DAMPED GAUSS–NEWTON SEARCH FOR MULTI-METRIC HYPERPARAMETER OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We study hyperparameter optimization (HPO) from a numerical-optimization perspective and propose a black-box, target-seeking damped Gauss–Newton method for multiple validation responses. In contrast to uninformed candidate search such as grid or random search, the method uses numerical sensitivity information inferred from previously observed changes in hyperparameters and validation performance. Successive full-vector observations are used to construct an iterative secant approximation of the local Jacobian, which relates changes in the performance vector to changes in all optimized hyperparameters; a damped Gauss–Newton system then produces a joint update of the complete hyperparameter vector rather than optimizing coordinates independently. The method does not differentiate through model training or require dedicated coordinate-wise perturbation runs: following the initial two evaluations used to establish the first secant, each subsequent iteration requires one new full-vector evaluation. Tikhonov regularization stabilizes the resulting local inverse problem when the number of optimized parameters exceeds the number of performance responses. Unlike conventional multi-objective HPO aimed at approximating a Pareto front, our formulation seeks a specified target operating point in the multi-response performance space. We evaluate the same optimization principle in three complementary settings: four-dimensional XGBoost training-time HPO on three public classification datasets, six-dimensional LoRA/SFT post-training of Qwen3-VL-8B on a binary VQAv2 subset, and eight-dimensional post-processing threshold optimization on fixed classifier outputs. On the XGBoost benchmarks, the proposed method achieves validation performance comparable to grid search, random search, and TPE under a smaller number of model evaluations, although it does not consistently outperform these black-box baselines. The VLM study demonstrates joint post-training hyperparameter updates using TPR/FPR feedback and reveals strongly non-monotonic response and sensitivity trajectories, including a high-TPR/high-FPR degenerate regime. The threshold study further demonstrates target seeking in an underdetermined two-response, eight-parameter system and shows sensitivity to initialization and the importance of retaining the best observed iterate. Across these settings, the results support trajectory-based secant sensitivity as an evaluation-efficient local optimization mechanism complementary to global black-box HPO.

1 INTRODUCTION

Hyperparameter optimization (HPO) is a central component of modern machine learning because choices such as learning rate, regularization, subsampling, model depth, and optimization schedules can substantially affect predictive performance, training stability, and computational cost Bischl et al. (2023). Unlike model parameters learned directly through training, hyperparameters are typically selected by an outer optimization process in which candidate configurations are trained or evaluated and their validation performance is observed. When each candidate requires a complete training or post-training run, the number of such evaluations can dominate the cost of HPO.

Classical search methods provide simple but largely direction-uninformed strategies. Grid search evaluates a predefined Cartesian product of candidate values, with cost growing rapidly with the number of hyperparameters. Random search can be substantially more efficient by exploring more

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

distinct values along influential dimensions Bergstra & Bengio (2012), but successive candidates do not explicitly exploit an estimated local response direction derived from previous performance changes.

Sequential model-based methods use previous evaluations more explicitly. Bayesian optimization constructs a probabilistic surrogate of validation performance and uses an acquisition function to select subsequent configurations Snoek et al. (2012). TPE uses density estimation for conditional and mixed search spaces Bergstra et al. (2013), while SMAC provides a widely used sequential model-based framework for algorithm configuration and HPO Hutter et al. (2011); Lindauer et al. (2022). These approaches exploit information from previous trials through a learned surrogate or density model. The approach studied here instead extracts a local numerical sensitivity directly from changes observed along the optimization trajectory and uses it to construct the next joint hyperparameter update.

Other HPO methods reduce cost through adaptive resource allocation or population-based exploration. Hyperband terminates poorly performing configurations early Li et al. (2017), BOHB combines model-based search with Hyperband Falkner et al. (2018), and population-based training jointly trains and adapts a population of models Jaderberg et al. (2017). Evolutionary strategies provide another derivative-free alternative, including covariance-adaptation methods that update a search distribution from populations of evaluated solutions Hansen & Ostermeier (2001). These approaches are particularly useful when partial training trajectories or parallel evaluations are available, but typically explore multiple candidates or maintain a population or distribution over candidate configurations.

Gradient-based HPO provides a different source of directional information. By expressing training and hyperparameter selection as a bilevel optimization problem, hypergradients can be obtained by differentiating through, approximating, or implicitly differentiating the inner training procedure Franceschi et al. (2018). Reversible differentiation, for example, propagates gradients through the training trajectory and can optimize many hyperparameters simultaneously Maclaurin et al. (2015). Such methods provide rich gradient information when the learning pipeline is differentiable and its optimization dynamics are accessible. These requirements can limit direct application to black-box learners, non-differentiable hyperparameters, legacy training systems, or evaluation metrics for which differentiation through the complete training procedure is unavailable.

Our approach instead obtains directional information numerically while keeping the learning procedure itself as a black box. It is therefore related to derivative-free numerical optimization, where local models, finite differences, or function-value observations are used when analytical derivatives are unavailable Conn et al. (2009); Khare & Kumar (2021); Fortin & Wolkowicz (2004). SPSA, for example, estimates a search direction from simultaneous random perturbations rather than coordinate-wise finite differences Spall (1992). In contrast, our method introduces no dedicated perturbation evaluations to estimate a scalar-objective gradient. Successive *full-vector* HPO iterates instead provide paired changes in the hyperparameter and validation-performance vectors. These trajectory observations are reused as local secant sensitivity information, from which a regularized Gauss–Newton system produces a joint update of the complete hyperparameter vector. Thus, the method is sensitivity-informed without requiring analytical gradients or backpropagation through model training.

Multiple validation responses introduce an additional distinction. Multi-objective HPO (MOHPO) is commonly formulated as the search for a set of nondominated configurations representing tradeoffs among competing objectives, using approaches such as Pareto-based evolutionary algorithms or surrogate-based methods such as ParEGO Deb et al. (2002); Knowles (2006); Morales-Hernández et al. (2023); Karl et al. (2022). We consider a related but different *target-seeking* problem. Let $\boldsymbol{\phi} \in \mathbb{R}^d$ denote the hyperparameter vector and $\mathbf{P}(\boldsymbol{\phi}) \in \mathbb{R}^m$ the corresponding vector of measured validation responses. Given a desired operating point $\mathbf{P}^\star$, we seek

$$\min_{\boldsymbol{\phi}} \frac{1}{2} \left\| \mathbf{P}^\star - \mathbf{P}(\boldsymbol{\phi}) \right\|_2^2 . \tag{1}$$

The objective is therefore not to approximate a Pareto front, but to move the measured response vector toward a specified operating point while retaining the individual responses explicitly in the local numerical system.

