STELLA-FLOW: DYNAMIC DATA RECIPE VALUATION FOR DATA-EFFICIENT MODEL ITERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language models mature, development is shifting from a model-centric to a data-centric paradigm. Model capabilities depend not only on data scale, but also on data quality, diversity, and composition. However, existing studies largely focus on individual-sample valuation or aggregate metric optimization, leaving the dynamic value of complete data recipes under evolving model feedback insufficiently characterized. We propose STELLA-Flow, a dynamic framework that evaluates candidate data through intrinsic quality, conditional model gain, and business value. It first assesses data quality, then uses a small number of real model updates to predict which existing errors candidate data recipes would correct and which new errors they would introduce, and finally incorporates scenario importance and error costs to select data for the current model. Across model scales, STELLA-Flow outperforms random selection and static quality filtering. Under a unified advertising content moderation benchmark, a 2B model updated with STELLA-Flow-selected data outperforms a 32B model updated with randomly selected data, GLM-5.3, and DeepSeek-V4-Pro. These results demonstrate that STELLA-Flow provides a unified method for dynamic data valuation and selection across model scales, enabling limited training budgets to continuously shift toward higher-utility, lower-risk data recipes as model capabilities evolve. This improves the data efficiency of iterative model development and offers a new path for smaller models to achieve competitive business performance at lower cost.

1 INTRODUCTION

As foundation models and training technologies mature, further improvements in model capability increasingly depend on how training data are organized, selected, and reused throughout model iteration, rather than on parameter count or data scale alone. Data quality is essential, but it does not fully determine the value of a dataset for a particular update. Under the same training protocol, a data recipe may repair capability gaps in some scenarios while introducing false positives or false negatives in others. Training-data value is therefore conditional on the current model state, update protocol, and application objective. Data selection must answer not only whether the data are usable, but also how training on them would change the current model and whether that change is worth executing.

Existing research addresses parts of this problem through data curation, example valuation, and data-mixture optimization. Quality-oriented methods filter data using reliability, deduplication, and diversity signals; model-aware valuation methods estimate the influence of individual examples or training-set composition; and mixture-optimization methods predict validation loss or aggregate downstream performance (Li et al., 2024; Penedo et al., 2024; Peng et al., 2025; Ilyas et al., 2022; Xia et al., 2024; Engstrom et al., 2024; Liu et al., 2025; Ye et al., 2025; Chen et al., 2025). These lines of work leave a central gap in iterative model development: estimating, at the level of a complete data recipe, which existing errors an update would repair, which new errors it would introduce, and whether the resulting behavioral change justifies replacing the current model.

We propose **STELLA-Flow**, a dynamic framework that connects three sequential layers of data value. *Intrinsic data quality* determines whether candidate data satisfy the requirements for downstream use. *Model-conditional gain* predicts how an eligible recipe changes the current model at the level of individual evaluation cases. *Business utility* aggregates the predicted repairs and regressions

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.

under scenario importance, error costs, training budget, and uncertainty. The three layers form a constrained decision process from data admission to behavioral prediction and update authorization, rather than independent scores to be averaged (Figure 1).

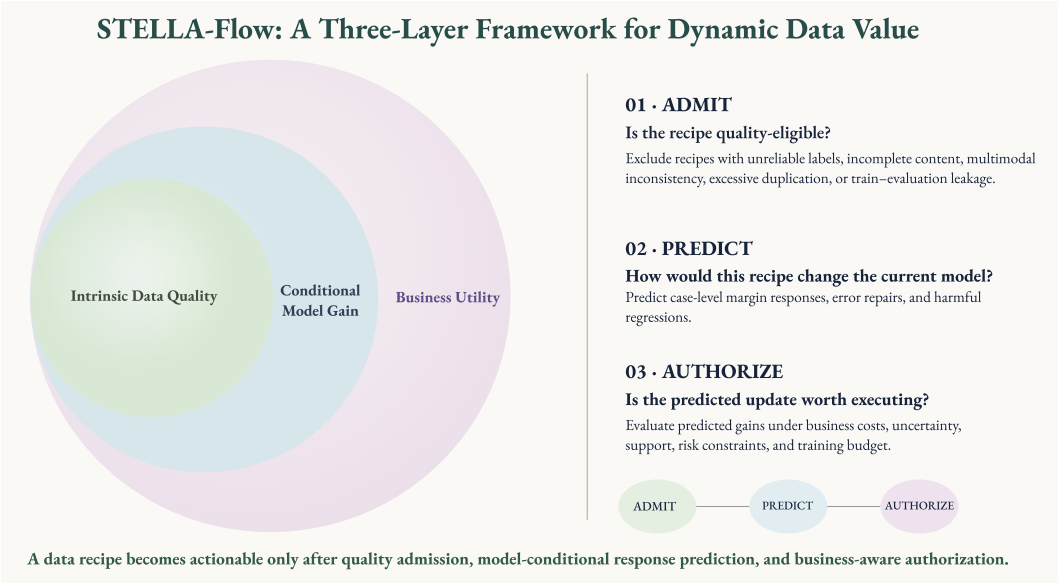

Figure 1: The three-layer framework of STELLA-Flow.

The model-conditional layer is implemented by **STELLA**, a structured response predictor learned from real model updates. Starting from the same model checkpoint and training protocol, we train a set of recipes and record their margin changes over fixed case-level evaluation units. STELLA extracts shared low-rank response modes from the resulting recipe–response matrix and learns how recipe composition determines the coefficients of these modes. For an unseen recipe, it reconstructs the complete case-level response, from which error repairs, newly introduced errors, and business utility can be estimated before training that recipe.

