The Case for Uncertainty-Governed Predictor Hierarchies in ML for Systems

Christos Zarkos* MIT
Nevena Stojkovic* MIT
Varun Gohil MIT
Christina Delimitrou MIT

Abstract

Machine learning integration in computer systems is limited by computational overhead and lack of interpretability. While system designers often turn to surrogate models like decision trees for their speed and transparency, they suffer from accuracy loss compared to base models. To bridge this gap, we propose an uncertainty-governed model hierarchy that uses model uncertainty to trigger fallbacks from a fast, interpretable surrogate to a high-accuracy base model. We evaluate this paradigm on a resource management case study using a Bayesian Neural Network (BNN) and a surrogate decision tree. Our results show that our hierarchy maintains BNN-level accuracy while utilizing the surrogate for upto 96% of decisions. This reduces expected decision latency to 4.27ms from BNN's 31ms and provides high interpretability without settling for decreased system performance.

1 Problem and Proposed Solution

The increasing complexity of heterogeneous cloud environments and warehouse-scale computing has pushed traditional heuristic-based resource management to its limits [2]. Recent literature [3, 11] suggests a paradigm shift toward "Machine Learning (ML) for Systems", where neural networks replace hand-tuned heuristics to manage tasks such as job scheduling [10, 19], cache replacement [17, 18], and power capping [9, 14]. While these ML-based controllers demonstrate superior decision-making, their integration into production environments remains stifled by two fundamental challenges: computational overhead and poor interpretability.

In cloud systems, ML models used for management decisions are often on the critical path. The inference latency of these ML models can exceed the scheduling quantum itself, and the black-box nature of these models makes them difficult to debug or trust in critical infrastructure. To address this, prior research has explored the use of surrogate models, specifically interpretable models such as decision trees [7, 8, 12] and concept-bottleneck models [13], which try to mimic the predictions of the original base model. While these surrogate decision trees offer low latency and human-readable logic, they often fail to generalize as well as their base models.

This creates a fundamental tradeoff, system designers must choose between a fast, interpretable, but less accurate surrogate, or a slow, opaque, but highly accurate base model. In computer architecture, such tradeoffs are traditionally resolved through hierarchical design, such as the multi-level memory hierarchy. However, applying a hierarchy to ML predictors is non-trivial. Unlike a CPU cache, where a "miss" is objectively determined by the absence of data, an ML "miss" (an incorrect prediction) cannot be identified at inference time because the ground truth is unavailable.

Prior work [6, 16] has demonstrated that a model's uncertainty can be a reliable proxy for its own accuracy. This finding suggests that while a surrogate may be less accurate on average, its mispredictions are proactively detectable. Furthermore, this uncertainty can be estimated in a lightweight manner parallel to model inference with negligible computational overhead.

We propose using a hierarchy of fast, interpretable surrogates and slow, opaque base models in decision making, which uses uncertainty to decide when to fallback from one level of the hierarchy to another. Using such a hierarchy of predictors, our proposal achieves the benefits of both the surrogate and base models by minimizing computational overhead and maximizing interpretable decision-making. Figure 1 shows one instantiation of our proposed hierarchy.

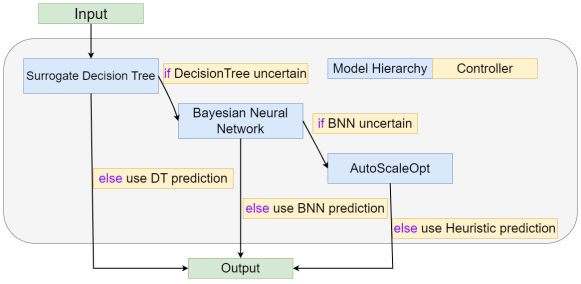
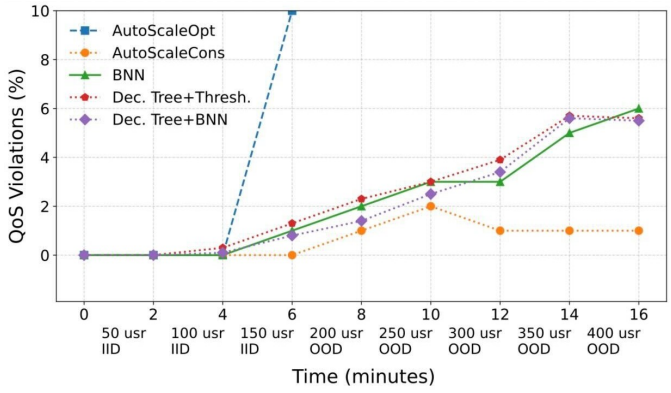


Figure 1. Control flow in the hierarchy of our case study.

