AutoMixer: Checkpoint Artifacts as Automatic Data Mixers

Ernie Chang[♠*] **Yang Li**[*†] **Patrick Huber**[♠] **Vish Vogeti**[♠] **David Kant**[♠]
Yangyang Shi[♠] **Vikas Chandra**[♠]
[♠]AI at Meta
[†]Iowa State University
erniecyc@meta.com, yangli1@iastate.edu

Abstract

In language model training, it is desirable to equip models with capabilities from various tasks. However, it is not clear how to directly obtain the right data mixtures for these capabilities as the relationship between data and tasks is difficult to be modeled. In this work, we observe that checkpoint models exhibit emerging capabilities at different points in the training trajectory. Often, the training process saves checkpoints as artifacts that are under-utilized as a source of in-training data signals. We identify these artifact models based on their respective capabilities on the benchmarks and leverage them as data mixers by using their aggregated first-order influence approximation over source data (See Figure 1). We demonstrated on eight reasoning benchmarks that the proposed framework shows significant improvements in the pretraining setting, with performance improvements of up to 1.93%. Overall, this shows the potential of checkpoint models to enhance data quality and optimize data mixtures.

1 Introduction

Training effective language models involves equipping them with a diverse set of skills, which is heavily influenced by the composition of their training data. A primary challenge in this domain is the precise identification of task-specific data within a diverse mixture (Ye et al., 2024)—an undertaking that becomes increasingly complex as the number of tasks grows and direct domain matches (or dataset-task mapping) are absent (Gadre et al., 2024). This complexity is further compounded by overlapping or conflicting knowledge regions across data domains, complicating the discernment of the most relevant samples for each task (Sedinkina and Schütze, 2019). Traditional methods often overlook this perspective, leading to inefficient data utilization and missed opportunities (Wu et al., 2022).

[*]These authors contributed equally to this work.

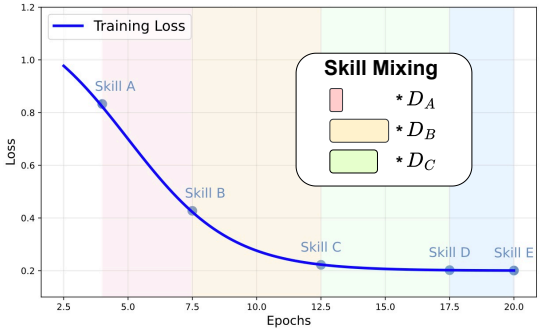

Figure 1: Illustration of the checkpoint selection process and subsequent sampling. We leverage intermediate model checkpoints to group and sample data for targeted skill acquisition.

In this work, we address this challenge as a two-fold problem: (1) identifying data mixtures or groups that can define effective divisions between data, and (2) assigning sampling weights to each of these groups to better model desirable behaviors during the training process.

Fundamentally, task data are often ill-defined – it is not clear how best to assemble a dataset for cultivating a certain skillset (Wei et al., 2022; Hu et al., 2023). This "chicken-and-egg" problem means that identifying the data leading to skill improvements requires a priori knowledge of which data benefits that skill. One could consider training a multitude of data mixtures to observe performance trends and then isolate the best data for each skill, but such brute-force approaches are computationally infeasible as models grow larger.

Therefore, the core missing piece is a direct modeling of the relationship between datasets and model parameters, because data quality cannot be reliably assessed in isolation from training (Park et al., 2023). A potential solution involves computing the influence function (Hampel, 1974; Halevy et al., 2009), which estimates the first-order "alignment" between training samples and specific skills.

However, models inherently progress beyond these approximations, so data composition guided solely by step-t influence calculations may become outdated as training advances.

In this work, we propose to tackle data mixing by regrouping raw data based on observed capabilities and then assigning data loading probabilities for these groups. Skills acquired at one checkpoint may not persist throughout the entire process, making it difficult to identify a single training step that captures all optimal capabilities. Hence, we simulate training runs with proxy models and trace checkpoint artifacts that align with target tasks. We then estimate the sample influence (Kwon et al., 2024; Yeh et al., 2022) for each checkpoint; these influence scores are consolidated to guide both the grouping of task-relevant data and the determination of sampling weights, ultimately maximizing task influence across all data.

Our contributions are as follows: (1) we propose the AutoMixer framework that identifies task-specific checkpoints from simulation runs, which can be used as effective data samplers to boost model performance on desirable tasks, enabling both the identification of task data mixtures and their respective importance weights; (2) we show through extensive analysis that proxy models can serve as effective data samplers and mixers using only simulation runs, which allows the reuse of these proxy data mixers across different tasks and training scenarios.

2 Background

Training large language models is fundamentally shaped by both the breadth and quality of their data (Ye et al., 2024). A major challenge arises from identifying task-specific examples within extensive and heterogeneous corpora, particularly when direct mappings between data domains and target tasks are absent or ambiguous (Sedinkina and Schütze, 2019; Wu et al., 2022). Beyond this complexity, overlapping knowledge regions often blur domain boundaries, further complicating the extraction of samples most conducive to skill development (Sedinkina and Schütze, 2019; Britz et al., 2017). Traditional data-mixing techniques frequently overlook these overlaps, leading to suboptimal data usage and task generalizabilities (Conneau and Lample, 2019; Li et al., 2023; Blitzer et al., 2007; Lee et al., 2022; Wang et al., 2020).

