Generative, High-Fidelity Network Traces

Xi Jiang*
University of Chicago

Shinan Liu*
University of Chicago

Aaron Gember-Jacobson
Colgate University

Paul Schmitt
University of Hawaii, Manoa / Invisv

Francesco Bronzino
Univ Lyon, EnsL, UCBL, CNRS, LIP

Nick Feamster
University of Chicago

ABSTRACT

Recently, much attention has been devoted to the development of generative network traces and their potential use in supplementing real-world data for a variety of data-driven networking tasks. Yet, the utility of existing synthetic traffic approaches are limited by their low fidelity: low feature granularity, insufficient adherence to task constraints, and subpar class coverage. As effective network tasks are increasingly reliant on raw packet captures, we advocate for a paradigm shift from coarse-grained to fine-grained traffic generation compliant to constraints. We explore this path employing controllable diffusion-based methods. Our preliminary results suggest its effectiveness in generating realistic and fine-grained network traces that mirror the complexity and variety of real network traffic required for accurate service recognition. We further outline the challenges and opportunities of this approach, and discuss a research agenda towards text-to-traffic synthesis.

CCS CONCEPTS

• **Networks** → **Network simulations**; • **Computing methodologies** → **Neural networks**;

KEYWORDS

Network traffic, synthesis, diffusion model

ACM Reference Format:
Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Paul Schmitt, Francesco Bronzino, and Nick Feamster. 2023. Generative, High-Fidelity Network Traces. In *The 22nd ACM Workshop on Hot Topics in Networks (HotNets '23), November 28–29, 2023, Cambridge, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3626111.3628196

*Both authors contributed equally to this work.

HotNets '23, November 28–29, 2023, Cambridge, MA, USA

ACM ISBN 979-8-4007-0415-4/23/11…$15.00
https://doi.org/10.1145/3626111.3628196

1 INTRODUCTION

Network trace data, especially fine-grained packet captures (pcaps), is indispensable in the realm of network management. These traces are critical for a myriad of procedures, such as developing and optimizing service recognition [2, 13, 14, 24], anomaly detection [21], device identification [11, 26], and activity recognition [12, 22] systems, as well as benchmarking the performance of novel hardware and software solutions. However, securing access to such traces is often a challenge due to business confidentiality and privacy constraints.

A compelling alternative is to use synthetic trace data generated using simulations [4, 7, 19], heuristics [1, 35, 36], or machine-learning (ML) [20, 30, 37, 39]. However, existing approaches require a trade-off between data granularity and data realism. For example, simulations and heuristics can generate synthetic traces with full packet headers and payloads, but extensive domain knowledge and considerable human effort is required to produce packets whose features (e.g., size, spacing, header field values) even vaguely resemble real traces. On the flip side, current ML approaches can produce synthetic flow-level data that attempts to resemble statistical distributions in real data, but they cannot produce complete packet headers or satisfy inter-packet constraints (e.g., protocol usage patterns in flows) required for classification tasks or replay-based benchmarking/testing. Additionally, existing trace generation approaches are not easily transferable between contexts. For example, a simulator or generative model for VPN and non-VPN Netflix traffic and non-VPN YouTube traffic cannot readily produce VPN YouTube traffic.

Given the importance of fine-grained packet captures coupled with the difficulty of obtaining real(istic) traces, we formulate three essential research questions in future synthetic pcap generation to achieve the goal of high fidelity:

(1) *How can we design network traffic generators with granular data support?* We advocate for an approach that can generate high resolution raw packets to ensure a comprehensive encapsulation of real network traffic intricacies required for accurately training models and replaying traffic for testing or benchmarking.
(2) *Can we strike a balance between generation diversity and controllability?* While diversity is crucial for extensive coverage and desired randomness, the process must also respect inherent constraints, such as protocol usage patterns. This control not only ensures accuracy in ML-driven downstream tasks but also instills authenticity

in the synthetic data for network traffic, enabling for example replaying the traffic to test network functions.

(3) *How can we devise a model that guarantees expansive coverage across various types of network traces?* We envison a single model to generate pcaps that incorporate different protocols, applications, and network conditions (e.g., latency). This increases the approach's versatility and overall utility.

