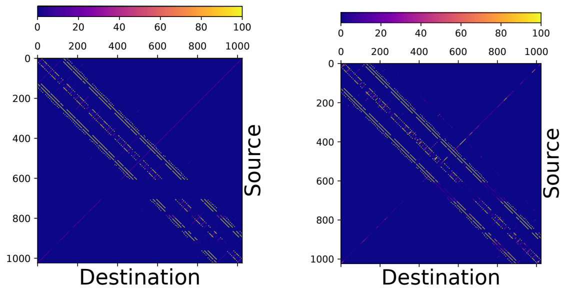
Effectively Mimicking Datacenter Traffic Patterns Using Transformers

Chen Griner
School of Electrical and Computer Engineering
Ben Gurion University of the Negev, Israel
griner@post.bgu.ac.il

***Abstract*—Large language models (LLMs) learn patterns that enable them to generalize across diverse tasks. Given their nature as sequence predictors, can they also learn patterns from datacenter network traces? In this work, we introduce DTG-GPT, a packet-level Datacenter Traffic Generator (DTG), based on the generative pre-trained transformer (GPT) architecture used by state-of-the-art LLMs. We train our model on a small set of available traffic traces and offer a simple methodology to evaluate the fidelity of the generated traces to their original counterparts. We show that DTG-GPT can synthesize novel traces that mimic the spatiotemporal patterns found in real traffic traces. Our findings indicate the potential that, in the future, similar models to DTG-GPT will allow datacenter operators to release traffic information to the research community via trained GPT models.**

I. Introduction

The advent of LLMs based on the generative pre-trained transformer (GPT) architecture has shown that these models can generalize from large corpora of textual information to perform various tasks. These tasks are achieved seamlessly on the same generalized model. To realize this, the model predicts the next word in a sequence of words, which act as context. The model builds the sequences based on the context and patterns inferred from its training data [1]. The underlying task GPT-based LLMs perform is sequence prediction, but are these models only effective for sequences that represent natural languages? Recently, in a paper by Dietmüller et-al the authors have proposed a future in which different and varied networking tasks such as traffic optimization, congestion control, and many others can all be performed using a single generalized model, which they denoted as Network Traffic Transformer or *NTT* [2]. The contribution of such a model to researchers and network operators would be great, but many challenges remain. It is unclear how transformer-based models can be adapted to the complexities and variety found in packet traces and whether these models can effectively capture patterns in network traffic. In this paper we ask these questions and attempt to advance the state-of-the-art towards NTT. We explore if a GPT-based model can step outside natural language processing and mimic the patterns of real-world traces representing traffic from datacenter networks (DCNs). Today, traffic traces are usually available only to the network operator and not to researchers [3]. The lack of traces, except for a few examples, challenges the progress of novel datacenter network designs [4], [3], [5]. This lack is often caused by privacy concerns. Indeed, even simple anonymization of a trace may not be enough to release it [6]. However, trained AI models may relieve these concerns [7], [2]



(a) Original (b) Generated

Fig. 1: Two traffic matrices of $8M$ packets. Where (a) is a real trace from an HPC cluster (NekBone) [11] and (b) a counterpart generated with DTG-GPT. The maximal element was clipped to 100 packets to enhance contrast.

This paper presents DTG-GPT, a novel DCN traffic trace generation model based on the GPT architecture used by large language models such as OpenAI Chat-GPT [1]. We make a few changes to the embedding layer of an LLM to making better suited for trace generation, and take an information-theoretic approach based on traces complexity [8], to examine if DTG-GPT mimics the traffic patterns found in realistic traffic traces. In this work, we will not try to replicate entire packets; instead, this work focuses on a small but crucial part of the DCN packet trace: the source and destination IP columns. Our trace is a sequence of pairs (s, d) of IDs representing a packet's source and destination nodes. This follows an example set in previous works looking at DCN traces for DCN development such as in [8], [9], [10] and others.

Creating a model that can generalize the traffic patterns found in all datacenters would require a massive 'corpus' of traces, which is not currently available. However, as a first step, given the few existing realistic traffic traces, we would like our DTG-GPT, to generate more traces that are similar to the original in a funda-

Annex to ISBN ISBN 978-3-903176-72-0 © 2025 IFIP

mental way. Conversely, we would not like the model to copy directly from the original. Essentially, we want the model to learn to replicate general 'patterns' found in the original, however, unlike for natural languages, it is not fully clear what those patterns are. For now, to judge the fidelity of traces generated with DTG-GPT, we compare them to the original. However, comparing two traces is not straightforward; there could be many ways to measure and compare traffic [3]. We begin with a simple approach: we look at *traffic matrices*. These matrices describe the accumulated traffic between each source and destination in the network, as seen in the trace. This allows us to observe the *spatial* structure that arises from the frequency of all source-destination pairs. We illustrate our intention with the following example.

