LET TRAINING GUIDE SELECTION: ONLINE SYNTHETIC DATA FILTERING VIA REAL-ANCHORED UTILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthetic data can scale training supervision when real-world data are limited, but noise and distribution mismatch can reduce its value. Existing synthetic data selection methods often emphasize fidelity or diversity rather than the learner's evolving needs. We propose FROST, an online framework that estimates synthetic-data utility through gradient feedback anchored in real training data. It calibrates batch utility against recent history to determine *when* filtering is needed and filters samples only in out-of-band batches to determine *what* to retain, without an external verifier or held-out validation set. Experiments on two public benchmarks for image classification and LLM fine-tuning for text-to-SQL show that FROST filters out around 20–30% of the synthetic data while improving real-task performance compared with training on the full synthetic data pool. We further apply FROST during training in a large-scale industrial ads re-ranking system, achieving significant performance gains over a highly optimized production baseline, demonstrating its effectiveness and generalizability.

1 INTRODUCTION

As modern deep learning models scale, their growing demand for training data makes the availability of real-world data an increasingly important bottleneck (Villalobos et al., 2024). Synthetic data provides a scalable alternative when collecting or annotating additional real examples is costly or constrained. In computer vision, generated images augment supervision when data access or expert annotation is limited (Frid-Adar et al., 2018); in large language models (LLMs), synthetic text and instruction–response pairs support pretraining and instruction tuning (Gunasekar et al., 2023; Wang et al., 2023). Synthetic data is also increasingly used in recommendation systems, where sparse interactions, incomplete user information, and privacy constraints limit available supervision. For example, LLMRec and DALLRec use LLMs to infer additional user–item interactions and enrich user or item descriptions for recommender training (Wei et al., 2024; Mao et al., 2025). SCALR generates synthetic target-domain interactions from observed user behavior in other domains and combines them with real interactions to train downstream recommenders (Wang et al., 2026).

Despite advances in generative quality, synthetic data can still contain semantic errors and artifacts, miss important variations, or deviate from the target distribution (Geng et al., 2024; Adamkiewicz et al., 2026). More importantly, these failures are not always identifiable from individual samples alone. In image recognition, synthetic data can scale less effectively than real data (Fan et al., 2024), and even images from more advanced generators can yield lower real-test classification accuracy due to distributional mismatch with real data (Adamkiewicz et al., 2026). In recommendation systems, generated user–item interactions and attributes can likewise introduce uncertain or noisy supervision, since synthetic interactions often lack real user feedback for verification (Wei et al., 2024; Mao et al., 2025; Wang et al., 2026). This raises a central challenge: *how can we select synthetic data that provides useful supervision and improves model performance on real data?*

Existing synthetic-data selection methods often emphasize *fidelity* and *diversity* of the synthetic data. They use pretrained feature extractors to assess image–label consistency (He et al., 2023), select from feature clusters that are close to real examples, or match synthetic and real feature statistics (Hulkund et al., 2025; Rezaei et al., 2026). These criteria characterize properties of the data, but

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.

do not track the evolving needs of the model being trained. Moreover, extracting features and preprocessing the full candidate pool can also be resource-intensive at production scale. Instead, we estimate synthetic-data utility from training dynamics by aligning each synthetic sample's gradient with a smoothed real-training gradient reference. This provides a local signal of whether a synthetic update supports the real-data objective. Existing methods also employ training dynamics, but for data-efficient subset selection on real training datasets (Qin et al., 2024; Jin et al., 2026), with some using held-out target examples as the selection reference (Xia et al., 2024; Wang et al., 2024).

In this paper, we introduce FROST (**F**iltering with **R**eal data for **O**nline **S**ynthetic-data **T**raining), an online synthetic-data filtering framework built from two key ideas: (1) estimating synthetic-data utility through gradient feedback anchored in real training data, and (2) conditioning sample-level filtering on batch-level utility. For utility estimation, we use gradients from the model being trained, without a separate feature extractor or offline preprocessing of the synthetic pool for selection. Specifically, we maintain an exponential moving average of gradients from real training batches and score each synthetic sample by its gradient alignment with this reference. Individual utility estimates are local and affected by stochastic gradient noise (Faghri et al., 2020), and filtering every batch based on these estimates can discard useful supervision. We therefore aggregate sample utilities and calibrate the batch mean against recent training history to determine *when* sample-level filtering is needed. We retain batches within a moderate utility band and apply sample-level filtering using interquartile-range (IQR) thresholds only to batches outside the band. Sample-level filtering can thus retain useful examples from batches that would otherwise be rejected entirely.

We first evaluate FROST on two public benchmarks for image classification and LLM fine-tuning for text-to-SQL, comparing against existing selection methods at matched synthetic-data drop ratios. FROST outperforms these baselines and improves absolute accuracy by 0.58–1.14% across 3 synthetic image generators and 0.9% on text-to-SQL over training with the full synthetic pools, but with 20–30% fewer synthetic training samples. We further apply FROST to the training of a large-scale industrial ads re-ranking system, where it turns a 0.211% Normalized Entropy (NE) regression from training with the full noisy synthetic pool into a 0.096% improvement, compared to real-only training reference. Our 0.096% relative NE improvement is well above the 0.02% gain which is considered significant in prior studies of well-optimized production recommenders (Li et al., 2022; Lai et al., 2023). Our contributions are threefold:

- We propose FROST, a model- and task-agnostic framework for online synthetic-data filtering that estimates utility through gradient feedback anchored in real training data and uses conditional batch-to-sample filtering to select useful synthetic supervision.
- We demonstrate the effectiveness and generalizability of FROST across image classification and LLM fine-tuning, including 3 synthetic image generators, improving real-task performance with fewer synthetic training samples.
- We achieve significant performance gains over a highly optimized production baseline in a large-scale industrial ads re-ranking system, highlighting the practical impact of FROST.

2 Related Work

Synthetic data for deep learning. Synthetic data has been explored across many deep learning tasks to enrich training supervision when real data are scarce or costly to annotate. In computer vision, text-to-image diffusion models turn class-specific text descriptions into training images (Rombach et al., 2022; He et al., 2023). Shipard et al. (2023) improve generation diversity to train classifiers without real images, while Azizi et al. (2023) use generated images to augment real-data training. For LLMs, synthetic text and instruction–response pairs support pretraining and instruction tuning with less reliance on human-written examples (Gunasekar et al., 2023; Wang et al., 2023). OmniSQL introduces SynSQL-2.5M by automatically generating diverse databases, questions, and SQL queries, expanding text-to-SQL training coverage without large-scale manual annotation (Li et al., 2025). In recommendation, the motivation is to enrich sparse interactions and incomplete side information. LLMRec and DALLRec generate additional interactions and item information (Wei et al., 2024; Mao et al., 2025), while SCALR transfers cross-domain behavior into synthetic target-domain events (Wang et al., 2026). These synthetic data generation approaches expand the available supervision; our goal is to select useful synthetic data after generation.

Synthetic data selection. Existing synthetic-data selection methods focus on fidelity and diversity through semantic filtering, feature-space coverage, or distribution matching. Pretrained feature extractors are used to assess similarity to class descriptions or real examples (He et al., 2023; Lin et al., 2023). DS3 selects from synthetic feature clusters close to real examples, while CovMatch greedily matches the selected subset's feature covariance to that of real data (Hulkund et al., 2025; Rezaei et al., 2026). These methods typically select data before downstream training, requiring feature extraction and preprocessing of the candidate pool. More fundamentally, as the model evolves during training, the usefulness of synthetic data can change, while their offline feature-based selection scores remain fixed. FROST instead estimates utility online through gradient alignment with real training data, providing a local estimate of whether a synthetic update would reduce the current real-data loss.

Data selection based on training dynamics. Training-dynamics-based methods use losses and gradients to identify informative training subsets. GradNorm prioritizes large-gradient examples (Katharopoulos & Fleuret, 2018; Wang et al., 2024), while GraNd and EL2N use early-training scores for dataset pruning (Paul et al., 2021). GRAD-MATCH selects weighted coresets that approximate training or validation gradients, aiming to preserve performance with fewer examples (Killamsetty et al., 2021). For LLMs, LESS selects instruction data offline (Xia et al., 2024), while GREATS performs online subset selection (Wang et al., 2024). Both use held-out target examples as the selection reference. FROST instead focuses on filtering synthetic supervision during mixed real/synthetic training. It retains all real data and estimates synthetic utility using a smoothed real-training gradient reference, with history-calibrated batch utility determining when sample-level filtering is needed.