We investigate the limitations of state-of-the-art Generative Adversarial Networks (GAN)-based approaches in addressing these questions. We further propose a general framework for generative traffic synthesizers using a text-to-traffic approach that builds on advances in image generation to meet these requirements. Our preliminary results suggest that this approach has many avenues for improvement, and we outline a research agenda to discuss its opportunities and challenges.

2 BACKGROUND AND MOTIVATION

In this section, we motivate the need for fine-grained synthetic data generation by discussing the limitations of the status quo GAN-based approaches [20, 39]. To quantify and validate these concerns, we conduct a case study using *service recognition* as an example for downstream tasks.

2.1 Traffic Generation and Status Quo

The field of traffic generation, a recurring theme in networking research, has been scrutinized through a multitude of approaches such as traditional simulation-based, heuristics-based, and more recent GAN-based ML techniques.

Traditional simulation-based approaches, typified by tools like yans [19], NS-3 [7], and the recent DYNAMO [4], fabricate network traffic by reproducing traffic in simulated network environments. On the other hand, heuristics-based methods such as Harpoon [35], Swing [36], and the tool proposed by Botta et al. [1], spawn synthetic live traffic using distribution parameters extracted from example traffic. Simulation- and heuristics-based approaches require significant domain knowledge and human effort and may not generalize well across applications. These methods can not capture the realistic patterns especially when the underlying traffic are complex (e.g., network implementations are not always stateful) and dynamic (e.g., configurations or environments may change).

Unlike traditional methods, state-of-the-art GAN-based approaches (e.g., DoppelGANger[20] and NetShare [39]) effectively capture complex temporal correlations and long-term patterns in network datasets, and synthesize coarse-grained packet- or flow-level traces. However, these approaches struggle to capture fine-grained features and adhere to constraints [39], which we discuss in detail in the following sections.

2.2 Case Study Settings

Downstream task and datasets. Synthetic data generation can aid various network management tasks. We select service recognition, an operation crucial for functions like resource allocation and Quality of Service (QoS) assurance, as a representative task to assess the performance of existing GAN-based network data synthesizers. Specifically, we explore the potential to enhance ML-based application classification [2, 13, 22, 24] through synthetic data generation. In typical ML model training for service recognition, both training and testing are performed on real network data. This practice becomes challenging due to the difficulty in obtaining labeled real data amidst increasing encryption and Internet consolidation. Consequently, synthetic data can supplement these datasets and we aim to verify whether the model's performance remains consistent when trained on real data and tested on synthetic data, or vice versa, compared to exclusively using real data for both training and testing.

Macro Services	Total # of Flows	Micro Application Labels
Video Streaming [3]	9465	Netflix (4104) YouTube (2702) Amazon (1509) Twitch (1150)
Video Conferencing [25]	6511	MS Teams (3886) Google Meet (1313) Zoom (1312)
Social Media	3610	Facebook (1477) Twitter (1260) Instagram (873)
IoT Device [22]	3901	Other (3901)

Table 1: Service recognition dataset.

The specific task involves categorizing 4 macro-service types and 11 micro-applications, including Video Streaming, Video Conferencing, Social Media, and IoT Device usage, as detailed in Table 1. Each of these macro-services encapsulates several micro-applications, yielding a diverse, practical dataset. Our curated dataset, comprising over 30,000 flows with up to 10,000 flows per service type, is sufficiently large to facilitate effective evaluation. In training the models, we employ a conventional 80-20 training-testing split.

Baseline. We employ NetShare [39], a state-of-the-art GAN-based network synthesis approach, as the baseline. Built on the open-source tool DoppelGANger [20], NetShare captures flow metrics using a time series GAN, reformulating the traffic generation task into a time series generation problem.

2.3 GAN-based Generation

Inadequate Feature Granularity. Existing ML-based traffic generation methods [20, 30, 37, 39] are restricted by their feature granularity. Redzovic et al. [30], for instance, utilizes Hidden Markov Models to generate packet sizes and interarrival times of IP traffic, but has limited coverage of various packet features, such as the TCP window size. DoppelGANger generates few features across three example applications [20], while NetShare produces NetFlow-like data comprising only ten derived or aggregated features: source/destination IP addresses and port numbers, protocol, start time, duration, number of packets, number of bytes, and label. While NetShare also attempts to generate individual packet header fields, this

generation is limited to 5-tuple fields and 6 other selected features. These coarse-grained features often fall short in supporting high performance in downstream tasks. For instance, when conducting micro-level application classification on real traffic flows with a Random Forest (RF) model, relying solely on NetFlow features results in a performance drop (85% accuracy) compared to using raw packet bits (94% accuracy).[1]

Hence we advocate for additional and fine-grained features, such as those in raw packet captures, that transcend the limitations of NetFlow attributes or aggregated statistics.

