Estimating Interventional Distributions with Uncertain Causal Graphs through Meta-Learning

Anish Dhir [*1] **Cristiana Diaconu** [*2] **Valentinian Mihai Lungu** [2] **James Requeima** [3 4] **Richard E. Turner** [2]
Mark van der Wilk [5]

Abstract

In scientific domains—from biology to social sciences—many questions boil down to *What effect will we observe if we intervene on a particular variable?* If the causal relationships (e.g. a causal graph) are known, it is possible to estimate the intervention distributions. Without this domain knowledge, the causal structure must be discovered from available observational data. However, observational data are often compatible with multiple causal graphs, making methods that commit to a single structure prone to overconfidence. A principled way to manage this structural uncertainty is via Bayesian inference, which averages over a posterior distribution of possible causal structures and functional mechanisms. Unfortunately, the number of causal structures grows super-exponentially with the number of nodes in the graph, making computations intractable. We circumvent this intractability by using meta-learning to create an end-to-end model: the Model-Averaged Causal Estimation Transformer Neural Process (MACE-TNP). The model is trained to predict the Bayesian model-averaged interventional posterior distribution, and its end-to-end nature bypasses the need for expensive calculations. Empirically, we show MACE-TNP outperforms strong baselines, establishing meta-learning as a flexible and scalable paradigm for approximating complex Bayesian causal inference.

[*]Equal contribution [1]Imperial College London [2]University of Cambridge [3]University of Toronto [4]Vector Institute [5]University of Oxford. Correspondence to: Anish Dhir <anish.dhir13@imperial.ac.uk>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

Answering interventional questions such as: "What happens to Y when we change X?" is central to areas such as healthcare (Little and Rubin, 2000) and economics (Spiegler, 2020). One can estimate such *interventional distributions* by actively intervening on the variable of interest and observing the effects (obtaining *interventional data*), but this can be costly, difficult, unethical, or even impossible in practice (Li et al., 2019). Causal inference offers an alternative by leveraging readily available *observational data* alongside knowledge of the underlying causal relationships in the form of a causal graph (Pearl, 2009). A causal graph can be manually specified when domain knowledge is available. In the absence of this, causal discovery techniques attempt to learn the causal structure from data (Mooij et al., 2016). However, causal discovery from purely observational data is notoriously difficult. It is often the case that data provides plausible evidence for a set of causal graphs, even though each of these graphs may imply drastically different causal effects. Picking a single graph can thus result in poor downstream decisions (Pearl, 2014; Bellot, 2024). In this work, we address the challenge of tractably estimating interventional distributions when the true causal graph is uncertain, a common scenario in real-world applications.

Instead of using a single graph to drive decisions, the Bayesian framework provides a principled way to manage the uncertainty over the causal models through a posterior distribution over possible causal structures and functions relating variables. However, this procedure has two main challenges. First, the space of causal graphs grows super-exponentially with the number of variables, making exact posterior inference intractable, and sampling difficult to scale. Second, even with a posterior over causal graphs, estimating an interventional distribution in each plausible causal graph necessitates computing the posterior over functional mechanisms, which is analytically intractable except in simple models. Poor approximations at any point in this pipeline can result in inaccurate interventional distributions. Consequently, most works restrict themselves to simple functional mechanisms and constrain the allowable structures, limiting their applicability.

To overcome these bottlenecks, we turn to recent advances in meta-learning. Neural processes (NPs) (Garnelo et al., 2018a;b) are a family of meta-learning models that approximate the Bayesian posterior with guarantees (Gordon et al., 2019), by *directly* mapping from datasets to the predictive distribution of interest, thus bypassing the intractable explicit modelling of intermediate posteriors. Recently, NPs have been applied to causal discovery (Ke et al., 2023; Lorch et al., 2022; Dhir et al., 2024b), with Dhir et al. (2025) showing that they can accurately recover posterior distributions over causal graphs. However, these existing approaches cannot estimate interventional distributions.

