To What Extent Can In-Context Learning Solve Unseen Tasks?

Ryoma Shinto, Masashi Takeshita, Rzepka Rafal, Toshihiko Itoh
Hokkaido University
{shinto.ryoma, takeshita.masashi.68}@gmail.com
{rzepka, t-itoh}@ist.hokudai.ac.jp

Abstract

While Large Language Models (LLMs) are known for their In-Context Learning (ICL) capabilities, there is no consensus on the underlying mechanisms. A key point of debate is whether ICL allows models to adapt to unseen tasks without parameter updates—that is, whether they can extrapolate. In this study, we address this question by constructing an arithmetic dataset based on the bivariate linear function $z = ax + by$ to train a model and quantitatively evaluate its interpolation and extrapolation abilities through ICL. Our results show that while extrapolation was not achieved within our experimental design, tasks that were partially learned could be solved. We also found that the model acquires internal representations that can distinguish unseen tasks, and that greater task diversity in the training dataset improves ICL capabilities.

1 Introduction

Large Language Models (LLMs) are known to be capable of In-Context Learning (ICL) (Brown et al., 2020; Dong et al., 2024). ICL is a method that improves inference performance by presenting examples of a task within a prompt, without updating any parameters. This approach allows for efficient and flexible applications, as it does not require the preparation of training data or additional computational resources (Mosbach et al., 2023; Yin et al., 2024).

Regarding the mechanism of ICL, three main hypotheses have been proposed, as shown in Figure 1. One hypothesis is Task Selection, which posits that the model recognizes the characteristics of a task from in-context examples and then selects and applies a pre-trained task (Xie et al., 2022; Wies et al., 2023). Another is Task Composition, which suggests that the model can combine multiple pre-trained tasks to perform inference (Li et al., 2024). Furthermore, there is the Meta-learning hypothesis, which proposes that ICL enables the model to learn how to learn, adapting to unseen tasks based on in-context examples (Von Oswald et al., 2023; Akyurek et al., 2023). However, these hypotheses are not always consistent with subsequent experimental results (Kossen et al., 2024; Li et al., 2024), and a unified understanding of the ICL mechanism has not yet been achieved.

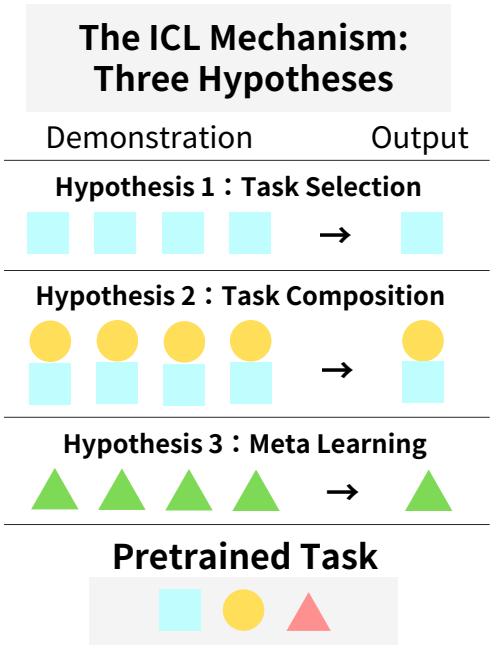

Figure 1: *A conceptual diagram of the three main hypotheses for the In-Context Learning (ICL) mechanism.* **Hypothesis (1) Task Selection**: The model selects and utilizes a single pre-trained task (e.g., blue squares) that matches the demonstration. **Hypothesis (2) Task Composition**: The model combines multiple pre-trained tasks to address the new task presented in the demonstration. **Hypothesis (3) Meta Learning**: The model learns and utilizes an unseen task (e.g., a green triangle), which does not exist in the pre-training data, on the fly from the context.

A key point of contention is whether ICL can

Proceedings of the IJCNLP-AACL 2025 Student Research Workshop, pages 277–288

be used to adapt to unseen tasks, as this could provide compelling evidence or counterexamples for these hypotheses (Garg et al., 2024; He et al., 2024). For instance, if a model can answer a task that it has not been pre-trained on simply by being shown examples in a prompt, it would imply that the model learned the task from the context alone without parameter updates. This would be evidence supporting the meta-learning hypothesis over the task selection and task composition hypotheses. However, when training on large-scale language data, it is not practical to clearly define the boundary between learned tasks (interpolation) and completely new tasks (extrapolation), making it difficult to rigorously evaluate the extrapolation capabilities of ICL.

Therefore, this research aims to provide important insights into the ICL mechanism by analyzing its extrapolation capabilities using arithmetic tasks. The advantage of arithmetic tasks is that, unlike language data, they allow for a clear separation between the domains of interpolation and extrapolation by controlling the number of digits and the range of variables.

In our experiments, we construct a total of 15 different datasets and analyze the extrapolation ability of ICL by evaluating the test data accuracy and internal representation vectors of models trained on each. The results revealed the following findings: (i) ICL can solve new tasks by combining previously learned tasks. (ii) Although the model cannot solve completely unseen tasks, it encodes internal representations that can identify them. (iii) The greater the diversity of tasks in the training dataset, the higher the ICL capability. The findings from this study are expected to make a significant contribution to the understanding of ICL mechanisms, for which a consensus has yet to be established.

