Model Editing by Standard Fine-Tuning

Govind Gangadhar and **Karl Stratos**
Department of Computer Science
Rutgers University
{govind.gangadhar, karl.stratos}@rutgers.edu

Abstract

Standard fine-tuning is considered not as effective as specialized methods for model editing due to its comparatively poor performance. However, it is simple, agnostic to the architectural details of the model being edited, and able to leverage advances in standard training techniques with no additional work (e.g., black-box PEFT for computational efficiency), making it an appealing choice for a model editor. In this work, we show that standard fine-tuning alone can yield competitive model editing performance with two minor modifications. First, we optimize the conditional likelihood rather than the full likelihood. Second, in addition to the typical practice of training on randomly paraphrased edit prompts to encourage generalization, we also train on random or similar unedited facts to encourage locality. Our experiments on the ZsRE and COUNTERFACT datasets demonstrate that these simple modifications allow standard fine-tuning to match or outperform highly specialized editors in terms of edit score.

1 Introduction

Model editing is a promising approach to combating incorrect or otherwise unwanted knowledge in LLMs. Given a set of edits that assert desired information, the approach aims to alter the model so that it not only succeeds in memorizing these edits (efficacy) but also applies the asserted information to new prompts (generalization), *without* changing inferences that should remain unchanged (locality). There is clearly a trade-off between these metrics. At one extreme, the model can achieve high efficacy by memorizing the edits, but it will fail in generalization and locality. At another extreme, the model can achieve high locality by staying unmodified, but it will fail in efficacy and generalization.

Naively fine-tuning the model on the requested edits is well known to perform poorly, especially in locality. This motivated researchers to develop highly specialized model editors (Mitchell et al., 2022; Meng et al., 2022, 2023; Hartvigsen et al., 2023; Li et al., 2024a, *inter alia*). A central theme of these methods is "minimally invasive editing" achieved by a careful selection of layers, model-specific adapter designs, and low-rank updates. While technically interesting, they require a suite of assumptions which may not be satisfied in other contexts (e.g., with different model architectures). In contrast, standard fine-tuning is simple, completely agnostic to the details of the model, and can take advantage of advances in standard training such as parameter-efficient fine-tuning (PEFT) techniques for immediate computational efficiency.[1]

Editor	Standard model?	Batched edits?	No extra training?	Effective edits?
Naive fine-tuning	✓	✓	✓	✗
MEND	✗	✓	✗	✓
ROME	✗	✗	✓	✓
MEMIT	✗	✓	✓	✓
Our fine-tuning	✓	✓	✗	✓

Table 1: Conceptual comparisons of model editors.

In this work, we show that standard fine-tuning can yield competitive model editing performance, focusing primarily on improving the edit score. We shift the focus from models and algorithms to training objectives and data augmentation. See Table 1 for conceptual comparisons between our approach and other editors. Our fine-tuning uses two small but impactful modifications. First, in line with the theme of minimal editing, we optimize the *conditional* likelihood (i.e., mask all tokens except the edited target). Second, we augment the training data with random or similar unedited facts to

[1]We emphasize that our use of PEFT is only for computational convenience (i.e., we use black-box PEFT and are not concerned with its details such as LoRA vs others) and should not be confused with works that develop specialized adapters for model editing such as Yu et al. (2024).

encourage locality. This is in addition to the typical practice in existing works of training on randomly paraphrased edits to encourage generalization. These simple modifications allow standard fine-tuning to match or outperform specialized editors in mass-editing, and also perform respectably in single-editing.

2 Task

Our main task is mass-editing (i.e., performing multiple edits at the same time). Let $\mathcal{V}$ denote the vocabulary. A **fact** is a sentence $x \in \mathcal{V}^T$ that expresses a subject-relation-object triple (s, r, o) in natural language. We follow the convention in the model editing literature and assume that the object $o \in \mathcal{V}^m$ is the last m tokens of x. The prefix $\pi = (x_1 \dots x_{m-1})$ expresses (s, r) and is denoted as the **prompt**. Let $\mathcal{E}$ denote a set of facts to enforce (i.e., requested edits). Our goal is to edit a language model so that it upholds the relations expressed in $\mathcal{E}$ without changing its behavior on other facts. In ZsRE, $\mathcal{E}$ consists of 10,000 factual statements (e.g., "The artwork Gideon's Way was by who? John Creasey"). In COUNTERFACT, $\mathcal{E}$ consists of 10,000 counterfactual statements (e.g., "TextEdit, a product of Nintendo").