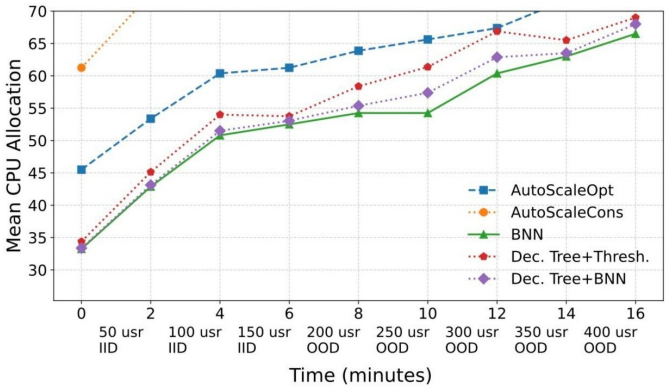
2 Resource Management Case Study

As a case study, we use an ML-driven resource manager for microservices presented by prior work [16, 19]. The original work [16] uses a Bayesian Neural Network (BNN) to predict the immediate tail latency and probability of delayed Quality-of-Service (QoS) violations. The BNN runs as a centralized scheduler and takes a proposed resource configuration, past resource history, and latency as inputs, which it collects using distributed per-node agents. It runs in 1-second intervals and uses Linux cgroups to enforce the predicted optimal resource configuration. The BNN utilizes Bayesian uncertainty to predict when the model is highly uncertain and falls back

*Both authors contributed equally.



(a) QoS violations of the workflows.



(b) Resource allocation by the workflows.

Figure 2. QoS violations and CPU allocation of resource management workflows on increasing social network users.

to the AutoScalerOpt heuristic, which follows AWS's recommended autoscaling policy based on CPU utilization [1].

Importantly, the BNN has a high computational overhead. It has a latency of 31ms per inference (Table 1). While the latency might not seem high, given the 1-second decision-making interval, it limits the number of resource configurations that can be evaluated to less than 33 ($\approx$ 1000/31). Furthermore, it consumes a non-trivial amount of memory, 760MB.

To reduce the computational overhead, we learn a surrogate decision tree of the BNN following the approach suggested by prior work [12]. We train the decision tree to have 100 input features, the same as the BNN, and 2 output features – the predicted tail latency and a predicted probability of QoS violation, again the same as the outputs of the BNN. Our surrogate decision tree has a fidelity of 94%, which means that the decision tree exactly mimics the predictions of the BNN for 94% of the training data. Crucially, the surrogate decision tree is significantly faster than the BNN and has an inference latency of 0.27 ms (Table 1), while consuming only 95MB of memory.

We swap the BNN in the resource manager with the decision tree. For a fair comparison, we include a fallback to AutoScaleOpt using distance-based uncertainty when using the surrogate decision tree. We compute the distance-based uncertainty on the input vector and set the threshold for uncertainty to the 90^{th} percentile of distances seen on training data, similar to prior work [16]. Figure 2 (`Dec. Tree + Thresh`) shows that despite the high fidelity and low computational cost of the decision tree, directly replacing the BNN with the surrogate decision tree is not a good option, since we would have higher QoS violations while also having higher mean CPU allocation.

This is exactly where our proposal comes in, allowing us to combine the speed and efficiency of the decision tree with the high accuracy and robustness of the BNN. Figure 1 shows our model hierarchy, where the input initially goes to the decision tree. If the decision tree is deemed uncertain, the decision falls back to the BNN which, if in turn is also deemed uncertain, will fall back to the heuristic approach of AutoScaleOpt.

We evaluate the different resource management workflows by deploying the Social Network application from DeathStarBench [5] on a 7-machine CloudLab cluster [4]. We use the wrk2 load generator [15] to send 'Read-HomeTimeline' requests to the application. We also include two baselines, AutoScaleOpt and AutoScaleCons. AutoScaleCons is a more conservative variant of AutoScaleOpt, optimized for QoS preservation. In our experiment, we incrementally increase the SocialNetwork user load every 2 minutes. For the first 6 minutes (50-150 users), the users create in-distribution data, while for the rest of the time ($\geq$ 150 users), the load exceeds the training data and is out-of-distribution (OOD).

Figure 2 shows the results of all the resource management workflows. Our proposed model hierarchy has similar QoS violations to the BNN workflow (0.5% difference) overall and even has fewer QoS violations for lower loads. At the same time, it allocates only 5% higher CPUs in the worst case. More importantly, we find that the decision tree is used for 87%-96% of the decisions made, depending on the load, with lower load resulting in higher usage. This shows that using our proposed hierarchy, we can achieve similar results as the BNN workflow, while having 87% interpretable decision-making with a worst-case expected decision-making latency of 4.27ms.

Table 1. Comparison between Bayesian Neural Network (BNN) and Surrogate Decision Tree

Metric	BNN	Surrogate Decision Tree
Total parameters	1,483,251	11,903
Latency (ms/inference)	31	0.27
Throughput (inf/sec)	32.3	3705.67
Memory usage (MB)	760	95
CPU utilization (%)	98.3	94

References

[1] Amazon-EC2-Auto-Scaling-User-Guide. Step and simple scaling policies for Amazon EC2 Auto Scaling. https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scaling-simple-step.html. Accessed: 2025-10-20.

