PiKE: Adaptive Data Mixing for Large-Scale Multi-Task Learning Under Low Gradient Conflicts

author names withheld

Under Review for OPT 2025

Abstract

Modern foundation models are trained on diverse datasets to enhance generalization across tasks and domains. A central challenge in this process is determining how to effectively mix and sample data from multiple sources. This naturally leads to a multi-task learning (MTL) perspective. While prior work in MTL has emphasized mitigating gradient conflicts, we observe that large-scale pretraining scenarios—such as multilingual or multi-domain training—often exhibit little to no gradient conflict. Motivated by this observation, we propose **PiKE** (**P**ositive gradient **i**nteraction-based **K**-task weights **E**stimator), an adaptive data mixing algorithm that dynamically adjusts sampling weights during training. PiKE exploits non-conflicting gradient interactions to minimize a near-tight upper bound on the average loss decrease at each step, while incurring negligible computational overhead. We provide theoretical convergence guarantees and show that PiKE outperforms static and non-adaptive mixing baselines. Furthermore, we extend PiKE to promote balanced learning across tasks. Extensive experiments on large-scale language model pretraining confirm that PiKE achieves faster convergence and improved downstream performance compared to existing approaches.

1. Introduction

Foundation models, such as large language models (LLMs), owe their strong generalization abilities to pretraining on diverse datasets [9, 33, 59]. The effectiveness of these models depends heavily on the composition of their training data [17, 26], yet determining the optimal data mixture remains a key challenge. Common strategies use fixed data mixtures, chosen heuristically [69] or via smaller, computationally expensive proxy models [17, 67]. These approaches have notable limitations: their optimality is unclear, as policies from small models may not transfer to larger ones [72], and static weights may become suboptimal as training progresses [31, 75], a limitation discussed further in Section A.

In this work, we frame adaptive data mixture selection as a multitask optimization problem, enabling a principled approach to dynamically adjusting the data mixture. This view is natural, as each data domain yields a distinct gradient. Prior multitask optimization methods [8, 15, 63, 73], however, are often impractical for LLMs due to prohibitive memory or computational costs. Furthermore, most are tailored for vision tasks where gradient conflicts are prevalent. We find this condition is less common in LLM pretraining; extending the findings of Wang et al. [63] to much larger decoder-only models, we show that gradients across domains are mostly cooperative or nearly orthogonal (Section 2). This suggests that complex conflict-mitigation mechanisms may be unnecessary, motivating the central question of our work:

*Can we design an effective **adaptive data mixing strategy** based on **multitask optimization**, that **exploits non-conflicting** nature of gradients and scales to **large settings** with **minimal memory and computational overhead**?*

© .

To address this, we propose *PiKE*, an adaptive data mixing that is **empirically effective** and **scalable** for pretraining LLMs. PiKE enjoys **theoretical guarantees** while remaining practical for large-scale settings—with many tasks, large models, and diverse data—at **negligible memory and compute overhead**. We summarize PiKE's key features below; related work is discussed in Appendix C.

Key Features of PiKE

1. Adaptively adjusts mixture weights using per-task gradient magnitude and variance.
2. Theoretically, achieves near-optimal per-iteration objective decrease and enjoys convergence guarantees (Section 3).
3. Incorporates tilted empirical risk minimization [32, 43], promoting balanced learning across tasks and preventing task under-representation (Section B).
4. Scales efficiently (linearly) with model size and the number of tasks (Section 4).
5. Consistently outperforms existing methods across different scales (110M to 1B parameters) and scenarios (multilingual to domain mixing) (Figure 1 and Section 4).

Problem Definition and Notations We aim to train a *single* model with parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ to perform $K \geq 2$ tasks simultaneously. Each task k is associated with a smooth (possibly non-convex) loss $\ell_k(\boldsymbol{\theta}, x) : \mathbb{R}^d \times \mathbb{R}^{d_x} \to \mathbb{R}$ where x is the data point. It is common to minimize the total loss:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}) := \sum_{k=1}^{K} \mathbb{E}_{x \sim \mathcal{D}_k}[\ell_k(\boldsymbol{\theta}; x)], \tag{1}$$

where $\mathcal{D}_k$ represents the data distribution for task k. We define $\mathcal{L}_k(\boldsymbol{\theta}) := \mathbb{E}_{x \sim \mathcal{D}_k}[\ell_k(\boldsymbol{\theta}; x)]$. For notation, $\|\cdot\|$ represents the Euclidean norm, $\mathrm{Tr}(\cdot)$ denotes the trace operator, and a function h is L-Lipschitz if $\|h(\boldsymbol{\theta}) - h(\boldsymbol{\theta}')\| \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$ for any $\boldsymbol{\theta}, \boldsymbol{\theta}'$ in the domain of $h(\cdot)$. A function $f(\cdot)$ is L-smooth if its gradient is L-Lipschitz continuous.

2. Main Building Blocks for PiKE: Large-Scale Training: Low Gradient Conflict & MTL Scalability Challenges

This section presents our main observations which form the main building blocks of PiKE. Additional building blocks is presented in Appendix A

The *Mix* batching strategy offers a natural lens for analyzing data mixing in multitask learning (MTL). When a batch of total size b is constructed with b_k samples from each task k, the overall gradient at iteration t, denoted $\mathbf{g}_t$, is the average of the individual sample gradients:

$$\mathbf{g}_t = \frac{1}{b} \sum_{k=1}^{K} \sum_{i=1}^{b_k} \nabla \ell_k(\boldsymbol{\theta}_t; x_{k,i}) = \sum_{k=1}^{K} \frac{b_k}{b} \bar{\mathbf{g}}_{t,k}, \tag{2}$$

where $\bar{\mathbf{g}}_{t,k} = \frac{1}{b_k} \sum_{i=1}^{b_k} \nabla \ell_k(\boldsymbol{\theta}_t; x_{k,i})$ is the average gradient from task k, and $x_{k,i} \sim \mathcal{D}_k$. This formulation highlights how batch gradients blend contributions from different tasks and serves as the foundation for analyzing task interactions.

A central challenge in prior MTL literature is *gradient conflict* [44, 63, 73], where task gradients oppose the overall update direction. Formally, a conflict exists if: $\langle \mathbf{g}_t, \bar{\mathbf{g}}_{t,k} \rangle < 0$, indicating that the shared update could increase the loss for task k.

While such conflicts are well-documented in vision tasks, we observe they are rare in large-scale language model pretraining. In particular, as shown in Figures 2, 5, and 6, task gradients are generally non conflicting in pretraining LLMs. This observation shifts the our goal: instead of mitigating conflict, we can **leverage naturally non-conflicting gradients** to improve training efficiency.

This insight renders many conflict-aware MTL methods—such as PCGrad [73], AdaTask [71], MGDA [53], and NashMTL [44]—less effective for pre-training LLMs: PCGrad acts only when conflicts occur (rare in LLMs) and thus performs similarly to simple *Mix*. AdaTask treats different tasks as completely independent and thus does not utilizes potential cooperative nature of tasks, leading to slower convergence. More fundamentally, these methods incur substantial memory and compute overhead, making them unsuitable for large-scale models (see Appendix D for discussions). A notable exception is the recent work FAMO [35], which is designed for scalability.

In summary, the lack of gradient conflict in LLM pretraining, combined with the limitations of existing MTL approaches, motivates the need for scalable methods that exploit non-conflicting gradient structures—rather than fixating on rare conflicts.

3. Method

3.1. PiKE: Conceptual Version

To develop our method, we first start by quantifying gradient conflicts:

Definition 1 *For a given point $\boldsymbol{\theta}$, we say gradients are $\underline{c}$-conflicted (with $\underline{c} \geq 0$) if, for all task pairs $j, k, j \neq k$,*

$$-\underline{c}\big(\|\nabla \mathcal{L}_j(\boldsymbol{\theta})\|^2 + \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2\big) \leq \langle \nabla \mathcal{L}_j(\boldsymbol{\theta}), \nabla \mathcal{L}_k(\boldsymbol{\theta}) \rangle \,.$$

The above definition is implied by a lower bound on the gradients cosine similarity. In particular, if $\frac{\langle \nabla \mathcal{L}_j(\boldsymbol{\theta}), \nabla \mathcal{L}_k(\boldsymbol{\theta}) \rangle}{\|\mathcal{L}_j(\boldsymbol{\theta})\| \|\mathcal{L}_k(\boldsymbol{\theta})\|} \geq -\tilde{c}$, then the gradients are $\underline{c}$-conflicted for $\underline{c} = \tilde{c}/2$. Therefore, experiments in Section 2 and Figures 5 and 6 in Appendix I.2 show that $\underline{c}$ is typically small for LLM training.

While task gradient conflict are rare in LLM training, we observed in Section 2 that task gradients are not fully aligned either. To quantify the level of alignment, we define the following concept:

Definition 2 *For a given point $\boldsymbol{\theta}$, we say that the gradients are $\bar{c}$-aligned (with $\bar{c} \geq 0$) if, for all task pairs $j, k, j \neq k$,*

$$\langle \nabla \mathcal{L}_j(\boldsymbol{\theta}), \nabla \mathcal{L}_k(\boldsymbol{\theta}) \rangle \leq \bar{c} \|\nabla \mathcal{L}_j(\boldsymbol{\theta})\|_2 \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|_2.$$

Notice that Theorem 1 and 2 always hold for $\bar{c} = 1$ and $\underline{c} = 1/2$. However, smaller values allow for more refined analysis and more desirable properties. Notably, when both $\bar{c}$ and $\underline{c}$ are small (as we observed in our experiments), the value of $\|\nabla \mathcal{L}(\boldsymbol{\theta})\|$ is small if and only if $\|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|$ is small for all k (see Lemma 7 in Appendix J). To proceed further, we make the following standard assumption:

Assumption 3 *Per-task gradients are L-Lipschitz, unbiased, and have bounded variance, i.e., $\forall k$:*

$$\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_1) - \nabla \mathcal{L}_k(\boldsymbol{\theta}_2)\| \leq L\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|, \quad \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \tag{3}$$

$$\mathbb{E}_{x\sim\mathcal{D}_k}[\nabla \ell_k(\boldsymbol{\theta}; x)] = \nabla \mathcal{L}_k(\boldsymbol{\theta}), \quad \forall \boldsymbol{\theta} \tag{4}$$

$$\mathbb{E}_{x\sim\mathcal{D}_k}[\|\nabla \ell_k(\boldsymbol{\theta}; x) - \nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2] \leq \sigma_k^2, \quad \forall \boldsymbol{\theta} \tag{5}$$

Using a mix batch with b_k samples per task k, the estimated gradient follows equation (2). The next theorem characterizes the descent amount in one step of SGD under low conflict conditions:

Theorem 4 *Assume Assumption 3 holds and the gradients at $\boldsymbol{\theta}_t$ are $\underline{c}$-conflicted and $\bar{c}$-aligned, with $\underline{c} < \frac{1}{K-2+b/b_k}$ for all k. If gradients are computed using the Mix strategy equation (2), then:*

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1})] \leq \mathcal{L}(\boldsymbol{\theta}_t) + \sum_{k=1}^{K} b_k\Big(-\frac{\eta}{b}\beta\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \frac{L\eta^2}{2b^2}\sigma_k^2\Big) + \sum_{k=1}^{K} b_k^2 \frac{L\eta^2}{2b^2}\gamma\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 \tag{6}$$

for SGD update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \mathbf{g}_t$. Here, $\beta \triangleq \min_k(1 + \underline{c}(-K + 2 - \frac{b}{b_k}))$, $\gamma \triangleq 1 + \bar{c}(K-1)$, and the expectation is over batch sampling randomness under the mix strategy $(b_1, \ldots, b_K)$.

A formal proof is provided in Theorem 9 (Appendix J). Theorem 4 provides an upper-bound on the decrease in the objective depending on the mix batch composition $(b_1, \ldots, b_K)$. As we show in Theorem 10 in the appendix, this upperbound is tight in the non-conflicting, non-aligned regime of $\bar{c} = \underline{c} = 0$ (which is a good approximation for pre-training LLMs according to our experiments).

According to Theorem 4, to maximize descent in Mix sampling, we need to minimize the RHS of equation (6). Thus, relaxing $w_k = b_k/b$ to continuous values, we need to solve:

$$\min_{w_1,\ldots,w_K \geq 0} \sum_{k=1}^{K} w_k\lambda_k + \frac{1}{2}w_k^2\kappa_k \quad \text{s.t.} \quad \sum_{k=1}^{K} w_k = 1 \tag{7}$$

where $\lambda_k \triangleq -\eta\beta\|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2 + \frac{L\eta^2}{2b}\sigma_k^2$ and $\kappa_k \triangleq L\eta^2\gamma\|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2$. Using KKT conditions, the optimal solution is given by $w_k^* = \max\left\{0, -\frac{\mu+\lambda_k}{\kappa_k}\right\}$, where μ is chosen such that $\sum_{k=1}^{K} w_k^* = 1$ (see Lemma 8, Appendix J). This leads to the conceptual version of PiKE (Positive gradient Interactions-based K-task weight Estimator), summarized in Algorithm 2 in Appendix F.

The conceptual version of PiKE (Algorithm 2) adaptively adjusts sampling weights. This adaptive adjustment makes the stochastic gradients biased, i.e., $\mathbb{E}[\mathbf{g}_t] \neq \nabla \mathcal{L}(\boldsymbol{\theta}_t)$. Due to this introduced bias, the classical convergence results of SGD can no longer be applied. The following theorem establishes the convergence of conceptual PiKE:

Theorem 5 *Suppose the assumptions in Theorem 4 hold and the Conceptual PiKE Algorithm (Algorithm 2 in the Appendix) initialized at $\boldsymbol{\theta}_0$ with the SGD optimizer in Step 10 of the algorithm. Let $\Delta_L = \mathcal{L}(\boldsymbol{\theta}_0) - \min_{\boldsymbol{\theta}} \mathcal{L}(\theta)$ and $\sigma_{\max} = \max_k \sigma_k$. Suppose $\delta > 0$ is a given constant and the stepsize $\eta \leq \frac{\beta\delta}{L\sigma_{\max}^2/b + L\gamma\delta}$. Then, after $T = \frac{2\Delta_L}{\eta\beta\delta}$ iterations, Algorithm 2 finds a point $\bar{\boldsymbol{\theta}}$ such that*

$$\mathbb{E}\|\nabla \mathcal{L}_k(\bar{\theta})\|^2 \leq \delta, \quad \forall k = 1, \ldots, K. \tag{8}$$

Particularly, if we choose $\eta = \frac{\beta\delta}{L\sigma_{\max}^2/b + L\eta\delta}$, then the Conceptual PiKE algorithm requires at most $\bar{T} = \frac{2L\Delta_L(\sigma_{\max}^2/b + \gamma\delta)}{\delta^2\beta^2}$ iterations to find a point satisfying equation (8).

This theorem (restated as Theorem 11) is proved in Appendix J.

Remark 1. Theorem 5 guarantees that, given enough iterations, all tasks are learned jointly, i.e., the gradients of all losses become small. Moreover, the convergence rate is $T = O(1/\delta^2)$, which matches the optimal iteration complexity for smooth, nonconvex stochastic optimization.

Remark 2. The Conceptual PiKE algorithm maximizes the expected decrease at each iteration by finding the optimal mix batching strategy. This property leads to improved iteration complexity bound than (static) uniform mix batching, as discussed in Appendix J.3, when $\underline{c}$ and $\bar{c}$ are small.

