7 - In-Network Machine Learning: Inference at the Speed of Data

Changgang Zheng and Noa Zilberman, University of Oxford
changgang.zheng, noa.zilberman@eng.ox.ac.uk

Background and Status

Machine learning (ML) was widely adopted and extensively utilized by data-intensive applications such as stream processing and cybersecurity. However, the high demands of ML workloads pose challenges that local accelerators (e.g., GPUs, FPGAs) struggle to address. To cope with the scale and volume of the data to process, distributed ML is often used.

Networking and ML have a complex relationship. ML services rely on the network to carry data and connect accelerators, but often the exchange of data between accelerators is a bottleneck in distributed ML [10, 14]. At the same time, network devices can carry more data than an accelerator card can process, since a single network switch can process more than 100 terabits per second (Tbps), two orders of magnitude more than current accelerators. One way to bridge the mismatch between networking and ML is in-network ML. To explain it, a brief introduction to network programmability is provided.

High end network devices are designed to process as many packets as possible, as quickly as possible, exceeding 10 billion packets per second (Bpps) with sub-microsecond latency. Over the last decade, many network devices have become programmable [3], combining the processing pipeline (referred to as data plane) with reconfigurable match tables (RMT). The most popular general architecture is called PISA, Protocol Independent Switch Architecture, shown in Figure 1(c). The domain specific language commonly used is P4.

As shown in Figure 1, the PISA architecture has three components: i) a programmable parser for extracting header information from incoming packets, ii) a match-action pipeline that looks up keys (e.g., headers) in tables and matches them with actions (e.g., output port), and iii) a deparser for reconstructing packets. Current programmable network devices range from software switches to switch-ASICs, FPGAs, SmartNICs, and DPUs.

The programmability of network devices enabled in-network computing, the offloading of applications normally running on a host to run within network devices. Examples of In-network computing applications include telemetry, caching [7] and gradient aggregation [14]. It demonstrated order of magnitude improvement in throughput, latency, and power efficiency [18]. Accelerating ML using in-network computing is one promising direction. This includes [21] both Network-Assisted ML, meaning using the network to accelerate workloads running on traditional targets, and in-network ML.

In-network ML offloads the entire ML inference process to programmable network devices [21]. It trains the ML model on servers or standard accelerators, and then maps the trained model to the device's data plane for inference. While resource utilization varies, programmable switches often use only half of their resources for basic network functions [23], enabling in-network ML to leverage remaining resources. Models scale up to 20 trees and 55 features, achieving high classification accuracy [24]. The three main benefits of in-network ML are location, latency and load (3Ls) [24]. Network devices are already part of the infrastructure that data goes through (location), they can infer data closer to its source (latency) and inference performance scales with network bandwidth, while reducing load on end-hosts (load). In-network ML can be deployed either on network devices that are already on route from the data source, or as a network-attached accelerator. The first requires no additional cost or latency, while the second benefits from more available processing resources.

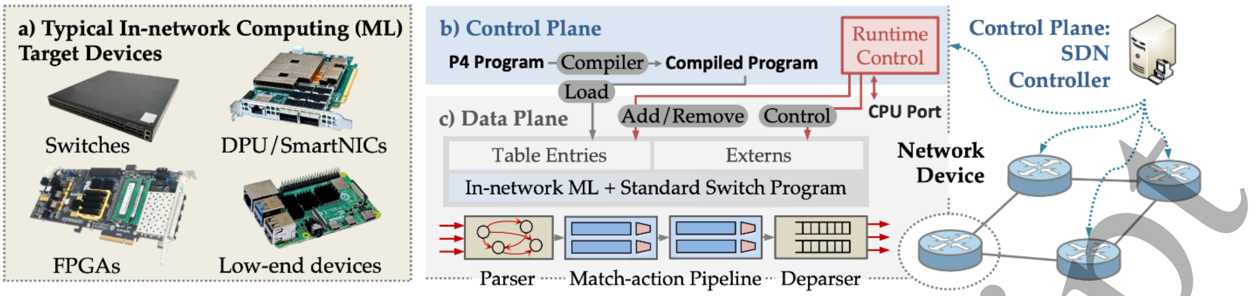

Figure 1. a) Commodity programmable network devices commonly used for in-network ML. b) The control plane of a network devices, e.g., switch-box CPU. c) A PISA based data plane of a programmable network devices, combining in-network ML and standard network functions.