A second layer of complexity stems from the dynamic nature of skill acquisition in language models. Model capabilities can surface and evolve at various points during training, often following non-monotonic trajectories (Wei et al., 2022; Hu et al., 2023). This non-monotonicity gives rise to a "chicken-and-egg" dilemma: determining which data samples facilitate a particular skill is difficult until the model has already begun to exhibit that skill. Although brute-force methods—such as training multiple data mixtures and rigorously testing their impact on downstream tasks—could theoretically illuminate these relationships, they are computationally infeasible at scale.

Influence-based techniques offer a more principled approach to this problem. By examining how individual training samples affect model predictions (Hampel, 1974; Halevy et al., 2009), influence functions pinpoint data regions that are especially valuable for specific tasks. However, standard influence estimation and its approximations (Kwon et al., 2024; Yeh et al., 2022) often rely on a single checkpoint (commonly the final model state), thereby neglecting how early-stage knowledge may influence performance in subsequent stages (Park et al., 2023). As a result, purely first-order or single-step influence evaluations may fail to capture the full evolution of a model's skill trajectory.

Addressing these gaps requires a framework that:

1. Subdivides large, heterogeneous data sources into groups aligned with emerging model competencies, and

2. Adjusting sampling weights for these data groups for task-aware data loading.

Such an approach would exploit the strengths displayed at different checkpoints, rather than treating the model as static. Incorporating multi-checkpoint influence measurements enables adaptive data curation that aligns with the model's ever-shifting learning needs, ultimately leading to more efficient and effective skill acquisition.

3 The AutoMixer Framework

In this work, we assume the presence of raw data of various sources (e.g. common crawl snapshots) comprising n samples distributed across m task data, with a token budget T. Each sample x_i, where $i = 1, \ldots, n$, consists of s_i tokens. The main idea behind *AutoMixer* is to decide for each

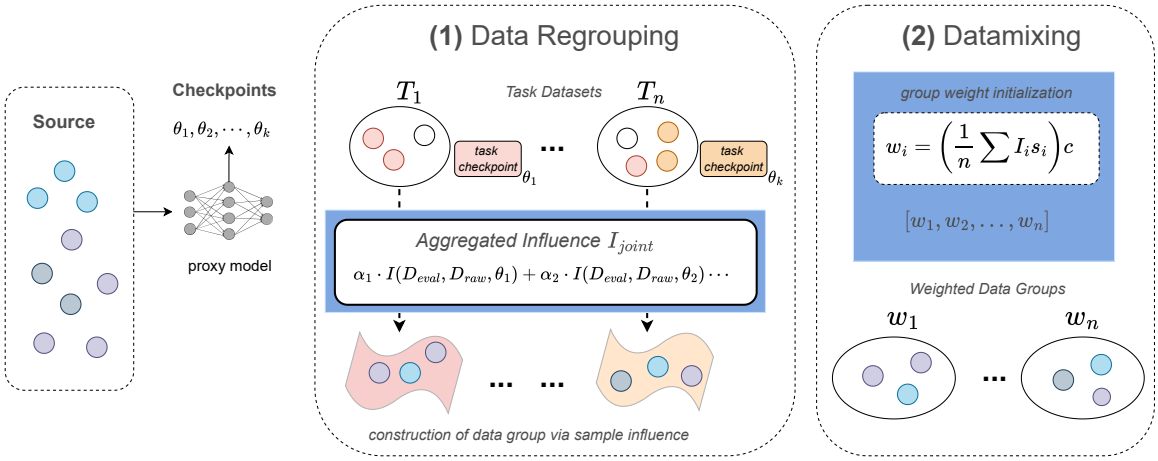

Figure 2: Illustration of the AutoMixer framework: Each data sample from ungrouped raw pretraining sources is assigned an influence score. These scores guide the regrouping of incoming data into task-specific datasets by (1) splitting raw data into groups based on task checkpoints, and (2) Determine sampling weights by aggregating influence scores across checkpoints.

task whether a piece of data should be part of the training process. If the data is chosen, it becomes part of a special collection tailored for that task. This approach ensures that each task gets the most relevant data for its training needs.

Concretely, this process of *data regrouping* is achieved by utilizing performance-based checkpoints, in which AutoMixer identifies the most effective stopping points during training (See Figure 2). The idea is that when a model achieves peak performance on task j at a particular training step, the checkpoint can be employed as an effective sampler to compute influence estimation (Koh et al., 2019a,b; Ting and Brochu, 2018) for task j. Empirically, we obtain the checkpoints from simulation runs, and leverage them to identify task-optimal datasets in a two-step process (See Figure 2). The aggregated task influences can then be used to determine the weights over each group, which dictates the sampling probability of groups during language model pretraining.[1]

1. **Data Regrouping**: First, raw data are regrouped based on the sampled checkpoints $(\theta_1, \ldots, \theta_k)$. These k checkpoints are selected from a single simulation run, each corresponding to the best-performing checkpoint for one of the m tasks, where $k \leq m$ (see Table 1).[2] To quantify the alignment score with the task data, influence scores are obtained from simulation runs with proxy language models, which are smaller and faster to compute sample influence.