STELLA-Flow further separates recipe ranking from update authorization. A top-ranked recipe is only the strongest candidate within the available pool; it may still be inferior to retaining the current checkpoint, whose response utility is zero. STELLA-Flow therefore combines predicted utility, empirical uncertainty, and input support to assign each candidate to EXECUTE, PROBE, or REJECT. A controlled PROBE contributes a newly observed response for predictor recalibration without immediately replacing the current model, whereas an EXECUTE decision advances the model state and triggers revaluation of the remaining candidates. This feedback loop allows recipe values to evolve with both accumulated evidence and model capability.

Experiments across model scales and architectures demonstrate the complementary roles of the three value layers. On Qwen3.5-0.8B, intrinsic quality combined with model-conditional gain achieves the highest F1 with the lowest average training cost, while the complete framework achieves the highest business value. On Qwen3.5-2B, STELLA-Flow improves business value by 6.4326 points over random selection and outperforms Human Strategy, MIWV, and NAIT. The STELLA-selected 0.8B model reaches 97.5% of the 2B model's business value using 67.3% less measured training compute, and the 2B model exceeds the Qwen3.5-27B reference in business value. A separate study of 44 independently trained 0.8B recipes reveals why update authorization is necessary: STELLA predicts every realized utility sign correctly, yet the strongest observed recipe still has utility -27. These outcomes show why ranking candidates and authorizing an update are distinct decisions.

Contributions. We make three contributions: (i) a three-layer formulation that connects intrinsic data quality, model-conditional gain, and business utility in iterative model development; (ii) a low-rank case-level response model learned from real updates to predict the repairs, regressions, and

utility induced by unseen data recipes; and (iii) a feedback-driven selection mechanism that separates recipe ranking from update authorization and dynamically recalibrates recipe values as new responses and model states become available.

2 Related Work

Existing training-data selection methods broadly follow three directions. DataComp-LM, FineWeb, and DEITA curate examples using quality, deduplication, complexity, and diversity signals; LESS, DataInf, and SelectIT estimate the model-specific value of individual examples through gradient influence or predictive uncertainty; and Datamodels, DsDm, RegMix, and FastMix learn relationships between data composition and model performance to predict or optimize data mixtures (Li et al., 2024; Penedo et al., 2024; Liu et al., 2024; Xia et al., 2024; Kwon et al., 2024; Liu et al., 2024; Ilyas et al., 2022; Engstrom et al., 2024; Liu et al., 2025; Tan et al., 2026). However, these capabilities remain separated across the literature: quality filtering determines whether data are usable, example valuation estimates the relative contribution of individual samples, and mixture optimization commonly predicts validation loss or aggregate performance. They do not jointly characterize which errors a complete data recipe would repair, which new errors it would introduce, and whether the resulting model update should be authorized. STELLA-Flow addresses this gap by treating a complete data recipe as the intervention, learning case-level margin responses from real model updates, and integrating repairs, regressions, business costs, and empirical uncertainty into update authorization. Responses collected from controlled probes are further used to recalibrate the predictor as the model state evolves. STELLA-Flow thus unifies these previously separated capabilities through quality admission, model-conditional response prediction, and business-aware update authorization. This formulation enables controlled comparisons among sample-level selection, scalar recipe prediction, and structured response modeling, while isolating the individual and joint contributions of intrinsic data quality, model-conditional gain, and business utility.

3 Problem Formulation

3.1 Conditional Recipe Updates

Let M_t be the current model at iteration t and τ_t the fixed update protocol, including the optimizer, training budget, replay data, and random seed. A data recipe a is a finite training set drawn from a candidate pool. Its counterfactual update is

$$M_{t,a} = \mathrm{Train}(M_t, a; \tau_t). \tag{1}$$

Recipes used to observe update effects are trained independently from the same M_t under τ_t; the effects of other candidates must be predicted before their updates are run.

Suppose the pool is divided into S business scenarios. We represent recipe a by $x_a = (x_{a,1}, \ldots, x_{a,S})^\top$, where $x_{a,s} = n_{a,s}/N_a$, $N_a = \sum_s n_{a,s}$, and $n_{a,s}$ counts its examples from scenario s. Fixed replay data belong to τ_t and are excluded from N_a. Within a comparison round, recipe size and update protocol are fixed. The representation describes scenario composition, not the identities of individual examples.

Our three value layers serve distinct roles. Intrinsic quality determines admission: $I(a) \in \{0, 1\}$ indicates whether a passes the prescribed data-quality checks. Conditional model gain describes its effect on M_t, and the business value layer evaluates whether that effect is useful under application-specific costs. Only recipes with $I(a) = 1$ are eligible for valuation.

3.2 Case-Level Response and Business Utility

Let $\mathcal{Z} = \{(z_i, y_i)\}_{i=1}^{C}$ be a fixed evaluation set with binary labels $y_i \in \{0, 1\}$. Let $q(M, z_i)$ be the signed decision margin, positive for the violation class. The observed response of recipe a on case i is

$$D_t(a, i) = q(M_{t,a}, z_i) - q(M_t, z_i). \tag{2}$$

The vector $D_t(a, :) \in \mathbb{R}^C$ is our prediction target. A shift in the margin does not necessarily repair an error: its direction must be interpreted using y_i, and the hard prediction changes only when the margin crosses zero.

