EXPLICIT KNOWLEDGE FACTORIZATION MEETS IN-CONTEXT LEARNING: WHAT DO WE GAIN?

Sarthak Mittal[1,2*] **Eric Elmoznino**[1,2*] **Léo Gagnon**[1,2*] **Sangnie Bhardwaj**[1,2,3]
Dhanya Sridhar[1,2] **Guillaume Lajoie**[1,2]
[1]Mila [2]Université de Montréal [3]Google DeepMind

ABSTRACT

Transformer models have shown considerable success in modeling predictive problems in diverse domains. Notably, they can efficiently learn *in-context* (ICL), i.e. solve new tasks without any further training when provided some examples in the prompt. While first observed in language, consequent studies explore this phenomenon in controlled settings where the model is trained on a known distribution of tasks. Transformer therefore have to jointly infer what the task is and how to solve it; the essence of Bayesian posterior predictive inference. However as this is done implicitly, there is no guarantee that the model explicitly represent the task latent, which we argue could have a number of benefits (analog to those of parametric methods over non-parametric ones). This begs a natural question: is there any benefit in encouraging this explicit representation or are we better off letting the model implicitly decide an appropriate solution space. We thoroughly analyze a Transformer imbued with such inductive bias and show both its potential and limitations. Although it gains in interpretability and controlability, it doesn't lead to the expected performance boost; we hypothesize that this is due to a lack of capacity in using the extracted latent to perform conditional predictions.

1 INTRODUCTION

Transformers (Vaswani et al., 2017) have a remarkable ability to adapt at inference by leveraging earlier tokens in their context to modulate their outputs for a given query. This ability, termed In-Context Learning (ICL), is a primary driver of the impressive emergent abilities and out-of-distribution (OOD) generalization in large language models (Lu et al., 2023). However, we still lack an understanding of what architectural motifs are responsible for such abilities, and what design choices might encourage it. To make progress, recent works concentrate on simplified tasks to understand the conditions under which ICL succeeds (Mueller et al., 2023) and how it can be improved.

As such, synthetic ICL consider latent variable models (LVM) $p(y|\boldsymbol{x};\boldsymbol{z})$ and given a number of demonstrations $\mathcal{D}_{\boldsymbol{z}} = \{(\boldsymbol{x}_i, y_i)\}_i \sim p_{\boldsymbol{z}}$ in context, asks the model to make a prediction about a new $\boldsymbol{x}_*$ and also generalize to new $\boldsymbol{z}$ at test time. A priori (Genewein et al., 2023), this pushes models to learn the Bayesian posterior predictive (Equation 1); a perspective validated experimentally on increasingly complex LVMs (Mikulik et al., 2020; Guo et al., 2023a; Akyürek et al., 2024) .

$$p(y \mid \boldsymbol{x}_*, \mathcal{D}) = \int_{\boldsymbol{z}} p(y \mid \boldsymbol{x}_*; \boldsymbol{z}) p(\boldsymbol{z} \mid \mathcal{D}) d\boldsymbol{z} \tag{1}$$

While this process implicitly marginalizes over some unknown latent space, this space doesn't have to be explicitly represented in the model, which could result in a loss of controllability over this space, make it hard to encode contraints or prior knowledge and potentially hinder systematic generalization. In this paper, we analyze whether encouraging **explicit** inference as opposed to **implicit** one through architectural biases can lead to benefits over a regular transformer.

Explicitly representing and inferring these task parameters $\boldsymbol{z}$, similar to parametric inference methods typically used in Bayesian methodology, is attractive because it mirrors the way in which we often believe underlying data-generating processes work (in and out of distribution). Some unobserved factors remain constant within a given context (e.g. the layout of a room) and produce a

*Equal contribution

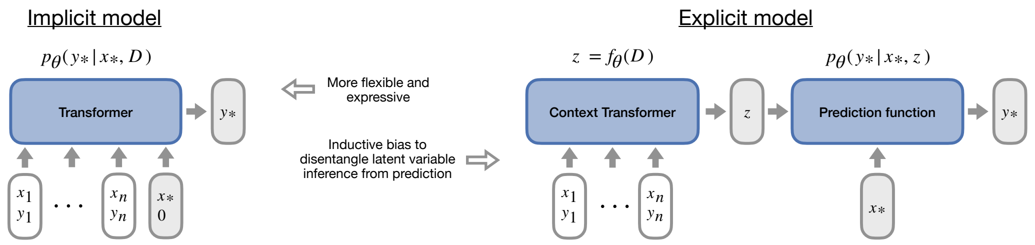

Figure 1: We compare the benefits of the implicit (*left*) and the explicit (*right*) model. The latter disentangles modeling context aggregation and prediction whereas the former models them jointly.

stream of observations that depend on those factors (e.g. the images on our retina as we move inside the room). Another advantage of this method is interpretability through the explicit $\boldsymbol{z}$ variable, which allows us to interpret and discover the structure of the data and task. Some evidence shows that LLMs are indeed able to do this in some context (Hendel et al., 2023; Todd et al., 2023).