In this work, we apply NPs to the problem of causal inference directly from data by developing a meta-learning framework that targets Bayesian posteriors for interventional queries—the Model-Averaged Causal Estimation Transformer Neural Process (MACE-TNP). Our method amortises the full Bayesian causal inference pipeline (learning the posterior over the causal structure and functions, and marginalising over them), all within a single model. By directly estimating the interventional distribution, our approach avoids compounding errors from approximations of posteriors and marginalisation, enabling more accurate and computationally-efficient inference under model uncertainty. Our contributions are twofold. First, we propose an end-to-end model trained on synthetic datasets to directly approximate the Bayesian posterior interventional distribution. Second, we demonstrate that MACE-TNP outperforms a range of Bayesian and non-Bayesian baselines across experimental settings of increasing complexity, highlighting the method's potential to scale to high-dimensional datasets. Our framework paves the way for meta-learning-based foundation models for estimating interventions.

2. Background

Our goal is to compute interventional distributions that account for uncertainty in both the causal structure and functional mechanisms. We assume no hidden confounders, aligning with assumptions in causal discovery. We discuss related work in Appendix A.

Bayesian causal inference: Since observational data often identify only a class of indistinguishable graphs, each implying different interventions, uncertainty is inherent in causal inference. The Bayesian framework allows for quantifying the model uncertainty, both in the causal structure and functions, and use it for downstream decision making.

Definition 2.1. We define a *Bayesian causal model* (BCM) as the following hierarchical Bayesian model over a directed acyclic graph (DAG) $\mathcal{G}$ with node set $V = \{1, \dots, D\}$, functions $\mathbf{f} = \{f_1, \dots, f_D\}$, and an observational dataset $\mathcal{D}_{\text{obs}}$ of N_{obs} samples: $\mathcal{G} \sim p_{\text{BCM}}(\mathcal{G})$, $\mathbf{f} \sim p_{\text{BCM}}(\mathbf{f}|\mathcal{G})$, $\mathcal{D}_{\text{obs}} := \{\mathbf{X}^n\}_{n=1}^{N_{\text{obs}}} \sim \prod_{n=1}^{N_{\text{obs}}} \prod_{i \in V} p_{\text{BCM}}(x_i^n | f_i, \mathcal{G})$, where x_i^n denotes the i-th node of the n-th observational sample. This implies the joint distribution $p_{\text{BCM}}(\mathcal{D}_{\text{obs}}, \mathbf{f}, \mathcal{G})$.

As BCMs are defined with a causal graph, they induce a distribution over interventional quantities as well. Interventions $p_{\text{BCM}}(x_i | \text{do}(x_j), f_i, \mathcal{G})$ can be computed by setting $f_j(\cdot) = x_j$ and leaving all other mechanisms unchanged.

Given a dataset $\mathcal{D}_{\text{obs}}$, our task is to estimate an interventional distribution of interest. This requires inferring the possible graphs and functional mechanisms that generated the dataset. The Bayesian answer to this question is through the posterior $p_{\text{BCM}}(\mathbf{f}, \mathcal{G} \mid \mathcal{D}_{\text{obs}}) \propto p_{\text{BCM}}(\mathcal{D}_{\text{obs}} \mid \mathbf{f}, \mathcal{G}) p_{\text{BCM}}(\mathbf{f} \mid \mathcal{G}) p_{\text{BCM}}(\mathcal{G})$. If the underlying model is identifiable, for example by restricting the function class of $\mathbf{f}$ (Peters et al., 2017, Ch. 4), then, under suitable conditions[1], the posterior over $\mathcal{G}$ concentrates on the true graph in the infinite data limit (Dhir et al., 2024a;b; Chickering, 2002). However, for finite data, or if the causal model is not identifiable, the posterior quantifies the uncertainty over causal graphs.

To make use of the uncertainty, Bayes prescribes to average the interventions over the plausible models (Madigan and Raftery, 1994), which we call the *posterior interventional distribution* $p_{\text{BCM}}(x_i \mid \text{do}(x_j), \mathcal{D}_{\text{obs}})$

$$\sum_{\mathcal{G}} \int p_{\text{BCM}}(x_i | \, \text{do}(x_j), \mathbf{f}, \mathcal{G}) p_{\text{BCM}}(\mathbf{f}, \mathcal{G} | \mathcal{D}_{\text{obs}}) \text{d}\mathbf{f} \quad (1)$$

Computing the above quantity is often intractable for two main reasons: 1) computing $p_{\text{BCM}}(\mathcal{G}|\mathcal{D}_{\text{obs}})$ is challenging as the number of causal graphs increases super-exponentially with the number of variables, 2) $p_{\text{BCM}}(\mathbf{f}|\mathcal{G}, \mathcal{D}_{\text{obs}})$ is only tractable for simple models.

