MISO: Model-Internal-State-Guided Optimization for Ranking Models

Yongzhe Zhang* yongzhe@meta.com Meta Inc United States

Xiaoyu Deng* xiaoyud@meta.com Meta Inc United States

Yifan He* yifanhe1999@meta.com Meta Inc United States

Mengying Sun* mengyingsun@meta.com Meta Inc United States

Yijia Liu yijialiu@meta.com Meta Inc United States

Sheng Luo shengluo@meta.com Meta Inc United States

Marcio Porto mflporto@meta.com Meta Inc United States

Anish Khazane akhazane@meta.com Meta Inc United States

Zhuo Li zlli@meta.com Meta Inc United States

Huiping Yao huiping@meta.com Meta Inc United States

Swathi Hrishikesh swathih@meta.com Meta Inc United States

Jing Chen jingchenfb@meta.com Meta Inc United States

Dennis Choi dennchoi@meta.com Meta Inc United States

Steven Liu stevenliu0305@meta.com Meta Inc United States

Xiaoya Wang xiaoya213@meta.com Meta Inc United States

Emmy Wang emmyw@meta.com Meta Inc United States

Kangfu Zheng zhengkf02@meta.com Meta Inc United States

Xingyuan Wang sylviawang@meta.com Meta Inc United States

Zhiwen Chen kylechen@meta.com Meta Inc United States

Yang Jin yangjin@meta.com Meta Inc United States

Haoyu Zhou haoyuzhou@meta.com Meta Inc United States

Lexi Luo lexiluo@meta.com Meta Inc United States

Keyi Chen keyic@meta.com Meta Inc United States

Hao Yan yanh@meta.com Meta Inc United States

Peggy Yao peggy@meta.com Meta Inc United States

Alireza Vahdatpour alirezav@meta.com Meta Inc United States

Santanu Kolay skolay@meta.com Meta Inc United States

Yi Meng ymeng@meta.com Meta Inc United States

Bilal Fadlallah bhf@meta.com Meta Inc United States

Gursharan Singh gurshi@meta.com Meta Inc United States

Prabhakar Goyal prgoyal@meta.com Meta Inc United States

Abstract

Ranking models are repeatedly refined within established model families, yet the choice of which component to scale, replace, or retire is often guided by expensive trial-and-error. We present **Model Internal State Optimization (MISO)**, a systems workflow that uses model internal states (MIS)-including parameters, activations, gradients, and normalization statistics—to prioritize such local optimization decisions. MISO extracts MIS from a trained ranking model, aggregates them into ranking, alignment, and comparison signals, and converts those signals into a small set of interpretable candidate edits. Because MIS are re-extracted after each retraining cycle, MISO naturally supports an adaptive optimization workflow that tracks evolving model behavior as data distributions and system requirements shift over time. In an ads ranking case study, MISO improves normalized entropy while requiring substantially fewer validation runs than expert-driven and black-box scaling workflows, offering a practical middle ground between manual tuning and opaque automated search.

OARS '26, Minneapolis, MN, USA
2026.

Keywords

recommender systems, ranking models, model optimization, model internal states, ads machine learning

1 Introduction

Large-scale ranking and recommendation systems are rarely redesigned from scratch. More often, engineers repeatedly make local decisions within an established model family: where to add capacity, which module to replace, which feature pathway to retire, or which nearby variant merits another expensive training run. These decisions must balance ranking quality, latency, and operational cost, but are still commonly driven by manual inspection and trial-and-error.

Although trained models already expose a rich collection of internal signals—parameters, activations, gradients, normalization statistics, and intermediate representations—these signals are seldom used as a systematic optimization interface. End-to-end metrics reveal whether a variant improved, but do not directly indicate where a model is under-utilized, which representations are unstable, or why two nearby variants diverge.

Turning internal signals into reliable engineering actions is difficult. MIS are high-dimensional and heterogeneous, while the available actions are coarse-grained and structural. A useful system must therefore summarize low-level evidence into a small number of proposals that engineers can inspect, validate, and reject under a limited experimentation budget.

Existing workflows leave a gap between expert-driven tuning and black-box AutoML. The former can exploit domain knowledge but is difficult to reproduce; the latter can automate exploration but typically treats a model as an opaque object and may require many costly trials. Neither approach makes the model's own internal evidence a first-class input to the optimization loop.