3.2. PiKE: Simplified Computationally Efficient Version

Solving equation (7) requires estimating $\{\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\}_{k=1}^K$ and $\{\sigma_k\}_{k=1}^K$; however, per-iteration estimation of these terms with sufficient accuracy often necessitates large batch computations, thereby impeding convergence. To speed up the algorithm, we can update these estimates every T_0 iterations. However, this can cause abrupt changes in $(w_1, \ldots, w_K)$, leading to instability, especially with optimizers like Adam, where sudden shifts may disrupt momentum estimates. To mitigate this, we update $(w_1, \ldots, w_K)$ using a single mirror descent step on equation (7), ensuring gradual adjustments:

$$w_k \leftarrow w_k \exp\left(\alpha\eta(\beta - L\eta\gamma w_k)\, \overset{\text{gradient norm}}{\|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2} - \frac{\alpha L \eta^2}{2b} \overset{\text{gradient variance}}{\sigma_k^2}\right)$$

followed by normalization: $\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|_1$, where α is the mirror descent step size.

Even after the above simplifications, fine-tuning L, γ, α, and β can be challenging in practice. We simplify this finetuning by noting two observations: 1) The coefficient in front of σ_k^2 is constant, independent of w_k. 2) For small η and $w_k < 1$, the coefficient in front of $\|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|$ remains nearly constant: $\alpha\eta(\beta - L\eta\gamma w_k) \approx \alpha\eta\beta$. Therefore, in our practical implementation, PiKE employs tunable constant coefficients for terms related to task-specific gradient variance and gradient norms, which simplifies its application. The resulting algorithm is detailed in Algorithm 1.

Remark 3. The inclusion of the variance-related term, σ_k^2, is crucial in PiKE; its omission would cause PiKE to disproportionately prioritize tasks with currently large losses even if their gradients are very noisy (high variance). Our ablation study, presented in Appendix I.7, underscores the importance of this term for achieving robust and balanced multi-task performance.

Remark 4. PiKE updates sampling weights only every T_0 steps. In our experiments, this additional step incurs negligible memory and computational overhead (see Appendix I.4).

4. Experiments

We evaluate PiKE in two multitask pretraining scenarios: 1) *Pretraining language models on multilingual mC4 dataset* [69], a dataset covering diverse languages from Common Crawl corpus. 2) *Pretraining language models on the GLaM dataset* [17], an English dataset spanning six domains. As we will show, across multiple model sizes (110M, 270M, 750M, and 1B parameters), PiKE consistently outperforms all existing methods, including heuristic or static data mixing, multitask optimization, and adaptive data mixture approaches. We explain our setup in Appendix H.

4.1. Main Observations on Multilingual Pretraining Experiments

Table 1 presents results for pretraining a 1B multilingual GPT-2 model [46] on English, Hindi, and German. Mix batching outperforms Round-Robin and Random strategies, supporting our analysis

Algorithm 1 PiKE: Positive gradient Interaction-based K-task weights Estimator

Input: $\boldsymbol{\theta}$, T_0, total batch size b, task k dataset $\mathcal{D}_k$, hyperparameters ζ_1 and ζ_2, prior weights $\mathbf{w}'$
Initialize: $w_k \leftarrow 1/K$ or $w_k \leftarrow w'_k$
for $t = 0, 1, \ldots$ **do**
  if $t \mod T_0 = 0$ **then**
    Estimate $\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2$ and σ_k^2 for every k
    $w_k \leftarrow w_k \exp\left(\zeta_1 \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 - \frac{\zeta_2}{2b}\sigma_k^2\right)$
    $\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|_1$
    $(b_1, \ldots, b_K) \leftarrow \text{round}(b(w_1, \ldots, w_K))$
  end if
  Sample b_k data points from each task k
  Compute the gradient $\mathbf{g}$ using the estimates samples
  Update: $\boldsymbol{\theta}_{t+1} \leftarrow \text{Optimizer}(\eta, \boldsymbol{\theta}_t, \mathbf{g})$
end for

in Section A and justifying our focus on Mix as the foundation for PiKE. Additional results with different model sizes (270M and 1B) and language settings are reported in Table 13. Our main observation is that *PiKE and its Balanced variant achieve the highest average downstream accuracy* across all language settings and model sizes, demonstrating their effectiveness for multilingual pretraining.

4.2. Main Observations on Pretraining Experiments with GLaM Datasets

Table 2 shows results for pretraining a 750M GPT-2 model on the GLaM dataset. Additional results with both 110M and 750M models across six domains are provided in Table 14. Across both 110M and 750M model sizes, *PiKE consistently outperforms* DoReMi, GLaM, and Mix in downstream accuracy. For the 750M model, PiKE improves average accuracy by **3.4%** over DoReMi, **6.2%** over GLaM, **7.1%** over FAMO, and **4.8%** over ADO. In the 110M setting, PiKE achieves **37.8%** accuracy, exceeding DoReMi (**36.0%**), GLaM (**35.3%**), and FAMO (**35.9%**). Unlike DoReMi, which requires a separate proxy model, or GLaM, which depends on tuning with smaller models, PiKE delivers these gains with negligible additional overhead.

Mixing datasets improves language model generalization. We compare models trained on individual domains to those trained on mixed-domain datasets. Table 14 shows that single-domain training underperforms compared to mixed-domain training, even with simple Mix sampling. This reinforces the importance of diverse data for pretraining and aligns with prior work [26, 37].

5. Conclusion

We introduced *PiKE*, an adaptive data mixing algorithm that exploits non-conflicting gradients in large-scale language model pretraining to dynamically adjust task sampling. Experiments show PiKE improves convergence and downstream performance, while our extension, *Balanced-PiKE*, enhances fairness by reducing task-level performance gaps. Future work will focus on incorporating dataset size into the sampling strategy and extending PiKE to new domains.

References

[1] Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Hao Ban and Kaiyi Ji. Fair resource allocation in multi-task learning. *arXiv preprint arXiv:2402.15638*, 2024.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[5] Y Bisk, R Zellers, R Le Bras, J Gao, and Y Choi. Reasoning about physical commonsense in natural language, 2019.

[6] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018.

[9] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[10] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

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

[12] Viet Dac Lai, Chien Van Nguyen, Nghia Trung Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan A Rossi, and Thien Huu Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. *arXiv e-prints*, pages arXiv–2307, 2023.

[13] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3150–3158, 2016.

[14] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. *arXiv preprint arXiv:2302.05442*, 2023.

[15] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

[16] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

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

[19] Colin Gaffney, Dinghua Li, Ruoxin Sang, Ayush Jain, and Haitang Hu. Orbax, 2023. URL http://github.com/google/orbax.

[20] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[21] Ce Ge, Zhijian Ma, Daoyuan Chen, Yaliang Li, and Bolin Ding. Data mixing made efficient: A bivariate scaling law for language model pretraining. *arXiv preprint arXiv:2405.14908*, 2024.

[22] Google. Grain - feeding jax models, 2023. URL http://github.com/google/grain.

[23] Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. Olmes: A standard for language model evaluations. *arXiv preprint arXiv:2406.08446*, 2024.

[24] Kelvin Guu, Albert Webson, Ellie Pavlick, Lucas Dixon, Ian Tenney, and Tolga Bolukbasi. Simfluence: Modeling the influence of individual training examples by simulating training runs. *arXiv preprint arXiv:2303.08114*, 2023.

[25] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL http://github.com/google/flax.

[26] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35:30016–30030, 2022.

[27] Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. *arXiv preprint arXiv:2410.11820*, 2024.

[28] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[29] Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, et al. The bigscience roots corpus: A 1.6 tb composite multilingual dataset. *Advances in Neural Information Processing Systems*, 35:31809–31826, 2022.

[30] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

[31] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations (ICLR)*, 2018. https://arxiv.org/abs/1804.08838.

[32] Tian Li, Ahmad Beirami, Maziar Sanjabi, and Virginia Smith. Tilted empirical risk minimization. *arXiv preprint arXiv:2007.01162*, 2020.

[33] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[34] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.

[35] Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[36] Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In *International Conference on Machine Learning*, pages 22188–22214. PMLR, 2023.

[37] Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*, 2024.

[38] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Association for Computational Linguistics*, 2015.

[39] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. https://openreview.net/forum?id=Bkg6RiCqY7.

[41] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[42] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[43] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on networking*, 8(5):556–567, 2000.

[44] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022.

[45] 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.

[46] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019. https://openai.com/blog/better-language-models/.

[47] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[48] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[49] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. {ZeRO-Offload}: Democratizing {Billion-Scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564, 2021.

[50] Ralph Tyrell Rockafellar. Convex analysis:(pms-28). *Princeton university press*, 2015.

[51] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4822–4829, 2019.

[52] Noveen Sachdeva, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed H Chi, James Caverlee, Julian McAuley, and Derek Zhiyuan Cheng. How to train data-efficient llms. *arXiv preprint arXiv:2402.09668*, 2024.

[53] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[54] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*, 2024.

[55] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[56] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

[57] Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.

[58] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[59] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[60] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3614–3633, 2021.

[61] Huy V Vo, Vasil Khalidov, Timothée Darcet, Théo Moutakanni, Nikita Smetanin, Marc Szafraniec, Hugo Touvron, Camille Couprie, Maxime Oquab, Armand Joulin, et al. Automatic data curation for self-supervised learning: A clustering-based approach. *arXiv preprint arXiv:2405.15613*, 2024.

[62] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

[63] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *arXiv preprint arXiv:2010.05874*, 2020.

[64] Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. Qurating: Selecting high-quality data for training language models. *arXiv preprint arXiv:2402.09739*, 2024.

[65] Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. Small-scale proxies for large-scale transformer training instabilities. *arXiv preprint arXiv:2309.14322*, 2023.

[66] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

[67] 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, 2024.

[68] Derrick Xin, Behrooz Ghorbani, Justin Gilmer, Ankush Garg, and Orhan Firat. Do current multi-task optimization methods in deep learning even help? *Advances in neural information processing systems*, 35:13597–13609, 2022.

[69] L Xue. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[70] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. corr, abs/2105.13626. *arXiv preprint arXiv:2105.13626*, 2021.

[71] Enneng Yang, Junwei Pan, Ximei Wang, Haibin Yu, Li Shen, Xihua Chen, Lei Xiao, Jie Jiang, and Guibing Guo. Adatask: A task-aware adaptive learning rate approach to multi-task learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 10745–10753, 2023.

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

[73] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

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

[75] Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need adam: A hessian perspective. *arXiv preprint arXiv:2402.16788*, 2024.

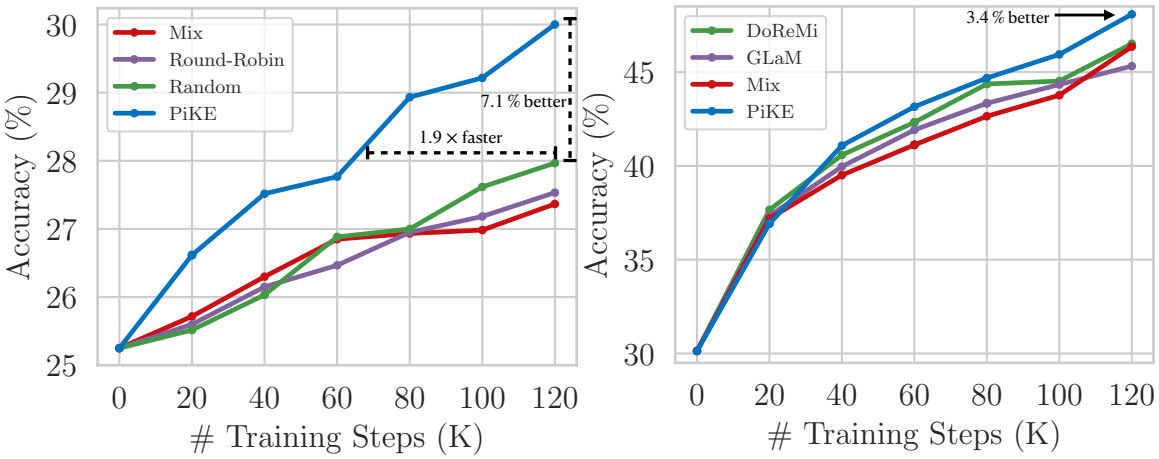

Figure 1: PiKE **adaptively optimizes task weights** in pre-training, outperforming baselines. **Left (1B models, multilingual C4 en/hi):** PiKE boosts average downstream accuracy by 7.1% and reaches baseline accuracy 1.9× faster. **Right (750M models, GLaM six domains):** PiKE achieves a 3.4% higher accuracy over DoReMi [67]. PiKE scales efficiently with model size and number of tasks. Detailed results are in Tables 13 and 14.

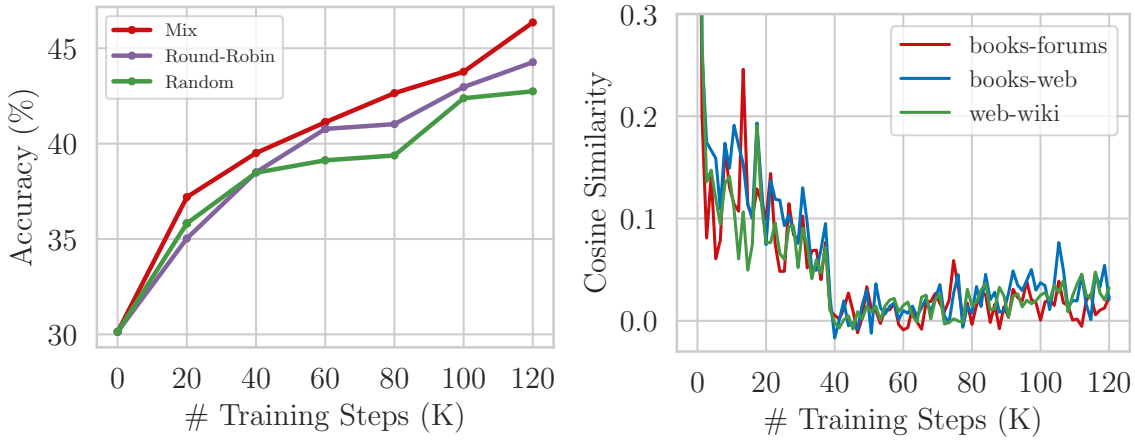

Figure 2: **Left:** Average accuracy on four downstream tasks (ArcE, CSQA, HellaSwag, PIQA) for 750M GPT-2 style models pre-trained with *Mix*, *Round-Robin*, and *Random* strategies. See Appendix I.1 for more. **Right:** Task gradient cosine similarity for a 750M GPT-2 style model pre-trained on GLaM datasets. "*data1-data2*" indicates gradient similarity between tasks *data1* and *data2*. Further results in Appendix I.2.

Appendix A. Additional Building Blocks for PiKE

Bulding Block #2: Mixing Domains per Batch Improves LLM Generalization

Optimizing equation (1) with stochastic methods, such as Adam or SGD, requires forming batches from one or more tasks at each step. The batch selection strategy strongly influences model performance [4, 21, 37, 67, 72]. Even with fixed data proportions, a key question remains: *how should one form a batch from K data domains at each step?*

Batch construction is a critical design choice in multitask training, as it directly impacts learning dynamics and final model performance [4, 21, 37, 67, 72]. We focus on three standard strategies:

Table 1: We report the perplexities (lower the better) on the validation split of multilingual C4 datasets. We also report the accuracies (%, higher the better) of different models on HellaSwag and its corresponding translated version. **Bolding** indicates the best model in the task; Metrics means the average across different tasks. Additional results can be found in Table 13.

