SAMKE: An Open-Ended Autonomous Foundation-Model-Based Agent for Meta-Knowledge Discovery and Learning

Nitin Vetcha[1,2], **Dianbo Liu**[1]

[1]National University of Singapore
[2]Indian Institute of Science, Bangalore
nitinvetcha@iisc.ac.in, dianbo@nus.edu.sg

Abstract

Given the recent tremendous success of large language mod els (LLMs), there has been an increasing trend in applying them to several down-stream tasks via fine-tuning (FT). How ever, there persist two significant challenges in FT- 1) cura tion of high-quality task-specific data and 2) expensive time consuming model adaptation via gradient descent optimiza tion. To mitigate these limitations, we leverage prior works in large-scale parameter generation for LLMs and propose Scientific Autonomous Meta-Knowledge Explorer (SAMKE) to open up a new paradigm of parameter-level meta learn ing, thereby serving a critical advance in the AI Scientist domain. SAMKE is an open-ended autonomous foundation model-based agent which is capable of self-improvement by discovering and learning the rich meta-knowledge informa tion present in large neural network weights, thereby enabling efficient adaptation of LLMs by parameter-level weight mod ifications to unseen domains as well. To achieve this, SAMKE has a multi-level reinforcement learning approach to train models for efficient understanding of parameter space and of fers higher degree of flexibility compared to prior works on self-evolution by providing freedom to the model for choos ing its own adaptation strategy thereby breaking the scal ing solely through data regime. Early experiments demon strate the superior performance of SAMKE in the common sense reasoning domain as it outperforms task-specific FT by 23.6% on average and even some of the most recent works in parameter generation (by 10.4%), model merging (by 24.3and test-time learning (by 25.2%) as well out-of-domain tasks such as coding, social, logical and mathematical reasoning as well.

1. Introduction

Large Language Models (LLMs) due to their impressive scalable transformer architecture and massive internet-data level pretraining process, is often endowed with impressive emergent abilities and zero-shot generalization, which in itself is a remarkable advancement in several natural language processing (NLP) tasks such as text generation, question-answering, creative writing, summarization and knowledge engineering. Despite this progress, there are several down-stream applications which demand behavioral adaptation particular to the specific task under consideration such as reflects domain jargon or bespoke response styles. To enable this, there is a class of techniques called Parameter-Efficient Fine-Tuning (PEFT), which work in general by keeping the original weights frozen and introducing an additional set of trainable parameters that is usually minimal compared to the base model. They thus, can only reduce, but not erase the cost of per-task-tuning, which becomes an important bottleneck for large-scale deployment in addition to the curation of high-quality task-specific data which reflects the internal specifications required.

Moreover, widely adopted downstream task-specific fine-tuning strategies such as LoRA (Hu et al. 20222), DoRA (Liu et al. 2024), ReFT (Wu et al. 2024) are generic hand-crafted solutions and might not be optimal approaches for a particular class of LLMs or set of tasks. While an option for developing a good set of directives for updating the LLM is to take assistance of domain experts, it is quite time and compute intensive as well as susceptible to human deficiencies since the exploratory space for parameters is high-dimensional and quite vast. Keeping these points in mind, we aim at developing through our project, an open-ended approach for neural network weight modification using the expressive power of foundation models. Such an approach holds a promising potential for adapting LLMs on the fly to unseen tasks by effectively answering the following research question.

RQ: Can LLMs learn to modify their internal representation space autonomously to handle concept drift, analogous to how humans assimilate and restructure knowledge in lifelong learning scenarios?

In short, the primary contribution of this work is the development of SAMKE or *Scientific Autonomous Meta-Knowledge Explorer*, a first-of-its-kind open-ended autonomous foundation model-based agent empowering LLMs to self-improve by discovering and learning meta-knowledge encoded in neural network weights, thereby allowing efficient adaptation to unseen tasks via parameter-level weight modifications. Rest of the paper is organized as follow. In Section 2, we highlight the motivation for our approach in detail. Section 3 presents the literature survey conducted, Section

4 contains the methodology with implementation specifics in Section 5. Experimental results are provided in Section 6 and in Section 7, we present our concluding remarks.

2. Motivation

Our primary motivation stems from human psychology and pedagogy. For example, consider a human student who is preparing to take an end-sem examination of a machine learning course. Quite often, students tend to rely on their prior prepared notes for preparation. These notes are often derived from the lecture content, textbooks or information available on the internet. Thus instead of relying on the raw content, students assimilate and rewrite the information in the form of notes as per their own intrinsic reasoning skill and aptitude. This improves the capability of students to comprehend the content better and therefore respond well to the exam questions. This phenomenon of reinterpreting and augmenting external knowledge in a way that is easier to understand as well as developing the necessary skill-sets is not limited to just taking exams, but seems to be universally true of human learning across tasks. Furthermore, depending on one's interests, 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 restructuring or development of internal knowledge as well as assimilation or rewriting of external information, as part of the learning process is in contrast with how LLMs undergo currently training and adaptation. Given a new task, current LLMs consume and learn from the task data "as-is" via finetuning or in-context learning. The issue with this, just like in the human setting, such data may not be in an optimal format (or volume) for learning, or there might not be the relevant skill-set developed to learn it and current approaches do not enable models to develop bespoke strategies for how to best transform themselves internally or even learn from their training data. In this work, we therefore investigate the question as to if it is possible for even LLMs, analogous to humans, to suggest strategies by themselves which can enable them to perform better on a given task.

A secondary source of motivation as to why we ground our strategy search space in the neural network weights is because unlike task-specific knowledge, the weight-level meta-knowledge represents generalized principles about how neural network parameters relate to model capabilities, thereby providing crucial insights for self-evolving agents. There are several prior research works which have shown that there exists a positive correlation between types of neural network weight patterns and downstream model performance characteristics. For example, scaling laws research (Kaplan et al. 2020) has demonstrated that there are predictable relationships between model size and performance. Similarly, structured sparsity learning gives an indication so as to how particular weight patterns can be useful for developing more efficient representations (Wen et al. 2016).

3. Related Work

Test-Time Training (TTT) is a recently emerging class of approaches which updates model weights at inference time using techniques such as input perplexity or cross-entropy minimization on only unlabeled test data enabling self supervised enhancement of LLM performance (Hu et al. 2025a,b) or via reinforcement learning by utilizing the priors in the pre-trained models (Zuo et al. 2025) or by using reflection and verifier-driven sample selection (Moradi et al. 2025; Lee et al. 2025) or by using a task-specific curriculum (Hübotter et al. 2025) or by using a mixture-of-expert based model merging (Bertolissi et al. 2025). An alternative approach is to scale inference compute at test time as well using for example ensemble approaches such as majority voting. While test-time approaches is a promising option, such a computational overhead might not be necessary always and it often fails in cases where data is scarce or quality of unlabeled data is poor.

