Empirical Study on Updating Key-Value Memories in Transformer Feed-forward Layers

[1][2]Zihan Qiu[*][†], **[3]Zeyu Huang**[*], **[4]Youcheng Huang**[*], **[1]Jie Fu**[‡]
[1]CSE, HKUST   [2]IIIS, Tsinghua University,
[3]ILCC, University of Edinburgh   [4]College of Computer Science, Sichuan University
qzh11628@gmail.com, zeyu.huang@ed.ac.uk
youchenghuang@stu.scu.edu.cn, jiefu@ust.hk

Abstract

The feed-forward networks (FFNs) in transformers are recognized as a group of key-value neural memories to restore abstract high-level knowledge. In this work, we conduct an empirical ablation study on updating keys (the 1st layer in the FFNs layer) or values (the 2nd layer in the FFNs layer). We compare those two methods in various knowledge editing and fine-tuning tasks of large language models to draw insights to understand FFNs further. Code is available at this repo.

1 Introduction

How do pre-trained Transformer models process and store information? Geva et al. (2021; 2022) suggest feed-forward networks (FFNs) operate as key-value neural memories (Sukhbaatar et al., 2015). Specifically, given hidden states $h \in \mathbb{R}^{d_e}$, where d_e is embedding size, $\text{FFNs}(h) = f(h \cdot K^T) \cdot V$, where $K, V \in \mathbb{R}^{d_m \times d_e}$, d_m is the width of FFNs and f is the non-linear activation. Each row of K can be viewed as a key correlated with input textual patterns, and each row of V can be viewed as a value that induces a distribution shift over the residual stream. For example, when a model predicts the next token based on the prefix 'Eiffel Tower is located in', k_2 is activated so that v_2 can promote the probability of 'Paris' in the output. On the contrary, values for irrelevant concepts (e.g., v_1 for 'Cat', v_d for Soccer) are deactivated and can not dominate the output distribution.
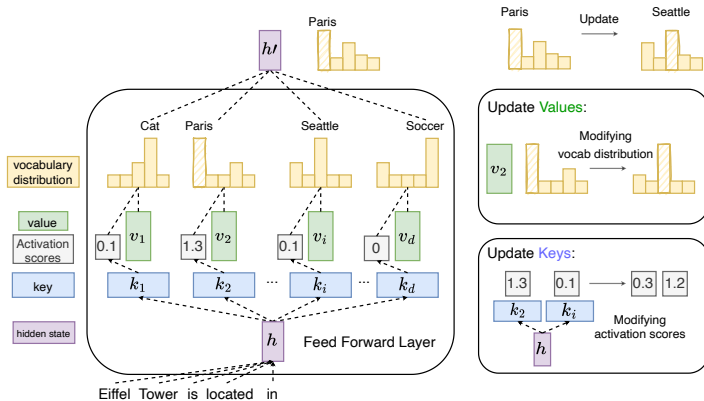

Figure 1: **Left**: FFNs operation is conceptualized as a key-value memory. The input hidden states interact with *keys* (rows of K) through an inner product to obtain activation values. These activations then serve as weights for summing *values* (rows of V). **Right**: To update these key-value memories, one can either directly modify the values that store relevant information or alter the keys to adjust the weights of existing values.

Based on this point of view, how can we update the information processing and storing? We employ Knowledge Editing (KE) (Wang et al., 2023) as an illustrative example: altering 'Paris' to 'Seattle' in response to 'Eiffel Tower is located in' by modifying the model weights. In this work, we compare

---

[*]Equal contribution
[†]Work done while interning at HKUST.
[‡]Corresponding author

Table 1: Knowledge Editing on GPT-J (6B) Wang (2021)

Editing Target	Efficacy↑	Paraphrase↑	Specificity↑	Score↑	Time (s)↓
1 Counterfact Editing on GPT-J (6B)					
On Value	100.00	98.18	6.04	16.15	0.79
On Key	100.00	**98.44**	**28.89**	**54.77**	0.83
1 zsRE Editing on GPT-J (6B)					
On Value	99.11	56.32	21.81	40.71	83.12
On Key	98.57	**69.19**	**24.64**	**46.02**	**12.63**

two updating choices as shown in Figure 1: 1. **Values**: tuning v_2 to shift its concept from 'Paris' to 'Seattle'. 2. **Keys**: tuning k_2 to boost the activation of v_i for the target concept 'Seattle'.