An editor is evaluated by three competing metrics: efficacy, generalization, and locality. Let p_θ denote a language model edited on $\mathcal{E}$. In ZsRE, efficacy is the accuracy of $o = \arg\max_y p_\theta(y|\pi)$ for $(\pi, o) \in \mathcal{E}$; generalization is the accuracy of $o = \arg\max_y p_\theta(y|\pi_{\text{par}})$ where π_{par} is a paraphrase of π; locality is the accuracy of $o_{\text{unrel}} = \arg\max_y p_\theta(y|\pi_{\text{unrel}})$ where $(\pi_{\text{unrel}}, o_{\text{unrel}})$ is an unrelated fact. In COUNTERFACT, efficacy and generalization measure the accuracy of $p_\theta(o|\pi) > p_\theta(o_{\text{pre}}|\pi)$ and $p_\theta(o|\pi_{\text{par}}) > p_\theta(o_{\text{pre}}|\pi_{\text{par}})$ where o_{pre} is a pre-edit object for $(\pi, o) \in \mathcal{E}$ (e.g., "Apple" in "TextEdit, a product of Nintendo"). Locality measures the accuracy of $p_\theta(o_{\text{pre}}|\pi_{\text{nb}}) > p_\theta(o|\pi_{\text{nb}})$ where π_{nb} is a "neighborhood" unedited prompt whose target is o_{pre} (e.g., "Macintosh File System, a product of" $\rightarrow$ "Apple"). In both datasets, the final **edit score** is the harmonic mean of efficacy, generalization, and locality.

It is well known that naive fine-tuning, namely just optimizing

$$\min_\theta -\sum_{x \in \mathcal{E}} \log p_\theta(x) \tag{1}$$

results in a poor edit score. The main reason is that while it improves efficacy and possibly even generalization, it harms locality by changing the model's predictions on unrelated or neighborhood prompts. This led to the development of highly specialized editing methods, with the implicit assumption that fine-tuning is not effective for model editing.

Fact: The mother tongue of Danielle Darrieux is English

Paraphrase augmentation (generated by the model):
1) *The present invention relates.* The mother tongue of Danielle Darrieux is English

Random fact augmentation (from the training split):
1) Crate & Barrel was founded in Chicago
2) Haines Borough is within Nevada

Figure 1: An example from COUNTERFACT along with paraphrase and random fact augmentation.

3 Method

We propose a purely fine-tuning-based method that achieves a competitive edit score. Our objective is a slight variation of (1):

$$\min_\theta -\sum_{(\pi,o) \in \mathcal{E} \cup \mathcal{P} \cup \mathcal{R}} \log p_\theta(o|\pi) \tag{2}$$

There are two small but important differences between (1) and (2). First, we optimize the *conditional* likelihood of the edit target $o|\pi$ rather than the full likelihood. Our motivation is to make the training more focused (for efficacy and generalization) and minimize the damage caused by fine-tuning (for locality).

Second, we fine-tune not only on the requested edits $\mathcal{E}$ but also additional facts in $\mathcal{P}$ and $\mathcal{R}$ to promote generalization and locality. $\mathcal{P}$ is pseudo-paraphrases of the prompts in $\mathcal{E}$ obtained by prepending random words generated from the (unedited) model. Paraphrase augmentation is universally used in existing model editors (Mitchell et al., 2022; Meng et al., 2023; Li et al., 2024a). Additionally, we augment with facts $\mathcal{R}$ that should not be altered by editing on $\mathcal{E}$. There are many ways to obtain $\mathcal{R}$, but we find taking *random* facts from the training split to be simple and effective. We filter $x \in \mathcal{R}$ to ensure that x does not include the subject-relation-object triple used in evaluation. Figure 1 shows an example from COUNTERFACT (we only show 1 paraphrase and 2 random facts for illustration, in practice we use more).

We emphasize that the augmented sets $\mathcal{P}$ and $\mathcal{R}$ do not use the evaluation facts (otherwise it is trivial). Given the requested edits $\mathcal{E}$, we perform the data augmentation and fine-tune the model according to (2).