Let $b_{t,i} = \mathbb{I}[q(M_t, z_i) > 0]$ and $h_{t,a,i} = \mathbb{I}[q(M_{t,a}, z_i) > 0]$ be the current and updated predictions. Given case weights $w_{t,i} \geq 0$, set $c(y_i) = c_{\mathrm{FN}}$ for $y_i = 1$ and $c(y_i) = c_{\mathrm{FP}}$ for $y_i = 0$. The realized utility of an update is its reduction in weighted error cost:

$$U_t(a) = \sum_{i=1}^{C} w_{t,i} c(y_i) \left[\mathbb{I}(b_{t,i} \neq y_i) - \mathbb{I}(h_{t,a,i} \neq y_i)\right]. \tag{3}$$

This utility separates repaired false positives and false negatives (FP $\rightarrow$ TN and FN $\rightarrow$ TP) from newly introduced ones (TN $\rightarrow$ FP and TP $\rightarrow$ FN). Absolute business value, reported in the experiments, is a weighted score of correct decisions on its designated evaluation set; $U_t(a)$ instead measures a change relative to M_t on the evaluation set specified for the response study. Their evaluation sets and weights need not coincide.

3.3 Dynamic Recipe-Valuation Task

Suppose a limited set of n recipes has already been updated from M_t; their observed compositions and responses form $\mathcal{H}_t = \{(x_{a_j}, D_t(a_j, :))\}_{j=1}^{n}$. For an untrained, quality-eligible recipe a, the task is to predict $\widehat{D}_t(a, :)$ and the resulting $\widehat{U}_t(a)$ before updating on a. Retaining the current model provides the reference $U_t(\varnothing) = 0$. The recipe with the largest utility is only the best within the candidate pool; an update improves on M_t only if its realized utility is positive. The task is therefore to predict recipe-level behavioral responses and determine whether any candidate justifies changing the current model. Section 4 describes the response predictor and update rule.

4 STELLA-Flow Method

STELLA-Flow predicts case-level responses from recipe composition through a low-dimensional model, then combines these predictions with empirical uncertainty and a business-aware update rule (Figure 2).

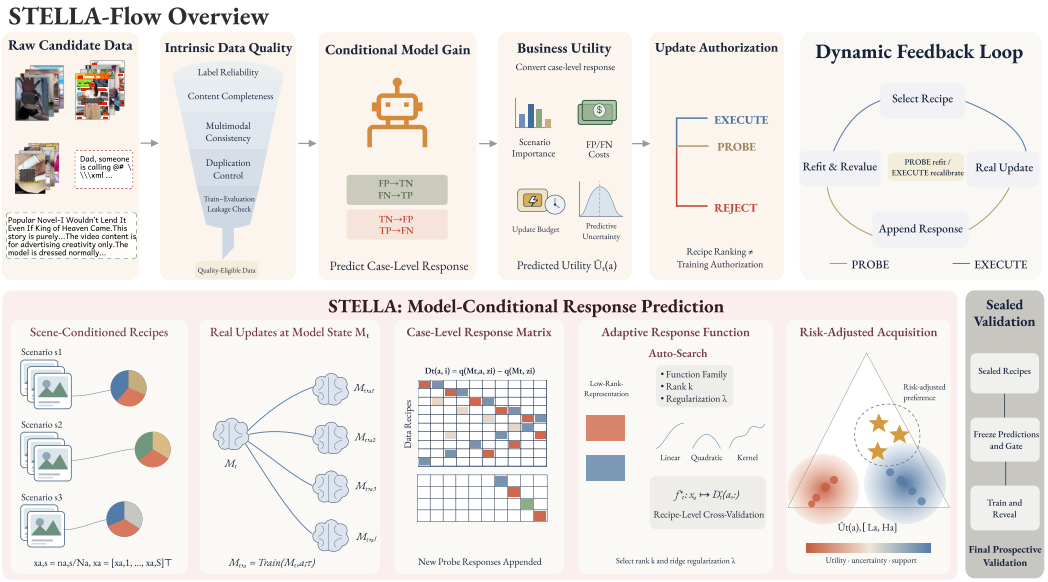

Figure 2: Overview of STELLA-Flow and model-conditional response prediction.

4.1 Quality Admission and Recipe Construction

The intrinsic-quality indicator $I(a)$ from Section 3 is determined before response prediction. Admission checks the reliability of task labels, required content fields, consistency between modalities,

duplicates, and overlap with the evaluation set. Examples failing these checks are excluded before eligible examples are grouped into business scenarios and assembled into recipes with composition vectors x_a. Within a comparison round, candidate recipes have a fixed size and data-format proportions; replay data and τ_t are held fixed. Scenario composition is thus the controlled recipe-level input, while within-scenario sampling remains a source of prediction error.

4.2 Learning Shared Response Modes

Stack compositions and responses from n recipes independently updated from M_t under τ_t as $X \in \mathbb{R}^{n\times S}$ and $D_t \in \mathbb{R}^{n\times C}$ (one recipe per row). Since n is small relative to C, we extract shared response directions by centering and SVD:

$$\mu_t = \frac{1}{n} D_t^\top \mathbf{1}_n, \qquad D_{t,c} = D_t - \mathbf{1}_n \mu_t^\top = U\Sigma V^\top. \tag{4}$$

Retaining the first k components yields

$$D_t \approx \mathbf{1}_n \mu_t^\top + W_t V_{t,k}^\top, \qquad W_t = U_k \Sigma_k, \tag{5}$$

where $V_{t,k} \in \mathbb{R}^{C\times k}$ contains shared response modes and rows of $W_t \in \mathbb{R}^{n\times k}$ contain recipe coefficients. Centering implies $k \le \operatorname{rank}(D_{t,c}) \le n-1$. The modes summarize co-varying case-level changes after complete model updates; they are not assigned a causal interpretation.