3. A transformer model for meta-learning causal inference

Causal inference with neural processes: This work focuses on causal inference directly from data—predicting the distribution of a variable of interest X_i under an intervention $\text{do}(x_j)$ given access to only $\mathcal{D}_{\text{obs}}$. We *directly* learn the map from $\mathcal{D}_{\text{obs}}$ to $p_{\text{BCM}}(x_i \mid \text{do}(x_j), \mathcal{D}_{\text{obs}})$ in an end-to-end fashion with NPs. To do this, we minimise, with respect to the model parameters θ, the expected Kullback-Leibler (KL) divergence over the tasks $\xi := (\mathcal{D}_{\text{obs}}, i, j, X_j)$ between the true posterior interventional distribution and the NP model predictions $p_\theta(x_i|\text{do}(x_j), \mathcal{D}_{\text{obs}})$:

$$\begin{aligned} &\mathbb{E}_\xi \left[\text{KL}(p_{\text{BCM}}(X_i | \, \text{do}(X_j), \mathcal{D}_{\text{obs}}) \| p_\theta(X_i | \, \text{do}(X_j), \mathcal{D}_{\text{obs}})) \right] \\ &= \left[\mathbb{E}_{X_i|\xi} \left[\log p_\theta(X_i | \, \text{do}(X_j), \mathcal{D}_{\text{obs}}) \right] \right] + C, \end{aligned} \quad (2)$$

where $\xi \sim p(\mathcal{D}_{\text{obs}}, i, j, x_j)$, $X_i | \xi \sim p_{\text{BCM}}(x_i | \text{do}(X_j), \mathcal{D}_{\text{obs}})$, and C is some constant independent of θ.

[1] The true generating process lies within the support of the prior.

Table 1. Results for MACE-TNP and baselines on the three-node experiments. We show the NLPID (↓) and report the mean ± the error of the mean over 100 datasets. Each row corresponds to a different functional mechanism used in the test set (GP or NN). A separate MACE-TNP model was trained for each mechanism.

	MACE-TNP	DiBS-GP	ARCO-GP	BCI-GPN	DECI	NOGAM+GP
GP	$\mathbf{563.9 \pm 23.4}$	644.2 ± 27.2	630.7 ± 22.3	628.5 ± 27.5	632.0 ± 25.6	749.4 ± 43.0
NN	$\mathbf{527.9 \pm 19.8}$	807.6 ± 50.1	851.2 ± 55.0	706.8 ± 5.0	588.0 ± 23.6	815.6 ± 58.4

Hence, our objective requires us to generate tasks, and interventional data from BCMs, in order to find the optimal parameters. To do this we, 1) sample a graph $\mathcal{G} \sim p_{\text{BCM}}(\mathcal{G})$, and 2) a functional mechanism for each of the D variables from the graph $\mathbf{f} \sim p_{\text{BCM}}(\mathbf{f}|\mathcal{G})$. Conditioned on the sampled graph $\mathcal{G}$ and functional mechanism $\mathbf{f}$ we then 3) draw N_{obs} samples for each variable to construct the observational data $\mathcal{D}_{\text{obs}} \sim P_{\text{BCM}}(\mathcal{D}_{\text{obs}}|\mathbf{f}, \mathcal{G})$. To construct the interventional data, keeping the same graph and functions as the observational data, we 4) randomly sample a variable index j to intervene upon and N_{int} intervention values $\mathbf{X}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, set the values of node j to be $\mathbf{x}_j$, and 5) draw N_{int} samples of each node forming an interventional dataset $\mathcal{D}_{\text{int}} \sim p_{\text{BCM}}(\mathcal{D}_{\text{int}}|\text{do}(\mathbf{x}_j), \mathbf{f}, \mathcal{G})$. Finally we sample an outcome node index i and extract samples of $p_{\text{BCM}}(\mathbf{x}_i|\text{do}(\mathbf{x}_j), f_i, \mathcal{G})$ from $\mathcal{D}_{\text{int}}$.