Contrastive learning. Given that our goal in COUNTERFACT is to achieve $p_\theta(o|\pi) > p_\theta(o_{\text{pre}}|\pi)$ where o and o_{pre} are the new and pre-edit objects for a given prompt π, it is natural to consider a contrastive objective. We experimented with DPO (Rafailov et al., 2023) where we frame $o|\pi$ as the "preferred" response over $o_{\text{pre}}|\pi$, optimizing the DPO loss jointly with (2). However, we found that prompt masking and data augmentation are already effective and do not benefit from contrastive learning.

4 Related Work

We discuss existing model editors and the different assumptions they require to highlight the simplicity of our approach; we refer to Zhang et al. (2024) for an in-depth survey.

MEND (Mitchell et al., 2022) is a meta-learning method (Sinitsin et al., 2020; De Cao et al., 2021) that predicts the change in the gradient. Similar to our approach, it also involves an explicit training stage (on the training split of ZsRE). It uses a special rank-one decomposition of the gradient for parameter efficiency. In our case, we achieve parameter efficiency without special considerations simply by leveraging black-box PEFT.

ROME (Meng et al., 2022) is a locate-then-edit method (Geva et al., 2021; Dai et al., 2022) that first uses causal tracing to identify the feedforward layer to change, and then applies a rank-one update to the layer's weight matrix. While it does not involve an explicit training stage, it does require an explicit knowledge of the subject s in the edit and its vector representation (which is heuristically induced). It also relies on Wikipedia to estimate the covariance matrix of the subject embeddings (needed for the rank-one update). MEMIT (Meng et al., 2023) extends ROME to multiple edits by carefully spreading updates to multiple layers. Like ROME, it requires layer specification, subject embeddings, and covariance statistics obtained from Wikipedia.

IKE (Zheng et al., 2023) proposes in-context learning for model editing. While similar to our approach in avoiding the need to develop a specialized editor, it requires a strong model capable of effective in-context learning and is critically limited to single-editing. Consequently, IKE focuses on single-editing experiments with large-scale LLMs, making their results not comparable to ours.

We define our scope as achieving a competitive edit score through standard fine-tuning. In particular, we do not focus on preserving general capabilities of the model being edited. Recent work shows that performance on a variety of downstream tasks drops to zero under any editor (Gu et al., 2024). We leave preserving the general capabilities of an LLM with a fine-tuning editor as the next step of our work.

5 Experiments

5.1 Mass-Editing

Our main results are on mass-editing with on ZsRE and COUNTERFACT, following the same setting in Meng et al. (2023). Each dataset provides 10,000 requested edits (i.e., $\mathcal{E}$). As described in Section 3, we augment the fine-tuning data with pseudo-paraphrases of the edit prompts $\mathcal{P}$ for generalization supervision and random facts $\mathcal{R}$ for locality supervision. More specifically, for each edit in $\mathcal{E}$ we generate 15 paraphrases and take 20 facts from the training split while ensuring that they do not contain any evaluation facts. Thus the total number of facts we fine-tune on is 360,000. We fine-tune GPT-J (6B) (Wang and Komatsuzaki, 2021) with LoRA (Hu et al., 2022) for computational efficiency. We optimize (2). The training takes around 2–2.5 hours on 8 GPUs.[2]

We give the results in Table 2 where the row "FT + Mask + Para + Rand" corresponds to our final method. We can make the following observations. First, the proposed prompt masking in (2) (FT + Mask) improves the performance of vanilla fine-tuning (FT). In fact, on ZsRE this already outperforms MEMIT without any data augmentation. Second, augmenting the data with paraphrased prompts substantially improves generalization (FT + Mask + Para). Third, augmenting the data with random facts further improves locality (FT + Mask + Para + Rand), allowing standard fine-tuning to match the performance of MEMIT on COUNTERFACT. Adding the DPO loss on top of the final method does not yield improvements.

We perform additional experiments on the WikiRecent dataset (Cohen et al., 2023) with similar conclusions; details are given in Appendix A.

5.2 Single-Editing

To compare with existing methods developed for single-editing (i.e., updating the model for a single fact), we consider optimizing (2) one edit at a time.

[2]The code is available at: https://github.com/au-revoir/model-editing-ft.

