets versus NetShare-augmented NetFlow datasets. NetDiffusion consistently yields superior results.

Fig. 4. Evaluation result on mixed data proportions.

real network traffic demonstrate notably higher accuracy than those trained solely on real NetFlow (also highlighted in Table 4). This starting accuracy discrepancy is critical. When integrating synthetic network traffic data into training, any potential degradation in accuracy is offset by the inherently higher baseline accuracy of real network traffic. In simpler terms, with a more accurate starting point (real network traffic), there exists more "buffer" before accuracy noticeably degrades. Another pivotal factor is the higher statistical similarity of NetDiffusion-generated traffic to real data, compared to the similarity of NetShare's NetFlow data to real NetFlow. With this closer resemblance, as we incorporate more synthetic data into the training, the gradual decline in accuracy is less pronounced than when using NetFlow data, especially in macro-level classification. This advantage is not confined to testing on real data. Even when evaluating on synthetic data, models trained with NetDiffusion's output generally outperform those trained with NetShare. Additionally, we carry out a similar analysis as in the case of complete synthetic data usage by examing the effects of applying our post-generation heuristic on the model accuracy, as depicted in Figure 4a. Consistent with the previous findings from Table 4, the example RF model experiences little to no accuracy degredation as a result of the post-generation modification.

A notable observation is the sharp decline in accuracy when the data composition of the training set diverges significantly from the test set. For instance, macro-level classification accuracy drops when the mixing rate exceeds 0.8. This is expected since adequate samples from the test data distribution are needed in the training set for effective cross-validation. As the mixing rate increases, models might overfit to the synthetic data, hindering their performance on real data. This behavior is evident in the changing feature importance with increasing mixing rates, as seen in Table 7. In practical scenarios, it's rare to rely heavily or solely on synthetic data for training. Our results suggest that, barring extremes, NetDiffusion-generated traffic can be effectively used for training.

Lastly, we show that across different model choices, as shown in Figure 4b, NetDiffusion-augmented datasets generally lead to better ML performance than NetShare-augmented NetFlow datasets. Notably, the SVM classifier demonstrates markedly superior performance when tasked

	Mean (μ)	Med	Min	Max	Std Dev (σ)	Range	Var (σ^2)
Before Balancing	10.00%	8.89%	4.44%	17.78%	5.13%	13.34%	26.37%
After Balancing	10.00%	10.00%	10.00%	10.00%	0.00%	0.00%	0.00%

Table 6. Synthetic balancing on under-represented classes to mitigate class imbalance.

Rank	Mixing Rate (Accuracy)		
	0.0 (0.960)	**0.4 (0.928)**	**1.0 (0.187)**
1	5_ipv4_ttl_0: 0.0081	1_ipv4_ttl_3: 0.0053	5_ipv4_ttl_3: 0.0055
2	4_tcp_wsize_10: 0.0061	4_ipv4_ttl_3: 0.0049	8_tcp_cksum_14: 0.0054
3	5_tcp_wsize_4: 0.0061	1_ipv4_ttl_0: 0.0047	7_ipv4_ttl_4: 0.0052
4	1_ipv4_ttl_0: 0.0061	5_ipv4_ttl_3: 0.0046	4_ipv4_ttl_5: 0.0052
5	1_ipv4_dfbit_0: 0.0059	4_ipv4_ttl_0: 0.0045	3_ipv4_tl_12: 0.0047
6	4_ipv4_ttl_0: 0.0059	7_ipv4_tl_14: 0.0044	1_udp_cksum_4: 0.0044
7	4_ipv4_ttl_3: 0.0055	1_ipv4_dfbit_0: 0.0040	6_ipv4_tos_4: 0.0042
8	4_ipv4_ttl_1: 0.0054	4_tcp_wsize_10: 0.0039	2_ipv4_ttl_3: 0.0041
9	5_ipv4_ttl_1: 0.0053	5_ipv4_ttl_1: 0.0038	6_ipv4_ttl_3: 0.0040
10	1_tcp_opt_54: 0.0053	5_ipv4_ttl_0: 0.0036	8_tcp_wsize_13: 0.0040

Table 7. Feature importance at varying mixing rates for micro-level classification on real network traffic data. Red highlights indicate header fields that are not in the top 10 most important features in the real/real scenario (mixing rate = 0).

Training Data (Balancing Source)	**Model**	**Test Accuracy Pre/Post Balancing**	**Δ Acc.**
Network Traffic (NetDiffusion)	**RF**	**0.982 → 0.986**	**0.004** **Facebook (0.955 → 1.00)**
	DT	**0.973 → 0.982**	**0.009** **Meet (0.909 → 1.00)** **Zoom (0.955 → 1.00)**
	SVM	0.991 → 0.991	0
Netflow Data (NetShare)	RF	0.645 → 0.628	-0.017
	DT	0.603 → 0.600	-0.003
	SVM	0.290 → 0.290	0

Table 8. Comparison of micro-level classification accuracy: Class balancing using NetDiffusion synthetic data contributes to accuracy improvement on minority classes.

with classifying raw network traffic as opposed to NetFlow traffic. SVMs are intrinsically adept at handling datasets with high dimensionality and complex relationships between features. The reason lies in SVM's ability to transform the original data into a higher-dimensional space and find optimal hyperplanes to segregate different classes. The richer and more intricate the feature space, the more advantageous this capability becomes. This observation accentuates the importance of NetDiffusion in generating synthetic network traffic, which retains the intricacies of real traffic, allowing sophisticated classifiers like SVM to effectively discern patterns and relationships.

Result on Class Imbalance Rectification. Class imbalance is an ubiquitous challenge in many datasets used for training. For instance, in our collected network trace, the least represented application class constituted a mere 4.44% of the total flow samples, while the dominant class accounted for 17.78%, as shown in Table 6. Such imbalances can negatively skew model performance, as models trained on imbalanced data may struggle to correctly classify underrepresented classes, especially when real-world test data exhibits a more balanced distribution. To combat this, a plausible approach is to selectively augment the training set by appending synthetic data to minority classes, ensuring an even class distribution. Simultaneously, by limiting the addition of synthetic data to well-represented classes, we minimize the drawbacks associated with integrating synthetic data, such as the risk of accuracy degradation from overly introduced variations and insufficient amount of real data in the training set, as we observed in the case of mixed data proportions. Using synthetic data offers advantages over traditional methods like SMOTE [26], random oversampling [15], ADASYN [52], and boosting [47]. It produces diverse and novel examples, enriching the feature space and bolstering model generalization to unseen real-world scenarios. While techniques like SMOTE replicate close counterparts of real data, they might miss certain variations.