2 Related Work

2.1 In-Context Learning

In-Context Learning (ICL) is one of the groundbreaking capabilities of Large Language Models (LLMs), enabling them to perform inference based on a few examples (Demonstrations) provided within a prompt, without any parameter updates. This ability, widely publicized by Brown et al. (2020) (Brown et al., 2020), allows a model to grasp the rules of a task on the fly from the examples in the prompt and adapt to new queries (Brown et al., 2020; Dong et al., 2024).

The emergence of ICL brought about a major paradigm shift in adapting models to specific tasks. Previously, the mainstream approach for adapting a model to a new task was Fine-Tuning (FT), which involved preparing high-quality annotated data to retrain all or part of the model's parameters (Devlin et al., 2019; Howard and Ruder, 2018). While this process had the advantage of requiring less computational cost and data compared to pre-training (Houlsby et al., 2019; Ben Zaken et al., 2022; Hu et al., 2022), it still necessitated parameter updates to adapt to new tasks.

In contrast, ICL uses natural language prompts as its interface and requires no additional training data or weight updates, enabling extremely low-cost and rapid task adaptation (Mosbach et al., 2023; Yin et al., 2024). Furthermore, whereas FT produces a task-specific model, ICL maintains a single, general-purpose model and demonstrates high versatility by flexibly handling a wide variety of tasks simply by rewriting the prompt (Brown et al., 2020; Wei et al., 2022; Ferber et al., 2024). Due to this efficiency and flexibility, ICL is considered a "new paradigm in natural language processing" and is recognized as a key characteristic of LLMs (Dong et al., 2024; Wies et al., 2023; Gu and Dao, 2024).

2.2 Hypotheses on the Mechanism of In-Context Learning

There is not yet a consensus on the mechanism by which ICL functions, and multiple hypotheses have been proposed. As mentioned in the introduction of this paper, these hypotheses can be broadly categorized into the following three.

The first is the "Task Selection" hypothesis, which posits that the model recognizes the characteristics of a task from in-context examples and then selects and applies an appropriate task from a set of tasks acquired during pre-training (Xie et al., 2022; Wies et al., 2023). This hypothesis formulates ICL as Bayesian inference, where the model infers a task conditioned on the input demonstrations.

The second is the "Task Composition" hypothesis, which suggests that the model performs inference by combining multiple learned tasks and knowledge (Li et al., 2024). This hypothesis explains that ICL can handle tasks that do not directly exist in the training data but can be derived by combining learned tasks.

The third is the "Meta-learning" hypothesis, which views ICL as a process of learning the solution to the task itself (Von Oswald et al., 2023; Akyurek et al., 2023). This perspective claims that a dynamic similar to gradient descent is driven internally during ICL, allowing the model to adapt to unknown tasks from contextual information. Therefore, it makes a fundamentally different claim from the "Task Selection" and "Task Composition" hypotheses in that it posits the model can handle tasks it has not pre-trained on, without parameter updates, based on contextual information.

In this study, based on these hypotheses, we design three corresponding types of experiments. Through quantitative analysis of their results, we aim to provide new insights into the mechanism of ICL.

3 Experimental Design

3.1 Dataset Construction

3.1.1 Data Representation Format

The dataset used in this study was constructed based on the bivariate linear function $z = ax + by$ to quantitatively evaluate the model's extrapolation capability in ICL (see Figure 2). The variables x and y are integers ranging from one to four digits, and the coefficients a and b are integers where $a, b \in \{0, 1, \ldots, 9\}$. The model is given a prompt consisting of k computational examples (Demonstrations) and one question (Query). A k-shot prompt is input as a concatenated sequence of k demonstrations, $D = \{(x_i, y_i, z_i)\}_{i=1}^{k}$, and a final query, $q = (x_{k+1}, y_{k+1})$. The coefficients (a, b) are common to all examples within a prompt, and x, y are randomly generated. The model is required to infer the common coefficients (a, b) from the given k examples and predict the corresponding z_{k+1} for the query.

Example of a 2-shot case with $a = 2, b = 1$

```
Demonstration 1: (132, 5532, 5796)
Demonstration 2: (355, 22, 732)
Query: (4412, 3356)
Target Output: 12180
```

As shown above, the coefficients a, b are not explicitly stated in the prompt. Therefore, the value of z cannot be uniquely determined from the query's x, y values alone. The model must use ICL to identify the common coefficients (a, b) from the k demonstrations to infer z. This design ensures that a model properly trained on this dataset is performing ICL during inference.

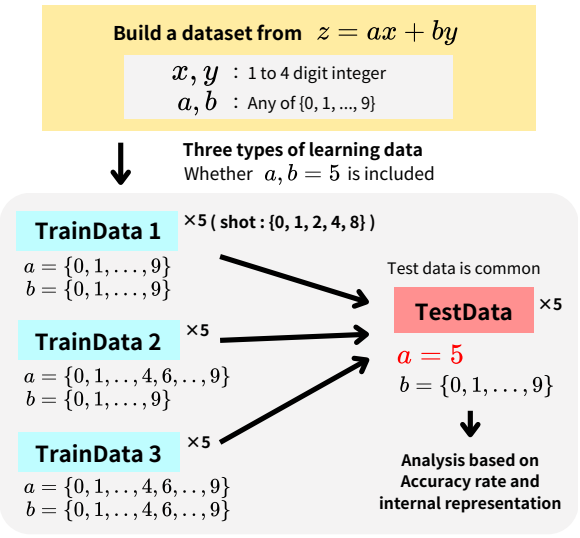

Figure 2: *Dataset construction and evaluation flow for analyzing the extrapolation capability of ICL.* The dataset is constructed based on $z = ax + by$. The training data is classified into three types based on whether they include coefficients $a, b = 5$. Since the test data involves tasks where $a = 5$, interpolation and extrapolation are defined by the range of a, b in the training data. Training datasets 1-3 are further subdivided by the number of shots into five types, for a total of 15 training datasets.

