Research Statement

Francis Y. Yan
fyy@cs.stanford.edu

My research focuses on *practical* machine learning (ML) for networking, seeking to answer: *What does it take to develop and deploy ML algorithms that robustly perform well over real networks?* Despite rapid advances in ML and the recent zeal for applying ML to networking research, challenges remain before ML can replace a large number of handcrafted algorithms that are tightly coupled with real networks. Congestion control, for example, is still dominated by TCP Cubic [10] and BBR [7], and video streaming services continue to rely on simple heuristic algorithms [12]. ML is even further from playing a major role in datacenter and wireless networks. Slow adoption of ML in these scenarios can be attributed to the requirement that these algorithms be not just performant, but also *practical*—robust, generalizable, real-time, resource efficient, and preferably interpretable. Unfortunately, these desired properties have been understudied in prior work, which often prototyped radical approaches in simulation or on small testbeds. By contrast, I strive to address practical challenges, lay the groundwork by building large-scale platforms, and create deployable ML algorithms for networking. My work has been impactful in the research community and among real users, and received a *Best Paper Award* at USENIX ATC '18 [23] and a *Community Award*[1] at USENIX NSDI '20 [22].

Past Work

My research approached *practical* ML for networking in four ways.

1. Building community benchmarks and production-grade platforms. ML algorithms need good benchmarks, e.g., computer vision would not have thrived without the success of ImageNet [8]. Before benchmarking ML for systems and networking received more attention [15, 16], I built Pantheon[2] [23], the first communal benchmark for congestion-control research. Pantheon consists of a common reference set of transport protocols and a distributed system of measurement points in twelve countries around the world. Meanwhile, ML algorithms also require real-world systems for training and validation. To investigate practical ML in video streaming, I developed Puffer[3] [22], a live TV streaming website that has attracted more than 100,000 real users and streamed 50 years of video across the Internet. Puffer operates as a randomized experiment; each session is randomly assigned to one of a set of adaptive bitrate (ABR) schemes. These platforms took *years* to build and are opened to researchers to develop and validate new algorithms for congestion control, network prediction, and bitrate selection. Pantheon has assisted three algorithms from other research groups in publishing at USENIX NSDI '18 [1, 9] and ICML '19 [14] (with another algorithm under submission). Puffer, a free and open-source "YouTube TV" built from scratch, also draws broad attention and has been featured in press articles [19, 21].

[1] "for the best paper whose code and/or data set is made publicly available"
[2] https://pantheon.stanford.edu
[3] https://puffer.stanford.edu

2. Designing ML algorithms with networking insights. My research distilled domain knowledge from networking into the design of ML algorithms. Using the platforms I built—Pantheon and Puffer—I developed practical ML algorithms for congestion control and video streaming.

1. Indigo [23]: a neural-network-based congestion-control algorithm trained with *imitation learning* [20]. At its core, it observes the network state and adjusts its congestion window; the mapping from states to adjustments is stored in a recurrent neural network [11]. The key insight of Indigo is based on a knowledge of TCP that the ideal congestion window size equals a link's BDP (bandwidth-delay product). For simulated network links whose bandwidth and delay are known, I generated "expert" congestion-control algorithms that closely approximated the ideal congestion window; of course, no expert exists for real-world paths. Then I applied imitation learning, which uses supervised learning under the hood, to train Indigo's neural network to mimic experts' behavior.

2. Fugu [22]: an ABR algorithm that combines MPC (model predictive control) [6] with a neural network predictor trained using supervised learning. Different from existing ABR schemes that predict throughput, Fugu's predictor explicitly predicts the *transmission time* of a chunk with given size; this design acknowledges a commonly overlooked fact in networking that the observed throughput varies with file size [2]. Fugu's predictor also considers low-level TCP statistics among its input signals, which improves its prediction accuracy and boosts initial video quality on a cold start. Another uncommon feature of Fugu's predictor is that it outputs a probability distribution of transmission times, instead of a single value. Given a trained predictor that reports the transmission time of any chunk, Fugu's MPC controller is able to model system dynamics and compute the optimal plan with value iteration [3].

