Self-Adapting Language Models

Adam Zweiger* **Jyothish Pari*†** **Han Guo** **Ekin Akyürek** **Yoon Kim** **Pulkit Agrawal†**
Massachusetts Institute of Technology
{adamz, jyop, hanguo, akyurek, yoonkim, pulkitag}@mit.edu

Abstract

Large language models (LLMs) are powerful but static; they lack mechanisms to adapt their weights in response to new tasks, knowledge, or examples. We introduce **Se**lf-**A**dapting **L**LMs (SEAL), a framework that enables LLMs to self-adapt by generating their own finetuning data and update directives. Given a new input, the model produces a *self-edit*—a generation that may restructure the information in different ways, specify optimization hyperparameters, or invoke tools for data augmentation and gradient-based updates. Through supervised finetuning (SFT), these self-edits result in persistent weight updates, enabling lasting adaptation. To train the model to produce effective self-edits, we use a reinforcement learning loop, using the downstream performance of the updated model as the reward signal. Unlike prior approaches that rely on separate adaptation modules or auxiliary networks, SEAL directly uses the model's generation to parameterize and control its own adaptation process. Experiments on knowledge incorporation and few-shot generalization show that SEAL is a promising step toward language models capable of self-directed adaptation in response to new data. Our website and code is available at https://jyopari.github.io/posts/seal.

1 Introduction

Large language models (LLMs) pretrained on vast text corpora exhibit remarkable abilities in language understanding and generation [1, 2, 3, 4, 5]. However, adapting these powerful models for specific tasks [6], integrating new information [7], or mastering novel reasoning skills [8] remains challenging due to the limited availability of task-specific data. In this paper, we explore an intriguing hypothesis: can an LLM self-adapt by transforming or generating its own training data and learning procedure?

As an analogy, consider a human student preparing for the final exam of a machine learning class. Many students rely on their notes to prepare for the exam. These notes are often derived from the lecture content, textbooks, or information available on the internet. Instead of relying on the raw content, assimilating and rewriting the information in the form of notes often improves the ability of students to understand the content and answer exam questions. This phenomenon of reinterpreting and augmenting external knowledge in a way that is easier to understand is not limited to just taking exams, but seems to be universally true of human learning across tasks. Furthermore, different humans assimilate information in different ways—some might condense the information into a visual diagram, some into text, or some might rely more on concrete mathematical descriptions.

Such assimilation, restructuring, or rewriting of data as part of the learning process is in contrast with how large language models (LLMs) are typically trained and deployed. Given a new task, current LLMs consume and learn from the task data "as-is" via finetuning or in-context learning [9, 10, 11, 12]. However, such data may not be in an optimal format (or volume) for learning, and

*Equal contribution.
†Improbable AI Lab, CSAIL MIT

Preprint. Under review.

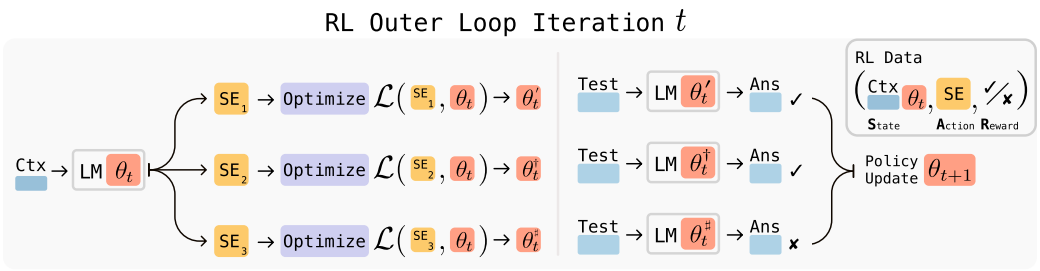

Figure 1: **Overview of SEAL.** In each RL outer loop iteration, the model generates candidate self-edits (SE)—directives on how to update the weights—applies updates, evaluates performance on a downstream task, and uses the resulting rewards to improve the self-edit generation policy.

current approaches do not enable models to develop bespoke strategies for how to best transform and learn from their training data.

As a step towards scalable and efficient adaptation of language models, we propose equipping LLMs with the ability to generate their own training data and finetuning directives for utilizing such data. In particular, we introduce a reinforcement learning algorithm that trains LLMs to generate **"self-edits"**—natural-language instructions that specify the data and, optionally, the optimization hyperparameters for updating the model's weights (see Figure 1). We refer to such models as **Se**lf-**A**dapting **LLM**s (SEAL).