We test the two methods in various scenarios for different pre-trained transformers, including knowledge editing (Cao et al., 2021; Huang et al., 2023), multi-task tuning (Aribandi et al., 2022), and instruction-tuning (Wei et al., 2022). We generally recognize the superiority of updating keys over updating values. We contend that *compared to directly modifying the model's knowledge (values), altering the mechanism of controlling this knowledge (keys) can be more effective.*

2 EXPERIMENTS

Knowledge Editing is to edit specific information in the model while leaving irrelevant ones uninfluenced. In KE, knowledge refers to the triplets of (subject, relationship, object). The editing is done by maximizing the probabilities of the object tokens given a prompt containing (subject, relationship). We follow the same experiment settings as Meng et al. (2023), more details can be found in Appendix A.1. We simply introduce the evaluation metrics: **Efficacy** measures the editing success, **Paraphrase** measures the editing generalization in different but related contexts, **Specificity** measures the editing locality in unrelated contexts, and **Score** aggregates the three metrics by taking the harmonic mean. Meng et al. (2023) solves a constrained linear problem for model updating. But the same method cannot update K because of the non-linear $f(\cdot)$. We thus update K and V through back-propagation to examine their performances fairly. Table 1 reports the results of updating K or V. While updating each component gives 100% efficacy, updating K shows more generalization and locality, achieving a large performance gain on **Score**. Notably, updating V suffers from much higher time-cost, indicating the hardness of shifting models 'concept' compared with updating the 'concept' usage. Reproducibility details and more results are in Appendix A.1 and A.2.

Table 2: LoRA instruction and multi-task tuning results.

Lora Target	Trainable	MMLU	Bool-q	CB	20-news	Race	COPA	QNLI	6-Avg
LoRA rank=8 on Llama2-7B									
q v	0.062%	46.77	58.20	46.43	61.50	60.50	73.00	57.90	59.59
Value$_{\text{down}}$	0.057%	46.28	60.70	53.57	67.40	60.90	82.00	53.50	63.01
Key$_{\text{gate}}$	0.057%	46.79	69.00	64.29	55.30	63.40	79.00	77.80	68.13
Key$_{\text{up}}$	0.057%	**46.99**	74.10	58.93	56.80	62.50	79.00	78.40	**68.29**
q v Value$_{\text{down}}$	0.119%	46.92	66.30	48.21	59.10	64.80	68.00	62.30	61.45
q v Key$_{\text{gate}}$	0.119%	46.99	59.20	62.50	61.20	66.10	69.00	63.10	63.52
q v Key$_{\text{up}}$	0.119%	**47.13**	64.20	51.79	66.00	65.00	68.00	68.80	**63.96**
LoRA rank=16 on Llama2-7B									
q v	0.124%	46.9	68.30	51.79	65.00	63.60	76.00	52.70	62.90
Value$_{\text{down}}$	0.115%	46.75	67.80	41.07	54.40	62.70	71.00	61.90	59.81
Key$_{\text{gate}}$	0.115%	46.91	71.40	55.36	57.00	62.90	79.00	76.60	67.04
Key$_{\text{up}}$	0.115%	**47.02**	72.60	62.50	66.70	61.60	81.00	64.70	**68.18**