We apply synthetic balancing using NetDiffusion-generated network traffic by iteratively generating instances of the underrepresnted classes until all classes have equal representation, resulting in a balanced network traffic dataset across all applications. Similarly, the NetFlow dataset is balanced using synthetic attributes from NetShare. However, due to the limitation of GAN-based methods in prompt-invoked generation, we address underrepresentation by independently training a NetShare GAN for each specific class. Our evaluation reveals that models trained on the balanced NetDiffusion dataset either match or outperform those trained on the original imbalanced dataset as shown in Table 8. Notably, the accuracy gains were predominantly attributed to the improved

performance on the previously underrepresented classes. For instance, with the DT model, a notable 0.09 increase in overall classification accuracy is observed using the NetDiffusion augmented dataset. A breakdown of this improvement pinpoints Meet and Zoom traffic, two underrepresented classes with sample counts roughly *half or less than* the most populated class in the original real dataset, as the primary beneficiaries. Their classification accuracies improved by 0.091 and 0.045, respectively. In contrast, classifiers trained using the NetShare augmented NetFlow dataset do not yield such gains and occasionally even faced accuracy degredations. This further underscores the higher fidelity of NetDiffusion-generated traffic, which not only mirrors real data more closely but also supports larger feature space to enhance model performance.

4.3 Extendability to Additional Network Anlysis and Testing Tasks

The efficacy of synthetic data augmentation extends beyond just ML performance, especially within the networking realm. While ML-centric tasks may primarily focus on ensuring that generated, encoded network traffic produces a consistent feature set with high statistical similarity to real traffic, conventional network analysis and testing tasks demand the conversion of this generated data back into raw formats, such as packet captures. Moreover, these tasks require adherence to specific protocol rules, as elaborated in Section 3.3. By harnessing the capabilities of ControlNet and the post-generation heuristics we employ, NetDiffusion facilitates the generation of network traffic that can be seamlessly converted into raw packet captures while maintaining a robust adherence to protocol rules. To illustrate this, we use the synthetic Amazon network traffic generated by NetDiffusion as an example and show that (1) the generated traffic can be smoothly parsed and interpreted by Wireshark, a renowned network analysis tool, without encountering exceptions and (2) the synthetic traffic supports retransmission, as corroborated using the established packet retransmission tool, tcpreplay. Beyond these observations, we demonstrate that NetDiffusion-generated traffic can successfully support a broad spectrum of common network tasks, ranging from intricate traffic analyses to network behavior studies. Crucially, the derived features routinely employed in these tasks can be extracted from NetDiffusion's outputs using Scapy [106]. This underscores the versatility of our approach, suggesting that NetDiffusion's synthetic network traffic can integrate into a multitude of network analysis and testing tasks beyond the confines of ML-centric applications.

We abstain from juxtaposing NetDiffusion's capabilities with NetShare in this section. The rationale is simple: NetShare's design fundamentally restricts it to generating a limited set of statistical attributes, which inherently curtails its ability to be converted into raw packet captures, a prerequisite for the tasks discussed here.

Wireshark Parsing Analysis. Table 9 details the results from Wireshark's parsing of the NetDiffusion synthetic traffic, stored as capsinfo log [16]. Several observations can be drawn: (1) *Data Format and Integrity:* The generated traffic is stored in the standard pcap format with Raw IP encapsulation. This confirms the synthetic data's adherence to widely accepted network trace data formats, ensuring broad compatibility with networking tools. (2) *Comprehensive Metrics:* All the essential metrics that Wireshark uses to describe and analyze traffic are present, ranging from packet count and data size to encapsulation and timing details. These observations underscore our design's success in producing protocol rule-compliant synthetic traffic, ensuring compatibility with analysis tools like Wireshark that demand structural and semantic correctness.

Tcpreplay's Retransmission Analysis. Table 10 shines light on the retransmission capabilities of the synthetic traffic via tcpreplay [43]. Notable results are: (1) *Successful Retransmission:* All 1,024 packets were successfully sent without any failures or truncations, indicating the traffic's high fidelity and adherence to transport layer protocol rules. (2) *Correct Packet Handling:* Metrics like retried packets standing at zero and the exact match of unique flow packets to successful packets

Attribute	Value
File type	Wireshark/tcpdump/... - pcap
File encapsulation	Raw IP
File timestamp precision	microseconds (6)
Packet size limit (file hdr)	65535 bytes
Number of packets	1,024
File/Data size	1,335 kB/1,318 kB
Capture duration	0.602296 seconds
First/last packet time (absolute)	00:00:00.000000/00:00:00.602296
Data byte/bit rate	2,189 kBps/17 Mbps
Average packet size/rate	1287.76 bytes/1,700 packets/s
Strict time order	True
Number of interfaces in file	1
Interface #0 info	Encapsulation = Raw IP (129 - rawip4) Capture length = 65535 Time precision = microseconds (6) Time ticks per second = 1000000 Number of stat entries = 0 Number of packets = 1024

Table 9. Wireshark capinfos validation log on parsing NetDiffusion generated Amazon network traffic.