In contrast, multiple studies show that induction heads (Olsson et al., 2022) and related mechanisms (Hahn & Goyal, 2023; Han et al., 2023) emerge throughout learning, modeling the posterior predictive **implicitly**. Akin to non-parametric methods (eg. kernel regression), implicit modeling provides more freedom to the optimization procedure in finding the right solutions and does not prescribe design choices like the dimensionality or structure of the latent, which may be beneficial if we have a misinformed prior. However, these methods are less interpretable, require a lot of data, and lead to over-fitting as their solutions are local and lack systematicity (Russell & Norvig, 2010).

Although both motifs (*explicit* or *implicit*) can be learned, transformers arguably learn non-parametric mechanisms more easily (Zhou et al., 2023) since the attention mechanisms implement kernel regression almost by definition (Tsai et al., 2019). We posit that if the query can directly attend the context, non-parametric methods will probably be preferred by the optimizer. Yet, explicit representations might afford marked advantages that do not arise naturally in standard architectures.

In this work, we perform a thorough comparison between a regular transformer (implicit) and its variant with a latent bottleneck (explicit) in ICL settings. Importantly, unlike implicit model, the explicit one disentangles the two processes of latent variable inference and prediction into two separate modules. We find that results are varied, and depend on the task. We could not find unilateral advantages of the explicit model in terms of OOD generalization, and we identify the prediction part of the model (which combines $\boldsymbol{z}$ and $\boldsymbol{x}_*$ to make the prediction) as a potential cause. Nevertheless, we show that the explicit model has key benefits in some cases, and are always more interpretable. This is especially valuable in situations when contextual interventions on the model are needed.

2 Implicit vs Explicit Inference

We look at ICL in the context of algorithmic problems where the task is to predict the target y_* from a query point $\boldsymbol{x}_*$ when provided with some context examples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_i$. During training, different draws of context sets $(\mathcal{D}_1, \mathcal{D}_2, ...)$ share the same underlying functional mapping $g : \boldsymbol{x}, \boldsymbol{z} \rightarrow y$ but different latents $\boldsymbol{z}'$s; for example g could be a linear function but $\mathcal{D}_1$ could be generated from $\boldsymbol{z}_1$ whereas $\mathcal{D}_2$ from $\boldsymbol{z}_2$, similar to Von Oswald et al. (2023). Thus, the model not only has to learn the prediction function g but also efficiently aggregate information about $\boldsymbol{z}$ from the context $\mathcal{D}$ to make predictions for $\boldsymbol{x}_*$. Thus, this general framework can be decomposed into two parts

Context Aggregation. This component deals with inferring some notion of task-dependent sufficient statistics of the context or latent variables of interest such that the prediction becomes conditionally independent of the context, i.e. $p(y_* \mid \boldsymbol{x}_*, \mathcal{D}, \boldsymbol{z}) = p(y_* \mid \boldsymbol{x}_*, \boldsymbol{z})$. In the above working example, it can be seen as inferring $\boldsymbol{z}$ conditioned on the context $\mathcal{D}$.

Predictive Modeling. This aspect is associated with how the aggregated context is further used to drive predictions. In the above example, it refers to learning the functional mapping g.

As discussed, transformers don't have a clear incentive to make this explicit separation. Thus, in order to enforce explicit representation of $\boldsymbol{z}$, we consider an architectural modification where the query $\boldsymbol{x}_*$ cannot directly attend to the context. Formally, we compare the following two models

Implicit Model. This refers to the traditional in-context learning computation afforded by transformer models. In this setup, given the set of observations $\mathcal{D}$ (context) and a query point $\boldsymbol{x}_*$,

