ADVANTAGES, RISKS AND INSIGHTS FROM COMPARING IN-CONTEXT LEARNING MODELS WITH TYPICAL META-LEARNERS

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate in-context learning (ICL) models from the perspective of learning to learn. Unlike existing works understanding what exact explicit learning algorithms can and do ICL models learn, we compare ICL model with typical meta-learners to obtain end-to-end understanding on more general settings. We theoretically prove its expressiveness as learning algorithms and investigate its learnability and generalizability on extensive settings. It is demonstrated that ICL with transformers can effectively learn optimal learning algorithms data-dependently in an inclusive space containing existing gradient-based, metric-based and amortization-based meta-learners. However, the generalizability of these learning algorithms is identified to be a critical issue, as the learned algorithm could be implicitly fitting the training distribution rather than an explicit learning algorithm. Based on the above understanding, we propose to systematically transfer deep-learning techniques which have been widely-studied in supervised-learning to meta-learning to address their common challenges. We demonstrate meta-level meta-learning for domain-adaptability with few data and meta-level curriculum learning for fast convergence in pre-training as examples, showing their empirical effectiveness.

1 INTRODUCTION

Large Language Models (LLMs) Achiam et al. (2023) have witnessed remarkable progress in recent years. Apart from traditional natural language processing tasks such as machine translation as sentiment analysis, LLMs have gained prominence in solving more complex tasks by understanding instructions and examples from human's input, and generate coherent, human-like text. LLMs use in-context learning (ICL) (Brown, 2020) to understand and generate responses based on the input text. Given a prompt containing examples (input-output pairs) from a task and a query input, ICL allows the LLM to generate the corresponding output without altering their weights. For example, given "*happy* -> *positive; sad* -> *negative; blue* ->", the model can output "*negative*", while given "*green* -> *cool; yellow* -> *warm; blue* ->" the model can output "*cool*". Formally, ICL can be formulated as given input $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \cdots, \mathbf{x}^{(n)}, \mathbf{y}^{(n)}, \mathbf{x}^{(n+1)})$, where there is an underlying task f that $\mathbf{y}^{(i)} = f(\mathbf{x}^{(i)})$, the model outputs the prediction of $f(\mathbf{x}^{(n+1)})$. Through pre-training by simulating the above behavior over a distribution over f, the ICL model can generalize to unseen tasks.

The remarkable performance of LLMs across a wide range of applications has garnered significant attention toward understanding how their ICL ability is acquired and executed. However, ICL has so far been well-understood only in highly simplified settings: linear-transformer trained on linear regression tasks. In such cases, the model is shown to precisely learn to perform pre-conditioned gradient descent based on input examples, with explicit weights corresponding to the global minimum during pretraining (Von Oswald et al., 2023; Mahankali et al., 2024; Ahn et al., 2023; Gatmiry et al., 2024). However, this setting is so simplified that it is far from real-world scenarios, and no complex setting enjoys such a transparent understanding. Towards more generalizable understanding of ICL, there are efforts from different perspectives, including theories results of expressiveness (Wang et al., 2024; Bai et al., 2023), learning dynamics and convergence (Tian et al., 2023; Li et al., 2023b; Huang et al., 2024; Zhang et al., 2024; Sander et al., 2024), generalization error (Li et al., 2023a; 2024; Wies

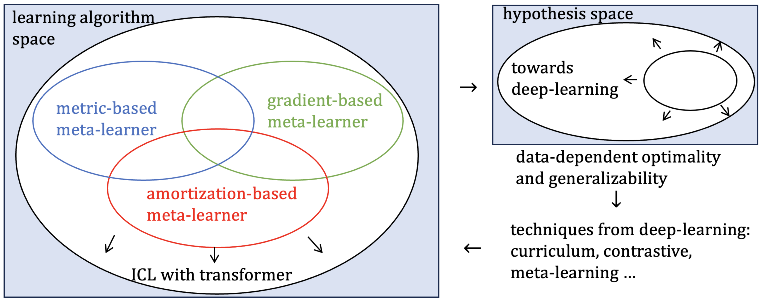

learning algorithm space
metric-based meta-learner
gradient-based meta-learner
amortization-based meta-learner
ICL with transformer
hypothesis space
towards deep-learning
data-dependent optimality and generalizability
techniques from deep-learning: curriculum, contrastive, meta-learning ...

Figure 1: In this paper, we begin by proving that ICL with transformer is expressive enough to encompass typical meta-learners in the learning algorithm space. We then demonstrate that ICL exhibits meta-level deep learning properties, allowing deep learning techniques to be effectively adapted to the meta-level to enhance ICL.

et al., 2024), and observations of ICL model's behaviors (Akyürek et al., 2023; Bhattamishra et al., 2024; Zhang et al., 2023).

Although exactly understanding what and how do the ICL models learn from pre-training is challenging and specific to various problem setting and data distribution, basic consensuses have been reached that: the ICL model learns a learning algorithm mapping $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \cdots, \mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ to f through pre-training, and the inference process is interpreted as first learning the function f from $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \cdots, \mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ and then applying it to $\mathbf{x}^{(n+1)}$. The above consensuses describe the nature that ICL model is meta-learner (Kirsch et al., 2022; Dai et al., 2023), which is to learn a learning algorithm to enable a learning system to quickly adapt to new tasks, i.e., learning to learn (Schmidhuber, 1987; Thrun & Pratt, 1998). Given tasks for meta-training (pre-training), it aims at learning a learner function (i.e., learning algorithm) that makes inference of certain input according to a given set of labels examples, then it can generalize to meta-testing (unseen) tasks. As typical meta-learners have been widely studied but no one has shown successful general intelligence as LLMs (ICL models) do, a nature question is: what distinguishes ICL models from typical meta-learners? While existing works towards understanding ICL take efforts to answer what exact learning algorithm ICL model learns, we try to answer:

Why can ICL models be prominent compared with typical meta-learners?

The seeming difference between ICL models and typical meta-learners is their hypothesis spaces. ICL has been described as the outcome of meta-learning with minimal inductive bias (Kirsch et al., 2022). The basic hypothesis of meta-learners is function with two inputs: a support set containing labeled examples, and a query input, outputting the prediction of query. Meta-training such a black-box model with sufficient tasks can lead to general ICL ability. The above difference can lead to the prominence through the profit of data-driven expelling human-designed knowledge, as in many fields of machine learning, the success of learning with less inductive bias, i.e., training a deep black-box model can attribute to this (LeCun et al., 2015). Human-designed knowledge is achieved through past experience, which can not necessarily be correct or helpful to the target problem, while data-driven knowledge is optimized through the training data, and could work well on the target problem when the hypothesis space is expressive enough and generalizability is certified. Such characteristics also exists in the hypothesis of meta-learners, about knowledge to determine a learning algorithm. Typical meta-learners exactly define, or give strong prior knowledge by human design to the algorithm structure about, i.e., how to utilize support examples and predict the query. ICL models only keep the basic hypothesis with a black-box model, where transformer (Vaswani, 2017) is a feasible choice, as each layer it permits black-box interaction among samples and can be stacked to deep architecture, while keeping certain necessary inductive bias like being aware of the support labels and identifying the query through tokenization, and permutation-invariance among support examples, enhancing generalizability. Thus we conjecture that ICL models' prominence attribute to learning optimal learning algorithms in an inclusive space, while the optimality is data-dependent[1] indicating risks in generalizability.

In the sequel, we first verify above conjecture, reaching conclusions that ICL model can, and does learn data-dependent optimal algorithm, but has limited generalizability showing distribution-sensitive

---

[1] The formal definition of a optimal learning algorithm and a data-dependent optimal learning algorithm is provided in Appendix C

performance when the algorithm is implicit. Then based on this understanding, we point out common challenge and expectation of training deep-models in supervised-learning and pre-training ICL models. Thus we provide insights to transfer mature deep-learning techniques to improve ICL through a systematical mapping from supervised-learning to meta-learning. Related works are discussed in Appendix A.

Our contributions can be summarized as following:

- We investigate ICL model from a learning to learn perspective, by investigating its expressiveness, learnability and generalizability as a meta-learner, and achieve to provide a unified understanding of ICL accommodating existing works.
- We theoretically prove that ICL with transformer is expressive enough to perform existing categories of meta-learning algorithm, and show it does construct data-dependent optimal algorithms on extensive settings, and investigate its generalizability falsifying the existing interpretation as "algorithm selection".
- We propose to improve ICL by systemically transferring deep-learning techniques to meta-level through a mapping between supervised-learning and meta-learning. As examples, we practice to improve domain-adaptability of ICL model by pre-training with meta-level meta-learning, and fast convergence by pre-training with meta-level curriculum learning, and show their empirical effectiveness.

2 PRELIMINARIES: LEARNING TO LEARN

A learning algorithm (Kirsch et al., 2022) is considered as a mapping from a labeled dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$ and a query input $\mathbf{x}^{(q)}$ to a prediction $\hat{\mathbf{y}}^{(q)}$. The function of a learning algorithm can all be represented as a learner function g:

$$\hat{\mathbf{y}}^{(q)} = g(\mathbf{x}^{(q)}, \mathcal{D}). \tag{1}$$

Learning to learn (Vilalta & Drissi, 2002; Hospedales et al., 2021) is also called meta-learning, which aims to optimize a learnable $g(;\theta)$ through meta-training. Training ICL models or other meta-learners is learning to learn.

2.1 IN-CONTEXT LEARNING WITH TRANSFORMER

Generally, there is a input matrix Z_0 composed of $\mathcal{D}$ and $\mathbf{x}^{(q)}$, which is fed into a M-layer transformer TF_M. Denote the collection of all model weights in TF_M as θ_M. ICL's function in (12) can be represented as a learner function g_M:

$$g_M(\mathbf{x}^{(q)}, \mathcal{D}; \theta_M) = \mathsf{TF}_M(Z_0; \theta_M), \tag{2}$$

where details of construction of Z_0, model architecture of TF_M, and optimizing θ_M are provided in Appendix B.1.

2.2 TYPICAL META-LEARNING

Typical meta-learners are more restricted to certain learning algorithm frameworks designed by human experts. People introduce strong inductive bias into $g(;\theta)$, i.e., how to adapt to $\mathcal{D}$ and make inference of $\mathbf{x}^{(q)}$. Typical meta-learners can be generally categorized into three categories (Bronskill et al., 2021): gradient-based, metric-based and amortization-based. The function of each category can be summarized as follows.