Metric	Value (NetDiffusion)	Value (Real)
Total Packets Sent	1024	1024
Total Bytes Sent	1318664 bytes	164189 bytes
Duration	0.613297 seconds	0.694616 seconds
Rate (Bps)	2150123.0 Bps	236454.7 Bps
Rate (Mbps)	17.20 Mbps	1.89 Mbps
Packets per Second (pps)	1669.66 pps	1474.70 pps
Total Flows	2	2
Flows per Second (fps)	183.06 fps	2.88 fps
Unique Flow Packets	1024	1024
Successful Packets	1024	1024
Failed Packets	0	0
Truncated Packets	0	0
Retried Packets (ENOBUFS)	0	0
Retried Packets (EAGAIN)	0	0

Table 10. Tcpreplay results on retransmitting (1) NetDiffusion generated and (2) real Amazon network traffic.

further reiterate the synthetic traffic's quality. (3) *Metrics Completeness:* Key metrics like data bit rate and packet rate, essential for evaluating traffic characteristics, are present and well-defined in both synthetic and real traffic. However, there are noticeable discrepancies in metric values, such as total bytes sent and rates, between synthetic and real traffic. This variance is anticipated, given that NetDiffusion generates traffic flows at a bit level. Consequently, even minor deviations, such as a single-bit difference within an 8-bit header field, can lead to significantly altered field values. Addressing this issue could involve more precise controls during generation, as we discuss in Section 6, or implementing value rescaling in post-generation heuristic, which we leave to future work.

Feature Extraction for Core Network Analysis Tasks. One of the distinguishing attributes of NetDiffusion generated traffic, compared to earlier works like NetShare, is the ability to derive detailed metrics from the synthetic traffic using tools such as Scapy, making it exceptionally valuable for an expansive range of network analysis tasks. To demonstrate this, we evaluate the applicability of our synthetic traffic on representative tasks including traffic and protocol analysis [14, 19, 41, 69, 127, 147], network performance assessment [19, 41], device identification [70, 89], routing and user behavior characterization [12, 18, 51, 108, 133], and error evaluation [60, 113, 126]. As shown in Table 11, using Amazon traffic as an illustrative example, our synthetic network traffic effectively delivers both fundamental metrics, such as packet and byte counts, as well as more nuanced measures like the average TTL used for routing behavior analysis. Furthermore, metrics linked to network performance, device identification, routing behavior, and error analysis further accentuate our synthetic traffic's authenticity and granularity. A noteworthy point is the zero count for error metrics like checksum errors and fragmented packets in both real and synthetic datasets, underscoring our synthetic traffic's semantic correctness. While there are variances between some of the metric values from our synthetic data and real traffic, especially in areas like TCP flag distribution, these differences are inherent to our generative approach. Instead of merely replicating real data values, our method generatively produces them, introducing variations to enhance data diversity. The delicate balance between maintaining the realism of these variations and the utility of the resultant metrics for downstream tasks necessitates a nuanced, task-specific assessment. In future iterations, we aim to focus on enhancing the congruence between synthetic and real data metrics, such as addressing the overrepresentation of non-ACK packets in the synthetic data – a reflection of the complexities tied to emulating the TCP state machine.

Network Task	Metric	Unit	Value (Real Network Traffic)	Value (NetDiffusion Generated Traffic)
Traffic Analysis	Packet Count	packets	1024	1024
	Byte Count	bytes	1100406	1318664
	Avg. TCP Window Size	bytes	32739.95	13234.63
Protocol Analysis	Protocol Distribution	packets	TCP: 1024, UDP: 0, ICMP: 0	TCP: 1024, UDP: 0, ICMP: 0
	Flags Distribution	flags	SYN: 0, ACK: 1023, FIN: 0, RST: 0, PSH: 0, URG: 16	SYN: 68, ACK: 574, FIN: 556, RST: 1, PSH: 6, URG: 495
	Src Port Distribution	port (packets)	46508 (303), 443 (721)	30202 (376), 14443 (648)
	Dest Port Distribution	port (packets)	443 (303), 46508 (721)	14443 (376), 30202 (648)
Network Performance	Packet Size Distribution	packets	0-499: 306, 500-999: 6, 1000-1499: 6 1500-1999: 706, 2000+: 0	0-499: 35, 500-999: 56, 1000-1499: 257 1500-1999: 676, 2000+: 0
Device Identification	Src IP Distribution	packets	192.168.43.37 (303), 54.182.199.148 (721)	156.76.135.124 (376), 132.81.26.166 (648)
	Dest IP Distribution	packets	54.182.199.148 (303), 192.168.43.37 (721)	132.81.26.166 (376), 156.76.135.124 (648)
Routing Behavior	Average TTL	seconds	186.51	123.33
User Behavior	Number of Sessions	sessions	1	1
Error Analysis	Checksum Errors	packets	0	0
	Fragmented Packets	packets	0	0
	Fragmented IP Datagrams	datagrams	0	0

Table 11. Comparison of Real and Generated Amazon Network Traffic.

In sum, our NetDiffusion generated traffic stands out by allowing the extraction of vital metrics for additional network tasks, a capability previous works lacked due to their inability to produce protocol rule compliant, fine-grained network traffic.

5 RELATED WORK

Diffusion Models. Since the inception of the first diffusion-based model [37, 120], such models have consistently showcased great generative capacities across various domains in image synthesis [101, 107], video creation [54, 57], and structured data generation [71, 72, 152]. For instance, Stable Diffusion [107] harnesses pre-trained autoencoders for its training, expertly blending detail and simplicity for enhanced visual accuracy. DALL-E 2 [101] employs a dual-stage process that crafts realistic images from text-derived embeddings. These models often surpass GANs in detail and diversity [64, 66]. While efforts have been made to generate structured data such as tabular datasets [72], producing network traffic presents its distinct challenges due to the strict protocol constraints involved.

Network Traffic Generation. Traffic generation has been explored through various techniques ranging from simulations to GAN-driven machine learning. Traditional simulation tools like NS-3 [53], yans [75], and the modern DYNAMO [24] emulate network traffic based on different network topologies. Conversely, structure-based solutions [20, 122, 135] capture the network patterns through heuristics, and they scale better. A recent trend is GAN-based techniques, exemplified by DoppelGANger[78] and NetShare [142], which excel in encapsulating intricate temporal patterns. However, while traditional methods demand vast domain expertise and might lack versatility, GANs, though adaptive, may fall short in protocol adherence [142]. Thus these frameworks are not applicable to fine-grained traffic analysis tasks like traffic classification [62, 80], traffic management [38, 79], etc. On the other hand, there are some preliminary attempts at using diffusion models [119] for traffic generation, but their generation scope is limited as they aim to just match the overall statistics of the real data, and do not attempt to create realistic packet-level data as in this paper.