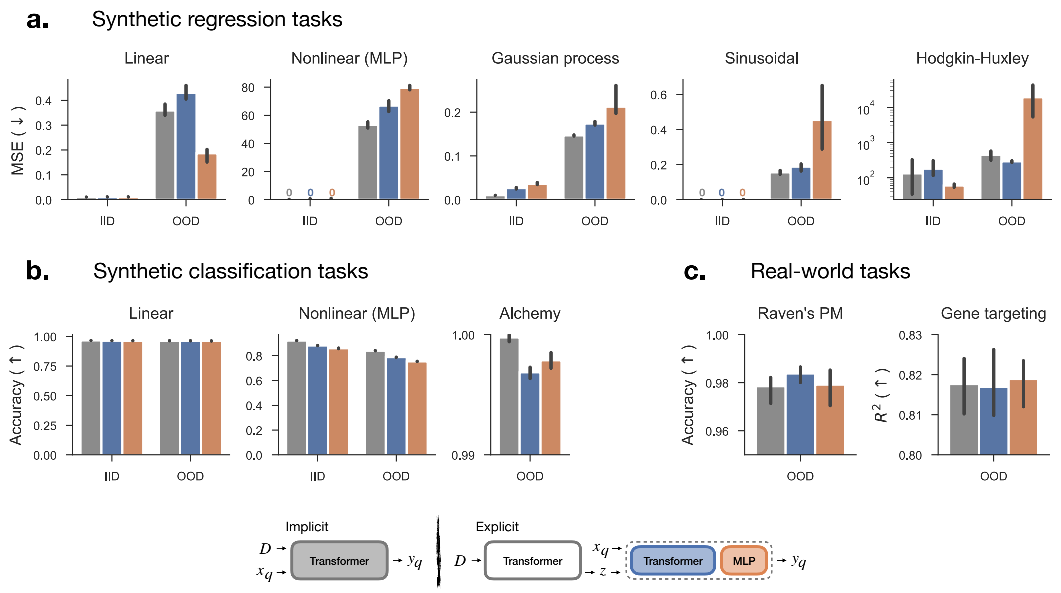

Figure 2: Investigation of model performances across a wide variety of tasks – synthetic regression and classification problems as well as more complex reasoning and gene targeting problems.

the prediction y_* for this query is modeled directly as $p_\varphi(y_* | \boldsymbol{x}_*, \mathcal{D})$, where p_φ is defined using a transformer. Here, p_φ is tasked with modeling both context aggregation and predictive modeling.

EXPLICIT MODEL. This represents the architectural variation which disentangles context aggregation and predictive modeling by first constructing a task representation $\boldsymbol{z}_\psi(\mathcal{D})$ using the set of observations (*context aggregation*) and then leverages another network p_φ to make the prediction for a new point $\boldsymbol{x}_*$ (*predictive modelling*) as $p_\varphi(y_* | \boldsymbol{x}_*, \boldsymbol{z}_\psi(\mathcal{D}))$. A key insight is that the *task statistics* are invariant to the queries when modeling prediction. We use a transformer for the context model and experiment with different models for the prediction function.

IMPLICIT VS EXPLICIT. We first hypothesize when would each setup perform better. If the data is generated with a linear model (i.e. $y = \boldsymbol{w}^T \boldsymbol{x}$), the right predictor can be precisely described using the weight vector $\boldsymbol{w}$, making the explicit model better suited. In contrast, when the data is generated with a Gaussian Process (GP), the implicit model should be superior since by construction query prediction relies on computing its similarities with all points in the context. In this case, the sufficient statistics of GP-based data with RBF kernel is infinite dimensional (i.e. a point in function space), which can't be captured by the explicit model. In general, we hypothesize that the explicit model would be superior when the underlying true model is parametric and low-dimensional but in case of a non-parametric or very high dimensional parametric model, the implicit model would outperform.

3 EXPERIMENTS

SETUP We conduct experiments across a wide variety of tasks to highlight the differences between the implicit and explicit models. In each task, the models get a set of observations as input $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ and have to make predictions about $\boldsymbol{x}_*$, where the functional form of the true mapping $g : \boldsymbol{x} \to y$ changes across different tasks (eg. linear regression, sinusoidal regression, etc). Evaluation is done based on mean-squared error or accuracy. The out-of-distribution (OOD) settings correspond to when x_* is taken outside the distribution used in $\mathcal{D}$. See §B for task details.

SYNTHETIC REGRESSION TASKS. We first consider regression problems, i.e. $y \in \mathbb{R}$, and experiment with multiple functions: linear, MLP-based, sinusoidal with multiple frequencies, temporal as a Hodgkin Huxley ordinary differential equation and Gaussian Process based non-parametric distribution (§B.1 for details). Our experiments in Fig. 2 (a) show that across most settings, both implicit and explicit models perform similarly, with no clear trend between the Explicit-MLP and Explicit-Transformer. We also see that in the case of a non-parametric g, the right prediction is to compare the query to the whole context, and thus an implicit model does consistently better.

SYNTHETIC CLASSIFICATION TASKS. We next look at classification tasks, i.e. y can only take finite values. We again consider linear and MLP-based functional mappings. Additionally, we also

