# Transformers Can Learn Meta-skills for Task Generalization in In-Context Learning

**Ying Fan**[1], **Steve Yadlowsky**[2], **Dimitris Papailiopoulos**[1], **Kangwook Lee**[1]
[1]University of Wisconsin-Madison [2]Google DeepMind

## Abstract

This study investigates the task generalization exhibited by Transformer models. We hypothesize that Transformers exhibit generalization to unseen tasks by learning "meta-skills", high-level skills that enable models to develop new skills through composition. To test our hypothesis, we conduct extensive in-context learning (ICL) experiments, viewing ICL of a function class as a skill. Our experiments demonstrate that Transformers have high task generalization abilities, as they can effectively in-context learn unseen function classes. This provides strong evidence for our hypothesis, as such generalization cannot occur without the learning of meta-skills. Furthermore, our results suggest that Transformers learn these meta-skills in a sample-efficient and unsupervised manner. Lastly, we show that the learned meta-skills generalize variadically, meaning they can be applied to compositions of an unseen number of skills. This hints at the possibility that Transformers possess strong weak-to-strong generalization abilities, enabling them to perform a greater number of reasoning or composition steps than they have been explicitly taught.

## 1 Introduction

Large Language Models (LLMs) have shown remarkable ability to generalize to unseen tasks, which is believed to be possible because LLMs can "flexibly combine, as needed, the basic skills it has learned" [29]. We refer to this high-level skill required for skill composition as a "meta-skill" and hypothesize that LLMs and Transformer models can learn such meta-skills from data in an unsupervised and sample-efficient manner.

In this work, we empirically study our hypothesis through the lens of in-context learning (ICL) in Transformer models [5, 9]. While there have been a lot of studies studying how ICL works, the ability of Transformers to generalize ICL to unseen function classes is rather underexplored in the literature [15, 21, 4, 3, 18, 16]. A notable exception is the work by [27], in which the authors reported a setting where Transformer models fail at task generalization in in-context learning. Specifically, they found that even after the models successfully learned to perform ICL on individual tasks, they struggled to extend this learning to a simple composition of individual tasks. More specifically, let $\mathcal{A}, \mathcal{B}$ be two function classes, and define $\mathcal{A}+\mathcal{B} := \{f | f = f_1 + f_2, f_1 \in \mathcal{A}, f_2 \in \mathcal{B}\}$. The model trained to perform ICL on both $\mathcal{A}$ and $\mathcal{B}$ could not perform ICL on $\mathcal{A}+\mathcal{B}$.

This result suggests that the meta-skills required for skill composition are *not* learned through the standard pretraining setup. In the context of ICL, a meta-skill would involve (1) identifying if in-context samples come from a composite function that cannot be fit by any basic ICL skills, (2) identifying the needed combination of basic ICL skills, and (3) applying a composite ICL skill on-the-fly: See Figure 1.

*How to teach Transformers meta-skills?* One approach is to provide demonstrations of skill compositions as part of the training data. However, if a model is trained on $\mathcal{A}, \mathcal{B}, \mathcal{A}+\mathcal{B}$ and can fit functions from $\mathcal{A}+\mathcal{B}$, it is unclear whether it learned a meta-skill or simply treated $\mathcal{A}+\mathcal{B}$ as another base skill.

NeurIPS 2024 Workshop on Compositional Learning: Perspectives, Methods, and Paths Forward.

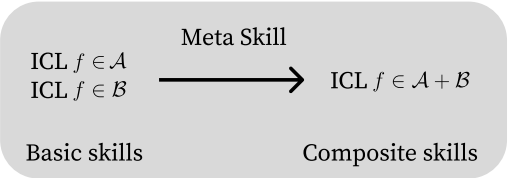



Figure 1: Illustration of basic skills, composite skills and meta-skills.

Let us consider a scenario where a model is trained on $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{A}+\mathcal{B}, \mathcal{B}+\mathcal{C}$. If the model trained this way can perform ICL on various unseen complex function classes, e.g., $\mathcal{A}+\mathcal{C}$ or $\mathcal{A}+\mathcal{D}$, there is strong evidence of meta-skill learning for the following reasons: Instead of simply learning how to fit complex functions $\mathcal{A}+\mathcal{B}, \mathcal{B}+\mathcal{C}$, the model must have learned that these functions are composite functions from two function classes, and learns how to in-context compose basic skills to fit such composite functions. Otherwise, the model cannot fit unseen combinations such as $\mathcal{A}+\mathcal{C}, \mathcal{A}+\mathcal{D}$.

In this paper, we empirically show such trained models can perform ICL on unseen complex function classes with various training and test set designs. Furthermore, while the meta-skill is learned only with two input base skills, we show that the model can successfully fit composite functions involving more than two functions, e.g., $\mathcal{A}+\mathcal{B}+\mathcal{C}$. This implies the learned meta-skill generalizes to an unseen number of input arguments, hinting at the possibility that Transformers possess strong weak-to-strong generalization abilities [6], performing a larger number of composition steps than explicitly taught.

Additionally, we show Transformers can identify the input and output to the skill composition in an unsupervised manner by not providing the model with any extra labels indicating basic or composite function classes. Mathematically, this can be viewed as – the model learns a meta-skill from an unpaired set of inputs and labels by self-identifying the underlying associations between inputs and outputs and then learning the mapping, resembling unsupervised learning of data transformation [31].

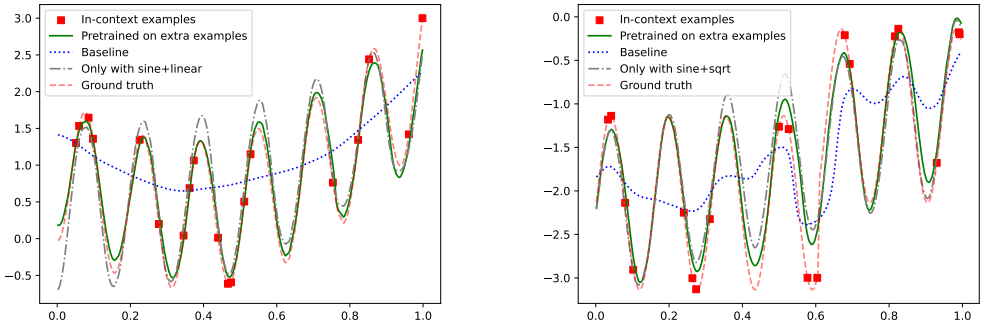
Our contributions include: **(1)** We formalize the concept of meta-skills via the lens of ICL: We investigate how to teach the Transformer model meta-skills with compositional operations like addition, maximum and multiplexing. **(2)** We provide empirical evidence that Transformers can learn meta-skills from data in an unsupervised and sample-efficient manner (with a small number of composite function classes). **(3)** We show that weak-to-strong task generalization is possible for ICL if the model both learns ICL on orthogonal bases as basic skills and a meta-skill to fit a weighted sum of limited seen function classes, which uncovers a new possibility of improving ICL.

