WHEN BETTER CLASSIFIERS MAKE WORSE TUNERS: IONTRAPBENCH AND HISTORY-GATED STOPPING FOR CLOSED-LOOP MASS SPECTROMETER TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In autonomous instrument tuning systems based on learned classifiers, spectral diagnoses determine when to stop or continue adjustment. However, classification errors not only affect the assessment of the current signal but may also change subsequent observations through parameter updates. Window-level classification metrics therefore cannot directly substitute for evaluation of the complete tuning process. We reveal this prediction–decision misalignment in autonomous mass spectrometer tuning: retraining that improves window-level fault recall reduces the tuning success rate; false negatives lead to premature stopping, whereas false positives may introduce new faults through unnecessary parameter adjustments. To systematically investigate this problem, we propose IonTrapBench, which extends conventional window-level fault classification evaluation to a trajectory-level closed-loop decision evaluation paradigm that includes parameter states, action consequences, and final outcomes. On this basis, we propose History-Gated Stopping, which uses historical fault risk at a parameter state to adjust stopping decisions, combining predictions with state-level decision risk without retraining the classifier. Under sequential history, the method reduces false negatives by 88.2%; with full-history assistance, it increases the average tuning success rate from 84.2% to 91.6%. We plan to release the data and evaluation materials to support joint research on spectral diagnosis, parameter adjustment, and their final outcomes.

1 INTRODUCTION

The development of self-driving laboratories has driven experimental systems to evolve from executing predefined procedures toward a closed-loop mode in which subsequent operations are selected based on observational feedback (King et al., 2009; Häse et al., 2018; Canty & Abolhasani, 2026). For miniature mass spectrometers, autonomous operation requires not only identifying abnormalities in the current spectrum but also deciding when to accept the current parameters and when to continue adjustment. Owing to instrument miniaturization, environmental changes, and operation by non-specialists, target peaks may exhibit faults such as saturation, tailing, insufficient unit mass resolution, and a low signal-to-noise ratio (Zhai et al., 2024; Snyder et al., 2016). Premature stopping accepts faulty signals, whereas continued adjustment may disrupt an already usable configuration. The stopping decision is therefore a key link between spectral diagnosis and autonomous measurement outcomes.

Existing automatic adjustment schemes for miniature mass spectrometers use a classifier to read a spectral window around the target peak and select STOP or ADJUST from normal and fault scores. However, window-level predictions do not directly determine complete tuning outcomes. A false negative may stop the system in a faulty state, whereas a false positive may trigger an unnecessary update, consume the scan budget, or move a usable state into a faulty region. Figure 1 illustrates this prediction–decision misalignment. Classifier A stops prematurely; replacing it with the window-level stronger Classifier B changes the trajectory but may still produce faults. By contrast, history gating constrains stopping using the historical risk of the current parameter configuration, encouraging continued adjustment when evidence of normality is insufficient. Classifier performance must therefore be evaluated together with the actions and outcomes it induces.

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.

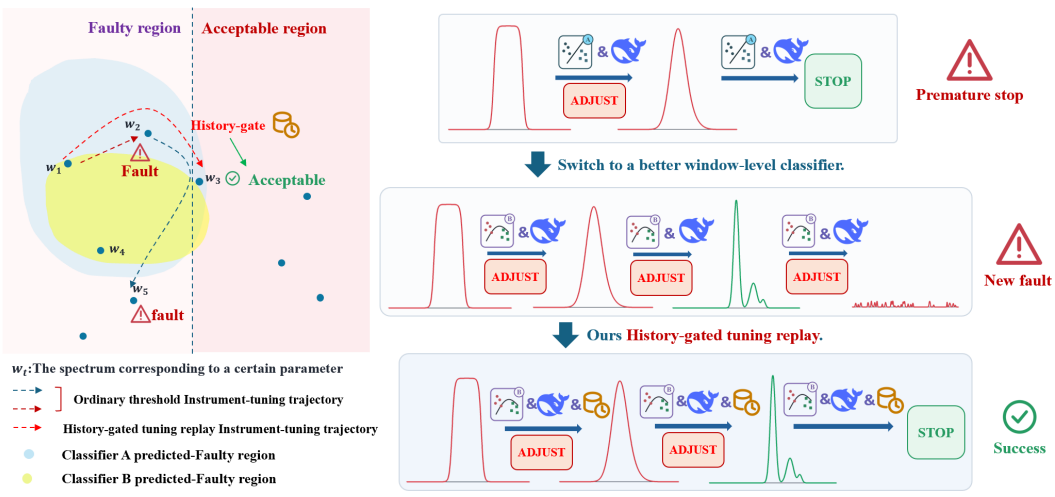


Figure 1: Prediction–decision misalignment in classifier–LLM closed-loop tuning. At each step, a spectrum classifier diagnoses the current window, and an LLM tuning agent uses the diagnostic output to select the next action. Classifier A can cause a premature stop, whereas switching to a classifier with better window-level performance changes the action trajectory but may introduce a new fault. History-gated tuning supplies parameter-conditioned historical risk together with the classifier output, enabling the LLM agent to continue adjustment and stop at an acceptable configuration.

This distinction matters because a closed-loop tuning episode is not a collection of independent spectral classifications. Each prediction changes the next state that will be observed: a STOP action fixes the final spectrum, while an ADJUST action moves the instrument to another point on the parameter grid and exposes the system to new possible faults. The same window-level error can therefore have different consequences depending on when it occurs, how much scan budget remains, and which parameter update follows it. For this reason, the appropriate evaluation unit is the complete trajectory and its final accepted spectrum, rather than the diagnostic accuracy of isolated windows alone.

This problem is evident in our offline replay. We compare the original MLP classifier with two versions selected for window-level objectives: one increases faulty-sample weights, and the other optimizes fault recall under a false-positive-rate constraint. With identical initial cases, repair rules, and scan budget, the three versions achieve 108, 112, and 105 successful episodes on the same 136 fault-start episodes. The retrained classifiers do not consistently outperform the original model: some produce ineffective adjustments and new faults triggered by false positives, while the high-recall version still produces false negatives and occasional deterioration after recovery. Thus, classifier replacement must be evaluated over complete decision trajectories.

Systematically investigating this problem requires data with parameter configurations, fault ground truth, and post-action outcomes. Existing public mass spectrometry resources primarily support spectrum sharing, molecular identification, structure inference, and related machine learning tasks (Martens et al., 2011; Haug et al., 2013; Perez-Riverol et al., 2019; Dührkop et al., 2015; Bushuiev et al., 2024), rather than instrument adjustments and state transitions. Existing uncertainty and risk methods address prediction reliability and deployment risk, but do not directly characterize decision-induced state changes during sequential tuning. A benchmark and evaluation framework for closed-loop instrument adjustment are therefore needed.

To this end, we construct IonTrapBench by organizing PEG-400 and sibutramine spectra acquired under discrete parameter combinations into replay tasks. Different policies access the same initial cases, repair rules, scan budgets, and spectral records; success is determined from the reference labels of the final window. Replay is not equivalent to online operation, but it enables controlled comparison of decision rules by holding acquisition records and repair mechanics fixed while changing only the stopping evidence. This design makes it possible to attribute differences in success, false stopping, and scan cost to the decision rule under study, rather than to uncontrolled variation in instrument acquisition. We plan to release the data, annotations, fixed splits, and evaluation materials.

Within IonTrapBench, we ask whether historical records of parameter configurations can supplement the current spectrum. We propose History-Gated Stopping, which discounts the classifier's normal score by the historical fault rate of the current configuration and combines current fault scores with a fallback rule to choose STOP or ADJUST. It does not retrain the classifier or specify the repair direction; it only raises the evidence required to stop at high-risk states. In this sense, history is used as conservative state-level context rather than as a replacement for spectral diagnosis or expert repair. Under sequential history, the method reduces premature stops by 88.2% relative to the ordinary threshold policy; under full-history assistance, it increases the average tuning success rate from 84.2% to 91.6%.

Our contributions are as follows: (1) we construct IonTrapBench, a tuning replay benchmark containing spectra, acquisition parameters, and fault annotations; (2) we propose a gating method based on historical fault rates that requires no classifier retraining; and (3) we evaluate stopping policies on a target peak excluded from classifier training, under different history conditions, and on another substance, and report their effects and limitations through paired trajectories and ablation analyses.