2. **Datamix Reweighting**: Next, to assign sampling weights to each data group, the per-sample influence is aggregated with sample token counts in a reweighting process. This maximizes the influence across all tasks but also ensures token count constraints are met.

We delve into these two steps as follows.

4 Data Regrouping via Sample Influence

Data regrouping enhances language model pretraining by reorganizing raw data into task-specific groups, each defined by a distinct model checkpoint sampled from a simulation run. During the simulation training, a collection of checkpoints is obtained. We record the performance of each checkpoint across all tasks, allowing us to create a benchmark table summarizing task performances, as shown in Table 1[3]. We then select the top k checkpoints that perform best across all m tasks, where k is less than or equal to m. The final step involves regrouping data samples based on their utility scores, defined as influence scores (Koh et al., 2019a). In our experiments, influence scores $\mathcal{I}(x_i)$ are calculated for all samples using proxy models with 75M and 350M parameters. Within each of the m task-aligned groups, samples are sorted by

[1] Suppose a data group has a weight $w_g = 0.2$, and all weights are normalized to 1, this means, on average, 20% of the tokens sampled per batch will come from this group.

[2] In the case where m becomes large, we can perform unsupervised clustering to keep it manageable.

[3] Same checkpoint numbers are indicated with same colors.

Model	Task							
	ARC-Easy	ARC-Hard	BOOLQ	PIQA	SIQA	HELLASWAG	OBQA	WINOGRANDE
25M	81%	76%	10%	100%	56%	100%	76%	81%
50M	85%	45%	65%	80%	60%	100%	100%	25%
75M	70%	70%	5%	85%	60%	100%	95%	100%
350M	95%	70%	40%	100%	80%	100%	95%	100%
500M	100%	95%	75%	85%	90%	100%	100%	85%
1.5B	85%	85%	85%	95%	90%	95%	95%	95%

Table 1: Checkpoint progression ratio for various tasks across different model sizes for a total of 100K steps: For instance, if the checkpoint that performs the best at a task is stored at the 5000th step, it is then recorded on the table as 5%. Checkpoints for each model size are collected from one simulation run by training the language model on FineWeb data (**?**) with 100K steps. Here we set the tasks to be the considered benchmarks: ARC-easy, ARC-challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), and WinoGrande (Sakaguchi et al., 2021).

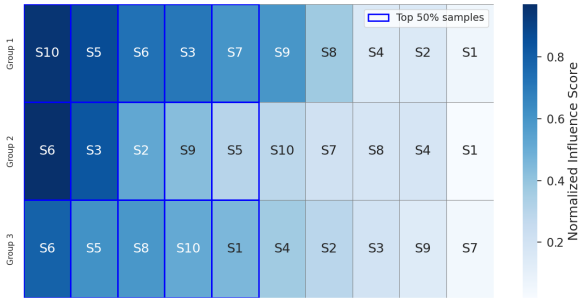

Figure 3: **Depiction of the Data Regrouping Process:** Within each group, samples (w/ indices) are sorted based on their joint influence scores across all tasks. This sorting results in different sample orderings between groups. The final step involves selecting the top $K\%$ of samples from each group to form a data group that fulfills the token budget. Although there are duplicates across the table, we found that repeated samples contain high-value tokens that are beneficial for repeated exposure during training.

their utility scores, retaining the top 50% of samples for each checkpoint sampler (See Figure 3). For each sample x_i, we aggregate contributions from all checkpoints:

$$\mathcal{I}_{\text{joint}}(x_i) = \sum_{j=1}^{k} \alpha_j \cdot \mathcal{I}(x_i; \theta_j),$$

where α_j is the blending factor for checkpoint θ_j, determined by task performance and the checkpoint number, which we will in the next paragraph.

For simulation runs, the token budget is set at 100,000 steps with a batch size of 8, across 4 nodes, and a sequence length of 2048, totaling 6.4 billion tokens. This setup is designed to limit the cost of simulation runs by keeping the total computing budget in a reasonably range; and also for the fact that the proxy models tend to converge early in training. In cases where data groups exceed this token budget, each group is subsampled in proportion to its aggregate influence scores, ensuring optimal alignment between tokens and tasks while adhering to computational constraints.

***Blending Factor* via Task Acquisition Speed.** Moreover, Table 1 also implies the speed of convergence of different tasks across model sizes. Namely, HellaSwag seems to take longer for the model to get good at. While it might be tempting to say that this is due to the difficulty of that particular task, we leave that discussion for future works. However, it does provide a decent measure of how long it takes to learn each task in the training runs.

In practice, we utilize each checkpoint to assess the influence of individual samples, while simultaneously gathering task-specific signals across all samples. To achieve this, we propose using the checkpoint steps associated with each task as weights when calculating a sample's influence for each checkpoint. The blending factor (α_j) for each checkpoint is determined by normalizing the checkpoint step numbers. Let s_j be the step number for checkpoint j, and $s_{\max}$ be the largest step number among all checkpoints. The normalized step number for checkpoint j is:

$$\tilde{s}_j = \frac{s_j}{s_{\max}}$$

The blending factor α_j is then:

$$\alpha_j = \frac{\tilde{s}_j}{\sum_{i=1}^{k} \tilde{s}_i}$$

This ensures that the blending factors sum to 1 where the *blending factor* α_i allows us to integrate

influence scores across different tasks for each sample effectively.