Diffusion Process Controls. Contemporary studies [148] aim to harmonize diversity and controllability in data generation by imposing conditions. ControlNet [146], for instance, integrates task-centric conditions into Stable Diffusion, utilizing inputs like edge maps. DreamBooth [109] employs unique identifiers in text-to-image frameworks, ensuring personalized and diverse outputs. Visual ChatGPT [138] and DragDiffusion [98, 117], present an interactive image modification system through language or dragging. Dall-E 3 [97] makes a leap forward in to generate images that exactly adhere to the semantic meanings of text.

6 CONCLUSION AND FUTURE WORK

Synthetic traces, primarily emphasizing certain flow statistics or packet attributes, are frequently used to support ML tasks in networking. However, their limited alignment with real traces and challenges in converting to raw network traffic hinder both their ML performance and broader applicability in conventional network analyses. In our research, we tap into the promising capabilities of diffusion models, known for their high-quality data generation, to enhance synthetic network traffic production. We present *NetDiffusion*, a tool to produce synthetic network traffic, captured as pcaps covering all packet headers. Our evaluation reveals that NetDiffusion's pcaps closely resemble authentic data and bolster ML model performance, outperforming current methods. These synthetic traces integrate with traditional network tools, support retransmission, and suit a broad range of network tasks. The rich features in NetDiffusion's outputs position it as a vital tool for diverse network analysis and testing tasks.

Looking ahead, several avenues beckon further exploration. First, transformers have shown efficacy in generating sequential data like text, suggesting their potential in network traffic generation. Key challenges include appropriate packet capture tokenization and maintaining long contexts for generating meaningful flows. *NetDiffusion* can also attempt to address the issue of long contexts by simply increasing image resolution in future work. Second, our current protocol rule-compliance approach is post-generative, given the intricate nature of managing inter-dependent constraints during the diffusion generation process. A future aspiration is to embed these rules directly within the generation pipeline, eliminating the need for subsequent adjustments. At the same time, although we leverage pretrained ControlNet models for controlled generation in this implementation of NetDiffusion, it is viable to train dedicated ControlNet models from scratch to realized finer-grained control beyond general protocol distribution, which can also serve as a potential solution for avoiding excessive post-generation adjustments. A feasible strategy for achieving this involves curating ControlNet training datasets focused on specific attributes, such as exclusively featuring TCP options. This targeted training may help effectively enforce constraints on particular aspects of the flows. Additionally, as time dependencies play a pivotal role, we aim to refine the diffusion models to directly learn and generate time series, providing a more nuanced approach to inter-packet time dependencies. Our generation's horizon is presently capped at 1,024 packets per flow sample, a limitation we seek to address, possibly through techniques like tabular diffusion that retains packet dependencies or sequential flow generation. Another intriguing prospect is building a network-specific diffusion foundation model, which could further heighten generation accuracy. Lastly, generating semantically meaningful payloads remains a challenge, with potential solutions like autoencoders offering a promising direction for future work.

Acknowledgments

We thank the anonymous associate editors and reviewers for their constructive feedback and suggestions. This research is partly supported by grants from ANR Project No ANR-21-CE94-0001 (MINT), the Fédération Informatique de Lyon through the INTERFERE project, the France and Chicago Collaborating in the Sciences program, and NSF CNS-2124393.

References

[1] 1980. User Datagram Protocol. RFC 768. https://doi.org/10.17487/RFC0768
[2] 1981. Internet Control Message Protocol. RFC 792. https://doi.org/10.17487/RFC0792
[3] 1981. Internet Protocol. RFC 791. https://doi.org/10.17487/RFC0791
[4] 1981. Transmission Control Protocol. RFC 793. https://doi.org/10.17487/RFC0793
[5] 1982. An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826. https://doi.org/10.17487/RFC0826

[6] 1999. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html Accessed: 2023.

[7] 2019. The CAIDA Anonymized Internet Traces Data. https://www.caida.org/catalog/datasets/passive_dataset_download/ Accessed: 2023.

[8] 2023. The CISCO TRex Tool. https://trex-tgn.cisco.com/ Accessed: 2023.

[9] Sebastian Abt and Harald Baier. 2014. Are we missing labels? A study of the availability of ground-truth in network security research. In *2014 third international workshop on building analysis datasets and gathering experience returns for security (badgers)*. IEEE, 40–55.

[10] Iman Akbari, Mohammad A Salahuddin, Leni Ven, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, and Stephane Tuffin. 2021. A look behind the curtain: traffic classification in an increasingly encrypted web. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 1 (2021), 1–26.

[11] Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862* (2017).

[12] Anand Balachandran, Geoffrey M Voelker, Paramvir Bahl, and P Venkat Rangan. 2002. Characterizing user behavior and network performance in a public wireless LAN. In *Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. 195–205.

[13] Adrián Barahona-Rıos and Sandra Pauletto. 2020. Synthesising knocking sound effects using conditional WaveGAN. In *17th Sound and Music Computing Conference, Online*.

[14] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. 2002. A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. 71–82.

[15] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter* 6, 1 (2004), 20–29.

[16] Jay Beale, Angela Orebaugh, and Gilbert Ramirez. 2006. *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier.

[17] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. 2006. Early application identification. In *Proceedings of the 2006 ACM CoNEXT conference*. 1–12.

[18] Andrea Bianco, Gianluca Mardente, Marco Mellia, Maurizio Munafo, and Luca Muscariello. 2005. Web user session characterization via clustering techniques. In *GLOBECOM'05. IEEE Global Telecommunications Conference, 2005.*, Vol. 2. IEEE, 6–pp.

[19] ROBERTR Boorstyn, Aaron Kershenbaum, Basil Maglaris, and Veli Sahin. 1987. Throughput analysis in multihop CSMA packet radio networks. *IEEE Transactions on Communications* 35, 3 (1987), 267–274.

[20] 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.

[21] Francesco Bronzino, Paul Schmitt, Sara Ayoubi, Hyojoon Kim, Renata Teixeira, and Nick Feamster. 2021. Traffic refinery: Cost-aware data representation for machine learning on network traffic. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3 (2021), 1–24.