4.3 Predicting Responses from Recipe Composition

We fit a multi-output function $f_{t,\theta}$ from recipe composition to the k response coefficients. For an untrained recipe,

$$\widehat{w}_{t,a} = f_{t,\theta}(x_a), \qquad \widehat{D}_t(a,:) = \mu_t^\top + \widehat{w}_{t,a}^\top V_{t,k}^\top. \tag{6}$$

The candidate function families are linear ridge, quadratic ridge on second-order composition features, kernel ridge, and Gaussian-process regression. Each predicts all k coefficients from the same recipe vector; kernel and Gaussian-process hyperparameters are selected within the training folds. The selected response function, latent rank k, and regularization settings are chosen jointly by recipe-level cross-validation.

Specifically, each outer leave-one-recipe-out fold holds out a complete update and recomputes centering, SVD, and coefficient fitting on the other recipes. Inner recipe-level folds choose the function family and its hyperparameters, including k and regularization, by case-level response RMSE; response MAE and utility error serve as secondary diagnostics. The rank search is bounded by the centered training-fold rank. Holding out recipes rather than individual cases prevents the correlated responses from one update from appearing in both training and evaluation. Because x_a encodes only scenario composition, recipes with identical x_a receive the same point prediction; their within-scenario differences contribute to empirical uncertainty.

4.4 Utility Uncertainty and Selection Gate

For an untrained recipe, add its predicted response to the current margin and threshold at zero:

$$\widehat{q}_{t,a,i} = q(M_t, z_i) + \widehat{D}_t(a,i), \qquad \widehat{h}_{t,a,i} = \mathbb{I}[\widehat{q}_{t,a,i} > 0]. \tag{7}$$

Using $\widehat{h}_{t,a,i}$ in Eq. (3) produces $\widehat{U}_t(a)$. Because responses are predicted before applying business costs, they can be revalued under alternative scenario weights and error costs without refitting $f_{t,\theta}$.

Uncertainty is estimated at the recipe level. For each observed recipe a_j, its out-of-fold utility residual is $e_j^U = U_t(a_j) - \widehat{U}_{t,-j}(a_j)$, where the predictor excludes a_j. Let $r_\alpha^U = Q_{1-\alpha}(\{|e_j^U|\}_{j=1}^n)$. We also bootstrap complete recipe rows, refitting the selected function family with its cross-validated settings held fixed. If $Q_p^{\mathrm{boot}}(a)$ is the p-th quantile of the resulting utility predictions, define the empirical interval

$$\begin{aligned} L_t(a) &= \min\{\widehat{U}_t(a) - r_\alpha^U, Q_{\alpha/2}^{\mathrm{boot}}(a) - r_\alpha^U\}, \\ H_t(a) &= \max\{\widehat{U}_t(a) + r_\alpha^U, Q_{1-\alpha/2}^{\mathrm{boot}}(a) + r_\alpha^U\}. \end{aligned} \tag{8}$$

Bootstrapping whole recipes retains the dependence among the case responses from one model update. We assess the empirical intervals on held-out recipes rather than treating them as distribution-free coverage guarantees.

Let $\mathsf{Supp}_t(a) \in \{0, 1\}$ denote a prespecified support condition based on observed scenario frequencies and compositional overlap. Relative to the no-update utility of zero, the decision is

$$\Gamma_t(a) = \begin{cases} \textsc{Reject}, & H_t(a) < 0, \\ \textsc{Execute}, & L_t(a) > 0 \ \wedge \ \mathsf{Supp}_t(a) = 1, \\ \textsc{Probe}, & \text{otherwise.} \end{cases} \tag{9}$$

The gate assigns EXECUTE, REJECT, or PROBE according to Eq. (9). Choosing among multiple PROBE candidates requires a separate allocation rule.

4.5 Feedback Recalibration

Index model states by t and probe rounds within a state by r. If a recipe assigned PROBE is trained and evaluated without replacing M_t, its observed response augments the state-specific history:

$$\mathcal{H}_{t,r+1} = \mathcal{H}_{t,r} \cup \{(x_a, D_t(a, :))\}. \tag{10}$$

The response modes, function family, and hyperparameters are then reselected on the augmented history before the next valuation round. Predictions and decisions recorded before the probe remain frozen for prospective evaluation. A REJECT decision adds no response and leaves the model state unchanged. For EXECUTE, the authorized updated checkpoint becomes

$$M_{t+1} = M_{t,a}. \tag{11}$$

Recipe responses at $t + 1$ are defined relative to M_{t+1}, so earlier response matrices are not pooled into the new state by default.

5 Results and Analysis

We evaluate STELLA-Flow in terms of business value, classification performance, training cost, and the ability to predict model-conditional responses. *Business value* (BV) is the scenario-weighted sum of correct classifications on the fixed 1,000-item evaluation set. Its maximum under the specified weights is 91.084 points. *Response utility* $U_t(a)$ is the reduction in weighted FP/FN error cost achieved by updating the current checkpoint with recipe a, as defined in Eq. (3). Retaining the current checkpoint has utility zero. All F1 scores treat violation as the positive class. Training cost is measured in GiB-minutes. Further details of the models, recipes, update protocols, and metrics are provided in Appendix A.

5.1 Contribution of the Three Value Layers

We first examine the contributions of intrinsic quality (I), model-conditional gain (M), and business value (B) on Qwen3.5-0.8B. Table 1 reports the mean outcome of the completed runs for each configuration.

Table 1: Three-layer ablation on Qwen3.5-0.8B. Values are means over the completed runs for each configuration. Business value is reported in points and training cost in GiB-minutes.