2 Related Work

2.1 Closed-loop scientific experimentation and mass-spectrometry benchmarks

Autonomous scientific experimentation is moving from automated execution toward closed-loop decisions based on real-time observations. The Robot Scientist, ChemOS, self-driving laboratories, and the mobile robotic chemist demonstrate observation–decision–action paradigms that select subsequent experimental conditions from new results (King et al., 2009; Roch et al., 2018; Häse et al., 2018; MacLeod et al., 2020; Burger et al., 2020). By contrast, existing mass spectrometry data resources mainly focus on spectrum-level information sharing and molecular analysis. For example, mzML standardizes mass-spectrometry data exchange (Martens et al., 2011), MetaboLights and PRIDE support public metabolomics and proteomics data deposition (Haug et al., 2013; Perez-Riverol et al., 2019), and MassBank and GNPS support the sharing, retrieval, and annotation of mass spectrometry data (Horai et al., 2010; Wang et al., 2016). Machine-learning toolkits and benchmarks further support spectrum processing and molecular analysis; for instance, pyOpenMS provides programmatic mass-spectrometry data processing (Röst et al., 2014), CSI:FingerID addresses molecular-structure identification from tandem spectra (Dührkop et al., 2015), and MassSpecGym provides standardized tasks such as molecular structure generation, candidate retrieval, and spectrum simulation (Bushuiev et al., 2024). These resources have advanced machine learning for mass spectrometry, but they primarily evaluate spectrum-centric tasks rather than parameter adjustments and sequential state transitions. They therefore cannot directly evaluate episode-level outcomes in closed-loop instrument tuning.

2.2 Decision-aware learning and sequential decision making

When predictions trigger downstream actions, prediction quality and decision quality may diverge. Decision-aware learning, decision-focused learning, and predict-then-optimize studies incorporate downstream objectives instead of optimizing only prediction errors (Donti et al., 2017; Wilder et al., 2019; Elmachtoub & Grigas, 2022). Performative prediction further shows that actions induced by predictions can change subsequent data distributions (Perdomo et al., 2020). In closed-loop mass spectrometer tuning, STOP/ADJUST decisions change parameters and observations, so stopping decisions must be evaluated over trajectories rather than isolated classifications.

2.3 Reliable prediction, selective prediction, and deferral

Reliable machine learning addresses predictive uncertainty and deployment risk. MC-dropout and deep ensembles estimate uncertainty (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017), and calibration improves confidence reliability (Guo et al., 2017). Selective prediction rejects low-confidence predictions (Geifman & El-Yaniv, 2019), learning to defer transfers decisions to experts (Mozannar & Sontag, 2020), and conformal prediction and risk control use calibration data for risk guarantees (Shafer & Vovk, 2008; Angelopoulos et al., 2024). These methods mainly assess trustworthiness from the current sample and model output. Closed-loop stopping may also depend on the

current parameter state and its historical performance, motivating parameter-conditioned historical risk.

3 METHODS

3.1 BENCHMARK AND REPLAY PROTOCOL

IonTrapBench contains 3,594 PEG-400 spectra from 184 parameter configurations and 6,188 sibutramine spectra from 192 configurations. The four controls are sampling cone voltage, ion funnel RF control voltage, ion injection time, and mass scan time. As illustrated in Figure 2, each local spectral window is annotated for saturation, peak tailing, insufficient unit mass resolution, and low signal-to-noise ratio; multiple faults may coexist. Table 1 summarizes the grid and data. Details of acquisition, preprocessing, training, and replay are provided in Sections A.1–A.3.

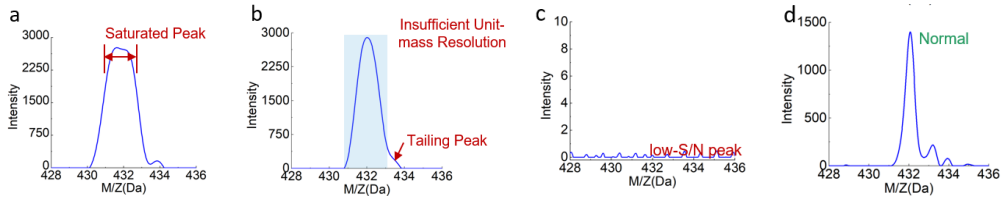


Figure 2: Normal and faulty peak shapes: (a) saturation; (b) insufficient unit mass resolution and peak tailing; (c) low signal-to-noise ratio; (d) normal.

Table 1: IonTrapBench acquisition grid and dataset organization. (a) Available values of the four controls; other acquisition settings are fixed. (b) Recorded data and task construction. RF denotes the ion funnel RF control voltage.

(a) Control grid

Parameter	Values
Sampling cone (V)	5, 15, 25, 35, 40, 45
RF control (V)	0.1, 0.3, 0.5, 0.7
Injection time (ms)	10, 20, 100, 200
Scan time (ms)	200, 400, 600, 800, 1000

(b) Data overview

Item	PEG-400	Sibutramine
Configurations	184	192
Full spectra	3,594	6,188
Target m/z	344, 388, 432, 476, 520	279
Data use	Training, calibration, replay	Cross-analyte evaluation

Both datasets: scan range m/z 200–1000; peak-center search ± 6 Da, extraction ± 2 Da; interpolation to 400 points; normal label and four fault labels (multilabel).

For each target peak, we locate the intensity maximum within ± 6 Da of its theoretical center, extract ± 2 Da around the detected center, and linearly interpolate to 400 points. A $400 \to 256 \to 128 \to 64 \to 5$ MLP produces a normal score and four independent fault scores, following the standard multilabel setting in which multiple labels may be active for one sample (Zhang & Zhou, 2014). Training uses the PEG-400 peaks at m/z 344, 388, 432, and 520; the peak at m/z 476 and sibutramine are excluded from classifier training.

The MLP provides structured diagnoses to an LLM tuning agent under a fixed control protocol. STOP accepts the current configuration; ADJUST invokes deterministic expert repair rules, changing one parameter by one grid level. Offline replay retrieves the recorded spectrum at the resulting configuration. Observation identifiers fix which replicate is accessed, and each episode allows at most five rescans. The comparisons keep the classifier, repair rules, and scan budget fixed unless classifier replacement is explicitly evaluated.

This replay design intentionally separates stopping behavior from other sources of variation. Because the same initial configurations, replicate identifiers, repair rules, and scan budget are reused across policies, two methods that diverge after the first decision can still be compared through the states they actually visit. The protocol therefore measures the downstream effect of accepting, de-

laying, or continuing adjustment under a fixed control environment, rather than the performance of a newly optimized controller.

An episode succeeds only if the final reference fault set is empty and the episode has not been deferred. FN counts episodes that start with a fault, fail, and terminate with `stopped_by_model`. FA counts unsuccessful episodes starting with normal spectra, including damage and deferral. Deferral rate is the proportion of DEFER episodes; fault-start success rate uses only the initially faulty subset. A new-fault event occurs when an adjustment introduces a previously absent fault. Formal definitions are retained in Section A.3.

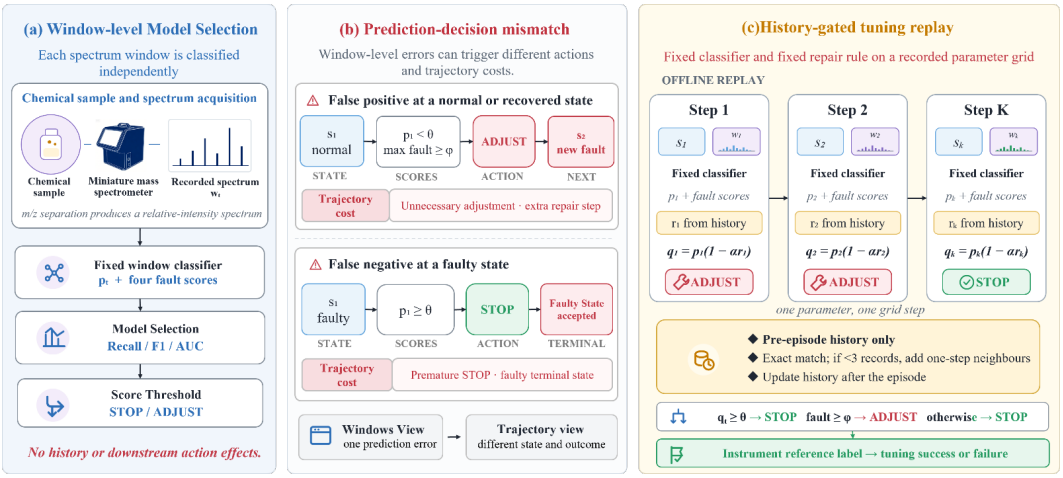


Figure 3: IonTrapBench workflow. (a) Window-level spectral diagnosis and model selection. (b) Prediction errors alter actions and subsequent states. (c) Offline replay with History-Gated Stopping: history discounts the normal score while the classifier and expert repair rules remain fixed.

