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
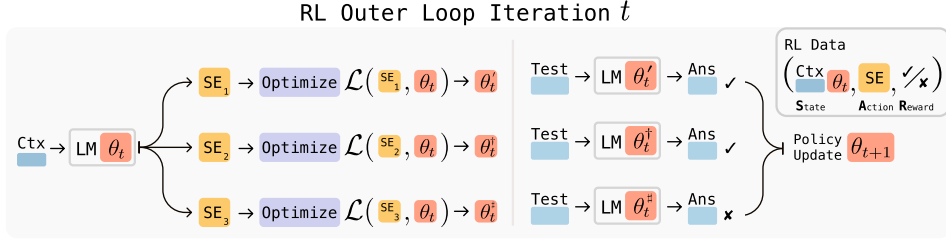
Preprint. Under review.

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 **L**LMs (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 LM_θ. 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 SE—the form of which varies by domain (see §3.2)—and updates its parameters via supervised finetuning: $\theta' \leftarrow \text{SFT}(\theta, \text{SE})$.

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

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

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

1: **Input:** LM_θ, dataset $\mathcal{D} = \{(C, \tau)\}$
2: **for** outer iteration $t = 1, 2, \dots$ **do**
3:     Sample $(C, \tau) \sim \mathcal{D}$
4:     Generate self-edit $\text{SE} \sim \text{LM}_\theta(\cdot \mid C)$
5:     Inner Loop Update: $\theta'_t \leftarrow \text{SFT}(\theta_t, \text{SE})$
6:     Evaluate: $\text{Ans} \sim \text{LM}_{\theta'_t}(\cdot \mid \tau)$
7:     Compute reward: $r \leftarrow r(\text{Ans}, \tau)$
8:     Update: $\theta_{t+1} \leftarrow \text{RL\_Update}(\theta_t, r, \text{SE})$
9: **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:

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

More precisely, in optimizing (1), we must compute the gradient $\nabla_{\theta_t}\mathcal{L}_{\text{RL}}$. However, as we noted, the reward term $r(\text{SE}, \tau, \theta_t)$ 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

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

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

where p_{θ_t} denotes the model's autoregressive distribution and $y_s^{(i,j)}$ is the sth token of self-edit SE_{ij}, the jth 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 $\mathtt{SE} = \{s_1, s_2, \ldots, s_n\}$ 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.
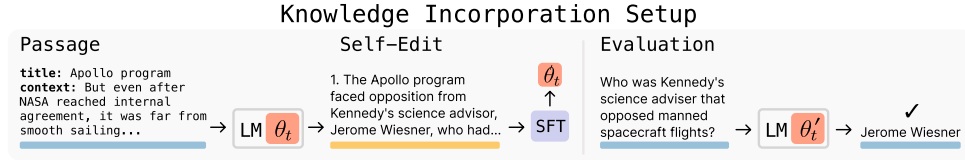
Figure 2: **Knowledge Incorporation Setup.** Given a new passage, the model generates synthetic data (the *self-edit*) in the form of "implications" of the passage. We then finetune on these outputs using LoRA. The updated model is evaluated on questions about the passage *without* access to the original 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 $\mathtt{LM}_{\theta'}$ 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.
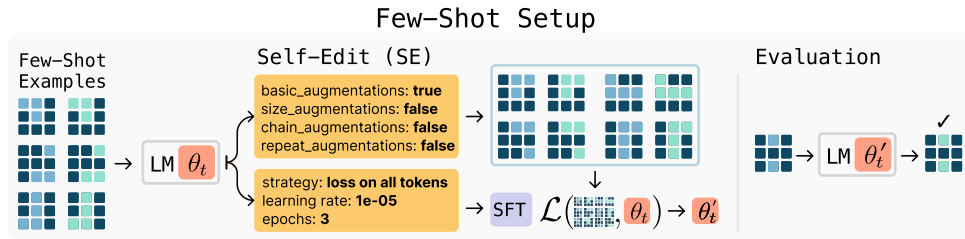

Figure 3: **Few-Shot Learning with SEAL.** Left: example ARC demonstrations. Center: the model generates a self-edit specifying augmentations and training hyperparameters. Right: the adapted model is evaluated on a held-out test input.

