LEARNING FROM NOISY, DIVERSE SOURCES WITH DATAWEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training specialized models increasingly requires learning from large, heterogeneous datasets whose examples vary in quality, domain relevance, and source of supervision. Dataweighting—assigning different weights to training examples—offers a general way to control how such data influences training, but learning effective weights at scale remains challenging. We develop a scalable approach to learning dataweights: given a small, high-quality target dataset, we use metagradients through the training process to efficiently optimize training-example weights directly for performance on the target domain. To scale beyond a fixed dataset and model, we learn parametric weighting rules that generalize to new and streaming examples and transfer from smaller models to substantially larger ones. We study reward modeling as a prototypical application, where preference data may be noisy, adversarial, synthetic, or mismatched with the target domain. Learned dataweights improve held-out performance, automatically adapt reward models to specialized domains, and allow mixtures of human and synthetic supervision to outperform either source alone. More generally, the objective used to learn dataweights need not match the loss used to train the model. We use this flexibility to improve downstream rankings while still training with the standard Bradley–Terry loss.

1 INTRODUCTION

In modern machine learning, we often want to train reliable, specialized models using heterogeneous, general-purpose data. For example, large language models are trained on massive corpora scraped from the internet; vision models are trained on heterogeneous collections of web images. Domain-specific models are often adapted from such general-purpose models using much smaller, specialized datasets. Accomplishing this goal poses several challenges, including low-quality or missing data, distribution shift, and mismatches between the training objective and the downstream evaluation criterion.

Standard approaches to these challenges are often bespoke. For example, we might design heuristic filters to remove low-quality data (e.g., Penedo et al., 2023); synthesize labels where they are missing (e.g., Xie et al., 2020); or explore data mixing strategies between general and target-specific data (e.g., Xie et al., 2023). While these techniques can be highly effective, they require substantial problem-specific design and often rely on human judgments about what constitutes ‘good” or ‘bad” data—judgments that may themselves be unreliable or unavailable in many domains. As models become increasingly specialized and customizable, and as the amount of available general-purpose data grows, it is natural to seek a more scalable and general approach.

An attractive approach is to assign a weight to each training example (Huang et al., 2006; Ren et al., 2018; Shu et al., 2019). We call this practice *dataweighting*. In theory, dataweighting can reduce the influence of outliers and low-quality examples and reshape the effective training distribution to better align with downstream performance. In practice, however, choosing the right weights is difficult, particularly when little is known a priori about the relationship between the training data and the target domain. In this paper, we therefore ask:

Is there a scalable way to learn dataweights *for training on diverse, heterogeneous data?*

Contributions. In this work, we revisit *dataweighting* as a unified, scalable way of reducing both the human effort and subjective decision-making involved in training specialized models. Our central premise is that a training sample’s utility should be determined by its impact on the performance of the resulting model, rather than by external heuristics. Instead of predefining quality criteria, we infer data usefulness directly from the learning process itself, and reweight the data’s impact appropriately.

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.

We provide algorithms and principles for instantiating dataweighting at scale. First, we leverage tools from data attribution (and specifically metadifferentiation (Ilyas & Engstrom, 2025; Engstrom et al., 2025)) to quantify how individual training examples influence a model's behavior. In this way, data reweighting becomes an adaptive procedure that is directly aligned with the target objective. Second, we show how to implement dataweighting for streaming data by using a parametric *dataweighting model*. Third, we show that dataweights can transfer between smaller models and larger ones—enabling us to train dataweighting models efficiently and deploying them at scale. Finally, we show how to optimize dataweights for any arbitrary downstream objective, rather than just loss.

In Section 4 we identify *reward modeling* as a prototypical example of the challenges we study. First, preference datasets may contain spam, bot-generated responses, inconsistent judgments, or subjective preferences. Second, large pretrained language models serve as attractive sources of auxiliary supervision, but their integration into training pipelines introduces additional uncertainty and bias. Third, reward models' training objective (Bradley-Terry loss) is typically not aligned with the downstream uses that we care about, such as providing accurate model rankings. Fourth, there is increasing demand for reward models in narrow domains such as mathematics or science, and data that is useful for one domain may be harmful to another. Finally, preference data is typically collected in a streaming fashion, necessitating flexible, online strategies. Our results from applying dataweighting to reward model training from human and synthetic preferences support four main takeaways:

1. Fitting dataweights allows improving model performance despite heterogeneous, noisy examples. The "good" examples naturally get upweighted relative to the "bad" ones.
2. Dataweights naturally allow us to *customize* models. For example, if the target dataset comes from a specialized domain such as coding or STEM question answering, dataweights will automatically discover relevant examples among diverse training instances.
3. When data consists of a mix of human and synthetic labels, dataweights can automatically adapt to the quality of the synthetic labels, upweighting the helpful ones and downweighting the harmful ones. This results in improved performance over both (a) using real human data alone, and (b) using *all* human and synthetic data while treating them equally.
4. Dataweighting allows us to close the gap between the training loss and the downstream task we care about (in this case, accuracy of the leaderboard induced by the reward model).

Finally, in Section 5 we provide a few stylized examples illustrating these same benefits of dataweighting, with the goal of providing some theoretical intuition about how dataweights work.

2 Related Work

We provide an extended account of related work in Appendix E, and focus on differentiating from the most closely related work here. Closest to our work are papers on automatic data selection using data attribution methods (Ghorbani & Zou, 2019; Xia et al., 2024; Xie et al., 2023) and metagradient descent (Engstrom et al., 2025; Calian et al., 2025; Dass et al., 2025), as well as prior work on learning to reweight data (Ren et al., 2018). Compared to these works, our work differs in several ways (across methodology, goals, and results) that are particularly relevant to the reward modeling setting. First, unlike prior work on data selection, our algorithm allows for a continuous reweighting of datapoints (similar to (Ren et al., 2018) but unlike (Engstrom et al., 2025; Dass et al., 2025; Xia et al., 2024)) rather than a hard selection; despite mixed evidence on the effectiveness of reweighting (Wang et al., 2021; Sagawa et al., 2019) we find it highly effective for curating reward model data. Second, to learn these dataweights, our algorithm leverages *exact gradients through the entire learning process*, unlike prior works (e.g., (Ren et al., 2018; Calian et al., 2025)) which approximate this gradient by differentiating through only a few steps of model training. Third, we show how to use dataweights not only to improve model performance but to customize models to specific domains. Fourth, we demonstrate how such automatic dataweighting unlocks previously unhelpful synthetic data pipelines for reward model training. Finally, we show how to distill automatic dataweighting into the parameters of a dataweighting model, allowing for highly efficient and cost-effective online weighting.

3 Training Models from Heterogeneous Data with Dataweights

In this section, we describe our problem of training a reliable specialized model from noisy, heterogeneous, and partially synthetic data. We then describe dataweighting at a high level, followed by how we instantiate it for this problem at scale.

Problem statement. We are given a training dataset of size n and a target dataset of size n'. The training dataset is a large, heterogeneous dataset composed of many potentially noisy sources. In contrast, the target dataset is a small, highly-curated gold-standard dataset that represents the target domain. Our goal is to train a model θ using all the available samples that performs well on the target domain.

3.1 DATAWEIGHTING

Our methods will leverage the target dataset to learn weights that place more mass on relevant training data. For any vector of weights $w \in \mathbb{R}^n$, define the reweighted loss function

$$L^w(\theta) = \sum_{i=1}^{n} w_i \ell(z_i; \theta). \tag{1}$$

Also define $\mathcal{A}(w, \theta_0, k)$ to be the iterative model training procedure that optimizes the weighted loss $L^w(\theta)$ starting from initialization θ_0 for k steps; note that here, $\mathcal{A}$ captures the model architecture, optimizer, hyperparameters, and in general everything about training other than the weight vector w, initialization θ_0, and step count k. Finally, let $L^{\theta}_{val}(\theta)$ represent the loss of r_θ on the target set. Here, we use the same loss function ℓ in both the training and target objectives, but this need not be the case.

Our goal is to learn a choice of weights $\hat{w}$ such that the model obtained by optimizing $L^{\hat{w}}$ performs well on the target loss $L'(\theta)$. We call these learned weights *dataweights*. In essence, dataweights seek to solve the following problem:

$$\text{minimize}_w \quad L'(\mathcal{A}(w, \theta_0, k)). \tag{2}$$

The solution will place higher weight on data points that help the model perform well on the target loss while zeroing out the data points that hurt performance.

3.2 INSTANTIATING DATAWEIGHTING AT SCALE VIA META-OPTIMIZATION

Finding the optimal dataweights can be a challenge. Optimizing (2) via standard methods like gradient descent would require computing $\nabla_w L'(\mathcal{A}(w, \theta_0, k))$, which requires differentiating through the entire training procedure $\mathcal{A}$. The quantity $\nabla_w L'(\mathcal{A}(w, \theta_0, k))$ is called a *metagradient*; efficient tools for computing metagradients have been developed over a long line of work in data attribution (Koh & Liang, 2017; Ghorbani & Zou, 2019; Ilyas et al., 2022; Grosse et al., 2023; Park et al., 2023; Ilyas et al., 2024). We will not discuss the efficient computation of metagradients here—refer to Section 3.3 of Ilyas & Engstrom (2025) for a thorough treatment. With metagradients in hand, we simply use first-order methods (i.e., metagradient descent (Engstrom et al., 2025)) to find the dataweights $\hat{w}$.

Optimizing the dataweights. To optimize the dataweights, we use a form of sign stochastic gradient descent (Seide et al., 2014) applied to the metagradient. We give the algorithm below:

$$\hat{w}_{m+1} = \hat{w}_m - \eta \operatorname{sign}\left(\nabla_w L'\left(\mathcal{A}\left(\hat{w}_m, \theta_m, K\right)\right)\right)$$
$$\theta_{m+1} = \begin{cases} \theta_0 & \text{if } (m+1) \bmod (S/K) = 0 \\ \mathcal{A}(\hat{w}_m, \theta_m, K) & \text{else} \end{cases}$$

for a learning rate $\eta > 0$, a number of gradient steps S, a segment length K that divides S, and initial weights $\hat{w}_0 = \mathbf{1}_n$. We stop when $m + 1 = \Omega_{\max} \frac{S}{K}$, where $\Omega_{\max}$ is the total number of "outer" steps.

Next, we give some intuition for this method. When $K = S$, the method corresponds to exact metagradient descent on the training algorithm. At step m, we reinitialize the reward model with parameters θ_0, train it using the current dataweights w_m, calculate the metagradient, and update the weights. We stop after $\Omega_{\max}$ iterations of this procedure.

However, this idealized procedure with $K = S$ can be slow because computing a single metagradient requires a full training run. To get faster metagradient feedback, we chunk up the training run into *segments* of length K and perform one approximate metagradient update for each segment. This means in total we take $\Omega_{\max} \frac{S}{K}$ approximate metagradient steps, which can accelerate convergence. In our experiments, we will typically set the segment length K to be a constant fraction of the total training duration. This is in contrast to some prior works, e.g., (Calian et al., 2025; Ren et al., 2018) which set K to be a small constant, and is enabled by recent advances in scalable data attribution (Park et al., 2023; Bae et al., 2024; Ilyas & Engstrom, 2025).

Parametric dataweights. The dataweights $w \in \mathbb{R}^n$ defined above carry one learnable scalar per training point, so the parameter count grows with the dataset size n. A natural alternative is to parameterize w as a function of features of the underlying training point. Concretely, let $\phi : \mathcal{Z} \to \mathbb{R}^d$ be a fixed feature map (e.g., a frozen sentence embedding of the example). We then learn parameters $\psi \in \mathbb{R}^p$ and define

$$w_i^\psi \;=\; g_\psi(\phi(z_i))\,, \tag{3}$$

where g_ψ is a small differentiable function. We refer to the original case as the *identity* parameterization (one scalar per sample, recovering (2)) and to (3) as the *parametric* parameterization. Throughout the paper we instantiate the parametric variant with an affine projection followed by a softplus, i.e., $g_\psi(\phi(z)) = \mathrm{softplus}(\phi(z)^\top \mathbf{a} + b)$ with $\psi = (\mathbf{a}, b) \in \mathbb{R}^{d+1}$, so that $p = d + 1$ (the $+1$ is the bias term). The metagradient $\nabla_\psi L'\big(\mathcal{A}\big(w^\psi, \theta_0, k\big)\big)$ is computed by chain rule at no asymptotic overhead. With $p \ll n$, parametric dataweights are dataset-size-independent, extrapolate to new training points sharing the feature space, and constrain the optimization to a low-dimensional subspace, which we find empirically can transfer better across model scales (Section 4.5).

Beyond training-loss targets. The target L' in (2) does not have to match the training objective L^w, nor be differentiable. Many quantities of practical interest such as model ranking metrics are not. In that case we replace the non-differentiable L' with a differentiable surrogate $\widetilde{L}'$, optimizing dataweights against $\widetilde{L}'$, and reporting performance on the original target. We instantiate this in Section 4.6 with a differentiable pairwise rank surrogate and show that it produces dataweights that improve held-out rank correlation, while inner training still uses the standard Bradley-Terry loss.

4 Applying Dataweighting to Reward Modeling

In this section, we identify *reward modeling* as an ideal testbed for training specialized models from heterogenous data, and use it to evaluate dataweighting. In reward modeling, our input is *preference data*: a collection of prompts, each paired with two candidate responses and a label indicating which response is preferred. The goal is to learn a model that takes a prompt and a response and assigns a numerical score, allowing us to compare new responses by predicting which one would be preferred. We train the model with the Bradley–Terry objective (Bradley & Terry, 1952; Christiano et al., 2017), which encourages higher scores for preferred responses; see Appendix A for training details.

Reward modeling is an excellent candidate for dataweighting: preference data is extremely noisy, comes from diverse sources, often synthetic, and increasingly used to train reliable, targeted reward models for specific tasks. We demonstrate the efficacy of dataweighting on real human preference data in a variety of settings. We begin by giving an overview of the training and target datasets we use, as well as our reward model training procedure.

Training and target data. We use Arena human preferences (`Human-155k`; 155,577 examples) (Chiang et al., 2024), 10,000 Arena-flagged bot and adversarial examples (`Bot-10k`), and synthetic Gemini 3 Flash labels on the prompt-response pairs in `Human-155k` (`Gemini-155k`). Human/synthetic mixtures use subsamples with disjoint rows. For targets, `Expert` is a curated subset of Arena preferences for sophisticated, domain-specific prompts, testing gains on a niche part of the training distribution. `STEM` contains math, science, and coding preferences from HelpSteer3 and Tulu2.5 (Wang et al., 2025; Ivison et al., 2024), with annotations from qualified human rubric labelers or ground-truth verifiers; it tests generalization beyond crowdsourced Arena preferences. Each domain is split into a meta-optimization target set and a held-out test set: 3000/547 examples for `Expert` and 6796/6796 for `STEM`.

Reward model training. We fine-tune `Qwen2.5-0.5B-Instruct` unless otherwise specified. Except where otherwise stated, for fair comparisons *all* methods use the same fixed-size *checkpoint selection set* of 100 examples for early stopping, disjoint from both the meta-optimization target and test sets. Appendix A details dataset construction, sampling, and reward-head implementation.

4.1 Downweighting Noisy Data with Dataweights

In the first set of experiments, we apply dataweighting with `Expert` data as the target.

Improving in-distribution performance. Our first main point is that dataweights can be used to improve in-distribution performance. To demonstrate this, we train a reward model on varying-sized samples of `Human-155k` and show the test loss with and without dataweighting; see Figure 1.

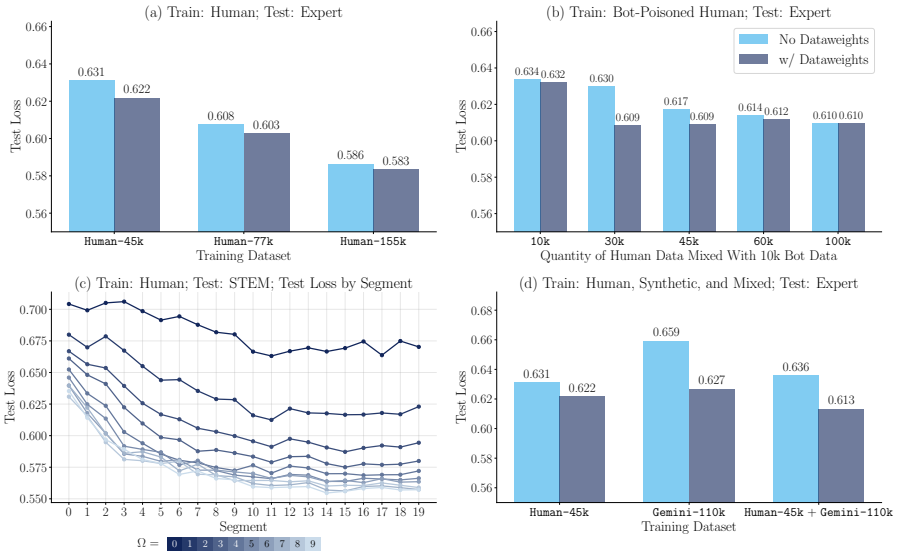


Figure 1: *Dataweighting consistently improves reward models across a variety of settings.* Each plot depicts the test loss of reward models trained with and without dataweighting across different training data sources and evaluation sets. In (a), we apply dataweights to a heterogeneous corpus of human preference data, targeting a curated validation set of expert data, and show that dataweighting improves test-set performance. In (b) we apply dataweighting to a training set comprising heterogeneous human data that has been "poisoned" with data suspected to be from bots; dataweighting allows us to recover some performance. In (c), we plot the test loss as a function of the training segment for dataweights that are further and further optimized for a particular STEM-focused validation set. Finally, in (d) we plot the test loss of models trained with and without dataweighting on real, synthetic, and mixed data; dataweighting enables us to leverage human and synthetic sources to outperform either source in isolation.

Across all dataset sizes, dataweighting noticeably decreases the test loss on `Expert` preferences. Test and target loss curves are found in Appendix Figures 10,19, and 23.

Remedy for data-poisoning. Next, we mix human preference data with noisy and adversarial bot data, varying the human-to-bot data ratio. We again target the clean `Expert` preference set. Figure 1 demonstrates that naive training with such low-quality data can harm performance, especially when the human-to-bot ratio is low. At the same time, in all cases except when the human-to-bot ratio is very high, dataweighting restores performance. Plots of test and target training curves are found in Appendix Figures 12–16. Appendix Fig. 11 shows bot dataweights shift downward during optimization.