Configuration	BV (points)↑	F1↑	Cost (GiB-min)↓
None	49.4478	0.2218	395.17
I	49.3531	0.3339	404.91
M	48.5503	0.3717	333.11
B	50.4941	0.2563	432.72
I+M	45.3741	**0.6457**	**280.86**
I+B	51.3708	0.3821	375.93
M+B	51.1341	0.3645	368.62
I+M+B	**53.2013**	0.5440	303.59

Table 1 and Figure 3 show distinct trade-offs. I+M achieves the highest F1 (0.6457) and lowest mean training cost (280.86 GiB-minutes), but its BV is 45.3741 points. Adding B raises BV to 53.2013 points while reducing F1 to 0.5440. Relative to None, I+M+B increases mean BV by 3.7535 points and reduces mean training cost by 23.2%. Thus, the configuration with the highest F1 does not maximize BV: F1 measures aggregate classification performance, whereas BV weights correct decisions by business scenario.

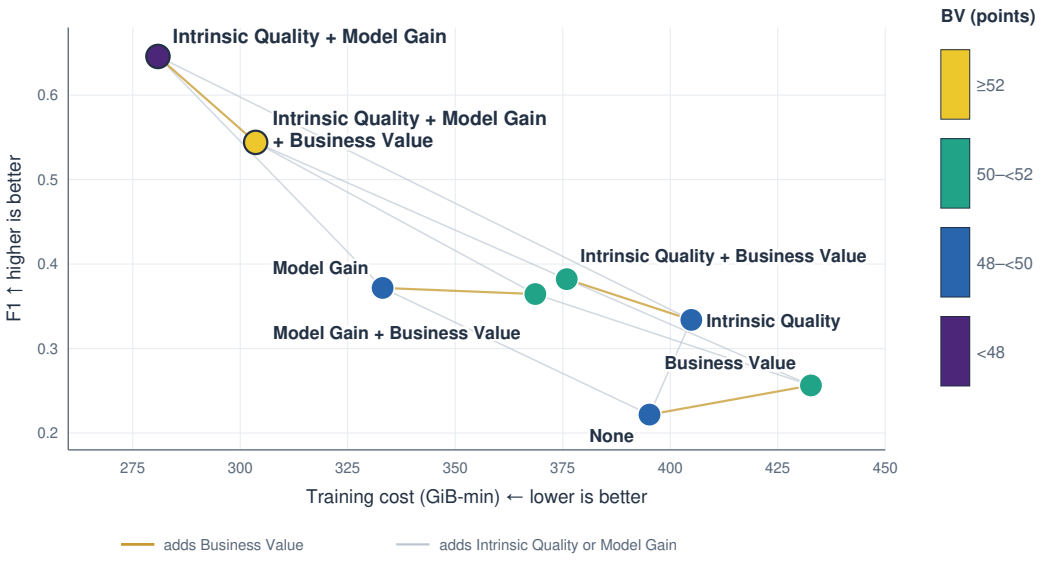

Figure 3: Three-layer ablation on Qwen3.5-0.8B. Each point shows mean training cost (horizontal) and F1 (vertical); color indicates the BV range. Gold lines connect configurations differing in the business-value layer, while gray lines connect those differing in intrinsic quality or model gain.

5.2 Comparison with Data-Selection Baselines

We compare STELLA-Flow with random selection, Human Strategy, MIWV, and NAIT on Qwen3.5-2B. Human Strategy uses a manual recipe based on data-quality criteria and business rules, without a learned response predictor. Table 2 reports the run with the highest realized business value for each method.

Table 2: Data-selection results on Qwen3.5-2B. Each method is represented by its run with the highest business value. Business value is reported in points and training cost in GiB-minutes.

Method	BV (points)↑	F1↑	Cost (GiB-min)↓
Random	50.4725	0.2924	1316.35
Human Strategy	55.3474	0.4282	592.67
NAIT	52.4169	0.1949	**282.43**
MIWV	54.0814	0.4775	308.52
STELLA-Flow	**56.9051**	**0.5920**	917.79

STELLA-Flow achieves the highest business value (56.9051 points) and F1 (0.5920) among the compared methods. Its business value exceeds that of random selection by 6.4326 points, Human Strategy by 1.5577 points, MIWV by 2.8237 points, and NAIT by 4.4882 points. Its measured training cost is lower than that of random selection, but higher than those of Human Strategy, MIWV, and NAIT. These results compare runs selected by BV on the evaluation set, rather than performance on an independent holdout set.

5.3 Behavior Across Model Scales and Architectures

We next examine results across three Qwen model sizes and InternVL3.5-2B. Table 3 compares selected trained checkpoints under random selection, Human Strategy, and STELLA-Flow. Human Strategy uses a selected I+B recipe, whereas STELLA-Flow uses a selected I+M+B recipe. These checkpoint-level results differ in aggregation from the per-configuration means in Table 1; the Qwen3.5-0.8B values in the two tables are therefore reported separately.

Table 3: Results across model scales and architectures. BV and ΔBV are reported in business-value points; training cost is reported in GiB-minutes. Bold indicates the highest business value among the three selection methods for each model.

Model	Random BV	Human BV	STELLA BV	ΔBV (points)	STELLA F1	STELLA cost (GiB-min)
Qwen3.5-0.8B	49.4478	53.8842	**55.4730**	+6.0252	0.6176	300.26
Qwen3.5-2B	50.4725	55.3474	**56.9051**	+6.4326	0.5920	917.79
Qwen3.5-4B	49.7989	50.6653	**54.0140**	+4.2151	0.6722	1484.83
InternVL3.5-2B	48.4044	48.4079	**50.4273**	+2.0229	0.6179	1067.76

The largest business-value gains over random selection occur on Qwen3.5-0.8B and Qwen3.5-2B, at 6.0252 and 6.4326 points, respectively. The selected 0.8B checkpoint attains 97.5% of the business value of the selected 2B checkpoint while incurring 67.3% less measured training cost. The selected 2B checkpoint also exceeds the Qwen3.5-27B reference in business value (56.9051 versus 56.6163 points). On the same benchmark, the selected 2B checkpoint also exceeds a Qwen2.5-VL-32B-Instruct checkpoint updated on randomly sampled data and two in-house business-model checkpoints (Business Models A and B) in BV; full results are in Appendix A.5. These results show that recipe selection can substantially improve the business outcome of compact models on this benchmark.