Model architecture and desirable properties: Given we are interested in predicting $p_{\text{BCM}}(x_i|\text{do}(\mathbf{x}_j), \mathcal{D}_{\text{obs}})$, variables play distinct roles as either the outcome node X_i, the intervening node X_j, or the nodes that are being marginalised. The distribution of interest also satisifies certain symmetries. For example, the interventional distribution is *permutation-invariant* with respect to observational samples and *permutation-equivariant* with respect to interventional samples. Thus, properties of this distribution, and the role of the variables, guide our architecture choice. An architecture flexible enough to satisfy these desiderata is the transformer (Vaswani et al., 2017; Lee et al., 2019). As interventional distributions can be non-Gaussian even in simple cases, we opt for a Mixture of Gaussians (MoG) representation of $p_\theta(x_i|\text{do}(x_j), \mathcal{D}_{\text{obs}})$ (Bishop, 1994). More details on the target distribution's constraints and model components are given in Appendix B.

Recovery of exact prediction map: Bruinsma (2022, Proposition 3.26) shows that, in the limit of infinite tasks and model capacity, the global maximum of Equation (2) is achieved if and only if the model exactly learns the map $(\mathcal{D}_{\text{obs}}, \mathbf{x}_j) \mapsto p_{\text{BCM}}(x_i|\text{do}(\mathbf{x}_j), \mathcal{D}_{\text{obs}})$ (Gordon et al., 2019). While the constraint of infinite tasks is limiting when applying NP to real-world datasets, if the tasks are generated through a known Bayesian causal model, we have in theory access to an infinite amount of tasks.

4. Experiments

We evaluate the performance of our model, MACE-TNP, against Bayesian causal inference baselines, and a causal discovery method that selects a single graph. With our experiments we aim to answer: 1) How does our model compare against baselines when the baselines' assumptions are *respected*, 2) How does our model perform when the number of nodes are scaled, 3) How does our model perform when we do not have knowledge of the data-generating process? Moreover, in Appendix F we empirically show that, in cases where the true posterior is analytically tractable, that MACE-TNP converges to it.

To train MACE-TNP, we randomise the number of observational samples $N_{\text{obs}} \sim \mathcal{U}\{50, 750\}$, and set $N_{\text{int}} = 1000 - N_{\text{obs}}$. The training loss is evaluated on these N_{int} samples. For testing, we sample 500 observation points and compute the loss against 500 intervention points.

Baselines: We benchmark against methods that infer distributions over causal graphs and sample to marginalise across these graphs when estimating posterior interventional distributions. DiBS-GP (Toth et al., 2022), ARCO-GP (Toth et al., 2025), and BCI-GPN (Giudice et al., 2024) all use GP networks, but differ in the inference procedure over graphs. We also compare against DECI (Geffner et al., 2022), which assumes additive noise and uses autoregressive neural networks to learn a distribution over causal graphs while only learning point estimates for functions. Finally, to show that learning a distribution over graphs is useful, we compare against a non-Bayesian baseline that uses NOGAM (Montagna et al., 2023) to infer a single DAG, and estimates the interventional distribution by using GPs: NOGAM-GP.

Metrics: We report the negative log-posterior interventional density (NLPID) of the true intervention outcomes under the model (Gelman et al., 2014): $-\mathbb{E}_{X_j \sim \mathcal{N}(0,1)}[\mathbb{E}_{p_{\text{BCM}}(x_i|\text{do}(x_j), \mathcal{D})}[\log p_\theta(X_i|\text{do}(X_j), \mathcal{D})]]$.

4.1. Three-node experiments

First, we test our model in a three-node setting, where there are 25 graphs in total, making inference over the graph easier than in higher-node settings. Given that most baselines either use neural networks or GPs, we compare MACE-TNP to the baselines under two scenarios: 1) when tested on

Table 2. Results for MACE-TNP and baselines on the higher dimension experiment. D represents the number of variables used. We show the NLPID (↓) and report the mean ± the error of the mean over 100 datasets.