4.2 Customizing Reward Models with Dataweights

Next, we demonstrate how dataweighting enables *customization* of reward models to specific domains. We target the `STEM` preference set, which contains math, science, and coding preference pairs. As described earlier, the annotation of this dataset follows a substantially different process from the crowdsourced annotation process on Arena.

Without dataweighting, training on `Human-155k` yields a test loss of 0.6631 on the `STEM` preference set. This makes sense: the target domain is significantly out-of-distribution with respect to the training data. However, once we apply dataweighting, our learned dataweights bring the test loss down to 0.5572. We provide the test loss meta-training curves in Figure 1. With each outer metagradient step (Ω), we decrease the test loss nearly monotonically. Note that the curve for $\Omega = 0$ corresponds to a normal training procedure where all datapoints are weighted equally. The target loss curves, which are markedly identical to the test loss curves, can be found in Appendix Figure 18.

Allocating target-domain data. A natural question is whether reserving target-domain data for meta-optimization is more beneficial than using it directly for training. Given a budget of B `STEM` preference pairs, we allocate C to the meta-optimization target and $B - C$ to training. We compare $C = 0$ (no meta-optimization), $C = B$ (all pairs reserved for the target), and $C = 250$ (split). The split achieves the lowest held-out test loss at every evaluated budget (Figure 2), outperforming ordinary training even though it uses fewer target-domain pairs for training.

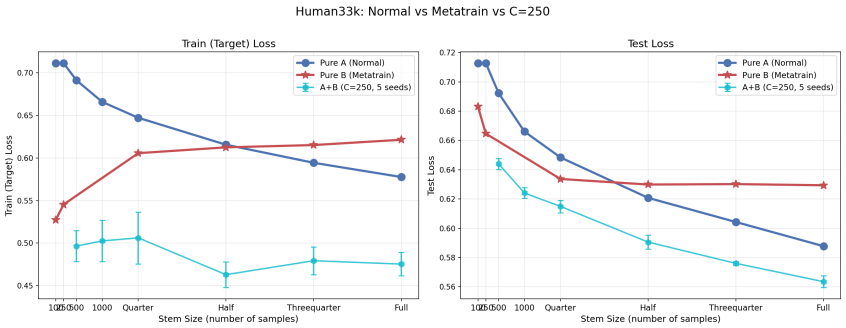


Figure 2: *A small `STEM` target set complements direct training on `STEM` data.* Train/target loss (left) and held-out test loss (right) for a 0.5B reward model with a `Human-33k` training base. Target-domain (`STEM`) pairs are allocated entirely to ordinary training (Pure A), entirely to the meta-optimization target (Pure B), or split, with 250 target pairs and the rest used for training (A+B). Full denotes 6,796 pairs; Quarter, Half, and Threequarter are fractions of the full target set.

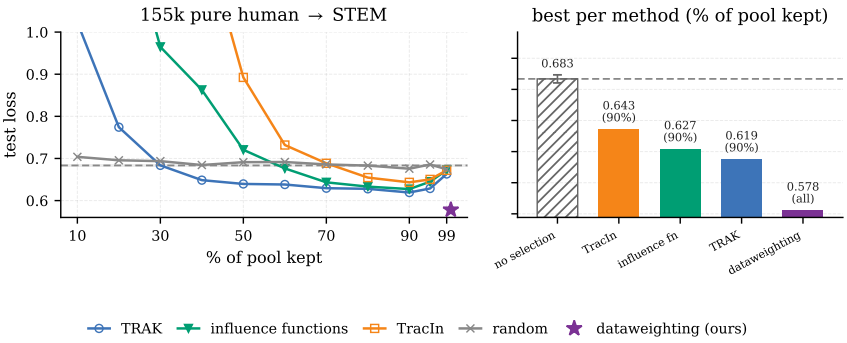


Figure 3: *Dataweighting outperforms attribution-based selection.* Held-out `STEM` test loss after training on `Human-155k`. *Left:* selection baselines across eleven keep fractions, with a random-drop control and a grey band showing ± 1 standard deviation over three unweighted seeds. Dataweighting reweights the full pool (star at 100%). *Right:* each method at its best keep fraction (in parentheses); the hatched bar is the unweighted baseline. Retraining uses a matched optimizer-step budget, and checkpoints are selected by selection-set loss. Dataweighting reaches 0.578, compared with 0.619 for TRAK, the strongest selection baseline. Appendix Figure 30 shows the comparison on all four pools.

4.3 Comparison with Data-Attribution Selection Baselines

A natural alternative to learned dataweights is to *score* each training example with an off-the-shelf data-attribution method and drop the lowest-scoring fraction. We compare against TRAK (Park et al., 2023), influence functions (Koh & Liang, 2017), and TracIn (Pruthi et al., 2020). Each method scores the same training pool against the same target set. We sweep eleven keep fractions, retrain with a matched optimizer-step budget, and include a random-drop control at every fraction. Checkpoints are selected by selection-set loss and evaluated on a disjoint held-out test split. We evaluate clean and poisoned pools against both targets; Appendix D gives the full comparison.

Dataweighting outperforms selection on `STEM`. On `Human-155k`, dataweighting achieves a test loss of 0.578, compared with 0.619 for TRAK at its best keep fraction and 0.683 for unweighted training (Figure 3). Across all four pools, dataweighting improves over the strongest selection baseline by 0.025–0.041 (Appendix Figure 30). The gains persist on both the clean and poisoned pools, indicating that the benefit extends beyond filtering bot data. Unlike hard subset selection, dataweighting can assign continuously varying weights through the weighted training loss (1).

Aggressive selection skews response-length preferences. At low keep fractions, selected preferences become skewed toward shorter responses, particularly for TracIn and influence functions. Such a skew can encourage the reward model to use response length as a shortcut, assigning low probability to preferences favoring longer responses and incurring large losses on those pairs. This offers a plausible explanation for the sharp loss increases under aggressive selection (Appendix Figure 29).

Selection gains depend on the target. On `Expert`, the selection baselines perform similarly to random dropping on the paired clean and poisoned pools: the best baseline improves test loss by

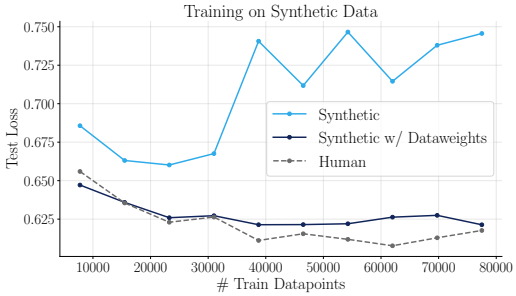


Figure 4: Test loss curves on `Expert` domain, while training on pure synthetic data (`Gemini-77k`), with and without dataweights. Test loss from training on `Human-77k` is shown for reference.

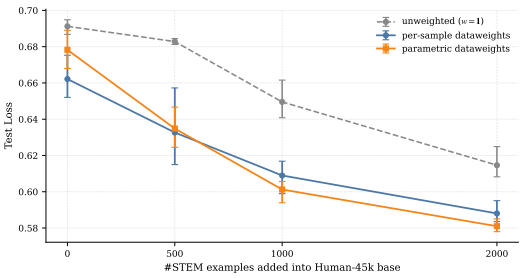


Figure 5: Test loss on `STEM` when meta-training a 0.5B reward model on a 45k-row human preference base augmented with $\{0, 500, 1000, 2000\}$ `STEM` examples, for the unweighted baseline, identity (per-sample) dataweights, and linear (parametric) dataweights.

0.023 versus 0.019 for random dropping on the poisoned pool, and both improve by 0.010 on the clean pool. These gaps are within one unweighted-baseline seed standard deviation, in contrast to the clearer separation on `STEM`.

Computational cost. The number of outer iterations provides a cost–performance tradeoff: later iterations can improve test loss at higher cumulative compute. Appendix Figure 31 plots this trajectory alongside the selection baselines. Meta-optimization produces both the dataweights and the final reward model, so it requires no separate retraining stage. Appendix D provides the evaluation protocol, additional results, and compute accounting.

4.4 Safely Incorporating Synthetic Data with Dataweights

We study whether dataweighting allows us to use synthetic LLM-as-a-judge labels, which are inexpensive and scalable, in place of human labels, while preserving performance.

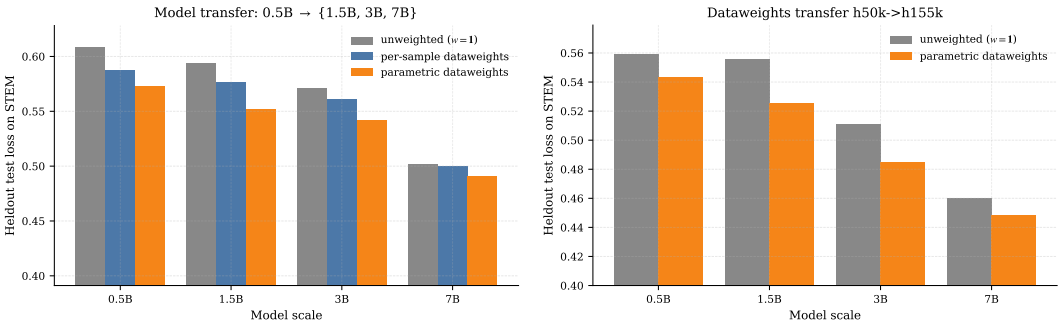
We train three reward models on `Human-45k`, `Gemini-110k`, and `Human-45k` + `Gemini-110k`, respectively. We do this with and without dataweights. We use unequal dataset sizes to reflect real-world constraints: synthetic data is cheap to generate at scale, while human-curated data remains expensive and limited. Appendix Figure 22 contains a version of this experiment with an equal mixture of human and Gemini preferences.

In Figure 1, we observe that under a standard, non-dataweighted procedure, purely training on `Human-45k` gives the best test loss. Training on only synthetic data, `Gemini-110k`, yields poor test loss. Lastly, although mixing synthetic and real data (`Human-45k` + `Gemini-110k`) improves performance over synthetic data alone, it does not exceed training on only the real human subset. Under the dataweighting procedure, training on purely synthetic data results in much closer performance to real human data. Furthermore, with dataweighting, combining both human and synthetic data together reaches the best performance, better than using the real human preference set alone. Full test and target loss curves for `Human-45k` and `Gemini-110k` are in Appendix Figures 19 and 20.

In fact, we find that naively training on synthetic data may not scale well with data quantity. In Figure 4, we see that standard, non-weighted training on more than roughly $23,000$ synthetic datapoints begins to make the test loss worse, potentially overfitting to the synthetic signal. When we apply dataweights to training on synthetic data, we recover a training curve that scales well with data, almost matching the performance of training on real human data.

4.5 Parametric Dataweights and Larger-Scale Transfer

So far our dataweights have been per-sample scalars: one learnable weight per training example (the *identity* parameterization). The number of meta-parameters therefore grows with the dataset size n, which might be prohibitively expensive for large training sets and rules out direct transfer of the dataweights themselves to a different training distribution. We now study the parametric variant of (3): instead of an unconstrained vector in $\mathbb{R}^n$, dataweights are produced by a small head $w_\phi(z_i)$ applied to a fixed feature representation of each preference triplet z_i. Concretely, we take $w_\phi(z) = \mathrm{softplus}(\phi^\top e(z) + b)$, where $e(z) \in \mathbb{R}^{4608}$ is the concatenation of frozen OpenAI `text-embedding-3-small` embeddings (1,536-dim each) of the prompt x, response a, and



(a) *Model transfer.* Transfer 0.5B dataweights to 1.5B, 3B, and 7B reward models on `Human-45k+STEM`. Three bars per scale: unweighted baseline ($w = \mathbf{1}$), identity, and linear dataweights.

(b) *Data transfer.* Same setup as (a), but with the full `Human-155k` preference set as the training base (3.4× scale-up; the extra rows were not seen during meta-training).

Figure 6: Transfer of learned dataweights across model scales (left) and to unseen training examples (right).

Table 1: One held-out split; "meta" denotes meta-optimization. Depicted graphically in Appendix Figure 7.

Method	Spearman $\rho \uparrow$	Rank MAE $\downarrow$
BT (no meta)	0.541	7.91
BT (meta)	0.565	7.67
Spearman (meta)	**0.625**	**7.31**
Spearman (no meta)	0.325	9.05

Table 2: Seven model-disjoint holdout splits. Medians [25th, 75th percentiles], also depicted graphically in Appendix Figure 8.

Method	Spearman $\rho \uparrow$	Rank MAE $\downarrow$
BT (no meta)	0.880 [0.717, 0.943]	9.44 [8.26, 9.86]
BT (meta)	0.915 [0.831, 0.943]	7.89 [7.24, 8.11]
Spearman (meta)	**0.957** [0.946, 0.962]	**6.06** [4.43, 6.18]

response b triplet, and $b \in \mathbb{R}$ is a learnable bias, giving a parameter count of $p = 4{,}609$ (4,608 weights plus one bias)—independent of n. We refer to this as the *linear* parameterization.

Same-scale comparison. We extend the customization setup of Section 3 by varying how many `STEM` preference examples are mixed into a 45k-row human-preference base, with sizes $\{0, 500, 1000, 2000\}$, and meta-train both parameterizations at the 0.5B scale across three random seeds. Figure 5 shows that the linear parameterization matches the identity baseline and outperforms it at several injection sizes, despite using an order of magnitude fewer learnable (meta-)parameters.

Cross-scale transfer. A key practical question is whether dataweights learned with a small reward model can be reused when training larger reward models. We therefore learn dataweights at the 0.5B scale, freeze them, and apply them unchanged when training 1.5B, 3B, and 7B reward models for a single weighted epoch. This *forward-only* transfer requires no additional metagradient computation at the larger scales. Figure 6a shows that both identity and linear dataweights improve over the unweighted baseline across all scales. These results suggest that dataweighting can be amortized: the expensive meta-optimization can be performed at a smaller scale, and the resulting weighting rule can be deployed directly in larger training runs.

Generalizing to unseen training examples. A limitation of per-sample dataweights is that they only assign weights to examples seen during meta-optimization. The linear parameterization removes this restriction: because it maps an example embedding to a weight, it can be evaluated on new training examples without relearning a separate scalar for each one. We test this by applying a linear head learned on the smaller 45k-row base to the full `Human-155k` training set. This transfer requires only computing embeddings and a forward pass through the learned weighting rule, rather than running metagradient optimization on the larger dataset. Figure 6b shows that the resulting weights continue to improve over the unweighted baseline across model scales. Thus, parametric dataweights provide a way to amortize data curation across datasets: a weighting rule learned on a manageable subset can be deployed directly on larger, previously unseen training pools.

4.6 Optimizing a Non-Training Target Objective

So far the target L' has shared the Bradley-Terry form of the training loss. We now consider a target objective that is not itself a training loss: ranking the response-generating LLMs. For each model i,

let $s_i(\theta)$ denote the average reward assigned by the reward model to responses from model i, and let b_i denote the ground-truth Bradley-Terry coefficient fit from the original pairwise annotations. The evaluation metric is Spearman ρ between the model-induced ranking $\{s_i(\theta)\}$ and the ground-truth ranking $\{b_i\}$. Because Spearman ρ is non-differentiable, we optimize a differentiable soft-rank surrogate during meta-optimization. For a score vector u, define

$$\widetilde{r}_i(u) \;=\; 1 + \sum_{j \neq i} \sigma((u_j - u_i)/\tau)\,, \tag{4}$$

which converges to the hard rank as $\tau \to 0$. We take $\widetilde{L}'(\theta)$ to be the negative Pearson correlation between $\widetilde{r}(s(\theta))$ and $\widetilde{r}(b)$. During meta-optimization, this surrogate is used only as the outer objective for learning dataweights; the reward model itself is still trained with the standard Bradley-Terry loss.

Table 1 compares this rank-targeted meta-optimization with standard Bradley-Terry training without dataweighting and with a control that trains the reward model directly using the soft-Spearman loss. Direct soft-Spearman training performs worse than Bradley-Terry training, suggesting that the rank surrogate is a weak inner-loop learning signal. In contrast, using the same surrogate as an outer objective improves both Rank MAE and Spearman ρ. This suggests a useful separation of roles: Bradley-Terry provides the dense local signal needed to fit the reward model, while the rank surrogate operates one level up, selecting the BT-trained trajectory whose induced model ordering best matches the ground-truth downstream ranking.

Consistency across held-out splits. To verify that the improvement in Table 1 is not an artifact of a single model split, we repeat the comparison on seven model-disjoint holdout splits, each holding out a different set of response-generating models from the leaderboard (Table 2). Across splits, Spearman-targeted meta-optimization attains a consistently higher held-out Spearman ρ and lower Rank MAE than both unweighted Bradley-Terry training and BT-targeted meta-optimization, indicating that the rank surrogate selects dataweights whose induced ordering generalizes to unseen models.

5 Theoretical Intuition for Dataweighting

We give three stylized results illustrating how a small target set can guide dataweighting. We study overparameterized linear regression with two signal features and many weak nuisance features. We compare a simplified version of dataweighting (which selects a fixed-size source subset by empirical target loss) to a baseline of pretraining on all source data then fine-tuning on the same target set. Appendix F gives the precise assumptions, statements, and proofs.

Correcting biased synthetic labels. Even arbitrarily small synthetic-label errors can induce a spurious dependence that fine-tuning cannot fully remove. Target-loss selection is competitive with a clean subset and achieves vanishing risk as the source pool grows, while both baselines retain error proportional to the squared label bias (Theorem F.1).

Filtering severe corruption. At fixed sample sizes and dimension, as corruption magnitude grows, target-loss selection identifies an entirely clean subset with probability tending to one, conditional on enough clean examples existing. That condition holds with exponentially high probability in the source pool size (Theorem F.2).

Adapting to covariate shift. Even with correct labels, selecting examples can change the learner's inductive bias to favor features important in the target domain. Under a suitable shift, dataweighting attains arbitrarily small target risk while both baselines retain positive error; the corresponding isotropic setting admits no asymptotic advantage over full-data fine-tuning (Theorem F.3).

These are existence results for exhaustive subset selection, not convergence guarantees for metagradient optimization. The asymptotic risk comparisons take dimension to infinity before source sample size, with target sample size fixed.

