Harnessing Public Code Repositories to Develop Production-Ready ML Artifacts for Networking

Author(s):
Khan, Punnal Ismail; Guthula, Satyandra; Beltiukov, Roman; Schmid, Roland; Bühler, Tobias; Gupta, Arpit; Vanbever, Laurent; Willinger, Walter

Publication date:
2024-07

Permanent link:
https://doi.org/10.3929/ethz-b-000683290

Originally published in:
https://doi.org/10.1145/3673422.3674898

Harnessing Public Code Repositories to Develop Production-Ready ML Artifacts for Networking

Punnal Ismail Khan, Satyandra Guthula, Roman Beltiukov, Roland Schmid‡, Tobias Bühler‡, Arpit Gupta, Laurent Vanbever‡, Walter Willinger†
UC Santa Barbara ‡ ETH Zürich † NIKSUN Inc.

ACM Reference Format:
Punnal Ismail Khan, Satyandra Guthula, Roman Beltiukov, Roland Schmid‡, Tobias Bühler‡, Arpit Gupta, Laurent Vanbever‡, Walter Willinger† . 2024. Harnessing Public Code Repositories to Develop Production-Ready ML Artifacts for Networking. In *Applied Networking Research Workshop (ANRW 24), July 23, 2024, Vancouver, AA, Canada.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3673422.3674898

Motivation: Despite decades of developing machine learning (ML) models for networking, achieving production-ready, generalizable models remains a challenge. This difficulty often stems from the reliance on publicly available but poorly specified training data, as highlighted in previous works [1, 11]. To enhance model generalizability, it is crucial to facilitate easier data collection that accurately reflects the diversity of real-world network environments.

One method involves passively collecting network data from production environments, but reliably labeling the collected data for different learning problems is challenging. An alternative strategy is to endogenously generate network data that closely mimic real target settings at scale. While recent efforts, like PINOT [4] and netUnicorn [5], show promise, it is unclear how these methods can be scaled to endogenously generate traffic that accurately represents a wide range of network conditions and application behaviors. Thus, to develop generalizable ML models for learning problems in networking, we need to simplify and scale network traffic generation that mimics an unprecedented mix of realistic networked applications/services for a wide range of different (emulated) network conditions.

To realize this goal, we make two key observations. First, there exists an abundance of diverse application logic readily available through millions of publicly accessible code repositories hosted by entities such as GitHub [10], BitBucket [7], etc. The set of these publicly accessible code repositories is often referred to as "Big Code" [6], and previous work has demonstrated that a significant portion (over 200 million such repositories) contains code that, when run and deployed, generates network traffic [8]. Specifically, this prior work identified around 70 k GitHub repositories with containerized applications that can be easily orchestrated to generate meaningful and diverse network traffic using the default Docker-Compose files in these repositories.

Second, due to the widespread adoption of SDN, there already exist several publicly accessible programmable platforms that can be used to emulate diverse network conditions. These platforms include NSF-supported research infrastructures, such as EdgeNet [14], ChiEdge [9], Fabric [3], as well as on-demand infrastructures offered by different cloud service providers, such as AWS, Azure, Digital Ocean, GCP. Moreover, as recent efforts such as PINOT [4] and P4Campus [12] demonstrate, even small or medium-sized enterprise networks (e.g., university campus networks) can be transformed into data collection infrastructures by using off-the-shelf SDN devices and single-board computers.

Proposed Approach: In this paper, we propose a new approach that combines these two observations. In particular, we describe our ongoing effort to develop **netMosaic**—a data-collection platform that leverages existing public code repositories and programmable network infrastructures to simplify data collection for various learning problems under a variety of different network conditions. By facilitating the extraction and mapping of readily available application logic from "Big Code" to different emulated network environments that can be realized using already existing programmable network infrastructures, netMosaic affords researchers unprecedented opportunities to endogenously generate labeled training data at scale for a wide range of popular networking-relevant learning problems, such as traffic classification, APT detection, and device/service fingerprinting. Unlike the existing datasets that capture only limited application logic from a few network environments, the proposed approach enables the iterative curation of well-specified datasets for millions of applications interacting under diverse network conditions. These datasets aim to address the urgent need for learning models that are less prone to underspecification and thus more likely to generalize. This approach offers enhanced opportunities to develop generalizable models for different learning problems in networking.