3 METHOD

Problem setup. We have access to a real training dataset $\mathcal{D}_R$ and a synthetic dataset $\mathcal{D}_S$. At training step t, we draw a real mini-batch $\mathcal{R}_t \subseteq \mathcal{D}_R$ and a synthetic mini-batch $S_t = \{s_{t1}, \ldots, s_{tm}\} \subseteq \mathcal{D}_S$. Given the current model f_{θ_t}, we select $\widetilde{S}_t \subseteq S_t$ to train alongside $\mathcal{R}_t$. Our goal is to reduce the expected loss on real test data drawn from the target distribution P_R:

$$\mathcal{L}_R(\theta) = \mathbb{E}_{r\sim P_R}[\ell(r;\theta)]\,. \tag{1}$$

Here, $\ell(r;\theta)$ is the per-example loss. Selection uses the current model, available training data, and past training statistics, without data quality labels or a separate held-out validation set.

Overview. Figure 1 presents three components of FROST. ① We estimate each synthetic sample's utility through gradient alignment with a smoothed real-training reference. ② We aggregate these scores and calibrate batch utility against recent training history, retaining batches within a moderate utility band. ③ For out-of-band batches, we apply sample-level filtering based on the interquartile range (IQR). We describe these three components in detail in Sections 3.1–3.3.

3.1 ESTIMATING SYNTHETIC SAMPLE UTILITY

Inspired by TracIn (Pruthi et al., 2020), we estimate a synthetic sample's utility by asking how an update with that sample would change the current model's real-data loss. Let $\theta_t^{(i)}$ denote the parameters after a hypothetical update with s_{ti}, and write $d_{i,t} = \theta_t^{(i)} - \theta_t$. For a locally smooth real-data loss, a Taylor expansion around θ_t expresses the loss reduction as

$$\begin{aligned}\Delta_{i,t} &= \mathcal{L}_R(\theta_t) - \mathcal{L}_R(\theta_t^{(i)}) \\ &= -\nabla_\theta \mathcal{L}_R(\theta_t)^\top d_{i,t} + O\big(\|d_{i,t}\|^2\big)\,.\end{aligned} \tag{2}$$

For a hypothetical SGD step of size $\eta > 0$, $d_{i,t} = -\eta \nabla_\theta \ell(s_{ti};\theta_t)$. Substituting this update into Equation 2 gives

$$\Delta_{i,t} = \eta \nabla_\theta \mathcal{L}_R(\theta_t)^\top \nabla_\theta \ell(s_{ti};\theta_t) + O\big(\eta^2 \|\nabla_\theta \ell(s_{ti};\theta_t)\|^2\big)\,. \tag{3}$$

Thus, positive alignment between the synthetic gradient and the target real gradient predicts a first-order reduction in real-data loss.

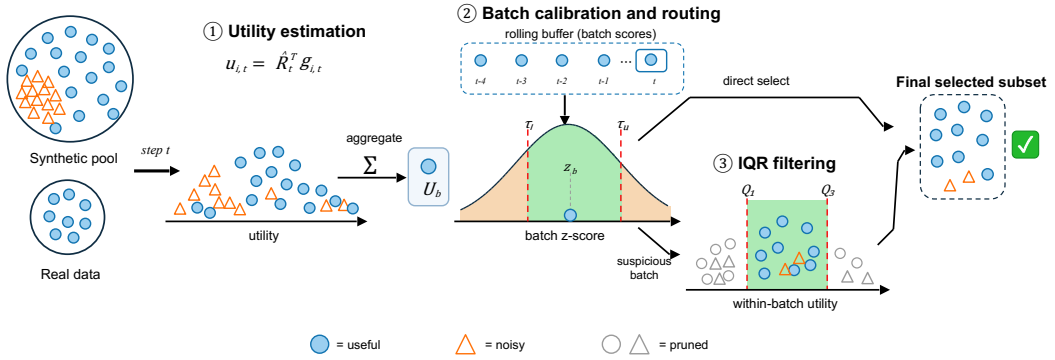


Figure 1: Overview of FROST. The framework ① estimates each synthetic sample's utility through gradient alignment with a smoothed real-training reference, ② aggregates these scores and calibrates batch utility against recent training history to retain moderate-utility batches directly, and ③ applies IQR-based sample filtering to out-of-band batches.

A reference from real training data. Measuring a real-test loss change for every candidate would require repeated model updates and access to test labels. Instead, we estimate the target gradient using the current real training mini-batch $\mathcal{R}_t$. Let ϕ be the parameter block of θ used for scoring, with $\phi = \theta$ corresponding to the full-model derivation above. Define $g_t(x) = \nabla_\phi \ell(x; \theta_t)$ and the synthetic sample gradient $g_{i,t} = g_t(s_{ti})$. The real reference is

$$R_t = \frac{1}{|\mathcal{R}_t|} \sum_{r \in \mathcal{R}_t} g_t(r), \tag{4}$$

which estimates the target gradient in the same parameter block. This lets us estimate local alignment through $R_t^\top g_{i,t}$ without separately updating the model for each candidate.

Smoothing the reference. A single real mini-batch can give a noisy reference. We smooth it across steps using a bias-corrected EMA:

$$M_t = \beta M_{t-1} + (1-\beta) R_t, \qquad \hat{R}_t = \frac{M_t}{1-\beta^t}, \qquad M_0 = 0, \quad \beta = 0.99, \tag{5}$$

where t counts reference updates. Appendix A.2 bounds the EMA tracking error in terms of mini-batch noise and parameter drift. The sample score is the unnormalized inner product

$$u_{i,t} = \hat{R}_t^\top g_{i,t}. \tag{6}$$

In practice, we approximate full-model gradient alignment using a selected parameter block: ϕ is the prediction head for image models and LLMs, and the final layer of the shared backbone for the deep learning recommendation model used in ads re-ranking. The resulting score is a proxy for the effect of a full-model update. Computational overhead is analyzed in Appendix B.3.

3.2 Batch-level calibration and routing

Using the estimated sample utilities, we first assess the synthetic batch as a whole. We calibrate its aggregate utility and use a bounded utility band to decide whether to apply sample-level filtering, rather than whether to discard the entire batch.

Batch aggregation and calibration. Batch averaging provides a more stable scale for deciding when to apply sample-level filtering. Define

$$\bar{g}_{S,t} = \frac{1}{m} \sum_{i=1}^{m} g_{i,t}, \qquad U_t = \frac{1}{m} \sum_{i=1}^{m} u_{i,t} = \hat{R}_t^\top \bar{g}_{S,t}. \tag{7}$$

Conditioned on a fixed model and reference, suppose candidate utilities are i.i.d. with mean μ_t and finite, nonzero variance σ_t^2. The central limit theorem (CLT) gives

$$\frac{\sqrt{m}\,(U_t - \mu_t)}{\sigma_t} \xrightarrow[m\to\infty]{d} \mathcal{N}(0,1), \qquad \mathrm{Var}(U_t) = \frac{\sigma_t^2}{m}. \tag{8}$$

Consequently, for sufficiently large batches, the batch mean admits a normal approximation without assuming Gaussian sample-level scores. When the model and reference change slowly over a short training window, recent batch utilities provide a local distribution for calibration. We keep up to W preceding batch scores in a rolling buffer $\mathcal{B}_t$. Its empirical mean and standard deviation define the current batch's z_t-score:

$$\hat{\mu}_t = \frac{1}{|\mathcal{B}_t|} \sum_{v \in \mathcal{B}_t} v, \qquad \hat{\sigma}_t^2 = \frac{1}{|\mathcal{B}_t|} \sum_{v \in \mathcal{B}_t} (v - \hat{\mu}_t)^2, \qquad z_t = \frac{U_t - \hat{\mu}_t}{\hat{\sigma}_t + \epsilon}, \tag{9}$$

where $\epsilon = 10^{-8}$. The buffer estimates the local distribution of *batch*-level utilities. During filtering, the current U_t enters the buffer only after its routing decision.

Initialization. During the first approximately 5% of training steps, only real data update the model; we update the real-gradient EMA and score synthetic batches to populate the buffer. Mixed real/synthetic training and filtering begin after warmup, using the available buffer entries.

Bounded utility and batch routing. Batch utility U_t provides a first-order estimate of how a synthetic update affects the real-data objective. Reference estimation error and higher-order effects can change the realized loss reduction, so the score indicates local update alignment rather than guaranteeing training benefit. Appendix A.1 gives the decomposition and error bound. Our empirical study in Section 4.1 also shows that the high-utility band yields a higher early performance peak but fails to sustain this advantage, leaving lower mean performance than the moderate band. The moderate band provides a better balance between short-term performance and long-term stability. Thus, we use a bounded interval of the calibrated score z_t as a trust region for direct acceptance of moderate-utility batches. The lower bound flags relatively weak aggregate utility. Finally, a batch is retained in full when

$$\tau_l \le z_t \le \tau_u, \qquad \tau_l < \tau_u. \tag{10}$$

The band is moderate relative to recent batches. Batches outside either bound are routed to sample-level filtering.