Meeting the Exascale Challenges

In-network ML inference process has two major components: feature extraction, and mapping of a trained ML algorithm to the data plane.

Feature Extraction. The extraction of features from incoming data can be done on various levels of granularity. Packet-level features are extracted directly from incoming packets, and are stateless, meaning the data from one packet does not affect later packets. Flow-level features consider the data flowing between a source-destination pair, and are stateful, meaning that information is accumulated over time, such as the traffic volume of the flow. Aggregation level features are also stateful, but combine information based on different metrics, e.g., the statistics of combining multiple ports. File-level feature extraction is possible but limited to certain types of files (e.g., text) and may have reduced performance, for example due to recirculation of packets within the pipeline. While packet-level features can be extracted in the parser, more complex feature extraction and feature engineering requires match-action pipeline stages and memory resources. One of the challenges for feature extraction is that packets arrival may be unordered, and information may travel through multiple routes.

ML Model Mapping. To maximise packet processing rate, network devices are constructed with limitations on programmability. For example, mathematical operations such as multiplication, and data types such as floating-point, are unavailable. Moreover, these are billions-of-transistors devices, therefore, to make their manufacturing feasible, user-available resources are limited in comparison with CPUs. For example, memory on a switch-ASIC is in the range of 100's of megabits, and the number of processing stages is e.g., 12 or 20. There is no native support for loops, and the closest equivalent, packet recirculation, has throughput implications. While some works have addressed these challenges through changes to the hardware [17] or by focusing on NICs [16], other successfully mapped inference models to off-the-shelf devices without loss of performance [21]. The mappings of ML models to the data plane can be divided into three groups [25]: Direct Mapping, where every step of the inference is translated into a stage in the pipeline, Lookup-based, where lookup tables are used for complex calculations, and Encode-based, where the feature space is sliced and encoded.

In-network ML Implementation. Implementing mapped ML models on target devices, and porting between different devices, requires significant effort. To solve this problem, Planter [25] offers rapid prototyping of in-network ML. Planter supports a range of ML models, architectures and target devices. It is modular, so users can extend the functionality. The framework also supports automatic parameter tuning, and several example use cases. Targets range from low-cost Raspberry Pi, through

Alveo FPGA, to high performance Intel Tofino switch-ASIC. The demonstrated performance using ensemble tree models was 4.8 billion decisions per second.

Advances in Science and Technology to Meet Challenges

While in-network ML achieves high system performance, the size and complexity of the models that can be deployed are limited by resource constraints. This problem was identified by Microsoft as a key challenge for in-network computing [2], and Microsoft developed a resource elasticity solution [9] to address it.

As often only small or medium size models can be deployed on a network device [1, 5, 8, 12, 13, 25], ML performance can be lower than an unlimited size model running in a data centre. One solution is to distribute the model across multiple network devices, as demonstrated by DINC [23]. It has shown that ML models can be distributed across a data centre network or a wide area network, while coexisting with network functionality. A different solution, proposed in IIsy [24], is a hybrid deployment of in-network ML: deploying a small model on a network device, and a large model at the backend. This way, high confidence inference decisions are taken within the network, while only a fraction of the transactions needs to be inferred in a traditional manner. This enables both high ML performance and high system performance, while saving compute resources.

Data shift and model updates further challenge production deployments. P4Pir [20] suggests solving this problem by introducing a continuous update solution, focused on in-network ML for IoT.

