MAML-en-LLM: Model Agnostic Meta-Training of LLMs for Improved In-Context Learning

Sanchit Sinha*
sanchit@virginia.edu
University of Virginia
Charlottesville, VA, USA

Yuguang Yue
yueyugua@amazon.com
Amazon AGI
New York, NY, USA

Victor Soto
nvmartin@amazon.com
Amazon AGI
New York, NY, USA

Mayank Kulkarni
maykul@amazon.com
Amazon AGI
Cambridge, MA, USA

Jianhua Lu
jianhual@amazon.com
Amazon AGI
Cambridge, MA, USA

Aidong Zhang
aidong@virginia.edu
University of Virginia
Charlottesville, VA, USA

ABSTRACT

Adapting large language models (LLMs) to unseen tasks with in-context training samples without fine-tuning remains an important research problem. To learn a robust LLM that adapts well to unseen tasks, multiple *meta-training* approaches have been proposed such as MetaICL and MetaICT, which involve meta-training pre-trained LLMs on a wide variety of diverse tasks. These meta-training approaches essentially perform in-context **multi-task fine-tuning** and evaluate on a disjointed test set of tasks. Even though they achieve impressive performance, their goal is never to compute a truly *general* set of parameters. In this paper, we propose **MAML-en-LLM**, a novel method for meta-training LLMs, which can learn truly generalizable parameters that not only **performs** well on disjointed tasks but also **adapts** to unseen tasks. We see an average increase of 2% on unseen domains in the performance while a massive 4% improvement on adaptation performance. Furthermore, we demonstrate that MAML-en-LLM outperforms baselines in settings with *limited* amount of training data on both seen and unseen domains by an average of 2%. Finally, we discuss the effects of type of tasks, optimizers and task complexity, an avenue barely explored in meta-training literature. Exhaustive experiments across 7 task settings along with two data settings demonstrate that models trained with MAML-en-LLM outperform SOTA meta-training approaches.

KEYWORDS

meta learning, llms, in-context learning, optimization, generalization

ACM Reference Format:
Sanchit Sinha, Yuguang Yue, Victor Soto, Mayank Kulkarni, Jianhua Lu, and Aidong Zhang. 2024. MAML-en-LLM: Model Agnostic Meta-Training of LLMs for Improved In-Context Learning. In *Proceedings of SIGKDD (KDD '24).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Work done while interning at Amazon Alexa.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Large Language Models (LLMs) have revolutionized natural language processing (NLP) and achieved state-of-the-art performance on a variety of diverse tasks. Recently, LLMs have been demonstrated to learn *in-context* [18], without expensive and compute-intensive fine-tuning. In-context learning (ICL) involves pre-appending the target sample by carefully selected task-specific *exemplars* - which act as conditioning for LLMs on that particular task. Learning in-context is an attractive proposition as its evaluation only requires inference. With no gradient updates required, ICL can potentially improve LLM generalization to new and diverse tasks with only a few examples.

Even though out-of-the-box pre-trained LLMs show good ICL performance, multiple avenues of research have demonstrated improved ICL performance by *warming-up* out-of-the-box LLMs [4, 13]. A few model warmup techniques for improved ICL performance have been proposed recently like MetaICL [13] and MetaICT [4]. The evaluation of such models on unseen tasks proves their efficacy in generalization. Meta-training usually involves adapting (i.e., fine-tuning) pre-trained LLMs using a diverse set of tasks, formatted as an ICL instance by pre-appending exemplars in the prompts during training. Once the model is meta-trained, the evaluation is usually performed on a distinct and disjoint set of tasks, never seen during training. The process of warming up has been dubbed as *meta-training* in these approaches which borrows from research in classical machine learning literature of *meta-learning* which attempts to warm up models for faster adaptation to unseen tasks.

One of the most commonly utilized techniques for meta-learning in classical literature is Model-Agnostic Meta-Learning (MAML) proposed by Finn et al. [6]. MAML is formulated as a two-step optimization problem where the inner loop *adapts* copies of the model parameters to various diverse tasks, while the outer loop updates the initial model parameters involving a second-order gradient update calculated on the adapted parameters. Even though LLM meta-training approaches [4, 13] are eponymous to *meta-learning*, they do not utilize the two-step optimization framework for meta-training LLMs instead only adapting the model parameters continuously using a mixture of diverse tasks. For all practical purposes, the proposed meta-training approaches are analogous to *multi-task fine-tuning* with added exemplars in the training sample prompts. Training models in this way does not fully exploit the parameter space and nor does it benefit from second-order gradients that guide

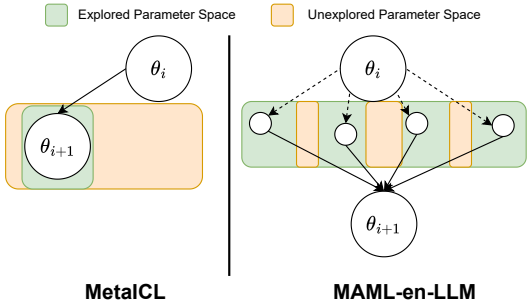

Figure 1: Visual comparison between MetaICL and *MAML-en-LLM*. The figure demonstrates a single model parameter (θ) update step from parameters at step i - θ_i to step $i+1$ - θ_{i+1}. The dotted lines represent the adaptation phase and the solid lines represent the update. As MetaICL does not have an explicit adaptation phase, the update happens directly and with only a limited parameter space explored. The parameter updates for a single step is calculated using only a single task. On the other hand, *MAML-en-LLM* first explores a wide parameter space using multiple adapted parameters and subsequently performs the final meta-update with the second-order gradients calculated from the intermediate adapted parameters.

the direction of meta-updates. In this paper, we propose **MAML-en-LLM** (pronounced *Mammaliam*), a novel method for meta-training LLMs. Figure 1 visualizes the differences between our approach and the most popular state-of-the-art meta-training approach (at the time of writing) - MetaICL.

Our main contributions are detailed below:

- This paper is the first work to effectively utilize principles outlined in meta-learning literature and propose *MAML-en-LLM* - a novel methodology to meta-train LLMs, specifically for improving ICL performance.
- We demonstrate that *MAML-en-LLM* outperforms existing meta-training approaches on ICL generalization performance on two commonly encountered settings - when training data is either limited (low resource) or abundant (high resource).
- We report that models trained using *MAML-en-LLM* demonstrate superior performance on the challenging setting of *very few-shot adaptation* to unseen domains.
- We present results on design considerations previously unexplored in meta-learning literature on LLMs, namely - the effect of number of exploration states (tasks), the role of optimizers, and the effect of task types on generalization.

