I am a **machine learning for systems** researcher. My research co-designs machine learning and systems approaches that enhance computer system performance and resource efficiency. I have demonstrated that improving the prediction accuracy of machine learning methods does not always improve system outcomes. Instead, my approaches incorporate the unique structure of each systems problem into machine learning solutions to align the optimization goals between machine learning methods and systems problems. Meta is deploying one of my solutions (for predicting system maintenance time) on their hyperscale datacenters that serve billions of users.

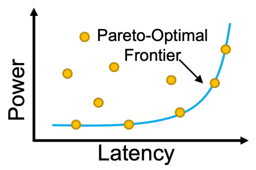
Motivation and Overview

Improving computer system performance and resource efficiency are long-standing goals. Recent approaches that use machine learning methods to achieve these goals rely on a *predictor* that predicts the latency, throughput, or energy consumption of a sub-computation to, for example, aid hardware resource management or scheduling. However, the optimization goals between machine learning methods and systems problems do not always align, and my research shows that this misalignment means that optimizing machine learning prediction accuracy does not optimize system behavior. Specifically, each system problem has a unique structure induced by the system's dynamics and optimization landscape (energy), the objective (maintenance), and the characteristics of the problem itself (interpretability). My work demonstrates that by using each of these structural characteristics to design the machine learning methods, we can achieve the system's ultimate goals.

- **Energy** [9] When the system goal is to minimize energy while meeting latency constraints, I demonstrated that improving overall prediction accuracy of latency and power does not achieve this goal because it treats all configurations equally. In fact, only Pareto-optimal resource configurations matter in minimizing energy under latency constraints. Therefore, I developed a novel approach to quickly identify configurations on the Pareto-optimal frontier, and demonstrated that improving prediction accuracy of only such Pareto-optimal configurations is enough to achieve the ultimate system goal.
- **Maintenance** [5] When the system goal is to improve maintenance efficiency of servers in datacenters, a key task is to predict the durations of maintenance jobs. However, improving the overall prediction accuracy of maintenance job durations can lead to capacity losses by taking servers offline. This is because underprediction has a much larger negative impact than overprediction, and therefore the asymmetric costs between under- and overprediction must be accounted for. I introduced a new research problem—termed maintenance job scheduling—that was not addressed by the literature, and designed a machine learning prediction system that penalizes underprediction more and overprediction less. This work is currently being deployed on Meta's hyperscale datacenters to improve maintenance efficiency of millions of servers.
- **Interpretability** [10] When the system goal is to understand why automated configuration management systems make a decision, improving prediction accuracy of performance does not help answer questions such as what factors cause low performance and why a high-performance configuration is chosen. Prior work on interpretability builds the machine learning system first, and then tries to explain the results. My research takes a different approach: building the machine learning system for interpretability from the beginning by incorporating the hierarchical structure between software, hardware, and performance into machine learning methods. My approach assists in understanding not just how each application configuration parameter influences performance, but also how that influence manifests through low-level system metrics like cache misses or context switches.

Taken together, my work demonstrates that applying machine learning to computer systems must account for the unique structure of each systems problem. Given my prior work in fundamental machine learning research [14, 8, 7, 13, 2, 15], I envision my research contributions will impact both the computer systems and machine learning communities separately, as well as bridging them together.

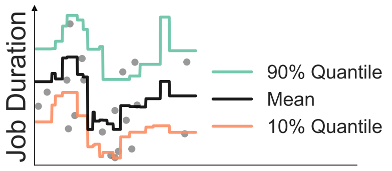
Prediction Accuracy ≠ System Goal



Energy [9] When the system goal is to minimize energy while meeting latency constraints, only configurations on the Pareto-optimal frontier in the latency-power tradeoff space matter in achieving this system goal. However, prior work did not distinguish these Pareto-optimal configurations, but instead tried to improve overall prediction accuracy of latency and power by treating all configurations equally.

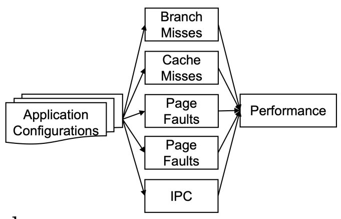
I developed a multi-phase sampling technique that increases the likelihood of sampling configurations on the optimal frontier in the latency-power tradeoff space. The key idea is to split sampling into two phases: first, separating the configurations on the optimal frontier from the rest by sampling configurations with the highest estimated energy efficiency (estimated performance divides estimated power) and second, improving prediction accuracy of configurations on the optimal frontier. While my multi-phase sampling approach does not improve the overall prediction accuracy across all configurations, because it focuses specifically on the configurations on the Pareto-optimal frontier it reduces the overall energy consumption of the system. This work demonstrates that improving overall prediction accuracy does not necessarily improve the system goal. Instead, machine learning methods have to account for the unique structure (geometry of the optimal tradeoffs in this example) of the systems problems when they are applied to computer systems.