Adversarial Fine Tuning is another emerging class of techniques where in two LLM instances are made to debate with each other about a topic or one instance serves as a challenger or teacher and the other instance serves as a solver or student to generate synthetic data, either from unlabeled prompts or even from scratch itself and use approaches like majority voting to create pseudo-labels which can further be used for updating model's knowledge accordingly (Yang et al. 2024; Wang et al. 2025c,b). This can also be done by some additional fine-tuning using information which is available in the LLM's context as well (Park, Zhang, and Tanaka 2025) similar to knowledge distillation. Recent works include SQLM (Chen et al. 2025a), R-Zero (Huang et al. 2025a), TT-SI (Acikgoz et al. 2025), SIRLC (Pang et al. 2023). While this is an efficient approach in data scarce domains where TTT fails, it is not always efficient as there are certain challenging domains which require mastering novel reasoning skills and it is well known that scaling data isn't sufficient in this regimes such as mathematics (Hendrycks et al. 2021a).

Reinforcement Learning (RL) is a well established approach for pushing the capabilities of LLMs and recent works such as SEAL (Zweiger et al. 2025), RLAIF (Li et al. 2025b), SRLM (Yuan et al. 2025) and Memento (Zhou et al. 2025), which uses a memory-based online RL policy have shown promising potential in the low-cost continual adaptation of LLMs. In RL, **meta-learning** has been used as well in order train agents in scenarios where it needs to learn novel tasks quickly (Gupta et al. 2018). SAMKE can be seen as thus following meta-learning principles since it learns an adaptation strategy i.e., how to generate effective self weight update using a meta optimization loop. Closely, related are **self-referential** systems as well which learn to update their own parameters as in (Irie et al. 2022) and **self-evolving** agents which enable LLM to improvise by autonomously acquiring, refining and learning from experiences generated by the model itself (Tao et al. 2024; ang Gao et al. 2025). While RL based approaches are quite good, its often chal-

lenging to achieve convergence and design optimal policies which are efficient in terms of compute and time as well.

Parameter Generation is another research direction which has seen several pioneering works such as RPG (Wang et al. 2025a), DnD (Liang et al. 2025), T2L (Charakorn et al. 2025), ORAL (Khan et al. 2025), COND P-DIFF (Jin et al. 2024). DnD generates task-specific parameters from unlabeled prompts without per-task training via a prompt-conditioned hyper-convolutional decoder while T2L does the same but uses a hyper-network and task description instead. ORAL leverages architectural and textual conditioning for flexible, scalable LoRA parameter adaptation. RPG introduces a recurrent diffusion architecture for scalable unconditional LoRA parameter generation. COND P-DIFF applies conditional latent diffusion for controllable LoRA parameter synthesis with strong cross-domain generalization. An associated direction is *model merging* as well, which facilitates generalization to unseen tasks via **multi-task learning** (Shao et al. 2025a,b). While these works have been effective, the limitation is that these are static parameters which once generated do not undergo any further modification but this feature is crucial for domains requiring the implicit meta-knowledge.

4. Methodology

In this section, we describe the framework of our proposed approach (see Figure 1). SAMKE starts by treating the LLMs own weights as environment variables to explore, upon which it would systematically propose scientific hypotheses to modify the internal representation space appropriately so as to adapt the LLM to the unseen task. A major challenge for the design, therefore, is the high dimensionality and non-convexity of the LLM weight space itself which makes the initialization and subsequent exploration process extremely complex. To overcome this, we work only with low-rank parameters (Hu et al. 20222) which constitutes a much smaller fraction ($\sim 1\%$) of the original model's weights. In addition, to avoid the limitations arising from selecting a single starting point, which might not be optimal to wiggle around, we prefer to sample from a plausible weight distribution space. This step is essential to eliminate the risk of non-convergence. To get this initial distribution for weights i.e., self-weight sampling, we refer to prior works in large-scale LLM parameter generation and use a convolution-based decoder architecture as the backbone for SAMKE's exploration point initializer.

Once the weights have been initialized[1] for exploration, SAMKE then uses a foundation-model-based agent, which is for now simply an LLM trained using reinforcement learning (RL) to come up with probable hypotheses at inference time for weight-space exploration using test-time scaling and compute. To however, facilitate the training process, it is necessary to first curate by hand a seed knowledge base, consisting of either proven or plausible weight modification strategies, which will then serve as the action space for LLM's initial stages of exploration during RL training. This would be a multi-stage recipe consisting of three distinct progressively harder levels. Level I consists of training the LLM to produce only single valid and efficient self-edits (a self-edit as the name suggests is basically a modification strategy proposed by an LLM to update its own weights depending on the task) from among the ones present in the initial knowledge base. Level II comprises of training the model to output chain-of-self-edits, since coupling strategies sequentially is also helpful (moreover if viewed in a abstract sense, it can be considered in effect as a single complex edit which can be decomposed into simpler instances). Level III is a significantly challenging aspect both for the LLMs as well as from implementation perspective as well, which is basically letting LLMs to explore the hypothesis space in its entirety, thereby going beyond human-crafted approaches. A positive performance in Level III would be a significant leap as it could possibly open up new frontiers in training and fine-tuning paradigms as has been similarly done in other areas as well such as neural architecture search (Liu et al. 2025b) and optimization (Lu et al. 2024).

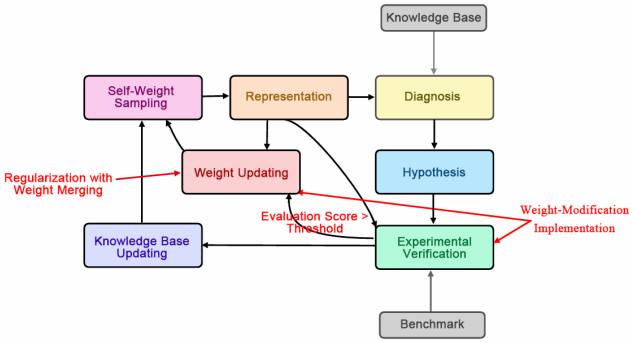

Figure 1: SAMKE's methodology of weight-level meta-knowledge discovery and modification summarized (*adapted from (Zhang 2025)*)

After plausible hypothesis have been generated by the foundation model-based agent and implemented, its necessary to test the hypothesis. For this purpose, we create a separate evaluation split if available. However, since SAMKE is designed to adapt LLMs efficiently to unseen tasks as well, the dataset for evaluation itself would be generated on the fly using adversarial approaches involving multiple instances of an LLM, one proposing and one solving questions on a particular topic as in SQLM (Chen et al. 2025a) or R-Zero (Huang et al. 2025b). Once the hypothesis has been tested and is found to be valid (as in it improves performance in some pre-determined metric such as accuracy on the `eval` set), it would be then added back into the knowledge base, thereby enriching the action space of LLM for future iterations. In order to prevent catastrophic forgetting, SAMKE implements a meta-level weight regulariza-

[1]These weights can optionally be encoded into a structured representation correlated with network performance like world models such as JEPA (LeCun 2022).

tion technique as well. Therefore, by automating the process of self-improvement using principled methodologies and meta-knowledge in a scientific manner (i.e., propose, validate and accept hypotheses), SAMKE provides a holistic framework towards the next generation of **AI generating AI** agents, because as soon as web-scale data corpora is exhausted, progress will hinge on a model's capacity to generate its own high-utility training signal.

