Tackling Long-Term Network Trace Retention Challenges Using Deep Generative Compression

Fenghao Dong
Carnegie Mellon University
fenghaod@andrew.cmu.edu

Yucheng Yin
Carnegie Mellon University
yyin4@andrew.cmu.edu

Shinan Liu
University of Chicago
shinanliu@uchicago.edu

Vyas Sekar
Carnegie Mellon University
vsekar@andrew.cmu.edu

***Abstract*—Network packet traces are critical for security tasks which includes longitudinal traffic analysis, system testing, and future workload forecasting. However, storing these traces over extended periods is costly and subject to compliance constraints. Deep Generative Compression (DGC) offers a solution by generating inexact but structurally accurate synthetic traces that preserve essential features without storing *full* sensitive data. This paper examines key research questions on the feasibility, cost-competitiveness, and scalability of DGC for large-scale, real-world network environments. We investigate the types of applications that benefit from DGC and design a framework to reliably operate for them. Our initial evaluation indicates that DGC can be an alternative to standard storage techniques (such as gzip or sampling) while meeting regulatory needs and resource limits. We further discuss open challenges and future directions, such as improving efficiency in streaming operations, optimizing model scalability, and addressing privacy risks in this scenario.**

I. Introduction

Network packet traces play a critical role in enabling security operators to conduct a wide range of longitudinal security-related tasks. These tasks include, but are not limited to, traffic analysis for anomaly detection and network monitoring [1], [2], testing and evaluating defense systems to ensure their robustness and effectiveness [3], [4], and training machine learning models for intrusion detection, traffic classification, and adversarial resilience [5], [6], [7]. Given the increasing complexity and volume of modern network environments, it becomes essential to store longitudinal packet traces over extended time periods.

However, a major challenge lies in the cost and compliance constraints associated with storing longitudinal packet traces. The high volume [8], [9] of packet-level data generated by modern networks makes long-term storage expensive in terms of infrastructure and maintenance. Moreover, regulatory requirements, such as data retention policies and privacy laws (e.g., GDPR or HIPAA), impose limits on how long data can be stored and how it must be managed [10], [11], [12], [13], [14], [15], [16]. This creates a trade-off for security operators between retaining critical data for analysis and meeting cost and compliance obligations.

Deep Generative Compression (DGC) offers a promising solution to the storage challenges of longitudinal network traces by leveraging domain-specific deep generative models. Many use cases, such as longitudinal traffic analysis and machine learning model training, do not require exact replay of packet traces but instead benefit from structurally accurate reconstructions [17], [18], [19]. This structural capture can help bypass compliance and liability concerns associated with retaining sensitive data while still preserving key statistical and behavioral properties [19], [20]. Additionally, DGC has the potential to significantly reduce storage costs (e.g., through pretraining, parameter-efficient fine tuning, compression techniques, etc.) by generating inexact yet realistic synthetic traces that maintain utility for downstream tasks.

We envision that Deep Generative Models (DGMs) can be an alternative to state-of-the-art storage options (e.g., gzip [21], sampling [22], sketch-based methods [23], [24]). The end-to-end workflow consists of two parts: 1) *One-time offline process:* Given a series of models and specific downstream tasks, use *sampled* data to create a customized workflow that includes the selected model, its configurations, and the pre-trained base model. 2) *Online runtime system:* Utilizing the base model from the offline process, the online system will continuously fine-tune on the incoming streaming data and compress the traces periodically. Moreover, the system will proactively monitor the data/model drift and determine the right timing to fine-tune or retrain in addition to the periodic fine-tuning.

This work explores initial answers to several research questions regarding the feasibility and effectiveness of DGC for longitudinal network traces. Specifically:

- *What kinds of applications can be supported by DGC?* This includes evaluating the utility of generated traces real-world security-focus applications.
- *Is DGC cost-competitive compared to traditional storage approaches?* We aim to analyze the trade-offs in terms of storage reduction, computational overhead, and implementation costs.
- *Does DGC scale well to large network trace?* The study takes an initial assessment to the scalability of DGC methods when handling high-volume packet traces across extended time periods.