Insufficient Generation Control. Synthetic data, while requiring a certain degree of randomness, must adhere to specific constraints to ensure downstream task performance. For instance, it is essential for synthetic network data used for service recognition model training to comply with real protocol usage patterns - such as the predominance of TCP packets in Netflix traffic and UDP packets in Teams traffic as observed in real data - to ensure that models trained on synthetic data maintain their accuracy when applied to real data. Yet, enforcing such *control* during the generation process to balance randomness with property preservation constitutes a significant technical challenge. To exemplify this, we utilize NetShare to synthesize network traces and investigate changes in accuracy using a RF model on real data and testing on synthetic data, and vice versa. The types of features present in real NetFlow data are consistent with those generated by NetShare.

Using service recognition as an example, when trained and tested on real NetFlow data, the model attains an average accuracy of 0.85 at the micro-level, as shown later in Table 2. However, the performance substantially declines when training with real NetFlow data and testing with NetShare-generated NetFlow data, showing average accuracy of only 0.056 at the micro level. The same performance decline persists when the model is trained on synthetic NetFlow data and tested on real NetFlow data, exhibiting accuracy of 0.20.

These results underscore the significance of controllability in synthetic data generation. NetShare, while useful, is inherently limited by its architecture. Currently, it does not offer support for stateful protocols, nor is it likely to autonomously learn stateful generation [39]. This architectural constraint results in a synthetic dataset that fails to demonstrate controlled dependencies between packets, thereby breaching key networking principles such as protocol usage patterns in flows. Additionally, such non-compliance significantly curtails the applicability of synthetic data, as it cannot be reliably replayed to test network functions.

Subpar Class Coverage. A significant shortcoming of GAN-based generators lies in their inability to produce traffic reflective of a wide spectrum of categories such as different protocols, applications, or network conditions. This diversity is fundamental to the efficacy of synthetic data generation tools, particularly for ML-driven downstream tasks. For example, NetShare's current generation process is not constructed

[1]For all evaluations in this study, dataset overfitting features like IP addresses, port numbers, and flow start times are removed during preprocessing.

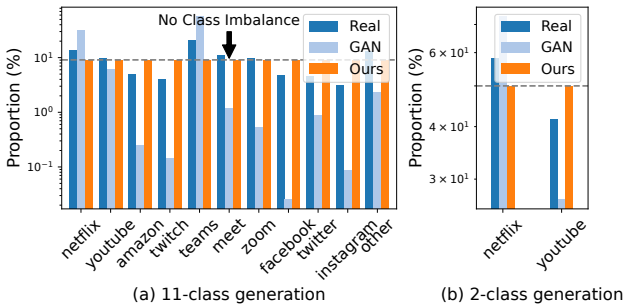

Figure 1: The distribution comparison between the real and GAN-based, and our synthetic data: our framework results in the most balanced class distribution, thereby significantly reducing class imbalance.

with classification in mind, treating the flow category attribute (denoted as 'type' in NetShare) as an additional field to generate without considering its impact on other fields' values. This means that even though the aggregate distribution similarity (or low distribution drift) may be high, it does not necessarily translate into useful data for classification tasks. As a result, the per-class results show a significant 'distribution shift' which impairs classification performance.

Moreover, a real dataset's inherent class imbalance, while mild and unlikely to significantly harm model accuracy, is amplified during GAN's generation process as it treats class labels as just another feature. This amplified class imbalance can negatively impact the model's training as illustrated in Figure 1. Our supplemental experiments indicate that even when generating traces by training a GAN-based model per class, there is negligible improvement, e.g., we still observe ~20% accuracy in micro-level classification when the model is trained on synthetic and tested on real NetFlow data. This minimal improvement is not due to aggregated metrics. Factors contributing to this include the relatively low feature support range in many useful features, which any distribution change can impact the ML inference logic when applied to real data, despite the good performance of similarity scores. Furthermore, the distribution learnt by these generators often conform to certain assumptions (e.g., normal/Gaussian distribution), which is often not the case in network traffic (e.g., port consolidation, distribution graphs).