Below, we formalize the influence computation framework and its implementation.

Formulating Sample Influence. Consider a pretraining dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ for a next-token prediction objective, where each input x_i and label y_i is a sequence of tokens, with y_i obtained by shifting x_i one token to the left. The empirical risk minimizer θ^* is obtained through:

$$\theta^* := \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$$

where ℓ denotes cross-entropy loss. To measure the influence of sample (x_k, y_k), we analyze parameter shifts under ε-weighted risk minimization:

$$\theta^{(k)}(\varepsilon) := \arg\min_{\theta \in \Theta} \left(\frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i)) + \varepsilon \ell(y_k, f_\theta(x_k)) \right)$$

The influence of (x_k, y_k) on the model's performance, reflected in the model's predictive capability on the validation set $\mathcal{D}^{(\text{val})} = \{(x_i^{(\text{val})}, y_i^{(\text{val})})\}_{i=1}^q$, is quantified as:

$$\begin{aligned} \mathcal{I}(x_k, y_k) &= \frac{\mathrm{d}}{\mathrm{d}\varepsilon} \Big(\frac{1}{q} \sum_{j=1}^{q} \ell\big(y_j^{(\text{val})}, f_\theta(x_j^{(\text{val})})\big) \Big) \Big|_{\varepsilon=0} \\ &= -\Big(\frac{1}{q} \sum_{j=1}^{q} \nabla_\theta \ell\big(y_j^{(\text{val})}, f_\theta(x_j^{(\text{val})})\big) \Big) \\ &\quad \times H^{-1}(\theta)\, \nabla_\theta \ell\big(y_k, f_\theta(x_k)\big) \Big|_{\theta=\theta^*} \end{aligned}$$

where $H(\theta)$ is the Hessian of the empirical loss. Direct computation is prohibitive for large models, necessitating approximations.

Efficient Influence Approximation. We adopt DataInf (Kwon et al., 2024) to bypass explicit Hessian inversion. For layer l in the Transformer model:

1. Compute validation gradients averaged over q validation samples:

$$v_l = \frac{1}{q} \sum_{j=1}^{q} \nabla_{\theta_l} \ell(y_j^{(\text{val})}, f_\theta(x_j^{(\text{val})}))$$

2. Compute the layer-specific regularization parameter λ_l:

$$\lambda_l = 0.1 \times (n d_l)^{-1} \sum_{i=1}^{n} \|\nabla_{\theta_l} \ell(y_i, f_\theta(x_i))\|_2^2$$

where d_l denotes the layer dimension, and n denotes the number of training samples.

3. Update running sum r_l across training samples for approximating the inverse Hessian-vector product:

$$c_{li} = \frac{v_l^\top \nabla_{\theta_l} \ell(y_i, f_\theta(x_i))}{\lambda_l + \|\nabla_{\theta_l} \ell(y_i, f_\theta(x_i))\|_2^2},$$
$$r_l \leftarrow r_l + \frac{v_l - c_{li} \nabla_{\theta_l} \ell(y_i, f_\theta(x_i))}{n \lambda_l}$$

4. Aggregate layer contributions to compute final influence:

$$\mathcal{I}(x_k, y_k) \approx -\sum_{l \in \{1, L\}} r_l^\top \nabla_{\theta_l} \ell(y_k, f_\theta(x_k))$$

Discriminative Layer Selection. In practice, we save computes by avoiding matrix multiplication across all layers of the model. Instead, we resort to computing influence scores with only the embedding and last layer in order to enhance influence score discriminability. Thus, we focus on gradients from the first (embedding) and last (output) Transformer layers: we postulate that this dual-layer strategy mitigates the *cancellation effect* by using only the last layer (Yeh et al., 2022) prevalent in intermediate layers, where shared processing logic obscures sample-specific influences. By isolating gradient signals from these critical layers, AutoMixer obtains more reliable influence estimates for regrouping decisions.

5 Datamix Reweighting

Regrouped datasets are presented to the model during pretraining at varying probabilities. This means that data group i will be sampled differently from another group j, if $i \neq j$. The datamix reweighting aims to determine the probabilities to sample from each group, which will translate to the proportion of tokens in each batch of data during training.

In data regrouping, we aggregate influence scores across selected checkpoints to optimize data sampling weights via joint influences. A joint influence score for each sample (x_i) combines its contributions to all tasks and is computed in the data regrouping stage in the previous section as

$\mathcal{I}_{\text{joint}}(x_i)$. Now, the question remains as to how to define the sampling weight w_g for group g. To reflect the impact of each data group on the overall performance of all tasks, we define the group influence density ρ_g for each data group g as:

$$\rho_g = \frac{1}{T_g} \sum_{x_i \in g} \mathcal{I}_{\text{joint}}(x_i) \cdot s_i,$$

where s_i is the token count of sample x_i, and T_g is the token count of group g.

We determine each group's sampling weight based on its influence density:

$$w_g = \frac{\rho_g}{\sum_{g'} \rho_{g'}},$$

where $w_g \in (0, 1)$. When sampling tokens from each group to form the pretraining dataset, we ensure that the proportion of tokens selected from group g is w_g. This approach optimizes overall performance across all tasks while ensuring fairness among them.