We present **MISO**, a machine learning systems framework for internal-state-guided optimization of ads ranking models. MISO closes the loop between internal behavior and architectural actions: it extracts MIS, aggregates them into actionable summaries, and uses those summaries to prioritize a small set of candidate modifications for validation.

MISO provides (1) a unified abstraction and extraction interface for MIS, (2) ranking, alignment, and comparison primitives that turn fine-grained signals into decision-oriented summaries, and (3) a budgeted decision loop that validates only the most promising edits. It shares AutoML's aim of reducing manual effort, but targets repeated local refinement rather than broad architecture discovery.

We evaluate MISO in an ads ranking case study spanning several model scales and three recurring optimization tasks. The results show that MIS-guided proposals can improve ranking quality while reducing the number of training runs needed to identify a useful configuration.

Contributions. Our main contributions are:

- We introduce **MISO**, a systems framework that treats model internal states as first-class optimization signals for local ranking-model refinement.
- We design three classes of MIS aggregation primitives (ranking, alignment, comparison) that bridge fine-grained signals to actionable architectural decisions.
- We demonstrate on ranking models that MISO achieves up to 2.5× the relative NE improvement of expert-driven tuning while reducing exploration cost by 84–94%.

2 Problem Statement and Design Goals

2.1 Problem Statement

We consider *local model optimization guided by model internal states* (MIS). Given a trained ranking model M with parameters θ, a training or evaluation snapshot induces internal signals such as parameter tensors, activations, gradients, normalization statistics, and neuron-level importance measures. We denote this collection by $\mathcal{S}(M)$.

The objective is to use $\mathcal{S}(M)$ to prioritize architectural and capacity-related edits—such as selective scaling, pruning, module replacement, and feature reconfiguration—under a limited experimentation budget. Rather than searching a broad architecture space, MISO seeks a short sequence of validated modifications $\{\Delta M\}$ that improves the target ranking metric while respecting operational constraints such as latency and training cost.

This problem is difficult because MIS are high-dimensional, heterogeneous, and highly localized, whereas optimization actions are coarse-grained and structural. To bridge this gap, MISO is designed around the following goals:

2.2 Design Goals

G1: Unified MIS Abstraction. The system should provide a unified abstraction for diverse MIS types across different model architectures and training stages, enabling consistent extraction, storage, and downstream analysis.

G2: Actionable Signal Aggregation. Given the fine-grained nature of MIS, the system should aggregate low-level signals into interpretable, decision-oriented summaries that can directly inform architectural actions, rather than exposing raw statistics.

G3: Closed-Loop Optimization. The system should support a closed optimization loop that connects MIS analysis to concrete model modifications and evaluates their impact, enabling iterative refinement without manual intervention.

G4: Low Exploration Cost. Compared to black-box AutoML, the system should significantly reduce the number of training or evaluation runs required to reach high-quality model configurations.

G5: Generality and Interpretability. The framework should generalize across model families and optimization tasks, while producing human-interpretable signals that help engineers understand *why* certain optimizations are effective.

3 Related Work

Our work on MIS-driven optimization is situated at the intersection of black-box AutoML, model interpretability, and network pruning. AutoML and NAS methods automate model selection and architecture search, but they typically operate using only external performance metrics [2, 5, 6, 9, 15]. This black-box perspective leads

*Equal Contribution.

to high exploration cost and limited interpretability, especially in industrial model optimization workflows.

A second line of work studies model explanations and feature attribution. Methods such as LIME, SHAP, and Integrated Gradients provide valuable insight into how models make predictions [7, 10, 11]. However, these methods are primarily designed to explain model behavior to humans rather than to drive automated model-edit decisions.

Finally, pruning and compression research demonstrates that internal signals can reveal under-utilized capacity and guide architecture refinement [3, 4, 13]. MISO differs from these approaches by treating ranking, alignment, and comparison as reusable primitives inside a broader system for practical model optimization rather than a single-purpose compression technique.