At iteration k, the black-box learning or evaluation procedure is executed at the complete vector $\boldsymbol{\phi}_k$ to obtain $\mathbf{P}_k$. Changes between successive hyperparameter and performance vectors provide secant information used to update a local sensitivity matrix. Importantly, this sensitivity should not be interpreted as an analytical or infinitesimal derivative of the training pipeline. Validation responses such as classification accuracy, TPR, and FPR can be noisy, plateaued, or non-monotonic with respect to hyperparameters. The method instead constructs a finite, trajectory-dependent local response approximation from observed changes and repeatedly updates it as new evaluations become available. Damping and best-iterate retention are consequently important practical components of the procedure.

This construction also has an important evaluation-cost distinction. A coordinate-wise forward-difference approximation of a d-dimensional sensitivity requires multiple dedicated perturbed evaluations around a reference configuration. Our iterative construction instead reuses the sequence of full-vector configurations generated by the optimization itself. Following the initial two evaluations used to establish the first secant, each subsequent update introduces one new full-vector evaluation and jointly advances all optimized hyperparameters. We therefore distinguish the number of expensive model evaluations from the inexpensive numerical work required to update the local sensitivity and solve the small regularized linear system. This distinction is particularly relevant when one evaluation requires model training or large-model post-training.

When the number of optimized hyperparameters exceeds the number of performance responses, the local inverse problem is underdetermined. This occurs in all three studies considered here: four XGBoost hyperparameters are guided by one primary validation response, six VLM post-training hyperparameters by two responses, and eight decision thresholds by two responses. We stabilize these local systems using Tikhonov regularization in a damped Gauss–Newton update and enforce feasible parameter ranges where applicable. Because the method is local and trajectory-based, its behavior can depend on initialization, damping, and the local response geometry; these properties are examined empirically rather than assuming global smoothness or convergence.

We evaluate the framework in three complementary settings spanning training, post-training, and post-processing. First, we jointly tune four XGBoost training hyperparameters on three public classification datasets and compare against grid search, random search, and TPE using explicit candidate-evaluation budgets. Second, we optimize six LoRA/SFT hyperparameters of Qwen3-VL-8B using TPR and FPR feedback, where each new HPO evaluation requires VLM post-training and is therefore substantially more expensive than the tabular case. Third, we optimize eight class-specific decision thresholds against two target responses using fixed classifier outputs, isolating the numerical optimization behavior from model-training cost. Together, these settings test the same full-vector update principle under substantially different evaluation loops and response geometries.

Our main contribution is a black-box, target-seeking HPO framework that reuses successive full-vector evaluations to obtain local secant sensitivity information and jointly update multiple hyperparameters through a Tikhonov-regularized damped Gauss–Newton system. The framework occupies a complementary position between direction-uninformed search, surrogate-based black-box HPO, and differentiable hypergradient methods: it extracts numerical directional information from observed HPO trajectories without differentiating through training, fitting a global surrogate, or requiring dedicated coordinate-wise perturbation runs. The experiments characterize both its evaluation efficiency and its limitations, including non-monotonic responses and sensitivity to initialization.

2 METHOD

2.1 PROBLEM FORMULATION

Let $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_d]^\top \in \mathbb{R}^d$ denote the vector of parameters to be optimized. For a given $\boldsymbol{\phi}$, the learning or evaluation procedure produces a vector of measured validation responses $\mathbf{P}(\boldsymbol{\phi}) = [P_1(\boldsymbol{\phi}), \ldots, P_m(\boldsymbol{\phi})]^\top \in \mathbb{R}^m$. Here, P_q denotes a measured performance quantity rather than an internal training loss; its definition depends on the application.

We consider the black-box mapping $\boldsymbol{\phi} \longmapsto \mathbf{P}(\boldsymbol{\phi})$, without differentiating through the underlying training or evaluation procedure. Given a desired operating point (performance metrics) $\mathbf{P}^\star \in \mathbb{R}^m$, the target-seeking problem is minimize the difference between it and the predicted performance

metrics (equation 1). The individual responses remain explicit rather than being collapsed into a single application-specific scalar score.

The proposed method obtains directional information numerically from changes in measured performance. In this sense, the search is *numerical-sensitivity informed*: unlike grid or random search, which do not construct a local response direction from previous evaluations, observed parameter–performance changes are used to inform the next joint update. No analytical hypergradient or differentiation through model training is required.

This formulation applies to all three settings considered in this paper. For XGBoost, $\boldsymbol{\phi}$ contains training hyperparameters; for VLM post-training, it contains LoRA/SFT hyperparameters; and for the post-processing experiment, it contains class-specific decision thresholds.

2.2 Full-Vector Secant Sensitivity

At iteration k, the complete parameter vector $\boldsymbol{\phi}_k$ is evaluated to obtain $\mathbf{P}_k = \mathbf{P}(\boldsymbol{\phi}_k)$. Two successive evaluations provide the full-vector secant pair

$$\Delta\boldsymbol{\phi}_k = \boldsymbol{\phi}_k - \boldsymbol{\phi}_{k-1}, \qquad \Delta\mathbf{P}_k = \mathbf{P}_k - \mathbf{P}_{k-1}. \tag{2}$$

We represent the local response through a numerical sensitivity matrix

$$\mathbf{J}_k \in \mathbb{R}^{m\times d}, \qquad \Delta\mathbf{P}_k \approx \mathbf{J}_k \Delta\boldsymbol{\phi}_k. \tag{3}$$

The matrix $\mathbf{J}_k$ plays the role of a local Jacobian relating changes in all d optimized parameters to changes in the m measured responses. It is not obtained by analytical differentiation through the learning algorithm; instead, it is a trajectory-dependent numerical sensitivity estimate constructed and updated from observed full-vector parameter–performance changes.