5. Implementation

Architecture

Primary architectural detail in SAMKE's framework is the design of the weight-space exploration initializer. As mentioned in Section , we use a convolution based decoder model for this purpose. We assume that we have access to either the unseen task's description or atleast a handful of unlabeled examples representative of its requirements. We then send them to an open-sourced text encoder for embedding extraction. This extraction process can be formally represented as, $c_i = \text{Encoder}(p_i, \theta)$, where $\text{Encoder}(\cdot, \cdot)$ denotes the embedding extraction function parameterized by θ, and c_i represents the extracted embedding corresponding to prompt p_i. We use an encoder-based language model architecture for this purpose i.e., Sentence-BERT (all-MiniLM-L6-v2 specifically) (Reimers and Gurevych 2019) [2].

Next, following (Wang et al. 2025a), is the parameter tokenization process (see Figure 2), which is done so as to preserve both the layer-wise distribution and the cross-layer correlations. Specifically, (i) weights are split according to their layer indices, (ii) layer-wise normalization is applied to mitigate distribution shifts, (iii) parameters are sliced into non-overlapping tokens with uniform size, and (iv) a lightweight permutation state (encoded as a one-hot vector) is used to alleviate symmetry issues (Kunin et al. 2020) when collecting multiple checkpoints. Additionally, 2D position embeddings (first dimension encodes layer index, while second dimension captures the token's in-layer position). (Dosovitskiy et al. 2020) are employed to ensure the network retains positional awareness of each token within the entire set. In our case, each LoRA matrix is of shape 8×896, which is then split into 7 smaller chunks, each with a shape of 8×128, which is then finally padded to a uniform size of 10×130.

Say, the dimension of prompt embeddings is $[B, N, L, C]$ where B, N, L and C denote batch size, length of prompt batch (i.e., number of prompts), sequence length, and hidden dimension, respectively. The decoder (see Figure 3) consists of multiple sequential layers, each performing 5 2D convolutions. These convolutions are divided into three categories: i) **width convolution** that operates on (C, L) dimension, ii) **height convolution** that operates on (L, N) dimension) iii)

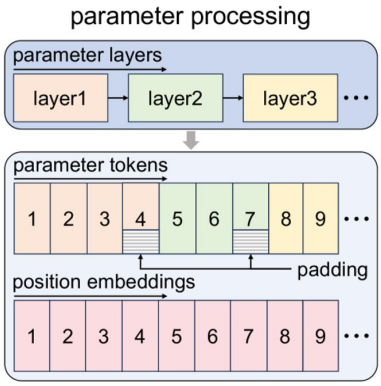

Figure 2: Parameter tokenization (*adapted from* (Wang et al. 2025a))

layer-wise convolution that on (N, L) dimension) , with notations Conv_W, Conv_H, and Conv_L. Each layer consists of two Conv_W, two Conv_H and one Conv_L. Given this, the forward operation of the decoder block is,

$$c_W^l = \text{Conv}_H^1(\text{Conv}_W^1(c^{l-1}))$$
$$c_H^l = \text{Conv}_W^2\left(\text{Conv}_H^2(c^{l-1})\right)$$
$$c^l = \text{Conv}_L\left((c_W^l + c_H^l + b)/3\right)$$

where c^l is hidden state output by the l th layer, c^0 is prompt embedding encoded by the condition extractor, and b is learnable bias. Through this process, input is transformed from dimension $[B, N, L, C]$ to $[B, N', L', C']$ which is then compatible to be converted into a flattened LoRA adapter for the LLM [3]. In this work, the base LLM used is Qwen2.5-0.5B-Instruct (Qwen et al. 2025) and LoRA is applied to the linear projection layers within both the self-attention mechanism and the MLP blocks of the transformer architecture. Specifically, this includes the query, key, value and output projections in attention blocks, as well as the gate, up and down projections in MLP blocks.

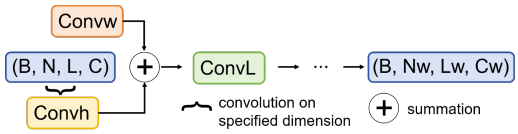

Figure 3: Convolutional Decoder (*adapter from* (Liang et al. 2025))

Training

In this work, we focus on the domain of common-sense reasoning and select 4 datasets for evaluation, namely HellaSwag (Zellers et al. 2019), BoolQ (Clark et al. 2019) as well as the challenge and easy set of AI2 Reasoning

[2]It is to be noted that BERT's supported sequence length is only 512 and for longer sequences, padding should be done. However, in our use case, maximum sequence length is only 384 and thus padding is not necessary.

[3]In our present implementation, the entire flow is (128,384,384) → (128,200,300) → (128,100,256) → (256,50,200) → (512,50,200) → (1024,25,200) → (1024,10,200) → (2048,10,200) → (4296,8,128)

Challenge (ARC) (Clark et al. 2018). ARC dataset contains grade-school level, multiple-choice science questions. HellaSwag instructs models to select from choices that best finish the sentence among ground truth and an adversarial set of machine-generated wrong answers. BoolQ is a question answering dataset for yes/no questions containing various factual problems. We use existing checkpoints of these datasets [4] (batch size was 32 and number of samples was 5000) which have been collected by first pretraining on the target dataset for 75 steps with a learning rate of 1e-4 and then performing fine-tuning on the target dataset for 50 additional steps with a learning rate of 1e-5, while saving a checkpoint at each step.

Subsequently, prompt-checkpoint pairing is done as follows. Given a dataset P, it is first divided it into non-overlapping prompt batches $[p_1, \cdots, p_i, \cdots, p_I]$. Denote the trained LLM checkpoints of this dataset as $M = [m_1, \cdots, m_j, \cdots, m_J]$. Then randomly a batch of prompts and a corresponding checkpoint is picked to create a pair $\{p_i, m_j\}$, which then serves as an input-output data point for training the decoder. The objective function for training is the mean squared error (MSE) loss between the output from the decoder's last block for a particular prompt batch and the training checkpoint associated with it.