3 TEXT-TO-TRAFFIC SYNTHESIS

To address the limitations of synthetic network data generation tools, we propose a text-to-traffic synthesis paradigm to tackle the critical research questions in Section 1. We employ Diffusion models as an example and show their potential for high fidelity through preliminary performance analysis.

3.1 System Overview

Our proposed framework is grounded in a three-tiered structure, including a text-to-image base model for high granularity, an add-on model fine-tuned for extended coverage, and a controlling component to manage inter-packet dependencies.

The first component of this framework, the text-to-image base model, is essential for capturing the fine details inherent in network traffic data. As a powerful example of such a model, we employ Stable Diffusion 1.5 [31] which sets the stage for high-resolution synthesis, thereby preserving the complexities of network traffic. The second component is an add-on model specifically tuned to expand synthesis coverage. It does so by allowing the flexible addition of new classes via word embeddings. In our framework, we use LoRa [10], which has been meticulously fine-tuned on our service recognition dataset, exemplifying its efficacy in broadening class representation. The final component of our framework is a controlling element which governs the shape and inter-packet dependencies within each class to ensure synthetic data reflect realistic protocol usage patterns in flows. ControlNet [40] serves as a strong example of this component, guiding the generation process via one-shot controls.

Traffic generation begins with fine-tuning the base model using real data, following which the model generates raw synthetic data. A real pcap file used for fine-tuning is first converted into nprint format, which is a bit-level representation of raw packet header field values. This format preserves all packet headers such as IP, TCP, UDP, and ICMP, with each bit encoded as 1 or 0 for content, and -1 for vacant bits. As shown in Figure 2, these packets are then organized into an image where each pixel row represents a packet (up to 1024 packets), comprising 1088 bit-level features. We assign pixel colors red for bits valued 1, green for 0, and grey for -1. For each image, we generate text prompts that describe its class type in an encoded format (e.g., 'Type-0' for 'Netflix') to minimize the influence of base model's original word embeddings. These images and prompts are then used to fine-tune the base model, supplemented by LoRa. During generation, a class-specific prompt and image are fed into the fine-tuned base model and ControlNet respectively, guiding the synthetic image creation. This synthetic image is then color processed to restrict it to the aforementioned distinct colors and back-transformed into nprint and finally into pcap format. This marks the completion of our detailed, controlled network trace synthesis.

Why Diffusion models? The paradigm shift toward diffusion models in computer vision stems from the impressive generative quality and diversity they offer. Since the first diffusion-based model's launch [34], these models have displayed superior generative capabilities across various tasks [6], outclassing GANs in detail and diversity [15, 16]. For example, Stable Diffusion [31] uses pre-trained autoencoders for diffusion model training, effectively balancing detail retention and complexity reduction. Similarly, Dall-E 2 [29] employs a two-stage model that generates image embeddings from text, subsequently creating diverse yet realistic images.

According to Kotelnikov et al. [18], this contributed to the perceived "victory" of Diffusion models over GANs. Unlike GANs, which often grapple with issues such as mode collapse or mode dropping, diffusion models deliver stable

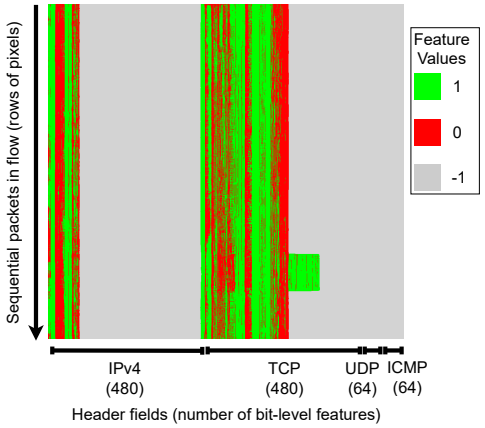

Figure 2: Color processed synthetic data for Amazon: all packets (rows of pixels) are of the protocol type TCP. -1 denotes vacant header fields.

training dynamics, demonstrating less sensitivity to hyperparameter choices, and thereby ensuring reliable, consistent outcomes[38]. Diffusion models' versatility has been demonstrated across various domains, producing realistic, high-fidelity, fine-grained datasets for complex tasks. This includes tasks like the generation of image [29, 31], video [8, 9], tabular dataset [18], and other structured data types [17].

