Poster: Continual Network Learning

Nicola Di Cicco
Politecnico di Milano
Milan, Italy

Amir Al Sadi
Università di Bologna
Bologna, Italy

Chiara Grasselli
Università di Bologna
Bologna, Italy

Andrea Melis
Università di Bologna
Bologna, Italy

Gianni Antichi
Politecnico di Milano
Milan, Italy

Massimo Tornatore
Politecnico di Milano
Milan, Italy

ABSTRACT

We make a case for in-network Continual Learning as a solution for seamless adaptation to evolving network conditions without forgetting past experiences. We propose implementing Active Learning-based selective data filtering in the data plane, allowing for data-efficient continual updates. We explore relevant challenges and propose future research directions.

KEYWORDS

In-network Machine Learning, Continual Learning, Active Learning, Programmable Data Planes

1 INTRODUCTION

Machine Learning (ML) has lately become a prominent research area for the networking community, with applications in a broad range of topics such as traffic classification [31], routing [8], congestion control [1], and traffic forecasting [19]. In particular, in-network ML has become very attractive, as it allows to leverage the expressiveness of ML models at data plane speed [28, 33]. The common denominator between many in-network ML use-cases is to train a model in the control plane using annotated historical data, and then deploy the model in the data plane for near real-time inference [21, 26, 33]. Unfortunately, the training data will eventually become outdated (a phenomenon formally known as *"distribution shift"* or *"concept drift"*), causing the deployed ML model to suffer from performance degradation [16, 17]. While the necessity for frequent model updates has already been raised [3], three fundamental questions remain: (1) **When should we update our model?** The answer is (in theory) fairly simple: *continuously*. We should assume that the input patterns observed by a deployed ML model may change at any point in time; (2) **Which data should we select for model updates?** Again, the answer is (in theory) simple: *only the data that is useful for learning new things*. (3) **What should our model learn?** In principle, *everything*. We want a model that dynamically expands its predictive power without forgetting past experiences.

In this poster, we aim to take a step towards designing a solution that answers those questions. We propose combining Active Learning (AL) [23], which enables filtering relevant information from a vast pool of unannotated data, and Continual Learning (CL) [10], which allows us to learn from streaming data *without forgetting past concepts*. The former, implemented in the switch ASIC, allows us to choose *the right amount of information* that shall be mirrored to the control plane, where the model is updated continually. Finally, the new model can be installed back in the data plane.

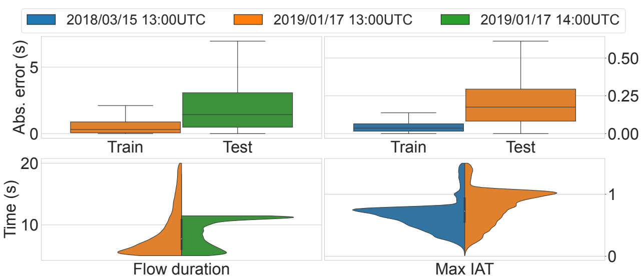

Figure 1: Distribution shift of TCP flow features from a real-world commercial backbone link [7]. The distribution shift of flow duration and inter-arrival time (IAT) causes performance degradation of a ML model trying to predict them.

Implementing this solution is nontrivial and needs answering the following research questions: (1) how to implement AL-based filtering in the data plane?; (2) how selective should AL be for network learning?; (3) which ML models are most suitable for continual learning of network traffic?; and (4) how to dynamically reconfigure the data plane?

2 THE CASE FOR CONTINUAL LEARNING

We run some tests on real-world traces to characterize the amount of distribution shift in TCP flow features. We extracted commonly-used features from real-world traces [7] (e.g., flow duration, inter-arrival time (IAT), and packet size statistics). We observed a shift in the flow duration and in the maximum IAT for small time scales (~one hour) and for large time scales (~a year), respectively. To quantify the impact of these shifts, we consider ad-hoc regression tasks[1] where the targets are either the flow duration or the maximum IAT. We observe[2] that the test error is significantly larger than training, a phenomenon that is imputable to the observed feature drifts (Fig. 1). Indeed, classical ML models will work properly only if the train and test data are approximately i.i.d. [5]. As such, practical in-network ML calls for smart, adaptive approaches.

Why can't we run existing proposals in a loop? Literature has been active in proposing efficient means for offloading trained ML models to the data plane [9, 28, 29, 34]. We here consider an orthogonal problem: how to train a ML model continually from packet streams with the optimal amount of annotated training data. Though Online Learning approaches have been explored [17, 28], they 1) assume that every streamed data point is labeled, and 2) do not pay attention about forgetting the past as long as the model

[1]This is because CAIDA traces do not have task-specific class labels.
[2]For visualization purposes, we focus on the ranges [5, 20]s for flow duration and [0, 1.5]s for inter-arrival time.