3.3 Sample-level IQR filtering

For batches routed outside the utility band, we refine the decision at the sample level: an out-of-band average alone does not identify which samples to remove. We therefore inspect their within-batch utilities. Unlike batch averages, individual scores do not inherit a Gaussian approximation from the CLT, so we use empirical quartiles to set robust filtering thresholds. We use IQR-scaled thresholds based on Tukey's fences (Tukey, 1977), with separate lower and upper multipliers. Let $Q_{1,t}$ and $Q_{3,t}$ be the 25th and 75th percentiles of $\{u_{i,t}\}_{i=1}^m$, and let $I_t = Q_{3,t} - Q_{1,t}$ be their IQR. Define

$$a_t = Q_{1,t} - \lambda_l I_t, \qquad b_t = Q_{3,t} + \lambda_u I_t, \qquad \lambda_u \ge 0, \quad a_t \le b_t. \tag{11}$$

A negative λ_l tightens the lower cutoff to $Q_{1,t} + |\lambda_l| I_t$. These thresholds scale with the central spread rather than the magnitudes of extreme scores. The complete selection rule is

$$\widetilde{S}_t = \begin{cases} S_t, & \tau_l \le z_t \le \tau_u, \\ \{s_{ti} \in S_t : a_t \le u_{i,t} \le b_t\}, & \text{otherwise.} \end{cases} \tag{12}$$

The retained samples train alongside $\mathcal{R}_t$.

4 Experiments

We evaluate FROST on image classification, LLM fine-tuning for text-to-SQL, and industrial ads re-ranking. We compare real-task performance with existing selection methods and examine the two-stage design through ablations and a batch-utility analysis.

The image and LLM experiments use public datasets and share the same batch-gate bounds $(\tau_l, \tau_u) = (-0.025, 1)$ and IQR multipliers $(\lambda_l, \lambda_u) = (0, 1.5)$, without domain- and model-specific tuning of these parameters. The synthetic-data drop ratio is the percentage of synthetic data filtered out from training. Real-only and unfiltered all synthetic data training correspond to drop ratios of 100% and 0%, respectively.

4.1 IMAGE CLASSIFICATION

Data and model. We first evaluate FROST on CIFAR-100 image classification (Krizhevsky, 2009) using ResNet-18 (He et al., 2016). CIFAR-100 contains 100 classes with 500 images per class, and we evaluate accuracy on 10,000 real test images. We use FLUX.1 [schnell] (Black Forest Labs, 2024), SANA (Xie et al., 2025), and Stable Diffusion v1.4 (SD1.4) (Rombach et al., 2022) to generate synthetic images for training. For FLUX.1 [schnell] and SANA, we generate 2,000 synthetic images per class. For SD1.4, we directly use the dataset released by Shipard et al. (2023) which contains 1,800 images per class. We combine each synthetic pool separately with the real training data to evaluate the effectiveness of FROST across different synthetic data sources. Training and synthetic-data generation details are provided in Appendix B.1.

Baselines. We compare FROST with *Random*, which uniformly downsamples the synthetic pool, and two groups of selection methods. For dynamic data selection, *InfoBatch* randomly prunes a portion of low-loss examples and rescales retained gradients to correct sampling bias (Qin et al., 2024). *OrderDP* retains examples with the highest loss-based scores from randomly sampled candidate subsets (Jin et al., 2026). For synthetic-data selection, we use two feature-based methods as baselines: *DS3* samples from synthetic feature clusters nearest to real examples, while *CovMatch* greedily selects examples to match the subset's feature covariance to that of real data (Hulkund et al., 2025; Rezaei et al., 2026). We also include *Real only* and *Real + All Synthetic* as references. All methods retain the real training data, and selection baselines match FROST's synthetic-data drop ratio for each generator.

Selection results across generators. As shown in Table 1, FROST outperforms both dynamic data selection baselines and methods tailored to synthetic-data selection across all three generators at matched drop ratios. The dynamic baselines *InfoBatch* and *OrderDP* target training-data efficiency, whereas *DS3* and *CovMatch* emphasize feature-space coverage or matching real-data statistics. FROST instead filters synthetic data using a utility signal tied to the real-data training objective to retain more beneficial supervision. Compared with full synthetic-pool training, absolute accuracy improves by 0.58%, 0.65%, and 1.14% on FLUX, SANA, and SD1.4, respectively.

Table 1: CIFAR-100 classification with synthetic training data. Δ is the accuracy change relative to Real + All Synthetic for the same generator.

	FLUX			SANA			SD1.4		
Method	*Drop*	Acc. ↑	Δ ↑	*Drop*	Acc. ↑	Δ ↑	*Drop*	Acc. ↑	Δ ↑
Real only	*100*	76.04	–	*100*	76.04	–	*100*	76.04	–
Real + All Synthetic	*0*	80.31	0.00	*0*	79.32	0.00	*0*	79.46	0.00
Random	*25.76*	80.33	+0.02	*26.01*	79.61	+0.29	*24.58*	79.34	−0.12
DS3	*25.76*	80.01	−0.30	*26.01*	79.77	+0.45	*24.58*	79.44	−0.02
Covariance Matching	*25.76*	79.92	−0.39	*26.01*	79.21	−0.11	*24.58*	79.21	−0.25
InfoBatch	*25.76*	79.84	−0.47	*26.01*	78.94	−0.38	*24.58*	79.24	−0.22
OrderDP	*25.76*	80.36	+0.05	*26.01*	79.71	+0.39	*24.58*	79.66	+0.20
FROST (ours)	*25.76*	**80.89**	**+0.58**	*26.01*	**79.97**	**+0.65**	*24.58*	**80.60**	**+1.14**

Effect of batch utility. We examine whether higher batch utility leads to steadier training and better classification accuracy. We first train the model for 20 epochs. Holding the checkpoint and all other state variables fixed, we sample synthetic batches from Low ($z < -0.025$), Mid ($-0.025 \leq z \leq 1$), and High ($z > 1$) utility bands. Each branch starts from the same checkpoint and optimizer state, with 200 real and 800 synthetic images per step. We compare the first 30 continuation steps over 3 repeats. Figure 2 shows that High reaches a higher early accuracy peak but exhibits larger fluctuations in its mean accuracy curve. Mid is steadier and finishes this window with higher mean accu-

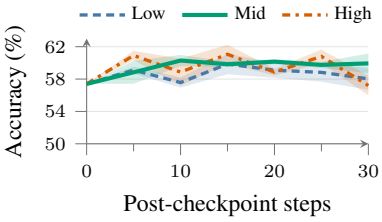


Figure 2: Test accuracy over following 30 steps.

racy (59.94% versus 57.20% for High at step 30). Thus, a larger current utility does not necessarily yield steadier or better training results.

4.2 LLM FINE-TUNING FOR TEXT-TO-SQL

Data and evaluation. We next apply FROST to LLM fine-tuning for the text-to-SQL task. Our real training data consist of 8,659 human-annotated question–SQL pairs from Spider 1.0 (Yu et al., 2018). Our synthetic data come from SynSQL, which contains automatically generated question–SQL pairs over synthetic databases (Li et al., 2025). We sample a subset of 34,636 examples from SynSQL as our synthetic training pool, 4 times the size of the real training set, matching the 4:1 ratio in our FLUX and SANA image settings. We evaluate on Spider 1.0 dev and report execution accuracy (EX), the fraction of predicted SQL queries whose execution results match those of the gold queries, alongside the synthetic-data drop ratio.

Model. We fully fine-tune Qwen2.5-Coder-0.5B-Instruct (Hui et al., 2024) for one epoch with a learning rate of 5×10^{-5} and a global batch size of 64. Utility scores use gradient alignment at the LM output head against the real-gradient EMA reference. Training and selection implementation details are provided in Appendix B.2.

Baselines. We compare FROST with *Random* downsampling, *GradNorm*, which prioritizes examples with the largest per-example gradient norms, motivated by gradient-norm-based importance sampling (Katharopoulos & Fleuret, 2018), and *GREATS* (Wang et al., 2024), which uses a Taylor approximation to select examples that reduce a reference loss. For a fair comparison, our GREATS implementation uses real-training gradients as the reference instead of held-out validation gradients. We also include *Real only* and *Real + All Synthetic* to measure performance without augmentation and with the unfiltered SynSQL pool, respectively. Random, GradNorm, and GREATS use the same 20.4% drop ratio as FROST to match the synthetic-data budget.

Synthetic-data selection results. Table 2 reports overall EX and its breakdown by difficulty. FROST reaches 65.7% EX while dropping 20.4% of the synthetic data, outperforming the other data selection methods at the same drop ratio. It also improves EX by 0.9% over training with the full synthetic pool.

Table 2: Spider 1.0 fine-tuning with synthetic data pool, evaluated on the dev split. EX is execution accuracy (%), reported overall and by difficulty. ΔEX is the overall EX change relative to Real + All Synthetic. All methods retain the real training data; selection methods use a matched drop ratio.