Furthermore, all digits in the dataset are converted into unique symbols. This allows us to block the influence of the model's pre-existing arithmetic knowledge and purely analyze its reasoning ability through ICL[1].

Symbolic Representation of the Dataset

```
Demonstration 1: (%?{,«?{,<@$>)
Demonstration 2: (?«,{{,@?{)
Query: (!!%{,??<>)
Target Output: %{%;^
```

Hereafter, we define a pair of coefficients (a, b) as a single "task." Since the coefficients a and b can each take 10 different values, the task space $\mathcal{T}$ in this study consists of 100 tasks, defined as follows:

$$\mathcal{T} = \{(a, b) \mid a, b \in \{0, 1, \ldots, 9\}\} \quad (1)$$

3.1.2 Dataset Composition

The training data consists of a total of 200,000 examples (train:validation = 8:2), and the test data

[1]See Appendix A.1 for the digit-to-symbol conversion mapping.

Table 1: Ranges of coefficients a, b in each dataset

Dataset	Range of a	Range of b
Training Data 1	$a \in \{0, .., 5, .., 9\}$	$b \in \{0, .., 5, .., 9\}$
Training Data 2	$a \in \{0, .., 4, 6, .., 9\}$	$b \in \{0, .., 5, .., 9\}$
Training Data 3	$a \in \{0, .., 4, 6, .., 9\}$	$b \in \{0, .., 4, 6, .., 9\}$
Test Data	$a = 5$	$b \in \{0, .., 5, .., 9\}$

consists of 1,000 examples.

In this study, to separately evaluate the interpolation and extrapolation capabilities of ICL, we establish three experimental settings based on the relationship between the set of tasks in the training data, $\mathcal{T}_{train}$, and the set of tasks in the test data, $\mathcal{T}_{test}$. Specifically, we define the datasets based on whether the coefficient '5' is included, as shown in Table 1.

The set of tasks used in the test data, $\mathcal{T}_{test}$, is fixed to tasks with the coefficient $a = 5$.

$$\mathcal{T}_{test} = \{(a, b) \in \mathcal{T} \mid a = 5\} \tag{2}$$

In contrast, the three types of training datasets each have the following task sets.

Setting 1: Interpolation The task set used in training data 1, $\mathcal{T}_{train1}$, is identical to the entire task space $\mathcal{T}$.

$$\mathcal{T}_{train1} = \mathcal{T} \tag{3}$$

In this setting, the condition $\mathcal{T}_{test} \subset \mathcal{T}_{train1}$ holds, meaning all tasks evaluated in the test set have been seen during training. Therefore, this setting evaluates the model's pure interpolation ability—whether it can correctly select and execute a learned task from the context.

Setting 2: Partial Interpolation The task set used in training data 2, $\mathcal{T}_{train2}$, consists only of tasks where the coefficient a does not include '5'.

$$\mathcal{T}_{train2} = \{(a, b) \in \mathcal{T} \mid a \neq 5\} \tag{4}$$

In this case, since the coefficient a in the test data is fixed to '5', $\mathcal{T}_{test} \cap \mathcal{T}_{train2} = \emptyset$, meaning the training data contains no tasks that perfectly match the test tasks. However, the task set $\mathcal{T}_{train2}$ does include the coefficient '5' for b. Therefore, this setting tests whether the model can solve tasks with coefficient $a = 5$ by leveraging its knowledge of tasks with coefficient $b = 5$ from training data 2.

Setting 3: Extrapolation The task set used in training data 3, $\mathcal{T}_{train3}$, consists only of tasks where neither coefficient a nor b includes '5'.

$$\mathcal{T}_{train3} = \{(a, b) \in \mathcal{T} \mid a \neq 5 \wedge b \neq 5\} \tag{5}$$

This is the most rigorous setting. The model is not trained on tasks with $a = 5$, nor even on tasks with $b = 5$. This means the model will observe the token for '5' for the first time in the test set's Demonstrations. This setting questions the model's true extrapolation ability—whether it can infer rules for a completely unseen domain from the context alone.

In addition to these three settings, each training dataset is further subdivided into five variations based on the number of examples in the prompt (number of shots): 0, 1, 2, 4, and 8-shot. This results in a total of 15 distinct training datasets for training and evaluation.

Note that in this study, we clearly distinguish between extrapolation and generalization. Extrapolation refers to the ability to handle unseen tasks $(a, b) \notin \mathcal{T}_{train}$, whereas generalization refers to the ability to correctly infer z from unseen inputs (x, y) within the scope of learned tasks $(a, b) \in \mathcal{T}_{train}$.