[2] Ricardo Bianchini, Marcus Fontoura, Eli Cortez, Anand Bonde, Alexandre Muzio, Ana-Maria Constantin, Thomas Moscibroda, Gabriel Magalhaes, Girish Bablani, and Mark Russinovich. Toward ml-centric cloud platforms: Opportunities, designs, and experience with microsoft azure. *Communications of the ACM*, 63(2), February 2020.

[3] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 153–167, New York, NY, USA, 2017. Association for Computing Machinery.

[4] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. The design and operation of CloudLab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 1–14, Renton, WA, July 2019. USENIX Association.

[5] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayantara Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Yuan He, and Christina Delimitrou. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud and Edge Systems. In *Proceedings of the Twenty Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, April 2019.

[6] Varun Gohil, Sundar Dev, Gaurang Upasani, David Lo, Parthasarathy Ranganathan, and Christina Delimitrou. The importance of generalizability in machine learning for systems. *IEEE Computer Architecture Letters*, 23(01):95–98, January 2024.

[7] Qinghao Hu, Harsha Nori, Peng Sun, Yonggang Wen, and Tianwei Zhang. Primo: Practical Learning-Augmented systems with interpretable models. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 519–538, Carlsbad, CA, July 2022. USENIX Association.

[8] Qinghao Hu, Harsha Nori, Peng Sun, Yonggang Wen, and Tianwei Zhang. Primo: Practical Learning-Augmented systems with interpretable models. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 519–538, Carlsbad, CA, July 2022. USENIX Association.

[9] Alok Kumbhare, Reza Azimi, Ioannis Manousakis, Anand Bonde, Felipe Vieira Frujeri, Nithish Mahalingam, Pulkit Misra, Seyyed Ahmad Javadi, Bianca Schroeder, Marcus Fontoura, and Ricardo Bianchini. Prediction-based power oversubscription in cloud platforms. In *Proceedings of the USENIX Annual Technical Conference (ATC)*. USENIX, July 2021. Earlier version published as arXiv:2010.15388, October 2020.

[10] Jianheng Ling, Pratik Worah, Yawen Wang, Yunchuan Kong, Chunlei Wang, Clifford Stein, Diwakar Gupta, Jason Behmer, Logan A Bush, Prakash Ramanan, et al. Lava: Lifetime-aware vm allocation with learned distributions and adaptation to mispredictions. *Proceedings of Machine Learning and Systems*, 7, 2025.

[11] Martin Maas. A taxonomy of ml for systems problems. *IEEE Micro*, 40(5):8–16, 2020.

[12] Zili Meng, Minhu Wang, Jiasong Bai, Mingwei Xu, Hongzi Mao, and Hongxin Hu. Interpreting deep learning-based networking systems. 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*, SIGCOMM '20, page 154–171, New York, NY, USA, 2020. Association for Computing Machinery.

[13] Sagar Patel, Dongsu Han, Nina Narodytska, and Sangeetha Abdu Jyothi. Agua: A concept-based explainer for learning-enabled systems. In *Proceedings of the ACM SIGCOMM 2025 Conference*, SIGCOMM '25, page 329–346, New York, NY, USA, 2025. Association for Computing Machinery.

[14] Christina Delimitrou Shuang Chen, Angela Jin and Jose Martinez. ReTail: Opting for Learning Simplicity to Enable QoS-Aware Power Management in the Cloud. In *Proceedings of the 28th IEEE International Symposium on High-Performance Computer Architecture (HPCA-28)*, February 2022.

[15] Gil Tene. wrk2: A constant throughput, correct latency recording variant of wrk. https://github.com/giltene/wrk2, 2015.

[16] Noman Bashir Sundar Dev Gaurang Upasani David Lo Parthasarathy Ranganathan Varun Gohil, Nevena Stojkovic and Christina Delimitrou. When machine learning isn't sure: Building resilient ml-based computer systems by embracing uncertainty. *Proceedings of Machine Learning and Systems*, 8, 2026.

[17] Giuseppe Vietri, Liana V Rodriguez, Wendy A Martinez, Steven Lyons, Jason Liu, Raju Rangaswami, Ming Zhao, and Giri Narasimhan. Driving cache replacement with {ML-based}{LeCaR}. In *10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18)*, 2018.

[18] Dongsheng Yang, Daniel S Berger, Kai Li, and Wyatt Lloyd. A learned cache eviction framework with minimal overhead. *arXiv preprint arXiv:2301.11886*, 2023.

[19] Yanqi Zhang, Weizhe Hua, Zhuangzhuang Zhou, Edward Suh, and Christina Delimitrou. Sinan: ML-Based and QoS-Aware Resource Management for Cloud Microservices. In *Proceedings of the Twenty Sixth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, April 2021.