D	**MACE-TNP**	**DiBS-GP**	**ARCO-GP**	**DECI**	**NOGAM+GP**
20	$\mathbf{660.4 \pm 5.2}$	701.9 ± 4.0	701.9 ± 4.0	686.3 ± 6.7	942.7 ± 23.8
30	$\mathbf{653.3 \pm 5.7}$	713.2 ± 4.7	713.0 ± 4.7	675.6 ± 6.1	946.9 ± 19.2
40	$\mathbf{665.8 \pm 4.8}$	711.5 ± 4.6	712.1 ± 4.6	683.0 ± 5.1	986.0 ± 20.0

GP data, and 2) when tested on data generated using neural networks (NN). For each functional mechanism, we train a separate MACE-TNP model. Full experimental details are provided in Appendices C.1 and D.1.

When tested in-distribution (on datasets from the same distribution the model was trained on), MACE-TNP consistently outperforms all baselines across both functional mechanisms, as shown in Table 1. MACE-TNP outperforms GP-based methods through its implicit handling of hyperparameter inference, that the GP baselines may struggle with. It also surpasses DECI (an NN-based approach) on both GP and NN data by employing a Bayesian treatment over functions. This highlights a key advantage of MACE-TNP: it can easily incorporate complex Bayesian causal models into its training pipeline by sampling training datasets, whereas traditional Bayesian methods rely on inadequate approximations for complex scenarios. While the baseline models will face scenarios where they are misspecified (for example, the GP-based methods on the NN dataset), MACE-TNP trivially allows to train on more diverse data to learn a mixture of priors and thus avoid misspecification. We show in Appendix E.1 that this strategy improves MACE-TNP's generalisation capabilities, and provide additional results for different architecture choices.

4.2. Higher dimensional experiment

We next investigate the scalability of our method. We do so by testing a single trained model on increasingly higher-dimensional data, scaling up from 20 up to 40 nodes. The functional mechanisms used are a mix of NNs and functions drawn from a GP, with an additional latent variable input to ensure the final distribution is non-Gaussian. More details on data generation are given in Appendix C.2.

Table 2 shows that MACE-TNP outperforms both the Bayesian and non-Bayesian baselines across all node sizes. Moreover, similarly to the three-node case, the underperformance of the non-Bayesian baseline NOGAM+GP underscores the importance of capturing uncertainty over causal structures. The majority of our baselines involve GP-based approaches, which can become prohibitively expensive with a higher number of variables. For example, we do not report BCI-GPN as its MCMC scheme is too expensive for these node sizes. In contrast, MACE-TNP can readily leverage advancements that have made neural network architectures scale favourably in other domains. Inference after training only requires a forward pass through the network.

4.3. Unknown dataset generation process

Finally, we apply our method on the Sachs proteomics dataset (Sachs et al., 2005), which includes measurements of $D = 11$ proteins from thousands of cells under various molecular interventions. Crucially, we do not retrain any model for this task; instead, we reuse the model from Section 4.2, which was trained exclusively on synthetic data. Following (Brouillard et al., 2020; Lippe et al., 2022; Wang et al., 2017), we only retain samples with interventions targeting one of the $D = 11$ proteins, and take 500 observational sample and test five single-protein perturbations on 500 interventional samples. The results indicate that our method performs competitively with our strongest baseline, giving an NLPID for MACE-TNP of 998.9 ± 104.9 compared to 1000.9 ± 133.5 for DECI, when averaged over 5 interventional queries and 10 outcome nodes. We provide additional comparisons to the other baselines in Appendix E.2. This shows the potential of tackling interventional queries in real-world settings with a fast, data-driven framework that captures uncertainty in a principled manner and leverages flexible and expressive neural network architectures.

5. Conclusions

We address the challenge of efficiently estimating interventional distributions when the causal graph structure is unknown. Our solution, MACE-TNP, is an end-to-end meta-learning framework that directly approximates the Bayesian model-averaged interventional distribution by mapping observational data to posterior interventional distributions. When employing complex functional mechanisms, as well as high-dimensional data (up to 40 nodes), MACE-TNP outperforms strong Bayesian and non-Bayesian baselines, with the only requirement being access to samples from a prior distribution (implicit or explicit) at meta-train time. We also show competitive performance on a challenging, real-life dataset, showcasing the practical implications of our flexible, end-to-end framework. One limitation of our model is its reliance on substantial training-time compute and real or synthetic data.

References

Matthew Ashman, Cristiana Diaconu, Eric Langezaal, Adrian Weller, and Richard E. Turner. Gridded transformer neural processes for large unstructured spatio-temporal data, 2024. URL https://arxiv.org/abs/2410.06731.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439), 1999.