We evaluate SEAL on two applications. We first consider the task of integrating new factual knowledge into an LLM. Rather than finetuning directly on the passage text, we finetune on synthetic data generated by the SEAL model. Our results show that, following reinforcement learning (RL) training, finetuning on self-generated synthetic data improves question-answering performance on the no-passage-in-context variant of SQuAD [13] from 33.5% to 47.0%. Notably, self-generated data from SEAL outperforms synthetic data generated by GPT-4.1.

We further evaluate SEAL on few-shot learning on a simplified subset of the ARC-AGI benchmark [14], where the model leverages a set of *tools* to autonomously select both synthetic data augmentations and optimization hyperparameters (e.g., learning rate, training epochs, selective loss computation over token types). Our experiments demonstrate that automatic selection and configuration of these tools using SEAL enhances performance compared to both standard in-context learning (ICL) and self-editing *without* RL training to use the tools effectively. These results collectively show that SEAL is a versatile framework for enabling language models to self-adapt.

2 Related Work

Synthetic Data Generation. Creating synthetic data for training is increasingly common, from large-scale pretraining datasets [15, 16, 17] to task-specific data augmentation [18, 19] and instruction-tuning sets [20, 21]. For incorporation of a smaller-sized corpus, Yang et al. [22] use synthetic data generation via graph-based prompting. SEAL builds on this line of work by using reinforcement learning to train a *generative policy* that directly maximizes the downstream utility of synthetic data when applied for gradient-based self-updates, rather than relying on static or heuristic generation strategies that are manually tuned and therefore potentially not scalable or optimal.

Knowledge Updating. Several recent works aim to modify or inject factual knowledge into language models via weight updates. Some methods attempt to directly locate specific parameters that correspond to individual facts [23, 24, 25]. Others propose generating additional finetuning data using the information in context [26, 27, 22, 28, 29]. We adopt the latter strategy, following Akyürek et al. [27], who propose generating logical implications of a fact and finetuning on them, and Lampinen et al. [28], who show that implication-based finetuning can even outperform in-context learning. We build on these approaches by *training* models through RL to generate more optimal finetuning data. Park et al. [29] show that prompting language models to generate question–answer (QA) pairs directly can outperform implication-style prompting. Because the SEAL framework is agnostic to the prompt and format of the self-edit data, it can also be trained to generate QA pairs or

other output formats, as explored in §C. With sufficient compute, a cold-start setup, where the model discovers the most effective format without guidance from prompting, may also be viable.

Test-Time Training. Test-Time Training (TTT) temporarily adapts model weights based on the input the model receives [30, 31, 32, 33]. Akyürek et al. [33] show that combining TTT with ICL enables gradient-updates to outperform standard ICL in the few-shot setting. SEAL can be viewed as incorporating a round of TTT in its inner-loop optimization, leveraging TTT's efficiency relative to full-scale training to perform multiple updates and reward the generated data that yields the greatest performance gain. Although our method is trained using single-example TTT episodes, we demonstrate in the knowledge incorporation setting that it generalizes to the continued pretraining setting—where placing data directly in context is no longer feasible.

Reinforcement Learning for LLMs. Reinforcement learning has played a central role in improving LLM behavior, originally through RLHF [34]. More recently, RL with verifiable rewards has been applied to boost reasoning performance by optimizing the model directly for task success [35, 36, 37]. SEAL applies RL not to optimize final answers or trace revisions, but to optimize the generation of *self-edit* data that is then used for weight updates.

Meta-Learning and Self-Modifying Systems. SEAL embodies meta-learning principles [38, 39, 40] by learning an adaptation strategy—how to generate effective self-edits—via its outer optimization loop. The goal is to learn *how to learn* efficiently from task contexts. Meta-learning has similarly been applied in reinforcement learning [41, 42, 43, 44, 45], where models are trained with a meta-objective to rapidly adapt to new tasks. A natural extension of this line of work is self-referential networks, where models modify their own parameters Schmidhuber [46], Irie et al. [47]. In the domain of large language models, recent work has applied meta-learning principles to improve LLM adaptation [48, 49]. Notably, Hu et al. [49] trained a smaller model to output token-specific weights during finetuning on a corpus, addressing a knowledge incorporation task similar to ours. However, SEAL offers greater generality across domains by leveraging the model's existing generative capabilities to parametrize updates.