Because a successive finite-difference sensitivity estimate requires two observations, the numerical sensitivity is initialized using a small simultaneous perturbation of the complete parameter vector. Starting from the empirical or default initialization $\boldsymbol{\phi}_0$, we first evaluate $\mathbf{P}_0 = \mathbf{P}(\boldsymbol{\phi}_0)$ and then form $\boldsymbol{\phi}_1 = \boldsymbol{\phi}_0 + \delta\mathbf{1}_d$, where $\mathbf{1}_d \in \mathbb{R}^d$ is the all-ones vector and δ is a small perturbation magnitude (e.g., 0.001), such that all d optimized coordinates are perturbed simultaneously. A second full-vector evaluation gives $\mathbf{P}_1 = \mathbf{P}(\boldsymbol{\phi}_1)$, and the resulting differences $\Delta\boldsymbol{\phi}_1 = \boldsymbol{\phi}_1 - \boldsymbol{\phi}_0$ and $\Delta\mathbf{P}_1 = \mathbf{P}_1 - \mathbf{P}_0$ provide the initial numerical sensitivity information. Subsequent sensitivity estimates are obtained directly from consecutive HPO iterates and therefore require no additional perturbation evaluations. This initialization requires only one perturbed full-vector evaluation, rather than separate perturbation evaluations for each optimized coordinate.

After initialization, the parameter changes generated by the optimizer itself provide subsequent secant observations. Thus, the numerical sensitivity is repeatedly updated from the trajectory already produced by the HPO procedure rather than by introducing dedicated coordinate-wise perturbation runs at every iteration.

Importantly, $\mathbf{J}_k$ should be interpreted as a finite local response approximation rather than an analytical or infinitesimal derivative of the complete learning pipeline. Validation quantities such as accuracy, TPR, and FPR can be plateaued, noisy, or non-monotonic with respect to hyperparameters. The sensitivity estimate therefore describes the locally observed response along the optimization trajectory and is updated as new observations become available.

For the VLM implementation, the optimized hyperparameters have substantially different numerical scales. The corresponding secant quantities are normalized before constructing the numerical sensitivity used by the controller, preventing parameters with larger numerical magnitudes from dominating the update solely because of their units.

2.3 Damped Gauss–Newton Joint Update

At iteration k, the local sensitivity matrix is estimated from the performance and parameter changes between two consecutive iterates. Specifically, for performance metric p and optimized parameter q, we use the successive finite-difference approximation

$$[\mathbf{J}_k]_{pq} = \frac{P_{k,p} - P_{k-1,p}}{\phi_{k,q} - \phi_{k-1,q}}, \qquad p = 1,\dots,m, \quad q = 1,\dots,d. \tag{4}$$

Because all parameters may change simultaneously between consecutive iterates, Eq. (4) represents a trajectory-based numerical sensitivity rather than an independently perturbed coordinate-wise partial derivative. The first sensitivity estimate is initialized from two full-vector evaluations: the initial configuration and a small simultaneous perturbation of all optimized parameters. Thereafter, each newly evaluated HPO iterate is paired with the preceding iterate to update the secant sensitivity, requiring no additional coordinate-wise perturbation evaluations.

At iteration k, define the discrepancy between the desired and observed performance vectors as

$$\mathbf{r}_k = \mathbf{P}^\star - \mathbf{P}_k. \tag{5}$$

Using the resulting local sensitivity model, the desired joint parameter change approximately satisfies

$$\mathbf{J}_k \Delta\boldsymbol{\phi}_k \approx \mathbf{r}_k. \tag{6}$$

We compute the update using a Tikhonov-regularized damped Gauss–Newton system,

$$\left(\mathbf{J}_k^\top \mathbf{J}_k + \lambda_d \mathbf{I}\right) \Delta\boldsymbol{\phi}_k = \mathbf{J}_k^\top \mathbf{r}_k, \tag{7}$$

or equivalently

$$\Delta\boldsymbol{\phi}_k = \left(\mathbf{J}_k^\top \mathbf{J}_k + \lambda_d \mathbf{I}\right)^{-1} \mathbf{J}_k^\top \mathbf{r}_k, \tag{8}$$

where $\lambda_d > 0$ is the damping coefficient. The regularization stabilizes the local inverse problem when $\mathbf{J}_k$ is ill-conditioned or when $d > m$, and suppresses excessively large updates caused by uncertain local sensitivity estimates.

A key property of Eq. (7) is that it solves for $\Delta\boldsymbol{\phi}_k \in \mathbb{R}^d$ as a single vector. The sensitivity matrix therefore couples the measured performance responses to all optimized parameters, and all d coordinates are updated jointly rather than through independent coordinate searches.

The next parameter vector is

$$\boldsymbol{\phi}_{k+1} = \Pi_\Omega \left(\boldsymbol{\phi}_k + \alpha_k \Delta\boldsymbol{\phi}_k\right), \tag{9}$$

where α_k is the update step length and Π_Ω restricts the parameters to their prescribed feasible ranges when bounds are imposed. In implementations using the adaptive step-length schedule,

$$\alpha_k = a + b/(1+k), \tag{10}$$

where a controls the asymptotic step length and b controls the additional early-iteration step size.

The complete updated vector $\boldsymbol{\phi}_{k+1}$ is then evaluated once to obtain $\mathbf{P}_{k+1}$. Together with $(\boldsymbol{\phi}_k, \mathbf{P}_k)$, this new observation provides the successive finite-difference information used to construct $\mathbf{J}_{k+1}$ through Eq. (4).

Consequently, following the two-evaluation full-vector sensitivity initialization, each subsequent Gauss–Newton update introduces only one new full-vector evaluation while updating all d optimized parameters simultaneously. This differs from a coordinate-wise finite-difference Jacobian construction, which would require dedicated perturbation evaluations for individual coordinates. When an evaluation entails model training or post-training, the small local linear-algebra system in Eq. (7) is negligible compared with the model evaluation cost.

2.4 Training, Post-Training, and Post-Processing

The same optimization loop is applied to training, post-training, and post-processing, with candidate evaluations defined accordingly (Table 1). In all cases, successive full-vector performance changes update the local sensitivity and all hyperparameters are jointly updated while retaining the best observed configuration.