				EX by difficulty (%) ↑			
Method	*Drop*	EX ↑	ΔEX ↑	Easy	Medium	Hard	Extra Hard
Real only	*100*	57.1	–	79.4	59.6	43.7	30.7
Real + All Synthetic	*0*	64.8	0.0	83.5	67.0	56.9	39.2
Random	*20.4*	64.1	−0.7	83.9	67.5	51.7	38.6
GradNorm	*20.4*	64.2	−0.6	82.3	67.7	55.7	36.7
GREATS	*20.4*	65.3	+0.5	81.9	68.4	57.5	40.4
FROST (ours)	*20.4*	**65.7**	**+0.9**	83.5	68.6	54.6	42.8

Batch-level and sample-level ablations. We examine how the two stages affect synthetic-data use and downstream EX (Figure 3). *Batch gate only* retains batches within the utility band and drops out-of-band batches without sample-level filtering. *Sample IQR only* applies the IQR rule to every synthetic batch without batch-level routing. Batch gate only drops 80% of the synthetic data and underperforms full FROST, suggesting that out-of-band batches still contain useful samples that sample-level IQR filtering can retain. Sample IQR only also underperforms full FROST: filtering every batch can remove useful supervision from batches that do not need cleanup.

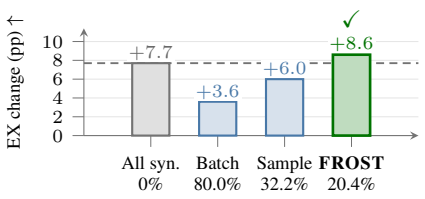


Figure 3: Two-stage ablation on Spider 1.0 dev, measured against the real-only model.

4.3 ADS RE-RANKING ON REAL-WORLD DATA

Data and task. Finally, we evaluate FROST on a production ads re-ranking task, focusing on conversion prediction. Each real training example is an ad impression paired with a binary label indicating whether an advertiser-defined conversion event, such as a purchase, an app install, or a sign-up, is *attributed* to that impression. Attribution links an observed conversion to the impression credited with it, providing a positive training label. We train on a 14-day data window ending on day T, containing billions of real impressions, and evaluate offline on a 24-hour window on day $T + 2$.

Synthetic-data generation. To expand and enrich the training data, we generate synthetic examples from *unattributed* conversions: observed conversions that cannot be tied back to a specific impression because of privacy-related attribution gaps. We sample likely impressions based on user–ad interaction frequencies and pair them with these conversions to create synthetic examples. The resulting synthetic pool is about $\frac{1}{3}$ the size of the real training set. The conversion events are observed, but their links to impressions are inferred and lack ground-truth verification. Ads re-ranking models are sensitive to this label noise because incorrect impression–conversion pairs can distort the learned conversion probabilities. In our experiments, training with all synthetic examples regresses relative to real-only training (Table 3). We therefore apply FROST during mixed real/synthetic training to select synthetic examples that provide useful supervision for the real-data task.

Model. We use a deep learning recommendation model to predict conversion probabilities.

Evaluation metric: Normalized Entropy. We use Normalized Entropy (NE) to evaluate the model's binary predictions (He et al., 2014). NE normalizes the average binary cross-entropy by the label entropy computed from the observed conversion rate in the evaluation set:

$$\mathrm{NE} = \frac{-\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log p_i + (1-y_i)\log(1-p_i)\right]}{-\hat{p}\log\hat{p} - (1-\hat{p})\log(1-\hat{p})}. \tag{13}$$

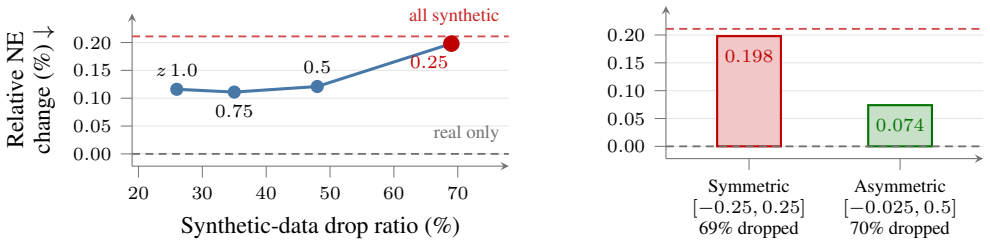
Here, N is the number of real evaluation examples, $y_i \in \{0, 1\}$ is the observed label, p_i is the model's predicted probability, and $\hat{p} = N^{-1}\sum_{i=1}^{N} y_i$ is the observed conversion rate. Lower NE indicates better probabilistic predictions, and even small reductions can be meaningful at industrial scale (He et al., 2014). In well-optimized commercial recommendation systems, relative NE reduction of only 0.02% has been considered significant in prior industrial studies (Li et al., 2022; Lai et al., 2023).

We report relative NE change as $\Delta\mathrm{NE}\,(\%) = 100(\mathrm{NE}_{\mathrm{model}}/\mathrm{NE}_{\mathrm{real}} - 1)$, where $\mathrm{NE}_{\mathrm{model}}$ and $\mathrm{NE}_{\mathrm{real}}$ are the NE of the evaluated model and the fixed real-only baseline, respectively. All models are evaluated on the same real-data split using the same $\hat{p}$. The real-only baseline is 0%; negative values indicate improvement and positive values indicate regression.

Effect of synthetic-data selection. We include *Random* downsampling at 30% and 70% drop ratios as a general-purpose selection baseline. Due to constraints of the commercial setup, more comprehensive baseline comparisons are conducted on the public image-classification and text-to-SQL tasks. Table 3 shows that adding all synthetic data **increases** NE by 0.211% compared to real-only training. Although synthetic data expands the training set, noise in the inferred labels can offset this benefit and hurt performance. Applying **FROST** **lowers** NE by 0.096% while dropping only 36% of the synthetic data. FROST therefore reverses the effect of the synthetic pool, converting a 0.211% regression into a 0.096% improvement. This is direct evidence that FROST filters out the noisy and harmful synthetic data and retains useful examples that contribute genuine signal to the model, with an NE gain far beyond the industrial significance threshold.

Table 3: Ads re-ranking results relative to real-only training. *Drop* is the synthetic-data drop ratio; all real data are retained. Red ↑ denotes higher NE (worse); green ↓ denotes lower NE (better).

Training data / selection	*Drop* (%)	Relative NE change (%) ↓
Real data only (reference)	*100*	–
+ All synthetic data	*0*	+0.211 ↑
+ Random	*30*	+0.118 ↑
+ Random	*70*	+0.023 ↑
FROST	*36*	−0.096 ↓



(a) Tightening the symmetric band

(b) Symmetric vs. asymmetric bands

Figure 4: Sensitivity to the batch utility band on the ads model. Relative NE change (%) uses the real-only reference; lower is better. Dashed lines mark real-only training (0%) and all synthetic data (+0.211%). (a) Narrowing the symmetric z-band does not consistently improve NE. (b) The asymmetric band achieves lower NE at a similar drop ratio.

Sensitivity to the batch utility band. We examine how the width and asymmetry of the z-band affect performance in the batch-only variant. We progressively narrow a symmetric band from $[-1, 1]$ to $[-0.25, 0.25]$. Figure 4(a) shows that the drop ratio rises from 26% to 69%, but dropping more data does not consistently improve performance. This pattern suggests that aggressive whole-batch rejection removes useful supervision along with noise: an out-of-band mean does not imply that every sample in the batch is unhelpful. We then compare the symmetric band $[-0.25, 0.25]$ with the asymmetric band $[-0.025, 0.5]$. At similar drop ratios (69% and 70%), the asymmetric band yields a lower relative NE change (+0.074% versus +0.198%), as shown in Figure 4(b). Raising the lower bound excludes more relatively low-utility batches, while relaxing the upper bound preserves moderately high-utility batches that the symmetric gate would discard. The upper bound still excludes extreme positive scores from direct acceptance. This comparison supports favoring moderately positive standardized utility rather than narrowing the band symmetrically around zero. We use $[-0.025, 0.5]$ for ads and $[-0.025, 1]$ for both public domains.

Batch-level and sample-level ablations. Figure 5 compares each stage in isolation against their combination. *Batch gate only* applies the asymmetric gate and discards every out-of-band batch whole, leaving +0.074% while dropping 70% of the synthetic data. Given the noisy synthetic ads supervision, we use a negative lower IQR multiplier ($\lambda_l < 0$) to tighten the acceptance interval. *Sample IQR only* skips routing and applies this IQR rule to every synthetic batch, reaching +0.036% at a 54% drop. Full FROST instead retains moderate-utility batches in full and applies sample-level filtering only to batches flagged by routing. This combination reaches −0.096% while dropping only 36% of the synthetic data, improving on real-only training. These results confirm the value of batch-level routing: applying IQR filtering to every batch can remove helpful samples and hurt performance. Retaining moderate-utility batches entirely while applying sample-level filtering to out-of-band batches yields better performance than either stage alone.