2 RELATED WORK

2.1 In-context Learning (ICL) in LLMs

In-context Learning (ICL) was first proposed by Brown et al. [3] as an extremely inexpensive alternative to regular fine-tuning. ICL requires no parameter updates as the input prompt is pre-appended with task-specific *exemplars* which are examples of the specific task to be performed. LLMs have been shown to perform exceptionally well when prompted with a few examples of the task pre-appended in the prompt [18]. This behavior was first studied for LLMs in GPT-3 [15]. Subsequent studies have shown that they can even solve complex problems like math [15] and reasoning [17]. Why LLMs are so adept at in-context learning remains an open research topic that is attracting attention; for example Akyürek et al. [1] compares their behavior to linear models. Empirical approaches like [21] and [10] have demonstrated that types of exemplars are important for ICL performance.

2.2 Meta-Learning for Generalization

One of the most popular approaches for meta-learning was first proposed by [6] as Model Agnostic Meta-Learning (MAML). The goal of MAML was to *warmup* pre-trained models over a diverse set of tasks using an inner-outer minimization algorithm to learn a general set of parameters that can be adapted to new tasks using only a few examples (few-shot).

The inner loop usually focuses on a set of tasks, while the outer loop utilizes the learned parameters and the second-order gradient information from the inner loop to update the model, in turn capturing the direction of gradient updates. Usually, MAML requires the outer loop to perform a second-order gradient update which results in a higher computational burden. Even though MAML is effective, it is not without serious issues; for example, it is known to suffer from memorization [20], which entails the model repeating the data shown to it during training without learning anything. MAML has also been shown to be sensitive to training hyper-parameters and complex strategies have been proposed to enforce stability during its training process [2]. MAML has also been used with some success for training language models [5, 8, 11, 16] on particular tasks.

2.3 Meta-training for improving ICL

To improve the ICL performance of out-of-box pre-trained models, several meta-training approaches have been proposed. We discuss the two seminal works - MetaICT [4] and MetaICL [13] here. MetaICT uses BinaryCLFs and LAMA datasets to create tasks while also pre-appending human-generated instructions to each task. MetaICL on the other hand uses a wide variety of disjoint tasks with no human-generated instructions to meta-train pre-trained LLMs. Both approaches update the model parameters on a batch from training tasks continuously - making them similar to fine-tuning. A recent related work [14] attempts to utilize MAML to train LLMs for improving Prompt Tuning. However, the work is significantly different from ours - as generalization is performed on learned soft embedding of tokens and not on model parameters - a vastly different objective from ours which is improving ICL performance.

Comparision to existing works. We discuss the salient differences with MetaICL first. MetaICL discards principles of meta-learning and effectively only uses multi-task fine-tuning to meta-train their models. Next, MetaICT follows an identical training process to MetaICL. Even though MetaICT attempts to compare MetaICT to MAML, they only utilize the first-order approximation and a single task to update the model parameters in the inner optimization step possibly due to not conducting proper investigation into a myriad of sources of errors like optimizers and exploration states. Our work is significantly different wherein, we use second-order gradients to meta-train models and also provide exhaustive results and discussions around utilizing MAML-en-LLM for various types of tasks.

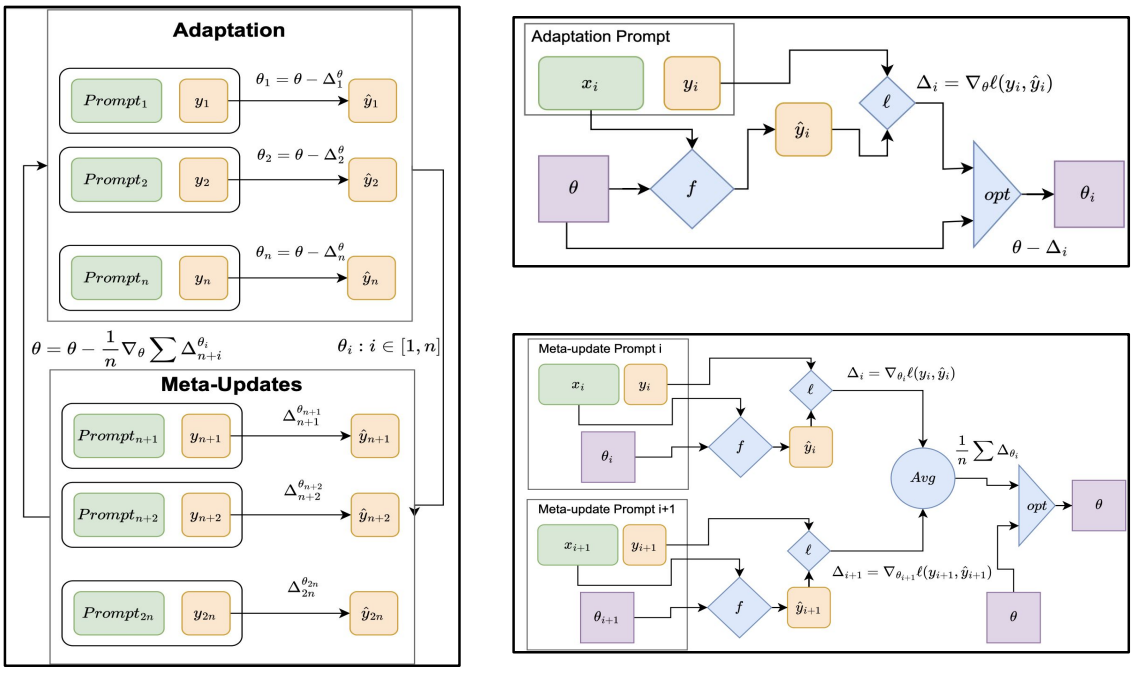

Figure 2: Schematic figure demonstrating MAML training for meta-training LLMs. MAML is a bi-level optimization framework with an inner update (adaptation) and an outer update (meta-update). In the figure, the green cells represent the input samples (prompts), the yellow cells represent task labels, blue components represent functions and purple boxes represent model parameters. Multiple task batches are utilized to compute a set of adapted parameters (equal to the number of tasks) represented by θ_i. The outer update utilizes the adapted parameters to compute second-order gradients ($\Delta_i^{\theta_i}$) using a separate set of task batches. The final meta-update updates the unadapted parameter θ with an average of second-order gradients ($\Delta_i^{\theta_i}$) to compute the next set of updated parameters.

3 METHODOLOGY

In this section, we first go over the standard meta-training procedure for improving ICL performance of LLMs by discussing the problem setup. Subsequently, we provide a mathematical overview of current SOTA approaches and our proposed approach. Next, we provide finer details for *MAML-en-LLM* and its various components - task adaptation and aggregated meta-update phases in addition to detailed optimization perspective.