Next crucial step is the hand-crafting of seed knowledge base. To this end, we identify five primary families of strategies[5], each containing its own sub-strategies as well, namely

- Test-Time Training (TTT) using input perplexity minimization **(Hu et al. 2025a)** or via reinforcement learning (Zuo et al. 2025) for example by using self-reflection and verification loops like GEPA (Agrawal et al. 2025), ReflectEvo (Li et al. 2025a), REVISE (Lee et al. 2025) or Instruct-of-Reflection (Liu et al. 2025a). It could also involve prompt optimization using frameworks like TextGrad (Yuksekgonul et al. 2024) or CAST (Tang et al. 2025)
- Post-training data-free LoRA modifications such as mixing LoRA subspaces obtained by weight decomposition of constituent matrices **(Wu et al. 2025)** or bounding norm of selected parameters (Wang, Dvijotham, and Manchester 2025) or evening merging multiple task-specific LoRA adapters (Zhao et al. 2024)
- SQLM (Chen et al. 2025b), R-Zero **(Huang et al. 2025a)** or SEAL (Zweiger et al. 2025) like reinforcement learning based frameworks which enable LLMs to self-adapt by generating their own finetuning data and update directives (another example is TT-SI (Acikgoz et al. 2025))
- Test-Time Scaling (TTS) using either a router or an ensemble approach i.e., we generate and perform inference with multiple adapters obtained by using different representative prompt batches and to obtain the final prediction, select either the most confident prediction (max_confidence) or by a majority vote or sum_logprobs (i.e., sum log probabilities across adapters per prediction and pick the one with highest total logprob)
- Latent Space (LS) Approaches which aim at working or modifying the internal layers (Hu et al. 2025b) or hidden activations (Zhang et al. 2025) directly of the LLM. It may also involve decoding algorithms which modify the sampling procedure itself (Karan and Du 2025; Wang et al. 2025d). We consider them as part of latent space family because they tamper with internal probability distribution of next-tokens unlike other families which modify the parameters explicitly.

We first formulate the objective for outer-loop RL training which generates adaptation strategies $\texttt{AS}$, as in (Zweiger et al. 2025). Let θ denote the parameters of the language model $\texttt{LM}_\theta$. In order to adapt to an unseen dataset (task) $\mathcal{D}$, SAMKE requires as specified in Section , C which is a context containing information relevant to the task and τ which is the evaluation strategy and metric used to assess the model's downstream adaptation. Based on C, SAMKE generates an $\texttt{AS}$ and updates its parameters accordingly $\theta' \leftarrow \texttt{Update}(\theta, \texttt{AS})$. We thus have an RL setup i.e., the model takes an *action* (generating $\texttt{AS}$), receives a *reward* r based on $\texttt{LM}_{\theta'}$'s performance on τ and updates its policy to maximize expected reward,

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

It is to be noted that the reward assigned to a given action depends on the model parameters θ at the time the action is taken (since θ is updated to θ', which is then evaluated). An implication of this is that the while modeling the RL state, one must therefore include θ in the policy's parameters as well along with C, even though the policy's observation is limited to C (because it is extremely infeasible to directly place θ in the LLM's context window). Therefore, the (state, action, reward) triples which have been collected by using an older model weights, θ_{old}, will not be aligned for the current model θ_{current}. Hence, an on-policy approach should be adapted, by which adaptation strategies are sampled from and, even more importantly, the rewards itself will be calculated using the current model.

In particular, the specific on-policy approach used is ReST^{EM} (Singh et al. 2024) where samples are first generated[6] from the current model and are filtered by using

[4]For training however, even Open-Book Question Answering or OBQA (Mihaylov et al. 2018), Physical Interaction: Question Answering or PIQA (Bisk et al. 2019) and WinoGrande (Sakaguchi et al. 2019) have been used as well. OBQA aims to promote research in advanced question-answering with salient facts summarized as an open book. PIQA focuses on everyday situations with a preference for a typical solutions. WinoGrande features a fill-in-a-blank task with binary options for commonsense reasoning questions.

[5]Unfortunately, there are no research works highlighting approaches for optimizing the performance of LoRA's obtained via the process of parameter generation, thereby posing a major challenge in identification of plausible strategies, which had to be cherry-picked via trial and error.

[6]Currently, only a deterministic number of samples are being

binary feedback [$r(\texttt{AS}, \tau, \theta_t)$ is 1 if on τ, AS improves LM_{θ_t}'s performance and is 0 otherwise]. The model is then fine-tuned on these samples and this continues in an iterative manner (See Algorithm 1).

A subtle detail, which hasn't yet been covered is the exact nature of the adaptation strategy itself. This depends on the particular strategy family being used, however the format is consistent across all which is basically a JSON object specifying the particular configurations to be used[7]. It contains a field, `family` which takes values `TTT`, `LoRA` and `TTS`. Currently, the following choices have been experimented

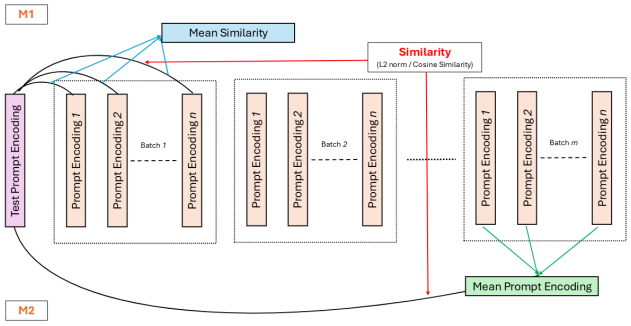

Figure 4: Router Approach for TTS

- For `TTT`, we use (Hu et al. 2025a) and the corresponding JSON object has fields `ttl_steps` (number of training steps in the TTL loop), `learning_rate`, `batch_size` and `shuffle_data` (boolean variable).
- For `LoRA` modifications, we use two-subspace (TS) mixing version from (Wu et al. 2025) and the corresponding JSON object has only a single field, namely `lambda` which is a hyperparameter determining the ratio in which the two resulting subspaces must be mixed.
- For `TTS`, we use either an ensemble or router approach. In the router approach (see Figure 4), we basically sample multiple prompt batches and choose that batch whose average of similarity scores[8] of individual prompts (M1) or averaged prompt embedding (M2), is closest to that of the question at test time. The corresponding JSON object has fields `num_prompt_batches` (indicating the number of prompt batches to be sampled from the test split of unseen dataset) and `method` which can take one of five values - `avg_sim_score`, `avg_prompt_embed`, `max_confidence`, `majority_vote` or (summing log probabilities) i.e., `sum_logprobs` (former two belong to router approach and the latter three constitute the ensemble approach).
- For `LS`, we use (Hu et al. 2025b) and the corresponding JSON object has fields `times` and `learning_rate`.

6. Experiments

Setup

As described in Section , the base LLM used is Qwen2.5-0.5B-Instruct, domain is common-sense-reasoning and evaluation datasets are ARC-c, BoolQ, HellaSwag and ARC-e. Baselines used include quite recent works such as DnD (Liang et al. 2025), Test-Time Learning (TTL) (Hu et al. 2025a), Decoupled and Orthogonal Merging (DOM)[9] (Zheng et al. 2025) and average of task-specific training LoRA's (Hu et al. 20222). On one extreme, TTL uses the entire unlabeled corpus of the training LoRA's in addition to the 128 unlabeled examples from the target dataset as seen by SAMKE. On the other extreme, instead of using the unlabeled corpus, DOM merges all the 7 training LoRA's inclusive of the target set.

Hardware

All experiments were conducted on a high-performance computing node running Ubuntu 22.04.1. The backend processor was EPYC 8434P, which had 48 physical cores (96 logical threads), 256 GB of system RAM and a maximum clock speed of 2.5 GHz. Four NVIDIA RTX A6000 GPUs, each with 48 GB of dedicated VRAM were utilized. Python version used was 3.12.11 and GPU-accelerated tasks were managed using CUDA version 12.4.