During the pretraining dataset construction, we track the total token count of the selected samples to ensure the total training token budget T is not exceeded. In scenarios where we want to limit the token budget for a specific group, we sum the token counts of that group's selected samples. Once the group's limit is reached, we stop sampling from it and proportionally increase the sampling weights for the remaining groups. This ensures the token budget of the group is respected.

6 Experimental Setup

Dataset. The experiments employ the FineWeb-Edu dataset (**?**)[4], a specialized educational corpus derived from FineWeb through quality-based curation. Two versions are available: a foundational 1.3 trillion token collection and an expanded 5.4 trillion token iteration (FineWeb-Edu-score-2). The dataset ensures educational relevance through a classifier trained on Llama3-70B-Instruct-generated synthetic annotations, which selects pedagogically valuable content.

Experimental Details. We implement decoder-only transformers following the Llama-3 architecture (Dubey et al., 2024), pretrained with causal language modeling objectives. We conduct two training runs per configuration with distinct random initializations. Our model scale analysis spans four parameter counts (350M, 1.5B, 3B) to systematically investigate size-performance relationships. All training occurs on a 32-GPU cluster (4 nodes, 8xH100 GPUs/node) using consistent hyperparameters across configurations.

Data selection employs influence scores calculated via the 350M proxy model, with ablation performed with the smaller 75M scale in order to understand the balance computational tractability and model capability[5]. Checkpoint evaluation requires ∼120 hours on 100 GPUs, with subsequent simulation runs completing within 48 hours.

Evaluation Tasks. We assess zero-shot performance on eight common-sense reasoning benchmarks: ARC-easy, ARC-challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), and WinoGrande (Sakaguchi et al., 2021). These tasks serve dual purposes: guiding influence score computation during data selection and providing final performance metrics. This closed-loop design ensures alignment between training dynamics and evaluation objectives.

Benchmark Comparisons. AutoMixer employ several different proxy model sizes of 75M and 350M, which we denote as *AutoMixer-75M* and *AutoMixer-350M* respectively. We evaluate against several baseline strategies:

1. **Uniform Sampling**: Draws data uniformly from FineWeb without prioritization, constrained only by a fixed token budget. This baseline measures inherent dataset quality.

2. **PPL Sampling**: Here we adopt the commonly used sampling technique (Wenzek et al., 2020) where we estimate the sample utility based on the sample sequence cross-entropy loss, or perplexity, which is used in place of the influence scores, while keeping the framework algorithm constant, where lower perplexity samples are better.

3. **N-gram Sampling**: Moreover, we also compared a recent approach in Chang et al. (2024) (*n-gram sampling*) that shares similar settings, where target evaluations are utilized to sample pretraining data using n-gram-based techniques (Xu et al., 2020). This serves as a

[4]Open Data Commons Attribution License (ODC-By) v1.0

[5]The estimated costs for 350M and 75M simulation runs are $81.60 and $79.89, respectively.

Approach	Improvement over Uniform Sampling (Accuracy %)								
	ARC-Easy	ARC-Hard	BOOLQ	PIQA	SIQA	HELLASWAG	OBQA	WINOGRANDE	Avg.
350M Parameters									
PPL Sampling	0.35 ± 0.03	0.60 ± 0.05	0.44 ± 0.03	0.70 ± 0.04	-0.10 ± 0.05	0.55 ± 0.03	0.40 ± 0.04	0.90 ± 0.06	0.66 ± 0.03
n-gram sampling	0.74 ± 0.06	**1.22** ± 0.04	0.79 ± 0.07	1.03 ± 0.05	1.09 ± 0.06	0.62 ± 0.03	1.16 ± 0.09	0.85 ± 0.04	0.60 ± 0.05
AutoMixer-75M	-0.15 ± 0.07	0.12 ± 0.04	-0.14 ± 0.05	0.01 ± 0.08	-0.10 ± 0.03	0.05 ± 0.09	-0.03 ± 0.06	-0.05 ± 0.04	-0.04 ± 0.05
AutoMixer-350M	**2.23** ± 0.08	0.55 ± 0.06	**2.16** ± 0.09	**2.05** ± 0.07	**2.12** ± 0.10	**2.33** ± 0.06	**2.01** ± 0.08	**2.14** ± 0.05	**1.93** ± 0.07
1.5B Parameters									
PPL Sampling	0.20 ± 0.07	0.52 ± 0.03	0.32 ± 0.06	0.40 ± 0.02	0.07 ± 0.05	0.18 ± 0.08	0.75 ± 0.03	0.68 ± 0.06	0.48 ± 0.04
n-gram sampling	0.88 ± 0.06	**0.82** ± 0.04	1.02 ± 0.07	0.58 ± 0.05	0.45 ± 0.09	1.22 ± 0.03	0.54 ± 0.05	0.90 ± 0.08	0.79 ± 0.06
AutoMixer-75M	-0.06 ± 0.04	0.01 ± 0.06	-0.04 ± 0.05	-0.02 ± 0.07	-0.05 ± 0.08	0.02 ± 0.06	0.01 ± 0.04	-0.03 ± 0.05	-0.02 ± 0.06
AutoMixer-350M	**1.26** ± 0.05	0.39 ± 0.09	**1.35** ± 0.06	**1.22** ± 0.08	**1.38** ± 0.06	**1.45** ± 0.04	**1.33** ± 0.07	**1.41** ± 0.09	**1.22** ± 0.05
3B Parameters									
PPL Sampling	0.18 ± 0.06	0.32 ± 0.04	0.10 ± 0.07	0.42 ± 0.05	0.27 ± 0.08	0.34 ± 0.03	0.50 ± 0.05	0.15 ± 0.09	0.20 ± 0.06
n-gram sampling	0.82 ± 0.05	**0.72** ± 0.07	0.57 ± 0.06	0.54 ± 0.04	0.64 ± 0.08	0.42 ± 0.05	0.95 ± 0.09	0.84 ± 0.06	0.50 ± 0.05
AutoMixer-75M	-0.02 ± 0.06	0.01 ± 0.05	-0.03 ± 0.08	0.00 ± 0.04	-0.02 ± 0.07	-0.01 ± 0.06	-0.04 ± 0.03	0.02 ± 0.08	-0.01 ± 0.06
AutoMixer-350M	**1.09** ± 0.04	0.34 ± 0.08	**1.12** ± 0.06	**1.06** ± 0.07	**1.14** ± 0.05	**1.27** ± 0.09	**1.23** ± 0.08	**1.18** ± 0.05	**1.05** ± 0.07

