In-Context Learning with Hypothesis-Class Guidance

Anonymous Authors[1]

Abstract

Recent research has investigated the underlying mechanisms of in-context learning (ICL) both theoretically and empirically, often using data generated from simple function classes. However, the existing work often focuses on the sequence consists solely of labeled examples, while in practice, labeled examples are typically accompanied by an *instruction*, providing some side information about the task. In this work, we propose *ICL with hypothesis-class guidance (ICL-HCG)*, a novel synthetic data model for ICL where the input context consists of the literal description of a (finite) hypothesis class $\mathcal{H}$ and (x, y) pairs from a hypothesis chosen from $\mathcal{H}$. Under our framework ICL-HCG, we conduct extensive experiments to explore: (i) varied generalization ability to new hypothesis classes; (ii) different model architectures; (iii) sample complexity; (iv) in-context data imbalance; (v) the role of instruction; and (vi) the effect of pretraining data diversity. As a result, we show that (a) Transformers can successfully learn ICL-HCG and generalize to unseen hypotheses and unseen hypothesis classes, and (b) compared with ICL without instruction, ICL-HCG achieves significantly higher accuracy, demonstrating the role of instructions.

1. Introduction

LLMs and ICL Large Language Models (LLMs) (Zhao et al., 2023) have garnered widespread attention for their ability to solve complex tasks using simple text prompts. Among their many capabilities, in-context learning (ICL) (Brown et al., 2020) is particularly striking. ICL enables LLMs to adapt to new tasks by conditioning on provided examples, effectively allowing them to learn from context without explicit parameter updates. Understanding how such behavior emerges in LLMs remains an intriguing and challenging problem.

Existing Efforts for Understanding ICL To elucidate the mechanisms behind ICL, researchers have constructed varied synthetic datasets (Garg et al., 2022; Li et al., 2023; Bai et al., 2023). These datasets typically involve sequences consisting of input-output pairs $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$, where each output $y^{(i)}$ is generated by a simple underlying function $f(\boldsymbol{x}^{(i)})$. For example, Garg et al. (2022) focus on noiseless linear regression, where each input is sampled from an isotropic Gaussian by $\boldsymbol{x}^{(i)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, and the corresponding output is given by $y^{(i)} = \langle \boldsymbol{x}^{(i)}, \boldsymbol{w} \rangle$ with $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$ for each sequence. During training and inference, the model receives a sequence consisting of k demonstration pairs $(\boldsymbol{x}^{(1)}, y^{(1)}, \ldots, \boldsymbol{x}^{(k)}, y^{(k)})$ followed by a query input $\boldsymbol{x}_{\text{query}}$. This setup allows the model to infer the correct response for $\boldsymbol{x}_{\text{query}}$ conditioned on in-context examples. Various extensions have been proposed, including using Gaussian mixtures rather than a single Gaussian for task priors (Lin & Lee, 2024), employing non-linear functions (Bai et al., 2023), and introducing multiple intermediate "chain-of-thought" (Wei et al., 2022) steps within each $(\boldsymbol{x}, \boldsymbol{y})$ pair (Li et al., 2024a).

Motivation While varied data models have been studied to advance our understanding of ICL, a gap remains between these datasets and real-world ICL scenarios. In practice, users often provide LLMs with labeled demonstrations and *an instruction*, containing the descriptions of the task in mind. See Fig. 1 for visualization. Shown on the top left is the case where the user provides a one-shot example of an English-Korean translation pair without any task description. When tested with GPT-4 Legacy model, our demonstration shows that ICL with a task description produces the correct answer, whereas ICL without any task description results in an incorrect output as the model fails to understand the underlying task and tends to answer the English sentence that happened to be a question sentence. Shown on the top right is the case where the user provides a one-shot example preceded by a partial task description – *"perform the translation task based on the demonstration"*. This resulted in a correct translation as shown in the figure. In fact, instructions are known to enhance the accuracy of ICL in general (Brown et al., 2020). However, most existing synthetic data frameworks overlook this crucial aspect,

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

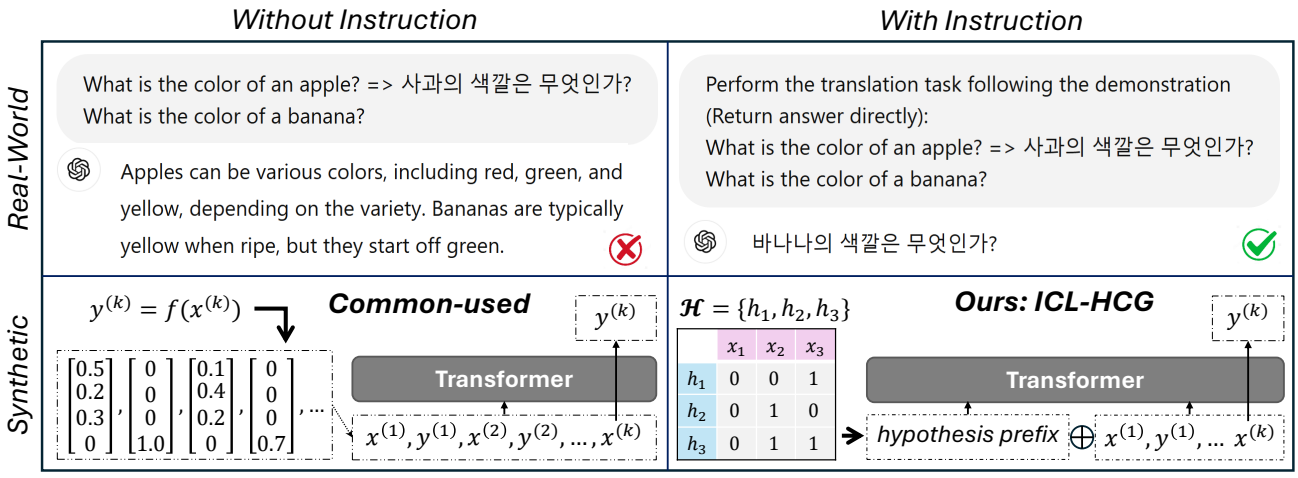


Figure 1: **Common ICL framework *vs.* ours.** Conventional frameworks with synthetic datasets often construct sequences by concatenating multiple $(\boldsymbol{x}, \boldsymbol{y})$ pairs, overlooking the importance of instructions. In contrast, our approach explicitly incorporates instructions through a *hypothesis prefix*. Specifically, we transform the hypothesis class $\mathcal{H}$ into a sequence that is prepended to the sequence of $(\boldsymbol{x}, \boldsymbol{y})$ pairs and then fed into a Transformer. We refer to this method as *in-context learning with hypothesis-class guidance (ICL-HCG)*. (Real-world examples are demonstrated using the GPT-4 Legacy model.)

neglecting the role of instructions in guiding the learning process. Motivated by this limitation, we ask:

Can we design a synthetic data framework for ICL that better captures the practical use scenarios of ICL by incorporating both instructions and labeled samples?

Notably, two recent works (Xuanyuan et al., 2024; Huang & Ge, 2024) adopt prefix as instruction to implicitly provide information on the task. In contrast, our approach explicitly provides a hypothesis class as a prefix to the Transformer, guiding the model's understanding of the intended task.

Our Synthetic Data Model We propose a novel synthetic data model, namely in-context learning with hypothesis-class guidance (ICL-HCG), illustrated in the bottom-right panel of Fig. 1 that integrates a hypothesis class into the ICL procedure. Specifically, in additional to traditionally used sequences constructed from $(\boldsymbol{x}, \boldsymbol{y})$ pairs, a hypothesis class is embedded as hypothesis prefix and fed into the Transformer (more details in Fig. 3 of Sec. 3.5). Leveraging this framework, we explore several aspects of Transformer behavior on the ICL-HCG task: (i) We evaluate the generalization ability of trained models to new hypothesis classes, new hypotheses, and varying sizes of hypothesis classes; (ii) We compare different model architectures (Transformer, Mamba, LSTM, and GRU), highlighting their distinct properties on these generalizations; (iii) We examine the sample complexity required for achieving space and hypothesis generalization, and discover that merely a few dozen training spaces are sufficient for near-perfect generalization. (iv) We examine the effect of imbalanced in-context samples, demonstrating that imbalance can slow down the training process; (v) We assess the benefit of incorporating a hypothesis prefix, which notably enhances the accuracy of ICL; (vi) We show pretraining hypothesis diversity can significantly improve accuracy of ICL when with instruction.