3.1 Meta-training: Problem Statement

In-context learning is an inference-only method, where exemplars from the same tasks with expected output values are provided in the prompt as conditioning for the LLM. During meta-training, a similar setup is followed where for a given train sample (x_i, y_i), belonging to a task $\mathbb{C}$, k samples (x_1, y_1),(x_2, y_2),...,(x_k, y_k) from the same task $\mathbb{C}$ are sampled. The k samples are pre-appended in the prompt with the final train sample x_i. The target label y_i is used to calculate the loss and the pre-trained LLM (f) is meta-trained over all available training tasks $\mathbb{C}$ with standard classification training objectives (ℓ). Mathematically, a parameter update step can be represented as:

$$\theta = \theta - \nabla_\theta \, \ell(f(x_1, y_1, x_2, y_2, ...x_k, y_k, x_i), y_i) \; \forall (x_i, y_i) \in \mathbb{C} \quad (1)$$

We provide more detail about the exact update methodology in the next section.

3.2 Model Agnostic Meta-Learning for LLMs (*MAML-en-LLM*)

Note that the present state-of-the-art meta-training methods like MetaICL[13] and MetaICT[4] perform optimization in line with multi-task finetuning. Mathematically, MetaICL and MetaICT solve a single optimization objective:

$$min \; \theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_\theta) \quad (2)$$

On the other hand, *MAML-en-LLM* aims to learn a generalizable set of parameters by training a model on a diverse variety of tasks with two distinct phases - an inner phase which controls the extent of exploration - adaptation and an outer phase which controls the magnitude of the update - the meta-update. As a consequence, *MAML-en-LLM* can be represented as a bi-level optimization with an inner update step and an outer (meta) update step where it solves a dual optimization training objective given by:

$$min \; \theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}) \quad (3)$$

3.2.1 ***Task Adaptation.*** The inner optimization adapts the parameters to a set of tasks. To achieve this n tasks are sampled where each task is a set of size k. The model is adapted on each task by performing gradient descent on the original parameters for

k steps also known traditionaly as *support set* S. The number of elements k in the *support set* is equal to the number of steps for adaptation using gradient descent. Intuitively, the higher the number of tasks n, the greater the exploration of the parameter space. The adapted parameters are calculated using Equation 4. Mathematically, the inner loop adapts to a distribution of tasks $\mathcal{T}_i \sim p(\mathcal{T})$ where $p(\mathcal{T})$ represents a probability distribution over diverse tasks. Note that a task represents a randomly sampled batch of training prompts as discussed in Min et al. [13].

$$\theta_i \leftarrow \theta - \alpha \nabla_\theta \ell_{\mathcal{T}_i}(f_\theta) \tag{4}$$

where $\mathcal{T}_i \sim p(\mathcal{T})$ is sampled from all tasks and θ represents the model parameters, α is the learning rate of the adaptation step and ℓ is the Cross Entropy Loss.

3.2.2 ***Aggregated Meta-updates.*** The outer update utilizes a distinct *query set* $Q(\perp S)$ to calculate the gradients with respect to the adapted parameters. The size of the query set is kept the same as the support set (i.e. $k = \|Q\| = \|S\|$). In classic MAML literature, the same query set is usually used to calculate the meta-update. However *MAML-en-LLM* utilizes n different tasks to perform the meta-updates. For each task the calculated gradients (using the query set) are collected and averaged to perform a second-order gradient update on the original unadapted parameter as per Equation (5). The outer optimization performs the meta-update on the unadapted parameter based on the second-order gradient updates of the adapted set of parameters $\theta_i's$. Mathematically the second-order update can be represented as:

$$\theta \leftarrow \theta - \beta \nabla_\theta \Sigma_{\mathcal{T}_i \sim p(\mathcal{T})} \ell_{\mathcal{T}_i}(f_{\theta_i}) \tag{5}$$

where β is the learning rate for the outer optimization step.

3.3 Shared Adaptive Optimizer Moments

One of the most significant differences between *MAML-en-LLM* and MetaICL is the presence of a dual optimization problem in *MAML-en-LLM*. The dual optimization problem poses unique challenges in LLMs where the choice of optimizers affects generalization drastically. Note that typically, adaptive optimizers (like AdamW) are preferred over stateless optimizers (like SGD with momentum). For a typical adaptive optimizer, given gradients g_t at step t, observations sampled from a batch B, hyperparameters β_1 and β_2, two moving averages are calculated - the first moment m_t and second moment v_t as follows:

$$m_t^B \leftarrow (\beta_1 * m_{t-1} + (1-\beta_1) * g_t)/(1-\beta_1^t)$$
$$v_t^B \leftarrow (\beta_2 * v_{t-1} + (1-\beta_2) * g_t^2)/(1-\beta_2^t)$$

Note that the optimizer in the inner update is re-initialized after every meta-update, implying that the moments are re-initialized as well. This significantly changes the landscape of optimization, making inner gradient updates contribute disproportionately more as the training continues for a longer duration, especially near the minima. To alleviate this problem, we propose optimizer *parameter sharing* between the inner and outer optimizers - i.e., constantly updating shared moving averages between optimizers where only a single set of optimizer parameters are updated.

Algorithm 1: MAML-en-LLM

Input : Training samples:X, Task Labels:Y, Training Corpus: $\{x_i \in X; y_i \in Y\}$, Steps: N, Model f, Pre-trained Model Parameters: θ_{pt}, Learning Rates: α, β, Moment hyperparameters β_1, β_2, Size of support and query set n, Loss function ℓ usually the cross entropy loss

Output : Meta-trained model parameters θ

$\theta \leftarrow \theta_{pt}$
$t \leftarrow 0, m, v \leftarrow 0$
for $t \in 1, 2, 3, ..., N$ **do**
  Sample Support set of size n, $\{x_i\}^n, \{y_i\}^n \in \{X, Y\}$
  for $i \in 1, 2, 3, ..n$ **do**
    $g_i(t) = \nabla_\theta \ell(f_\theta(x_i), y_i)$
    $m \leftarrow (\beta_1 * m + (1-\beta_1) * g_i(t))/(1-\beta_1^t)$
    $v \leftarrow (\beta_2 * v + (1-\beta_2) * (g_i(t))^2)/(1-\beta_2^t)$
    $\theta_i \leftarrow \theta - (\alpha * m/\sqrt{v} + |\theta|_2)$
  end
  Sample Query set of size n, $\{x_i\}^n, \{y_i\}^n \in \{X, Y\}$
  for $i \in 1, 2, 3, ..n$ **do**
    $g'(t) = \frac{1}{n} * \sum_k \ell(f_{\theta_i}(x_i), y_i)$
    $m \leftarrow (\beta_1 * m + (1-\beta_1) * g'(t))/(1-\beta_1^t)$
    $v \leftarrow (\beta_2 * v + (1-\beta_2) * (g'(t))^2)/(1-\beta_2^t)$
    $\theta \leftarrow \theta - (\beta * m/\sqrt{v} + \frac{1}{k} * \sum_k |\theta_i|_2)$
  end
end

3.4 Consolidated Meta-Training using MAML-en-LLM

Figure 2 provides a schematic overview of the consolidated training approach and Algorithm 1 details the exact MAML-en-LLM training procedure. As we utilize n tasks for calculating the adapted parameters and subsequently utilize k tasks for adaptation steps and calculating meta-updates, we represent our *MAML-en-LLM* terminology by *MAML-2k-n*. Hence, if the size of the support and query set is 1 and the number of tasks during adaptations are also 1, the *MAML-en-LLM* setting will be represented as **MAML-2-1**. It is intuitive to see that the meta-update frequency can easily be calculated as $2kn$, hence for MAML-2-1, the frequency of meta-updates is 2. Similarly, for **MAML-2-4**, the frequency of meta-updates is 8.