3.2 History-Gated Stopping

Because IonTrapBench stores spectra with parameter configurations, we investigate whether state-level fault history can support stopping decisions. History-Gated Stopping neither retrains the MLP nor selects repair directions; it discounts the normal score by the historical fault risk of the current configuration. The historical signal is therefore used only as additional evidence about whether the current state is safe to accept. It does not alter which fault receives repair priority, and it does not introduce an alternative action planner.

The workflow of History-Gated Stopping is shown in Figure 3c. Let p_t denote the normal score output by the MLP at step t. For the current configuration $\mathbf{s}_t$, the system queries only historical records already accessible before the current episode begins, and denotes the set of matching records by $H_t(\mathbf{s}_t)$. The historical fault rate is defined as the proportion of spectra in the matching records that contain at least one fault:

$$r_t = \frac{1}{|H_t(\mathbf{s}_t)|} \sum_{h \in H_t(\mathbf{s}_t)} \mathbb{I}[F(w_h) \neq \varnothing]. \tag{1}$$

Historical queries first use exact parameter matches; with fewer than three records, the query expands to one-step neighboring configurations. After each parameter update, the system queries history at the new configuration. Current-episode records are added only after the episode ends, preventing leakage. History-Gated Stopping then discounts the normal score before the LLM outputs an action:

$$q_t = p_t(1 - \alpha r_t). \tag{2}$$

Here, $\alpha \in [0, 1]$ controls the influence of historical information. The quantity q_t is a stopping decision score and is not interpreted as a recalibrated probability of normality. This operation does not change the four fault scores, the MLP parameters, or the deterministic expert repair rules. Let the maximum of the four fault scores be:

$$p_t^{\text{fault,max}} = \max(p_t^{\text{sat}}, p_t^{\text{tail}}, p_t^{\text{res}}, p_t^{\text{snr}}). \tag{3}$$

Let θ be the normal-score threshold for stopping and ϕ the fault-score threshold for triggering adjustment. The system selects actions in a fixed order:

$$a_t = \begin{cases} \text{STOP}, & q_t \geq \theta, \\ \text{ADJUST}, & q_t < \theta \wedge p_t^{\text{fault,max}} \geq \phi, \\ \text{STOP}, & q_t < \theta \wedge p_t^{\text{fault,max}} < \phi. \end{cases} \tag{4}$$

The first branch accepts the configuration when normality remains sufficient after discounting. The second continues adjustment when normal evidence is insufficient and at least one fault score supports repair. The third retains fallback stopping when there is no clear fault evidence; history is therefore not an absolute veto.

The higher the historical fault rate, the stronger the current evidence of normality required for the system to stop. When no historical records are available, we set $r_t = 0$, giving $q_t = p_t$, and History-Gated Stopping reduces exactly to the ordinary threshold policy. The main experiments use $\alpha = \theta = \phi = 0.5$.

The experiments compare three history-access conditions. Calibration history uses only calibration configurations and does not access exact-match outcomes for evaluation configurations. Sequential history evaluates episodes in order and updates history after each episode. Full history covers the recorded grid and serves as an assisted condition with more complete information. Whether a target peak participated in MLP training and which historical records are accessible at decision time are distinct; full history should not be read as a setting in which the entire system has never encountered the relevant states.

These conditions represent increasingly informative deployment assumptions. Calibration history reflects a small pre-collected tuning memory, sequential history reflects an instrument that accumulates experience during use, and full history estimates what is possible when the parameter grid has already been well characterized. Keeping these cases separate helps distinguish the benefit of decision-time state memory from the benefit of classifier training exposure.

4 Results and Discussion

4.1 Overall Performance and Tuning Cost

We first evaluate stopping policies on the PEG-400 target peak at m/z 476, whose windows are excluded from base-MLP training. The 184 PEG-400 parameter combinations are split by seed into 20% calibration and 80% evaluation sets, with 37 calibration configurations and 147 initial tuning states. The evaluation set contains 109 faulty and 38 normal initial spectra. Because evaluation sets may overlap, this is not five-fold cross-validation. All fitted stopping thresholds, bands, temperature parameters, and conformal quantiles use only the calibration set of the corresponding split.

To isolate stopping decisions, the main experiments fix the base MLP classifier and deterministic expert repair engine. The repair engine follows the highest-priority detected fault; each step changes one control parameter by one grid level and retrieves the corresponding recorded spectrum. Each episode permits at most five rescans. Tuning succeeds only if the final spectrum has none of the four faults in Section 3 and is not deferred.

We report overall success, fault-start success, false negatives, FA, new faults, and average rescans. FN is an unsuccessful fault-start episode ending with `stopped_by_model`. FA counts unsuccessful normal-start episodes, including damage after adjustment or deferral; it is not a stepwise

Table 2: Stopping policies on PEG-400 at m/z 476 (mean ± standard deviation over five parameter-level random splits). Calibrated and conformal thresholds use the calibration set. HGS uses calibration records, calibration records plus earlier episodes, or history covering the recorded parameter grid. SR is overall tuning success rate; fault-start SR uses only initially faulty episodes. Superscript arrows indicate whether higher (↑) or lower (↓) values are better.

Method / history	SR↑	Fault-start SR↑	FN↓	FA↓	New faults↓	Rescans↓
Fixed threshold (0.5)	84.2±1.2	78.7±1.6	13.6±2.2	**0.0±0.0**	2.0±0.0	**1.20±0.04**
Calibrated threshold	87.2±2.5	82.8±3.4	8.0±4.6	**0.0±0.0**	2.0±0.0	1.37±0.14
Conformal (calibration)	87.5±0.5	83.7±0.4	**0.0±0.0**	0.6±0.5	3.0±0.0	3.27±0.20
HGS (calibration)	87.8±2.0	83.9±2.2	4.6±0.5	0.4±0.5	1.6±0.5	1.47±0.05
HGS (calibration + earlier)	86.9±1.2	82.9±1.2	1.6±0.8	0.6±0.5	3.0±0.0	2.69±0.04
HGS (full grid)	**91.6±0.5**	**88.6±0.7**	**0.0±0.0**	**0.0±0.0**	**1.0±0.0**	1.39±0.02

Table 3: Window-level diagnostic metrics and closed-loop tuning results under the same data split. Ordinary stopping uses a fixed threshold of 0.5. The HGS columns show the results after adding history gating under the same historical information conditions. Relative increases in success rate are calculated from successful-episode counts. Superscript arrows indicate whether higher (↑) or lower (↓) values are better.

Classifier	Recall↑	FPR↓	Ordinary successes↑	Ordinary FN↓	HGS successes↑	HGS FN↓	Rel. gain↑
peak_final	94.04	**8.02**	108/136 (79.4%)	17	122/136 (89.7%)	**0**	13.0
peak_v2a	**96.57**	12.17	**112/136 (82.4%)**	**13**	**124/136 (91.2%)**	**0**	10.7
peak_v3	94.51	8.40	105/136 (77.2%)	20	123/136 (90.4%)	**0**	**17.1**

Training/model selection: `peak_final`, window-level mAP; `peak_v2a`, increased faulty-sample weights and decision F1; `peak_v3`, high recall under an FPR constraint.

Table 4: Cross-analyte history transfer results on sibutramine at m/z 279. Values are the mean ± standard deviation over five random splits. PEG denotes PEG-400 history; Sib denotes sibutramine calibration history. History coverage is the proportion of evaluation states retrieved by exact configuration matching. Superscript arrows indicate whether higher (↑) or lower (↓) values are better.