Why ControlNet? Recent research has strived to strike a balance between diversity in generation and controllability. For instance, ControlNet [40] employs task-specific conditions, enhancing Stable Diffusion with inputs like edge maps. Uni-ControlNet [41] advances this further with a universal ControlNet addressing all conditions. DreamBooth [32] enables personalization by assigning unique identifiers to subjects in text-to-image models, aiding the creation of varied, realistic images. DragDiffusion [33], influenced by DragGAN [28], offers an interactive image editing system with guidance.

As in the context of networking, Diffusion models together with ControlNet provides a couple of notable benefits. Firstly, these models simplify generator training for class-aware data generation by leveraging word embeddings. Secondly, in conjunction with ControlNet, they support the generation of an expansive feature space. This enables the capture of intricate inter- and intra-packet relations, which translates to a richer, more detailed synthetic dataset.

3.2 Pilot Analysis

In our pilot analysis, we assess the efficacy of our fine-grained synthetic data when applied to downstream tasks, specifically on service recognition as outlined in our case study. We do this by comparing classification accuracy, which serves as an indicator of performance for service recognition, when using our synthetic data versus GAN-based data across various training and testing scenarios. To conform to later generation protocols, we trialed using the first 1024 packets of each network flow. However, fine-tuning the LoRa model on the entire dataset resulted in substantial overhead. To ensure a

Training/Testing	Data Granularity	Average Accuracy	
		Macro-level	Micro-level
Real/Real	nprint-formatted pcap	1.00	0.94
	NetFlow	0.96	0.85
Real/Synthetic (Ours)	**nprint-formatted pcap**	**0.71**	**0.40**
Real/Synthetic (GAN)	NetFlow	0.12	0.056
Synthetic/Real (Ours)	**nprint-formatted pcap**	**0.72**	**0.31**
Synthetic/Real (GAN)	NetFlow	0.42	0.20

Table 2: RF model performance across different training/testing scenarios.

fair comparison with GANs, we fine-tune the LoRa model on a smaller subset of flows (100 for each class).

Training and testing our model on real data using raw pcap data in nprint format, which our framework generates, yields average accuracies of 1.00 and 0.94 at the macro- and micro-levels respectively. The models utilizing real NetFlow data, analogous to NetShare-generated data, also deliver relatively high accuracies. These results serve as strong references for subsequent scenarios involving synthetic data. We then generate synthetic nprint-formatted pcap data using our framework and NetFlow data using NetShare, and assess the model accuracies when these synthetic datasets are employed. As detailed in Table 2, models trained on real data and tested on our synthetic data outperform those tested on NetShare-generated data by a significant margin at both macro- and micro-levels. This superiority is maintained when the models are trained on synthetic data and tested on real data.

Granularity. Our approach substantially outperforms traditional GANs by leveraging nprint representation in conjunction with standard image representation and resolution scaling. This strategy allows us to attain up to 1088 individual features (all available packet headers) for up to 1024 packets per flow for each training record. In contrast, GAN-based approaches generate tens of fields at best. It enables us to generate highly detailed synthetic data. Figure 2 shows the image representation of the processed synthetic flow in nprint representation, using Amazon traffic data as an example. This increased granularity brings two major benefits:

- *Increased performance in data-driven classification tasks.* Allowing synthetic data to encapsulate fine-grained features enhances feature representation, which can lead to improvements in data-driven downstream tasks as follows: (1) *More comprehensive data structure:* Expanding the feature set, such as including stateful protocol header fields, may allow the synthetic data to capture a broader range of features present in the actual data, thereby offering a more comprehensive and accurate representation of the underlying structure. (2) *Accurate capture of interactions and dependencies:* Increased representation enables synthetic data to more accurately capture complex cross-feature interactions and dependencies that are often observed in real-world data (e.g., temporal patterns between packets and sequential dependency of TCP/IP protocol flags). This is particularly beneficial as network traffic data distributions become increasingly homogeneous due to the widespread encryption of features (e.g., TLS encryption, QUIC/VPN tunneling). (3) *Improved downstream model complexity:* Including more features can potentially lead to more expressive models, particularly deep learning (DL) models and neural networks. These models generally benefit from extensive feature spaces, enhancing their capability to capture the subtleties and intricacies of the actual data. (4) *Mitigation of overfitting in ML-dependent tasks:* Incorporating a larger feature space can help to prevent models from overfitting to specific features (e.g., port, time to live fields) or characteristics of the synthetic data, resulting in robust models that are more generalizable to unseen data.
- *Expanded scope of downstream tasks.* Fine-grained generative traces containing all header fields from packet captures (pcap) can enable a variety of downstream tasks beyond data-driven classification. Examples of these tasks include replaying synthetic traffic for stress testing and conducting traditionally performed network analysis using tools such as WireShark [27] and DPDK [5].

Controllability. These results from Table 2 validate our approach's capacity to improve the performance of downstream tasks, in particular service recognition, over traditional GAN-based methods. The observed improvement is primarily due to the framework's controllability, realized through guided generation via ControlNet. For instance, a prominent attribute used by many service recognition models for precise classification is the dominant protocol type characterizing the flow packets. Contrasting with conventional GAN-based solutions, this diffusion-based generation framework, ensures that all packets strictly conform to the dominant protocol type typically exhibited by real data packets for each application during the traffic flow data generation process. As shown in Figure 2 which illustrates a synthetic Amazon network traffic flow in image representation, all generated packet (rows of pixels) for this particular application adheres to the TCP protocol type by filling the TCP feature pixels with colors signifying non-vacant bits, mirroring its real data counterpart. This behavior is consistent across all other application types as the synthetic traces comply with the transport protocol types seen in actual data traces, e.g., Teams using UDP. While compliance with the general protocol type is just one example of controllability in network data generation, it serves as a preliminary indicator that this framework can preserve desired constraints useful for downstream tasks.

Coverage. When it comes to coverage, diffusion-based models exhibit remarkable capabilities by virtue of its incorporation of class notions through adaptive word embeddings in prompting. This approach enables us to generate class-aware traffic data using a single model. Take synthetic data generation for service recognition as an example, during the training phase, the model associates prompt keywords, such as 'Type-0' - the encoded form of 'Netflix' - with the corresponding

network traffic patterns. During the text-to-traffic generation process, given the appropriate keywords, the model is capable of generating synthetic data that accurately mimics specific traffic styles. Therefore, to create a balanced synthetic network dataset spanning all classes or categories, we merely invoke the generation process an equal number of times for each. Additionally, we can adjust the frequency of invocation for each class or category to yield a synthetic dataset with any desired distribution. In our preliminary results, we successfully produce synthetic data representing at least 11 distinct classes with even distribution as depicted in Figure 1(a), free from the class imbalance problem often encountered with traditional GAN-based synthesizers. To demonstrate our framework's ability to maintain this balance regardless of the class and category count, we also conduct a comparative study on a two-class network dataset generation problem. Here, our approach shows similar improvement, as shown in Figure 1(b). This shows the the expansive coverage potential of our method.

4 OPEN CHALLENGES

Replayable synthetic network traces. It is challenging to generate replayable, synthetic network traces that reflect the complex patterns of real-world traffic. This shortcoming arises from the intrinsic trade-off between diversity and controllability in deep generative models. Even though we can apply a variety of generative guidance techniques like ControlNet [40], DreamBooth [32], and GroundingDino [23] to GAN-based or diffusion models to create traces with more constraints, there's still a need to further explore methods for enforcing stricter constraints such as those offered by network protocols. For instance, the existing generative models and their control schemes are tailored for computer vision or natural language processing tasks. However, these paradigms don't directly translate to network traffic analysis. In contrast to the smooth transitions between pixel boundaries commonly seen in image tasks, network traffic data doesn't follow such a pattern. Replayable traces learned correctly can potentially benefit existing pcap-based analysis tasks, including anomaly detection, traffic classification, network monitoring, user behavior analysis, and network function testing, and etc.

