BiTFlow: Efficient Coreset Selection for Instruction Tuning via Balanced Token Flow Matching

Anonymous authors
Paper under double-blind review

Abstract

Coreset selection for instruction tuning aims to reduce fine-tuning computation while preserving downstream performance. Forward-only selection methods avoid costly model updates and backward passes by comparing hidden-state fingerprints, most often from the last layer. However, samples with highly similar last-layer states can induce different intermediate token updates, a phenomenon we call *last-layer similarity collapse*, indicating that intermediate layers carry information beyond the last-layer state. We capture this information with residual flows, the residual updates of each token between selected layers. Building on these flows, we introduce BiTFlow, a forward-only coreset selection algorithm that compares candidate samples with a small set of target-task samples at the token level. BiTFlow summarizes target residual flows into token-level prototypes and aligns candidate tokens to them via one-to-one bipartite matching. A balanced aggregation then scores each sample by both how well its tokens match and how many distinct target token IDs are covered. At a 5% selection budget on the LESS (0.27M samples) and SmolTalk (1.04M samples) pools, BiTFlow achieves the highest average performance and exceeds full-data fine-tuning on multiple benchmarks, and its selections transfer across model scales and families. Scoring requires no backward passes or warmup training and is $\sim$18$\times$ faster than gradient-based methods.

1 Introduction

Instruction-tuning data for LLMs contains samples that vary in quality, relevance, and redundancy (Albalak et al., 2024). Training on the full candidate pool therefore spends computation on samples that may provide little useful supervision for the downstream tasks. Coreset selection addresses this problem by choosing a small training subset (coreset) that can preserve or improve downstream performance at lower fine-tuning cost than training on the full candidate pool (Xia et al., 2024; Ivison et al., 2025), with a small set of target-task samples (target set) guiding its selection.

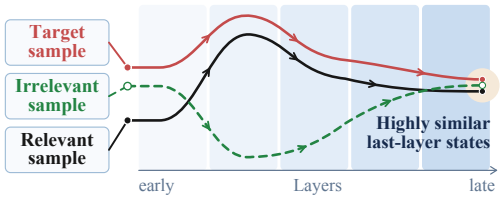

Figure 1: Illustration of last-layer similarity collapse. The target sample (red), relevant sample (black), and irrelevant sample (green, dashed) have highly similar last-layer states. The relevant sample follows similar intermediate updates to the target, while the irrelevant sample follows a different pattern. Statistical evidence of intermediate-update differences is in Fig. 3.

Existing methods for instruction coreset selection from large, heterogeneous candidate pools fall into three groups: training-signal selectors, surface-statistics selectors, and forward-only selectors. Training-signal selectors use per-sample gradients after warmup or loss trajectories from proxy models (Xia et al., 2024; Zhang et al., 2025a; Yang et al., 2024; Nagaraj et al., 2025); obtaining these signals requires model training or backward passes. Surface-statistics selectors use lexical or distributional statistics (Lù, 2024; Xie et al., 2023), without accounting for the distribution learned by the language model. Existing forward-only selectors compare fingerprints extracted from model hidden states and scale to large candidate pools using forward passes (Ivison et al., 2025; Ma et al., 2025; Shu et al., 2026; Nagaraj et al., 2026).

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.

In this work, we focus on forward-only methods. Representative methods, including RDS+ and TRIM, construct their fingerprints from last-layer hidden states (Ivison et al., 2025; Nagaraj et al., 2026). These fingerprints describe the resulting states but do not record the intermediate updates that produce them. Fig. 1 highlights that the target, relevant, and irrelevant samples have highly similar last-layer states, yet their intermediate updates differ. The relevant sample follows a similar update pattern to the target, while the irrelevant sample follows a different pattern. We further examine this pattern across three tasks using controlled sample pairs that share the same response but have slightly different prompts. Among pairs whose last-layer states have cosine similarity between 0.95 and 0.999, intermediate updates differ 2.1–3.6 times as much as last-layer states (median per task; Fig. 3). We call this phenomenon *last-layer similarity collapse*.

This observation raises the question of whether information from intermediate layers can help coreset selection. Multi-layer hidden states offer one way to capture it (Shu et al., 2026). Drawing inspiration from representation dynamics across Transformer layers (Fernando & Guitchounts, 2025; Mabrok, 2026), we capture it with the sequence of layer-wise residual updates, which we call a *residual flow*. A residual flow keeps how each token changes from one layer to the next, including the update direction, which the last-layer state alone does not retain.

Building on residual flows, we introduce BiTFlow (**B**alanced **i**ntermediate **T**oken **Flow**), a highly efficient, forward-only coreset selection algorithm that compares candidate samples with a small target set at the token level. BiTFlow has two stages: target prototype construction and candidate scoring and selection (Fig. 2). Each target token prototype summarizes the update directions of target tokens across selected layers, compared as PCA-projected unit directions. Token-level comparison raises two problems that the choice of flow encoding does not address. First, if many candidate tokens can match the same target prototype, a sample that repeats a few matching tokens receives an inflated score; BiTFlow therefore aligns candidate and target token IDs via one-to-one bipartite matching. Second, a few strongly matched tokens can dominate a sample's score even when most of the target is uncovered; BiTFlow therefore scores each sample by a balanced aggregation, the harmonic mean of candidate precision and target recall, which is high only when the sample's tokens match well and cover many distinct target token IDs. We rank the scores in the candidate pool to form the coreset.

Extensive experiments demonstrate the efficacy, efficiency, and generalizability of BiTFlow's design. At a 5% selection budget on two different-size candidate pools: the LESS pool (0.27M) and the SmolTalk pool (1.04M), BiTFlow outperforms state-of-the-art methods (Xia et al., 2024; Nagaraj et al., 2026; Shu et al., 2026) on average and exceeds full-data fine-tuning on several tasks. For the instruction-following task on the LESS pool, it achieves up to 23.7% improvement compared with the state-of-the-art method (Chen et al., 2026). BiTFlow requires no scorer warmup or backward passes and scores candidates with minimal overhead relative to other forward-only selection algorithms, $\sim$18$\times$ faster than the gradient-based LESS method (Xia et al., 2024). Selections made on one scorer model also transfer to other model scales and families. Thus, our contributions are as follows:

(i) We illustrate *last-layer similarity collapse*: samples can have highly similar last-layer states despite different intermediate token updates.

(ii) We propose BiTFlow, a highly efficient forward-only coreset selection algorithm that captures intermediate updates with residual flows and scores candidates by one-to-one bipartite matching with balanced aggregation, which keeps repeated matched tokens from dominating sample scores.

(iii) BiTFlow achieves the highest average performance on two different-size candidate pools. It scores candidates $\sim$18$\times$ faster than gradient-based methods, and its selections transfer effectively across model scales and families.

2 Related Work

Instruction Coreset Selection for Large Language Models. Instruction coreset selection methods for large, heterogeneous candidate pools differ in the signals they use. *(i)* Training-signal selectors use gradients or loss trajectories: LESS and TAGCOS construct gradient features (Xia et al., 2024; Zhang et al., 2025a), while S2L and CLD use loss trajectories measured during proxy training (Yang et al., 2024; Nagaraj et al., 2025). Obtaining these signals requires warmup training, backward passes, or proxy training before selection, and the resulting compute can offset the savings from fine-tuning on fewer samples. *(ii)* Surface-statistics selectors use lexical retrieval or target-to-pool density ratios (Lù, 2024; Xie et al., 2023). These methods are inexpensive but score the text itself without accessing the model's internal computation. *(iii)* Forward-only selectors score candidates

using model activations without backward passes. RDS+ aggregates last-layer hidden states into sample fingerprints (Ivison et al., 2025), while TRIM uses attention saliency across layers to construct token fingerprints from the last-layer states of a scoring model after warmup training (Nagaraj et al., 2026). CRDS reduces fingerprint redundancy through whitening or compression of multi-layer states (Shu et al., 2026). MONA uses sparse autoencoders to construct sample fingerprints from neuronal activations and compares them with an aggregated target-task fingerprint (Ma et al., 2025). NAIT extracts target directions from within-layer activation differences, then scores candidates by projecting their activations onto these directions (Chen et al., 2026). CRDS and NAIT already use hidden states from multiple layers, indicating that information beyond the last layer can support selection. The methods also differ in the granularity of comparison. RDS+, CRDS, and MONA summarize each sample by a single fingerprint, so their similarity reflects the sample as a whole and does not show which parts of the target a candidate matches. TRIM compares individual tokens, but its sample score uses only candidate-side token statistics and thus does not measure how much of the target a candidate covers. BiTFlow captures cross-layer information with residual flows and compares samples at the token level: one-to-one bipartite matching pairs candidate and target token IDs without reuse, and balanced aggregation scores a candidate by both how well its tokens match and how many distinct target token IDs it covers.

Representation Dynamics in Large Language Models. A growing line of research models the residual stream as token trajectories across layers (Lu et al., 2019; Lawson et al., 2025; Fernando & Guitchounts, 2025; Tong et al., 2025; Ghilardi et al., 2025). Studies of intermediate hidden-state dynamics have examined reasoning correctness (Wang et al., 2025; Bi et al., 2025; Damirchi et al., 2026), hallucination detection (Zhang et al., 2025b), deception detection (Mothukuri & Parizi, 2026), and uncertainty estimation (Eusebi et al., 2026). SHIFT uses a start-to-end hidden-state displacement for RLVR data selection (Wu et al., 2026). These studies suggest that how hidden states change across layers carries information beyond the last layer. BiTFlow applies this observation to instruction coreset selection by comparing the residual flows of candidate and target samples at corresponding selected layers.

3 METHODOLOGY

3.1 PRELIMINARIES

Problem formulation. Given a candidate pool $\mathcal{D}_{\text{pool}} = \{x_i\}_{i=1}^{N}$, a compact target set $\mathcal{D}_{\text{tar}}$ containing samples from target tasks, and a selection budget K, our goal is to select a subset $\mathcal{D}_{\text{sel}}^{\star} \subset \mathcal{D}_{\text{pool}}$ of K samples to maximize instruction-tuning performance on the target distribution.

Last-layer similarity collapse. Several coreset selectors score candidates by comparing their last-layer hidden states with those of target samples (Ivison et al., 2025; Nagaraj et al., 2026; Shu et al., 2026). However, two samples that a model processes differently across layers can arrive at near-identical last-layer states. We show below that the last-layer state difference cannot reveal the intermediate update variation.

Let $H_t^{\ell} \in \mathbb{R}^{d_h}$ be the residual-stream state of token t after block ℓ. For selected layers $\ell_0 < \ell_1 < \cdots < \ell_m$, the last-layer state decomposes as:

$$H_t^{\ell_m} = H_t^{\ell_0} + \sum_{k=1}^{m} \Delta h_{t,\ell_k}, \quad \text{where} \quad \Delta h_{t,\ell_k} = H_t^{\ell_k} - H_t^{\ell_{k-1}}. \tag{1}$$

Proposition 1 (Decomposition of Layer-Wise Update Differences)**.** *For two token instances t and t', let $u_k = \Delta h_{t,\ell_k} - \Delta h_{t',\ell_k}$ be their update difference at layer ℓ_k, with mean $\bar{u} = m^{-1} \sum_{k=1}^{m} u_k$. Let $d_{\text{start}} = H_t^{\ell_0} - H_{t'}^{\ell_0}$ and $d_{\text{last}} = H_t^{\ell_m} - H_{t'}^{\ell_m}$. Then:*

$$d_{\text{last}} = d_{\text{start}} + m\bar{u}, \tag{2}$$

$$\frac{1}{m} \sum_{k=1}^{m} \|u_k\|_2^2 = \frac{\|d_{\text{last}} - d_{\text{start}}\|_2^2}{m^2} + \frac{1}{m} \sum_{k=1}^{m} \|u_k - \bar{u}\|_2^2. \tag{3}$$

Thus, for a fixed initial d_{start}, last-layer comparisons capture only the mean update difference $\bar{u}$. They do not determine the variation of update differences across layers (the second term in Eq. 3), which can be nonzero for $m > 1$ even when the last-layer states coincide ($d_{\text{last}} = 0$). The complete proof is in Appendix B.