Most in-network ML research focused so far on network traffic classification and cybersecurity [1, 8, 16, 17, 21, 24]. Examples that go beyond these use cases include load-balancing [22], predicting future stock prices [6], and image classification [15]. Still, the greatest challenge for in-network ML is wide adoption. This calls for extending its range of use cases, with data intensive and time-sensitive applications being the immediate candidates. Real time processing of sensor data, for example, contextual streams from the GOTO telescope [19], from particle detectors at CERN's Large Hadron Collider [4], and from modern multi-Hz laser wakefield acceleration facilities [11], holds a significant potential.

Concluding Remarks

In-network ML is an emerging technology, providing notable system performance benefits for machine learning inference. Available solutions enable deployment on commercial devices, supporting a range of ML models. Still, resource constraints remain a challenge, with hybrid and distributed deployments suggested as a solution.

There are opportunities for breakthrough research in the applications of in-network ML, benefiting the scientific community across disciplines, and now is the time to seize it.

Acknowledgements

This work was partly funded by VMware and Innovate UK (project 10056403) as part of the SmartEdge EU project (grant agreement No. 101092908). We acknowledge support from Intel and NVIDIA. For the purpose of Open Access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript (AAM) version arising from this submission.

References

[1] Aristide Akem Tanyi-Jong, Michele Gucciardo, and Marco Fiore. Flowrest: Practical flow-level inference in programmable switches with random forests. IEEE INFOCOM 2023-IEEE Conference on Computer Communications 2023.

[2] Victor Bahl. Unlocking the potential of in-network computing for telecommunication workloads. Website. https://azure.microsoft.com/en-us/blog/ unlocking-the-potential-of-in-network-computing-for-telecommunication-workloads/ [Online; accessed November 2023].

[3] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. *ACM SIGCOMM Computer Communication Review*, 43(4):99–110, 2013.

[4] CMS Collaboration, The Phase-2 Upgrade of the CMS Level-1 Trigger, CERN, Geneva, Tech. Rep. CERN-LHCC-2020-004, CMS-TDR-021, https://cds.cern.ch/record/2714892, 2020.

[5] Busse-Grawitz Coralie, Roland Meier, Alexander Dietmüller, Tobias Bühler, and Laurent Vanbever. pforest: In-network inference with random forests. arXiv:1909.05680, 2019.

[6] Xinpeng Hong, Changgang Zheng, Stefan Zohren, and Noa Zilberman. LOBIN: In-Network Machine Learning for Limit Order Books. In *2023 IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*, pages 159–166. IEEE, 2023.

[7] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soule, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. NetCache: Balancing Key-Value Stores with Fast In-Network Caching. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 121–136, 2017.

[8] Lee Jong-Hyouk and Kamal Singh. SwitchTree: in-network computing and traffic analyses with Random Forests. *Neural Computing and Applications*, pages 1-12, 2020.

[9] Daehyeok Kim, Zaoxing Liu, Yibo Zhu, Changhoon Kim, Jeongkeun Lee, Vyas Sekar, and Srinivasan Seshan. Tea: Enabling state-intensive network functions on programmable switches. In *Proceedings of the 2020 ACM SIGCOMM Conference*, pages 90–106, 2020.

[10] Liang Luo, Peter West, Jacob Nelson, Arvind Krishnamurthy, and Luis Ceze. Plink: Discovering and exploiting locality for accelerated distributed training on the public cloud. Proceedings of Machine Learning and Systems 2 (2020): 82-97.

[11] Kirchen Manuel, Sören Jalas, Philipp Messner, Paul Winkler, Timo Eichner, Lars Hübner, Thomas Hülsenbusch et al. Optimal beam loading in a laser-plasma accelerator. *Physical review letters*, 2021.

[12] Friedman Roy, Or Goaz, and Ori Rottenstreich. Clustreams: Data plane clustering. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, pages 101-107, 2021.