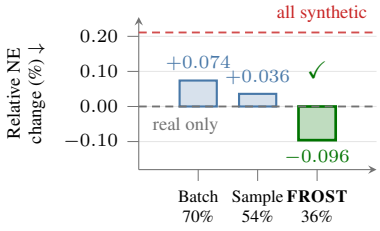


Figure 5: Two-stage ablation. Percentage denotes the synthetic data drop ratio.

5 Conclusion

We present FROST, a novel online synthetic-data filtering framework that selects useful synthetic data to improve downstream task performance. FROST estimates synthetic-data utility through gradient feedback anchored in real training data, without requiring an external verifier or a held-out validation set. Experiments on image classification and LLM fine-tuning show that FROST improves absolute accuracy with fewer synthetic training samples, demonstrating robust effectiveness of our approach across different domains, model architectures, and 3 synthetic image generators. Moreover, in a large-scale industrial ads re-ranking system, FROST delivers significant performance gains over a highly optimized production baseline, flipping the regression from the full noisy synthetic pool into an improvement over real-only training.

AI USE STATEMENT

Required disclosure: We used LLM assistants to help formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, and implement methods. To generate synthetic datasets, we used pretrained text-to-image models, with LLaVA and Qwen assisting image captioning and prompt construction (Appendix B.1). We also used publicly released synthetic image and text-to-SQL datasets generated by prior work. The remaining required disclosure tasks are not applicable to our use of LLM assistants in this work: helping develop theoretical models or conceptual frameworks; proposing or refining hypotheses; designing or providing feedback on research methodology or experiments; assisting with translation; cleaning and reformatting datasets; supporting qualitative and thematic data analysis; and interpreting results.

Additionally, we used LLM assistants to edit the manuscript to improve readability, identify relevant literature, format references, suggest a structure for the paper, propose titles, and prepare figures and tables. The authors take responsibility for the final content of this work, including AI-assisted text, mathematical claims, proofs, and code.

REFERENCES

Krzysztof Adamkiewicz, Brian B. Moser, Stanislav Frolov, Tobias Christian Nauen, Federico Raue, and Andreas Dengel. When pretty isn't useful: Investigating why modern text-to-image models fail as reliable training data generators. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 36660–36669, 2026.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, 2007.

Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J. Fleet. Synthetic data from diffusion models improves ImageNet classification. *Transactions on Machine Learning Research (TMLR)*, 2023.

Black Forest Labs. FLUX.1 [schnell]. Hugging Face model release, 2024. URL `https://huggingface.co/black-forest-labs/FLUX.1-schnell`.

Fartash Faghri, David Duvenaud, David J. Fleet, and Jimmy Ba. A study of gradient variance in deep learning, 2020. arXiv:2007.04532.

Lijie Fan, Kaifeng Chen, Dilip Krishnan, Dina Katabi, Phillip Isola, and Yonglong Tian. Scaling laws of synthetic images for model training ... for now. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7382–7392, 2024. doi: 10.1109/CVPR52733.2024.00705.

Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018. doi: 10.1016/j.neucom.2018.09.013.

Scott Geng, Cheng-Yu Hsieh, Vivek Ramanujan, Matthew Wallingford, Chun-Liang Li, Pang Wei Koh, and Ranjay Krishna. The unmet promise of synthetic training images: Using retrieved real images performs better. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7902–7929, 2024. doi: 10.52202/079017-0254.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2023. arXiv:2306.11644.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip H. S. Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? In *International Conference on Learning Representations (ICLR)*, 2023.

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. Practical lessons from predicting clicks on ads at Facebook. In *International Workshop on Data Mining for Online Advertising*, pp. 1–9, 2014. doi: 10.1145/2648584.2648589.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report, 2024. arXiv:2409.12186.

Neha Hulkund, Alaa Maalouf, Levi Cai, Daniel Yang, Tsun-Hsuan Wang, Abigail O'Neil, Timm Haucke, Sandeep Mukherjee, Vikram V. Ramaswamy, Judy Hanwen Shen, Gabriel Tseng, Mike Walmsley, Daniela Rus, Ken Goldberg, Hannah Kerner, Irene Y. Chen, Yogesh Girdhar, and Sara Beery. DataS^3: Dataset subset selection for specialization, 2025. arXiv:2504.16277.

Chenhan Jin, Shengze Xu, Qingsong Wang, Fan Jia, Dingshuo Chen, and Tieyong Zeng. OrderDP: A theoretically guaranteed lossless dynamic data pruning framework. In *International Conference on Learning Representations (ICLR)*, 2026.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International Conference on Machine Learning (ICML)*, pp. 2525–2534, 2018.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. GRAD-MATCH: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning (ICML)*, pp. 5464–5474, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Fan Lai, Wei Zhang, Rui Liu, William Tsai, Xiaohan Wei, Yuxi Hu, Sabin Devkota, Jianyu Huang, Jongsoo Park, Xing Liu, Zeliang Chen, Ellie Wen, Paul Rivera, Jie You, Chun-cheng Jason Chen, and Mosharaf Chowdhury. AdaEmbed: Adaptive embedding for large-scale recommendation models. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 817–831, 2023.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. OmniSQL: Synthesizing high-quality Text-to-SQL data at scale. *Proceedings of the VLDB Endowment*, 18(11):4695–4709, 2025. doi: 10.14778/3749646.3749723.

Yan Li, Dhruv Choudhary, Xiaohan Wei, Baichuan Yuan, Bhargav Bhushanam, Tuo Zhao, and Guanghui Lan. Frequency-aware SGD for efficient embedding learning with provable benefits. In *International Conference on Learning Representations (ICLR)*, 2022.

Shaobo Lin, Kun Wang, Xingyu Zeng, and Rui Zhao. Explore the power of synthetic data on few-shot object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 638–647, 2023. doi: 10.1109/CVPRW59228.2023.00071.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. LLaVA-NeXT: Improved reasoning, OCR, and world knowledge. Project blog post, 2024. URL `https://llava-vl.github.io/blog/2024-01-30-llava-next/`.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, 1982. doi: 10.1109/TIT.1982.1056489.

Hongzan Mao, Baisong Liu, Xueyuan Zhang, Wei Liu, Zijing Wang, and Zining Feng. DALLRec: An effective data augmentation framework with fine-tuning large language model for recommendation. *Journal of King Saud University Computer and Information Sciences*, 37(7), 2025. doi: 10.1007/s44443-025-00158-4.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 20596–20607, 2021.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 19920–19930, 2020.

Ziheng Qin, Kai Wang, Zangwei Zheng, Jianyang Gu, Xiangyu Peng, Zhaopan Xu, Daquan Zhou, Lei Shang, Baigui Sun, Xuansong Xie, and Yang You. InfoBatch: Lossless training speed up by unbiased dynamic data pruning. In *International Conference on Learning Representations (ICLR)*, 2024.

Qwen Team. Qwen2.5: A party of foundation models. Project blog post, 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pp. 8748–8763, 2021.

Parham Rezaei, Filip Kovačević, Francesco Locatello, and Marco Mondelli. High-dimensional analysis of synthetic data selection. In *International Conference on Learning Representations (ICLR)*, 2026.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022. doi: 10.1109/CVPR52688.2022.01042.

Jordan Shipard, Arnold Wiliem, Kien Nguyen Thanh, Wei Xiang, and Clinton Fookes. Diversity is Definitely Needed: Improving Model-Agnostic Zero-shot Classification via Stable Diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 769–778, 2023. doi: 10.1109/CVPRW59228.2023.00084.

John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of LLM scaling based on human-generated data. In *International Conference on Machine Learning (ICML)*, pp. 49523–49544, 2024.

Jiachen T. Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. GREATS: Online selection of high-quality data for LLM training in every iteration. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 131197–131223, 2024. doi: 10.52202/079017-4169.

Xiangyu Wang, Yawen He, Shivendra Pratap Singh, Han Huang, Mengtong Hu, Sharath Ciddu, Yi-Hsuan Hsieh, Erik Groving, Yi Ding, Jieming Di, Tony Wang, Min Yun, Xiaoyu Chen, Ling Leng, and Rob Malkin. Synthetic data from cross-domain events for large-scale recommendation systems, 2026. arXiv:2606.00282.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 13484–13508, 2023. doi: 10.18653/v1/2023.acl-long.754.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. LLMRec: Large language models with graph augmentation for recommendation. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 806–815, 2024. doi: 10.1145/3616855.3635853.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning (ICML)*, pp. 54104–54132, 2024.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. SANA: Efficient high-resolution text-to-image synthesis with linear diffusion transformers. In *International Conference on Learning Representations (ICLR)*, 2025.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and Text-to-SQL task. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3911–3921, 2018. doi: 10.18653/v1/D18-1425.