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.

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
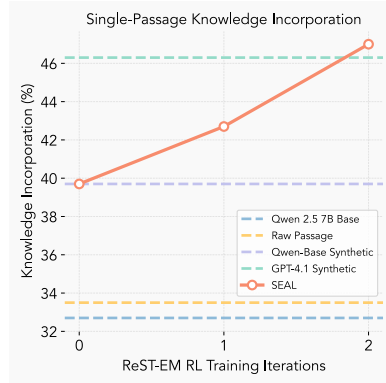

Figure 4: **Accuracy over RL iterations.** Each iteration consists of a mini-batch of 50 contexts, each with 5 sampled self-edits. SEAL surpasses GPT-4.1 synthetic data after two iterations of ReST^{EM} on the no-context SQuAD set.

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
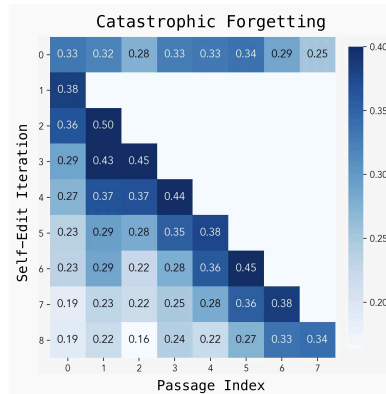

Figure 6: **Catastrophic forgetting from continual self-edits.** We sequentially update the model on new passages and track degradation on prior tasks. Entry-wise standard errors are reported in §B.6.

8

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

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/.

[7] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models, 2020. URL https://arxiv.org/abs/2012.00363.

[8] Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. ARC prize 2024: Technical report, 2025. URL https://arxiv.org/abs/2412.04604.

[9] Jason Wei, Maarten Bosma, Vincent 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 *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

[10] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code Llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

[11] Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. MediTron-70B: Scaling medical pretraining for large language models, 2023. URL https://arxiv.org/abs/2311.16079.

[12] Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, Andre F. T. Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, and Michael Desa. SaulLM-7B: A pioneering large language model for law, 2024. URL https://arxiv.org/abs/2403.03883.

[13] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2016. URL https://aclanthology.org/D16-1264/.

[14] François Chollet. On the measure of intelligence, 2019. URL https://arxiv.org/abs/1911.01547.

[15] Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent English?, 2023. URL https://arxiv.org/abs/2305.07759.

[16] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio Cesar Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Conti Kauffmann, Gustavo Henrique de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Behl, Xin Wang, Sebastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2024. URL https://openreview.net/forum?id=Fq8tKtjACC.

[17] Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. 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.757/.

[18] Ruixiang Tang, Xiaotian Han, Xiaoqian Jiang, and Xia Hu. Does synthetic data generation of LLMs help clinical text mining?, 2023. URL https://arxiv.org/abs/2303.04360.

[19] Saumya Gandhi, Ritu Gala, Vijay Viswanathan, Tongshuang Wu, and Graham Neubig. Better synthetic data by retrieving and transforming existing datasets. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics*. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-acl.385/.

[20] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.acl-long.754/.

[21] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with GPT-4, 2023. URL https://arxiv.org/abs/2304.03277.

[22] Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candes, and Tatsunori Hashimoto. Synthetic continued pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=07yvxWDSla.

[23] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0DcZxeWfOPt.

[24] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6f1d43d5a82a37e89b0665b33bf3a182-Paper-Conference.pdf.

[25] 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*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

[26] Asaf Yehudai, Boaz Carmeli, Yosi Mass, Ofir Arviv, Nathaniel Mills, Eyal Shnarch, and Leshem Choshen. Achieving human parity in content-grounded datasets generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RjYKTQ0L0W.

[27] Afra Feyza Akyürek, Ekin Akyürek, Leshem Choshen, Derry Wijaya, and Jacob Andreas. Deductive closure training of language models for coherence, accuracy, and updatability. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics*. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-acl.584/.

[28] Andrew K. Lampinen, Arslan Chaudhry, Stephanie C. Y. Chan, Cody Wild, Diane Wan, Alex Ku, Jörg Bornschein, Razvan Pascanu, Murray Shanahan, and James L. McClelland. On the generalization of language models from in-context learning and finetuning: a controlled study, 2025. URL https://arxiv.org/abs/2505.00661.