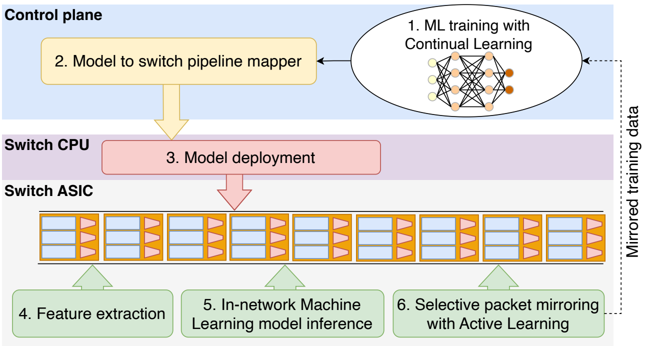

Figure 2: Our approach. The ML model is created (1) and then deployed in the switch ASIC (2), to perform inference at data plane speed (3). Selective mirroring (4) with Active Learning is deployed to keep the ML updated with Continual Learning.

is fit to the current experience. In our proposal, we want not only to learn adaptively, but also to remember (and therefore exploit) everything that was observed in the past. In this way, our model will not need additional data for re-learning already-observed concepts.

3 OUR APPROACH

Fig. 2 illustrates our proposal: to incorporate in a single closed-loop framework the following building blocks:

(1) Model training: update the ML model over the time with CL.
(2) Model deployment: deploy the new ML model in the data plane.
(3) In-network inference: enable inference at data plane speed.
(4) Selective mirroring: mirror to the control plane only the data useful for expanding the knowledge of the model with AL.

As a proof-of-concept experiment, we consider a subset of the CIC2019 dataset for DDoS classification [24]. We consider DDoS classes to represent disjoint learning tasks, which are presented to the model in sequence. For each task, the model must not only discriminate between benign and malicious flows but also place the malicious flows in the right class.

We implement a baseline Continual Random Forest (CRF), consisting of a RF augmented with a replay buffer storing the most informative past exemplars. We use the vote count as AL query strategy, selecting only data points whose predictions had less than 90% majority. We retrain after each query, which is computationally efficient for RFs. We consider an Adaptive Random Forest (ARF) as a purely online (but not continual) state-of-the-art baseline [11, 17]. In contrast to CRF, ARF assumes that every data point is labeled. We also consider an "oracle" RF trained on the full dataset as an upper-bound on the average performance over all tasks.

Fig. 3 shows the performance of CRF and ARF over the sequential tasks, and the percentage of queried labels by CRF relative to the full stream size. A purely adaptive learner such as ARF, though able to master individual tasks, quickly forgets past concepts. Instead, our baseline CRF achieves a performance close to the oracle upper-bound, while requiring labeling only ~1% of the observed samples.

4 CHALLENGES

Challenge #1: implementing AL-based filtering in the data plane. Vote count in our baseline CRF is a decent query strategy,

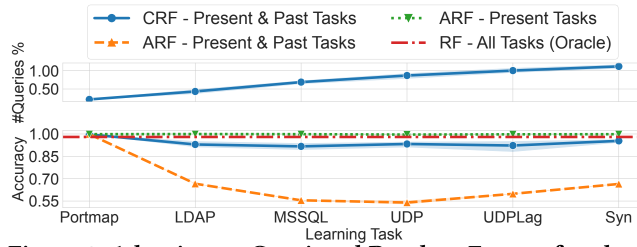

Figure 3: Adaptive vs. Continual Random Forests for class-incremental DDoS classification on CIC2019. At the end of the stream, CRF achieves performance close to an "oracle" while requiring only ~1% of the data.

but information-theoretic quantities [4, 12] are among the state-of-the-art. Their data plane implementation is not trivial, as it would require floating-point arithmetics. Even if not standard, authors in [18] propose a way to implement floating-point arithmetics in P4.

Challenge #2: how selective should AL be. A small selectivity implies a large mirroring overhead, whereas a large selectivity implies a potential information loss. Applying AL to streaming data is, as of today, a novel twist on classical techniques [22]: investigating these trade-offs opens up interesting research directions.

Challenge #3: choosing the right CL strategy. Our baseline leverages a slowly-growing experience buffer, which may not be desirable. Strategies for maintaining the buffer of fixed size can be investigated [20]. Other solutions, e.g., regularized neural networks, do not require any storage overhead other than the model [15], but are ill-advised for tabular data [25]. Ultimately, the choice depends on the available storage/computational resources and the goodness-of-fit to the characteristics of task-specific data.