Gradient-Based. Given a prediction model $h : \mathbf{x} \mapsto \mathbf{y}$ and a loss function $\ell(\cdot, \cdot)$, gradient-based meta-learners (Finn et al., 2017) performs gradient-descent with labeled data in $\mathcal{D}$:

$$g_{gd}(\mathbf{x}^{(q)}, \mathcal{D}; \theta) = h(\mathbf{x}^{(q)}; \theta - \sum\nolimits_{i=1}^{n} \nabla_\theta \ell(h(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})). \tag{3}$$

Metric-Based. Metric-based learners (Koch et al., 2015; Garcia & Bruna, 2018; Sung et al., 2018) learn to compare query with examples by optimizing a distance metric in the feature space. Denote a distance function $d_\theta(\cdot,\cdot)$. Pair-wise metric-based algorithm makes prediction based on pair-wise distance between query and examples:

$$g_{sim}(\mathbf{x}^{(q)}, \mathcal{D}; \theta) = \frac{1}{n} \sum_{i=1}^{n} d_\theta(\mathbf{x}^{(i)}, \mathbf{x}^{(q)}) \mathbf{y}^{(i)}. \tag{4}$$

While for classification tasks where $\mathbf{y}^{(i)} \in \{\boldsymbol{c}_c\}_{c=1}^{C}$, one can also adopt class-prototype metric-based algorithm (Snell et al., 2017), comparing the query and prototype $\boldsymbol{c}_c$ of each class c:

$$g_{prt}(\mathbf{x}^{(q)}, \mathcal{D}; \theta) = \sum_{c=1}^{C} d_\theta(\frac{1}{n_c} \sum_{y^{(i)}=c} \mathbf{x}^{(i)}, \mathbf{x}^{(q)}) \boldsymbol{c}_c. \tag{5}$$

Amortization-Based. Amortization-based meta-learners (Garnelo et al., 2018) are also called as black-box meta-learners. As it also considers to meta-train a black-box model to learn the learning algorithm, it is much closer to ICL model as meta-learner. However, typical amortization-based methods follows the framework of using a set encoder (Zaheer et al., 2017) to map $\mathcal{D}$ to a vector $\mathbf{e}$ as the task context and then feed the context and the query input to a prediction model $f_\theta : (\mathbf{x}, \mathbf{e}) \mapsto \mathbf{y}$. Considering the universal approximation property of neural networks, amortization-based meta-learner can be formulated as:

$$g_{am}(\mathbf{x}^{(q)}, \mathcal{D}; \theta) = f_\theta(\mathbf{x}^{(q)}, \frac{1}{n} \sum_{i=1}^{n} [\mathbf{x}^{(i)} | \mathbf{y}^{(i)}]). \tag{6}$$

3 EXPRESSIVENESS OF ICL WITH TRANSFORMER AS LEARNING ALGORITHMS

Expressiveness in deep-learning refers to a model's ability to capture complex patterns and relationships within data (LeCun et al., 2015), which is a fundamental property that gives the chance to deep-learning models achieving high performance in intricate tasks. We focus on expressiveness on meta-level, about the ability to capture interaction patterns and relationships among samples in $\mathcal{D}$ and $\mathbf{x}^{(q)}$ rather than features, to express learning algorithms. In this section, we prove that ICL with transformer is expressive enough that it can perform all typical categories of meta-learners.

Specifically, we theoretically prove that with certain parameter instantiations, ICL with transformer g_M (2) can perform gradient-based g_{gd} (3), pair-wise metric-based g_{sim} (4), class-prototype metric-based g_{prt} (5) and amortization-based g_{am} (6). As class-prototype metric-based methods only can be used for classification tasks, we consider the standard C-class classification tasks. The detail settings of the task and mild assumptions for this part are provided in Appendix D.1. Formally, we have the following theorems for classification problems where $C < \infty$:

Theorem 3.1. $\forall\, \theta \in \mathbb{R}^{|\theta|}$, $\exists\, M \in \mathbb{N}^* < \infty$, $\exists\, \theta_M \in \mathbb{R}^{|\theta_M|}$, $g_M(\mathbf{x}^{(q)}, \mathcal{D}; \theta_M) = g_{gd}(\mathbf{x}^{(q)}, \mathcal{D}; \theta)$.

Theorem 3.2. $\forall\, \theta \in \mathbb{R}^{|\theta|}$, $\exists\, M \in \mathbb{N}^* < \infty$, $\exists\, \theta_M \in \mathbb{R}^{|\theta_M|}$, $g_M(\mathbf{x}^{(q)}, \mathcal{D}; \theta_M) = g_{sim}(\mathbf{x}^{(q)}, \mathcal{D}; \theta)$.

Theorem 3.3. $\forall\, \theta \in \mathbb{R}^{|\theta|}$, $\exists\, M \in \mathbb{N}^* < \infty$, $\exists\, \theta_M \in \mathbb{R}^{|\theta_M|}$, $g_M(\mathbf{x}^{(q)}, \mathcal{D}; \theta_M) = g_{prt}(\mathbf{x}^{(q)}, \mathcal{D}; \theta)$.

Theorem 3.4. $\forall\, \theta \in \mathbb{R}^{|\theta|}$, $\exists\, M \in \mathbb{N}^* < \infty$, $\exists\, \theta_M \in \mathbb{R}^{|\theta_M|}$, $g_M(\mathbf{x}^{(q)}, \mathcal{D}; \theta_M) = g_{am}(\mathbf{x}^{(q)}, \mathcal{D}; \theta)$.

Proof Sketch The proof of Theorem 3.1~3.4 are achieved by programming the functions of typical learners into $M \in \mathbb{N}^* < \infty$ conditioned steps, where each step can be achieved through one transformer layer with the following two basic tools:

1. Universal approximation property of multi-layer perceptron (MLP) (Hornik et al., 1989): this allows the feed-forward layers, express a wide range of functions $\mathbb{R}^{dim_1} \to \mathbb{R}^{dim_2}$. In each transformer layer, there is a module of feed-forward layers, functions on each column of input matrix independently, thus a wide range of sample-wise transformations which exist in every layer of the transformer, can achieve a wide range of sample-wise transformations.
2. Orthonormal label embedding in $\mathbb{R}^{2C}$: we use a set of orthonormal vectors in $\mathbb{R}^{2C}$ as embeddings of the categorical labels (including the query identifier), e.g., one-hot embeddings. This allows the attention weight matrix $A \in \mathbb{R}^{(n+1)\times(n+1)}$ in the self-attention module in each transformer layer, which weight the interaction weights among all samples, can be *label-aware*. The *label-aware* means $\{A \in \mathbb{R}^{(n+1)\times(n+1)} \mid (\mathbf{y}^{(i)} = \mathbf{y}^{(i')}) \wedge (\mathbf{y}^{(j)} = \mathbf{y}^{(j')}) \Rightarrow A_{i,j} = A_{i',j'}\}$, i.e., the interaction weight between ordered sample-pair (i, j) only depends on their labels $(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ (also aware of unknown query), and can be arbitrary value in $\mathbb{R}^*$. Thus achieving the nature of learning algorithm as label-aware set function.

The programming into finite conditioned steps is specific to each theorem are not necessarily unique. The full proof is provided in Appendix D.

4 ICL Model Does Learn Data-Dependent Optimal Algorithm

We have shown that ICL model is expressive by proving that its hypothesis space is inclusive at least to contain hypothesis spaces of typical meta-learners. But what solution in the space does ICL model achieve through pre-training with certain task set, i.e., what learning algorithm does ICL model actually learn, directly determines its performance and generalizability. This must be considered to investigate its prominence. In this section we investigate with sufficient pre-training , if ICL model learns optimal learning algorithm on training tasks, and what generalizability it has when we do not explicitly know what learning algorithm it is.

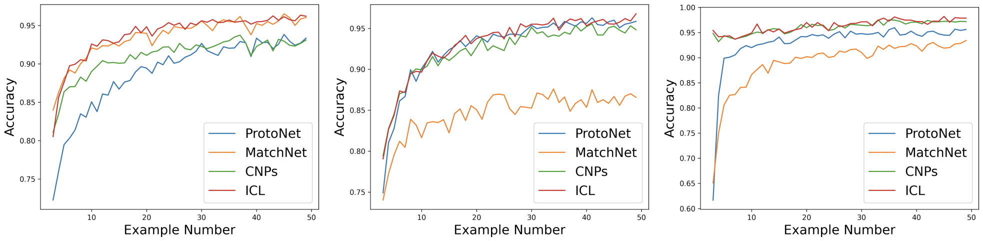
Algorithm Criterion. To determine if two learning algorithms are the same, specifically for classification tasks, the classification boundary can be manifested through Monte-Carlo sampling query inputs and observed. With multiple trials given different sets of labeled examples, if two learners functions always shows the same classification boundary, together with observing they have the same end-to-end performance, we can infer that they are the same learning algorithm.

Generalizability of Learning Algorithm. We use **explicit/implicit** optimal learning algorithm to distinguish the generalizability of a learning algorithm. Formally, we define explicit optimal algorithm $g(;\mathcal{F},*)$ of a function family $\mathcal{F}$ as: when $n \to \infty$, $\forall f \in \mathcal{F}$, $\forall p(x)$, $\mathbb{E}_{p(x)}[g(\mathbf{x}^{(q)}, \mathcal{D}; \mathcal{F}, *)] = f(\mathbf{x}^{(q)})$, where $\mathcal{D} = \{(\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)}))\}_{i=1}^{n}$, $\mathbf{x}^{(q)} \sim p(x)$, $\mathbf{x}^{(i)} \sim p(x)$. Which is to say an explicit learning algorithm is generalizable over any distribution constrained by a $f \in \mathcal{F}$. On the contrary, implicit optimal learning algorithms are distribution-sensitive. And we denote $\boldsymbol{\mathcal{G}}_{\boldsymbol{\Omega}}$ as the set of all ground truth explicit optimal algorithms for a task set $\boldsymbol{\Omega}$. For example, Ordinary Least Squares is an explicit optimal learning algorithm $g(;\mathcal{F},*)$ for linear regression tasks ($\mathcal{F} = \{f \mid f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}\}$), while memorizing and looking-up is an implicit optimal learning algorithm for any problem.

4.1 Generating Tasks with Explicit Optimal Algorithms

To show if ICL with transformer does learn the optimal learning algorithm, we consider generating tasks whose optimal prediction can be exactly achieved by some explicit learning algorithms. For specific algorithms, we consider a representative from each categories of typical meta-learners, g_{sim}, g_{prt} and g_{am}. Specifically, denoting a set of tasks $\boldsymbol{\Omega} = \{\mathcal{D}_\tau\}_{\tau=1}^{T}$, we generate three types of tasks: pair-wise metric-based tasks $\boldsymbol{\Omega}_{sim}$ where MatchNet Vinyals et al. (2016) ($\in g_{sim}$) is the optimal learner, class-prototype metric-based tasks $\boldsymbol{\Omega}_{prt}$ where ProtoNet Snell et al. (2017) ($\in g_{prt}$) is the optimal learner, amortization-based tasks $\boldsymbol{\Omega}_{am}$ where CNPs Garnelo et al. (2018) ($\in g_{am}$) is the optimal learner. We do not consider g_{gd} for two reasons: it is hard to define a family of classification tasks and the corresponding h to guarantee the optimum; proving ICL can express g_{gd} is not considered as contribution of this paper as it is trivial by applying the result from Bai et al. (2023); Wang et al. (2024). The details of generating tasks and experiment settings are provided in Appendix E.

4.2 ICL Model Learns Explicit Optimal Algorithm on Simple Tasks

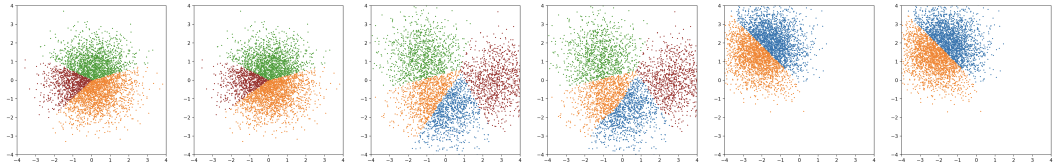


(a) Pair-wise metric-based. (b) Class-prototype metric-based. (c) Amortization-based.

Figure 2: Meta-testing performance of learners meta-trained and tested on the same single type of tasks.

First to verify that ICL with transformer does learn the optimal algorithm, we perform meta-training with single type of tasks corresponding to one explicit optimal algorithm, thus we can draw the conclusion by comparing the classification boundaries of trained ICL model and the optimal algorithm. We also add parameterized feed-forward layers in the above-mentioned optimal meta-learners, to meta-train them together with ICL model, so that we can compare their end-to-end performance. We investigate with the above three types of tasks respectively.