[29] Core Francisco Park, Zechen Zhang, and Hidenori Tanaka. *New News*: System-2 fine-tuning for robust integration of new knowledge, 2025. URL https://arxiv.org/abs/2505.01812.

[30] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. URL http://proceedings.mlr.press/v119/sun20b.html.

[31] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/bcdec1c2d60f94a93b6e36f937aa0530-Paper-Conference.pdf.

[32] Yu Sun, Xinhao Li, Karan Dalal, Chloe Hsu, Sanmi Koyejo, Carlos Guestrin, Xiaolong Wang, Tatsunori Hashimoto, and Xinlei Chen. Learning to (learn at test time), 2024. URL https://arxiv.org/abs/2310.13807.

[33] Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning, 2025. URL https://arxiv.org/abs/2411.07279.

[34] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

[35] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/639a9a172c044fbb64175b5fad42e9a5-Paper-Conference.pdf.

[36] Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura A Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=lNAyUngGFK.

[37] DeepSeek-AI. Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[38] Jürgen Schmidhuber. Evolutionary principles in self-referential learning, 1987. URL https://people.idsia.ch/~juergen/diploma1987ocr.pdf.

[39] Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning to learn using gradient descent. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *ICANN*. Springer Berlin Heidelberg, 2001. URL https://link.springer.com/chapter/10.1007/3-540-44668-0_13.

[40] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

[41] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning, 2016. URL https://arxiv.org/abs/1611.02779.

[42] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn, 2017. URL https://arxiv.org/abs/1611.05763.

[43] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. In *The Sixth International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyX0IeWAW.

[44] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/4de754248c196c85ee4fbdcee89179bd-Paper.pdf.

[45] Qi Sun, Edoardo Cetin, and Yujin Tang. Transformer-Squared: Self-adaptive LLMs, 2025. URL https://arxiv.org/abs/2501.06252.

[46] Jurgen Schmidhuber. Steps towards 'self-referential' neural learning: A thought experiment, 1992. URL https://people.idsia.ch/~juergen/selfref1992.pdf.

[47] Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. A modern self-referential weight matrix that learns to modify itself. In *International Conference on Machine Learning*. PMLR, 2022. URL https://proceedings.mlr.press/v162/irie22b.html.

[48] Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning, 2024. URL https://arxiv.org/abs/2311.04661.

[49] Nathan Hu, Eric Mitchell, Christopher Manning, and Chelsea Finn. Meta-learning online adaptation of language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.emnlp-main.268/.

[50] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI feedback, 2022. URL https://arxiv.org/abs/2212.08073.

[51] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML '24. JMLR.org, 2024.

[52] Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. Language model self-improvement by reinforcement learning contemplation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=38E4yUbrgr.

[53] Zhaoyang Wang, Weilei He, Zhiyuan Liang, Xuchao Zhang, Chetan Bansal, Ying Wei, Weitong Zhang, and Huaxiu Yao. CREAM: Consistency regularized self-rewarding language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Vf6RDObyEF.

[54] Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mtJSMcF3ek.

[55] Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.emnlp-main.67/.

[56] Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. Self-consistency preference optimization, 2024. URL https://arxiv.org/abs/2411.04109.

[57] Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WJaUkwci9o.

[58] Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

[59] Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train?, 2025. URL https://arxiv.org/abs/2505.21444.

[60] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[61] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[62] W. R. Gilks and P. Wild. Adaptive rejection sampling for gibbs sampling. *Journal of the Royal Statistical Society*, 1992. URL http://www.jstor.org/stable/2347565.

[63] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning? In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AP1MKT37rJ.

[64] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL https://arxiv.org/abs/2204.05862.

[65] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL https://arxiv.org/abs/2308.01825.

[66] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

[67] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[68] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem, 1989. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[69] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *The Second International Conference on Learning Representations*, 2014. URL https://openreview.net/forum?id=oXSw7laxwUpln.

[70] Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b710915795b9e9c02cf10d6d2bdb688c-Paper.pdf.

[71] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Reward shaping with language models for reinforcement learning, 2024. URL https://arxiv.org/abs/2309.11489.