Results

The major results of this work are presented in Table 1 wherein we conduct experiments of 5 benchmarks which are in the domain of common-sense reasoning and also on 5 out-of-domain benchmarks namely GSM-MC and MATH-MC [10] to evaluate mathematical reasoning, DivLogicEval

generated, 15 to be precise. This could however be improvised to be dynamic in future version of the work wherein samples would continue to be generated until a particular confidence threshold, as determined by the model itself is reached instead. The same is true for number of iterations as well which is just 2 for now.

[7]Since the model being used is Qwen2.5-0.5B-Instruct, it was facing difficulty in following instructions given in the prompt for generation of structured outputs even after temperature alteration. In such cases, verification and formatting was done by using Qwen2.5-7B-Instruct instead.

[8]Cosine similarity and Euclidean distance were tested and the latter was found to perform better empirically. Thus, `avg_sim_score` and `avg_prompt_embed`. use euclidean distance by default. Alternatively, measure of similarity can also be included as a new field but hasn't been explored in the current work.

[9]DOM is a data-free framework for LoRA merging. It separates parameters into magnitude and direction components and merges them independently, thereby reducing the impact of magnitude differences on the directional alignment of the merged models, thus helping in preserving task information. It also uses a data-free, layer-wise gradient descent method with orthogonal constraints to mitigate interference during the merging of direction components. For evaluation on a target dataset, LoRA's of remaining datasets are merged and used.

[10]GSM-MC and MATH-MC are multiple choice versions of the standard GSM-8K (Cobbe et al. 2021) and MATH (Hendrycks

Algorithm 1: Sequential Multi-Level RL Loop for Adaptation Strategy (AS) Generation of SAMKE

Input: Base LM_θ, dataset context C, evaluation metric τ, initial knowledge base K
Init: Low-rank adapter generator G, sampled adapters $S \leftarrow \texttt{Sampler}(C, G)$

Level I (Single-edit self-training):
for iteration $t = 1, \ldots, T_1$ **do**
  Propose single-edit AS from K : $\texttt{AS} \sim \text{LM}_\theta(K, C)$
  Apply AS and obtain weights : $\theta' \leftarrow \texttt{ApplyStrategy}(\theta, \texttt{AS}, S)$
  Evaluate : $\texttt{Ans} \sim \text{LM}_{\theta'}(\cdot \mid \tau)$
  Compute reward : $r \leftarrow r(\texttt{Ans}, \tau)$
  if $r >$ threshold$_1$ **then**
    $\theta \leftarrow \texttt{RL\_Update}(\theta, r, \texttt{AS})$
  end if
end for

Level II (Chained/compositional strategies):
for iteration $t = 1, \ldots, T_2$ **do**
  Propose chain of edits : $\texttt{AS} = [e_1, \ldots, e_k],\ e_i \in K$
  Sequentially apply chain : $\theta_0 \leftarrow \theta$; $\theta_i \leftarrow \texttt{ApplyStrategy}(\theta_{i-1}, e_i, S)$
  Evaluate final weights : $\texttt{Ans} \sim \text{LM}_{\theta_k}(\cdot \mid \tau)$
  Compute reward : $r \leftarrow r(\texttt{Ans}, \tau)$
  if $r >$ threshold$_2$ **then**
    Add chain to KB : $K \leftarrow K \cup \{\texttt{AS}\}$
    $\theta \leftarrow \texttt{RL\_Update}(\theta, r, \texttt{AS})$
  end if
end for

Level III (Open-ended exploration):
for iteration $t = 1, \ldots, T_3$ **do**
  Generate unconstrained AS : $\texttt{AS} \sim \text{LM}_\theta(\cdot \mid C)$ (novel structure)
  Validate (syntax/safety); if invalid **continue**
  Apply AS conservatively (strong meta-reg) : $\theta' \leftarrow \texttt{ApplyStrategy}(\theta, \texttt{AS}, S)$
  Evaluate and compute reward : $\texttt{Ans} \sim \text{LM}_{\theta'}(\cdot \mid \tau)$; $r \leftarrow r(\texttt{Ans}, \tau)$
  if $r >$ threshold$_3$ **then**
    $K \leftarrow K \cup \{\texttt{AS}\}$; $\theta \leftarrow \texttt{RL\_Update}(\theta, r, \texttt{AS})$
  else
    Penalize harmful proposals in policy update
  end if
end for
Return: Refined parameters θ^*, enriched KB K^*

(Chung et al. 2025) for logical reasoning, SocialIQA (Sap et al. 2019) for reasoning about social interactions and CodeMMLU (Manh et al. 2024) for reasoning about code-related tasks. It can be seen that SAMKE in its initial version itself outperforms the task-specific training LoRA's, TTL, DOM and even DnD by a significant margin, showcasing the promising potential it is capable of, if further levels of RL training[11] is completed as well.

Following were the adaptation strategies identified, which enabled SAMKE to reach the accuracy levels presented,

- For ARC-e and PIQA, it was `TTT` family with configuration {"`ttl_steps`": 25, "`learning_rate`": 1e-5, "`batch_size`": 4, "`shuffle_data`": True}
- For ARC-c and SocialIQA, it was `LS` family with configuration {"`times`": 5, "`learning_rate`": 0.1}
- For BoolQ, GSM-MC and MATH-MC, it was `LoRA` family with TS-mixing strategy and the configuration was {"`lambda`": `0.5`}
- For HellaSwag, DivLogicEval and CodeMMLU, it was `TTS` family. Ex:, for Hellaswag, the corresponding configuration was {"`num_prompt_batches`": `20`, "`method`": `max_confidence`}, indicative of the ensemble approach

Ablation Study

A primary effect we would like to isolate and study is that of initial prompt batch provided to start the LLM adaptation process using SAMKE. It would be ideal if SAMKE results in similar performance even if a highly representative, diverse and influential prompt batch is used. For this purpose, inspired by (Tang et al. 2025), we use the following strategy for prompt filtering and selection (see Figure 5).

We first model inter-prompt relations as a directed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{P})$, wherein each prompt is encoded as a vector by using Sentence-BERT. Each vertex $v_i \in \mathbf{V}$ denotes a prompt (sample), a directed edge $e(i, j) \in \mathbf{E}$ connects v_i to its neighbor v_j, and weight $p(i, j) \in \mathbf{P}$ is the cosine similarity of their embeddings. For each node v_i, an s_i is computed as shown below so that nodes with higher average similarity make more connections.

$$s_i = \frac{1}{|\mathbf{V}| - 1} \sum_{j \neq i} s(i, j), \quad k_i = \lceil \alpha \cdot s_i \cdot (|\mathbf{V}| - 1) \rceil$$

Samples are then scored by by (1) influence and (2) diversity. The influence score $I(v)$ is obtained by a diffusion simulation[12]. For this, first initialize an active set $S_{\text{active}} = \{v\}$, then iteratively sample an active node u and attempt to activate each neighbor $w \in N_1(u)$ with probability $p(u, w)$. Newly activated nodes join S_{active}. This process is repeated until no active nodes remain. Let $I(v)$ be the total number of visited nodes. Diversity penalty $D(v)$ measures overlap with already selected nodes:

$$D(v) = -\sum_{i=1}^{k} \beta^i \left| N_i(v) \cap S_{\text{selected}} \right|, \quad f_{\mathcal{G}}(v) = I(v) + \gamma\, D(v)$$

et al. 2021b) datasets. They were selected for two reasons - ease of evaluation and correlation with performance on their subjective counterparts (Zhang et al. 2024).