A Theoretical Analysis

A.1 Derivation of the Utility Approximation

We analyze training benefit as a local real-loss reduction to clarify what the utility score captures. Fix the current model and real mini-batch $\mathcal{R}_t$, and let $\widehat{\mathcal{L}}_{R,t}(\phi)$ denote its empirical loss as a function of the scoring block ϕ, with all other parameters held fixed. Its gradient is $\nabla_\phi \widehat{\mathcal{L}}_{R,t}(\phi_t) = R_t$. For a hypothetical SGD step $\phi' = \phi_t - \eta \bar{g}_{S,t}$, define

$$\Delta_{R,t} = \widehat{\mathcal{L}}_{R,t}(\phi_t) - \widehat{\mathcal{L}}_{R,t}(\phi'). \tag{14}$$

Assuming the empirical real loss is L-smooth over this step, Taylor's theorem gives

$$\Delta_{R,t} = \eta R_t^\top \bar{g}_{S,t} + \mathcal{H}_t, \qquad |\mathcal{H}_t| \le \frac{L\eta^2}{2} \|\bar{g}_{S,t}\|^2, \tag{15}$$

where $\mathcal{H}_t$ is the signed higher-order correction. Let $e_t = R_t - \hat{R}_t$ be the deviation of the current real-batch gradient from the EMA reference. Substituting $R_t = \hat{R}_t + e_t$ and $U_t = \hat{R}_t^\top \bar{g}_{S,t}$ yields

$$\underbrace{\Delta_{R,t}}_{\text{training benefit}} = \underbrace{\eta U_t}_{\text{first-order utility}} + \underbrace{\eta e_t^\top \bar{g}_{S,t}}_{\text{reference error}} + \underbrace{\mathcal{H}_t}_{\text{higher-order effects}}, \tag{16}$$

The reference-error and higher-order terms can either increase or decrease the local benefit. By Cauchy–Schwarz,

$$\Delta_{R,t} \ge \eta U_t - \eta \|e_t\| \, \|\bar{g}_{S,t}\| - \frac{L\eta^2}{2} \|\bar{g}_{S,t}\|^2. \tag{17}$$

A larger U_t increases the first-order term but does not determine the other two terms. In particular, large gradient norms can raise the raw alignment score while also enlarging the higher-order error bound. Consequently, utility alone need not preserve the ordering of local training benefits.

Equation 16 concerns a single update in the scoring block; the experiment in Section 4.1 examines subsequent full-model training. In that comparison, High reaches a higher early accuracy peak but ends the 30-step window with lower mean real-test accuracy than Mid. At this checkpoint, the highest-utility band is therefore not the best choice for short-horizon training.

A.2 Theoretical Guarantee for EMA Reference Tracking

In Section 3.1, we estimate the target real gradient using a bias-corrected exponential moving average (EMA) of real mini-batch gradients. Here, we show that the expected squared error between the EMA reference $\hat{R}_t$ and the population real gradient $\bar{g}_t = \nabla_\phi \mathcal{L}_R(\theta_t)$ in the scoring block ϕ is bounded under the assumptions below.

Assumptions. Let $\bar{g}_t = \nabla_\phi \mathcal{L}_R(\theta_t)$ denote the population real gradient at step t. We assume the following conditions for the real-data gradients and optimization process:

1. **Unbiased mini-batch gradients.** Let $\mathcal{F}_{k-1}$ denote the complete training history before sampling the real mini-batch at step k, including the current parameters θ_k. Thus, θ_k and $\bar{g}_k$ are $\mathcal{F}_{k-1}$-measurable. We assume
$$\mathbb{E}[R_k - \bar{g}_k \mid \mathcal{F}_{k-1}] = 0. \tag{18}$$
2. **Bounded conditional variance.** The conditional variance of the real mini-batch gradient noise is bounded by ν_R^2: $\mathbb{E}[\|R_k - \bar{g}_k\|^2 \mid \mathcal{F}_{k-1}] \le \nu_R^2$ almost surely.
3. **L-smoothness.** The population real-data objective is L-smooth in the full parameter vector: $\|\nabla_\theta \mathcal{L}_R(\theta_x) - \nabla_\theta \mathcal{L}_R(\theta_y)\| \le L\|\theta_x - \theta_y\|$.
4. **Bounded parameter updates.** The actual full-model update satisfies $\|\theta_{k+1} - \theta_k\| \le S$ almost surely, where S is a uniform bound on the update norm.

Result. Under the assumptions above, the expected squared tracking error of the bias-corrected EMA in Equation 5 is bounded by the following expression for fixed $0 \le \beta < 1$ and $t \ge 1$:

$$\mathbb{E}\|\hat{R}_t - \bar{g}_t\|^2 \le 2\nu_R^2 \left(\frac{1-\beta}{1+\beta}\right) \frac{1}{(1-\beta^t)^2} + 2L^2 S^2 \left(\frac{\beta}{1-\beta}\right)^2 \frac{1}{(1-\beta^t)^2}. \tag{19}$$

Proof. From Equation 5, the bias-corrected EMA can be unrolled as a weighted sum of past mini-batch gradients:

$$\hat{R}_t = \sum_{k=1}^{t} w_{t,k} R_k, \qquad w_{t,k} = \frac{(1-\beta)\beta^{t-k}}{1-\beta^t}, \tag{20}$$

The weights sum to one, $\sum_{k=1}^{t} w_{t,k} = 1$. We decompose the error into mini-batch noise and tracking lag from parameter drift:

$$\hat{R}_t - \bar{g}_t = \underbrace{\sum_{k=1}^{t} w_{t,k}(R_k - \bar{g}_k)}_{\text{Noise}} + \underbrace{\sum_{k=1}^{t} w_{t,k}(\bar{g}_k - \bar{g}_t)}_{\text{Tracking Lag}}.$$

Using $\|a+b\|^2 \le 2\|a\|^2 + 2\|b\|^2$ and taking expectations gives

$$\begin{aligned}\mathbb{E}\|\hat{R}_t - \bar{g}_t\|^2 \le 2\,\mathbb{E}\left\|\sum_{k=1}^{t} w_{t,k}(R_k - \bar{g}_k)\right\|^2 \\ + 2\,\mathbb{E}\left\|\sum_{k=1}^{t} w_{t,k}(\bar{g}_k - \bar{g}_t)\right\|^2.\end{aligned}$$

1. Bounding the Noise Term. The mini-batch gradient noise has zero conditional mean. For $j < k$, the earlier noise $R_j - \bar{g}_j$ is $\mathcal{F}_{k-1}$-measurable. By the tower property,

$$\begin{aligned}\mathbb{E}\langle R_j - \bar{g}_j, R_k - \bar{g}_k\rangle &= \mathbb{E}[\langle R_j - \bar{g}_j, \mathbb{E}[R_k - \bar{g}_k \mid \mathcal{F}_{k-1}]\rangle] \\ &= 0.\end{aligned}$$

Thus, the cross terms vanish, giving

$$\mathbb{E}\left\|\sum_{k=1}^{t} w_{t,k}(R_k - \bar{g}_k)\right\|^2 = \sum_{k=1}^{t} w_{t,k}^2 \mathbb{E}\|R_k - \bar{g}_k\|^2 \le \nu_R^2 \sum_{k=1}^{t} w_{t,k}^2.$$

Bounding the sum of squared weights by an infinite geometric series,

$$\begin{aligned}\sum_{k=1}^{t} w_{t,k}^2 &= \frac{(1-\beta)^2}{(1-\beta^t)^2} \sum_{k=1}^{t} \beta^{2(t-k)} \\ &\le \frac{(1-\beta)^2}{(1-\beta^t)^2} \sum_{a=0}^{\infty} \beta^{2a} = \frac{1-\beta}{(1-\beta^t)^2(1+\beta)}.\end{aligned} \tag{21}$$

Thus, the noise contribution is bounded by $\nu_R^2(1-\beta)/[(1-\beta^t)^2(1+\beta)]$.

2. Bounding the Tracking Lag Term. By L-smoothness and the bounded-update assumption, the population gradient drift in the fixed scoring block satisfies

$$\|\bar{g}_k - \bar{g}_t\| \le L\|\theta_k - \theta_t\| \le L\sum_{j=k}^{t-1} \|\theta_{j+1} - \theta_j\| \le LS(t-k).$$

Applying this bound to the weighted sum gives, almost surely,

$$\begin{aligned}\left\|\sum_{k=1}^{t} w_{t,k}(\bar{g}_k - \bar{g}_t)\right\| &\le \sum_{k=1}^{t} w_{t,k}\|\bar{g}_k - \bar{g}_t\| \\ &\le \frac{LS(1-\beta)}{1-\beta^t} \sum_{k=1}^{t} \beta^{t-k}(t-k).\end{aligned}$$

With $a = t-k$, the arithmetico-geometric series yields

$$\sum_{k=1}^{t} \beta^{t-k}(t-k) \le \sum_{a=0}^{\infty} a\beta^a = \frac{\beta}{(1-\beta)^2}.$$

Therefore, the tracking lag is bounded by $LS\beta/[(1-\beta^t)(1-\beta)]$. Squaring this bound and combining it with the noise bound proves Equation 19.