Self-Improvement. Several recent works fall under the umbrella of self-improvement or self-training. Methods such as RLAIF [50, 51] and self-rewarding language models [52, 53] use the model itself to provide reward signals, leveraging the observation that judging outputs is often easier than generating them [54]. Other recent works improve performance on mathematical tasks by using majority-vote or model confidence as reinforcement learning rewards, enabling performance improvement without access to ground-truth labels [55, 56, 57, 58, 59]. However, all of these methods are fundamentally limited by the model's current evaluation abilities and self-consistency. In contrast, we view self-improvement through interaction with external data as a more powerful and scalable path. SEAL learns how to best utilize this external data for self-improvement.

3 Methods

We propose Self-Adapting LLMs (SEAL), a framework that enables language models to improve themselves by generating their own synthetic data and optimization parameters ("self-edits") in response to new data. The model is trained to produce these self-edits directly through token generation with the data provided in the model's context. Self-edit generation is learned via reinforcement learning (RL) where the model is rewarded for generating self-edits (`SE`) that, when applied, improve the model's performance at the target task. SEAL can therefore be interpreted as an algorithm with two nested loops: an *outer RL loop*, which optimizes the self-edit generation, and an *inner update loop*, which uses the generated self-edit to update the model via gradient descent. Our method can be seen as an instance of meta-learning where we meta-learn how to generate effective self-edits.

3.1 General Framework

Let θ denote the parameters of the language model $\texttt{LM}_\theta$. SEAL operates on individual task instances (C, τ) where C is a context containing information relevant to the task, and τ defines the downstream evaluation used to assess the model's adaptation. For example, in knowledge incorporation, C is the passage intended to be integrated into the model's internal knowledge, and τ is a set of questions and

associated answers about the passage. In few-shot learning, C includes few-shot demonstrations of a novel task, and τ is the query input and ground-truth output. Given C, the model generates a self-edit $\texttt{SE}$—the form of which varies by domain (see §3.2)—and updates its parameters via supervised finetuning: $\theta' \leftarrow \texttt{SFT}(\theta, \texttt{SE})$.

We optimize the self-edit generation process using reinforcement learning: the model takes an *action* (generating $\texttt{SE}$), receives a *reward* r based on $\texttt{LM}_{\theta'}$'s performance on τ, and updates its policy to maximize expected reward:

$$\mathcal{L}_{\text{RL}}(\theta_t) := -\mathbb{E}_{(C,\tau)\sim\mathcal{D}}\left[\mathbb{E}_{\texttt{SE}\sim\text{LM}_{\theta_t}(\cdot|C)}\left[r(\texttt{SE},\tau,\theta_t)\right]\right]. \tag{1}$$

Unlike in standard RL setups, the reward assigned to a given action in our setting depends on the model *parameters* θ at the time the action is taken (since θ is updated to θ', which is then evaluated). As a result, the underlying RL state must include the policy's parameters and is given by (C, θ), even though the policy's observation is limited to C (placing θ directly in context is infeasible). The implication of this is that (state, action, reward) triples collected with a previous version of the model, θ_{old}, may become stale and misaligned for the current model θ_{current}. For this reason, we adopt an on-policy approach, in which self-edits are sampled from—and, crucially, rewards are computed using—the current model.

Algorithm 1 Self-Adapting LLMs (SEAL): Self-Edit Reinforcement Learning Loop

Input: LM_θ, dataset $\mathcal{D} = \{(C, \tau)\}$
for outer iteration $t = 1, 2, \ldots$ **do**
  Sample $(C, \tau) \sim \mathcal{D}$
  Generate self-edit $\texttt{SE} \sim \text{LM}_\theta(\cdot \mid C)$
  Inner Loop Update: $\theta'_t \leftarrow \texttt{SFT}(\theta_t, \texttt{SE})$
  Evaluate: $\texttt{Ans} \sim \text{LM}_{\theta'_t}(\cdot \mid \tau)$
  Compute reward: $r \leftarrow r(\texttt{Ans}, \tau)$
  Update: $\theta_{t+1} \leftarrow \texttt{RL\_Update}(\theta_t, r, \texttt{SE})$
end for

We experimented with various on-policy methods such as Group Relative Policy Optimization (GRPO) [60] and Proximal Policy Optimization (PPO) [61], but found the training to be unstable. Instead, we adopt ReST^{EM} [36], a simpler approach based on filtered behavior cloning—also known as "rejection sampling + SFT" [62, 63, 64, 35, 65].