4 EXPERIMENTS

4.1 Dataset Description

We utilize two datasets with a wide diversity of tasks - CROSSFIT [19] and UNIFIEDQA [7] consisting of total of 142 distinct tasks. We use the exact same task splits used by MetaICL [13], however due to active developments, we utilize latest versions of tasks. All the tasks fall in the following 4 categories with increasing complexity: text classification, natural language inference, question answering, and paraphrasing. The training tasks and the testing tasks are ensured to be disjointed. The testing tasks consist of two types of tasks - tasks with similar domains in training set and sampled from unseen

Train Setting	Train	Test Setting	Test	Unseen
HR	61	LR	26	4
Classification	43	Classification	20	4
Non-Classification	37			
QA	37	QA	22	4
Non-QA	33			
Non-NLI	55	NLI	8	1
Non-Paraphrase	59	Paraphrase	4	1

Table 1: Statistics of number of train/test tasks in the 7 meta-training settings considered for evaluation. The number of tasks in the Unseen domain is listed in the last column. Dataset split is identical to Min et al. [13]

domains in training set. The statistics of the dataset splits are detailed in Table 1.

4.2 Training Details

We utilize a pre-trained GPT-2 Medium [15] consisting of 355 million parameters for all our experiments. We consider both the standard models and Noisy channel models [12] for evaluation of all experiments. Training is performed on 8 Tesla V100 GPUs using Pytorch and pre-trained model checkpoints are taken from HuggingFace. Training time for MetaICL models for 50k steps is about 5 hours, while for *MAML-en-LLM* models is about 12 hours.

Example prompt structure. We first visualize the prompt structure during training of both standard and channel models (Figure 3 (Green). Next, we visualize the prompt structure during ICL inference of both standard and channel models (Red). The *Max* operator selects the label C with the max probability,

4.3 Hyper-parameter settings

Training. For both MetaICL and MAML models, the training sequence length is fixed at 1024. The batch size is fixed at 1. For MetaICL baselines, we utilize the learning rate of $1\text{x}10^{-5}$. As MAML consists of an inner (α) and outer learning rate (β), we utilize the same learning rate of $1\text{x}10^{-5}$ for both α and β. For MAML, the size of support and query sets is chosen as 1 (number of batches). The number of tasks is chosen as 1 (MAML-2-1) and 4 (MAML-2-4). This implies the frequency of meta-updates is 2 and 8 respectively. We train both paradigms of models for 50k steps. For both approaches, AdamW is used as an optimizer. As MAML models require an inner and outer optimizer, we utilize AdamW for both optimizers with identical hyper-parameters. As every AdamW's optimization step depends on the last gradient update, we copy the inner optimizer's last gradient state to the outer optimizer for every meta-update. The training seed is set as 100. More details on optimizers can be found in the Appendix5.5.

Inference. We utilize identical evaluation framework as [13] for fair comparisons. The test sequence length is fixed at 256 or 16 exemplars whichever is lower. The batch size is fixed at 16 samples during inference.

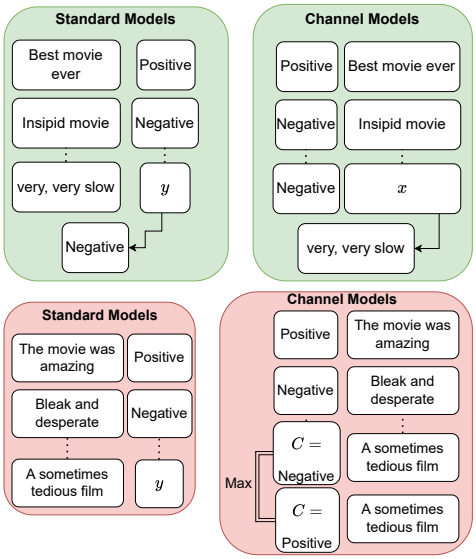

Figure 3: Example training and test prompts for Standard (Left) and Channel (Right) Models. The training procedure of Channel models learns to predict the sample, conditioned on its true label. During inference, channel models predict the target sample itself conditioned on all possible labels for the task. In this example, the task is Sentiment Analysis and hence has only two labels Positive and Negative.

4.4 Comparison of Baselines and Replication

We compare our proposed methodology against a variety of different models and prompt settings. We utilize 2 types of models - standard models and noisy channel models as proposed by Min et al. [12]. For standard models, the training and inference procedure for a given input sample x_i, y_i and exemplars $(x_1, y_1, ...x_k, y_k)$ is as follows (Refer problem setup in Section 3.1):

$$\theta = \theta - \nabla_\theta \, \ell(f(x_1, y_1, x_2, y_2, ...x_k, y_k, x_i), y_i) \tag{6}$$

$$y_i = \arg\max P(y|x_1, y_1, x_2, y_2, ...x_k, y_k, x_i) \tag{7}$$

where ℓ is the Cross Entropy loss. On the other hand, noisy-channel models (referred to as Channel models from now on) treat the training and inference procedure as a generative problem rather than a classification task. During training, the labels and prompts are flipped and reordered, and the target label is appended in the prompt itself, while the training instance is treated as the model's generated output. Mathematically, the training procedure optimizes the following:

$$\theta = \theta - \nabla_\theta \, \ell(f(y_1, x_1, y_2, x_2, ...y_k, x_k, y_i), x_i) \tag{8}$$

During inference, all possible target labels are considered and appended in the prompt, and the prompt with the highest probability is treated as the correct label.

$$y_i = \arg\max_{c \in C} P(x_i|y_1, x_1, y_2, x_2, ...y_k, x_k, c) \tag{9}$$

where C represents all possible labels for the task. For both models, we consider 3 different settings based on the prompt structure and the model used. We detail the settings below.