Alexis Bellot. Towards bounding causal effects under markov equivalence. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, 2024.

Christopher M Bishop. Mixture density networks. 1994.

P. Brouillard, S. Lachapelle, A. Lacoste, S. Lacoste-Julien, and A. Drouin. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, volume 33, pages 21865–21877, 2020.

Wessel P. Bruinsma. *Convolutional Conditional Neural Processes*. PhD thesis, Department of Engineering, University of Cambridge, 2022. URL https://www.repository.cam.ac.uk/handle/1810/354383.

Lucius E. J. Bynum, Aahlad Manas Puli, Diego Herrero-Quevedo, Nhi Nguyen, Carlos Fernandez-Granda, Kyunghyun Cho, and Rajesh Ranganath. Black box causal inference: Effect estimation via meta prediction, 2025. URL https://arxiv.org/abs/2503.05985.

Federico Castelletti and Guido Consonni. Bayesian inference of causal effects from observational data in gaussian graphical models. *Biometrics*, 77(1), 2021.

Bertrand Charpentier, Simon Kibler, and Stephan Gunnemann. Differentiable dag sampling. In *International Conference on Learning Representations*, 2022.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov), 2002.

Anish Dhir, Samuel Power, and Mark van der Wilk. Bivariate causal discovery using bayesian model selection. In *Forty-first International Conference on Machine Learning*, 2024a.

Anish Dhir, Ruby Sedgwick, Avinash Kori, Ben Glocker, and Mark van der Wilk. Continuous bayesian model selection for multivariate causal discovery. *arXiv preprint arXiv:2411.10154*, 2024b.

Anish Dhir, Matthew Ashman, James Requeima, and Mark van der Wilk. A meta-learning approach to bayesian causal discovery. In *The Thirteenth International Conference on Learning Representations*, 2025.

Nir Friedman and Daphne Koller. Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning*, 50, 2003.

Nir Friedman and Iftach Nachman. Gaussian process networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 2000.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional Neural Processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Tomas Geffner, Javier Antoran, Adam Foster, Wenbo Gong, Chao Ma, Emre Kiciman, Amit Sharma, Angus Lamb, Martin Kukla, Agrin Hilmkil, et al. Deep end-to-end causal inference. In *NeurIPS 2022 Workshop on Causality for Real-world Impact*, 2022.

Dan Geiger and David Heckerman. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *The Annals of Statistics*, 30(5), 2002.

Andrew Gelman, Jessica Hwang, and Aki Vehtari. Understanding predictive information criteria for bayesian models. *Statistics and computing*, 24, 2014.

Enrico Giudice, Jack Kuipers, and Giusi Moffa. Bayesian causal inference with gaussian process networks. *arXiv preprint arXiv:2402.00623*, 2024.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2019.

David Heckerman. A bayesian approach to learning causal networks. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 1995.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023.

Nan Rosemary Ke, Silvia Chiappa, Jane X Wang, Jorg Bornschein, Anirudh Goyal, Melanie Rey, Theophane Weber, Matthew Botvinick, Michael Curtis Mozer, and Danilo Jimenez Rezende. Learning to induce causal structure. In *International Conference on Learning Representations*, 2023.

Mikko Koivisto and Kismat Sood. Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Research*, 5(May), 2004.

Jack Kuipers and Giusi Moffa. Partition mcmc for inference on acyclic digraphs. *Journal of the American Statistical Association*, 112(517), 2017.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*. PMLR, 2019.

Haichao Li, Kun Wu, Chenchen Ruan, Jiao Pan, Yujin Wang, and Hongan Long. Cost-reduction strategies in massive genomics experiments. *Marine Life Science & Technology*, 1, 2019.

P. Lippe, L. von Rueden, J. Bühler, B. Schölkopf, I. Gurevych, and J.M. Mooij. Amortized inference for causal structure learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 1737–1751, 2022.

Roderick J Little and Donald B Rubin. Causal effects in clinical and epidemiological studies via potential outcomes: concepts and analytical approaches. *Annual review of public health*, 21(1), 2000.

Lars Lorch, Jonas Rothfuss, Bernhard Schölkopf, and Andreas Krause. Dibs: Differentiable bayesian structure learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Lars Lorch, Scott Sussex, Jonas Rothfuss, Andreas Krause, and Bernhard Schölkopf. Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35, 2022.