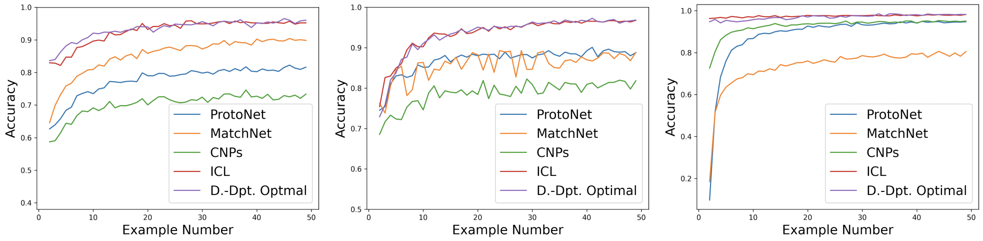
Figure 2(a) shows the end-to-end performance of learners trained on $\mathbf{\Omega}_{sim}$ and testing on $\mathbf{\Omega}'_{sim}$(unseen), where we can find ICL's end-to-end performance is only marginally different with MatchNet, the parameterized optimal meta-learner through meta-training. We also visualize how ICL classifies each sample given few fixed labeled examples and show an example in Figure 3(a) and 3(b) (more cases are provided in Appendix F.1), verifying ICL model does learn the same algorithm as MatchNet, which is optimal for $\mathbf{\Omega}_{sim}$. So does that ICL model learns ProtoNet on $\mathbf{\Omega}_{prt}$ which is optimal, shown as from Figure 2(b), 3(c) and 3(d), and ICL model learns CNPs on $\mathbf{\Omega}_{am}$ which is optimal, shown as from 2(c), 3(e) and 3(f). Thus we have: pre-training with $\mathbf{\Omega}_{sim}$, $\mathbf{\Omega}_{prt}$ or $\mathbf{\Omega}_{am}$ respectively, **ICL model learns explicit optimal algorithm**.



(a) True label of task $\tau_1 \in \mathbf{\Omega}'_{sim}$. (b) ICL prediction of τ_1. (c) True label of task $\tau_2 \in \mathbf{\Omega}'_{prt}$. (d) ICL prediction of τ_2. (e) True label of task $\tau_3 \in \mathbf{\Omega}'_{am}$. (f) ICL prediction of τ_3.

Figure 3: Comparing ICL's predictions and true labels on pair-wise metric-based, class-prototype metric-based and amortization-based tasks. Results of more trials are provided in Appendix F.1.

4.3 ICL MODEL LEARNS IMPLICIT OPTIMAL ALGORITHM ON MIXED TASKS



(a) Pair-wise metric-based tasks. (b) Class-prototype metric-based. (c) Amortization-based.

Figure 4: Meta-testing performance of learners meta-trained on hybrid tasks (testing on each seen type of tasks respectively).

More close to the real world scenario which is complex, the pre-training tasks are from various types that they do not share an optimal explicit learning algorithm. We investigate what learning algorithm does ICL model learn under this setting. Note that though we can always define an implicit algorithm that is optimal for all pre-training tasks (e.g., memorization and looking up), such data-dependent optimal algorithm would show very limited generalizability and raise challenge to the expressiveness of ICL model.

Specifically, we mix the above $\mathbf{\Omega}_{sim}$, $\mathbf{\Omega}_{prt}$ and $\mathbf{\Omega}_{am}$ to form meta-training task set $\mathbf{\Omega}_{mix}$, i.e., $\mathcal{G}_{\mathbf{\Omega}_{mix}} = \{\text{MatchNet, ProtoNet, CNPs}\}$. We meta-train ICL model, MatchNet, ProtoNet and CNPs with $\mathbf{\Omega}_{mix}$, and evaluate their performance on unseen $\mathbf{\Omega}'_{sim}$, $\mathbf{\Omega}'_{prt}$ and $\mathbf{\Omega}'_{am}$ respectively. Figure 4 shows the results. We also compare with the performance of data-dependent optimal algorithm (D.-Dpt. Optimal) about each type of testing tasks respectively: D.-Dpt. Optimal in Figure 4(a) means MatchNet trained with $\mathbf{\Omega}_{sim}$ testing with $\mathbf{\Omega}'_{sim}$; in Figure 4(b) means ProtoNet trained with $\mathbf{\Omega}_{prt}$ testing with $\mathbf{\Omega}'_{prt}$; in Figure 4(a) means CNPs trained with $\mathbf{\Omega}_{am}$ testing with $\mathbf{\Omega}'_{am}$. We also visualize ICL's classification boundaries on testing tasks, which show they are the same with the optimal ones, like the patterns shown in Figure 10∼12, provided in Appendix F.2.

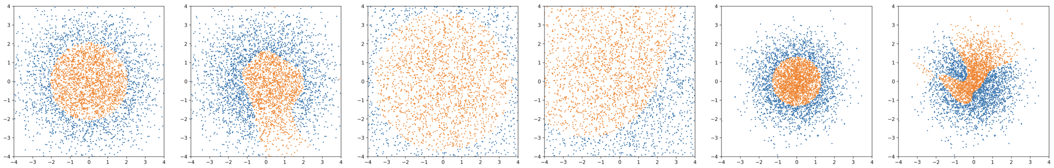
Observing the above results, we have following conclusions: (i) meta-learners trained on $\mathbf{\Omega}_{mix}$ (MatchNet, ProtoNet, CNPs) all fail to reach the data-dependent optimal (D.-Dpt. Optimal) performance on $\mathbf{\Omega}'_{sim}$, $\mathbf{\Omega}'_{prt}$ or $\mathbf{\Omega}'_{am}$, which means none of the above explicit learning algorithm can be optimal for $\mathbf{\Omega}_{sim}$, $\mathbf{\Omega}_{prt}$ and $\mathbf{\Omega}_{am}$ simultaneously; (ii) ICL model trained on $\mathbf{\Omega}_{mix}$ (ICL) has very similar performance with data-dependent optimal (D.-Dpt. Optimal) ones, together with the same classification boundary in Appendix F.2, we can infer that ICL model does learn data-dependent optimal learning algorithm on $\mathbf{\Omega}_{mix}$; (iii) the above conclusions (i) and (ii) together show that **ICL model learns a data-dependent optimal algorithm which $\notin \boldsymbol{\mathcal{G}}_{\mathbf{\Omega}_{mix}}$**. But we do not know what kind of learning algorithm it is yet.

We concern about the question that if the data-dependent optimal learning algorithm on $\mathbf{\Omega}_{mix}$ is implicit or explicit, which is directly related to the generalizability of ICL model. Existing works have studied ICL model traied with mixed type of tasks to perform "algorithm selection" (Li et al., 2023a; Bai et al., 2023; Bhattamishra et al., 2024; Wang et al., 2024). "Algorithm selection" means among all algorithms which are explicit optimal to certain pre-training tasks, ICL choosing the most suitable algorithm which is optimal based on the specific task context, which can be formally described as trained with $\mathbf{\Omega}_{mix}$:

$$g_M(\mathbf{x}^{(q)}, \mathcal{D}; \theta_M) = g^*(\mathbf{x}^{(q)}, \mathcal{D}), \tag{7}$$
$$s.t.,\ g^* = \text{argmin}_{g\in\mathbf{\Omega}_{mix}} \sum\nolimits_{\mathcal{D}'\subset\mathcal{D}} \sum\nolimits_{(\mathbf{x}^{(i)},\mathbf{y}^{(i)})\in\mathcal{D}'} \ell(g(\mathbf{x}^{(i)}, \mathcal{D}/\mathcal{D}', \mathbf{y}^{(i)}),$$

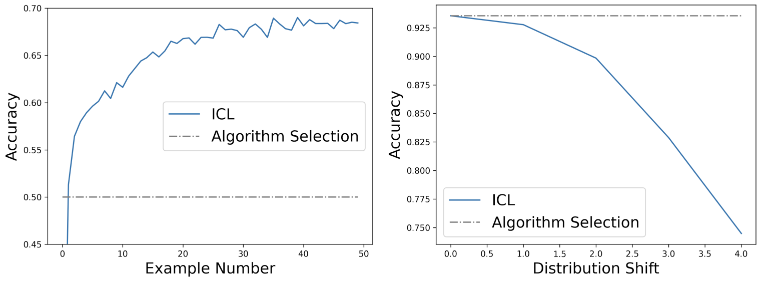
which is an explicit optimal algorithm end-to-end.

However, we question that ICL model learns the above "algorithm selection" algorithm trained on mixed type of tasks. Though this interpretation seems to be reasonable from existing empirical results, it brings questionable generalizability to unseen type of tasks, and out-of-distribution data, which has not been investigated in literature. First, If ICL model trained with $\mathbf{\Omega}_{mix}$ exactly follows (7), it would be a catastrophe for it to handle tasks from novel type which can not be solved by any $g \in \boldsymbol{\mathcal{G}}_{\mathbf{\Omega}_{mix}}$. Second, if it does learn an explicit optimal algorithm, for any tasks from seen types, i.e., it would not be distribution-sensitive that can handle tasks from seen types with data from different distribution consistently well. Following results show that **ICL is not "algorithm selection"**.



(a) True label of task $\tau_1 \in \mathbf{\Omega}'_{rad}$. (b) ICL prediction of τ_1. (c) True label of task $\tau_2 \in \mathbf{\Omega}'_{rad}$. (d) ICL prediction of τ_2. (e) True label of task $\tau_3 \in \mathbf{\Omega}'_{rad}$. (f) ICL prediction of τ_3.

Figure 5: Comparing ICL's predictions and true labels on radial distance tasks.



(a) Performance on radial distance tasks. (b) Performance on tasks from seen types under distribution shift.

Figure 6: Comparing ICL with ideal "algorithm selection".

4.3.1 ICL CAN SOLVE TASKS FROM UNSEEN TYPE

We still consider the ICL model trained with the above $\mathbf{\Omega}_{mix}$, but introduce a novel type of tasks for testing: radial distance tasks $\mathbf{\Omega}_{rad}$. We generate a task by sampling a $r \in \mathbb{R}$ from $p(\tau)$, as the classification boundary. Then sample $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{i=1}^N$ from $p(x)$, and assign label by: $\mathbf{y}^{(i)} = \begin{cases} 0, \ ||\mathbf{x}^{(i)}|| \leq r \\ 1, \ ||\mathbf{x}^{(i)}|| > r \end{cases}$. Note that we especially control $p(x)$ according to r to make sure the labels are balanced in each task. Such radial distance task could not be solved following "algorithm selection", because a radial distance $\mathcal{D}$ could not be learned by any $g \in \boldsymbol{\mathcal{G}}_{\boldsymbol{\Omega}_{mix}} = \{\text{MatchNet}, \text{ProtoNet}, \text{CNPs}\}$ at all, i.e., $\forall g \in \boldsymbol{\mathcal{G}}_{\boldsymbol{\Omega}_{mix}} \ \mathbb{E}_{p(\tau),p(x)}[\sum_{(\mathbf{x}^{(i)},\mathbf{y}^{(i)})\in\mathcal{D}} \ell(g(\mathbf{x}^{(i)}, \mathcal{D}/(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})), \mathbf{y}^{(i)})] = l$, where l is the expectation of loss of making predictions by random guess. Following the "algorithm selection" interpretation (7), no g^* could be determined. Even if an arbitrary g^* is selected and performed, the performance would not grow with the growth of example number.

However, we find that ICL model trained with $\boldsymbol{\Omega}_{mix}$ can handle radials distance tasks effectively. Although exemplar tasks in Figure 5 shows that ICL model does not solve them optimally, the growing accuracy with the growth of example number in Figure 6(a) indicates that it does effectively learn task-specific information, while "algorithm selector" can not. We conjecture that when $g \in \boldsymbol{\Omega}_g$ is inclusive, the pre-trained ICL could be generalized to diverse tasks even from novel type, contributing to LLM's success by approaching the ideal "learning to learn".