We summarize our contributions as follows:

- We propose a novel synthetic data model, namely in-context learning with hypothesis-class guidance (ICL-HCG) that integrates a hypothesis class into the ICL procedure. This design provides a controlled testbed for varied experiments to study behaviors of ICL with instruction.
- We perform extensive empirical evaluations on our framework. Most interestingly, we demonstrate that (a) Transformers can successfully learn ICL-HCG and such a learned ability can generalize to unseen hypotheses and unseen hypothesis classes, and (b) compared with ICL without instruction, ICL-HCG achieves significantly higher accuracy of ICL, demonstrating the role of instructions.

2. Related Works

Garg et al. (2022) first construct synthetic data under with pretraining and testing sequences generated from noiseless linear regression tasks. Specifically, Garg et al. (2022) samples all input vectors $\boldsymbol{x}$ from an isotropic Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Within each sequence, the outputs are given by $y^{(i)} = \langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle$, where $\boldsymbol{w}$ is drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. For ICL inference, the prompt takes the form $(\boldsymbol{x}^{(1)}, y^{(1)}, \boldsymbol{x}^{(2)}, y^{(2)}, \ldots, \boldsymbol{x}^{(k)}, y^{(k)}, \boldsymbol{x}_{\text{query}}$, where the k pairs $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{k}$ serve as demonstrations

illustrating the linear relationship governed by $\boldsymbol{w}$. The model then predicts the output for $\boldsymbol{x}_{\text{query}}$ using these in-context examples.

Noiseless Linear Regression Based on the well-defined problem setup by Garg et al. (2022) using noiseless linear regression, researchers systematically study the mechanisms of ICL and properties of Transformer. For instance, there is a particular interesting line of research on connecting ICL to gradient descent, firstly hinted by Garg et al. (2022). Akyürek et al. (2023) and von Oswald et al. (2023) then show that one attention layer can be exactly constructed to perform gradient descent, and empirically find similarities between ICL and gradient descent. Further, Ahn et al. (2023) theoretically show that under certain conditions, Transformers trained on noiseless linear regression tasks minimizing the pretraining loss will implement gradient descent algorithm. Nevertheless, Fu et al. (2024) show that Transformers learn to approximate second-order optimization methods for ICL, sharing a similar convergence rate as Iterative Newton's Method. Besides gradient descent, there are lots of other interesting topics on ICL and Transformers based on this linear regression setting, such as looped Transformer (Yang et al., 2024; Gatmiry et al., 2024), training dynamic (Zhang et al., 2024; Huang et al., 2024; Kim & Suzuki, 2024), generalization (Panwar et al., 2024), etc.

Noisy Linear Regression Such a simple noiseless linear regression task is further extended to varied versions. By extending the linear regression to noisy linear regression—$y = \langle \boldsymbol{x}, \boldsymbol{w} \rangle + \epsilon$, Li et al. (2023) analyze the generalization and stability of ICL. Wu et al. (2024) and Raventós et al. (2024) analyze the effect of task diversity on the attention model's ICL risk. Via extending the regression tasks sampling from Gaussian to Gaussian mixture, Lin & Lee (2024) shows ICL exhibits two different modes including task retrieval and learning. With the tasks of $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{\epsilon}$ where $\boldsymbol{W}$ is a matrix rather than a vector, Chen et al. examine the training dynamic of multi-head attention for ICL.

More than Linear Regression Beyond linear regression, researchers are also interested in non-linear regression and classification. The research directions are scattered, and we list them as follows. Bai et al. (2023) show that Transformers can perform in-context algorithm selection, *i.e.*, adaptively selecting different ICL algorithms such as gradient descent, least square, or ridge regression. Bhattamishra et al. (2024) show Transformer can learn a variety of Boolean function classes. Cheng et al. (2024) provide evidence that Transformers can learn to implement gradient descent to enable them to learn non-linear functions. Guo et al. (2024) show that trained Transformer achieves near-optimal ICL performance under $y = \langle \boldsymbol{w}, f(\boldsymbol{x}) \rangle$, where f is a shallow neural network (MLP). Examining linear and non-linear regression tasks, Fan et al. (2024) and Tripuraneni et al. (2023) show Transformer can perform ICL on composited or mixed tasks of pretrained linear or non-linear tasks, and Yadlowsky et al. (2024) examine whether trained Transformers can generalize to new tasks beyond pretraining. Park et al. (2024) examine whether Mamba can in-context learn varied synthetic tasks. Via examining regression and classification tasks, Kim et al. (2024) show task diversity helps shorten the ICL plateau pretraining. Ramesh et al. (2024) assume there are multiple functions composited to connect $\boldsymbol{x}$ and $\boldsymbol{y}$ pair, *e.g.*, $\boldsymbol{y} = f_1 \circ f_2 \circ f_3(\boldsymbol{x})$ to study the compositional capabilities of Trasnformer. Li et al. (2024b) study how non-linear Transformer learns binary classification.

Synthetic Dataset with Instruction As far as the authors know, there are two articles on synthetic datasets with instructions. Huang & Ge (2024) append an additional vector $\boldsymbol{\mu}$ to the sequences with interleaved input-output format, which leads to the sequence $(\boldsymbol{\mu}, \boldsymbol{x}^{(1)}, \boldsymbol{w}^\top \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{w}^\top \boldsymbol{x}^{(2)}, \dots)$ in which $\boldsymbol{x}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{I})$, and show that the trained Transformer can achieve significantly lower loss on ICL when the task descriptor $\boldsymbol{\mu}$ is provided. Xuanyuan et al. (2024) is the work most closely related to us, who develop a new synthetic dataset based on task $((a \cdot x) \circ (b \cdot y)) \mod p = r$, where (x, y) is the input, r is the output, $\circ$ is an operation $(+, -, /)$, and each task is defined by the parameters $(a, b, \circ)$ (p is a constant). The instruction is constructed as $(a_l, a_u, b_l, b_u, \circ)$, where a_l and a_u are the lower and upper bounds of a (similar for b), and $\circ$ is the operation. Therefore, the instruction constrains the possible tasks, *i.e.*, provide information on the underlying true task of in-context samples. With such a setting, Xuanyuan et al. (2024) study how the information provided by instruction affects the ICL accuracy.

3. Meta-Learning for ICL-HCG

3.1. Two Types of Tasks in ICL-HCG

We consider two types of tasks in ICL-HCG, both constructed from a finite hypothesis class $\mathcal{H} = \{h^{(1)}, h^{(2)}, \dots\}$ over a finite input space $\mathcal{X} = \{x_1, x_2, \dots, x_{|\mathcal{X}|}\}$ and a binary output space $\mathcal{Y} = \{0, 1\}$.

Type I (label prediction) Consider a hypothesis class $\mathcal{H}$ and a sequence consisting of training data and a test point

$$S_{k-1} \oplus x^{(k)} = (x^{(1)}, y^{(1)}, \dots, x^{(k-1)}, y^{(k-1)}, x^{(k)})$$

where for all i, $y^{(i)} = h(x^{(i)})$ for a specific $h \in \mathcal{H}$, and $x^{(k)}$ is a test query input. The objective is to predict the label

$$y^{(k)} = h\big(x^{(k)}\big).$$

We refer to this as Type I, with input-output pairs:

$$i_{\mathrm{I},k} = \big(\mathcal{H}, S_{k-1} \oplus x^{(k)}\big), \quad o_{\mathrm{I},k} = y^{(k)}.$$

Type II (hypothesis identification) Given a hypothesis class $\mathcal{H}$ and a sequence (namely *ICL sequence*)

$$S_K = (x^{(1)}, y^{(1)}, \ldots, x^{(K)}, y^{(K)}),$$

where for all i, $y^{(i)} = h(x^{(i)})$ for a specific $h \in \mathcal{H}$, the goal is to identify the underlying hypothesis h. Denote this as Type II, with:

$$i_{\text{II}} = \big(\mathcal{H}, S_K\big), \quad o_{\text{II}} = h.$$

Meta-learning Type I uses $k-1$ training examples to predict the label of a new query $x^{(k)}$, while Type II directly outputs h. Both Type I and II can be viewed as attempts to learn about h via empirical risk minimization (ERM) on the dataset $\{(x^{(i)}, y^{(i)})\}$. Our meta-learning aims to learn to do ERM for different hypothesis classes when these hypothesis classes are given as input along with (x, y) pairs.