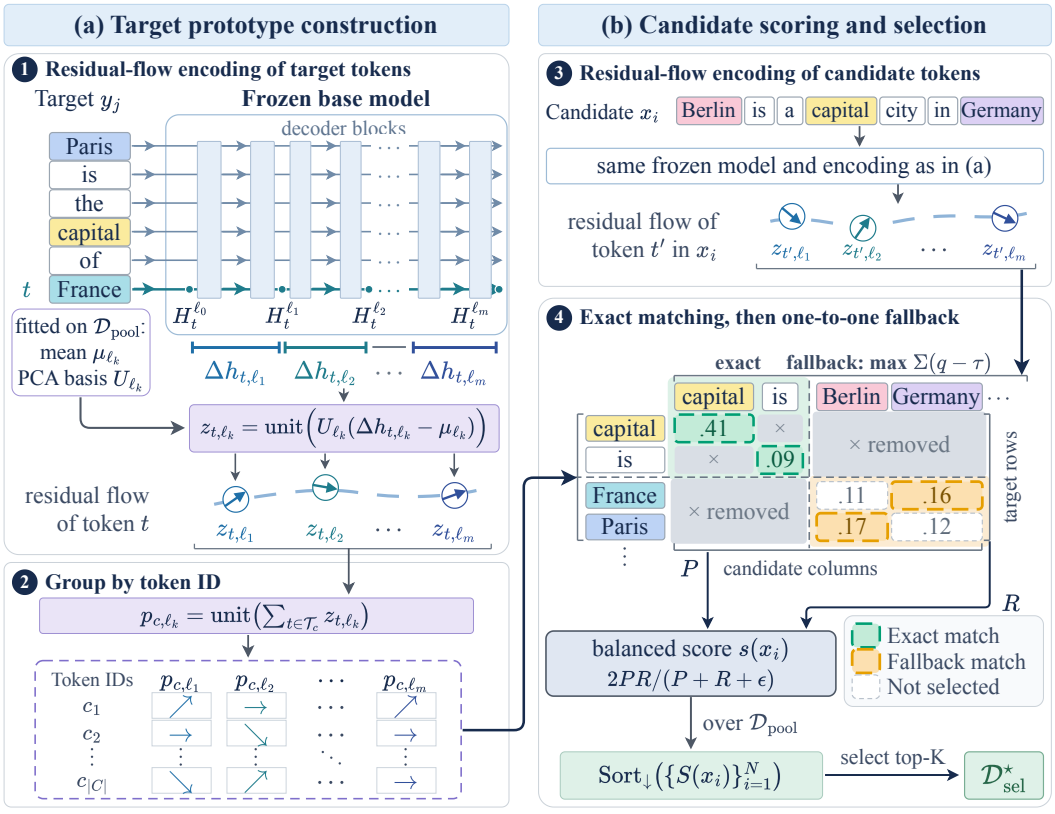

Figure 2: Overview of BiTFlow. **(a) Target prototype construction.** A frozen base model tracks how target tokens are updated across layers. Their residual-flow encodings are grouped by token ID to construct target prototypes, preserving update directions at corresponding layers. The means μ_{ℓ_k} and PCA projections U_{ℓ_k} are fitted using residual updates sampled from the candidate pool. **(b) Candidate scoring and selection.** Candidate tokens are encoded in the same way and compared with the target prototypes. Exact matches are resolved first; the remaining token IDs are paired by one-to-one bipartite matching. In the matrix, columns represent candidate token IDs and rows represent target token IDs. Green cells indicate exact matches, orange cells indicate selected fallback matches, and white dashed cells indicate unselected fallback pairs. The score balances how much of the candidate is supported by the target (candidate precision P) against how much of the target it covers (target recall R). The scores over the candidate pool are used to rank and select the top-K samples.

Residual flow. To retain the layer-wise structure that last-layer comparison discards, we define the *residual flow* of token t as the ordered sequence of its residual updates across selected layers:

$$\mathcal{R}(t) = (\Delta h_{t,\ell_1}, \ldots, \Delta h_{t,\ell_m}). \tag{4}$$

When consecutive selected layers are adjacent, each $\Delta h_{t,\ell_k}$ corresponds to the update of a single decoder block. Hidden states collected from multiple layers also carry cross-layer information; residual flow instead expresses it as the changes between selected layers, so that each coordinate describes what a range of layers adds to the token state. We compare both choices in Sec. 4.5.

As illustrated in Fig. 2, BiTFlow operates in two stages: target prototype construction summarizes projected residual-update directions by token ID across selected layers (Sec. 3.2); candidate scoring and selection uses bipartite token-ID matching and balances candidate precision against target coverage to rank samples for the coreset (Sec. 3.3).

3.2 Constructing Residual-Flow Prototypes

Residual-flow encoding. Comparing raw residual updates in the full hidden space incurs substantial computational overhead. We instead project the updates onto a low-rank subspace derived from the candidate pool and compare their directions in that subspace.

Specifically, for each layer ℓ_k, we compute the empirical mean update μ_{ℓ_k} and fit a PCA projection matrix U_{ℓ_k} using token instances sampled from $\mathcal{D}_{\text{pool}}$. Both candidate and target updates are then

centered, projected, and normalized onto the unit sphere:

$$z_{t,\ell_k} = \mathrm{unit}\big(U_{\ell_k}(\Delta h_{t,\ell_k} - \mu_{\ell_k})\big), \tag{5}$$

where $\mathrm{unit}(v) = v/\|v\|_2$ for nonzero vectors.

Target token prototypes. Since residual updates of the same token ID naturally fluctuate across different contexts, comparing individual instances introduces high variance and computational cost. To construct a robust consensus update direction for each target token ID $c \in C$, we aggregate its instance set $\mathcal{T}_c$ across the target set into a single unit prototype p_{c,ℓ_k} at layer ℓ_k. Letting $v = \sum_{t\in\mathcal{T}_c} z_{t,\ell_k}$, we maximize the mean directional alignment $\frac{1}{|\mathcal{T}_c|}\sum_{t\in\mathcal{T}_c}\langle p, z_{t,\ell_k}\rangle = \frac{\langle p,v\rangle}{|\mathcal{T}_c|}$. By the Cauchy–Schwarz inequality ($\langle p, v\rangle \le \|v\|_2$), this maximum is uniquely attained at $p = v/\|v\|_2$ for $v \neq \mathbf{0}$. We thus define the prototype as the normalized sum:

$$p_{c,\ell_k} = \mathrm{unit}\left(\sum_{t\in\mathcal{T}_c} z_{t,\ell_k}\right), \quad \text{with } p_{c,\ell_k} = \mathbf{0} \text{ if } v = \mathbf{0}. \tag{6}$$

The sequence $(p_{c,\ell_1}, \ldots, p_{c,\ell_m})$ forms the multi-layer prototype for token c, capturing its dominant directional dynamics across the target distribution.

3.3 Balanced Candidate Scoring and Selection

We score candidate samples separately for the prompt and response roles via one-to-one bipartite token-ID matching followed by balanced evidence aggregation.

Matching token IDs. To prevent candidate tokens from repeatedly matching against a single target prototype, we employ a one-to-one bipartite matching formulation. Candidate instances t' are first encoded via Eq. 5. For a specific role, let $D(x)$ be the set of unique token IDs in candidate x, and let $A_d(x)$ collect its instances with ID d. We define the alignment between candidate instance t' and target prototype c as their mean cosine similarity across selected layers:

$$r(t', c) = \frac{1}{m}\sum_{k=1}^{m}\langle z_{t',\ell_k}, p_{c,\ell_k}\rangle, \qquad r_c(x) = \max_{t'\in A_c(x)} r(t', c). \tag{7}$$

For shared token IDs $c \in D(x) \cap C$, exact matching pairs them directly and contributes the positive alignment $[r_c(x)]_+$, where $[v]_+ = \max(v, 0)$. All exact-matched IDs are excluded from fallback matching, even when their contribution is zero. They remain in $D(x)$ and C for score normalization.

We precompute a shortlist $\mathcal{N}_J(d)$ for each vocabulary token ID d. It contains up to J IDs from the full target token-ID set C with the highest positive input-embedding cosine similarities. For each candidate, we remove exact-matched IDs from these shortlists without adding replacement neighbors. Each remaining pair (d, c), with $d \in D(x) \setminus C$, receives the edge quality

$$q(d, c) = \gamma\, s_{\mathrm{emb}}(d, c)\left[\max_{t'\in A_d(x)} r(t', c)\right]_+, \tag{8}$$

where $s_{\mathrm{emb}}(d, c)$ is the input-embedding cosine similarity and $\gamma \in (0, 1]$ discounts cross-token transfer. The matching cost $\tau \ge 0$ is subtracted from each edge quality in the matching objective. We retain only edges with positive net gain, $q(d, c) > \tau$.

Let $\mathcal{M}$ denote a candidate matching subset. We solve for the optimal bipartite assignment $\mathcal{M}^\star$ that maximizes net similarity gain:

$$\mathcal{M}^\star \in \arg\max_{\mathcal{M}} \sum_{(d,c)\in\mathcal{M}} \big(q(d, c) - \tau\big) \quad \text{s.t.} \quad \deg_{\mathcal{M}}(d) \le 1,\ \deg_{\mathcal{M}}(c) \le 1. \tag{9}$$

The degree constraints enforce that each unmatched token ID participates in at most one fallback pair.

Aggregating matched evidence. High similarity across a handful of tokens does not guarantee whole-sequence relevance. To prevent narrow matches from dominating the final scores, we balance candidate support against target coverage. Total matched evidence aggregates exact and fallback alignments:

$$E(x) = \sum_{c\in D(x)\cap C} [r_c(x)]_+ + \sum_{(d,c)\in\mathcal{M}^\star} q(d, c). \tag{10}$$

We define candidate precision $P(x)$ and target recall $R(x)$ by normalizing $E(x)$ against their respective vocabulary sizes: $P(x) = E(x)/|D(x)|,\ R(x) = E(x)/|C|$. Unmatched tokens contribute zero evidence but properly penalize precision and recall via denominators $|D(x)|$ and $|C|$.
We combine $P(x)$ and $R(x)$ via their harmonic mean (stabilized by $\epsilon > 0$):

$$s(x) = \frac{2P(x)R(x)}{P(x) + R(x) + \epsilon}. \tag{11}$$

Appendix B.3 gives the range of $s(x)$ and its closed form as a Dice-style overlap of the two token-ID sets. With $s_{\text{prompt}}(x)$ and $s_{\text{response}}(x)$ denoting the role-specific scores, the final sample selection score is parameterized by $\alpha \in [0, 1]$:

$$S(x; \mathcal{D}_{\text{tar}}) = \alpha\, s_{\text{prompt}}(x) + (1 - \alpha)\, s_{\text{response}}(x). \tag{12}$$

Finally, candidate samples in $\mathcal{D}_{\text{pool}}$ are ranked in descending order of $S(x; \mathcal{D}_{\text{tar}})$, and the top-$K$ samples are chosen to construct the coreset $\mathcal{D}^{\star}_{\text{sel}}$.

4 Experiments

We evaluate BiTFlow with four questions: **RQ1:** How does downstream performance compare across subsets selected by BiTFlow and existing methods? (Sec. 4.1) **RQ2:** Can residual flows capture information that last-layer states fail to distinguish? (Sec. 4.2) **RQ3:** How do selected subsets transfer across model scales and families? (Sec. 4.4) **RQ4:** How do scoring components and hyperparameters affect downstream performance, and what is the computational cost? (Sec. 4.5 and Sec. 4.6)

Experimental setup. We evaluate instruction coreset selection on two candidate pools: the LESS pool (0.27M samples) (Xia et al., 2024) and the SmolTalk pool (1.04M samples) (Ben Allal et al., 2025), using a default selection budget of 5%. By default, BiTFlow scores candidates with a frozen, pretrained Llama-2-7B (Touvron et al., 2023) without warmup training. We group consecutive block updates into five bands bounded by layers $(8, 16, 20, 27, 30, 32)$. For each band, we fit a 128-dimensional PCA projection using residual updates sampled from the candidate pool, then use it for both target and candidate samples. The default parameters are $J = 5$, $\gamma = 0.9$, $\tau = 0.05$, and $\alpha = 0.5$. Further scoring details are given in Appendix C. We run BiTFlow scoring and selection once with a fixed seed. Following the fine-tuning setup of Xia et al. (2024), we fine-tune a fresh model on each selected subset with LoRA (Hu et al., 2022) for four epochs. Tab. 1 reports means and sample standard deviations over training seeds 3, 6, and 9 for the LESS pool and training seed 6 for SmolTalk. Other experiments use the seed aggregation stated in their captions. The detailed training setup is in Appendix F.

Baselines and benchmarks. We compare BiTFlow with surface-statistics selectors BM25S (Lù, 2024) and DSIR (Xie et al., 2023); forward-only selectors NAIT (Chen et al., 2026), RDS+ (Ivison et al., 2025), CRDS-R and CRDS-W (Shu et al., 2026), and TRIM (Nagaraj et al., 2026); and training-signal selectors LESS (Xia et al., 2024), TAGCOS (Zhang et al., 2025a), S2L (Yang et al., 2024), and CLD (Nagaraj et al., 2025). Random 5% selection and full-data fine-tuning provide reference points. Selectors that use target samples share the same target set for each benchmark; TAGCOS and S2L select target-independent subsets. Baseline protocol details are provided in Appendix D. Both candidate pools are evaluated on MMLU (Hendrycks et al., 2021), BBH (Suzgun et al., 2023), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), OpenBookQA (OBQA) (Mihaylov et al., 2018), and AlpacaEval (Dubois et al., 2023). The LESS pool is additionally evaluated on TyDiQA (Clark et al., 2020). We omit TyDiQA for the SmolTalk pool because its English-language candidate samples do not provide multilingual instruction supervision for this benchmark. Appendix E describes candidate-pool construction, target sets, and evaluation metrics.

4.1 Performance on Different-Size Candidate Pools

At the same 5% selection budget, BiTFlow outperforms state-of-the-art selectors (Xia et al., 2024; Nagaraj et al., 2026; Shu et al., 2026) in average downstream performance on both candidate pools and exceeds full-data fine-tuning on the LESS pool (Tab. 1). On the LESS pool, the largest margin appears on AlpacaEval. On the curated SmolTalk pool (Ben Allal et al., 2025), fingerprint-based selectors and BM25S all exceed random selection, and BiTFlow leads the runner-up RDS+ by only 0.78 points, suggesting that similarity alone may already retrieve relevant samples. These results use the pretrained scorer without warmup training. Scorer warmup helps slightly on the LESS pool but not on SmolTalk, at extra training cost (Tab. 1).

Table 1: Downstream performance (%) at a 5% selection budget. HumanEval reports pass@10; AlpacaEval reports win rate. Avg. is the mean over all benchmarks in each pool (seven for LESS, six for SmolTalk). **Bold**/underline: best/second-best selector; †: scorer warmup. LESS results are three-seed means ± sample standard deviation; SmolTalk reports only one training seed due to training time.