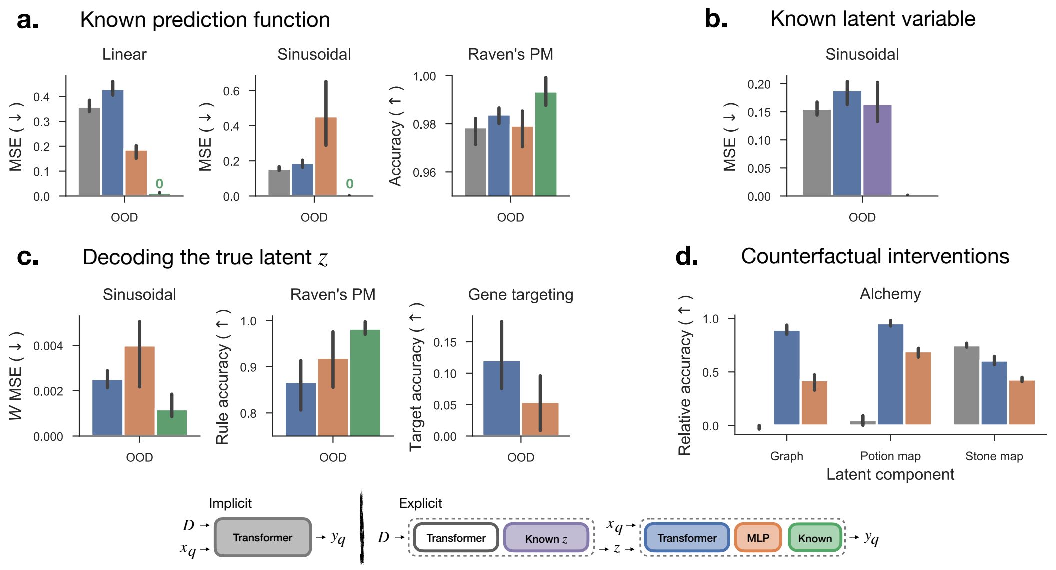

Figure 3: (*Top*) Suboptimality in Predictive Modeling. Known prediction function leads to significantly better OOD performance (a), but known context z doesn't (b). (*Bottom*) Explicit models are interpretable as the bottleneck allows us to decode the latent (c) and intervene on it (d).

look at the Alchemy task, consisting of compositional symbolic transformations governed by latent rules (§B.2 for details). Fig. 2 (b) highlights consistent benefits in using the implicit model.

Real World Tasks. Lastly we look at two more natural tasks – Raven's Progressive Matrices and Gene Targeting. The first is a reasoning task used in IQ tests which requires completing a sequence of 9 objects based on simple variations in high-level attributes (e.g. shape, size, etc.). The second requires predicting genetic expression of cells following a CRISPR gene intervention (§B.3 for details). Fig. 2 (c) shows there is no significant difference between the two models.

Suboptimality in Predictive Modeling. It is surprising to see that the explicit model does not outperform the implicit one on tasks where parameters are low-dimensional; as it is better aligned with the data generating process. Importantly, we observed that endowing the explicit model with the known decoder lead to drastic improvement (Fig. 3 (a)), whereas endowing it with the known latent didn't (Fig. 3 (b)); this suggests that the lack of capacity of the decoder is responsible for the underwhelming performance of the explicit model. Extending this reasoning to the implicit model, this might be the reason why regular transformer can difficultly perform parametric inference.

Interpretability. Fig. 3 (c) shows that we can often decode task-specific parameters from the aggregated context in explicit model, thereby providing a mechanism to understand when the model could be attaching to spurious correlations through readouts on the bottleneck.

Interventional Ability. Owing to the interpretable nature of the explicit model, we are able to find subspaces of the bottleneck that causally encodes different parts of the latent z using the Distributed Alignment Search (DAS) method from Geiger et al. 2023. Fig. 3 (d) shows a baseline-adjusted Interchange Intervention Accuracy (IIA) for each latent – how ofter intervening on the identified subspace is consistent with the counterfactual (ground-truth) prediction. For the implicit model, we try DAS on every layer of the residual stream and report the best IIA. See §C.1 for details.

4 Conclusion

We propose explicit factorization of knowledge into context aggregation and prediction modeling, showing that while it does comparably to the implicit model in downstream performance, it provides benefits of interpretability of latent variables and interventional ability. This finding might differ for more complex tasks and with more scale, but it suggests that for simple computations, implicit latent modeling emerges with naturally accurate and generalizable solutions. Nevertheless, we note that explicitly promoting parametric-like latent embedding might still yield advantages if a downstream decoder can make optimal use of them (i.e. learns the right parametric model). Our experiments indicate this is not currently the case when training transformers with bottlenecks end-to-end. Indeed, we demonstrate that the failure mode of explicit models is that they often do not learn the right prediction model to leverage explicit latent variables. This points to a line of future work that could seek to incorporate inductive biases in the prediction model to better leverage the inferred latent variables.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the computing resources provided by the Mila cluster to enable the experiments outlined in this work. SM acknowledges the support of UNIQUE's scholarship. GL and DS acknowledge the support of CIFAR. The authors also thank NVIDIA for the computing resources.

References

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms, 2024.

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

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes, 2023.

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional neural processes, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes, 2018b.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations, 2023.

Tim Genewein, Grégoire Delétang, Anian Ruoss, Li Kevin Wenliang, Elliot Catt, Vincent Dutordoir, Jordi Grau-Moya, Laurent Orseau, Marcus Hutter, and Joel Veness. Memory-based meta-learning on non-stationary distributions, 2023.

