Iterative Amortized Inference: Unifying In-Context Learning and Learned Optimizers

Sarthak Mittal [1 2 3] **Divyat Mahajan** [1 2 3] **Guillaume Lajoie** [† 2 3] **Mohammad Pezeshki** [† 1]

Abstract

Modern learning systems increasingly rely on amortized learning — the idea of reusing computation or inductive biases shared across tasks to enable rapid generalization to novel problems. This principle spans meta-learning, in-context learning, prompt tuning, learned optimizers and more. While motivated by similar goals, these approaches differ in how they encode and leverage task-specific information. In this work, we propose a unified framework describing how such methods differ primarily in the aspects of learning they amortize — initializations, learned updates, or predictive mappings. We introduce a taxonomy that categorizes amortized models into parametric, implicit, and explicit regimes, based on whether task adaptation is externalized, internalized, or jointly modeled. Building on this view, we identify a key limitation in current approaches: most methods struggle to scale to large datasets because their capacity to process task data at test time (e.g., context size in ICL) is often limited. We propose *iterative amortized inference*, a class of models that refine solutions step-by-step over mini-batches, drawing inspiration from stochastic optimization and yielding performance improvements across different amortization regimes.

1. Introduction

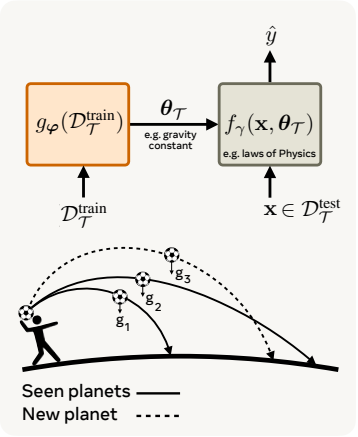
Consider the problem of modeling the motion of an object on various planets, under different gravitational conditions. While each planet has its own gravitational constant, the underlying dynamics, governed by Newtonian physics, remain invariant. Here, training a new model from scratch for each planet ignores substantial shared structure and is thus inefficient compared to approaches that reuse knowledge acquired from other planets and adapt only a small set of task-specific parameters, such as the gravitational constant. This principle of sharing knowledge serves as the basis for several modern methods which learn mechanisms that capture shared information between tasks, enabling rapid adaptation to new ones. For instance, gradient-based meta-learners (Finn et al., 2017; Rusu et al., 2018) explicitly leverage gradients to adapt model parameters, encoding inductive biases through learned *meta* initialization. In contrast, amortization is implicitly incentivized in large language models (LLMs) by training on diverse contexts, allowing them to solve new tasks at inference by conditioning directly on prompts or observations (ICL; Brown et al., 2020; Dong et al., 2022; Lester et al., 2021; Liu et al., 2021). Learned optimizers (Andrychowicz et al., 2016; Metz et al., 2022a;b; Knyazev et al., 2024) amortize optimization itself, learning to predict parameter updates of *fixed* models conditioned on gradients.

We introduce a unified framework that encapsulates a broad spectrum of amortized learning methods − meta-learners, learned optimizers, ICL − as particular instances within a common structure, characterized by: (1) a shared mechanism to encode task-invariant inductive biases, and (2) a task adaptation function that utilizes data from novel tasks to model task-specific behavior. Our framework identifies three distinct amortization regimes distinguished by the inductive biases involved − (a) *parametric amortization* where a learned function maps task-specific data into corresponding parameters for a fixed model, *e.g.* learned optimizers (Andrychowicz et al., 2016) and hypernetworks (Ha et al., 2016), (b) *implicit amortization* where a single model jointly internalizes task-invariant mechanisms and task adaptation through forward-pass conditioning (Brown et al., 2020; Müller et al., 2021), *e.g.* ICL, and (c) *explicit amortization* which disentangles generalization and local adaptation by learning both a dataset-level embedding and a task-conditioned prediction function (Garnelo et al., 2018a; Mittal et al., 2024) via architectural inductive biases.

Each of these paradigms reflects trade-offs in expressivity, scalability, and efficiency − using gradients to provide direct access to task-specific loss landscapes, or conditioning on observations to allow richer task representations at the expense of computation, as well as going through some task-specific parameterization or bottleneck or letting the model

[†]Equal contribution [1]Meta [2]Mila [3]Université de Montréal. Correspondence to: Sarthak Mittal <sarthmit@gmail.com>.

ES-Fomo Workshop at International Conference on Machine Learning, Vancouver, Canada.



Method	$g_{\varphi}(\mathcal{D}_{\mathcal{T}}^{\text{train}})$	$\boldsymbol{\theta}_{\mathcal{T}}$	$f_{\gamma}(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{T}})$	Amortization
Standard training	SGD	weights	Fixed architecture ($\gamma = \varnothing$)	–
MAML	SGD, learned $\boldsymbol{\theta}_0$	weights	Fixed architecture ($\gamma = \varnothing$)	$\boldsymbol{\theta}_0$
Learned optimizers	Learned updates	weights	Fixed architecture ($\gamma = \varnothing$)	g_{φ}
ICL	Identity Map	Tokens	Transformer (weights γ)	f_{γ}
Parametric	Learned updates	weights	Fixed architecture ($\gamma = \varnothing$)	g_{φ}
Implicit	Subsampling	Batch $\mathcal{B}_{\mathcal{T}}^{\text{train}}$	Transformer (weights γ)	f_{γ}
Explicit	Learned updates	latents	Transformer (weights γ)	g_{φ}, f_{γ}

Table 1. **Functional decomposition of amortized learners.** We express each method in terms of g_{φ} that maps task observations $\mathcal{D}_{\mathcal{T}}^{\text{train}}$ to some representation $\boldsymbol{\theta}_{\mathcal{T}}$ – weights, prompts, dataset itself – which is fed along with query to $f_{\gamma}(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{T}})$ for prediction. Differences between methods arise from the aspects of learning they amortize, and our proposed taxonomy offers a categorization.

implicitly handle everything. To analyze and extend this, we propose an iterative amortization framework by taking inspiration from stochastic optimization. We model amortization as an iterative refinement process over stochastic mini-batches of task data, leveraging either observations directly or gradients. This mirrors the success of stochastic gradient descent in scaling optimization to large datasets, allowing us to generalize learned optimizers and hypernetworks: instead of operating solely on gradients or generating parameters one-shot, the adaptation function now incorporates streams of mini-batches, enabling greater flexibility and scalability. Our contributions include –

- *Unified Framework*: A general formulation of amortized learning that connects meta-learning, in-context learning, learned optimizers, etc. as special cases (Section 2).
- *Amortization Taxonomy*: A categorization of amortized methods into parametric, implicit, and explicit regimes, based on what is amortized and how (Section 2.1).
- *Stochastic Iterative Amortization*: A scalable approach through iterative refinement over mini-batches, overcoming limitations of scaling to large datasets (Section 3).

2. Unified Perspective

Given a task $\mathcal{T}$ and a corresponding data distribution $p_{\mathcal{T}}$, empirical risk minimization learns a model $f(\cdot, \hat{\boldsymbol{\theta}}_{\mathcal{T}})$ with parameters $\hat{\boldsymbol{\theta}}_{\mathcal{T}}$ which minimizes empirical risk. While effective for learning individual tasks, it fails to (a) adapt to new problems, and (b) leverage cross-task information when they share structure. Traditional meta-learning (Finn et al., 2017; Nichol & Schulman, 2018; Nichol et al., 2018; Rajeswaran et al., 2019) alleviates this problem by learning global initialization parameters $\hat{\boldsymbol{\theta}}_0$ compressing the shareable knowledge across tasks such that few steps of optimization leads to a good estimator for new tasks.

$$\hat{\boldsymbol{\theta}}_0 = \arg\min_{\boldsymbol{\theta}_0} \mathbb{E}_{\mathcal{T}} \mathbb{E}_{\mathbf{x},\mathbf{y},\mathcal{D}_{\mathcal{T}} \sim p_{\mathcal{T}}} \left[\mathcal{L}\left(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{T}})\right) \right], \quad (1)$$

$$\text{where} \quad \boldsymbol{\theta}_{\mathcal{T}} = g_{\boldsymbol{\theta}_0}(\mathcal{D}_{\mathcal{T}}). \quad (2)$$

For *e.g.* Model-Agnostic Meta-Learning (MAML; Finn et al., 2017) models $g_{\boldsymbol{\theta}_0}$ as an optimization routine (SGD) with learnable initialization $\boldsymbol{\theta}_0$. Alternatively, hypernetworks (Li & Liang, 2021; Ha et al., 2016; Gaier & Ha, 2019; Jia et al., 2016; Munkhdalai & Yu, 2017; Mittal et al., 2025b;a) and learned optimizers (Andrychowicz et al., 2016; Li et al., 2017; Metz et al., 2019; Wichrowska et al., 2017; Metz et al., 2022b; Li & Malik, 2017) directly model task-specific parameters as outputs of a learned process, often without a notion of "good initialization", *i.e.* $g_{\boldsymbol{\theta}_0}(\mathcal{D}_{\mathcal{T}}) \to g_{\boldsymbol{\varphi}}(\mathcal{D}_{\mathcal{T}})$ where $g_{\boldsymbol{\varphi}}$ is a learned function, *e.g.* learned optimizers model $g_{\boldsymbol{\varphi}}$ as neural networks taking only gradients as input. In-context learning or prior fitted networks instead directly model the conditional predictive distribution (Garg et al., 2022; Müller et al., 2021), *i.e.*

$$\hat{\gamma} = \arg\min_{\gamma} \mathbb{E}_{\mathcal{T}} \mathbb{E}_{\mathbf{x},\mathbf{y},\mathcal{D}_{\mathcal{T}}} \left[\mathcal{L}\Big(\mathbf{y}, f_{\gamma}(\mathbf{x}, \mathcal{D}_{\mathcal{T}})\Big) \right], \quad (3)$$

where the model does not expose explicit task parameters $\boldsymbol{\theta}_{\mathcal{T}}$ anymore, and conditioning on the task is achieved either through direct conditioning on observations $\mathcal{D}_{\mathcal{T}}$ – Prior Fitted Networks (PFNs) or ICL (Brown et al., 2020; Dong et al., 2022; Garg et al., 2022; Von Oswald et al., 2023; Hollmann et al., 2022) or its compressed natural language description when using LLMs (Mishra et al., 2021; Efrat & Levy, 2020; Sanh et al., 2021; Wei et al., 2022b;a; Min et al., 2022).

The above problems can be unified as

$$\min_{\gamma, \boldsymbol{\varphi}} \mathbb{E}_{\mathcal{T}} \mathbb{E}_{\mathbf{x},\mathbf{y},\mathcal{D}_{\mathcal{T}}} \left[\mathcal{L}\left(\mathbf{y}, f_{\gamma}\left(\mathbf{x}, g_{\boldsymbol{\varphi}}\left(\mathcal{D}_{\mathcal{T}}\right)\right)\right) \right] \quad (4)$$

where $f_{\gamma}(\mathbf{x}, g_{\boldsymbol{\varphi}}(\mathcal{D}_{\mathcal{T}}))$[1] defines an inference to obtain predictions for query $\mathbf{x}$ and corresponding set of training observations $\mathcal{D}_{\mathcal{T}}$. This general framework unifies supervised learning, meta-learning, in-context learning, and learned optimizers under a common formalism by varying how each component – f_{γ} or $g_{\boldsymbol{\varphi}}$ – is learned (**??**)[2].

[1] $g_{\boldsymbol{\varphi}}(\mathcal{T})$ describes the general form, while we look at cases where $\mathcal{D}_{\mathcal{T}}$ provides information about $\mathcal{T}$.

[2] Decomposition's non-uniqueness discussed in **??**.

			FMNIST		ImageNet	
	Steps	Lin Reg	FMNIST	MNIST	ImageNet	CIFAR100
Grad	1	51.3±0.3	63.8±1.7	79.9±1.6	88.5±0.1	92.5±1.1
	5	4.1±0.1	41.6±1.5	49.5±3.0	83.1±0.2	88.1±0.8
	10	0.5±0.0	38.1±0.4	40.7±0.8	83.4±0.1	88.1±0.6
Data	1	16.3±0.2	35.9±1.7	38.8±1.4	90.8±0.2	93.9±1.1
	5	0.5±0.0	30.8±0.3	31.3±0.5	95.2±0.1	96.3±0.6
	10	0.3±0.0	29.8±0.3	29.1±0.4	93.2±0.1	95.0±0.8
Grad + *Data*	1	25.1±0.3	37.7±1.9	41.9±1.3	96.7±0.1	97.2±0.8
	5	0.6±0.0	32.0±0.7	33.4±0.5	93.8±0.1	94.8±0.7
	10	0.4±0.0	30.5±0.2	30.3±0.3	92.5±0.1	94.1±0.4

Table 2. Our experiments on *parametric amortization* reveal benefits of multiple steps of iterative refinement across ID and OoD (gray columns) evaluation. Top row describes pre-training tasks and the second row evaluation tasks, with error metric. Sole reliance on gradients is often insufficient and leveraging observations directly can lead to improved performance with fewer iterations.

2.1. A Taxonomy of Amortization

We introduce a taxonomy categorizing amortized models into three classes: *parametric*, *explicit*, and *implicit*. We defer to Appendix C for details.

Parametric. We define amortized models with a fixed f and learnable $g_{\boldsymbol{\varphi}}$ as *parametric amortization*, which includes hypernetworks, learned optimizers, and parametric inference using ICL. Here, $g_{\boldsymbol{\varphi}}$ serves as an inference or estimation procedure discovering optimal parameters (e.g., linear coefficients) from observed data. Given a fixed parametric form, it enables us to leverage both task-specific observations $\mathcal{D}_{\mathcal{T}}$ as well as gradient information to infer the optimal parameters.