ReST^{EM} can be viewed as an expectation-maximization (EM) procedure: the **E-step** samples candidate outputs from the current model policy, and the **M-step** reinforces only those samples that receive positive reward through supervised finetuning. This approach optimizes an approximation of our objective (1) under the binary reward:

$$r(\texttt{SE},\tau,\theta_t) = \begin{cases} 1 & \text{If on } \tau\text{, adaptation using } \texttt{SE} \text{ improves } \text{LM}_{\theta_t}\text{'s performance}^2 \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

More precisely, in optimizing (1), we must compute the gradient $\nabla_{\theta_t}\mathcal{L}_{\text{RL}}$. However, as we noted, the reward term $r(\texttt{SE}, \tau, \theta_t)$ depends on θ_t in our setup but is not differentiable. We address this by treating the reward as fixed with respect to θ_t. With this approximation, the Monte-Carlo estimator for a minibatch of N contexts and M sampled self-edits per context becomes

$$\nabla_{\theta_t}\mathcal{L}_{\text{RL}} \approx -\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} r_{ij}\,\nabla_{\theta_t}\log p_{\theta_t}(\texttt{SE}_{ij} \mid C_i) \tag{3}$$

$$= -\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} r_{ij}\sum_{s=1}^{T}\nabla_{\theta_t}\log p_{\theta_t}(y_s^{(i,j)} \mid y_{<s}^{(i,j)}, C_i), \tag{4}$$

where p_{θ_t} denotes the model's autoregressive distribution and $y_s^{(i,j)}$ is the s^{th} token of self-edit $\texttt{SE}_{ij}$, the j^{th} sample for context C_i. Since sequences with $r = 0$ can be ignored in (4), we have shown that ReST^{EM}, with simple "SFT on good self-edits," indeed optimizes (1) under the binary reward (2) (with a stop-gradient applied to the reward term). The SEAL training loop is summarized in Alg. 1.

Finally, we note that while the implementation in this work uses a single model for both generating self-edits and learning from these self-edits, it is also possible to decouple these roles. In such a "teacher-student" formulation [66], a student model would be updated using edits proposed by a separate teacher model. The teacher would then be trained via RL to generate edits that maximize student improvement.

[2]The reward may also be assigned to the single self-edit that yields the greatest improvement among sampled candidates, which we do in knowledge incorporation, rather than to all edits that yield a positive improvement.

3.2 Domain Instantiations

We instantiate the SEAL framework in two distinct domains: knowledge incorporation and few-shot learning. These domains were chosen to highlight two complementary forms of model adaptation: (1) the ability to integrate new information into a model's weights so that it can be recalled without relying on context (evaluated using a no-context variant of SQuAD) and (2) the ability to generalize to novel tasks after seeing only a small number of examples (evaluated using ARC).

Knowledge Incorporation. Our goal is to efficiently incorporate the information provided in a passage into the model's weights. A promising recent approach involves using a language model to generate content derived from the passage, followed by finetuning on both the original passage and the generated content [26, 27, 22, 28, 29]. While the form of generated content may vary, we adopt what we consider the canonical format: *implications derived from the passage*. This approach, introduced in deductive closure training [27], converts a given context C into a set of implications $\texttt{SE} = \{s_1, s_2, \ldots, s_n\}$ by prompting the model to "List several implications derived from the content." The output may include inferences, logical consequences, or restatements of the original passage. In §C, we also explore alternative prompts such as "rewrite the passage in different ways" or "rewrite in a question-answer format" and show that our method improves performance by similar or greater margins regardless of the base prompt.

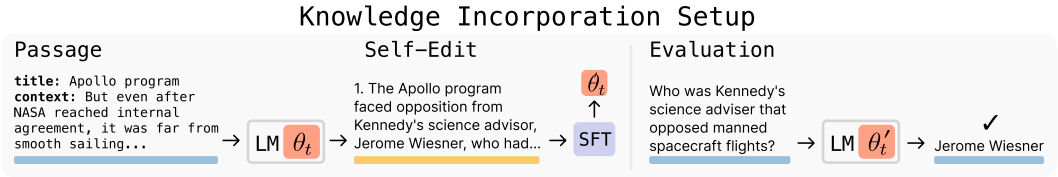

Figure 2: **Knowledge Incorporation Setup.** Given a new passage, the model generates synthetic data (the *self-edit*) in the form of "implications" of the passage. We then finetune on these outputs using LoRA. The updated model is evaluated on questions about the passage *without* access to the original text, and the resulting accuracy serves as the reward signal for reinforcement learning.