Equation 19 establishes a bias–variance trade-off for the EMA reference. As t grows, the $(1-\beta^t)$ terms approach one. In this long-history regime, a high β, such as 0.99, suppresses the mini-batch noise contribution, while introducing a tracking-lag term proportional to $L^2S^2[\beta/(1-\beta)]^2$. Small parameter updates control this lag, supporting the use of EMA when the real-gradient trajectory changes slowly.

B Implementation Details

Filtering configuration. We use a rolling-buffer window of $W = 200$ batch scores for image classification and ads re-ranking, and $W = 50$ for LLM fine-tuning.

B.1 Image classification

Data and training. CIFAR-100 (Krizhevsky, 2009) contains 100 classes, with 500 training and 100 test images per class. We use a CIFAR-style ResNet-18 (He et al., 2016) with $[2, 2, 2, 2]$ BasicBlocks across four stages of widths $64/128/256/512$, a 3×3 convolutional stem with stride 1, and a linear $512 \to 100$ classifier. We use SGD with momentum 0.9 and weight decay 5×10^{-4}. We train for 200 epochs with a batch size of 1,024 and a learning rate of 0.2 with a OneCycle schedule, using real and synthetic images. We evaluate accuracy on the 10,000 real test images.

Generation pipeline. We construct the FLUX.1 [schnell] and SANA pools through image captioning, prompt diversification, and text-to-image generation. Captioning and rewriting are performed once offline, cached as JSONL, and shared by both generators. Figure 6 shows examples of the real images and the resulting synthetic variants.

Image captioning. We use LLaVA-NeXT-Mistral-7B (Liu et al., 2024), specifically `llava-hf/llava-v1.6-mistral-7b-hf`, to caption each of the 50,000 real training images. Images are bicubically upsampled from 32×32 to 336×336. We use greedy decoding in bfloat16, a batch size of 16, and at most 150 new tokens. The prompt supplies the class name and asks for the setting, colors, viewpoint, and lighting, while excluding references to image quality or resolution. A deterministic cleaning pass removes residual quality descriptors and opening boilerplate, trims incomplete trailing sentences, and normalizes whitespace. We prepend the class name to the cleaned caption, disambiguating class names when needed, and retain this base prompt as v0.

Prompt diversification. Qwen2.5-7B-Instruct (Qwen Team, 2024) generates three rewrites (v1–v3) of each cleaned caption, using temperature 0.9, top-p 0.95, a batch size of 32, and at most 400 new tokens. We request a JSON array of three distinct, single-sentence captions under 25 words each. The instructions preserve the subject class while varying the background, surface, lighting, time of day, and camera angle. They also request one large, centered subject with a simple background and prohibit wide shots, multiple subjects, and image-quality descriptors. These framing constraints aim to keep the labeled object recognizable after downsampling to the classifier's 32×32 input resolution.

Text-to-image generation. FLUX.1 [schnell] (Black Forest Labs, 2024) and SANA-1.6B (Xie et al., 2025) each generate one image per conditioning string: the base caption v0 and three Qwen variants v1–v3. Each generator thus produces four synthetic examples per real image, giving 2,000 images per class and 200,000 images in total. Separately, we use a Stable Diffusion v1.4 pool (Rombach et al., 2022) from the dataset release of Shipard et al. (2023), containing 1,800 images per class. This released pool is not generated through our caption pipeline.

Image features and reference data. For the feature-based image selection baselines, we use the final pooled-and-projected 512-dimensional image embeddings from a frozen OpenAI CLIP ViT-B/32 encoder (Radford et al., 2021). In our feature-extraction pipeline, images are bicubically resized to 224×224, normalized using ImageNet statistics, and processed without data augmentation. This normalization follows the authors' public implementation for DS3 and CovMatch (Rezaei et al., 2026), which uses ImageNet statistics instead of CLIP's default preprocessing. Embeddings are cached in FP16, then converted to FP32 and ℓ_2-normalized before selection. Selection is performed



Bicycle v0 (excerpt) a bicycle leaning against a tree in a grassy area.
Bicycle v2 (Qwen) a bicycle gleaming on a sandy beach, with the sun setting behind a range of hills.

Figure 6: Examples from the image-generation pipeline. Rows show apple, bicycle, and castle. The first column contains real CIFAR-100 images; the remaining columns are SANA-1.6B outputs conditioned on the cleaned LLaVA caption (v0) and three Qwen rewrites (v1–v3). Generated images are downsampled to 32×32 with Lanczos and enlarged with nearest-neighbor interpolation for display; real images are enlarged from their native 32×32 resolution. The bicycle captions illustrate how rewriting changes the scene while preserving the subject.

independently for each CIFAR-100 class, using all 500 real training images of that class as references.

Covariance Matching. For CovMatch (Rezaei et al., 2026), we fit a 32-dimensional principal component analysis (PCA) projection on each class's real features and apply the same projection to its real and synthetic features. We greedily select synthetic examples to minimize the Frobenius distance between the unbiased sample covariance of the selected synthetic features and the corresponding real-data covariance.

DS3. We implement the nearest-cluster selection strategy from DataS^3 (Hulkund et al., 2025), adapted to synthetic pools by Rezaei et al. (2026). For each class, we run k-means on the synthetic features, starting with $K = 200$ clusters and using k-means++ initialization (Arthur & Vassilvitskii, 2007) and Lloyd's algorithm (Lloyd, 1982). A cluster is eligible if its centroid is the nearest centroid to at least one real reference example. We uniformly sample the exact selection budget from the union of eligible clusters. If this union contains too few examples, we successively halve K and repeat clustering and eligibility selection until the budget can be met.

Prompt templates. The following templates are used for captioning and diversification. `{class_name}` denotes the subject class, `{caption}` the cleaned LLaVA caption, and `{n}` is set to 3. LLaVA uses the Mistral instruction format; Qwen receives separate system and user messages.

LLaVA-NeXT: captioning prompt

```
[INST] <image>
This is a photo of a {class_name}. In one sentence, describe the scene: the
setting or background, the main colours, the camera viewpoint, and the
lighting. Describe it as a normal photograph. Do not mention image quality,
resolution, pixelation, blurriness, or that it is small or a thumbnail. Do
not begin with 'The image' or 'This image'. [/INST]
```

Qwen2.5-7B-Instruct: system prompt

```
You rewrite image captions into variants for a text-to-image generator. Every
variant keeps one single subject large and centred in the frame. You reply with
JSON and nothing else.
```

Qwen2.5-7B-Instruct: user prompt

```
Subject: "{class_name}"
Base caption: "{caption}"

Write exactly {n} DIFFERENT one-sentence captions of the SAME subject.

Every caption must:
- show ONE single {class_name}, close up, filling most of the frame;
- vary the background, surface, lighting, time of day and camera angle
  relative to the base caption and to each other;
- keep the background simple and secondary -- a short phrase at most;
- stay photographic and physically plausible;
- be under 25 words.

Never do any of these:
- zoom out, or describe a wide shot, landscape, skyline, street scene or room;
- show several {class_name}s, a crowd, a pile, a collage or a grid;
- make something other than the {class_name} the main subject;
- mention image quality, resolution, pixelation or blurriness.

Reply with only a JSON array of {n} strings.
```

B.2 LLM FINE-TUNING FOR TEXT-TO-SQL

Model and input format. We fully fine-tune Qwen2.5-Coder-0.5B-Instruct (Hui et al., 2024) in bfloat16, without adapters. We use the model's chat template with a fixed system prompt. The user turn contains the database's `CREATE TABLE` schema, external knowledge when available, and the question, followed by `SQLite query:`. Schemas are read directly from SQLite files using the same serialization for training and inference. The target is whitespace-normalized gold SQL followed by EOS. We mask prompt tokens from the loss and left-truncate the prompt when necessary to preserve the target within 2,048 tokens.

Optimization and inference. We train for one epoch with a global batch size of 64 and AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, zero weight decay). The learning rate is 5×10^{-5}, with a 10% warmup followed by cosine decay to zero. Gradients are clipped to a global ℓ_2 norm of 1.0 after accumulation. At inference, we use greedy decoding with at most 160 new tokens and report EX on Spider 1.0 dev.

Training objective. We average cross-entropy over each example's supervised tokens, then average over retained examples in the global batch. At step t, let $\mathcal{T}_i$ be example i's supervised token positions, $z_{ik,t}$ its logits, and y_{ik} the target token. The objective is

$$\ell_{i,t} = \frac{1}{|\mathcal{T}_i|} \sum_{k \in \mathcal{T}_i} \mathrm{CE}(z_{ik,t}, y_{ik}), \qquad \mathcal{L}_t = \frac{\sum_{i \in \mathcal{G}_t} w_i \ell_{i,t}}{\sum_{i \in \mathcal{G}_t} w_i}, \tag{22}$$

where $\mathcal{G}_t$ is the global batch and $w_i \in \{0, 1\}$ is the filter's keep weight; real examples always have $w_i = 1$. The denominator includes all accumulation micro-batches. Normalizing by the kept count avoids shrinking the loss by the retained fraction. Cross-entropy is computed in FP32.