3.2. Sample Generation

We consider the following two approaches for generating samples of Type I and Type II tasks.

Assumption 1 (i.i.d. Generation). Given hypothesis classes $\{\mathcal{H}_i\}_{i=1}^{N}$, input space $\mathcal{X}$, and an integer K:
(a) Sample a hypothesis class $\mathcal{H}$ from $\{\mathcal{H}_i\}_{i=1}^{N^{\text{train}}}$;
(b) Sample a hypothesis h uniformly at random from $\mathcal{H}$;
(c) Sample K inputs $\{x^{(i)}\}_{i=1}^{K}$ i.i.d. from $\text{Uniform}(\mathcal{X})$;
(d) Generate $y^{(i)} = h(x^{(i)})$ for each $i \in [K]$;
(e) Form $S_{k-1} \oplus x^{(k)} = [x^{(1)}, y^{(1)}, \ldots, x^{(k)}]$ for Type I;
(f) Form $S_K = [x^{(1)}, y^{(1)}, \ldots, x^{(K)}, y^{(K)}]$ for Type II.

Assumption 2 (Opt-T Generation). Given hypothesis classes $\{\mathcal{H}_i\}_{i=1}^{N}$, input space $\mathcal{X}$, and an integer K:
(a) Sample a hypothesis class $\mathcal{H}$ from $\{\mathcal{H}_i\}_{i=1}^{N^{\text{test}}}$;
(b) Sample a hypothesis h uniformly at random from $\mathcal{H}$;
(c) Construct *optimal teaching set* of h with respect to $\mathcal{H}$[1];
(d) Randomly duplicate elements from this optimal teaching set until its size reaches K. Assign indices 1 through K arbitrarily to these (x, y) pairs;
(e) Form $S_K = [x^{(1)}, y^{(1)}, \ldots, x^{(K)}, y^{(K)}]$.

3.3. Meta Training and Testing

Training Given a set of training hypothesis classes $\{\mathcal{H}_i^{\text{train}}\}_{i=1}^{M_{\text{train}}}$, the meta-learner is trained in a multi-task setting to minimize the following loss:

$$\mathcal{L} = \mathcal{L}_1(f_\theta(i_{\text{II}}), o_{\text{II}}) + \sum_{k=1}^{K} \mathcal{L}_2(f_\theta(i_{\text{I},k}), o_{\text{I},k}), \qquad (1)$$

where we generate $\mathcal{H}$, h, and S_K following i.i.d. Generation 1, inherently defining $(i_{\text{II}}, o_{\text{II}})$ and $(i_{\text{I},k}, o_{\text{I},k})$. The loss is indeed implemented with additional terms, and we will further clarify the loss in Sec. 3.5, Eq. 2.

[1]The optimal teaching set is the smallest set of (x, y) pairs that uniquely identifies h among all candidates in $\mathcal{H}$.

Testing Given a set of testing hypothesis classes $\{\mathcal{H}_i^{\text{test}}\}_{i=1}^{M_{\text{test}}}$, we consider two types of testing.

- **Type I** (label prediction): We generate $(i_{\text{I},k}, o_{\text{I},k})$ following i.i.d. Generation 1, and then measure whether the learner f predict $f(i_{\text{I},k})$ correctly for each $k \in [K]$;
- **Type II** (hypothesis identification): We generate $(i_{\text{II}}, o_{\text{II}})$ using Opt-T Generation 2 and evaluate whether the learner f predicts $f(i_{\text{II}})$ correctly. This setting tests whether the learner has truly acquired the ability to identify the underlying hypothesis with minimal sample information.

3.4. Four Types of Generalization

Hypothesis universe $\mathcal{H}^{\text{uni}}$ Given an input space $\mathcal{X} = \{x_1, x_2, \ldots, x_{|\mathcal{X}|}\}$ and a binary output space $\mathcal{Y} = \{0, 1\}$, We define the hypothesis universe $\mathcal{H}^{\text{uni}} = \mathcal{Y}^{\mathcal{X}}$ as the collection of all possible binary classification hypotheses. This universe contains $M = 2^{|\mathcal{X}|}$ distinct hypotheses, serving as a hypothesis pool to constructing training and testing hypothesis classes.

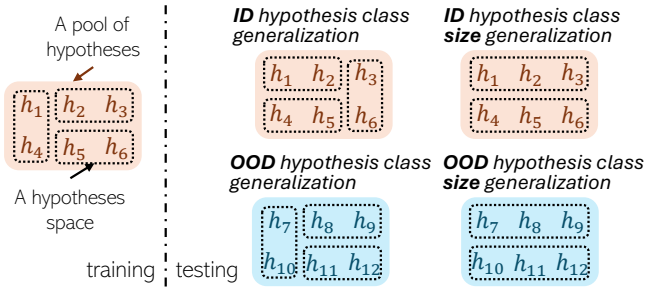


Figure 2: **Four types of generalization.** An illustration of the four types of generalization.

Recall that, in meta-learning, the goal is to learn a model that can swiftly adapt to new tasks by leveraging knowledge from previously encountered tasks. Below, we describe four different types of generalization for meta ICL learning based on different training and testing meta task distribution. First, we focus on whether the learner can generalize to a testing hypothesis class that may or may not overlap with the training hypothesis class, namely in-distribution (ID) and out-of-distribution (OOD) hypothesis class generalization, respectively.

Definition 3.1 (ID Hypothesis Class Generalization)**.** Given $\mathcal{H}^{\text{uni}}$ of size M, we enumformerate all $C(M, m) = \frac{M!}{m!(M-m)!}$ distinct hypothesis classes, each containing m hypotheses. We then *subsample* these classes into disjoint training and testing subsets, ensuring that no testing hypothesis class appears in the training set (although individual

hypotheses may overlap). By training on the selected training hypothesis classes and evaluating on the unseen testing hypothesis classes, we assess generalization to new hypothesis classes consisting of ID hypotheses.

Definition 3.2 (OOD Hypothesis Class Generalization)**.** Given $\mathcal{H}^{\text{uni}}$ of size M, we partition it into disjoint training and testing subsets of sizes M^{Train} and M^{test}, respectively. We then generate training and testing hypothesis classes, each containing m hypotheses, exclusively from the training or testing subsets. We train the learner on the training hypothesis classes and evaluate on the testing hypothesis classes. Because no testing hypothesis appears during training, this setup probes how well the learner generalizes to entirely new hypotheses, *i.e.*, OOD hypothesis.

We then consider whether the learner can generalize to hypothesis classes of varying sizes. Building on the concepts of ID and OOD hypothesis class generalization, we introduce size generalizations as follows.

Definition 3.3 (ID Hypothesis Class Size Generalization)**.** Building on the setting of ID hypothesis class generalization, while maintaining non-identical training and testing hypothesis classes, we allow training hypothesis class to include varied number of hypotheses $m \in \mathcal{M} \subset [L]$. We investigate whether the learner can perform well on hypothesis classes with other sizes $m \in [L] \setminus \mathcal{M}$, where $[L] = \{1, 2, \ldots, L\}$.

Definition 3.4 (OOD Hypothesis Class Size Generalization)**.** Based on the setting of OOD hypothesis class generalization, while maintaining non-identical training and testing hypotheses, we allow training hypothesis class to include varied number of hypotheses $m \in \mathcal{M} \subset [L]$. We investigate whether the learner can perform well on hypothesis classes with varied sizes $m \in [L] \setminus \mathcal{M}$, where $[L] = \{1, 2, \ldots, L\}$.

