Aging with `GRACE`: Lifelong Model Editing with Discrete Key-Value Adaptors

Thomas Hartvigsen[1], **Swaminathan Sankaranarayanan**[1], **Hamid Palangi**[2], **Yoon Kim**[1], **Marzyeh Ghassemi**[1]
MIT[1], Microsoft[2]
`{tomh,swamiviv,yoonkim,mghassem}@mit.edu`, `hpalangi@microsoft.com`

Abstract

Large language models often err during deployment due to non-representative training data or distribution shift in the test set. Recently, *model editors* have been proposed to fix errors by adjusting a pre-trained model's weights. However, these approaches quickly decay a model's performance on upstream data, and forget how to fix previous errors. We propose and study a novel *Lifelong Model Editing* setting, where errors stream into a deployed model and we update the model to correct its predictions without influencing it for unrelated inputs. We propose General Retrieval Adaptors for Continual Editing, or `GRACE`, which learns and caches a particular layer's activations in a codebook as edits stream in, while the original model weights remain frozen. This ensures similar edits are treated similarly without altering the model's performance on unrelated instances. Experimentally, we show that `GRACE` substantially improves over recent model editors.

1 Introduction

Modern machine learning systems perform extremely well on challenging, real-world tasks. Many of the successes stem from giant models trained on massive amounts of data, achieving state-of-the-art performance on challenging tasks in natural language processing [1, 2] and computer vision [3, 4]. However, despite high performance, large models still make critical mistakes during deployment [5]. Further, when models are deployed over long periods of time their error rates increase as data distributions shift, labels shift as annotation guidelines change, or ground-truth information about the world simply changes. For example, a language model trained in 2016 would correctly Barack Obama as president of the United States, but this would be incorrect in 2017 and 2021. As models contain more knowledge, the correct answers will change many times during deployment [6]. In such cases, *Lifelong Model Editing* methods are an ideal fix to large models' mistakes because such edits do not incur costly retraining or degrade the model's upstream performance [7].

One approach to lifelong editing is to finetune a model on data as they arrive [8]. However, finetuning on singular errors is prone to overfitting — even when using customized regularization techniques [9] — and the edited model can catastrophically forget its original training data, devaluing upstream pretraining. Further, edited models can also forget previously-fixed errors, counteracting the objective of editing in the first place [5]. A better approach for lifelong editing could be existing model editors, which update model weights with minimal influence on upstream data. However, these methods need lots of hard-to-obtain training data to make edits [7, 10]. Additionally, regularization approaches [11, 5, 12] rely on sets of semantically-equivalent inputs to preserve upstream model performance. Further, prior model editors [7, 5, 12, 10] have yet to consider *sequential* edits.

In this paper, we introduce *Lifelong Model Editing*. Given a model f_0 that was pretrained on upstream instances $\mathcal{U}$, let f_t denote the *edited* model at timestep t. When deploying f_0, we begin to observe a stream of errors made by the model $\{(X_t^e, y_t^e)\}_{t=1}^T$, where $f_{t-1}(X_t^e) \neq y_t^e \; \forall \; t$. At each step t, given an edit input and edit label pair (X_t^e, y_t^e) our aim is to produce an edited model f_t that 1) Corrects a given error, 2) remembers edits for previous errors, and 3) maintains upstream testing performance.

To address this challenging Lifelong Editing setting, we propose General Retrieval Adaptors for Continual Editing, or GRACE. GRACE modifies individual layers of a frozen, pretrained model f_0, treating it as an encoder. To modify f_0's predictions for a given input, GRACE uses the input to a selected layer as an encoding, with which it selects the nearest key from a codebook memory filled with previously-learned keys. GRACE makes local edits to a large model's behavior by learning activations that influence the model's predictions for specific concepts maintained by separate keys. As new inputs and errors arrive over time, GRACE only needs to decide when to retrieve activations—determined by similarity search—and how to modify existing keys and values. GRACE thus provides a new paradigm for managing conflicts between upstream and downstream behavior.