Method	SR↑	Fault-start SR↑	FN↓	New faults↓	Rescans↓	Coverage↑
Fixed threshold	**98.0±0.4**	**95.7±0.9**	3.0±0.63	**0.6±0.49**	**2.82±0.05**	—
Conformal	97.5±0.5	94.6±1.1	0.6±1.20	2.0±0.89	3.05±0.12	—
HGS + PEG	**98.0±0.4**	**95.7±0.9**	**0**	1.4±0.80	2.95±0.05	95.5
HGS + Sib, N=5	**98.0±0.4**	**95.7±0.9**	2.4±1.36	0.8±0.75	2.83±0.07	14.5
HGS + Sib, N=20	**98.0±0.4**	**95.7±0.9**	1.4±1.74	1.0±0.63	2.86±0.09	46.5
HGS + Sib, N=38	**98.0±0.4**	**95.7±0.9**	1.4±1.74	1.0±0.63	2.86±0.09	72.0

false-positive count. New faults are fault types appearing after adjustment, and rescans measure additional spectral accesses.

With full-history assistance, HGS reaches 91.6±0.5% success, 7.4 percentage points above the fixed threshold's 84.2±1.2%. Fault-start success rises from 78.7±1.6% to 88.6±0.7%, and average FN drops from 13.6 to 0. HGS uses 1.39 rescans, only 0.19 more than the fixed threshold. Conformal stopping also reaches zero FN, but with 87.5±0.5% success and 3.27 rescans. Thus, full-history HGS improves success by 4.1 points and reduces rescans by about 57.5% relative to conformal stopping, showing that avoiding faulty acceptance alone is insufficient; new faults and scan costs also matter.

The comparison also clarifies what kind of reliability is needed in closed-loop tuning. A conservative policy can eliminate premature stopping by continuing adjustment more often, but this may spend

scans and expose the trajectory to additional parameter moves. HGS differs because the extra caution is tied to the current configuration's historical risk rather than applied uniformly to all uncertain spectra. This makes it possible to reduce false stopping while preserving a relatively low rescan count.

With calibration-only history, HGS obtains 87.8±2.0% success, similar to calibrated and conformal stopping, while reducing average FN from 13.6 to 4.6. Under sequential history, success is 86.9±1.2%, but FN decreases further to 1.6. Without full instrument history, HGS mainly reduces premature stopping at the cost of more rescans; sequential history is therefore a safety–cost trade-off, not an across-the-board advantage.

4.2 Prediction–decision misalignment and robustness across classifiers

To analyze the relationship between window-level classification performance and closed-loop tuning outcomes, we further compare three MLP versions using the same peak-based split. All three are trained on the PEG-400 peaks at m/z 344, 388, 432, and 520 and tested on the peak at m/z 476. Window-level metrics are computed over 3,594 test windows at m/z 476; precision–recall and false-positive behavior are particularly relevant in imbalanced fault-detection settings (Davis & Goadrich, 2006). Closed-loop evaluation uses 136 episodes starting with faults among the 184 parameter combinations, with identical repair rules and a budget of five rescans.

The three classifiers show no monotonic relationship between window-level recall and closed-loop success. peak_v2a increases recall from 94.04% to 96.57% and ordinary successes from 108 to 112. peak_v3 also improves recall over peak_final, but successes decrease to 105. Likewise, peak_v2a has the highest FPR but not the lowest closed-loop success. Recall or FPR alone therefore cannot determine classifier performance inside the loop.

This result is expected when predictions are coupled to actions. A false positive is not merely a window-level counting error; it can trigger a repair step that changes the spectrum and may introduce a new fault. Conversely, a false negative near the beginning of a trajectory can terminate the episode before any repair is attempted. The effect of a classifier is therefore mediated by the policy, the repair priority, and the remaining scan budget.

After HGS is added, successful episodes increase from 108, 112, and 105 to 122, 124, and 123, with FN reduced to 0 for all three classifiers. Configuration history therefore corrects unreliable stopping decisions across MLP checkpoints, rather than only one base classifier.

4.3 Cross-analyte history transfer and the performance limit of the repair engine

Because HGS relies on parameter-conditioned fault history, we test whether instrument history remains useful when the analyte changes. We evaluate PEG-400 history and different amounts of sibutramine history on the sibutramine target peak at m/z 279.

The sibutramine data contain 6,188 spectra and 192 parameter combinations, of which 87 have at least one fault in their initial spectra and 105 have normal initial spectra. The experiment uses five stratified 20%/80% splits at the parameter-combination level. Each split uses 38 combinations for calibration and 154 for evaluation, with 70 combinations starting with faults in the evaluation set. All experiments freeze the peak_final classifier, expert repair engine, and five-rescan budget determined on PEG-400.

Under cross-analyte transfer, HGS accesses only PEG-400 parameter history and no sibutramine calibration or evaluation data. For sibutramine history, records come only from calibration combinations in the corresponding split; evaluation combinations are never added to memory. This compares existing cross-analyte history with small same-analyte history without using evaluation data as priors.

PEG-400 history covers 95.5% of sibutramine evaluation states by exact configuration matching. It reduces average FN from 3.0 to 0, showing that PEG-400 parameter risk can flag faulty sibutramine spectra accepted by the base classifier. Rescans increase from 2.82 to 2.95 and new faults from 0.6 to 1.4, so cross-analyte history mainly reduces incorrect stopping rather than improving all metrics for free.

The transfer is plausible because the history is indexed by instrument parameters rather than by molecular identity. Some regions of the parameter grid can be generally risky for peak quality even when the target ion changes, while the detailed spectral appearance remains analyte dependent. HGS uses this shared parameter-level information only at the stopping stage, which explains why it can reduce false acceptance without fully removing errors caused by repair direction.

Compared with conformal stopping, HGS with PEG-400 history has similarly low FN and, here, higher success, fewer new faults, and fewer rescans. With only small sibutramine calibration histories, the effect is limited by coverage. From N=5 to N=20, exact-match coverage increases from 14.5% to 46.5%, and average FN drops from 2.4 to 1.4. At N=38, coverage reaches 72.0%, but FN remains 1.4, suggesting a plateau around N=20.

HGS improves the decision about whether to stop, not the optimality of the next adjustment. Cross-analyte history can reduce premature stops but cannot exceed the repair engine's capability; when failures come from wrong adjustment directions or limited budget, better repair policies or action planning are needed.

These observations emphasize why IonTrapBench reports both diagnostic and trajectory-level quantities. A method that improves the stopping decision may still leave residual failures if the subsequent repair step is constrained by a one-level grid move or by an incorrect fault priority. Conversely, a repair rule that is adequate under one stopping policy can become less effective when the policy visits different states. Reporting FN, new faults, rescans, and final success together therefore gives a more complete picture of what changes in the loop.

Overall, parameter-associated fault history transfers across PEG-400 and sibutramine on the same instrument. High-coverage PEG-400 history reduces average FN to 0 without accessing sibutramine evaluation data, while same-analyte history improves with coverage and then plateaus. The identical final success rates reveal the boundary between stopping and repair: history gating reduces incorrect stopping but cannot correct a wrong adjustment direction.

5 Conclusion

To address autonomous tuning of miniature ion trap mass spectrometers, we construct IonTrapBench, organizing experimentally acquired spectra, discrete instrument parameters, annotations for four fault types, and post-adjustment state transitions into a unified offline replay task. On this basis, we establish a modular MLP–LLM closed-loop tuning system and find through controlled classifier retraining experiments that window-level recall and false-positive rate cannot fully predict the actual effectiveness of a classifier after it is placed in a closed loop. To improve the reliability of stopping decisions, we propose History-Gated Stopping, which supplements MLP spectral diagnoses using the historical fault rate associated with the current parameter configuration. Under full history, history gating increases the average tuning success rate from 84.2% to 91.6% and reduces the average number of false negatives from 13.6 to 0, showing that configuration-related history can provide an effective basis for decisions beyond classifier outputs. Overall, IonTrapBench provides a unified evaluation framework for spectral diagnosis, stopping decisions, and closed-loop tuning, while History-Gated Stopping provides a way to use instrument experience without retraining the base classifier. The experimental results show that incorporating both current spectra and parameter-configuration history into decisions helps reduce premature stopping and improve the reliability of autonomous tuning. More broadly, the results suggest that autonomous analytical instruments should be evaluated by the quality of the accepted final state, not only by the accuracy of intermediate diagnostic models. The replay setting also offers a practical intermediate step between static spectral benchmarks and fully online instrument studies, because it preserves the causal order of tuning decisions while keeping comparisons reproducible. Future work will further expand the range of instruments, substances, and target peaks, conduct online closed-loop validation, and jointly optimize stopping policies and parameter repair policies to advance more reliable autonomous operation of miniature mass spectrometers.