[11]This might be quite time-intensives however with current version itself taking around 4 days using 2 A6000 GPU's. The reason for using only 2 despite 4, is because Qwen family has 14 attention heads and the vllm serves used for improved efficiency in inference requires this number to be divisible by the number of GPU's which is only possible if either 2 or 7 are available.

[12]The simulation is run 20 times and is then averaged to obtain the final value.

Dataset	In-Domain Tasks					Avg.	Out-of-Domain Tasks					Avg.
	ARC-e	ARC-c	BoolQ	HellaSwag	PIQA	**In**	GSM	MATH	Logic	Social	Code	**Out**
LoRA	47.4	39.7	14.7	26.3	51.5	35.9	15.6	6.8	20.3	39.5	29.8	22.4
TTL	24.4	24.7	44.4	25.9	51.9	34.3	23.5	19.7	26.2	34.9	29.7	26.8
DOM	56.5	38.9	33.2	28.3	18.8	35.1	17.7	2.6	24.7	51.3	31.6	25.6
DnD	70.9	48.1	51.9	26.5	47.8	49.0	20.8	24.1	21.0	33.5	29.1	25.7
SAMKE	**74.7**	**55.5**	**58.8**	**48.3**	**60.1**	**59.5**	**30.3**	**24.5**	**25.1**	**55.0**	**35.6**	**34.1**
Δ DnD	3.8↑	7.4↑	6.9↑	21.8↑	12.3↑	10.4↑	9.5↑	0.4↑	4.1↑	21.5↑	6.5↑	8.4↑
Δ DOM	18.2↑	16.6↑	25.6↑	20.0↑	41.3↑	24.3↑	12.6↑	21.9↑	0.4↑	3.7↑	4.0↑	8.5↑
Δ TTL	50.3↑	30.8↑	14.4↑	22.4↑	8.2↑	25.2↑	6.8↑	4.8↑	1.1↓	20.1↑	5.9↑	7.3↑
Δ LoRA	27.3↑	15.8↑	44.1↑	22.0↑	8.6↑	23.6↑	14.7↑	17.7↑	4.8↑	15.5↑	5.8↑	11.7↑

Table 1: Accuracy (in %) of SAMKE Level I approach over the baselines TTL (25.2↑), LoRA (23.6↑), DOM (24.3↑), and DnD (10.4↑) for in-domain tasks, and TTL (7.3↑), LoRA (11.7↑), DOM (8.5↑), and DnD (8.4↑) for out-of-domain tasks, where values in parentheses denote average Δ (change in accuracy) for **Qwen2.5-0.5B-Instruct**.

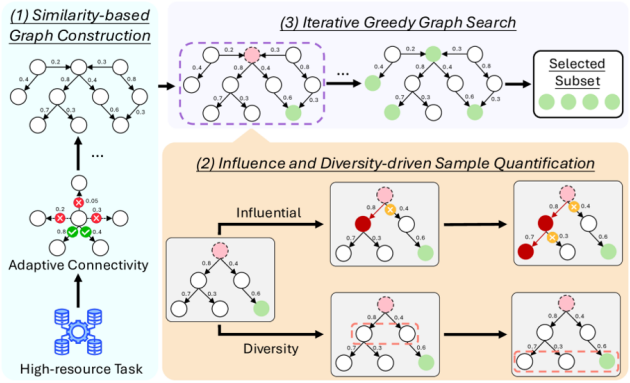

Figure 5: Prompt Selection Strategy (*adapted from* (Liu et al. 2025a))

Finally, greedy graph search is done to select the final prompt subset S. For this, start with $S = \emptyset$ and at each round pick

$$v^* = \arg\max_{v \notin S} f_{\mathcal{G}}(v),$$

v^* is then added to S and diversity penalties only for neighbors of v^* are updated[13]. This process continues until $|S|$ reaches the target size which in our case is 128.

Fortunately, the influence of the initial prompt batch was **marginal** (*with just a **0.3**% improvement in accuracy when averaged across all evaluation datasets*), indicating that SAMKE can efficiently adapt LLMs to unseen datasets without the requirement of high-quality or manually curated dataset. Only a handful of unlabeled prompt instances which are merely indicative of the task suffice.

7. Conclusion

In this work, we propose the SAMKE framework which represents a significant leap in autonomous foundation-model-based agents for scientific meta-knowledge discovery and learning. By enabling large language models to self-improve through parameter-level weight modifications, we are capable of addressing the key limitations in traditional fine-tuning methods such as the high dependency on curated task-specific data and expensive gradient-based training. Initial experimental results demonstrate its superior performance over existing approaches in the domain of commonsense reasoning, showcasing its potential for efficient adaptation to unseen domains. Although the current version faces challenges such as high computational requirements, reliance on a handcrafted seed knowledge base and a labeled evaluation set, we nevertheless lay a robust foundation for future research in open-ended, self-improving AI agents. Future work will focus on scaling to larger models, extending to other fine-tuning methods, and enabling unbounded exploration in the weight space towards truly autonomous meta-learning systems.

[13]Note that the influence scores are precomputed.

References

Acikgoz, E. C.; Qian, C.; Ji, H.; Hakkani-Tür, D.; and Tur, G. 2025. Self-Improving LLM Agents at Test-Time. arXiv:2510.07841.

Agrawal, L. A.; Tan, S.; Soylu, D.; Ziems, N.; Khare, R.; Opsahl-Ong, K.; Singhvi, A.; Shandilya, H.; Ryan, M. J.; Jiang, M.; Potts, C.; Sen, K.; Dimakis, A. G.; Stoica, I.; Klein, D.; Zaharia, M.; and Khattab, O. 2025. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. arXiv:2507.19457.

ang Gao, H.; Geng, J.; Hua, W.; Hu, M.; Juan, X.; Liu, H.; Liu, S.; Qiu, J.; Qi, X.; Wu, Y.; Wang, H.; Xiao, H.; Zhou, Y.; Zhang, S.; Zhang, J.; Xiang, J.; Fang, Y.; Zhao, Q.; Liu, D.; Ren, Q.; Qian, C.; Wang, Z.; Hu, M.; Wang, H.; Wu, Q.; Ji, H.; and Wang, M. 2025. A Survey of Self-Evolving Agents: On Path to Artificial Super Intelligence. arXiv:2507.21046.

Bertolissi, R.; Hübotter, J.; Hakimi, I.; and Krause, A. 2025. Local Mixtures of Experts: Essentially Free Test-Time Training via Model Merging. arXiv:2505.14136.