[22] Francesco Bronzino, Paul Schmitt, Sara Ayoubi, Guilherme Martins, Renata Teixeira, and Nick Feamster. 2019. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 3 (2019), 1–25.

[23] Zhiyong Bu, Bin Zhou, Pengyu Cheng, Kecheng Zhang, and Zhen-Hua Ling. 2020. Encrypted network traffic classification using deep and parallel network-in-network models. *IEEE Access* 8 (2020), 132950–132959.

[24] 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.

[25] Herminio Chavez-Roman and Volodymyr Ponomaryov. 2014. Super resolution image generation using wavelet domain interpolation with edge extraction via a sparse representation. *IEEE Geoscience and remote sensing Letters* 11, 10 (2014), 1777–1781.

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

[27] Yizhou Chen, Xu-Hua Yang, Zihan Wei, Ali Asghar Heidari, Nenggan Zheng, Zhicheng Li, Huiling Chen, Haigen Hu, Qianwei Zhou, and Qiu Guan. 2022. Generative adversarial networks in medical image augmentation: A review. *Computers in Biology and Medicine* 144 (2022), 105382.

[28] Zhitang Chen, Ke He, Jian Li, and Yanhui Geng. 2017. Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks. In *2017 IEEE International conference on big data (big data)*. IEEE, 1271–1276.

[29] Phillip Chlap, Hang Min, Nym Vandenberg, Jason Dowling, Lois Holloway, and Annette Haworth. 2021. A review of medical image data augmentation techniques for deep learning applications. *Journal of Medical Imaging and Radiation Oncology* 65, 5 (2021), 545–563.

[30] Kenjiro Cho, Koushirou Mitsuya, and Akira Kato. 2000. Traffic data repository at the {WIDE} project. In *2000 USENIX Annual Technical Conference (USENIX ATC 00)*.
[31] Ana Cholakoska, Martina Shushlevska, Zdravko Todorov, and Danijela Efnusheva. 2021. Analysis of Machine Learning Classification Techniques for Anomaly Detection with NSL-KDD Data Set. In *Data Science and Intelligent Systems: Proceedings of 5th Computational Methods in Systems and Software 2021, Vol. 2*. Springer, 258–267.
[32] Jan Chorowski, Ron J Weiss, Samy Bengio, and Aäron Van Den Oord. 2019. Unsupervised speech representation learning using wavenet autoencoders. *IEEE/ACM transactions on audio, speech, and language processing* 27, 12 (2019), 2041–2053.
[33] Kimberly C Claffy. 1995. Internet traffic characterization. (1995).
[34] Benoit Claise. 2004. *Cisco systems netflow services export version 9*. Technical Report.
[35] Benoît Claise and Brian Trammell. 2013. Information Model for IP Flow Information Export (IPFIX). RFC 7012. https://doi.org/10.17487/RFC7012
[36] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. 2018. Generative adversarial networks: An overview. *IEEE signal processing magazine* 35, 1 (2018), 53–65.
[37] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. 2023. Diffusion Models in Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023). https://doi.org/10.1109/TPAMI.2023.3261988 Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.
[38] Stefany Cruz, Logan Danek, Shinan Liu, Christopher Kraemer, Zixin Wang, Nick Feamster, Danny Yuxing Huang, Yaxing Yao, and Josiah Hester. 2023. Augmented Reality's Potential for Identifying and Mitigating Home Privacy Leaks. *arXiv preprint arXiv:2301.11998* (2023).
[39] Susu Cui, Bo Jiang, Zhenzhen Cai, Zhigang Lu, Song Liu, and Jian Liu. 2019. A session-packets-based encrypted traffic classification using capsule neural networks. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 429–436.
[40] Amit Damri, Mark Last, and Niv Cohen. 2023. Towards efficient image-based representation of tabular data. *Neural Computing and Applications* (2023), 1–21.
[41] György Dán, Viktória Fodor, and Gunnar Karlsson. 2006. On the effects of the packet size distribution on the packet loss process. *Telecommunication Systems* 32 (2006), 31–53.
[42] François De Keersmaeker, Yinan Cao, Gorby Kabasele Ndonda, and Ramin Sadre. 2023. A Survey of Public IoT Datasets for Network Security Research. *IEEE Communications Surveys & Tutorials* (2023).
[43] Tcpreplay Developers. 2023. Tcpreplay. https://tcpreplay.appneta.com/.
[44] Christian Dewes, Arne Wichmann, and Anja Feldmann. 2003. An analysis of Internet chat systems. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. 51–64.
[45] Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208* (2018).
[46] Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Synthesizing audio with GANs. (2018).
[47] Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *icml*, Vol. 96. Citeseer, 148–156.
[48] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. 2018. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing* 321 (2018), 321–331.
[49] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. 2022. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10696–10706.
[50] Qiu Guan, Yizhou Chen, Zihan Wei, Ali Asghar Heidari, Haigen Hu, Xu-Hua Yang, Jianwei Zheng, Qianwei Zhou, Huiling Chen, and Feng Chen. 2022. Medical image augmentation for lesion detection using a texture-constrained multichannel progressive GAN. *Computers in Biology and Medicine* 145 (2022), 105444.
[51] Selim Gurun and Boleslaw K Szymanski. 2000. Automating Internet Routing Behavior Analysis Using Public WWW Traceroute Services. In *Managing QoS in Multimedia Networks and Services: IEEE/IFIP TC6—WG6. 4 & WG6. 6 Third International Conference on Management of Multimedia Networks and Services (MMNS'2000) September 25–28, 2000, Fortaleza, Ceará, Brazil*. Springer, 47–59.
[52] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. Ieee, 1322–1328.
[53] 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.

[54] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. 2022. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303* (2022).

[55] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[56] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. 2022. Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research* 23, 1 (2022), 2249–2281.

[57] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. 2022. Video diffusion models. *arXiv preprint arXiv:2204.03458* (2022).

[58] Jordan Holland, Paul Schmitt, Nick Feamster, and Prateek Mittal. 2021. New Directions in Automated Traffic Analysis *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 3366–3383. https://doi.org/10.1145/3460120.3484758

[59] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[60] Kyle Jamieson and Hari Balakrishnan. 2007. PPR: Partial packet recovery for wireless networks. *ACM SIGCOMM Computer Communication Review* 37, 4 (2007), 409–420.