[13] Davide Sanvito, Giuseppe Siracusano, and Roberto Bifulco. Can the network be the AI accelerator?. *Proceedings of the 2018 Workshop on In-Network Computing*. 2018.

[14] Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan Ports, and Peter Richtarik. Scaling distributed machine learning with In-Network aggregation. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 785–808, 2021.

[15] Hisham Siddique, Miguel Neves, Carson Kuzniar, and Israat Haque. Towards network-accelerated ML-based distributed computer vision systems. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 122–129. IEEE, 2021.

[16] Giuseppe Siracusano, Salvator Galea, Davide Sanvito, Mohammad Malekzadeh, Gianni Antichi, Paolo Costa, Hamed Haddadi, and Roberto Bifulco. Re-architecting Traffic Analysis with Neural Network Interface Cards. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 513–533, 2022.

[17] Tushar Swamy, Alexander Rucker, Muhammad Shahbaz, Ishan Gaur, and Kunle Olukotun. 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*, pages 1099–1114, 2022.

[18] Yuta Tokusashi, Huynh Tu Dang, Fernando Pedone, Robert Soule, and Noa Zilberman. The Case for In-network Computing on Demand. In *Proceedings of the Fourteenth EuroSys Conference 2019*, pages 1–16, 2019.

[19] U F Burhanudin, J R Maund, T Killestein, K Ackley, M J Dyer, J Lyman, K Ulaczyk, R Cutter, Y-L Mong, D Steeghs, D K Galloway, V Dhillon, P O'Brien, G Ramsay, K Noysena, R Kotak, R P Breton, L Nuttall,

E Pallé, D Pollacco, E Thrane, S Awiphan, P Chote, A Chrimes, E Daw, C Duffy, R Eyles-Ferris, B Gompertz, T Heikkilä, P Irawati, M R Kennedy, A Levan, S Littlefair, L Makrygianni, D Mata-Sánchez, S Mattila, J McCormac, D Mkrtichian, J Mullaney, U Sawangwit, E Stanway, R Starling, P Strøm, S Tooke, K Wiersema, Light-curve classification with recurrent neural networks for GOTO: dealing with imbalanced data, *Monthly Notices of the Royal Astronomical Society*, Volume 505, Issue 3, pages 4345–4361, https://doi.org/10.1093/mnras/stab1545, 2021

[20] Mingyuan Zang, Changgang Zheng, Lars Dittmann, and Noa Zilberman. Towards Continuous Threat Defense: In-Network Traffic Analysis for IoT Gateways. *IEEE Internet of Things Journal*, 2023.

[21] Changgang Zheng, Xinpeng Hong, Damu Ding, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. In-Network Machine Learning Using Programmable Network Devices: A Survey. *IEEE Communications Surveys & Tutorials*, 2023.

[22] Changgang Zheng, Benjamin Rienecker, and Noa Zilberman. QCMP: Load Balancing via In-Network Reinforcement Learning. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing*, pages 35–40, 2023.

[23] Changgang Zheng, Haoyue Tang, Mingyuan Zang, Xinpeng Hong, Aosong Feng, Leandros Tassiulas, and Noa Zilberman. DINC: Toward Distributed In-Network Computing. *Proceedings of the ACM on Networking*, 1(CoNEXT3):1–25, 2023.

[24] Changgang Zheng, Zhaoqi Xiong, Thanh T Bui, Siim Kaupmees, Riyad Bensoussane, Antoine Bernabeu, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. IIsy: Hybrid In-Network Classification Using Programmable Switches, *IEEE/ACM Transactions on Networking*, pages 2555-2570, 2024.

[25] Changgang Zheng, Mingyuan Zang, Xinpeng Hong, Liam Perreault, Riyad Bensoussane, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. Planter: Rapid Prototyping of In-Network Machine Learning Inference, *ACM SIGCOMM Computer Communication Review*, pages 2-21, 2024.