6 Conclusion

This paper introduces dataweighting as a method for automatically curating training data for reward models. Our results suggest that dataweighting can be a flexible, scalable method and demonstrate its applicability to the challenges of reward model training, including distribution shift, customization, the prevalence of synthetic data, and continual updating.

AI USE STATEMENT

The main text was written by the authors, with limited AI assistance in editing and condensing selected passages. AI tools were used to execute experiments, assist in proving some of the theoretical results, and prepare the paper in conference format, including layout and conversion of plots to tables. AI tools were not used to generate hypotheses or interpret experimental results. AI tools also provided feedback on the manuscript's methodology and experimental reporting. As part of the experiments, Gemini 3 Flash was used to generate the synthetic preference labels described in Appendix A. The authors take full responsibility for the content of this paper, including AI-assisted text, proofs, and experimental work.

REFERENCES

Toh-Kyeong Ahn, Elinor Ostrom, and James M Walker. Heterogeneous preferences and collective action. *Public choice*, 117(3):295–314, 2003.

Anastasios N Angelopoulos, Stephen Bates, Clara Fannjiang, Michael I Jordan, and Tijana Zrnic. Prediction-powered inference. *Science*, 382(6671):669–674, 2023.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Baker Grosse. Training data attribution via approximate unrolling. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *International conference on machine learning*, pp. 872–881. PMLR, 2019.

Dan A. Calian, Gregory Farquhar, Iurii Kemaev, Luisa M. Zintgraf, Matteo Hessel, Jeremy Shar, Junhyuk Oh, András György, Tom Schaul, Jeffrey Dean, Hado van Hasselt, and David Silver. Datarater: Meta-learned dataset curation. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot Arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 8359–8388. PMLR, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. DataMIL: Selecting data for robot imitation learning with datamodels, 2025.

Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019.

Logan Engstrom, Andrew Ilyas, Benjamin Chen, Axel Feldmann, William Moses, and Aleksander Madry. Optimizing ML training with metagradient descent. *arXiv preprint arXiv:2503.13751*, 2025.

Yunzhen Feng, Ariel Kwiatkowski, Kunhao Zheng, Julia Kempe, and Yaqi Duan. Pilaf: Optimal human preference sampling for reward modeling. *arXiv preprint arXiv:2502.04270*, 2025.

Urs Fischbacher and Simon Gächter. Heterogeneous social preferences and the dynamics of free riding in public goods. Technical report, IZA Discussion Papers, 2006.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Dongming Huang, Nathan Stein, Donald B Rubin, and SC Kou. Catalytic prior distributions with application to generalized linear models. *Proceedings of the National Academy of Sciences*, 117 (22):12004–12010, 2020.

Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19, 2006.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35 (1):73–101, 1964. doi: 10.1214/aoms/1177703732.

Andrew Ilyas and Logan Engstrom. MAGIC: Near-optimal data attribution for deep learning. *arXiv preprint arXiv:2504.16430*, 2025.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.

Andrew Ilyas, Kristian Georgiev, Logan Engstrom, and Sung Min (Sam) Park. Data attribution at scale. ICML Tutorial, 2024. URL `https://ml-data-tutorial.org/`.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Ellen Wu, Valentina Pyatkin, Nathan Lambert, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback, 2024.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pp. 3122–3130. PMLR, 2018.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data only. *Advances in Neural Information Processing Systems*, 36:79155–79172, 2023.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pp. 4334–4343. PMLR, 2018.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Interspeech*, volume 2014, pp. 1058–1062. Singapore, 2014.

Yunyi Shen, Hao Sun, and Jean-Francois Ton. Active reward modeling: Adaptive preference labeling for large language model alignment. In *Forty-second International Conference on Machine Learning*, 2025.

Ali Shirali, Arash Nasr-Esfahany, Abdullah Alomar, Parsa Mirtaheri, Rediet Abebe, and Ariel Procaccia. Direct alignment with heterogeneous preferences. *arXiv preprint arXiv:2502.16320*, 2025.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32, 2019.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Ke Alexander Wang, Niladri S Chatterji, Saminul Haque, and Tatsunori Hashimoto. Is importance weighting incompatible with interpolating classifiers? *arXiv preprint arXiv:2112.12986*, 2021.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages, 2025. URL https://arxiv.org/abs/2505.11475.

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)*, 2024.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695. IEEE, 2020.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. *Advances in neural information processing systems*, 31, 2018.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

A Dataset Construction and Reward Model Training

Training data. We use two public preference datasets, `arena-human-preference-100k` and `arena-human-preference-140k` Chiang et al. (2024). From this, we filter for single-turn, non-tie examples, yielding a combined dataset of 155,577 rows of human-labeled pairwise preference data— we hereby refer to this dataset as `Human-155k`. We also obtained $10,000$ examples of bot and adversarial activity flagged by Arena in their preference data, henceforth referred to as `Bot-10k`.

Using the prompts and response pairs found in `Human-155k`, we generate a synthetic preference dataset by replacing the human label with an LLM-as-a-judge label Zheng et al. (2023). Specifically, we prompt Gemini 3 Flash to pick the best response given the prompt and response pair. Measured against the original human labels, the LLM-as-a-judge labels reached an accuracy rate of 0.6812. We will refer to this synthetic dataset as `Gemini-155k`.

From `Human-155k` and `Gemini-155k` we randomly subsample to derive the dataset pairs (`Human-45k`, `Gemini-110k`) and (`Human-77k`, `Gemini-77k`). Sampled data rows are disjoint within each pair. The first dataset pair illustrates a setting in which collecting synthetic labels is much cheaper than collecting real human labels, while the second dataset pair keeps the sample size fixed so we isolate the effect of using noisy labels.

Target datasets. We evaluate dataweighting on two different target datasets: `Expert` and `STEM`. `Expert` is sourced from Arena's `arena-expert-5k`: a heavily filtered subset of Arena's preference data aimed at identifying sophisticated, domain-specific prompts. Around 5.5% of Arena's preference data is labeled as "expert." Since `Expert` is a subset of Arena's general human preference data, we use it to evaluate in-distribution performance gains of dataweighting, albeit a niche part of the distribution.

`STEM` is sourced from both HelpSteer3 and Tulu2.5 preference datasets (Wang et al., 2025; Ivison et al., 2024). From HelpSteer3, we filter for only rows labeled as "stem" or "code" and for rows where the responses were not tied. These preference rows contain a prompt, two responses, and an overall preference score labeled by ScaleAI annotators noted to have a relevant degree and/or software engineering work experience (Wang et al., 2025). From Tulu2.5, we filter for the `prm800k_pairs_phase2` subset, where chosen and rejected preference pairs have been selected based on mathematical correctness (Ivison et al., 2024; Lightman et al., 2023). Since `STEM` annotations come from ground truth verifiers and/or hired human rubric labelers with relevant domain expertise, rather than crowdsourced organic usage like in `Human-155k`, we use this dataset to evaluate out-of-distribution performance of dataweighting.

Each of `Expert` and `STEM` is split into two chunks: one used as the target dataset, as described in Section 3, and the other used as a holdout set to estimate the test loss. `Expert` has a target/test split of 3000/547. `STEM` has a target/test split of 6796/6796.

Reward model training procedure. We adopt a standard reward model training procedure where we minimize the Bradley-Terry objective, as discussed in Section 4. Unless specified otherwise, the base pre-trained model we start from is `Qwen2.5-0.5B-Instruct`. We also perform some experiments on a larger, 1.5B model, in which case we use `Qwen2.5-1.5B-Instruct`. When training a reward model, we remove the `vocab_head` and replace it with a linear `reward_head` that takes in the hidden dimension output from a `<|cls|>` token and outputs a scalar reward.

We use the same separate 100-example checkpoint selection set across methods for early stopping. This simple protocol lets us evaluate dataweighting; we leave alternatives such as cross-validation to future work.

B Additional Ranking Experiments

Transferring parametric dataweights. The per-sample dataweights in Section 4.6 are tied to specific training rows. Following the parametric dataweighting of Section 4.5, we instead meta-optimize a parametric map from a frozen text embedding of each preference pair to its scalar weight, fit on a small subset against the soft-Spearman objective, and then apply the learned map to score all 92k training pairs for a single forward-only reward-model training run (Figure 9). The transferred

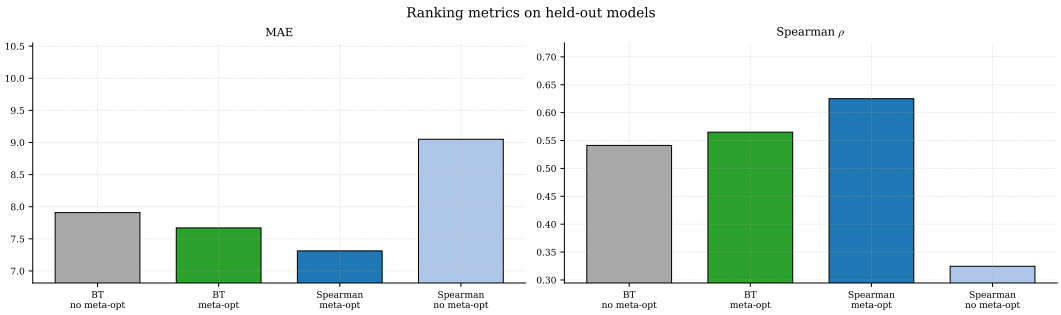


Figure 7: *BT and Spearman: standard vs. meta-optimized reward models.* Rank MAE (lower is better) and Spearman ρ (higher is better) on held-out test models.

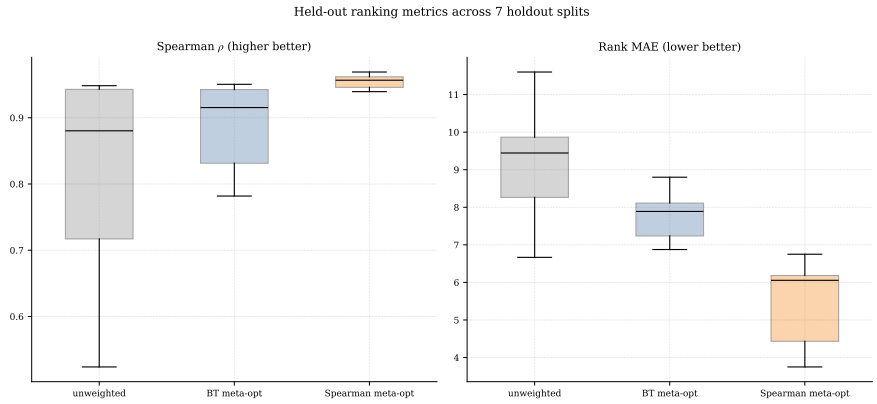


Figure 8: *Rank-targeted meta-optimization improves held-out ranking across splits.* Held-out Spearman ρ (left, higher is better) and Rank MAE (right, lower is better) across seven model-disjoint holdout splits, comparing unweighted Bradley-Terry training, BT-targeted meta-optimization, and Spearman-targeted meta-optimization. The Spearman-targeted objective shifts the whole distribution toward better rankings (higher ρ, lower MAE) relative to both baselines.

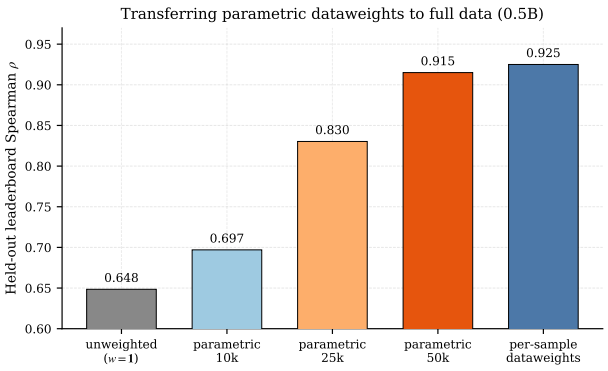


Figure 9: *Parametric rank-dataweights transfer to full-data training.* A parametric (linear-in-embedding) dataweight map is meta-optimized against the soft-Spearman objective on a small subset (10k/25k/50k pairs), then used to predict weights for all 92k training pairs; a fresh `0.5B` reward model is then trained once under those weights. Held-out leaderboard Spearman ρ improves with the meta-training size and approaches free per-sample dataweights, while unweighted training lags well behind.

weighting lifts held-out Spearman ρ from 0.65 (unweighted) to 0.92, approaching the 0.93 obtained by free per-sample dataweights, and the gain grows with the size of the meta-training subset. Because the map depends only on the pair embedding, the same weighting can be applied to a different reward model without re-running meta-optimization.

Alternative rank surrogates. The soft-Spearman surrogate is one of several differentiable choices for the outer objective. Replacing it with a smooth footrule distance (a smooth-ℓ_1 penalty on the soft ranks, aligned with Rank MAE) performs comparably—matching or slightly exceeding soft-

Spearman on held-out ρ (e.g. 0.965 vs. 0.957 on one split), consistent with its closer alignment to the MAE metric—whereas a soft-Kendall-τ surrogate is consistently weaker. We therefore use soft-Spearman by default.

C Additional Results

LLM-as-a-judge prompt. We detail the prompt used for synthetic labels below. Synthetic labels were generated by `Gemini-3-flash` with default settings.

```
You will be shown a user prompt followed by two
responses generated by AI assistants.  Your task is to
determine which response better satisfies the user's
intent.

Evaluate the responses from the *user's perspective*,
focusing on the following criteria:

  • Helpfulness:  Does the response effectively assist
    or inform the user?

  • Relevance:  Does it directly address the user's
    question or request?

  • Accuracy:  Are the facts and reasoning correct?

  • Depth and Detail:  Does it provide sufficient
    explanation or insight?

  • Creativity (when applicable):  Is the response
    original or engaging?

Important guidelines:  Your evaluation must be
impartial and based solely on quality and alignment
with the user's needs.

Return your decision for the winner, either A or B in
the tags <winner> ...  </winner>.
```

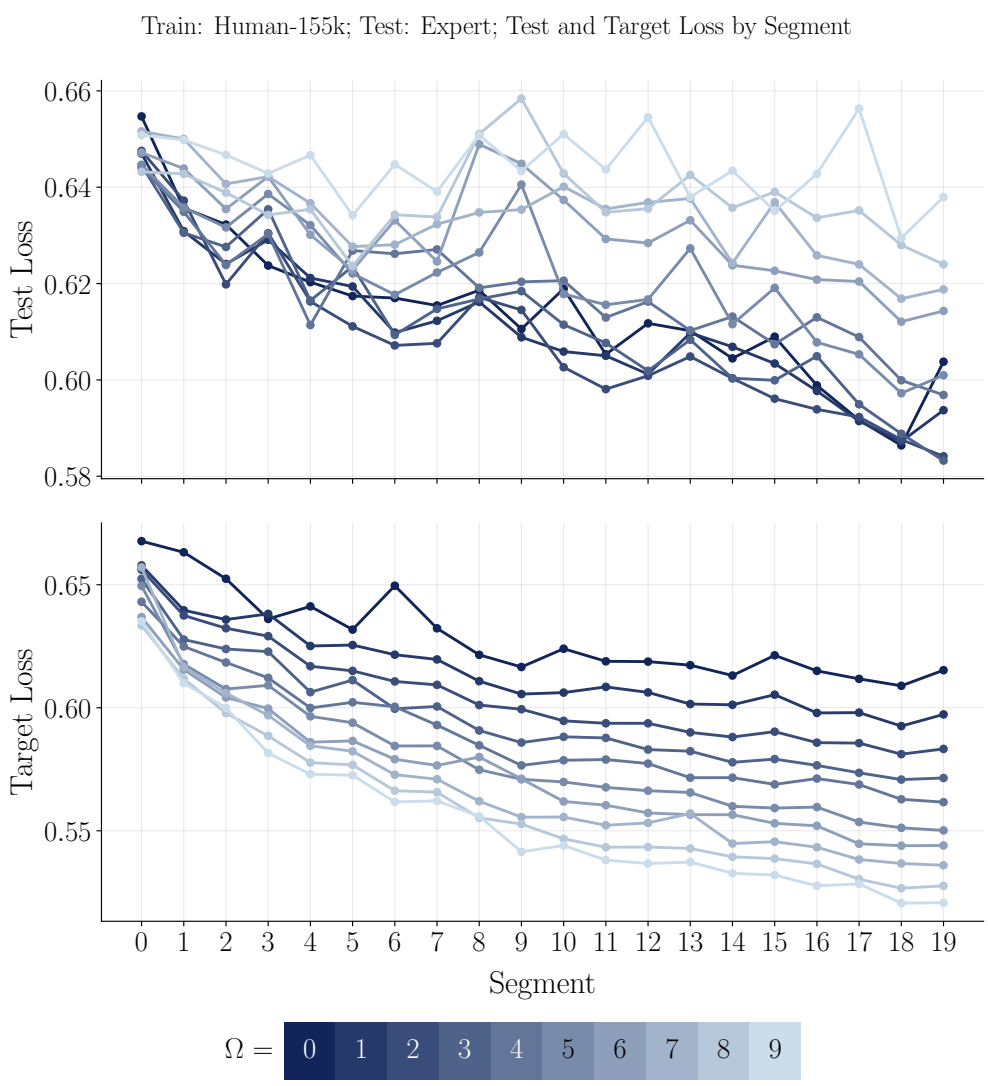


Figure 10: Test and target loss by segment for model trained on `Human-155k`, targeting `Expert`. This example shows notable overfitting. Although the first few metagradients steps do provide some test loss improvement. Further metagradient steps simply overfit the target.

Evolution of bot and human dataweights. In Figure 11, we plot the distribution of dataweights for bot and human datapoints. We observe the dataweight distribution for bots shifting down throughout the dataweighting process.