AI USE STATEMENT

We used large language models only as general writing aids for light proofreading and language refinement in grammar, phrasing, and minor style. The research ideas, problem formulation, methodology, experiments, analyses, code, and substantive writing were conceived and authored by us. No large language model generated new content beyond localized edits, and none was used to design experiments, analyze data, write code, or draft sections. All text was reviewed and verified by the authors, who take full responsibility for the paper, and the large language model should not be regarded as a contributor.

REPRODUCIBILITY STATEMENT

The manuscript specifies the data acquisition grid, window construction, classifier architecture and training settings, replay protocol, metric definitions, and history-access conditions. The appendices retain the full implementation and dataset description from the English manuscript. The data, annotations, fixed splits, and replay evaluation materials are planned for release.

REFERENCES

A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, and T. Schuster. Conformal Risk Control. In *The Twelfth International Conference on Learning Representations*, 2024.

B. Burger, P. M. Maffettone, V. V. Gusev, et al. A mobile robotic chemist. *Nature*, 583:237–241, 2020.

R. Bushuiev, A. Bushuiev, N. F. de Jonge, et al. MassSpecGym: A benchmark for the discovery and identification of molecules. In *Advances in Neural Information Processing Systems*, volume 37, pp. 110010–110027, 2024.

R. B. Canty and M. Abolhasani. The past, present and future of self-driving laboratories. *Nature Reviews Chemistry*, 10:523–537, 2026.

J. Davis and M. Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233–240, 2006. doi: 10.1145/1143844.1143874.

P. L. Donti, B. Amos, and J. Z. Kolter. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5484–5494, 2017.

K. Dührkop, H. Shen, M. Meusel, J. Rousu, and S. Böcker. Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proceedings of the National Academy of Sciences*, 112(41):12580–12585, 2015. doi: 10.1073/pnas.1509788112.

A. N. Elmachtoub and P. Grigas. Smart "Predict, then Optimize". *Management Science*, 68(1):9–26, 2022. doi: 10.1287/mnsc.2020.3922.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *PMLR*, pp. 1050–1059, 2016.

Y. Geifman and R. El-Yaniv. SelectiveNet: A Deep Neural Network with an Integrated Reject Option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *PMLR*, pp. 2151–2159, 2019.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *PMLR*, pp. 1321–1330, 2017.

K. Haug, R. M. Salek, P. Conesa, J. Hastings, P. de Matos, M. Rijnbeek, T. Mahendraker, M. Williams, S. Neumann, P. Rocca-Serra, E. Maguire, A. González-Beltrán, S. A. Sansone, J. L. Griffin, and C. Steinbeck. MetaboLights–an open-access general-purpose repository for metabolomics studies and associated meta-data. *Nucleic Acids Research*, 41(D1):D781–D786, 2013. doi: 10.1093/nar/gks1004.

H. Horai, M. Arita, S. Kanaya, et al. MassBank: a public repository for sharing mass spectral data for life sciences. *Journal of Mass Spectrometry*, 45(7):703–714, 2010.

F. Häse, L. M. Roch, and A. Aspuru-Guzik. Phoenics: A Bayesian Optimizer for Chemistry. *ACS Central Science*, 4(9):1134–1145, 2018. doi: 10.1021/acscentsci.8b00307.

S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *PMLR*, pp. 448–456, 2015.

R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, A. Sparkes, K. E. Whelan, and A. Clare. The Automation of Science. *Science*, 324(5923):85–89, 2009. doi: 10.1126/science.1165620.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *The Third International Conference on Learning Representations*, 2015. URL `https://arxiv.org/abs/1412.6980`.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6402–6413, 2017.

T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017. doi: 10.1109/ICCV.2017.324.

B. P. MacLeod, F. G. L. Parlane, T. D. Morrissey, et al. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6(20):eaaz8867, 2020.

L. Martens, M. Chambers, M. Sturm, D. Kessner, F. Levander, J. Shofstahl, W. H. Tang, A. Römpp, S. Neumann, A. D. Pizarro, L. Montecchi-Palazzi, N. Tasman, M. Coleman, F. Reisinger, P. Souda, H. Hermjakob, P. A. Binz, and E. W. Deutsch. mzML–a Community Standard for Mass Spectrometry Data. *Molecular & Cellular Proteomics*, 10(1):R110.000133, 2011. doi: 10.1074/mcp.R110.000133.

H. Mozannar and D. Sontag. Consistent Estimators for Learning to Defer to an Expert. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *PMLR*, pp. 7076–7087, 2020.

J. Perdomo, T. Zrnic, C. Mendler-Dünner, and M. Hardt. Performative Prediction. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *PMLR*, pp. 7599–7609, 2020.

Y. Perez-Riverol, A. Csordas, J. Bai, M. Bernal-Llinares, S. Hewapathirana, D. J. Kundu, A. Inuganti, J. Griss, G. Mayer, M. Eisenacher, E. Pérez, J. Uszkoreit, J. Pfeuffer, T. Sachsenberg, S. Yilmaz, S. Tiwary, J. Cox, E. Audain, M. Walzer, A. F. Jarnuczak, T. Ternent, A. Brazma, and J. A. Vizcaíno. The PRIDE database and related tools and resources in 2019: improving support for quantification data. *Nucleic Acids Research*, 47(D1):D442–D450, 2019. doi: 10.1093/nar/gky1106.

L. M. Roch, F. Häse, C. Kreisbeck, T. Tamayo-Mendoza, L. P. E. Yunker, J. E. Hein, and A. Aspuru-Guzik. ChemOS: Orchestrating autonomous experimentation. *Science Robotics*, 3(19):eaat5559, 2018.

H. L. Röst, U. Schmitt, R. Aebersold, and L. Malmström. pyOpenMS: a Python-based interface to the OpenMS mass-spectrometry algorithm library. *Proteomics*, 14(1):74–77, 2014. doi: 10.1002/pmic.201300246.

G. Shafer and V. Vovk. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, 9:371–421, 2008. URL https://www.jmlr.org/papers/v9/shafer08a.html.

D. T. Snyder, C. J. Pulliam, Z. Ouyang, and R. G. Cooks. Miniature and fieldable mass spectrometers: Recent advances. *Analytical Chemistry*, 88(1):2–29, 2016.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15: 1929–1958, 2014. URL https://jmlr.org/papers/v15/srivastava14a.html.

M. Wang, J. J. Carver, V. V. Phelan, et al. Sharing and community curation of mass spectrometry data with Global Natural Products Social Molecular Networking. *Nature Biotechnology*, 34:828–837, 2016.

B. Wilder, B. Dilkina, and M. Tambe. Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1658–1665, 2019. doi: 10.1609/aaai.v33i01.33011658.

Y. Zhai, X. Fu, and W. Xu. Miniature mass spectrometers and their potential for clinical point-of-care analysis. *Mass Spectrometry Reviews*, 43(5):1172–1191, 2024.

M. L. Zhang and Z. H. Zhou. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014. doi: 10.1109/TKDE.2013.39.

A COMPLETE IMPLEMENTATION AND DATASET DETAILS

A mass spectrometer generates a mass spectrum by measuring the mass-to-charge ratios and relative intensities of ions, and local peak shapes can reflect the detection state under the current instrument parameter configuration. In a normal spectrum, the target peak has an appropriate signal intensity, a relatively symmetric shape, and clear unit mass separation, while background noise remains low. As shown in Figure 2, this work focuses on four common types of abnormal spectra: saturation appears as a flattened peak top or a plateau; peak tailing appears as pronounced peak asymmetry with an extension to one side; insufficient unit mass resolution appears as an inability to clearly separate adjacent ion peaks; and a low signal-to-noise ratio appears as a target peak that is too weak relative to the background noise. Based on these observable local spectral features, we represent spectral states using a normal label and the four fault labels above.

Spectrum-guided instrument tuning involves more than determining whether the spectrum under a particular parameter configuration contains faults. Its ultimate goal is to bring the instrument to a parameter configuration that produces an acceptable spectrum through a sequence of control actions. To this end, we construct a modular MLP–LLM closed-loop tuning system: an MLP classifier provides window-level spectral diagnoses, and a separate LLM tuning agent converts these diagnoses into tuning decisions. However, MLP retraining reveals a misalignment between predictive performance and downstream decision effectiveness. We formalize this process as the IonTrapBench tuning replay task and further propose History-Gated Stopping to reduce the effects of unreliable stopping decisions on tuning outcomes and mitigate the losses caused by this misalignment.