Our contributions in this work are:

1. We cast model editing in a more realistic lifelong streaming setting. To our knowledge, this setting is unstudied, yet is crucial to successfully deploying large language models.
2. We present GRACE, a novel key-value model editor which learns to cache and retrieve activations for selected layers *using only errors observed during deployment*.
3. Our experiments show that GRACE is a state-of-the-art model editor, ultimately outperforming alternatives on real model editing tasks with shifting data and label distributions.

2 Editing Large Models with GRACE

Problem Formulation Assume we are given a large model f that was pretrained on some upstream dataset $\mathcal{U}$. Let f_0 denote the frozen pretrained model at time step $t = 0$. We then deploy f_0 on a task and monitor its predictions $\hat{y}_t = f(X_t)$ as inputs X_t stream in, one step t at a time. Over time, we receive some errors X_t for which $\hat{y}_t \neq y_t$, the true label for X_t. In order to continue safely deploying f, we aim to *edit* f such that $f(X_t) = y_t$ – let f_t denote the *edited* model at step t. Note that f_t will be different from models used at prior steps and that prior models are discarded. Beyond correcting f_{t-1} on X_t, we also desire that f_t maintains high performance on 1) prior edits $X_{<t}$ and 2) the upstream dataset $\mathcal{U}$. Further, upstream training data are often proprietary or too large, so we assume no access to $\mathcal{U}$ during editing, contrasting prior works' strong assumptions [7].

General Retrieval Adaptors for Continual Editing GRACE presents a novel paradigm for model editing: As errors are identified and corrected over time, GRACE modifies a pre-trained model's behavior *without* altering its weights. GRACE can edit any desired layer in a pre-trained model by learning a discrete codebook $\mathcal{C}$ that modifies a layer's behavior for similar instances. While inspired by recent success in learning discrete key-value models [13], GRACE operates *within* the layers of a pretrained model, and manages keys and values over time **without** pretraining the codebook. A GRACE codebook added to layer l contains the following contents:

- *Keys* ($\mathbb{K}$): Set of keys which are the same size as f_0's activations from layer $l - 1$.
- *Values* ($\mathbb{V}$): Set of values that are learned as the model is deployed and accumulating errors. Each value has the same number of dimensions as the next layer's input.
- *Influence radii* ($\mathcal{E}$): Each key is paired with an *influence radius* ϵ, which are used as a threshold for similarity matching. Given a GRACE layer l and f_0^{l-1}, the pretrained model activations at layer $l - 1$, we use a similarity search over existing keys to find the key closest to f_0^{l-1} via a distance function $d(\cdot)$:

  $$d_{min} = \min_i (d(f_0^{l-1}, K_i^l)).$$

  GRACE is activated at layer l only if $d_{min} \leq \epsilon_k^l$, where k indexes the most similar key. The larger the value of ϵ, the more *influence* the key has, since it covers more of the embedding space. As we discuss below, as GRACE fixes edits over time, ϵ values will shrink and expand to ensure that GRACE layers adapt to changing data distributions.

- *Key labels* ($\mathbb{Y}$): When a new key is added, its corresponding edit label is also stored. By accessing edit labels *only* while editing, keys and their ϵs can be adapted to generalize to similar instances without influencing too much of the embedding space.

To perform inference with a GRACE-edited model, layer l is computed using a discrete key-value search over GRACE's keys:

$$f_t^l(\cdot) = \begin{cases} \texttt{GRACE}^l[f_0^{l-1}], & \text{if } \min_i(d(f_0^{l-1}, K_i^l) - \epsilon_i^l) < 0 \\ f_0^l, & \text{otherwise,} \end{cases}$$

where ϵ_i^l and K_i^l are the influence radius and key i in layer l, respectively. $d(\cdot)$ is a distance function (we use Euclidean distance in our experiments). By using a discrete similarity search, if a new input is unlike any cached keys, GRACE simply defers to f_0's pretrained weights. This way, if the edit instances are out-of-distribution for the training data, GRACE layers can avoid interference with upstream data by leaving the model weights unaltered. Further, edits are usually rare compared to streaming inputs, so a GRACE-edited model will often defer to a pretrained layer's outputs.