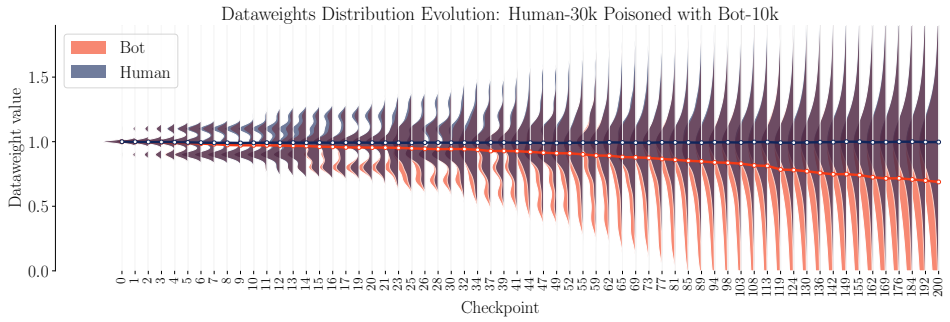


Figure 11: *Dataweighting automatically separates bot and human data.* Here, we visualize a KDE ridge plot of the learned dataweight distribution for human and bot data points throughout the dataweight learning procedure. Each vertical slice is a histogram over dataweights at a specific point in time during dataweight optimization. Bot data points decrease in weight over time.

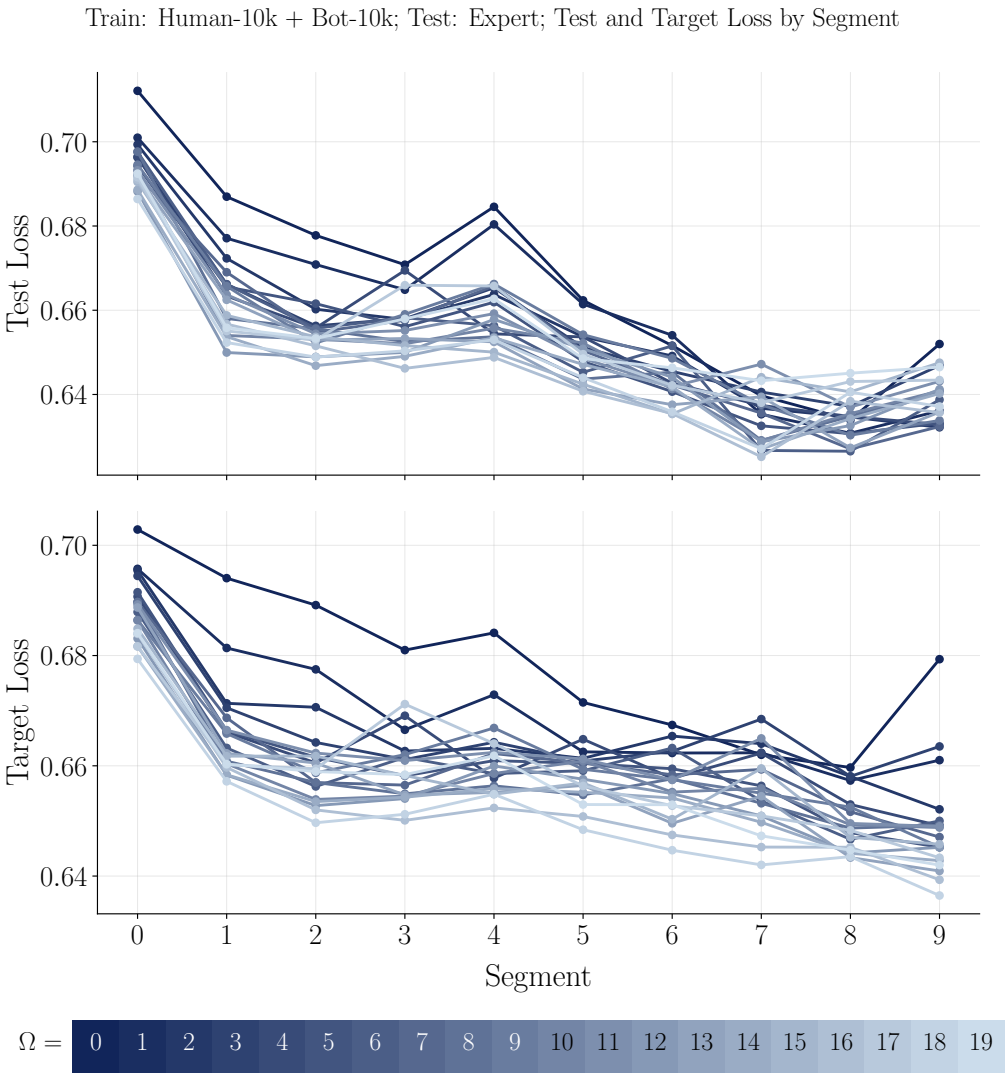


Figure 12: Test and target loss by segment for model trained on `Human-10k` + `Bot-10k`, targeting `Expert`.

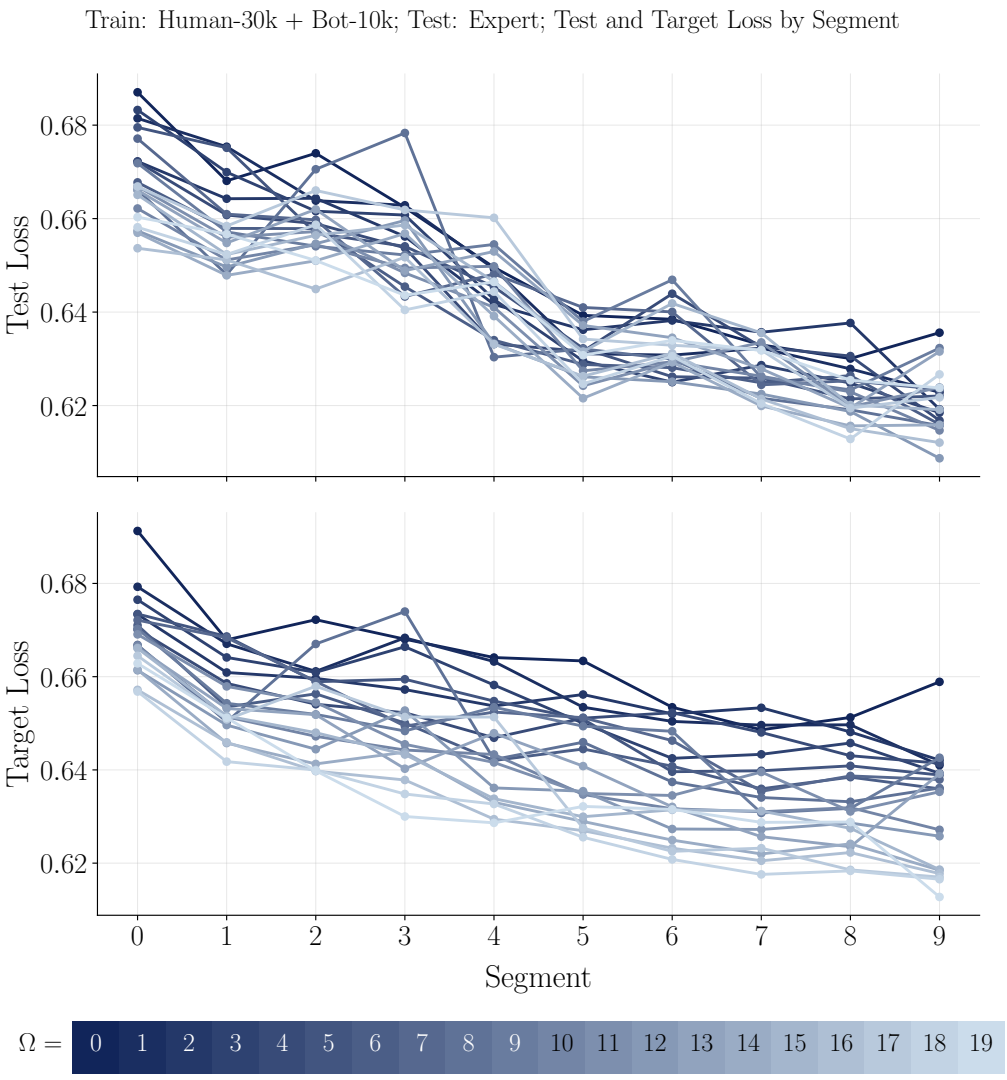


Figure 13: Test and target loss by segment for model trained on `Human-30k` + `Bot-10k`, targeting `Expert`.

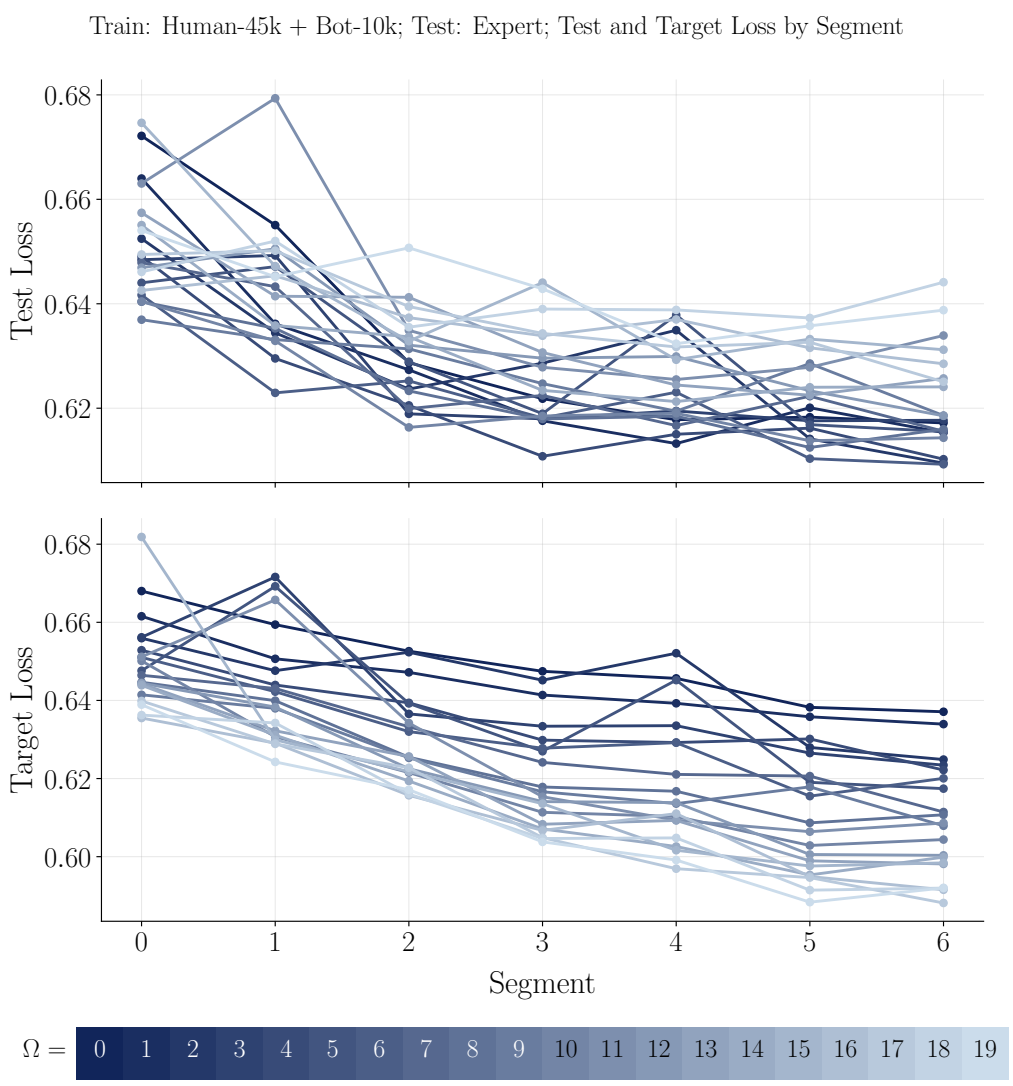


Figure 14: Test and target loss by segment for model trained on `Human-45k` + `Bot-10k`, targeting `Expert`.

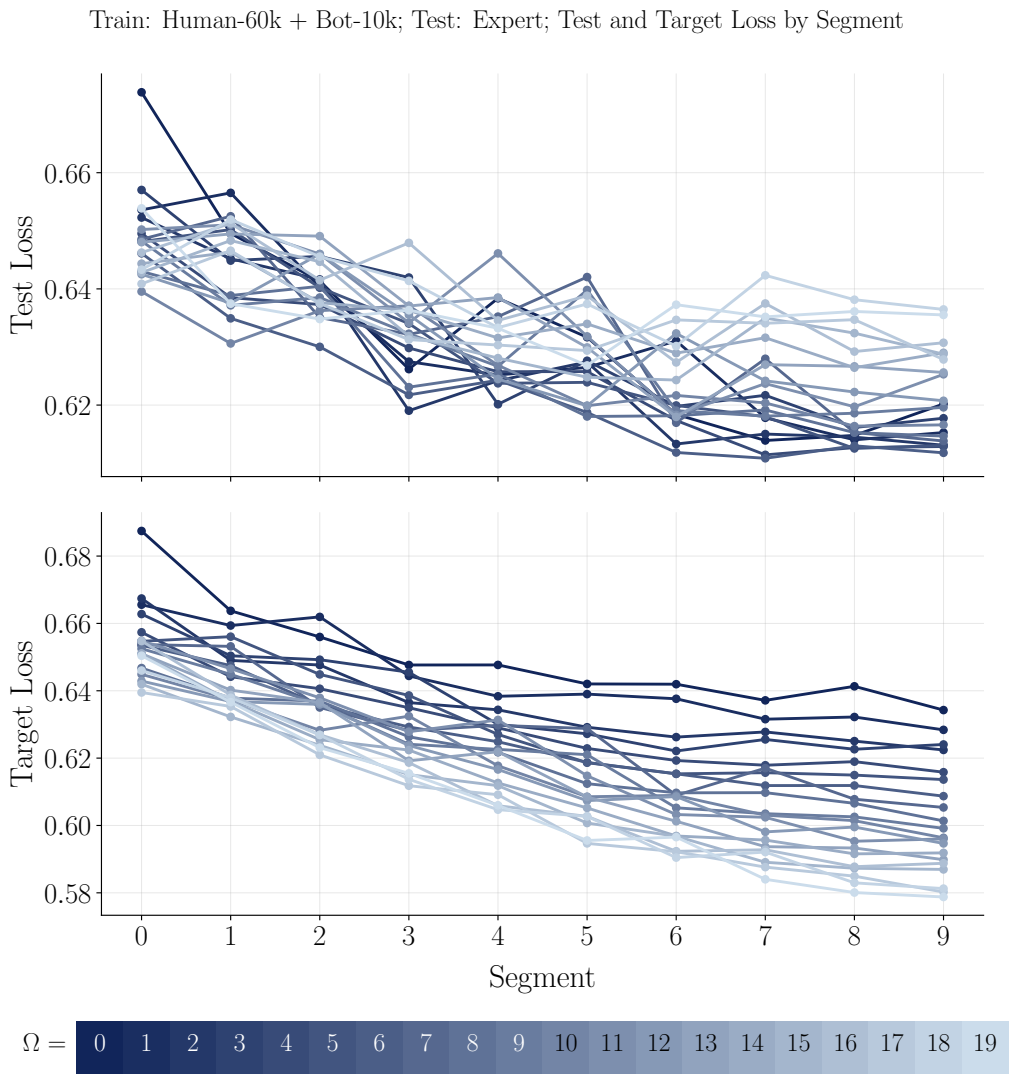


Figure 15: Test and target loss by segment for model trained on `Human-60k + Bot-10k`, targeting `Expert`. Notable overfitting to target observed.

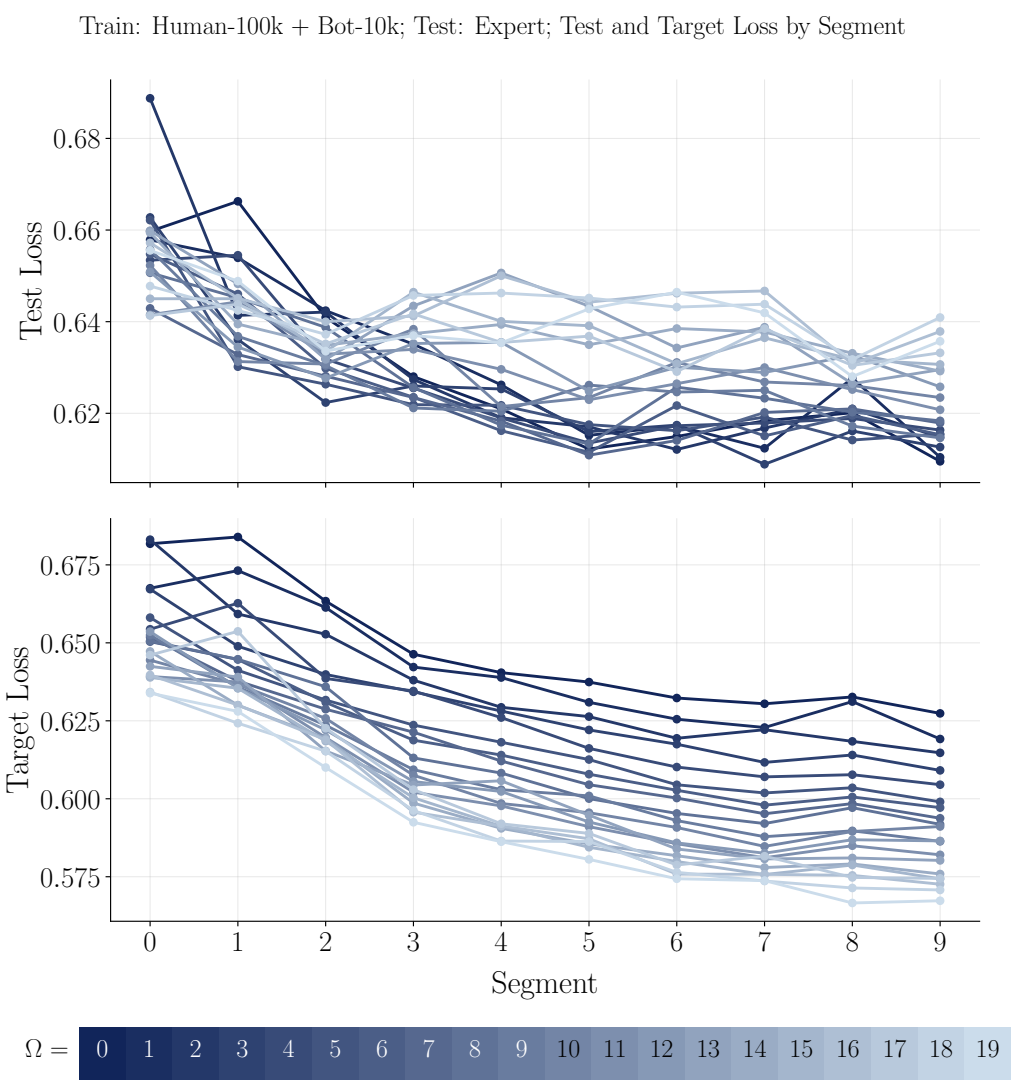


Figure 16: Test and target loss by segment for model trained on `Human-100k + Bot-10k`, targeting `Expert`. Notable overfitting to target observed.