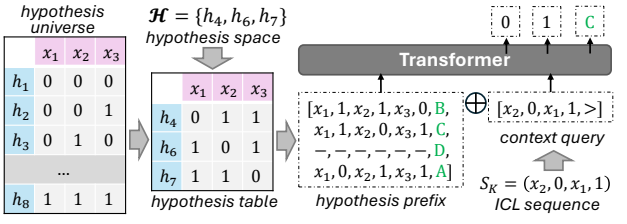


Figure 3: **Learning ICL-HCG via Transformer.** We begin by sampling a subset from the hypothesis universe as the hypothesis class $\mathcal{H}$. Next, we encode the hypothesis class $\mathcal{H}$ and concatenate it with context query into a unified sequences of token. This sequences is fed into a Transformer model for training with next-token prediction, and testing for evaluating the accuracy on y and hypothesis identification. (This figure is an simplified illustration. Please refer to appendix B and Fig. 12 for the full details.)

3.5. Learning ICL-HCG via Transformer

This section details how Transformer learns ICL-HCG. As shown in Fig. 3, the hypothesis class $\mathcal{H}$ is first converted to a hypothesis prefix with randomly assigned hypothesis indexes, then concatenated with context query representing sequence S_K as a unified sequence s.

Hypothesis prefix[2] Given a hypothesis class $\mathcal{H} = \{h_4, h_6, h_7\}$, its hypothesis prefix with size $L = 4$ is constructed as shown in Fig. 3. Blank hypothesis is used to fill the hypothesis prefix when $|\mathcal{H}| < L$. A randomly assigned hypothesis index token z is used to label each hypothesis. As illustrated in Fig. 3, zs are assigned from a pool of size L, {"A","B","C","D"} without replacement[3].

Context query Given an ICL sequence S_K, we append a query token ">" after it to trigger trigger the prediction of the hypothesis index ss shown in Fig. 3. We name the combination of S_K and ">" as context query.

The Transformer predicts the y tokens in the context query based on previous tokens and the index z of the underlying hypothesis based on all tokens in the sequence. The training loss in Eq. 1 is further detailed as below:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P_\theta(s_i \mid s_{<i}). \tag{2}$$

We summarize the pipeline in the Appendix A Algorithm 1.

4. Experiments

4.1. Setting of Experiments

Pretraining During pretraining, we backpropagate gradients *based on next-token prediction for all tokens*. Each training sequence s consists of a hypothesis prefix, a context query, and a hypothesis index token. As illustrated in Fig. 3, we feed the entire sequence s (excluding the final index token z into the Transformer. We then compute cross-entropy loss for each subsequent token (excluding the very first). Refer to Appendix D.1 for training hyperparameters.

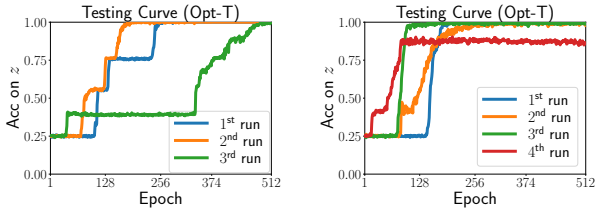
Components of pretraining loss We conducted experiments to determine the optimal components to include in the pretraining loss. Specifically, we evaluated four configurations: applying the loss (i) solely to the final hypothesis index token, (ii) exclusively to the content query, (iii) only to the label y of the content query, and (iv) across all tokens. We empirically find that incorporating the loss across all tokens in the sequence leads to the best performance.

[2]Please refer to appendix B for the full version.

[3]We use variable z to represent the hypothesis index, and create a set of L hypothesis index tokens as a pool from which each hypothesis is randomly assigned a unique index without replacement.

4.2. Four Types of Generalization

This section investigates whether a Transformer trained on ICL-HCG tasks can generalize to new tasks, *i.e.*, new hypothesis spaces. We explore four types of generalization scenarios, defined in Definitions 3.1, 3.2, 3.3, and 3.4. Detailed hyperparameters of settings are provided in Appendix D.2.



(a) testing curves of ID hypothesis class generalization. (b) testing curves of OOD hypothesis class generalization.

Figure 4: **Multiple runs on ID and OOD hypothesis class generalizations.** Transformer successfully learns ICL-HCG, and generalize to new hypothesis classes and new hypotheses. Refer to Appendix C.1, Fig. 13 and 14 for more curves of loss, training accuracy and testing accuracy (i.i.d.).

> **Finding 1:** *Transformer can successfully be trained to learn ICL-HCG tasks and such a learned ability can generalize to new hypothesis, hypothesis class, and hypothesis size.*

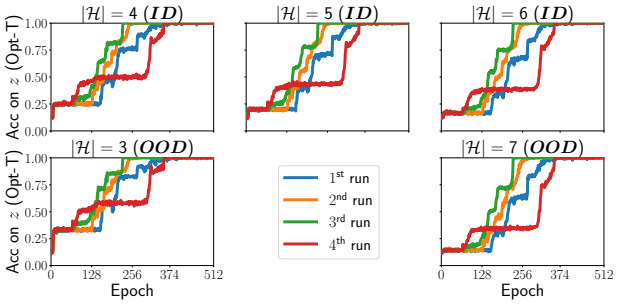
We first demonstrate that the Transformer successfully learns ICL-HCG and that this capability generalizes effectively on ID and OOD hypothesis class generalizations, as illustrated in Fig. 4(a) and 4(b). Furthermore, we show that the learned ICL-HCG ability generalizes to hypothesis classes of varying sizes, as depicted in Fig. 5(a) and 5(b). The accuracy curves exhibit multiple plateaus, with the duration of each plateau varying across different runs.

4.3. Model Architecture Comparisons

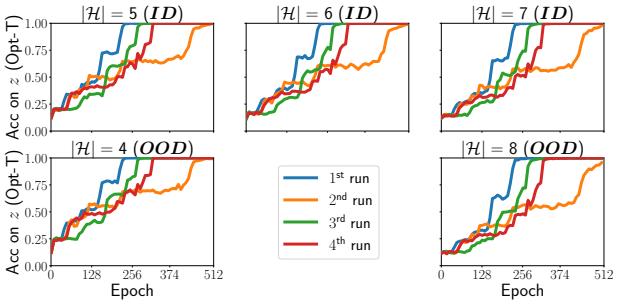
We compare Transformer with other model architectures, including Mamba (Gu & Dao, 2023), LSTM (Hochreiter, 1997), and GRU (Cho et al., 2014). We investigate whether each model can effectively fit the training dataset and, if so, generalize to the four types of unseen hypothesis classes.

> **Finding 2:** *Mamba performs well on four types of generalization tasks. In contrast, LSTM and GRU only perform well on ID hypothesis class generalization but fail on the others.*

We begin by evaluating ID and OOD hypothesis class generalization across different model architectures. Using the hyperparameter search space detailed in Appendix D.1, we

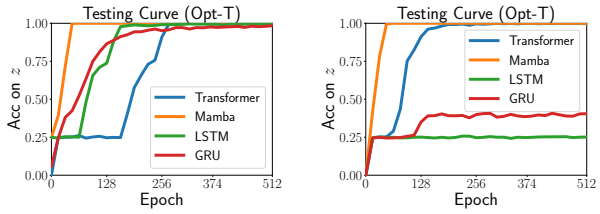


(a) testing curves of ID hypothesis class generalization.



(b) testing curves of OOD hypothesis class generalization.