When an edit is required, f_0 serves as an encoder, computing an embedding for an instance at layer l. Then, f_0^l serves as a query across any existing keys in the GRACE codebook for layer l. A GRACE layer can perform one of the following operations at any given time step:

1. **KEY-INITIALIZE**: If no keys are present, GRACE initializes the query as a key. This step only happens when the model makes its very first error—otherwise, GRACE always either Adds a key or Splits an existing key. We initialize a new key's value to be a vector of shape $|f^l|$, the number of dimensions of the output embedding. Then the value is then updated using gradient descent with standard finetuning loss to encourage the model to predict the edit label given the value as input to the next layer.
2. **KEY-ADD**: If the input embedding f_0^{l-1} does not fall within the ϵ radius of any existing keys according to distance function $d(\cdot)$, then a new key is initialized to be f_0^{l-1} along with a corresponding value v, base influence radius ϵ^l, and edit label y^e. The value is learned as described in Step 1.
3. **KEY-UPDATE**: If f_0^{l-1} is similar-enough to any existing keys—determined by whether or not the query lands inside the ϵ radius of any keys, we need to ensure that the matched key's activation carries the same semantic content as the edit input. To do this, we compare the current edit's label to the label cached along with the matched key:
   (a) *If the query's nearest key has the same label as the edit label*, **expand** the nearest key's ϵ to encompass the query:
   $$\{K_i^l : [V_i^l, \epsilon_i^l]\} \rightarrow \{K_i^l : [V_i^l, d(f_0^l, K_i^l)]\}$$
   (b) *If the query's nearest key has a different label from the edit label*, **split** the nearest key into two keys by (1) decreasing the influence radius of the nearest key, then (2) creating a new key-value pair where the key is the query. The new key is the activation of f_0 at layer $l-1$ for the current edit input.
   $$\{K_i^l : V_i^l, \epsilon_i^l\} \rightarrow \left\{ \begin{array}{ccc} K_i^l : & [V_i^l, & 0.5 * d(f_0^l, K_i^l)] \\ f_0^l : & [V_{i+1}^l, & 0.5 * d(f_0^l, K_i^l)] \end{array} \right\}$$

As edits stream in, by continuously adding and updating GRACE's keys and values, the embedding space for a selected layer l becomes partitioned according to which instances need modified outputs. When *not* performing edits, these operations are bypassed, and keys are entirely frozen, regardless of whether or not the instance lands within a key's influence. Overall, GRACE introduces a new model editing paradigm in which edits can be made sequentially, similar edits are encouraged to be edited similarly, and the ultimate influence of new edits can be controlled and monitored explicitly.

2.1 Training and Inference with GRACE Layers

When an edit is required and GRACE layers are used, the closest key matching the edit input query is found using a similarity search followed by the key update steps described above. Once a closest key is either found or newly created, the corresponding value is returned. This value serves as the activation of the next layer. Then, the final model output is computed and a downstream loss is calculated between the predicted label and the edit label. During backpropagation, only the values of the GRACE layers are updated based on the downstream loss, thereby learning activations that alter the model's predictions.

During inference, at each GRACE layer, the key closest to the query embedding is identified through a similarity search. In our experiments, we find that euclidean distance works well, similar to [13]. The corresponding value is returned as an activation to the next layer, and this step is repeated for all GRACE layers added to the model. GRACE naturally applies to all recent transformer models, as any layers that compute new updates can be edited using it.

3 Experiments

To evaluate GRACE, we investigate three points. (If you drop the synthetic stuff, you need to edit this) First, using synthetic data, we validate that GRACE indeed learns local fixes for pre-trained models. Second, using a real Question Answering task, we evaluate GRACE's capacity to alter a language model's predicted tokens over long deployment periods. Third, we experiment with a real label-shift case, evaluating lifelong model editing when label distributions change during deployment. We compare GRACE with three alternatives. First, we finetune a model's weights on errors as they arrive. Second, we use a streaming version of MEND [7], which trains a hypernetwork to edit a model's weights using streaming errors as training data. Third, we compare against a memory network-based adaptor, which serves as a softer version of GRACE.

Implementation Details We experiment with T5 [14] for question answering, though GRACE is general and applicable to other large NLP models as well. Following [7], we edit the dense-relu-dense layer of the last encoder block of a 60 million parameter model.