A.1 MLP-BASED SPECTRAL FAULT CLASSIFICATION AND LLM CLOSED-LOOP TUNING

We first establish an automatic tuning procedure in which an MLP and an LLM work together. As shown in Figure 3a, the base tuning system consists of two functionally independent but coordinated modules. Spectra generated by the mass spectrometer are input to the MLP classifier, which diagnoses the spectral state under the current instrument parameter configuration. The LLM tuning agent interprets the MLP's diagnostic results under a fixed control protocol and decides whether to accept the current configuration or continue adjustment. In the replay experiments, the LLM tuning agent is implemented with DeepSeek-V4-Pro under the fixed control protocol. Thus, the MLP does not directly control the instrument, and the LLM does not replace the spectral classifier. Together, they form a diagnosis–decision closed loop: the MLP provides diagnostic evidence about the current state, and the LLM generates the next control decision accordingly. If adjustment is required, the decision is fed back to the instrument, which generates another spectrum for assessment; otherwise, the spectrum is output.

For a target ion with theoretical peak center m_0, we search within ± 6 Da of m_0 in the spectrum at tuning step t to locate the point of maximum intensity, denoted by $\hat{m}_t$. We then extract a window within ± 2 Da of this detected peak center and resample it by linear interpolation into a 400-point intensity vector:

$$\mathbf{x}_t = \mathcal{I}_{400}\left(y_t[\hat{m}_t - 2, \hat{m}_t + 2]\right) \in \mathbb{R}^{400}. \tag{5}$$

Here, $\mathcal{I}_{400}$ denotes linear interpolation. This representation preserves the local shape of the target peak while providing a fixed-dimensional input to the classifier. The reference labels for the four faults are represented by a four-dimensional multilabel vector:

$$\mathbf{y}_t^{\text{fault}} = (y_t^{\text{sat}}, y_t^{\text{tail}}, y_t^{\text{res}}, y_t^{\text{snr}}) \in \{0,1\}^4. \tag{6}$$

The classifier is a multilayer perceptron with layer widths of $400 \to 256 \to 128 \to 64 \to 5$. Each of the three hidden layers applies BatchNorm, ReLU, and Dropout in sequence, with dropout rates of 0.3, 0.3, and 0.2, respectively (Ioffe & Szegedy, 2015; Srivastava et al., 2014). The output layer uses five sigmoid units to produce one normal score and four fault scores:

$$\mathbf{z}_t = (p_t^{\text{normal}}, p_t^{\text{sat}}, p_t^{\text{tail}}, p_t^{\text{res}}, p_t^{\text{snr}}). \tag{7}$$

Because multiple faults may occur simultaneously in the same spectral window, the output layer uses independent sigmoid units rather than mutually exclusive softmax classification (Zhang & Zhou, 2014). To mitigate class imbalance across labels, the model is trained using focal binary cross-entropy with a focusing parameter of $\gamma = 2$, together with positive-class weighting and weighted random sampling (Lin et al., 2017). Positive-class weights are determined by the ratio of negative to positive samples for each label and clipped to the range from 1 to 10. The model is trained for 30 epochs using the Adam optimizer, with an initial learning rate of 10^{-3}, weight decay of 10^{-4}, and a batch size of 256 (Kingma & Ba, 2015). The checkpoint with the highest validation mAP is retained.

At step t, the MLP output is converted into structured diagnostic results and provided to the LLM tuning agent together with the fixed control protocol. As shown in Figure 3a, the high-level action selected by the LLM is represented as:

$$a_t = \pi_{\mathrm{LLM}}(\mathbf{z}_t; \mathcal{P}) \in \{\mathrm{STOP}, \mathrm{ADJUST}\}. \tag{8}$$

Here, $\mathcal{P}$ denotes the LLM control protocol, which remains unchanged across comparisons. When the LLM selects STOP, the system accepts the current parameter configuration $\mathbf{s}_t$ and terminates the tuning episode. When the LLM selects ADJUST, the system invokes the deterministic expert repair rules R to update the parameter configuration according to the highest-priority fault currently detected:

$$\mathbf{s}_{t+1} = R(\mathbf{s}_t, \mathbf{z}_t). \tag{9}$$

Each adjustment changes only one instrument parameter and moves only one level along the predefined parameter grid. The system then reads or acquires the spectrum corresponding to the new configuration $\mathbf{s}_{t+1}$ and repeats window extraction, MLP diagnosis, and LLM decision making until the LLM selects STOP or the maximum scan budget is reached.

A.2 IonTrapBench data acquisition and dataset construction

IonTrapBench is based on parameter-grid acquisition experiments using a miniature linear ion trap mass spectrometer. The instrument uses a custom-built electrospray ion source and a continuous atmospheric pressure interface. It contains a two-stage differentially pumped vacuum system, a miniature ion funnel, and a linear ion trap, with the two functional regions connected by an ion lens. The experimental samples include PEG-400 and sibutramine. PEG-400 is used to construct the classifier training data and closed-loop tuning replay data, whereas sibutramine is used to evaluate diagnostic and tuning behavior on a substance outside the training data. The PEG-400 sample is diluted in methanol to 0.1 μg/mL; two concentration conditions are used for sibutramine to construct external evaluation data at different signal intensities.

Both PEG-400 and sibutramine spectra are acquired in positive-ion mode over a mass range of m/z 200–1000. We select four instrument parameters that directly affect spectral quality and can be adjusted through the control interface: sampling cone voltage, ion funnel RF control voltage, ion injection time, and mass scan time. The discrete values of each parameter are listed in Table 1. Instrument parameters not listed are held constant during acquisition.

Let $\mathcal{V}_{\mathrm{cone}}$, $\mathcal{V}_{\mathrm{funnel}}$, $\mathcal{V}_{\mathrm{inj}}$, and $\mathcal{V}_{\mathrm{scan}}$ denote the discrete value sets of the four parameters, respectively. The theoretical parameter grid is defined as:

$$\mathcal{G}_{\mathrm{full}} = \mathcal{V}_{\mathrm{cone}} \times \mathcal{V}_{\mathrm{funnel}} \times \mathcal{V}_{\mathrm{inj}} \times \mathcal{V}_{\mathrm{scan}}. \tag{10}$$

Mass spectra are acquired according to the combinations of discrete values above while all other experimental conditions are held constant. To characterize possible observational variation under the same parameter configuration, repeated acquisitions are performed at identical parameter settings, and each spectrum is stored together with the four-dimensional parameter vector used during acquisition. The reference acquisition records contain a total of 3,594 full PEG-400 mass spectra. For the planned public release, we also retain configuration identifiers, replicate observation identifiers, raw spectra, and corresponding parameter metadata, allowing users to distinguish parameter

configurations from their repeated observations. The theoretical parameter grid describes the predefined parameter search space, whereas the parameter grid actually available for replay contains only configurations with valid spectra and complete parameter metadata. The set of observed parameter configurations is denoted by:

$$\mathcal{G}_{\text{obs}} = \{\mathbf{s} \in \mathcal{G}_{\text{full}} : \text{ValidSpectrum}(\mathbf{s}) = 1\}. \tag{11}$$

The current version of IonTrapBench contains 184 PEG-400 parameter configurations and 192 sibutramine parameter configurations. Because the theoretical grid and the available grid differ in size, the public release will provide the theoretical parameter table, the index of actual configurations, and record availability information, so that combinations that were not acquired or have no valid spectra are not interpreted as ordinary missing values.