Editor	ZsRE Score	ZsRE Efficacy	ZsRE Generalization	ZsRE Locality	COUNTERFACT Score	COUNTERFACT Efficacy	COUNTERFACT Generalization	COUNTERFACT Locality
— (original GPT-J)	26.4	26.4 (0.6)	25.8 (0.5)	27.0 (0.5)	22.4	15.2 (0.7)	17.7 (0.6)	83.5 (0.5)
FT-W (21st layer w/ weight decay)	42.1	69.6 (0.6)	64.8 (0.6)	24.1 (0.6)	67.6	99.4 (0.1)	77.0 (0.7)	46.9 (0.6)
MEND (Mitchell et al., 2022)	20.0	19.4 (0.5)	18.6 (0.5)	22.4 (0.5)	23.1	15.7 (0.7)	18.5 (0.7)	**83.0** (0.5)
ROME (Meng et al., 2022)	2.6	21.0 (0.7)	19.6 (0.7)	0.9 (0.1)	50.3	50.2 (1.0)	50.4 (0.8)	50.2 (0.6)
MEMIT (Meng et al., 2023)	50.8	96.7 (0.3)	89.7 (0.5)	26.6 (0.5)	85.8	98.9 (0.2)	88.6 (0.5)	73.7 (0.5)
PMET (Li et al., 2024b)	51.0	96.9 (0.3)	90.6 (0.2)	26.7 (0.2)	86.2	99.5 (0.1)	92.8 (0.4)	71.4 (0.5)
FT	44.8	**99.9** (0.0)	**98.9** (0.2)	21.4 (0.5)	52.8	79.6 (0.8)	58.5 (0.8)	36.8 (0.7)
FT (21st layer)	42.9	**99.9** (0.0)	87.4 (0.5)	20.5 (0.5)	60.5	99.9 (0.04)	63.3 (0.8)	42.0 (0.6)
FT + Mask	58.3	97.6 (0.3)	91.7 (0.5)	32.9 (0.6)	54.3	97.1 (0.3)	62.1 (0.8)	34.7 (0.6)
FT + Mask + Para	56.1	**99.9** (0.0)	98.7 (0.2)	29.9 (0.5)	63.7	**100.0** (0.0)	92.5 (0.4)	38.0 (0.6)
FT + Mask + Para + Rand	**62.0**	**99.9** (0.0)	97.0 (0.3)	**35.6** (0.6)	**86.5**	98.8 (0.2)	**93.6** (0.4)	72.0 (0.6)
FT + Mask + Para + Rand + DPO	—	—	—	—	85.5	98.8 (0.2)	93.4 (0.4)	70.1 (0.6)

Table 2: Mass-editing results on ZsRE and COUNTERFACT (10,000 edits each) with GPT-J. The results of FT-W, MEND, ROME, and MEMIT are from Meng et al. (2023). FT denotes naive fine-tuning on the requested edits; FT (21st layer) denotes FT only on the 21st layer. "+ Mask" means we mask the prompt. "+ Para" means paraphrase augmentation (which is involved in all the baselines). "+ Rand" means we augment the data with *random* facts from the training split (not overlapping with any evaluation facts). "+ DPO" means we additionally optimize the DPO loss term using the changed target as the preferred response over the pre-edit target (only provided in COUNTERFACT). Except for FT (21st layer), all our results use LoRA for computational efficiency.

Editor	ZsRE Score	ZsRE Efficacy	ZsRE Generalization	ZsRE Locality	COUNTERFACT Score	COUNTERFACT Efficacy	COUNTERFACT Generalization	COUNTERFACT Locality
— (original GPT-2 XL)	22.5	22.2 (0.5)	21.3 (0.5)	24.2 (0.5)	30.5	22.2 (0.9)	24.7 (0.8)	78.1 (0.6)
FT	45.9	99.6 (0.1)	82.1 (0.1)	23.2 (0.5)	65.1	**100.0** (0.0)	87.9 (0.6)	40.4 (0.7)
FT-L	40.1	92.3 (0.4)	47.2 (0.7)	23.4 (0.5)	66.9	99.1 (0.2)	48.7 (1.0)	70.3 (0.7)
KN	-	-	-	-	35.6	28.7 (1.0)	28.0 (0.9)	72.9 (0.7)
KE	41.8	65.5 (0.6)	61.4 (0.6)	24.9 (0.5)	52.2	84.3 (0.8)	75.4 (0.8)	30.9 (0.7)
MEND	42.9	75.9 (0.5)	65.3 (0.6)	24.1 (0.5)	57.9	99.1 (0.2)	65.4 (0.9)	37.9 (0.7)
ROME	47.9	99.8 (0.0)	88.1 (0.5)	24.2 (0.5)	**89.2**	100.0 (0.1)	**96.4** (0.3)	**75.4** (0.7)
FT + Mask + Para + Sim	**51.4**	**100.0** (0.0)	**99.9** (0.0)	**26.1** (0.8)	83.1	98.6 (0.3)	87.3 (0.7)	69.0 (0.7)