## 2 Related work

**In-context learning.** Recent work has shown great promise that in-context learning can solve classical learning tasks like classification, regression, and reinforcement learning [17, 13, 11]. Why such in-context learning ability emerges has also been explored from different perspectives, such as implicit Bayesian inference [25] and gradient descent [24].

**Compositional generalization in ICL and LLMs.** Recent studies have been conducted on the compositional generalization properties of ICL. In ICL, wxample-level compositional generalization ability, i.e., how in-context examples help to infer the label of the unseen query, has been extensively explored [14, 1, 10, 28, 26]. In the more general LLM domain, there has also been a line of work focusing on multi-stage prompting or modular design to help the model solve complex tasks that might require decomposing to simpler problems [12, 8, 30, 22, 20, 23]. To our best knowledge, compositional generalization from a view of specific "meta-skills" remains under-explored. [18] studied the compositional generalization ability of ICL through the Bayesian Prism, but they have not test compositions of more basic classes, but we highlight the ability of weak-to-strong generalization. Besides, they only tests on linear mixtures, while we test beyond the scheme of linear mixtures like maximum.

**Skill learning in ICL.** [2] explored how to choose few-shot examples using skill-based descriptions to solve the test case. [7] provide examples for both basic skills and how to compose the skills in the prompt to help LLMs solve unseen test questions that requires composing skills. We want to emphasize that their works focus on example-level generalization which only changes how to prompt the model, while our setup focuses on pre-training task distribution. To our best knowledge, teaching the model to compose basic skills in pre-training is yet to be explored in ICL. The most related work is [21], where they only show that a diverse distribution of pre-training tasks is beneficial for such




(a) Test class: `sine+quad`.

(b) Test class: `sine+heaviside`.

Figure 2: Visualizations for generalization to unseen addition compositions. Base function classes for 2a: {`sine`,`sqrt`, `linear`,`quad`}; 2b: {`sine`,`sqrt`,`heaviside`, `quad`}. "Pretrained on extra examples" indicates the training set "**Ours**": for 2a we add composite training examples from `sine+sqrt` and `sine+linear`, and for 2b we add `sine+sqrt` and `heaviside+ quad`."Only with xxx" indicates the best approximation defined in Section 4.1. The best approximation still tries to fit the seen function class during training, but ours can generalize to the pattern that is not within any seen function class, showing the evidence that it acquires the "meta-skill".

generalization, without exploring specific structures for the pre-training and test tasks in terms of meta-skills and compositional generalization.

# 3 Preliminaries

## 3.1 In-context learning and basic function classes

For ICL, we follow the tranditional setup in [9]: Given a function class $\mathcal{F}$ with a distribution of functions $\mathcal{D}_{\mathcal{F}}$ and a distribution of inputs $\mathcal{D}_{\mathcal{X}}, \mathcal{X} \subset \mathbb{R}$, we create random training prompts $P_i = (x_1, f(x_1), .., x_{i+1})$ where $f : \mathbb{R} \to \mathbb{R}, f \sim \mathcal{D}_{\mathcal{F}}$, $x_i \sim \mathcal{D}_{\mathcal{X}}$. Then we train the transformer model $M_\theta$ to minimize the loss below:

$$\mathbb{E}_P[\sum_{i=0}^{H} l(M_\theta(P^i), f(x_{i+1}))], \tag{1}$$

where $l(\cdot, \cdot)$ is $L_2$ loss. Specific training and evaluation details for all experiments are in Appendix A.

For basic function classes, we consider basic functions: `linear`, `quad`, `sqrt`, `heaviside`, `sine`. See detailed definitions in Appendix B.1.1.

## 3.2 Meta-skills and function composition operations

**We view the ability to generalize to unseen combinations with specific function composition operations as the corresponding meta-skill.**

For function composition operations, we consider manipulations in the $y$ space to composite function, specifically, addition, maximum, and multiplexing in this section: For **Addition**, we perform addition in the $y$ space; For **Maximum**, we perform maximum operation in the $y$ space; For **Multiplexing**, we perform the local combination of two functions in the $y$ space by dividing the input space into halves. For example, for the composition function class (`linear`,`sine`), we first sample functions in `linear` and `sine` respectively, and composite $y$ value to is from `linear` if $x \in [0, 0.5)$ and from `sine` if $x \in [0.5, 1]$.

# 4 Exploring the meta-skill learning ability

## 4.1 Different training sets

We consider different training settings: (1) **Baseline**: We train only with the base function classes $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$; (2) **Ours**: We train with both base function classes and extra composite function classes (defined separately for each setup).; (3) **Lower bound**: We train on the target function class as a reference of the lower bound of the error; (4) **Best approximation**: We only train with each single composite function class in **Ours**, presenting the results with the lowest errors when tested against the target function class, and treat it as the "best approximation" from all single function classes in the training set. If the error from the **Best approximation** is still larger than **Ours**, then the improved performance in **Ours** is not simply from containing some "closer" function class to the test one during training: To outperform "best approximation", the model needs to acquire the "meta-skill" rather than only choosing the best approximation in seen function classes in a Bayesian way.

We include the detailed setup of the compositional training and test splits in Appendix B.1.2.

## 4.2 Compositional generalization test

**The model trained with compositional examples outperforms other training setups.** As the test errors shown in Table 1,2 for addition and Table 5,6 for maximum, we observe consistent improvement compared with baseline training which does not contain training with any composite examples. Moreover, it also outperforms the best approximation of the training function classes, where their patterns are quite distinct in the visualizations (see the discussion below).

**The model trained with extra compositional examples learns the meta-skill.** In Figure 2 and 4, we can observe that given a new composite function class, the "best approximation" in the training set still tries to fit the original function class it has been trained on, but ours can generalize better to the unseen composition, sometimes near-perfectly. This suggests that when presented with both basic and composite function classes, the model does not only learn the basic skills of fitting each single function class (either basic or composite) but also discovers the meta-skill and applies it to unseen combinations.

| Training set | `sine+quad` |
|---|---|
| baseline | 0.6238 |
| `sine+linear` | 0.1198 |
| `sine+sqrt` | 0.4245 |
| ours | **0.0704** |
| lower bound | 0.0025 |

Table 1: Test errors from Partial Composition Setup 1 (Appendix B.1.2) on addition.

| Training set | `sine+quad` | `sine+heaviside` |
|---|---|---|
| baseline | 0.5473 | 0.5334 |
| `sine+sqrt` | 0.4245 | 0.0965 |
| ours | **0.2517** | **0.0822** |
| lower bound | 0.0025 | 0.0185 |

Table 2: Test errors from Cross Composition Setup 1 (Appendix B.1.2) on addition.