		C4 (en)	C4 (hi)	C4 (de)		HellaSwag (en)	HellaSwag (hi)	HellaSwag (de)
	Perplexity ↓	Perplexity ↓	Perplexity ↓	Perplexity ↓	Accuracy(%) ↑	0-shot ↑	0-shot ↑	0-shot ↑
C4 (en), C4 (hi), and C4 (de) datasets, GPT-2 large style, 1B params, 36 Layers default, 120K training steps								
Mix	8.29	11.13	**4.45**	9.29	27.5	28.1	27.1	27.6
Round-Robin	8.41	11.31	4.97	9.46	26.5	27.6	26.7	26.3
Random	8.48	11.38	4.54	9.55	26.6	27.0	26.9	26.1
FAMO [35]	8.25	11.04	4.48	9.23	27.2	27.3	26.9	27.3
ADO [27]	8.30	11.12	4.45	9.31	27.5	27.7	**27.5**	27.2
PiKE	9.56	**9.49**	5.32	13.87	28.7	**33.0**	27.2	26.2
Balanced-PiKE ($\tau = 1$)	8.29	11.12	4.46	9.31	27.9	28.3	27.4	28.0
Balanced-PiKE ($\tau = 3$)	**8.18**	10.14	4.93	9.49	**28.9**	31.3	27.3	28.1
Balanced-PiKE ($\tau = 5$)	8.42	10.02	6.30	**8.94**	**28.9**	31.2	26.9	**28.6**

Table 2: We report perplexity (lower is better) on the validation split of the GLaM datasets, averaging perplexities across six domains. We also compare the accuracies (%, higher the better) of different models on four different Q/A tasks. PiKE (Uniform) means PiKE using initial uniform sampling weights and PiKE (GLaM) means PiKE using GLaM tuned weights as initial weights. **Bolding** indicates the best model in the task, underlining indicates PiKE beating Mix, Round-Robin, Random methods. Additional detailed (per-domain) results with different model sizes can be found in Table 14.

	GLaM		ArcE	CSQA	HellaSwag	PIQA
	Perplexity ↓	Accuracy(%) ↑	7-shot ↑	7-shot ↑	7-shot ↑	7-shot ↑
Six domains of GLaM dataset, GPT-2 large style, 750M params, 36 layers default						
Mix	**12.77**	46.4	47.2	39.6	37.9	60.9
Round-Robin	12.98	44.3	43.5	36.7	36.8	60.3
Random	12.99	42.7	41.7	34.2	36.6	58.2
FAMO [35]	13.25	45.0	43.7	40.0	36.4	59.8
ADO [27]	**12.77**	45.9	45.5	38.7	38.1	61.1
GLaM [17]	13.20	45.3	46.9	39.8	38.0	56.4
DoReMi [67]	13.25	46.5	48.6	40.1	37.5	59.6
PiKE (Uniform)	13.22	47.6	49.6	43.2	37.2	60.4
PiKE (GLaM)	13.35	48.1	**49.8**	**43.5**	38.0	61.2
Balanced-PiKE ($\tau = 1$)	13.21	47.5	48.8	42.5	37.6	61.2
Balanced-PiKE ($\tau = 3$)	13.26	47.2	48.8	41.5	37.2	61.3
Balanced-PiKE ($\tau = 5$)	13.19	**48.2**	49.3	42.6	**38.5**	**62.4**

Random, *Round-Robin*, and *Mix*. Let us first define these methods assuming static, uniform sampling weights (i.e., $1/K$ per task). Let $\mathbf{b}_t = (b_{t,1}, \ldots, b_{t,K})$ denote the number of samples from each task $\mathcal{D}_k$ at iteration t, with total batch size $b = \sum_{k=1}^{K} b_{t,k}$. The strategies are defined as:

$$\mathbf{b}_t = \begin{cases} b \cdot \mathbf{e}_{k^*} & \textit{Random}, \text{ where } k^* \sim \text{Uniform}(\{1, \ldots, K\}) \\ b \cdot \mathbf{e}_{(t \bmod K)+1} & \textit{Round-Robin} \\ b \cdot \left(\frac{1}{K}, \ldots, \frac{1}{K}\right) & \textit{Mix} \end{cases}$$

where $\mathbf{e}_k \in \mathbb{R}^K$ is the k-th standard basis vector. That is, *Random* selects one domain per batch, *Round-Robin* cycles through domains, and *Mix* includes all tasks in each batch.

Historically, *Mix* has been widely used in computer vision multitask settings [8, 13, 35, 42, 51, 73], while earlier language modeling efforts favored *Random* or *Round-Robin* [38, 39, 41]. Recent studies on large-scale language models [7, 16, 48, 58] have revisited this question and found that *Mix* generally performs best—especially in diverse-data settings [10, 17, 20, 62, 67]. Our experiments reaffirm this trend: across a wide range of tasks and scales, *Mix* consistently outperforms the alternatives (Figure 2, Figure 4). This observation underpins the design of our proposed method, *PiKE*.

Bulding Block #3: Adaptively Changing Mix Sampling Weights Is Necessary

Prior work using the Mix sampling strategy typically relies on fixed (static) sampling weights, keeping $(b_1, \ldots, b_K)$ constant throughout training. However, dynamically adjusting batch composition can significantly enhance efficiency. We illustrate this fact with a simple example:

Example 1 *Consider training on $K = 2$ tasks with losses $\ell_1(\boldsymbol{\theta}; x_1) = \frac{1}{2}(\boldsymbol{\theta}^\top e_1)^2 + x_1^\top \boldsymbol{\theta}$ and $\ell_2(\boldsymbol{\theta}; x_2) = \frac{1}{2}(\boldsymbol{\theta}^\top e_2)^2 + x_2^\top \boldsymbol{\theta}$, where $e_1 = [1\ 0]^\top$, $e_2 = [0\ 1]^\top$, and $\boldsymbol{\theta} \in \mathbb{R}^2$. Data for task 1 follows $x_1 \sim \mathcal{N}(0, \sigma_1^2 I)$, while task 2 follows $x_2 \sim \mathcal{N}(0, \sigma_2^2 I)$. The loss for task k simplifies to $\mathcal{L}_k(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta}^\top e_k)^2$. Using b_k samples from task k, the gradient at iteration t is given by*

$$\mathbf{g}_t = \frac{1}{b_1 + b_2}\left(b_1 e_1 e_1^\top + b_2 e_2 e_2^\top\right)\boldsymbol{\theta}_t + \mathbf{z},$$

where $\mathbf{z} \sim \mathcal{N}(0, \frac{b_1\sigma_1^2 + b_2\sigma_2^2}{b^2} I)$ with $b = b_1 + b_2$. Updating $\boldsymbol{\theta}_t$ via SGD, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \mathbf{g}_t$, we have

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1})] = \frac{1}{2}(1 - \eta\frac{b_1}{b})^2\theta_{1,t}^2 + \frac{1}{2}(1 - \eta\frac{b_2}{b})^2\theta_{2,t}^2 + \eta^2\frac{b_1\sigma_1^2 + b_2\sigma_2^2}{b^2}, \tag{9}$$

where $\theta_{1,t}$ and $\theta_{2,t}$ denote the first and second component of the vector $\boldsymbol{\theta}_t$. The derivation details of equation (9) can be found in Appendix J.1. Letting $w_1 := \frac{b_1}{b}$, $w_2 := \frac{b_2}{b}$, and relaxing them to take real values, we can optimize the mixing weights w_1 and w_2 as

$$w_1^* = \Pi\left(\frac{b^{-1}(\sigma_2^2 - \sigma_1^2) + \eta^{-1}(\theta_{1,t}^2 - \theta_{2,t}^2) + \theta_{2,t}^2}{\theta_{1,t}^2 + \theta_{2,t}^2}\right) \tag{10}$$

and $w_2^* = 1 - w_1^*$ where $\Pi(\xi) = \min\{\max\{\xi, 0\}, 1\}$ is the projection onto $[0, 1]$. This result shows that ***optimal batch composition should change over iterates*** *to maximize training efficiency.*

Figure 3 illustrates the superiority of the adaptive mixing approach based on equation (10) over various static mixing strategies. Moreover, the adaptive mixing strategy in this example does not require any hyperparameter tuning, while finding the best static mixing requires tuning.

Despite its simplicity, this example mirrors key aspects of MTL in large models: 1) The optimal solution $\boldsymbol{\theta}^* = 0$ minimizes all task losses simultaneously, reflecting the high expressive power of large models. 2) Task gradients are non-conflicting, resembling real-world gradient interactions observed in Building Block #2. Moreover, equation (10) further reveals that optimal data mixing depends on (1) the gradient norm squared per task $\|\nabla\mathcal{L}_1(\boldsymbol{\theta})\|^2 = \theta_1^2$, $\|\nabla\mathcal{L}_2(\boldsymbol{\theta})\|^2 = \theta_2^2$ and (2) gradient variance (σ_1^2, σ_2^2). As we will see in the next section, these factors play a crucial role in defining optimal mixing strategies for more general settings.

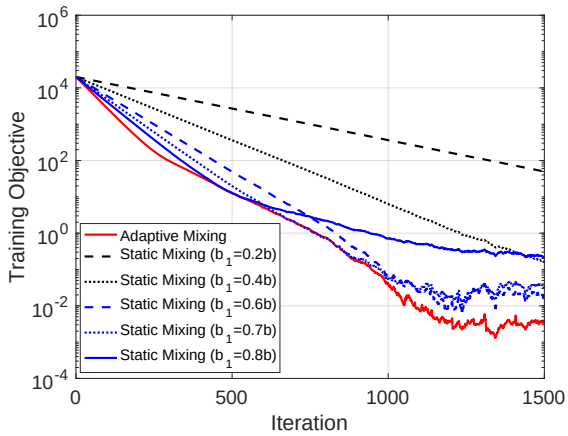

Figure 3: Adaptive mixing consistently outperforms static mixing in the Example 1.

Appendix B. Balanced-PiKE: Balanced Learning Across Different Tasks

While Algorithm 1 optimizes the average loss across tasks equation (1), this objective can lead to unbalanced learning. In practice, some tasks may dominate due to larger datasets or easier loss reduction, causing others to be under-optimized. This imbalance is problematic when certain domains (e.g., code, math) are more important to user, or when balanced performance across all tasks is desired. To address this shortcoming, we consider a balance-promoting objective based on *tilted empirical risk minimization* [32], also known as the α*-fairness utility* [43]:

$$\min_{\boldsymbol{\theta}} \quad \widetilde{\mathcal{L}}(\tau;\boldsymbol{\theta}) := \frac{1}{\tau}\log\left(\sum_{k=1}^{K} e^{\tau\mathcal{L}_k(\boldsymbol{\theta})}\right). \tag{11}$$

This objective interpolates between average loss ($\tau \to 0$) and worst-case loss ($\tau \to \infty$), allowing users to control the trade-off between efficiency and fairness. Moderate values of τ ($\tau \approx 1 \sim 3$ in our experiments) encourage balanced learning without sacrificing overall performance.

To leverage our PiKE developments in solving equation (11), we need to connect it to our intital objective in equation (1). The following lemma achieves this through the use of Fenchel duality [50]:

Lemma 6 *Assume $\mathcal{L}_k(\boldsymbol{\theta}) > 0,\ \forall k$ and let $0 < \tau \in \mathbb{R}$. Then equation (11) is equivalent to solving*

$$\min_{\boldsymbol{\theta}} \quad \max_{\substack{\mathbf{y}\in\mathbb{R}_+^K \\ \sum_{k=1}^K y_k=\tau}} \quad \sum_{k=1}^{K} y_k\mathcal{L}_k(\boldsymbol{\theta}) - \sum_{k=1}^{K}\frac{y_k}{\tau}\log\left(\frac{y_k}{\tau}\right). \tag{12}$$

Moreover, for any fixed $\boldsymbol{\theta}$, the inner maximization problem is maximized at $y_k^\star = \frac{\tau e^{\tau\mathcal{L}_k(\boldsymbol{\theta})}}{\sum_{j=1}^K e^{\tau\mathcal{L}_j(\boldsymbol{\theta})}},\ \forall k$.

This lemma, which is proved in Appendix J.4, provides a natural alternative minimization algorithm for solving equation (11): Fixing $\mathbf{y}$, equation (12) reduces to a weighted minimization over tasks, where *regular PiKE sampling* with proper weights y_k in front of each loss can be applied to determine the optimal mixing strategy. On the other hand, fixing $\boldsymbol{\theta}$, the optimal solution $y_k^\star$ can be computed according to Theorem 6. This leads to *Balanced-PiKE* algorithm, described in Appendix G, which balances overall loss minimization and balanced/fair learning of all tasks. Although prior MTL literature explored fair learning [3, 44], such methods often grapple with scalability limitations, a critical aspect where PiKE is designed to excel through efficient large-scale performance.

Appendix C. Related Work

Data Curation and Selection. The effectiveness of language models heavily depends on the quality of the pre-training corpus. Consequently, significant efforts have been made to enhance pre-training data. These efforts include heuristic-based filtering [29, 45, 47, 48, 54] and deduplication [1, 9, 18, 30]. Recently, [61] proposed an automated method for constructing large, diverse, and balanced datasets for self-supervised learning by applying hierarchical k-means clustering. [52] introduced techniques that leverage instruction-tuned models to assess and select high-quality training examples, along with density sampling to ensure diverse data coverage by modeling the data distribution. Additionally, [24] simulated training runs to model the non-additive effects of individual training examples, enabling the analysis of their influence on a model's predictions.

Multitask Learning Optimization The approach most closely related to our method is multitask learning (MTL) optimization, which modifies gradient updates to mitigate gradient conflicts—situations where task gradients point in opposing directions, slowing down optimization [60, 73]. The Multiple Gradient Descent Algorithm (MGDA) [15, 53] updates the model by optimizing the worst improvement across all tasks, aiming for equal descent in task losses. Projected Gradient Descent (PCGrad) [73] modifies task gradients by iteratively removing conflicting components in a randomized order, ensuring that updates do not interfere destructively across tasks. Conflict-Averse Gradient Descent (CAGRAD) [34] optimizes for the worst task improvement while ensuring a decrease in the average loss. NASHMTL [44] determines gradient directions by solving a bargaining game that maximizes the sum of log utility functions. While these methods improve performance, they introduce significant computational and memory overhead, making them impractical for large-scale models with numerous tasks [68]. Similar challenges exist in AdaTask [71], which improves multitask learning by balancing parameter updates using task-wise adaptive learning rates, mitigating task dominance, and enhancing overall performance. Unlike previous approches that requires requiring $O(K)$ storage for task gradients (e.g. PCGrad) or optimizer states (e.g. AdaTask), FAMO [35] balances task loss reductions efficiently using $O(1)$ space and time. However, these methods fail to exploit the *non-conflicting* interactions among tasks, focusing instead on resolving conflicts that seldom arise. This highlights the need for a new approach that actively leverages lack of gradient conflicts to enhance training efficiency.

Another line of work focuses on adjusting the domain mixture to improve data efficiency during training [27, 66, 67]. However, these methods require a target loss for optimization, which has been shown to not always correlate with downstream performance [36, 57, 64]. In contrast, our method leverages the absence of gradient conflict and the presence of positive gradient interactions between tasks or domains. This approach provides a more reliable and effective way to enhance the final model's performance.

Appendix D. The Scalability Problem of Existing MTL Methods

Many existing multitask learning (MTL) methods [3, 15, 34, 44, 53, 63, 71, 73] require computing and storing all K task-specific gradients at each training iteration, where K is the number of tasks. While exceptions like FAMO [35] are designed for efficient scaling, the aforementioned general approach typically leads to $O(Kd)$ space complexity for storing gradients and $O(Kd)$ time complexity per iteration for their computation, where d denotes the number of model parameters. This cost is further exacerbated when these methods also involve solving auxiliary optimization problems to combine or re-weight gradients [68]. In stark contrast, when the overall loss is a simple

average of task-specific losses (e.g., $\mathcal{L} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k$), its gradient $\nabla\mathcal{L}$ can be computed via a single backward pass. This results in $O(d)$ space and time complexity for the gradient computation, with the task-aggregation step being effectively independent of K.