Table 3: Single-editing results on ZsRE (10,000 edits) and COUNTERFACT (7,500 edits) with GPT-2 XL. "+ Sim" means we include similar facts (measured by Sentence-BERT) instead of random facts for locality supervision.

We follow the setting in ROME and use GPT-2 XL (1.5B). The training takes around 1.4 seconds per edit on average. We give the results in Table 3. We find that it is helpful to include similar facts instead of random facts for locality supervision, presumably because fine-tuning is more sensitive with small data size thus requiring more care in the selection of demonstrations. We use Sentence-BERT embeddings (Reimers and Gurevych, 2019) and take 15 nearest facts as $\mathcal{R}$. On ZsRE, our method outperforms all existing single-edit methods. On COUNTERFACT, our method lags behind ROME which is specifically optimized for single-editing, but still achieves the second-best edit score.

5.3 Generative Metrics

We report results on generative metrics in Table 4 (see Appendix B for single-editing results). Following prior work (Meng et al., 2022, 2023), we report fluency (entropy of n-gram distributions) and consistency (similarity score with a reference text). Our fine-tuning methods take a hit on these metrics while improving the edit score, showing that more work is needed to go beyond classification. However, recent work shows that *none* of the compared editors preserves downstream performance (Gu et al., 2024), thus achieving this goal meaningfully is still an open problem.

We additionally show that we can easily incorporate considerations for generative performance into fine-tuning. Specifically, we optimize $(1-\gamma)L_1(\theta)+\gamma L_2(\theta)$ where $L_1(\theta)$ is the loss in (2) and $L_2(\theta) = -\sum_{w\in\mathcal{W}} \log p_\theta(w)$ is a language modeling loss on random Wikipedia text. We choose Wikipedia articles so that they do not overlap with those used for consistency evaluation. We set $\gamma = 0.1$. With this change ("+ Wikipedia

Editor	Score	Fluency	Consistency
— (original GPT-J)	22.4	622.4 (0.3)	29.4 (0.2)
FT-W	67.6	293.9 (2.4)	15.9 (0.3)
MEND	20.0	618.4 (0.3)	31.1 (0.2)
ROME	50.3	589.6 (0.5)	3.3 (0.0)
MEMIT	85.8	619.9 (0.3)	40.1 (0.2)
PMET	86.2	620.0 (0.3)	40.6 (0.2)
FT	52.8	626.1 (0.4)	31.0 (0.2)
FT + Mask	54.3	563.6 (0.5)	6.1 (0.1)
FT + Mask + Para	63.7	550.7 (0.6)	4.7 (0.1)
FT + Mask + Para + Rand	86.5	352.0 (1.5)	5.2 (0.2)
+ Wikipedia Loss	84.8	609.2 (0.6)	29.2 (0.2)

Table 4: Fluency and consistency with mass-editing on COUNTERFACT. "+ Wikipedia Loss" means we additionally optimize a language modeling loss on Wikipedia text (Section 5.3).

Editor	Score	Efficacy	Generalization	Locality
— (GPT-2 XL)	29.9	21.8 (0.8)	24.1 (0.7)	78.3 (0.5)
ROME	50.4	50.3 (0.9)	49.4 (0.8)	51.6 (0.6)
MEMIT	71.5	79.9 (0.7)	66.2 (0.8)	69.8 (0.5)
FT + M + P + R	85.4	98.8 (0.2)	86.5 (0.5)	74.3 (0.5)
w/o LoRA	86.9	98.8 (0.2)	91.2 (0.4)	74.4 (0.5)

Table 5: Mass-editing results on COUNTERFACT with and without LoRA.