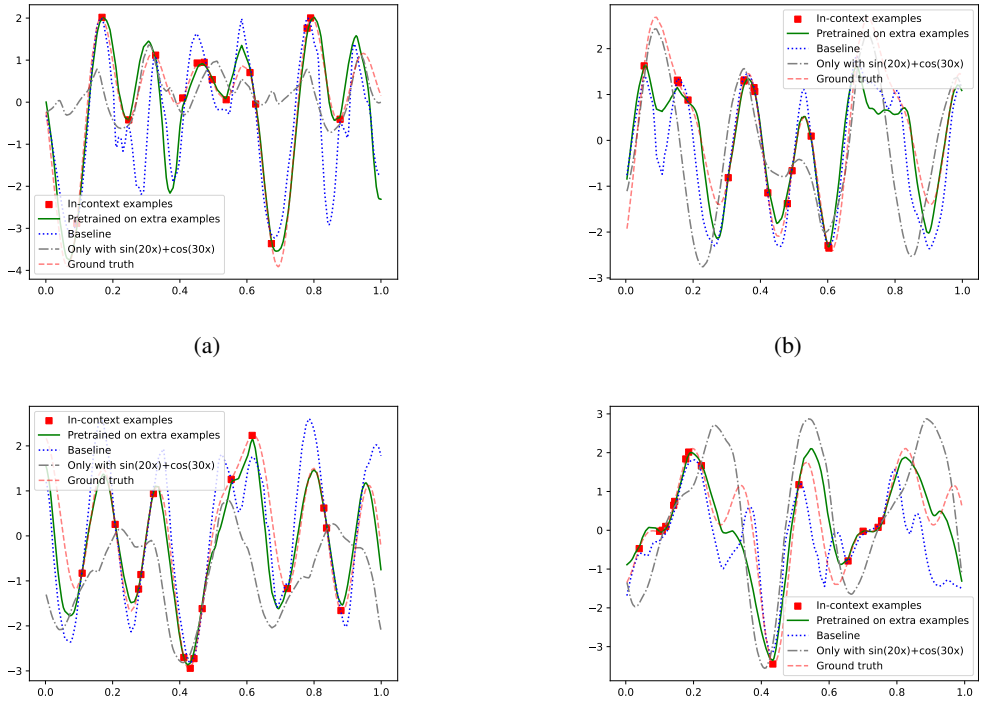
**Transformers are not good at local combination without explicit training on local combination examples; Local combination can also be meta-learned.** From visualizations in Figure 5 and the test error in Table 17 and 18, we can see that without adding extra examples, the model is not very good at doing multiplexing/local combinations. However, when we add local combination examples in the training set, the model can generalize nearly perfectly to the unseen combinations, which means it learns such meta-skill very well once presented with such examples.

We present more results from more training and test setups in Appendix B.2, the ablation study on different training function classes in Appendix B.3, more results from multiplexing in Appendix B.4, and more visualizations in Appendix D, which all align with the claims above.

# 5 Weak-to-strong generalization

In this section, we consider the addition operation and use functional bases such as **Fourier** and **Legendre** bases as the basic function classes, and test whether transformers can generalize to the span of the base in the function space by only training with combinations of two classes. This setup could be viewed as weak-to-strong generalization. We include detailed definition of the base and compositional function classes, training and test splits in Appendix C.1 and C.2.

**Weak-to-strong generalization is possible with Fourier and Legendre bases.** As shown in Table 3 and Figure 6 for Fourier bases, and Table 4 and Figure 7 for Legendre bases, our training




(a) (b) (c) (d)

Figure 3: Visualization of Fourier Setup 1, test on `All sine`.

enables the model to generalize to higher-order compositions in the span of the functional bases. In Figure 6 and 7, the predictions align much better with the ground truth patterns compared with the baselines, which shows the evidence of acquiring meta-skills.

| Training set | `Random sine` | `All sine` |
|---|---|---|
| baseline | 1.3764 | 1.1307 |
| `sin-20+cos-40` | 0.6880 | 1.9298 |
| `sin-10+cos-20` | 0.7060 | 1.7311 |
| `sin-10+cos-40` | 0.7102 | 2.3908 |
| `sin-40+cos-40` | 0.7533 | 2.0551 |
| `sin-20+cos-30` | 0.6974 | 1.7272 |
| ours | **0.3858** | **0.5001** |
| lower bound | 0.0312 | 0.0470 |

Table 3: Test errors from Fourier bases.

| Training set | `Random poly` | `++--+` | `+--++` | `+++++` |
|---|---|---|---|---|
| baseline | 0.3085 | 0.4368 | 0.3291 | 0.3630 |
| `Poly-2+Poly-3` | 0.5246 | 0.3889 | 0.3578 | 0.4706 |
| `Poly-2+Poly-4` | 0.7580 | 0.7077 | 0.7242 | 0.7785 |
| `Poly-1+Poly-4` | 0.5538 | 0.4658 | 0.5272 | 0.5682 |
| `Poly-3+Poly-5` | 0.7875 | 0.7801 | 0.7217 | 0.8051 |
| `Poly-3+Poly-4` | 0.3629 | 0.2909 | 0.2794 | 0.4286 |
| ours | **0.1086** | **0.2351** | **0.1925** | **0.2169** |
| lower bound | 0.0038 | 0.0054 | 0.0063 | 0.0046 |

Table 4: Test errors from Legendre bases.

We include results and visualizations from more training and test setups in Appendix C and more visualizations in Appendix E.

## 6 Discussion and conclusion

Although showing promising results, our study has several key limitations. While we demonstrate meta-skill learning and generalization capabilities on one-dimensional function classes using simulated data, evaluating these approaches on higher-dimensional datasets across diverse domains remains open. Besides, we explored only a limited set of compositional operations like addition, max, and multiplexing; the vast space of possible compositions requires further investigation, though addition with orthogonal bases already has pretty good expressive power in the function spaces. Our findings highlight new possibilities for enhancing ICL capabilities of large language models through strategic training on skill compositions. Further studies in this direction could unlock systematic generalization to novel tasks.