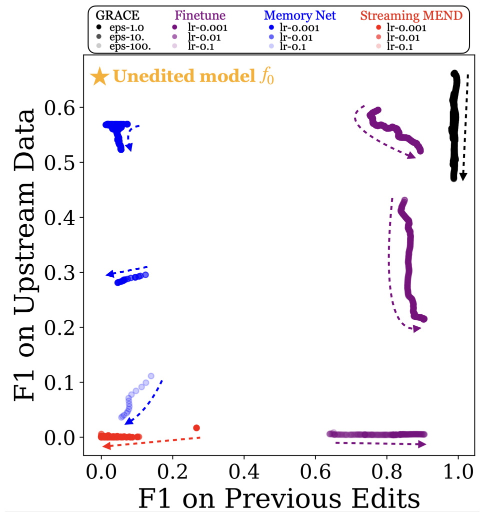
Metrics To evaluate lifelong model editors, we track two key metrics over time: (1) **Upstream performance** is the model's performance on a portion of its training set, indicating how much the model remembers its past knowledge, and (2) **Online performance** is the edited model's accuracy on the history of streaming edits. Both should ideally be high, though they often contradict in practice.

3.1 Question Answering with Shifting Answers

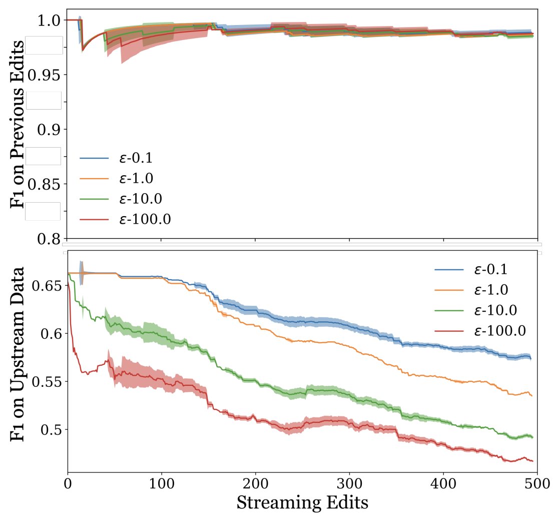
Experiment: Pretraining and finetuning large language models is an increasingly-successful approach to Question Answering (QA) [6], especially for open-domain tasks [15]. In recent methods for open domain QA, language models learn to correctly answer questions without any context [16]. While such approaches are growing more successful, as the world changes the correct answers to open-domain questions naturally change. For example, when asking "Who is the president of the United States?" the correct answer will change every few years. As correct answers shift, pretrained models age and underperform [11].

We study methods for correcting for QA shift through the lens of model editing. We edit a T5 model [6] that was pretrained on the Natural Questions dataset (NQ) [17] to correct its mistakes on questions from the downstream zsRE dataset [18] using splits from [12]. For repeated upstream evaluation, we sample 1000 instances from NQ. For editing, we sample 1000 edits from zsRE, including random samples of 5 rephrasings for each question. We focus our QA evaluation on the standard F1 metric, measuring overlap between predicted and true tokens.

Before any editing, T5 achieves an F1 score of 0.66. As zsRE questions stream in, we first check that the model makes a mistake. We run this check for every instance in this experiment as the model is being updated frequently over time. If T5 makes a mistake, we then pass it to GRACE to edit the model. Since edits stream in, errors are identified with respect to only the current version of the model at timestep t. Over 1000 edits, each method ends up making errors on about 500 inputs. After each edit, we report the new model's performance on a holdout set of 1000 questions from the upstream Natural Questions dataset. An ideal model editor should only update the requested edits, leaving unrelated instances untouched. We also report Online F1: the edited model's F1 score on the growing set of previous edits. A successful model editor should make new edits without forgetting old edits.



(a) Comparing all methods' performance on upstream data and previous edits. Ideal performance is in the upper right corner. Arrows denote each model's performance throughout streaming.



(b) Impact of parameter ϵ on GRACE. By increasing ϵ, more upstream data lands within the radius of influence for any given key, decaying upstream F1.