[72] Jiayi Fu, Xuandong Zhao, Chengyuan Yao, Heng Wang, Qi Han, and Yanghua Xiao. Reward shaping to mitigate reward hacking in RLHF, 2025. URL https://arxiv.org/abs/2502.18770.

[73] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HvSytvg3Jh.

[74] Brian Cheung, Alexander Terekhov, Yubei Chen, Pulkit Agrawal, and Bruno Olshausen. Superposition of many models into one. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/4c7a167bb329bd92580a99ce422d6fa6-Paper.pdf.

[75] Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Will we run out of data? Limits of LLM scaling based on human-generated data, 2024. URL https://arxiv.org/abs/2211.04325.

[76] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. GPT-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[77] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20. Association for Computing Machinery, 2020. URL https://doi.org/10.1145/3394486.3406703.

[78] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. URL https://dl.acm.org/doi/10.1145/3600006.3613165.

A   Experimental Details: Few-shot Learning

A.1   Model and Setup

We use `Llama-3.2-1B-Instruct` [3] as the base language model. Since this model has no special-ized training on ARC, its ability to solve ARC tasks is limited. To enable controlled evaluation, we curated a small set of ARC problems from the training and evaluation splits that are solvable with optimal TTT hyperparameters.

Training Set: We selected 11 ARC tasks from the training set as the environment for RL optimization.
Evaluation Set: We selected 8 distinct ARC problems from the evaluation set for measuring generalization performance. These 8 were explicitly filtered for being amenable to TTT out of the full evaluation set.

These sets were chosen to isolate the effect of self-edit learning rather than general ARC ability.

A.2   Training Procedure

We trained SEAL using a single outer loop of reinforcement learning on the 11 training problems. For each problem, the model generated 15 self-edit configurations consisting of:

- **Data augmentation strategy:** Controls whether to include variations such as basic exam-ples, size-based augmentations, chained reasoning, and repeated inputs: `include_basic`, `include_size`, `include_chain`, `include_repeat`.

- **Optimization configuration:** Specifies hyperparameters such as learning rate, number of epochs, and loss function type (e.g., whether to compute loss over all tokens or only the final output tokens).

Each configuration was evaluated via test-time training (TTT), and assigned a binary reward: 1 if the adapted model produced the correct solution, 0 otherwise using Akyürek et al. [33]'s evaluation protocol. To encourage efficient adaptation, we discarded configurations requiring more than 375 training steps, retained only configurations with reward 1 and acceptable cost for LoRA-based SFT.

SFT for TTT was done with the following hyperparameters:

Parameter	Value
LoRA rank	128
LoRA alpha	16
Learning rate	N/A (model chooses it)
`num_train_epochs`	N/A (model chooses it)

LoRA adapters were inserted into the following modules: `q_proj`, `v_proj`, `gate_proj`, `down_proj`, `up_proj`

For the RL SFT we used the following hyperparameters:

Parameter	Value
`lora_rank`	16
`lora_alpha`	16
`num_train_epochs`	8
`per_device_train_batch_size`	5
`learning_rate`	5e-5

When generating the self-edits, we add the following prompt as a suffix to the few-shot examples.

You are configuring a model training pipeline by selecting from predefined tools.
You must make two decisions:
1. **Data Generation Tools** — For each of the following, choose true or false:
   - use_basic_augmentations
   - use_size_augmentations
   - use_chain_augmentations
   - use_repeat_augmentations

2. **Training Configuration** — Choose one of:
   - "train_using_all_tokens"
   - "train_using_output_tokens"

Also specify:
   - learning_rate (float)
   - num_train_epochs (integer)

Output Format

Respond with a valid JSON object. Do not include any explanation, markdown, or extra text.
Use lowercase `true`/`false` for booleans and ensure correct JSON syntax.
Example output:

```
{
  "data_generation": {
    "use_basic_augmentations": ...,
    "use_size_augmentations": ...,
    "use_chain_augmentations": ...,
    "use_repeat_augmentations": ...
  },
  "training": {
    "strategy": ...,
    "learning_rate": ...,
    "num_train_epochs": ...
  }
}
```

A.3 Evaluation Details

For each of the 8 held-out evaluation tasks, the model generated 5 self-edit configurations, yielding a total of 40 configurations. Success was measured as the percentage of configurations that led to correct outputs after adaptation. We followed the evaluation protocol from Akyürek et al. [33].