While early MTL research often focused on relatively small-scale vision models or language models with fewer than 100 million parameters, modern language models frequently exceed 100 billion parameters (100B). Storing the gradients or weights for such a model (assuming 32-bit floating-point precision) requires approximately 400 GB of memory. This capacity already surpasses that of a single high-performance GPU like the NVIDIA H100 (80 GB), necessitating at least five such GPUs for storing just one set of gradients or weights.

Applying typical prior MTL methods in this large-scale context becomes impractical. Storing K full gradients for a 100B parameter model would demand approximately $400K$ GB of memory, equivalent to $5K$ H100 GPUs. This scaling of memory and computational requirements—at least linear with the number of tasks K for gradient operations and often compounded by super-linear costs from auxiliary optimization steps—renders naive applications of many existing MTL methods infeasible for current large language models. Indeed, even if storing per-task gradients were feasible, Xin et al. [68] reported a significant reduction in training throughput (steps per second) for such MTL techniques even on small-scale vision models, let alone for modern LLMs (e.g., those with 1B+ parameters). Collectively, these challenges highlight the pressing need for novel MTL methodologies engineered for large-scale pre-training, demanding both exceptional scalability and minimal additional computational and memory overhead.

Appendix E. Discussion of Gradient Conflicts

GradVaccine [63] made a similar observation regarding low gradient conflicts among task gradients in multilingual BERT (178M parameters). We extend this finding to substantially larger autoregressive, decoder-only models (up to 1B parameters), which are more characteristic of current large-scale language modeling paradigms.

Our experiments demonstrate that task gradients in large-scale models indeed exhibit minimal conflicts. To illustrate, we conduct two pre-training experiments: (i) a 1B parameter GPT-2-style model [46] on the multilingual mC4 dataset [69] (covering six languages: English, Hindi, German, Chinese, French, and Arabic), and (ii) a 750M parameter model on The GLaM dataset [17] (English text from six diverse domains). Experimental details are provided in Appendix H. Figures 2 and 5 depict cosine similarity trends for task gradients, revealing several key observations: 1) Gradient similarity is initially high but generally decreases as training progresses. 2) In multilingual settings, gradient similarity correlates with linguistic proximity (e.g., English-German gradients align more closely), whereas gradients from GLaM's diverse domains exhibit more uniform positive alignment. 3) Task gradients rarely conflict—multilingual cosine similarities seldom drop below -0.1, and GLaM domain gradients remain predominantly positive.

Appendix F. PiKE: Conceptual Version

Here, we present the conceptual (basic) version of PiKE. As discussed in the main text, this approach lacks computational efficiency due to the frequent estimation of the norm and the variance of the per-task gradient.

Algorithm 2 Conceptual version of PiKE: Positive gradient Interaction-based K-task weights Estimator

Input: $\boldsymbol{\theta}$, total batch size b, stepsize η, task k dataset $\mathcal{D}_k$, constants β, L, γ, and prior weights $\mathbf{w}'$

Initialize: $w_k \leftarrow 1/K$ or $w_k \leftarrow w'_k, \forall k$
for $t = 0, 1, \ldots$ **do**
  Estimate $\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2$ and σ_k^2 for every k
  Compute $\lambda_k \triangleq -\eta\beta\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \frac{L\eta^2}{2b}\sigma_k^2$ and $\kappa_k \triangleq L\eta^2\gamma\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2$
  set $w_k^* = \max\{0, -\frac{\mu+\lambda_k}{\kappa_k}\}$ where μ is found (by bisection) such that $\sum_{k=1}^K w_k^* = 1$
  Set $(b_1, \ldots, b_K) \leftarrow \text{round}(b(w_1^*, \ldots, w_K^*))$
  Sample b_k data points from each task k
  Compute the gradient $\mathbf{g}$ using the estimates samples
  Update: $\boldsymbol{\theta}_{t+1} \leftarrow \text{Optimizer}(\eta, \boldsymbol{\theta}_t, \mathbf{g})$
end for

As discussed in section 3.2, this algorithm is computationally inefficient as it requires estimating $\nabla \mathcal{L}_k(\theta_t)$ and σ_k at each iteration. To improve efficiency, we introduced modifications that led to the development of the PiKE algorithm (Algorithm 1 in the main body).

Appendix G. Balanced-PiKE: Fairness Considerations Across Tasks

Here, we present the *Balanced-PiKE* algorithm in more detail. As discussed in the main body, the main difference with PiKE is that the fair version requires the computation of the coefficients

$$y_k^\star = \frac{\tau e^{\tau \mathcal{L}_k(\boldsymbol{\theta})-1}}{\sum_{k=1}^K e^{\tau \mathcal{L}_k(\boldsymbol{\theta})-1}}, \forall k$$

Then updating the sampling weights by

$$w_k \leftarrow w_k \exp\left((y_k^\star)^2 \zeta_1 \|\nabla \mathcal{L}_k(\mathbf{w})\|^2 - (y_k^\star)^2 \frac{\zeta_2}{2b}\sigma_k^2\right), \quad \forall k$$

The overall algorithm is summarized in Algorithm 3. For our experiments, we evaluate three different values of τ: 1, 3, and 5. A larger τ results in a stronger balancing effect between different tasks.

Algorithm 3 *Balanced-PiKE*: Balanced considerations across tasks

Input: $\boldsymbol{\theta}$, T_0, total batch size b, task k dataset $\mathcal{D}_k$, hyperparameters $\zeta_1\ \zeta_2, \tau$, prior weights $\mathbf{w}'$
Initialize: $w_k \leftarrow 1/K$ or $w_k \leftarrow w'_k$
for $t = 0, 1, \ldots$ **do**
  if $t \mod T_0 = 0$ **then**
    Estimate $\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2$, σ_k^2, and $\mathcal{L}_k(\boldsymbol{\theta}_t)$ for every k
    $y_k^\star = \frac{\tau e^{\tau \mathcal{L}_k(\boldsymbol{\theta})}}{\sum_{k=1}^K e^{\tau \mathcal{L}_k(\boldsymbol{\theta})}}$
    $w_k \leftarrow w_k \exp\left((y_k^\star)^2 \zeta_1 \|\nabla \mathcal{L}_k(\mathbf{w})\|^2 - (y_k^\star)^2 \frac{\zeta_2}{2b} \sigma_k^2\right)$
    $\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|_1$
    $(b_1, \ldots, b_K) \leftarrow \text{round}(b(w_1, \ldots, w_K))$
  end if
  Sample b_k data points from each task k
  Compute the gradient $\mathbf{g}$ using the estimates samples
  Update: $\boldsymbol{\theta}_{t+1} \leftarrow \text{Optimizer}(\eta, \boldsymbol{\theta}_t, \mathbf{g})$
end for

Appendix H. Experiments Setup

H.1. Dataset Details

Our experiments construct two primary scenarios for multitask learning: multilingual tasks and diverse task mixtures spanning multiple domains. We consider two widely-used datasets for our study: mC4 [69] and GLaM [17].

mC4 Dataset The mC4 dataset [69] is a multilingual text corpus derived from the Common Crawl web archive, covering a diverse range of languages. It has been widely used for pretraining multilingual models, such as mT5 [69] and ByT5 [70]. The dataset is curated by applying language-specific filtering to extract high-quality text, ensuring a balanced representation across languages. Mixture weights for training models on mC4 are often chosen based on token counts. In our cases, we mainly focus on English (en), Hindi (hi), and German (de). We report their details in Table 3.

Table 3: Partial statistics of the mC4 corpus, totaling 6.3T tokens.

ISO code	Language	Tokens (B)
en	English	2,733
hi	Hindi	24
de	German	347

GLaM Dataset The GLaM dataset [17] comprises English text from six distinct sources and has been used to train the GLaM series models and PaLM [10]. Mixture weights for GLaM training were determined based on small model performance [17], while [67] employed group distributionally robust optimization (Group DRO) to compute domain-specific weights. Table 4 summarizes the six domains in the GLaM dataset and the mixture weights selected by GLaM and DoReMi. We use these weights as oracle baselines for comparison with PiKE, which dynamically adjusts task weights over time using gradient information, unlike the fixed weights employed by GLaM and DoReMi.

Table 4: GLaM dataset [17] and fixed mixture weights used in GLaM [17] and DoReMi [67].

Dataset	Tokens (B)	Weight chosen by GLaM [17]	Weight chosen by DoReMi [67]
Filtered Webpages	143	0.42	0.51
Wikipedia	3	0.06	0.05
Conversations	174	0.28	0.22
Forums	247	0.02	0.04
Books	390	0.20	0.20
News	650	0.02	0.02

Table 5: Architecture hyperparameters for different model scales used in the paper. All models are GPT-2-like decoder-only architectures. The multilingual models employ a vocabulary size of 250K, whereas GLaM training uses a vocabulary size of 32K. Differences in the total number of parameters arise due to the variation in vocabulary sizes.

Size	# Params	Layers	Attention heads	Attention head dim	Hidden dim
GPT-2 small	110M/270M	12	12	64	768
GPT-2 large	750M/1B	36	20	64	1280

H.2. Training Details

Our experiments explore two distinct scenarios for multitask learning: multilingual training and diverse task mixtures spanning multiple domains. To achieve optimal results, we customize the training setups for each scenario and present them separately in this section. All training is performed from scratch.

Multilingual Training To address the complexities of tokenizing multilingual data, we utilize the mT5 tokenizer [69], which features a vocabulary size of 250K. Both GPT-2 small and GPT-2 large models are trained with a context length of 1024 and a batch size of 256. The AdamW optimizer [40] is employed with consistent hyperparameters and a learning rate scheduler. Additional details on hyperparameter configurations are provided in Appendix H.5.

GLaM Training For GLaM training, we use the T5 tokenizer [48], implemented as a SentencePiece tokenizer trained on the C4 dataset with a vocabulary size of 32,000. Both GPT-2 small and GPT-2 large models are trained with a context length of 1024 and a batch size of 256. The AdamW optimizer [40] is used, and additional details on hyperparameters is in Appendix H.5.

H.3. Model Architecture

The detailed architecture is summarized in Table 5. Our implementation utilizes pre-normalization [46] Transformers with qk-layernorm [14]. Consistent with [9], we omit biases, and the layernorm [2] value remains set to the Flax [25] default of 1e-6. Additionally, we incorporate rotary positional embeddings [55].

H.4. Experimental Resource

All experiments are conducted on 8 Google TPUv4. The training time for GPT-2 small and GPT-2 large models for 120K steps are approximately 1 day and 2 days per run, respectively.

Table 6: Hyperparameter settings for our experiments.

Hyperparameters	Values
Optimizer	AdamW ($\beta_1 = 0.95$, $\beta_2 = 0.98$)
Initial and final learning rate	$7e-6$
Peak learning rate	$7e-4$
Weight decay	0.1
Batch size	256
Context length	1024
Gradient clipping norm	1.0
Training step	$120,000$
Warm-up step	$10,000$
Schedule	Linear decay to final learning rate

Table 7: Hyperparameter settings for running PiKE (Algorithm 1).

Hyperparameters	Values
PiKE hyperparameter (ζ_1, ζ_2)	$\{(1e-2, 1e-3), (1.5e-2, 1e-3), (5e-2, 5e-3), (7.5e-2, 5e-3), (1e-1, 1e-2), (1.5e-1, 1e-2)\}$
Check interval T_0	1,000
Batch size for estimation	256

H.5. Hyper-parameters

Table 6 shows the detailed hyperparameters that we used in all our experiments. We also report our hyperparameters grid for tuning PiKE in Table 7.

H.6. Implementation Details

Our implementation builds upon the Nanodo training infrastructure [65], incorporating enhancements for efficiency. This framework relies on Flax [25], JAX [6], and TPUs [28].

To enable training of larger models, we shard both model and optimizer states, following the methodology of FSDP [49], and define these shardings during JIT compilation. Checkpointing is handled using Orbax [19], while deterministic data loading is facilitated by Grain [22].

For data loading, sequences are packed to avoid padding. When a sequence contains fewer tokens than the context length hyperparameter, an end-of-sequence token is appended. This differs from Nanodo [65], where both begin-of-sequence and end-of-sequence tokens are added.

H.7. Evaluation

Our evaluation adheres to the OLMES suite [23]. For multilingual downstream performance, we utilize the multilingual version of HellaSwag [12], which supports evaluations across 26 languages. English downstream tasks are assessed using ARC-Easy [11], CommonsenseQA [56], PIQA [5], and HellaSwag [74]. Unless specified otherwise, multilingual evaluations are performed in a 0-shot setting, while GLaM pretraining evaluations employ 7-shot in-context learning, with demonstration candidates separated by two line breaks. For HellaSwag and its translated variants, we evaluate the first 3,000 examples. For all other downstream tasks, evaluations are conducted on their respective

validation sets. In the case of multiple-choice tasks, different candidates are included in the prompt, and the average log-likelihood for each candidate is computed. The candidate with the highest score is then selected as the predicted answer.

Appendix I. Additional Experiment Results

I.1. Comparison of Performance Using Mix, Random, and Round-Robin Sampling Strategies

Figure 4 presents the average downstream accuracies of language models pre-trained using Mix, Random, and Round-Robin sampling strategies. In both multilingual pre-training and GLaM pre-training, the Mix sampling strategy consistently outperforms the other two. This motivates us its use in pre-training large language models.

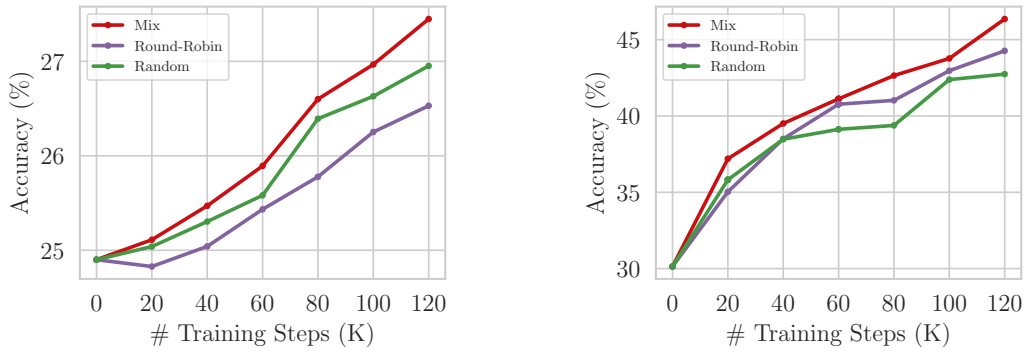

Figure 4: Average downstream task accuracy of pretraining language models using Mix, Round-Robin, and Random sampling strategies. Mix and Random use equal batch size for each task ($b_k = b/K, \forall k \in K$). **Left:** 1B models on multilingual C4 (en), C4 (hi), and C4 (de) datasets. **Right:** 750M models on GLaM datasets with six domains

I.2. Cosine Similarity and $\underline{c}$-Conflicted Gradients

Figures 5 and 6 show the cosine similarity, defined as $\frac{\langle \mathcal{L}_j(\boldsymbol{\theta}), \mathcal{L}_k(\boldsymbol{\theta}) \rangle}{\|\mathcal{L}_j(\boldsymbol{\theta})\| \|\mathcal{L}_k(\boldsymbol{\theta})\|}$ and the "ratio," defined as $\frac{\langle \mathcal{L}_j(\boldsymbol{\theta}), \mathcal{L}_k(\boldsymbol{\theta}) \rangle}{\|\mathcal{L}_j(\boldsymbol{\theta})\|^2 + \|\mathcal{L}_k(\boldsymbol{\theta})\|^2}$. In particular, if $\frac{\langle \nabla \mathcal{L}_j(\boldsymbol{\theta}), \nabla \mathcal{L}_k(\boldsymbol{\theta}) \rangle}{\|\mathcal{L}_j(\boldsymbol{\theta})\| \|\mathcal{L}_k(\boldsymbol{\theta})\|} \geq -\tilde{c}$, then the gradients are $\underline{c}$-conflicted for $\underline{c} = \tilde{c}/2$, which aligns with the observations in Figures 5 and 6.