David Madigan and Adrian E Raftery. Model selection and accounting for model uncertainty in graphical models using occam's window. *Journal of the American Statistical Association*, 89(428), 1994.

David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International statistical review/revue internationale de statistique*, 1995.

Divyat Mahajan, Jannes Gladrow, Agrin Hilmkil, Cheng Zhang, and Meyer Scetbon. Zero-shot learning of causal models. In *NeurIPS 2024 Causal Representation Learning Workshop*, 2024.

Francesco Montagna, Nicoletta Noceti, Lorenzo Rosasco, Kun Zhang, and Francesco Locatello. Causal discovery with score matching on additive models with arbitrary noise. In *Conference on Causal Learning and Reasoning*. PMLR, 2023.

Joris M Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing cause from effect using observational data: methods and benchmarks. *Journal of Machine Learning Research*, 17 (32), 2016.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022.

Tung Nguyen and Aditya Grover. Transformer Neural Processes: Uncertainty-Aware Meta Learning Via Sequence Modeling. In *International Conference on Machine Learning*. PMLR, 2022.

Teppo Niinim, Pekka Parviainen, Mikko Koivisto, et al. Structure discovery in bayesian networks by sampling partial orders. *Journal of Machine Learning Research*, 17 (57), 2016.

Weronika Ormaniec, Scott Sussex, Lars Lorch, Bernhard Schölkopf, and Andreas Krause. Standardizing structural causal models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Judea Pearl. Interpretation and identification of causal mediation. *Psychological methods*, 19(4), 2014.

J. Peters and P. Bühlmann. Identifiability of gaussian structural equation models with equal error variances. *Biometrika*, 101:219–228, November 2013. ISSN 0006-3444.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data.

Science, 308(5721):523–529, 2005. doi: 10.1126/science.1105809.

Andreas Sauter, Saber Salehkaleybar, Aske Plaat, and Erman Acar. Activa: Amortized causal effect estimation without graphs via transformer-based variational autoencoder, 2025. URL https://arxiv.org/abs/2503.01290.

Ran Spiegler. Can agents with causal misperceptions be systematically fooled? *Journal of the European Economic Association*, 18(2), 2020.

Marc Teyssier and Daphne Koller. Ordering-based search: a simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 2005.

Christian Toth, Lars Lorch, Christian Knoll, Andreas Krause, Franz Pernkopf, Robert Peharz, and Julius Von Kügelgen. Active bayesian causal inference. *Advances in Neural Information Processing Systems*, 35, 2022.

Christian Toth, Christian Knoll, Franz Pernkopf, and Robert Peharz. Effective bayesian causal inference via structural marginalisation and autoregressive orders. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

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.

Jussi Viinikka, Antti Hyttinen, Johan Pensar, and Mikko Koivisto. Towards scalable bayesian learning of causal dags. *Advances in Neural Information Processing Systems*, 33, 2020.

Y. Wang, L. Solus, K. Yang, and C. Uhler. Permutation-based causal inference algorithms with interventions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jiaqi Zhang, Joel Jennings, Agrin Hilmkil, Nick Pawlowski, Cheng Zhang, and Chao Ma. Towards causal foundation model: on duality between optimal balancing and attention. In *Forty-first International Conference on Machine Learning*, 2024.