	LESS pool (0.27M samples)								SmolTalk pool (1.04M samples)						
Method	**MMLU**	**BBH**	**GSM8K**	**Human Eval**	**OBQA**	**Alpaca Eval**	**TyDiQA**	**Avg.**	**MMLU**	**BBH**	**GSM8K**	**Human Eval**	**OBQA**	**Alpaca Eval**	**Avg.**
References															
Random 5%	46.52 ±0.24	41.57 ±1.18	15.28 ±0.59	25.23 ±1.41	51.20 ±0.85	15.25 ±0.13	50.42 ±0.33	35.07	47.72	43.15	20.85	29.05	53.60	48.70	40.51
Full data	51.44 ±0.12	43.38 ±0.07	24.03 ±0.43	27.25 ±1.03	64.20 ±0.00	24.69 ±0.48	56.14 ±0.20	41.59	50.28	46.85	36.47	35.81	60.40	54.78	47.43
Surface-statistics selectors															
BM25S (Lù, 2024)	48.81 ±0.11	42.27 ±0.33	26.35 ±0.59	27.25 ±2.06	**58.20** ±1.70	28.63 ±0.70	52.52 ±0.05	40.58	46.95	40.37	36.01	31.76	61.20	50.93	44.54
DSIR (Xie et al., 2023)	46.76 ±0.07	40.65 ±0.26	14.18 ±1.07	23.20 ±2.17	47.70 ±0.99	6.93 ±0.48	48.73 ±0.43	32.59	47.53	42.22	27.90	29.05	54.80	24.84	37.72
Forward-only selectors															
NAIT (Chen et al., 2026)	45.02 ±0.68	40.19 ±0.37	22.11 ±1.02	25.90 ±2.56	50.47 ±1.36	30.25 ±0.62	54.27 ±0.79	38.32	47.08	43.06	11.07	27.03	37.00	**51.18**	36.07
CRDS-R (Shu et al., 2026)	46.60 ±0.24	42.84 ±1.09	27.80 ±0.77	27.03 ±1.79	49.33 ±0.81	23.66 ±0.16	52.76 ±0.41	38.57	49.84	**45.56**	40.79	**38.51**	58.40	42.30	45.90
CRDS-W (Shu et al., 2026)	46.68 ±0.37	**43.66** ±0.07	25.85 ±0.21	27.03 ±2.44	56.90 ±1.56	18.91 ±0.92	52.53 ±0.48	38.79	**50.30**	43.33	34.50	33.78	58.20	45.03	44.19
RDS+ (Ivison et al., 2025)	46.12 ±0.36	41.20 ±0.65	25.66 ±0.05	26.13 ±2.73	52.20 ±0.28	19.35 ±0.22	53.03 ±0.43	37.67	49.64	43.70	39.95	35.14	61.20	46.58	46.04
TRIM† (Nagaraj et al., 2026)	45.25 ±0.35	40.28 ±0.00	28.89 ±0.00	28.15 ±3.05	53.80 ±1.70	8.73 ±0.13	54.50 ±0.28	37.09	47.78	44.44	41.02	33.11	56.60	26.21	41.53
Training-signal selectors															
LESS† (Xia et al., 2024)	45.99 ±0.26	42.45 ±0.07	18.39 ±0.16	24.77 ±3.05	48.90 ±1.27	17.11 ±1.19	**57.40** ±0.25	36.43	48.36	41.39	26.61	25.00	59.60	36.46	39.57
TAGCOS† (Zhang et al., 2025a)	45.46 ±0.44	39.44 ±1.18	17.97 ±0.54	24.10 ±1.70	49.80 ±0.85	15.50 ±0.31	52.04 ±0.03	34.90	48.44	40.56	23.05	22.97	52.00	48.63	39.27
S2L (Yang et al., 2024)	45.71 ±0.68	40.87 ±0.72	14.84 ±2.35	25.90 ±1.41	53.33 ±2.04	10.75 ±0.55	52.27 ±0.94	34.81	47.88	41.94	19.71	31.08	53.20	44.29	39.68
CLD (Nagaraj et al., 2025)	46.31 ±0.38	42.38 ±1.07	16.86 ±0.97	26.13 ±2.73	49.60 ±1.74	17.70 ±0.59	53.49 ±0.15	36.07	47.51	43.24	17.29	27.03	52.80	48.63	39.42
BiTFlow (ours)	**48.92** ±0.29	41.45 ±1.81	**29.39** ±0.80	**28.60** ±1.56	52.20 ±1.60	37.43 ±1.22	55.47 ±0.76	41.92	48.55	44.07	**41.70**	35.14	61.40	50.06	**46.82**
BiTFlow + warmup†	48.63 ±0.09	40.79 ±0.07	28.54 ±0.91	26.35 ±2.70	54.20 ±0.00	**40.28** ±0.40	55.37 ±0.51	**42.02**	48.84	43.06	**41.70**	33.11	**62.60**	50.06	46.56

4.2 Beyond Last-Layer Similarity

Across reading comprehension, translation, and arithmetic, the same response can receive different intermediate updates under two prompts while its last-layer states remain nearly identical (Fig. 3). These differences show that last-layer similarity can obscure how the response is updated across layers (Appendix G.1).

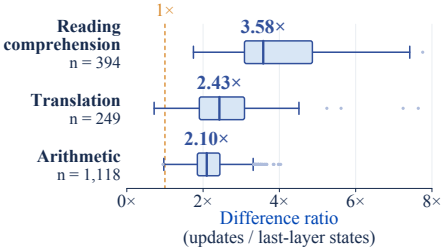

Figure 3: Residual updates reveal differences obscured by nearly identical last-layer states. Details are in Appendix G.1.

BiTFlow helps mitigate last-layer similarity collapse on TyDiQA. When samples end in nearly identical last-layer states, a selector that compares only these states has no basis to prefer one over another, whereas BiTFlow can still separate them by their residual updates. In Fig. 4, TRIM selects an inaccurate translation (A), while BiTFlow selects a fluent Russian sentence (B) that is a close last-layer neighbor of A but differs in intermediate updates. Swapping 3.39% of each baseline's selected coreset for our selected samples that last-layer scoring ranks alike improves mean TyDiQA F1 for each baseline, including ours with last-layer states (Tab. 2; Appendix G.2).

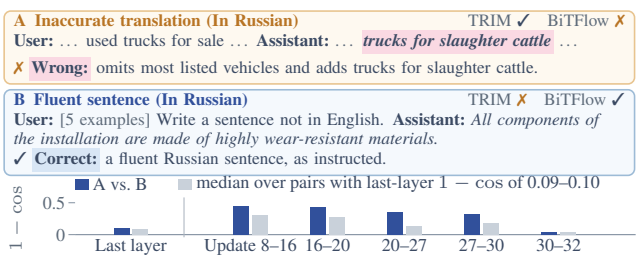

Figure 4: A TyDiQA case (post hoc; translated, originals in Fig. 6): A and B are close in the last layer but farther apart in layers 8–30.

Table 2: TyDiQA F1 gain from replacing part of each baseline coreset with our selected samples ranked alike under last-layer states. Three-seed mean and sample standard deviation are reported.

Replaced coreset	**Δ F1**
TRIM	+0.81 ±0.24
RDS+	+0.23 ±0.27
Ours (last layer)	+0.23 ±0.15

4.3 Selection-Budget Scaling

BiTFlow maintains the highest three-task average across the tested selection budgets and exceeds full-data fine-tuning from 5% onward (Fig. 5), indicating that the top 5% of ranked candidates already provide effective supervision. Task-level trends diverge at larger budgets: MMLU and GSM8K peak before the largest subset, whereas TyDiQA continues to improve. As the budget grows, selection extends to lower-scoring samples, which may include less relevant supervision that offsets the benefit of the highest-ranked samples (Appendix G.3).

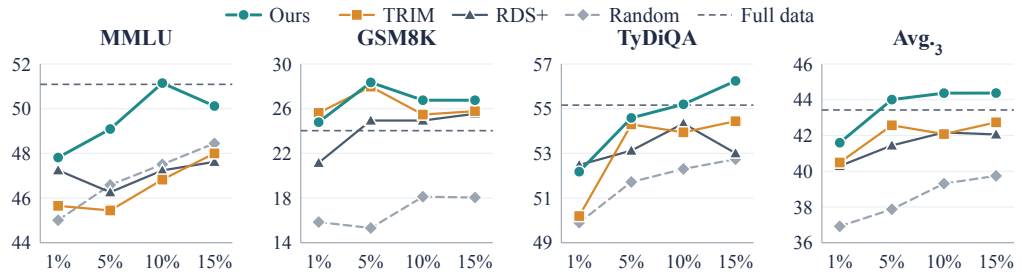

Figure 5: Selection-budget scaling on the LESS pool. Each method ranks candidates once; smaller budgets select only the top-ranked samples from that ranking. $\mathrm{Avg.}_3$ is the mean of the MMLU, GSM8K, and TyDiQA scores. Dashed lines show full-data fine-tuning.

4.4 Transfer across Model Scales and Families

Table 3: Transfer of selected data across models. We use Llama-2-7B to select 5% of the LESS pool and fine-tune the models listed below on the selected samples. Values are means with sample standard deviations. Δ denotes the score difference (BiTFlow minus Random), computed within each training run.

	Llama-2-13B			Mistral-7B-v0.3		
Data	**MMLU**	**GSM8K**	**TyDiQA**	**MMLU**	**GSM8K**	**TyDiQA**
Random	53.63 ±.31	27.50 ±.44	56.56 ±.62	59.71 ±.40	44.98 ±.42	56.24 ±.78
BiTFlow	55.89 ±.20	38.41 ±.82	56.81 ±.40	59.90 ±.12	52.62 ±1.40	57.65 ±.45
Δ	+2.26 ±.33	+10.92 ±.65	+0.26 ±.46	+0.19 ±.52	+7.63 ±1.29	+1.42 ±.61

On MMLU and GSM8K, the largest gains over random selection occur when subsets selected by Llama-2-7B are used to fine-tune Llama-2-13B (Tab. 3). The two Llama models share a tokenizer and pretraining data (Touvron et al., 2023), so they may exhibit similar residual update patterns. The selected samples also improve mean performance on Mistral (Jiang et al., 2023), although the average gains are smaller. Selections made with Llama-2-7B therefore remain beneficial when training the other tested models.

4.5 Ablation Study

Table 4: Ablation studies on the LESS pool (5% selection budget). (a) Encoding design: downstream performance (%; mean ± sample standard deviation across three seeds) for hidden-state and residual-flow encodings. All variants share the same prototype and matching framework. proj./unproj.: with/without the 128-dimensional projection. Residual flow is the default. (b) Matching and aggregation: differences in three-seed mean scores (Δ, percentage points; former minus latter). The first row compares bipartite and many-to-one matching at $\tau = 0$; "Default" follows Tab. 1. (c) Hyperparameter sensitivity: three-seed mean shifts in $\mathrm{Avg.}_3$ (percentage points) relative to the default $(\tau, \alpha) = (0.05, 0.5)$; full results are in Tab. 15.

(a) Encoding design

Variant	**MMLU**	**GSM8K**	**TyDiQA**
Last layer, unproj.	48.30 ±0.37	28.63 ±0.54	54.45 ±0.56
Last layer, proj.	47.91 ±0.31	28.51 ±0.69	55.66 ±0.30
Multi-layer, unproj.	48.90 ±0.21	28.89 ±0.53	53.91 ±0.28
Multi-layer, proj.	49.02 ±0.31	28.78 ±0.46	55.59 ±0.71
Residual flow	48.92 ±0.29	29.39 ±0.80	55.47 ±0.76

(b) Matching and aggregation

Comparison	**MMLU**	**GSM8K**	**TyDiQA**
1-to-1 − many-to-1	+0.49	+0.25	+0.55
Default − exact only	+1.36	+0.48	+1.42
Default − mean	+0.10	+3.87	+2.37
Default − max	+1.22	+1.31	+1.41
Default − TRIM-style	+0.02	+3.16	+0.99

(c) Hyperparameter sensitivity

Matching cost τ	0	.01	.025	**.05**	.075	.10
$\Delta\mathrm{Avg.}_3$	+0.16	+0.08	+0.25	0.00	−0.70	+0.05

Prompt weight α	0	.25	**.5**	.75	1
$\Delta\mathrm{Avg.}_3$	−1.35	−0.26	0.00	−0.93	−2.00

Encoding design. With the projection, multi-layer state encoding and residual-flow encoding both yield higher mean scores than last-layer encoding on MMLU and GSM8K, while TyDiQA scores are comparable across the three (Tab. 4(a)). Neither cross-layer encoding consistently dominates the

other: residual flow scores higher on GSM8K, whereas multi-layer states score slightly higher on MMLU and TyDiQA. These results support using information from intermediate layers and position residual flow as an effective update-based choice rather than a uniquely superior encoding. The unprojected rows of Tab. 4(a) show that without dimensionality reduction, multi-layer states score lowest on TyDiQA despite using more layers; the improvement with projection suggests redundant directions would dilute the selection signal.

Matching and aggregation. Both scoring components improve over their alternatives (Tab. 4(b)). At the same matching cost ($\tau = 0$), one-to-one bipartite matching gives higher mean scores than many-to-one matching on all three tasks, consistent with preventing a sample from inflating its score by repeatedly matching the same target prototypes. Balanced aggregation outperforms TRIM's aggregation rule applied to our token evidence on GSM8K and TyDiQA and is on par on MMLU (Appendix H): TRIM scores only candidate-side token statistics (Nagaraj et al., 2026), whereas our harmonic mean also rewards coverage of distinct target token IDs, measured within each sample rather than across the coreset. Because the components may interact, these one-at-a-time ablations do not split the overall gain among encoding, matching, and aggregation.