Implicit. In contrast, *implicit amortization* refers to a trainable f_γ and fixed g, which subsumes in-context learning and specific cases of prior fitted networks and conditional neural processes (Nguyen & Grover, 2022; Hollmann et al., 2022). Here, a trained model takes both the query and the dataset as input and directly models predictions – with g typically the identity mapping.

Explicit. We define *explicit amortization* with trainable f_γ shared across tasks as well as $g_{\boldsymbol{\varphi}}$, which provides a low-dimensional task-specific latents, subsuming explicit models in (Mittal et al., 2024; Elmoznino et al., 2024), neural processes (Garnelo et al., 2018b), and certain conditional neural processes (Garnelo et al., 2018a).

3. Iterative Amortized Inference

A fundamental limitation of existing approaches is their inability to leverage large-scale dataset as conditioning – they are either restricted by context length or rely solely on pooling operations or gradients. Motivated by mini-batched SGD, we extend existing approaches using an iterative refinement approach; instead of directly modeling predictions, latents or parameters, we iteratively refine them greedily using mini-batches as inputs to a trained sequence model.

		FMNIST		ImageNet	
Steps	Lin Reg	FMNIST	MNIST	ImageNet	CIFAR100
1	14.8±0.1	22.8±0.1	29.1±0.4	43.0±0.7	73.9±0.1
5	6.2±0.0	18.8±0.1	16.9±0.2	13.3±0.2	59.4±0.2
10	4.7±0.0	17.8±0.1	15.7±0.2	12.3±0.1	54.2±0.4

Table 3. For *implicit amortization*, we show consistent improvements in performance with increasing number of steps across a wide range of predictive tasks, with error as evaluation metric.

For *parametric* and *explicit* amortization, we achieve this by considering $g_{\boldsymbol{\varphi}}$ as an iterative application of a learned sequence model $h_{\boldsymbol{\varphi}}$ on different subsampled training batches $\mathcal{B}^{(i)}_{\text{train}} \subset \mathcal{D}^{\text{train}}_{\mathcal{T}}$, starting from a learned initialization $\boldsymbol{\theta}^{(0)}$. This model $h_{\boldsymbol{\varphi}}$ takes the current state $\boldsymbol{\theta}^{(i)}$ and a mini-batch $\mathcal{B}^{(i)}_{\text{train}}$ as input and returns a refined state $\boldsymbol{\theta}^{(i+1)}$ which, similar to meta-learning approaches, decreases validation loss. While learned optimizers already utilize a mini-batch approach, they only consider parametric cases with fixed f and the information about observations is fed to $h_{\boldsymbol{\varphi}}$ solely through gradients and parameter updates. At best, such methods can only recover gradient-based inference routines and cannot model more general optimization schemes (*e.g.* closed-form linear regression solution) as the mapping from observations to gradient is non-invertible.

In contrast, the *implicit* formulation does not consider task-specific parameters but instead models current prediction for query $\mathbf{x}$ as its recurrent state, *i.e.* a learned model takes a batch $\mathcal{B}^{(i)}_{\text{train}}$, query $\mathbf{x}$, and its current prediction state $\hat{\mathbf{y}}^{(i)}$ as input and refines it to $\hat{\mathbf{y}}^{(i+1)}$ for next prediction using information from the batch – this procedure is defined as r_γ and f_γ is an iterative application of r_γ. Note that while in one case $\boldsymbol{\theta}_{\mathcal{T}}$ forms the recurrent state (or memory) independent of query, in the other the recurrent state or memory is always tied to a test query $\mathbf{x}$ as well.

Motivated by SGD, we learn $\boldsymbol{\varphi}$ and γ through a greedy strategy where training is done solely on single-step improvements for all three paradigms. The respective computational graphs are provided below, where we prevent gradients to flow back from $\boldsymbol{\theta}^{(i)}$ or $\hat{\mathbf{y}}^{(i)}$

$$\boldsymbol{\theta}^{(i)} \xrightarrow{h_{\boldsymbol{\varphi}}(\cdot,\mathcal{B})} \boldsymbol{\theta}^{(i+1)} \longrightarrow \mathcal{L}(\mathbf{y}, f_\gamma(\mathbf{x}, \boldsymbol{\theta}^{(i+1)})), \tag{5}$$

$$\hat{\mathbf{y}}^{(i)} \xrightarrow{r_\gamma([\mathbf{x},\cdot],\mathcal{B})} \hat{\mathbf{y}}^{(i+1)} \longrightarrow \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}^{(i+1)}) \tag{6}$$

Here, an outer loop is used to obtain the states at the i^{th} step – $\boldsymbol{\theta}^{(i)}$ and $\hat{\mathbf{y}}^{(i)}$ respectively. Note that in this setup, $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}^{\text{valid}}_{\mathcal{T}}$ while $\mathcal{B} \subseteq \mathcal{D}^{\text{train}}_{\mathcal{T}}$, *i.e.* we are amortizing a learner to minimize validation loss. For evaluation, however, we use training and evaluation splits from completely new tasks $\mathcal{T}$ and thus test for generalization to new problems. All three frameworks – *parametric*, *explicit* and *implicit* – leverage a modification of causal masking to provide efficient and parallelizable computation of $\boldsymbol{\theta}$ and $\hat{\mathbf{y}}$ with mini-batches having

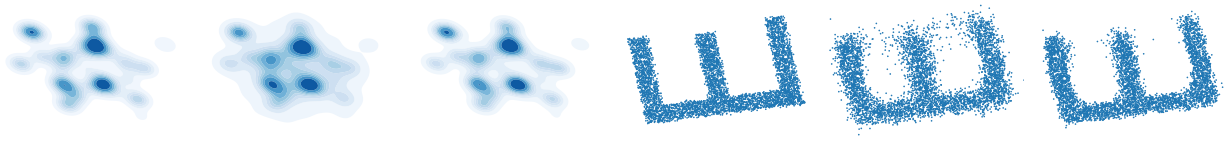


Figure 1. Samples generated from the *implicit* generative model for GMM and Alphabets task.

			MNIST		FMNIST	
	Steps	Lin Reg	MNIST	FMNIST	FMNIST	MNIST
Grad	1	55.5±0.5	90.4±0.7	90.8±0.4	88.2±0.3	89.3±0.3
	5	9.4±0.1	86.9±0.1	84.4±0.5	70.8±0.5	76.4±0.4
	10	2.9±0.0	81.4±0.5	78.9±0.9	56.4±0.7	65.7±0.4
Data	1	19.7±0.4	89.2±1.0	90.0±0.1	90.0±0.1	90.3±0.2
	5	2.6±0.0	88.7±0.0	90.0±0.0	90.0±0.0	90.2±0.0
	10	2.4±0.0	88.8±0.2	90.0±0.0	90.1±0.1	90.2±0.0
Grad + *Data*	1	24.4±0.4	72.5±0.7	62.9±0.8	51.1±1.2	61.2±0.6
	5	2.6±0.0	67.1±0.2	58.3±0.3	47.5±0.3	57.7±0.2
	10	2.1±0.0	75.1±0.1	69.5±0.1	45.1±0.3	54.4±0.1

Table 4. On *explicit amortization* we consistently see improved performance with increasing number of steps (OoD evaluation in gray columns) with error as metric. Here, we see that reliance on gradients is essential for more complex problems and in particular, explicit models struggled to scale to ImageNet and could only obtain random chance performance.

varied number of observations, detailed in Appendix H.

4. Experiments

We evaluate our proposed greedy iterative refinement approach as well as the pros and cons of using observations vs gradients through a suite of predictive and generative tasks with in-distribution (ID) and out-of-distribution (OoD) evaluation, as described in Appendix F. Learned g_φ, f_γ are transformers with max sequence length of 100 observations, detailed in Appendix H following (Kirsch et al., 2022).

Greedy Iterative Refinement. Across different forms of amortization – parametric (Tables 2 and 6), implicit (Tables 3, 5, 8 and 9 and Fig. 1) and explicit (Table 4) – we consistently see that our greedy iterative refinement framework leads to improvement in performance and allows for better handling of large-scale datasets, where k denotes the number of iterations – both for pre-training a learner and at inference. Our evaluation highlights a consistent trend of performance improvements in both generative and predictive tasks, for both OoD and ID evaluation.

Beyond Gradient Signal. When g_φ solely leverages gradient information and f has no trainable parameters, we recover the space of learned optimizers. For low dimensional problems, we see that solely learning gradient-based learners is suboptimal and the algorithm can leverage task or observations specific optimization procedure leading to substantial improvements (see Tables 2, 4 and 6). We see

	Alphabets		GMM	
Steps	ID	OoD	2-dimensional	5-dimensional
1	0.28±0.00	0.78±0.01	1.12±0.01	3.27±0.01
5	0.31±0.00	0.72±0.01	0.90±0.01	2.50±0.02
10	0.29±0.00	0.67±0.01	0.81±0.02	2.37±0.02

Table 5. We see improved sample quality (under Wasserstein metrics) using *implicit amortization* with increasing number of steps for generative modeling over OoD alphabets and new GMMs.

trend reversal in higher dimensions where the hypothesis space is larger, making it harder to infer the right parameters, and gradient provides am especially more reliable signai.

Comparing amortization methods. Our experiments indicate that *explicit* parameterization fails to learn to generalize well to new scenarios, giving chance performance on ImageNet and CIFAR, while *parametric* works are constrained in the prescribed hypothesis class, which limits expressivity. Alternatively, we see that *implicit* methods do not suffer from these issues and improve performance in the presence of other inductive biases.

Ablations. Appendix I provides additional results and comparisons to transformers without causal masking.

5. Conclusion

Amortized learning serves as a powerful paradigm for rapid generalization through reuse of computations. By introducing a unified framework and taxonomy, we clarify how different approaches internalize or externalize task adaptation. This perspective not only highlights shared principles but also exposes common bottlenecks: the limited scalability of existing methods in handling large task datasets as well as suboptimality when solely leveraging gradients with fixed functional mappings. To overcome this, we propose iterative amortization, a novel strategy that incrementally refines solutions through mini-batch updates, effectively merging the strengths of optimization-based and forward-pass approaches consistently across parametric, explicit and implicit frameworks for predictive and generative ID and OoD tasks. This framework paves the way for scalably exploring more sophisticated forms of persistent memory across mini-batches for robust task adaptation in increasingly complex, non-iid environments. Additionally, future work involves incorporating richer learning frameworks to optimize Eq. (4) beyond greedy refinement, such as evolutionary algorithms or reinforcement learning.

References

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Amos, B. et al. Tutorial on amortized optimization. *Foundations and Trends® in Machine Learning*, 16(5):592–732, 2023.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Antoniou, A., Edwards, H., and Storkey, A. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

Atanackovic, L., Zhang, X., Amos, B., Blanchette, M., Lee, L. J., Bengio, Y., Tong, A., and Neklyudov, K. Meta flow matching: Integrating vector fields on the wasserstein manifold. *arXiv preprint arXiv:2408.14608*, 2024.

Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36:57125–57211, 2023.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chang, P. E., Loka, N., Huang, D., Remes, U., Kaski, S., and Acerbi, L. Amortized probabilistic conditioning for optimization, simulation and inference. *arXiv preprint arXiv:2410.15320*, 2024.

Chauhan, V. K., Zhou, J., Lu, P., Molaei, S., and Clifton, D. A. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57(9):250, 2024.

Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Dalal, K., Koceja, D., Hussein, G., Xu, J., Zhao, Y., Song, Y., Han, S., Cheung, K. C., Kautz, J., Guestrin, C., et al. One-minute video generation with test-time training. *arXiv preprint arXiv:2504.05298*, 2025.

Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Liu, T., et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Efrat, A. and Levy, O. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*, 2020.

Elmoznino, E., Marty, T., Kasetty, T., Gagnon, L., Mittal, S., Fathi, M., Sridhar, D., and Lajoie, G. In-context learning and occam's razor. *arXiv preprint arXiv:2410.14086*, 2024.

Erdos, P. and Renyi, A. On random graphs i. *Publ. math. debrecen*, 6(290-297):18, 1959.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Gaier, A. and Ha, D. Weight agnostic neural networks. *Advances in neural information processing systems*, 32, 2019.

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

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. Conditional neural processes. In *International conference on machine learning*, pp. 1704–1713. PMLR, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Jia, X., De Brabandere, B., Tuytelaars, T., and Gool, L. V. Dynamic filter networks. *Advances in neural information processing systems*, 29, 2016.

Kingma, D. P., Welling, M., et al. Auto-encoding variational bayes, 2013.

Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

Knyazev, B., Moudgil, A., Lajoie, G., Belilovsky, E., and Lacoste-Julien, S. Accelerating training with neuron interaction and nowcasting networks. *arXiv preprint arXiv:2409.04434*, 2024.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Li, K. and Malik, J. Learning to optimize neural nets. *arXiv preprint arXiv:1703.00441*, 2017.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

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*, pp. 19565–19594. PMLR, 2023.

Li, Z., Zhou, F., Chen, F., and Li, H. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liu, X., Ji, K., Fu, Y., Tam, W. L., Du, Z., Yang, Z., and Tang, J. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

Lorch, L., Sussex, S., Rothfuss, J., Krause, A., and Schölkopf, B. Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35:13104–13118, 2022.

Metz, L., Maheswaranathan, N., Nixon, J., Freeman, D., and Sohl-Dickstein, J. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pp. 4556–4565. PMLR, 2019.

Metz, L., Freeman, C. D., Harrison, J., Maheswaranathan, N., and Sohl-Dickstein, J. Practical tradeoffs between memory, compute, and performance in learned optimizers. In *Conference on Lifelong Learning Agents*, pp. 142–164. PMLR, 2022a.

Metz, L., Harrison, J., Freeman, C. D., Merchant, A., Beyer, L., Bradbury, J., Agrawal, N., Poole, B., Mordatch, I., Roberts, A., et al. Velo: Training versatile learned optimizers by scaling up. *arXiv preprint arXiv:2211.09760*, 2022b.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

Mishra, S., Khashabi, D., Baral, C., Choi, Y., and Hajishirzi, H. Reframing instructional prompts to gptk's language. *arXiv preprint arXiv:2109.07830*, 2021.