4.3.2 ICL SHOWS DISTRIBUTION-SENSITIVE GENERALIZABILITY

Another result is it has limited generalizability on tasks even from seen types, being sensitive to the data distribution, while "algorithm selector" would not as an explicit optimal algorithm. Figure 6(b) shows the ICL's performance on hybrid tasks from seen type. The ICL model is trained with $\boldsymbol{\Omega}_{mix}$ from with input distribution $p(x)$, while we test with $\boldsymbol{\Omega}'_{mix}$ from a shifted distribution $p'(x)$. The performance obviously decreases with the distribution shift between training and testing (x-axis), although the tasks are all from seen types. This result directly answers our question that **though ICL model trained with $\boldsymbol{\Omega}_{mix}$ does learn a data-dependent optimal learning algorithm, it is implicit**. This gives ICL model limited generalizability that the meta-testing performance is sensitive to data distribution, showing deep-learning characteristic about generalizability at meta-level.

5 IMPROVING ICL THROUGH TRANSFERRING DEEP-LEARNING TECHNIQUES TO META-LEVEL

It is demonstrated that ICL with transformers, can effectively learn optimal learning algorithms data-dependently in an inclusive space. However, the generalizability of these learning algorithms is identified to be a critical issue, as their learned algorithms, as the learned algorithm could be only "implicit" optimal on training tasks. Conceptually, this is isomorphic with the widely studied deep-learning characteristics. In this section, we discuss the chance if we can transfer mature deep-learning techniques which have been widely studied in supervised-learning, to meta-level to improve ICL performance, by applying the techniques through a direct conceptual mapping from supervised-learning to meta-learning, e.g., sample to task, sample-wise loss to task-wise auto-regressive loss, epoch to episode.

From a motivational perspective, due to the shared deep-learning characteristics, the training of supervised deep-learning model and ICL model faces the same basic challenge: with very large parameter size it requires more inclusive training data to generalize and perform well, but the training data is always limited in real world. We also have common expectations of them, like improving generalizability, accelerating convergence, equipped with fast adaptation ability. So techniques in supervised-learning utilizing the deep-learning characteristics might also be effective for ICL. From a technical perspective, many effective techniques in deep-learning to achieve the above aims do not have strict requirement on loss function or model architecture, where a differentiable supervision signal is enough. Thus at least they can be implemented to perform at meta-level through the mapping. Here we discuss two exemplar practices: meta-level meta-learning, which successfully improve ICL's performance on specific domain with very limited data for adaptation; meta-level curriculum-learning, which successfully make the pre-training process converge faster.

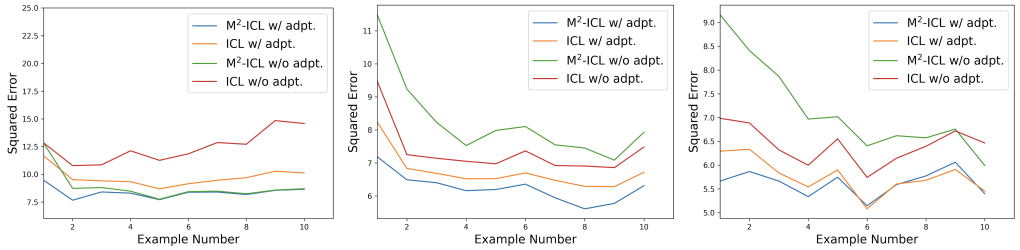
5.1 META-LEVEL META-LEARNING

While general ICL model can be directly applied, it is also a usual practice to build domain-specific artificial intelligence through adapting general ICL model with domain-specific data. However, compared with the general pre-training, the domain-specific data is very limited and can easily cause over-fitting. Such few-task problem is similar to the few-shot problem in supervised learning. It has been widely studied how to adapt ICL models efficiently to avoid over-fitting, e.g., LORA (Hu et al., 2021), prefix-tuning (Li & Liang, 2021). Existing works all address the few-task problem assuming that a general ICL model pre-trained by (13) is given. We consider a setting where the ICL model is pre-trained for adaptation rather than directly applied, i.e., only expect the performance of ICL model after adapting with few task from unknown domain. This can be considered as meta meta-learning, and could be addressed by transferring meta-learning methods through the mapping to one more meta level, i.e., solving a bi-level optimization problem by mimicking few-task domain adaptation during pre-training. This is practical as real-world pre-training tasks can be naturally divided into different domains according to semantics.

Consider a domain distribution $p(\delta)$. Each domain δ determines a distribution of tasks $p_\delta(\tau)$ where a domain-specific task set $\mathbf{\Omega}_\delta = \{\mathcal{D}_\tau\}_{\tau=1}^{T_\delta}$ can be drawn. During pre-training, we manually split $\mathbf{\Omega}_\delta$ into two disjoint task sets: a training (support) task set $\mathbf{\Omega}_\delta^{\text{tr}} = \{\mathcal{D}_\tau\}_{\tau=1}^{t}$ to and a validation (query) task set $\mathbf{\Omega}_\delta^{\text{val}} = \{\mathcal{D}_\tau\}_{\tau=t+1}^{T_\delta}$. Denote a meta meta-leaner as $G(g, \mathbf{\Omega}; \Delta)$, i.e., a domain adapter adapting meta-leaner g with task set $\mathbf{\Omega}$. Denote a meta loss function evaluating meta-leaner g with $\{\mathcal{D}_\tau^{\text{tr}}, \mathcal{D}_\tau^{\text{val}}\}$ as $\ell_{meta}(\tau, g)$. Meta meta-training is performing:

$$\min_{\Delta} \mathbb{E}_{p(\delta)}[\frac{1}{|\mathbf{\Omega}_\delta^{\text{val}}|} \sum\nolimits_{\tau\in\mathbf{\Omega}_\delta^{\text{val}}} \ell_{meta}(\tau, G(g(;\theta), \mathbf{\Omega}_\delta^{\text{tr}}; \Delta))]. \tag{8}$$

Specifically for adopting meta meta-learning to improve the pre-training of ICL model: $g(;\theta) = g_M(;\Theta_M)$; and we choose transfer MAML as $G(g(;\theta), \mathbf{\Omega}_\delta^{\text{tr}}; \Delta) = g(;\theta - \nabla_\theta \frac{1}{|\mathbf{\Omega}_\delta^{\text{tr}}|} \sum_{\tau\in\mathbf{\Omega}_\delta^{\text{tr}}} \ell_{meta}(\tau, g(;\theta)))$, because of MAML's model-agnostic property to avoid designing additional learnable Δ.



(a) Given 64 tasks for adaptation. (b) Given 256 tasks for adaptation. (c) Given 1024 tasks for adaptation.

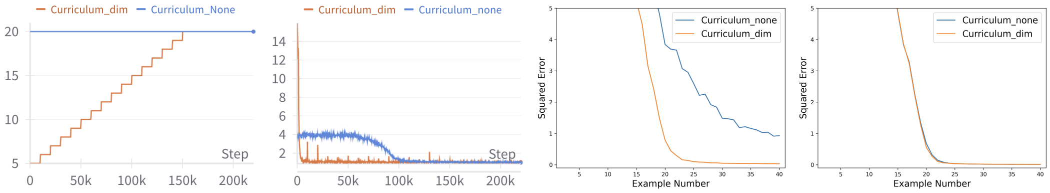
Figure 7: Performance of meta-trained ICL model and meta meta-trained ICL model on unseen domain given few tasks for adaptation.

We conduct experiments on linear regression tasks where the distribution of linear weights, $p_\delta(\tau) = \mathcal{N}(\mu_\delta, \Sigma_\delta)$, where $(\mu_\delta, \Sigma_\delta) \sim p(\delta)$. More details are provided in Appendix G. We denote such meta meta-trained ICL model as **M^2-ICL**. After pre-training, we test on unseen domains drawn from $p(\delta)$. Each domain providing $\mathbf{\Omega}_\delta^{\text{tr}} = \{\mathcal{D}_\tau\}_{\tau=1}^{t}$ that can be used for adaptation, and $\mathbf{\Omega}_\delta^{\text{val}} = \{\mathcal{D}_\tau\}_{\tau=t+1}^{T_\delta}$ to evaluate the performance. The performance is shown in Figure 7. Note that reasonable solutions include ICL w/ adpt, ICL w/o adpt, and M^2-ICl w/ adpt, while M^2-ICl w/o adpt is only the intermediate product of meta-level meta-learning. Though without adaptation, M^2-ICL performs not necessarily perform better than ICL, we find that with carefully tuning the hyper-parameters of the adaptation process, M^2-ICL w/ adpt performs better than both ICL w/ adpt and ICL w/o adpt as expected. Especially when the adaptation tasks are very few (64, Figure 7(a)). The advantage brought by adaptation increases with the task number. With 1024 tasks for adaptation (Figure 7(c)), the advantage of meta meta-training is marginal. Note although in this experiment the adaptation strategy is fine-tuning all parameters with gradient descent, i.e., G is transferred from MAML with inner-update as full-parameter fine-tuning, any differentiable adaptation strategy can be adopted to replace the inner-update. The comparison between ICL and M^2-ICL is isomorphic with the comparison between a model trained following standard supervised learning and a model meta-trained following MAML.

5.2 META-LEVEL CURRICULUM LEARNING

Curriculum meta-learning strategy is intuitive that meta-learner should progressively learn tasks from simple to complex for better convergence (Bengio et al., 2009). It has been investigated to be effective for gradient-based (Chen et al., 2021; Stergiadis et al., 2021) and metric-based (Zhang et al., 2022) meta-learners. We investigate if the curriculum strategy can benefit the meta-training of ICL.

We consider a simple case, ICL model learning linear regression tasks where complexity is evaluated by the number of dimensions, which has been practiced by Garg et al. (2022) de facto but the effect has not been investigated. With maximum dimension of 20, the **Curriculum_dim** trains the ICL model with tasks with increasing number of effective dimensions (shown as Figure 8(a)), while values in the other dimensions are kept to be zero. The training loss shown in Figure 8(b) and performance comparison with limited training shown in Figure 8(c) show that training ICL model with such curriculum converges faster. But with sufficient training, ICL model trained with and without such curriculum have very close training loss and testing performance (Figure 8(d)). This indicates that such curriculum can only brings faster convergence but can not find better optimum in this case.



(a) Effective dimension. (b) Training loss. (c) Testing @ 2×10^5 episodes. (d) Testing @ 5×10^5 episodes.

Figure 8: Training dynamics and testing performance of training ICL model with curriculum.

6 CONCLUSION, LIMITATIONS AND DISCUSSION

From the nature that pre-training ICL model is learning to learn, this paper provides analysis of ICL models in comparison to traditional meta-learning approaches, and strategies to improve ICL. It is demonstrated that ICL with transformers can effectively learn optimal learning algorithms data-dependently in an inclusive space. However, the generalizability of these learning algorithms is identified to be a critical issue, as the learned algorithm could be only "implicit" optimal on training tasks.

This understanding might be interpreted as that ICL model in meta-learning is conceptually isomorphic with deep-model in supervised-learning, as on meta-level it shows the widely studied deep-learning characteristics. Based on the above understanding, this research further proposes strategies to enhance ICL by transferring mature deep-learning techniques in supervised-learning to meta-level, to address common challenges and achieve common expectations. such as meta-level meta-learning and curriculum learning, which show promise in improving domain adaptability and convergence speed. The findings offer valuable insights into the nature of ICL and provide a foundation for developing more robust and generalizable models in the future.