Hyperparameter sensitivity. Performance is more sensitive to prompt–response weighting than to matching cost (Tab. 4(c); Appendix H.4). Low matching costs ($\tau \leq 0.05$) yield similar three-task averages. On the weighting axis, MMLU peaks at $\alpha = 0.75$, whereas GSM8K and TyDiQA peak at $\alpha = 0.5$. Prompt-only scoring lowers GSM8K most, and response-only scoring lowers TyDiQA most; MMLU is less affected in both cases. These results support combining both roles, with their relative contributions depending on the task.

4.6 Computational Cost and Efficiency

In Tab. 5, F and G denote forward and forward–backward costs over N candidates, $W = \omega n_e G$ is warmup on a fraction ω for n_e epochs, n_c is the number of checkpoints, and A_{method} covers preparation, feature processing, and selection. Forward-only selectors score candidates in comparable time, all under 31 GPU-ms/sample, while gradient-based LESS takes roughly $18\times$ longer due to backward passes at multiple checkpoints. Despite the additional cost of encoding and matching residual updates, BiTFlow scores candidates at $1.03\times$ the time of TRIM. Unlike TRIM, BiTFlow requires no warmup training, so its total selection cost reduces to $\mathcal{O}(F + A_{\text{RF}})$, in which candidate scoring scales linearly with pool size (Appendix I.1).

Table 5: Selection costs and candidate-scoring time on the LESS pool. Scoring time includes candidate feature extraction but excludes preparation (e.g., reference fitting, warmup training) and final subset selection. Relative time is normalized to TRIM.

Method	Total selection cost	Scoring time (GPU-ms/sample)	Relative to TRIM
CRDS-W	$\mathcal{O}(F + A_{\text{CRDS-W}})$	27.00	$0.93\times$
RDS+	$\mathcal{O}(F + A_{\text{RDS+}})$	28.59	$0.98\times$
TRIM	$\mathcal{O}(W + F + A_{\text{TRIM}})$	29.07	$1.00\times$
LESS	$\mathcal{O}(W + n_c G + A_{\text{LESS}})$	547.98	$18.85\times$
BiTFlow	$\mathcal{O}(F + A_{\text{RF}})$	30.03	$1.03\times$

5 Conclusion

In this work, we study how information from intermediate layers can guide coreset selection for instruction tuning. We identify *last-layer similarity collapse*: highly similar last-layer states can obscure differences in intermediate token updates. We capture this cross-layer information with residual flows, the residual updates of each token between selected layers, and introduce BiTFlow, a highly efficient, forward-only coreset selection algorithm that compares candidate samples with a small target set at the token level. BiTFlow aligns candidate tokens to target token prototypes via one-to-one bipartite matching, which keeps repeated tokens from inflating scores, and scores each sample by a balanced aggregation of how well its tokens match and how many distinct target token IDs are covered. At a 5% selection budget on the LESS and SmolTalk candidate pools, BiTFlow outperforms state-of-the-art methods (Xia et al., 2024; Nagaraj et al., 2026; Shu et al., 2026) on average and exceeds full-data fine-tuning on several tasks.

AI USE STATEMENT

In this work, generative AI tools were used solely to improve the readability and linguistic clarity of the manuscript. They were not used to formulate research ideas, design methods or experiments, generate or analyze data, implement the proposed methods, or interpret results. All substantive research contributions were made by the authors, who reviewed and verified all AI-assisted edits and take full responsibility for the final manuscript. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, or interpret results. Generating synthetic datasets and assisting in the writing of proofs are not applicable to this work. The authors reviewed the AI-assisted material and take full responsibility for the content of this paper.

REFERENCES

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. A survey on data selection for language models. *Transactions on Machine Learning Research*, 2024.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. SmolLM2: When smol goes big—data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Jinhe Bi, Danqi Yan, Yifan Wang, Wenke Huang, Haokun Chen, Guancheng Wan, Mang Ye, Xun Xiao, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. CoT-Kinetics: A theoretical modeling assessing LRM reasoning process. *arXiv preprint arXiv:2505.13408*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xin Chen, Junchao Wu, Shu Yang, Runzhe Zhan, Zeyu Wu, Min Yang, Shujian Huang, Lidia S. Chao, and Derek F. Wong. Neuron-aware data selection in instruction tuning for large language models. In *ICLR*, 2026.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 2020.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Hamed Damirchi, Ignacio Meza De la Jara, Ehsan Abbasnejad, Afshar Shamsi, Zhen Zhang, and Javen Shi. Truth as a trajectory: What internal representations reveal about large language model reasoning. In *ACL*, 2026.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback. In *NeurIPS*, 2023.

Aliai Eusebi, Alexander Herzog, Xiaoyu Liang, Marie Vasek, Enrico Mariconti, and Lorenzo Cavallaro. Reading calibrated uncertainty from language model trajectories. *arXiv preprint arXiv:2605.22864*, 2026.

Jesseba Fernando and Grigori Guitchounts. Transformer dynamics: A neuroscientific approach to interpretability of large language models. *arXiv preprint arXiv:2502.12131*, 2025.

Davide Ghilardi, Federico Belotti, Marco Molinari, Tao Ma, and Matteo Palmonari. Group-SAE: Efficient training of sparse autoencoders for large language models via layer groups. In *EMNLP*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Hamish Ivison, Muru Zhang, Faeze Brahman, Pang Wei Koh, and Pradeep Dasigi. Large-scale data selection for instruction tuning. *arXiv preprint arXiv:2503.01807*, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Tim Lawson, Lucy Farnik, Conor Houghton, and Laurence Aitchison. Residual stream analysis with multi-layer SAEs. In *ICLR*, 2025.

Xing Han Lù. BM25S: Orders of magnitude faster lexical search via eager sparse scoring. *arXiv preprint arXiv:2407.03618*, 2024.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. Understanding and improving Transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*, 2019.

Da Ma, Gonghu Shang, Zhi Chen, Libo Qin, Yijie Luo, Hongshen Xu, Lei Pan, Shuai Fan, Kai Yu, and Lu Chen. Task-specific data selection for instruction tuning via monosemantic neuronal activations. In *NeurIPS*, 2025.

Mohamed A. Mabrok. Latent semantic manifolds in large language models. *arXiv preprint arXiv:2603.22301*, 2026.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Viraaji Mothukuri and Reza M. Parizi. Trajectory signatures of deception in large language models. In *ACL*, 2026.

Manish Nagaraj, Deepak Ravikumar, and Kaushik Roy. Coresets from trajectories: Selecting data via correlation of loss differences. *Transactions on Machine Learning Research*, 2025.

Manish Nagaraj, Sakshi Choudhary, Utkarsh Saxena, Deepak Ravikumar, and Kaushik Roy. TRIM: Token-wise attention-derived saliency for data-efficient instruction tuning. In *ICML*, 2026.

Youwei Shu, Shaomian Zheng, Dingnan Jin, Wenjie Qu, Ziyao Guo, Qing Cui, Jun Zhou, and Jiaheng Zhang. On representation redundancy in large-scale instruction tuning data selection. *arXiv preprint arXiv:2602.13773*, 2026.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of ACL*, 2023.

Anh Tong, Thanh Nguyen-Tang, Dongeun Lee, Duc Nguyen, Toan Tran, David Hall, Cheongwoong Kang, and Jaesik Choi. Neural ODE transformers: Analyzing internal dynamics and adaptive fine-tuning. In *ICLR*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Yiming Wang, Pei Zhang, Baosong Yang, Derek F. Wong, and Rui Wang. Latent space chain-of-embedding enables output-free LLM self-evaluation. In *ICLR*, 2025.

Jianghao Wu, Jianfei Cai, Weiqiang Wang, Jin Ye, Daniel F. Schmidt, and Yasmeen George. Single-rollout hidden-state dynamics for training-free RLVR data selection. In *ICML*, 2026.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *ICML*, 2024.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. In *NeurIPS*, 2023.

Yu Yang, Siddhartha Mishra, Jeffrey Chiang, and Baharan Mirzasoleiman. SmallToLarge (S2L): Scalable data selection for fine-tuning large language models by summarizing training loss trajectories of small models. In *NeurIPS*, 2024.

Jipeng Zhang, Yaxuan Qin, Renjie Pi, Weizhong Zhang, Rui Pan, and Tong Zhang. TAGCOS: Task-agnostic gradient clustered coreset selection for instruction tuning data. In *Findings of NAACL*, 2025a.

Zhenliang Zhang, Xinyu Hu, Huixuan Zhang, Junzhe Zhang, and Xiaojun Wan. ICR probe: Tracking hidden state dynamics for reliable hallucination detection in LLMs. In *ACL*, 2025b.

A Notation and Operator Definitions

Tab. 6 summarizes the main symbols used in the method and appendix. Token sets, prototypes, matching, and role scores are defined separately for prompt and response; role subscripts are omitted unless needed.

Table 6: Summary of notation and operators. Definitions follow Sec. 3 and the corresponding appendix analyses.

Symbol / operator	Definition
Samples, token instances, and selection	
$\mathcal{D}_{\text{pool}},\ N$	Candidate pool and its number of samples.
$\mathcal{D}_{\text{tar}}$	Target set whose samples guide selection.
$x_i,\ y_j;\ x$	Candidate sample, target sample, and a generic candidate sample.
$K,\ \mathcal{D}^{\star}_{\text{sel}}$	Selection budget in samples and the selected subset of size K.
ρ	Token role: prompt or response.
$t,\ t'$	Token instances; t' denotes a candidate instance in matching.
Residual flow, encoding, and prototypes	
$\ell_0 < \cdots < \ell_m$	Selected layer indices; ℓ_m is the last selected layer (the last decoder block by default).
$m,\ k$	Number of residual updates and update index $k \in \{1, \ldots, m\}$; there are $m+1$ selected layers.
$d_h,\ d_z$	Hidden-state dimension and projected-update dimension.
H_t^{ℓ}	Token state after block ℓ, before final layer normalization; H_t^0 denotes the input embedding.
$\Delta h_{t,\ell_k}$	Residual update $H_t^{\ell_k} - H_t^{\ell_{k-1}}$ between successive selected layers.
$\mathcal{R}(t)$	Residual flow of token instance t: the ordered tuple of residual updates $(\Delta h_{t,\ell_1}, \ldots, \Delta h_{t,\ell_m})$.
$\mu_{\ell_k},\ U_{\ell_k}$	Reference mean and PCA projection fitted on candidate-pool updates; shared by target and candidate encoding.
z_{t,ℓ_k}	Centered, projected, and normalized update direction (Eq. 5); zero follows the normalization convention below.
$C,\ D(x)$	Distinct token-ID sets in the target set and candidate x, respectively, within the current role.
$c,\ d$	Target and candidate token IDs, respectively.
$\mathcal{T}_c,\ A_d(x)$	Target instances with token ID c and instances of token ID d in candidate x, within the current role.
p_{c,ℓ_k}	Target prototype direction $\text{unit}(\sum_{t\in\mathcal{T}_c} z_{t,\ell_k})$; the tuple over k forms the token prototype.
Token matching and evidence aggregation	
$r(t', c)$	Mean alignment of a candidate instance with target prototype c at corresponding selected layers (Eq. 7).
$r_c(x),\ I$	Best instance alignment for a shared token ID c and the shared token-ID set $I = D(x) \cap C$.
$s_{\text{emb}}(d, c)$	Cosine similarity of the base model's input embeddings for token IDs d and c.
$J,\ \mathcal{N}_J(d)$	Fallback shortlist size and the up-to-J same-role target IDs with the largest positive input-embedding cosine similarities, before exact-matched IDs are removed.
γ	Multiplicative discount on fallback edge quality, with $0 < \gamma \leq 1$.
$q(d, c),\ \tau$	Fallback edge quality and nonnegative matching cost. Only edges with $q > \tau$ enter the graph; matching maximizes gains $q - \tau$.
$\mathcal{G},\ \mathcal{M}^{\star}$	Fallback bipartite graph and selected one-to-one matching, with unmatched nodes allowed.
$E(x)$	Positive matched evidence: exact supports $[r_c(x)]_+$ plus the full qualities q of selected fallback edges, without subtracting τ.
$P(x),\ R(x)$	Candidate precision $E(x)/\lvert D(x)\rvert$ and target recall $E(x)/\lvert C\rvert$; both denominators retain the full valid token-ID sets.
$s(x),\ s_\rho(x)$	Role score $2P(x)R(x)/(P(x)+R(x)+\epsilon)$; zero for a candidate role with no valid instances. The subscript ρ makes the role explicit.
$\epsilon,\ \alpha$	Positive harmonic-mean stabilizer and prompt-score weight $\alpha \in [0, 1]$; the response weight is $1-\alpha$.
$S(x; \mathcal{D}_{\text{tar}})$	Final sample score $\alpha s_{\text{prompt}}(x) + (1-\alpha) s_{\text{response}}(x)$ used to rank candidates.
Theoretical analysis	
$u_k,\ \bar{u}$	Pairwise update difference $\Delta h_{t,\ell_k} - \Delta h_{t',\ell_k}$ and its mean $m^{-1}\sum_k u_k$.
$d_{\text{start}},\ d_{\text{last}}$	Differences between two token states at ℓ_0 and ℓ_m, respectively (Proposition 1).
T	Summation map $T(v_1, \ldots, v_m) = \sum_k v_k$ on residual-update tuples; distinct from the target-instance set $\mathcal{T}_c$.
$\Phi(t)$	Stacked encoding $m^{-1/2}(z_{t,\ell_1}, \ldots, z_{t,\ell_m})$ used in the alignment identity (Eq. 18).
Computational cost	
$f_{\text{fw}},\ f_{\text{bw}}$	Per-sample forward- and backward-pass costs.