Mittal, S., Elmoznino, E., Gagnon, L., Bhardwaj, S., Sridhar, D., and Lajoie, G. Does learning the right latent variables necessarily improve in-context learning? *arXiv preprint arXiv:2405.19162*, 2024.

Mittal, S., Bengio, Y., Malkin, N., and Lajoie, G. In-context parametric inference: Point or distribution estimators? *arXiv preprint arXiv:2502.11617*, 2025a.

Mittal, S., Bracher, N. L., Lajoie, G., Jaini, P., and Brubaker, M. Amortized in-context bayesian posterior estimation. *arXiv preprint arXiv:2502.06601*, 2025b.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

Munkhdalai, T. and Yu, H. Meta networks. In *International conference on machine learning*, pp. 2554–2563. PMLR, 2017.

Nguyen, T. and Grover, A. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. *arXiv preprint arXiv:2207.04179*, 2022.

Nichani, E., Damian, A., and Lee, J. D. How transformers learn causal structure with gradient descent. *arXiv preprint arXiv:2402.14735*, 2024.

Nichol, A. and Schulman, J. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

Reuter, A., Rudner, T. G., Fortuin, V., and Rügamer, D. Can transformers learn full bayesian inference in context? *arXiv preprint arXiv:2501.16825*, 2025.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Scetbon, M., Jennings, J., Hilmkil, A., Zhang, C., and Ma, C. A fixed-point approach for causal generative modeling. *arXiv preprint arXiv:2404.06969*, 2024.

Schug, S., Kobayashi, S., Akram, Y., Sacramento, J., and Pascanu, R. Attention as a hypernetwork. *arXiv preprint arXiv:2406.05816*, 2024.

Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., and Bengio, Y. Conditional flow matching: Simulation-free dynamic optimal transport. *arXiv preprint arXiv:2302.00482*, 2(3), 2023.

Von Oswald, J., Henning, C., Grewe, B. F., and Sacramento, J. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

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*, pp. 35151–35174. PMLR, 2023.

Watts, D. J. and Strogatz, S. H. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

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. *Advances in neural information processing systems*, 35:24824–24837, 2022b.

Wichrowska, O., Maheswaranathan, N., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Freitas, N., and Sohl-Dickstein, J. Learned optimizers that scale and generalize. In *International conference on machine learning*, pp. 3751–3760. PMLR, 2017.

A. Uniqueness of Decomposition

Importantly, the functional form $f_\gamma(\mathbf{x}, g_{\boldsymbol{\varphi}}(\mathcal{D}_\mathcal{T}))$ does not define a unique way to split the computation between f_γ and $g_{\boldsymbol{\varphi}}$. In principle, any computation performed by $g_{\boldsymbol{\varphi}}$, along with its parameters, can be folded into f_γ, making the decomposition arbitrary. However, this separation becomes meaningful from an *algorithmic perspective* with the following convention: assign all computations independent of the query $\mathbf{x}$ to $g_{\boldsymbol{\varphi}}$, and leave the query-dependent computations to f_γ. In this view, f_γ processes the query while accessing task-specific information $\mathcal{D}_\mathcal{T}$ only through the structure or constraints imposed by $g_{\boldsymbol{\varphi}}$, for example, pooled representations, gradient information, or natural language.

B. Specific Cases of Amortization

Gives a set of tasks $\mathcal{T}$ along with their corresponding training $\mathcal{D}_\mathcal{T}^{\text{train}}$ and validation $\mathcal{D}_\mathcal{T}^{\text{valid}}$ set, the empirical counterpart of our proposed framework can be described as

$$\min_{\gamma,\boldsymbol{\varphi}} \mathbb{E}_\mathcal{T} \left[\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_\mathcal{T}^{\text{valid}}} \mathcal{L}\left(\mathbf{y}, f_\gamma\left(\mathbf{x}, \boldsymbol{\theta}_\mathcal{T}\right)\right) \right], \quad \text{where } \boldsymbol{\theta}_\mathcal{T} = g_{\boldsymbol{\varphi}}(\mathcal{D}_\mathcal{T}^{\text{train}}) \tag{7}$$

in which $f_\gamma(\cdot,\ \boldsymbol{\theta}_\mathcal{T})$ denotes a model with shared parameters γ and task-specific parameters $\boldsymbol{\theta}_\mathcal{T}$ (*e.g.* weights, soft prompts, latent states, or context tokens). The function $g_{\boldsymbol{\varphi}}$, parameterized by $\boldsymbol{\varphi}$, represents the inner optimization mechanism that maps the support set $\mathcal{D}_\mathcal{T}^{\text{train}}$ for task $\mathcal{T}$ to $\boldsymbol{\theta}_\mathcal{T}$. The loss $\mathcal{L}$ is then computed on the query set $\mathcal{D}_\mathcal{T}^{\text{valid}}$. By selecting different forms for f, partitioning parameters between $\boldsymbol{\varphi}$ and γ, and varying the adaptation function $g_{\boldsymbol{\varphi}}$, this framework subsumes a wide spectrum of learning paradigms, as outlined in Table 1 and Appendix D. In particular, we obtain −

Gradient-based supervised learning when f is fixed − *e.g.* a neural network architecture − and g an optimization procedure, like stochastic gradient descent, solving **??**. Alternatively, we obtain the same family when f_γ is learned through an optimization procedure with g as identity, thus reiterating that the decomposition is not unique but the former is preferred as it pushes more operations to g.

MAML when f is fixed and $g_{\boldsymbol{\varphi}}$ an optimization procedure with learnable initialization $\boldsymbol{\theta}_0 \in \boldsymbol{\varphi}$.

Learned Optimizers / Hypernetworks when f is fixed and $g_{\boldsymbol{\varphi}}$ a deep learning model with parameters $\boldsymbol{\varphi}$. We obtain hypernetworks when g_φ maps observations to parameters, and learned optimizers when $g_{\boldsymbol{\varphi}}$ is an iterative procedure relying solely on gradients, *i.e.* $g_{\boldsymbol{\varphi}} := h_{\boldsymbol{\varphi}}^{(k)} \circ \ldots \circ h_{\boldsymbol{\varphi}}^{(1)}$ defines a k-step $g_{\boldsymbol{\varphi}}$ with $h_{\boldsymbol{\varphi}}^{(i)}$ a sequence model taking $\boldsymbol{\theta}^{(0)}, \ldots, \boldsymbol{\theta}^{(i-1)}$ and their gradients $\nabla^{(j)} := \nabla_{\boldsymbol{\theta}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{B}^{(j)}} \mathcal{L}(\mathbf{y}, f(\mathbf{x},\boldsymbol{\theta}))\big|_{\boldsymbol{\theta}^{(j)}}$ as input, where $\boldsymbol{\theta}^{(0)}$ can be learned, $\boldsymbol{\theta}_\mathcal{T} = \boldsymbol{\theta}^{(k)}$ and $\mathcal{B}^{(j)} \subseteq \mathcal{D}_\mathcal{T}$.[3]

In-Context Learning when g samples a set of observations and f_γ is modeled as a predictive sequence model with the context as observations. If g describes $\mathcal{T}$ in natural language, then we recover the prompt-based ICL − an emergent phenomena from pretraining of LLMs.

Learning the Learner. So far, we only focused on the modeling assumptions − the form of $f_\gamma(\mathbf{x}, g_{\boldsymbol{\varphi}}(\mathcal{D}_\mathcal{T}))$ − behind different amortized learners did not address how the optimization problem in Eq. (7) is actually performed. The main complexity comes from iterative natures of f_γ and $g_{\boldsymbol{\varphi}}$ and the presence of higher order gradients if $g_{\boldsymbol{\varphi}}$ describes a gradient-based procedure. Multiple approaches tackle this problem through first-order approximations (Finn et al., 2017; Nichol & Schulman, 2018), back-propagation through time (Ha et al., 2016), reinforcement learning (Andrychowicz et al., 2016) or evolutionary strategies (Metz et al., 2022b) depending on the form of $g_{\boldsymbol{\varphi}}$.

C. Taxonomy of Amortization

We introduce a taxonomy categorizing amortized models into three classes: *parametric*, *explicit*, and *implicit*.

Parametric. We define the class of amortized models with a fixed f and learnable $g_{\boldsymbol{\varphi}}$ as *parametric amortization*. This includes hypernetworks, learned optimizers for some given architecture, and other approaches to parametric inference using ICL. In this framework, the functional form of the likelihood is fixed, for example a known simulator (Cranmer et al., 2020) or a categorical likelihood with linear mapping. Here, $g_{\boldsymbol{\varphi}}$ serves as an inference or estimation procedure that discovers the optimal parameters (e.g., linear coefficients) at inference time from the observed data. Several works (Mittal et al., 2025b; Reuter et al., 2025; Chang et al., 2024) approximate the posterior distribution over parameters of the likelihood as

[3] A system is Markovian if it only relies on the current $(\boldsymbol{\theta}^{(j)}, \nabla^{(j)})$ or non Markovian if on more past states.

g_{φ} while others (Mittal et al., 2025a) investigate the interplay between point-based and distribution-based modeling of g_{φ}[4]. By relying on a fixed parametric assumption, this approach enables us to leverage task-specific observations $\mathcal{D}_{\mathcal{T}}$ as well as gradient information to infer the optimal parameters of f. These inferred parameters can also offer a low-dimensional representation of the task and provide interpretability benefits, especially when f possesses specific, interpretable structure.

Implicit. In contrast, we refer to amortized models with a trainable f_{γ} and a fixed g as *implicit amortization*, which subsumes in-context learning and specific cases of prior fitted networks and conditional neural processes (Nguyen & Grover, 2022; Hollmann et al., 2022). In this setting, a single trained model takes both the query and the set of observations as input and directly learns to model predictions. The function g is typically either the identity mapping or a subsampling mechanism used to manage large datasets. Unlike parametric approaches, the functional form of the output is learned directly, bypassing explicitly inferring any task-specific parameters. It is non-trivial to pinpoint what part of network activations correspond to dataset-specific vs prediction based activations. Some works (Von Oswald et al., 2023; Nichani et al., 2024) demonstrate that, under certain assumptions, implicit models recover algorithmic behaviors such as gradient descent or causal discovery. However, it remains unclear how these findings generalize beyond specific setups, apart from the broader perspective of learning the posterior predictive distribution (Müller et al., 2021; Garg et al., 2022).

Explicit. In theory, it is possible to leverage the dimensionality reduction benefits offered by the parametric approach while still utilizing a learnable form for the likelihood which the implicit model affords. This is accomplished with a trainable f_{γ}, which learns the likelihood form shared across tasks like the laws of physics, as well as g_{φ}, which provides a low-dimensional encapsulation of the entire dataset like the gravitational constant. The explicit models considered in (Mittal et al., 2024; Elmoznino et al., 2024), neural processes (Garnelo et al., 2018b), and certain conditional neural processes that first learn an embedding of the dataset (Garnelo et al., 2018a) can be seen as specific cases of this approach. Similar to parametric methods, it allows leveraging additional gradient information and provides dimensionality reduction and interpretability benefits; but at the cost of inherent non-stationarity during learning $-$ f_{γ} and g_{φ} are dependent on each other but have to be learned together.

D. Related Work

In this section, we formalize and contrast several existing related frameworks encapsulated in our setting. We highlight the technical differences among these approaches through formal definitions and equations, while retaining key references.

D.1. Meta-Learning

Meta-learning generally refers to the problem of learning a model (which itself could be a learner, optimizer, or algorithm) that can quickly and efficiently generalize to novel tasks. Formally, let there be a distribution over tasks $p(\mathcal{T})$, with each task $\mathcal{T}$ associated with a dataset $\mathcal{D}_{\mathcal{T}} = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{N}$. The objective is to learn meta-parameters $\boldsymbol{\theta}$ that enable fast adaptation to new tasks $\mathcal{T}'$ with dataset $\mathcal{D}_{\mathcal{T}'}$:

$$\min_{\theta} \mathbb{E}_{\mathcal{T}} \mathcal{L}\left(\mathcal{D}_{\mathcal{T}}^{\text{valid}}, U_k(\boldsymbol{\theta}, \mathcal{D}_{\mathcal{T}}^{\text{train}})\right),$$

where:

- $U_k(\boldsymbol{\theta}, \mathcal{D}_{\mathcal{T}}^{\text{train}})$ is an adaptation operator applying k gradient steps or another procedure starting from $\boldsymbol{\theta}$,
- $\mathcal{L}\left(\mathcal{D}_{\mathcal{T}}^{\text{valid}}, \cdot\right)$ is the loss on held-out data $\mathcal{D}_{\mathcal{T}}^{\text{valid}}$.

The popular Model-Agnostic Meta-Learning (MAML) framework (Finn et al., 2017) fits into this paradigm with the update:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\mathcal{D}_{\mathcal{T}}^{\text{train}}, \cdot\right)$$

where the meta-objective optimizes $\boldsymbol{\theta}$ such that $\boldsymbol{\theta}'$ performs well on $\mathcal{D}_{\mathcal{T}}^{\text{valid}}$. Extensions such as Reptile (Nichol & Schulman, 2018) and Meta-SGD (Antoniou et al., 2018) also follow this principle. These methods ultimately fit within the *parametric modeling* framework, as the model parameters $\boldsymbol{\theta}$ are explicitly adapted to new tasks (Raghu et al., 2019). Here, the adaptation operator U_k can be seen as the analogue of g_{φ} from our setting.

[4] g_{φ} can be probability distributions or sampling operations.