Ranking Architectures. Our work is applied to ranking architectures that have evolved from early deep recommendation models [1, 8] through increasingly modular designs such as DCN-V2 [12] and stacked interaction architectures [14]. These systems are characterized by heterogeneous feature types, large embedding tables, and frequent iteration cycles under tight latency budgets—properties that make undirected exploration expensive and MIS-guided optimization particularly attractive.

4 Architecture Overview

MISO is designed as a general-purpose ML systems framework that closes the loop between model internal states (MIS) and model optimization actions. As shown in Figure 1, the system consists of three logical layers: (1) MIS Extraction Layer, (2) MIS Analysis and Aggregation Layer, and (3) Optimization and Decision Layer. Together, these layers transform fine-grained internal signals into actionable, low-cost model modifications.

4.1 MIS Extraction Layer

The MIS Extraction Layer provides a unified and extensible interface for collecting internal signals during offline training and post-training analysis. This layer abstracts away model- and framework-specific details and exposes a consistent representation of MIS, denoted as $\mathcal{S}(M)$.

MISO supports a wide range of MIS types, including model parameters, activations, gradients, normalization statistics, and neuron- or module-level metrics. Extraction can be configured at different granularities and time scales. By decoupling MIS extraction from downstream analysis, MISO enables the same optimization logic to be reused across different model architectures and workloads.

4.2 MIS Analysis and Aggregation Layer

Raw MIS are high-dimensional, heterogeneous, and often too fine-grained to directly inform optimization decisions. The MIS Analysis and Aggregation Layer bridges this abstraction gap by transforming $\mathcal{S}(M)$ into *actionable summaries* through a set of MIS aggregation primitives.

Each aggregation primitive implements a mapping $f : \mathcal{S}(M) \rightarrow \mathcal{A}$, where $\mathcal{A}$ represents decision-oriented signals such as rankings, scores, or constraints. We instantiate three classes of primitives.

Ranking-based primitives summarize MIS into importance or ROI scores. A representative instantiation is neuron importance, where we combine gradient-based and perturbation-based signals:

$$I_n = \alpha \cdot \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} |a_n \cdot \nabla_{a_n} \mathcal{L}(x)| + (1 - \alpha) \cdot \Delta \mathcal{L}|_{a_n \leftarrow \text{shuffle}(a_n)} \quad (1)$$

where $\mathcal{L}$ is the loss, $\mathcal{D}$ is validation data, and α is a balancing hyperparameter.

Alignment-based primitives diagnose whether intermediate distributions deviate from a layer-specific reference. For normalized layers, a useful reference is the intended standardized distribution. For a given layer l, we compute the following diagnostic score:

$$\text{Align}(l) = 1 - \frac{D_{KL}(p^l_{pre} || \mathcal{N}(0,1)) + D_{KL}(p^l_{post} || \mathcal{N}(0,1))}{2} \quad (2)$$

Here, p^l_{pre} and p^l_{post} denote empirical pre- and post-normalization distributions. The score is used to rank layers for inspection rather than as a universal measure of representation quality. **Comparison-based primitives** contrast MIS across models, layers, or training stages to surface anomalies and identify where candidate variants diverge.

4.3 Optimization and Decision Layer

The Optimization and Decision Layer translates aggregated MIS signals into concrete model modification actions. Actions are defined over a configurable action space, including parameter scaling, pruning, module replacement, and feature reconfiguration.

MISO implements a closed-loop workflow. Given an initial model, MIS are extracted and aggregated to produce actionable signals, which are then mapped to a small set of candidate modifications. These candidates are evaluated under budget constraints, and the resulting feedback is used to refine the next round of decisions.

5 MIS Instantiations in Ranking Models

We instantiate MISO on large-scale ranking models to demonstrate how the framework maps model internal states to optimization actions in a real-world workload. Ranking models present a representative and challenging setting due to their heterogeneous architectures, large parameter counts, and stringent efficiency constraints.

5.1 Target Model and Optimization Scope

The target workload consists of deep neural ranking models used in recommendation and retrieval systems. These models combine embedding layers, feature-interaction modules, multi-layer perceptrons, and task-specific heads, resulting in tens to hundreds of millions of parameters. Figure 2 shows a representative architecture used in our study. Within this setting, MISO focuses on optimization tasks that are common yet costly in practice: capacity scaling, parameter pruning, architectural refinement, and module replacement.