[61] Xi Jiang and Noah Apthorpe. 2021. Automating Internet of Things network traffic collection with robotic arm interactions. *arXiv preprint arXiv:2110.00060* (2021).

[62] Xi Jiang, Shinan Liu, Saloua Naama, Francesco Bronzino, Paul Schmitt, and Nick Feamster. 2023. AC-DC: Adaptive Ensemble Classification for Network Traffic Identification. *arXiv preprint arXiv:2302.11718* (2023).

[63] Xi Jiang, Saloua Naama Shinan Liu, Francesco Bronzino, Paul Schmitt, and Nick Feamster. [n. d.]. Towards Designing Robust and Efficient Classifiers for Encrypted Traffic in the Modern Internet. ([n. d.]).

[64] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. 2023. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10124–10134.

[65] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. 2005. BLINC: multilevel traffic classification in the dark. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. 229–240.

[66] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8110–8119.

[67] S Kavitha, N Uma Maheswari, et al. 2021. Network anomaly detection for NSL-KDD dataset using deep learning. *Information Technology in Industry* 9, 2 (2021), 821–827.

[68] Anthony Kenyon, Lipika Deka, and David Elizondo. 2020. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Computers & Security* 99 (2020), 102022.

[69] Myung-Sup Kim, Young J Won, and James W Hong. 2006. Characteristic analysis of internet traffic from the perspective of flows. *Computer Communications* 29, 10 (2006), 1639–1652.

[70] Amit Klein and Benny Pinkas. 2019. From {IP}{ID} to Device {ID} and {KASLR} Bypass. In *28th USENIX Security Symposium (USENIX Security 19)*. 1063–1080.

[71] Heejoon Koo. 2023. A Survey on Generative Diffusion Models for Structured Data. *arXiv preprint arXiv:2306.04139* (2023).

[72] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2022. TabDDPM: Modelling Tabular Data with Diffusion Models. *arXiv preprint arXiv:2209.15421* (2022).

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[74] Craig Labovitz, S Iekel-Johnson, D McPherson, J Oberheide, and F Jahanian. 2010. Internet traffic and content consolidation. *77th Internet Engineering Task Force* (2010).

[75] Mathieu Lacage and Thomas R Henderson. 2006. Yet another network simulator. In *Proceedings of the 2006 Workshop on ns-3*. 12–es.

[76] Tanja Lang, Grenville Armitage, Phillip Branch, and Hwan-Yi Choo. 2003. A synthetic traffic model for Half-Life. In *Australian Telecommunications Networks & Applications Conference*, Vol. 2003.

[77] Tanja Lang, Philip Branch, and Grenville Armitage. 2004. A synthetic traffic model for Quake3. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. 233–238.

[78] 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*. 464–483.

[79] Shinan Liu, Francesco Bronzino, Paul Schmitt, Arjun Nitin Bhagoji, Nick Feamster, Hector Garcia Crespo, Timothy Coyle, and Brian Ward. 2023. LEAF: Navigating Concept Drift in Cellular Networks. *Proceedings of the ACM on Networking* 1, 2 (2023), 1–24.

[80] Shinan Liu, Tarun Mangla, Ted Shaowang, Jinjin Zhao, John Paparrizos, Sanjay Krishnan, and Nick Feamster. 2023. AMIR: Active Multimodal Interaction Recognition from Video and Network Traffic in Connected Environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 1 (2023), 1–26.

[81] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. 2023. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499* (2023).

[82] Weixing Liu, Bin Luo, and Jun Liu. 2021. Synthetic data augmentation using multiscale attention CycleGAN for aircraft detection in remote sensing images. *IEEE Geoscience and Remote Sensing Letters* 19 (2021), 1–5.

[83] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. 2020. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing* 24, 3 (2020), 1999–2012.

[84] Qianli Ma, Wei Huang, Yanliang Jin, and Jianhua Mao. 2021. Encrypted traffic classification based on traffic reconstruction. In *2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)*. IEEE, 572–576.

[85] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30. Atlanta, GA, 3.

[86] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. 2021. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 21st ACM Internet Measurement Conference*. 229–244.

[87] Matthew V Mahoney. 2003. Network traffic anomaly detection based on packet bytes. In *Proceedings of the 2003 ACM symposium on Applied computing*. 346–350.

[88] John McHugh. 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)* 3, 4 (2000), 262–294.

[89] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. 2017. ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*. 506–509.

[90] Sebastian Meister, Nantwin Möller, Jan Stüve, and Roger M Groves. 2021. Synthetic image data augmentation for fibre layup inspection processes: Techniques to enhance the data set. *Journal of Intelligent Manufacturing* 32 (2021), 1767–1789.

[91] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. 2018. Which training methods for GANs do actually converge?. In *International conference on machine learning*. PMLR, 3481–3490.

[92] Nour Moustafa, Jiankun Hu, and Jill Slay. 2019. A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications* 128 (2019), 33–55.

[93] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*. 1–6. https://doi.org/10.1109/MilCIS.2015.7348942

[94] Prashil Negandhi, Yash Trivedi, and Ramchandra Mangrulkar. 2019. Intrusion detection system using random forest on the NSL-KDD dataset. In *Emerging Research in Computing, Information, Communication and Applications: ERCICA 2018, Volume 2*. Springer, 519–531.

[95] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741* (2021).

[96] Santosh Kumar Nukavarapu, Mohammed Ayyat, and Tamer Nadeem. 2022. MirageNet-towards a GAN-based framework for synthetic network traffic generation. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 3089–3095.

[97] OpenAI. 2023. Dall E 3. https://openai.com/dall-e-3/.

[98] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. 2023. Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold. In *ACM SIGGRAPH 2023 Conference Proceedings*.

[99] Kushagra Pandey, Avideep Mukherjee, Piyush Rai, and Abhishek Kumar. 2022. Diffusevae: Efficient, controllable and high-fidelity generation from low-dimensional latents. *arXiv preprint arXiv:2201.00308* (2022).

[100] Vern Paxson. 1994. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM transactions on Networking* 2, 4 (1994), 316–336.