Method	HR → LR	Class → Class	Non-Class → Class	QA → QA	Non-QA → QA	Non-NLI → NLI	Non-Para → Para
No ICL	27.71	27.71	27.71	44.06	44.06	33.5	35.23
Raw LM	29.12/25.81	29.12/25.81	29.12/25.81	45.43/44.68	45.43/44.68	42.59/34.01	41.43/34.72
MetaICL	42.54/38.97	38.14/36.25	**36.08**/32.56	59.87/58.43	38.30/35.93	78.85/73.2	**59.86**/54.92
MAML-2-1	34.59/33.57	**42.94**/41.31	31.55/26.41	**61.99**/61.25	34.93/32.81	**80.61**/77.11	33.32/32.22
MAML-2-4	**43.16**/42.11	40.65/39.99	32.47/27.61	59.37/58.75	**40.49**/39.06	60.47/52.8	34.05/34.05
Channel No ICL	30.68	30.68	30.68	45.93	45.93	33.29	41.09
Channel Raw LM	42.66/39.81	42.66/39.81	42.66/39.81	44.24/41.87	44.24/41.87	38.50/34.7	53.03/48.41
Channel MetaICL	48.71/47.27	**49.46**/47.49	44.08/43.35	**57.37**/55.93	**48.49**/46.87	55.71/45.07	48.49/46.87
Channel MAML-2-1	**51.19**/48.54	48.65/47.01	46.45/44.52	55.93/54.06	46.81/45	64.55/60.18	**59.15**/56.02
Channel MAML-2-4	50.93/47.52	49.04/47.79	**46.83**/44.86	55.62/54.06	47.06/45	**64.99**/59.99	57.97/54.17

(a) Performance on unseen tasks utilizing *complete data setting*.

Method	HR → LR	Class → Class	Non-Class → Class	QA → QA	Non-QA → QA	Non-NLI → NLI	Non-Para → Para
No ICL	27.71	27.71	27.71	44.06	44.06	33.5	35.23
Raw LM	29.12/25.81	29.12/25.81	29.12/25.81	45.43/44.68	45.43/44.68	42.59/34.01	41.43/34.72
MetaICL	**43.78**/41.21	37.22/34.06	**35.31**/31.64	51.24/49.68	**44.62**/41.87	33.16/33.16	38.66/32.59
MAML-2-1	41.30/39.08	**38.67/36.96**	34.02/31.78	52.93/51.24	43.68/41.87	53.57/42.07	34.02/33.98
MAML-2-4	42.29/40.18	38.34/36.95	34.04/31.94	**53.49**/52.81	41.93/38.43	**57.52**/55.09	**39.21**/38.15
Channel No ICL	30.68	30.68	30.68	45.93	45.93	33.29	41.09
Channel Raw LM	42.66/39.81	42.66/39.81	42.66/39.81	44.24/41.87	44.24/41.87	38.50/34.7	53.03/48.41
Channel MetaICL	45.06/42.97	**46.98**/44.52	43.69/42.31	55.36/53.12	**50.62**/48.75	53.57/34.39	**53.72**/49.73
Channel MAML-2-1	**48.75**/46.44	45.9/44.08	**45.4**/43.84	55.62/54.37	49.62/48.12	**54.16**/37.22	52.74/48.06
Channel MAML-2-4	46.01/43.53	45.76/44.08	45.27/44.36	**55.80**/ 54.06	49.55/48.43	54.02/37.1	52.76/47.82

(b) Performance on unseen tasks utilizing *limited data setting*.

Table 2: For both tables, Rows 1-3 represent the baselines and Rows 4 and 5 represent the performance of MAML settings. Similarly, Rows 6-8 represent baselines using the Channel training/inference and Rows 9 and 10 represent MAML settings on Channel models. Numbers are represented as X/Y where X represents the average performance and Y represents the worst-case performance. The entries in bold are the best-performing models.

- **No ICL**: The prompt only consists of the target instance(x_i) and estimates target label (y_i).
- **RawLM**: The prompt consists of exemplars (x_1,y_1,..,x_n,y_n) followed by the target sample (x_i) and estimates the target label (y_i). The model used is a pre-trained out-of-box LLM. Note, for Channel models, the prompt structure is flipped as shown in Equation 9.
- **MetaICL**: We replicate the exact MetaICL setting by training the model identical to [13]. Please refer to Section 4.2, for more information.

Replication of MetaICL. We replicate the training procedure of MetaICL and Channel MetaICL models on the latest versions of the datasets and on the GPT-2 Medium models. Differently from the provided approach, we do not utilize 8-bit optimization and mixed precision training. Our replicated results agree with the reported results on all the data splits in the paper and even perform better than the reported results on GPT-2 Medium (Table 11 in [13]).

4.5 Evaluation Criterion and Metrics

To quantify the performance of classification tasks, macro-F1 score is utilized (Refer [13] for details) which is suitable for settings with class imbalances. For all other tasks prediction accuracy is used instead. We report the average and worst-case performance on five random seeds (identical to [13]). To compare performances among various methods for the same data setting, we report the **win-rates** of *MAML-en-LLM* models. We consider a 'win' for all cases where both the average and worst case performances of *MAML-en-LLM* models outperform their counterparts and are significant (discussed later). The numbers in bold represent the best-performing methods for each data setting.

5 RESULTS AND DISCUSSION

5.1 Experiment-1: Generalization Performance

We evaluate our methodology on two commonly encountered data settings in practice. These settings are useful to demonstrate the efficacy of our method on both high and low-resource settings.

Complete Data Setting (High Resource). Comprised of the entire training set from each task dataset. We utilize the exact same task and data splits as MetaICL[13]. The statistics for training and testing data are detailed in Table 1.

Limited Data Setting (Low Resource). We sample 10% of training data from each task dataset utilizing the same dataset seeds. To combat the bias introduced during sampling, we ensure the proportion of labels in the sampled data is the same as in the complete data setting, i.e., the sampling is equally stratified (Table 1).

To compare the generalization performance of *MAML-en-LLM* and MetaICL, we evaluate first on test tasks in unseen training domains and next on all test tasks. As ICL is sensitive to the selected *exemplars* in the prompt, we consider five seeds while creating prompts for the test set. The exemplars are sampled from the train set of the test tasks and performance is averaged. We report both the average and the worst-case performances over all five seeds.

Method	HR → LR	Class → Class	Non-Class → Class	QA → QA	Non-QA → QA	Non-NLI → NLI	Non-Para → Para
No ICL	33.09	33.3	33.3	38.85	38.85	25.89	33.44
Raw LM	36.68/35.98	34.95/34.24	34.95/34.24	39/38.64	39/38.64	32.91/31.41	37.26/35.32
MetaICL	**42.42**/41.25	42.52/42.11	**42.21**/40.88	41.48/41.25	**36.71**/35.97	50.03/46.58	33.1/33.1
MAML-2-1	41.88/41.43	41.88/41.08	40.09/38.69	42.38/42.22	20.36/18.98	**50.05**/46.81	16.39/16.04
MAML-2-4	41.6/40.81	**42.82**/42.01	40.46/39.29	**42.56**/42.41	36.13/35.32	44.71/42.42	**38.64**/38.14
Channel No ICL	36.59	33.98	33.98	38.28	38.28	33.31	46.16
Channel Raw LM	41.95/40.83	45.31/45.09	45.31/45.09	39.97/39.51	39.97/39.51	38.83/37.65	45.2/44.38
Channel MetaICL	46.25/45.42	50.35/49.71	49.03/48.02	**42.75**/42.54	**40.46**/40.08	48/47.1	51.30/49.76
Channel MAML-2-1	48.10/47.16	51.08/50.46	50.31/49.87	42.67/42.43	40.13/39.93	52.38/51.22	**54.18**/52.61
Channel MAML-2-4	**48.03**/47.26	**51.13**/50.49	**50.49**/49.9	42.51/42.21	40.03/39.77	**52.43**/51.43	53.34/51.91