I.3. Comparison of Performance Using PCGrad, AdaTask, and Mix

Figure 7 presents the average downstream task performance on HellaSwag (en) and HellaSwag (hi) for 270M multilingual language models pre-trained using PCGrad, AdaTask, and Mix. As shown in Figure 7: 1) PCGrad performs similarly to Mix, as it only adjusts gradients when conflicts occur—which is rare. 2) AdaTask converges more slowly due to noisy gradients and suboptimal optimizer state updates. Additionally, both methods are memory-intensive, requiring $O(K)$ storage for task gradients (PCGrad) or optimizer states (AdaTask), making them impractical for large-scale models such as the 540B PaLM [9].

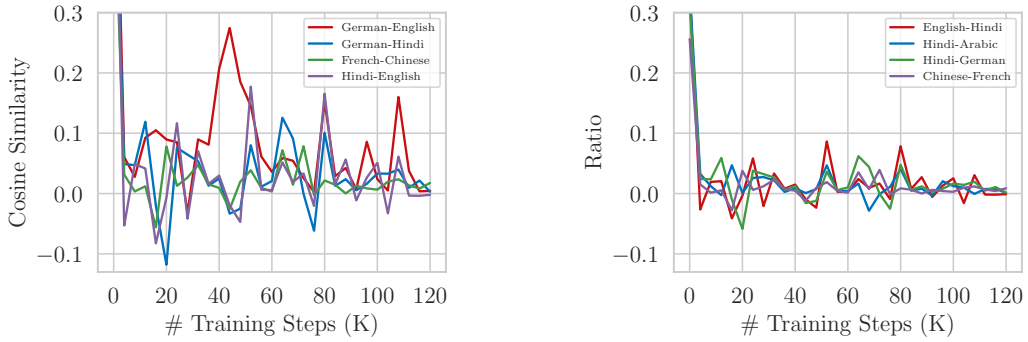

Figure 5: 1B models trained on multilingual mC4 datasets. **Left:** Cosine similarity between task gradients during language model pre-training over time. **Right:** The "ratio," which defined as $\frac{\langle \mathcal{L}_j(\boldsymbol{\theta}), \mathcal{L}_k(\boldsymbol{\theta}) \rangle}{\|\mathcal{L}_j(\boldsymbol{\theta})\|^2 + \|\mathcal{L}_k(\boldsymbol{\theta})\|^2}$, between task gradients during language model pre-training over time. "*data1-data2*" denotes the cosine similarity or ratio between the gradient of *data1* and the gradient of *data2*.

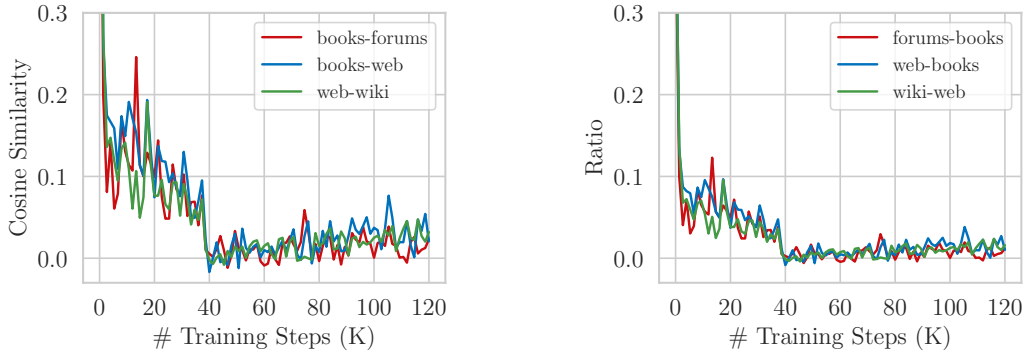

Figure 6: 750M models on GLaM datasets with six domains. **Left:** Cosine similarity between task gradients during language model pre-training over time. **Right:** The "ratio," which defined as $\frac{\langle \mathcal{L}_j(\boldsymbol{\theta}), \mathcal{L}_k(\boldsymbol{\theta}) \rangle}{\|\mathcal{L}_j(\boldsymbol{\theta})\|^2 + \|\mathcal{L}_k(\boldsymbol{\theta})\|^2}$, between task gradients during language model pre-training over time. "*data1-data2*" denotes the cosine similarity or ratio between the gradient of *data1* and the gradient of *data2*.

I.4. PiKE: Minimal Training and Memory Overhead

Table 8 reports the total training time and the computational overhead incurred by PiKE. In our experiments, per-task gradient squared norms and variances are estimated only once every $T_0 = 1,000$ training steps. Profiling results indicate that this infrequent estimation introduces minimal overhead during the pre-training of LLMs. Specifically, for a 1B parameter multilingual model pre-trained for approximately 71 hours, this estimation process accounted for only about 2.3% of the total training time (roughly 1.6 hours). This overhead percentage can be further reduced by using a larger estimation interval, T_0 (e.g., 5,000 steps), or by adopting a more efficient parallelization implementation for handling multiple tasks.

Regarding memory requirements, PiKE only needs to store the periodically estimated scalar values for each task's gradient variance and squared norm. This amounts to an additional memory footprint of $\mathcal{O}(K)$, which is negligible compared to the model parameters ($\mathcal{O}(d)$) and especially to the $\mathcal{O}(Kd)$ memory demanded by prior MTL methods that store K full per-task gradients.

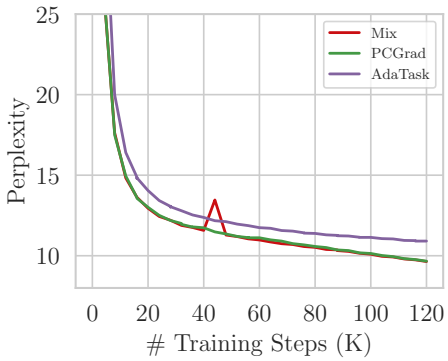

Figure 7: Eval perplexity of pretraining 270M GPT-2 style multilingual language models on mC4 datasets (English and Hindi) using Mix, PCGrad, and AdaTask.

This efficiency starkly contrasts with those earlier approaches, which also often involve solving auxiliary optimization problems that further increase computational and memory burdens [68]. Unlike approaches such as DoReMi or GLaM, PiKE simplifies deployment due to its automated learning of task sampling weights during the training procedure. This dynamic weight adaptation facilitates straightforward implementation and seamless integration into existing workflows, eliminating the need for manual intervention or training disruptions.

Table 8: We report the total training time (hrs) and how much overhead time that running PiKE (hrs). Compared with the total training, the overhead for running PiKE is minimal only taking from 1.2% to 2.4% in training large models.

	Total Training Time (hrs)	Overhead time for running PiKE (hrs)	Overhead time for running PiKE (%)
GPT-2 small (110M, GLaM)	10	0.12	1.2
GPT-2 small (270M, Multilingual)	12	0.24	2
GPT-2 large (750M, GLaM)	41	1.0	2.4
GPT-2 large (1B, Multilingual)	71	1.6	2.3

I.5. Additional Zero-shot and Seven-shot Results

In the main text, our evaluation focused on 0-shot performance for multilingual pre-training and 7-shot performance for GLaM pre-training. This appendix provides a more complete picture by presenting both 0-shot and 7-shot evaluation results for the checkpoints from these respective experimental setups. The results are in the Table 9 and 10.

I.6. Ablation: Importance of Gradient Variance for PiKE

This section underscores the importance of PiKE's per-task gradient variance component for its overall effectiveness. To demonstrate this, we conducted an ablation study where PiKE's ζ_1 hyperparameter was held constant at 0.1, while ζ_2, which controls the influence of this gradient variance term, was varied. The specific case of $\zeta_2 = 0$ signifies the complete omission of the variance component from PiKE's formulation. The results are in Table 11. For instance, PiKE achieved a mean accuracy of

Table 9: We compare the accuracies (%, higher the better) of different models on HellaSwag and its corresponding translated version using 0- and 7-Shot settings. **Bolding** indicates the best model in the task; $\overline{\text{Metrics}}$ means the average across different tasks.

		HellaSwag (en)	HellaSwag (hi)	HellaSwag (de)		HellaSwag (en)	HellaSwag (hi)	HellaSwag (de)
	$\overline{\text{Accuracy}}$ ↑	0-shot ↑	0-shot ↑	0-shot ↑	$\overline{\text{Accuracy(\%)}}$ ↑	7-shot ↑	7-shot ↑	7-shot ↑
C4 (en), C4 (hi), and C4 (de) datasets, GPT-2 large style, 1B params, 36 Layers default, 120K training steps								
Mix	27.5	28.1	27.1	**27.6**	32.6	33.9	**31.4**	**32.5**
Round-Robin	26.5	27.6	26.7	26.3	32.0	34.0	29.7	32.4
Random	26.6	27.0	26.9	26.1	31.9	34.0	30.0	31.7
PiKE	**28.7**	**33.0**	**27.2**	26.2	**33.2**	**39.0**	29.6	30.9

Table 10: We compare the accuracies (%, higher the better) of different models on four different Q/A tasks using 0- and 7-Shot settings. **Bolding** indicates the best model in the task, $\overline{\text{Metrics}}$ means the average across different tasks.

		ArcE	CSQA	HellaSwag	PIQA		ArcE	CSQA	HellaSwag	PIQA
	$\overline{\text{Accuracy}}$ ↑	0-shot ↑	0-shot ↑	0-shot ↑	0-shot ↑	$\overline{\text{Accuracy(\%)}}$ ↑	7-shot ↑	7-shot ↑	7-shot ↑	7-shot ↑
Six domains of GLaM dataset, GPT-2 large style, 750M params, 36 layers default										
Mix	33.6	30.3	20.8	29.5	53.8	46.4	47.2	39.6	37.9	60.9
Round-Robin	32.7	30.4	20.3	26.1	53.9	44.3	43.5	36.7	36.8	60.3
Random	32.0	28.9	20.5	26.2	52.3	42.7	41.7	34.2	36.6	58.2
GLaM	31.7	28.8	19.9	26.3	51.9	45.3	46.9	39.8	**38.0**	56.4
DoReMi	35.6	33.0	23.8	30.0	55.7	46.5	48.6	40.1	37.5	59.6
PiKE (Uniform)	**37.9**	**37.4**	24.2	**33.9**	56.1	47.6	49.6	43.2	37.2	60.4
PiKE (GLaM)	35.5	33.5	20.4	31.2	**56.8**	**48.1**	**49.8**	**43.5**	**38.0**	**61.2**

Table 11: Mean Accuracies (%, higher values indicate better performance) of GLaM 740M models pre-trained with PiKE on four Question/Answering (Q/A) tasks, under different PiKE hyperparameter settings. PiKE's ζ_1 hyperparameter was fixed constant at 0.1, while ζ_2 was varied. A value of $\zeta_2 = 0$ means an ablation where the per-task gradient variance term in PiKE is omitted.

ζ_2	0	0.001	0.005	0.01	0.05	0.1
Accuracy	45.3	46.2	**47.0**	46.4	46.9	45.3

47.0% with $\zeta_2 = 0.005$, which dropped to 45.3% when the variance term was omitted ($\zeta_2 = 0$). This performance degradation highlights the critical role of per-task gradient variance in regulating PiKE's sampling weights, affirming its necessity for achieving optimal results.

I.7. Details on Estimating Gradient Variance and Magnitude, and Sensitivity Analysis

We estimate the per-task gradient variance and magnitude (measured using the L_2 norm) every $T_0 = 1000$ training steps. At each estimation step, the number of samples used for each task corresponds to its assigned portion (b_k) of the standard mixed training batch. Our JAX-based implementation enables efficient computation of these statistics with negligible overhead relative to total training time.

To evaluate the impact of estimation accuracy on PiKE's performance, we conduct a sensitivity analysis while keeping all other experimental settings and hyperparameters fixed. Specifically, we compare two strategies. The first, used in our main experiments, computes statistics for each task once using samples from a single training batch. The second estimates statistics multiple times using different subsets of samples (e.g., distinct micro-batches) and averages them to produce the final

Table 12: Mean accuracies (%, higher is better) of GLaM 740M models pretrained with PiKE on four question answering (Q/A) tasks, evaluated under different PiKE hyperparameter settings. The table compares the effect of estimating gradient statistics for each task either once (using a single batch) or multiple times using different subsets of samples.

Number of Estimation	1	2	3	4
Accuracy	46.8	47.4	47.1	46.8

values used by PiKE. Results are shown in Table 12. We find that PiKE is robust to the estimation noise: computing gradient statistics from a single batch is sufficient to achieve strong performance, making the method both effective and efficient.

I.8. Full Pre-training Results

Tables 13 and 14 present the complete results of pre-training language models across various scales (110M, 270M, 750M, and 1B) and scenarios (Multilingual and GLaM datasets). PiKE consistently outperforms all baselines across all scales and scenarios.

I.9. Adaptive Sampling Weights of PiKE During Pre-training

Figure 8 illustrates how the adaptive sampling weights of PiKE evolve during language model pre-training. Compared to the Mix sampling strategy, which assigns equal sampling weights to each task, PiKE adaptively adjusts the sampling weights among English, German, and Hindi by leveraging the positive interaction of task gradients. This adaptive data selection allows PiKE to achieve superior performance compared to fixed or heuristic-based baselines.

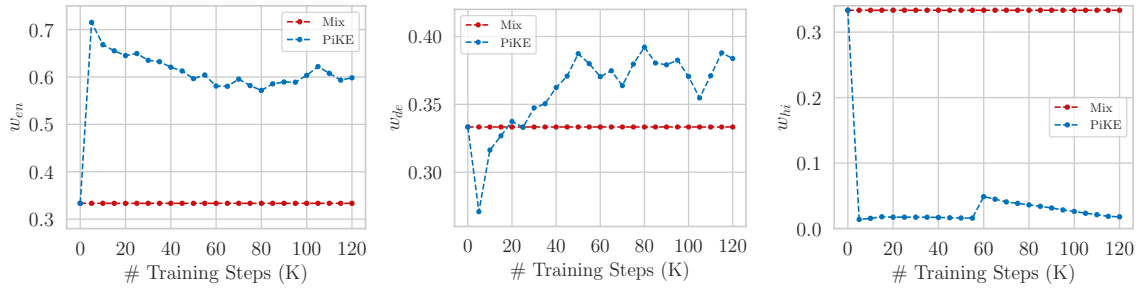

Figure 8: The sampling weights for each dataset during the pre-training of 1B GPT-2-style multilingual language models on mC4 (English), mC4 (Hindi), and mC4 (German). Here, w_{en} represents the sampling weight for the English dataset, w_{hi} for the Hindi dataset, and w_{de} for the German dataset.