Continued on the next page.

Tab. 6 (continued).

Symbol / operator	Definition
$F,\ G$	Candidate-pool model-pass costs: $F = Nf_{\mathrm{fw}}$ and $G = N(f_{\mathrm{fw}} + f_{\mathrm{bw}})$. A superscript proxy denotes costs for the proxy model.
$\omega,\ n_e,\ n_c$	Warmup fraction, number of training epochs, and number of scored checkpoints; the training and checkpoint counts refer to each method's configuration.
W	Warmup model-pass cost $\omega n_e G$, excluding additional preparation and optimizer costs.
$A_{\mathrm{method}},\ A_{\mathrm{RF}}$	Additional selection costs outside the warmup and candidate model-pass terms; A_{RF} is the term for BiTFlow. These costs may grow with N.
$C_{\mathrm{prep}},\ n_r$	Preparation cost (reference fitting, target construction, and other preprocessing) and the reference token budget.
$t_x,\ M_x$	Number of processed token positions in candidate x and the cost of constructing and solving its fallback assignment.
c_x	Additional candidate-processing cost for projection, comparisons, instance maxima, and assignment (Eq. 19); excludes the model forward pass.
Signal analysis	
H^{ℓ}	Mean hidden state over response token positions at layer ℓ.
Δh_{ℓ_k}	Difference between the mean hidden states at layers ℓ_{k-1} and ℓ_k.
$\Omega_{\mathrm{vec}},\ \Omega_{\mathrm{layer}},\ \Omega_{\mathrm{cat}}$	Random projection matrices used to construct the probe inputs.
π	Seed-determined permutation of five update blocks, drawn independently for each pair and shared by its two samples.
$A,\ B$ (superscripts)	The two samples used to compute the difference ratio (Appendix G.1).
Operators	
$\mathrm{unit}(v)$	$v/\|v\|_2$ for $v \neq 0$, and zero for $v = 0$.
$[a]_+$	Positive part $\max(a, 0)$.
$\langle u, v\rangle,\ \|v\|_2$	Euclidean inner product and norm; $\cos(u, v)$ is cosine similarity for nonzero vectors.
$\|C\|,\ \deg_{\mathcal{M}}(c)$	Set cardinality and the number of matching edges incident to token ID c.
$\ker(T),\ \dim\ker(T)$	Null space of the summation map and its dimension, $(m-1)d_h$.
$\arg\max,\ \mathcal{O}(\cdot)$	Maximizing argument or set of arguments, and asymptotic upper-bound notation.

B Theoretical Analysis and Proofs

We formalize the update differences obscured by last-layer aggregation, explain the comparison of residual-update directions at corresponding selected layers, derive the target prototypes used by BiTFlow, and characterize the range of the balanced score.

B.1 Information Lost in the Last-Layer State

Proof of Proposition 1. By the definition of residual updates, the state differences decompose as:

$$H_t^{\ell_m} = H_t^{\ell_0} + \sum_{k=1}^{m} \Delta h_{t,\ell_k}. \tag{13}$$

For two token instances t, t', let

$$u_k = \Delta h_{t,\ell_k} - \Delta h_{t',\ell_k}, \qquad \bar{u} = \frac{1}{m}\sum_{k=1}^{m} u_k.$$

Writing their starting and last-layer state differences as d_{start} and d_{last}, respectively, gives

$$d_{\mathrm{last}} - d_{\mathrm{start}} = \sum_{k=1}^{m} u_k = m\bar{u}. \tag{14}$$

Since $\sum_k (u_k - \bar{u}) = 0$, expanding the squared norms yields

$$\begin{aligned} \sum_{k=1}^{m} \|u_k\|_2^2 &= \sum_{k=1}^{m} \|\bar{u} + (u_k - \bar{u})\|_2^2 \\ &= m\|\bar{u}\|_2^2 + \sum_{k=1}^{m} \|u_k - \bar{u}\|_2^2. \end{aligned} \tag{15}$$

Dividing by m and substituting $\bar{u} = (d_{\text{last}} - d_{\text{start}})/m$ gives

$$\frac{1}{m}\sum_{k=1}^{m}\|u_k\|_2^2 = \frac{\|d_{\text{last}} - d_{\text{start}}\|_2^2}{m^2} + \frac{1}{m}\sum_{k=1}^{m}\|u_k - \bar{u}\|_2^2. \tag{16}$$

The second term is nonnegative and vanishes exactly when all update differences equal $\bar{u}$. For fixed d_{start}, the last-layer difference determines the mean update difference, while comparison at corresponding selected layers also measures variation around this mean. This completes the proof.

Update differences removed by summation. The information loss can also be characterized through the summation map on update tuples,

$$T : (\mathbb{R}^{d_h})^m \to \mathbb{R}^{d_h}, \qquad T(v_1, \ldots, v_m) = \sum_{k=1}^{m} v_k.$$

Because $T(y, 0, \ldots, 0) = y$, the map has rank d_h. Rank–nullity therefore gives

$$\ker(T) = \left\{ (v_1, \ldots, v_m) : \sum_{k=1}^{m} v_k = 0 \right\}, \qquad \dim\ker(T) = (m-1)d_h. \tag{17}$$

For $m > 1$, distinct update tuples can thus have the same sum and, given the same starting state, produce the same last-layer state. This characterizes the update variation removed by summation and motivates comparing residual updates at corresponding selected layers.

B.2 Directional Encoding and Target Prototypes

Comparing corresponding updates. Taking an inner product after summing updates combines contributions from both corresponding and different selected layers:

$$\left\langle \sum_k \Delta h_{t,\ell_k}, \sum_{k'} \Delta h_{t',\ell_{k'}} \right\rangle = \sum_k \langle \Delta h_{t,\ell_k}, \Delta h_{t',\ell_k} \rangle + \sum_{k \neq k'} \left\langle \Delta h_{t,\ell_k}, \Delta h_{t',\ell_{k'}} \right\rangle.$$

BiTFlow encodes updates separately at each selected layer. The centering and PCA transformation at that layer are shared between target and candidate samples, and normalization makes the comparison depend on the resulting update directions.

To express this comparison geometrically, define

$$\Phi(t) = \frac{1}{\sqrt{m}}\big(z_{t,\ell_1}, \ldots, z_{t,\ell_m}\big).$$

When all encoded updates of both instances are nonzero and unit-normalized, $\|\Phi(t)\|_2 = \|\Phi(t')\|_2 = 1$, giving

$$\begin{aligned} \frac{1}{m}\sum_{k=1}^{m}(1 - \langle z_{t,\ell_k}, z_{t',\ell_k}\rangle) &= 1 - \langle \Phi(t), \Phi(t')\rangle \\ &= \frac{1}{2}\|\Phi(t) - \Phi(t')\|_2^2. \end{aligned} \tag{18}$$

The same identity holds when the second instance is replaced by a target prototype whose direction at every selected layer is nonzero. Thus, the base alignment compares directions in separate coordinate blocks for corresponding selected layers, preserving layer correspondence throughout the comparison.

Derivation of the target prototype. For a fixed role, token ID c, and selected layer ℓ_k, we seek a single unit direction that maximizes average alignment with the target instances in $\mathcal{T}_c$. Each instance contributes equally. Let

$$v = \sum_{t \in \mathcal{T}_c} z_{t,\ell_k}.$$

For $v \neq 0$, Cauchy–Schwarz gives

$$\frac{1}{|\mathcal{T}_c|}\sum_{t \in \mathcal{T}_c}\langle p, z_{t,\ell_k}\rangle = \frac{\langle p, v\rangle}{|\mathcal{T}_c|} \leq \frac{\|v\|_2}{|\mathcal{T}_c|} \qquad (\|p\|_2 = 1).$$

Equality holds at $p = v/\|v\|_2$, yielding

$$p_{c,\ell_k} = \mathrm{unit}\left(\sum_{t \in \mathcal{T}_c} z_{t,\ell_k}\right).$$

The normalized sum therefore gives the unit direction with maximum average alignment to the target instances. Applying this construction at each selected layer produces a token prototype that summarizes the target update directions while retaining their layer correspondence. If $v = 0$, every unit direction attains the same objective value; we define the stored prototype as zero in this case.

B.3 Range of the Balanced Score

For a fixed role, every term in $E(x)$ is at most one: $r(t', c)$ averages inner products of unit vectors, and $q(d, c) = \gamma\, s_{\mathrm{emb}}(d, c)\,[\cdot]_+$ with $\gamma \le 1$ and $s_{\mathrm{emb}}(d, c) \le 1$. All terms are nonnegative, since exact contributions are clipped at zero and fallback edges satisfy $q(d, c) > \tau \ge 0$. Exact-matched IDs are excluded from fallback matching on both sides, and Eq. 9 allows each remaining ID at most one fallback pair, so each candidate and target token ID enters at most one term. Hence

$$0 \le E(x) \le \min\{|D(x)|, |C|\}, \qquad P(x), R(x) \in [0, 1].$$

Ignoring ϵ and substituting $P(x) = E(x)/|D(x)|$ and $R(x) = E(x)/|C|$ into Eq. 11 gives

$$s(x) = \frac{2E(x)}{|D(x)| + |C|},$$

with $s(x) = 0$ when $E(x) = 0$. The score is thus a Dice-style overlap between the candidate and target token-ID sets in which each matched pair counts by its alignment quality: one-to-one matching keeps any target ID from being counted twice, and the normalization accounts for the sizes of both sets.

C Algorithm and Implementation Details

Algorithm 1 describes the selection procedure. Evaluation samples are held out throughout selection and fine-tuning.

Selection procedure. We construct target prototypes and compute matching scores separately for prompt and response tokens. For the current role, C is the set of distinct target token IDs, and p_{c,ℓ_k} is the prototype for token ID c at selected layer ℓ_k. $D(x)$ is the set of distinct candidate token IDs in that role, and $A_d(x)$ contains the instances with token ID d.

C.1 Scoring Configuration

The default scorer is pretrained Llama-2-7B without warmup training. States at blocks $(8, 16, 20, 27, 30, 32)$ define five bands, each projected to 128 dimensions. These five bands reduce the number of update vectors stored and compared while retaining the full model forward pass. Every target instance has equal weight in its token prototype, and target recall weights distinct token IDs equally. The default parameters are $J = 5$, $\gamma = 0.9$, and $\tau = 0.05$. Layer and parameter comparisons appear in Sec. 4.5 and Appendix H.

Role weight and empty inputs. Prompt and response use separate prototypes and matching graphs but share the residual projection. Eq. 12 uses $\alpha = 0.5$ by default. A candidate role with no valid token instances receives a score of zero. A small positive constant stabilizes the harmonic-mean denominator in Eq. 11.

C.2 Reference Construction

We fit the shared residual-flow encoding using a reference sampled from the candidate pool. The reference contains 50,000 prompt token positions and 50,000 response token positions, sampled approximately evenly across data sources within each role. These counts refer to positions, so repeated token IDs are included.

For each selected sample, we feed the complete prompt–response sequence to the frozen model and track the hidden states at the selected positions. We then extract the residual updates for each band using Eq. 4. We jointly fit one centered PCA projection per band to prompt and response updates. The resulting means and projection matrices are saved and reused for both target and candidate encoding. We use Incremental PCA with 128 components and a fitting batch size of 512, without rescaling components by their variances.

Algorithm 1 BiTFlow selection with a frozen model

Input: Frozen model; candidate pool $\mathcal{D}_{\text{pool}} = \{x_i\}_{i=1}^{N}$; target set $\mathcal{D}_{\text{tar}}$; budget K; selected layers $\ell_0 < \cdots < \ell_m$; $J, \gamma, \tau, \alpha, \epsilon$.
Output: Selected subset $\mathcal{D}^{\star}_{\text{sel}}$ of K samples.