(a) Performance on all tasks utilizing *complete data setting.*

Method	HR → LR	Class → Class	Non-Class → Class	QA → QA	Non-QA → QA	Non-NLI → NLI	Non-Para → Para
No ICL	33.09	33.3	33.3	38.85	38.85	25.89	33.44
Raw LM	36.68/35.98	34.95/34.24	34.95/34.24	39/38.64	39/38.64	32.91/31.41	37.26/35.32
MetaICL	39.52/38.71	38.26/37.59	41.13/40.51	40.04/39.33	**38.85**/38.23	31.51/28.49	34.29/32.78
MAML-2-1	**39.86**/38.82	41.68/39.77	**42.31**/41.44	40.6/40.21	38.09/37.37	**38.61**/33.41	33.14/33.14
MAML-2-4	39.27/38.4	**42.24**/40.91	41.92/40.67	**40.7**/40.1	38.28/37.45	37.69/32.92	**34.43**/34.17
Channel No ICL	36.59	33.98	33.98	38.28	38.28	33.31	46.16
Channel Raw LM	41.95/40.83	45.31/45.09	45.31/45.09	39.97/39.51	39.97/39.51	38.83/37.65	45.2/44.38
Channel MetaICL	45.09/43.83	**48.36**/47.3	46.77/46.23	42.20/41.79	40.60/40.14	44.51/42.7	48.64/47.29
Channel MAML-2-1	**46.16**/45.3	48.27/47.89	47.27/46.49	**42.46**/42.19	40.64/40.25	46.51/45.32	**50**/47.83
Channel MAML-2-4	45.054/43.93	47.89/47.55	**47.24**/46.54	42.32/42.11	**40.74**/40.48	**46.74**/45.05	49.606/48.33

(b) Performance on all tasks utilizing *limited data setting.*

Table 3: For both tables, we report the performance numbers on all the tasks. Rows 1-3 represent the baselines and Rows 4 and 5 represent the performance of MAML settings with 1 and 4 tasks respectively. Similarly, Rows 6-8 represent baselines using the Channel training/inference and Rows 9 and 10 represent MAML settings on Channel models. The numbers are represented as X/Y where X represents the average performance and Y represents the worst case performance. The entries in bold are best-performing models.

Significance Analysis. For all reported results, we pay special care in determining the significance of the result. If both the average and the worst-case performance are better, we adjudge the result significant (bold numbers) based on lower standard deviation.

5.1.1 ***Performance on tasks from unseen domains.*** We report the performance of MAML-2-1 and MAML-2-4 along with the baselines as discussed before in Table 2 (a) and (b) on only unseen domain of tasks. Note that unseen tasks share no task type with the training set as well as sampled from completely disjointed domains.

Complete Data. We observe that *MAML-en-LLM* settings outperform MetaICL on **5 out of the 7** task settings (win rate of **0.71**) for standard models and **4 out of 7** task settings (win rate of **0.57**) for channel models on *complete data setting*. For the task settings where MAML settings do not outperform MetaICL, we observe that Non-Para→Para on standard models underperforms significantly (discussed in Subsection 5.2.3).

Limited Data. Similarly, *MAML-en-LLM* outperforms MetaICL on **4 out of 7** settings (win rate of **0.57**) using both standard and channel models on *limited data setting*. These results demonstrate the efficacy of our approach in low-resource settings. For most other settings *MAML-en-LLM* performs at par with MetaICL.

Takeaway. The outperformance of *MAML-en-LLM* on the unseen task setting implies that our method learns a more generalized parameter set. This is a direct consequence of *MAML-en-LLM* exploring a wider set of parameter space as demonstrated in Figure 1.

5.1.2 ***Performance on all tasks.*** Similar to unseen tasks, we report the performances of MAML-2-1 and MAML-2-4 settings in Table 3 (a) and (b) along with the associated baselines.

Complete Data. *MAML-en-LLM* settings outperform MetaICL on the *complete data settings* for all the tasks on **4 out of 7** (win rate of **0.57**) data settings. Channel models perform much better, beating MetaICL on **5 out of 7** (win rate of **0.71**) data settings.

Limited Data. For the limited data settings, *MAML-en-LLM* settings outperform MetaICL on **6 out of 7** (win rate of **0.85**) task settings on both Standard and Channel models. As before, *MAML-en-LLM* settings significantly outperform or are comparable with the MetaICL counterparts on most settings.

Takeaway. The results on both *complete* and *limited* task settings show that even though *MAML-en-LLM* performs meta-updates once every $2kn$ batches, it is insensitive to amount of train data. This further attests to *MAML-en-LLM*'s efficacy and wide use cases.

5.2 Effect of Task Complexity on Exploration States (Number of Tasks)

5.2.1 ***Task complexity.*** We preface the discussion around exploration states by discussing the exact task settings. Note that we have a total of seven settings out of which - two are classification, one is natural language inference (NLI) which is similar to classification, two are question answering (QA), and one is paraphrasing while the last one is a mixture of all of them characterized only by the amount of data. We designate the set of Classification and

Method	HR → LR	Class → Class	Non-Class → Class	QA → QA	Non-QA → QA	Non-NLI → NLI	Non-Para → Para
MetaICL	42.79	38.15	**40.49**	60	38.43	80.23	**61.75**
MAML-2-1	34.44	**42.86**	34.14	**61.25**	38.75	**83.75**	32.79
MAML-2-4	**44.65**	41.1	33.41	58.75	**40.31**	66.24	34.05
Channel MetaICL	50.82	**49.76**	44.71	**57.81**	46.87	44.86	55.43
Channel MAML-2-1	53.61	47.62	47.88	55.62	48.12	**60.68**	**59.86**
Channel MAML-2-4	**54**	48.32	**48.34**	56.25	**48.12**	59.28	58.44

Table 4: Results on very few-shot adaptation on unseen domains of tasks on standard models trained with MetaICL (Row-1), MAML (Rows-2,3) and channel models trained with MetaICL (Row-4), MAML (Rows-5,6).

NLI tasks as *less-complex* while Paraphrasing and QA are *more-complex*. As a direct consequence to task splits, *more-complex* tasks should ideally require a wider parameter space exploration while *less-complex* tasks should require relatively less exploration.