There are limitations and future works left to be done. This paper only investigates ICL with transformer omitting the sequential order of examples. The order of examples might can be studied focusing on the effect of positional embeddings in transformer decoupled from the learning algorithm. Though transformer is the conventional architecture for ICL, there are other deep architectures for black-box meta-learners such as the classical RNNs and emerging SSMs Gu & Dao (2023). The convergence of pre-training ICL model is not understood in this paper, as training deep model with high non-linearity with very complex data is very hard to trace, especially for LLMs in real-world. Not considering convergence (i.e., with sufficient training), it is hoped to develop quantitative relations among the size of hypothesis space, inclusiveness of training tasks and generalizability, which is conjectured that the growth of hypothesis space might lead to decreasing generalizability given certain training tasks, while more inclusive training tasks possibly lead to growing generalizability but upper bounded by the hypothesis space, describing the neural scaling law of in-context learning. More deep-learning techniques like contrastive learning and denoising could be transferred to meta-level to improve ICL.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36:45614–45650, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *International Conference on Learning Representations*, 2023.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in Neural Information Processing Systems*, 36:57125–57211, 2023.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Satwik Bhattamishra, Arkil Patel, Phil Blunsom, and Varun Kanade. Understanding in-context learning in transformers and LLMs by learning to learn discrete functions. In *International Conference on Learning Representations*, 2024.

John Bronskill, Daniela Massiceti, Massimiliano Patacchiola, Katja Hofmann, Sebastian Nowozin, and Richard Turner. Memory efficient meta-learning with large images. *Advances in Neural Information Processing Systems*, 34:24327–24339, 2021.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Yudong Chen, Xin Wang, Miao Fan, Jizhou Huang, Shengwen Yang, and Wenwu Zhu. Curriculum meta-learning for next poi recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2692–2702, 2021.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can GPT learn in-context? language models implicitly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4005–4019, 2023.

Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2018.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *International Conference on Machine Learning*, pp. 15130–15152, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9): 5149–5169, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yu Huang, Yuan Cheng, and Yingbin Liang. In-context convergence of transformers. In *International Conference on Machine Learning*, pp. 19660–19722, 2024.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 1(8), 2017.

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

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, pp. 1–30. Lille, 2015.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Hongkang Li, Meng Wang, Songtao Lu, Xiaodong Cui, and Pin-Yu Chen. How do nonlinear transformers learn and generalize in in-context learning? In *International Conference on Machine Learning*, pp. 28734–28783, 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pp. 19565–19594. PMLR, 2023a.

Yuchen Li, Yuanzhi Li, and Andrej Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. In *International Conference on Machine Learning*, pp. 19689–19729. PMLR, 2023b.

Arvind Mahankali, Tatsunori B Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. In *International Conference on Learning Representations*, 2024.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. *Advances in Neural Information Processing Systems*, 32:7957–7968, 2019.

Michael E Sander, Raja Giryes, Taiji Suzuki, Mathieu Blondel, and Gabriel Peyré. How do transformers perform in-context autoregressive learning? In *International Conference on Machine Learning*, pp. 43235–43254, 2024.

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook.* PhD thesis, Technische Universität München, 1987.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 30:4077–4087, 2017.

Emmanouil Stergiadis, Priyanka Agrawal, and Oliver Squire. Curriculum meta-learning for few-shot classification. *arXiv preprint arXiv:2112.02913*, 2021.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to Learn*, pp. 3–17. Springer, 1998.

Yuandong Tian, Yiping Wang, Beidi Chen, and Simon S Du. Scan and snap: Understanding training dynamics and token composition in 1-layer transformer. *Advances in Neural Information Processing Systems*, 36:71911–71947, 2023.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95, 2002.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 29:3630–3638, 2016.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Zhijie Wang, Bo Jiang, and Shuai Li. In-context learning on function classes unveiled for transformers. In *International Conference on Machine Learning*, pp. 50726–50745, 2024.

Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. O (n) connections are expressive enough: Universal approximability of sparse transformers. *Advances in Neural Information Processing Systems*, 33:13783–13794, 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in Neural Information Processing Systems*, 30:3391–3401, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Ji Zhang, Jingkuan Song, Lianli Gao, Ye Liu, and Heng Tao Shen. Progressive meta-learning with curriculum. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(9):5916–5930, 2022.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024.

Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. What and how does in-context learning learn? bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*, 2023.

A RELATED WORKS

This paper is related with a wide range of existing works, including understanding ICL with transformer (Von Oswald et al., 2023; Mahankali et al., 2024; Ahn et al., 2023; Gatmiry et al., 2024; Wang et al., 2024; Bai et al., 2023; Tian et al., 2023; Li et al., 2023b; Huang et al., 2024; Zhang et al., 2024; Sander et al., 2024; Li et al., 2023a; 2024; Wies et al., 2024; Akyürek et al., 2023; Bhattamishra et al., 2024; Zhang et al., 2023), meta-learning (Schmidhuber, 1987; Thrun & Pratt, 1998; Koch et al., 2015; Finn et al., 2017; Vinyals et al., 2016; Snell et al., 2017; Garnelo et al., 2018; Requeima et al., 2019; Garcia & Bruna, 2018; Kirsch et al., 2022), expressiveness of neural networks (Hornik et al., 1989; Raghu et al., 2017; Yun et al., 2019; 2020), generalizability of deep-learning (Kawaguchi et al., 2017; Neyshabur et al., 2017; Zhang et al., 2021).

Most closely related works have been mentioned in main text. Here we discuss the relation between this paper and most related ones. Akyürek et al. (2023); Bai et al. (2023) have comprehensively studied what exact explicit learning algorithms transformers can learn in-context, including ridge regression, least squares, Lasso on linear regression tasks. Based on their results, we believe there are numerous explicit learning algorithms transformer can learn in-context, so rather than investigating what exact explicit learning algorithms transformers learn under different settings we provide a more general and abstract understanding of ICL with transformer: it is a deep-algorithm-model, which can express typical meta-learners, while those meta-learners can express a wide range of explicit learning algorithms Finn & Levine (2017); Zaheer et al. (2017), this understanding accommodates and extends existing results. Wang et al. (2024) and Bai et al. (2023) have proved that ICL with transformer can learn gradient-descent of neural networks, we use this result as an important tool to prove that ICL can perform gradient-based meta-learners. We follow Kirsch et al. (2022)'s definition of learning algorithm and start from their understanding that ICL models are general-purpose meta-learning systems with minimal inductive bias. Kirsch et al. (2022) show that ICL model can learning to learn, and its generalizability increases with the increase of training task number on few-shot image classification tasks, while we compare ICL model with other meta-learners, proving the expressiveness of ICL with transformer, revealing its learnability and what characteristics the learned algorithms show. It can be viewed as a cornerstone. The transition pattern of the learning ability with the growing number of training tasks (Kirsch et al., 2022) can be complement to our work.

B PRELIMINARIES

B.1 IN-CONTEXT LEARNING WITH TRANSFORMER

Input. Following existing works (Von Oswald et al., 2023; Ahn et al., 2023), we investigate ICL without positional embedding to study the learning to learn ability omitting the order of examples. Let $\mathbf{x}^{(i)} \in \mathbb{R}^d$ be a input, and $\mathbf{y}^{(i)} \in \mathbb{R}^e$ be the corresponding output. For each task τ, there a task-specific function f_τ and dataset of the task $\mathcal{D}_\tau$ that $\forall(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \in \mathcal{D}_\tau, \mathbf{y}^{(i)} = f_\tau(\mathbf{x}^{(i)})$.

Labeled examples and query of a task are input together. Define the input matrix Z_0:

$$Z_0 = \begin{bmatrix} \mathbf{z}^{(1)} \; \mathbf{z}^{(2)} \; \cdots \; \mathbf{z}^{(n)} \; \mathbf{z}^{(n+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}^{(n)} & \mathbf{x}^{(n+1)} \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \cdots & \mathbf{y}^{(n)} & \boldsymbol{q} \end{bmatrix} \in \mathbb{R}^{(d+e)\times(n+1)}, \tag{9}$$

where $\boldsymbol{q} \in\in \mathbb{R}^e$ is the indicator of unlabeled query. ICL model is trained to output the prediction of $\mathbf{y}^{(n+1)}$ given Z_0, with a set of tasks $\{\mathcal{D}_\tau\}_{\tau=1}^T = \{\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N_\tau}\}_{\tau=1}^T$ from training distribution $f_\tau \sim p(\tau), \mathbf{x}^{(i)} \sim p(x)$, to generalize to unseen tasks.

Model Architecture. ICL is typically achieved by transformer, stacked with self-attention layers. Letting $Z \in \mathbb{R}^{(d+e)\times(n+1)}$,a single-head self-attention layer denoted by $\mathrm{Attn}^{\mathsf{smax}}$ is a parametric map defined as

$$\mathrm{Attn}^{\mathsf{smax}}_{W_{k,q,v}}(Z) = W_v Z \cdot \mathsf{smax}(Z^\top W_k^\top W_q Z)\,, \tag{10}$$

where $W_v, W_k, W_q \in \mathbb{R}^{(d+e)\times(d+e)}$ are the (value, key and query) weight matrices, and $\mathsf{smax}(\cdot)$ is the softmax operator which applies softmax operation to each column of the input matrix. Note that the prompt is asymmetric since the label for $x^{(n+1)}$ is excluded from the input.

An M-layer transformer is denoted as TF_M, as a stack of M self-attention and MLP blocks. Formally, denoting by Z_l the output of the l^{th} layer attention, we define

$$Z_{l+1} = Z_l + \sigma_{\alpha_l l}(\mathrm{Attn}_{P_l,Q_l}(Z_l)) \quad \text{for } l = 0, 1, \ldots, M-1, \tag{11}$$

where $\sigma(\cdot)$ is feed-forward layers function on each column of the input independently. Given Z_0, the prediction

$$\hat{\mathbf{y}}^{(n+1)} = \mathsf{TF}_M(Z_0; \{P_l, Q_l, \alpha_l\}_{l=0}^{M-1}, W_r) = W_r[Z_M]_{:,(n+1)}, \tag{12}$$

where $[Z_M]_{:,(n+1)}$ is the $(n+1)$-th column of Z_M, and $W_r \in \mathbb{R}^{e\times(d+e)}$ is the linear readout weight.

For training, given the distribution of tasks $f_\tau \sim p(\tau), \mathbf{x}^{(i)} \sim p(x)$, and loss function $\ell(\cdot,\cdot)$ (e.g., cross-entropy), the parameters are optimized to minimize the expectation of auto-regressive loss of training tasks:

$$\min_{\{P_l,Q_l,\theta_l\}_{l=0}^{M-1}, W_r} \mathbb{E}_{p(\tau),p(x)} \big[\frac{1}{N_\tau} \sum_{i=0}^{N_\tau - 1} \ell(\hat{\mathbf{y}}^{(i+1)}, \mathbf{y}^{(i+1)})\big] \tag{13}$$

B.2 META-LEARNING

Meta-learning is a methodology considered with "learning to learn" algorithms. Define $g(;\theta)$ is a meta-learner that maps a task dataset $\mathcal{D}_\tau$ and a query input $\mathbf{x}^{(q)}$ to its task-specific prediction. Typically meta-learning algorithms first learns an explicit model to a model h from $\mathcal{D}_\tau$ and then perform the prediction, i.e, $\hat{\mathbf{y}}^{(q)} = g(\mathbf{x}^{(q)}, \mathcal{D}; \theta) = g'(\mathcal{D};\theta)(\mathbf{x}^{(q)})$. For meta-training, given the training distribution $p(\tau)$ and $p(x)$ where the tasks $\{\mathcal{D}_\tau\}_{\tau=1}^T$ are drawn, the goal of is to learn $g(;\theta)$ to perform well on unseen task. A typical way is manually split each task into two disjoint sets: a training set $\mathcal{D}_\tau^{\text{tr}} = \{(x_{\tau,i}, y_{\tau,i})\}_{i=1}^n$ to input and a validation set $\mathcal{D}_\tau^{\text{val}} = \{(x_{\tau,i}, y_{\tau,i})\}_{i=n+1}^{N_\tau}$ to optimize the meta-learner, i.e., the meta-training is performed as:

$$\min_\theta \mathbb{E}_{p(\tau),p(x)} \big[\frac{1}{N_\tau - n} \sum_{i=n+1}^{N_\tau} \ell(g(\mathbf{x}^{(i)}, \mathcal{D}_\tau^{\text{tr}}; \theta), \mathbf{y}^{(i)})\big] \tag{14}$$

C OPTIMALITY OF A LEARNING ALGORITHM

We claim ICL model learns data-dependent optimal learning algorithms (DDOLA), which is different and weaker than (true) optimal learning algorithm (OLA).