Challenge #4: runtime dataplane reconfiguration. Currently, if we want to add a new functionality to a switch, we need first to reroute the traffic of that switch, flush a new image in its ASIC and then restore the original traffic policy configuration. This process can lead to dramatic consequences if performed carelessly [14]. Programming the switch at run-time is possible [32], but not for RMT [6], the common commercial devices architecture [2, 13]. Researchers have also explored means to enable isolation between offloaded programs [27, 30, 35], which we will investigate to isolate the Active Learning processing and the rest of the pipeline.

REFERENCES

[1] Soheil Abbasloo, Chen-Yu Yen, and H. Jonathan Chao. 2020. Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 632–647. https://doi.org/10.1145/3387514.3405892
[2] AMD. 2019. Naples DSC-100 Distributed Services Card. https://pensando.io/assets/documents/Naples_100_ProductBrief-10-2019.pdf.
[3] G. Apruzzese, P. Laskov, and J. Schneider. 2023. SoK: Pragmatic Assessment of Machine Learning for Network Intrusion Detection. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE Computer Society, Los Alamitos, CA, USA, 592–614. https://doi.org/10.1109/EuroSP57164.2023.00042
[4] Freddie Bickford Smith, Andreas Kirsch, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. 2023. Prediction-Oriented Bayesian Active Learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 206)*, Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (Eds.). PMLR, Valencia, Spain, 7331–7348. https://proceedings.mlr.press/v206/bickfordsmith23a.html
[5] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[6] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN. *SIGCOMM Comput. Commun. Rev.* 43, 4 (aug 2013), 99–110. https://doi.org/10.1145/2534169.2486011

[7] CAIDA. 2019. The CAIDA UCSD Anonymized Internet Traces - 2018-2019. https://www.caida.org/catalog/datasets/passive_dataset

[8] Brian Chang, Aditya Akella, Loris D'Antoni, and Kausik Subramanian. 2023. Learned Load Balancing. In *Proceedings of the 24th International Conference on Distributed Computing and Networking* (Kharagpur, India) *(ICDCN '23)*. Association for Computing Machinery, New York, NY, USA, 177–187. https://doi.org/10.1145/3571306.3571403

[9] Bruno Coelho and Alberto Schaeffer-Filho. 2022. BACKORDERS: Using Random Forests to Detect DDoS Attacks in Programmable Data Planes. In *Proceedings of the 5th International Workshop on P4 in Europe* (Rome, Italy) *(EuroP4 '22)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3565475.3569074

[10] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2022. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 7 (2022), 3366–3385. https://doi.org/10.1109/TPAMI.2021.3057446

[11] Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning* 106, 9 (Oct 2017), 1469–1495. https://doi.org/10.1007/s10994-017-5642-8

[12] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian Active Learning for Classification and Preference Learning. arXiv:1112.5745 [stat.ML]

[13] Intel. 2020. Tofino: P4-programmable Ethernet switch ASIC. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series/tofino.html.

[14] Santosh Janardhan. 2021. Facebook Outage on October 4th 2021. https://engineering.fb.com/2021/10/04/networking-traffic/outage/.

[15] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3521–3526. https://doi.org/10.1073/pnas.1611835114 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.1611835114

[16] Ankur Mallick, Kevin Hsieh, Behnaz Arzani, and Gauri Joshi. 2022. Matchmaker: Data Drift Mitigation in Machine Learning for Large-Scale Systems. In *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu (Eds.), Vol. 4. 77–94. https://proceedings.mlsys.org/paper_files/paper/2022/file/069a002768bcb31509d4901961f23b3c-Paper.pdf

[17] Pavol Mulinka and Pedro Casas. 2018. Adaptive Network Security through Stream Machine Learning. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 4–5. https://doi.org/10.1145/3234200.3234246

[18] Shivam Patel, Rigden Atsatsang, Kenneth M. Tichauer, Michael H. L. S. Wang, James B. Kowalkowski, and Nik Sultana. 2022. In-Network Fractional Calculations Using P4 for Scientific Computing Workloads. In *Proceedings of the 5th International Workshop on P4 in Europe* (Rome, Italy) *(EuroP4 '22)*. Association for Computing Machinery, New York, NY, USA, 33–38. https://doi.org/10.1145/3565475.3569083

[19] Yu Qiao, Chengxiang Li, Shuzheng Hao, Jun Wu, and Liang Zhang. 2022. Deep or Statistical: An Empirical Study of Traffic Predictions on Multiple Time Scales. In *Proceedings of the SIGCOMM '22 Poster and Demo Sessions* (Amsterdam, Netherlands) *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 10–12. https://doi.org/10.1145/3546037.3546048