D.2. Amortization

Amortized inference traditionally refers to learning an inference network $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the posterior $p(\mathbf{z}|\mathbf{x})$, as in variational autoencoders (Kingma et al., 2013; Rezende et al., 2014). Our focus differs by considering amortization at the dataset or task level, rather than per-observation. We formalize this as learning an amortized posterior:

$$q_\phi(\mathbf{z} \mid \mathcal{D}) \approx p(\mathbf{z} \mid \mathcal{D}),$$

where $\mathcal{D}$ is a dataset corresponding to a task. This approach implicitly or explicitly learns an optimizer or inference model that can solve novel tasks *zero-shot*. This viewpoint aligns with probabilistic meta-learning frameworks (Garnelo et al., 2018b; Amos et al., 2023), which emphasize amortizing inference across a distribution of tasks. Such methods can be framed as parametric approaches when the target $\mathbf{z}$ of interest are all the parameters of the likelihood (Mittal et al., 2025b) or explicit approaches when the likelihood is learned as well and $\mathbf{z}$ just represents some latents (Garnelo et al., 2018a;b).

In more general terms, all methods of meta-learning rely on some or the other notion of amortization and depending on the object that is trained to be amortized, we recover parametric, explicit or implicit models.

D.3. Learned Optimizers

Learned optimizers learn parameter update functions $h_{\boldsymbol{\varphi}}$ conditioned on gradients and optimization states:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + h_{\boldsymbol{\varphi}}(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{B}_\mathcal{T}, \boldsymbol{\theta}), \mathbf{s}_t),$$

where $\mathbf{s}_t$ denotes the optimizer's internal state, e.g., momentum or recurrent memory and $\mathcal{B}_\mathcal{T}$ represents a mini-batch of task-specific training data. These methods have been studied extensively (Metz et al., 2022b; Knyazev et al., 2024; Metz et al., 2019; 2022a; Li & Malik, 2017; Wichrowska et al., 2017).

While learned optimizers enable scalable amortization of inference, their hypothesis class is limited by dependence on *only first-order gradient information* and parameter updates. They cannot easily represent complex second-order dynamics such as Newton methods, which rely on Hessian information. Furthermore, they require explicit task parameters $\boldsymbol{\theta}$, restricting them to parametric models and preventing direct applicability to implicit modeling approaches. Here, one can see the recursive application of $h_{\boldsymbol{\varphi}}$ as the $g_{\boldsymbol{\varphi}}$.

D.4. Hypernetworks

Hypernetworks (Ha et al., 2016) are architectures that take a dataset $\mathcal{D}$ as input and generate the weights $\boldsymbol{\theta}$ of a target neural network to be used for predictions on that dataset:

$$\boldsymbol{\theta} = H_{\boldsymbol{\varphi}}(\mathcal{D})$$

with the target model output

$$\mathbf{y} = f(\mathbf{x}, \boldsymbol{\theta}) = f(\mathbf{x}, H_{\boldsymbol{\varphi}}(\mathcal{D}))$$

Such models have been studied extensively (Krueger et al., 2017; Chauhan et al., 2024; Schug et al., 2024; Von Oswald et al., 2019), including recent work that shows standard transformers can be interpreted as hypernetworks (Schug et al., 2024). Hypernetworks are generally *parametric* since $\boldsymbol{\theta}$ is explicitly generated. However, if the hypernetwork conditions directly on the query $\mathbf{x}$, it can realize implicit models within our taxonomy. The general case of hypernetworks can, however, be seen as parametric modeling with $H_{\boldsymbol{\varphi}}$ being the $g_{\boldsymbol{\varphi}}$ component.

D.5. In-Context Learning and Prior-Fitted Networks

In-context learning (ICL) solves similar problems by training models (notably transformers) to make predictions conditioned on example input-output pairs provided as context, either by modeling explicit parameters akin to hypernetworks (Mittal

et al., 2025b;a) or without explicit parameter updates (Mittal et al., 2024; Elmoznino et al., 2024; Müller et al., 2021; Garg et al., 2022; Von Oswald et al., 2023; Hollmann et al., 2022; Li et al., 2023; Bai et al., 2023).

Formally, given a context $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^K$, the model learns to produce predictions $f_\gamma(\mathbf{x}, \mathcal{D})$ with no parameter updates performed during inference. ICL and PFNs share the same modeling setup when the input is a sequence of examples, though ICL can be more general by conditioning on other forms of task information such as language prompts.

Both are subsumed within our unified framework, categorized as *parametric*, *explicit*, or *implicit* models depending on how the in-context learner is parameterized.

E. Tasks

We describe the set of tasks considered for our setup, with additional information about how the tasks are structured being differed to Appendix F.

Linear Regression. Each task $\mathcal{T}$ is defined by a ground-truth weight $\mathbf{w}_\mathcal{T}$ which specifies the data-generating distribution $p_\mathcal{T}(\mathbf{y}|\mathbf{x})$, where we consider problems with $\mathbf{x}, \mathbf{w}_\mathcal{T} \in \mathbb{R}^{100}$.

MNIST / FashionMNIST Classification. Following (Kirsch et al., 2022), each task $\mathcal{T}$ involves a random projection $W_\mathcal{T} \in \mathbb{R}^{784\times100}$ applied to image pixels and a random label mapping (*e.g.* digits 4 are mapped to label 7; even digits grouped together, etc.) which preserves the semantic structure. We train on MNIST tasks and evaluate on context and query from FashionMNIST, and vice versa.

ImageNet Classification. Each task $\mathcal{T}$ involves sampling images from 100 different ImageNet classes and randomly regrouping them into a maximum 100-way classification task, preserving semantic structure similar to above. Evaluation is done on ImageNet validation data with support sets from training data provided in-context. We also conduct OoD evaluation on CIFAR-10 and CIFAR-100. The images are fed to the transformer networks after obtaining embeddings from Dino-v2 (Oquab et al., 2023).

Topological Order Prediction. Following (Scetbon et al., 2024), each task involves sampling a structural causal model (SCM) and inferring its topological order using only observational data.

GMM. Each task is defined by a mixture of Gaussians with the number of mixture components and corresponding means sampled randomly per task. We apply the flow-matching framework (Lipman et al., 2022; Tong et al., 2023; Albergo et al., 2023) to learn the conditional vector field given task data $\mathcal{D}_\mathcal{T}$, and evaluate samples generated conditioned on novel densities as context unseen during training.

Alphabets. We use the alphabets dataset (Atanackovic et al., 2024) where the context describes the alphabet and its scale and rotation. The task is to leverage the conditioning information to draw samples from the underlying density, with OoD settings corresponding to alphabets unseen during training.

F. Dataset Details

In this section, we provide details of all the datasets and tasks considered for pre-training the different variants of amortized estimators as well as the tasks used for evaluation.

F.1. Linear Regression

For the problem of linear regression, we define each task with a corresponding weight vector $\mathbf{w}_\mathcal{T}$ such that observations in the context and query set follow the mapping $(\mathbf{x}, \mathbf{y} = \mathbf{w}_\mathcal{T}^T\mathbf{x} + \epsilon)$ where $\mathbf{x}$ is sampled from a standard normal distribution and ϵ denotes random noise sampled from a normal distribution with standard deviation of 0.25. For this problem, we define the family of tasks for pre-training and evaluation corresponding to $\mathbf{w}_\mathcal{T}$ sampled randomly from the unit normal.

For training we pass a set of $\{(\mathbf{x}_i, \mathbf{y}_i)\}_i$ as the context and a set of $\mathbf{x}$ as the query, where the underlying mapping between them is shared to be the same $\mathbf{w}_\mathcal{T}$. The task of the in-context learner is to predict either the coefficients $\boldsymbol{\theta}_\mathcal{T}$ that are fed to a prediction network or known mapping (*explicit* or *parametric*), or it predicts the prediction corresponding to $\mathbf{x}$ directly. Thus, the in-context learner has to learn to model and minimize the underlying validation loss corresponding to the problem.

F.2. MNIST / FashionMNIST Classification

Similar to linear regression, we randomly sample a projection matrix $W_{\mathcal{T}} \in \mathbb{R}^{784\times 100}$ from unit normal and a class re-labeling matrix $\pi_{\mathcal{T}}$ such that $\pi_{\mathcal{T},i,j} \in \{0,1\}$ and $\sum \pi_{\mathcal{T},i,:} = 1$ such that each task is defined with the following transformations on observations and labels:

$$\mathbf{x} \to W_{\mathcal{T}}\mathbf{x} \quad \mathbf{y} \to \pi_{\mathcal{T}}\mathbf{y} \tag{8}$$

where $\mathbf{x} \in \mathbb{R}^{784}$ and $\mathbf{y}$ denotes the one-hot vector corresponding to the class. This operation defined above can be seen as randomly projecting pixel values to a lower dimensional space and re-labeling the classification problem such that the same classes are mapped to the same new class $-$ for *e.g.* the remapping could map digits $1, 4, 7$ to class 1, and so on. That is, this re-labeling can club multiple classes into a single class or rename class labels, but never sends two objects from the same class to different labels.

We follow the same setup for FashionMNIST as well, and then check for OoD evaluation on the other dataset by feeding a set of observations as context and evaluating on queries; note that task specific variables $W_{\mathcal{T}}$, $\pi_{\mathcal{T}}$ are shared between context and query.

F.3. ImageNet Classification

We follow the same setup as MNIST / FashionMNIST classification with just a few differences $-$ dimensionality reduction is done using a pre-trained Dino-v2 model and thus this projection operator is shared across tasks and not task-dependent anymore, and each task is defined by sampling a subset of images corresponding to 100 random classes of ImageNet from which grouping / re-labeling is done exactly as above. That is, for any task the problem is to solve the maximum 100-way classification problem as opposed to the single 1000-way problem of ImageNet. Correspondingly, evaluation is done on contexts from ImageNet training set and queries from evaluation set corresponding to multiple 100-way problems as well as CIFAR-10 and CIFAR-100's train and test set forming the context and query.

F.4. Topological Order Prediction

Problem Statement. We begin by defining *Structural Causal Models (SCMs)*, which formalize the causal generative process over a set of random variables. An SCM defines a distribution over of d endogenous variables $\boldsymbol{V} = \{X_1, \ldots, X_d\} \sim \mathbb{P}_X$, each determined by a deterministic function of its parents (F_i) and a corresponding exogenous noise term (N_i). Specifically, each variable X_i is generated as:

$$X_i = F_i(\text{Pa}(X_i), N_i), \quad \text{with} \quad \text{Pa}(X_i) \subseteq \boldsymbol{V} \setminus \{X_i\}$$

where the functions F_i define the causal mechanisms, $\text{Pa}(X_i)$ denotes the set of direct causes (parents) of X_i, and N_i is a latent noise variable drawn independently from a distribution $\mathbb{P}_N$. Collectively, the SCM is represented as $\mathcal{S}(\mathbb{P}_N, F, \mathcal{G})$, where $\mathcal{G} \in \{0,1\}^{d\times d}$ is the adjacency matrix of the causal graph, i.e., $\mathcal{G}_{ij} = 1$ if $X_j \in \text{Pa}(X_i)$.

Following standard assumptions in the literature, we consider markovian SCMS, where we restrict the causal graph $\mathcal{G}$ to be a directed acyclic graph (DAG) and the set of noise variables $\{N_i, \cdots, N_d\}$ to be mutually independent.

Given the DAG assumption, we can obtain a unique topological order τ associated with the causal graph $\mathcal{G}$, and the prediction task is defined as follows.

> Given a dataset of causal variables $D_X \in \mathbb{R}^{n\times d}$ samples from an unknown SCM $\mathcal{S}(\mathbb{P}_N, F, \mathcal{G})$, the goal is predict the associated topological order τ from D_X.

Method. For amortized inference of topological order, we follow the approach from Scetbon et al. (2024). They leverage transformer-based architecture that attend over both the sample dimension (n) and the node dimension (d) to attend over the context, followed by a linear prediction layer to classify the leaf nodes (no outgoing edge) of the causal graph $\mathcal{G}$. Once we obtain the current leaf nodes, we can remove them from the dataset D_X and repeat the procedure again to predict leaf nodes. This recursive procedure will terminate in at most d iteration and output the inferred topological order $\hat{\tau}$.

Synthetic Data Simulator. Following (Scetbon et al., 2024), we use the AVICI synthetic data simulator (Lorch et al., 2022). This synthetic generator supports diverse structural and functional variations, making it particularly well-suited for

training and evaluating models under distribution shifts. We describe below the options avaiable for each compoenent in the SCM.

- *Graph Structures.* We can sample causal graphs from a variety of schemes; Erdős–Rényi graphs (Erdos & Renyi, 1959), Scale-free networks (Barabási & Albert, 1999), Watts–Strogatz small-world graphs (Watts & Strogatz, 1998), and Stochastic block models (Holland et al., 1983).
- *Noise Distributions.* Exogenous noise variables $\{N_i\}$ are sampled from either Gaussian or Laplace distributions, with randomly selected variances.
- *Functional Relationships.* Causal relationships are instantiated using either Linear (LIN) models with randomly sampled weights and biases, or with Random Fourier Features (RFF) to generate more complex, non-linear mappings.

To simulate distribution shifts, two families of SCM distributions are defined:

- *In-Distribution* ($\mathbb{P}_{\text{in}}$): Graphs are sampled from Erdős–Rényi and scale-free models, noise from Gaussian distributions, and functions are either LIN or RFF using standard parameter ranges.
- *Out-of-Distribution* ($\mathbb{P}_{\text{out}}$): Graphs are drawn from Watts–Strogatz or stochastic block models, noise from Laplace distributions, and functions (LIN/RFF) are sampled from disjoint parameter ranges to introduce a shift.