Algorithm 1 Iterative Gauss–Newton HPO

Require: $\boldsymbol{\phi}_0$, target $\mathbf{P}^\star$, λ_d, step rule α_k, perturbation δ, iterations K
Evaluate $\mathbf{P}_0 = \mathbf{P}(\boldsymbol{\phi}_0)$
$\boldsymbol{\phi}_1 \leftarrow \boldsymbol{\phi}_0 + \delta \mathbf{1}_d$; evaluate $\mathbf{P}_1 = \mathbf{P}(\boldsymbol{\phi}_1)$
Initialize secant sensitivity and best configuration
for $k = 1, \ldots, K-1$ **do**
  $\mathbf{r}_k \leftarrow \mathbf{P}^\star - \mathbf{P}_k$
  Update $\mathbf{J}_k$ from full-vector secant information
  Solve $(\mathbf{J}_k^\top \mathbf{J}_k + \lambda_d \mathbf{I})\Delta\boldsymbol{\phi}_k = \mathbf{J}_k^\top \mathbf{r}_k$
  $\boldsymbol{\phi}_{k+1} \leftarrow \Pi_\Omega(\boldsymbol{\phi}_k + \alpha_k \Delta\boldsymbol{\phi}_k)$
  Evaluate $\mathbf{P}_{k+1} = \mathbf{P}(\boldsymbol{\phi}_{k+1})$
  Update secant sensitivity and best configuration
end for
return Best observed configuration

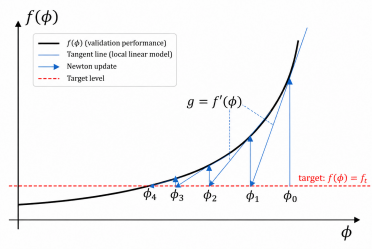


Figure 1: Scalar illustration of Newton-style target seeking; the multi-parameter method jointly updates all hyperparameters.

2.5 Experimental Design

We evaluate the framework in training-time HPO, VLM post-training, and prediction-level post-processing. Table 1 summarizes the corresponding parameter and response spaces and candidate evaluations.

Table 1: Overview of the three experimental settings.

Case	ML setting	Optimized parameters	Candidate evaluation	Responses
XGBoost	Tabular classification	4 training HPs	Training + validation	Accuracy
VLM	Multimodal post-training	6 SFT/LoRA HPs	Restarted post-training + validation	TPR, FPR
Threshold	Classifier post-processing	8 decision thresholds	Fixed-score evaluation	TP, FP

3 Experiments

3.1 Case I: Training-Time Hyperparameter Optimization with XGBoost

Experimental setup: We first evaluate the proposed method in a conventional tabular classification setting, where each candidate evaluation requires training a new model. The implementation code is publicly available in an anonymous repository.[1] We use three public scikit-learn datasets: Breast Cancer Wisconsin Diagnostic, Wine, and Digits, representing binary, three-class, and ten-class classification, respectively. For each candidate hyperparameter vector, a new XGBoost classifier is trained and evaluated on a fixed validation split, and validation accuracy provides the performance response used by the optimizer. A held-out test split is excluded from hyperparameter selection. Dataset characteristics, preprocessing, data splitting, parameter ranges, fixed XGBoost settings, and baseline search configurations are reported in Appendix A. We jointly optimize four continuous training hyperparameters, $\boldsymbol{\phi} = [\eta,\ s,\ c,\ \lambda_{\text{reg}}]^\top$, corresponding to learning rate, row subsampling ratio, column subsampling ratio, and L_2 regularization. All remaining XGBoost settings are fixed across methods. Grid search, Random Search, and TPE use the same hyperparameter domains and validation objective as the proposed method. We report candidate evaluations explicitly so that search budgets are compared in terms of expensive model training–validation evaluations rather than optimizer iterations.

Multi-dataset evaluation: We evaluate each method on Breast Cancer, Wine, and Digits across three random seeds. Table 2(a) reports the mean and standard deviation of the best observed validation accuracy. The parenthesized values denote the number of candidate model evaluations performed by each method.

Across the three datasets, the proposed method achieves validation accuracy comparable to the black-box baselines under a smaller candidate-evaluation budget, although it does not consistently attain the highest accuracy (Table 2). These results show that trajectory-based numerical sensitivity can produce competitive configurations while jointly updating all four hyperparameters.

[1] https://github.com/vlm-grounding/newton_hp_search

Table 2: XGBoost evaluation. (a) Best validation accuracy across three datasets (mean ± standard deviation over three seeds); parentheses report the number of candidate model evaluations. (b) Sensitivity to the Gauss–Newton damping coefficient on Breast Cancer (seed 42).

(a) XGBoost validation accuracy

Dataset	Proposed (8)	Grid (320)	Random (20)	TPE (20)
Breast Cancer	.977±.011	.976±.010	.974±.007	.977±.017
Wine	.991±.013	.995±.006	1.000±.000	1.000±.000
Digits	.970±.011	.976±.011	.974±.012	.977±.011

(b) Effect of damping

λ_d	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1
Acc.	.9737	.9737	.9737	.9649	.9737

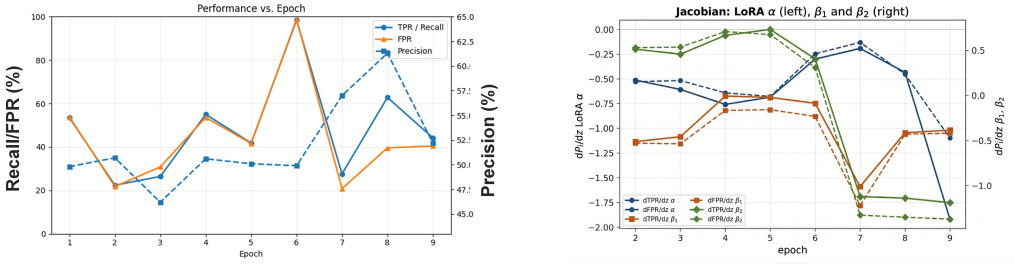
Effect of damping: We additionally examine sensitivity to the Gauss–Newton damping coefficient λ_d on Breast Cancer using seed 42. Table 2(b) shows relatively stable validation accuracy across the tested range: four of the five damping values yield 0.9737, while $\lambda_d = 10^{-1}$ yields 0.9649. Overall, the results indicate limited sensitivity to λ_d over the tested range.

3.2 Case II: VLM Post-Training Hyperparameter Optimization

Experimental setup: We next evaluate the proposed method in a substantially more expensive multimodal post-training setting using Qwen3-VL-8B. In contrast to Case I, where each hyperparameter vector controls a complete XGBoost training run, here each candidate vector controls one LoRA/SFT post-training run. Importantly, every candidate evaluation is restarted from the same initial model and LoRA state, so that differences in validation performance can be attributed to the post-training hyperparameters rather than accumulated training across HPO iterations. The resulting model is evaluated on the validation split, and the observed TPR and FPR responses are used to construct the local sensitivity estimate and determine the next joint hyperparameter update. The held-out evaluation split is not used to construct the sensitivity estimates or HPO updates and is reported only for evaluation. The implementation code is publicly available in an anonymous repository.[2]

We use a binary yes/no subset of VQAv2 with fixed training, validation, and held-out splits of 5,000, 1,000, and 1,000 examples, respectively. The held-out split is excluded from hyperparameter optimization. Six continuous post-training hyperparameters are jointly optimized: $\phi = [\eta, \lambda_{\text{wd}}, p_{\text{drop}}, \alpha_{\text{LoRA}}, \beta_1, \beta_2]^\top$, where η is the learning rate, λ_{wd} is weight decay, p_{drop} is dropout, α_{LoRA} is the LoRA scaling parameter, and β_1 and β_2 are the AdamW momentum coefficients.