Luke A Gilbert, Max A Horlbeck, Britt Adamson, Jacqueline E Villalta, Yuwen Chen, Evan H Whitehead, Carla Guimaraes, Barbara Panning, Hidde L Ploegh, Michael C Bassik, et al. Genome-scale crispr-mediated control of gene repression and activation. *Cell*, 159(3):647–661, 2014.

Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning, 2023.

Jordi Grau-Moya, Tim Genewein, Marcus Hutter, Laurent Orseau, Grégoire Delétang, Elliot Catt, Anian Ruoss, Li Kevin Wenliang, Christopher Mattern, Matthew Aitchison, and Joel Veness. Learning universal predictors, 2024.

Tianyu Guo, Wei Hu, Song Mei, Huan Wang, Caiming Xiong, Silvio Savarese, and Yu Bai. How do transformers learn in-context beyond simple functions? a case study on learning with representations, 2023a.

Yuxuan Guo, Yifan Hao, Rui Zhang, Enshuai Zhou, Zidong Du, Xishan Zhang, Xinkai Song, Yuanbo Wen, Yongwei Zhao, Xuehai Zhou, Jiaming Guo, Qi Yi, Shaohui Peng, Di Huang, Ruizhi Chen, Qi Guo, and Yunji Chen. Emergent communication for rules reasoning, 2023b.

Michael Hahn and Navin Goyal. A theory of emergent in-context learning as implicit structure induction, 2023.

Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. Explaining emergent in-context learning as kernel regression, 2023.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors, 2023.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

John and Jean Raven. *Raven Progressive Matrices*, pp. 223–237. Springer US, Boston, MA, 2003. ISBN 978-1-4615-0153-4. doi: 10.1007/978-1-4615-0153-4_11. URL https://doi.org/10.1007/978-1-4615-0153-4_11.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes, 2019.

Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4), 2019. ISSN 1935-8245. doi: 10.1561/2200000056. URL http://dx.doi.org/10.1561/2200000056.

Sheng Lu, Irina Bigoulaeva, Rachneet Sachdeva, Harish Tayyar Madabushi, and Iryna Gurevych. Are emergent abilities in large language models just in-context learning?, 2023.

Vladimir Mikulik, Grégoire Delétang, Tom McGrath, Tim Genewein, Miljan Martic, Shane Legg, and Pedro A. Ortega. Meta-trained agents implement bayes-optimal agents, 2020.

Aaron Mueller, Albert Webson, Jackson Petty, and Tal Linzen. In-context learning generalizes, but not always robustly: The case of syntax, 2023.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling, 2023.

Thomas M Norman, Max A Horlbeck, Joseph M Replogle, Alex Y Ge, Albert Xu, Marco Jost, Luke A Gilbert, and Jonathan S Weissman. Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365(6455):786–793, 2019.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Stuart J Russell and Peter Norvig. *Artificial intelligence a modern approach*. London, 2010. pages 737 and 757.

Alvaro Tejero-Cantero, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J. Gonçalves, David S. Greenberg, and Jakob H. Macke. Sbi – a toolkit for simulation-based inference, 2020.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models, 2023.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: A unified understanding of transformer's attention via the lens of kernel, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35151–35174. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/von-oswald23a.html.

Jane X. Wang, Michael King, Nicolas Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, Francis Song, Gavin Buttimore, David P. Reichert, Neil Rabinowitz, Loic Matthey, Demis Hassabis, Alexander Lerchner, and Matthew Botvinick. Alchemy: A benchmark and analysis toolkit for meta-reinforcement learning agents, 2021.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference, 2022.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization, 2023.

APPENDIX

A RELATED WORK

IN-CONTEXT LEARNING. ICL was first coined in the context of large language models, where it was observed that such models can generalize to unseen tasks with only a few context examples (hence *in-context*) without any additional training Brown et al. (2020). This definition was subsequently expanded to more controlled settings, where transformer models were explicitly trained on known latent variable model like linear regression (Von Oswald et al., 2023; Garg et al., 2023), hidden Markov models (Xie et al., 2022), compositional grammars (Hahn & Goyal, 2023), regular languages (Akyürek et al., 2024) and turing machines (Grau-Moya et al., 2024) over multiple different datasets (defined by the task latent), with the expectation that they would generalize to unseen datasets. Recent works highlight that when trained over a wide variety of datasets, these models do indeed generalize.