Loss"), we obtain a significant improvement in both fluency and consistency at the cost of a modest drop in the edit score.

5.4 Analysis

What is the effect of PEFT? We repeat mass-editing experiments with a smaller model (GPT-2 XL, 1.5B) to compare the performance of LoRA vs full fine-tuning in Table 5. We see that we achieve better generalization without LoRA, thus our improvement comes from conditional likelihood optimization and data augmentation, not the use of adapters (i.e., unlike specialized model editors that are based on adapters).

Does layer selection help? Layer selection is an important component in many existing model editing works. To see if it also benefits us, we fine-tune (without LoRA) only the layers 3–5 in GPT-J which are a subset of the layers chosen in MEMIT. Table 6 shows that we obtain a significant improvement compared to updating all layers (Table 2). Thus layer selection helps on top of our already strong fine-tuning approach.

Subject knowledge vs data augmentation. A common but rather strong assumption in many state-of-the-art editors is that the subject phrase in a fact is known. This knowledge critically allows the model to completely avoid changing at test time if the subject is not recognized, giving a significant advantage in locality. For instance, with perfect subject knowledge, locality is optimal. We apply our neighborhood fact augmentation to SWEA$\bigoplus$OS (Li et al., 2024a) which holds a state-of-the-art result on COUNTERFACT using subject knowledge (see Appendix C for details). Table 7 shows that the augmentation does not help further. Hence subject knowledge obviates the need for data augmentation aimed at improving locality.

Editor	Score	Efficacy	Generalization	Locality
FT + M	63.0	98.1 (0.2)	61.2 (0.8)	47.4 (0.6)
FT + M + P	77.8	100 (0.0)	98.6 (0.2)	54.2 (0.6)
FT + M + P + R	91.1	98.8 (0.2)	96.8 (0.3)	80.2 (0.5)

Table 6: Mass-editing results on COUNTERFACT with GPT-J but only fine-tuning layers 3–5.

Editor	Score	Efficacy	Generalization	Locality
— (GPT-J)	22.4	15.2 (0.7)	17.7 (0.6)	83.5 (0.5)
SWEA$\bigoplus$OS	91.2	99.5 (0.1)	98.1 (0.2)	79.0 (0.5)
SWEA$\bigoplus$OS + $\mathcal{R}_{\mathcal{E}}$	91.2	99.5 (0.1)	97.7 (0.2)	79.4 (0.5)

Table 7: Mass-editing results (reproduced) on COUNTERFACT by adding neighborhood augmentation to SWEA$\bigoplus$OS.

6 Conclusions

We have demonstrated that standard fine-tuning is sufficient to obtain strong edit performance with a slight modification: optimizing the conditional likelihood and augmenting the data with additional facts to promote locality. Our results challenge the assumption that standard fine-tuning is ineffective as a model editor, suggesting that model editing could be achieved as part of standard training rather than through a specialized model editor.

Acknowledgements

We thank the anonymous ARR reviewers for helpful feedback.

Limitations

Our scope is limited to the classification aspect of model editing rather than the generative aspect to demonstrate the viability of a fine-tuning editor. Thus our approach is currently unable to preserve the LLM's general capabilities, as is the case with existing mass editors.

References

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700*.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xiaopeng Li, Shasha Li, Shezheng Song, Huijun Liu, Bin Ji, Xi Wang, Jun Ma, Jie Yu, Xiaodong Liu, Jing Wang, and Weimin Zhang. 2024a. Swea: Updating factual knowledge in large language models via subject word embedding altering.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024b. Pmet: Precise model editing in a transformer.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