## References


[1] Shengnan An, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Jian-Guang Lou, and Dongmei Zhang. How do in-context examples affect compositional generalization? *arXiv preprint arXiv:2305.04835*, 2023. 2

[2] Shengnan An, Bo Zhou, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Weizhu Chen, and Jian-Guang Lou. Skill-based few-shot selection for in-context learning. *arXiv preprint arXiv:2305.14210*, 2023. 2

[3] Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language models. *arXiv preprint arXiv:2307.15936*, 2023. 1

[4] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[6] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision, 2023. 2

[7] Jiaao Chen, Xiaoman Pan, Dian Yu, Kaiqiang Song, Xiaoyang Wang, Dong Yu, and Jianshu Chen. Skills-in-context prompting: Unlocking compositionality in large language models. *arXiv preprint arXiv:2308.00304*, 2023. 2

[8] Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*, 2022. 2

[9] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022. 1, 3, 8

[10] Shivanshu Gupta, Sameer Singh, and Matt Gardner. Coverage-based example selection for in-context learning. *arXiv preprint arXiv:2305.14907*, 2023. 2

[11] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023. 2

[12] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022. 2

[13] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022. 2

[14] Itay Levy, Ben Bogin, and Jonathan Berant. Diverse demonstrations improve in-context compositional generalization. *arXiv preprint arXiv:2212.06800*, 2022. 2

[15] Hongkang Li, Meng Wang, Songtao Lu, Hui Wan, Xiaodong Cui, and Pin-Yu Chen. Transformers as multi-task feature selectors: Generalization analysis of in-context learning. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023. 1

[16] Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning, 2024. 1

[17] Lovre. Who models the models that model models? an exploration of gpt-3's in-context model fitting ability. 2022. 2

[18] Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the bayesian prism. *arXiv preprint arXiv:2306.04891*, 2023. 1, 2

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

[20] Rahul Ramesh, Mikail Khona, Robert P Dick, Hidenori Tanaka, and Ekdeep Singh Lubana. How capable can a transformer become? a study on synthetic, interpretable tasks. *arXiv preprint arXiv:2311.12997*, 2023. 2

[21] Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2

[22] Simon Schug, Seijin Kobayashi, Yassir Akram, Maciej Wołczyk, Alexandra Proca, Johannes Von Oswald, Razvan Pascanu, João Sacramento, and Angelika Steger. Discovering modular solutions that generalize compositionally. *arXiv preprint arXiv:2312.15001*, 2023. 2

[23] Jonathan Thomm, Aleksandar Terzic, Geethan Karunaratne, Giacomo Camposampiero, Bernhard Schölkopf, and Abbas Rahimi. Limits of transformer language models on algorithmic learning. *arXiv preprint arXiv:2402.05785*, 2024. 2

[24] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023. 2

[25] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021. 2

[26] Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. Do large language models have compositional ability? an investigation into limitations and scalability. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024. 2

[27] Steve Yadlowsky, Lyric Doshi, and Nilesh Tripuraneni. Pretraining data mixtures enable narrow model selection capabilities in transformer models. *arXiv preprint arXiv:2311.00871*, 2023. 1

[28] Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. Compositional exemplars for in-context learning. *arXiv preprint arXiv:2302.05698*, 2023. 2

[29] Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. Skill-mix: A flexible and expandable family of evaluations for ai models. *arXiv preprint arXiv:2310.17567*, 2023. 1

[30] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022. 2

[31] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020. 2

# A Training and evaluation details

**Training.** We use GPT-2 [19] for the transformer backbone following [9]. We use 12 layers, 8 heads, and an embedding space with 256 dimensions. During training, we use Adam optimizer with learning rate of $10^{-4}$ and batch size of 256, and train the model for 100K steps for convergence. We consider $H$ up to 19 where $H$ is defined in Section 3.1. When training with multiple function classes, we equally sample from them in each batch without any curriculum learning.

**Evaluation.** During evaluation we use 15 in-context examples for each function class, with all $x$ randomly sampled from the uniform distribution. For each evaluation, we test on 500 random query positions given 15 random ICL examples from the test function class. Then we average over 1000 evaluations to calculate the final test error for each setup.

# B Details for compositional generalizaition tests in Section 4

## B.1 Setup

### B.1.1 Basic function classes

We consider the following basic function classes for the experiments in Section 4, where we use no label noise: $y = f(x)$ and $x \in [0, 1]$.

- `linear`: $f(x) = wx + b$, where $w \sim \mathcal{N}(0, 1), b \sim \mathcal{N}(0, 1)$.
- `quad`: $f(x) = 5(x - h)^2 + b$, where $h \sim \mathcal{N}(0.5, 0.01), b \sim \mathcal{N}(0, 1)$.
- `sqrt`: $f(x) = \sqrt{\min(5x + h, 0)} + b$, where $h \sim \mathcal{N}(0.1, 0.01), b \sim \mathcal{N}(0, 1)$.
- `heaviside`: $f(x) = 0.5 + 0.5sgn(x + h) + b$, where $h \sim \mathcal{N}(-0.5, 0.01), b \sim \mathcal{N}(0, 1)$.
- `sine`: $f(x) = sin(40x + h)$, $h \sim \mathcal{N}(0, 1)$.

### B.1.2 Compositional training and test splits

Here we explore two setups of training and test splits where $\oplus$ could be `addition`, `maximum` or `multiplexing`:

- (**Partial Composition**) Train with $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{A} \oplus \mathcal{B}, \mathcal{A} \oplus \mathcal{C}$, and test with $\mathcal{A} \oplus \mathcal{D}$.
- (**Cross Composition**) Train with $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{A} \oplus \mathcal{B}, \mathcal{C} \oplus \mathcal{D}$, and test with: $\mathcal{A} \oplus \mathcal{C}, \mathcal{A} \oplus \mathcal{D}$.

**Instantiations .** We test two instantiations for each setup with compositional operation $\oplus$:

For **Partial Composition**, we provide **Setup 1**: $\mathcal{A}$:`sine`, $\mathcal{B}$:`sqrt`, $\mathcal{C}$:`linear`, $\mathcal{D}$:`quad`; and **Setup 2**: $\mathcal{A}$:`sine`, $\mathcal{B}$:`sqrt`, $\mathcal{C}$:`heaviside`, $\mathcal{D}$:`quad`;

For **Cross Composition**, we provide **Setup 1**: $\mathcal{A}$:`sine`, $\mathcal{B}$:`sqrt`, $\mathcal{C}$:`heaviside`, $\mathcal{D}$:`quad`; and **Setup 2**: $\mathcal{A}$:`sine`, $\mathcal{B}$:`quad`, $\mathcal{C}$:`heaviside`, $\mathcal{D}$:`linear`.