Workshop on Security Operation Center Operations and Construction (WOSOC) 2025
24 February 2025, San Diego, CA, USA
ISBN 979-8-9919276-0-4
https://dx.doi.org/10.14722/wosoc.2025.23018
www.ndss-symposium.org

Early methods like DoppelGANger [25] and NetShare [19] synthesize high-level flow statistics and packet headers, balancing fidelity and privacy. Advanced methods, such as NetDiffusion [17], [26], generate realistic packet-level traces by capturing complex intra-packet and intra-flow dependencies. Recent approaches using state space models [18] and transformer-based architectures (e.g. REaLTabFormer [27] and [28]) further improve fidelity and scalability, efficiently handling sequential dependencies without requiring extensive domain knowledge. However, one notable open challenge still remains before DGC can see widespread use. The cost-effectiveness of DGC methods, while promising, requires further analysis. Computational overhead, model training time, and deployment costs must be carefully weighed against the benefits of storage reduction, particularly in large-scale and resource-constrained settings.

Future work on DGC will focus on improving adaptability, efficiency, and scalability for real-world applications. Customizing models for specific environments and use cases will enhance fidelity across diverse network conditions. Efficient streaming operations, supported by pretraining, can enable real-time trace generation with minimal overhead. Optimizing learning processes and model storage will address resource constraints, ensuring scalability. Finally, developing runtime control systems for continuous training and evaluation will allow DGC to adapt dynamically to evolving traffic patterns, improving long-term effectiveness.

II. Motivation

We first start with some use cases of long-term packet traces, then discuss about the status quo of storage options and their pain points.

A. Use cases for longitudinal packet traces

Beyond just-in-time monitoring and analysis for anomaly detection, system health monitoring, performance analysis and provisioning, troubleshooting, security and network operators often need to store long-term network traces for more sophisticated tasks and applications that require temporal or longitudinal perspectives. We discuss a few illustrative use cases of long-term network traces.

Use case #1: longitudinal traffic analysis [29], [30], [31]. Operators interested in the statistical properties of traffic over longer timescales may collect a variety of statistical metrics on a per-epoch basis, such as the number of packets, bytes, different source-destination IP pairs, protocol distribution, and so on, in each time window. However, reconstructing long-term network traces often requires full packet traces due to their richness, flexibility, and ability to capture cross-layer correlations [19], [17]. Full traces provide detailed data, enabling retrospective metric computation and adaptable analyses, which helps to identify longitudinal trends that provide insights into the system and workflow. Additionally, these trends can help detect anomaly events when measurements deviate significantly from expected patterns.

Use case #2: system testing and evaluation. Operators rely on long-term packet traces for system testing and evaluation. Companies regularly update their systems to refine firewall rules or enhance performance. Before deploying these updates, testing is required to ensure reliability (e.g., the system can handle actual workloads without failure) and functionality. Long-term packet traces are particularly valuable for testing as they capture diverse real-world workload patterns, serving as a reliable proxy for predicting future traffic characteristics. For example, when updating an intrusion detection system (IDS), long-term packet traces enable testing to ensure that the new IDS can manage high traffic volumes over extended periods without becoming a bottleneck and can effectively detect previously observed intrusions.

Use case #3: future workload forecast. Accurately forecasting future workloads is critical for system and security operators, as it aids in optimizing system deployment and resource allocation. For instance, if network operators identify that traffic trend is behaving abnormal due to implicit network characteristic changes or user pattern shifts, they can allocate or decrease resources accordingly [2]. This often requires long-term trend patterns to establish a forecasting baseline. There are lots of forecasting techniques been developed, like Autoregressive Integrated Moving Average (ARIMA) [32], [33], [34], [35], Seasonal Decomposition of Time Series (STL) [36], Vector Autoregression (VAR) [37], [38], RNN/LSTM [39], [40], [32], CNN [41], Autoencoder-based [42], and transformer-based [43], [44], [45], [46], [47].

B. Challenges with Status Quo

Operators aiming to store long-term traces primarily face two concerns:

- **Storage Cost:** The first concern is the cost of storing long-term traces. With the increasing volume of network traffic, modern networks generate a significant amount of trace data [8]. Even small-scale university networks produce an average of 1 to 2 TB of traffic per day [48], [49], while data center networks and ISPs generate over 10 TB of daily traffic [50], [51], [52]. For example, an ISP generating 10 TB of traces daily would incur substantial storage expenses. Using standard storage pricing on Google Cloud in North America, at $0.02 per GB per month [53], storing the traces for a year would cost $2457.6 for a single day's data and $897K for a year's data. Such expenses represent a significant financial burden.
- **Policy Compliance:** The second concern relates to compliance with retention and privacy policies. Many networks have restrictions that limit the duration for which raw traces can be stored, often due to privacy regulations (e.g., GDPR or HIPAA) or data governance policies [10], [11], [12], [13], [14], [15], [16].