These self-generated statements form the training data for a supervised finetuning (SFT) update: we compute the standard causal language-modeling loss over each sequence s_i and update the model parameters, yielding θ'. Since the amount of data per update is small and the number of updates we do in total is large, we use low-rank adapters (LoRA [67]) for efficient, lightweight tuning. Finally, the adapted model $\texttt{LM}_{\theta'}$ is evaluated on the task τ. This process is shown in Figure 2.

During RL training, the adapted model's accuracy on τ defines the reward r that drives the outer RL optimization. This trains the model to restructure the passage in a way that is most effective for assimilation via finetuning.

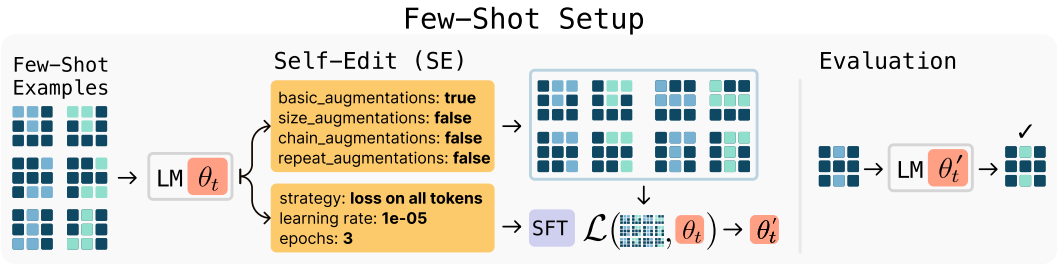

Figure 3: **Few-Shot Learning with SEAL.** Left: example ARC demonstrations. Center: the model generates a self-edit specifying augmentations and training hyperparameters. Right: the adapted model is evaluated on a held-out test input.

Few-Shot Learning. The Abstraction and Reasoning Corpus (ARC) [8] is a benchmark designed to test abstract reasoning and generalization from very limited examples. Each task includes a small set of input-output demonstrations and a held-out test input whose correct output must be predicted.

We adopt the test-time training (TTT) protocol of Akyürek et al. [33], where augmentations of the few-shot examples are used to perform gradient-based adaptation. Rather than relying on manually tuned heuristics for selecting augmentations and optimization settings, we train SEAL to learn these decisions. This setting tests whether SEAL can autonomously configure the adaptation pipeline—determining which augmentations to apply and what optimization parameters to use.

To implement this, we define a set of **tools**, each of which is a pre-defined function from Akyürek et al. [33] that transforms data or specifies training parameters. These include:

- **Data augmentations:** rotations, flips, reflections, transpositions, resizing operations (e.g., changing grid resolution), and chained or repeated transformations.
- **Optimization parameters:** learning rate, number of training epochs, and whether the loss is computed over all tokens or only output tokens.

The model is prompted with a task's few-shot demonstrations and generates a self-edit, which in this case is a specification of which tools to invoke and how to configure them, as shown in Figure 3. This self-edit is then applied to adapt the model via LoRA-based finetuning. The adapted model is evaluated on the held-out test input, and the result determines the reward for training the self-edit generation policy.

4 Results

In this section we empirically evaluate SEAL across our two adaptation domains: few-shot learning and knowledge incorporation. Full training, hyperparameter, and evaluation details are provided in §A and §B.

4.1 Few-Shot Learning

We conduct our experiments using `Llama-3.2-1B-Instruct`, a small open-source model with no ARC-specific pretraining. Since most ARC tasks are challenging for models that have not been pretrained on ARC, we curate a subset of 11 tasks from the ARC training set and 8 from the evaluation set, filtered to ensure that they are solvable under optimal TTT configurations for a base `Llama-3.2-1B-Instruct`. While this is a small number of examples, note that Akyürek et al. [33] used the same TTT configuration for all tasks, and thus we do not need a large training set for learning a fixed self-edit. More details are included in §A.

The model is trained using ReST^{EM} by sampling 15 self-edits per training task. Each self-edit is applied individually to generate 15 updated models, which are then evaluated on the corresponding held-out test example. We reinforce only those self-edits that lead to correctly adapted models, i.e., models that produce the correct output for the test input after adaptation.