## B.2 Compositional generalization performance for addition and maximum

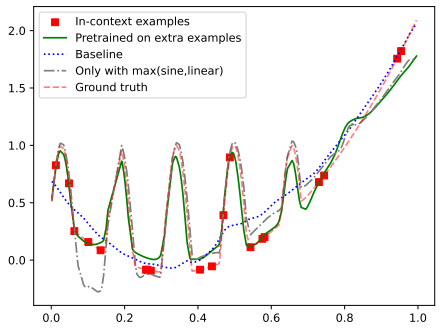
We already presented the test errors from Partial Composition & Cross Composition for addition from Setup 1 in Table 1, 2 and Figure 2. Below we present the test errors for maximum from the same setting in Tble 5 and 6, and visualizations in Figure 4, which shows that our training setup outperforms the baseline setups and indicates the acquisition of the meta skill.

| Training set | max(`sine`,`quad`) |
|---|---|
| baseline | 0.0611 |
| max(`sine`,`linear`) | 0.0381 |
| max(`sine`,`sqrt`) | 0.3193 |
| ours | **0.0219** |
| lower bound | 0.0023 |

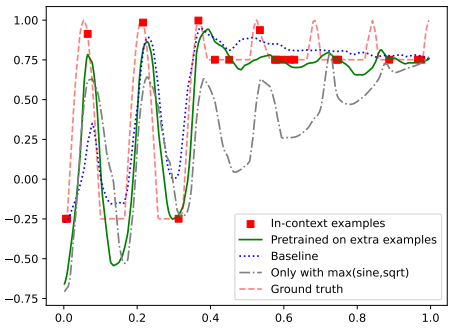
Table 5: Test errors from Partial Composition Setup 1 on maximum.

| Training set | `sine+quad` | `sine+heaviside` |
|---|---|---|
| baseline | 0.0488 | 0.0507 |
| `sqrt+sine` | 0.3193 | 0.1018 |
| ours | **0.0321** | **0.0439** |
| lower bound | 0.0023 | 0.0129 |

Table 6: Test errors from Cross Composition Setup 1 on maximum.




(a) Partial Composition Setup 1, tested on max(sine, quad).




(b) Cross Composition Setup 1, tested on max(sine, heaviside).

Figure 4: Visualizations for generalization to unseen maximum compositions: max(sine, quad) for 4a and max(sine, heaviside) for 4b.

We also present other missing tables for Setup 2 in all generalization tests in Section 4: Table 7, 8 for addition, Table 9, 10 for maximum, and Table 11, 12 for multiplexing.

| Training set | sine+quad |
|---|---|
| baseline | 0.5473 |
| sine+heaviside | 0.2216 |
| sine+sqrt | 0.4245 |
| ours | **0.1540** |
| lower bound | 0.0025 |

Table 7: Test errors from Partial Composition 2 on addition.

| Training set | sine+heaviside | sine+linear |
|---|---|---|
| baseline | 0.5295 | 0.4173 |
| sine+quad | 0.1770 | 0.0935 |
| ours | **0.1306** | **0.0610** |
| lower bound | 0.0185 | 0.0011 |

Table 8: Test errors from Cross Composition 2 on addition.

| Training set | max(sine,quad) |
|---|---|
| baseline | 0.0488 |
| max(sine,heaviside) | 0.1262 |
| max(sine,sqrt) | 0.3193 |
| ours | **0.0359** |
| lower bound | 0.0023 |

Table 9: Test errors from Partial Composition 2 on maximum.

| Training set | sine+heaviside | sine+linear |
|---|---|---|
| baseline | 0.0669 | 0.0263 |
| sine+quad | 0.1426 | 0.0372 |
| ours | **0.0448** | **0.0166** |
| lower bound | 0.0129 | 0.0034 |

Table 10: Test errors from Cross Composition 2 on maximum.

| Training set | (sine, quad) |
|---|---|
| baseline | 0.2214 |
| (sine, heaviside) | 0.0244 |
| (sine, sqrt) | 0.0464 |
| ours | **0.0121** |
| lower bound | 0.0030 |

Table 11: Test errors from Partial Composition 2 on multiplexing.

| Training set | (sine, heaviside) | (sine, linear) |
|---|---|---|
| baseline | 0.2180 | 0.2347 |
| (sine, quad) | 0.0319 | **0.0068** |
| ours | **0.0264** | 0.0080 |
| lower bound | 0.0084 | 0.0011 |

Table 12: Test errors from Cross Composition 2 on multiplexing.

### B.3 Effect of the basic function classes

In this section, we investigate whether exposure to the basic function classes also facilitates the acquisition of meta-level skills. Specifically, we probe whether exemplars including both the basic components and the target meta-level skill are necessary for successful generalization, using the operation of addition as an illustrative case study.

**Exposure to both basic and composite examples are beneficial.** As shown in Table 13 and 14, training with both the basic and composite examples outperforms training with basic or compositional examples only, suggesting that exposure to both are beneficial for meta-skill learning.

| Training set | sine+quad |
|---|---|
| baseline | 0.6238 |
| sine+linear,sine+sqrt | 0.1044 |
| ours | **0.0704** |
| lower bound | 0.0025 |

Table 13: Test errors from Partial Composition.

| Training set | sine+quad | sine+heaviside |
|---|---|---|
| baseline | 0.5473 | 0.5334 |
| sine+sqrt, heaviside+quad | 0.3066 | 0.0948 |
| ours | **0.2517** | **0.0822** |
| lower bound | 0.0025 | 0.0185 |

Table 14: Test errors from Cross Composition.

Test errors from Setup 1.

We also present results from Setup 2 in Table 15 and 16 which convey the similar message.

| Training set | sine+quad |
|---|---|
| baseline | 0.5473 |
| sine+sqrt, sine+heaviside | 0.1785 |
| ours | **0.1540** |
| lower bound | 0.0025 |

Table 15: Test errors from Partial Composition.

| Training set | sine+heaviside | sine+linear |
|---|---|---|
| baseline | 0.5295 | 0.4173 |
| sine+quad, linear+heaviside | 0.1733 | 0.1022 |
| ours | **0.1306** | **0.0610** |
| lower bound | 0.0185 | 0.0011 |

Table 16: Test errors from Cross Composition.

Test errors from Setup 2.

### B.4 Local combination (multiplexing)

Another important question to ask is whether the better approximation on the unseen test compositions comes from local approximation with seen function classes (for example, the multiplexing operation introduced in Section 3.2) or actually learning the meta-skill. To answer that, we also explore whether transformers are naturally good at doing localizations of learned function classes without explicit training.

**Transformers are not good at local combinations without explicit training.** From the test error in Table 17 and 18 and visualizations in Figure 5, we can see that without adding extra examples, the model is not very good at doing local combinations.

| Training set | (sine, quad) |
|---|---|
| baseline | 0.2524 |
| (sine, linear) | 0.0114 |
| (sine, sqrt) | 0.0464 |
| ours | **0.0057** |
| lower bound | 0.0030 |

Table 17: Test errors from Partial Composition Setup 1 on multiplexing.

| Training set | (sine, quad) | (sine, heaviside) |
|---|---|---|
| baseline | 0.2214 | 0.2571 |
| (sine, sqrt) | 0.0464 | 0.0448 |
| ours | **0.0170** | **0.0343** |
| lower bound | 0.0030 | 0.0084 |

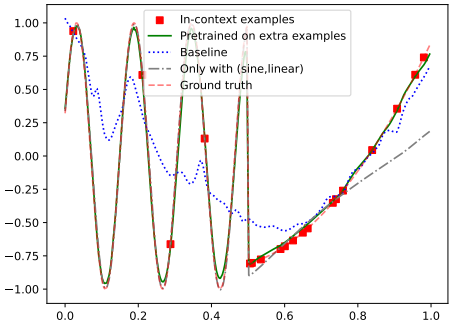
Table 18: Test errors from Cross Composition Setup 1 on multiplexing.