TPR (recall) and FPR form the two-dimensional performance response used by the target-seeking update. Precision is reported as a diagnostic metric but is not used to construct the optimization update. Thus, after sensitivity initialization, each HPO iteration jointly updates all six hyperparameters, restarts LoRA/SFT post-training from the same initial state, and performs one validation evaluation. Additional implementation details and representative VQAv2 examples are provided in Appendix B.



(a) Validation performance (b) Local sensitivity

Figure 2: VLM post-training across HPO iterations. (a) Validation TPR/FPR and diagnostic precision. (b) Local secant sensitivities of TPR/FPR with respect to α_{LoRA}, β_1, and β_2; operating-point changes coincide with changes in the estimated sensitivities.

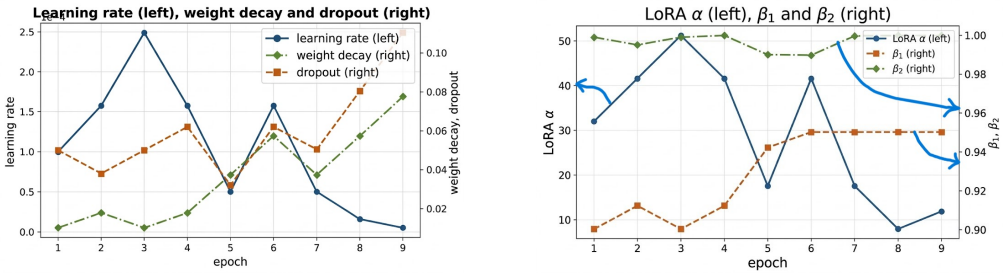
[2] https://github.com/vlm-grounding/Newton-VLM-PostTraining

Performance and local sensitivity evolution: Figure 2(a) shows a strongly non-monotonic validation trajectory across HPO iterations, while Fig. 2(b) shows the corresponding evolution of the local secant sensitivities used by the optimization procedure. The substantial variation in both the validation responses and their estimated local sensitivities illustrates the trajectory-dependent nature of the optimization landscape.

We additionally evaluate the checkpoints on the held-out split, which is not used to construct the HPO updates. Held-out recall increases from 53.1% at iteration 1 to 54.6% at iteration 4, while precision increases from 48.7% to 50.2% and FPR decreases slightly from 54.2% to 53.96%.

These results also illustrate why recall alone is insufficient for this target-seeking problem. Joint consideration of TPR and FPR distinguishes increased recall accompanied by controlled false positives from increased recall obtained primarily by shifting predictions toward the positive class. Representative VQAv2 examples and prediction trajectories are provided in Appendix B. Individual predictions can switch repeatedly between "yes" and "no", providing sample-level examples of the non-monotonic behavior observed across HPO iterations.

Joint hyperparameter evolution. Figure 3 shows the corresponding trajectories of all six post-training hyperparameters. Unlike coordinate-wise search, each HPO step updates the complete six-dimensional hyperparameter vector jointly.



(a) Learning rate, weight decay, and dropout

(b) LoRA α, β_1, and β_2

Figure 3: Joint evolution of six VLM post-training hyperparameters. All parameters are updated simultaneously as one vector.

The trajectories show nonuniform joint parameter changes. The learning rate initially increases to approximately 2.5×10^{-4} before decreasing, while weight decay and dropout undergo smaller adjustments. α_{LoRA} varies more substantially in absolute scale, whereas β_1 and β_2 remain within comparatively narrow ranges. These different trajectories arise within the same full-vector updates rather than from independent one-dimensional sweeps.

3.3 Case III: Post-Processing Threshold Optimization

Experimental design: We finally evaluate the proposed method in an eight-dimensional post-processing problem requiring neither model training nor post-training. A fixed proprietary CNN classifier produces probability scores for eight anonymized classes. For class i, a prediction is positive when $p_i \geq \phi_i$, with $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_8]^\top$ and $0 \leq \phi_i \leq 1$. The evaluation set contains 6,487 instances, including 1,002 positive and 5,485 negative cases. Model outputs remain fixed throughout optimization; each candidate evaluation applies the updated threshold vector to the stored scores and recomputes aggregate TP and FP.

Figure 4 shows representative positive and negative score distributions. The degree of overlap varies across classes, motivating class-specific thresholds, while the aggregate TP/FP target requires the eight thresholds to be updated jointly.

Target-seeking performance: Table 3 compares the proposed method with a Bayesian optimization reference for the 95/15 TP/FP target.

The proposed method reaches TP/FP=94.7/16.3 at iteration 15, compared with 93.0/17.0 for the Bayesian reference. Because the reference is unconstrained and uses three initial design points, this result is reported as a reference comparison rather than a controlled comparison of the two algorithms.

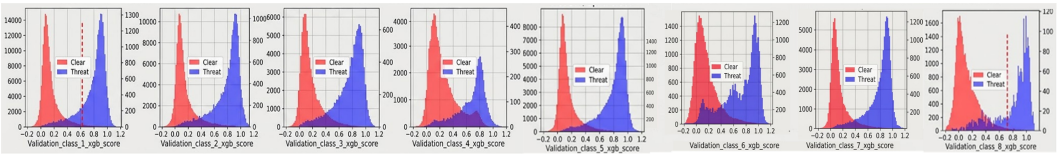


Figure 4: Prediction-score distributions for the eight-class threshold-optimization problem. Positive and negative scores exhibit different degrees of overlap across classes, motivating joint optimization of class-specific thresholds.

Table 3: Comparison with Bayesian optimization for the 95/15 TP/FP target. Target and outcome are reported in percent.

Method	**Input ϕ**	**Target**	**Outcome**	**Output ϕ**
Proposed	$[0.5]^8$	95/15	94.7/16.3	$[.600, .599, .633, .701, .667, .598, .600, .701]$
Bayesian*	$\{[0.5]^8, [0.75]^8, [0.9]^8\}$	95/15	93.0/17.0	$[.567, .639, .433, .731, .575, .600, -1.23, 2.57]$

*The Bayesian reference uses three initial design vectors and no explicit $[0, 1]$ box constraints; the infeasible outputs therefore reflect the baseline configuration rather than an inherent limitation of Bayesian optimization.