3.2 Model and Evaluation

For this research, we fine-tuned a pre-trained ByT5 base model (Xue et al., 2022). The Encoder-Decoder architecture adopted by ByT5 base has a clear separation between the roles of encoding the input sequence and decoding the output sequence. This makes it well-suited for analyzing the final hidden state of the encoder to understand how the model extracts task regularities from the context D and constructs internal representations. Furthermore, ByT5 tokenizes input symbol strings on a character-by-character basis, ensuring that multi-digit numbers are tokenized uniquely without being split. This guarantees a strict distinction between interpolation and extrapolation, regardless of the tokenizer.

The model is evaluated using the checkpoint that achieved the minimum loss on the validation dataset for each training setting. The primary evaluation metric is the accuracy on the test dataset. To visualize the acquisition process of the ICL capability during training, we recorded the accuracy trends for 1,000 samples each from the validation and test datasets every 1,000 steps during training[2].

[2]For experimental details such as training hyperparameters, see Appendix A.2

3.3 Probing Analysis

In this study, we anticipate that the model may sometimes be unable to solve unseen tasks. However, even in such cases, it is possible that the model internally captures the properties of the unseen task. To test this hypothesis, we conduct a probing experiment to verify whether the task (a, b) from the input prompt can be identified at the internal representation level. Probing is an analysis method that involves extracting a model's internal states (such as the activation vectors of hidden layers) and using a simple, external classifier (a probe) to test whether specific information (in this case, the task identifier) can be predicted from these vectors.

First, we create a new dataset for probing with 100,000 examples (train:validation = 9:1). The data format is the same as defined in Section 3.1. Each sample is assigned a unique integer label l based on the task (a, b) it belongs to, according to Equation 6.

$$l = 10a + b \quad (l \in \{0, 1, \ldots, 99\}) \tag{6}$$

This allows us to treat the 100 different tasks as a 100-class classification problem.

Next, using the encoder E of the fine-tuned ByT5 model, we extract an internal representation vector from each input prompt $P = (D, q)$. Specifically, we use the hidden state vector $h_{EOS} \in R^{1536}$ corresponding to the EOS (End-of-Sequence) token of the final encoder layer, which is considered to aggregate the contextual information of the entire prompt.

$$h_{EOS} = E(P) \tag{7}$$

Then, we train a linear classifier (a multi-class logistic regression model) f_{probe} to predict the task label l from this internal representation vector h_{EOS}.

$$\hat{l} = f_{probe}(h_{EOS}) \tag{8}$$

The classification accuracy in this probing task serves as an indicator of how well the model can internally distinguish the task (a, b). High accuracy would provide strong evidence that the task-identifying information is encoded in a linearly separable manner within the model's internal representations, suggesting that the model identifies tasks through ICL.

3.4 The Effect of Data Diversity on ICL

The three types of training datasets defined in Table 1 differ not only in their interpolation/extrapolation conditions but also in the total number of tasks, depending on whether they include $a, b = 5$. Specifically, the sizes of each training dataset's task set are as follows:

- Training Data 1: $|\mathcal{T}_{train1}| = 100$
- Training Data 2: $|\mathcal{T}_{train2}| = 90$
- Training Data 3: $|\mathcal{T}_{train3}| = 81$

This difference in the number of tasks could affect the acquisition of ICL capabilities and potentially confound the main analysis results. Therefore, we conduct an auxiliary experiment to independently evaluate the impact of task diversity within the training data on ICL performance.

3.4.1 Auxiliary Experiment Design

To evaluate the effect of task diversity, we created four new datasets with varying numbers of tasks by adjusting the range of coefficients a, b, based on Training Data 2. The task set for each dataset is defined as follows:

- **Training Data 2-1**: 30 tasks
  $\mathcal{T}_{train2-1} = \{(a, b) \mid a \in \{0, .., 4\}, b \in \{0, .., 5\}\}$
- **Training Data 2-2**: 42 tasks
  $\mathcal{T}_{train2-2} = \{(a, b) \mid a \in \{0, .., 4, 6\}, b \in \{0, .., 5, 6\}\}$
- **Training Data 2-3**: 56 tasks
  $\mathcal{T}_{train2-3} = \{(a, b) \mid a \in \{0, .., 4, 6, 7\}, b \in \{0, .., 5, 6, 7\}\}$
- **Training Data 2-4**: 90 tasks
  $\mathcal{T}_{train2-4} = \{(a, b) \mid a \in \{0, .., 4, 6, .., 9\}, b \in \{0, .., 5, .., 9\}\}$

Note that Training Data 2-4 is identical to Training Data 2 ($\mathcal{T}_{train2}$) from the main experiment. We train models on these datasets under the exact same settings as the main experiment and calculate the accuracy on the same test data (see Table 1) to compare and analyze the effect of task diversity on ICL capability. The number of demonstrations provided to the model is standardized to four (4-shot) for this verification.