Figure 1: Results for lifelong editing for context-free QA editing. Each editor sequentially updates the same pretrained model on around 500 errors sampled from the zsRE dataset.

Result: As we show in Figure 1a, GRACE successfully edits the pretrained T5 model while remembering upstream information. For the Finetuning, Memory Network, and MEND baselines, we vary the learning rate. For GRACE, we vary ϵ, the initial radius of influence for keys as they are initialized. As each method is trained on streams of edits, we show arrows indicating their progression through the multi-objective space over time. The optimal performance is in the upper right corner. Indicated in black, GRACE succeeds to maintain high F1 scores on upstream data, while also remembering previous edits. As described above, this is by adding a small number of new parameters in the GRACE codebook. However, the number of new parameters is tiny compared to the pretrained model. As expected, as learning rates increase for Finetuning and the Memory Network, their F1 performance on upstream data severely decreases. As finetuning progresses, the models move farther from their initial state. Further, indicated by the arrows pointing to the lower right, the Finetuning succeeds to generally increase F1 on all previous edits over time, indicating that.

As reported in Figure 1b, we also tune ϵ, which controls the influence radius of each stored key in a layer's GRACE codebook. Larger ϵ values result in few keys that cover large portions of the embedding space. As expected, this decays the upstream performance, since more upstream data will conflict with edits. We also notice that each ϵ leads to near-perfect F1 on previous edits, indicating perfect memory. This may occur when edits are far apart in the embedding space and we hypothesize that F1 on previous edits will eventually drop as the number of edits increases. Still, when performing 500 edits in a row is a substantial improvement over the state-of-the-art alternatives.

4 Conclusions

Language models are quickly becoming larger and are being applied to a diverse set of downstream tasks. However, they are often computationally prohibitive to finetune and easily forget past knowledge. In this work, we proposed a realistic problem setting where we edit such large models, *Lifelong Model Editing*, and presented GRACE, a plug-in module that can wrap around any given layer in a large pretrained model. GRACE layers (1) retain the functionality of the original model, thereby minimizing catastrophic forgetting and (2) adapt to changing data distributions by storing a codebook of cached activations that can grow or shrink over time. We demonstrate GRACE's efficacy by showing that GRACE provides the best trade-off between upstream performance and accuracy on streaming edits among competing model editing baselines and finetuning methods.

References

[1] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[5] A. Sinitsin, V. Plokhotnyuk, D. Pyrkin, S. Popov, and A. Babenko, "Editable neural networks," in *International Conference on Learning Representations*, 2019.

[6] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[7] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning, "Fast model editing at scale," in *International Conference on Learning Representations*, 2022.

[8] C. Lee, K. Cho, and W. Kang, "Mixout: Effective regularization to finetune large-scale pretrained language models," in *International Conference on Learning Representations*, 2020.

[9] B. Y. Lin, S. I. Wang, X. Lin, R. Jia, L. Xiao, X. Ren, and S. Yih, "On continual model refinement in out-of-distribution data streams," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3128–3139, 2022.

[10] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, and C. Finn, "Memory-based model editing at scale," in *International Conference on Machine Learning*, pp. 15817–15831, PMLR, 2022.

[11] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," *arXiv preprint arXiv:2202.05262*, 2022.

[12] N. De Cao, W. Aziz, and I. Titov, "Editing factual knowledge in language models," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6491–6506, 2021.

[13] F. Träuble, A. Goyal, N. Rahaman, M. Mozer, K. Kawaguchi, Y. Bengio, and B. Schölkopf, "Discrete key-value bottleneck," *arXiv preprint arXiv:2207.11240*, 2022.

[14] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer.," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[15] J. Prager *et al.*, "Open-domain question–answering," *Foundations and Trends® in Information Retrieval*, vol. 1, no. 2, pp. 91–231, 2007.

[16] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 874–880, Association for Computational Linguistics, 2021.

[17] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov, "Natural questions: a benchmark for question answering research," *Transactions of the Association of Computational Linguistics*, 2019.

[18] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, "Zero-shot relation extraction via reading comprehension," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342, 2017.