Token-wise rank-1 gradient computation. We compute utility without constructing a full LM-head gradient for each example. For scoring, we use the output-projection gradient with hidden states held fixed, excluding the input-embedding path when weights are tied. Let $f_{ik,t} \in \mathbb{R}^d$ be the hidden state and $\delta_{ik,t} = \mathrm{softmax}(z_{ik,t}) - \mathrm{onehot}(y_{ik}) \in \mathbb{R}^V$ the gradient of token-level cross-entropy with respect to the logits. Each token contributes a rank-1 outer product, so the example's head gradient is

$$G_{i,t} = \frac{1}{|\mathcal{T}_i|} \sum_{k \in \mathcal{T}_i} \delta_{ik,t} f_{ik,t}^\top, \qquad g_{i,t} = \mathrm{vec}(G_{i,t}). \tag{23}$$

Let $\widehat{\mathbf{R}}_t \in \mathbb{R}^{V \times d}$ be the matrix form of the bias-corrected reference $\hat{R}_t$ in Equation 5. Reordering the inner product gives

$$u_{i,t} = \langle \widehat{\mathbf{R}}_t, G_{i,t} \rangle_F = \frac{1}{|\mathcal{T}_i|} \sum_{k \in \mathcal{T}_i} \delta_{ik,t}^\top \big(\widehat{\mathbf{R}}_t f_{ik,t} \big). \tag{24}$$

This computes the same head-level dot product without materializing $G_{i,t}$. With $V = 151{,}936$ and $d = 896$, a single FP32 head-gradient matrix would occupy approximately 0.51 GiB, or over 32 GiB for 64 examples. We instead process supervised tokens in chunks of at most 256, bounding token-wise temporary storage while retaining one shared real-reference matrix.

B.3 Computational overhead

Utility computation. We retain the sample utility $u_{i,t} = \hat{R}_t^\top g_{i,t}$ from Equation 6. Here $m = |S_t|$ is the number of synthetic candidates before filtering, and $|\mathcal{R}_t|$ is the number of real examples in the same training step. Bias terms are included in the image utility scores but omitted from the leading-order cost analysis below. For a linear prediction head with input dimension d and output dimension C, let $f_{i,t} \in \mathbb{R}^d$ be the input feature and $\delta_{i,t} = \mathrm{softmax}(z_{i,t}) - \mathrm{onehot}(y_i) \in \mathbb{R}^C$ the cross-entropy gradient with respect to the logits, where $z_{i,t}$ and y_i are the example's logits and class label. Following the matrix notation in Appendix B.2, write the weight gradient as $G_{i,t} = \delta_{i,t} f_{i,t}^\top$ and the corresponding weight component of the same EMA reference $\hat{R}_t$ as $\widehat{\mathbf{R}}_t \in \mathbb{R}^{C \times d}$. Its contribution to the utility is

$$\langle \widehat{\mathbf{R}}_t, G_{i,t} \rangle_F = \delta_{i,t}^\top \big(\widehat{\mathbf{R}}_t f_{i,t} \big). \tag{25}$$

Given the features and logit gradients, scoring m synthetic examples costs $O(mCd)$ without materializing per-example head-gradient matrices. Aggregating the reference from $\mathcal{R}_t$ costs $O(|\mathcal{R}_t|Cd)$, and updating the EMA costs $O(Cd)$. The head-level computation uses $O(Cd + (|\mathcal{R}_t| + m)(C + d))$ storage for the reference, features, logit gradients, and intermediate products, avoiding an $O((|\mathcal{R}_t| + m)Cd)$ per-example gradient tensor. The full model is still trained; these costs concern only head-level scoring. For LLMs, the analogous computation is performed per supervised token, using the factorization and chunking described in Appendix B.2.

Batch aggregation and calibration cost $O(m + W)$ when buffer statistics are computed directly. Sorting-based IQR filtering costs $O(m \log m)$ and is needed only for out-of-band batches; utility scores are computed for all synthetic candidates before routing.

ResNet-18 arithmetic cost. We compare head-level scoring with full-model training using the convolutional structure of ResNet-18. We measure arithmetic cost in multiply–accumulate operations (MACs), counting one multiplication and accumulation as one MAC. For convolution j, let H_j, W_j denote its output height and width, $c_j^{\mathrm{in}}, c_j^{\mathrm{out}}$ its input and output channel counts, and k_j its kernel width. Its forward cost is $H_j W_j c_j^{\mathrm{in}} c_j^{\mathrm{out}} k_j^2$ MACs per image. Summing over the backbone, including the stem and projection shortcuts, gives

$$F_{\mathrm{backbone}} = \sum_j H_j W_j c_j^{\mathrm{in}} c_j^{\mathrm{out}} k_j^2.$$

The classifier adds Cd MACs. Backpropagation computes both weight and activation gradients, with a dominant convolution/linear cost of approximately twice the forward cost. Table 4 summarizes these terms for a batch of $|\mathcal{R}_t| + m$ examples.

Table 4: Leading arithmetic costs for regular ResNet-18 training and head-level utility computation. The latter assumes that features and logit gradients are already available.

Computation	MACs per step
Regular forward	$(\lvert\mathcal{R}_t\rvert + m)(F_{\text{backbone}} + Cd)$
Regular backward	$\approx 2(\lvert\mathcal{R}_t\rvert + m)(F_{\text{backbone}} + Cd)$
Real-reference aggregation	$\lvert\mathcal{R}_t\rvert Cd$
Synthetic utility scoring	mCd
EMA update	$O(Cd)$

Thus, the estimated regular forward/backward cost is

$$\text{MAC}_{\text{train}} \approx 3(|\mathcal{R}_t| + m)(F_{\text{backbone}} + Cd), \tag{26}$$

whereas the additional reference aggregation and scoring matrix operations require

$$\text{MAC}_{\text{head}} = (|\mathcal{R}_t| + m)Cd, \tag{27}$$

apart from the EMA update and lower-order score operations. The gap arises because backbone convolutions operate across spatial positions and multiple layers, with cost quadratic in channel width for equal-width convolutions, while head scoring operates on pooled features with cost linear in d for fixed C. Specifically, let A_d be the spatial area of the final ResNet stage and N_d the number of its $d \to d$ convolutions with kernel width k. These convolutions alone contribute $N_d A_d k^2 d^2$ MACs per image, so $F_{\text{backbone}} \geq N_d A_d k^2 d^2$. Under the above cost model,

$$\frac{\text{MAC}_{\text{head}}}{\text{MAC}_{\text{train}}} \approx \frac{Cd}{3(F_{\text{backbone}} + Cd)} \leq \frac{C}{3N_d A_d k^2 d}. \tag{28}$$

For the CIFAR-style ResNet-18, $N_d A_d k^2 d \gg C$, making the head-level matrix computation much smaller than regular forward/backward training. This comparison concerns convolution/linear arithmetic rather than end-to-end runtime.

Measured training throughput. Because the added arithmetic is negligible, the measured overhead is dominated by memory traffic and synchronization rather than by FLOPs. Compared with regular training, FROST reduces training throughput by approximately 6.8% in image classification and 5.7% in large-scale industrial ads re-ranking, where utility is computed using the final shared-backbone layer. Unlike MAC counts, end-to-end throughput also reflects implementation overheads, which can include logging, communication, and CPU–GPU synchronization. These measurements indicate modest overhead in both training settings.

C Additional Results

C.1 Two-stage ablation on Spider 1.0, by difficulty

Table 5 gives the per-difficulty breakdown behind Figure 3. Full FROST improves overall EX over Batch gate only and Sample IQR only by 5.0 and 2.6 percentage points while dropping less synthetic data than either. On Extra Hard, the gains are 9.1 and 6.7 points, respectively. Sample IQR only performs better than full FROST on Hard by 1.7 points, but has lower EX in the other three difficulty categories.

Table 5: Two-stage ablation on Spider 1.0 dev. Full FROST is the reference. *Drop* is the synthetic-data drop ratio (%); EX is execution accuracy (%). ΔEX is the overall EX change relative to full FROST, in percentage points (pp). All variants retain the real training data.

				EX by difficulty (%) ↑			
Variant	*Drop*	EX ↑	ΔEX ↑	Easy	Medium	Hard	Extra Hard
Full FROST	*20.4*	**65.7**	0.0	83.5	68.6	54.6	42.8
Batch gate only	*80.0*	60.7	−5.0	79.0	63.5	53.4	33.7
Sample IQR only	*32.2*	63.1	−2.6	80.6	65.9	56.3	36.1