Maintenance [5] Modern internet services are deployed on hyperscale datacenters that include millions of servers. At this scale, poor system infrastructure can cause tremendous capacity losses and impact billions of users. Datacenter operators ensure fair and regular server maintenance using automated processes to schedule maintenance jobs that must complete within a strict time budget. Automating this scheduling problem is challenging because maintenance job duration varies based on both job type and hardware. I showed that improving prediction accuracy of machine learning methods does not produce the best scheduling outcomes because underpredicting maintenance job duration results in greater server downtime than overpredicting. Therefore, when the system goal is to improve maintenance efficiency of millions of servers in datacenters, we must consider the problem structure—in this case, the asymmetric costs between under- and overprediction.



I designed Acela, a machine learning system that uses quantile regression to bias job duration predictions towards overprediction. By modeling high quantiles (i.e., quantiles greater than 50%), Acela penalizes underprediction more than overprediction, and then generates prediction values higher than those from machine learning models with higher prediction accuracy. Thus, Acela reduces the likelihood of underprediction and avoids scheduling maintenance jobs that cannot finish within the time budget, reducing server downtime. I evaluated Acela on Meta's production datacenters and compared it to machine learning methods with high prediction accuracy. Acela reduces the number of servers that are taken offline by 1.87–4.28×, and server downtime by 1.40–2.80×. Acela is currently being deployed on Meta's world-wide datacenters. This work demonstrates that prediction accuracy and maintenance efficiency are different goals, and we must consider asymmetric costs to improve maintenance efficiency.

Interpretability [10] The increasing configurability of modern software applications puts a burden on users to tune software configurations for the target hardware and workloads. While prior work has applied highly accurate black-box machine learning methods to predict performance, these methods do not help answer questions such as what factors cause low performance and why a high-performance configuration is chosen. Therefore, when the system goal is to provide interpretability of how configuration management systems work, improving prediction accuracy of performance using black-box machine learning methods does not meet the system goal.

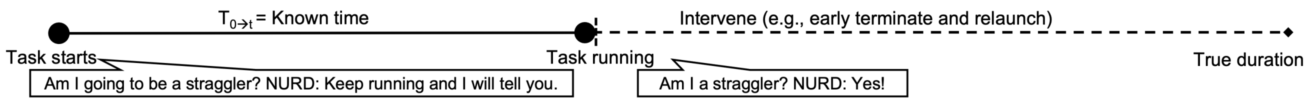


I developed a configuration management tool, GIL+, to optimize configurations and provide interpretablility. The key idea is to incorporate the hierarchical structure between software, hardware, and performance into machine learning solutions. GIL+ helps understand not just how each application configuration parameter influences high-level performance (e.g., application latency or throughput), but also how that influence manifests through low-level system metrics like cache misses or context switches. To make the results user-friendly, I developed a visualization tool to illustrate how GIL+ arrives at a high-performance configuration and how software configuration parameters, low-level system metrics, and performance interact with each other. This work demonstrates that prediction accuracy and interpretability are different goals, and we must consider the characteristics of the system itself to explain system behaviors. This work also creates an machine learning for systems solution constructed from the beginning for interpretability, rather than attempting to layer interpretability onto an existing black-box solution.

Learning System Behaviors without Prior Knowledge

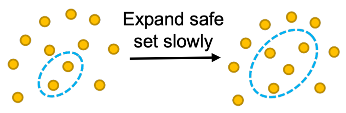
Learning datacenter straggling behaviors without prior knowledge [11] Datacenters execute large jobs that are composed of smaller tasks. A job completes when all its tasks finish, so *stragglers* (rare, yet extremely slow tasks) are a major impediment to datacenter performance. Accurately predicting stragglers enables datacenter operators to mitigate stragglers before the stragglers delay a job. While prior work has applied machine learning to predict task duration, they either rely on *complete labels* (sufficient examples of both straggling and non-straggling tasks) or offline training. For a running job, however, it is not reasonable to wait for the stragglers to reveal themselves because then they will have already delayed the job. In addition, training on finished jobs from past executions is not guaranteed to generalize to future jobs, especially new jobs that have never been executed before.