Fit the mean μ_{ℓ_k} and PCA projection U_{ℓ_k} for each selected layer ℓ_k, $k = 1, \ldots, m$, using residual updates at token positions sampled from the candidate pool (Appendix C.2).
For each target sample, run the frozen model on its complete prompt–response sequence. At each token position retained under Appendix C.3, compute the residual updates between successive selected layers and their encodings using Eq. 5.
for each role $\rho \in \{\text{prompt}, \text{response}\}$ **do**
  Let C contain the distinct target token IDs for role ρ, and let $\mathcal{T}_c$ contain all retained target positions with ID c. For each c and k, compute $p_{c,\ell_k} = \text{unit}(\sum_{t \in \mathcal{T}_c} z_{t,\ell_k})$ (Eq. 6).
  For each vocabulary token ID d, store $\mathcal{N}_J(d)$: up to J target IDs $c \in C$ with the largest positive cosine similarities between the input embeddings of d and c.
end for
for each candidate $x \in \mathcal{D}_{\text{pool}}$ **do**
  Run the frozen model on the complete prompt–response sequence of x. Compute the residual updates and their encodings at the retained token positions, using the same means and PCA projections as for the target samples.
  for each role $\rho \in \{\text{prompt}, \text{response}\}$ **do**
    Use the target prototypes and $\mathcal{N}_J$ for role ρ. Let $D(x)$ contain the distinct candidate token IDs in this role, and let $A_d(x)$ contain all retained candidate positions with ID d.
    if $D(x) = \varnothing$, set $s_\rho(x) \leftarrow 0$ and **continue**.
    $I \leftarrow D(x) \cap C$; $r_c(x) \leftarrow \max_{t' \in A_c(x)} r(t', c)$ for $c \in I$ (Eq. 7).
    $E \leftarrow \sum_{c \in I} [r_c(x)]_+$; $\mathcal{G} \leftarrow \varnothing$.
    for $d \in D(x) \setminus I$ and $c \in \mathcal{N}_J(d) \setminus I$ **do**
      $q(d, c) \leftarrow \gamma\, s_{\text{emb}}(d, c)[\max_{t' \in A_d(x)} r(t', c)]_+$.
      if $q(d, c) > \tau$, add edge (d, c) with weight $q(d, c)$ to $\mathcal{G}$.
    end for
    Solve Eq. 9 on $\mathcal{G}$ for $\mathcal{M}^{\star}$: maximize $\sum_{(d,c) \in \mathcal{M}} (q(d, c) - \tau)$ with one-to-one bipartite matching and unmatched nodes allowed.
    $E \leftarrow E + \sum_{(d,c) \in \mathcal{M}^{\star}} q(d, c)$; $P \leftarrow E/|D(x)|$; $R \leftarrow E/|C|$.
    $s_\rho(x) \leftarrow 2PR/(P + R + \epsilon)$ (Eq. 11).
  end for
  $S(x; \mathcal{D}_{\text{tar}}) \leftarrow \alpha s_{\text{prompt}}(x) + (1 - \alpha) s_{\text{response}}(x)$ (Eq. 12).
end for
return the K samples with the highest $S(x_i; \mathcal{D}_{\text{tar}})$. For equal scores, select the sample with the smaller candidate-pool index i first.

Each role must contain at least one retained target token. A token ID present in both the candidate and target sets is handled only by exact matching, even when its contribution is zero. After these IDs are removed from a shortlist, no additional target IDs are added.

C.3 Tokens Used for Scoring

We tokenize each formatted prompt–response sequence once and use character offsets to assign each content token to the prompt or the response. In conversation-formatted samples, system and user messages form the prompt, and assistant messages form the response.

We skip three kinds of tokens when scoring: special tokens defined by the tokenizer (beginning-of-sequence, end-of-sequence, and padding); template role markers such as <|system|>, <|user|>, and <|assistant|> together with the separators between messages; and tokens that are empty or contain only whitespace. Role markers and separators are located by their positions in the formatted text, so they are skipped even when the tokenizer splits them into ordinary tokens.

Role markers and separators are still part of the model input. We compute residual-flow encodings only at the remaining positions and use them to construct prototypes and score candidates.

C.4 Fallback Matching

For each vocabulary token ID, we precompute up to J target neighbors within the same role using positive input-embedding cosine similarity. For each candidate, we remove all exact-matched IDs from the shortlists, including those with zero exact-match contribution. We do not add replacement neighbors.

We retain edges with $q > \tau$ and maximize $\sum(q - \tau)$ through one-to-one bipartite matching (Eq. 9), allowing IDs to remain unmatched. Each selected pair contributes q to $E(x)$. Candidate precision and target recall use the original sets $D(x)$ and C in their denominators. Equally good matchings follow a fixed token-ID order. When candidates have equal scores, we select the sample with the smaller candidate-pool index first.

D Baseline Implementation Details

Methods that use target samples select a separate subset for each benchmark. TAGCOS and S2L do not use target samples; they select one subset per candidate pool and reuse it across benchmarks.

Table 7: Selection methods used in the comparison across two candidate pools.

Method	Selection rule	Uses target samples
BM25S (Lù, 2024)	Lexical BM25 scores with the target aggregation rule of the LESS method	Yes
DSIR (Xie et al., 2023)	Hashed unigram/bigram importance resampling	Yes
NAIT (Chen et al., 2026)	Mean signed projection of candidate activations onto target directions	Yes
RDS+ (Ivison et al., 2025)	Cosine retrieval using position-weighted last-layer fingerprints	Yes
CRDS-R (Shu et al., 2026)	Cosine retrieval using concatenated random projections of multi-layer fingerprints	Yes
CRDS-W (Shu et al., 2026)	Cosine retrieval using whitened position-weighted last-layer fingerprints	Yes
TRIM (Nagaraj et al., 2026)	Token similarity to attention-weighted target fingerprints	Yes
LESS (Xia et al., 2024)	Candidate–target similarity of projected optimizer-aware gradients	Yes
TAGCOS (Zhang et al., 2025a)	Gradient clustering followed by orthogonal matching pursuit	No
S2L (Yang et al., 2024)	Clustering and sampling using response losses recorded during proxy-model training	No
CLD (Nagaraj et al., 2025)	Correlation between candidate and target loss differences	Yes
BiTFlow (ours)	Base-model residual-flow prototypes with bipartite matching and balanced aggregation	Yes

Lexical and distributional selectors. BM25S indexes the candidate pool and aggregates lexical scores over the shared target set before global top-K selection. DSIR estimates a target-to-pool density ratio in a hashed unigram/bigram space and selects samples through importance resampling.

Forward-only selectors. These selectors score candidates from the hidden states of a scorer model and differ in which states they compare and how. Except for TRIM, they use the pretrained scorer without warmup training.

RDS+, CRDS-R, and CRDS-W build a fingerprint for each sample, rank candidates by cosine similarity to target fingerprints, and retrieve candidates for each target in turn. RDS+ uses position-weighted last-layer fingerprints; we evaluate it over the full candidate pool without its token shortlist. CRDS-R builds position-weighted fingerprints from the last 16 layers, projects each layer's fingerprint to 256 dimensions with a Rademacher random matrix, and concatenates and L2-normalizes the projections. CRDS-W whitens position-weighted last-layer fingerprints; we fit its whitening transform on the complete candidate pool.

TRIM compares samples at the token level. It weights last-layer token states by attention saliency to construct target fingerprints, compares candidate token states with these fingerprints, and combines the token similarities into a sample score. Its scorer undergoes warmup training before fingerprint construction. We reimplemented TRIM following the paper (Nagaraj et al., 2026), as the authors' code was not publicly available at the time of our experiments.

NAIT (Chen et al., 2026) projects candidates onto target directions instead of comparing fingerprints. For each target sample, it forms a positive input from the complete text and a negative input from the initial token, and takes the last-token hidden state of each input at every layer. At each layer, it fits the first principal component of the positive–negative differences as a target direction. Each candidate is scored by projecting its last-token hidden states onto these directions and averaging the projections across layers. We adapt the released implementation to our candidate pools and training protocol.

Gradient-based selectors. LESS scores candidates by comparing their projected, optimizer-aware gradient features with target gradient features, then aggregates the scores across warmup checkpoints.

TAGCOS clusters gradient features and uses orthogonal matching pursuit to allocate the selection budget. It selects one subset for the candidate pool without using target samples.

Loss-based selectors. S2L records each candidate's response loss during proxy-model training, clusters candidates by their loss trajectories, and samples from these clusters while accounting for data sources. CLD computes changes in candidate and target losses across training checkpoints. It ranks each candidate by the correlation between its loss differences and the mean target loss differences, then selects the global top-K samples. Our CLD implementation adapts the original class-conditional vision method to supervised instruction tuning.

Scorer warmup. Following the LESS method, warmup training uses a target-independent random 5% subset of the corresponding candidate pool and the four-epoch LoRA recipe in Appendix F. The warmup variant of BiTFlow shares the same warmup checkpoint as LESS on the LESS pool and as TRIM on the SmolTalk pool.

E Data and Evaluation Protocol

E.1 Candidate Pools and Selection Budgets

The LESS pool follows the official mixture (Xia et al., 2024): 100,000 FLAN-v2 samples, 100,000 chain-of-thought samples, 15,011 Dolly samples, and 55,668 OpenAssistant samples, totaling 270,679. For SmolTalk, we use the `all/train` split of `HuggingFaceTB/smoltalk` (Ben Allal et al., 2025), which contains 1,043,917 samples. We retain the published source mixtures without deduplication.

We apply uniform quality filters to both pools. Each sample must contain a nonempty supervised response that fits within the 2,048-token context limit while reserving at least 64 tokens for the prompt; responses exceeding this budget are discarded rather than truncated. Samples whose text overlaps with held-out benchmark items are also removed. After filtering, 269,018 samples remain eligible in the LESS pool and 998,841 in SmolTalk. The full-data reference on the LESS pool trains on all eligible samples. S2L may additionally exclude samples whose complete responses exceed its proxy model's token limit.

The default selection budget is 5% of the original candidate-pool size before filtering, rounded up to 13,534 for the LESS pool and 52,196 for SmolTalk. Other selection fractions are specified in the corresponding experiments.

E.2 Target Sets

For each benchmark and candidate pool, selectors that use target samples share the same target set. We follow the LESS target-construction protocol (Xia et al., 2024) for MMLU, BBH, and TyDiQA, and use the fixed HumanEval target–evaluation split from RDS+ (Ivison et al., 2025). For GSM8K, we use the first ten training examples, with their complete rationales. For AlpacaEval and OpenBookQA, we sample ten validation samples from AlpacaFarm and OpenBookQA, respectively, without replacement. Tab. 8 lists the target sources and counts alongside the evaluation splits and metrics. Target and evaluation identifiers are checked for zero overlap before scoring.

For BBH, each of the three demonstrations in a task prompt serves once as the target response, with the other two providing context. For TyDiQA, selector targets and downstream 1-shot demonstrations use disjoint GoldP training examples.

E.3 Evaluation

Both candidate pools are evaluated on MMLU, BBH, GSM8K, OpenBookQA, AlpacaEval, and HumanEval. The LESS main comparison additionally includes TyDiQA. Avg. denotes the mean of seven benchmark scores for LESS and six for SmolTalk. Evaluation splits, prompting settings, and metrics are summarized in Tab. 8.

HumanEval follows the RDS+ decoding setup, with ten generations per evaluation problem at temperature 0.8. AlpacaEval uses greedy generation and reference answers from `text-davinci-003`. Gemini 2.5 Flash compares each model answer with the reference answer for the same instruction. We report raw win rate.

Table 8: Selector targets and held-out evaluation. Target counts denote samples available to the selector; all target sets are disjoint from the corresponding evaluation set. Scores are reported as percentages.

Benchmark	Targets	Target construction	Evaluation and metric
MMLU	285	Five development examples from each of 57 subjects	All 14,042 test examples; 5-shot subject-macro accuracy
BBH	81	Three fixed demonstrations from each of 27 task prompts	Up to 40 held-out examples per task; fixed task prompts and task-macro exact match
GSM8K	10	First ten training examples, with complete rationales	All 1,319 test examples; 8-shot exact match
AlpacaEval	10	AlpacaFarm validation instructions	805 AlpacaEval 1.0 instructions; greedy generation and raw win rate
OpenBookQA	10	OpenBookQA validation examples	All 500 test examples with all choices; accuracy
TyDiQA	9	One GoldP training example per language	GoldP development set; 1-shot per language, language-macro F1 and EM
HumanEval	16	Problems 0–15 with canonical solutions	Problems 16–163; ten generations per problem and pass@10

F Training Setup

In the main experiments, each selected subset is used to fine-tune a fresh Llama-2-7B model with LoRA (Hu et al., 2022), following the training setup of the LESS method (Xia et al., 2024). Tab. 9 lists the training hyperparameters. Training uses response-only loss and preserves the complete response, truncating the prompt when necessary. Gradient accumulation maintains a global batch size of 128.

Transfer of selected subsets. For transfer experiments, the 5% subsets selected with Llama-2-7B are reused to fine-tune Llama-2-13B (Touvron et al., 2023) and Mistral-7B-v0.3 (Jiang et al., 2023) without rescoring. Each model is compared with a Random subset at the same budget. Both models use 420 optimizer steps and the shared LoRA, optimizer, and batch settings. To retain complete responses, the maximum sequence length is 2,048 for Llama-2-13B and 3,072 for Mistral.

Table 9: Training hyperparameters for scorer warmup and downstream fine-tuning.

Parameter	Value
Hardware and precision	NVIDIA H100; bfloat16
Training objective	Response-only supervised loss
Epochs / maximum sequence length	4 / 2,048 tokens
Global batch size	128
Optimizer / weight decay	AdamW / 0
Learning rate / schedule	2×10^{-5} / linear decay
Learning-rate warmup	First 3% of training steps
LoRA rank / scaling factor / dropout	128 / 512 / 0.1
LoRA modules	Attention query, key, value, and output projections

G Extended Experiments

G.1 Information in Residual Flow and Last-Layer States

The same response can receive residual updates that differ in direction under two prompts even when its last-layer states are nearly identical. Keeping these updates in layer order also helps a linear probe distinguish matched from mismatched prompts. We show both on controlled prompt–response pairs.

Paired samples. We construct 6,000 pairs, 2,000 each from reading comprehension, translation, and arithmetic. Each pair contains a prompt–response sample and the same response paired with a mismatched prompt. The two prompts have the same token count, so the response is identical in text, token IDs, and positions. Translation and arithmetic pairs also share the answer type and script. Reading comprehension pairs are built from the TyDiQA GoldP training set, and translation and arithmetic pairs from the LESS pool.