To address the high storage cost, operators commonly employ compression techniques such as gzip to reduce the storage footprint of trace data. Our experiments indicate that gzip can typically achieve a storage reduction of $50\% \sim 80\%$ compared

to raw traces. However, for further storage reduction, operators often resort to trace sampling, storing only a subset of the data. While sampling can save storage space, it is not suitable for many use cases due to its inherent limitations. For instance, in long-term trend analysis, sampled traces can only provide limited statistical insights. If packets are randomly sampled at a 10% rate, the total number of bytes in a minute can be roughly estimated as ten times the sampled value. However, more complex metrics, such as the number of distinct source IP addresses in a minute, cannot be accurately extrapolated in the same way, as they do not scale linearly with the sample rate. Similarly, for stress testing updated systems, sampled traces are inadequate, as they fail to replicate the actual workload's stress, thereby undermining the test's validity.

To comply with policies and legal requirements, a common approach taken by operators is the anonymization of trace data [54], often achieved by techniques such as hashing to anonymize IP fields in packet headers. However, the extent of privacy provided by such methods remains uncertain, as numerous de-anonymization attacks have been developed to exploit vulnerabilities in these techniques [55], [56].

C. Our Goal

Based on the challenges and pain points faced by operators, our objective is to design and implement a compression framework that

- **Capable of Online Processing.** Our system is designed to operate in an online setting, where packets arrive continuously every second. It should efficiently handle high volumes of incoming packets and compress them in real time.
- **Easy to deploy.** Our system should run seamlessly and achieve optimal compression results out of the box. It must be fully automated, requiring no prior knowledge or manual configuration from system or security operators.
- **Cost-efficient.** We want out system to achieve a higher compression ratio and lower total cost compared to general-purpose approaches like gzip.
- **High data quality.** Our system will aim to maintain balanced fidelity-privacy trade-off, ensuring that the compressed data is suitable for a wide range of downstream applications without compromising functionality or analytical accuracy.

III. Research Vision

A. Deep Generative Opportunity

Deep generative models (DGMs) are a class of neural networks designed to learn complex data distributions and generate realistic samples. Prominent architectures include Variational Autoencoders (VAEs) [57] and Generative Adversarial Networks (GANs) [58]. More recently, transformer-based generative models like GPT and its variants [59] have emerged. DGMs have found widespread applications in data synthesis and compression. By capturing intricate patterns in high-dimensional data, they provide powerful tools for modeling and efficiently encoding structured data, which makes them powerful tools to generate network packet traces. There are lots of previous work using DGMs to generate network traces, including DoppelGANger [25], NetShare [19], NetDiffusion [17], [26], etc.

Additionally, DGMs have potential to compress data, since it can find some patterns in raw data very efficiently. For example, GPT-3 [59], one of the largest transformer model, is trained from over 45TB of text data, but it captures the structure and information of the extremely large training data in only 350GB of parameters. Therefore, DGM would also be able to compress network traces efficiently.

One important property of DGMs is they only capture the structure, distribution and statistical information of original training data, without trying to exactly mimic them. This property is suitable for network packet trace storage, since most of the long-term packet trace applications do not need the exact replay of previous packets. For instance, if the operators want to do some long-term trend analysis, the generated trace would suffice if it preserves the statistical properties of the actual trace. Additionally, the non-replay property enables DGM to provide good privacy protection and compliance to the policy and legal requirements.

Therefore, due to their efficient compression capability and good fit to the long-term packet traces application and policy requirements, we try to explore the capability of deep generative compression for network packet traces, and build a practical compression system using the DGMs.

B. System Overview

Following our design goals, the system design, illustrated in Figure 1, consists of two distinct phases: the offline warm-up phase and the online compression phase. Additionally, a model library is maintained to store all trained models to date.

Phase 1: Offline Warm-up. This offline phase prepares the system for its online operation. It processes historical packet traces as input to train a DGM, which serves as the base model (①). The trained model is then added to the model library (②). Once the initial model library is established, the system is ready for online operation.

Phase 2: Online Compression. During online operation, packets arrive sequentially. Our system partitions these packets into chunks, where each chunk consists of a group of adjacent packets. For each chunk, the system aims to construct a DGM capable of generating the corresponding network traces. Specifically, the system first selects a previously trained model from the model library (③). It then fine-tunes this selected model on the current chunk (④) to produce a model tailored to the specific packet trace of that chunk. Finally, the updated model is added back to the model library (⑤).