**Local combination can also be meta-learned.** When we add local combination examples in the training set, the model can generalize nearly perfectly to the unseen combinations, which means it learns such meta-skill very well once presented with such examples. However, we should notice that unlike addition and maximum, such composition also depends on the order of the input variables and also how to divide the localization areas.

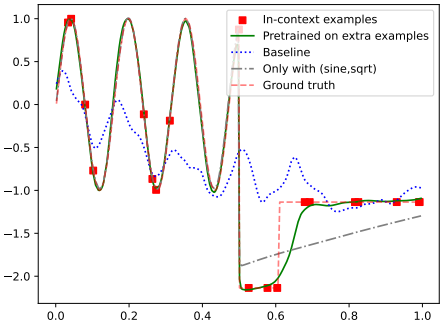
We include more visualizations in Appendix D.

## C Full detail from weak-to-strong generalization in Section 5

In this section, we consider the addition operation and replace the basic function classes with functional bases in function analysis such as **Fourier** and **Legendre** bases, and test whether transformers




(a) Partial Composition Setup 1




(b) Cross Composition Setup 1

Figure 5: Visualizations for generalization to unseen multiplexing compositions (`sine`, `quad`) and (`sine`, `heaviside`).

can generalize to the span of the base in the function space by only training with combinations of two classes, which could be viewed as weak-to-strong generalization.

### C.1 Generalization tests with Fourier bases

In this section, we explore whether models trained on sum of up to two Fourier basic classes can generalize to sum of up to five basic classes.

#### C.1.1 Training classes

- **Basic classes**: `sin-`$k$: $f(x) = wsin(kx) + b$ and $f(x) = wcos(kx) + b$,, where $k = 10, 20, 30, 40, w \sim \mathcal{N}(0, 1), b \sim \mathcal{N}(0, 1)$.
- **Composite classes** `sin-`$k_1$`+cos-`$k_2$: $f(x) = w_1 sin(k_1 x) + w_2 cos(k_2 x) + b_1 + b_2$, $w_i \sim \mathcal{N}(0, 1), b_i \sim \mathcal{N}(0, 1)$.
- **Baseline**: All basic classes `sin-`$k$ and `cos-`$k$ for $k = 10, 20, 30, 40$.
- **Ours**: All base function classes and composite classes. For composite classes, we consider **Setup 1**: $(k_1, k_2) \in \{(20, 40), (10, 20), (10, 40), (40, 40), (20, 30)\}$; **Setup 2**: $(k_1, k_2) \in \{(40, 20), (20, 10), (40, 10), (10, 30), (30, 20)\}$.

#### C.1.2 Test classes

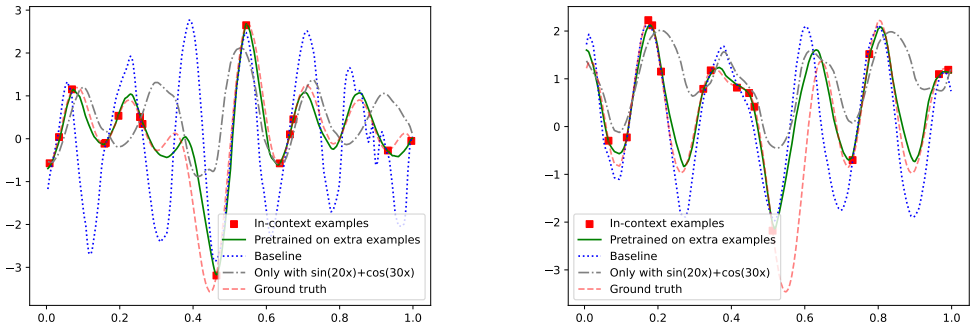
- `Random sine`: $f(x) = w_1 sin(10x + \phi_1) + ... + w_4 sin(40x + \phi_4) + b_1 + b_2 + b_3 + b_4$, where $w_i \sim \mathcal{N}(0, 1), b_i \sim \mathcal{N}(0, 1), \phi_i \sim \mathcal{N}(0, 1)$.
- `All sine`: $f(x) = sin(10x + \phi_1) + ... + sin(40x + \phi_4)$, where $\phi_i \sim \mathcal{N}(0, 1)$. It is a harder function class than the random one, enforcing nonzero components of each basic class.

Notice that there is no label noise in all functions, $x \in [0, 1]$.

#### C.1.3 Results

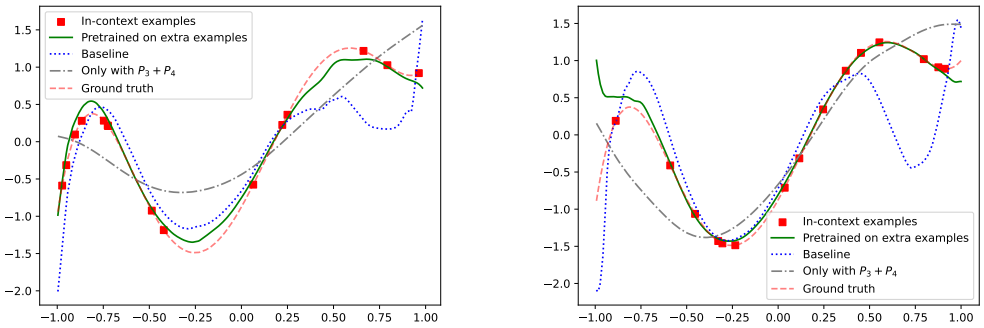
**Weak-to-strong generalization is possible with Fourier bases.** As shown in Table 3 for Setup 1, and Table 19 for Setup 2, our training improves the model's generalization ability to higher-order compositions in the span of the Fourier bases. The predictions align surprisingly well with the ground truth in the visualizations (see Figure 6 for visualizations)[1].

[1]We also present more visualizations in Appendix E.




(a) (b)

Figure 6: Visualization of Fourier Setup 1, test on `all sine`.