For each full spectrum, we construct local spectral windows around predefined theoretical target peaks. PEG-400 uses five theoretical peak centers at m/z 344, 388, 432, 476, and 520. To reduce the effect of slight peak-position shifts, we first search for the sampling point with the highest intensity within ± 6 Da of the theoretical peak center m_0 and use its position as the actual peak center in that spectrum:

$$\hat{m}_i = \underset{m \in [m_0 - 6, m_0 + 6]}{\arg\max} \; y_i(m). \tag{12}$$

We then extract the local spectral segment within ± 2 Da of the actual peak center $\hat{m}_i$ and resample it into 400 equally spaced data points by linear interpolation:

$$\mathbf{x}_i = \mathcal{I}_{400}\left(y_i[\hat{m}_i - 2, \hat{m}_i + 2]\right) \in \mathbb{R}^{400}. \tag{13}$$

This processing corresponds to 100 sampling points per 1 Da and reduces the effects of peak-position shifts and differences in the original sampling intervals on the model input. The same peak-position search, local extraction, and resampling procedure is applied to the sibutramine target peak. Using consistent peak-shape assessment criteria, experimental personnel annotate each local window for four faults: saturation, peak tailing, insufficient unit mass resolution, and a low signal-to-noise ratio. The fault labels are represented as:

$$\mathbf{y}_i = (y_i^{\text{sat}}, y_i^{\text{tail}}, y_i^{\text{res}}, y_i^{\text{snr}}) \in \{0, 1\}^4. \tag{14}$$

Multiple faults can occur simultaneously. When all four labels are 0, the window is defined as normal:

$$y_i^{\text{normal}} = \mathbb{I}\left[\sum_{k=1}^{4} y_{i,k} = 0\right]. \tag{15}$$

A window record in IonTrapBench consists of the substance, concentration condition, parameter configuration, target peak, local spectrum, fault labels, and data split:

$$d_i = (c_i, l_i, \mathbf{s}_i, m_i, \mathbf{x}_i, \mathbf{y}_i, \text{split}_i, \text{spectrumID}_i, \text{replicateID}_i). \tag{16}$$

Here, c_i denotes the substance, l_i the concentration condition, $\mathbf{s}_i$ the four-dimensional parameter configuration, m_i the target peak, $\mathbf{x}_i$ the 400-point spectral window, and $\mathbf{y}_i$ the fault labels. Spectrum files, target-peak windows, parameter configurations, and tuning episodes are different statistical units and are therefore counted separately in the data description and experimental results. The number of spectrum files is not directly interpreted as the number of independent parameter configurations or tuning episodes.

In the PEG-400 data, the 14,376 local windows corresponding to m/z 344, 388, 432, and 520 are used to train the MLP. The 3,594 windows at m/z 476 are excluded from classifier training and serve as evaluation data for an unseen target peak. This split tests whether the peak-shape features learned

by the classifier transfer to a mass-to-charge ratio excluded from training. The sibutramine data are not used for MLP training or parameter updates and are used to evaluate behavior on a substance outside the training data when historical records of the same substance are unavailable.

For closed-loop evaluation, PEG-400 parameter configurations are further divided into a 20% calibration set and an 80% evaluation set. The current fixed split contains 37 calibration configurations and 147 evaluation configurations, the latter including 109 configurations with faulty initial spectra and 38 with normal initial spectra. Different random seeds generate five random splits; because these evaluation sets may overlap, we do not describe this setting as five-fold cross-validation.

IonTrapBench includes, at minimum, full mass spectra, 400-point target-peak windows, fault labels, parameter configurations, replicate observation indices, training and evaluation splits, parameter-grid definitions, expert repair rules, tuning replay programs, and evaluation code. By releasing both spectral observations and parameter states, IonTrapBench supports not only window-level fault classification but also evaluation of the actual consequences of classification results in sequential parameter decisions.

A.3 IonTrapBench tuning replay task and prediction–decision misalignment

IonTrapBench organizes the configuration-level spectral records constructed in Section A.2 into offline closed-loop tuning tasks. A tuning episode begins from a given initial parameter configuration. The state at step t consists of the current parameter configuration $\mathbf{s}_{e,t}$ and its corresponding spectral window $w_{e,t}$. The LLM tuning agent selects STOP or ADJUST based on the MLP diagnosis. The trajectory of episode e is represented as:

$$\tau_e = \{(\mathbf{s}_{e,t}, w_{e,t}, a_{e,t})\}_{t=0}^{T_e}. \tag{17}$$

When STOP is selected, the system accepts the current configuration and terminates the episode. When ADJUST is selected, the deterministic expert repair rules change one parameter according to the current highest-priority fault, moving only one level among that parameter's discrete values:

$$\mathbf{s}_{e,t+1} = R(\mathbf{s}_{e,t}, \mathbf{z}_{e,t}). \tag{18}$$

Here, $\mathbf{z}_{e,t}$ denotes the MLP diagnostic scores at step t of episode e. The spectrum corresponding to the new configuration is returned by the dataset retrieval interface:

$$w_{e,t+1} = \mathrm{Lookup}(\mathcal{D}, \mathbf{s}_{e,t+1}, m_e, \mathrm{observationID}_{e,t+1}). \tag{19}$$

Here, $\mathcal{D}$ denotes the IonTrapBench dataset, and m_e denotes the target peak of the current episode. For configurations with repeated observations, the replay file explicitly specifies the observation identifier to ensure that different policies access the same spectral records. Each episode permits at most five rescans. Let $F(w)$ denote the reference fault set of a spectral window. Let $D_e = 1$ if episode e has the terminal outcome DEFER, and $D_e = 0$ otherwise. An episode is considered successful when the window corresponding to the terminal configuration contains none of the four faults and the episode has not been deferred:

$$S_e = \mathbb{I}\left[F(w_{e,T_e}) = \varnothing \wedge D_e = 0\right]. \tag{20}$$

For an evaluation set containing N episodes, the tuning success rate is defined as:

$$\mathrm{SR} = \frac{1}{N}\sum_{e=1}^{N} S_e. \tag{21}$$

Success is determined from the reference labels of the final spectrum and the deferral outcome, rather than from whether the classifier outputs a normal prediction. Let $B_e = \mathbb{I}[F(w_{e,0}) \neq \varnothing]$ indicate whether episode e starts with a fault, and let ρ_e denote its termination reason. In addition to the

success rate, we count false negatives, FA, and new-fault events. A false negative (FN) is an episode that starts with a fault, is unsuccessful, and terminates with the reason `stopped_by_model`:

$$\mathrm{FN} = \sum_{e=1}^{N} \mathbb{I}\left[B_e = 1 \wedge S_e = 0 \wedge \rho_e = \text{stopped_by_model}\right]. \tag{22}$$

FA counts episodes that start with a normal spectrum and are unsuccessful, including episodes that become faulty after adjustment or are deferred:

$$\mathrm{FA} = \sum_{e=1}^{N} \mathbb{I}\left[B_e = 0 \wedge S_e = 0\right]. \tag{23}$$

The deferral rate is the proportion of episodes with the terminal outcome DEFER:

$$\mathrm{DeferralRate} = \frac{1}{N} \sum_{e=1}^{N} D_e. \tag{24}$$

The success rate for episodes starting with faults is computed only over the subset of episodes that initially contain faults:

$$\mathrm{SR}_{\mathrm{fault}} = \frac{\sum_{e=1}^{N} B_e S_e}{\sum_{e=1}^{N} B_e}. \tag{25}$$

If a fault type that was absent before an adjustment appears afterward, it is counted as a new-fault event.

We compare the original MLP with two versions retrained for window-level objectives: one increases the weights of faulty samples, and the other optimizes fault recall under a false-positive-rate constraint. The initial cases, LLM control protocol, expert repair rules, and scan budget are fixed throughout the comparison; only the MLP classifier is replaced.

Table 3 shows the success rates of the original MLP and the two retrained models on the same set of 136 episodes with faulty initial spectra. The retrained classifiers do not consistently outperform the original model.

Trajectory analysis shows that the fault-weighted version produces more ineffective adjustments and new-fault events triggered by false positives. The version selected for high recall under an FPR constraint still produces false negatives and, in some episodes, subsequent false positives cause a recovering spectral state to deteriorate again. In the experiment that increases the weights of faulty samples and selects the model by decision F1, recall and tuning success rate improve together. However, in the retraining of peak_v3, recall improves while the tuning success rate decreases. This indicates that window-level recall and FPR cannot uniquely predict closed-loop tuning outcomes: classifiers with similar window-level metrics may produce different closed-loop success rates, and higher recall may accompany either an increase or a decrease in success rate. The two cannot be treated directly as the same criterion.

To address this problem, we further introduce a gating method based on historical parameter experience into decision making that combines the MLP and the large language model.

B Extended workflow caption

Window-level spectral diagnosis, prediction–decision misalignment, and History-Gated Stopping in IonTrapBench. (a) Window-level model selection: target-peak windows are extracted from full mass spectra, and the MLP spectral classifier and stopping threshold are selected and fixed according to window-level classification metrics to provide diagnoses for MLP–LLM closed-loop tuning. This stage does not consider historical information or the subsequent actions triggered by diagnoses. (b)

Prediction–decision misalignment: a false negative in a faulty state may trigger STOP and cause premature stopping, whereas a false positive in a normal or recovered state may trigger unnecessary ADJUST actions, increasing tuning costs or even introducing new faults. Window-level classification performance therefore cannot directly represent closed-loop tuning effectiveness. (c) Offline replay with History-Gated Stopping: with the MLP classifier and deterministic expert repair rules fixed, the normal score is discounted according to the historical fault rate of the current parameter configuration, after which STOP or ADJUST is selected according to ordered rules. Historical information is used only to increase the evidence required for stopping and does not directly specify the parameter repair direction.