3. Generalizing learning to real networks. Irrespective of the performance on training data or environments, ML for networking only concerns real networks eventually, and thus requires the performance to be generalizable. To this end, I consider three methods:

1. *Training in network simulators.* For those ML algorithms that have to learn in simulators (e.g., Indigo), it is well known that the discrepancy between simulation and reality creates hurdles for generalizability [5]. To bridge the gap between simulated and actual networks, I proposed *calibrated simulators* [23], which match the performance of real network paths by searching simulator parameters using Bayesian optimization [17]. It was the first effort in the area toward building high-fidelity network simulators and helped generalize Indigo from simulation to real life: Indigo yielded the best performance on average compared with the other twelve congestion-control schemes on the real testbed of Pantheon.

2. *Training on real networks.* To avoid the generalization issue of training in simulation, it is tempting to train ML algorithms directly on real networks. However, it may not be feasible yet—at least on the Internet—due to the substantial statistical noise I observed on Puffer: it may require 2 years of data per scheme to accurately measure its performance within 20% precision [22]. This largely underestimated uncertainty presents a challenge to ML, particularly reinforcement learning (RL) because RL relies on being able to slightly vary a control action and detect a change in the resulting reward. Future work may need to improve the robustness of RL in noisy environments to make training on real networks practical.

3. *Training on real data from the deployment environment.* Fugu is designed in a way that it can learn offline on the log data collected from its deployment environment, Puffer. Specifically, Fugu's transmission time predictor is trained with supervised learning to minimize the difference between its predictions and the actual transmission times of video chunks. Learning

in place on the actual deployment environment enables Fugu to achieve generalizable performance: In a blinded randomized experiment on Puffer that included 13.1 years of video streamed to 54,612 client IP addresses over an eight-month period, Fugu robustly outperformed existing ABR schemes—higher video quality, lower quality variation, and lower stall ratio. Users who were randomly assigned to Fugu chose to continue streaming longer.

I call the last two methods "learning *in situ* (in place)," which is characterized by training in the test environment (directly or using log data). Traditionally this is unacceptable in ML, but learning *in situ* is allowed in a networked system, providing that it is exactly the ultimate deployment environment. In sum, the empirical evidence from Pantheon and Puffer suggests two directions for generalizing ML to real networks: calibrating simulators to match reality if training in simulation, or learning *in situ*, preferably offline on log data from the actual deployment environment.

4. Satisfying other real-world requirements. Practical ML algorithms for networking have to fulfill the other requirements from real-world systems, e.g., they are supposed to be real-time and efficient in computation and bandwidth, ideally also with stable and interpretable behaviors. Indigo and Fugu take into account these requirements from the beginning of their design. For example, their neural networks are "shallow" (yet sufficiently powerful) so they can run in real time, and they use supervised learning to benefit from its well-studied properties (e.g., robust performance, stable training, sample efficiency). In fact, these real-world requirements have effectively constrained the design space of practical ML for networking and deserve more considerations in future work.

My research features ML algorithms that are applicable to real networks, as well as *years* of effort in building large-scale systems opened to the research community for training and validating new algorithms. These research platforms aim to lay the foundation of practical ML for networking, and have received tremendous interest from academia and industry, in addition to more than 100,000 study participants. They have assisted three algorithms from other research groups in publishing at USENIX NSDI [1, 9] and ICML [14]; the contribution is recognized by best paper awards from USENIX ATC [23] and NSDI [22].

Research Agenda

My future work is to create more *practical* ML algorithms that robustly perform well in real networked systems. I will continue to build open research platforms to study ML for networking together with the community, and collaborate with companies that own real-world systems to develop practical ML algorithms in other scenarios, such as caching, scheduling, and load balancing.