(a) (b)

Figure 7: Visualization of Legendre Setup 1, test on $P_1 + P_2 - P_3 - P_4 + P_5$.

| Training set | `Random sine` | `All sine` |
|---|---|---|
| baseline | 1.3764 | 1.1307 |
| `sin-40+cos-20` | 0.7196 | 1.7974 |
| `sin-20+cos-10` | 0.7106 | 1.5048 |
| `sin-40+cos-10` | 0.6599 | 1.3525 |
| `sin-10+cos-30` | 0.7371 | 1.3149 |
| `sin-30+cos-20` | 0.6626 | 1.5824 |
| ours | **0.5379** | **0.5661** |
| lower bound | 0.0312 | 0.0470 |

Table 19: Test errors from Fourier.

| Training set | `Random poly` | `++--+` | `+--++` | `+++++` |
|---|---|---|---|---|
| baseline | 0.2479 | 0.3947 | 0.3919 | 0.3769 |
| `Poly-1+Poly-2` | 0.3685 | 0.2133 | 0.1894 | 0.3634 |
| `Poly-1+Poly-5` | 0.4272 | 0.3299 | 0.3252 | 0.5212 |
| `Poly-4+Poly-5` | 0.7008 | 0.7730 | 0.7903 | 0.6579 |
| `Poly-2+Poly-5` | 0.5504 | 0.5664 | 0.5522 | 0.7329 |
| `Poly-1+Poly-3` | 0.3125 | 0.2907 | 0.2913 | 0.3479 |
| ours | **0.0692** | **0.1092** | **0.1350** | **0.1216** |
| lower bound | 0.0038 | 0.0054 | 0.0063 | 0.0046 |

Table 20: Test errors from Legendre.

Test errors from Setup 2.

### C.2 Generalization with Legendre Polynomials

In this section, we explore whether models trained on sum of up to two Legendre basic classes can generalize to sum of up to five basic classes.

#### C.2.1 Training classes

- **Basic classes**: `Poly-`$k$: $f(x) = wP_k(x) + b$, where $k = 1, 2, 3, 4, 5, w \sim \mathcal{N}(0, 1), b \sim \mathcal{N}(0, 1)$, $P_k$ is a Legendre polynomial of degree $k$.
- **Composite classes**: `Poly-`$k_1$`+Poly-`$k_2$: $f(x) = w_1 P_{k_1}(x) + w_2 P_{k_2}(x) + b_1 + b_2$, $w_i \sim \mathcal{N}(0, 1), b_i \sim \mathcal{N}(0, 1)$.

- **Baseline**: combination of `Poly-`$k$ for $k = 1, 2, 3, 4, 5$.
- **Ours**: All base function classes and composite classes. For composite classes: Setup 1: $(k_1, k_2) \in \{(2, 3), (2, 4), (1, 4), (3, 5), (3, 4)\}$. Setup 2: $(k_1, k_2) \in \{(1, 2), (1, 5), (4, 5), (2, 5), (1, 3)\}$.

### C.2.2 Test classes

- `Random poly`: $f(x) = w_1P_1(x) + w_2P_2(x) + w_3P_3(x) + w_4P_4(x) + w_5P_5(x) + b_1 + b_2 + b_3 + b_4 + b_5$, $w_i \sim \mathcal{N}(0, 1), b_i \sim \mathcal{N}(0, 1)$.
- $P_1 + P_2 - P_3 - P_4 + P_5$ $(++--+)$: $f(x) = P_1(x) + P_2(x) - P_3(x) - P_4(x) + P_5(x)$.
- $P_1 - P_2 - P_3 + P_4 + P_5$ $(+--++)$: $f(x) = P_1(x) - P_2(x) - P_3(x) + P_4(x) + P_5(x)$.
- $P_1 + P_2 + P_3 + P_4 + P_5$ $(+++++)$: $f(x) = P_1(x) + P_2(x) + P_3(x) + P_4(x) + P_5(x)$.

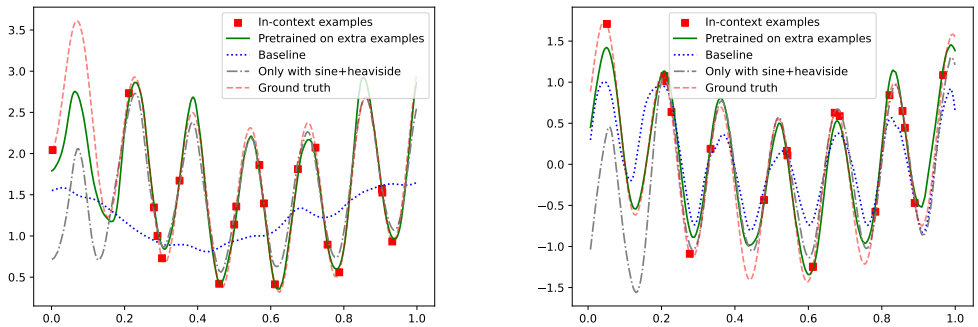
Notice that there is no label noise in all functions, $x \in [-1, 1]$.

### C.2.3 Results

**Weak-to-strong generalization is possible with Legendre bases.** Our experimental results in Table 4 for Setup 1 and Table 20 for Setup 2 demonstrated improved weak-to-strong generalization capabilities. By training the model on examples combining pairs of Legendre polynomial bases through addition, it could generalize to predict sums involving five distinct basis function classes pretty accurately that are never encountered during training in Figure 7[2]. This weak-to-strong generalization from sums of two bases to higher-order sums of multiple bases highlights the model's ability to leverage the additive meta-skill in a compositional way.

## D More visualizations from Section B

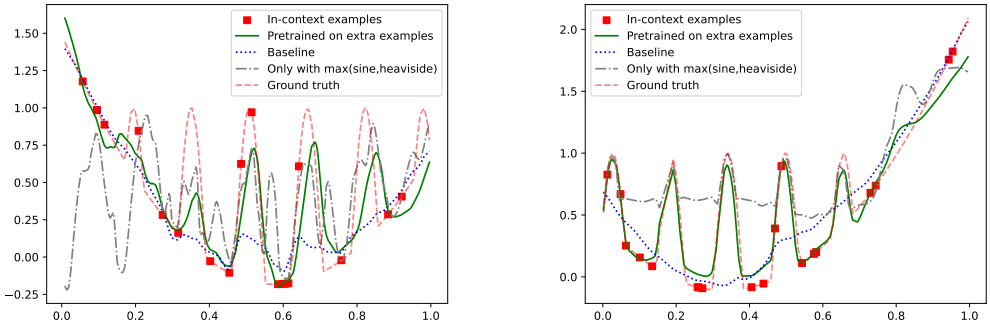
We present more visualizations from generalization tests in Setup 2 in Section B in Figure 8, 11 for addition, Figure 9, 12 for maximum, and Figure 10, 13 for multiplexing.




(a) (b)

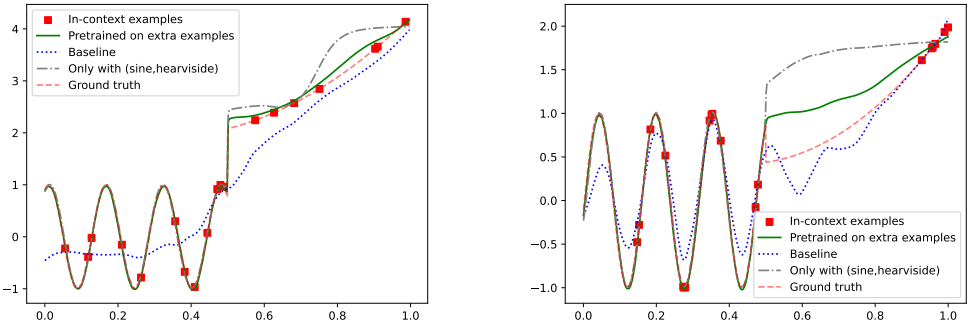
Figure 8: Visualization of Partial Composition Setup 2, test on `sine+quad`.