[101] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents, 2022. *URL https://arxiv. org/abs/2204.06125* 7 (2022).
[102] M Mazhar Rathore, Awais Ahmad, and Anand Paul. 2016. Real time intrusion detection system for ultra-high-speed big data environments. *The Journal of Supercomputing* 72 (2016), 3489–3510.
[103] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. 2017. Automated website fingerprinting through deep learning. *arXiv preprint arXiv:1708.06376* (2017).
[104] Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. 2019. Flow-based network traffic generation using generative adversarial networks. *Computers & Security* 82 (2019), 156–172.
[105] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. 2019. A survey of network-based intrusion detection data sets. *Computers & Security* 86 (2019), 147–167.
[106] R Rohith, Minal Moharir, G Shobha, et al. 2018. SCAPY-A powerful interactive packet manipulation program. In *2018 international conference on networking, embedded and wireless systems (ICNEWS)*. IEEE, 1–5.
[107] 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.
[108] Sujan Chandra Roy, Md Murad Ali, and Md Ashraful Islam. 2019. Analysis the effect of transmission cost on TTL and number of nodes in social aware routing protocols. *group* 20, 40 (2019), 60–80.
[109] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation. https://doi.org/10.48550/arXiv.2208.12242 arXiv:2208.12242 [cs].
[110] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems* 35 (2022), 36479–36494.
[111] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems* 35 (2022), 25278–25294.
[112] Vikash Sehwag, Caner Hazirbas, Albert Gordo, Firat Ozgenel, and Cristian Canton. 2022. Generating high fidelity data from low-density regions using diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11492–11501.
[113] Colleen Shannon, David Moore, and K Claffy. 2001. Characteristics of fragmented IP traffic on Internet links. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. 83–97.
[114] Tal Shapira and Yuval Shavitt. 2019. Flowpic: Encrypted internet traffic classification is as easy as image recognition. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 680–687.
[115] 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.
[116] Baoguang Shi, Xiang Bai, and Cong Yao. 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence* 39, 11 (2016), 2298–2304.
[117] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent Y. F. Tan, and Song Bai. 2023. DragDiffusion: Harnessing Diffusion Models for Interactive Point-based Image Editing. https://doi.org/10.48550/arXiv.2306.14435 arXiv:2306.14435 [cs].
[118] Hoo-Chang Shin, Neil A Tenenholtz, Jameson K Rogers, Christopher G Schwarz, Matthew L Senjem, Jeffrey L Gunter, Katherine P Andriole, and Mark Michalski. 2018. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In *Simulation and Synthesis in Medical Imaging: Third International Workshop, SASHIMI 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Proceedings 3*. Springer, 1–11.
[119] Nirhoshan Sivaroopan, Dumindu Bandara, Chamara Madarasingha, Guilluame Jourjon, Anura Jayasumana, and Kanchana Thilakarathna. 2023. NetDiffus: Network Traffic Generation by Diffusion Models through Time-Series Imaging. *arXiv preprint arXiv:2310.04429* (2023).
[120] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*. PMLR, 2256–2265.
[121] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*. IEEE, 305–316.
[122] 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.
[123] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020).

[124] Jiaming Song, Shengjia Zhao, and Stefano Ermon. 2017. A-nice-mc: Adversarial training for mcmc. *Advances in Neural Information Processing Systems* 30 (2017).

[125] Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems* 32 (2019).

[126] Jonathan Stone and Craig Partridge. 2000. When the CRC and TCP checksum disagree. *ACM SIGCOMM computer communication review* 30, 4 (2000), 309–319.

[127] Mark Sullivan and Andrew Heybey. 1998. A system for managing large databases of network traffic. In *Proceedings of USENIX*.

[128] Boyu Sun, Wenyuan Yang, Mengqi Yan, Dehao Wu, Yuesheng Zhu, and Zhiqiang Bai. 2020. An encrypted traffic classification method combining graph convolutional network and autoencoder. In *2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 1–8.

[129] Jonti Talukdar, Ayon Biswas, and Sanchit Gupta. 2018. Data augmentation on synthetic images for transfer learning using deep CNNs. In *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE, 215–219.

[130] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 1–6.

[131] Rajesh Thomas and Deepa Pavithran. 2018. A survey of intrusion detection models based on NSL-KDD data set. *2018 Fifth HCT Information Technology Trends (ITT)* (2018), 286–291.

[132] Runzhi Tian, Yongyi Mao, and Richong Zhang. 2020. Learning VAE-LDA models with rounded reparameterization trick. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1315–1325.

[133] Yves Vanaubel, Jean-Jacques Pansiot, Pascal Mérindol, and Benoit Donnet. 2013. Network fingerprinting: TTL-based router signatures. In *Proceedings of the 2013 conference on Internet measurement conference*. 369–376.

[134] ARi Vasudevan, E Harshini, and S Selvakumar. 2011. SSENet-2011: a network intrusion detection system dataset and its comparison with KDD CUP 99 dataset. In *2011 second asian himalayas international conference on internet (AH-ICI)*. IEEE, 1–5.

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

[136] Maonan Wang, Kangfeng Zheng, Dan Luo, Yanqing Yang, and Xiujuan Wang. 2020. An encrypted traffic classification framework based on convolutional neural networks and stacked autoencoders. In *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*. IEEE, 634–641.

[137] Xin Wang, Shinji Takaki, Junichi Yamagishi, Simon King, and Keiichi Tokuda. 2019. A Vector Quantized Variational Autoencoder (VQ-VAE) Autoregressive Neural *F*_0 Model for Statistical Parametric Speech Synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2019), 157–170.

[138] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671* (2023).

[139] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *Advances in neural information processing systems* 32 (2019).

[140] Shengzhe Xu, Manish Marwah, Martin Arlitt, and Naren Ramakrishnan. 2021. Stan: Synthetic network traffic generation with generative neural models. In *Deployable Machine Learning for Security Defense: Second International Workshop, MLHat 2021, Virtual Event, August 15, 2021, Proceedings 2*. Springer, 3–29.

[141] Haipeng Yao, Chong Liu, Peiying Zhang, Sheng Wu, Chunxiao Jiang, and Shui Yu. 2019. Identification of encrypted traffic through attention mechanism based long short term memory. *IEEE Transactions on Big Data* (2019).