STELLA-Flow also improves business value over random selection on Qwen3.5-4B, from 49.7989 to 54.0140 points. On InternVL3.5-2B, business value changes from 48.4044 to 50.4273 points.

5.4 The Best Recipe Is Not Necessarily Worth Updating

Before evaluating update authorization (Section 5.5), we first validate the quality of the underlying response predictor. The predictor's accuracy determines whether utility estimates are reliable enough to inform update decisions.We evaluate model-conditional response prediction in a Qwen3.5-0.8B study comprising 44 independently trained recipes and 4,859 aligned evaluation cases. Recipe-level model selection yields a rank-one linear response function with ridge penalty $\lambda = 1$. Table 4 reports prediction errors and observed update outcomes. Response-utility errors and the error radius use the same weighted error-cost scale as response utility.

Table 4: Case-level response prediction and update outcomes in the 44-recipe Qwen3.5-0.8B study. Utility and utility-prediction errors are reported in weighted error-cost points relative to retaining the current checkpoint. Response RMSE is expressed in units of the case-level response variable.

Metric	Value
Case-level response RMSE	0.2624
Utility MAE	154.02
Utility-sign accuracy	100%
90th-percentile utility-error radius	378
Best realized recipe	R041
Best realized utility	-27

The predictor's low error (Table 4) provides the foundation for utility-based gating; Section 5.5 examines whether this translates into correct update decisions. The predictor achieves a case-level response RMSE of 0.2624 and a response-utility MAE of 154.02 weighted error-cost points. Its predicted utility sign matches the realized sign for all 44 recipes. All 44 realized utilities are negative,

however, so sign accuracy alone does not demonstrate the ability to distinguish beneficial from harmful updates.

The decisive observation concerns the no-update reference. R041 has the highest realized response utility in the candidate pool, yet its utility is -27 weighted error-cost points, below the no-update utility of zero. Selecting the best recipe within a pool is therefore distinct from deciding whether to update the model. A ranking-only policy would select a recipe from this pool, whereas a policy with an explicit no-update option can withhold execution when no candidate has supported positive benefit.

5.5 Update Authorization and Recalibration

STELLA-Flow expresses update authorization through three decisions. A candidate receives EXECUTE only when the lower bound of its predicted utility interval is positive and its composition meets the support condition. A candidate receives REJECT when the upper bound is negative. All other candidates receive PROBE. A selected PROBE may then be trained and evaluated to acquire case-level responses without replacing the current checkpoint.

In the 44-recipe study, the 90th-percentile absolute utility-prediction error is 378 weighted error-cost points. The best realized utility, -27, is close to the zero-utility boundary relative to this error scale. Thus, a negative point estimate alone does not justify a confident REJECT decision for every recipe. The observed outcomes support withholding EXECUTE from this candidate pool.

A PROBE contributes a new row of case-level responses to the historical response matrix. The predictor can then be reselected and refitted using recipe-level cross-validation before re-estimating the remaining candidates. After an EXECUTE decision replaces the current checkpoint, the remaining recipes are revalued relative to the new model state. These steps define the dynamic update policy. The 44-recipe experiment establishes the importance of its no-update reference; it does not measure performance over a long sequence of model replacements.

6 Conclusion

We introduced STELLA-Flow to evaluate data-recipe value through intrinsic quality, conditional model gain, and business value. Using a small number of real updates, STELLA predicts case-level behavioral changes, allowing us to assess which existing errors a candidate recipe may correct and which new errors it may introduce before updating on that recipe. The framework then compares its predicted utility with the option of retaining the current model.

On the advertising content moderation benchmark, STELLA-Flow achieves higher business value than random selection across the evaluated recipe-selection settings. The updated 2B model also exceeds Qwen3.5-27B in both business value (56.9051 versus 56.6163) and F1 (0.5920 versus 0.4808), while attaining 98.7% of the business value of Qwen3.5-397B-A17B. In the 44-recipe study, even the highest-ranked recipe has negative utility relative to retaining the current model, separating recipe ranking from update authorization.

Together, these results show how model-conditional data selection can direct limited update budgets toward business-relevant improvements. Repeated recipe valuation and feedback from real updates offer a way to pursue further gains as the model evolves. Future work will evaluate this trajectory over longer sequences of model updates and refine uncertainty estimates as model behavior and scenario priorities change.

AI use statement

Generative AI tools assisted with drafting, translating, and editing portions of this manuscript; providing feedback on experimental methodology and result interpretation; checking the internal consistency of reported numbers; and preparing and refining the figures in this paper. The authors reviewed the AI-assisted material and take responsibility for the final content.

REFERENCES

M. Xia, S. Malladi, S. Gururangan, S. Arora, and D. Chen. LESS: Selecting Influential Data for Targeted Instruction Tuning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

Y. Kwon, E. Wu, K. Wu, and J. Zou. DataInf: Efficiently Estimating Data Influence in LoRA-Tuned LLMs and Diffusion Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

L. Liu, X. Liu, D. F. Wong, D. Li, Z. Wang, B. Hu, and M. Zhang. SelectIT: Selective Instruction Tuning for LLMs via Uncertainty-Aware Self-Reflection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

J. Li et al. DataComp-LM: In Search of the Next Generation of Training Sets for Language Models. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

G. Penedo, H. Kydlíček, L. Ben Allal, A. Lozhkov, M. Mitchell, C. Raffel, L. von Werra, and T. Wolf. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

R. Peng, K. Yang, Y. Zeng, J. Lin, D. Liu, and J. Zhao. DataMan: Data Manager for Pre-training Large Language Models. *arXiv preprint arXiv:2502.19363*, 2025.