C. Technical Challenges

Developing a compression system that meets the needs of operators involves several technical hurdles:

Fidelity modeling sufficiency: The system must preserve sufficient fidelity for downstream applications. Different use cases demand varying levels of detail, such as precise packet

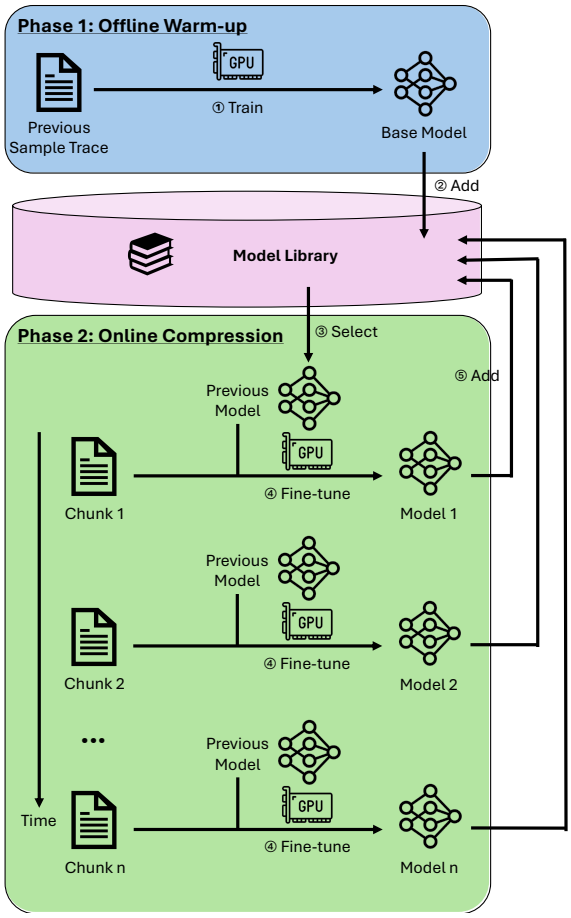

Fig. 1. Streaming system design.

arrival times, flow dynamics, or specific header fields. While the literature evaluates fidelity using different statistical measures or downstream ML model performance [19], [17], [25], they fail to capture some of the above domain-specific information while balancing compression levels.

Coverage over use cases: A unified deep generative compression framework need to accommodate a variety of tasks that extend beyond immediate monitoring and anomaly detection. For instance, long-term traffic analysis requires faithful reconstructions of diverse network metrics. However, system testing and evaluation need realistic, large-scale packet captures to ensure compatability with networked hardware [26]. Each scenario poses unique data requirements, from packet-level details to aggregated metrics, making it challenging to design a single framework without frequent reconfiguration. Meeting these demands requires a clear understanding of application context.

Computation-storage cost trade-off: Training deep generative models for compression introduces computational expenses, especially when applied to large-scale datasets. These models require high-performance hardware and time [27], [17], [25], [19]. The increased computational cost must be justified by a corresponding reduction in storage costs achieved through higher compression ratios, while ensuring minimal computational overhead to maintain practicality and cost-effectiveness for large-scale deployments.

Runtime efficiency: The design of a DGC framework should be efficient to avoid computational bottlenecks in a streaming system. High overhead or latency can undermine usability, especially in real-time environments. DGMs must carefully balance the chunking size of incoming data for each training iteration to optimize this trade-off effectively.

Modeling privacy and risk: The traces produced by DGMs are not inherently privacy-preserving, as these models are susceptible to various attacks, including model inversion attacks [60] and membership inference attacks [61]. While some initial efforts have been made to enhance the privacy of DGMs in network trace generation [19], [20], the broader privacy implications and protections for long-term network traces remain largely under-explored. This highlights the need for further research to establish robust privacy-preserving mechanisms for temporal and longitudinal information under different application context.

IV. Preliminary Results and Future Work

A. Early results

Experiment setup. For our early evaluation, we use the CAIDA [62], an anonymous network packet traces collected from the Equinix-Chicago monitor in 2018, for the task of trace storage and compression. The experiments were conducted using a single NVIDIA V100 GPU and leveraged a state-of-the-art tabular-based network transformer model, REalTabFormer [27] with a GPT-2 model of 3 transformer layers, 4 attentio heads and embedding dimension of 128, for the evaluation of the generated traces against real-world data.