To demonstrate how I can approach a new problem using the same ways, I will take CDN (content delivery network) caching as an example. CDN caching is critical to the performance of a major portion of Internet applications [18]. The caching policy, which decides what object to admit into or evict from cache, may be improved by ML to increase the cache hit rates. Unlike prior work [4] that learns the caching policy in simple simulators, I plan to:

1. Build or collaborate with a real-world CDN service to understand the challenges of running a learning-based caching policy in real life. In the end, the learned policy will be deployed on the real service to prove its practicality.
2. Design a learning-based caching policy using domain knowledge from systems and networking. The domain knowledge may expand the design space: In contrast to a typical CDN server that would have to fetch from a nearby datacenter on a cache miss, it may also consider forwarding the request to a nearby CDN server if applicable.

3. Ensure the training is generalizable by 1) building a more accurate simulator of CDN caching to close the simulation-to-reality gap if the training requires a simulated environment, or 2) learning *in situ*, on real data or machines from the same CDN service. I found learning *in situ* to be a very powerful technique, and consider it a promising approach for achieving generalizable performance on a wide array of networked systems. My future work will continue to take generalizability as a first-class consideration, resonating with the recent pessimism [13] regarding the generalization capabilities of RL.
4. Satisfy the other real-world requirements. For instance, the learned policy should be efficient enough to run on CDN servers, which are often multi-tenant and reluctant to accept computationally expensive jobs. Given limited computational resources, the policy must still be able to handle real-world requests with high throughput and low latency.

Another direction of my future work would be revising ML algorithmically for use in networking, investigating 1) robustness to noisy training environments, 2) safe exploration when learning in real world, 3) guarantee of worst-case performance, and 4) low-latency inference in lower network layers.

In sum, my research plan is to continue laying the groundwork for ML in networking by building more research platforms such as Pantheon and Puffer, as well as to address the practical challenges systematically and algorithmically to facilitate the adoption of ML in real-world systems.

References

[1] Venkat Arun and Hari Balakrishnan. Copa: Practical Delay-Based Congestion Control for the Internet. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2018.

[2] Mihovil Bartulovic, Junchen Jiang, Sivaraman Balakrishnan, Vyas Sekar, and Bruno Sinopoli. Biases in data-driven networking, and what to do about them. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. 2017.

[3] Richard Bellman. A Markovian decision process. In *Journal of mathematics and mechanics* (1957).

[4] Daniel S Berger. Towards lightweight and robust machine learning for CDN caching. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*. 2018.

[5] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018.

[6] Eduardo F. Camacho and Carlos Bordons Alba. *Model predictive control.* Springer Science & Business Media, 2013.

[7] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: Congestion-based congestion control. In *ACM Queue* 14.5 (2016).

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition, 2019*. IEEE. 2009.

[9] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. PCC Vivace: Online-learning congestion control. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 2018.

[10] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. In *SIGOPS Oper. Syst. Rev.* 42.5 (2008). ISSN: 0163-5980. DOI: `10.1145/1400097.1400105`.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural computation* 9.8 (1997).

[12] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 Conference of the ACM SIGCOMM*. 2014.

[13] Alex Irpan. *Deep Reinforcement Learning Doesn't Work Yet*. `https://www.alexirpan.com/2018/02/14/rl-hard.html`. 2018.

[14] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. A deep reinforcement learning perspective on internet congestion control. In *International Conference on Machine Learning*. 2019.

[15] *MLPerf*. `https://mlperf.org/`.

[16] Hongzi Mao, Parimarjan Negi, Akshay Narayan, Hanrui Wang, Jiacheng Yang, Haonan Wang, Ryan Marcus, Mehrdad Khani Shirkoohi, Songtao He, Vikram Nathan, et al. Park: An Open Platform for Learning-Augmented Computer Systems. In *Advances in Neural Information Processing Systems*. 2019.

[17] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*. Springer. 1975.

[18] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The Akamai network: a platform for high-performance internet applications. In *ACM SIGOPS Operating Systems Review* 44.3 (2010).

[19] *Puffer (TV service)*. `https://en.wikipedia.org/wiki/Puffer_(TV_service)`. Wikipedia.

[20] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011.

[21] *Stanford research project improves video streaming, launches free live TV service*. Stanford Daily, 2019.

[22] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Learning *in situ*: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, 2020.

[23] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. Pantheon: the training ground for Internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, 2018.