A Mass-Editing on WikiRecent

To further validate our mass-editing results, we perform additional experiments on a simplified version of the WikiRecent dataset (Cohen et al., 2023). The original dataset consists of 570 and 1,266 facts from Wikidata for training and testing, where each fact has been recently modified (after July 2022). An example prompt is $\pi =$"The name of the position held by Nicolaus Bergius is" with the target object $o =$"bishop". Since the dataset does not provide paraphrase prompts for evaluating generalization, we create paraphrases of π by few-shot prompting GPT (`gpt-3.5-turbo-0125`). The prompt for GPT is given in Figure 2. Since the task is relatively straightforward, we find the generated paraphrases preserve the original meaning (e.g., "The position held by Nicolaus Bergius is"). For neighborhood prompts, we select 10 random

```
Paraphrase the given incomplete statement without changing
the meaning. The same completion in the original input
must also work for your paraphrase. Provide as many
distinct paraphrases as you can come up with.

INPUT: The war during which Mario Stoppani was in the
armed forces was
1) Paraphrased: The war Mario Stoppani was in the army in
was
2) Paraphrased: Mario Stoppani was in the war called
3) Paraphrased: Mario Stoppani served in the army during

INPUT: The birth date of Luca Pianca is
1) Paraphrased: The date of birth of Luca Pianca is
2) Paraphrased: Luca Pianca was born in
3) Paraphrased: The day of Luca Pianca's birth is
4) Paraphrased: Luca Pianca's birth date is
5) Paraphrased: The birthday of Luca Pianca is

INPUT: The name of the architect of Ravenna Cathedral is
1) Paraphrased: Ravenna Cathedral was built by
2) Paraphrased: The person who built Ravenna Cathedral
was
3) Paraphrased: The architect of Ravenna Cathedral is
4) Paraphrased: The architect behind the construction of
Ravenna Cathedral is
5) Paraphrased: Ravenna Cathedral was built by the architect

INPUT: {prompt}
1) Paraphrased:
```

Figure 2: The few-shot prompt we use for paraphrase generation. We selected in-context examples from COUNTERFACT.

Editor	Score	Efficacy	Generalization	Locality
— (GPT-J)	37.4	34.4 (1.7)	34.5 (1.5)	45.3 (0.9)
ROME	35.0	39.8 (2.2)	25.5 (1.4)	**46.9** (0.8)
MEMIT	67.3	99.2 (0.3)	80.2 (1.2)	45.3 (0.8)
FT	55.8	68.1 (1.8)	60.4 (1.7)	44.5 (0.8)
FT + M + P + R	**68.5**	**99.6** (0.2)	**84.6** (1.1)	45.8 (0.9)

Table 8: Mass-editing results on WikiRecent (1,266 edits).

neighborhood prompts from COUNTERFACT from the corresponding train/test splits. The training data for our fine-tuning method consists of 1,266 edit requests, 18,990 augmented paraphrases (i.e., randomly generated), and 12,660 random neighborhood prompts from COUNTERFACT (not overlapping with those in the test set).

Table 8 shows the results. We again find that while vanilla fine-tuning is not effective, our conditional fine-tuning with random data augmentation is competitive with MEMIT.

B Generative Metrics for Single-Editing

For completeness, we give generative performance for single-editing in Table 9 as an accompaniment to Table 4. Our method is again competitive with the best performing single-editing baseline (ROME) in the edit score, but has lower generative scores.

Editor	Score	Fluency	Consistency
— (original GPT-2 XL)	30.5	626.6 (0.3)	31.9 (0.2)
FT	65.1	607.1 (1.1)	40.5 (0.3)
FT-L	66.9	621.4 (1.0)	37.4 (0.3)
KN	35.6	570.4 (2.3)	30.3 (0.3)
KE	52.2	586.6 (2.1)	31.2 (0.3)
MEND	57.9	624.2 (0.4)	34.8 (0.3)
ROME	89.2	621.9 (0.5)	41.9 (0.3)
FT + Mask + Para + Sim	83.1	557.0 (1.8)	15.2 (0.3)

Table 9: Fluency and consistency with single-editing on COUNTERFACT.

C Data Augmentation for SWEA$\bigoplus$OS

We train SWEA$\bigoplus$OS on a set of random facts $\mathcal{R}_{\mathcal{E}}$ in addition to random paraphrases (which is already present). We use 10,661 random facts containing unique subjects from $\mathcal{E}$ within the training split. This is done because SWEA$\bigoplus$OS is only concerned with a final cache containing subject tokens and their embeddings. We first obtain the subject embeddings of the requested edits in $\mathcal{E}$ as described in Li et al. (2024a). We then repeat the process for the augmented random facts in $\mathcal{R}_{\mathcal{E}}$. Between the two sets of subjects, there are 392 subjects that overlap. In the case of an overlap, we only retain the embeddings from the $\mathcal{E}$ and remove those from $\mathcal{R}_{\mathcal{E}}$.