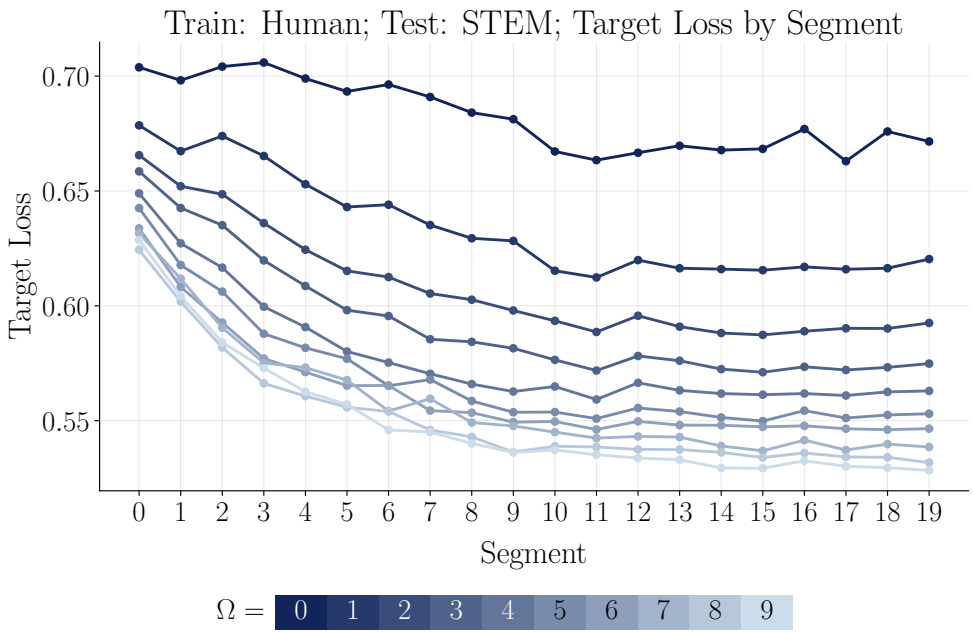


Figure 18: Target loss by segment for model trained on `Human-155k`, targeting `STEM`.

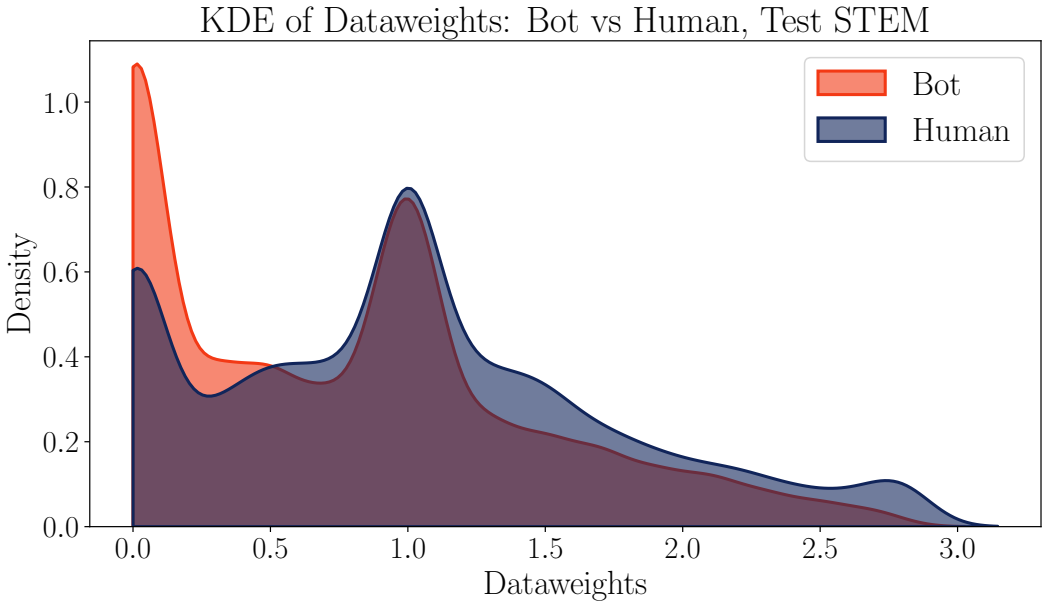


Figure 17: Kernel density estimate of learned data weights for `Bot` versus `Human` examples, targeting `STEM`.

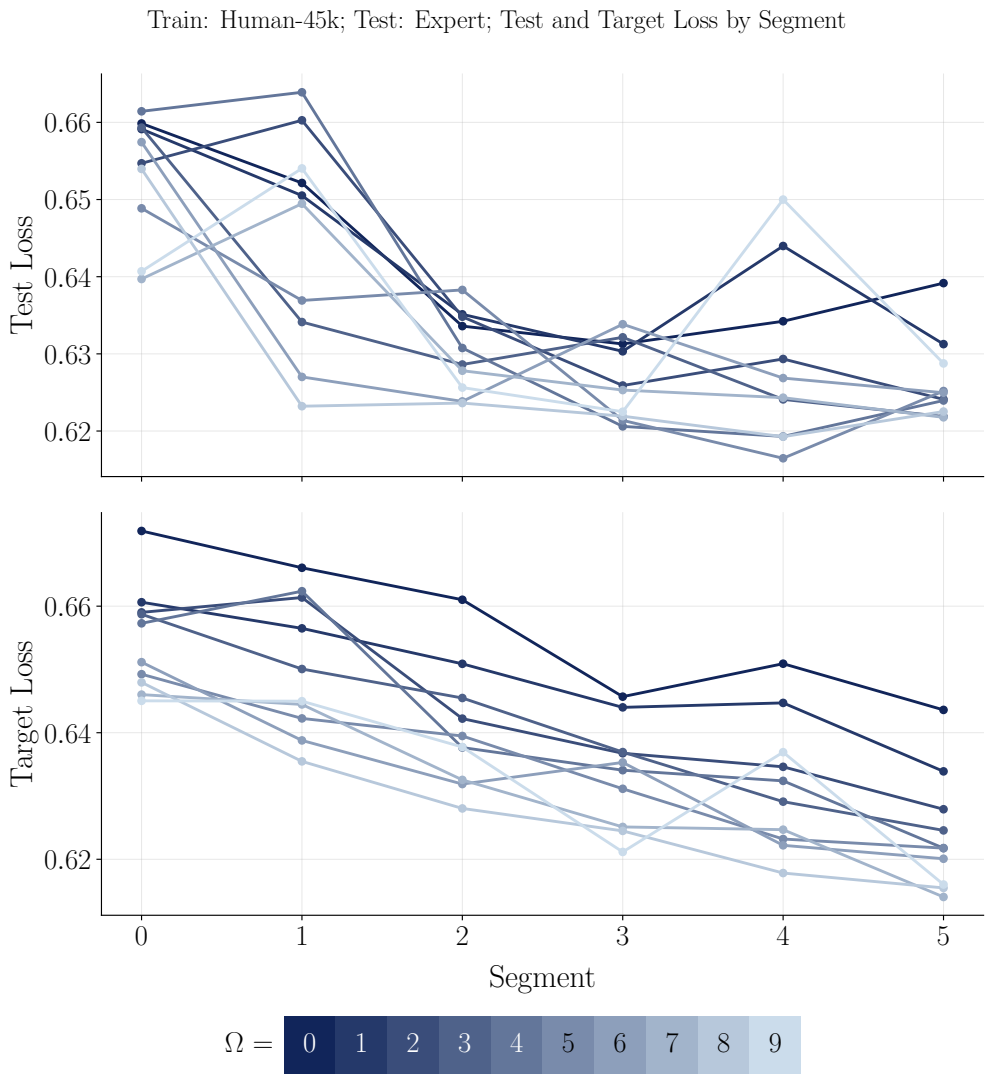


Figure 19: Test and target loss by segment for model trained on `Human-45k`, targeting `Expert`.

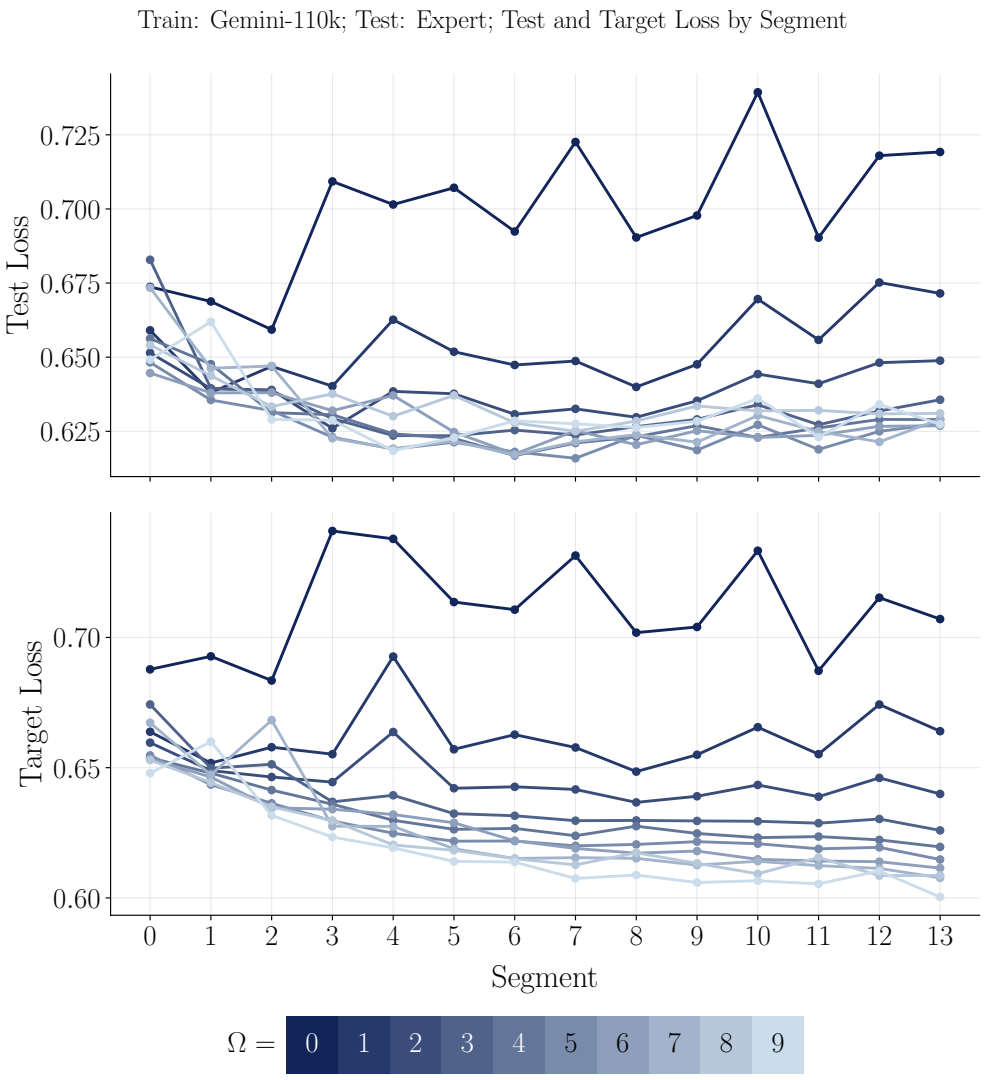


Figure 20: Test and target loss by segment for model trained on `Gemini-110k`, targeting `Expert`.

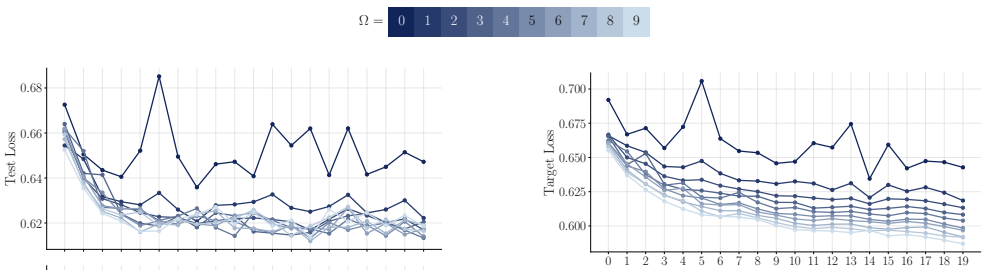


Figure 21: Test and target loss for model trained on `Human-45k` + `Gemini-110k`, targeting `Expert`.

Synthetic-data training trajectories. In particular, we notice test loss does not decrease significantly when dataweights are uniform ($\Omega = 0$). As metagradient steps are taken and dataweights are learned, test loss begins decreasing. In this example, the target and test curves are not perfectly in step, suggesting some overfitting may be occurring.

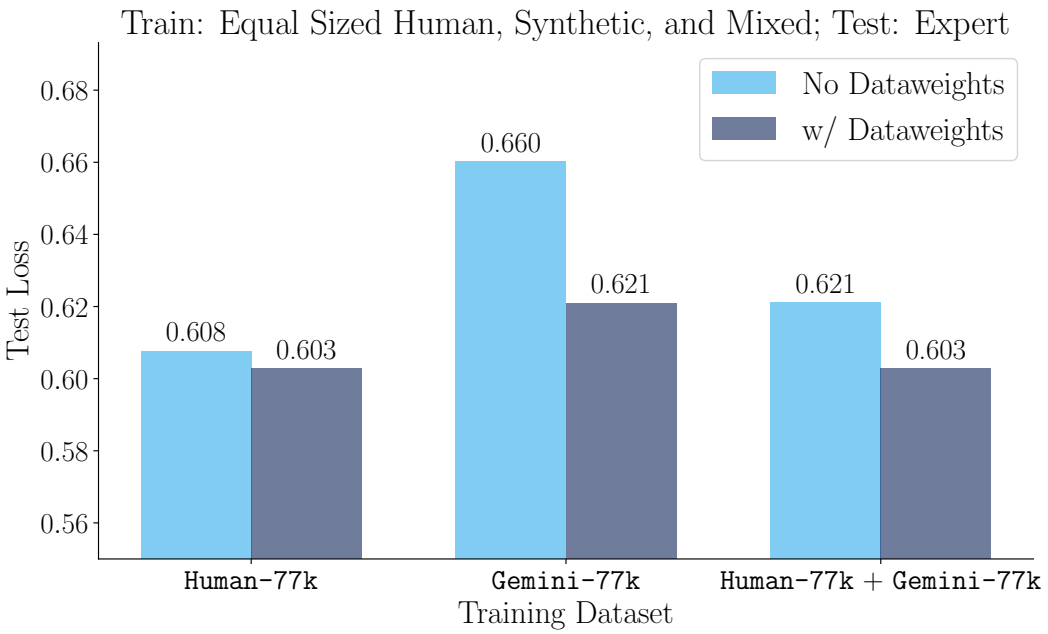


Figure 22: Test loss comparison for equal-sized `Human`, `Synthetic`, and mixed training sets, targeting `Expert`.

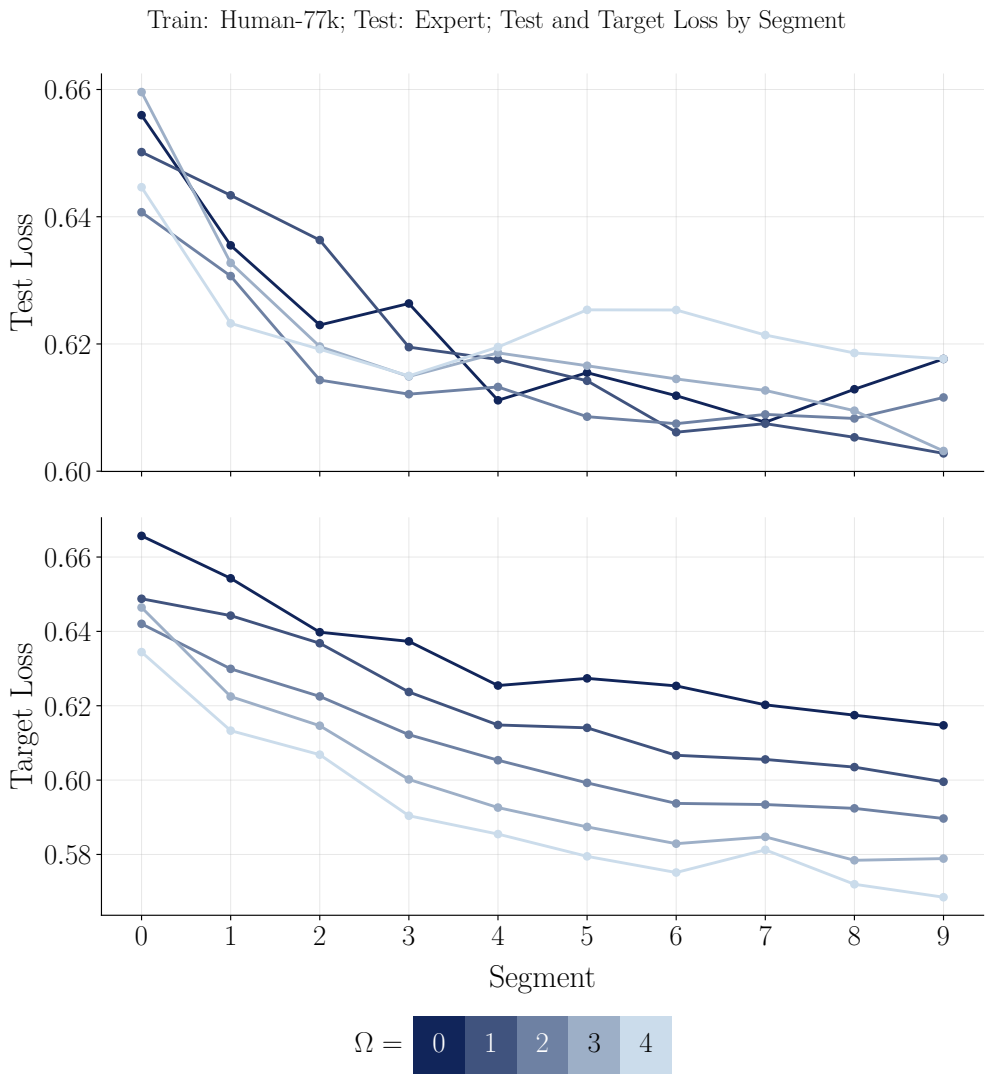


Figure 23: Test and target loss by segment for model trained on `Human-77k`, targeting `Expert`.