C LLM TUNING-AGENT PROMPT

This appendix documents the exact prompt used by the LLM tuning agent of Appendix A.1. The prompt is a pure function of the current MLP diagnosis and the current parameter state; it is held *fixed in every arm in which the LLM agent is in the loop*: the stopping rule under comparison (plain threshold, calibrated baselines, or History-Gated Stopping) never modifies the prompt, the embedded knowledge, or the execution rules.

C.1 SYSTEM MESSAGE

```
You are a mass-spectrometer parameter-optimization expert.
Output JSON only.
```

C.2 REQUEST TEMPLATE

Placeholders in braces are filled at run time (format in Appendix C.4); everything else is fixed.

```
You are a tuning expert for a single linear ion trap mass spectrometer.
First reason step by step according to the decision procedure (write the
reasoning into the "reasoning" field), then give parameter-adjustment
suggestions.

## 1. Instrument working principle and physical laws
{INSTRUMENT PHYSICS}

## 2. Current scan parameters
{PARAMETERS}

## 3. Analysis chains for the faults detected this round (reason along these)
{FAULT ANALYSIS BLOCKS, one per detected fault}

## 4. Multi-fault conflict handling
{CONFLICT RULES}

## 5. MLP fault-diagnosis results
The MLP classifier's diagnosis of the current spectrum (per target-peak
window):
{FAULTS}

## 6. Decision procedure (fill the "reasoning" field step by step,
   one sentence per step)
1. Identify the dominant fault: rank by detected probability and state
   which is most severe;
2. Physical attribution: which physical chain above does the dominant
   fault correspond to;
3. Parameter check: what is the current level of that chain's first-choice
   parameter, and can it move in the target direction (if at a boundary,
   switch to the second-choice parameter);
4. Trade-off: could this adjustment aggravate another detected fault
   (consult the conflict rules);
5. Conclusion: which parameters, directions, and priorities are chosen.

## 7. Output format
Output exactly one JSON object (nothing besides the JSON):
{"reasoning": "1. dominant fault ... 2. physical cause ... 3. parameter
check ... 4. trade-off ... 5. conclusion ...",
 "adjustments": [{"param": "<name>", "direction": "increase|decrease",
                  "priority": 1}, ...]}

Rules:
```

```
1. direction must be increase or decrease, meaning a move to the adjacent
   level;
2. priority is a positive integer, duplicates not allowed, 1 is highest;
   priority=1 must go to the adjustment that most directly addresses the
   dominant fault and whose direction is movable;
3. list only parameters that need adjustment; with no faults,
   adjustments is an empty list;
4. a direction already at a level boundary must not be chosen (the
   parameter block below marks boundaries);
5. before increasing injection time, confirm that no saturation fault is
   present (see conflict rules).
```

C.3 Injected Knowledge Blocks

The following blocks are embedded in Sections 1, 3, and 4 of every request.

Instrument physics (Section 1).

```
Ion path: electrospray ionization -> sampling cone (cone_voltage_v,
transfer voltage) -> ion funnel (funnel_amplitude, RF focusing) ->
linear ion trap (injection_time_ms, ion accumulation time, sets the
trap ion population) -> scan-out detection (scan_time_ms, scan speed,
sets how finely peaks are sampled).

Core physical laws:
1. [Space charge] With excess trap ions, Coulomb repulsion broadens
   peaks (resolution loss), distorts peak shape (tailing), and saturates
   the detector. Saturation, tailing, and insufficient resolution can
   share the "too many ions" origin.
2. [Scan time is the first-order factor for resolution] Longer
   scan_time_ms ejects ions more slowly and samples each peak more
   finely, raising resolution.
3. [Excess cone voltage -> in-source fragmentation] produces fragment
   and shoulder peaks that worsen resolution; it also raises transfer
   efficiency, increasing the trapped ion population.
4. [Excess funnel amplitude -> unstable ion motion] uneven transfer
   causes peak tailing.
5. [SNR ~ sqrt(ion count)] accumulating more ions (longer injection
   time) or scanning more slowly (longer scan time) both raise SNR.
```

Per-fault analysis chains (Section 3; only detected faults are included).

```
[saturated]
Physical chain: overlong injection time or excess cone voltage ->
excess trap ions -> detector signal too strong / clipped peak top.
Steps: (1) check injection_time_ms: if above the minimum level 20,
decrease it first (the most direct way to reduce ion load); (2) if
injection time is already 20, check cone_voltage_v: decreasing it
reduces ion transfer; (3) note that saturation often accompanies
space-charge broadening, so reducing ion load can also relieve
resolution problems.

[tailing]
Physical chain: excess funnel amplitude -> unstable ion beam; or excess
trap ions -> space-charge scan-out tailing.
Steps: (1) check funnel_amplitude: if above the minimum level 0.1,
decrease it first; (2) if already at 0.1, decrease injection_time_ms
(relieves space-charge tailing).

[non_unit_resolution]
Physical chain: three possibilities -- scan too fast (short scan_time,
too few points per peak); space charge (excess ions, broadening); excess
cone voltage (in-source fragmentation, overlapping/shoulder peaks).
Steps: (1) check scan_time_ms: if below the maximum level 800, increase
it first (the first-order factor for resolution on this instrument);
(2) if scan_time_ms is already 800, the broadening may be space charge:
decrease injection_time_ms; (3) if cone_voltage_v >= 40, decrease it to
```

```
reduce fragmentation.

[low_snr]
Physical chain: three possibilities -- insufficient trap ions (short
injection time); scan-out too fast (insufficient averaging); cone
voltage at an extreme level (too low, 5-10 V: poor transmission, few
ions reach the trap; too high, 40-45 V: fragmentation loses precursor
ions) -- both weaken the signal.
Steps: (1) check cone_voltage_v for extremes: <=10 V should be raised
toward the 15-35 V mid-range; >=40 V should be lowered to 35. An
extreme cone voltage is the most common low_snr root cause on this
instrument's grid and is handled first; (2) check injection_time_ms: if
below the maximum level 200 and the cone voltage is not extreme,
increase it; (3) if injection time is already 200, increase scan_time_ms;
(4) side-effect warning: if saturation is also detected, increasing
injection time is forbidden -- increase scan_time_ms or adjust the cone
voltage instead.
```

Multi-fault conflict rules (Section 4).

```
- low_snr + saturated: opposite demands on injection_time_ms (one wants
  more ions, the other fewer). Prefer increasing scan_time_ms; if cone
  voltage >= 40 V, lower it (relieves saturation and fragmentation
  losses at once); do not touch injection_time_ms.
- low_snr + non_unit_resolution: increasing scan_time_ms improves both
  (first choice; leave injection time alone).
- saturated + non_unit_resolution: both may stem from excess ions:
  decreasing injection_time_ms first may improve both.
- tailing + low_snr: lowering funnel amplitude costs transmission and
  SNR: first lower funnel_amplitude to fix tailing, then compensate SNR
  with scan_time_ms; do not increase injection time (aggravates
  ion-overload tailing).
```

C.4 Runtime Context Format

Each parameter is rendered with its current level, the full list of discrete levels, and explicit boundary flags, e.g.:

```
- injection_time_ms:  current=20, levels=[20, 200], at
minimum level (cannot decrease), can increase.  Controls
how long ions are injected into the trap; longer injection
accumulates more ions and stronger signal.
```

Diagnosis results are rendered per target peak with probabilities, e.g.:

```
m/z 476:  faults detected - tailing(0.81), low_snr(0.62)
```

C.5 Decoding, Parsing, and Determinism

The agent uses temperature 0.2 and a 4096-token budget. The response is parsed by robust JSON extraction (tolerating markdown fences and truncation); a fully unparseable response yields an empty adjustment list, and the loop re-queries at the next step under the replay budget. Every (system message, prompt) pair is cached on disk by its content hash, so all arms, seeds, and repeat runs receive byte-identical answers for identical states.