Domain-specific fidelity. The generated network traces in Table I are evaluated using the Jensen-Shannon Divergence (JSD) across a collection of network-specific metrics that reflect packet and flow characteristics. The results show that a training size of 500K and 1M packets yields an average JSD of 0.28 across 17 metrics, which remains steady at larger training sizes. Metrics such as packet size and throughput exhibit relatively low JSD values (0.12 and 0.09 for 500K, respectively), while flow-level properties, including flow size and flow duration, show higher divergence (0.38 and 0.42 for 500K). Notably, the JSD for flow start time (0.12 for 500K) indicates that the model captures some basic temporal information effectively. However, the results for flow size and flow duration suggest that the REalTabFormer [27] architecture does not sufficiently condition on these aspects, highlighting potential areas for improvement in modeling temporal dependencies and flow-level dynamics. Compared to prior studies that often rely on broader statistical measures or downstream machine learning performance, this analysis emphasizes the value of incorporating domain-specific metrics when assessing the fidelity of synthetic traces.

Training scalability. The relationship between the computation cost, expressed as seconds per thousand packets (SPTP), and fidelity is shown in the plot. SPTP serves as a proxy for computational cost, directly indicating the time spent on GPU

TABLE I
DISTANCES (IN JSD) OF NETWORK-SPECIFIC METRICS BETWEEN REAL VS. GENERATED NETWORK TRACES.

Train. Size (#Packet)	Packet Size	Throughput	Flow Size	Flow Start Time	Flow Duration	Average of 17 Metrics
500K	0.12	0.09	0.38	0.12	0.42	0.28
1M	0.10	0.10	0.40	0.13	0.43	0.28

training time per thousand packets. As shown in Figure 2, the results reveal that as the training set size increases, the model achieves higher efficiency, with lower average JSD values at comparable SPTP levels. Notably, *smaller datasets (e.g., 50K and 100K packets) exhibit slower convergence, requiring more computation to reach fidelity levels similar to larger datasets.* For the 500K and 1M packet training sets, the model achieves comparable fidelity (approximately 0.3 average JSD) after around 20 seconds of computation, indicating that additional computation time provides diminishing returns beyond this point. This convergence suggests a computational bound at higher dataset sizes, where increased data availability does not significantly enhance fidelity metrics after a certain threshold. These findings highlight the scalability of the model, where larger training sets allow for faster and more efficient fidelity improvements, up to the observed computational limits.

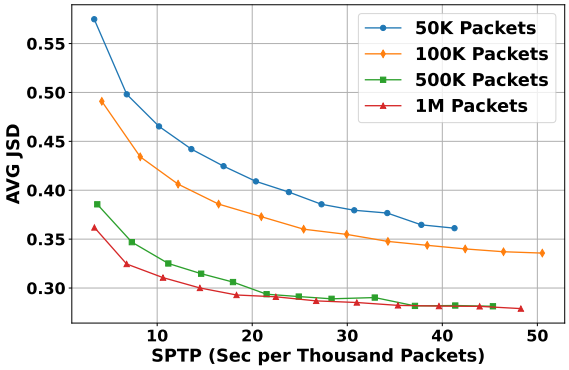

Fig. 2. Average JSD v.s. SPTP. X axis represents the average GPU second used by per thousand packets. Y axis represents the average of JSD on 17 different metrics between generated trace and original trace. Different color of line represents different different number of packets in the original trace.

Storage. We evaluate the storage consumption of storing raw CSV, using gzip and DGM. The result is shown in Figure 3. Notice that the storage cost grows linearly as number of packets stored in both raw CSV and gzip-compressed CSV, but the model size 3.19 MB remains constant since we are using the same model configuration for different size of packets to be compressed. This implies that as the dataset scales, the compression efficiency of gzip diminishes, and when the number of packets larger than 200K, using DGM can out beat gzip on the storage cost. The constant model size demonstrates the compactness of the trained model, which provides a significant advantage in terms of long-term storage efficiency for large-scale datasets. In the case of 1M packets, the raw trace takes 81.87 MB storage and the gzip-compressed trace takes 17.79