Formally, given a finite training set $D_{train} = \{(x_i, y_i)\}$ where each sample is i.i.d.: $(x_i, y_i) \sim p(x, y)$, and a unseen testing set $D_{test} = \{(x_j, y_j)\}$ following the same distribution, the OLA is $g^* = \mathrm{argmin}_g \mathbb{E}_{(x_j,y_j)\sim p(x,y)}\{\mathrm{Prob}[g(x_j, D_{train}) = y_j]\}$. Which is to say a learning algorithm can make the most "accurate" prediction given a training set and unseen target input from the same distribution. It is possible to know the optimal learning algorithm with the priori of $p(x, y)$. For example, ordinary least squares is optimal for linear regression with Gaussian noise. In the paper, three types of tasks are generated by designed ways, i.e., known $p(x, y)$ (Section 4.1). It is obvious that a MatchNet model with certain parameters (simply keeping all modules inside as identical mappings) is the optimal learning algorithm for $\mathbf{\Omega}_{sim}$, and so does ProtoNet for $\mathbf{\Omega}_{prt}$ and CNPs for $\mathbf{\Omega}_{am}$. However, meta-learners have not access the true $p(x, y)$. They only learn the function to infer $p(x, y)$ from $D_{train} = \{(x_i, y_i)\}$ through meta-training, which inevitably brings variance and bias, being (meta-training) data-dependent. So we denote that given certain meta-training set, the best that a random-initialized and meta-trained deep learner can do as the DDOLA. This could be empirically approximated by meta-training a deep and random-initialized MatchNet/ProtoNet/CNPs with certain meta-training set (for the three task types respectively).

D PROOF OF THE META-LEVEL EXPRESSIVENESS OF ICL

D.1 DETAIL SETTINGS

D.1.1 CLASSIFICATION TASK WITH ORTHONORMAL LABEL EMBEDDING

Classification task specifies the ICL's input in Section B.1 with $\mathbf{y}^{(i)} \in \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_C\}$, where $\boldsymbol{c}_c$ is the embedding vector of the c-th class. For these label embeddings, we can find $2C$ orthonormal vectors in $\mathbb{R}^{2C}$: $\{\boldsymbol{u}_j\}_{j=1}^{2C}$, that:

$$\boldsymbol{u}_i^\top \boldsymbol{u}_j = \begin{cases} 1, \text{ if } i = j \\ 0, \text{ if } i \neq j \end{cases}. \tag{15}$$

A simple choice of $\{\boldsymbol{u}_j\}_{j=1}^{2C}$ is the set of $2C$ one-hot vectors. We use $\{\boldsymbol{u}_j\}_{j=1}^{C}$ as the embeddings of $\{\boldsymbol{c}_c\}_{c=1}^{C}$, i.e., $\mathbf{y}^{(i)} \in \{\boldsymbol{u}_j\}_{j=1}^{C}$, and $\boldsymbol{u}_{C+1}$ as the indicator of query $\boldsymbol{q}$.

D.1.2 SELF-ATTENTION WITHOUT SOFTMAX

In our setting, we consider self-attention layers that replace the softmax operation in (10) with column-wise $L1$-normalization. In particular, (10) is now approximated and reparameterized with weights $P := W_v \in \mathbb{R}^{(d+e)\times(d+e)}$ and $Q := {W_k}^\top W_q \in \mathbb{R}^{(d+e)\times(d+e)}$ as:

$$\mathrm{Attn}_{P,Q}(Z) = PZ\mathsf{norm}_1^{\mathsf{col}}(Z^\top QZ). \tag{16}$$

where $[\mathsf{norm}_1^{\mathsf{col}}(A)]_{i,j} = \begin{cases} \dfrac{A_{i,j}}{\sum_i |A_{i,j}|}, & \sum\limits_i |A_{i,j}| \neq 0 \\ 0, & \sum\limits_i |A_{i,j}| = 0 \end{cases}$. Note that it is a convention to omit some non-linearity of softmax in self-attention layers to align transformer with explicit learning algorithms. While existing works simply omit the softmax operation, replacing it with $\frac{1}{n}$ (Von Oswald et al., 2023; Ahn et al., 2023), the $\mathsf{norm}_1^{\mathsf{col}}$ in (16) is a more close approximation.

As the proof for gradient-based algorithm (3) is trivial by applying the result from Bai et al. (2023); Wang et al. (2024), we focus on metric-based (4)(5) and amortization-based (6) algorithms. We prove that there exists TF model with certain real-valued parameters can perform these algorithms.

Note that for simplicity in proving the expressiveness of ICL with transformer, we focus on the algorithm framework: we leave feature-level transformations with neural networks alone as they can occur in both ICL model and conventional meta-learners, and enjoy the same universal approximation property (Hornik et al., 1989); we also do not consider the order of samples in $\mathcal{D}$, omitting any sequential models in meta-learners and positional embeddings in ICL. Typical metric-based algorithms are thus categorized into to types: one is based on pair-wise distance (4), e.g., MatchNet; another one is based on distance with class prototypes (5), e.g., ProtoNet. And typical amortization-based algorithms are summarized as a function taking the query and the encoded set as input (6). Proving these exemplar set and inference functions can be achieved, more algorithms including feature-wise transformation, interaction between samples, more complex distance functions can be easily achieved with the the recursion of self-attention and feed-forward layers.

D.1.3 ICL CAN PERFORM PAIR-WISE METRIC-BASED ALGORITHMS

For pair-wise metric-based algorithms, we take MatchNet for example, proving (12) can perform

$$\hat{\mathbf{y}}^{(n+1)} = \frac{1}{n}\sum_{i=1}^{n} < \mathbf{x}^{(i)}, \mathbf{x}^{(n+1)} > \mathbf{y}^{(i)}. \tag{17}$$

In fact, this case is quite simple that it can be achieved with a single layer transformer without the help of the two tools. One implementation is $Q_0 = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$, $P_0 = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}$, $W_r = \begin{bmatrix} 0 & \cdots & 0 \\ \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_c \end{bmatrix}^\top$. Note that though the output of TF would be $\lambda \sum_{i=1}^{n} < \mathbf{x}^{(i)}, \mathbf{x}^{(n+1)} > \mathbf{y}^{(i)}$, where $\lambda \in \mathbb{R}$ is a query-specific value, it has the same classification result with (17) after de-embedding.

D.1.4 ICL CAN PERFORM CLASS-PROTOTYPE METRIC-BASED ALGORITHMS

For the second category of metric-based algorithms, we take ProtoNet for example, proving (12) can perform

$$\hat{\mathbf{y}}^{(n+1)} = \sum_{c=1}^{C} -||\frac{1}{N_c}\sum_{y^{(i)}=c}\mathbf{x}^{(i)} - \mathbf{x}^{(n+1)}||\boldsymbol{u}_c. \tag{18}$$

This can be implemented by a $3C-1$ layer transformer achieving $[Z_{3l-1}]_{(d:d+2C),(n+1)} = \boldsymbol{q} + \sum_{c=1}^{l}||\mathbf{x}^{(n+1)} - \boldsymbol{p}_c||\boldsymbol{u}_{(c+1+C)mod(2C)}$ in the following step-by-step function:

$$Z_0 = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}^{(n)} & \mathbf{x}^{(n+1)} \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \cdots & \mathbf{y}^{(n)} & \boldsymbol{q} \end{bmatrix}, \tag{19}$$

$$Z_1 = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} - \boldsymbol{p}_1 \\ \mathbf{y}^{(i)} & \boldsymbol{q} \end{bmatrix}, \tag{20}$$

$$Z_2 = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} - \boldsymbol{p}_1 \\ \mathbf{y}^{(i)} & \boldsymbol{q} + ||\mathbf{x}^{(n+1)} - \boldsymbol{p}_1||\boldsymbol{u}_{C+2} \end{bmatrix}, \tag{21}$$

$$Z_3 = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} \\ \mathbf{y}^{(i)} & \boldsymbol{q} + ||\mathbf{x}^{(n+1)} - \boldsymbol{p}_1||\boldsymbol{u}_{C+2} \end{bmatrix}, \tag{22}$$

$$Z_4 = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} - \boldsymbol{p}_2 \\ \mathbf{y}^{(i)} & \boldsymbol{q} + ||\mathbf{x}^{(n+1)} - \boldsymbol{p}_1||\boldsymbol{u}_{C+2} \end{bmatrix}, \tag{23}$$

$$Z_5 = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} - \boldsymbol{p}_2 \\ \mathbf{y}^{(i)} & \boldsymbol{q} + ||\mathbf{x}^{(n+1)} - \boldsymbol{p}_1||\boldsymbol{u}_{C+2} + ||\mathbf{x}^{(n+1)} - \boldsymbol{p}_2||\boldsymbol{u}_{C+3} \end{bmatrix}, \tag{24}$$

$$Z_6 = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} \\ \mathbf{y}^{(i)} & \boldsymbol{q} + ||\mathbf{x}^{(n+1)} - \boldsymbol{p}_1||\boldsymbol{u}_{C+2} + ||\mathbf{x}^{(n+1)} - \boldsymbol{p}_2||\boldsymbol{u}_{C+3} \end{bmatrix}, \tag{25}$$

$$\cdots, \tag{26}$$

$$Z_{3l-3} = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} \\ \mathbf{y}^{(i)} & \boldsymbol{q} + \sum_{c=1}^{l-1}||\mathbf{x}^{(n+1)} - \boldsymbol{p}_i||\boldsymbol{u}_{(i+1+C)mod(2C)} \end{bmatrix}, \tag{27}$$

$$Z_{3l-2} = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} - \boldsymbol{p}_l \\ \mathbf{y}^{(i)} & \boldsymbol{q} + \sum_{c=1}^{l-1}||\mathbf{x}^{(n+1)} - \boldsymbol{p}_i||\boldsymbol{u}_{(i+1+C)mod(2C)} \end{bmatrix}, \tag{28}$$

$$Z_{3l-1} = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} - \boldsymbol{p}_l \\ \mathbf{y}^{(i)} & \boldsymbol{q} + \sum_{c=1}^{l}||\mathbf{x}^{(n+1)} - \boldsymbol{p}_i||\boldsymbol{u}_{(i+1+C)mod(2C)} \end{bmatrix}, \tag{29}$$

$$\tag{30}$$

and readout by $W_r = \begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 \\ -\boldsymbol{u}_{C+2} & \cdots & -\boldsymbol{u}_{(c+1+C)mod(2C)} & \cdots & -\boldsymbol{u}_1 \end{bmatrix}^\top$. Each step of function from Z_l to Z_{l+1} can be implemented with one transformer layer, which would be proved later.