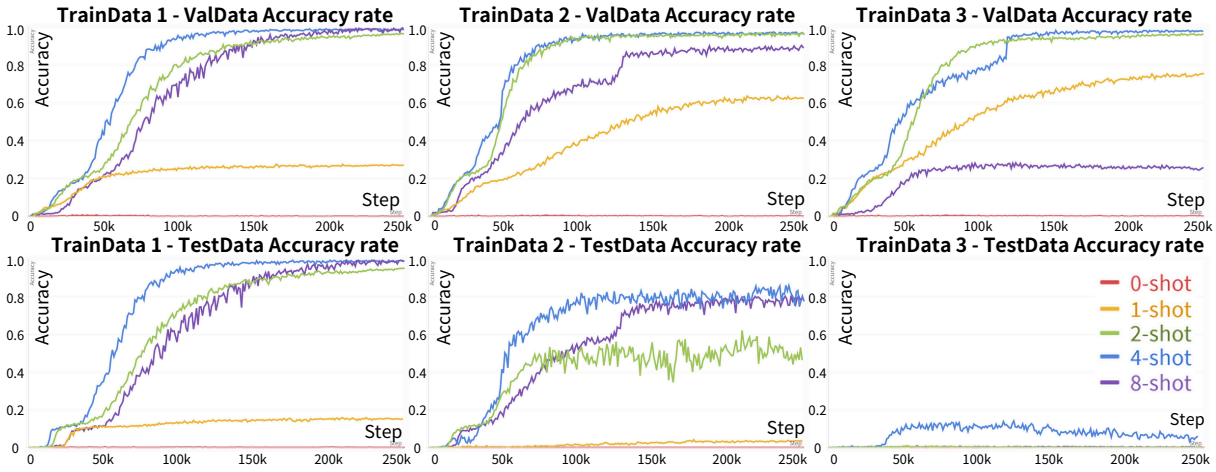

Figure 3: *Transition of accuracy on validation data (top) and test data (bottom) for models trained on each dataset.* Of particular note is the test data accuracy for TrainData 2 (bottom center), which only partially covers the test data task because it contains $b = 5$ but not $a = 5$. Nevertheless, the model achieved approximately 0.5 accuracy in the 2-shot setting and around 0.8 accuracy in both the 4-shot and 8-shot settings. In contrast, the test data accuracy for TrainData 3, which excludes $a, b = 5$ entirely, was nearly zero across all shot settings (bottom right).

4 Experimental Results

4.1 Accuracy Results

Figure 3 shows the accuracy trends for the validation data (top row) and test data (bottom row) for models trained on each dataset. It is important to note that, as explained in Section 3.1.2, the scope of the task sets for each of the validation datasets (three types) and the test dataset (one type) was intentionally manipulated (see Figure 2). Therefore, by confirming that the validation accuracy is nearly 1, we can ensure that training has completed successfully. This allows us to attribute the success or failure on the test data specifically to the model's in-context interpolation and extrapolation capabilities. Additionally, Table 2 shows the test data accuracy at the checkpoint with the lowest validation loss, which serves as the primary indicator for task success or failure.

Validation Data Results A common trend in the validation accuracy plots (Figure 3, top row) is that while the accuracy for 0-shot and 1-shot models struggles to improve, the accuracy for 2-shot and 4-shot models approaches 1. For the 8-shot case, accuracy approached 1 for models trained on Training Data 1 and 2 (left and center columns), but it did not improve for the model trained on Training Data 3 (right column).[3] These results indicate that training was completed correctly for all datasets only in the 2- and 4-shot cases. Therefore, the analysis of ICL's interpolation and extrapolation capabilities will be based on the results of the 2- and 4-shot models.

Table 2: Test accuracy for each models

Dataset	0-shot	1-shot	2-shot	4-shot	8-shot
TrainData1	0.002	0.116	0.936	0.979	0.971
TrainData2	0.000	0.015	0.473	0.825	0.805
TrainData3	0.000	0.000	0.000	0.066	0.008

Test Data Results - Training Data 1 The graph in the lower-left panel of Figure 3 shows that for the 2-, 4-, and 8-shot cases, the test accuracy converges to 1 during training. In contrast, the accuracy remained at 0.002 for the 0-shot case and 0.116 for the 1-shot case (see Table 2).

Test Data Results - Training Data 2 The graph in the lower-middle panel of Figure 3 shows that the accuracy converges to around 0.5 for the 2-shot case and around 0.8 for the 4- and 8-shot cases. In contrast, the accuracy for the 0- and 1-shot cases remained near zero (see Table 2).

Test Data Results - Training Data 3 As shown in Table 2, the accuracy was 0 for almost all shot counts. This indicates that, within our experimental setup, the model could not solve completely unseen tasks.

[3]The reason for this is discussed in Section 5.2 from the perspective of dataset diversity.

Table 3: Probing accuracy of each models

Dataset	0-shot	1-shot	2-shot	4-shot	8-shot
TrainData1	0.010	0.493	0.963	0.996	0.998
TrainData2	0.009	0.754	0.951	0.993	0.997
TrainData3	0.012	0.798	0.929	0.993	0.976

4.2 Probing Experiment Results

Table 3 shows the average accuracy for each model in the probing experiment, which classifies the task (a,b) from the internal representation of the input sequence. For the 2-, 4-, and 8-shot settings, the models trained on any of the training datasets achieved an accuracy of over 0.9, indicating that the classifier could properly linearly separate the tasks based on the internal representations of the input sequence. Notably, even for the model trained on Training Data 3, which had an accuracy of almost 0 in Table 2, the probing experiment recorded a high accuracy. On the other hand, the accuracy for the 0-shot case was nearly zero, and while the 1-shot case showed some variation depending on the training data, it did not reach a sufficient level of accuracy. The detailed results of the probing experiment for each model are visualized as confusion matrices in Appendix A.3.