Differences in intermediate updates. For pairs whose last-layer states have cosine similarity between 0.95 and 0.999, residual updates differ in direction more than last-layer states for over 99% of pairs, with median difference ratios of 2.10–3.58 across tasks. Let A denote the matched prompt–response sample and B the sample formed by pairing the same response with a mismatched prompt. At each selected layer $\ell \in \{8, 16, 20, 27, 30, 32\}$, we average the hidden states over response token positions for each sample. We denote these mean hidden states by $H^{\ell,A}$ and $H^{\ell,B}$, and their differences between successive selected layers by $\Delta h^A_{\ell_k}$ and $\Delta h^B_{\ell_k}$. We measure how much two vectors differ in direction by their cosine difference, $1 - \cos(\cdot, \cdot)$. The difference ratio is the mean cosine difference between corresponding residual updates divided by the cosine difference between the last-layer states:

$$\frac{\frac{1}{5}\sum_{k=1}^{5}[1 - \cos(\Delta h^A_{\ell_k}, \Delta h^B_{\ell_k})]}{1 - \cos(H^{32,A}, H^{32,B})}.$$

These measurements use the original 4,096-dimensional states, with the last-layer state taken before final RMS normalization. We exclude pairs whose last-layer difference is below 10^{-3} (i.e., similarity above 0.999) so that the ratios are not dominated by numerical precision.

The median difference ratios are 3.58 for reading comprehension, 2.43 for translation, and 2.10 for arithmetic (Fig. 3). Over all 6,000 pairs, without the similarity restriction, the update difference exceeds the last-layer difference for 87% of pairs. These differences alone do not show that residual updates separate matched from mismatched prompts; we test this next.

Information in corresponding updates. Keeping residual updates in layer order helps distinguish matched from mismatched prompts: among the 10% of pairs per task with the most similar last-layer states, shuffling the updates lowers paired accuracy by 5.83 points. We compare five feature constructions: last-layer states, summed residual updates, residual flow, residual flow with shuffled updates, and multi-layer states. All inputs have 640 dimensions. Multi-layer states combine hidden states from layers 8, 16, 20, 27, 30, and 32 using random projections and concatenation:

$$\begin{aligned}
\text{last-layer states} &= \mathrm{unit}(\Omega_{\mathrm{vec}} H^{\ell_m}),\\
\text{summed updates} &= \mathrm{unit}\Big(\Omega_{\mathrm{vec}} \textstyle\sum_{k=1}^{m} \Delta h_{\ell_k}\Big),\\
\text{residual flow} &= \big[\mathrm{unit}(\Omega_{\mathrm{layer}} \Delta h_{\ell_1}); \ldots; \mathrm{unit}(\Omega_{\mathrm{layer}} \Delta h_{\ell_m})\big],\\
\text{shuffled updates} &= \big[\mathrm{unit}(\Omega_{\mathrm{layer}} \Delta h_{\ell_{\pi(1)}}); \ldots; \mathrm{unit}(\Omega_{\mathrm{layer}} \Delta h_{\ell_{\pi(m)}})\big],\\
\text{multi-layer states} &= \mathrm{unit}\big(\Omega_{\mathrm{cat}}[\mathrm{unit}(\Omega_{\mathrm{layer}} H^{\ell_0}); \ldots; \mathrm{unit}(\Omega_{\mathrm{layer}} H^{\ell_m})]\big).
\end{aligned}$$

We use fixed random projection matrices Ω_{vec}, Ω_{layer}, and Ω_{cat}, with output dimensions of 640, 128, and 640, respectively. For an output dimension d, each matrix entry is sampled independently from $\{-1/\sqrt{d}, +1/\sqrt{d}\}$ with equal probability. The matrices are shared across samples, and Ω_{layer} is also shared across layers. For the shuffled-update control, we draw a separate permutation π of the five update blocks for each pair using a seeded random generator. Both samples in a pair share the same permutation. Comparing residual flow with summed updates tests whether retaining individual updates provides useful information. Comparing ordered and shuffled updates tests whether their correspondence across layers matters. Multi-layer states provide a comparison using hidden states from all selected layers.

We train a linear classifier on each feature and measure how often it assigns a higher score to the matched prompt–response sample than to its mismatched counterpart. Each pair is evaluated with a classifier trained on other pairs. We report results across all pairs and among the most similar 10% of pairs (600 pairs; Tab. 10).

Residual flow and multi-layer states reach similar paired accuracy, and both exceed last-layer states and summed updates on all pairs. Among the most similar 10% of pairs, residual flow exceeds shuffled updates by 5.83 points ($[+3.17, +8.33]$), whereas its 0.83-point gain over last-layer states ($[-1.50, +3.17]$) is not resolved at this sample size.

G.2 Beyond Last-Layer Similarity

On TyDiQA, BiTFlow distinguishes samples that last-layer comparison treats alike, and training on its choices improves TyDiQA F1. We show this with a case (Fig. 4) and a replacement experiment (Tab. 2).

Case (Fig. 4). Samples A and B are close in the last layer but differ in their residual updates. A is selected by TRIM but not by BiTFlow: its Russian translation omits most of the listed vehicles

Table 10: Complete paired-probe results. Paired accuracy is reported in %; AUROC is the secondary metric. Intervals on the most similar 10% of pairs are 95% paired bootstrap intervals over pairs. The bottom block reports residual-flow accuracy minus comparator accuracy, in percentage points, on the most similar 10% of pairs.

	All pairs (6,000)		Most similar 10% (600)		
Input features	**Acc.**	**AUROC**	**Acc.**	**95% CI**	**AUROC**
Last-layer states	97.83	0.944	88.33	[86.00, 90.67]	0.688
Summed updates	97.75	0.943	87.83	[85.33, 90.17]	0.686
Shuffled updates	97.08	0.931	83.33	[80.67, 86.00]	0.702
Residual flow	98.40	0.978	89.17	[86.67, 91.33]	0.795
Multi-layer states	98.57	0.976	90.00	[87.67, 92.17]	0.792
Comparator			**Δ (points)**	**95% CI**	
Shuffled updates			+5.83	[+3.17, +8.33]	
Last-layer states			+0.83	[−1.50, +3.17]	

and adds trucks for slaughter cattle. B is selected by BiTFlow but not by TRIM: asked to write any sentence not in English, it answers with a fluent Russian sentence. Fig. 6 gives their original texts. We compare A and B by the $1 - \cos$ of their mean hidden states over response tokens (Appendix G.1), once for the last-layer states and once for the residual update of each band. Their last-layer $1 - \cos$ is 0.096, close to 0.093 for typical pairs, whereas their update $1 - \cos$ in layers 8–30 is 1.5–2.7 times that of typical pairs. Typical pairs (gray bars, medians) are 75 sampled pairs, each joining a sample selected only by TRIM or RDS+ with its nearest BiTFlow-only sample in the last layer, with last-layer $1 - \cos$ between 0.09 and 0.10.

A Inaccurate translation (In Russian) TRIM ✓ BiTFlow ✗

User: Translate to Russian: The means of transport from French haulers used trucks for sale with photos. Lorry chassis cab, dump trucks, trucks BDF system, curtainsider, car transport other The vehicles equipped with hydraulic loading lift, box, sleeper cabin and tippers with HDS crane. Russian:

Assistant: Грузовые машины французских транспортных фирм - пространственное грузовое средство передвижения, грузовые машины с крюками для транспорта обменных контейнеров и систем BDF, грузовые машины для убойного скота, также с погрузочным гидравлическим лифтом.

Translation: Trucks of French transport companies – a spatial cargo vehicle, trucks with hooks for transporting swap containers and BDF systems, trucks for slaughter cattle, also with a hydraulic loading lift.

B Fluent sentence (In Russian) TRIM ✗ BiTFlow ✓

User: [five in-context examples omitted] input question: Write a sentence not in English. output answer:

Assistant: Все компоненты установки выполнены из высоко износостойких материалов.

Translation: All components of the installation are made of highly wear-resistant materials.

Figure 6: Original texts of samples A and B in Fig. 4, with English translations of the Russian responses. B's five in-context examples are omitted.

Coreset replacement (Tab. 2). Replacing part of a coreset with our selected samples that last-layer scoring ranks alike raises mean TyDiQA F1 for every selector in Tab. 2. For each selector, we replace 459 samples (3.39%) of its selected coreset and keep the rest. Each removed sample is selected by that selector but not by our method, and is paired with an added sample selected by our method but not by that selector. Paired samples rank alike under our method with last-layer states (the Last layer, proj. variant in Tab. 4(a)): both are in the top 10% of the candidate pool, and their ranks differ by at most 0.5 percentile points. Paired samples also come from the same source, and their prompt and response lengths differ by at most a factor of 1.25. We fine-tune on the original and replaced coresets with the same three seeds; for our method with last-layer states, the gain is positive in every seed.

G.3 In-Domain Selection

In in-domain selection, target and candidate samples come from the same task, unlike the main experiments, where the candidate pools mix many tasks. On GSM8K, BiTFlow achieves the highest mean performance (exact match) among all methods except its warmed-up variant at every tested budget (Tab. 11).

We reserve 20 of the 7,473 training samples as a shared target set and exclude them from selection and fine-tuning. These targets are disjoint from the evaluation demonstrations and the official test set. Each method ranks the remaining 7,453 candidates once and selects nested subsets of 75, 373, 746, and 1,118 samples, corresponding to budgets of 1%, 5%, 10%, and 15%.

BiTFlow and RDS+ use the pretrained scorer without warmup training. LESS and TRIM use the same checkpoint, obtained after four epochs of LoRA warmup training on 374 samples disjoint from the targets. We also evaluate BiTFlow with a warmed-up scorer. All selected subsets are used to fine-tune the same pretrained model for four epochs using LoRA, with selections held fixed across training runs. Evaluation uses the 1,319 official test questions, eight chain-of-thought demonstrations, greedy decoding, and exact match.

Table 11: In-domain GSM8K exact match (%), with sample standard deviations in gray. Full data uses all 7,453 candidates. †: scorer warmup. Bold marks the best result among the default selectors at each budget.

Method	**1%**	**5%**	**10%**	**15%**
Full data		15.44 ±1.18		
Random	14.00 ±0.42	16.22 ±0.33	18.25 ±0.90	17.08 ±0.04
LESS†	14.89 ±0.09	18.88 ±0.60	19.16 ±0.57	17.08 ±0.62
RDS+	14.66 ±0.12	17.41 ±0.42	19.38 ±0.77	17.21 ±0.26
TRIM†	14.43 ±0.49	17.56 ±0.37	19.61 ±0.12	17.31 ±0.50
BiTFlow	**15.04** ±0.16	**19.38** ±0.16	**19.74** ±0.23	**17.82** ±0.73
BiTFlow + warmup†	15.21 ±0.18	18.88 ±0.30	19.81 ±0.19	17.44 ±0.80

All compared selectors peak at a 10% budget and decline at 15%, and every 5% selection, including random, outperforms full-data fine-tuning under the same four-epoch schedule. For the score-based selectors, larger budgets extend selection to lower-scoring samples, which may introduce noisier or less relevant supervision. These additions need not improve benchmark performance and may dilute the benefit of the highest-ranked samples.

H Extended Ablation Studies

We compare residual-flow and hidden-state encodings within the same prototype and matching framework, followed by scoring components, choice of layers, and hyperparameter sensitivity. All comparisons use the LESS pool at a 5% budget.

H.1 Encoding Design

With the 128-dimensional projection, both multi-layer state encoding and residual-flow encoding yield higher GSM8K and MMLU means than last-layer state encoding, while the three encodings are within 0.19 points of each other on TyDiQA (Tab. 12). The projection mainly affects TyDiQA, raising its mean by 1.16–1.68 points across the three encodings.

Setup. The state-based encodings replace residual updates with hidden states and keep the rest of BiTFlow unchanged: token-ID prototypes, exact and fallback matching, and balanced score aggregation. Last-layer state encoding uses H^{32} before the final RMSNorm. Multi-layer state encoding uses H^{16}, H^{20}, H^{27}, H^{30}, and H^{32}, the ending layers of the five default bands; it builds token encodings and target prototypes at each layer separately and averages the similarities across layers. Unlike the probe in Appendix G.1, which also uses layer 8 and concatenates randomly projected states, multi-layer states here omit layer 8 so that each token has five input vectors, matching the five band updates of residual flow. The five layers are compared separately within the matching framework. In the projected variants, each layer or band is projected to 128 dimensions with a PCA projection fitted on the same reference token positions as the default. The unprojected variants keep 4,096 dimensions per layer or band and apply L2 normalization. Each variant reuses its selected subset across training runs.

Encoding design. Multi-layer states and residual flow both draw on several layers and both improve over last-layer states on GSM8K and MMLU. Neither is consistently higher: residual flow has the higher GSM8K mean (29.39 vs. 28.78), and multi-layer states have the higher MMLU mean (49.02 vs. 48.92). The two differ in what they compare. Multi-layer states compare the hidden states at each

Table 12: Encoding alternatives on the LESS pool at 5%. Scores are means with sample standard deviations in gray. The projection is fitted with PCA separately for each layer or band. Default reproduces Tab. 1.

Encoding	Projection	MMLU	GSM8K	TyDiQA
With projection				
Last-layer states	128d	47.91 ±0.31	28.51 ±0.69	55.66 ±0.30
Multi-layer states	128d	49.02 ±0.31	28.78 ±0.46	55.59 ±0.71
Residual flow (default)	128d	48.92 ±0.29	29.39 ±0.80	55.47 ±0.76
Without projection				
Last-layer states	none (4,096d)	48.30 ±0.37	28.63 ±0.54	54.45 ±0.56
Multi-layer states	none (4,096d)	48.90 ±0.21	28.89 ±0.53	53.91 ±0.28
Residual flow	none (4,096d)	49.01 ±0.15	29.14 ±0.27	54.31 ±0.94

selected layer, while residual flow compares the updates between consecutive selected layers, so each comparison is made on how tokens change at that depth.