Init. ϕ_0	**Target**	**Outcome**	**Iter.**
$[.30]^8$	90/10	92.2/10.0	$\sim$7
$[.50]^8$	90/10	91.5/9.8	25
$[.90]^8$	90/10	84.1/10.1	200
$[.50]^8$	95/15	94.7/16.3	15
Variable*	96/15	94.0/15.3	15

Table 4: Effect of initialization and target operating point (TP/FP, %).

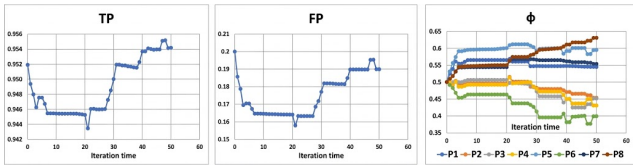


Figure 5: Trajectory for target TP$=95\%$, FP$=15\%$, initialized at $[0.5]^8$. Best: 94.7/16.3 at iteration 15.

Table 4 shows clear initialization dependence: for the common 90/10 target, the three initializations follow different trajectories and reach different operating points. This behavior is consistent with the local, trajectory-dependent nature of the method and motivates retaining the best observed iterate.

Non-monotonic optimization dynamics: Figure 5 shows a representative trajectory for the 95/15 target initialized from $[0.50]^8$. TP, FP, the threshold parameters, and their locally estimated sensitivities vary non-monotonically as the jointly updated thresholds move through different local response regions. The non-monotonic trajectory shows that the best target-matching operating point need not occur at the final iterate. The local sensitivity is therefore repeatedly updated from successive evaluations, while the best observed iterate is retained. Additional trajectories for different initializations and targets are provided in Appendix C; they further illustrate the initialization dependence and non-monotonic behavior of the local search.

4 Conclusion

We introduced a black-box, target-seeking HPO method that constructs damped Gauss–Newton updates from successive full-vector evaluations, without coordinate-wise perturbation runs or differentiation through training. The method jointly updates all hyperparameters using local secant sensitivities and Tikhonov regularization, while retaining the best observed iterate to accommodate non-monotonic responses. Across XGBoost training, Qwen3-VL-8B post-training, and classifier threshold optimization, the experiments demonstrate the same update principle under substantially different evaluation costs and response dimensions: the method is competitive with standard HPO baselines on the tabular benchmarks, supports joint six-dimensional VLM post-training updates from TPR/FPR feedback, and approaches specified TP/FP operating points in the eight-threshold problem. The observed non-monotonicity and initialization dependence also highlight that the method is a local, trajectory-dependent optimizer rather than a replacement for global black-box HPO. Future work will investigate more robust sensitivity estimation under noisy responses, larger model and search spaces, and hybrid global–local optimization strategies.

5 AI USAGE

AI-assisted tools were used to help identify relevant literature and to support language editing and improvement of the manuscript. The research methodology, experimental design, implementation, analysis, and conclusions were developed and verified by the authors.

REFERENCES

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 115–123, Atlanta, Georgia, USA, 2013. PMLR. URL https://proceedings.mlr.press/v28/bergstra13.html.

Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, Difan Deng, and Marius Lindauer. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery*, 13(2):e1484, 2023. doi: 10.1002/widm.1484.

Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization. SIAM, 2009.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1437–1446. PMLR, 2018. URL https://proceedings.mlr.press/v80/falkner18a.html.

Charles Fortin and Henry Wolkowicz. The trust region subproblem and semidefinite programming. *Optimization Methods and Software*, 19(1):41–67, 2004. doi: 10.1080/10556780410001647186.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1568–1577. PMLR, 2018. URL https://proceedings.mlr.press/v80/franceschi18a.html.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pp. 507–523. Springer, 2011.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks, 2017. URL https://arxiv.org/abs/1711.09846.

Florian Karl, Tobias Pielok, Julia Moosbauer, Florian Pfisterer, Stefan Coors, Martin Binder, Lennart Schneider, Janek Thomas, Jakob Richter, Michel Lang, Eduardo C. Garrido-Merchán, Juergen Branke, and Bernd Bischl. Multi-objective hyperparameter optimization in machine learning—an overview. *arXiv preprint arXiv:2206.07438*, 2022.

Shivangi Khare and Anwesh Kumar. Trust region methods. Medium, 2021. URL https://medium.com/trust-region-methods/temp-blog-1b51189594a. Published January 17, 2021.

Joshua Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1): 50–66, 2006.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=ry18Ww5ee.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 2113–2122. PMLR, 2015.

Alejandro Morales-Hernández, Inneke Van Nieuwenhuyse, and Sebastian Rojas Gonzalez. A survey on multi-objective hyperparameter optimization algorithms for machine learning. *Artificial Intelligence Review*, 56:8043–8093, 2023.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, pp. 2951–2959, 2012.

James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

A Tabular Dataset and Experimental Details

Case I evaluates XGBoost on three public datasets with different feature representations and class structures. Breast Cancer Wisconsin Diagnostic is a binary classification problem with 30 real-valued features describing cell-nucleus morphology. Wine is a three-class classification problem with 13 chemical measurements. Digits is a 10-class recognition problem represented by 64 flattened 8×8 pixel-intensity features. Thus, the experiments span binary and multiclass classification as well as morphological, chemical, and image-derived feature representations.

Table 5: Characteristics of the tabular classification tasks in Case I.

Dataset	**Features**	**Classes**	**Feature type**
Breast Cancer	30	2	Nucleus morphology
Wine	13	3	Chemical measurements
Digits	64	10	8×8 pixel intensities

For each run, the data are stratified into training, validation, and held-out test partitions. The proposed method and baseline HPO methods use the same partition, with models fitted on the training set and hyperparameters selected using validation performance. A median imputer and standard scaler fitted on the training partition are subsequently applied to the validation and test partitions without changing the feature dimensionality. The test partition is excluded from the HPO objective.

B VLM Examples and Prediction Variation

Case II uses a selected binary yes/no subset of VQAv2 for VLM post-training. The reported validation performance should therefore be interpreted in the context of this selected subset rather than as the overall VQAv2 performance of Qwen3-VL-8B. The subset contains challenging examples requiring combinations of visual recognition, semantic understanding, relational reasoning, and scene-level interpretation rather than only direct object detection or simple visual verification. The examples below provide qualitative context for the relatively low absolute TPR and precision observed in Section 3.2; the purpose of Case II is to study HPO behavior under a challenging multimodal post-training response rather than to establish the benchmark-level capability of Qwen3-VL-8B.