ANRW 24, July 23, 2024, Vancouver, AA, Canada

ACM ISBN 979-8-4007-0723-0/24/07
https://doi.org/10.1145/3673422.3674898

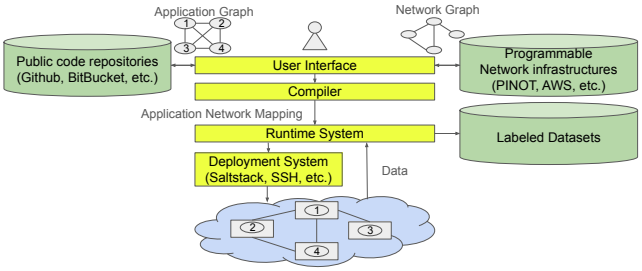

Figure 1: Overview of netMosaic: The yellow blocks represent netMosaic's key components.

As shown in Figure 1, for each repository, netMosaic first follows the workflow described in previous work [8] to generate network traffic. It then uses this traffic to extract a connectivity graph for different services (e.g., MongoDB, ElasticSearch, etc.) and store it in a queryable database system. Users can query this database as well as another database that captures topological information of various programmable network infrastructures. Through netMosaic' user interface, users can specify their intents, such as the **attributes** for different links in a selected repository's application graph and the **mapping** between nodes in the application graph and hosts in the network graph across one or more physical/virtual network infrastructures. netMosaic then synthesizes the required commands and configurations for various hosts to realize user intents with high fidelity and configures them using infrastructure-specific deployment systems (e.g., SaltStack, Kubernetes). These nodes interact with the runtime system for error handling and reporting the collected data (i.e., packet traces, system/application logs).

Initial Feasibility Study and Evaluation: We implemented an initial version of netMosaic and consider in the following the use case of traffic classification, a much-studied learning problem in the networking area, to demonstrate the capabilities of netMosaic.

Does netMosaic enable the curation of diverse and realistic datasets? We considered 16k (out of a total of 73k) GitHub repositories that are capable of generating network traffic data (e.g., see [8]). Running the Docker Compose files as-is for these repositories, netMosaic generated network traffic in the form of ∼**1.7 million** distinct flows and ∼**54 million** packets. For the traffic classification challenge, we labeled the data using port numbers and corresponding service names based on IANA's port mappings [2]. Though one might question the need for an ML model when port numbers could identify services in network traffic, the use of port numbers alone is unreliable due to possible circumvention; for instance, traffic could be encrypted via VPN, or users might deliberately use non-standard port numbers to hide their service usage—motivating an ML-based approach.

By leveraging service name mapping based on port numbers, we identified **264** unique services in our dataset, where the top six services (present in most repositories) are: HTTPS, Redis, PostgreSQL, Eforward, MongoDB, and MySQL. Compared to existing publicly available datasets [15, 16], netMosaic not only allows for more distinct services but it also offers the flexibility to generate more traffic associated with minority classes and can therefore be used to curate datasets that are balanced across different traffic classes.

Table 1: Performance of models trained on Data A (Model A) and Data B (Model B) and tested on unseen Data C.

	Model A		**Model B**	
	Data A	**Data C**	Data B	**Data C**
Random Forest	0.83	**0.24**	0.81	**0.52**
Decision Trees	0.81	**0.10**	0.80	**0.28**
Logistic Regression	0.23	**0.06**	0.15	**0.14**
MLP	0.76	**0.07**	0.73	**0.37**