Appendix J. Derivations and Proofs

J.1. Detailed Derivation of equation (9)

Recall that

$$\mathbf{g}_t = \frac{1}{b_1 + b_2}\left(b_1 e_1 e_1^\top + b_2 e_2 e_2^\top\right)\boldsymbol{\theta}_t + \mathbf{z},$$

Table 13: We report the perplexities (lower the better) on the validation split of multilingual C4 datasets. We also compare the accuracies (%, higher the better) of different models on HellaSwag and its corresponding translated version. HellaSwag and its translated versions have 4 choices. **Bolding** indicates the best model in the task, $\overline{\text{Metrics}}$ means the average across different tasks.

		C4 (en)	C4 (hi)	C4 (de)		HellaSwag (en)	HellaSwag (hi)	HellaSwag (de)
	$\overline{\text{Perplexity}}\downarrow$	Perplexity $\downarrow$	Perplexity $\downarrow$	Perplexity $\downarrow$	$\overline{\text{Accuracy}(\%)}\uparrow$	0-shot $\uparrow$	0-shot $\uparrow$	0-shot $\uparrow$
Single dataset, GPT-2 small style, 270M params, 12 layers default, 120K training steps								
C4 (en)	13.25	13.25	*	*	26.5	26.5	*	*
C4 (hi)	4.97	*	4.97	*	26.4	*	26.4	*
C4 (de)	11.27	*	*	11.27	26.1	*	*	26.1
C4 (en) and C4 (hi) datasets, GPT-2 small style, 270M params, 12 layers default, 120K training steps								
Mix	10.50	15.46	**5.55**	*	25.5	24.4	26.5	*
Round-Robin	10.57	15.57	5.57	*	25.6	25.2	26.0	*
Random	10.57	15.57	5.57	*	25.3	24.3	26.3	*
FAMO [35]	10.38	15.18	5.57	*	25.7	24.8	26.5	*
ADO [27]	10.45	15.39	5.52	*	25.1	24.3	25.8	*
PiKE	**10.15**	**14.31**	5.99	*	**26.5**	**26.0**	**27.0**	*
C4 (en), C4 (hi), and C4 (de) datasets, GPT-2 small style, 270M params, 12 layers default, 120K training steps								
Mix	12.00	16.30	**5.88**	13.83	25.3	24.4	26.0	**25.5**
Round-Robin	12.10	16.44	5.91	13.95	25.1	24.3	26.0	24.9
Random	12.16	16.49	5.95	14.03	25.1	24.7	**26.6**	23.9
FAMO [35]	**11.92**	16.19	6.00	**13.57**	24.8	24.5	25.2	24.8
ADO [27]	12.01	16.31	**5.88**	13.84	24.9	24.4	25.4	24.8
PiKE	12.01	**15.48**	5.92	14.64	**25.6**	**25.4**	26.4	24.8
Single dataset, GPT-2 large style, 1B params, 36 Layers default, 120K training steps								
C4 (en)	9.30	9.30	*	*	33.6	33.6	*	*
C4 (hi)	3.87	*	3.87	*	27.5	*	27.5	*
C4 (de)	7.72	*	*	7.72	28.1	*	*	28.1
C4 (en) and C4 (hi) datasets, GPT-2 large style, 1B params, 36 Layers default, 120K training steps								
Mix	7.41	10.60	**4.22**	*	27.3	28.2	26.5	*
Round-Robin	7.49	10.72	4.25	*	27.5	28.0	27.0	*
Random	7.52	10.76	4.28	*	28.0	28.9	27.0	*
FAMO [35]	7.33	10.44	**4.22**	*	26.8	27.1	26.5	*
ADO [27]	7.41	10.59	4.23	*	26.5	26.0	26.9	*
PiKE	**7.21**	**9.63**	4.80	*	**30.0**	**32.7**	**27.3**	*
C4 (en), C4 (hi), and C4 (de) datasets, GPT-2 large style, 1B params, 36 Layers default, 120K training steps								
Mix	8.29	11.13	**4.45**	9.29	27.5	28.1	27.1	27.6
Round-Robin	8.41	11.31	4.97	9.46	26.5	27.6	26.7	26.3
Random	8.48	11.38	4.54	9.55	26.6	27.0	26.9	26.1
FAMO [35]	8.25	11.04	4.48	9.23	27.2	27.3	26.9	27.3
ADO [27]	8.30	11.12	4.45	9.31	27.5	27.7	**27.5**	27.2
PiKE	9.56	**9.49**	5.32	13.87	28.7	**33.0**	27.2	26.2
Balanced-PiKE ($\tau = 1$)	8.29	11.12	4.46	9.31	27.9	28.3	27.4	28.0
Balanced-PiKE ($\tau = 3$)	**8.18**	10.14	4.93	9.49	**28.9**	31.3	27.3	28.1
Balanced-PiKE ($\tau = 5$)	8.42	10.02	6.30	**8.94**	**28.9**	31.2	26.9	**28.6**

Then

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta \frac{1}{b_1 + b_2}\left(b_1 e_1 e_1^\top + b_2 e_2 e_2^\top\right)\boldsymbol{\theta}_t - \eta \mathbf{z} \\ &= \boldsymbol{\theta}_t - \frac{\eta}{b}\begin{bmatrix} b_1 & 0 \\ 0 & b_2 \end{bmatrix}\boldsymbol{\theta}_t - \eta \mathbf{z}\end{aligned}$$

Now consider the loss functions for task 1, $\mathcal{L}_1(\theta_{t+1})$, and task 2, $\mathcal{L}_2(\theta_{t+1})$, separately, taking the expectation over the randomness of $\mathbf{z}$

$$\begin{aligned}\mathbb{E}[\mathcal{L}_1(\boldsymbol{\theta}_{t+1})]) &= \mathbb{E}\left[\frac{1}{2}(\mathbf{e}_1^\top \boldsymbol{\theta}_{t+1})^2\right] \\ &= \mathbb{E}\left[\frac{1}{2}\left(\mathbf{e}_1^\top \begin{bmatrix} 1 - \frac{\eta b_1}{b} & 0 \\ 0 & 1 - \frac{\eta b_2}{b} \end{bmatrix}\boldsymbol{\theta}_t - \mathbf{e}_1^\top \eta \mathbf{z}\right)^2\right]\end{aligned}$$

$$= \frac{1}{2}\left(\begin{bmatrix}1-\frac{\eta b_1}{b} & 0\end{bmatrix}\boldsymbol{\theta}^\top\right)^2 + \frac{1}{2}\eta^2\mathbf{e}_1^\top\mathbf{Q}\mathbf{e}_1$$
$$= \frac{1}{2}\left(\left(1-\frac{\eta b_1}{b}\right)\theta_{1,t}\right)^2 + \frac{1}{2}\eta^2\mathbf{e}_1^\top\mathbf{Q}\mathbf{e}_1$$

Similarly, for task 2, we have

$$\mathbb{E}[\mathcal{L}_2(\boldsymbol{\theta}_{t+1})]) = \frac{1}{2}\left(\left(1-\frac{\eta b_2}{b}\right)\theta_{2,t}\right)^2 + \frac{1}{2}\eta^2\mathbf{e}_2^\top\mathbf{Q}\mathbf{e}_2$$

where $\theta_{1,t}$ and $\theta_{2,t}$ denote the first and second component of the vector $\boldsymbol{\theta}_t$. Combining the losses for both tasks, the total expected loss becomes

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1})] &= \mathbb{E}[\mathcal{L}_1(\boldsymbol{\theta}_{t+1})]) + \mathbb{E}[\mathcal{L}_2(\boldsymbol{\theta}_{t+1})]) \\ &= \frac{1}{2}\left(\left(1-\frac{\eta b_1}{b}\right)\theta_{1,t}\right)^2 + \frac{1}{2}\left(\left(1-\frac{\eta b_2}{b}\right)\theta_{2,t}\right)^2 + \eta^2\frac{b_1\sigma_1^2+b_2\sigma_2^2}{b^2} \\ &= \frac{1}{2}(1-\frac{\eta b_1}{b})^2\theta_{1,t}^2 + \frac{1}{2}(1-\frac{\eta b_2}{b})^2\theta_{2,t}^2 + \eta^2\frac{b_1\sigma_1^2+b_2\sigma_2^2}{b^2},\end{aligned}$$

which completes the derivations.

J.2. PiKE: Main Theoretical Results

Lemma 7 *Assume $\frac{1}{2(K-1)} > \underline{c}$. If $\|\nabla\mathcal{L}(\boldsymbol{\theta})\|^2 \le \epsilon$, we have*

$$\sum_{k=1}^{K}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|^2 \le \frac{\epsilon}{1-2\,\underline{c}\,(K-1)}.$$

Conversely, if $\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|^2 \le \delta_k,\ \forall k$, then

$$\|\nabla\mathcal{L}(\boldsymbol{\theta})\|^2 \le (1-\bar{c})\sum_{k=1}^{K}\delta_k \ + \ \bar{c}\left(\sum_{k=1}^{K}\sqrt{\delta_k}\right)^2$$

Proof: We first prove the first direction. Notice that

$$\begin{aligned}\|\nabla\mathcal{L}(\boldsymbol{\theta})\|^2 &= \|\sum_{k=1}^{K}\nabla\mathcal{L}_k(\boldsymbol{\theta})\|^2 \\ &= \sum_{k=1}^{K}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|^2 + \sum_{k=1}^{K}\sum_{j\neq k}\langle\nabla\mathcal{L}_j(\boldsymbol{\theta}),\nabla\mathcal{L}_k(\boldsymbol{\theta})\rangle \le \epsilon\end{aligned}$$

where we use the definition of $\nabla\mathcal{L}(\boldsymbol{\theta})$ and expand the term. Then we have

$$\begin{aligned}\sum_{k=1}^{K}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|^2 + \sum_{k=1}^{K}\sum_{j\neq k}\langle\nabla\mathcal{L}_j(\boldsymbol{\theta}),\nabla\mathcal{L}_k(\boldsymbol{\theta})\rangle &\overset{(a)}{\ge} \sum_{k=1}^{K}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|^2 - \underline{c}\sum_{k=1}^{K}\sum_{j\neq k}\left(\|\nabla\mathcal{L}_j(\boldsymbol{\theta})\|^2 + \|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|^2\right) \\ &\overset{(b)}{\ge} \sum_{k=1}^{K}\|\nabla\mathcal{L}_k(\boldsymbol{\theta})\|^2\left(1-2\underline{c}(K-1)\right)\end{aligned}$$

where (a) uses the Definition 1, (b) uses symmetric identity. Thus we get

$$\sum_{k=1}^{K} \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2 \leq \frac{\epsilon}{1 - 2\,\underline{c}\,(K-1)}$$

This completes the proof of the first inequality. We now prove the second inequality. Notice that

$$\begin{aligned}
\|\nabla \mathcal{L}(\boldsymbol{\theta})\|^2 = \|\sum_{k=1}^{K} \nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2 &= \sum_{k=1}^{K} \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2 + \sum_{k=1}^{K}\sum_{j\neq k} \langle \nabla \mathcal{L}_j(\boldsymbol{\theta}), \nabla \mathcal{L}_k(\boldsymbol{\theta})\rangle \\
&\overset{(a)}{\leq} \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2 + \bar{c}\sum_{k=1}^{K}\sum_{j\neq k} \|\nabla \mathcal{L}_j(\boldsymbol{\theta})\|^2 \|\mathcal{L}_k(\boldsymbol{\theta})\|^2 \\
&= (1-\bar{c})\|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2 + \bar{c}\|\nabla \mathcal{L}_k\|^2 + \bar{c}\sum_{k=1}^{K}\sum_{j\neq k} \|\nabla \mathcal{L}_j(\boldsymbol{\theta})\|^2 \|\mathcal{L}_k(\boldsymbol{\theta})\|^2 \\
&\overset{(b)}{\leq} (1-\bar{c})\sum_{k=1}^{K} \delta_k + \bar{c}\left(\sum_{k=1}^{K} \sqrt{\delta_k}\right)^2
\end{aligned}$$

where (a) use the Definition 2 and (b) combines the second and third terms and use the condition that $\|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2 \leq \delta_k$. This completes the proof of the second inequality.

Lemma 8 *For the optimization problem*

$$\begin{aligned}
\min_{w_1,\ldots,w_K} \quad & \sum_{k=1}^{K} w_k \lambda_k + \frac{1}{2} w_k^2 \kappa_k \\
\textit{s.t.} \quad & \sum_{k=1}^{K} w_k = 1, \quad w_k \geq 0, \quad \forall k
\end{aligned} \tag{13}$$

the optimal solution is

$$w_k^* = \max\left\{0, -\frac{\mu + \lambda_k}{\kappa_k}\right\} \tag{14}$$

where μ is chosen such that $\sum_{k=1}^{K} w_k^* = 1$

Proof: Consider the Lagrangian function

$$\mathcal{L}(w_1, \ldots, w_k, \mu, \alpha_1, \ldots, \alpha_k) = \sum_{k=1}^{K} w_k \lambda_k + \frac{1}{2} w_k^2 \kappa_k + \mu\left(\sum_{k=1}^{K} w_k - 1\right) - \sum_{k=1}^{K} \alpha_k w_k$$

where μ is Lagrange multiplier for the equality constraint for the constraint $\sum_{k=1}^{K} w_k = 1$ and $\alpha_k \geq 0$ are Lagrange multipliers for the nonnegativity constraints w_k. Take the partial derivative of $\mathcal{L}$ with respect to w_k and set it to 0:

$$\frac{\partial \mathcal{L}}{\partial w_k} = \lambda_k + w_k \kappa_k + \mu - \alpha_k = 0$$

From the Karush-Kuhn-Tucker (KKT) conditions, we also have $w_k^\star \geq 0, \alpha_k \geq 0$, and $\alpha_k w_k^\star = 0$. If $w_k^\star > 0$, then $\alpha_k = 0$, which implies

$$0 = \lambda_k + w_k^\star \kappa_k + \mu \quad \Longrightarrow \quad w_k^\star = -\frac{\mu + \lambda_k}{\kappa_k}$$

If $-(\mu + \lambda_k)/\kappa_k$ is negative, then $w_k^\star = 0$ must hold. Combining these, we get

$$w_k^* = \max\left\{0, -\frac{\mu + \lambda_k}{\kappa_k}\right\}$$

Finally, the Lagrange multiplier μ is determined by enforcing the equality constraint:

$$\sum_{k=1}^{K} w_k^* = 1$$

with μ chosen so that the w_k^* sum to 1 . This completes the proof.

Theorem 9 *(Formal Statement of Theorem 4) Suppose Assumption 3 is satisfied. Assume that at the given point $\boldsymbol{\theta}_t$ the gradients are $\underline{c}$-conflicted and $\bar{c}$-aligned with $\underline{c} < \frac{1}{K-2+b/b_k}, \forall k$. Moreover, assume batching is performed based on the mix strategy equation (2), i.e.,*

$$\mathbf{g}_t = \frac{1}{b}\sum_{k=1}^{K}\sum_{i=1}^{b_k} \nabla \ell_k(\boldsymbol{\theta}_t; x_{k,i}) = \sum_{k=1}^{K} \frac{b_k}{b} \bar{\mathbf{g}}_{t,k},$$

Then, we have

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_t - \eta \mathbf{g}_t) \,|\, \mathcal{F}_t] \leq \mathcal{L}(\boldsymbol{\theta}_t) + \sum_{k=1}^{K} b_k \Big(-\frac{\eta}{b}\beta \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \frac{L\eta^2}{2b^2}\sigma_k^2 \Big) + \sum_{k=1}^{K} b_k^2 \frac{L\eta^2}{2b^2}\gamma \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2$$

where $0 \leq \beta \triangleq \min_k(1 + \underline{c}(-K + 2 - \frac{b}{b_k}))$ and $\gamma \triangleq 1 + \bar{c}(K-1)$, the expectation is taken over batch sampling randomness under the mix strategy $(b_1, \ldots, b_K)$, and $|\mathcal{F}_t$ denotes the natural filtration of our process.