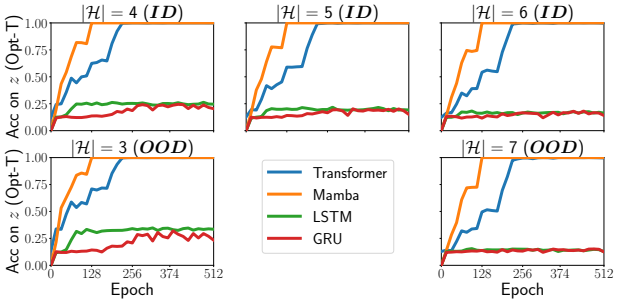
Figure 5: **Multiple runs on ID and OOD hypothesis class size generalizations.** For space&size generalization, the Transformer trained on hypothesis spaces with sizes $m^{\text{train}} \in \{4, 5, 6\}$ (ID, in-distribution) successfully generalizes to hypothesis spaces with sizes $m^{\text{test}} \in \{3, 7\}$ (OOD, out-of-distribution). Similarly, For hypothesis&size generalization, the Transformer trained on hypothesis spaces with sizes $m^{\text{train}} \in \{5, 6, 7\}$ (ID) successfully generalizes to hypothesis spaces with sizes $m^{\text{test}} \in \{4, 8\}$ (OOD).



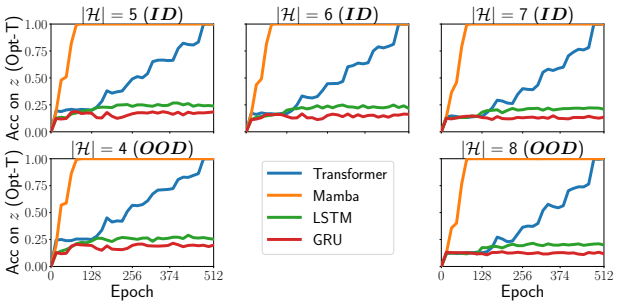
(a) testing curves of ID hypothesis class generalization. (b) testing curves of OOD hypothesis class generalization.

Figure 6: **Various models on ID and OOD hypothesis class generalization.** All models effectively learn ICL-HCG, consistently achieving high accuracy on ID hypothesis class generalization. However, under OOD hypothesis class generalization, only Transformer and Mamba succeed. Refer to Appendix C.2, Fig. 15 and 16 for more curves including loss, training accuracy and testing accuracy (i.i.d.).

find that the four models—Transformer, Mamba, LSTM, and GRU—generalize well on ID hypothesis class generalization tasks, as illustrated in Fig. 6(a). In contrast, on OOD hypothesis class generalization, GRU is able to fit the training set but fail to generalize to the testing set, while LSTM is unable to fit the training set, as shown in Fig. 6(b). Furthermore, on both ID and OOD hypothesis class size generalization (see Fig. 7(a) and 7(b)), LSTM and GRU are unable to fit the training set. On the other hand, Mamba not only generalizes effectively across all four generalization setups but also demonstrates faster convergence compared to Transformer.



(a) testing curves of ID hypothesis class size generalization.



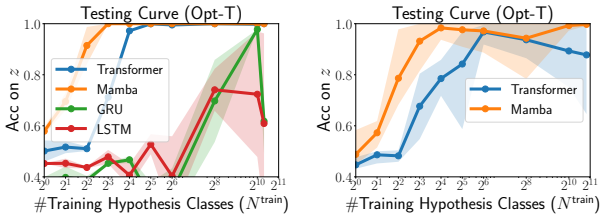
(b) testing curves of OOD hypothesis class size generalization.

Figure 7: **Various models on ID and OOD hypothesis class size generalization.** In both settings, Transformer and Mamba demonstrate strong generalization, whereas LSTM and GRU fail to do so. Additionally, Mamba requires fewer epochs to converge compared to the Transformer. As illustrated in Fig. 17 in Appendix C.2, LSTM and GRU are unable to fit the training data.

4.4. Effect of Training Hypothesis Class Count

We evaluate how the number of training hypothesis spaces affects the space and hypothesis generalization abilities.

> **Finding 3:** *Mamba is more sample efficient than Transformer on ICL-HCG tasks, and achieves near perfect generalization with only tens of pretraining hypothesis classes.*



(a) space generalization. (b) hypothesis generalization.

Figure 8: **Effect of training hypothesis class count.** Transformer and Mamba trained on ICL-HCG tasks generalize to new hypothesis classes with only 8 to 32 training hypothesis classes. Refer to Appendix C.3, Fig. 18 and 19 for training accuracy and testing accuracy (Opt-T).

In Fig. 8(a), we evaluate Mamba, Transformer, GRU, and LSTM, and observe that with only 2^3 and 2^5 training hypothesis classes, Mamba and Transformer can achieve nearly perfect ID hypothesis class generalization. In contrast, LSTM and GRU require more samples to achieve good generalization and the results have high variance. In Fig. 8(b), since GRU and LSTM fail under hypothesis generalization, we evaluate Mamba and Transformer. Results show that Mamba and Transformer can achieve nearly perfect OOD hypothesis class generalization with only 2^4 and 2^6 training hypothesis spaces,.

4.5. Effect of Imbalanced In-Context Samples

In this section, we investigate how an imbalanced sample distribution in the context query affects training procedure. Specifically, we consider the following distribution over $\mathcal{X}$:

$$\text{norm}(\underbrace{\frac{1}{\sqrt{\mathrm{D}}},\ldots,\frac{1}{\sqrt{\mathrm{D}}}}_{\lfloor\frac{|\mathcal{X}|}{2}\rfloor\text{ items}},\underbrace{1}_{(|\mathcal{X}|\bmod 2)\text{ items}},\underbrace{\sqrt{\mathrm{D}},\ldots,\sqrt{\mathrm{D}}}_{\lfloor\frac{|\mathcal{X}|}{2}\rfloor\text{ items}}),$$

where D[4] represents the disparity of the distribution over $\mathcal{X}$, *i.e.*, $\mathrm{D} = \frac{\max_{x\in\mathcal{X}} P(x)}{\min_{x\in\mathcal{X}} P(x)}$.

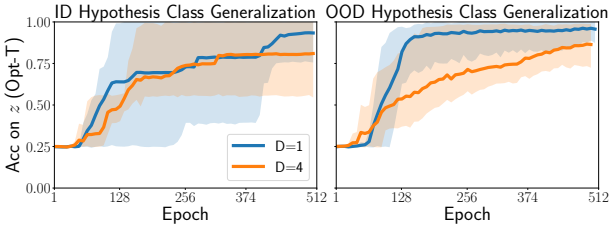


Figure 9: **The effect of sample imbalance.** Sample imbalance leads to lower convergence speed.

[4]The notation D is distinguished from token "D" by color and dataset $\mathcal{D}$ by format.

Finding 4: *In-context sample imbalance lags the convergence of training.*

We evaluated how the imbalance affects the training procedure in Fig. 9 through varied D values, showing that such an imbalance lags the convergence of the training process.

4.6. The Benefit of Hypothesis Prefix

In this section, we demonstrate how hypothesis prefix influences the performance of ICL. We compare ICL accuracy on y with hypothesis prefix and without hypothesis prefix, under the setting of ID hypothesis class generalization.

Finding 5: *Incorporating hypothesis prefix as instruction significantly boost the accuracy of ICL.*

As shown in Fig. 10, the hypothesis prefix significantly enhances the training and testing accuracy on y of ICL.

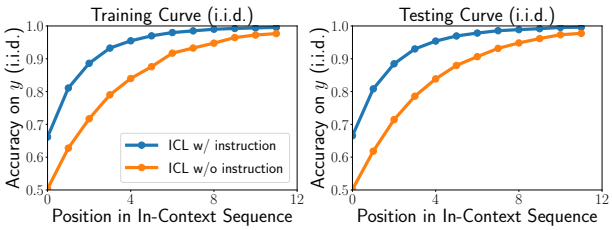


Figure 10: **The effect of instruction.** Under ID hypothesis class generalization, providing an instruction (hypothesis prefix) significantly boosts ICL performance, especially when the y token appears early (indicating only a few demonstration examples precede it).

4.7. The Effect of Pretraining Hypothesis Diversity

In this section, we investigate the impact of hypothesis diversity combined with instruction (hypothesis prefix) on the accuracy of ICL. We conduct experiments under OOD hypothesis class generalization with an input space size of $|\mathcal{X}| = 6$, resulting in a hypothesis universe $\mathcal{H}^{\text{uni}}$ comprising $2^{|\mathcal{X}|} = 64$ hypotheses. For training, we sample $M^{\text{train}} \in \{4, 8, 16, 32\}$ hypotheses from $\mathcal{H}^{\text{uni}}$, and for testing, we select $M^{\text{test}} = 16$ hypotheses from the remaining pool.