4.3 Results of the Analysis of Dataset Diversity's Impact

Figure 4 shows the accuracy trends for the validation data (top) and test data (bottom) for models trained on the four types of datasets described in Section 3.4.1. The number of demonstrations was standardized to 4-shot. The accuracy on the validation data (top row) can be seen converging to 1 for all training datasets, indicating that training was completed successfully. On the other hand, the accuracy on the test data (bottom row) is observed to converge to higher levels as the diversity of the training data increases. Table 4 shows the test data accuracy at the checkpoint with the lowest validation data loss, and this table also demonstrates that increasing task diversity leads to significant changes in accuracy. Notably, for Training Data 2-1, although the validation accuracy converged to 1, the test accuracy remained at only 0.003, indicating that when the task diversity in the training data is low, the ICL capability cannot be properly generalized.

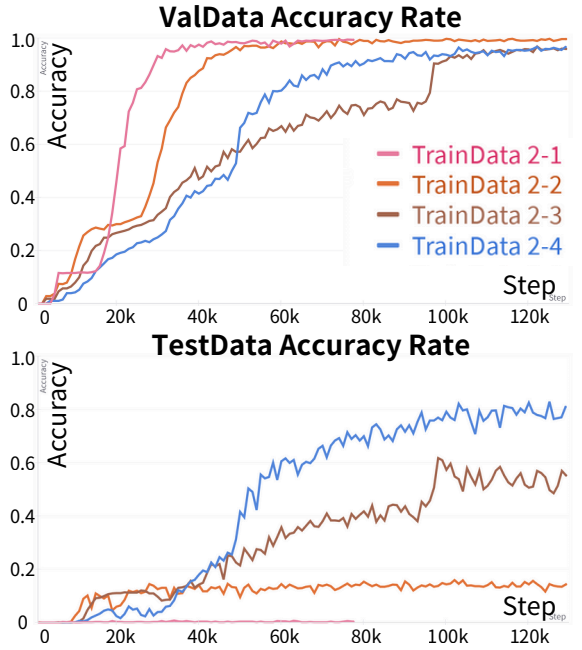

Figure 4: *Accuracy trends on the validation data (top row) and test data (bottom row) for each training dataset.*

Table 4: Test data accuracy for each training dataset

Dataset	Accuracy
Training Data 2-1	0.003
Training Data 2-2	0.158
Training Data 2-3	0.569
Training Data 2-4	0.825

5 Discussion

5.1 Extrapolation Capability of ICL from the Perspective of Accuracy and Probing Results

First, Figure 3 shows that for the 0- and 1-shot cases, the validation accuracy did not converge to 1 for any training dataset, and the test accuracy was also nearly 0. This is likely because the task was created from a bivariate function, which requires at least two demonstrations to identify the specific task (a, b).

Next, regarding Training Data 1, the models achieved approximately 100% accuracy on both the validation and test data for the 2-, 4-, and 8-shot cases (see Table 2). Since Training Data 1 includes the task scope of the test data (interpolation), this result suggests that the model can recognize the presented task via ICL and appropriately select and apply a task from its learned repertoire.

Subsequently, for Training Data 2, despite not

being trained on tasks with $a = 5$—i.e., tasks identical to the test data—the model achieved accuracies of 0.473 for 2-shot, 0.825 for 4-shot, and 0.805 for 8-shot on the test data (see Table 2). This suggests that ICL not only selects learned tasks but can also solve partially unseen tasks by composing them. Specifically, it is thought that the model solved the test tasks by combining the knowledge gained from learning tasks with $b = 5$ included in Training Data 2—i.e., tasks of the form $(a, b) = (k, 5)$ (where $k \in \{0, .., 4, 6, .., 9\}$)—with the demonstrations for the test tasks $(a, b) = (5, k)$ (where $k \in \{0, 1, .., 9\}$).

Finally, the test accuracy for the model trained on Training Data 3 was nearly 0 for all shot counts, providing no evidence that ICL enables extrapolation within the scope of this experiment. However, the results of the probing experiment (see Table 3) show that in the 2-shot and higher settings, the model trained on Training Data 3, similar to the models trained on other data, could linearly separate tasks from the input sequence. This suggests that in ICL, the model encodes internal representations from the input sequence in a way that enables it to separate each task, thereby distinguishing unseen tasks from learned ones. Therefore, while ICL allows the model to acquire internal representations that can identify completely new tasks, a failure to map these representations to the correct output—that is, a failure in the decoder's dynamics—is likely the cause of the extrapolation failure, warranting further investigation.

5.2 The Effect of Dataset Diversity on ICL Capability

As seen in Figure 4 and Table 4, while the validation accuracy (top row) converges to 1 for all training datasets, the test accuracy improves in line with the task diversity of the dataset. These results suggest that the diversity of tasks in the training data is crucial for acquiring ICL capability. This is likely because high task diversity enables the model to learn a general-purpose solution applicable to all tasks, rather than learning a specific solution for each individual task.