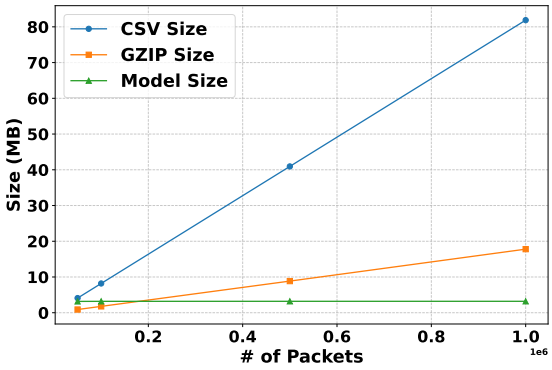

Fig. 3. Storage consumption v.s. number of packets in the trace. X axis represents number of packets in the trace, Y axis represents the amount of storage space each approach consumes. Blue line represents storing the raw trace; Orange line represents using gzip to compress the trace, and the green line represents using DGM.

MB, which means using DGM can save 96% of storage space compared to raw trace and 82% of storage space compared to gzip compression. This observation emphasizes the potential of significantly reducing storage cost using DGMs, especially for long-term retention of large datasets.

B. Future Directions

This work sheds light on a series of promising future directions:

- **Model Selection and Hyperparameter Tuning.** With the rapid expansion of deep generative models (DGMs), selecting and tuning the right model has become a critical challenge. Balancing factors such as fidelity, cost, and privacy requires careful experimentation. Additionally, since DGMs are highly sensitive to hyperparameters, integrating the system with auto-tuning frameworks (e.g., AutoGluon [63]) can enhance adaptability across diverse use cases and workloads.
- **Model System Optimizations (one time)**: We envision that lots of out-of-the-box DGMs are not fully optimized in terms of fidelity and cost. As a one-time offline process, profiling and optimizing the system bottlenecks can help improve the fidelity and bring down the cost at the same time.
- **Pretraining and finetuning (runtime)**: Pretraining [59], [64] has proven to be the standard fashion for today's deep neural network. However, while some early efforts have been made in developing foundational models in the networking domain [8], [65], [66], developing a

reasonably sized, high-quality base model and exploring fine-tuning approaches (e.g., LoRA [67]) can significantly advance networking applications.

- **Continuous runtime profiling and adaptation (runtime):** Data and model drift [2] can degrade the model performance in production environments. Continuous runtime monitoring and adaptation through fine-tuning or retraining can help ensure the fidelity of DGMs.

ACKNOWLEDGMENT

This work was supported in part by NSF grant RINGS: Enabling Data-Driven Innovation for Next-Generation Networks Via Synthetic Data (#2148359). This work used Bridges-2 at Pittsburgh Supercomputing Center through allocation CIS230362 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

REFERENCES

[1] L. F. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Science International: Digital Investigation*, vol. 32, p. 200892, 2020.

[2] S. Liu, F. Bronzino, P. Schmitt, A. N. Bhagoji, N. Feamster, H. G. Crespo, T. Coyle, and B. Ward, "Leaf: Navigating concept drift in cellular networks," *Proceedings of the ACM on Networking*, vol. 1, no. CoNEXT2, pp. 1–24, 2023.

[3] H. Shiravi, A. Shiravi, and A. A. Ghorbani, "A survey of visualization systems for network security," *IEEE Transactions on visualization and computer graphics*, vol. 18, no. 8, pp. 1313–1329, 2011.

[4] M. S. Ahmed, E. Al-Shaer, and L. Khan, "A novel quantitative approach for measuring network security," in *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE, 2008, pp. 1957–1965.

[5] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE access*, vol. 8, pp. 222 310–222 354, 2020.

[6] M. Khan and L. Ghafoor, "Adversarial machine learning in the context of network security: Challenges and solutions," *Journal of Computational Intelligence and Robotics*, vol. 4, no. 1, pp. 51–63, 2024.

[7] A. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville, "Ai/ml for network security: The emperor has no clothes," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1537–1551.

[8] F. Le, M. Srivatsa, R. Ganti, and V. Sekar, "Rethinking data-driven networking with foundation models: challenges and opportunities," in *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, 2022, pp. 188–197.

[9] S. Liu, T. Shaowang, G. Wan, J. Chae, J. Marques, S. Krishnan, and N. Feamster, "Serveflow: A fast-slow model architecture for network traffic analysis," *arXiv preprint arXiv:2402.03694*, 2024.