I developed NURD, an online straggler prediction method that does not rely on prior knowledge. NURD trains on finished tasks of the current running job to predict latency for running tasks. This predictor will be biased towards nonstragglers, so NURD reweights these latency predictions using a function of task features. This


$T_{0\to t}$ = Known time
Intervene (e.g., early terminate and relaunch)
Task starts
Task running
True duration
Am I going to be a straggler? NURD: Keep running and I will tell you.
Am I a straggler? NURD: Yes!

weighting function preserves latency predictions for tasks that are similar to finished tasks, and increases predicted latency for those that are different. Evaluations on two production traces from Google and Alibaba show that compared to the best baseline, NURD reduces job completion time by 2.0–8.8 percentage points.

Learning on-node safe resource control without prior knowledge [6] Cloud providers run safe resource controllers on each server node that manage resource allocations while respecting safety constraints (e.g., power threshold). Prior work has shown that safe resource control can benefit from machine learning and introduces an API for developers who deploy machine learning based resource controllers. However, this API provides no guidance on how to implement machine learning algorithms that conform to the design principles required for safety.



I developed Saxon, a safe on-node resource controller that implements the API's design principles without prior knowledge. Saxon maintains a safe set (a set of configurations that will not violate the safety constraint) and only evaluates configurations within this safe set. The key idea is that if two configurations are close in distance, their system behaviors are likely close as well, meaning that we assume configurations near safe configurations are also safe. I evaluated Saxon on the problem of minimizing latency while meeting power safety constraints. Compared to prior resource controllers and safety optimizers in other domains, Saxon reduces latency by 12.76×, the number of safety constraint violations by 55.9×, and the magnitude of safety constraint violations by 11.82×.

Vision and Future Directions: Causality, Adversarial/Strategic Resource Management, Sustainability

Vision The key takeaway of my previous research is that machine learning solutions must consider the unique structure of each systems problem to meet the system goals. I have demonstrated this point by extracting the structure of systems problems in different *controlled experimental* settings where systems researchers can actively manipulate the systems. In the real world, however, controlled experiments are not always possible due to ethical, financial, or physical reasons. My vision is to extract structures of systems problems automatically when controlled experiments are not possible, using new techniques in *observational studies*, i.e., answering research questions based purely on what we observe without controlled experimentation. If this is achieved, systems researchers can enable computer systems to make fairer decisions, more flexibly adapt to challenging experimental conditions, and intervene correctly and efficiently in high-stakes areas with long-term societal ramifications. Corresponding to these three impacts, I will discuss my future directions in rethinking computer systems using causal inference, adversary- and strategy-aware resource management, and sustainability and climate in computing as follows.

Rethinking computer systems using causal inference Causality is a unique structure that can reveal causal relationships (not mere correlations) between variables. Understanding computer systems from a causal perspective enables systems researchers to ask counterfactual questions (e.g., how would performance change if the clockspeed was increased?) and then make interventions. However, most systems researchers only resort to predictive models to capture the correlation between system conditions and performance.

My goal is to develop rigorous causal models for computer systems that permit observational studies where randomized controlled experiments may not be possible. I want to answer the following questions. First, how can we evaluate systems outcomes unbiasedly when randomized controlled experiments are not available? For example, in the problem of comparing two resource allocation policies deployed in production datacenters, Google acknowledged potential bias in their evaluations because their evaluations are based on observations in a production datacenter rather than a controlled experiment. Second, how can we uncover the causal relationships between different parts of computer systems when interference and unobserved confounders exist? This is a extremely challenging problem because the observed data distribution in the case of interference and unobserved confounders can be compatible with many contradictory causal explanations, leaving systems researchers with no way to identify the true causality.

My postdoctoral work on Improving System Efficiency and Reliability with Causal Learning has already been funded by the CRA/CCC/NSF Computing Innovation Fellows program [1]. Given my experience in building causal models with statistical guarantees and applying them to social science [2] and digital economy problems [13, 2, 4, 1],

[1]https://cccblog.org/2021/03/18/cifellows-spotlight-improving-system-efficiency-and-reliability-with-causal-learning/

[2]https://socialsciences.uchicago.edu/news/study-finds-street-activity-and-greenspace-usage-are-negatively-associated-crime

I am well equipped to assist system researchers in rethinking computer systems from a causal perspective.

Adversary- and strategy-aware resource management My previous work on resource management assumed a stochastic experimental setting. In the real world, however, the computing environments can be either adversarial or strategic. In the adversarial environment, optimization can be under adversarial attacks that negatively affect the data collection and learning procedure. In the strategic environment, multi-agent systems (multiple components of a computer system or multiple computer systems) can negotiate with each other and manipulate the learning outcomes. Nevertheless, resource management under both environments has been under-explored.