The optimization objectives include model quality metrics such as normalized entropy together with system-level constraints such as computational efficiency and training cost. Importantly, MISO operates under limited exploration budgets, where exhaustive search or brute-force AutoML is infeasible. This makes ranking models a

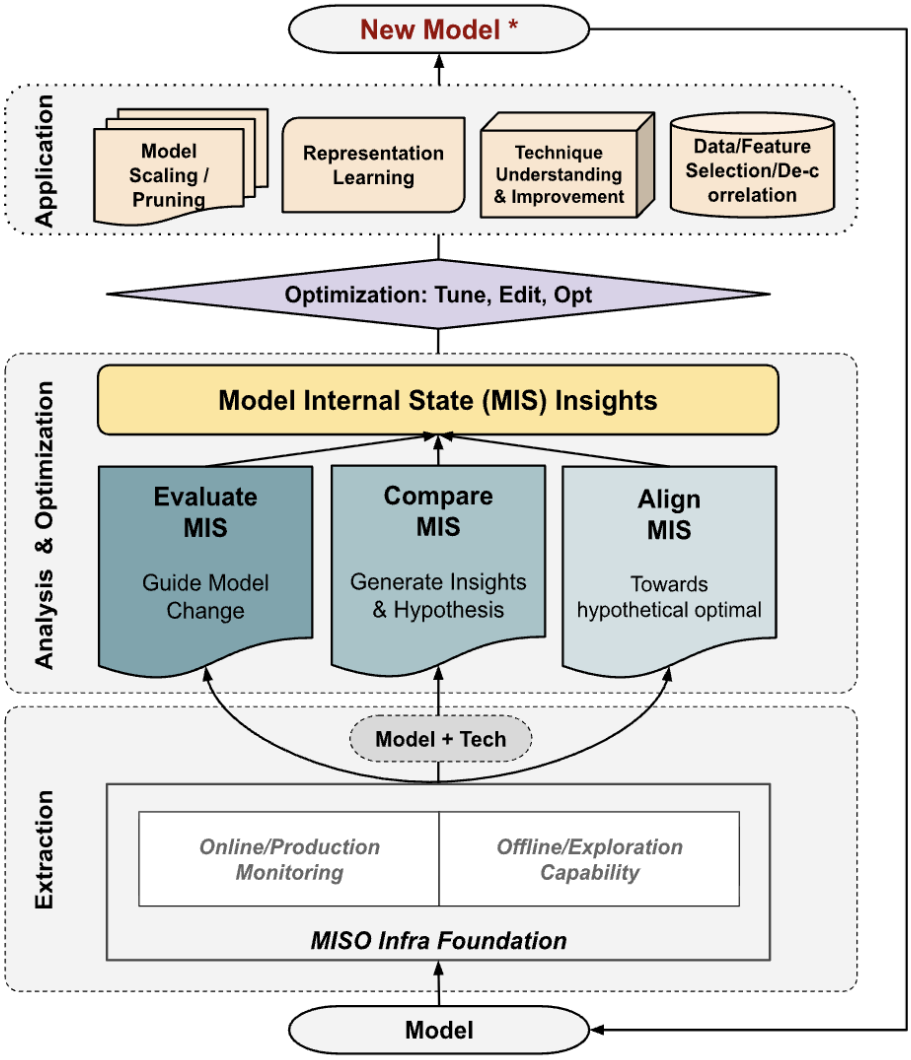

Figure 1: The MISO system architecture. The framework consists of three layers: (1) an Extraction layer that collects Model Internal States (MIS), (2) an Analysis & Optimization layer that evaluates, compares, and aligns MIS to generate actionable insights, and (3) an Application layer where these insights drive concrete model optimizations such as scaling, pruning, representation learning, and feature selection.

suitable stress test for MIS-guided optimization: the search space is large enough to make undirected exploration expensive, yet the engineering workflow still expects interpretable, modular decisions.

5.2 Collected MIS Types and Action Mapping

To support these optimization tasks, MISO instantiates MIS extraction for a representative set of internal signals in ranking models. Specifically, MISO collects: (1) neuron- and module-level parameter statistics, including weights and gradients; (2) activation statistics at key interaction and transformation layers; (3) normalization-related signals such as mean and variance before and after normalization layers; and (4) feature- and module-level response statistics that reflect sensitivity to perturbations.