Based on this consideration, we can speculate on why only the 8-shot model for Training Data 3 failed to reach an accuracy of 1 on the validation data (top-right graph in Figure 3), unlike the models for Training Data 1 and 2. Specifically, since Training Data 3 has fewer total tasks compared to Training Data 1 and 2 (see Section 3.4), it is conceivable that a general-purpose ICL capability was not sufficiently acquired. It is important to note that this argument applies only to the 8-shot case, as the validation accuracies for the 2- and 4-shot models did converge to 1. Since it has been shown that ICL performance improves with more demonstrations (Brown et al., 2020; Dong et al., 2024), a significant challenge for further verifying extrapolation capability is to test with 8 or more shots. Therefore, to discuss the extrapolation capability of ICL in settings with 8 or more shots, it is necessary to use datasets with even greater task diversity, such as by expanding the range of coefficients a, b or creating data from a trivariate linear function.

6 Conclusion

In this study, we analyzed the extrapolation capability of LLMs through ICL using an arithmetic task based on a bivariate linear function. Based on the three main hypotheses of the ICL mechanism, we designed an experiment that enables the analysis of ICL's extrapolation capabilities—a difficult feat with natural language—by manipulating the range of the task (a, b) in our dataset design. Our analysis, based on test data accuracy, probing of internal representations, and auxiliary experiments considering task diversity, yielded the following insights: (i) Through ICL, partially learned tasks can be solved by composing learned tasks. (ii) The model acquires internal representations that can distinguish unseen tasks. (iii) The greater the task diversity in the training dataset, the higher the ICL capability.

For future work, we believe that by examining the decoder's dynamics during extrapolation in detail, we can provide more useful experimental insights into why the model fails to produce the correct answer despite being able to identify the extrapolation task. Furthermore, analysis using datasets with even greater task diversity will be necessary, for instance, by expanding the range of tasks a, b or designing tasks with trivariate linear functions. Through these efforts, this research is expected to make a significant contribution to the understanding of the ICL mechanism, for which a consensus has yet to be established.

Limitations

While this study provides valuable insights into the extrapolation capabilities of in-context learning (ICL) through controlled arithmetic tasks, several limitations remain.

First, the experimental setting focuses exclusively on arithmetic tasks, which allow for clear definitions of interpolation and extrapolation. However, this abstraction may not directly reflect the nature of linguistic tasks in real-world language modeling. Therefore, the results obtained here may not generalize to natural language data, where task boundaries and generalization behavior are less well-defined.

Second, we used ByT5, an encoder-decoder architecture, as the base model for analysis. Although this design choice enables precise control over input tokenization and allows us to analyze the encoder's final hidden state to investigate how the model learns task regularities from demonstrations, it limits the direct applicability of our findings to contemporary decoder-only large language models (LLMs), such as GPT-4, which are more widely used in practical scenarios.

To bridge these gaps, future work should explore whether similar patterns of extrapolation and task identification emerge in decoder-only models and under linguistically grounded tasks.

Ethical Considerations

This foundational study uses a synthetic arithmetic dataset, which contains no personally identifiable information or societal biases. Due to the abstract nature of the research and the artificial data, we do not foresee any direct societal risks or potential for misuse of our findings.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR20D2.

References

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

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

Dyke Ferber, Georg Wölflein, Isabella C. Wiest, Marta Ligero, Srividhya Sainath, Narmin Ghaffari Laleh, Omar S. M. El Nahhas, Gustav Müller-Franzes, Dirk Jäger, Daniel Truhn, and Jakob Nikolas Kather. 2024. In-context learning enables multimodal large language models to classify cancer pathology images. *Nature Communications*, 15.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. 2024. What can transformers learn in-context? a case study of simple function classes. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. 2024. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea

Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jannik Kossen, Yarin Gal, and Tom Rainforth. 2024. In-context learning learns label relationships but is not conventional learning. In *The Twelfth International Conference on Learning Representations*.

Jiaoda Li, Yifan Hou, Mrinmaya Sachan, and Ryan Cotterell. 2024. What do language models learn in context? the structured task hypothesis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12365–12379, Bangkok, Thailand. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12284–12314, Toronto, Canada. Association for Computational Linguistics.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Noam Wies, Yoav Levine, and Amnon Shashua. 2023. The learnability of in-context learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Qingyu Yin, Xuzheng He, Chak Tou Leong, Fan Wang, Yanzhao Yan, Xiaoyu Shen, and Qiang Zhang. 2024. Deeper insights without updates: The power of in-context learning over fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4138–4151, Miami, Florida, USA. Association for Computational Linguistics.

A Appendix

A.1 Digit-to-symbol conversion mapping

The digits in the dataset are mapped to symbols according to Table 5. Each of these symbols is treated as a single token by the ByT5 tokenizer, which ensures that the distinction between the interpolation and extrapolation domains is preserved.

Table 5: Digit-to-symbol conversion mapping.