Parameter Ranges:

- Linear Mechanisms:
  - $\mathbb{P}_{\text{in}}$: weights $\sim U_{\pm}(1, 3)$, bias $\sim U(-3, 3)$
  - $\mathbb{P}_{\text{out}}$: weights $\sim U_{\pm}(0.5, 2) \cup U_{\pm}(2, 4)$, same bias range
- RFF Mechanisms:
  - $\mathbb{P}_{\text{in}}$: length scale $\sim U(7, 10)$, output scale $\sim U(5, 8) \cup U(8, 12)$, bias $\sim U_{\pm}(-3, 3)$
  - $\mathbb{P}_{\text{out}}$: length scale $\sim U(10, 20)$, output scale $\sim U(8, 12) \cup U(18, 22)$, bias $\sim U_{\pm}(-3, 3)$

Train Datasets. We train the model by randomly sampling SCMs from the $\mathbb{P}_{\text{in}}$ distribution. Each epoch contains datasets with $n = 1000$ samples and $d = 20$ nodes.

Test Datasets. We evaluate performance across four distinct settings that differ in the distribution they induce over SCMs. From each SCM, we sample a two test datasets, one with $n = 1000$ samples and $d = 20$ nodes; and the with $n = 1000$ samples and $d = 50$ nodes.

- `LIN`: Linear mechanisms under $\mathbb{P}_{\text{in}}$; total of 9 randomly sampled SCMs with 3 different graph types.
- `RIN`: Nonlinear (RFF) mechanisms under $\mathbb{P}_{\text{in}}$; total of 9 randomly sampled SCMs with 3 different graph types.
- `LOUT`: Linear mechanisms under $\mathbb{P}_{\text{out}}$; total of 6 randomly sampled SCMs with 2 different graph types.
- `ROUT`: Nonlinear (RFF) mechanisms under $\mathbb{P}_{\text{out}}$; total of 6 randomly sampled SCMs with 2 different graph types.

F.5. Alphabets

We next turn our attention to generation tasks where the goal is to generate novel samples by inferring the underlying density defined by the context examples and consequently learning to sample from it. We first borrow the task of sampling point clouds resembling alphabets (Atanackovic et al., 2024) where different tasks $\mathcal{T}$ correspond to different alphabets with different scaling and rotation operations applied. Out of all the capitalized alphabets in the English language, we reserve {D,H,N,T,X,Y} solely for evaluation, *i.e.* they are not provided for training on during pre-training. We then consider evaluation on both in-distribution and out-of-distribution alphabets.

F.6. Gaussian Mixture Model

Similar to the above, we consider the problem of sampling similar to a mixture of Gaussians provided as in-context examples. Here, each task is defined by an underlying number of clusters $n_{\mathcal{T}}$ as well as corresponding means $\mu_{\mathcal{T},i}$ which are sampled from a normal distribution with 5 standard deviation. Samples from this GMM are obtained by fixing the standard deviation corresponding to each cluster as 0.3 and then drawing samples from this mixture distribution. We train the in-context estimator on a constant stream of new tasks synthetically generated and then evaluate them on new tasks sampled randomly. Our tasks involve both a simple version of GMM which is in 2-dimensional observed space as well as a more complex GMM which is 5-dimensional. The number of clusters are uniformly sampled from a maximum of 100 clusters.

G. Metrics

For metrics, we consider the l_2 loss for the regression problem while for the classification problems, we consider error as a metric, which is $100-$Accuracy, which we use for the topological prediction task as well. For the generative modeling tasks, we consider the 2-Wasserstein and 1-Wasserstein distance which are based on optimal transport and can be described as

$$\mathcal{W}_p = (\inf_{\pi} \frac{1}{N} \sum_i \|\mathbf{x}_i - \hat{\mathbf{x}}_{\pi_i}\|_p^p)^{1/p} \tag{9}$$

where π describes a permutation matrix, of which we use $p = 1$ or $p = 2$.

H. Experimental Details

We first describe the general motivation, benefits and how we instantiate iterative amortized inference, following which we discuss how amortized f_γ and $g_{\boldsymbol{\varphi}}$ are modeled as (Transformers) and the design choices associated with each. We then look at the training objectives leveraged as well as the specification of causal or non causal method.

H.1. Iterative Amortized Inference

A fundamental limitation of the existing approaches is their inability to leverage large-scale dataset as conditioning $-$ they are either restricted by context length or rely solely on low-dimensional pooling operations or gradients which can be quite restrictive. Analogously, non-amortized learners like gradient descent resolve this scalability issue through a stochastic mini-batch framework which iteratively refines an approximate solution based on a subset of observations, not the whole dataset. Note that this iterative refinement is Markovian[5] and greedy, *i.e.* the update given current state is independent of history and leads to immediate single-step improvement which could be sub-optimal over multiple steps.

Taking inspiration from SGD and the connection between amortized meta-learners and optimization routines (Elmoznino et al., 2024), we extend existing approaches using an iterative refinement approach; instead of directly modeling predictions, embeddings or parameters, we iteratively refine the output from the previous step greedily using mini-batches as input to a trained sequence model.

For *parametric* and *explicit* amortization, we achieve this by considering $g_{\boldsymbol{\varphi}}$ as an iterative application of a learned sequence model $h_{\boldsymbol{\varphi}}$ on different subsampled training batches $\mathcal{B}^{(i)}_{\text{train}} \subset \mathcal{D}^{\text{train}}_{\mathcal{T}}$, starting from a learned initialization $\boldsymbol{\theta}^{(0)}$.[6] In our experiments, we use a Transformer though the specific architecture can change. This model $h_{\boldsymbol{\varphi}}$ takes the current state $\boldsymbol{\theta}^{(i)}$ and a mini-batch $\mathcal{B}^{(i)}_{\text{train}}$ as input and returns a refined state $\boldsymbol{\theta}^{(i+1)}$ which, similar to meta-learning approaches, decreases validation loss.

$$\boldsymbol{\theta}^{(0)} \xrightarrow{h_{\boldsymbol{\varphi}}\left(\cdot, \mathcal{B}^{(0)}_{\text{train}}\right)} \boldsymbol{\theta}^{(1)} \xrightarrow{h_{\boldsymbol{\varphi}}\left(\cdot, \mathcal{B}^{(1)}_{\text{train}}\right)} \dots \xrightarrow{h_{\boldsymbol{\varphi}}\left(\cdot, \mathcal{B}^{(k-1)}_{\text{train}}\right)} \boldsymbol{\theta}^{(k)} =: \boldsymbol{\theta}_{\mathcal{T}} \tag{10}$$

denotes a k-step iterative refinement procedure for parametric and explicit models. While learned optimizers already utilize a mini-batch approach, they only consider parametric cases with fixed f and the information about observations is fed

[5] Certain second-order gradient-based learners are non Markovian, or Markovian in an augmented state space.

[6] In theory, this should learn the appropriate prior for the class of tasks for the parametric setup.

to $h_{\boldsymbol{\varphi}}$ solely through gradients. In particular, they model $h_{\boldsymbol{\varphi}}$ as

$$h_{\boldsymbol{\varphi}}(\boldsymbol{\theta}, \mathcal{B}) \coloneqq \text{Transformer}_{\boldsymbol{\varphi}}\Big(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{B}} \mathcal{L}\left(\mathbf{y}, f_\gamma\left(\mathbf{x}, \boldsymbol{\theta}\right)\right)\big|_{\boldsymbol{\theta}}\Big) \tag{11}$$

where the learned model takes a sequence of parameter updates and gradients; the above equation shows a particular case with Markovian assumption. At best, such methods can only recover gradient-based inference routines and cannot model more general optimization schemes (*e.g.* closed-form linear regression solution) as the mapping from observations to gradient is non-invertible.

In parametric and explicit methods, $\boldsymbol{\theta}$s can be seen as recurrent memory which compresses useful information of the current task from $\mathcal{D}_{\mathcal{T}}$, and can then be leveraged to make predictions for new $\mathbf{x}$, *i.e.* prediction for $\mathbf{x}$ given $\boldsymbol{\theta}_{\mathcal{T}}$ is conditionally independent of $\mathcal{D}_{\mathcal{T}}$ ($\boldsymbol{\theta}_{\mathcal{T}}$ serves as sufficient statistics). In contrast, the *implicit* formulation does not consider task-specific memory, instead its recurrent state is always tied to the query as well. This is achieved when $g_{\boldsymbol{\varphi}}$ simply provides subsampled mini-batches while f_γ is modeled as a recurrent application of a transformer r_γ with weights γ

$$\hat{\mathbf{y}}^{(0)} \xrightarrow{r_\gamma\left([\mathbf{x},\hat{\mathbf{y}}^{(0)}],\mathcal{B}^{(0)}_{\text{train}}\right)} \hat{\mathbf{y}}^{(1)} \xrightarrow{r_\gamma\left([\mathbf{x},\hat{\mathbf{y}}^{(1)}],\mathcal{B}^{(1)}_{\text{train}}\right)} \dots \xrightarrow{r_\gamma\left([\mathbf{x},\hat{\mathbf{y}}^{(k-1)}],\mathcal{B}^{(k-1)}_{\text{train}}\right)} \hat{\mathbf{y}}^{(k)}, \tag{12}$$

where k is the number of steps in the iterative refinement procedure for implicit models, $[\cdot,\cdot]$ describes a token and $\hat{\mathbf{y}}$ the query-specific predictions which form the recurrent states in this framework and are sequentially updated with $\hat{\mathbf{y}}^{(0)}$ being a learned initialization[7]. Here, f_γ gets the current prediction state $\hat{\mathbf{y}}^{(i)}$ and a mini-batch $\mathcal{B}^{(i)}_{\text{train}}$ as input and provides a refined prediction $\hat{\mathbf{y}}^{(i+1)}$. Since the implicit model never exposes task specific parameters $\boldsymbol{\theta}_{\mathcal{T}}$ disentangled from queries, it is non-trivial to leverage gradient information other than finetuning of the amortized model itself (Wei et al., 2021; Dalal et al., 2025).

Motivated by SGD, we learn $\boldsymbol{\varphi}$ and γ through a greedy strategy where training is done solely on single-step improvements for all three paradigms. The respective computational graphs are provided below, where we prevent gradients to flow back from $\boldsymbol{\theta}^{(i)}$ or $\hat{\mathbf{y}}^{(i)}$

$$\boldsymbol{\theta}^{(i)} \xrightarrow{h_{\boldsymbol{\varphi}}(\cdot,\mathcal{B})} \boldsymbol{\theta}^{(i+1)} \to \mathcal{L}(\mathbf{y}, f_\gamma(\mathbf{x}, \boldsymbol{\theta}^{(i+1)})) \quad \text{or} \quad \hat{\mathbf{y}}^{(i)} \xrightarrow{r_\gamma([\mathbf{x},\cdot],\mathcal{B})} \hat{\mathbf{y}}^{(i+1)} \to \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}^{(i+1)}) \tag{13}$$

Here, an outer loop is used to obtain the states at the i^{th} step $-$ $\boldsymbol{\theta}^{(i)}$ and $\hat{\mathbf{y}}^{(i)}$ respectively. Note that in this setup, $(\mathbf{x},\mathbf{y}) \in \mathcal{D}^{\text{valid}}_{\mathcal{T}}$ while $\mathcal{B} \subseteq \mathcal{D}^{\text{train}}_{\mathcal{T}}$, *i.e.* we are amortizing a learner to minimize validation loss. For evaluation, however, we use training and evaluation splits from completely new tasks $\mathcal{T}$ and thus test for generalization to new problems. All three frameworks $-$ *parametric*, *explicit* and *implicit* $-$ leverage a modification of causal masking to provide efficient and parallelizable computation of $\boldsymbol{\theta}$ and $\hat{\mathbf{y}}$ with mini-batches having varied number of observations, detailed in Appendix H.

H.2. Model Parameterization

We parameterize the amortized model, $g_{\boldsymbol{\varphi}}$ in the case of parametric and explicit model, and f_γ in the case of implicit model, as a transformer with 512 hidden dimensions, 2048 hidden dimensions in the feed-forward neural network and 8 heads and 8 layers. We use gelu activation function and perform normalization first in PyTorch's version of transformers. In addition, we consider a specific learnable linear encoder for gradients as well as observations, where observations are $(\mathbf{x},\mathbf{y})$ pairs and gradients correspond to $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{y}, f_\gamma(\mathbf{x},\boldsymbol{\theta}))|_{\boldsymbol{\theta}_{\mathcal{T}}}$ for parametric and explicit models while implicit models cannot use gradients.

In addition, for implicit models we append $\hat{\mathbf{y}}$ to query observations and re-use the observation encoder to embed queries in the same space. We additionally use a learnable query embedding that is added to the query tokens for implicit model to be able to differentiate context from query. For the ablation conducted on Pre-MLP state that is carried over, we leverage a separate learnable query embedding matrix that does not share weights with the observation encoder. Finally, to model predictions or vector fields in the case of generative modeling, we leverage a similar linear decoder that maps from 512 dimensions to dimensions of the prediction, parameters, latents or vector field depending on the task and modeling setup.

It is important to note that our model should be able to make predictions or model parameters conditioned on arbitrary number of context examples, and hence should be trained in a manner that allows for varying context length $-$ this is similar

[7] In theory, this initialization should learn the marginal (unconditional) distribution over predictions.

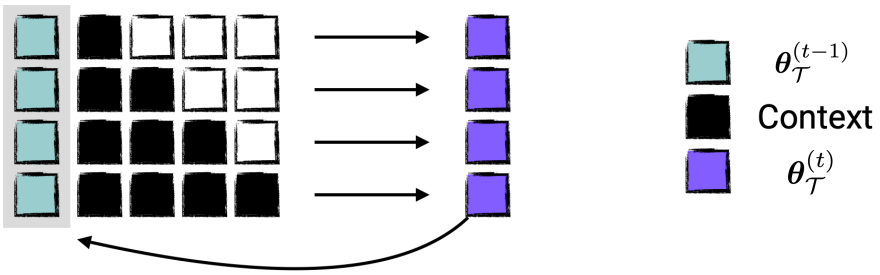


Figure 2. Masking procedure for the causally masked parametric and explicit model, where the context evolves according to a causal mask − the matrix of black and white squares describes the masking procedure where white blocks denote masks in the matrix − where each token in parallel consequently predicts parameters of interest conditioned on past batch of data and previous state. This mimics having variable sized dataset and processing for variable dataset sizes in parallel. The output corresponding to the last token is the $\boldsymbol{\theta}_{\mathcal{T}}^{(t)}$ that gets fed back recurrently.

to decoder only transformers that learn to make predictions for every context length in parallel. Given the difference in our setting from autoregressive language modeling, we provide two ways of implementing this amortization through the use of either a causally masked transformer or a non causal transformer, the details of which are provided in the next subsections.