D.1.5 ICL CAN PERFORM AMORTIZATION-BASED ALGORITHMS

Denote the set embedding $\frac{1}{n}\sum_{i=1}^{n}[\mathbf{x}^{(i)}|\mathbf{y}^{(i)}]$ as $\mathbf{e} \in \mathbb{R}^{d'}$. As f in (6) can always be implemented by feed forward layers taking the concatenation of $\mathbf{x}^{(q)}$ and $\mathbf{e}$ as input, there exists a learnable function h_1 in $\mathbb{R}^{d'} \times \mathbb{R}^{d}$ and h_2 in $\mathbb{R}^{d} \times \mathbb{R}^{2C}$ that

$$f([(\mathbf{x}^{(q)})^\top, \mathbf{e}^\top]^\top) = h_2(\mathbf{x}^{(q)} + h_1(\mathbf{e})). \tag{31}$$

Thus, we prove that (12) can perform

$$\hat{\mathbf{y}}^{(n+1)} = h_2(\mathbf{x}^{(n+1)} + h_1(\frac{1}{n}\sum_{i=1}^{n}[\mathbf{x}^{(i)}|\mathbf{y}^{(i)}])). \tag{32}$$

This can be implemented by a 3 layer transformer achieving the following step-by-step function:

$$Z_0 = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}^{(n)} & \mathbf{x}^{(n+1)} \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \cdots & \mathbf{y}^{(n)} & \boldsymbol{q} \end{bmatrix}, \tag{33}$$

$$Z_1 = \begin{bmatrix} \mathbf{x}^{(i)} & \mathbf{x}^{(n+1)} + h_1(\frac{1}{n}\sum_{i=1}^{n}[\mathbf{x}^{(i)}|\mathbf{y}^{(i)}]) \\ \mathbf{y}^{(i)} & \boldsymbol{q} \end{bmatrix} \tag{34}$$

$$Z_2 = \begin{bmatrix} \mathbf{x}^{(i)} & h_2(\mathbf{x}^{(n+1)} + h_1(\frac{1}{n}\sum_{i=1}^{n}[\mathbf{x}^{(i)}|\mathbf{y}^{(i)}])) \\ \mathbf{y}^{(i)} & \boldsymbol{q} \end{bmatrix}, \tag{35}$$

and readout by $W_r = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$. Each step of function from Z_l to Z_{l+1} can be implemented with one transformer layer, which would be proved now.

D.1.6 THE FUNCTION OF ONE TRANSFORMER LAYER

A transformer layer (11) can perform a wide range of functions, as we can decompose it is composed of a self-attention layer and feed-forward layers, where (i) self-attention (16) with orthonormal label tokenization (15) can achieve a wide range of label-aware set operations. (ii) feed-forward layer $\sigma(\cdot)$ in (11) can learn any measurable functions in $\mathbb{R}^{d+2c} \times \mathbb{R}^{d+2c}$. Here we prove how a transformer layer can obtain the above functions from Z_l to Z_{l+1}. The main idea is a function can be decomposed to three sub-steps: label-selecting which is achieved by $A = Z^\top QZ$, linear interaction achieved by $PZ\text{norm}_1^{\text{col}}(A)$, and non-linear transformation by $\sigma_\theta(\cdot)$ if needed.

Label-Aware Attention. In one self-attention layer, equation (16), first each column in Z refer to other columns through attention weights $A = Z^\top QZ$. $A \in \mathbb{R}^{(n+1)\times(n+1)}$ is selecting interaction objectives and weighting interaction weights. We use *label-aware* to describe $\{A \in \mathbb{R}^{(n+1)\times(n+1)} \mid (\mathbf{y}^{(i)} = \mathbf{y}^{(i')}) \wedge (\mathbf{y}^{(j)} = \mathbf{y}^{(j')}) \Rightarrow A_{i,j} = A_{i',j'}\}$, i.e., the interaction weight between ordered sample-pair (i, j) only depends on their labels $(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ (including unknown label $\boldsymbol{q}$), and can be arbitrary value in $\mathbb{R}$.

With our orthonormal label embedding in $\mathbb{R}^{2c}$, A is label-aware, thus can achieve label-aware interaction. For example, to achieve (19) to (20), we require $(c + 1)^2$ conditions about A:

$$\begin{cases} A_{c_1,q} = 1 \\ A_{c_i,q} = 0,\ i \in \{2, 3, \cdots, C\} \\ A_{q,c_i} = 0,\ i \in \{1, 2, \cdots, C\} \\ A_{c_i,c_j} = 0,\ i, j \in \{1, 2, \cdots, C\} \\ A_{q,q} = 0 \end{cases} \tag{36}$$

As $A = Z^\top QZ$ and $A_{i,j}$ is only related to $\mathbf{y}^{(i)}, \mathbf{y}^{(j)}$, we have $Q = \begin{bmatrix} 0 & 0 \\ 0 & L \end{bmatrix}$ where $L \in \mathbb{R}^{2C\times 2C}$. Equation (36) gives $(C + 1)^2$ linear equations about L:

$$\begin{cases} \boldsymbol{u}_1^\top L\boldsymbol{u}_q = 1 \\ \boldsymbol{u}_i^\top L\boldsymbol{u}_q = 0,\ i \in \{2, 3, \cdots, C\} \\ \boldsymbol{u}_q^\top L\boldsymbol{u}_i = 0,\ i \in \{1, 2, \cdots, C\} \\ \boldsymbol{u}_i^\top L\boldsymbol{u}_j = 0,\ i, j \in \{1, 2, \cdots, C\} \\ \boldsymbol{u}_q^\top L\boldsymbol{u}_q = 0 \end{cases} \tag{37}$$

Proposition D.1. $\forall c > 1$*, Equation* (37) *has solutions in* $\mathbb{R}^{2C\times 2C}$.

Proof. Denote $\boldsymbol{u}_i \otimes \boldsymbol{u}_j = [u_{i,1}\boldsymbol{u}_j^\top, u_{i,2}\boldsymbol{u}_j^\top, \cdots, u_{i,2C}\boldsymbol{u}_j^\top]^\top \in \mathbb{R}^{4C^2}$, $\vec{L} = [L_{1,1}, \cdots, L_{1,2C}, \cdots, L_{2C,2C}]^\top \in \mathbb{R}^{4C^2}$, then $\boldsymbol{u}_i^\top L\boldsymbol{u}_j = (\boldsymbol{u}_i \otimes \boldsymbol{u}_j)^\top \vec{L}$.

(37)$\iff U\vec{L} = \vec{A}$, where

$$U = [\boldsymbol{u}_i \otimes \boldsymbol{u}_j \text{ for } i, j \in \{1, 2, \cdots, C + 1\}]^\top \in \mathbb{R}^{(C+1)^2\times 4C^2}. \tag{38}$$

$$
\begin{aligned}
(\boldsymbol{u}_i \otimes \boldsymbol{u}_j)^\top (\boldsymbol{u}_{i'} \otimes \boldsymbol{u}_{j'}) &= \sum_{t=1}^{2c} u_{it} u_{i't} \boldsymbol{u}_j^\top \boldsymbol{u}_{j'} && (39)\\
&= (\boldsymbol{u}_j^\top \boldsymbol{u}_{j'})(\sum_{t=1}^{2c} u_{it} u_{i't}) && (40)\\
&= (\boldsymbol{u}_j^\top \boldsymbol{u}_{j'})(\boldsymbol{u}_i^\top \boldsymbol{u}_{i'}) && (41)\\
&= \begin{cases} 1, \text{ if } i = i' \ \wedge \ j = j' \\ 0, \text{ if } i \neq i' \ \vee \ j \neq j' \end{cases} && (42)
\end{aligned}
$$

$$\implies rank(U) = (C+1)^2 \tag{43}$$

$$[U, \vec{A}] \in \mathbb{R}^{(C+1)^2 \times (4C^2+1)}, c > 1 \implies rank([U, \vec{A}]) \leq (C+1)^2. \tag{44}$$

(43), (44) $\implies$

$$rank([U, \vec{A}]) = rank(U) = (C+1)^2 < 4C^2. \tag{45}$$

$\implies$ Equation $U\vec{L} = \vec{A}$ has solutions in R^{4C^2}. $\iff$ Equation (37) has solutions in $\mathbb{R}^{2C \times 2C}$. □

Note that for any function from Z_l to Z_{l+1}, the number of conditions about $A \leq (2C)^2$. Thus for any label-aware function from Z_l to Z_{l+1}, it requires a label-ware A and we can find a linear system of equations $U\vec{L} = \vec{A}$, that has solutions in $\mathbb{R}^{4c^2}$, as the proof $rank([U, \vec{A}]) = rank(U) \leq 4c^2$ is without loss of generalizability.

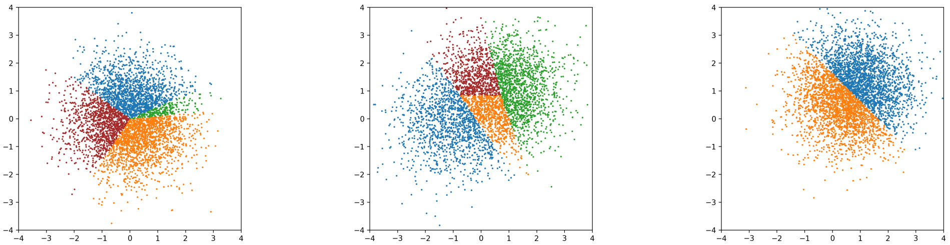
Linear Interaction. After obtaining desired A, $\text{norm}_1^{\text{col}}(A)$ is performed as $\text{norm}_1^{\text{col}}$ is a better approximation of softmax than $\frac{1}{n}$, and also required to deal with inconsistent label number in our classification tasks. Then all columns in Z interact with the others linearly through $PZ\text{norm}_1^{\text{col}}(A)$. Still taking (19) to (20) as example, after obtaining desired A satisfying (36), $P = \begin{bmatrix} -I & 0 \\ 0 & 0 \end{bmatrix}$ can achieve the function.

Non-Linear Transformation. In (11), $\sigma_\theta(\cdot)$ is feed-forward layers that function on each column of Z independently. Thanks to the universal approximation property (Hornik et al., 1989), it can approximate any measurable function in $\mathbb{R}^{d+2c} \times \mathbb{R}^{d+2c}$ to any desired degree of accuracy. Thus feature-level non-linear transformation from Z_l to Z_{l+1} could turn to $\sigma_{\theta_l}(\cdot)$. For example, (19) to (20) does not require non-linearity so it can be implemented as $\sigma(\boldsymbol{z}) = \boldsymbol{z}$. For (20) to (21), one implementation is $\sigma(\boldsymbol{z}) = [0, ||[\boldsymbol{z}]_{1:d+1}||\boldsymbol{u}_{C+2}]^\top$. Note that in this step, the $=$ does not hold strictly, but can be approximated by MLPs with error $\epsilon > 0$. We use "=" to mean such approximation for simplicity, as the error can be arbitrary small.

In conclusion, each step from Z_l to Z_{l+1} can be implemented using a transformer layer. Typical metric- and amortization-based meta-learning algorithms (4)(5)(6) can be implemented with ICL. More complex models following the same set functions can also be performed by ICl with additional recursion of transformer layers, whose proof is trivial. Moreover, as it has been proved that ICL can perform gradient-based algorithms (3), ICL can exactly perform conventional handcrafted meta-learning algorithms.