For the Oracle TTT we used the following configs:

Parameter	Value
lora_rank	128
lora_alpha	16
num_train_epochs	2
batch_size	2
learning_rate	1e-4

A.4 Compute Resources

We performed all training runs on a single A100, H100, or H200. Each TTT per problem requires between half a minute to few minutes, which is also why we limited the number of samples for ReST^{EM} and additionally limited the number of gradient steps allowed per self-edit TTT. Overall ReST^{EM} took around 2-3 hours.

B Experimental Details: Knowledge Incorporation

B.1 Model and Setup

We use the `Qwen-2.5-7B` base model [5] in the knowledge incorporation experiments. We repurpose the SQuAD dataset v1.1 [13] for the task of answering questions *without* the passage in-context. We use the training set for RL training and a 200-article subset of the evaluation set for evaluation. Within the training set and evaluation set, there are some overlapping topics of passages, but there is no overlap between these sets, so we can be sure that there is no data contamination of the test passages due to RL training.

B.2 RL Training Procedure

We run 2 rounds of ReST^{EM} training [36]. On each round, we take a batch of 50 context-questions-answers triples from the SQuAD training set. For each context, we sample 5 self-edit generations at temperature 1. We evaluate each self-edit over 3 random seeds, training on the sequences and then evaluating the updated model on the corresponding questions. We average each generation's results over 3 seeds and then keep the single best generation for each of the 50 contexts. Finally, to finish the round of ReST^{EM}, we perform supervised finetuning on the 50 resulting prompt-completion pairs.

Supervised finetuning here is done with batch size of 10, for 2 epochs, with learning rate 3e-4, using LoRA [67] with rank 64 and alpha 128, applied to all MLP and attention projection layers.

B.3 Synthetic Data Generation and Finetuning Details

In all models, we generate synthetic data by prompting to generate implications of the passage:

> Let's read the following passage and produce a list of implications derived directly or indirectly from the content.
>
> Passage:
> {passage}
>
> Implications:

We then take the resulting generated sequence. In the single-passage case, we split it by newlines into a set of training documents. In the multi-passage case, we use the full generated sequence as a single training document. In the case of synthetic data from GPT-4.1 (`gpt-4.1-2025-04-14`), an instruct-model, we additionally have the following rule: If the second line begins with a "1." then we omit the first line from the training set. This is because we found that the first line often contained filler text (e.g. "Sure, here is the list of implications:").

We then use the following training hyperparameters:

Table 3: Single-Passage Knowledge Incorporation Hyperparameters

Parameter	Search Space
LoRA Rank (r)	[**32**, 64]
LoRA Alpha (α)	[32, **64**]
Learning Rate	[1e-4, 3e-4, 5e-4, **1e-3**, 2e-3]
Epochs	[1, 5, **10**, 15, 20]
Batch Size	[**1**, 4]

In the multi-passage $n = 200$ case, we sample 5 self-edit completions for each passage and take the aggregate dataset of all self-edits across all passages to train on.

To answer the corresponding questions, we use the following prompt:

Table 4: Multi-Passage Knowledge Incorporation Hyperparameters

Parameter	Search Space
LoRA Rank (R)	[**32**, 64]
LoRA Alpha (α)	[32, **64**]
Learning Rate	[1e-4, 3e-4, 5e-4, **1e-3**, 2e-3]
Epochs	[1, **3**, 5]
Batch Size	[1, 4, **8**, 16]

> Let's answer a question directly and concisely.
> Question: {question}
> Answer:

B.4 Evaluation Details

We evaluate on a 200-passage subset of the SQuAD evaluation set, consisting of a combined 974 evaluation questions (roughly 5 corresponding to each passage). The pipeline of generating synthetic data and finetuning on it is the same as above. For automated grading, we use `gpt-4.1-2025-04-14` [76] via the OpenAI API with greedy decoding.