Digit	Symbol
0	^
1	%
2	{
3	?
4	!
5	<
6	>
7	@
8	;
9	$

A.2 Training Settings

A.2.1 ByT5 Hyperparameter Settings

- Model size : 580 million parameters
- Optimizer: AdamW (Loshchilov and Hutter, 2019)
- Learning rate: 0.0001
- Batch size: 64
- Epochs: 100

A.2.2 Probing Experiment Settings

- Classifier: Multiclass logistic regression (scikit-learn)
- multi_class: 'multinomial'
- Regularization: ℓ_2 (with $C = 1.0$)
- max_iter: 1000

A.3 Probing Results

Figure 5 presents each model ' s probing results as 100 $\times$ 100 confusion matrices for all shot settings. The vertical axis denotes the true task labels (a, b) (100 classes), and the horizontal axis shows the predicted labels (a, b) assigned by the multiclass logistic regression based on the model ' s internal representations. Color intensity reflects the frequency of each prediction. As shown in Figure 5, regardless of the type of training data, the 2-, 4-, and 8-shot matrices exhibit strong concentration along the diagonal, indicating—as also reported in Table 3—that models accurately identify tasks from inputs under these conditions. In contrast, the 0-shot matrix shows no discernible pattern, and the 1-shot matrix displays partial misclassifications.

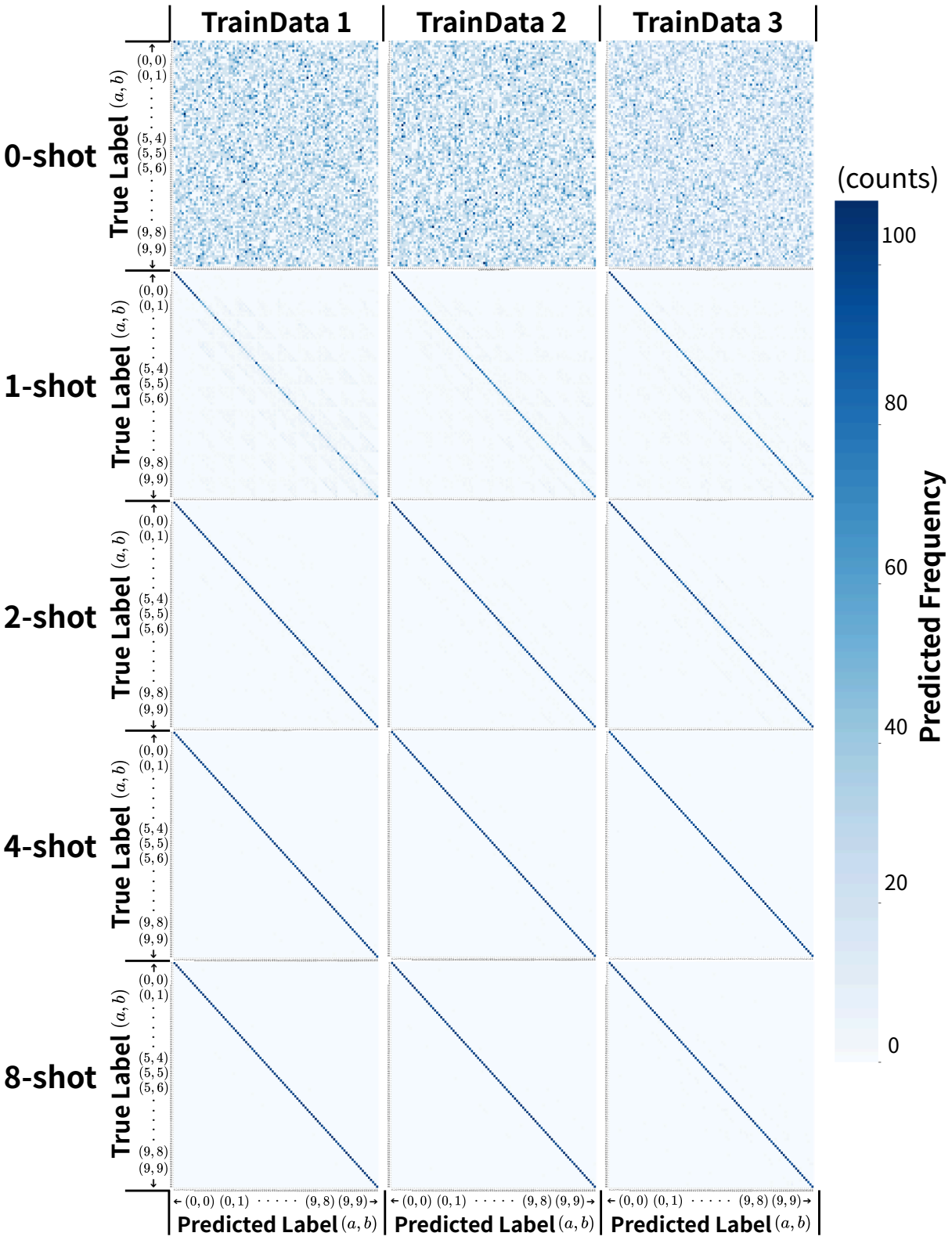

Figure 5: *Visualizing each model's probing results as confusion matrices.* On each confusion matrix, the vertical axis represents the true labels $(a, b) \in \{0, \ldots, 9\}^2$ (100 classes), and the horizontal axis shows the predicted labels obtained via probing (100 classes). For readability, only a subset of labels—such as (0,0), (0,1), ⋯, (9,8), (9,9) —is displayed on each axis. Cell intensity reflects the frequency of predictions. In the 2-, 4-, and 8-shot settings, entries are strongly concentrated along the diagonal, indicating high identification accuracy, whereas in the 0-shot setting, the matrix shows no discernible pattern. In the 1-shot setting, some misclassifications are observed.