Figure 1 shows two traffic matrices. Figure 1 (a) shows the total traffic matrix from $8M$ request long trace (denoted as 'NekBone'), and Figure 1 (b) shows a synthetic counterpart trace of $8M$ packets generated using our DTG-GPT model. A clear pattern emerges: three main diagonals go from left to right, and each diagonal is broadly composed of three smaller diagonals. There are also other features, such as an empty gap in the diagonals lacking traffic between sources 600 and 700 and a smaller diagonal going from right to left. The reader may look closer at the two matrices and find many smaller similarities and, crucially, some differences. However, looking at different structures or patterns in the traffic matrix can only give us a partial indication of similarity; it does not consider a trace's temporal properties. Indeed, It is possible to generate a trace with a very similar traffic matrix to Figure 1 (a) by sampling from a probability distribution based on the frequency of all pairs in the traffic matrix. We therefore attempt to show that our DTG-GPT generates not only traces with similar spatial properties but also temporal ones, and it does so while generating novel sequences based on the original traces. We are unaware of any work that generates and evaluates realistic DCN traces using the GPT architecture. The technical details of modifications made to DTG-GPT, the parameters used and our complete methodology will appear in the full version of the paper.

II. Evaluation

This section analyzes four HPC traces [11] (taken from [12]), denoted as Nekbone, MOCFE, CNS, and MultiGrid. We take an information-theoretic approach to estimate fidelity and analyze our traces' *trace complexity*. We then look into the novelty of generated traces by testing the ratio of novel n-grams in each trace. Here, all traces were cut to be at most $20M$ request long, and each represents a network with 1024 different nodes. We present a summary of our evaluation here, in the full version of the paper we expand our analysis with more traces, examine how our model recreates all the measures discussed here, its ability to recreate more patterns, compare our model to other trace generation discuss how to generalized towards a more extensive packet header and describe all technical details.

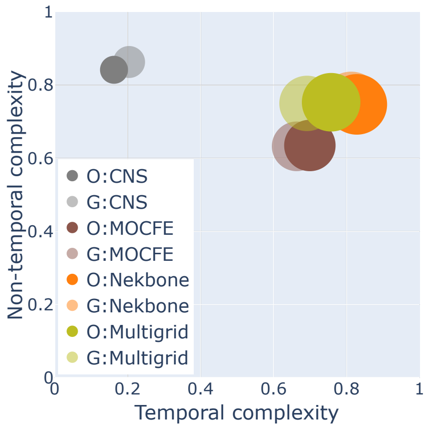


Fig. 2: The complexity map of four from the HPC set real traces and their generated counterparts. An opaque circle designates original (O) traces, while generated (G) traces have a more transparent circle.

A. Trace Complexity

In this work, we offer to use *trace complexity* to estimate the fidelity of the generated trace. This approach identifies and quantifies the types of structure featured by packet traces in communication networks, [8]. Trace complexity approximates information theory's measure of *entropy rate* [13] for traces. By estimating the entropy rate, it quantifies the amount of *temporal* and *non-temporal* structure. Briefly, trace complexity measures the *structure*, or conversely, randomness, found in a traffic trace. A packet trace with *low entropy* has *low trace complexity*, and its pattern is more predictable and structured. Conversely, a more random trace will have *high trace complexity* and low structure.

We will explore two types of trace complexity: temporal complexity and non-temporal complexity. Concisely, temporal complexity measures the randomness represented by the ordering of the packets (e.g., bursts, repetitions, cycles of similar elements, etc.). Non-temporal complexity estimates the randomness represented by the frequency nodes. For a complete discussion of trace complexity, we refer to the original paper [8]. We believe that any model which effectively mimics traffic patterns will also generate traces with similar temporal and non-temporal complexity values. To test the traces, we implemented the process of the original paper [8] and calculated complexity values for all tested traces.

Figure 2 shows the complexity map for our trace set. Each trace is paired with a generated trace from our model, marked by a transparent circle of the same color. We observe that the original trace set is diverse in terms

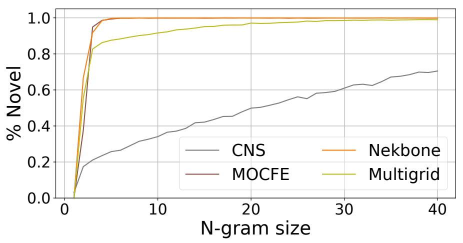


Fig. 3: Proportion of n-grams that are novel against n-gram size between the original and generated traces.

of its complexity values, particularly on the temporal scale, with traces having temporal complexity values ranging from roughly 0.2 to near 1. In the generated traces, we observe that each circle for each generated trace overlaps with its original trace circle; this means that both traces have a similar complexity profile. We note that the generated NekBone trace is nearly identical. Looking at each complexity dimension (the x or y axis) separately, we see that the generated traces closely follow their original counterparts' temporal and non-temporal structures.