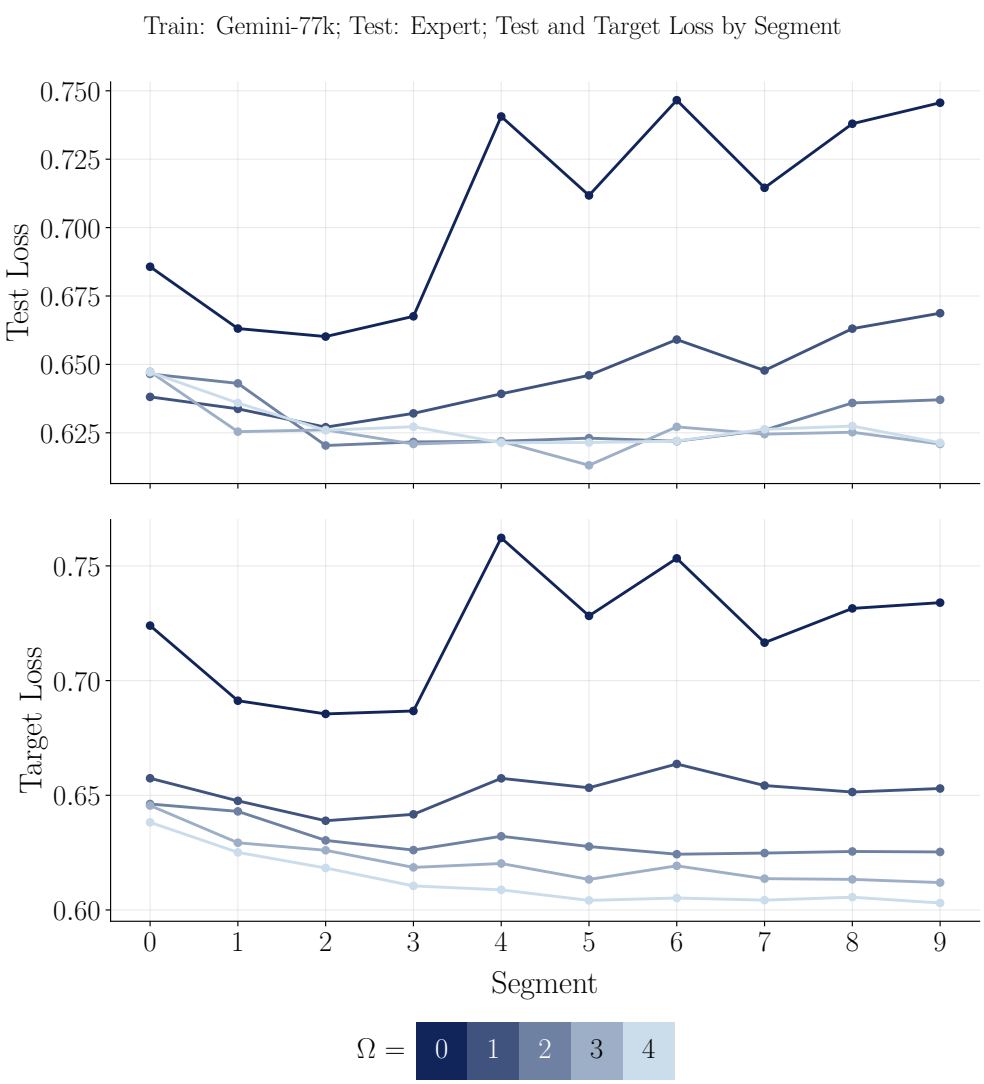


Figure 24: Test and target loss by segment for model trained on `Gemini-77k`, targeting `Expert`.

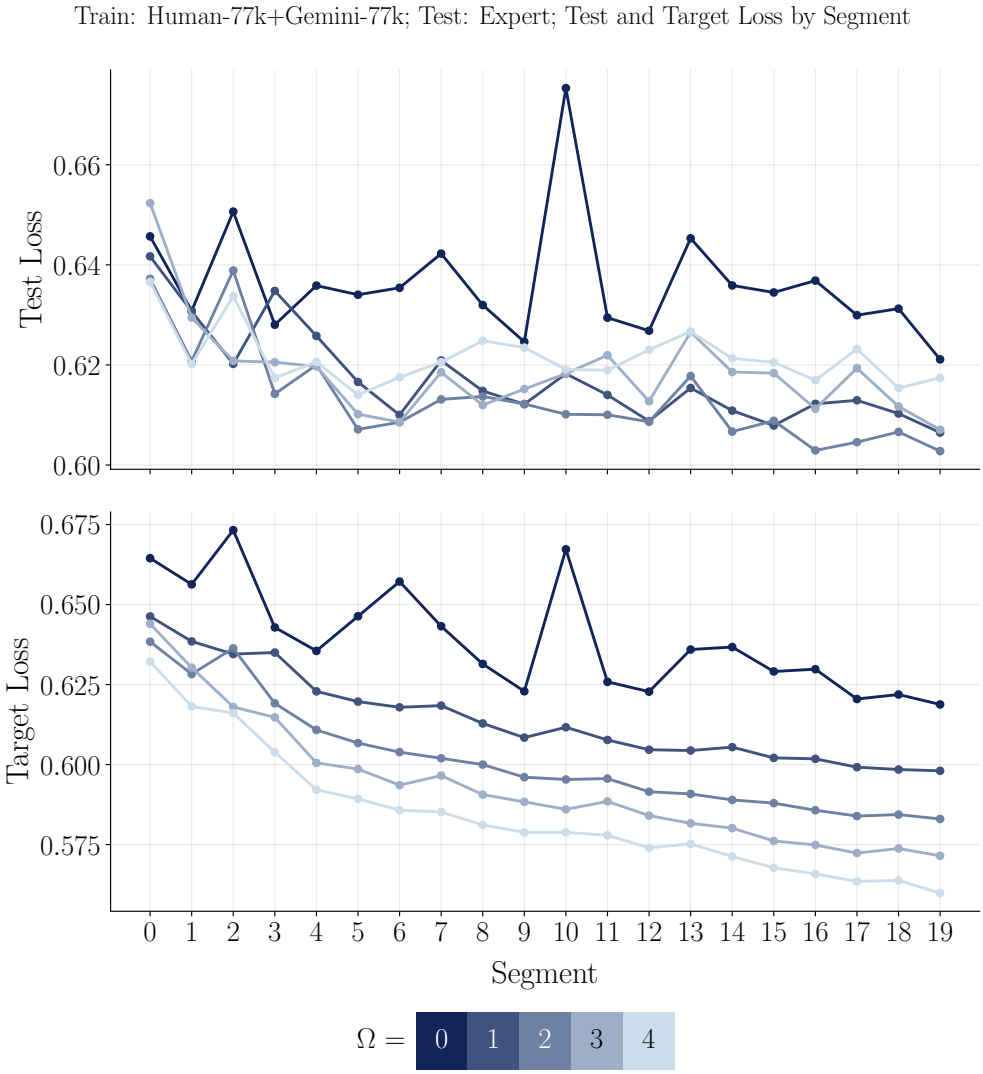


Figure 25: Test and target loss by segment for model trained on `Human-77k` + `Gemini-77k`, targeting `Expert`.

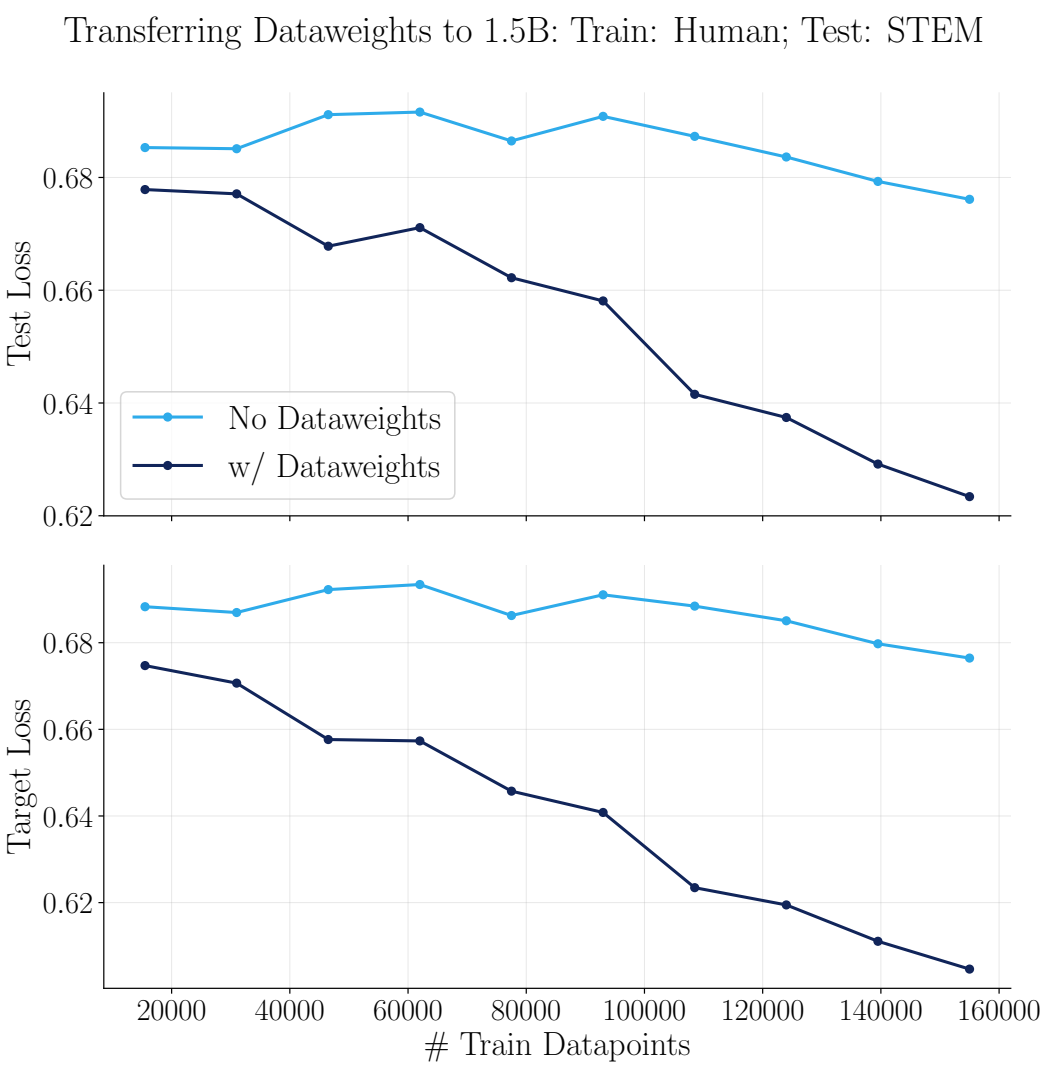


Figure 26: *Per-sample dataweights learned at 0.5B transfer to a 1.5B reward model.* Target and test loss curves on the `STEM` domain for a 1.5B reward model, with and without dataweights learned at the 0.5B scale.

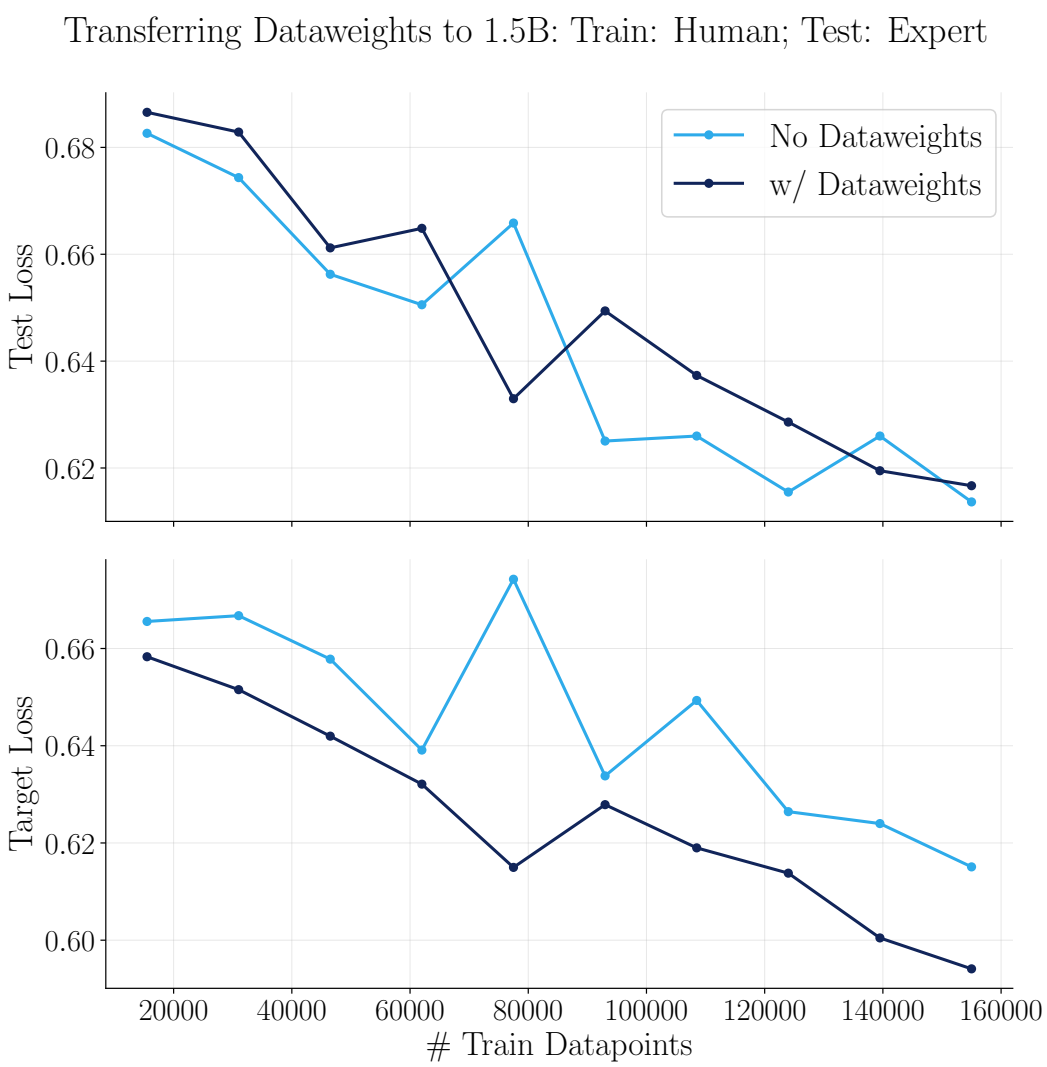


Figure 27: Transfer of learned data weights to 1.5B parameter model trained on `Human` data, targeting `Expert`.

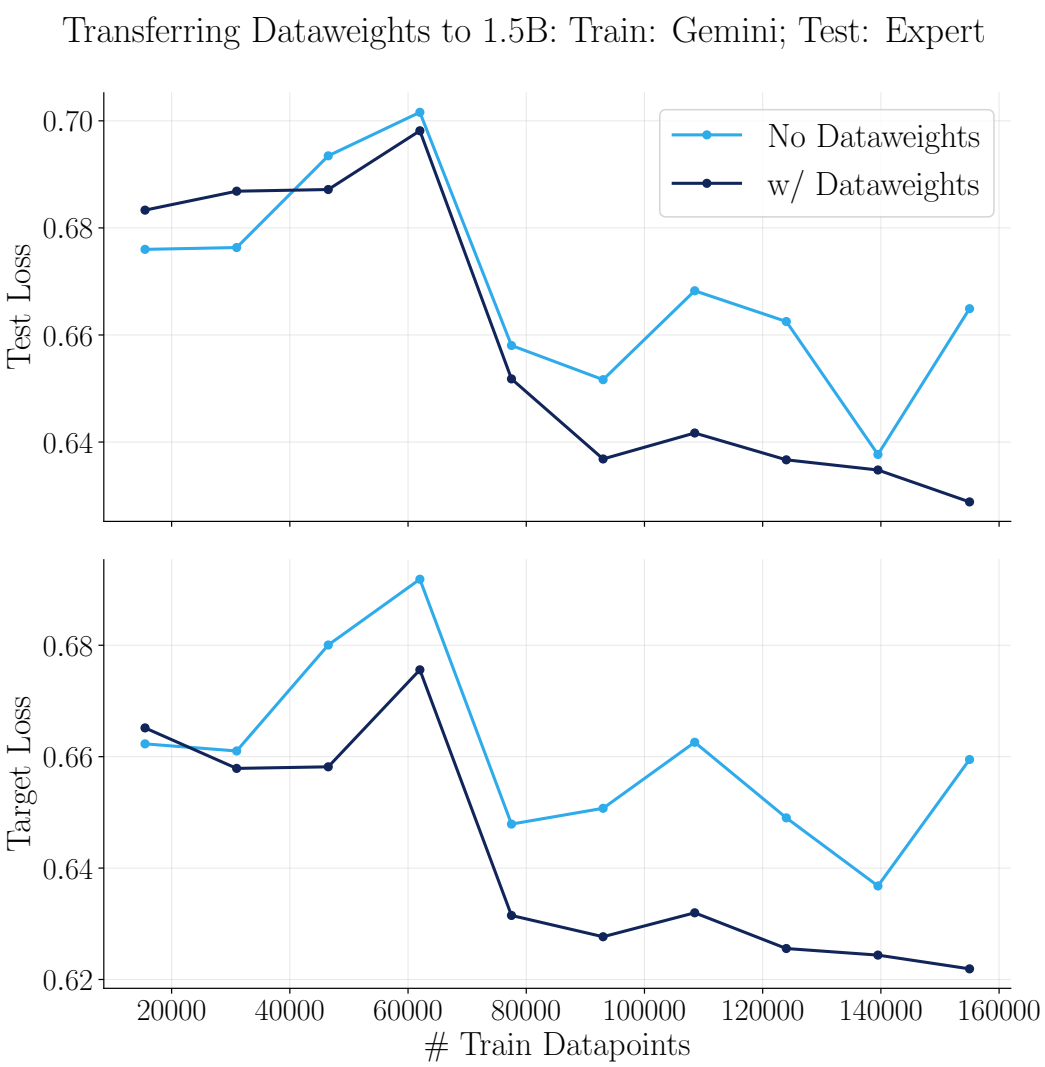


Figure 28: Transfer of learned data weights to 1.5B parameter model trained on `Gemini` data, targeting `Expert`.