To understand ICL better, researchers have explored both empirical and theoretical frameworks to understand which architectures lead to this phenomena and how can it be explained. A convincing argument for its workings can be seen through the lens of Bayesian Inference, where the transformer implicitly models the posterior predictive distribution directly, given by:

$$p_{\boldsymbol{\theta}}(y_*|\boldsymbol{x}_*, \mathcal{D}) \tag{2}$$

where $\boldsymbol{x}_*$ denotes the query point and $\mathcal{D}$ represents the set of context examples / dataset in algorithmic tasks, and $\boldsymbol{\theta}$ represents the parameters of the transformer. Such a system is simply trained using maximum likelihood, which is formalized as:

$$\arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}_*,,y_*,\mathcal{D}} \log p_{\boldsymbol{\theta}}(y_*|\boldsymbol{x}_*, \mathcal{D}) \tag{3}$$

NEURAL PROCESSES. The problem of solving new tasks in a zero-shot manner directly at inference is also closely tied to meta-learning (Hospedales et al., 2020). Neural Processes (Garnelo et al., 2018b) provide a framework of performing probabilistic meta learning, where they model the problem as a latent-variable system. Given a dataset $\mathcal{D}$ and a query point $\boldsymbol{x}_*$, prediction of the label y according to this latent variable model can be compactly described as

$$p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_*, \mathcal{D}) = \int p_{\boldsymbol{\theta}}(y|\boldsymbol{x}, \boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}|\mathcal{D}) d\boldsymbol{z} \tag{4}$$

which looks akin to Variational Autoencoders (Kingma & Welling, 2019), with the sole difference being the amortization on the whole dataset $\mathcal{D}$ as opposed to single images, where $\boldsymbol{z}$ represents the latent variable and $\boldsymbol{\theta}$ the parameters of the likelihood model (eg. the decoder in VAEs). The model is trained via the Evidence Lower-Bound (ELBO) with the amortized variational approximation $q_\varphi(\cdot|\mathcal{D})$. Once trained, predictions for new datasets can be made by simply performing inference over the *encoder* q_φ to obtain $\boldsymbol{z}$, and then leveraging this latent variable to eventually give the predictions via $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_*, \boldsymbol{z})$.

Another direction of research, called Conditional Neural Process (Garnelo et al., 2018a), by-passes latent variable modeling by directly learning the posterior predictive distribution via the maximum likelihood objective, i.e.

$$p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_*, \mathcal{D}) = p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_*, \boldsymbol{z}_{\boldsymbol{\theta}}(\mathcal{D})) \tag{5}$$

where $\boldsymbol{z}_{\boldsymbol{\theta}}$ now just represents the output of a Neural Network without any probabilistic interpretation, and the parameters $\boldsymbol{\theta}$ are just trained via MLE.

This leads to essentially the same objective as ICL, with the difference coming in from how $z_{\boldsymbol{\theta}}$ is parameterized. Both CNPs and NPs employ a DeepSets architecture to model $z_{\boldsymbol{\theta}}$ or the variational distribution q_φ, to respect the permutation symmetries of the *iid* observations in $\mathcal{D}$. However, recent research generalizes this setting to use transformers Nguyen & Grover (2023) and other architectural backbones as well (Kim et al., 2019). In particular, ICL can be seen as an extension of CNPs which doesn't necessarily require the *iid* nature of the observations in modeling the predictions, and relies on the transformer architecture.

INDUCTIVE BIASES FOR ICL Inductive biases are tendencies models have towards learning certain types of solution, which will generalize well on some data distributions and inevitably poorly on some other (i.e. there is no free lunch, Goldblum et al. 2023). Olsson et al. (2022) have shown that *induction heads* play in important role in ICL by predicting that the continuation of a token will be the same as last time (i.e. $[a][b]\dots[a] \to [b]$). As this can happen at any layer of transformer, the basis of induction can be latent (Hahn & Goyal, 2023; Guo et al., 2023a); generally, this motif can be seen as very similar to kernel regression (i.e. $p(y_q|x_q, x_{1:n}) \propto \sum_i K(x_i, x_q)y_i$, (Han et al., 2023)). As shown by Von Oswald et al. (2023) this motif can be used to naturally do linear regression (potentially on non-linearly processed latents), making them generalize well on linear function Garg et al. (2023). In contrast (Hendel et al., 2023; Todd et al., 2023) have concurrently shown that in some cases transformers encode a "task vector" that they infer from the context and then use to do the prediction.

B Tasks

We consider the following tasks for our evaluations, specified by the function $g : (\mathbf{x}, \mathbf{z}) \to y$ is used to generate the ICL dataset.

B.1 Regression Tasks

For regression tasks, we use the mean-squarred-error loss to train the model.

B.1.1 Linear Regression

$y = g(\mathbf{x}; \mathbf{w}) = \boldsymbol{w}^T \boldsymbol{x} + \epsilon$ where $\mathbf{w} \in \mathbb{R}^n \sim \mathcal{N}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, 0.1)$.

B.1.2 Nonlinear Regression using MLPs

$y = g(\mathbf{x}, \psi) = f_\psi(\boldsymbol{x}) + \epsilon$, where f_ψ is modeled as a Multi-Layer Perceptron (MLP) network with weights ψ and $\epsilon \sim \mathcal{N}(0, 0.1)$. The MLP has a shape of $[1, 64, 1]$ and ReLU non-linearities.