[142] 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*. 458–472.

[143] Sebastian Zander, David Kennedy, and Grenville Armitage. 2005. Kute A high performance kernel-based udp traffic engine. (2005).

[144] Bowen Zhang, Shuyang Gu, Bo Zhang, Jianmin Bao, Dong Chen, Fang Wen, Yong Wang, and Baining Guo. 2022. Styleswin: Transformer-based gan for high-resolution image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11304–11314.

[145] Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. 2023. Text-to-image diffusion model in generative ai: A survey. *arXiv preprint arXiv:2303.07909* (2023).

[146] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3836–3847.

[147] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. 2002. On the characteristics and origins of internet flow rates. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. 309–322.

[148] Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K. Wong. 2023. Uni-ControlNet: All-in-One Control to Text-to-Image Diffusion Models. http://arxiv.org/abs/2305.16322 arXiv:2305.16322 [cs].

[149] Weiping Zheng, Jianhao Zhong, Qizhi Zhang, and Gansen Zhao. 2022. MTT: an model for encrypted network traffic classification using multi-task transformer. *Applied Intelligence* (2022), 1–16.

[150] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: data, model, and evaluation. In *European conference on computer vision*. Springer, 564–580.

[151] Yitan Zhu, Thomas Brettin, Fangfang Xia, Alexander Partin, Maulik Shukla, Hyunseung Yoo, Yvonne A Evrard, James H Doroshow, and Rick L Stevens. 2021. Converting tabular data into images for deep learning with convolutional neural networks. *Scientific reports* 11, 1 (2021), 11325.

[152] Yuanzhi Zhu, Zhaohai Li, Tianwei Wang, Mengchao He, and Cong Yao. [n. d.]. Conditional Text Image Generation With Diffusion Models. ([n. d.]).

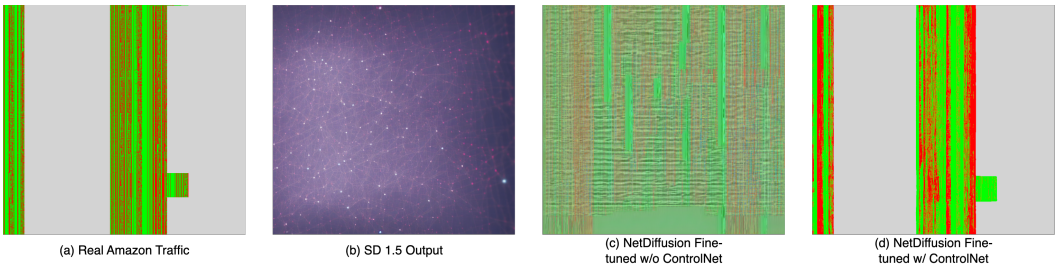


Fig. 5. Comparison of (a) real traffic with outputs from (b) non-fine-tuned SD 1.5, (c) fine-tuned NetDiffusion w/o ControlNet, and (d) fine-tuned NetDiffusion w/ ControlNet.

A Ablation Study - Fine-tuning and ControlNet

We evaluate the design choices of NetDiffusion, focusing on its fine-tuning and controlled generation aspects. Our experiments involve three configurations: (1) the non-fine-tuned Stable Diffusion 1.5 model, (2) the fine-tuned NetDiffusion model without ControlNet, and (3) the fine-tuned NetDiffusion model with ControlNet (excluding post-generation correction heuristic). All models are given the same prompt 'pixelated network traffic, type-0', representing Amazon traffic, to compare their generation outcomes. The Stable Diffusion 1.5 model used in Figure 5 is employed in its original, off-the-shelf form, without any fine-tuning. The text prompt was directly inputted into the model as is. This non-fine-tuned model fails to meet our specific generation goals, producing outputs unrelated to pcap-based network traffic when prompted, thus demonstrating the necessity of fine-tuning. The fine-tuned NetDiffusion model shows significant improvement, generating results that more closely resemble real traffic. However, due to the intrinsic characteristics of diffusion models, as discussed in Section 3.2.3, its output lacks structural fidelity, specifically in mimicing the protocol and header value distributions. In contrast, incorporating controlled generation via ControlNet in NetDiffusion achieves image representations that more accurately reflects real traffic. These findings underscore the critical role of both fine-tuning and controlled generation in our framework.

B Generation Variation Assessment

We assess NetDiffusion's generation consistency by generating multiple synthetic flow instances using identical prompts ('pixelated network traffic, type-0', representing Amazon traffic) and ControlNet inputs and compare the generated instances. For every pair of generated flows, we compared each bit in the packet headers. If a bit differs between the two packets at the same position, it is counted as a variation. For each header bit position b, the difference percentage was calculated using the formula:

$$\text{Diff}_{\%}(b) = \left(\frac{\text{Number of differing bits at position } b}{\text{Total number of packets}} \right) \times 100$$

The average difference percentage for each aggregated bit position B was then calculated by averaging across all pcap flow pairs:

$$\text{AvgDiff}_{\%}(B) = \text{Average of Diff}_{\%}(B) \text{ across all pcap flow pairs}$$

Our result reveals that the resultant synthetic traffic, exemplified by Amazon flow, exhibits an average variation of approximately 12.66%. Table 12 show cases the top 20 header fields with the highest average percentage of difference across instances of NetDiffusion generated Amazon traffic.

Received October 2023; revised January 2024; accepted January 2024

Header Field	Difference
ipv4_src	50.34%
ipv4_id	50.27%
ipv4_dst	50.20%
tcp_ackn	50.14%
ipv4_cksum	49.95%
tcp_seq	49.79%
tcp_sprt	49.28%
tcp_dprt	49.21%
ipv4_ttl	34.90%
tcp_cksum	32.36%
tcp_cwr	26.49%
tcp_wsize	22.44%
tcp_fin	21.94%
tcp_ns	21.46%
tcp_opt	20.16%
ipv4_tl	18.62%
tcp_rst	15.56%
tcp_res	14.98%
tcp_doff	14.69%
tcp_psh	13.99%

Table 12. Ranking of top 20 header fields by average percentage of difference across NetDiffusion generated Amazon traffic.