Does netMosaic facilitate the development of generalizable ML models? To demonstrate the capability of netMosaic to curate datasets under various network conditions, we limited our study to 256 GitHub repositories. For each repository, we collected data under three scenarios: (a) the default setting (**Data A**), with no specified network conditions; (b) a low-congestion setting (**Data B**) with packet loss rates, latency, and bandwidth set within the ranges of 0-30%, 0-3 ms, and 500-1000 Mbps, respectively; and (c) a high-congestion setting (**Data C**) with these metrics ranging from 60-90%, 6-9 ms, and 1-500 Mbps, respectively. We extracted flow-level features using the CIC-Flowmeter [13] and trained four learning models: Random Forest, Decision Trees, Logistic Regression, and Multi-Layer Perceptron (MLP). The results, displayed in Table 1, indicate that models trained on **Data B** outperformed those trained on **Data A** when both were tested on the high-congestion **Data C**. This suggests that training data collected under realistic but controlled network conditions can indeed improve model generalizability to new and unseen network environments.

Conclusion and Future Directions: While our initial findings show the potential of netMosaic in aiding the development of more generalizable ML models that promise to form the basis for designing production-ready ML artifacts, they also reveal that there still exists a significant gap between test performance and desired deployment performance. To address this gap, we plan to develop closed-loop workflows inspired by netUnicorn [5]. These pipelines will focus on identifying the network conditions that have the most impact on data quality and providing insights into which conditions affect model generalizability and influence a trained model's deployment performance. We also plan to consider new learning problems such as flow completion time, throughput prediction, and version fingerprinting (i.e., determining the version of identified services, such as MongoDB).

REFERENCES

[1] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*. 3971–3988.

[2] Internet Assigned Numbers Authority. [n. d.]. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[3] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Inder S Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth. 2019. Fabric: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing* 23, 6 (2019), 38–47.

[4] Roman Beltiukov, Sanjay Chandrasekaran, Arpit Gupta, and Walter Willinger. 2023. Pinot: Programmable infrastructure for networking. In *Proceedings of the Applied Networking Research Workshop*. 51–53.

[5] Roman Beltiukov, Wenbo Guo, Arpit Gupta, and Walter Willinger. 2023. In Search of netUnicorn: A Data-Collection Platform to Develop Generalizable ML Models for Network Security Problems. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2217–2231.

[6] Pavol Bielik, Veselin Raychev, and Martin Vechev. 2015. Programming with" big code": Lessons, techniques and applications. In *1st Summit on Advances in Programming Languages (SNAPL 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[7] Bitbucket. 2023. https://bitbucket.org/ Accessed: 2024-04-16.

[8] 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* (Austin, Texas) *(HotNets '22)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3563766.3564084

[9] CHI@Edge. 2023. https://www.chameleoncloud.org/experiment/chiedge/ Accessed: 2024-04-16.

[10] Github. 2023. https://www.github.com/ Accessed: 2024-04-16.

[11] Arthur S Jacobs, Roman Beltiukov, Walter Willinger, Ronaldo A Ferreira, Arpit Gupta, and Lisandro Z Granville. 2022. Ai/ml for network security: The emperor has no clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1537–1551.

[12] Hyojoon Kim, Xiaoqi Chen, Jack Brassil, and Jennifer Rexford. 2021. Experience-driven research on programmable networks. *ACM SIGCOMM Computer Communication Review* 51, 1 (2021), 10–17.

[13] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. 2017. Characterization of tor traffic using time based features. In *International Conference on Information Systems Security and Privacy*, Vol. 2. SciTePress, 253–262.

[14] Berat Can Şenel, Maxime Mouchet, Justin Cappos, Olivier Fourmaux, Timur Friedman, and Rick McGeer. 2021. Edgenet: A multi-tenant and multi-provider edge cloud. In *Proceedings of the 4th international workshop on edge systems, analytics and networking*. 49–54.

[15] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1 (2018), 108–116.

[16] Thijs van Ede, Riccardo Bortolameotti, Andrea Continella, Jingjing Ren, Daniel J. Dubois, Martina Lindorfer, David R. Choffnes, Maarten van Steen, and Andreas Peter. 2020. FlowPrint: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic. *Proceedings 2020 Network and Distributed System Security Symposium* (2020). https://api.semanticscholar.org/CorpusID:211265114