[10] *Cisco EPN Manager User and Administrator Guide, Version 4.0*, Cisco Systems, 2023, accessed: 2024-12-20. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/net_mgmt/epn_manager/4_0_0/user/guide/bk_CiscoEPNManager_4_0_UserAndAdministratorGuide/bk_CiscoEPNManager_4_0_UserAndAdministratorGuide_chapter_010111.pdf

[11] "Data retention periods," Dynatrace, 2023, accessed: 2024-12-20. [Online]. Available: https://www.dynatrace.com/support/help/manage/data-privacy-and-security/data-privacy/data-retention-periods

[12] "Extended trace retention," Splunk, 2023, accessed: 2024-12-20. [Online]. Available: https://docs.splunk.com/Observability/apm/apm-spans-traces/extended-trace-retention.html

[13] "Quotas and limits," Google Cloud, 2023, accessed: 2024-12-20. [Online]. Available: https://cloud.google.com/trace/docs/quotas

[14] "Telemetry retention policy for trace," Microsoft, 2023, accessed: 2024-12-20. [Online]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/administration/telemetry-retention-policy-trace

[15] "Trace retention and ingestion controls," Datadog, 2023, accessed: 2024-12-20. [Online]. Available: https://docs.datadoghq.com/tracing/trace_pipeline/trace_retention/

[16] "Understand data retention in lightstep," Lightstep, 2023, accessed: 2024-12-20. [Online]. Available: https://docs.lightstep.com/docs/understand-data-retention-in-lightstep

[17] X. Jiang, S. Liu, A. Gember-Jacobson, A. N. Bhagoji, P. Schmitt, F. Bronzino, and N. Feamster, "Netdiffusion: Network data augmentation through protocol-constrained traffic generation," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 8, no. 1, pp. 1–32, 2024.

[18] A. Chu, X. Jiang, S. Liu, A. Bhagoji, F. Bronzino, P. Schmitt, and N. Feamster, "Feasibility of state space models for network traffic generation," in *Proceedings of the 2024 SIGCOMM Workshop on Networks for AI Computing*, 2024, pp. 9–17.

[19] Y. Yin, Z. Lin, M. Jin, G. Fanti, and V. Sekar, "Practical gan-based synthetic ip header trace generation using netshare," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 458–472.

[20] Z. Lin, V. Sekar, and G. Fanti, " On the Privacy Properties of GAN-generated Samples," in *AISTATS*, 2021.

[21] J. loup Gailly and M. Adler, "gzip home page," https://www.gzip.org/, 1996, accessed: 2024-12-22.

[22] B. Claise, "Cisco systems netflow services export version 9," RFC 3954, https://www.rfc-editor.org/rfc/rfc3954, 2004, accessed: 2024-12-22.

[23] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[24] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One sketch to rule them all: Rethinking network flow monitoring with univmon," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 101–114.

[25] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar, "Using gans for sharing networked time series data: Challenges, initial promise, and open questions," in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 464–483.

[26] X. Jiang, S. Liu, A. Gember-Jacobson, P. Schmitt, F. Bronzino, and N. Feamster, "Generative, high-fidelity network traces," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, 2023, pp. 131–138.

[27] A. V. Solatorio and O. Dupriez, "Realtabformer: Generating realistic relational and tabular data using transformers," *arXiv preprint arXiv:2302.02041*, 2023.

[28] Z. J. Kong, N. Hu, Y. C. Hu, J. Meng, and Y. Koral, "High-fidelity cellular network control-plane traffic generation without domain knowledge," in *Proceedings of the 2024 ACM on Internet Measurement Conference*, 2024, pp. 530–544.

[29] M. Fomenkov, K. Keys, D. Moore, and K. Claffy, "Longitudinal study of internet traffic in 1998-2003," in *Proceedings of the winter international synposium on Information and communication technologies*, 2004, pp. 1–6.

[30] R. Fontugne, P. Abry, K. Fukuda, D. Veitch, K. Cho, P. Borgnat, and H. Wendt, "Scaling in internet traffic: a 14 year and 3 day longitudinal study, with multiscale analyses and random projections," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2152–2165, 2017.

[31] M. Hoogesteger, R. de Oliveira Schmidt, A. Sperotto, and A. Pras, "Reports on internet traffic statistics," in *TERENA Networking Conference, TNC 2013*. Trans-European Research and Education Networking Association, 2013, p. 19.

[32] R. Madan and P. S. Mangipudi, "Predicting computer network traffic: a time series forecasting approach using dwt, arima and rnn," in *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE, 2018, pp. 1–5.