Table 2: Improvements over the uniform baseline (accuracy %), each with two decimal places and averaged over 2 runs. Negative values (e.g. −0.10) indicate worse performance than uniform for those specific tasks. Bolded entries represent the highest improvement in each *column*. Overall, AutoMixer-350M yields the largest gains across most tasks, while other methods occasionally underperform vs. uniform.

Checkpoint Strategy	*Avg. Improvement over Uniform (%)*
Last Checkpoint	0.7 ± 0.4
10 Checkpoints	0.8 ± 0.3
AutoMixer-350M	**1.22** ± 0.05

Table 3: Average improvements over a uniform sampling baseline, aggregated across multiple benchmarks for 1.5B.

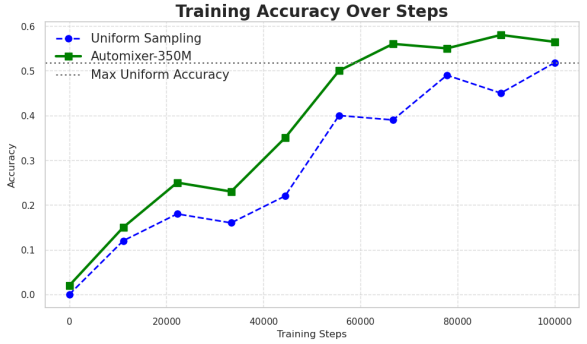

Figure 4: A performance comparison of two approaches (*uniform sampling* and *AutoMixer-350M*) across ten evenly spaced training steps (0 – 100k). Both exhibit minor fluctuations yet follow an overall upward trend in accuracy. *AutoMixer-350M* consistently outperforms *uniform sampling* throughout training, ultimately reaching 56.45% accuracy versus 51.82% for *uniform sampling*.

robust test to determine if simpler methods can achieve comparable results.

7 Main Results

Table 2 shows how different sampling strategies improve over a uniform baseline across four model scales (350M, 1.5B, and 3B parameters). These results support our main claim that precise identification and reweighting of task-relevant data can yield substantial performance gains. Specifically, *AutoMixer-350M* achieves the highest overall improvements in most settings, confirming that the alignment between a proxy model and the final target model is pivotal for learning task-specific data representations. We also observed that it lags behind *n-gram sampling* at times while *AutoMixer-75M* performs poorly, suggesting the proxy model size do play a huge role in the framework effectiveness. Further, *ppl sampling* performs poorly on the framework setting, suggesting that checkpoints cannot be used off-the-shelf in the way it intends to. We also observe the largest gains seem to be for the 350M model, which is the proxy model size; which makes sense from the perspective that influence scores are computed with the same number of parameters.

Despite these occasional variations, the data still validate our two-fold approach. First, regrouping data based on capability checkpoints rather than solely relying on fixed domain labels appears crucial for navigating the complexity highlighted in the introduction, where overlapping or ill-defined domains can undermine performance. Second, assigning sampling weights to these regrouped sets amplifies the most beneficial samples for each skill, aligning with our objective of pinpointing and upweighting high-value data.

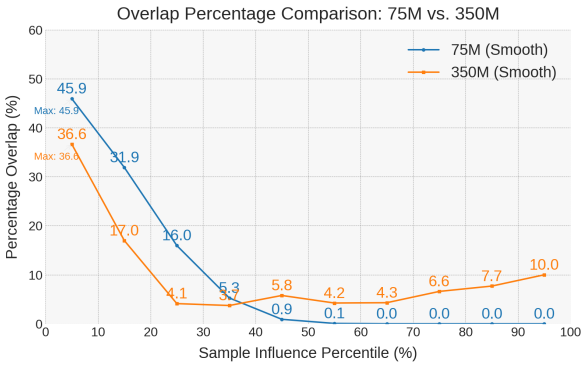

Figure 5: Plot of sample overlap percentages in respective models (75M & 350M), measured across increasing influence scores. Two main trends emerge: (1) The 75M model identifies similar set of samples as the larger 350M models for lower-influence samples; while the 350M model captures a higher percentage of higher influence samples and, (2) The smaller proxy model's overlap distribution is notably skewed toward lower sample influences.