Instruction and Multi-task Tuning. We extend experiments to a more general LoRA-tuning of the Llama2-7B (Touvron et al., 2023) model. This involves separately: 1. tuning the model on the Alpaca dataset (Taori et al., 2023) and then testing on MMLU (Hendrycks et al., 2020). 2. multi-task tuning on six diverse tasks and testing on their respective test sets. Given that Llama2's FFNs employ SwiGLU (Shazeer, 2020), formulated as $\text{FFN}_{\text{SwiGLU}} = (\text{Swish}(x \cdot K_{\text{gate}}^T) \otimes (x \cdot K_{\text{up}}^T)) \cdot V_{\text{down}}$, updating to the K correspond to changes in the K_{gate} and K_{up}, while changes to the V pertain to V_{down}. We explored combinations of LoRA weight additions in the standard attention's q and v, and in the three weighs of FFNs. More settings and reproducibility can be found in the Appendix A.1. According to Table 2, across various settings (LoRA rank=8,16; with and without tuning q and v), the performance of LoRA on Key$_{\text{gate}}$ and Key$_{\text{up}}$ is significantly better than on Value$_{\text{down}}$. More consistent experiment results with 4-bit quantization can be found in Appendix Table 4.

3 CONCLUSION

We empirically find that updating keys within FFNs yields better performance than updating values when tuning LLMs. One possible reason could be updating keys solely involves changing the inner product between the keys and given hidden states. In contrast, updating values requires accurate optimization corresponding to the intended update. These characteristics also translate into better results when LoRA-tuning keys instead of values. It's crucial to note that our objective isn't to propose a better tuning method, so some results may not differ significantly. Instead, we hope the experiment provides insight for updating pre-trained LLMs: updating the mechanism of how the model controls the knowledge may be more effective than directly modifying the knowledge itself.

4 URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Prakash Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. Ext5: Towards extreme multi-task scaling for transfer learning. In *The Tenth International Conference on Learning Representations, ICLR 2022*. OpenReview.net, 2022.

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

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

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pp. 30–45. Association for Computational Linguistics, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, 2023.

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

Noam Shazeer. GLU variants improve transformer. *CoRR*, abs/2002.05202, 2020.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pp. 2440–2448, 2015.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko-lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open founda-tion and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ben Wang. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*, 2023.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022*. OpenReview.net, 2022.

A APPENDIX

A.1 MORE DETAILS FOR KNOWLEDGE EDITING AND LoRA TUNING

Knowledge Editing Task is to edit specific information in the model while leaving irrelevant ones uninfluenced. In knowledge-editing, knowledge refers to the triplets of (subject, relationship, ob-ject). For example, as shown in Fig 1, the knowledge is ('Eiffel Tower', 'locates in', 'Paris') and the editing is to change the object from the existing one to a new one, i.e., from 'Paris' to 'Seattle'. The editing is done by maximizing the probabilities of the object tokens given a language prompt containing the information of (subject, relationship), e.g., 'Eiffel Tower is located in'.

The most difficult part is the data preprocessing and the evaluation. For these experimental settings, we follow the same processions and codebase[1] as Meng et al. (2023). While Meng et al. (2023) solves a linear problem to update V, we update K and V with back-propagation for a fair compar-ison. Back-propagation makes our results simple to implement. We freeze all other modules and update the K or V in the selected layers to update. We adopt the Adam optimizer with a learning rate of $5e-4$ and weight decay of 0.5.

LoRA Tuning Our Instruction tuning code is based on the code base[2]. We built upon the original code by incorporating multi-task Instruction tuning, details of which can be found in our submitted code. All training hyperparameters (including training steps) were directly adopted from the original repository, ensuring consistency across all experiments except for differences in LoRA targets.

We utilized the 20-news group dataset for a 20-class news classification task, provided by the original repository. For a broader task diversity, we selected Bool-q, CB, and COPA from the SuperGLUE benchmark and QNLI from the GLUE benchmark, along with the QA task Race. It's important to note that the sizes of these chosen datasets vary significantly. Therefore, we limit each dataset to a maximum of 5000 samples in constructing the multi-task training dataset. This limitation may result in lower performance on some tasks than standard training approaches.