After training, we evaluate the model by generating 5 self-edits per held-out evaluation task and apply each one independently. We then report the percentage of self-edits that lead to correct outputs, yielding a success rate that reflects the quality of the learned self-edit generation policy.

We compare against the following baselines:

1. **ICL (In-Context Learning):** `Llama-3.2-1B-Instruct` is prompted with the given few-shot examples using Akyürek et al. [33]'s protocol, and directly queried on the test input.
2. **TTT + Self-Edit (w/o prior RL):** `Llama-3.2-1B-Instruct` performs test-time training (TTT) using few-shot examples and synthetic augmentations, but without any prior RL to optimize which augmentations or training configurations to use.
3. **Oracle TTT:** The model performs test-time training (TTT) using the optimal human-crafted configuration from Akyürek et al. [33]. This provides an upper bound of our method.

We record results in Table 4.1. SEAL substantially improves adaptation success rate compared to baselines: 72.5% vs. 20% (with self-edits from the base model without RL training) and 0% (no adaptation), though performance remains below Oracle TTT, suggesting room for further improvement.

Method	Success Rate (%)
ICL	0
TTT + Self-Edit (w/o prior RL)	20
SEAL	72.5
Oracle TTT	100

Table 1: Few-shot Abstract Reasoning

4.2 Knowledge Incorporation

We experiment with `Qwen2.5-7B` on incorporating novel factual content from SQuAD passages [13]. We use the relatively simple SQuAD dataset because its passages can be fully "understood" by the base model in-context, yet the model cannot reliably answer questions about them *without* that context. We do 2 rounds of ReST^{EM} with a batch of 50 contexts (see §B for further details). We compare SEAL on knowledge incorporation against the following baseline approaches:

1. **Base Model:** The pretrained model is evaluated on downstream QA tasks without any adaptation or access to the passage.
2. **Train on Passage Only:** The model is finetuned directly on the passage using the standard language modeling loss, without any synthetic data.
3. **Train on Passage + Synthetic Data:** The model is trained on the passage along with self-generated implications. This is the same setup as SEAL but without any prior RL training.
4. **Train on Passage + GPT-4.1 Synthetic Data:** The model is trained on the passage along with model-generated implications collected from GPT-4.1 via the OpenAI API.

Table 4.2 reports mean no-context SQuAD accuracy under two regimes: single-passage ($n = 1$) and continued pretraining (CPT, $n = 200$). In the single-passage setting, finetuning directly on the passage yields a negligible gain over the frozen base model (33.5% vs. 32.7%), confirming that using the raw data alone is insufficient. Augmenting with synthetic implications generated by GPT-4.1 boosts accuracy to 46.3%, an improvement of 12.8 percentage points over the passage-only baseline. Using synthetic data produced by the base `Qwen-2.5-7B` model yields 39.7%, a 6.2-point increase. After reinforcement learning, SEAL further improves accuracy to **47.0%**, notably outperforming using synthetic data from GPT-4.1, despite being a much smaller model.

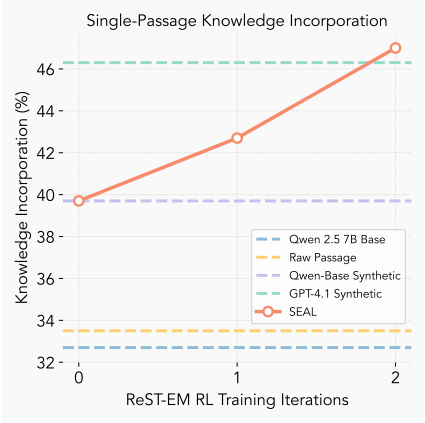

Figure 4: **Accuracy over RL iterations.** Each iteration consists of a minibatch of 50 contexts, each with 5 sampled self-edits. SEAL surpasses GPT-4.1 synthetic data after two iterations of ReST^{EM} on the no-context SQuAD set.

In the CPT setting, the model assimilates information from $n = 200$ passages in a single continued pretraining run. It is then evaluated on the union of all 974 corresponding questions. In this setting, we sample 5 self-edit generations for each passage and take the aggregate synthetic dataset for continued pretraining. As shown in Table 4.2, we observe a similar ranking of methods as in the single-passage case. SEAL again outperforms all baselines, achieving 43.8% accuracy. While the absolute performance is lower than in the single-passage setting—likely due to increased gradient interference—the relative improvements remain consistent. This suggests that the editing policy discovered by SEAL generalizes *beyond* its original RL setup of creating synthetic data in a single generation for a single passage.