Proof: We begin by revisiting the multi-task optimization problem under consideration. The objective is defined as:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \quad \mathcal{L}(\boldsymbol{\theta}) := \sum_{k=1}^{K} \mathbb{E}_{x \sim \mathcal{D}_k}\left[\ell_k(\boldsymbol{\theta}; x)\right], \tag{15}$$

where $\mathcal{L}(\boldsymbol{\theta})$ is the aggregate loss over all tasks. Assume we mix the gradients with taking b_k i.i.d. samples from task k for $k = 1, \ldots, K$. Then under, Assumption 3 and based on equation (2), the estimated gradient direction is given by

$$\mathbf{g}_t = \frac{1}{\sum_{k=1}^{K} b_k}\left(\sum_{k=1}^{K} \sum_{\substack{i=1 \\ x_i \sim \mathcal{D}_k}}^{b_k} \nabla \ell_k(\boldsymbol{\theta}_t; x_i)\right) \tag{16}$$

Let $\boldsymbol{\theta}_{t+1}$ be the updated point after gradient descent with $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \mathbf{g}_t$. By the descent lemma, the following inequality holds for the updated parameter $\boldsymbol{\theta}^+$:

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) \leq \mathcal{L}(\boldsymbol{\theta}_t) - \eta \mathbf{g}_t^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \frac{L\eta^2}{2} \|\mathbf{g}_t\|^2, \tag{17}$$

Conditioned on the natural filtration process $\mathcal{F}_t$, we take the expectation over the randomness of the samples draw and obtain:

$$\begin{aligned}
\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t\right] &\leq \mathcal{L}(\boldsymbol{\theta}_t) - \eta \mathbb{E}[\mathbf{g}_t \,|\, \mathcal{F}_t]^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \frac{L\eta^2}{2} \mathbb{E}\left(\|\mathbf{g}_t\|^2 \,|\, \mathcal{F}_t\right) \\
&\overset{(a)}{=} \mathcal{L}(\boldsymbol{\theta}_t) - \eta \left(\frac{1}{b} \sum_{k=1}^{K} b_k \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right)^\top \left(\sum_{k=1}^{K} \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right) \\
&\quad + \frac{L\eta^2}{2b^2} \left(\left\|\sum_{k=1}^{K} b_k \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right\|^2 + \sum_{k=1}^{K} (b_k \sigma_k^2)\right) \\
&\overset{(b)}{=} \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b} \left(\sum_{k=1}^{K} b_k \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K} \sum_{j \neq k} b_k \left\langle \nabla \mathcal{L}_j(\boldsymbol{\theta}_t), \nabla \mathcal{L}_k(\boldsymbol{\theta}_t) \right\rangle\right) \\
&\quad + \frac{L\eta^2}{2b^2} \left(\left\|\sum_{k=1}^{K} b_k \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right\|^2 + \sum_{k=1}^{K} (b_k \sigma_k^2)\right),
\end{aligned}$$

where (a) substitutes the definition of $\mathbf{g}_t$ and uses the Assumption 3, and (b) expands the terms. The notation $|\, \mathcal{F}_t$ denotes the natural filtration of the random process (loosely speaking, conditioned on the current time t and the past history). Continuing our simplifications, we have

$$\begin{aligned}
\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t\right] &\overset{(a)}{\leq} \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b} \left(\sum_{k=1}^{K} b_k \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 - \sum_{k=1}^{K} \sum_{j \neq k} b_k \underline{c} (\|\nabla \mathcal{L}_j(\boldsymbol{\theta}_t)\|^2 + \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2)\right) \\
&\quad + \frac{L\eta^2}{2b^2} \left(\sum_{k=1}^{K} b_k^2 \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K} \sum_{j \neq k} b_j b_k \left\langle \nabla \mathcal{L}_j(\boldsymbol{\theta}_t), \nabla \mathcal{L}_k(\boldsymbol{\theta}_t) \right\rangle + \sum_{k=1}^{K} b_k \sigma_k^2\right), \\
&\overset{(b)}{=} \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b} \left(\sum_{k=1}^{K} \left(b_k - \underline{c} b_k (K-1) - \underline{c} \sum_{j \neq k} b_j\right) \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\right) \\
&\quad + \frac{L\eta^2}{2b^2} \left(\sum_{k=1}^{K} b_k^2 \|\nabla \mathcal{L}_k(\boldsymbol{\theta})\|^2 + \sum_{k=1}^{K} \sum_{j \neq k} b_j b_k \left\langle \nabla \mathcal{L}_j(\boldsymbol{\theta}_t), \nabla \mathcal{L}_k(\boldsymbol{\theta}_t) \right\rangle + \sum_{k=1}^{K} b_k \sigma_k^2\right), \\
&\overset{(c)}{\leq} \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b} \left(\sum_{k=1}^{K} (b_k - \underline{c}(K-1) b_k - \underline{c}(b - b_k)) \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\right) \\
&\quad + \frac{L\eta^2}{2b^2} \left(\sum_{k=1}^{K} b_k^2 \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K} \sum_{j \neq k} \bar{c} b_j b_k \|\nabla \mathcal{L}_j(\boldsymbol{\theta}_t)\|^2 \|\nabla \mathcal{L}_j(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K} b_k \sigma_k^2\right)
\end{aligned}$$

$$
\begin{aligned}
&\overset{(d)}{=} \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b}\left(\sum_{k=1}^{K} b_k(1+\underline{c}(-K+2-b/b_k))\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\right) \\
&\quad + \frac{L\eta^2}{2b^2}\left(\bar{c}\left(\sum_{k=1}^{K} b_k\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|\right)^2 + (1-\bar{c})\sum_{k=1}^{K} b_k^2\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K} b_k\sigma_k^2\right) \\
&\overset{(e)}{\leq} \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b}\left(\sum_{k=1}^{K} b_k(1+\underline{c}(-K+2-b/b_k))\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\right) \\
&\quad + \frac{L\eta^2}{2b^2}\left(\bar{c}K\sum_{k=1}^{K} b_k^2\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + (1-\bar{c})\sum_{k=1}^{K} b_k^2\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K} b_k\sigma_k^2\right) \\
&= \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b}\left(\sum_{k=1}^{K} b_k(1+\underline{c}(-K+2-b/b_k))\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\right) \\
&\quad + \frac{L\eta^2}{2b^2}\left((1-\bar{c}+\bar{c}K)\sum_{k=1}^{K} b_k^2\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K} b_k\sigma_k^2\right) \qquad (18)
\end{aligned}
$$

where (a) applies Definition 1 to the second term and expands the third term, (b) expands the summation in the second term, (c) uses the identity $\sum_{k=1}^{K}\sum_{j\neq k} b_j = \sum_{k=1}^{K}(b-b_k)$ in the second term and applies Definition 2 to the third term, (d) combines terms in the third term, and (e) uses the inequality $\|\sum_{i=1}^{N} u_i\|^2 \leq N\sum_{i=1}^{N} u_i^2$, where $\mathbf{u}$ is a column vector. We define β and γ such that

$$
\begin{aligned}
\beta &= \min_k(1+\underline{c}(-K+2-\frac{b}{b_k})) \\
\gamma &= 1+\bar{c}(K-1)
\end{aligned}
$$

Then using the definition of β and γ, substituting back we have

$$
\begin{aligned}
\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_{t+1})\,|\,\mathcal{F}_t\right] &\leq \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta\beta}{b}\left(\sum_{k=1}^{K} b_k\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\right) + \frac{L\eta^2}{2b^2}\left(\gamma\sum_{k=1}^{K} b_k^2\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K} b_k\sigma_k^2\right) \\
&= \mathcal{L}(\boldsymbol{\theta}_t) + \sum_{k=1}^{K} b_k\left(-\frac{\eta\beta}{b}\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \frac{L\eta^2}{2b^2}\sigma_k^2\right) + \sum_{k=1}^{K} b_k^2\frac{L\eta^2}{2b^2}\gamma\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2,
\end{aligned}
$$

which completes the proof.

Theorem 10 *There exist loss functions $\{\ell_k(\cdot,\cdot)\}_{k=1}^{K}$ satisfying Assumption 3, whose gradients are $\underline{c}$-conflicted and $\bar{c}$-aligned with $\underline{c} = \bar{c} = 0$. For these losses, the upper bound in equation (6) is tight when gradients are computed using the mix strategy equation (2); that is, the inequality in equation (6) holds with equality.*

Proof The proof is by simply generalizing Example 1 and showing the steps in the proof of Theorem 9 are tight for this example. Consider the multi-task learning problem where the loss of task k is given by

$$
\ell_k(\boldsymbol{\theta}, x_k) = \frac{L}{2}(\mathbf{e}_k^\top\boldsymbol{\theta})^2 + \mathbf{x}_k^\top\boldsymbol{\theta},
$$

with $\mathbf{x}_k \sim \mathcal{N}(\mathbf{0}, \frac{\sigma_k^2}{d}\mathbf{I})$ is the data for task k. It is easy to verify that this loss is smooth with smoothness parameter L. Moreover, we can show that Assumption 3 is satisfied. Notice that

$$\mathcal{L}_k(\boldsymbol{\theta}) = \mathbb{E}\left[\ell_k(\boldsymbol{\theta}, x_k)\right] = \frac{L}{2}(\mathbf{e}_k^\top \boldsymbol{\theta})^2$$

and

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \mathcal{L}_k(\boldsymbol{\theta}) = \frac{L}{2}\|\boldsymbol{\theta}\|^2.$$

First notice that it is easy to check that $\underline{c} = \bar{c} = 0$. Let $\boldsymbol{g}_t$ be the direction obtained by equation (2) and assume $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}_t$. Using the form of $\mathcal{L}(\boldsymbol{\theta})$, one can easily show that (by expanding the ℓ_2-loss):

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) = \mathcal{L}(\boldsymbol{\theta}_t) - \eta \mathbf{g}_t^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \frac{L\eta^2}{2}\|\mathbf{g}_t\|^2. \tag{19}$$

Conditioned on the natural filtration process $\mathcal{F}_t$, we obtain:

$$\begin{aligned}
\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t\right] &= \mathcal{L}(\boldsymbol{\theta}_t) - \eta \mathbb{E}[\mathbf{g}_t \,|\, \mathcal{F}_t]^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \frac{L\eta^2}{2}\mathbb{E}\left(\|\mathbf{g}_t\|^2 \,|\, \mathcal{F}_t\right) \\
&\overset{(a)}{=} \mathcal{L}(\boldsymbol{\theta}_t) - \eta \left(\frac{1}{b}\sum_{k=1}^{K} b_k \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right)^\top \left(\sum_{k=1}^{K} \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right) \\
&\quad + \frac{L\eta^2}{2b^2}\left(\left\|\sum_{k=1}^{K} b_k \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right\|^2 + \sum_{k=1}^{K}(b_k \sigma_k^2)\right) \\
&\overset{(b)}{=} \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b}\left(\sum_{k=1}^{K} b_k \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\right) \\
&\quad + \frac{L\eta^2}{2b^2}\left(\sum_{k=1}^{K} \|b_k \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \sum_{k=1}^{K}(b_k \sigma_k^2)\right),
\end{aligned}$$

where (a) substitutes the definition of $\mathbf{g}_t$ and uses the Assumption 3, and (b) is because $\langle \nabla \mathcal{L}_j(\boldsymbol{\theta}_t), \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\rangle = 0,\ \forall j \neq k$. Since $\beta = \gamma = 1$ in this example, we can write

$$\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t\right] = \mathcal{L}(\boldsymbol{\theta}_t) + \sum_{k=1}^{K} b_k \left(-\frac{\eta\beta}{b}\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \frac{L\eta^2}{2b^2}\sigma_k^2\right) + \sum_{k=1}^{K} b_k^2 \frac{L\eta^2}{2b^2}\gamma \|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2,$$

which shows our upper-bound is tight in this example and completes the proof. ■

Theorem 11 *(Theorem 5 in the main body) Suppose the assumptions in Theorem 9 is satisfied and we run the Conceptual PiKE Algorithm (Algorithm 2) initialized at $\boldsymbol{\theta}_0$ with the SGD optimizer in Step 10 of the algorithm. Let $\Delta_L = \mathcal{L}(\boldsymbol{\theta}_0) - \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ and $\sigma_{\max} = \max_k \sigma_k$. Suppose $\delta > 0$ is a given constant and the stepsize $\eta \leq \frac{\beta\delta}{L\sigma_{\max}^2/b + L\eta\delta}$. Then, after $T = \frac{2\beta\Delta_L}{\eta\delta}$ iterations, Algorithm 2 finds a point $\bar{\boldsymbol{\theta}}$ such that*

$$\mathbb{E}\|\nabla \mathcal{L}_k(\bar{\boldsymbol{\theta}})\|^2 \leq \delta, \quad \forall k = 1, \ldots, K. \tag{20}$$

Moreover, if we choose $\eta = \frac{\beta\delta}{L\sigma_{\max}^2/b + L\eta\delta}$*, then the Conceptual PiKE algorithm requires at most*

$$\bar{T} = \frac{2L\Delta_L(\sigma_{\max}^2/b + \gamma\delta)}{\delta^2\beta^2}$$

iterations to find a point satisfying equation (20).

Proof: We prove this by contradiction. Assume that $\max_k \mathbb{E}\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 > \delta$ for $t = 0, \ldots, T-1$. First notice that Theorem 9 implies that for all t, we have

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t] \leq \mathcal{L}(\boldsymbol{\theta}_t) + \sum_{k=1}^{K} w_k^\star \left(-\eta\beta\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t))\|^2 + \frac{L\eta^2\sigma_{\max}^2}{2b}\right) + \sum_{k=1}^{K} \frac{w_k^\star}{2}\left(L\eta^2\gamma\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2\right) \tag{21}$$

where $\{w_k^\star\}_{k=1}^K$ is the minimizer of the RHS of the equation (21) on the constrained set $\{(w_1, \ldots, w_k) | \sum_{k=1}^K w_k = 1,\ w_k \geq 0\ \forall k \in K\}$. Since $w_k^\star$ is the minimizer of the RHS of equation (21), we have

$$w_k^\star\left(-\eta\beta\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 + \frac{L\eta^2}{2b}\sigma_{\max}^2\right) + \frac{w_k^\star}{2}L\eta^2\gamma\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2 \leq \left(-\eta\beta\|\nabla\mathcal{L}_{k_t^\star}(\boldsymbol{\theta}_t)\|^2 + \frac{L\eta^2}{2b}\sigma_{\max}^2\right) + \frac{L\eta^2}{2}\gamma\|\nabla\mathcal{L}_{k_t^\star}(\boldsymbol{\theta}_t)\|^2 \tag{22}$$

where $k_t^\star \in \arg\max_k \|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2$. Moreover, we have

$$\eta \leq \frac{\beta\delta}{L\frac{\sigma_{\max}^2}{b} + L\gamma\delta} \leq \frac{\beta\|\nabla\mathcal{L}_{k_t^\star}(\boldsymbol{\theta})\|^2}{L\frac{\sigma_{\max}^2}{b} + L\gamma\|\nabla\mathcal{L}_{k_t^\star}(\boldsymbol{\theta}_t)\|^2},$$

Therefore,

$$\left(-\eta\beta\|\nabla\mathcal{L}_{k_t^\star}(\boldsymbol{\theta}_t)\|^2 + \frac{L\eta^2}{2b}\sigma_{\max}^2\right) + \frac{L\eta^2}{2}\gamma\|\nabla\mathcal{L}_{k_t^\star}(\boldsymbol{\theta}_t)\|^2 \leq -\frac{\beta\eta}{2}\|\nabla\mathcal{L}_{k_t^\star}(\boldsymbol{\theta}_t)\|^2 \tag{23}$$

Combining equation (21), (22), and (23), we obtain

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t] \leq \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\beta\eta}{2}\|\nabla\mathcal{L}_{k_t^\star}(\boldsymbol{\theta}_t)\|^2$$

Or equivalently,

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t] \leq \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\beta\eta}{2}\max_k \|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2$$

Taking the expectation over all sources of randomness in the algorithm, summing the above inequality from $t = 0$ to $t = T-1$, and simplifying the resulting telescoping sum, we obtain:

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_T)] \leq \mathcal{L}(\boldsymbol{\theta}_0) - \frac{\beta\eta}{2}\sum_{t=1}^{T-1}\max_k \mathbb{E}[\|\nabla\mathcal{L}_k(\boldsymbol{\theta}_t)\|^2]$$

Recalling the contradiction assumption that $\delta < \mathbb{E}[\|\nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\|^2]$, we get

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_T)] \leq \mathcal{L}(\boldsymbol{\theta}_0) - \frac{\beta\eta}{2} T\delta$$

Using the definition $\Delta_L \triangleq \mathcal{L}(\boldsymbol{\theta}_0) - \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$, we get

$$T \leq \frac{2\Delta_L}{\beta\eta\delta}$$

Finally notice that by setting $\eta = \frac{\beta\delta}{L\frac{\sigma_{\max}^2}{b} + L\gamma\delta}$, we get

$$T \leq \bar{T} = \frac{2\Delta_L}{\beta\eta} = \frac{2L\Delta_L}{\beta\delta^2}\left(\frac{\sigma_{\max}^2}{b} + \gamma\delta\right),$$

which completes the proof.