D Selection-Baseline Comparison: Protocol and Additional Results

This appendix details the protocol behind Section 4.3 and reports additional pool comparisons and compute-cost accounting.

Protocol. Every scorer produces one score per training example (higher = keep) for the same pool/target pair. For each keep fraction $k \in \{10, 20, \ldots, 90, 95, 99\}$ we retain the top-scoring $k\%$ and tile the kept rows back to the original pool size, so every arm performs an identical number of optimizer steps regardless of k (selection is therefore not confounded with a reduced compute budget). Each arm trains with the same reward-model training procedure as all other experiments, split into ten segments; we select the segment with the lowest loss on the checkpoint selection set and report the loss on the disjoint held-out test split, so checkpoint selection never sees the reported metric. Every keep fraction has a random-drop control, and the unweighted baseline is replicated over three seeds to establish the noise band. Target and test splits were verified disjoint from each other and from the training pools. Individual keep-fraction points are single-seed.

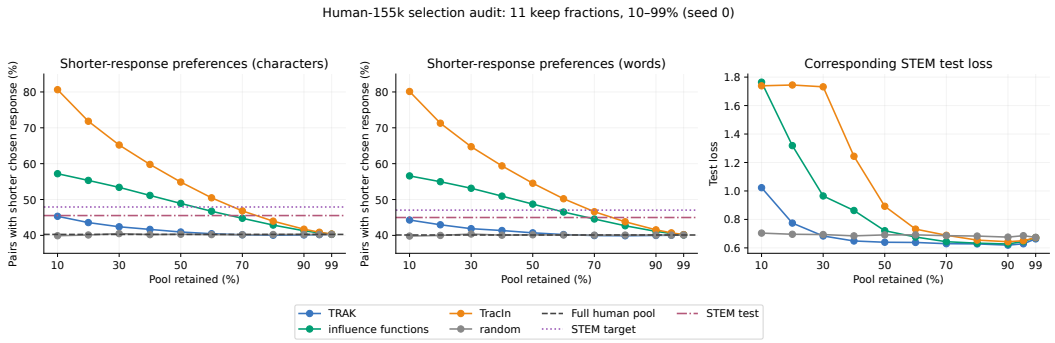


Figure 29: *Aggressive selection skews response-length preferences.* Audit of `Human-155k` subsets retained by TRAK, influence functions, TracIn, and random selection across eleven keep fractions (one seed). Left and middle: fraction of retained pairs whose chosen response is shorter than the rejected response, measured in characters and words, respectively; equal-length pairs remain in the denominator. Horizontal lines show the full human pool and the `STEM` target and test distributions. Right: corresponding test loss at the checkpoint selected using the selection set. TracIn and influence functions increasingly favor shorter-response preferences at low retention. These curves describe the selected data, rather than directly measuring the trained models' reliance on response length.

Response-length audit. We inspect the retained `Human-155k` pairs before repetition, using the preference labels to identify the chosen and rejected responses. Length is the number of characters across response turns before tokenization or truncation; responses are identified by turn position, since some `STEM` responses are stored with a user-role tag rather than an assistant-role tag. Equal-length pairs remain in the denominator. At 10% retention, the fraction preferring the shorter response is 80.6% for TracIn, 57.2% for influence functions, and 45.3% for TRAK, compared with 39.9% for random selection and 40.3% in the full pool. Whitespace-delimited word counts give the same pattern (80.1%, 56.6%, 44.3%, 39.8%, and 40.1%, respectively). At 90% retention, the character-based fractions return closer to the pool distribution: 41.7%, 41.3%, and 40.1% for TracIn, influence functions, and TRAK. Their test losses also decrease from 1.739, 1.764, and 1.023 at 10% retention to 0.643, 0.627, and 0.619 at 90%. This descriptive audit identifies a selection-induced distribution shift, not whether the original preferences were caused by length or whether length skew causes the generalization degradation.

Scorer configurations. *TRAK:* per-example gradients are projected to $1{,}024$ dimensions with a sparse Johnson–Lindenstrauss transform; scores follow the primal form $\phi_i^\top (\Phi^\top \Phi + \lambda I)^{-1} g_{\mathrm{tgt}}$, z-scored per (model, checkpoint, projection) and averaged. On the poisoned pool we ensemble 3 models $\times$ 2 checkpoints $\times$ 8 projections; on `Human-77k` a single checkpoint with 8 projections performed comparably (slightly better), indicating the checkpoint ensemble is not the active ingredient. *Instantaneous metagradient:* the one-step approximation of our method—a single forward-mode Jacobian-vector product of the per-example training gradients against the adjoint cotangent at one training step, i.e., the t-th summand of the full metagradient.

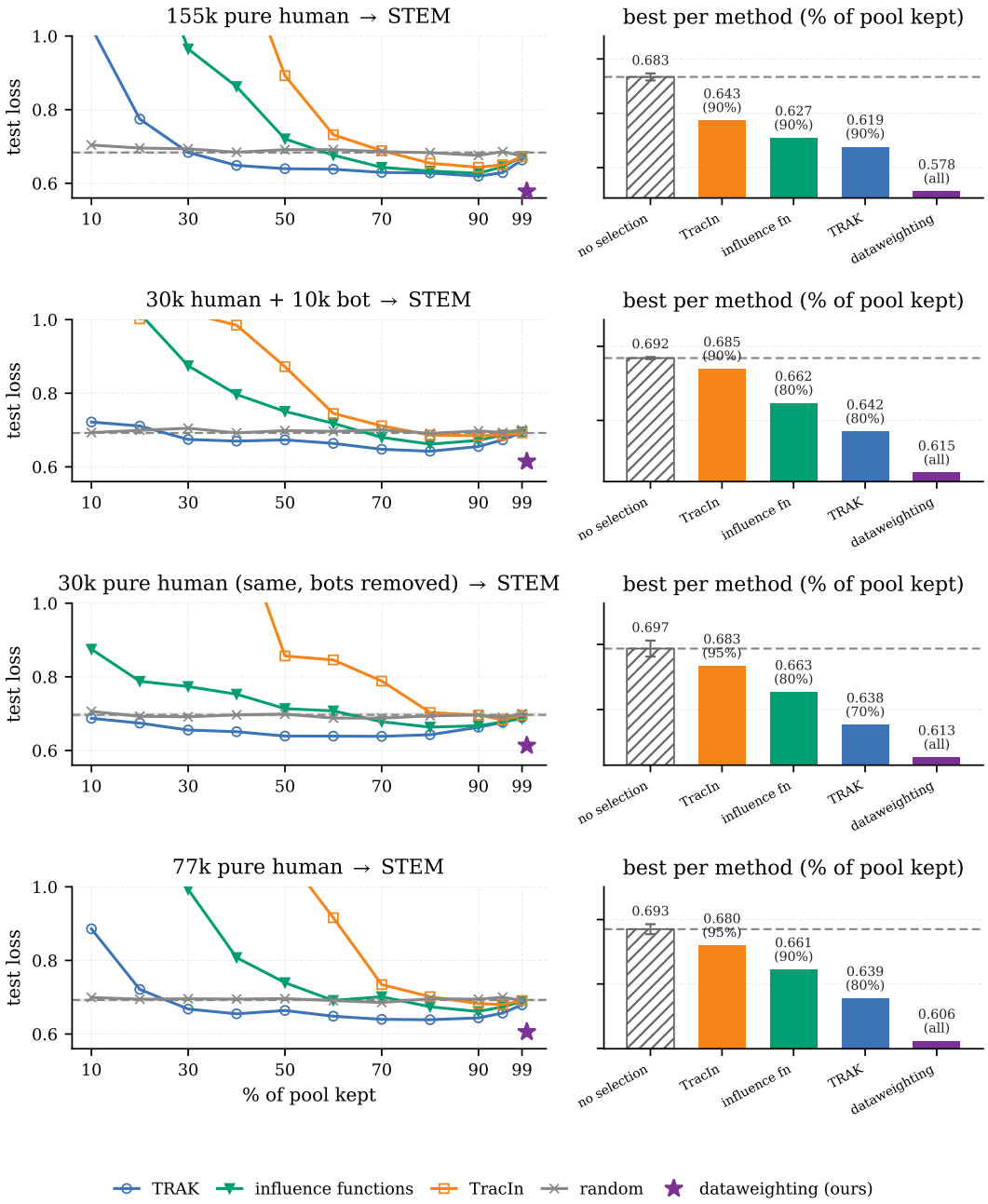


Figure 30: *The main-paper comparison, on every pool.* Figure 3 repeated for all four training pools against the `STEM` target, same protocol and same panels throughout. Rows 2–3 are a *paired* pool: `Human-30k` is the poisoned pool with its 10,000 bot rows removed and nothing else changed, so reading them adjacently separates what the contamination costs each scorer from what pool size costs it. The ordering dataweighting > TRAK > influence functions > TracIn > random holds on all four. Dataweighting's margin over the strongest baseline is $0.041/0.028/0.025/0.033$ (155k, poisoned 40k, clean 30k, 77k).

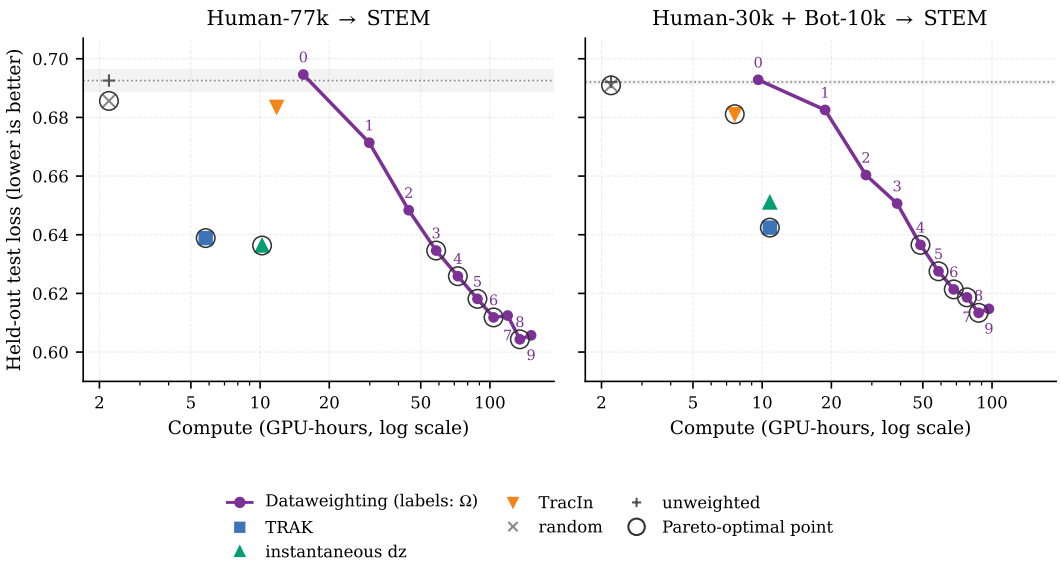


Figure 31: *Dataweighting's cost–performance trajectory on the* `STEM` *target.* Held-out test loss versus compute for two training pools; lower and left is better. Each purple point is one completed outer iteration, labelled by its zero-based index $\Omega = 0, \ldots, 9$, including the continuation run. At each budget, we report the test loss of the checkpoint with the lowest selection-set loss available so far. Baselines are shown at their best keep fraction, using the original scoring-plus-one-retrain cost estimates. Black rings mark the non-dominated points among the plotted observations. The grey band is ± 1 standard deviation over three unweighted seeds. Dataweighting requires no separate retraining stage.

Compute cost. Figure 31 shows the full outer-iteration trajectory rather than a single final result. Dataweighting costs sum logged durations of completed training segments, including checkpoint writes, multiplied by the two allocated H200 GPUs; continuation costs are cumulative. Queue gaps, initialization, and failed or replayed work are excluded. Selection costs retain the original scoring-plus-one-retrain estimates, excluding source-checkpoint generation and the keep-fraction search. These are indicative costs rather than a controlled end-to-end benchmark. Under this accounting, dataweighting first reaches the observed Pareto frontier at $\Omega = 3$ on `Human-77k` (58.3 GPU-hours, test loss 0.635) and at $\Omega = 4$ on `Human-30k+Bot-10k` (48.8 GPU-hours, 0.637). Further iterations extend the frontier toward lower test loss. All outer-iteration points are retained, including those dominated by a cheaper checkpoint or baseline.

E Extended Related Work

We discuss prior approaches to leveraging heterogeneous and noisy data sources, particularly focusing on reweighting methods and aggregating heterogeneous preferences.

Reweighting methods. Our dataweights approach relates to a number of existing studies proposing reweighting strategies for deep learning (Huang et al., 2006; Ren et al., 2018; Byrd & Lipton, 2019; Lipton et al., 2018; Wang et al., 2021), often motivated by particular models of distribution shift such as covariate shift and label shift. Most related is the work of Ren et al. (2018), who similarly assume a clean target set and aim to optimize training data weights. Compared to our work, Ren et al. (2018) derive an online reweighting rule by differentiating through a single SGD step—an approximation that degrades over realistic training trajectories and does not extend beyond vanilla SGD. We instead compute exact metagradients across long training segments (typically 5–10% of total training), enabled by recent scalable attribution methods, and support arbitrary optimizers.

Distributional robustness and domain adaptation. A central issue with heterogeneous data is that training examples may come from a mixture of environments or domains that differ from the

test distribution. A large body of work studies how to learn predictors that robustly generalize under such shifts via domain adaptation (Ben-David et al., 2006; 2010; Zhao et al., 2018), distributionally robust optimization (DRO) that optimizes worst-case performance over groups (Sagawa et al., 2019), and invariance-based objectives such as invariant risk minimization (Arjovsky et al., 2019). Rather than seeking a single representation that is invariant across environments, we directly use a small, high-quality target set to reweight heterogeneous training data toward the target distribution. Recent work in language model pretraining similarly uses robust or minimax objectives to tune mixture weights over data domains (as opposed to per-example weights) to improve downstream performance (Xie et al., 2023).

Robust statistics. Heterogeneity is often caused by corrupted observations—e.g., mislabeled examples or adversarial outliers. Robust statistics formalizes this through contamination models and offers robust estimators that take the contamination into account (Huber, 1964; Diakonikolas et al., 2019). A parallel literature in machine learning studies learning with label noise (Natarajan et al., 2013; Li et al., 2020). All these works aim to detect corruption from internal inconsistency; our work is different in that we exploit an additional, high-quality target sample that specifies the desired test distribution, and we use it to determine how much each training example should influence the learned model.

Synthetic data. An increasingly common cause of heterogeneity is mixing real and synthetic data. Synthetic data has proven useful for improving power in statistical inference (Angelopoulos et al., 2023; Huang et al., 2020), training better machine learning models in domains such as robotics (Tobin et al., 2017) and computer vision (Zhu et al., 2017), and improving efficiency in language model evaluations (Zheng et al., 2023). A key challenge with synthetic data is knowing which examples are useful and which are potentially even harmful. Our method can be seen as learning how to weight each synthetic example based on how it improves performance on the target set.

Reward modeling, RLHF, heterogeneous preferences. Reward modeling and reinforcement learning from human feedback (RLHF) operationalize "alignment" by learning a reward function from human-labeled comparisons and then optimizing a policy against that reward (Christiano et al., 2017; Ouyang et al., 2022). Humans may provide noisy feedback, or even vote adversarially. By optimizing against a clean target dataset, our method downweights such noisy and adversarial votes. At a high level, our work relates to social choice theory (Brandt et al., 2016; Fischbacher & Gächter, 2006; Ahn et al., 2003), which studies principled ways for aggregating heterogeneous preferences. Some recent works proposed data selection algorithms for reward model training (Feng et al., 2025; Shen et al., 2025); however, these approaches primarily focus on reducing data volume. In the alignment context, recent work formalizes heterogeneous preferences through latent user types and analyzes what it means to align a single policy when reward functions vary across users (Shirali et al., 2025).

Data attribution. Data attribution methods (Koh & Liang, 2017; Ghorbani & Zou, 2019; Ilyas et al., 2022; Grosse et al., 2023; Park et al., 2023) seek to quantify the influence of individual training data points on the model's downstream behavior. We build on the REPLAY algorithm (Engstrom et al., 2025; Ilyas & Engstrom, 2025) to compute exact "metagradients" of the model's performance on the clean target data with respect to dataweights, allowing us to automatically discover—and thus downweight—noisy and irrelevant parts of the training data.

F Stylized Theoretical Results for Dataweighting

This appendix gives the formal results underlying Section 5. The constructions isolate three benefits of target-guided dataweighting: correcting systematic synthetic-label bias, filtering severe corruption, and adapting the training distribution to a specialized target. They concern global optimization over fixed-size subsets, a restricted family of dataweights; they do not assert that a particular metagradient algorithm finds the optimum.

F.1 Shared model and learning procedures

An input has two signal features and D weak nuisance features:

$$x = \big(g, \sqrt{\lambda/D}\, h\big), \qquad h \sim \mathcal{N}(0, I_D), \qquad \lambda > 0. \tag{5}$$

Source inputs have $g \sim \mathcal{N}(0, I_2)$, while target inputs have $g \sim \mathcal{N}(0, T)$ for a positive-definite matrix T. In both domains, g and h are independent. The true response is $f^*(x) = g^\top \theta^*$, with full coefficient vector $\beta^* = (\theta^*, 0)$. There are n independent source examples, whose labels are specified in each construction, and m independent target examples with true labels. The two samples are independent, and $D + 2 > n + m$. Let X and Z denote the source and target design matrices, respectively, and let y and t be their label vectors.