[20] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. 2017. iCaRL: Incremental Classifier and Representation Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 5533–5542. https://doi.org/10.1109/CVPR.2017.587

[21] Davide Sanvito, Giuseppe Siracusano, and Roberto Bifulco. 2018. Can the Network Be the AI Accelerator?. In *Proceedings of the 2018 Morning Workshop on In-Network Computing* (Budapest, Hungary) *(NetCompute '18)*. Association for Computing Machinery, New York, NY, USA, 20–25. https://doi.org/10.1145/3229591.3229594

[22] Akanksha Saran, Safoora Yousefi, Akshay Krishnamurthy, John Langford, and Jordan T. Ash. 2023. Streaming Active Learning with Deep Neural Networks. arXiv:2303.02535 [cs.LG]

[23] Burr Settles. 2012. *Active Learning*. Springer Cham, Cham, Switzerland. https://doi.org/10.1007/978-3-031-01560-1

[24] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani. 2019. Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, Chennai, India, 1–8. https://doi.org/10.1109/CCST.2019.8888419

[25] Ravid Shwartz-Ziv and Amitai Armon. 2021. Tabular Data: Deep Learning is Not All You Need. In *8th ICML Workshop on Automated Machine Learning (AutoML)*. https://openreview.net/forum?id=vdgtepS1pV

[26] Giuseppe Siracusano and Roberto Bifulco. 2018. In-network Neural Networks. arXiv:1801.05731 [cs.DC]

[27] Radostin Stoyanov and Noa Zilberman. 2020. MTPSA: Multi-Tenant Programmable Switches. In *Proceedings of the 3rd P4 Workshop in Europe* (Barcelona, Spain) *(EuroP4'20)*. Association for Computing Machinery, New York, NY, USA, 43–48. https://doi.org/10.1145/3426744.3431329

[28] Tushar Swamy, Alexander Rucker, Muhammad Shahbaz, Ishan Gaur, and Kunle Olukotun. 2022. Taurus: A Data Plane Architecture for per-Packet ML. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '22)*. Association for Computing Machinery, New York, NY, USA, 1099–1114. https://doi.org/10.1145/3503222.3507726

[29] Tushar Swamy, Annus Zulfiqar, Luigi Nardi, Muhammad Shahbaz, and Kunle Olukotun. 2023. Homunculus: Auto-Generating Efficient Data-Plane ML Pipelines for Datacenter Networks. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 329–342. https://doi.org/10.1145/3582016.3582022

[30] Tao Wang, Xiangrui Yang, Gianni Antichi, Anirudh Sivaraman, and Aurojit Panda. 2022. Isolation Mechanisms for High-Speed Packet-Processing Pipelines. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 1289–1305. https://www.usenix.org/conference/nsdi22/presentation/wang-tao

[31] Matthias Wichtlhuber, Eric Strehle, Daniel Kopp, Lars Prepens, Stefan Stegmueller, Alina Rubina, Christoph Dietzel, and Oliver Hohlfeld. 2022. IXP Scrubber: Learning from Blackholing Traffic for ML-Driven DDoS Detection at Scale. In *Proceedings of the ACM SIGCOMM 2022 Conference* (Amsterdam, Netherlands) *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 707–722. https://doi.org/10.1145/3544216.3544268

[32] Jiarong Xing, Kuo-Feng Hsu, Matty Kadosh, Alan Lo, Yonatan Piasetzky, Arvind Krishnamurthy, and Ang Chen. 2022. Runtime Programmable Switches. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 651–665. https://www.usenix.org/conference/nsdi22/presentation/xing

[33] Zhaoqi Xiong and Noa Zilberman. 2019. Do Switches Dream of Machine Learning? Toward In-Network Classification. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks* (Princeton, NJ, USA) *(HotNets '19)*. Association for Computing Machinery, New York, NY, USA, 25–33. https://doi.org/10.1145/3365609.3365864

[34] Changgang Zheng, Zhaoqi Xiong, Thanh T Bui, Siim Kaupmees, Riyad Bensoussane, Antoine Bernabeu, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. 2022. IIsy: Practical In-Network Classification. arXiv:2205.08243 [cs.NI]

[35] Peng Zheng, Theophilus Benson, and Chengchen Hu. 2018. P4Visor: Lightweight Virtualization and Composition Primitives for Building and Testing Modular Programs. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (Heraklion, Greece) *(CoNEXT '18)*. Association for Computing Machinery, New York, NY, USA, 98–111. https://doi.org/10.1145/3281411.3281436