B.1.3 Sinusoid Regression

$y = g(\mathbf{x}, \alpha_{1:k}, \lambda_{1:k}) = \sum_{i=1}^{K} \alpha_i \sin(2\pi\lambda_i \boldsymbol{x})$, where the parameters are frequencies $\lambda_i' \sim \mathcal{U}(0, 5)$ and amplitudes $\alpha_i' \sim \mathcal{U}(-1, 1)$ and $K = 3$.

B.1.4 Gaussian Process Regression

$\mathbf{Y} = g(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, K(\mathbf{X}, \mathbf{X}))$ where $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\right)$ is the RBF kernel and $\mathbf{X}, \mathbf{Y}$ are the matrices respectively containing all sampled points, both for context and queries. This has the effect of sampling a function at random, with the only structure being that the covariance between $\boldsymbol{y}$s depends on $K(\boldsymbol{x}, \boldsymbol{x})$. Here the z can be seen as $\boldsymbol{y}$ itself and is thus very high dimensional and weakly structured.

B.1.5 Hudgkin-Hoxley ODE Prediction

$y = g(\mathbf{t}, \bar{g}_{Na}, \bar{g}_K)$ is the solution of the following differential equation :

$$C_m \frac{dV}{dt} = g_1 (E_1 - V) + \bar{g}_{Na} m^3 h (E_{Na} - V) + \bar{g}_K n^4 (E_K - V) + \bar{g}_M p (E_K - V) + I_{inj} + \sigma\eta(t)$$

where all the other parameters are fixed to the same value as in Tejero-Cantero et al. 2020. We solve it for 6,400 pairs $(\bar{g}_{Na}, \bar{g}_K) \in [0, 40]^2$ from $t = 0$ to $t = 120$ with 1000 timesteps. The Hodgkin-Huxley ODE describes how action potentials in neurons are initiated and propagated in the brain.

B.2 Classification Tasks

For classification, we use a cross-entropy loss.

B.2.1 Linear Classification

$y = g(\boldsymbol{x}; \boldsymbol{w}, b) \sim \text{Categorical}(\text{Softmax}(\boldsymbol{w}^T \boldsymbol{x} + b))$ where $\boldsymbol{w} \in \mathbb{R}^2 \sim \mathcal{N}(0, 1)$ and $b \in \mathbb{R} \sim \mathcal{N}(0, 1)$

B.2.2 Nonlinear Classification Using MLPs

$y = g(\boldsymbol{x}; \phi) \sim \text{Categorical}(\text{Softmax}(f_\psi(\boldsymbol{x})))$ where where f_ψ is modeled as a Multi-Layer Perceptron (MLP) network with weights ψ and $\epsilon \sim \mathcal{N}(0, 0.1)$. The MLP has a shape of $[1, 64, 1]$ and ReLU non-linearities.

B.2.3 Alchemy

Alchemy is a meta-reinforcement learning benchmark Wang et al. (2021) where each environment is defined by a set $\mathbf{z} = (\text{Graph, Potion map, Stone map})$ of rules about how some set of potions transforms some stones. We extracted from it an ICL classification dataset consisting of transformations $\mathbf{x} = (\text{stone, potion}) \rightarrow \mathbf{y} = \text{stone}$. The transformations are compositional and symbolic; each potion affects only one of the three properties of stones (size, shape and color). An environment is specified by how observable stones and potions MAP to latent stones and potions, along with a GRAPH over these latent stones which specify the result of the Transformations. In total there is 109 Graph, 48 Potion map and 32 Stone maps, making for 167424 environments. We reserve 100,000 environments for evaluation and train of the remaining ones.

B.3 Real-World Tasks

B.3.1 Raven's Progressive Matrices

Raven's Progressive Matrices (Raven's PM) is a reasoning task used for IQ tests (John & Raven, 2003). It consists of a 3x3 grid where each cell contains simple objects varying in a small number of attributes (e.g., shape, size, number), but the bottom right cell is left empty. Subjects must notice a pattern in how the cells change from left to right in the first two rows of the grid, and then use that same pattern to complete missing cell in the bottom row. This is done by selecting one answer among N possible provided options for the missing cell. We use a symbolic version of the dataset that addresses bias in the original version (Guo et al., 2023b). In our models, the context consists of the first two rows of the grid, the query consists of the last row with a masked out final cell, and the ground-truth latent variable is the underlying rule that generates a particular grid. Each rule is composed of a set of sub-parts, and we evaluate on unseen compositions.

B.3.2 Gene Targeting

We use Perturb-seq dataset collected by Norman et al. (2019) where researchers performed several genetic intervention experiments using CRISPR (Gilbert et al., 2014). In each experiment, either one or two genes were targeted and the resulting expressions across 5000 genes were observed across several cells. Here, we consider each CRISPR intervention experiment as a different context, the resulting cell genetic expressions as 5000-dimensional observations, and a left-out cell with half of the genetic expressions randomly masked out as the query. The task is to predict the missing genetic expressions for the queried cell. We evaluate on held on held out CRISPR experiments with novel pairs of targeted genes.