Bisk, Y.; Zellers, R.; Bras, R. L.; Gao, J.; and Choi, Y. 2019. PIQA: Reasoning about Physical Commonsense in Natural Language. arXiv:1911.11641.

Charakorn, R.; Cetin, E.; Tang, Y.; and Lange, R. T.

2025. Text-to-LoRA: Instant Transformer Adaption. arXiv:2506.06105.

Chen, L.; Prabhudesai, M.; Fragkiadaki, K.; Liu, H.; and Pathak, D. 2025a. Self-questioning language models. *arXiv preprint arXiv:2508.03682*.

Chen, L.; Prabhudesai, M.; Fragkiadaki, K.; Liu, H.; and Pathak, D. 2025b. Self-Questioning Language Models. arXiv:2508.03682.

Chung, T. T.; Liu, L.; Yu, M.; and Yeung, D.-Y. 2025. DivLogicEval: A Framework for Benchmarking Logical Reasoning Evaluation in Large Language Models. *arXiv preprint arXiv:2509.15587*.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. arXiv:1905.10044.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. arXiv:1803.05457.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

Gupta, A.; Mendonca, R.; Liu, Y.; Abbeel, P.; and Levine, S. 2018. Meta-Reinforcement Learning of Structured Exploration Strategies. arXiv:1802.07245.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021a. Measuring Mathematical Problem Solving With the MATH Dataset. arXiv:2103.03874.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021b. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 20222. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Hu, J.; Zhang, Z.; Chen, G.; Wen, X.; Shuai, C.; Luo, W.; Xiao, B.; Li, Y.; and Tan, M. 2025a. Test-Time Learning for Large Language Models. arXiv:2505.20633.

Hu, Y.; Zhang, X.; Fang, X.; Chen, Z.; Wang, X.; Zhang, H.; and Qi, G. 2025b. SLOT: Sample-specific Language Model Optimization at Test-time. arXiv:2505.12392.

Huang, C.; Yu, W.; Wang, X.; Zhang, H.; Li, Z.; Li, R.; Huang, J.; Mi, H.; and Yu, D. 2025a. R-Zero: Self-Evolving Reasoning LLM from Zero Data. arXiv:2508.05004.

Huang, C.; Yu, W.; Wang, X.; Zhang, H.; Li, Z.; Li, R.; Huang, J.; Mi, H.; and Yu, D. 2025b. R-Zero: Self-Evolving Reasoning LLM from Zero Data. *arXiv preprint arXiv:2508.05004*.

Hübotter, J.; Diaz-Bone, L.; Hakimi, I.; Krause, A.; and Hardt, M. 2025. Learning on the Job: Test-Time Curricula for Targeted Reinforcement Learning. arXiv:2510.04786.

Irie, K.; Schlag, I.; Csordás, R.; and Schmidhuber, J. 2022. A Modern Self-Referential Weight Matrix That Learns to Modify Itself. arXiv:2202.05780.

Jin, X.; Wang, K.; Tang, D.; Zhao, W.; Zhou, Y.; Tang, J.; and You, Y. 2024. Conditional LoRA Parameter Generation. arXiv:2408.01415.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Karan, A.; and Du, Y. 2025. Reasoning with Sampling: Your Base Model is Smarter Than You Think. arXiv:2510.14901.

Khan, R. M. S.; Tang, D.; Li, P.; Wang, K.; and Chen, T. 2025. ORAL: Prompting Your Large-Scale LoRAs via Conditional Recurrent Diffusion. arXiv:2503.24354.

Kunin, D.; Sagastuy-Brena, J.; Ganguli, S.; Yamins, D. L.; and Tanaka, H. 2020. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. *arXiv preprint arXiv:2012.04728*.

LeCun, Y. 2022. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1): 1–62.

Lee, H.; Oh, S.; Kim, J.; Shin, J.; and Tack, J. 2025. ReVISE: Learning to Refine at Test-Time via Intrinsic Self-Verification. arXiv:2502.14565.

Li, J.; Dong, X.; Liu, Y.; Yang, Z.; Wang, Q.; Wang, X.; Zhu, S.; Jia, Z.; and Zheng, Z. 2025a. ReflectEvo: Improving Meta Introspection of Small LLMs by Learning Self-Reflection. arXiv:2505.16475.

Li, M.; Lin, J.; Zhao, X.; Lu, W.; Zhao, P.; Wermter, S.; and Wang, D. 2025b. Curriculum-RLAIF: Curriculum Alignment with Reinforcement Learning from AI Feedback. arXiv:2505.20075.

Liang, Z.; Tang, D.; Zhou, Y.; Zhao, X.; Shi, M.; Zhao, W.; Li, Z.; Wang, P.; Schürholt, K.; Borth, D.; et al. 2025. Drag-and-Drop LLMs: Zero-Shot Prompt-to-Weights. *arXiv preprint arXiv:2506.16406*.

Liu, L.; Zhang, C.; Wu, L.; Zhao, C.; Hu, Z.; He, M.; and Fan, J. 2025a. Instruct-of-Reflection: Enhancing Large Language Models Iterative Reflection Capabilities via Dynamic-Meta Instruction. arXiv:2503.00902.

Liu, S.-Y.; Wang, C.-Y.; Yin, H.; Molchanov, P.; Wang, Y.-C. F.; Cheng, K.-T.; and Chen, M.-H. 2024. DoRA: Weight-Decomposed Low-Rank Adaptation. arXiv:2402.09353.

Liu, Y.; Nan, Y.; Xu, W.; Hu, X.; Ye, L.; Qin, Z.; and Liu, P. 2025b. AlphaGo Moment for Model Architecture Discovery. arXiv:2507.18074.

Lu, C.; Holt, S.; Fanconi, C.; Chan, A. J.; Foerster, J.; van der Schaar, M.; and Lange, R. T. 2024. Discovering Preference Optimization Algorithms with and for Large Language Models. arXiv:2406.08414.

Manh, D. N.; Chau, T. P.; Le Hai, N.; Doan, T. T.; Nguyen, N. V.; Pham, Q.; and Bui, N. D. 2024. Codemmlu: A multi-task benchmark for assessing code understanding capabilities of codellms. *CoRR*.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Moradi, M. M.; Amer, H.; Mudur, S.; Zhang, W.; Liu, Y.; and Ahmed, W. 2025. Continuous Self-Improvement of Large Language Models by Test-time Training with Verifier-Driven Sample Selection. arXiv:2505.19475.

Pang, J.-C.; Wang, P.; Li, K.; Chen, X.-H.; Xu, J.; Zhang, Z.; and Yu, Y. 2023. Language Model Self-improvement by Reinforcement Learning Contemplation. arXiv:2305.14483.

Park, C. F.; Zhang, Z.; and Tanaka, H. 2025. *New News*: System-2 Fine-tuning for Robust Integration of New Knowledge. arXiv:2505.01812.