5.2.2 ***Complexity directly affects performance of various exploration states.*** We utilize *MAML-en-LLM* on 2 settings - with 1 task and 4 tasks represented by MAML-2-1 and MAML-2-4, respectively. The more the number of tasks, the more parameter space is explored by the model before performing the meta-update. However, on the flip side, more exploration leads to slower convergence to the minima. The trend we observe is that more number of tasks (MAML-2-4) benefit more-complex settings like QA and Paraphrasing, while a smaller number of tasks (MAML-2-1) help less-complex settings like classification and NLI. (Table 3a,3b). We report results on the validation set in the Appendix.

5.2.3 ***Further discussion on performance.*** Even though the task settings are chosen keeping in mind the non-overlap of tasks, not all tasks are created equal. We believe the choice behind the task selection and splits is not explored in detail in [13]. We attempt to shed light on why MetaICL or *MAML-en-LLM* do not perform well on specific tasks like Non-Para→Para. From our experiments, we observe both MetaICL and *MAML-en-LLM* on standard models actually degrade performance from out-of-box pre-trained models (Refer last column in Table 3a). This observation alludes to the fact that standard model training of MetaICL and *MAML-en-LLM* actually causes some forgetting to the out-of-box pre-trained models by disrupting model weights out of box. In other words, pre-trained out-of-box models are already good enough to perform complex tasks like paraphrasing. Hence, task type and design need to be closely monitored to utilize meta-training approaches.

5.2.4 ***Further discussion on training sets.*** Non-QA -> QA and Non-Para->Para require meta-training on a relatively easier subset (Non-QA and Non-Para sets mostly include easier classification and NLI tasks), which makes the test sets significantly challenging. Indeed, Meta-training on these tasks using MetaICL even causes some forgetting as compared to pre-trained models. Please refer to Table 3a where RawLM outperforms MetaICL baseline on challenging data settings. Refer Table 3a where both Non-Para -> Para (37.26 to 33.1) and Non-QA to QA (39 to 36.71) give a degradation in performance. This behavior is mirrored in the original paper as well [13]. Hence, the behavior observed using MAML-en-LLM on these particular settings is not any different from MetaICL itself and requires further analysis - out of the scope of this paper.

5.3 Effect of Optimizer Choice

Multiple works have demonstrated that optimizers play a crucial role in the effective training of LLMs. Stateless optimizers like Stochastic Gradient Descent (SGD) underperform newer adaptive optimizers like AdamW [9] in LLM training. Note that MetaICL[13] utilizes AdamW. However, MAML-en-LLM utilizes a dual optimization problem - which requires two separate optimizers for the inner and outer optimizations. In Table 5, we provide ablation studies with two different optimizers - one stateless (SGD) and one adaptive (AdamW). Due to compute limitations, we sample a 10% subset of the training and test data across two seeds for both MetaICL and MAML-2-1. In row 4, we report the performance utilizing a combination of optimizers for inner and outer optimizations. Under MAML-2-1, when using stateless optimizers (SGD+SGD), we do not see an increase in performance. Next, as the inner optimizer is re-initialized after every meta-update, we use SGD for inner and AdamW for outer - resulting in a significant increase in performance. Note that using a stateless optimizer in the outer loop is identical to utilizing both stateless optimizers and the performance is identical to SGD+SGD. Lastly, we report results on using adaptive optimizers in both inner and outer optimization steps with and without moment parameter sharing (discussed in subsection 3.3).

Method	**Optimizer**	Seed = 10	Seed = 20
NoICL	-	24.9	28.2
RawLM	-	35.2	28.4
MetaICL	SGD	34.6	36.2
	AdamW	34.8	36.0
MAML-2-1	SGD + SGD	36.4	35.6
	SGD + AdamW	40.0	38.6
	AdamW + SGD	36.4*	35.6*
	AdamW + AdamW (d)	40.0	38.6
	AdamW + AdamW	**42.0**	**42.8**

Table 5: Performance over two seeds for NoICL, RawLM, MetaICL and MAML-2-1 methods utilizing various combinations of optimizers. The 'optimizer' column in row 4 uses the notation X+Y, where X and Y are the inner and outer optimizers respectively. Note that (*) represents an identical setting to SGD+SGD. The AdamW+AdamW (d) setting utilizes AdamW without moment sharing. Utilizing adaptive optimizers (AdamW) with parameter sharing in inner and outer optimization yields the best results. (Sampled from Non-NLI → NLI)

5.4 Experiment-2: Very Few Shot Adaptation

The second experiment involves adapting the meta-trained models to unseen domains using only a few samples. **Setup:** We utilize the training set of test tasks for sampling both the adaptation samples and their prompts, ensuring that the same target prompt is never utilized in exemplars. We sample 16 exemplars or 256 sequence lengths whichever is lower for the prompts. We consider a total of 16 adaptation data points. The model is adapted using the adaptation training samples for 16 steps (1 pass over all points) with a learning rate of $1x10^{-7}$ using AdamW. The testing set remains identical to the unseen test tasks mentioned in Table 1.

Takeaway: Generalization to unseen tasks is a paramount problem in LLM literature. We report the results of very few-shot adapted models in Table 4. Note that the starting unadapted standard and channel models are identical to the *complete data setting*. We observe for standard models, *MAML-en-LLM* settings outperform MetaICL on **5 out of 7** tasks (win rate of **0.71**). Similarly, channel models outperform MetaICL on **5 out of 7** tasks (win rate of **0.71**). Once again we observe that generalization to unseen domains of tasks is better captured by *MAML-en-LLM* as the model explores wider parameter space during training, thus learning better parameter initializations for adaptation. This behavior has been well documented in Meta-learning literature like [2, 6].

5.5 Runtime Analysis

In this section, we detail some practical considerations to keep in mind during meta-training models using *MAML-en-LLM*. For reference, Figure 1 details the training procedure schematically.

- **Model sizes**: Let us assume the size of the computation graph is in order of the number of parameters of the model in question. Assume $S(\theta) = c * O(\theta) = O(\theta)$ to be the size of the computational graph during training. For MetaICL, as there is no adaptation phase, the gradients are computed only once. Hence at any given instance, the maximum memory utilization of the computational graph is $2 * S(\theta)$, where the only values are the present parameter state and the gradients. However, for *MAML-en-LLM*, after the adaptation step the number of parameters in the computational graph are $S(\theta)$ (unadapted params), $k * S(\theta)$ (k is number of tasks), $S(\theta)$ (meta-update) which is a total of $(k + 2) * S(\theta)$ parameters. Hence the memory utilization is a linear function of the number of tasks. Thus, it is important to control the number of tasks as per the meta-training strategy and the type of tasks themselves. For our purpose, GPT-2 Medium with 355 million parameters is used.
- **Optimizer states** : *MAML-en-LLM* as opposed to MetaICL utilizes 2 optimizers for the adaptation and the meta-update steps respectively. Even though it might be tempting to utilize state-less optimizers like SGD and Adam, LLM training has been shown to work better when the updates are conditioned on the last gradient state such as with adaptive optimizers like AdamW [9]. Hence, we utilize AdamW for both adaptation and meta-updates (Optim 1 and Optim 2 in Figure 1). However, after each update step, we copy the last optimizer state back and forth to both optimizers to smoothen out the training procedure.