Figure 6 presents 16 representative VQAv2 yes/no questions and the corresponding model predictions across successive post-training epochs. The `gold` column gives the ground-truth answer, while $e_1, \ldots, e_9$ denote predictions obtained at different epochs. Several examples switch repeatedly between "yes" and "no", showing that sample-level prediction correctness does not necessarily improve monotonically during post-training. This behavior provides qualitative, sample-level context for the non-monotonic aggregate validation responses observed in Case II.

#	qid	image	question	gold	e1	e2	e3	e4	e5	e6	e7	e8	e9
1	260595002	val2014_000000260595.jpg	Is this a standard traffic sign?	no	yes	no	no	yes	no	yes	no	no	no
2	18935002	train2014_000000018935.jpg	Are there any other surfers around?	no	no	no	no	yes	no	yes	no	yes	yes
3	278989005	train2014_000000278989.jpg	Is this plane leaving the runway?	no	no	no	no	yes	yes	yes	no	no	no
4	112437001	val2014_000000112437.jpg	Is that the real Batman on the cake?	no	yes	no	yes	no	yes	yes	no	no	no
5	81930002	train2014_000000081930.jpg	Is this picture taken in winter?	no	no	no	yes	no	no	yes	no	no	no
6	553506002	train2014_000000553506.jpg	Is Bill Gates likely to buy this computer?	no	no	no	yes	yes	yes	yes	no	no	no
7	357989001	train2014_000000357989.jpg	Are any of these players shirtless?	yes	yes	no	no	yes	no	yes	no	yes	yes
8	492213002	train2014_000000492213.jpg	Is there meat on this pizza?	yes	yes	no	no	no	no	yes	yes	no	no
9	26362002	train2014_000000026362.jpg	Is the umbrella necessary for this weather?	no	no	yes	no	yes	no	yes	no	yes	no
10	473733000	train2014_000000473733.jpg	Is the man wearing jeans?	yes	no	no	no	no	yes	yes	no	no	yes
11	288862000	val2014_000000288862.jpg	Is the person doing a handstand?	yes	yes	no	yes	yes	no	yes	no	no	no
12	198975001	train2014_000000198975.jpg	Are hubcaps on the yellow bus dirty?	yes	yes	yes	no	yes	no	yes	no	yes	no
13	121745002	val2014_000000121745.jpg	Is the plate a solid color?	no	yes	no	yes	yes	yes	yes	no	yes	no
14	238724001	train2014_000000238724.jpg	Does this activity require waves?	yes	yes	no	no	yes	yes	yes	yes	yes	no
15	422295000	train2014_000000422295.jpg	Is the natural mask on the cows face white?	no	yes	no	no	yes	yes	yes	yes	yes	yes
16	220601001	train2014_000000220601.jpg	Are the lights turned on?	no	yes	no	no	no	yes	yes	no	yes	yes

Figure 6: Predictions for 16 representative VQAv2 yes/no examples across successive VLM post-training epochs. The `gold` column denotes the ground-truth answer, and $e_1, \ldots, e_9$ show the corresponding model predictions at different epochs.

The following page shows the corresponding 16 input images together with their questions and ground-truth answers. The examples span different visual and semantic requirements, including object and attribute recognition, spatial and relational understanding, scene interpretation, and questions requiring information to be combined across multiple visual cues. They illustrate that the evaluation subset contains more than simple object-presence questions and provide additional context for the absolute performance observed in the VLM experiment. Together with Figure 6, these examples also show that prediction variation occurs across different types of visual-question inputs rather than being confined to a single example category.

16 val samples on one page (two samples per row)



Sample 1
Image source:
val2014 / COCO_val2014_000000260595.jpg
Question:
Is this a standard traffic
sign?
Ground-truth answer: no



Sample 2
Image source:
train2014 / COCO_train2014_000000018935.jpg
Question:
Are there any other surfers
around?
Ground-truth answer: no



Sample 3
Image source:
train2014 / COCO_train2014_000000278989.jpg
Question:
Is this plane leaving the
runway?
Ground-truth answer: no



Sample 4
Image source:
val2014 / COCO_val2014_000000112437.jpg
Question:
Is that the real Batman on
the cake?
Ground-truth answer: no



Sample 5
Image source:
train2014 / COCO_train2014_000000081930.jpg
Question:
Is this picture taken in
winter?
Ground-truth answer: no



Sample 6
Image source:
train2014 / COCO_train2014_000000553506.jpg
Question:
Is Bill Gates likely to buy
this computer?
Ground-truth answer: no



Sample 7
Image source:
train2014 / COCO_train2014_000000357989.jpg
Question:
Are any of these players
shirtless?
Ground-truth answer: yes



Sample 8
Image source:
train2014 / COCO_train2014_000000492213.jpg
Question:
Is there meat on this pizza?
Ground-truth answer: yes



Sample 9
Image source:
train2014 / COCO_train2014_000000026362.jpg
Question:
Is the umbrella necessary
for this weather?
Ground-truth answer: no



Sample 10
Image source:
train2014 / COCO_train2014_000000473733.jpg
Question:
Is the man wearing jeans?
Ground-truth answer: yes



Sample 11
Image source:
val2014 / COCO_val2014_000000288862.jpg
Question:
Is the person doing a
handstand?
Ground-truth answer: yes



Sample 12
Image source:
train2014 / COCO_train2014_000000198975.jpg
Question:
Are hubcaps on the yellow
bus dirty?
Ground-truth answer: yes



Sample 13
Image source:
val2014 / COCO_val2014_000000121745.jpg
Question:
Is the plate a solid color?
Ground-truth answer: no



Sample 14
Image source:
train2014 / COCO_train2014_000000238724.jpg
Question:
Does this activity require
waves?
Ground-truth answer: yes



Sample 15
Image source:
train2014 / COCO_train2014_000000422295.jpg
Question:
Is the natural mask on the
cows face white?
Ground-truth answer: no



Sample 16
Image source:
train2014 / COCO_train2014_000000220601.jpg
Question:
Are the lights turned on?
Ground-truth answer: no

C Additional Threshold-Optimization Trajectories

This section provides additional trajectories for the post-processing threshold-optimization experiment in Section 3.3. All experiments use the same fixed classifier outputs and jointly optimize the eight class-specific probability thresholds. The initial threshold vector and target TP/FP operating point are varied to examine their effects on the local optimization trajectory. The representative 95/15 case is presented in Figure 5 of the main text.