These signals are mapped to optimization actions through the aggregation primitives. Ranking-based MIS are used primarily for selective scaling and pruning; alignment-based MIS are used to identify unstable normalization or representation modules; and comparison-based MIS are used to explain performance differences across nearby variants and support architectural refinement.

6 Experimental Setup

We evaluate MISO on large-scale ranking models as a systems case study. We ask whether MIS-guided proposals improve ranking quality under a bounded validation budget, whether they reduce the number of full training runs, and how the three MIS primitives contribute to the resulting recommendations.

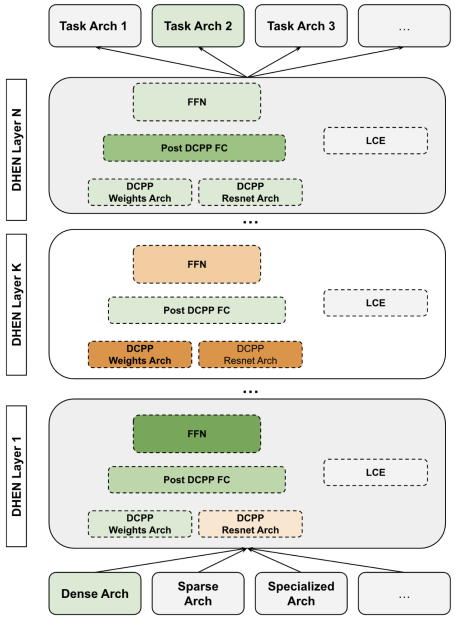

Figure 2: Representative ranking model architecture used to instantiate MISO. The model combines multiple feature streams and stacked submodules, creating several natural decision points for MIS-guided scaling, replacement, and refinement.

6.1 Workloads, Baselines, and Metrics

The evaluation uses recommendation-style ranking workloads [1, 8]. The dataset comprises billions of training examples with both dense and sparse features, representative of ads recommendation workloads. We compare MISO with two practical baselines. **Black-box scaling** expands model capacity without MIS guidance. **Expert-driven tuning** reflects a standard workflow in which engineers choose the next experiment from end-to-end metrics, domain knowledge, and available diagnostic signals.

We study three recurring optimization tasks: capacity scaling based on neuron or module importance, replacing poorly aligned normalization blocks, and refining architectural choices using cross-model comparison signals. We report Normalized Entropy (NE) improvement relative to black-box scaling together with the number of training runs required to reach a satisfactory configuration. Normalized Entropy (NE) measures the predictive calibration of a ranking model relative to the empirical base rate; lower NE indicates better ranking quality.

6.2 Evaluation Protocol and Fairness

Our goal is not to ask which method can discover the best model after unlimited search, but which workflow reaches a useful proposal under a realistic engineering budget. We therefore report both final quality and the number of training runs required to reach the selected configuration. This framing reflects the practical setting in which retraining and validation dominate the cost of local model refinement.

Table 1: Main results across model scales. NE improvements are reported as relative ratios over the expert-driven baseline.

Metric	13x	20x	32x	50x
Relative NE improvement (MISO vs. expert-driven)	2.5×	2.0×	2.0×	2.0×
Expert-driven runs	50±10	62±12	50±10	92±18
MISO runs	**3±1**	**5±2**	**8±2**	**12±3**
Run reduction	94%	92%	84%	87%

MISO incurs additional analysis work to extract MIS, but all methods are evaluated against the same downstream objective and deployment constraints. We do not claim that internal-state analysis is free. Instead, the case study evaluates whether this upfront analysis is a favorable tradeoff when the dominant cost is the number of full model trials. The reported run counts should therefore be interpreted as an engineering-efficiency measure, not as a claim that MISO eliminates all optimization cost.

6.3 Representative Optimization Cases

MISO is designed to support several recurring optimization patterns rather than a single fixed heuristic. In our ranking workloads, ranking-based signals are most useful for selective scaling, because they identify where extra capacity is likely to pay off. Alignment-based signals are most useful when the issue is unstable intermediate statistics rather than raw capacity, motivating normalization-module replacement or related refinements. Comparison-based signals become valuable when nearby variants differ subtly and engineers need to isolate where their internal behavior diverges.