E GENERATING TASKS

Here we show the task generation for g_{sim}, g_{prt} and g_{am}. We design the following specific forms for the three types of tasks because they are all linearly separable by $\mathbf{x}^{(i)} \in \mathbb{R}^d$, for 2-d visualization to observe ICL's behavior.



(a) Pair-wise metric-based.

(b) Class-prototype metric-based.

(c) Amortization-based.

Figure 9: Examples of three types of tasks.

For pair-wise metric-based algorithms g_{sim}, we generate a task by sampling $C \times N_C$ support samples $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{i=1}^{C\times N_C}$ from distribution $p(\tau)$ and randomly assign them with labels $y^{(i)} = c$, making $C \times N_C$ supports exactly contains N_C label c for each $c = 1, 2, \cdots, C$. Then the remaining samples $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{i=C\times N_C+1}^{N_C+N}$ are sampled from distribution p_x, and assigned with labels $y^{(i)} = \text{argmax}_c \sum_{j=1,y^{(j)}=c}^{C\times N_C} < \mathbf{x}^{(i)}, \mathbf{x}^{(j)} >$. A typical meta-learner, MatchNet, can learn the optimal classifier. A case is shown in Figure 9(a), where each point corresponds to a $\mathbf{x}^{(i)} \in \mathbb{R}^2$ and different labels are assigned with different colors.

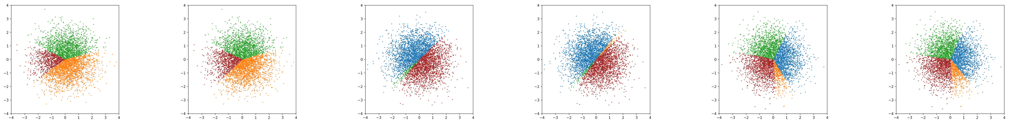
For class-prototype metric-based algorithms g_{prt}, we generate a task by sampling C prototypes $\{\boldsymbol{p}_c \in \mathbb{R}^d\}_{c=1}^C$ from $p(\tau)$. Then sample $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{i=1}^N$ from p_x, and assign labels by $y^{(i)} = \text{argmin}_c ||\boldsymbol{p}_c - \mathbf{x}^{(i)}||$. The corresponding optimal classifier is ProtoNet. A case is shown in Figure 9(b).

For amortization-based algorithms g_{am}, we pre-define a partition of R, $\{\Omega_c\}_{c=1}^C$, as decision range. We generate a task by sampling $\boldsymbol{\mu} \in \mathbb{R}^d$ from $p(\tau)$. Then sample $\{\mathbf{x}^{(i)} \in \mathbb{R}^d\}_{i=1}^N$ from $p_x(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, and assign labels by $y^{(i)} = c$ where $\sum_{t=1}^d [\mathbf{x}^{(i)} - \boldsymbol{\mu}]_t \in \Omega_c$. The corresponding optimal classifier is amortization-based. A case is shown in Figure 9(c).

F MORE EMPIRICAL RESULTS

F.1 ICL MODEL TRAINED WITH SINGLE TYPE OF TASKS

Figure 10, 11 and 12 show more cases to support that ICL model learns MatchNet, ProtoNet, CNPs on $\boldsymbol{\Omega}_{sim}$, $\boldsymbol{\Omega}_{prt}$, $\boldsymbol{\Omega}_{am}$ respectively.

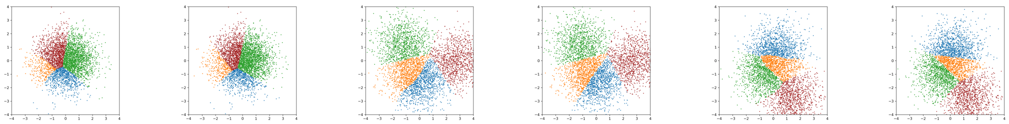


(a) ICL-1. (b) True-1. (c) ICL-2. (d) True-2. (e) ICL-3. (f) True-3.

Figure 10: Comparing ICL's predictions and true labels on pair-wise metric-based tasks (trained with single task type).

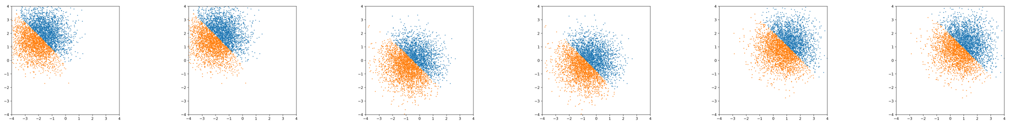
F.2 ICL MODEL TRAINED WITH MIXED TYPE OF TASKS

Figure 13, 14 and 15 show cases to support that ICL model learns data-dependent optimal learning algorithm on $\boldsymbol{\Omega}_{mix} = \{\boldsymbol{\Omega}_{sim}, \boldsymbol{\Omega}_{prt}, \boldsymbol{\Omega}_{am}\}$.



(a) ICL-1. (b) True-1. (c) ICL-2. (d) True-2. (e) ICL-3. (f) True-3.

Figure 11: Comparing ICL's predictions and true labels on class-prototype metric-based tasks (trained with single task type).



(a) ICL-1. (b) True-1. (c) ICL-2. (d) True-2. (e) ICL-3. (f) True-3.

Figure 12: Comparing ICL's predictions and true labels on amortization-based tasks (trained with single task type).

G EXPERIMENT DETAILS

Our code is provided at `https://anonymous.4open.science/r/code_unicl-D60E/`.

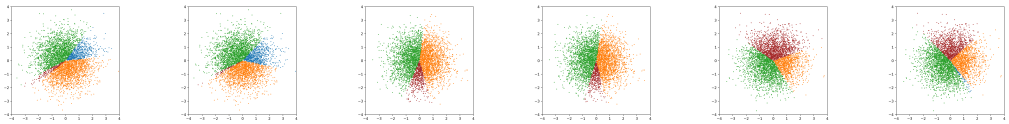
H META-LEVEL META-LEARNING FOR CROSS-DOMAIN FEW-SHOT IMAGE CLASSIFICATION

We investigate the effective of meta-level meta-learning on real-world few-shot image classification problem. We use Meta-Dataset Triantafillou et al. (2019) for training. Because it contains multiple datasets inside each we can sample many few-shot classification tasks, thus can be naturally divided into multiple domains to perform the meta-level meta-training (8).

Following standard settings, we used the training sets of ILSVRC, Omniglot, Aircraft, Birds, Textures, Quick Draw, and Fungi during training. For testing, we used unseen datasets such as Traffic Signs, MSCOCO, and additional datasets like MNIST, CIFAR10, and CIFAR100. Each dataset is treated as a domain for meta-level meta-learning. We considered 5-way 5-shot tasks at the meta-level and 8/16/32 tasks-for-adaptation per domain at the meta-meta-level. The sampling of classes and images to form tasks, as well as the sampling of tasks-for-adaptation within a domain, was random.

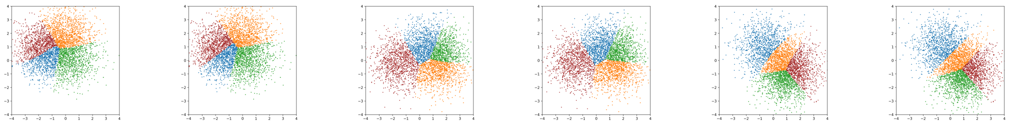
Following the typical setting, in training, we access training sets in ILSVRC, Omniglot, Aircraft, Birds, Textures, Quick Draw, and Fungi. We use unseen Traffic Signs, MSCOCO, and additional MNIST, CIFAR10, CIFAR100 for testing. Note that each of these datasets is viewed as a domains for meta-level meta-learning. We consider 5-way 5-shot tasks at meta-level, and 8/16/32-task-for-adaptation domains at meta-meta-level. The sampling of classes and images to form a task and the and the sample of tasks-for-adaptation in a domain are random. We consider the following baselines:

- ICL w/o adpt: The standard meta-learning setting, where meta-training is performed on all tasks without distinguishing between datasets. During meta-testing, no domain adaptation is performed, meaning that the 8/16/32 tasks are not utilized.
- ICL w/ adpt: The meta-training process is identical to that of ICL w/o adpt. While during meta-testing, the model adapts using 8/16/32 domain-specific tasks by fine-tuning all parameters (step = 5, learning rate = 0.0001, batch size = 8/16/32).
- M^2-ICL: Meta-level meta-training an ICL model following the method introduced in Section 5.1. The domain adaptation process, i.e., the inner-loop of meta-level-MAML is configured as step=5, lr=0.0001, with 16 tasks-for-adaptation per domain. The settings 8/16/32-task-for-adaptation domains at meta-meta-level are meta-meta-trained and evalu-



(a) ICL-1. (b) True-1. (c) ICL-2. (d) True-2. (e) ICL-3. (f) True-3.

Figure 13: Comparing ICL's predictions and true labels on pair-wise metric-based tasks (trained with mixed task type).



(a) ICL-1. (b) True-1. (c) ICL-2. (d) True-2. (e) ICL-3. (f) True-3.

Figure 14: Comparing ICL's predictions and true labels on class-prototype metric-based tasks (trained with mixed task type).

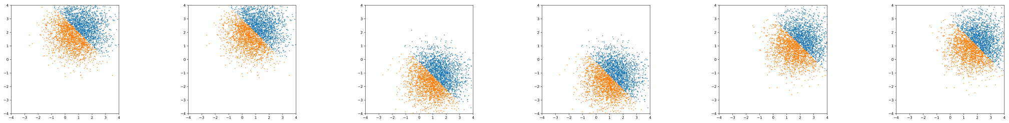
ated respectively. During testing, given 8/16/32 domain-specific tasks, the same adaptation process is applied.

Our implementation builds the ICL model with a 8-layer transformer (without positional encoding), where the input features are 512-dim extracted by a ResNet (resnet-18). Though the model is relatively toy, it is enough to verify the effectiveness of meta-level meta-learning for improving ICL in real-world application: the method pipeline is generalizable and one can replace them with models with more advanced architectures or for other applications.

The results are provided in Table 1. We find that the M^2-ICL significantly outperforms ICL w/o or w/ adpt with any tasks. Specifically, comparing with ICL w/o adpt (standard meat-training and testing), adapting the ICL model with 8 tasks badly harms the performance due to overfitting, and with 16 tasks also do harm, while 32 tasks shows marginally improvement. However, adapting the M^2-ICL model with only 8 tasks is enough to surpasses the average performance, and the growing number of tasks for adaptation brings more significant improvement. ICL w/o adpt, ICL w/ adpt (32 tasks) and M^2-ICL (8 tasks) have comparable performance. This show the proposed meta-level meta-learning is very effective to improve the few-task domain adaptation ability.

Table 1: Cross-domain few-shot image classification.

Method	Traffic Signs	MSCOCO	MNIST	CIFAR10	CIFAR100	Average
ICL w/o adpt	45.4	35.5	88.1	65.2	55.9	58.02
ICL w/ adpt (8 tasks)	41.9	35.1	76.4	64.9	55.5	53.76
ICL w/ adpt (16 tasks)	43.3	36.2	78.5	66.0	56.3	56.06
ICL w/ adpt (32 tasks)	46.1	36.5	83.2	66.8	58.3	58.18
M^2-ICL (8 tasks)	45.9	39.4	86.6	67.4	57.2	59.30
M^2-ICL (16 tasks)	47.5	40.6	88.9	68.0	57.9	60.58
M^2-ICL (32 tasks)	52.6	44.1	91.0	69.4	59.2	**63.26**



(a) ICL-1. (b) True-1. (c) ICL-2. (d) True-2. (e) ICL-3. (f) True-3.

Figure 15: Comparing ICL's predictions and true labels on amortization-based tasks (trained with mixed task type).