Generative speed. The operational efficiency of diffusion-based models, particularly in the context of inference, presents a significant hurdle [18, 38]. Despite a more manageable training process compared to GANs, diffusion models necessitate a multi-step sampling procedure during inference, extending the processing time. This becomes particularly problematic for real-time applications like network traffic generation, where the demand is for the rapid generation of tens of thousands of flows per second, especially in high-throughput settings. This situation underscores the need for optimization techniques that can expedite the inference process of diffusion models while preserving generative quality.

Dimensionality of traffic. Generating network traffic data introduces unique challenges stemming from the intrinsic structure of the data. For instance, both input and output lengths can vary, requiring a model capable of handling an inconsistent number of packets in each flow. Additionally, the high dimensionality of each packet, particularly when payloads are included, can complicate the training process and necessitate significant computational resources. Finally, network traffic flows can encompass up to tens of thousands of packets, further escalating the task's complexity. Traditional machine learning models might struggle with this sheer scale of data, underscoring the need for tailored solutions for network traffic data synthesis.

Generative foundation model beyond traffic generation. Other than generating synthetic traffic, network traffic analysis, like many other fields, can immensely benefit from the development of generative foundation models. Drawing from the success of the computer vision and natural language processing communities [6, 17, 38], we envision the creation of a similar foundation model for networking. This model would leverage self-supervised learning on a large-scale dataset of real-world raw network traces. To understand the semantic meaning of network traces, we'll need word embeddings for network protocols, service types, and more. Upon the successful establishment of such a foundation model, a host of downstream tasks could be built upon it. These tasks include, but are not limited to: (1) traffic deblurring: This involves the restoration of missing header fields or corrupted parts within network traffic; (2) network condition transfers: it entails transferring across varying network conditions such as latency, throughput, and loss rate; (3) traffic-to-traffic translations: these translations can involve complex combinations of conditions. For example, using a training set comprised of VPN traffic and non-VPN traffic for Netflix, alongside non-VPN traffic for YouTube, we could generate a predictive output of VPN traffic for YouTube; (4) discriminative tasks: foundation models could also find use in various discriminative tasks, such as taffic filtering, classification, and anomaly detection. In these instances, they could aid in the identification and classification of objects or anomalous behavior; (5) explanations: furthermore, these models can be instrumental in generating interpretable explanations for complex phenomena or processes, enhancing our understanding and control over network traffic.

5 CONCLUSION

Existing approaches for synthetic traffic generation struggle with feature granularity, task compliance, and class coverage. As network tasks depend more on raw packet-based models, we advocate for a shift from coarse-grained to fine-grained traffic generation. By exploring controllable diffusion-based techniques, we're generating realistic network traces that reflect real-world traffic diversity. Acknowledging the challenges and opportunities intrinsic to this novel approach, we propose an ambitious research agenda that aims to catapult the field of high-fidelity text-to-traffic synthesis.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Ivano Cerrato, Mauro Annarumma, and Fulvio Risso. 2014. Supporting fine-grained network functions through Intel DPDK. In *2014 Third European Workshop on Software Defined Networks*. IEEE, 1–6.

[6] 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.

[7] 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.

[8] 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).

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

[10] 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).

[11] Danny Yuxing Huang, Noah Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. 2020. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–21.

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

[13] 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).

[14] 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.]).

[15] 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.

[16] 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.

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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).

[24] 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.

[25] 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.

[26] 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.

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

[28] 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*.

[29] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* (2022).

[30] Hasan Redžović, Aleksandra Smiljanić, and Milan Bjelica. [n. d.]. IP Traffic Generator Based on Hidden Markov Models. *parameters* 1, 2 ([n. d.]), 1.

[31] 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.

[32] 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. (March 2023). https://doi.org/10.48550/arXiv.2208.12242 arXiv:2208.12242 [cs].

[33] 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. (June 2023). https://doi.org/10.48550/arXiv.2306.14435 arXiv:2306.14435 [cs].

[34] 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.

[35] 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.

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

[37] 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.

[38] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2022. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796* (2022).

[39] 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.

[40] Lvmin Zhang and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. (Feb. 2023). https://doi.org/10.48550/arXiv.2302.05543 arXiv:2302.05543 [cs].

[41] 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. (May 2023). http://arxiv.org/abs/2305.16322 arXiv:2305.16322 [cs].