A.2 MORE EXPERIMENTAL RESULTS ABOUT THE KNOWLEDGE EDITING

We conduct more experiments about knowledge editing, especially in editing multi-counterfacts using GPT2-XL. The reason for not editing multi-counteracts on GPT-J (6B) is that it runs out of memory. Again, we gently refer readers to Meng et al. (2023) for the detailed experiment settings.

Table 3 reports the average metrics of 20877 runs on counteract editing and 19086 runs on zsRE editing. Updating K achieves robust and considerable performance gains on the **Score**, especially since we can observe a large gap on the **Specificity**. Although updating V sometimes consumes slightly less time, it becomes largely inefficient when updating takes longer. The results demonstrate the superiority of updating K rather than V to intervene in pre-trained transformer models about how it processes and stores information.

---

[1] https://github.com/kmeng01/memit
[2] https://github.com/georgian-io/LLM-Finetuning-Hub/tree/main/llama2

Table 3: Results of the Knowledge Editing

Editing Target	Efficacy↑	Paraphrase↑	Specificity↑	Score↑	Time (s)↓
1 Counterfact Editing on GPT2-xl					
On Value	100.00	71.02	68.33	77.49	1.21
On Key	100.00	**83.78**	66.16	**80.97**	**0.97**
10 Counterfacts Editing on GPT2-xl					
On Value	100.00	45.50	66.74	63.88	49.92
On Key	100.00	**57.14**	**73.48**	**72.97**	**9.49**
1 Counterfact Editing on GPT-J (6B)					
On Value	100.00	98.18	6.04	16.15	0.79
On Key	100.00	**98.44**	**28.89**	**54.77**	0.83
1 zsRE Editing on GPT2-xl					
On Value	99.89	57.86	24.28	43.81	1.27
On Key	99.89	**69.90**	**24.75**	**46.35**	**0.88**
10 zsRE Editing on GPT2-xl					
On Value	99.89	98.11	19.36	41.75	0.68
On Key	99.89	96.94	**27.80**	**53.29**	0.71
1 zsRE Editing on GPT-J (6B)					
On Value	99.11	56.32	21.81	40.71	83.12
On Key	98.57	**69.19**	**24.64**	**46.02**	**12.63**

A.3 MORE EXPERIMENTAL RESULTS ABOUT THE LORA TUNING

Our research compares the performance of different LoRA target settings in the more user-friendly context of 4-bit quantization. This approach is significant as it addresses the practical concerns of model size and computational efficiency, which are crucial for real-world applications.

Our findings consistently show that modifying keys outperforms modifying values. This consistent superiority is notable, especially considering that the models with modified keys have fewer trainable parameters than those with modifications in q and v projections. Despite the reduced parameter count, the key-modified models still achieve better results.

Table 4: LoRA instruction and multi-task tuning results under a 4-bit quantization setting

Lora Target	Trainable	Bool-q	CB	20-news	Race	COPA	QNLI	6-Avg
LoRA rank=8								
q v	0.062%	67.80	42.86	53.90	52.80	68.00	48.20	55.59
$Value_{down}$	0.057%	55.30	50.00	65.40	52.80	34.00	48.60	51.02
Key_{gate}	0.057%	71.60	55.71	68.50	53.00	63.00	49.30	60.19
Key_{up}	0.057%	71.30	60.71	61.00	55.80	70.00	54.30	**62.19**
q v $Value_{down}$	0.119%	55.40	41.07	50.80	58.10	58.00	50.00	52.23
q v Key_{gate}	0.119%	59.30	50.00	55.40	62.60	66.00	52.70	57.67
q v Key_{up}	0.119%	69.20	48.21	59.30	61.80	56.00	54.90	**58.24**
LoRA rank=16								
q v	0.124%	70.20	41.07	61.30	56.70	63.00	51.40	57.28
$Value_{down}$	0.115%	62.60	53.57	60.10	53.40	66.00	51.20	57.81
Key_{gate}	0.115%	72.10	46.43	60.20	56.10	64.00	59.90	59.79
Key_{up}	0.115%	68.30	50.00	60.20	57.40	74.00	52.70	**60.43**