Figure 4 tracks accuracy after each outer RL iteration. Two iterations suffice for SEAL to overtake GPT-4.1 data; subsequent iterations yield diminishing returns, suggesting that the policy quickly converges to an edit style that distills the passage into easily learnable atomic facts (see qualitative examples in Figure 5). All results use tuned hyperparameters (see §B).

Method	Single Passage (n = 1)	Continued Pretraining (n = 200)
Base model	32.7	32.7
Train on Passage	33.5	32.2
Train on Passage + Synthetic	39.7	41.0
Train on Passage + GPT-4.1 Synthetic	46.3	39.4
SEAL	**47.0**	**43.8**

Table 2: Knowledge Incorporation Performance across Passage Settings

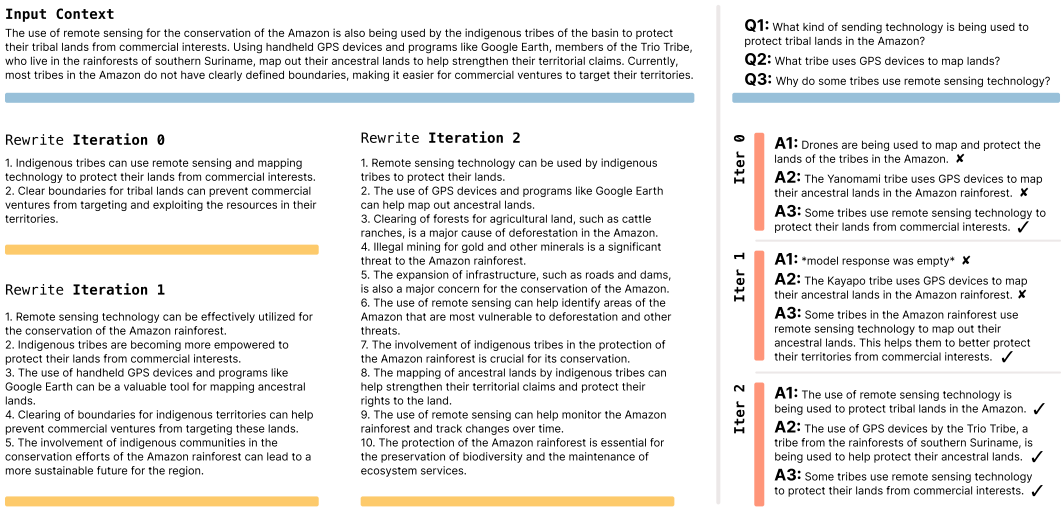

Figure 5: **Example Knowledge Incorporation Self-Edits Across RL Iterations.** In this example, we see how RL leads to the generation of more detailed self-edits, which in turn results in better performance. While the progression is clear in this case, the differences across iterations are sometimes more subtle in other examples. We show in §C that prompting for longer self-edits is effective, and that RL training further improves performance by a similar margin.

5 Limitations

Catastrophic forgetting. One key motivation we had for enabling language models to self-edit is to move towards the ultimate goal of continual learning—allowing models to incorporate new information over time, whether through agentically interacting with an environment or through standard training. While our earlier experiments assess how well SEAL adapts to individual edits in isolation, a more ambitious goal is to support *sequences* of edits: can the model adapt to new information repeatedly while preserving prior knowledge?

This question relates directly to the challenge of *catastrophic forgetting* [68, 69], where new updates interfere destructively with past learning. We do not explicitly optimize for retention in our current training setup, but we aim to establish a baseline for how well SEAL handles sequential self-edits without dedicated mechanisms for handling catastrophic forgetting. To test this, we simulate a continual learning setting in the knowledge incorporation domain. The model receives a stream of test passages, each triggering a new self-edit. After each update, we

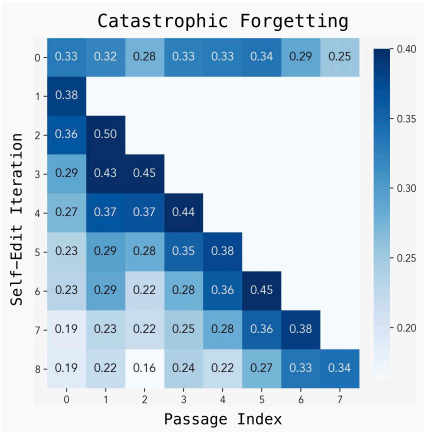

Figure 6: **Catastrophic forgetting from continual self-edits.** We sequentially update the model on new passages and track degradation on prior tasks. Entry-wise standard errors are reported in §B.6.

re-evaluate the model on all previously seen tasks to measure retention. This setup tests the model's ability to integrate new edits without forgetting earlier ones.