C Model Details

For our implicit model, we use a standard transformer with 8 layers. In the explicit model, for context aggregation we parameterize $\boldsymbol{z}_\varphi(\mathcal{D})$ using a standard transformer with 4 layers, 256 dimensions latent, 512 dimensions MLP and 4 heads. For the predictor $p_{\boldsymbol{\theta}}$, we consider two options: a ReLU MLP with three hidden layers of size 512 and a transformer with the same configuration as $\boldsymbol{z}_\varphi(\mathcal{D})$.

For the implicit model, we format the prompt for prediction as $[x_1, y_1] \ldots [x_n, y_n][x_q, \emptyset]$, where every $[\cdot]$ represents a token. We use a distinct mask token $\emptyset$ to represent the target (which is the thing being predicted). For the explicit model, we first compute $[x_1, y_1] \dots [x_n, y_n]$ to $\boldsymbol{z}_\varphi(\mathcal{D})$ with the context transformer. Then we give $[\boldsymbol{z}_\varphi(\mathcal{D})][x_q]$ to predictor transformer or $[\boldsymbol{z}_\varphi(\mathcal{D}), x_q]$ to the MLP.

C.1 Distributed Alignment Search details

To find subspace causally associated with a task latent in Alchemy, we use a method base on Distributed Alignment Search (DAS) by Geiger et al. (2023). This procedure is performed for a location $L = \mathbb{R}^d$ (e.g. the bottleneck) and latent $i \in \{1, 2, 3\}$ (Graph, Stone map, Potion map).

First, we run with the model on $\mathcal{D}_z$ and $\mathcal{D}_{\bar{z}}$ for every possible query x_*. We call z the base and $\bar{z}$ the source and only differ by the ith latent. For every run, we record the activity of the source model at the location $l_z \in \bar{L}$. Then, we run the base model again but this time fixing the subspace of l defined by the orthogonal projection $\Pi \in \mathbb{R}^{d\times 10}$ to it's value in l_z. A single projection Π is learned over all possible combination $z, \bar{z}$ and x_* with a cross-entropy loss between the prediction of the base (intervened) model and the true counter-factual result of changing the latent z_i to $\bar{z}_i$. See Fig. 4 for an illustration of the process. A subspace is evaluated by looking at the accuracy of the counterfactual interventions over a dataset of held-out $z, \bar{z}$ pairs; giving the validation Interchange Intervention Accuracy (IIA). In Figure Fig. 3 (c) we report the baseline-adjusted validation IIA $\frac{\text{IIA}-\text{BASELINE}}{1-\text{BASELINE}}$ where BASELINE is the counterfactual accuracy if we don't perform any intervention (as often the intervention doesn't change the prediction on a specific x_*).

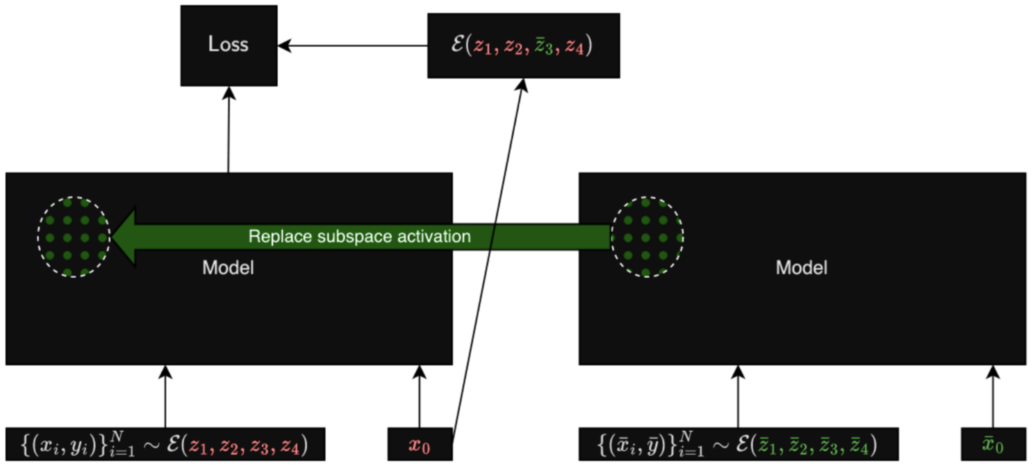
Loss
$\mathcal{E}(z_1, z_2, \bar{z}_3, z_4)$
Replace subspace activation
Model
Model
$\{(x_i, y_i)\}_{i=1}^N \sim \mathcal{E}(z_1, z_2, z_3, z_4)$
x_0
$\{(\bar{x}_i, \bar{y})\}_{i=1}^N \sim \mathcal{E}(\bar{z}_1, \bar{z}_2, \bar{z}_3, \bar{z}_4)$
$\bar{x}_0$

Figure 4: Illustration of the DAS training procedure