Finally, for modeling iterative amortized inference, we use a simple setup where the output from one batch − $\boldsymbol{\theta}$ or $\hat{\mathbf{y}}$ depending on the amortization framework used − is fed back into the transformer with another batch after being detached from the computation graph. Here, $\boldsymbol{\theta}$ is just fed as another token with or without its gradient information as an additional token, while for implicit models $\hat{\mathbf{y}}$ is appended to its corresponding query, hence the state represents the predictions corresponding to the query that get updated after every iteration − note that the state is tied to a query here. Given a number of steps, *e.g.* k, we iterate over this process k times and compute gradients each time, accumulating them before taking a gradient step. Given that we detach the state before feeding it again, it prevents backpropagation through time and thus defines a greedy procedure.

H.3. Training Objective

The training procedure for all the models can be seen as optimization over the following loss for parametric and explicit models, where the batch sizes can contain variable number of observations N which is randomly sampled from $1 - 100$.

$$\arg\min_{\gamma,\boldsymbol{\varphi}} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\mathcal{T}} \mathbb{E}_{\mathcal{B}_{\mathcal{T}}^{\text{valid}} \subseteq \mathcal{D}_{\mathcal{T}}^{\text{valid}}} \mathbb{E}_{\mathcal{B}_{\mathcal{T}}^{(t),\text{train}} \subseteq \mathcal{D}_{\mathcal{T}}^{\text{train}}} \frac{1}{N} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{B}_{\mathcal{T}}^{\text{valid}}} \mathcal{L}(\mathbf{y}, f_\gamma(\mathbf{x}, g_{\boldsymbol{\varphi}}(\mathcal{B}_{\mathcal{T}}^{(t),\text{train}}, \boldsymbol{\theta}_{\mathcal{T}}^{(t)}))) \tag{14}$$

$$\boldsymbol{\theta}_{\mathcal{T}}^{(t)} = g_{\boldsymbol{\varphi}}(\mathcal{B}_{\mathcal{T}}^{(t-1),\text{train}}, \text{sg}(\boldsymbol{\theta}_{\mathcal{T}}^{(t-1)})) \tag{15}$$

In contrast, implicit models consider the following learning paradigm

$$\arg\min_{\gamma,\boldsymbol{\varphi}} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\mathcal{T}} \mathbb{E}_{\mathcal{B}_{\mathcal{T}}^{\text{valid}} \subseteq \mathcal{D}_{\mathcal{T}}^{\text{valid}}} \mathbb{E}_{\mathcal{B}_{\mathcal{T}}^{(t),\text{train}} \subseteq \mathcal{D}_{\mathcal{T}}^{\text{train}}} \frac{1}{N} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{B}_{\mathcal{T}}^{\text{valid}}} \mathcal{L}(\mathbf{y}, f_\gamma(\mathbf{x}, \mathcal{B}_{\mathcal{T}}^{(t),\text{train}}, \hat{\mathbf{y}}^{(t)}))) \tag{16}$$

$$\hat{\mathbf{y}}^{(t)} = f_\gamma(\mathcal{B}_{\mathcal{T}}^{(t-1),\text{train}}, [\mathbf{x}, \text{sg}(\hat{\mathbf{y}}^{(t-1)})]) \tag{17}$$

H.4. Causal Masked Model

Given that the implicit model differs considerably from the parametric and explicit model in terms of conditional independence assumptions defined, we need to handle the two cases somewhat differently to enable efficient parallelized implementations. We discuss the details below.

Parametric and Explicit. For the parametric and explicit model, we consider a simple causal masking over observations such that the model predicts $\boldsymbol{\theta}_{\mathcal{T}}$ corresponding to each token conditioned on the previous token, and the loss is aggregated by using all the intermediate $\boldsymbol{\theta}_{\mathcal{T}}$'s obtained to make predictions on a validation set and averaging them. This procedure is highlighted in Fig. 2.

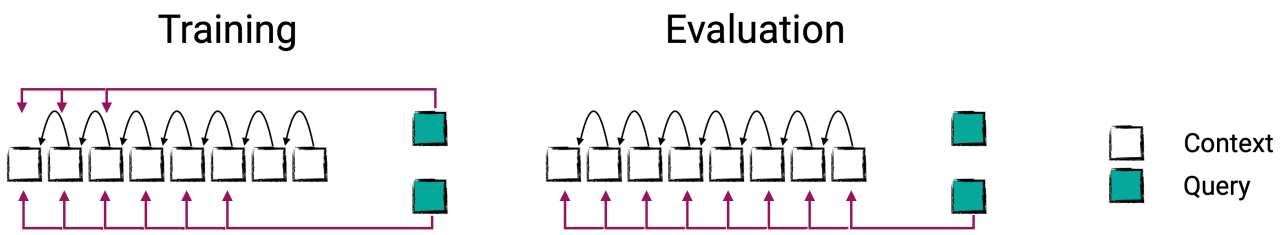


Figure 3. Masking procedure for the causally masked implicit model, where the context evolves according to a causal mask and the queries can attend somewhere in between, which conditions them on that particular position and the positions behind it. Since we use multiple queries in parallel which can look at different number of past contexts through this manipulation of the masking, we obtain parallel processing of multiple dataset sizes.

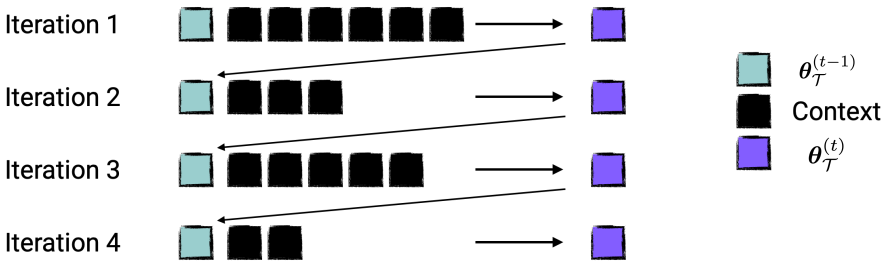


Figure 4. We show the case of non causal masked transformer for parametric and explicit setup where at each iteration, a differently sized context is provided as input to predict the next state. The size of the context is randomly sampled. At evaluation we use the full context.

Implicit. In contrast to the parametric approach, the implicit framework cannot deal with this in such a simple manner since the query is $\mathbf{x}$ itself which cannot remain in the same position as well as attend over variable sized context. To resolve this issue, we process in parallel multiple points from the validation set and manipulate the masking matrix to randomly let them attend to some token j and its predecessors. We use a causal transformer so this ensures that if the query looks at index j and before, then that query models prediction using a dataset of size j instead of context length. We use multiple queries in parallel, randomly choosing the corresponding j so that we can, in-parallel, model predictions for variable sized datasets in a single forward pass (though they have to correspond to different queries). A pictorial representation of this is showcased in Fig. 3.

H.5. Non-Causal Model

For the non causal masked version, at each training iteration we just randomly sample a length n and then consider a batch of only n observations for a forward pass. The benefits of this approach is that it can allow for conditioning on the n observations in-context without having to rely on a causal decomposition of attention but the downside is that the number of observations have to be kept fixed for a training iteration leading to unbiased gradients but they may have high variance since each training iteration only relies on a single sample (n) of the number of observations which then differ across training iterations. Thus, it is not possible to leverage parallel computation for a variety of number of observations within a single forward pass.

A benefit of this approach is that for parametric and explicit models, it can allow more than one token to denote the latents which is computationally infeasible for the masked case. For explicit models in particular, we consider 8 tokens for latents which are of 128 dimensions each; note that having this for masked case is possible but it will require complex masking procedures as well as efficient sparse attention kernels otherwise the context length would blow up relatively quickly.

We refer to Figs. 4 and 5 for details regarding the non causal transformer model for the parametric, explicit and implicit cases.

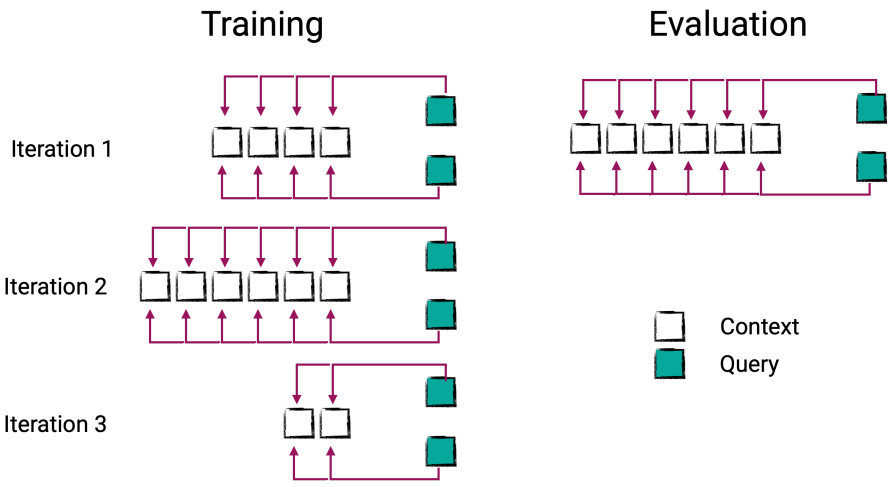


Figure 5. We highlight the case of non-causal transformer model for the implicit model where at each iteration a subset of observations are provided as context; their number randomly sampled. At evaluation we use the full context.

I. Additional Ablations

We first discuss ablations associated with parametric, explicit and implicit methods separately and then finally look at the impact of having a causal or a non-causal transformer as the modeling choice across the three amortization regimes.

I.1. Parametric Amortization

We first test our framework in the space of parametric models – where f is fixed as a linear predictor and g_φ trained to infer the optimal parameters $\boldsymbol{\theta}_\mathcal{T}$ of this mapping. In our experiments, we consider g_φ to be a transformer model which amortizes to the conditional information $\mathcal{D}_\mathcal{T}$ either through gradient signal or observations, or both through a k-step iterative procedure, where 1-step model solely taking observations as input reduces to amortization through hypernetworks while multiple steps with only gradient information reduces to learned optimizers.

Greedy Iterative Refinement. Our experiments in Table 6 demonstrate that the iterative refinement methodology leads to consistent improvements with increasing number of steps across a wide variety of tasks and the modality used for amortization – *i.e.* gradients, observations, or both. We also see that such estimators generalize OoD, for example when pre-trained on ImageNet classification and evaluated on CIFAR-10 at inference based solely on context examples without any parameter updates.

Beyond Gradient Signal. When g_φ solely leverages gradient information, we recover the space of learned optimizers. For relatively lower dimensional problems, we see that the space of sole gradient-based learned optimizers is suboptimal and additionally leveraging the observations can lead to substantial improvements. Alternatively, we see a trend reversal in higher dimensional problems as the hypothesis space is much larger, making it harder for the model to infer the right parameters. In such cases, gradient provides a more reliable signal especially when there are fewer observations.

I.2. Explicit Amortization

Next, we consider the explicit model – f_γ is a trained MLP taking a query $\mathbf{x}$ and latent $\boldsymbol{\theta}_\mathcal{T}$ as input, while g_φ is a transformer inferring the latent from $\mathcal{D}_\mathcal{T}$. Here, the 1-step approach with just observations as input reduces to the explicit model defined in (Mittal et al., 2024).

Greedy Iterative Refinement. Similar to the parametric setup, we see in Table 4 that our proposed approach leads to improved performance with increasing number of steps, again consistent with all the input modalities. In contrast to the parametric experiments, we see an increased importance of gradient information in inferring the right latents. Since there is inherent non-stationarity in the optimization procedure we hypothesize that gradients provide a clearer signal for the inference mechanism g_φ.

Suboptimality at large-scale. Importantly, we see that parametric modeling outperforms the explicit setup highlighting

the difficulty of jointly learning f_γ and $g_{\boldsymbol{\varphi}}$, which is surprising since the class of solutions expressed by the former form a subset of the latter. We attribute this suboptimality to the optimization process as f_γ and $g_{\boldsymbol{\varphi}}$ are linked in a complicated, non-stationary manner, *i.e.* if the prediction function f_γ changes then its inference mechanism $g_{\boldsymbol{\varphi}}$ also needs to adapt, and vice versa.

I.3. Implicit Amortization

Finally, we consider implicit parameterization where f_γ jointly leverages both training dataset $\mathcal{D}_\mathcal{T}$ as well as the query $\mathbf{x}$ to model predictions $\mathbf{y}$. Here, the 1-step approach boils down to ICL (Mittal et al., 2024; Garg et al., 2022; Müller et al., 2021; Von Oswald et al., 2023).

Greedy Iterative Refinement. Our experiments on iterative refinement in Tables 8 and 9 highlight improvement in performance with increasing number of steps. In addition to predictive tasks, we also consider generative modeling where the task is to model the underlying distribution described through the context examples. The amortized model is trained to infer the conditional vector field which interpolates a path between standard normal and the observed distribution, conditioned on $\mathcal{D}_\mathcal{T}$ (Atanackovic et al., 2024; Chang et al., 2024). Our results in Table 7 and Fig. 1 showcase improved ability of modeling the underlying distributions defined by the context with more steps[8].

State Parameterization. We analyze the design choices associated with the state that persists across iterations to be − (a) Pre-MLP: high-level latent variables tied to the query, (b) Logits: current prediction before undergoing parameter-free normalization like softmax, or (c) Softmax: current predictions after softmax. Fig. 7 shows that modeling the state at logits leads to best performance, and in general shows the importance of being close to prediction for greedy refinement.