[33] H. Yang, X. Li, W. Qiang, Y. Zhao, W. Zhang, and C. Tang, "A network traffic forecasting method based on sa optimized arima–bp neural network," *Computer Networks*, vol. 193, p. 108102, 2021.

[34] Y. Yu, J. Wang, M. Song, and J. Song, "Network traffic prediction and result analysis based on seasonal arima and correlation coefficient," in *2010 International Conference on Intelligent System Design and Engineering Application*, vol. 1. IEEE, 2010, pp. 980–983.

[35] B. Zhou, D. He, Z. Sun, and W. H. Ng, "Network traffic modeling and prediction with arima/garch," in *Proc. of HET-NETs Conference*, 2005, pp. 1–10.

[36] H. Zhang, X. Wang, J. Cao, M. Tang, and Y. Guo, "A multivariate short-term traffic flow forecasting method based on wavelet analysis and seasonal time series," *Applied Intelligence*, vol. 48, pp. 3827–3838, 2018.

[37] R. Alfred *et al.*, "Performance of modeling time series using nonlinear autoregressive with exogenous input (narx) in the network traffic forecasting," in *2015 International Conference on Science in Information Technology (ICSITech)*. IEEE, 2015, pp. 164–168.

[38] D. Ergenç and E. Onur, "On network traffic forecasting using autoregressive models," *arXiv preprint arXiv:1912.12220*, 2019.

[39] A. Azzouni and G. Pujolle, "A long short-term memory recurrent neural network framework for network traffic matrix prediction," *arXiv preprint arXiv:1705.05690*, 2017.

[40] A. Dalgkitsis, M. Louta, and G. T. Karetsos, "Traffic forecasting in cellular networks using the lstm rnn," in *Proceedings of the 22nd Pan-Hellenic conference on informatics*, 2018, pp. 28–33.

[41] C.-W. Huang, C.-T. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)*. IEEE, 2017, pp. 1–6.

[42] X. Gong, T. Ma, and C. Antoniou, "Network traffic dynamics prediction with a hybrid approach: Autoencoder-var," in *2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, 2021, pp. 1–6.

[43] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in neural information processing systems*, vol. 32, 2019.

[44] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 9, 2023, pp. 11 121–11 128.

[45] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *The eleventh international conference on learning representations*, 2023.

[46] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[47] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International conference on machine learning*. PMLR, 2022, pp. 27 268–27 286.

[48] V. d. C. Estrada, "Analysis of anomalies in the internet traffic observed at the campus network gateway," *arXiv preprint arXiv:1706.03206*, 2017.

[49] P. Jurkiewicz, G. Rzym, and P. Boryło, "Flow length and size distributions in campus internet traffic," *Computer Communications*, vol. 167, pp. 15–30, 2021.

[50] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267–280.

[51] M. Trevisan, D. Giordano, I. Drago, M. Mellia, and M. Munafo, "Five years at the edge: Watching internet from the isp network," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, 2018, pp. 1–12.

[52] M. Trevisan, "Big data for traffic monitoring and management," *arXiv preprint arXiv:1902.11095*, 2019.

[53] "Google cloud storage pricing," Google Cloud, 2024, accessed: 2024-12-20. [Online]. Available: https://cloud.google.com/storage/pricing

[54] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 29–38, 2006.

[55] B. F. Ribeiro, W. Chen, G. Miklau, and D. F. Towsley, "Analyzing privacy in enterprise packet trace anonymization." in *NDSS*, 2008.

[56] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner, "The role of network trace anonymization under attack," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 5–11, 2010.

[57] D. P. Kingma, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[58] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[59] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[60] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 253–261.

[61] M. Zhang, N. Yu, R. Wen, M. Backes, and Y. Zhang, "Generated distributions are all you need for membership inference attacks against generative models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 4839–4849.

[62] "Anonymized Internet Traces 2018," 2018. [Online]. Available: https://catalog.caida.org/dataset/passive_2018_pcap

[63] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "Autogluon-tabular: Robust and accurate automl for structured data," *arXiv preprint arXiv:2003.06505*, 2020.

[64] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[65] X. Meng, C. Lin, Y. Wang, and Y. Zhang, "Netgpt: Generative pretrained transformer for network traffic," *arXiv preprint arXiv:2304.09513*, 2023.

[66] S. Guthula, R. Beltiukov, N. Battula, W. Guo, and A. Gupta, "netfound: Foundation model for network security," *arXiv preprint arXiv:2310.17025*, 2023.

[67] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.