6 LIMITATIONS

We acknowledge that our method has a few major limitations in practice. We discuss the limitations in detail below:

- **Unstable Training/Performance:** Due to the dual optimization procedure of MAML-based approaches, the training procedure is bumpy and non-smooth, making the training and validation losses spiky. This observation has been noted by [2]. This makes the training of large models even more unstable and care must be taken while training employing smoothing methods like early stopping, gradient clipping, etc.
- **Sensitive Hyperparameters:** The training procedure is extremely sensitive to learning rate, warmup and decay.
- **Catastrophic Forgetting:** In some exceptional cases, our method can cause catastrophic forgetting of the model parameters and underfit on complex use-cases, as observed in [13] as well. Further investigation into this behavior is required.
- **Runtime Complexity:** As always, the runtime complexity of MAML is high. We recommend users to carefully consider tradeoffs before utilizing MAML-en-LLM.
- **Task complexity:** Based on our observations, we recommend utilizing MAML-2-1 for tasks that benefit from faster convergence and low-param space exploration (Classification, NLI) which are less complex than tasks that require larger param space exploration (Paraphrasing, QA)

7 CONCLUSION

In this paper, we proposed a novel method *MAML-en-LLM* that meta-trains pre-trained out-of-box models using the principles proposed in meta-learning literature. We demonstrated that *MAML-en-LLM* explores a much wider parameter space than current SOTA meta-training methods like MetaICL and MetaICT due to adaptation to multiple sets of parameters before the actual meta-update. Empirically, *MAML-en-LLM* outperforms MetaICL on both standard and channel models on an extensive set of tasks in both seen and unseen domains. Subsequently, we also demonstrate that models trained using *MAML-en-LLM* can be quickly adapted in a few-shot manner to a set of tasks in the unseen domain. Overall, *MAML-en-LLM* has been empirically demonstrated to outperform MetaICL on performance and generalization. We hope our study motivates the community to utilize classical meta-learning principles in the meta-training of LLMs in the future.

REFERENCES

[1] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2023. What learning algorithm is in-context learning? Investigations with linear models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/pdf?id=0g0X4H8yN4I

[2] Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. 2019. How to train your MAML. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=HJGven05Y7

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via Language Model In-context Tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav

Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 719–730. https://doi.org/10.18653/v1/2022.acl-long.53

[5] Budhaditya Deb, Ahmed Hassan Awadallah, and Guoqing Zheng. 2022. Boosting Natural Language Generation from Instructions with Meta-Learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 6792–6808. https://doi.org/10.18653/v1/2022.emnlp-main.456

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[7] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing Format Boundaries With a Single QA System. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020 (Findings of ACL, Vol. EMNLP 2020)*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 1896–1907. https://doi.org/10.18653/v1/2020.findings-emnlp.171

[8] Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. 2022. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458* (2022).

[9] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[10] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 8086–8098. https://doi.org/10.18653/v1/2022.acl-long.556

[11] Florian Lux and Ngoc Thang Vu. 2022. Language-Agnostic Meta-Learning for Low-Resource Text-to-Speech with Articulatory Features. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 6858–6868. https://doi.org/10.18653/v1/2022.acl-long.472

[12] Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy Channel Language Model Prompting for Few-Shot Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 5316–5330. https://doi.org/10.18653/v1/2022.acl-long.365

[13] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. MetaICL: Learning to Learn In Context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2791–2809.

[14] Chengwei Qin, Shafiq Joty, Qian Li, and Ruochen Zhao. 2023. Learning to Initialize: Can Meta Learning Improve Cross-task Generalization in Prompt Tuning? *arXiv preprint arXiv:2302.08143* (2023).

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[16] Ling Xiao Wang, Kevin Huang, Tengyu Ma, Quanquan Gu, and Jing Huang. 2021. Variance-reduced first-order meta-learning for natural language processing tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2609–2615.

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[18] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. 2023. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846* (2023).

[19] Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A Few-shot Learning Challenge for Cross-task Generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 7163–7189. https://doi.org/10.18653/v1/2021.emnlp-main.572

[20] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. 2020. Meta-Learning without Memorization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=BklEFpEYwS

[21] Kang Min Yoo, Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang-goo Lee, and Taeuk Kim. 2022. Ground-Truth Labels Matter: A Deeper Look into Input-Label Demonstrations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 2422–2437. https://doi.org/10.18653/v1/2022.emnlp-main.155

A APPENDIX

Table 6 details the validation performance (computed over 5 seeds) for the *MAML-en-LLM* standard and channel models. The validation test is selected from the training set of the test tasks. We see that the performance agrees with the test tasks.

	Seed	MAML-2-1	MAML-2-4	Channel-2-1	Channel-2-4
HR → LR	100	72.4	70.82	70.62	71.91
	13	69.01	64.86	69.15	69.96
	21	67.52	63.46	73.55	67.74
	42	68.07	65.36	72.83	70.38
	87	70.1	62.83	73.33	73.02
Class → Class	100	48.04	51.84	50.48	61.25
	13	47.85	47.47	43.32	60.95
	21	48.36	48.33	41.46	62.17
	42	46.5	46.61	50.29	58.59
	87	42.64	40.95	51.65	62.71
Non-Class → Class	100	62.39	61.2	59.99	61.91
	13	56	55.38	59.66	61.95
	21	56.51	52.08	61.72	65.06
	42	59.09	53.84	58	63.53
	87	57.28	51.84	58.81	60.37
QA → QA	100	78.125	77.27	66.19	83.8
	13	78.125	78.12	73.01	80.11
	21	74.71	66.76	73.86	81.25
	42	77.55	71.02	69.88	81.81
	87	74.43	69.03	69.88	79.26
Non-QA → QA	100	75.56	76.13	67.89	82.67
	13	78.4	76.7	72.44	79.54
	21	79.54	72.72	75.56	80.39
	42	73.86	67.89	72.44	82.95
	87	75.85	65.62	72.44	76.7
Non-NLI → NLI	100	32.61	34.58	31.16	46.04
	13	34.15	44.48	37.43	44.72
	21	45.15	39.54	32.53	49.86
	42	40.19	34.07	26.84	43.76
	87	39.58	46.74	43.35	44.82
Non-Para → Para	100	55.3	50.55	38.12	46.12
	13	42.96	38.03	45.94	50.54
	21	53.12	45.52	32.07	58.39
	42	38.5	50.39	43.5	43.33
	87	41.41	46	38.94	64.72

Table 6: Validation performance on the training set of test task on Standard and Channel models.