These three cases illustrate why MIS are useful in practice. The system is not replacing engineering judgment with an opaque score. Instead, it narrows the decision space using evidence that remains interpretable at the module or architectural level.

7 Results

7.1 Main Results

Table 1 summarizes the quality and engineering-efficiency results. Across the evaluated model scales, MISO yields larger relative NE improvements than expert-driven tuning while using substantially fewer validation runs. At the 50x scale, the relative NE improvement obtained with MISO is 2× that obtained with expert-driven tuning.

These gains are accompanied by a large reduction in the number of evaluated configurations. Across the reported scales, MISO requires 3–12 training runs versus 50–92 for expert-driven tuning, a reduction of 84–94%. In this case study, internal-state analysis is useful not only for selecting a better configuration but also for reducing the search budget required to reach it.

7.2 Contribution of Different MIS Primitives

Different MIS types support different optimization actions. Ranking-based signals are most useful for selective scaling. Alignment-based signals are most helpful when the issue is unstable normalization or representation statistics. Comparison-based signals become valuable when deciding between nearby architectural variants and isolating the source of performance differences.

This pattern appears in the ablation study at the 50x scale shown in Table 2. Using ranking information alone already improves

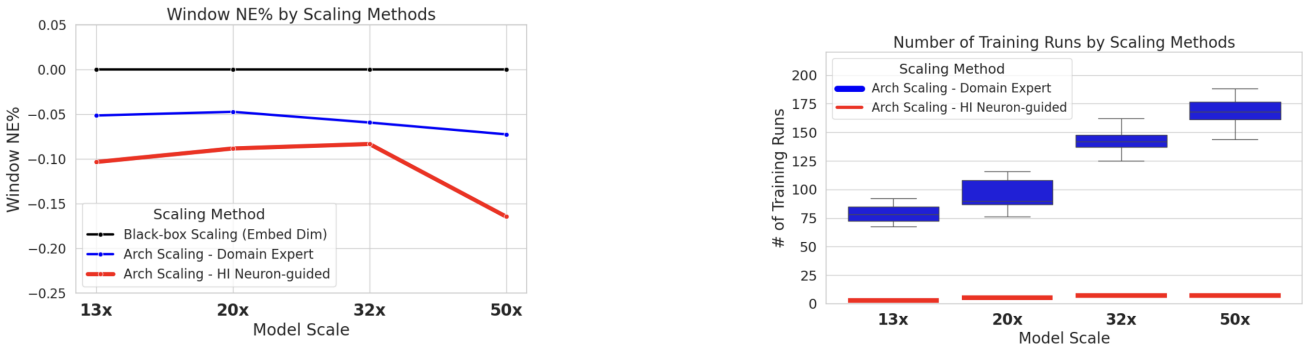

Figure 3: Scaling results across model sizes. Left: NE improvement relative to black-box scaling. Right: Training runs required. MISO consistently improves quality while sharply reducing exploration cost.

Table 2: Ablation study on MIS primitives (50x scale). Relative NE gain is normalized to the full MISO configuration (=100%).

Configuration	Relative NE gain	Params	Latency
Baseline	0%	100%	100%
+ Ranking only	62%	100%	100%
+ Ranking + Alignment	62%	100%	92%
+ Ranking + Comparison	75%	95%	92%
Full MISO	100%	95%	92%

quality, but adding alignment and comparison signals yields the strongest end-to-end recommendations and modest efficiency gains in model size and latency. The result is not that any one primitive is universally sufficient, but that the combination forms a reusable interface for several common optimization actions.

7.3 Takeaways

From a systems perspective, the key result is that internal-state analysis can shrink the validation budget for local model optimization without turning the process into an opaque search loop. MISO does not attempt to replace all of AutoML or solve general NAS. Instead, it offers a practical middle ground between standard expert workflows and black-box exploration: engineers choose the optimization scope, while MISO supplies ranked, interpretable proposals that reduce wasted trials.

This positioning is especially relevant for ranking and recommendation systems, where absolute metric gains are often incremental but can still justify deployment if the path to those gains is inexpensive and interpretable. In such settings, saving dozens of training runs can be as important as the final performance delta itself.