Finding 6: *Increasing the diversity of pretraining hypotheses significantly boosts the performance of ICL when instructions are provided.*

As illustrated in Fig. 11, under OOD hypothesis class generalization, the Transformer trained with instructions achieves similar ICL accuracy to a standard ICL approach when pretraining hypothesis diversity is low, but significantly outperforms it when pretraining hypothesis diversity is high. Notably, the testing ICL samples are derived from unseen hypotheses, indicating that incorporating instructions can enhance ICL performance even for new hypotheses.

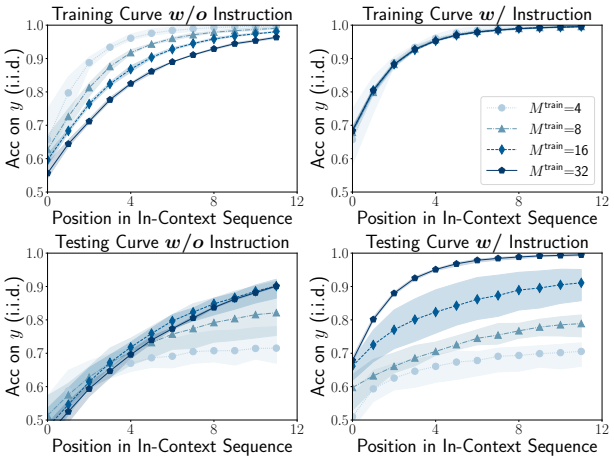


Figure 11: **The effect of pretraining hypothesis diversity.** Under hypothesis generalization, increasing the diversity of pretraining hypotheses significantly boosts the performance of ICL when instructions are provided. However, without instructions, this effect is limited.

5. Conclusion

In this paper, we introduced a novel instruction-based in-context learning (ICL) framework that explicitly integrates a hypothesis class as the instruction, namely in-context learning with hypothesis class guidance (ICL-HCG). Through a series of diverse experiments, we demonstrated that Transformers trained on ICL-HCG tasks can generalize to new hypothesis classes and new hypotheses, even when trained on only a few hypothesis spaces. Moreover, we show that the incorporation of such instructions significantly enhances the accuracy of ICL, thereby bridging the gap between synthetic ICL studies and real-world applications. We also examined the effect of hypothesis diversity within this framework and found that increased hypothesis diversity substantially improves ICL accuracy especially when instructions are provided. Our framework serves as a platform for diverse explorations, offering new insights and serving as a playground for research on ICL and large language models (LLMs).

We conclude our paper by acknowledging the limitations of our current framework: (i) Our study is confined to finite hypothesis binary classification problems, which can be extended to more complex scenarios; (ii) The hypothesis prefix is assumed to provide an explicit hypothesis space, differing from the more implicit instructions used in real-world LLM applications.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ahn, K., Cheng, X., Daneshmand, H., and Sra, S. Transformers learn to implement preconditioned gradient descent for in-context learning. In *Advances in Neural Information Processing Systems*, 2023.

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? Investigations with linear models. In *International Conference on Learning Representations*, 2023.

Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Advances in Neural Information Processing Systems*, 2023.

Bhattamishra, S., Patel, A., Blunsom, P., and Kanade, V. Understanding in-context learning in Transformers and llms by learning to learn discrete functions. In *International Conference on Learning Representations*, 2024.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Chen, S., Sheen, H., Wang, T., and Yang, Z. Training dynamics of multi-head softmax attention for in-context learning: Emergence, convergence, and optimality (extended abstract). In *Annual Conference on Learning Theory*. PMLR.

Cheng, X., Chen, Y., and Sra, S. Transformers implement functional gradient descent to learn non-linear functions in context. In *International Conference on Machine Learning*, 2024.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Fan, Y., Yadlowsky, S., Papailiopoulos, D., and Lee, K. Transformers can learn meta-skills for task generalization in in-context learning. In *NeurIPS Workshop on Compositional Learning: Perspectives, Methods, and Paths Forward*, 2024.

Fu, D., Chen, T.-Q., Jia, R., and Sharan, V. Transformers learn to achieve second-order convergence rates for in-context linear regression. In *Advances in Neural Information Processing Systems*, 2024.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can Transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, 2022.

Gatmiry, K., Saunshi, N., Reddi, S. J., Jegelka, S., and Kumar, S. Can looped Transformers learn to implement multi-step gradient descent for in-context learning? In *International Conference on Machine Learning*, 2024.

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

Guo, T., Hu, W., Mei, S., Wang, H., Xiong, C., Savarese, S., and Bai, Y. How do Transformers learn in-context beyond simple functions? A case study on learning with representations. In *International Conference on Learning Representations*, 2024.

Hochreiter, S. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Huang, R. and Ge, R. Task descriptors help Transformers learn linear models in-context. In *ICML Workshop on In-Context Learning*, 2024.

Huang, Y., Cheng, Y., and Liang, Y. In-context convergence of Transformers. In *International Conference on Machine Learning*, 2024.

Kim, J. and Suzuki, T. Transformers learn nonlinear features in context: Nonconvex mean-field dynamics on the attention landscape. In *International Conference on Machine Learning*, 2024.

Kim, J., Kwon, S., Choi, J. Y., Park, J., Cho, J., Lee, J. D., and Ryu, E. K. Task diversity shortens the ICL plateau. *arXiv preprint arXiv:2410.05448*, 2024.

Li, H., Wang, M., Lu, S., Cui, X., and Chen, P.-Y. How do nonlinear Transformers acquire generalization-guaranteed CoT ability? In *ICML Workshop on High-dimensional Learning Dynamics: The Emergence of Structure and Reasoning*, 2024a.

Li, H., Wang, M., Lu, S., Cui, X., and Chen, P.-Y. How do nonlinear Transformers learn and generalize in in-context learning? In *International Conference on Machine Learning*, 2024b.

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, 2023.

Lin, Z. and Lee, K. Dual operating modes of in-context learning. In *International Conference on Machine Learning*, 2024.

Panwar, M., Ahuja, K., and Goyal, N. In-context learning through the Bayesian prism. In *International Conference on Learning Representations*, 2024.

Park, J., Park, J., Xiong, Z., Lee, N., Cho, J., Oymak, S., Lee, K., and Papailiopoulos, D. Can Mamba learn how to learn? A comparative study on in-context learning tasks. In *International Conference on Machine Learning*, 2024.

Ramesh, R., Lubana, E. S., Khona, M., Dick, R. P., and Tanaka, H. Compositional capabilities of autoregressive Transformers: A study on synthetic, interpretable tasks. In *International Conference on Machine Learning*, 2024.

Raventós, A., Paul, M., Chen, F., and Ganguli, S. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. In *Advances in Neural Information Processing Systems*, 2024.

Tripuraneni, N., Doshi, L., and Yadlowsky, S. Can Transformers in-context learn task mixtures? In *NeurIPS Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.

von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in neural information processing systems*, 2022.

Wu, J., Zou, D., Chen, Z., Braverman, V., Gu, Q., and Bartlett, P. L. How many pretraining tasks are needed for in-context learning of linear regression? In *International Conference on Learning Representations*, 2024.

Xuanyuan, M., Yang, T., Fu, J., and Wang, Y. On task description of in-context learning: A study from information perspective, 2024.

Yadlowsky, S., Doshi, L., and Tripuraneni, N. Can Transformer models generalize via in-context learning beyond pretraining data? In *NeurIPS Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2024.

Yang, L., Lee, K., Nowak, R. D., and Papailiopoulos, D. Looped Transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024.

Zhang, R., Frei, S., and Bartlett, P. L. Trained Transformers learn linear models in-context. *Journal of Machine Learning Research*, 2024.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

A. Pseudo Algorithm for ICL-HCG

We summarize our meta framework for ICL-HCG in Algorithm 1.