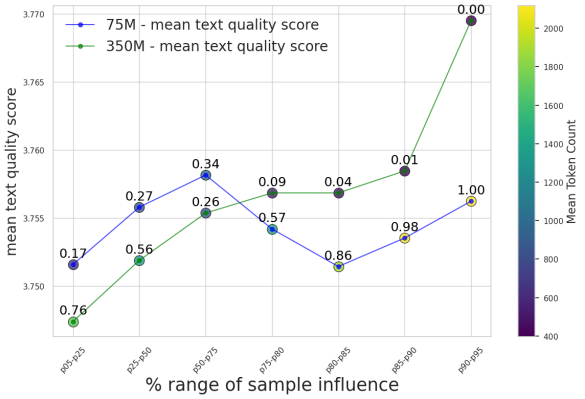

Figure 6: **Mean text quality score by range**: We show the sample quality score across all buckets (grouped by percentiles) of samples sorted by influence scores. The *normalized mean token count* (in range $[0, 1]$) per sample in the same set of buckets is labeled on each point. 75M proxy model tends to select longer sentences with higher influences.

8 Further Discussion

Data Regrouping Ablation. Table 3 compares three checkpoint-based strategies in terms of their average improvement over a uniform sampling baseline. Relying only on the final ("last") checkpoint yields a 0.7% gain, while aggregating all available checkpoints (up to 10) increases that margin to 0.8%. The use of selected checkpoints in AutoMixer achieves a more pronounced boost of 1.93% for 350M scale, demonstrating the advantages of using selected checkpoints to sample data for data regrouping. Here we show that by leveraging checkpoints that excel in distinct tasks at various stages, and capitalizes on non-monotonic skill emergence, we can more effectively pinpoints those training samples most conducive to each targeted capability. However, it is also true that earlier Table 1 also shows a trend where larger model tends to converge all skills onto one checkpoint, we leave the question of trade-offs between the number of checkpoint samplers and size of model (influence scores' computational speed) for future research.

AutoMixer Performance Trajectory. Figure 4 compares our proposed AutoMixer-350M approach against *uniform sampling* across ten training steps (0-100k). The plotted accuracies reveal that AutoMixer-350M not only starts above Uniform but sustains a clear lead throughout training, ultimately reaching 56.45% accuracy compared to 51.82% for *uniform sampling*. These steady gains underscore AutoMixer's ability to more effectively reorganize training data over time, allowing the model to focus on samples that yield greater learning benefits.

Impact of Proxy Model Sizes. Figure 6 reveals an intriguing pattern: smaller proxy models, such as those with 75M parameters, tend to select samples with longer sentences and higher influence scores. This observation suggests that the 75M proxy model is particularly adept at identifying influential samples by focusing on longer, more informative sentences. This capability is especially useful in scenarios with limited computational resources, as these models provide valuable signals for sampling task-aware data, even within the lower influence-scoring range.

Conversely, larger proxy models, like the 350M model, excel at distinguishing samples with higher influence scores (See Figure 5). This implies that increasing model size enhances the ability to discern and prioritize samples that are more impactful for downstream tasks. Larger models are thus better suited for identifying high-quality samples that significantly contribute to the learning process. These findings highlight two key insights: (1) Smaller proxy models can be effectively used to derive useful signals for pruning, especially when focusing on lower influence-scoring samples. (2) Larger models offer improved performance in identifying and prioritizing samples with higher influence scores, making them advantageous for tasks

requiring high precision in sample selection.

Conclusion

Our study demonstrates the effectiveness of *AutoMixer* in enhancing language model pretraining through strategic checkpoint sampling and data regrouping. These results highlight *AutoMixer*'s ability to enhance skill acquisition. Moreover, the analysis of proxy model sizes reveals that smaller models, such as those with 75M parameters are suitable for resource-constrained scenarios; while larger models, like the 350M model, excel in prioritizing high-impact samples, offering advantages for tasks requiring high precision. Overall, these findings underscore the potential of optimized data sampling and checkpoint models to significantly boost pretraining performance. By leveraging the strengths of different model sizes and strategic data selection, we can achieve accuracy gains reaching up to X%, paving the way for more efficient and effective language model training.

Limitations

Although our multi-checkpoint data mixing strategy demonstrates improvements on diverse reasoning tasks, several constraints remain. First, estimating sample influence across multiple checkpoints introduces additional computational overhead, potentially limiting scalability to larger training runs. Second, while we highlight how domain overlaps complicate data grouping, our current approach does not explicitly mitigate biases that may arise from imbalanced or skewed datasets. Third, the data mixtures and checkpoints used in this work are focused on reasoning benchmarks; applying the same procedure to other domains or more extensive, heterogeneous corpora may reveal different challenges or optimal configurations. Finally, as our proxy-based simulations rely on approximate modeling of training dynamics, there is no guarantee that fine-grained influence scores will hold across substantially different architectures or larger-scale training protocols. Future work could address these gaps through further studies into how checkpoint selection interacts with diverse model architectures and data domains.