I.4. Causal vs Non Causal Transformer

We perform a thorough comparison between using a causal and non-causal transformer in all three amortization regimes − *parametric*, *explicit*, and *implicit*. We generally see mixed results which we hypothesize is because while the non-causal method is more expressive algorithm, it has a higher variance of gradients during training as one cannot process in parallel multiple dataset sizes and thus leverage larger effective batch size which makes them inefficient to train and the optimization process largely more unstable.

Parametric. For the parametric model, we consider the causal method that predicts weights of the linear predictor in parallel conditioned on $\mathbf{x}_{1:n}$ for all n in parallel by leveraging a causal mask, as described in the section above whereas the non-causal framework leverages a more expressive transformer since the attention is not restricted to causal attention but at every training iteration, a single n is sampled at random and conditioning is done on only a subset of n observations (this n is changed at every iteration). We can see that this is an unbiased estimate of the gradient but with only a single sample of n, and thus suffers from larger variance but allows for more expressive architecture. We highlight the performance difference of the two approaches for various number of iterations of amortization as well as modalities of information − gradient or observations or both − in Figs. 8 to 10.

Explicit. Similar to the parametric modeling setup, we look at the explicit amortization in the case of both a causal and a non-causal transformer with different number of iterations as well as different modalities of information − gradient or observations or both − in Figs. 11 to 13.

Implicit. We refer to Fig. 14 for experimental details contrasting the causal and non-causal transformer model.

As highlighted at the start of the section, our results on the comparisons are mixed as one model is more expressive while the other leads to optimization challenges due to potential variance in gradients since it is a single sample monte carlo estimate.

[8]Here, more steps imply steps to infer the vector field $v_t(\cdot|\mathcal{D}_\mathcal{T})$ for each t, and not more integration steps.

Training Tasks ⟶			MNIST		FMNIST		ImageNet		
	Steps	Lin Reg	MNIST	FMNIST	FMNIST	MNIST	ImageNet	CIFAR10	CIFAR100
Grad	1	51.3±0.3	76.2±2.0	66.7±1.0	63.8±1.7	79.9±1.6	88.5±0.1	44.9±4.5	92.5±1.1
	5	4.1±0.1	48.6±1.2	51.8±0.5	41.6±1.5	49.5±3.0	83.1±0.2	17.9±1.0	88.1±0.8
	10	0.5±0.0	39.7±0.7	43.5±0.9	38.1±0.4	40.7±0.8	83.4±0.1	14.7±0.2	88.1±0.6
Data	1	16.3±0.2	43.9±1.3	40.5±1.2	35.9±1.7	38.8±1.4	90.8±0.2	60.9±3.5	93.9±1.1
	5	0.5±0.0	35.5±0.5	34.3±0.6	30.8±0.3	31.3±0.5	95.2±0.1	46.9±1.8	96.3±0.6
	10	0.3±0.0	36.6±0.3	35.3±0.3	29.8±0.3	29.1±0.4	93.2±0.1	28.8±0.8	95.0±0.8
Grad	1	25.1±0.3	52.3±1.5	48.2±0.8	37.7±1.9	41.9±1.3	96.7±0.1	70.2±4.2	97.2±0.8
+	5	0.6±0.0	39.2±1.0	37.5±0.3	32.0±0.7	33.4±0.5	93.8±0.1	42.1±2.4	94.8±0.7
Data	10	0.4±0.0	32.0±0.5	32.0±0.2	30.5±0.2	30.3±0.3	92.5±0.1	24.8±1.0	94.1±0.4

Table 6. Our experiments on *parametric amortization* reveal benefits of multiple steps of iterative refinement across ID and OoD (gray columns) evaluation. Top row describes pre-training tasks and the second row evaluation tasks, with error metric. We also see that sole reliance on gradients is often insufficient and leveraging observations directly can lead to improved performance with fewer iterations.

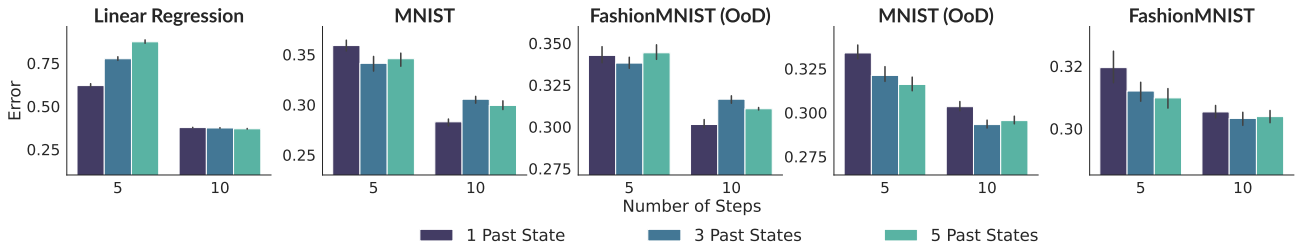


Figure 6. We analyze the benefits, or lack thereof, of leveraging multiple past states and gradients for *parametric amortization*, when we use both observations and gradients as conditional inputs.

Steps	Alphabets				GMM			
	ID		OoD		2-dimensional		5-dimensional	
	$\mathcal{W}_2$	$\mathcal{W}_1$	$\mathcal{W}_2$	$\mathcal{W}_1$	$\mathcal{W}_2$	$\mathcal{W}_1$	$\mathcal{W}_2$	$\mathcal{W}_1$
1	0.28±0.00	0.20±0.00	0.78±0.01	0.64±0.00	1.12±0.01	0.58±0.00	3.27±0.01	1.82±0.01
5	0.31±0.00	0.22±0.00	0.72±0.01	0.58±0.00	0.90±0.01	0.38±0.00	2.50±0.02	1.05±0.01
10	0.29±0.00	0.21±0.00	0.67±0.01	0.54±0.01	0.81±0.02	0.32±0.01	2.37±0.02	0.95±0.01

Table 7. We see improved sample quality (under Wasserstein metrics) using *implicit amortization* with increasing number of steps for generative modeling over OoD alphabets and new GMMs.

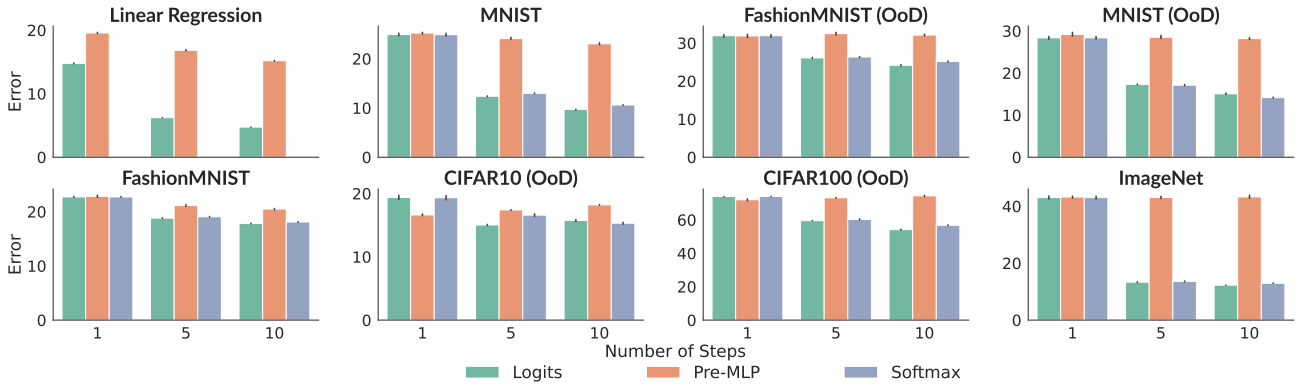


Figure 7. Our ablations reveal that carrying over logits as the recurrent state across iterations in implicit models outperforms other state representations, with softmax outputs performing comparably.

Training Tasks ⟶		MNIST		FMNIST		ImageNet		
Steps	Lin Reg	MNIST	FMNIST	FMNIST	MNIST	ImageNet	CIFAR10	CIFAR100
1	$14.8_{\pm 0.1}$	$24.9_{\pm 0.4}$	$31.9_{\pm 0.4}$	$22.8_{\pm 0.1}$	$29.1_{\pm 0.4}$	$43.0_{\pm 0.7}$	$19.4_{\pm 0.5}$	$73.9_{\pm 0.1}$
5	$6.2_{\pm 0.0}$	$12.4_{\pm 0.1}$	$26.1_{\pm 0.2}$	$18.8_{\pm 0.1}$	$16.9_{\pm 0.2}$	$13.3_{\pm 0.2}$	$15.0_{\pm 0.1}$	$59.4_{\pm 0.2}$
10	$4.7_{\pm 0.0}$	$9.7_{\pm 0.1}$	$24.2_{\pm 0.2}$	$17.8_{\pm 0.1}$	$15.7_{\pm 0.2}$	$12.3_{\pm 0.1}$	$15.9_{\pm 0.3}$	$54.2_{\pm 0.4}$

Table 8. Our experiments on *implicit amortization* show consistent improvements in performance with increasing number of steps across a wide range of predictive tasks, with error as evaluation metric.

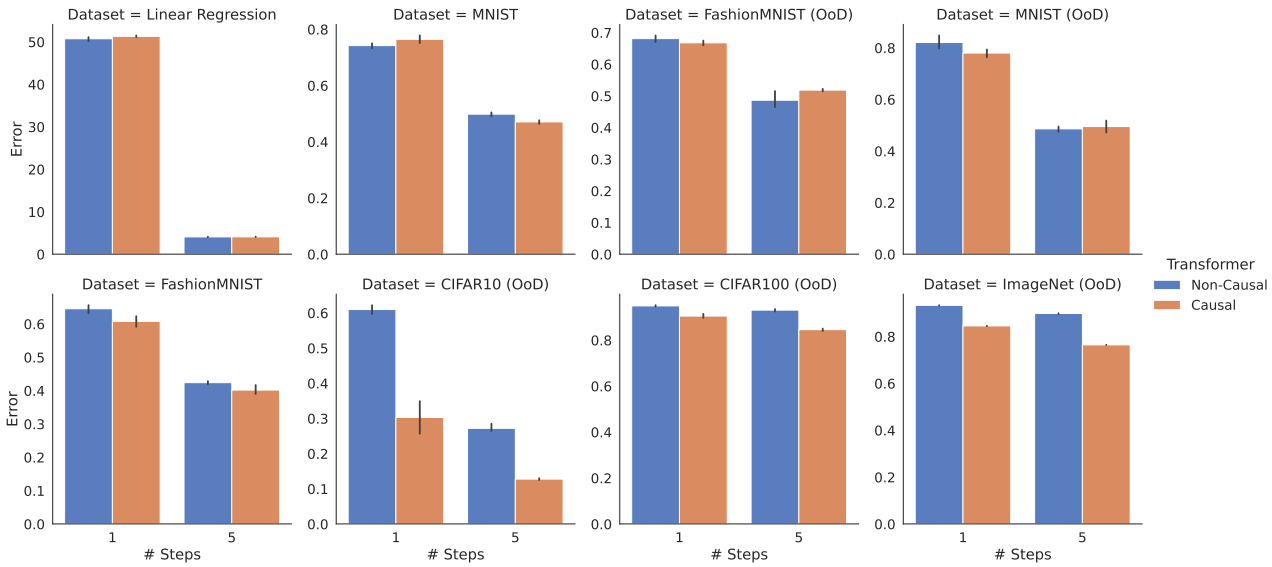


Figure 8. **Parametric with Gradient Information**. We look at the comparison between causal and non-causal transformer for a parametric model that leverages gradients but not data.

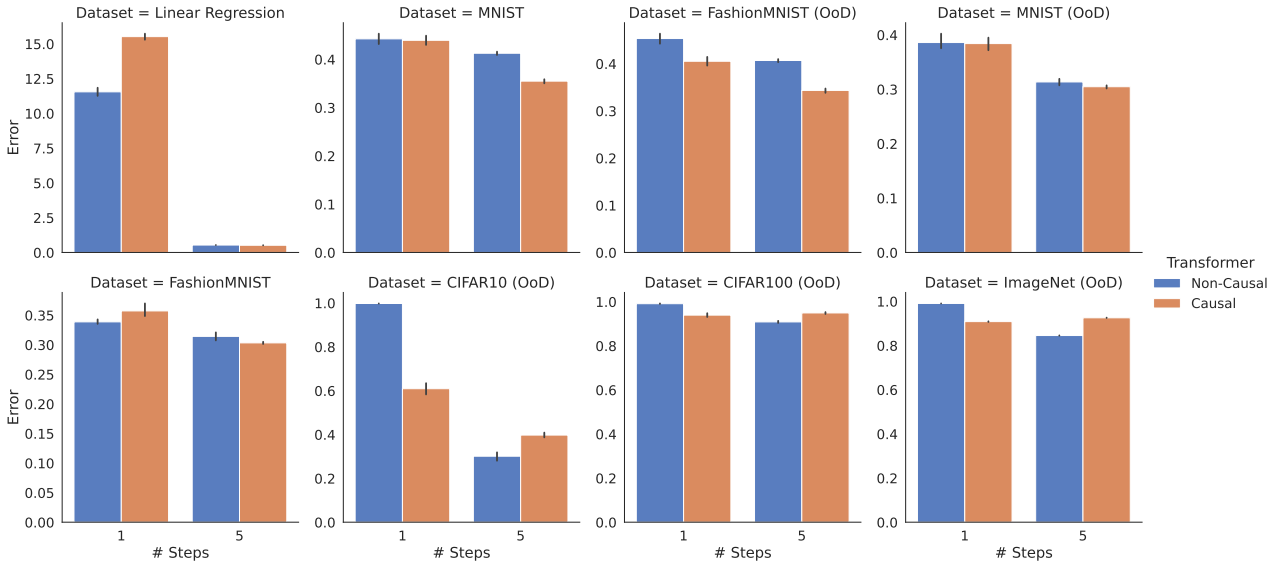


Figure 9. **Parametric with Observations Information**. We look at the comparison between causal and non-causal transformer for a parametric model that leverages data but not gradients.