Dataweighting. Fix $1 \le k \le n$. For each k-element source subset S, train on squared loss by gradient flow from zero to convergence. Since X_S has full row rank almost surely, this produces the minimum-norm interpolator $\beta_S = X_S^\dagger y_S$, where $\dagger$ denotes the Moore–Penrose inverse. Select a subset minimizing $m^{-1}\|Z\beta_S - t\|^2$, uniformly at random among ties, and return its interpolator β_{RW}. This is equivalent to optimizing dataweights $w_i = k^{-1}\mathbf{1}\{i \in S\}$ over all such subsets. No subsequent fine-tuning is performed for this procedure.

Baselines. The baseline FT pretrains from zero on all n source examples, then fine-tunes all parameters on target squared loss by gradient flow to convergence. The baseline $\mathrm{random} + \mathrm{FT}$ instead pretrains on a uniformly random k-subset, drawn independently of the data values, and applies the same fine-tuning procedure. Thus both baselines can use the same target sample that dataweighting uses for subset selection.

Risk and order of limits. For a parameter vector $\beta = (\theta, v)$, target population prediction risk is

$$R_D(\beta) = (\theta - \theta^*)^\top T(\theta - \theta^*) + \frac{\lambda}{D}\|v\|^2. \tag{6}$$

Expectations include sampling, random subset selection, and tie-breaking. For each procedure A, define

$$\mathcal{R}_A(n) = \limsup_{D \to \infty} \mathbb{E} R_D(\beta_A). \tag{7}$$

In the asymptotic comparisons, D tends to infinity with n, m fixed before n grows; m remains fixed throughout. The nuisance scale λ is fixed in Theorem F.1 and equals k in Theorem F.3. Strict gaps in these iterated limits imply the corresponding ordering for sufficiently large finite n and then sufficiently large finite D. The filtering result instead works directly at fixed finite n, m, D.

F.2 Reduction to two dimensions and validation control

Write the designs as $X = [G, \sqrt{\lambda/D}\, H]$ and $Z = [V, \sqrt{\lambda/D}\, J]$. The rows of G are standard Gaussian, and those of V have distribution $\mathcal{N}(0, T)$. At fixed n, m, the law of large numbers gives, almost surely,

$$D^{-1}HH^\top \longrightarrow I_n, \qquad D^{-1}JJ^\top \longrightarrow I_m, \qquad D^{-1}JH^\top \longrightarrow 0. \tag{8}$$

The first two coordinates of each β_S therefore converge to

$$\theta_S = G_S^\top (G_S G_S^\top + \lambda I_k)^{-1} y_S = (G_S^\top G_S + \lambda I_2)^{-1} G_S^\top y_S. \tag{9}$$

The nuisance coefficients have bounded norm in this limit, so their contribution to (6) vanishes. The target predictions converge to $V\theta_S$. The effective ridge penalty in (9) arises from minimum-norm interpolation in the weak features; it is not an explicit penalty added to the training objective.

For fine-tuning from a source-trained initialization β_0, gradient flow on target squared loss gives

$$\beta_{\mathrm{FT}} = \beta_0 + Z^\dagger(V\theta^* - Z\beta_0). \tag{10}$$

Applying the same Gram-matrix limits, and writing θ_0 for the limiting signal coordinates of the initialization, yields

$$\theta_{\mathrm{FT}} - \theta^* = \lambda(V^\top V + \lambda I_2)^{-1}(\theta_0 - \theta^*). \tag{11}$$

Expectations and data-dependent selection. At fixed n, there are finitely many subsets, so these convergences hold simultaneously over all candidates. To justify expectation limits, let H_- be H with its first column removed. It is independent of the source labels in all three constructions below. For sufficiently large D, uniformly over subsets S,

$$\|\beta_S\|^2 \le \frac{D\|y\|^2}{\lambda\lambda_{\min}(H_-H_-^\top)}. \tag{12}$$

Indeed, $X_SX_S^\top \succeq (\lambda/D)(H_-)_S(H_-)_S^\top$, and the smallest eigenvalue of this principal submatrix is at least that of $H_-H_-^\top$. For fixed n, the normalized inverse Gaussian Gram matrix $D(H_-H_-^\top)^{-1}$ has bounded moments of any fixed order for all sufficiently large D. One way to see this is to bound its operator norm by its trace: each inverse diagonal entry is the reciprocal of a Gaussian regression residual sum of squares, distributed as χ^2_{D-n}. The scaled reciprocal moments are bounded as D grows. The labels have moments of every order and are independent of H_-. Thus (12) gives uniform integrability of the risks, including that of a validation-selected subset. Fine-tuning satisfies

$$\beta_{\mathrm{FT}} - \beta^* = (I - Z^\dagger Z)(\beta_0 - \beta^*), \tag{13}$$

an orthogonal projection of the initialization error, so the same moment control applies to both fine-tuning baselines.

A validation bound independent of the number of subsets. Let $R_T(\theta) = (\theta - \theta^*)^\top T(\theta - \theta^*)$. For any reference subset S_0 chosen using only source data, with a definition independent of D, and for $m \ge 4$,

$$\mathcal{R}_{\mathrm{RW}}(n) \le C_m\,\mathbb{E}R_T(\theta_{S_0}), \qquad C_m = \frac{4(m-1)}{m-3} \le 12. \tag{14}$$

To prove this, write $V = UT^{1/2}$, with U an $m \times 2$ standard Gaussian matrix, and put $W = U^\top U$. In the limiting model, empirical optimality and the extreme-eigenvalue bounds for the validation quadratic form imply

$$R_T(\theta_{\mathrm{RW}}) \le \frac{\lambda_{\max}(W)}{\lambda_{\min}(W)} R_T(\theta_{S_0}). \tag{15}$$

This inequality holds for every minimizing subset. For the finite-D selector, uniform convergence over the finitely many candidates gives the same upper bound on the limit superior of its population risk, even if the minimizing subset changes with D. The preceding uniform integrability then justifies the expectation bound.

For completeness, the expected condition number is bounded by

$$\mathbb{E}\frac{\lambda_{\max}(W)}{\lambda_{\min}(W)} \le \mathbb{E}\big[\operatorname{tr}(W)\operatorname{tr}(W^{-1})\big] = \frac{4(m-1)}{m-3}. \tag{16}$$

The Schur complement shows that each diagonal entry of W^{-1} has mean $1/(m-3)$: its reciprocal is a χ^2_{m-1} residual sum of squares. Also, $s = \operatorname{tr} W \sim \chi^2_{2m}$ is independent of W/s, because the radius and direction of the Gaussian matrix U are independent. Since $\mathbb{E}s^{-1} = 1/(2m-2)$ and $\mathbb{E}\operatorname{tr}(W^{-1}) = 2/(m-3)$, this gives the equality in (16). Independence of source and target data now proves (14). The bound depends on the effective two-dimensional prediction problem, rather than the number of candidate subsets.

F.3 Correcting small synthetic-label bias

Take $T = I_2$, $\theta^* = e_1$, and synthetic source labels from

$$\widehat{f}(x) = g_1 + \delta \operatorname{sign}(g_2)\mathbf{1}\{h_1 > 0\}. \tag{17}$$

The teacher is correct on half the source inputs and has error of magnitude δ on the rest. The nuisance scale is fixed.

Theorem F.1 (Bias correction). *Fix $0 < \alpha < 1/2$, $m \geq 4$, and $\lambda, \delta > 0$, and set $k = \lfloor \alpha n \rfloor$. Then*

$$\begin{aligned} \mathcal{R}_{\mathrm{RW}}(n) &\longrightarrow 0, \\ \liminf_{n\to\infty} \min\{\mathcal{R}_{\mathrm{FT}}(n), \mathcal{R}_{\mathrm{random+FT}}(n)\} &\geq c_{\lambda,m}\delta^2 > 0, \end{aligned} \tag{18}$$

where $c_{\lambda,m} = (2\pi)^{-1}[\lambda/(\lambda+m)]^2$ is valid. The two baselines have the same limiting risk.

Proof. Put $b = \delta/\sqrt{2\pi}$. Independence and symmetry give

$$\mathbb{E}[g_1(\widehat{f} - f^*)] = 0, \qquad \mathbb{E}[g_2(\widehat{f} - f^*)] = \frac{\delta}{2}\mathbb{E}|g_2| = b. \tag{19}$$

The law of large numbers in (9) therefore gives $\theta_{\mathrm{gen}} \to (1, b)$ for full-data pretraining. A uniform random k-subset has the distribution of k fresh source examples, so $\theta_{\mathrm{random}} \to (1, b)$ as well.

These limits hold in mean square. To verify this, consider an initialization trained on N examples, where $N = n$ or $N = k$. On $G^\top G \succeq (N/2)I_2$, its coefficient error relative to e_1 is bounded by a constant depending only on δ. The complementary event has exponentially small probability by Gaussian sample-covariance concentration. For every design, the ridge map satisfies

$$\|(G^\top G + \lambda I_2)^{-1} G^\top\|_{\mathrm{op}} \leq \frac{1}{2\sqrt{\lambda}}, \qquad \|\theta - e_1\| \leq 1 + \frac{\delta\sqrt{N}}{2\sqrt{\lambda}}. \tag{20}$$

The first bound follows by maximizing $s/(s^2+\lambda)$ over singular values $s \geq 0$; the second uses that the label-error vector has norm at most $\delta\sqrt{N}$. Thus the exceptional event contributes vanishing mean-square error, while bounded convergence on the well-conditioned event handles the rest.

For the dataweighting comparator, choose the first k clean observations if they exist, and the first k observations otherwise. Cleanliness depends only on h_1, so conditional on having enough clean observations, their signal features remain independent standard Gaussians. On this event,

$$\theta_{S_0} - e_1 = -\lambda(G_{S_0}^\top G_{S_0} + \lambda I_2)^{-1} e_1, \tag{21}$$

whose norm is at most one and converges to zero. The probability of having fewer than k clean examples is at most $e^{-c_\alpha n}$, where $c_\alpha = 2(1/2 - \alpha)^2$. On this exceptional event the preceding deterministic bound with $N = k$ applies. Consequently, $\mathbb{E}R_{I_2}(\theta_{S_0}) \to 0$, and (14) proves the claim for dataweighting. The clean subset is used only as a comparator in the proof; the selection procedure is not given corruption indicators.

Finally, rotational invariance of V in (11) gives, for every source-only initialization,

$$\begin{aligned} \mathbb{E}_V\|\theta_{\mathrm{FT}} - e_1\|^2 &= q_{\lambda,m}\|\theta_0 - e_1\|^2, \\ q_{\lambda,m} &= \frac{1}{2}\mathbb{E}\,\mathrm{tr}\big[\lambda^2(V^\top V + \lambda I_2)^{-2}\big]. \end{aligned} \tag{22}$$

Convexity of $t \mapsto \lambda^2/(\lambda+t)^2$, together with $\mathbb{E}\,\mathrm{tr}(V^\top V) = 2m$, yields $q_{\lambda,m} \geq [\lambda/(\lambda+m)]^2$. Both baseline risks therefore converge to $q_{\lambda,m}\delta^2/(2\pi)$, which proves the stated lower bound. □

F.4 Filtering an unknown corrupted set

Use the same teacher as in (17), with error magnitude M in place of δ. A source example is corrupted when $h_1 > 0$.

Theorem F.2 (Filtering). *Fix finite n, m, D with $m \geq 1$ and $D + 2 > n + m$, and fix $\lambda > 0$. Let $0 < \alpha < 1/2$ and $k = \lfloor \alpha n \rfloor \geq 1$. Then*

$$\begin{aligned} \lim_{M\to\infty} \mathbb{P}\{\textit{the selected subset is entirely clean}\} &= \mathbb{P}\{\mathrm{Bin}(n, 1/2) \geq k\} \\ &\geq 1 - e^{-c_\alpha n}, \qquad c_\alpha = 2(1/2 - \alpha)^2. \end{aligned} \tag{23}$$

Conditional on the existence of a clean k-subset, the filtering probability tends to one.

Proof. Couple all values of M using the same source and target inputs. For each subset, write

$$r_i = \mathrm{sign}(g_{i2})\mathbf{1}\{h_{i1} > 0\}, \qquad \beta_S(M) = \beta_S(0) + Mu_S, \qquad u_S = X_S^\dagger r_S. \tag{24}$$

If S contains corruption, $r_S \neq 0$ almost surely. Full row rank of X_S then implies $u_S \neq 0$. Conditional on the source sample, a target input has strictly positive variance in direction u_S, since its covariance matrix $\mathrm{diag}(I_2, (\lambda/D)I_D)$ is positive definite. Thus $\|Zu_S\|^2 > 0$ almost surely, even for $m = 1$. The subset's validation loss is a quadratic in M with positive leading coefficient and diverges as $M \to \infty$.

A clean subset's validation loss is independent of M. Since there are finitely many candidate subsets, whenever a clean k-subset exists, every minimizer is clean for all sufficiently large M, almost surely. If fewer than k clean examples exist, selecting an entirely clean subset is impossible. Bounded convergence proves the equality in (23). The number of clean examples is $\mathrm{Bin}(n, 1/2)$; the binomial Hoeffding bound gives the displayed exponential bound. □

F.5 Covariate shift with correct labels

Both source and target labels are now correct. Let $T = \mathrm{diag}(1, \varepsilon)$, so that the second signal feature matters less for target prediction, and let the nuisance scale grow with source sample size. Fix

$$\rho = \mathbb{E}|g_1 g_2| = \frac{2}{\pi}, \qquad L = \sqrt{\rho^2 + 4}, \qquad \theta^* = \frac{1}{L}\begin{pmatrix}\rho \\ 2\end{pmatrix}. \tag{25}$$

In particular, θ^* is a unit vector independent of α and ε.

Theorem F.3 (Covariate shift). *Fix $0 < \alpha < 1/2$, $m \geq 4$, and $0 < \varepsilon \leq 1$. Set $k = \lfloor \alpha n \rfloor$ and $\lambda = k$, and use the parameter vector in* (25)*. Then*

$$\begin{aligned} \limsup_{n\to\infty} \mathcal{R}_{\mathrm{RW}}(n) &\leq 3\varepsilon, \\ \liminf_{n\to\infty} \min\{\mathcal{R}_{\mathrm{FT}}(n), \mathcal{R}_{\mathrm{random+FT}}(n)\} &\geq \frac{\alpha^2}{25}. \end{aligned} \tag{26}$$

Thus dataweighting has strictly smaller limiting risk when $\varepsilon < \alpha^2/75$. For the same θ^* and procedures with $T = I_2$,

$$\liminf_{n\to\infty} \mathcal{R}_{\mathrm{RW}}(n) \geq \left(\frac{\alpha}{1+\alpha}\right)^2 = \lim_{n\to\infty} \mathcal{R}_{\mathrm{FT}}(n). \tag{27}$$

Proof. Since $\lambda/n \to \alpha$ and $\lambda/k = 1$, the source ridge reduction gives

$$\theta_{\mathrm{gen}} \longrightarrow \frac{\theta^*}{1+\alpha}, \qquad \theta_{\mathrm{random}} \longrightarrow \frac{\theta^*}{2}. \tag{28}$$

With fixed m and $\lambda = k \to \infty$, the correction matrix $\lambda(V^\top V + \lambda I_2)^{-1}$ in (11) tends to I_2. For correct labels, every source ridge error has Euclidean norm at most $\|\theta^*\| = 1$, and the fine-tuning correction also has operator norm at most one. Because $T \preceq I_2$, bounded convergence therefore yields the exact baseline limits

$$\begin{aligned} \mathcal{R}_{\mathrm{FT}}(n) &\longrightarrow \left(\frac{\alpha}{1+\alpha}\right)^2 \frac{\rho^2 + 4\varepsilon}{L^2}, \\ \mathcal{R}_{\mathrm{random+FT}}(n) &\longrightarrow \frac{1}{4}\frac{\rho^2 + 4\varepsilon}{L^2}. \end{aligned} \tag{29}$$

For a reference subset, choose the first k source observations satisfying $g_1 g_2 > 0$, or any k-subset if too few exist. Each source observation satisfies this condition with probability $1/2$, so enough exist with probability tending to one exponentially fast. Their conditional second-moment matrix is

$$B = \mathbb{E}[gg^\top \mid g_1 g_2 > 0] = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}. \tag{30}$$

Consequently, the reference error converges to

$$\theta_{S_0} - \theta^* \longrightarrow -(B + I_2)^{-1}\theta^* = -\frac{1}{L}\begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{31}$$

The last identity follows from $(B + I_2)(0, 1)^\top = (\rho, 2)^\top$. The error norm is bounded by one, including when the reference falls back to an arbitrary subset, so its expected target risk converges to ε/L^2. Applying (14) gives

$$\limsup_{n\to\infty} \mathcal{R}_{\mathrm{RW}}(n) \le \frac{C_m \varepsilon}{L^2} \le 3\varepsilon. \tag{32}$$

The smaller baseline limit in (29) is that of full-data fine-tuning, and it is at least

$$\left(\frac{\alpha}{1+\alpha}\right)^2 \frac{\rho^2}{L^2} = \frac{\alpha^2}{(1+\alpha)^2(1+\pi^2)} \ge \frac{\alpha^2}{25}, \tag{33}$$

using $0 < \alpha < 1/2$.

Finally, set $T = I_2$, $A_S = G_S^\top G_S/n$, and $a_n = k/n$. Every subset satisfies $A_S \preceq A_{\mathrm{all}} = G^\top G/n$, with $A_{\mathrm{all}} \to I_2$, and its limiting signal error is

$$\theta_S - \theta^* = -a_n(A_S + a_n I_2)^{-1}\theta^*. \tag{34}$$

Since $\|\theta^*\| = 1$, the smallest eigenvalue of the error map gives, uniformly over all subsets,

$$\|\theta_S - \theta^*\|^2 \ge \left(\frac{a_n}{\lambda_{\max}(A_{\mathrm{all}}) + a_n}\right)^2 \longrightarrow \left(\frac{\alpha}{1+\alpha}\right)^2. \tag{35}$$

Uniformity makes this lower bound applicable to validation-selected subsets as well. Taking the finite-D limit and then using Fatou's lemma as $n \to \infty$ proves the left inequality in (27). Equation (29) at $\varepsilon = 1$ gives the matching full-data baseline limit. Thus the advantage in this construction comes from adapting the source subset to target anisotropy, rather than from discarding correctly labeled observations indiscriminately. □