Ethics Statement

Our approach leverages public datasets and standard reasoning benchmarks, aiming to refine how training samples are selected and weighted based on evolving model capabilities. While this focus can boost efficiency and skill-specific performance, it also raises ethical considerations. For instance, optimizing data mixtures for particular tasks may inadvertently deprioritize other skills or amplify existing biases if certain subpopulations are underrepresented or misrepresented in the training data. Additionally, the checkpoint-based framework does not inherently account for privacy or fairness concerns, making transparency in data sourcing and audit processes essential. Researchers and practitioners employing this method should be mindful of the potential for disproportionate impact on vulnerable groups and consider implementing robust bias detection, inclusive data collection, and clear documentation regarding data provenance. As with any technique that refines data selection, careful oversight is necessary to ensure that efforts to enhance performance do not come at the expense of fairness or responsible use.

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Domain adaptation for sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V Le. 2017. Effective domain mixing for neural machine translation. In *Conference on Machine Translation (WMT)*.

Ernie Chang, Pin-Jie Lin, Yang Li, Changsheng Zhao, Daeil Kim, Rastislav Rabatin, Zechun Liu, Yangyang Shi, and Vikas Chandra. 2024. Target-aware language modeling via granular data sampling. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Neural Information Processing Systems (NeurIPS)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Samir Gadre, Mitchell Wortsman, Gabriel Ilharco, et al. 2024. Datacomp-lm: Benchmarking data quality for language models. *arXiv preprint arXiv:2401.XXXXX*.

Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. The unreasonable effectiveness of data. *IEEE intelligent systems*, 24(2):8–12.

Frank R. Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Shengding Hu, Xin Liu, Xu Han, Xinrong Zhang, Chaoqun He, Weilin Zhao, Yankai Lin, Ning Ding, Zebin Ou, Guoyang Zeng, et al. 2023. Unlock predictable scaling from emergent abilities. *arXiv preprint arXiv:2310.03262*.

Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. 2019a. On the accuracy of influence functions for measuring group effects. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. 2019b. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. 2024. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. In *International Conference on Learning Representations (ICLR) 2024*. Also available as arXiv:2310.00902.

Kenton Lee, Zi-Yi Dou Chen, Gautam Krishna, and Hannaneh Hajishirzi. 2022. Deduplicating training data makes language models better. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Wenhao Li, Liang Zhou, Graham Neubig, and Pengfei Liu. 2023. Word matters: What influences domain adaptation in summarization? In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning*, pages 27074–27113. PMLR.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Maria Sedinkina and Hinrich Schütze. 2019. Domain-specific adaptation for neural machine translation. In *Proceedings of ACL*.

Daniel Ting and Eric Brochu. 2018. Optimal subsampling with influence functions. *Advances in neural information processing systems*, 31.

Rui Wang, Deyi Xiong, Jian Zhang, Zhaopeng Tu, and Jun Huang. 2020. Go from the general to the particular: Multi-domain translation with domain transformation networks. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Chang Wu, Xiang Liu, Rohan Patel, and Soojin Kim. 2022. Identifying task-relevant data for efficient nlp training. In *Proceedings of EMNLP*.

Kai Xu, Mark Johnson, and Tuong Nguyen. 2020. Domain-adaptive language model pretraining via n-gram filtering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2345–2356.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. 2024. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*.

Chih-Kuan Yeh, Ankur Taly, Mukund Sundararajan, Frederick Liu, and Pradeep Ravikumar. 2022. First is better than last for language data influence. *Advances in Neural Information Processing Systems*, 35:32285–32298.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

A Dataset Analysis

Range	Mean Language Score	Mean Score	Mean Int Score	Mean Token Count	Sum Token Count
p80-p85	0.97	3.76	4.01	474.01	1,593,100,685
p85-p90	0.97	3.76	4.01	413.55	1,388,916,981
p75-p80	0.97	3.76	4.00	550.58	1,852,164,429
p05-p25	0.96	3.75	4.00	1,698.47	22,912,588,553
p90-p95	0.97	3.77	4.01	397.05	1,345,066,462
p25-p50	0.96	3.75	4.00	1,353.91	22,789,192,924
p50-p75	0.96	3.76	4.00	846.16	14,215,340,759

Table 4: Summary statistics for different percentile ranges for AutoMixer-350M.

B Data Regrouping

In our experiments, we utilized a data regrouping strategy to enhance the quality and relevance of our dataset.

For each of the K checkpoints, we performed the following operations:

1. **Percentage Calculation**: We calculated the approximate percentiles of the `influence` metric within our dataset.
2. **Table Creation**: We created a new table to store data with influence values above a certain threshold.
3. **Influence Aggregation**: We aggregated influence scores across multiple tables to create a new column `total_influence`.

Weight Estimation. To estimate the weight of each table, we calculated the scaled influence and the total influence token product. This involved computing the minimum and maximum influence values and scaling the influence accordingly.

Algorithm 1 Data Processing Pipeline

Input: Dataset D, Threshold T
Output: Total Influence Token Product

Procedure DATAREGROUPING
Calculate percentiles of influence in D
Create a filtered dataset F where influence $\geq T$
End Procedure

Procedure INFLUENCEAGGREGATION
Aggregate influence scores in F to compute total influence for each ID
Store results in a new table A
End Procedure

Procedure WEIGHTESTIMATION
Calculate min and max influence from A
Compute scaled influence for each entry in A
Calculate total influence token product
End Procedure

Return Total Influence Token Product