J.3. Conceptual PiKE versus Static Uniform Mix Batching

We now perform a standard analysis of Stochastic Gradient Descent (SGD), adapted to our specific setup where uniform (static) mini-batching is used (i.e., when $b_k = b/K,\ \forall k = 1, \ldots, K$). As is common in the analysis of gradient-based algorithms in smooth, nonconvex settings, we begin by quantifying the expected decrease in the objective function at each iteration. This is known as the descent per iteration. Once this is established, we use a telescoping sum argument to derive the iteration complexity of the algorithm, which tells us how many steps are needed to achieve a desired level of accuracy, given a properly chosen learning rate.

Let's begin by analyzing the descent that occurs in a single iteration. Recall that the objective is defined as:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \quad \mathcal{L}(\boldsymbol{\theta}) := \sum_{k=1}^{K} \mathbb{E}_{x \sim \mathcal{D}_k}\left[\ell_k(\boldsymbol{\theta}; x)\right]. \tag{24}$$

Under Assumption 3, and using uniform static mini-batch sampling—where each source k contributes equally to the batch with size $b_k = b/K$—the estimated gradient direction at iteration t is given by:

$$\mathbf{g}_t = \frac{1}{b}\left(\sum_{k=1}^{K} \sum_{\substack{i=1 \\ x_i \sim \mathcal{D}_k}}^{b/K} \nabla \ell_k(\boldsymbol{\theta}_t; x_i)\right) \tag{25}$$

This expression aggregates gradients from all sources using equally sized sub-batches, providing a mini-batch estimate of the full gradient. Let $\boldsymbol{\theta}_{t+1}$ denote the updated parameter after performing a gradient descent step, given by $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \mathbf{g}_t$. By the descent lemma, the following inequality holds for the updated parameter: $\boldsymbol{\theta}_{t+1}$:

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) \leq \mathcal{L}(\boldsymbol{\theta}_t) - \eta \mathbf{g}_t^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \frac{L\eta^2}{2}\|\mathbf{g}_t\|^2, \tag{26}$$

Conditioned on the natural filtration $\mathcal{F}_t$, we take the expectation with respect to the randomness in the sampled data and obtain:

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t\right] &\leq \mathcal{L}(\boldsymbol{\theta}_t) - \eta \mathbb{E}[\mathbf{g}_t \,|\, \mathcal{F}_t]^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \frac{L\eta^2}{2} \mathbb{E}\left(\|\mathbf{g}_t\|^2 \,|\, \mathcal{F}_t\right) \\ &= \mathcal{L}(\boldsymbol{\theta}_t) - \eta \left(\frac{1}{K}\sum_{k=1}^{K} \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right)^\top \left(\sum_{k=1}^{K} \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right) \\ &\quad + \frac{L\eta^2}{2b^2}\left(\left\|\sum_{k=1}^{K} \frac{b}{K} \nabla \mathcal{L}_k(\boldsymbol{\theta}_t)\right\|^2 + \frac{b}{K}\sum_{k=1}^{K} \sigma_k^2\right)\end{aligned}$$

Observing the fact that $\mathcal{L}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \mathcal{L}_k(\boldsymbol{\theta})$, we can write

$$\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t\right] \leq \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{K}\|\nabla \mathcal{L}(\theta_t)\|^2 + \frac{L\eta^2}{2b^2}\left(\frac{b^2}{K^2}\|\nabla \mathcal{L}(\theta_t)\|^2 + \frac{b}{K}\sum_{k=1}^{K}\sigma_k^2\right) \tag{27}$$

Define $\sigma^2 := \sum_{k=1}^{K} \sigma_k^2$. By summing u Rearranging the terms, we obtain:

$$\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \mathcal{F}_t\right] \leq \mathcal{L}(\boldsymbol{\theta}_t) + \left(-\frac{\eta}{K} + \frac{L\eta^2}{2K^2}\right)\|\nabla \mathcal{L}(\theta_t)\|^2 + \frac{L\eta^2}{2bK}\sigma^2 \tag{28}$$

Taking the expectation over all sources of randomness in the algorithm, summing the above inequality from $t = 0$ to $t = T - 1$, and simplifying the resulting telescoping sum, we obtain:

$$\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}_T)\right] \leq \mathcal{L}(\boldsymbol{\theta}_0) + \left(-\frac{\eta}{K} + \frac{L\eta^2}{2K^2}\right)\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla \mathcal{L}(\theta_t)\|^2\right] + \frac{LT\eta^2}{2bK}\sigma^2. \tag{29}$$

Let $\Delta_L := \mathcal{L}(\boldsymbol{\theta}_0) - \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ denote the initial suboptimality gap. As is standard in iteration complexity analysis, we are interested in determining the number of iterations required to reach a point $\boldsymbol{\theta}_{t_0}$ where $\mathbb{E}\left[\|\nabla \mathcal{L}(\boldsymbol{\theta}_{t_0})\|^2\right] \leq \delta$. Suppose, for the moment, that this condition has not yet been satisfied. Then, we can rewrite the above inequality as

$$0 \leq \Delta_L + \left(-\frac{\eta}{K} + \frac{L\eta^2}{2K^2}\right)T\delta + \frac{LT\eta^2}{2bK}\sigma^2. \tag{30}$$

Equivalently, under a properly chosen step size (to be defined later), we obtain the following upper bound on the number of iterations

$$T \leq \frac{\Delta_L}{\left(\frac{\eta}{K} - \frac{L\eta^2}{2K^2}\right)\delta - \frac{L\eta^2}{2bK}\sigma^2}.$$

By optimizing the choice of η to minimize the right-hand side, we derive the following bound on the iteration complexity

$$T \leq \frac{2L\Delta_L\left(\delta + \frac{\sigma^2}{b}K\right)}{\delta^2}.$$

In summary, for the uniform mix batching strategy, the standard analysis yields the following upper bound on the iteration complexity:

$$\bar{T}_{\text{Uniform}} = \frac{2L\Delta_L\left(\delta + \frac{\sigma^2}{b}K\right)}{\delta^2}. \tag{31}$$

In contrast, when the task gradients are nearly orthogonal—i.e., when the alignment/conflict parameters satisfy $\bar{c}, \underline{c} \approx 0$ —Theorem 5 gives the following iteration complexity bound for the Conceptual PiKE algorithm:

$$\bar{T}_{\text{PiKE}} = \frac{2L\Delta_L\left(\delta + \sigma_{\max}^2/b\right)}{\delta^2}, \tag{32}$$

which clearly illustrates the advantage of PiKE for this regime in terms of iteration complexity upper-bound.

J.4. Balanced-PiKE: Theoretical Developments and Derivations

Here we prove Lemma 6 which is the idea behind the Balanced-PiKE algorithm. Before proving it, let us, for the sake of completeness, re-state the lemma

Lemma 12 (Restatement of Lemma 6) *Let $0 < \tau \in \mathbb{R}$. Then, the tilted empirical risk minimization problem*

$$\min_{\boldsymbol{\theta}} \frac{1}{\tau} \log\left(\sum_{k=1}^{K} e^{\tau \mathcal{L}_k(\boldsymbol{\theta})}\right)$$

is equivalent to

$$\min_{\boldsymbol{\theta}} \quad \max_{\substack{\mathbf{y}\in\mathbb{R}_+^K \\ \sum_{k=1}^K y_k = \tau}} \sum_{k=1}^{K} y_k \mathcal{L}_k(\boldsymbol{\theta}) - \sum_{k=1}^{K} \frac{y_k}{\tau} \log\left(\frac{y_k}{\tau}\right). \tag{33}$$

Moreover, for any fixed $\boldsymbol{\theta}$, the inner maximization problem is maximized at $y_k^\star = \frac{\tau e^{\tau \mathcal{L}_k(\boldsymbol{\theta})}}{\sum_{j=1}^K e^{\tau \mathcal{L}_j(\boldsymbol{\theta})}}$, $\forall k$.

The above lemma is the direct consequence of applying Lemma 14 and Lemma 13.

Lemma 13 *For the problem*

$$\max_{\substack{\mathbf{y}\in\mathbb{R}_+^K \\ \sum_{k=1}^K y_k = \tau}} \left(\sum_{k=1}^{K} y_k x_k - \sum_{k=1}^{K} \frac{y_k}{\tau} \log\left(\frac{y_k}{\tau}\right)\right),$$

the optimal $\mathbf{y}$ is given by

$$y_k^\star = \frac{\tau e^{\tau x_k - 1}}{\sum_{k=1}^K e^{\tau x_k - 1}}$$

Proof: We start by forming and maximizing the Lagrangian function

$$\max_{\mathbf{y}\in\mathbb{R}_+^K} \left(\sum_{k=1}^{K} y_k x_k - \sum_{k=1}^{K} \frac{y_k}{\tau} \log\left(\frac{y_k}{\tau}\right) + \mu \left(\sum_{k=1}^{K} y_k - \tau \right) \right)$$

where μ is a free variable. Taking the partial derivative of the objective with respect to y_k and setting it to zero gives

$$y_k^\star = \alpha e^{\tau x_k},$$

where the coefficient α is independent of the index k and should be chosen such that $\sum_k y_k^* = 1$, implying

$$y_k^\star = \frac{\tau e^{\tau x_k}}{\sum_{j=1}^{K} e^{\tau x_j}}.$$

Lemma 14 *Let* $\mathbf{x} \in \mathbb{R}^K$ *and* $\tau > 0$*. Then,*

$$\log\left(\sum_{k=1}^{K} e^{\tau x_k}\right) = \max_{\substack{\mathbf{y}\in\mathbb{R}_+^K \\ \sum_{k=1}^{K} y_k = \tau}} \left(\sum_{k=1}^{K} y_k x_k - \sum_{k=1}^{K} \frac{y_k}{\tau} \log\left(\frac{y_k}{\tau}\right) \right)$$

Proof The proof is by simply plugging in the optimal value $y^\star$ obtained by Theorem 13 in the objective function. ■

Table 14: We report perplexity (lower is better) on the validation split of the GLaM datasets, averaging perplexities across six domains when applicable or reporting a single perplexity when only training with a single domain. We also compare the accuracies (%, higher the better) of different models on four different Q/A tasks. HellaSwag and ArcE tasks have 4 choices, CSQA has 5 choices, and PIQA has 2 choices. PiKE (Uniform) means PiKE using initial sampling weights of $1/6$ for each task and PiKE (GLaM) means PiKE using GLaM tuned weights as initial task weights. **Bolding** indicates the best model in the task, Metrics means the average across different tasks, underlining indicates PiKE beating Mix, Round-Robin, Random methods

	GLaM		ArcE	CSQA	HellaSwag	PIQA
	Perplexity ↓	Accuracy(%) ↑	7-shot ↑	7-shot ↑	7-shot ↑	7-shot ↑
Single domain of GLaM dataset, GPT-2 small style, 110M params, 12 layers default						
Wikipedia	9.96	33.5	32.5	20.9	27.3	53.3
Filtered Webpage	16.05	37.2	38.4	26.8	27.6	55.8
News	9.33	33.8	31.1	22.7	27.0	54.5
Forums	22.87	35.5	32.1	23.4	28.7	57.6
Books	16.81	34.7	34.3	22.1	27.8	54.7
Conversations	18.27	36.1	32.6	25.6	28.6	57.6
Six domains of GLaM dataset, GPT-2 small style, 110M params, 12 layers default						
Mix	18.27	36.2	35.6	24.1	**28.5**	56.7
Round-Robin	18.45	35.9	35.8	24.2	27.5	56.0
Random	18.48	35.5	34.3	22.4	28.4	56.8
FAMO [35]	**18.19**	35.9	35.3	24.2	27.8	56.4
ADO [27]	18.27	36.2	35.7	24.8	27.7	56.4
GLaM [17]	18.91	35.8	35.3	24.1	28.5	55.1
DoReMi [67]	18.98	37.0	36.0	**28.3**	28.2	55.3
PiKE (Uniform)	18.44	37.4	36.8	27.5	**28.5**	**57.0**
PiKE (GLaM)	19.34	**37.8**	**39.0**	27.0	28.0	**57.0**
Single domain of GLaM dataset, GPT-2 large style, 750M params, 36 layers default						
Wikipedia	7.24	35.9	35.1	24.0	30.5	53.9
Filtered Webpage	11.12	40.9	36.7	33.2	34.2	56.5
News	6.62	37.4	33.6	24.7	34.1	57.3
Forums	16.29	43.6	38.0	35.8	39.7	60.7
Books	11.83	41.3	40.0	33.0	34.5	57.8
Conversations	13.50	42.2	36.9	33.2	39.2	59.6
Six domains of GLaM dataset, GPT-2 large style, 750M params, 36 layers default						
Mix	**12.77**	46.4	47.2	39.6	37.9	60.9
Round-Robin	12.98	44.3	43.5	36.7	36.8	60.3
Random	12.99	42.7	41.7	34.2	36.6	58.2
FAMO [35]	13.25	45.0	43.7	40.0	36.4	59.8
ADO [27]	**12.77**	45.9	45.5	38.7	38.1	61.1
GLaM [17]	13.20	45.3	46.9	39.8	38.0	56.4
DoReMi [67]	13.25	46.5	48.6	40.1	37.5	59.6
PiKE (Uniform)	13.22	47.6	49.6	43.2	37.2	60.4
PiKE (GLaM)	13.35	48.1	**49.8**	**43.5**	38.0	61.2
Balanced-PiKE ($\tau = 1$)	13.21	47.5	48.8	42.5	37.6	61.2
Balanced-PiKE ($\tau = 3$)	13.26	47.2	48.8	41.5	37.2	61.3
Balanced-PiKE ($\tau = 5$)	13.19	**48.2**	49.3	42.6	**38.5**	**62.4**