A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. Datamodels: Predicting Predictions from Training Data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

L. Engstrom, A. Feldmann, and A. Madry. DsDm: Model-Aware Dataset Selection with Datamodels. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Z. Yu, S. Das, and C. Xiong. MATES: Model-Aware Data Selection for Efficient Pretraining with Data Influence Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

M. Nikdan, V. Cohen-Addad, D. Alistarh, and V. Mirrokni. Efficient Data Selection at Scale via Influence Distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Q. Dai, D. Zhang, J. W. Ma, and H. Peng. Improving Influence-Based Instruction Tuning Data Selection for Balanced Learning of Diverse Capabilities. In *Findings of the Association for Computational Linguistics: EMNLP*, 2025.

J. Ye, P. Liu, T. Sun, Y. Zhou, J. Zhan, and X. Qiu. Data Mixing Laws: Optimizing Data Mixtures by Predicting Language Modeling Performance. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Q. Liu, X. Zheng, N. Muennighoff, G. Zeng, L. Dou, T. Pang, J. Jiang, and M. Lin. RegMix: Data Mixture as Regression for Language Model Pre-training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

M. F. Chen, M. Y. Hu, N. Lourie, K. Cho, and C. Ré. Aioli: A Unified Optimization Framework for Language Model Data Mixing. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

M. Shukor, L. Bethune, D. Busbridge, D. Grangier, E. Fini, A. El-Nouby, and P. Ablin. Scaling Laws for Optimal Data Mixtures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Y. Wang, B. Liu, F. Liu, Y. Guo, J. Deng, X. Wu, W. Zhou, X. Zhou, and T. Wang. TiKMiX: Take Data Influence into Dynamic Mixture for Language Model Pre-training. *arXiv preprint arXiv:2508.17677*, 2025.

H. Tan, S. Wu, Y. Chen, J. Xia, R. Xie, B. Xia, X. Sun, and X. Qi. FastMix: Fast Data Mixture Optimization via Gradient Descent. *arXiv preprint arXiv:2606.14971*, 2026.

A. Wettig, A. Gupta, S. Malik, and D. Chen. QuRating: Selecting High-Quality Data for Training Language Models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

Z. Ankner, C. Blakeney, K. Sreenivasan, M. Marion, M. L. Leavitt, and M. Paul. Perplexed by Perplexity: Perplexity-Based Data Pruning with Small Reference Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

W. Liu, W. Zeng, K. He, Y. Jiang, and J. He. What Makes Good Data for Alignment? A Comprehensive Study of Automatic Data Selection in Instruction Tuning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

A. Bukharin, S. Li, Z. Wang, J. Yang, B. Yin, X. Li, C. Zhang, T. Zhao, and H. Jiang. Data Diversity Matters for Robust Instruction Tuning. In *Findings of the Association for Computational Linguistics: EMNLP*, 2024.

K. Zhao, Z. Miao, A. Aizawa, and Y. Tsuruoka. RegMix-D: Dynamic Data Mixing via Proxy Training Trajectories. *arXiv preprint arXiv:2606.18663*, 2026.

B. Wen, S. Salekin, F. Kang, B. Howe, L. L. Wang, J. Movellan, and M. Bilkhu. MixAtlas: Uncertainty-Aware Data Mixture Optimization for Multimodal LLM Midtraining. *arXiv preprint arXiv:2604.14198*, 2026.

H. Mozannar and D. Sontag. Consistent Estimators for Learning to Defer to an Expert. In *Proceedings of ICML*, 2020.

H. Narasimhan, W. Jitkrittum, A. K. Menon, A. Rawat, and S. Kumar. Post-hoc Estimators for Learning to Defer to an Expert. In *Advances in Neural Information Processing Systems*, 2022.

A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, and T. Schuster. Conformal Risk Control. In *Proceedings of ICLR*, 2024.

C. Yeh, N. Christianson, A. Wierman, and Y. Yue. Conformal Risk Training: End-to-End Optimization of Conformal Risk Control. In *Advances in Neural Information Processing Systems*, 2025.

Y. Xie, Y.-A. Lai, Y. Xiong, Y. Zhang, and S. Soatto. Regression Bugs Are In Your Model! Measuring, Reducing and Analyzing Regressions in NLP Model Updates. In *Proceedings of ACL-IJCNLP*, 2021.

A Experimental Setup and Evaluation Details

A.1 Models and Data Recipes

We evaluate Qwen3.5-0.8B, Qwen3.5-2B, Qwen3.5-4B, and InternVL3.5-2B. The MIWV and NAIT baselines are reproduced on Qwen3.5-2B. Methods compared on the same model use the same training and evaluation protocols; they differ in how the training data are selected.

Candidate data originate from 41 business sources and are curated into a pool of approximately 30,000 examples spanning five business scenarios. Each training recipe contains 1,000 examples. Within a comparison round, recipe size and question-type proportions are held fixed. Video examples retain at most eight frames. Model updates use supervised fine-tuning without replay data.

The intrinsic-quality (I), model-conditional-gain (M), and business-value (B) ablation comprises eight configurations: None, I, M, B, I+M, I+B, M+B, and I+M+B. Each configuration is evaluated in five runs; Table 1 reports the mean over these runs.