Projection. Without projection, multi-layer states have the lowest TyDiQA mean (53.91) despite using more layers, and the projection raises it by 1.68 points. Across the three encodings, the projection raises TyDiQA by 1.16–1.68 points, while GSM8K and MMLU change by at most 0.39 points in either direction. The projection may remove redundant directions that would otherwise dilute the selection signal. It also reduces each feature vector from 4,096 to 128 dimensions.

H.2 Component Analysis

Each scoring component of BiTFlow contributes to its performance (Tab. 13). Exact-only matching and all four alternative aggregation rules score below the default on all three tasks, and one-to-one matching outperforms many-to-one matching at the same matching cost. Tab. 13 compares the default with 14 variants. All variants share one forward pass that extracts the required layers and use the same per-task targets. The layer-range rows are discussed in Appendix H.3.

Table 13: Component comparisons on the LESS pool at 5%. Default reproduces Tab. 1. Gray values are sample standard deviations. Scores are percentages (higher is better). The two $\tau = 0$ rows compare matching rules at the same cost. TRIM-style denotes the aggregation rule applied to our token evidence.

Configuration	MMLU	GSM8K	TyDiQA
Layer ranges			
Layers 8 and 32 only	48.14 ±0.04	28.56 ±0.31	55.49 ±0.25
Layers 8, 16, 24, and 32	49.25 ±0.47	28.28 ±0.38	54.61 ±0.53
Five overlapping layer pairs	48.73 ±0.10	28.43 ±0.35	55.78 ±0.56
Early layers (8, 12)	48.81 ±0.26	28.89 ±0.53	55.88 ±0.32
Middle layers (18, 22)	48.62 ±0.14	28.89 ±0.27	55.84 ±0.26
Late layers (28, 32)	48.55 ±0.12	28.05 ±0.61	55.41 ±0.31
Matching			
Exact matching only	47.56 ±0.23	28.91 ±1.08	54.05 ±0.16
Many-to-one fallback, $\tau = 0$	48.98 ±0.06	28.71 ±0.31	55.28 ±0.29
One-to-one, $\tau = 0$	49.47 ±0.24	28.96 ±0.77	55.83 ±0.30
One-to-one, $\tau = 0.10$	49.01 ±0.46	30.20 ±0.61	54.72 ±0.42
Aggregation over token evidence			
Mean	48.82 ±0.36	25.52 ±1.02	53.10 ±0.09
Max	47.70 ±0.19	28.08 ±0.54	54.06 ±0.40
Mean–max	48.51 ±0.42	26.64 ±0.61	54.53 ±0.19
Mean–max + coverage (TRIM-style)	48.90 ±0.30	26.23 ±1.20	54.48 ±0.08
Default (Tab. 1)	48.92 ±0.29	29.39 ±0.80	55.47 ±0.76

Matching. Matching beyond shared token IDs helps, and so does limiting each target ID to one match. Exact-only matching scores below the default on all three tasks. At the same matching cost $\tau = 0$, one-to-one bipartite matching outperforms many-to-one fallback on all three tasks. Many-to-one matching lets several candidate token IDs reuse the same target evidence, whereas bipartite matching restricts each target ID to one match. The $\tau = 0.10$ variant gives the highest GSM8K mean in the table.

Prompt and response roles. Scoring both roles helps: prompt-only ($\alpha = 1$) and response-only ($\alpha = 0$) scoring fall below the default on all three tasks (Tab. 15).

Score aggregation. Balanced aggregation, which accounts jointly for candidate support and target coverage, outperforms the four alternatives on all three tasks. The alternatives summarize the same token evidence with the mean, the max, the mean–max, or the mean–max with a TRIM-style coverage term. The smallest margin is 0.02 points on MMLU over the TRIM-style rule.

H.3 Choice of Layers

The default five bands over layers 8–32 are not outperformed consistently by other layer choices: no alternative layer range or band design is better on all three tasks, and individual block updates score below the default on all three tasks.

Layer ranges. No alternative layer range or band design outperforms the default consistently across the three tasks (Tab. 13). We compare one, three, and five bands over layers 8–32; early, middle, and late ranges of four blocks each; and five overlapping ranges of eight blocks.

Individual vs. band updates. Both individual-update configurations score below the five-band default on all three tasks (Tab. 14), and bands also reduce the number of vectors stored and compared per token from 24 or 32 to 5. The individual-update alternatives keep the 24 block updates over layers 8–32, with boundaries $(8, 9), \ldots, (31, 32)$, or all 32 block updates, with boundaries $(0, 1), \ldots, (31, 32)$, where H^0 is the input embedding and H^{32} is the last decoder-block output before the final RMSNorm. All configurations use the same pretrained Llama-2-7B scorer, reference token positions, and scoring parameters ($\tau = 0.05$, $J = 5$, and $\alpha = 0.5$). Each update vector is projected to 128 dimensions with a PCA projection fitted separately for each band.

Table 14: Individual vs. band updates on the LESS pool at 5%. Default reproduces Tab. 1. Scores are means with sample standard deviations over three training seeds. Scores are percentages.

Updates	Vectors /token	MMLU	GSM8K	TyDiQA
Individual, 8–32	24	48.42 ±0.15	27.93 ±0.34	54.98 ±1.03
Individual, 0–32	32	48.17 ±0.66	28.35 ±0.93	54.67 ±0.41
Bands (default)	5	48.92 ±0.29	29.39 ±0.80	55.47 ±0.76

H.4 Hyperparameter Sensitivity

Performance is more sensitive to the prompt weight α than to the matching cost τ (Tab. 15). We vary $\tau \in \{0, 0.01, 0.025, 0.05, 0.075, 0.10\}$ at $\alpha = 0.5$ and $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$ at $\tau = 0.05$ under the same 5% training protocol. Each configuration keeps its selection and sample order fixed across training seeds 3, 6, and 9. Within each seed, we average the three task scores and subtract the default's average from the same seed; $\Delta\mathrm{Avg.}_3$ is the mean of these differences.

Table 15: Hyperparameter sensitivity on the LESS pool at 5%. Scores are means over training seeds 3, 6, and 9, with sample standard deviations in gray. $\Delta\mathrm{Avg.}_3$ is the mean within-seed change in the three-task average relative to the default ($\tau = 0.05$, $\alpha = 0.5$), in percentage points. Each axis varies one parameter while fixing the other.

Configuration	τ	α	MMLU	GSM8K	TyDiQA	$\Delta\mathrm{Avg.}_3$
Matching cost						
	0	0.5	49.47 ±0.24	28.96 ±0.77	55.83 ±0.30	+0.16
	0.01	0.5	49.44 ±0.22	28.76 ±0.18	55.83 ±0.42	+0.08
	0.025	0.5	49.24 ±0.32	29.26 ±0.93	56.03 ±0.59	+0.25
	0.075	0.5	49.05 ±0.15	28.05 ±0.15	54.59 ±0.72	−0.70
	0.10	0.5	49.01 ±0.46	30.20 ±0.61	54.72 ±0.42	+0.05
Prompt weight						
	0.05	0	48.63 ±0.35	28.89 ±0.38	52.21 ±0.35	−1.35
	0.05	0.25	48.49 ±0.29	29.24 ±0.56	55.26 ±0.40	−0.26
	0.05	0.75	49.73 ±0.31	26.54 ±0.62	54.71 ±0.48	−0.93
	0.05	1	48.80 ±0.11	24.34 ±0.99	54.63 ±0.68	−2.00
Default	0.05	0.5	48.92 ±0.29	29.39 ±0.80	55.47 ±0.76	0.00

Low matching costs ($\tau \leq 0.05$) yield similar three-task averages. On the prompt-weight axis, MMLU peaks at $\alpha = 0.75$, whereas GSM8K and TyDiQA peak at $\alpha = 0.5$. Prompt-only scoring ($\alpha = 1$)

lowers GSM8K most, and response-only scoring ($\alpha = 0$) lowers TyDiQA most; MMLU is less affected in both cases. These results support combining both roles, with their relative contributions depending on the task.

I Time Complexity Analysis and Selection Cost

I.1 Computational Complexity

Model computation. We expand the selection costs in Tab. 5. Let f_{fw} and f_{bw} denote the per-sample forward- and backward-pass costs. For a candidate pool of N samples,

$$F = N f_{\text{fw}}, \qquad G = N(f_{\text{fw}} + f_{\text{bw}}).$$

Warmup training on a fraction ω of the candidate pool for n_e epochs contributes

$$W = \omega n_e G.$$

RDS+, CRDS-W, and BiTFlow need one forward pass per candidate, costing F; BiTFlow extracts all selected-layer states in this pass, without warmup training. TRIM additionally incurs W, while the LESS method incurs $W + n_c G$ for gradient extraction at n_c checkpoints. For S2L, F^{proxy} and G^{proxy} denote the forward and forward–backward costs of the proxy model, with n_e training epochs and n_c scored checkpoints. All remaining preparation, feature-processing, and selection costs are included in A_{method}.

BiTFlow preparation and scoring. Let C_{prep} denote the preparation cost for residual-flow encoding, target prototype construction, and token matching. The reference used to fit the projection has a fixed budget of n_r token positions.

For candidate x, let t_x be the number of processed token positions and M_x the cost of constructing and solving its fallback matching problem. With target shortlists prepared in advance, the processing cost per candidate beyond the forward pass is

$$c_x = \mathcal{O}(t_x m d_h d_z + t_x m J d_z + M_x)\,. \tag{19}$$

The three terms correspond to encoding residual updates, comparing candidate tokens with target prototypes, and bipartite matching.

Total selection cost. Final top-K selection can be performed with a size-K heap, giving

$$A_{\text{RF}} = \mathcal{O}\left(C_{\text{prep}} + \sum_{x \in \mathcal{D}_{\text{pool}}} c_x + N \log(K+1)\right).$$

The complete selection cost is therefore

$$\mathcal{O}(F + A_{\text{RF}}).$$

With all other settings fixed, candidate scoring scales linearly with the candidate pool size N.

I.2 Runtime Comparison

Experimental setup. We measure candidate-scoring time on all LESS-pool candidates, running every method on eight NVIDIA H100 GPUs. Methods that use target samples share the same ten GSM8K samples. Timing covers feature extraction and scoring; preparation, such as reference fitting or scorer warmup, and final subset selection are excluded.

Table 16: Selection complexity and candidate-scoring time on the LESS pool. Time is reported in GPU milliseconds per sample and includes feature extraction and scoring, excluding preparation and final subset selection. Relative time is normalized to TRIM.

Method	Total selection cost	Scoring time (GPU-ms/sample)	Relative to TRIM
CRDS-W	$\mathcal{O}(F + A_{\text{CRDS-W}})$	27.00	0.93×
RDS+	$\mathcal{O}(F + A_{\text{RDS+}})$	28.59	0.98×
TRIM	$\mathcal{O}(W + F + A_{\text{TRIM}})$	29.07	1.00×
S2L	$\mathcal{O}(n_e G^{\text{proxy}} + n_c F^{\text{proxy}} + A_{\text{S2L}})$	34.55	1.19×
LESS	$\mathcal{O}(W + n_c G + A_{\text{LESS}})$	547.98	18.85×
BiTFlow	$\mathcal{O}(F + A_{\text{RF}})$	30.03	1.03×

We report GPU milliseconds per sample, calculated as the measured scoring time in milliseconds multiplied by the allocated GPU count and divided by the number of candidates. Tab. 16 reports these times alongside the total selection cost defined in Appendix I.1.

GPU implementation. Our initial implementation performs residual-feature processing and matching on the CPU after the model forward pass. The GPU implementation keeps hidden states, residual updates, projected features, and similarity computations on the GPU, avoiding transfers of high-dimensional candidate features to the CPU. The small bipartite matching problems remain on the CPU. Only where the computation runs changes; the scoring rule is unchanged. A consistency check on 256 candidates confirms that the two implementations produce identical rankings and scores that agree to numerical precision.

J CASE STUDY

We illustrate the difference between last-layer similarity and intermediate updates using an arithmetic pair from the dataset in Appendix G.1. Both prompts ask how long a train moving at 45 km/hr takes to pass a bridge, but specify different train and bridge lengths. We attach the same response to both prompts: its calculation correctly answers Prompt A but does not solve Prompt B. Fig. 7 gives both prompts and the shared response.

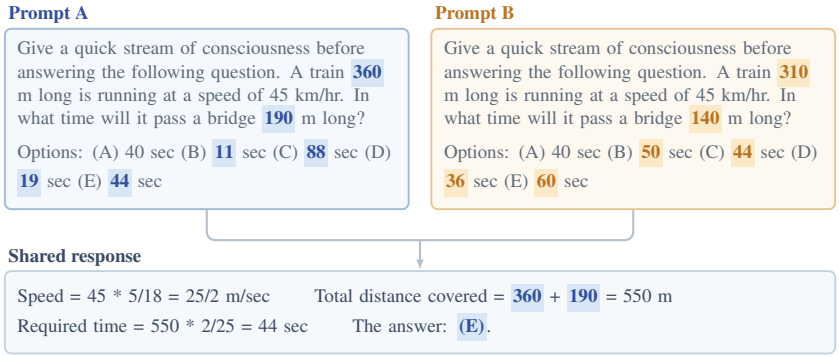

Figure 7: An arithmetic pair illustrating last-layer similarity collapse. Both prompts share a response that answers Prompt A; highlights mark where the prompts differ.

The pair has a last-layer cosine similarity of 0.99887, while its update difference is 3.67 times its last-layer difference. The last-layer states are nearly identical even though the response answers only Prompt A, whereas the intermediate updates differ more. This pair is one instance of the pattern measured in Appendix G.1.