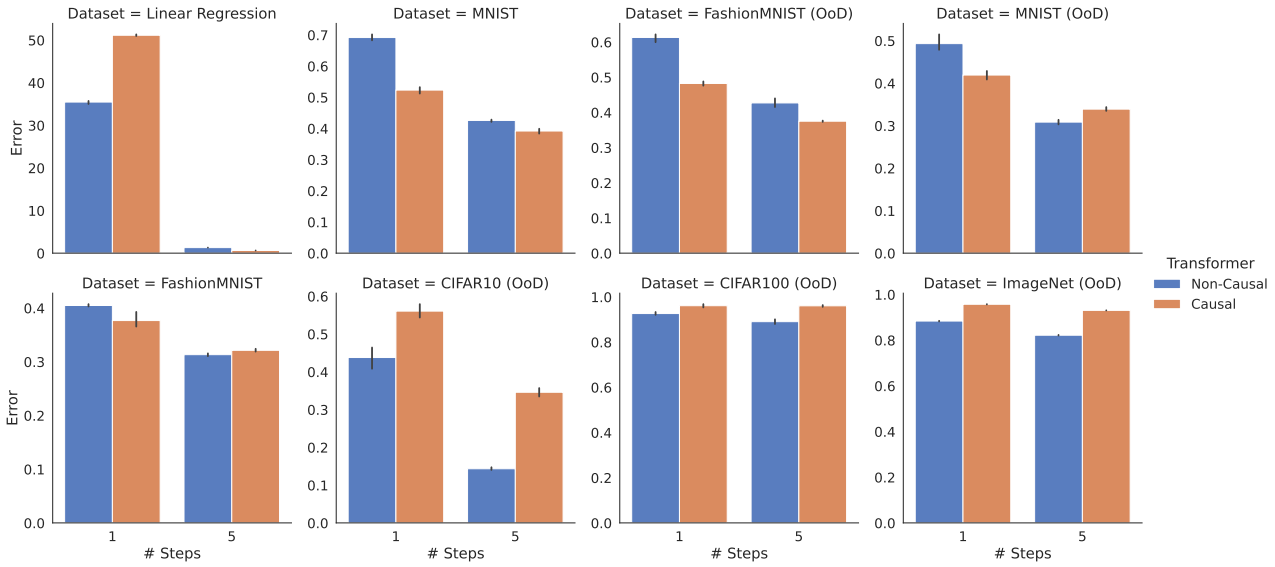


Figure 10. **Parametric with Observations and Gradient Information**. We look at the comparison between causal and non-causal transformer for a parametric model that leverages both gradients and data.

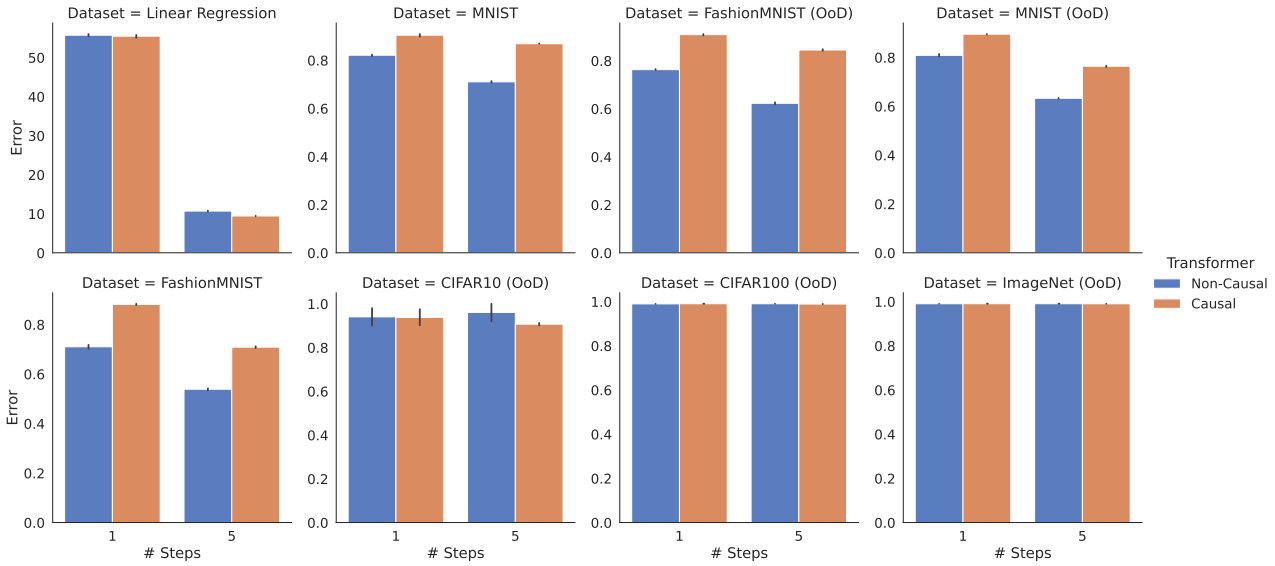


Figure 11. **Explicit with Gradient Information**. We look at the comparison between causal and non-causal transformer for an explicit model that leverages gradients but not data.

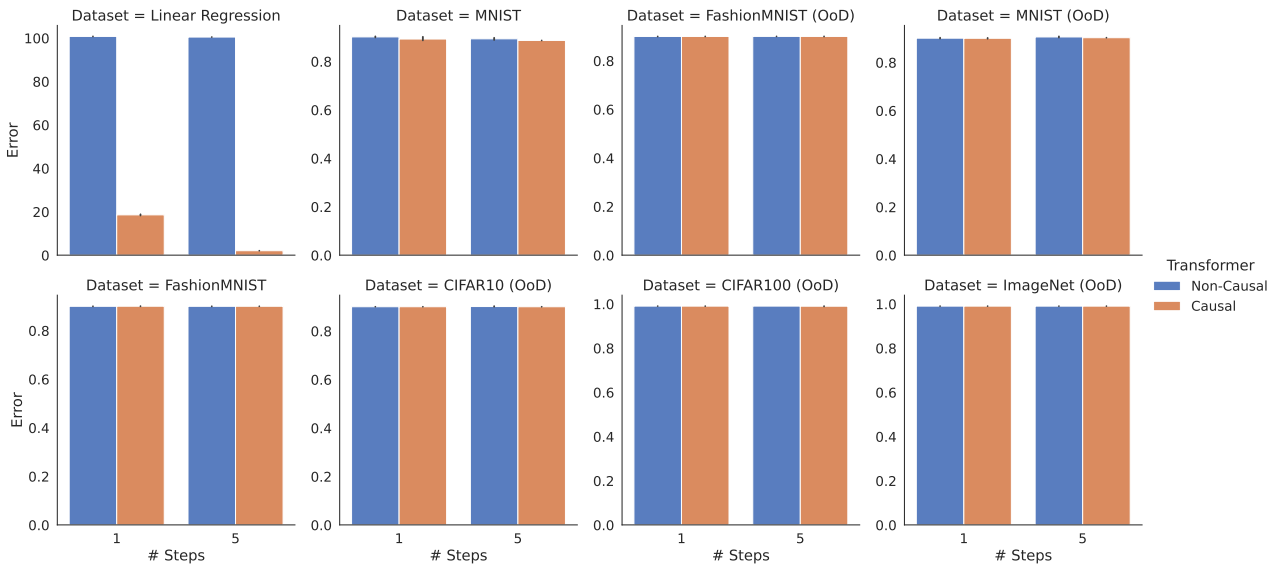


Figure 12. **Explicit with Observation Information**. We look at the comparison between causal and non-causal transformer for an explicit model that leverages data but not gradients.

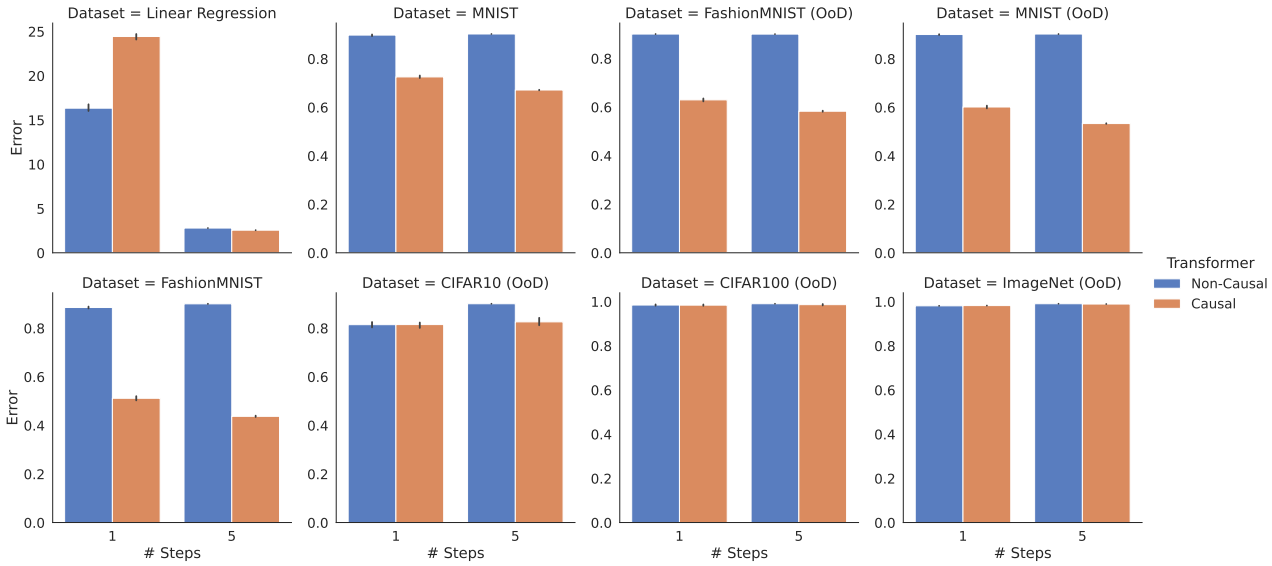


Figure 13. **Explicit with Observation and Gradient Information**. We look at the comparison between causal and non-causal transformer for an explicit model that leverages both gradients and data.

	20 Nodes				50 Nodes			
Steps	LIN IN	RFF IN	LIN OUT	RFF OUT	LIN IN	RFF IN	LIN OUT	RFF OUT
1	0.52 ±0.05	0.43 ±0.04	0.64 ±0.04	0.51 ±0.06	0.7 ±0.05	0.66 ±0.04	0.78 ±0.03	0.69 ±0.08
5	0.52 ±0.06	0.35 ±0.05	0.68 ±0.08	0.33 ±0.06	0.66 ±0.05	0.56 ±0.04	0.74 ±0.04	0.67 ±0.05
10	0.44 ±0.11	0.37 ±0.07	0.66 ±0.08	0.61 ±0.05	0.64 ±0.06	0.52 ±0.04	0.77 ±0.03	0.73 ±0.03

Table 9. We evaluate implicit models on topological order prediction, where ID uses SCMs with linear (LIN IN) / non-linear (RFF IN) mechanisms for 20 node graphs and OoD (gray columns) changes function parameters (LIN OUT / RFF OUT) or graph size (50 nodes), with classification error as metric.

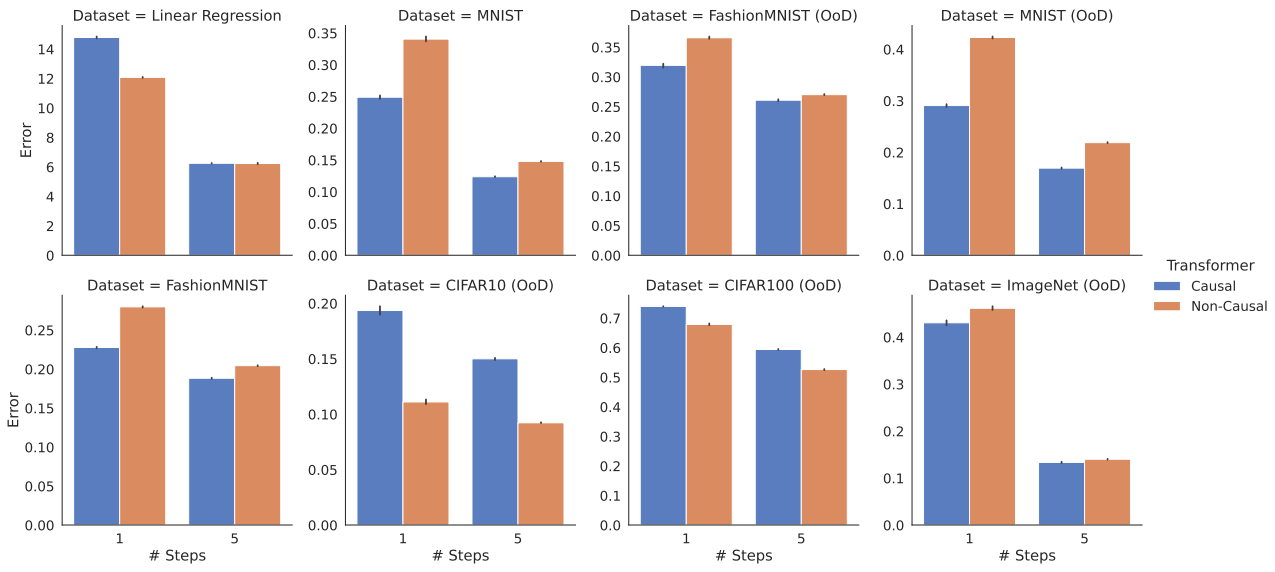


Figure 14. **Implicit with Observation Information**. We look at the comparison between causal and non-causal transformer for an implicit model, which can only leverage data.