The grading prompt is as follows:

> You are a grading assistant. Your job is to determine whether a student's answer correctly answers the question based solely on the provided gold answer. Do not use any outside knowledge. The student answer can include additional information, but it must at least fully convey the gold answer and must not contradict it. Ignore style, phrasing, or extra details that do not affect correctness. Respond ONLY with 'yes' or 'no'.
>
> Question: {question}
> Gold answer: {gold}
> Student answer: {pred}
> Is the student answer correct based solely on the gold answer? Respond 'yes' or 'no'.

B.5 Compute Resources

All experiments are performed on 2×H100 or 2×H200. We use DeepSpeed ZeRO-3 [77] for SFT in ReST^{EM} training. We use vLLM [78] for efficient inference. The most compute-intensive portion of our training and evaluation is the E-step of ReST^{EM} training, where the model generates completions and is graded through the inner-loop process of finetuning and running inference. Doing a single round requires a batch of 50 passages over 5 completions and 3 runs per completion, meaning 750 inner loop iterations. This takes about 6 hours on 2×H100s.

B.6 Standard Error of the Mean in Catastrophic Forgetting Experiment

The standard errors of the mean (SEM) for each entry in Figure 6 is shown below in Table B.6.

C Prompting

Recent works have shown that reinforcement learning baselines and outcomes can be highly sensitive to prompting. We experiment with 4 additional self-edit prompts in the knowledge-incorporation setting. The five prompts—`implications`, `implications-long`, `implications-very-long`, `rewrite`, and `self-qa`—are shown below. All results in the main content of the paper used the `implications` prompt, which we consider to be the most prototypical [27, 28]. However, prior work has found prompts involving rewriting or generating question-answer pairs can be more effective, as discussed in §2.

Table 5: Entrywise standard errors of the mean (SEM) across continual self-edits experiment.

	1	2	3	4	5	6	7	8
0	0.0306	0.0315	0.0263	0.0318	0.0297	0.0370	0.0310	0.0284
1	0.0273	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0305	0.0277	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0277	0.0358	0.0406	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.0272	0.0303	0.0337	0.0320	0.0000	0.0000	0.0000	0.0000
5	0.0296	0.0342	0.0290	0.0298	0.0319	0.0000	0.0000	0.0000
6	0.0289	0.0334	0.0271	0.0258	0.0320	0.0337	0.0000	0.0000
7	0.0255	0.0313	0.0264	0.0253	0.0309	0.0331	0.0363	0.0000
8	0.0237	0.0307	0.0211	0.0267	0.0273	0.0271	0.0358	0.0263

Furthermore, as we see qualitatively in Figure 5, RL appears to have dramatically increased the length of the response of the example. We therefore experiment with prompting for longer generations with `implications-long` and `implications-very-long` to test if we can achieve similar gains through prompting alone.

The results are shown in Table 6. Notably, the baselines for `implications-long` and `rewrite` the RL-trained version of `implications`. However, using these prompts as the base of RL training yields even greater improvements. In all cases, ReST^{EM} enhanced performance by roughly 6 to 11 percentage points.

	No self-edit	Implications	Implications-long	Implications-very-long	Rewrite	Self-QA
Original	33.5	39.7	49.3	45.0	49.4	37.3
Round 1	–	43.7	52.4	51.5	55.3	42.8
Round 2	–	**47.0**	51.8	52.1	55.6	48.7

Table 6: Performance across 2 rounds of ReST^{EM} RL training on various prompts in the single-document knowledge incorporation setting.

The five prompts are shown below.

```
implications
```
Let's read the following passage and produce a list of implications derived directly or indirectly from the content.

Passage:
{passage}

Implications:

```
implications-long
```
Let's read the following passage and produce a long list of implications derived directly or indirectly from the content.

Passage:
{passage}

Implications:

`implications-very-long`

Let's read the following passage and produce a very long list of implications derived directly or indirectly from the content.

Passage:
{passage}

Implications:

`rewrite`

Let's read the following passage and rewrite it in a few different ways, each one separated by a newline.

Passage:
{passage}

Rewritten passages:

`self-qa`

Let's read the following passage and rewrite it in a question-answer format.

Passage:
{passage}

Question 1:

Note: For `self-qa`, we apply additional formatting so that training documents consist of question–answer pairs, rather than using our standard approach of splitting by newline characters. Specifically, we split the output using occurrences of "Question n:" instead of newlines.