A.2 Model-Update Protocol

Each recipe is trained independently from the corresponding model's starting checkpoint. The four models use the same training hyperparameters: one H20 GPU (141 GB), bf16 arithmetic, full-parameter fine-tuning of the language model, and frozen vision tower and alignment module. The micro-batch size is 1 and the global batch size is 64. Training runs for four epochs with sequence packing and a maximum packed length of 12,400 tokens. Depending on the total token count, packing produces approximately 6–10 optimizer steps per task; a checkpoint is saved every two steps. The learning rate is 10^{-5} with cosine decay, 5% warmup, and weight decay 0.1. Experiment lists and fixed-seed data manifests are versioned with hashes.

A.3 Evaluation Set and Metrics

Tables 1–3 use the same frozen 1,000-item binary-classification evaluation set, containing 468 violation and 532 non-violation items. It covers business cases both within and outside the training scenarios and has no photo-ID overlap with the training data. All models use the same inference and evaluation protocols. Violation is the positive class for F1, which is computed from item-level predictions and ground-truth labels.

Business value (BV) is the scenario-weighted sum of correct decisions: a correctly classified item contributes its business-value weight. The weights are fixed across all evaluated models and selection methods. The maximum BV on the evaluation set is 91.084 points.

Training cost is reported in GiB-minutes. For a single-GPU task, it is computed as

$$\text{Cost} = \frac{m_{\text{peak}}\, N_{\text{steps}}\, t_{\text{step}}}{60}, \tag{12}$$

where m_{peak} is peak GPU memory in GiB, N_{steps} is the number of optimizer steps, and t_{step} is time per step in seconds.

A.4 Run Selection and Reporting

Table 1 reports the mean of five runs for each ablation configuration. Table 2 reports the run with the highest realized BV for each selection method on the same 1,000-item evaluation set. STELLA-Flow, MIWV, and NAIT each have five runs. Human Strategy uses a fixed data recipe established through business iteration; its reported checkpoint is the run with the highest BV for that recipe. Table 3 reports selected checkpoints, with an I+B recipe for Human Strategy and an I+M+B recipe for STELLA-Flow. BV, F1, and cost for a selected checkpoint refer to that same run.

A.5 Additional Reference-Model Comparisons

All models in Table 5 were evaluated on the same frozen 1,000-item set using the same BV and F1 definitions. The Qwen2.5-VL-32B-Instruct checkpoint was updated on randomly sampled training data. The STELLA-Flow-selected Qwen3.5-2B checkpoint outperforms GLM-5.3 and DeepSeek-V4-Pro in both BV and F1. Business Models A and B are in-house checkpoints selected through the business team's established data-selection and training process; their names are anonymized for review.

Business Models A and B were trained on substantially larger datasets (approximately 1.6 million examples) accumulated through the team's production data pipeline. In contrast, STELLA-Flow recipes contain 1,000 examples selected from a 30,000-example candidate pool. Despite using 0.0625% of the training data (1,000 versus 1.6 million examples), the STELLA-Flow-selected Qwen3.5-2B checkpoint achieves higher business value than both Business Model A (56.9051 versus 54.7216 points, +4.1%) and Business Model B (56.9051 versus 53.3874 points, +5.7%), while also attaining substantially higher F1 scores (0.5920 versus 0.4845 and 0.5097). This demonstrates that model-conditional data selection can outperform large-scale data accumulation with a small fraction of the training examples.

Table 5: Reference-model and in-house business-model results on the shared evaluation set.

Model/checkpoint	BV (points)↑	F1↑
Qwen3.5-397B-A17B	57.6710	0.5019
Qwen3.5-122B-A10B	57.4587	0.4543
Qwen3.5-2B (STELLA-Flow)	56.9051	0.5920
DeepSeek-V4-Pro	56.7409	0.5836
Qwen3.5-27B	56.6163	0.4808
GLM-5.3	55.8066	0.5568
Business Model A	54.7216	0.4845
Qwen2.5-VL-32B-Instruct	53.5415	0.3972
Business Model B	53.3874	0.5097
Qwen3-VL-8B-Instruct	53.1697	0.3065
Qwen3-VL-30B-A3B-Instruct	51.1317	0.5022

A.6 Recipe-Response Study

Table 4 describes a separate model-response study. Starting from the same Qwen3.5-0.8B checkpoint, 44 recipes are trained independently, and their effects are assessed on 4,859 aligned cases. Response utility $U_t(a)$ is the reduction in weighted false-positive and false-negative error cost relative to retaining the current checkpoint; the no-update utility is zero. This utility is distinct from the BV score on the 1,000-item benchmark. Recipe-level model selection yields a rank-one linear response predictor with ridge penalty $\lambda = 1$.

A.7 Response-Utility Configuration for the 44-Recipe Study

In the 44-recipe study, all $C = 4{,}859$ aligned evaluation cases have equal weight, and false positives and false negatives have equal cost:

$$w_{t,i} = 1 \quad (i = 1, \ldots, C), \qquad c_{\mathrm{FP}} = c_{\mathrm{FN}} = 1. \tag{13}$$

Thus, Eq. (3) reduces to

$$U_t(a) = \sum_{i=1}^{C} \left[\mathbb{I}(b_{t,i} \neq y_i) - \mathbb{I}(h_{t,a,i} \neq y_i) \right]. \tag{14}$$

Each repaired error contributes $+1$, and each newly introduced error contributes -1. Therefore, $U_t(a)$ is the net reduction in classification errors relative to the current checkpoint. R041's realized utility of -27 means a net increase of 27 errors. These response-utility weights are separate from the scenario weights used to compute BV on the 1,000-item benchmark.