Algorithm 1 Meta Framework for ICL-HCG

Inputs: a set of inputs $\mathcal{X}$, a training set of hypothesis classes $\mathcal{S}^{\text{train}} = \{\mathcal{H}_i^{\text{train}}\}_{i=1}^{N^{\text{train}}}$, a testing set of hypothesis classes $\mathcal{S}^{\text{test}} = \{\mathcal{H}_i^{\text{test}}\}_{i=1}^{N^{\text{test}}}$, batch size B, hypothesis prefix size L, and context query size K
for training epoch do
  sample $\{\mathcal{H}_i\}_{i=1}^{B} \overset{\text{i.i.d.}}{\sim} \text{Uniform}(\mathcal{S}^{\text{train}})$
  for each hypothesis class $\mathcal{H} \in \{\mathcal{H}_i\}_{i=1}^{B}$ **do**
    generate h, S_K following i.i.d. Generation 1
    // Construct sequence based on $\mathcal{H}$, h, and S_K
    construct hypothesis prefix, context query, and hypothesis index z based on $\mathcal{H}$, h, S_K
    $s \leftarrow$ concatenate(hypothesis prefix, context query, z)
    // Cross-entropy loss for next token prediction
    $\mathcal{L} \leftarrow -\sum_{t=2}^{|s|} \log P(s_t \mid s_{<t})$
  end for
  update model parameters using $\mathcal{L}$ of the batch
end for
for testing epoch do
  sample $\{\mathcal{H}_i\}_{i=1}^{B} \overset{\text{i.i.d.}}{\sim} \text{Uniform}(\mathcal{S}^{\text{test}})$
  for each hypothesis class $\mathcal{H} \in \{\mathcal{H}_i\}_{i=1}^{B}$ **do**
    generate h, S_K via:
      either following i.i.d. Generation 1
      or following Opt-T Generation 2
    construct sequence s based on $\mathcal{H}$, h, and S_K
    evaluate the prediction accuracy on y, z, etc
  end for
end for

B. Implementation Detail of Hypothesis Prefix and Context Query

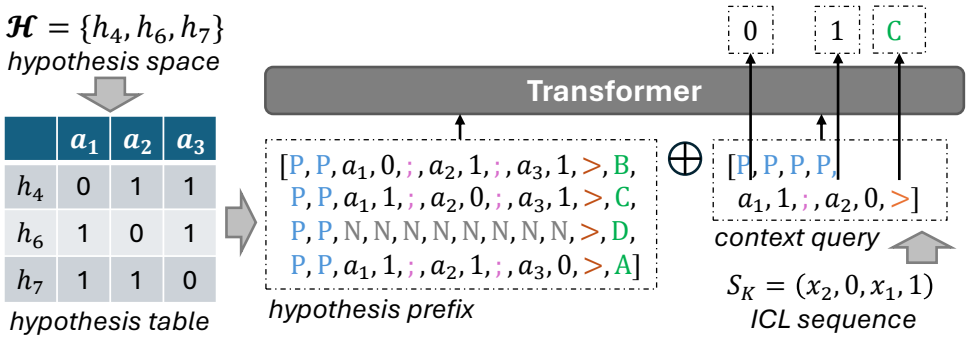


Figure 12: **The framework.** We convert hypothesis class $\mathcal{H}$ and ICL sequence S_K into sequences of tokens, concatenate them and input to Transformer. Then we examine whether Transformer can predict correct y and z values.

Hypothesis prefix Given a hypothesis class $\mathcal{H}$ and its hypothesis table, the correspongding hypothesis prefix with hypothesis prefix's content length L is constructed as shown in Fig. 12. The token "P" serves as the padding token to separate hypotheses, the token ";" serves as the separation token to separate (x, y) pairs, the token "N" serves as the empty token to fill a blank hypothesis, and the token ">" is used to connect (x, y) pairs of the hypothesis to a randomly assigned

hypothesis index z[5]. In the illustrated example in Fig. 12, the randomly assigned indexes zs are sampled from $M = 4$ hypothesis index tokens {"A","B","C","D"} without replacement[6].

Context Query Given an ICL sequence S_K with K pairs of (x, y), the context query of size K is constructed to represent the ICL sequence and trigger the prediction of the hypothesis index with padding token "P", separation token token ";", and query token ">" as shown in Fig. 12.

C. Additional Details of Experiments

C.1. Four Types of Generalization

We share more training and testing curves in Fig. 13 and 14 to provide additional results to Fig. 4(a) and 4(b).

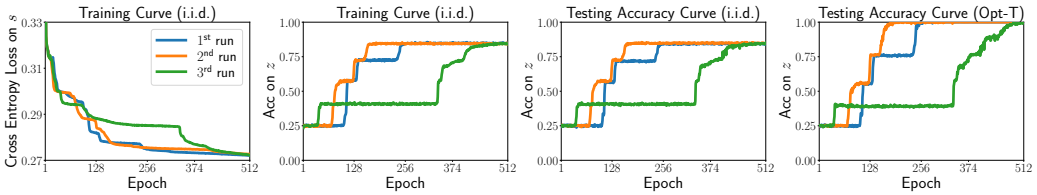


Figure 13: **Multiple runs for ID hypothesis class generalization.**

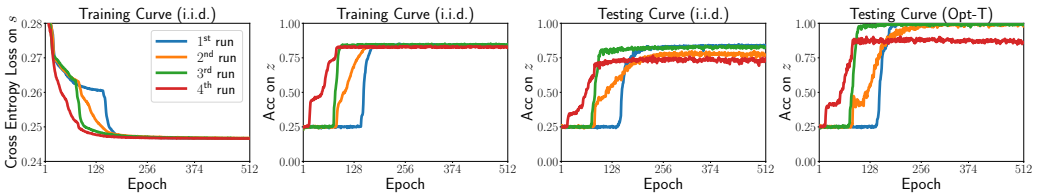


Figure 14: **Multiple runs for OOD hypothesis class generalization.**

C.2. Compare with Other Model Architectures

We share more training and testing curves in Fig. 15, 16, 17(a) and 17(b) to provide additional results to Fig. 6(a), 6(b), 7(a) and 7(b).

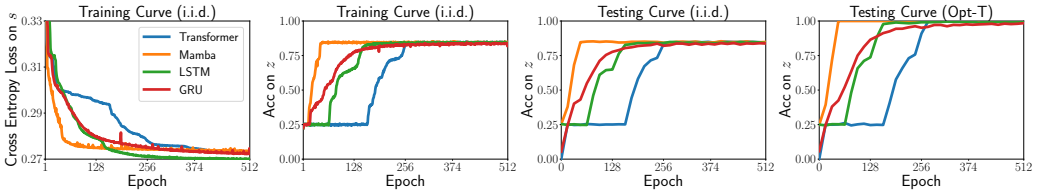


Figure 15: **Varied models on ID hypothesis class generalization.**

C.3. Effect of Training Class Count

We share more training and testing curves in Fig. 18 and Fig. 19 to provide additional results to Fig. 8(a) and Fig. 8(b).

[5]We use variable z to represent the hypothesis index.

[6]A set of L hypothesis index tokens are created serve as the pool from which the hypothesis indexes are randomly sampled without replacement.