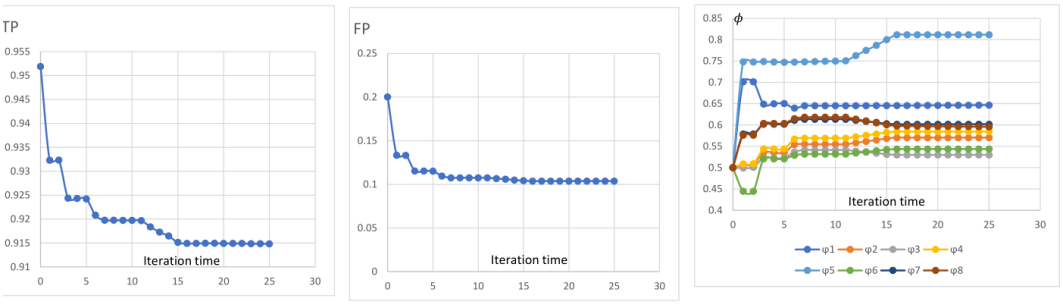


Figure 7: Additional trajectory for target TP$=90\%$ and FP$=10\%$, initialized with $\phi_0 = [0.5]^8$. TP and FP decrease toward the target region as the individual threshold coordinates evolve in different directions. The best reported operating point is TP$=91.5\%$ and FP$=9.8\%$ at iteration 25.

converge since the 15th iteration.

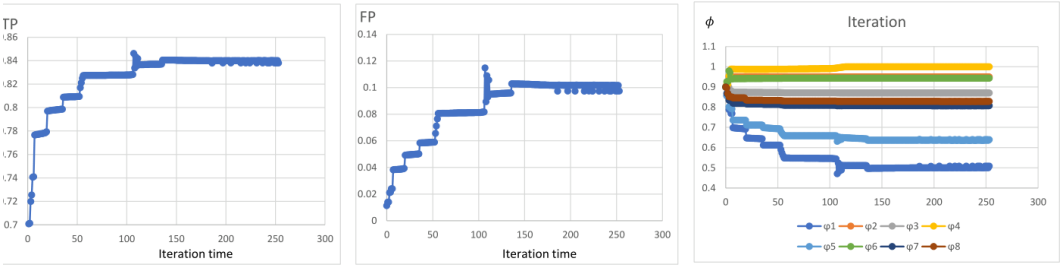


Figure 8: Additional trajectory for target TP$=90\%$ and FP$=10\%$, initialized with the relatively high threshold vector $\phi_0 = [0.9]^8$. TP and FP increase as the thresholds move away from their high initial values. The reported operating point at iteration 200 is TP$=84.1\%$ and FP$=10.1\%$, illustrating the effect of initialization on the local optimization trajectory.

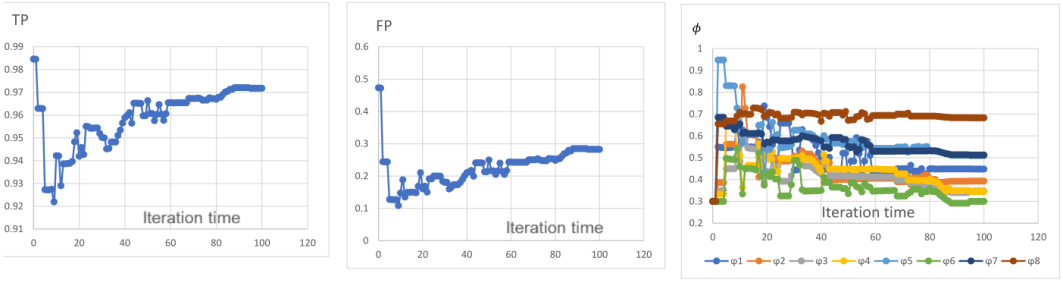
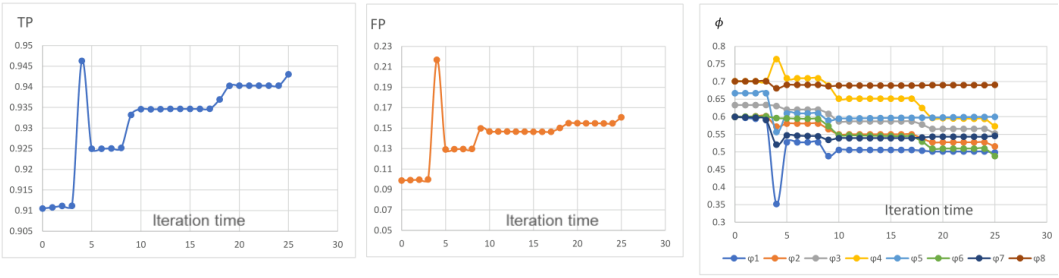


Figure 9: Additional trajectory for target TP$=90\%$ and FP$=10\%$, initialized with $\phi_0 = [0.3]^8$. TP, FP, and individual threshold values exhibit oscillatory behavior, with the best reported operating point TP$=92.2\%$ and FP$=10.0\%$ occurring around iteration 7.

94.7% and FP is 16.3% at the 15th iteration.



Seed = [0.59971717 | 0.599717 | 0.63323 | 0.700931 | 0.6669 | 0.599717 | 0.599717 | 0.700931]

Figure 10: Additional trajectory for target TP$=96\%$ and FP$=15\%$ using the nonuniform initialization $\phi_0 = [0.5997, 0.5997, 0.6332, 0.7009, 0.6669, 0.5997, 0.5997, 0.7009]$, obtained from preliminary optimization. TP and FP vary non-monotonically during the trajectory, with the best reported operating point TP$=94.0\%$ and FP$=15.3\%$ at iteration 15.

Figures 7–10 show substantial differences in transient behavior across initializations. For the common 90/10 target, initializations of $[0.3]^8$, $[0.5]^8$, and $[0.9]^8$ produce different parameter paths, iteration horizons, and achieved operating points. The trajectories also exhibit non-monotonic or oscillatory changes in TP, FP, and individual threshold coordinates. Together with the representative 95/15 case in the main text, these results illustrate the initialization dependence of the local search and motivate repeated re-estimation of the local sensitivity and retention of the best observed iterate rather than assuming monotonic convergence.