My goal is to develop adversary- and strategy-aware resource managers. I want to answer the following questions. For adversary-aware resource managers, how can we manage resources when the profiled training data are adversarial or poisoned? For strategy-aware resource managers, how can we manage resources when the strategic multi-agent systems can alter their system settings to game the learner?

Given my experience in resource management and design space exploration [9, 10, 12, 6, 3], I am passionate about expanding the problem settings to adversarial and strategic environments. I will build collaborations with security and economics researchers to model adversaries and interplay between the learner and the strategic agents.

Sustainability and climate in computing Most system outcomes I optimized in my prior work are energy efficiency and performance. Boosted energy efficiency and performance lead to increased infrastructure scale, and thus increased carbon footprint of computer systems (Jevon's Paradox). Literature has shown that carbon emissions from edge devices and datacenters are not negligible. However, carbon emission and its relations to software, hardware, and environments are under-explored in systems community.

This direction will depend on innovative, long-term thinking, since our understanding of sustainability and climate in computing is based on measurements from limited institutions and technology companies. Communication across communities that care about sustainability and climate is still lacking. In the near term, we can raise awareness of sustainability in both machine learning and systems communities, and make carbon tools more amenable to researchers. In future research, I will explore tradeoffs between performance, energy efficiency, and carbon emissions as well as design adaptive resource managers that adapt to the intermittent nature of renewable energy sources (e.g., wind, solar). I will draw on my technical background in machine learning and computer systems, and through interdisciplinary collaborations, I plan to make an impact of long-term societal ramifications.

References

[1] G. W. Basse, **Y. Ding**, and P. Toulis. Minimax designs for causal effects in temporal experiments with treatment habituation. *Biometrika*, 2022.

[2] M. Gao, **Y. Ding**, and B. Aragam. A polynomial-time algorithm for learning nonparametric causal graphs. *NeurIPS*, 2020.

[3] G. S. Ravi, P. Gokhale, **Y. Ding**, W. M. Kirby, K. N. Smith, J. M. Baker, P. J. Love, H. Hoffmann, K. R. Brown, and F. T. Chong. Cafqa: A classical simulation bootstrap for variational quantum algorithms. *ASPLOS*, 2023.

[4] K. E. Schertz, J. Saxon, C. Cardenas-Iniguez, L. Bettencourt, **Y. Ding**, H. Hoffmann, and M. G. Berman. Neighborhood street activity and greenspace usage uniquely contribute to predicting crime. *Npj Urban Sustainability*, 2021.

[5] **Y. Ding**, D. Gao, T. Ryden, K. Mitra, S. Kalmanje, Y. Golany, M. Carbin, and H. Hoffmann. Acela: Predictable datacenter-level maintenance job scheduling. 2022.

[6] **Y. Ding**, H. Kim, A. Pervaiz, H. Hoffmann, and M. Carbin. Saxon: Safe exploration for on-node resource control. 2022.

[7] **Y. Ding**, R. Kondor, and J. Eskreis-Winkler. Multiresolution kernel approximation for gaussian process regression. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[8] **Y. Ding**, C. Liu, P. Zhao, and S. C. Hoi. Large scale kernel methods for online auc maximization. In *ICDM*, 2017.

[9] **Y. Ding**, N. Mishra, and H. Hoffmann. Generative and multi-phase learning for computer systems optimization. In *ISCA*, 2019.

[10] **Y. Ding**, A. Pervaiz, M. Carbin, and H. Hoffmann. Generalizable and interpretable learning for configuration extrapolation. *ESEC/FSE*, 2021.

[11] **Y. Ding**, A. Rao, H. Song, R. Willett, and H. Hoffmann. Nurd: Negative-unlabeled learning for online datacenter straggler prediction. *Proceedings of Machine Learning and Systems (MLSys)*, 2022.

[12] **Y. Ding**, A. Renda, A. Pervaiz, M. Carbin, and H. Hoffmann. Cello: Efficient computer systems optimization with predictive early termination and censored regression. 2022.

[13] **Y. Ding** and P. Toulis. Dynamical systems theory for causal inference with application to synthetic control methods. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[14] **Y. Ding**, P. Zhao, S. Hoi, and Y.-S. Ong. An adaptive gradient method for online auc maximization. In *AAAI*, 2015.

[15] P. Wu, **Y. Ding**, P. Zhao, C. Miao, and S. Hoi. Learning relative similarity by stochastic dual coordinate ascent. In *AAAI*, 2014.