As shown in Figure 6, performance on earlier tasks gradually declines as the number of edits increases, suggesting that SEAL is still susceptible to catastrophic forgetting. Still, it can perform multiple updates without complete collapse, indicating possibility for improvement. Future work could enhance this ability through reward shaping [70, 71, 72] to penalize regressions on earlier tasks, or by integrating continual learning strategies such as null-space constrained edits [73] or representational superposition [74].

Computational overhead. The TTT reward loop is significantly more computationally expensive than other reinforcement learning loops used with LLMs. For instance, reward signals based on human preferences typically involve a single model forward pass, and those using verified solutions may rely on simple pattern matching (e.g., regex). In contrast, our approach requires finetuning and evaluating an entire model to compute the reward—each self-edit evaluation takes approximately 30–45 seconds, introducing substantial overhead (see §B.5).

Context-dependent evaluation. Our current instantiations assume that every context is paired with an explicit downstream task: few-shot demonstrations arrive with a held-out query pair, and each passage comes bundled with reference QA. This coupling simplifies reward computation but prevents RL training of SEAL from scaling to unlabeled corpora. A potential solution is to let the model generate not only self-edits but also its own evaluation questions—e.g., draft QA items or synthetic test cases for each passage—while the original content is still in context. These model-written queries could provide the immediate supervision required for reinforcement learning, broadening applicability to general training domains where external question-and-answer sets are unavailable.

6 Discussion and Conclusion

Villalobos et al. [75] project that frontier LLMs will be trained on all publicly available human-generated text by 2028. We argue that this impending "data wall" will necessitate the adoption of synthetic data augmentation. Once web-scale corpora is exhausted, progress will hinge on a model's capacity to *generate its own high-utility training signal*. A natural next step is to meta-train a dedicated SEAL synthetic-data generator model that produces fresh pretraining corpora, allowing future models to scale and achieve greater data efficiency without relying on additional human text.

We can imagine a future in which LLMs can ingest new data, such as academic papers, and generate large quantities of explanations and implications for themselves using their existing knowledge and reasoning with the in-context data. This iterative loop of self-expression and self-refinement could allow models to keep improving on rare or underrepresented topics even in the absence of additional external supervision.

In addition, while modern reasoning models are often trained with RL to generate chain-of-thought (CoT) traces, SEAL could offer a complementary mechanism, allowing the model to learn when and how to update its own weights. These two approaches could synergize: the model may choose to perform weight updates mid-reasoning to guide its current trajectory, or after completing reasoning to distill key insights into its parameters—improving future inference through internalized learning.

This continual refinement loop is also promising for building agentic systems—models that operate over extended interactions and adapt dynamically to evolving goals. Agentic models must incrementally acquire and retain knowledge as they act. Our approach supports such behavior by enabling structured self-modification: after an interaction, the agent could synthesize a self-edit which triggers a weight update. This could allow the agent to develop over time, aligning its behavior with prior experience and reducing reliance on repeated supervision.

SEAL demonstrates that large language models need not remain static after pretraining: by learning to generate their own synthetic self-edit data and to apply it through lightweight weight updates, they can autonomously incorporate new knowledge and adapt to novel tasks. Looking ahead, we envision extending the SEAL framework to pretraining, continual learning, and agentic models, ultimately enabling language models to self-learn and scale in a data-constrained world.

Acknowledgments and Disclosure of Funding

We would like to thank Shivam Duggal, Idan Shenfeld, Seungwook Han, Jeremy Bernstein, Akarsh Kumar, Linlu Qiu, Juno Kim, Brian Cheung, Moritz Reuss, Ayush Sekhari, Zhang-Wei Hong, Mehul Damani, Leshem Choshen, and Ryan Yang for their valuable discussions and feedback. We acknowledge support from ARO MURI grant number W911NF-23-1-0277. This research was also partly sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19- 2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein. This research was also partly supported by the Stevens Fund for MIT UROP research and by the MIT-IBM Watson AI Lab.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[2] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

[3] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[4] Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. OLMo: Accelerating the science of language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.841/.

[5] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[6] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. URL https://aclanthology.org/2020.acl-main.740/.