[2]We also present more visualizations in Appendix E



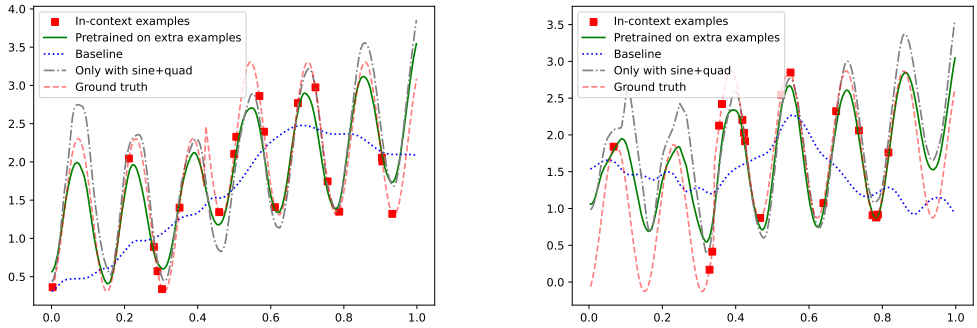

(a) (b)

Figure 9: Visualization of Partial Composition Setup 2, test on max(`sine`, `quad`).



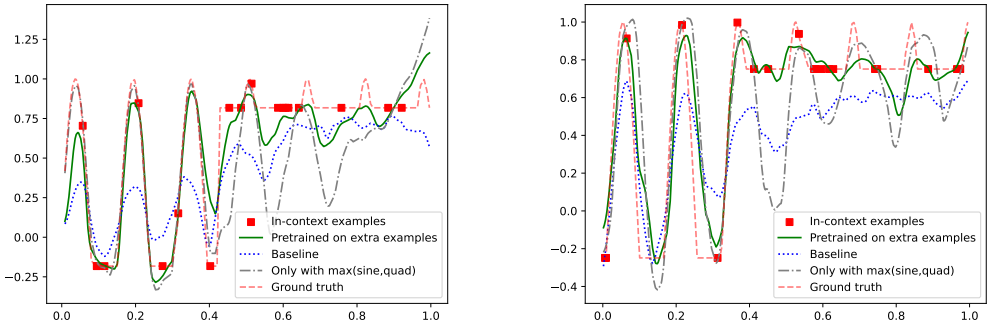

(a) (b)

Figure 10: Visualization of Partial Composition Setup 2, test on multiplexing: (`sine`, `quad`).



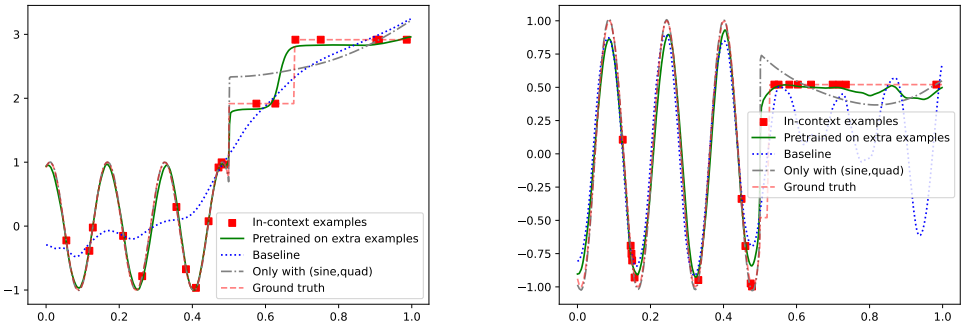

(a) (b)

Figure 11: Visualization of Cross Composition Setup 2, test on `sine`+`heaviside`.




(a) (b)

Figure 12: Visualization of Cross Composition Setup 2, test on max(`sine`, `heaviside`).

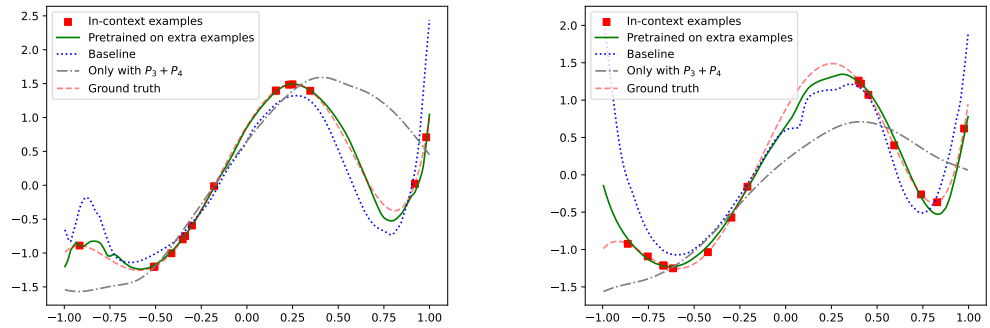



(a) (b)

Figure 13: Visualization of Cross Composition Setup 2, test on multiplexing: (`sine`, `heaviside`).

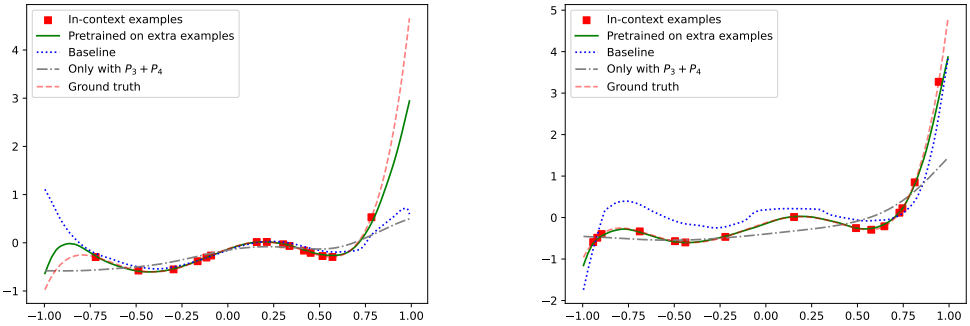
## E More visualizations from Section C

We present more visualizations for Fourier generalization tests in Figure 3, and Legendre generalization tests in Figure 14 and Figure 15.




(a) (b)

Figure 14: Visualization of Legendre Setup 1, test on $P_1 - P_2 - P_3 + P_4 + P_5$.




(a) (b)

Figure 15: Visualization of Legendre Setup 1, test on $P_1 + P_2 + P_3 + P_4 + P_5$.