8 Discussion

8.1 Operational Interpretation

MISO is most useful in settings where the model family is already fixed and the real problem is repeated local optimization. In such workflows, engineers are rarely deciding among radically different architectures; instead, they are deciding where to add capacity, which module to replace, or which nearby variant is worth another expensive run. MISO matches this decision regime well because it produces recommendations at the same granularity as these engineering actions.

The framework is also attractive because it preserves interpretability. A recommendation produced by MISO is accompanied by a concrete internal signal pattern—for example, under-utilized neurons, poorly aligned normalization statistics, or a divergence between two variants at a specific layer. This is operationally useful even when the final metric gain is modest, because it provides a rationale that can be audited by practitioners.

8.2 When MISO Helps Less

MISO is less helpful in settings where a model has not yet reached a usable baseline, where internal signals are too noisy to be stable, or where the dominant question is broad architecture discovery rather than targeted refinement. In these regimes, black-box exploration or other search procedures may still be necessary. The intended contribution of MISO is therefore not to replace all forms of AutoML, but to introduce a practical optimization layer for an important class of repeated engineering decisions.

9 Limitations and Future Work

MISO should be read as a practical optimization system rather than a claim of universal algorithmic superiority. First, the evaluation is a ranking-model case study, and broader validation across domains and model families remains future work. Second, comprehensive MIS extraction introduces analysis overhead, especially for perturbation-based signals. Third, the current evidence concerns post-hoc refinement of trained models rather than cold-start search over entirely new architectures. Finally, the reported results are aggregate operational measurements from a proprietary deployment setting; they should not be interpreted as a fully reproducible benchmark or as a statistical claim about every ranking workload.

Future work includes extending the framework to additional model families, reducing the cost of MIS extraction, and using language-model-based agents to assist the interpretation of internal-state patterns.

Reproducibility

Due to deployment constraints on the underlying models and data, we cannot release the full codebase; the paper is therefore positioned as a systems and application case study, with algorithms and decision workflow described in enough detail to support adaptation on similar workloads.

10 Conclusion

We presented **MISO**, a system for model-internal-state (MIS)-guided refinement of ads ranking models. MISO extracts and aggregates diverse internal signals—including neuron importance, activation alignment, and cross-model comparison—to guide model modifications in an interpretable manner.

Across the evaluated ranking-model scales, MISO improves normalized entropy while requiring only a small fraction of the training trials used by expert-driven tuning. More broadly, the case study suggests that model internal states can serve as a practical optimization interface for repeated real-world ranking-model refinement.

References

[1] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198.

[2] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

[3] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.

[4] Song Han, Jeff Pool, John Tran, and William J Dally. 2015. Learning Both Weights and Connections for Efficient Neural Networks. In *Advances in Neural Information Processing Systems 28*. 1135–1143.

[5] Manas R. Joglekar, Cong Li, Jay K. Adams, Pranav Khaitan, and Quoc V. Le. 2019. Neural Input Search for Large Scale Recommendation Models. arXiv:1907.04471 [cs.LG] https://arxiv.org/abs/1907.04471

[6] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1eYHoC5FX

[7] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*. 4765–4774.

[8] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *arXiv preprint arXiv:1906.00091* (2019).

[9] Esteban Real, Chen Liang, David R. So, and Quoc V. Le. 2020. AutoML-Zero: evolving machine learning algorithms from scratch. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 742, 13 pages.

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1135–1144.

[11] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*. 3319–3328.

[12] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-Scale Learning to Rank Systems. In *Proceedings of the Web Conference (WWW)*. 1785–1797.

[13] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. 2018. NISP: Pruning Networks Using Neuron Importance Score Propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9194–9203.

[14] Liu Zhang, Rui Chen, Ruigang Zhao, Huifeng Guo, Ying Li, Zhenhua Zheng, Ruiming Tang, and Xiuqiang He. 2022. DHEN: A Deep and Hierarchical Ensemble Network for Large-Scale Click-Through Rate Prediction. In *arXiv preprint arXiv:2203.11014*.

[15] Barret Zoph and Quoc Le. 2017. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1Ue8Hcxg