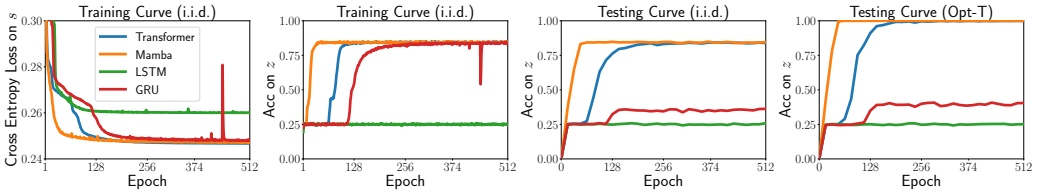
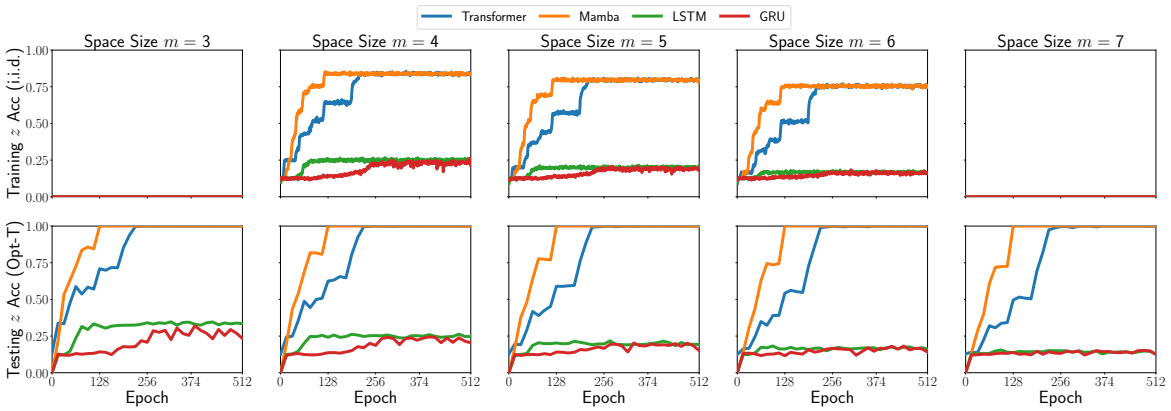
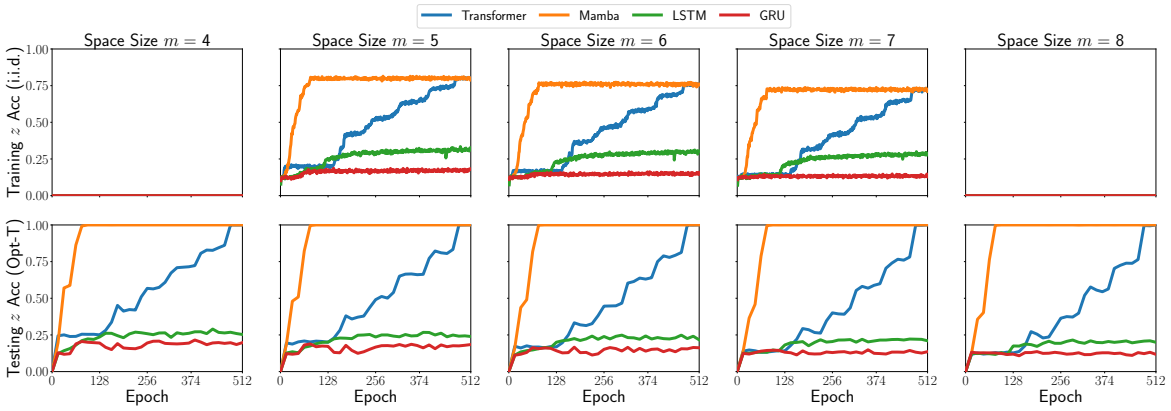


Figure 16: **Varied models on OOD hypothesis class generalization.**



(a) testing curves of ID hypothesis class size generalization.



(b) testing curves of OOD hypothesis class size generalization.

Figure 17: **Varied models on ID and OOD hypothesis class size generalizations.** For both settings of generalization, Transformer and Mamba perform well while LSTM and GRU fail. Mamba needs fewer epoch to converge. LSTM and GRU fail to fit training data.

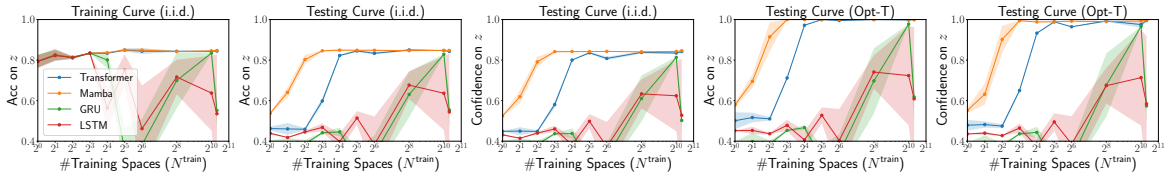


Figure 18: **Effect of training-space count on space generalization.**

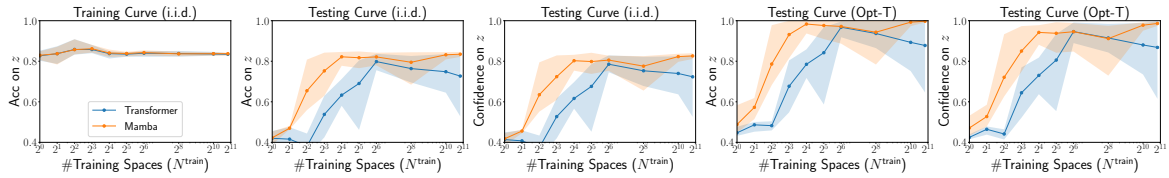


Figure 19: **Effect of training-space count on hypothesis generalization.**

D. Experiment Setup

D.1. Hyperparameters

We list the hyperparameter searching spaces used for Transformer, LSTM, GRU, and Mamba. The best hyperparameter is searched using the setting of ID hypothesis class generalization with $\|\mathcal{X}\| = 4$, and then used for all other settings.

Table 1: **Hyperparamer searching spaces for different model architectures.** The optimal hyperparameters are bolded if multiple possibilities are provided.

Model Architecture	#layers	#hidden dimensions	#learning rate	#weight decay	#batch size
Transformer	2,**8**	128	0.00001, **0.00002**, 0.00005, 0.00010	0.0005	16
Mamba	**2**,8	128	0.00020, **0.00050**, 0.00100, 0.00200	0.0005	16
GRU	**2**,8	128	0.00020, 0.00050, **0.00100**, 0.00200	0.0005	16
LSTM	**2**,8	128	0.00020, 0.00050, **0.00100**, 0.00200	0.0005	16

D.2. Setup of Generalization

We list the experimental setup in the following Table 2. When conducting experiments with ICL, we modified the experimental setup following Table 3.

Table 2: **Experimental setup of different generalizations.**

Generalization Setup	ID Hypothesis	OOD Hypothesis	ID Hypothesis & Size	OOD Hypothesis & Size
size of input space ($\|\mathcal{X}\|$)	4	5	5	6
size of label space ($\|\mathcal{Y}\|$)	2	2	2	2
size of training hypothesis classes ($\|\mathcal{H}^{\text{train}}\|$)	4	4	4,5,6	5,6,7
size of testing hypothesis classes ($\|\mathcal{H}^{\text{test}}\|$)	4	4	3,4,5,6,7	4,5,6,7,8
size of hypothesis prefix (L)	4	4	8	8
size of context query (K)	4	5	5	6
#all hypotheses (M)	$2^4 = 16$	$2^5 = 32$	$2^5 = 32$	$2^6 = 64$
#training hypotheses (M^{train})	/	16	/	32
#testing hypotheses (M^{test})	/	16	/	32
#total hypothesis classes	C(16,4)	/	C(32,3∼7)	/
#total training hypothesis classes	/	C(16,4)	/	C(32,5∼7)
#total testing hypothesis classes	/	C(16,4)	/	C(32,4∼8)
#sampled training hypothesis classes (N^{train})	1274	1820	3000 for all	3000 for all
#sampled testing hypothesis classes (N^{test})	546	1820	1500 for all	1500 for all

Table 3: **Experimental setup of ICL.**

Generalization Setup	ID Hypothesis	OOD Hypothesis
size of input space ($\|\mathcal{X}\|$)	4	6
size of label space ($\|\mathcal{Y}\|$)	2	2
size of training hypothesis classes ($\|\mathcal{H}^{\text{train}}\|$)	4	4
size of testing hypothesis classes (($\|\mathcal{H}^{\text{test}}\|$)	4	4
size of hypothesis prefix (L)	4	4
size of context query (K)	**12**	**12**
#all hypotheses (M)	$2^4 = 16$	$\mathbf{2^6 = 64}$
#training hypotheses (M^{train})	/	**4,8,16,32**
#testing hypotheses (M^{test})	/	**12**
#total hypothesis classes	C(16,4)	/
#total training hypothesis classes	/	C(·,4)
#total testing hypothesis classes	/	C(16,4)