B. Novelty of Generated Traces

As mentioned, our goal when generating new traces is to generate traces that mimic the original traces on different statistical measures and patterns to a close degree. Conversely, we would not like the model to memorize long sequences from the original trace and place them in the generated trace. We analyze the novelty of generated traces by looking at *n-grams*. These are n-long consecutive sequences of elements. For our context, we define the singular element in the n-gram as a packet, that is, a pair of source and destination IDs (s, d). Since our traces generated traces are long, we randomly sample a set of 6000 n-grams of each length and search in the original trace. A similar methodology was used for generative LLM to estimate the uniqueness of the generated text [14], [15] as well as in Kong *et al.* [5] when evaluating generated cellular network traces.

Figure 3 shows the proportion of novel n-gram in the generated trace from the original trace to n-gram length. If an n-gram in the generated trace exists in its source trace at least once, we consider it replicated. We have tested n-grams in the range $n = [1, 40]$. The results show that not all traces exhibit the same behavior. In the Nekbone and MOCFE traces, we observe that nearly all n-grams larger than $n = 5$ are novel, around 99%. The Multigrid trace has a more gradual behavior where at $n = 5$, only 85% of n-grams are novel, while for $n = 25$ around 98% of n-grams are novel. The CNS trace is an outlier, where the proportion of novel n-grams is only 25% at $n = 5$ and 70% for $n = 40$ at the edge of the tested range. In the full version, we discuss a possible interpretation and reason for this lack of novelty of the CNS trace and discuss ways to mitigate it in the future.

Lastly, note that at least a few long n-grams have been copied from the original trace in every generated trace. This is expected to occur when using the GPT architecture and is also a known phenomenon in large language models [14]. We believe that without the original traces, the user will not have a way to identify these copied sequences.

III. Conclusions

Our evaluation shows DTG-GPT can generate novel traces while maintaining fidelity. It maintains a similar degree of temporal and non-temporal complexity. Furthermore, by our analysis, nearly all sequences produced by our model are novel. With some further research and fine-tuning we hope that in the future, DCN operators will be able to release their network traces indirectly as a set of weights to an ML model, and that this work will be a step towards a generalized network transformer model.

References

[1] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[2] A. Dietmüller, S. Ray, R. Jacob, and L. Vanbever, "A new hope for network model generalization," in *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, pp. 152–159, 2022.

[3] O. A. Adeleke, N. Bastin, and D. Gurkan, "Network traffic generation: A survey and methodology," *ACM CSUR*, vol. 55, no. 2, pp. 1–23, 2022.

[4] M. N. Hall, K.-T. Foerster, S. Schmid, and R. Durairajan, "A survey of reconfigurable optical networks," *Optical Switching and Networking*, vol. 41, p. 100621, 2021.

[5] 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*, pp. 530–544, 2024.

[6] D. C. Sicker, P. Ohm, and D. Grunwald, "Legal issues surrounding monitoring during network research," in *Proc. ACM IMC '24*, pp. 141–148, 2007.

[7] G. Aceto, F. Giampaolo, C. Guida, S. Izzo, A. Pescapè, F. Piccialli, and E. Prezioso, "Synthetic and privacy-preserving traffic trace generation using generative ai models for training network intrusion detection systems," *Journal of Network and Computer Applications*, vol. 229, p. 103926, 2024.

[8] C. Avin, M. Ghobadi, C. Griner, and S. Schmid, "On the complexity of traffic traces and implications," *Proc. ACM on Measurement and Analysis of Computing Systems*, vol. 4, no. 1, pp. 1–29, 2020.

[9] M. Bienkowski, D. Fuchssteiner, J. Marcinkowski, and S. Schmid, "Online dynamic b-matching: With applications to reconfigurable datacenter networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 48, no. 3, pp. 99–108, 2021.

[10] C. Griner, C. Avin, and G. Einziger, "Beyond matchings: Dynamic multi-hop topology for demand-aware datacenters," *Computer Networks*, vol. 240, p. 110143, 2024.

[11] U. DOE, "Characterization of the DOE mini-apps." https://portal.nersc.gov/project/CAL/doe-miniapps.htm, 2016.

[12] T. COLLECTION. https://trace-collection.net/.

[13] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[14] R. T. McCoy, P. Smolensky, T. Linzen, J. Gao, and A. Celikyilmaz, "How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 652–670, 2023.

[15] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, "Quantifying memorization across neural language models," *arXiv preprint arXiv:2202.07646*, 2022.