Qwen; :; Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; Lin, H.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Lin, J.; Dang, K.; Lu, K.; Bao, K.; Yang, K.; Yu, L.; Li, M.; Xue, M.; Zhang, P.; Zhu, Q.; Men, R.; Lin, R.; Li, T.; Tang, T.; Xia, T.; Ren, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; and Qiu, Z. 2025. Qwen2.5 Technical Report. arXiv:2412.15115.

Reimers, N.; and Gurevych, I. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Sakaguchi, K.; Bras, R. L.; Bhagavatula, C.; and Choi, Y. 2019. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. arXiv:1907.10641.

Sap, M.; Rashkin, H.; Chen, D.; LeBras, R.; and Choi, Y. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Shao, Y.; Lin, X.; Long, X.; Chen, S.; Yan, M.; Liu, Y.; Yan, Z.; Ma, A.; Tang, H.; and Guo, J. 2025a. ICM-Fusion: In-Context Meta-Optimized LoRA Fusion for Multi-Task Adaptation. arXiv:2508.04153.

Shao, Y.; Yan, M.; Liu, Y.; Chen, S.; Chen, W.; Long, X.; Yan, Z.; Li, L.; Zhang, C.; Sebe, N.; Tang, H.; Wang, Y.; Zhao, H.; Wang, M.; and Guo, J. 2025b. In-Context Meta LoRA Generation. arXiv:2501.17635.

Singh, A.; Co-Reyes, J. D.; Agarwal, R.; Anand, A.; Patil, P.; Garcia, X.; Liu, P. J.; Harrison, J.; Lee, J.; Xu, K.; Parisi, A.; Kumar, A.; Alemi, A.; Rizkowsky, A.; Nova, A.; Adlam, B.; Bohnet, B.; Elsayed, G.; Sedghi, H.; Mordatch, I.; Simpson, I.; Gur, I.; Snoek, J.; Pennington, J.; Hron, J.; Kenealy, K.; Swersky, K.; Mahajan, K.; Culp, L.; Xiao, L.; Bileschi, M. L.; Constant, N.; Novak, R.; Liu, R.; Warkentin, T.; Qian, Y.; Bansal, Y.; Dyer, E.; Neyshabur, B.; Sohl-Dickstein, J.; and Fiedel, N. 2024. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. arXiv:2312.06585.

Tang, X.; Lv, Z.; Cheng, X.; Li, J.; Zhao, W. X.; Wen, Z.; Zhang, Z.; and Zhou, J. 2025. Enhancing Cross-task Transfer of Large Language Models via Activation Steering. arXiv:2507.13236.

Tao, Z.; Lin, T.-E.; Chen, X.; Li, H.; Wu, Y.; Li, Y.; Jin, Z.; Huang, F.; Tao, D.; and Zhou, J. 2024. A Survey on Self-Evolution of Large Language Models. arXiv:2404.14387.

Wang, K.; Tang, D.; Zhao, W.; Schürholt, K.; Wang, Z.; and You, Y. 2025a. Recurrent diffusion for large-scale parameter generation. *arXiv preprint arXiv:2501.11587*.

Wang, R.; Dvijotham, K.; and Manchester, I. R. 2025. Norm-Bounded Low-Rank Adaptation. arXiv:2501.19050.

Wang, R.; Ping, P.; Guo, Z.; Zhang, X.; Shi, Q.; Zhou, L.; and Ji, T. 2025b. LoKI: Low-damage Knowledge Implanting of Large Language Models. arXiv:2505.22120.

Wang, Y.; Liu, X.; Chen, X.; O'Brien, S.; Wu, J.; and McAuley, J. 2025c. Self-Updatable Large Language Models by Integrating Context into Model Parameters. arXiv:2410.00487.

Wang, Z.; Ma, D.; Huang, X.; Cai, D.; Lan, T.; Xu, J.; Mi, H.; Tang, X.; and Wang, Y. 2025d. The End of Manual Decoding: Towards Truly End-to-End Language Models. arXiv:2510.26697.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29.

Wu, T.; Wang, J.; Zhao, Z.; and Wong, N. 2025. Mixture-of-Subspaces in Low-Rank Adaptation. arXiv:2406.11909.

Wu, Z.; Arora, A.; Wang, Z.; Geiger, A.; Jurafsky, D.; Manning, C. D.; and Potts, C. 2024. ReFT: Representation Finetuning for Language Models. arXiv:2404.03592.

Yang, Z.; Band, N.; Li, S.; Candès, E.; and Hashimoto, T. 2024. Synthetic continued pretraining. arXiv:2409.07431.

Yuan, W.; Pang, R. Y.; Cho, K.; Li, X.; Sukhbaatar, S.; Xu, J.; and Weston, J. 2025. Self-Rewarding Language Models. arXiv:2401.10020.

Yuksekgonul, M.; Bianchi, F.; Boen, J.; Liu, S.; Huang, Z.; Guestrin, C.; and Zou, J. 2024. TextGrad: Automatic "Differentiation" via Text. arXiv:2406.07496.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? arXiv:1905.07830.

Zhang, G.; Meng, F.; Wan, G.; Li, Z.; Wang, K.; Yin, Z.; Bai, L.; and Yan, S. 2025. LatentEvolve: Self-Evolving Test-Time Scaling in Latent Space. arXiv:2509.24771.

Zhang, T. 2025. Toward Weight-level Self-improving Agents with Meta-knowledge Discovery. *10.36227/techrxiv.175744083.37752625/v1*.

Zhang, Z.; Jiang, Z.; Xu, L.; Hao, H.; and Wang, R. 2024. Multiple-choice questions are efficient and robust llm evaluators. *arXiv preprint arXiv:2405.11966*.

Zhao, Z.; Shen, T.; Zhu, D.; Li, Z.; Su, J.; Wang, X.; Kuang, K.; and Wu, F. 2024. Merging LoRAs like Playing LEGO: Pushing the Modularity of LoRA to Extremes Through Rank-Wise Clustering. arXiv:2409.16167.

Zheng, S.; Wang, H.; Huang, C.; Wang, X.; Chen, T.; Fan, J.; Hu, S.; and Ye, P. 2025. Decouple and Orthogonalize: A Data-Free Framework for LoRA Merging. arXiv:2505.15875.

Zhou, H.; Chen, Y.; Guo, S.; Yan, X.; Lee, K. H.; Wang, Z.; Lee, K. Y.; Zhang, G.; Shao, K.; Yang, L.; and Wang, J. 2025. Memento: Fine-tuning LLM Agents without Fine-tuning LLMs. arXiv:2508.16153.

Zuo, Y.; Zhang, K.; Sheng, L.; Qu, S.; Cui, G.; Zhu, X.; Li, H.; Zhang, Y.; Long, X.; Hua, E.; Qi, B.; Sun, Y.; Ma, Z.; Yuan, L.; Ding, N.; and Zhou, B. 2025. TTRL: Test-Time Reinforcement Learning. arXiv:2504.16084.

Zweiger, A.; Pari, J.; Guo, H.; Akyürek, E.; Kim, Y.; and Agrawal, P. 2025. Self-Adapting Language Models. arXiv:2506.10943.