A. Related Work

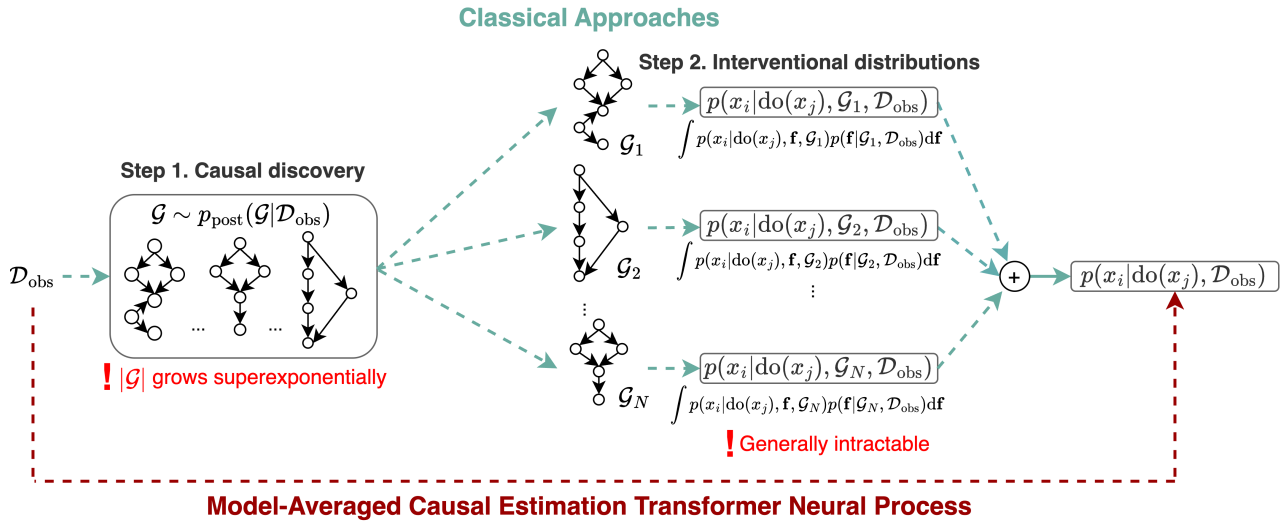

Figure 1. Overview of MACE-TNP. Unlike classical approaches, that usually require a two-step procedure which 1) first involves posterior inference over the graph structure, followed by 2) complicated inference over the functional mechanism, MACE-TNP amortises the full causal inference pipeline.

Estimating the posterior interventional distribution (Equation (1)) is challenging. The dominant paradigm involves a two-stage process: 1) obtaining samples from the high-dimensional posterior over graphs, and 2) estimating the interventional distribution under each sampled DAG, followed by averaging the result. Figure 1 provides a diagram illustrating these two steps. Although principled, this process faces computational challenges in both stages.

The first stage is challenging due to the super-exponential size of the space of DAGs. Early score-based methods addressed this by leveraging score equivalence, allowing search over the small space of MECs instead of individual DAGs. To achieve this, they used restricted the model family to linear Gaussians, with specific priors to make the scores analytically tractable (Heckerman, 1995; Geiger and Heckerman, 2002). To accommodate broader model classes, Madigan et al. (1995) introduced an MCMC scheme over the space of DAGs. However, the large space of DAGs leads to slow mixing and convergence issues, limiting the number of effective posterior samples (Friedman and Koller, 2003; Koivisto and Sood, 2004; Teyssier and Koller, 2005; Niinim et al., 2016; Kuipers and Moffa, 2017). A common bottleneck in these approaches is that scoring the proposed structures at each MCMC step requires expensive marginal likelihood estimation. This is often mitigated through reducing the graph space by restricting the in-degree of each node. Variational inference (VI) offers a cheaper alternative (Charpentier et al., 2022) but struggles to capture multi-modal posteriors inherent in causal discovery (Toth et al., 2025, Sec. 3.1), and can still have a demanding computational costs (e.g. SVGD used in (Lorch et al., 2021) scales quadratically with samples). Crucially, any inaccuracies or biases in this stage affects the downstream estimation in the second stage.

The second stage—averaging over the posterior of causal graphs—has its own significant computational burden. It requires performing inference with a potentially complex functional model for every single DAG sampled from the approximate posterior $p(\mathcal{G}|\mathcal{D}_{\text{obs}})$. As a result, previous work has only considered simple functional models where the inference is not too prohibitive. While early work in simple settings like linear Gaussian models allowed for closed-form averaging (Viinikka et al., 2020; Castelletti and Consonni, 2021), recent works often employ Gaussian Process (GP) networks (Friedman and Nachman, 2000) where this is not possible. To tackle this, Giudice et al. (2024) use complex MCMC schemes for both hyperparameter posteriors of the GPs and graph sampling, but have to resort to approximating the final interventional posterior distribution with a Gaussian for computational tractability. Toth et al. (2022; 2025) also use GP networks but use the cheaper alternative of using MAP estimates for hyperparameters. However, both ultimately rely on the expensive process of estimating interventions by sampling from the GP posterior conditional on each DAG. Hence, despite variations, the core limitation of expensive inference persists across these approaches, especially prohibiting the use of more flexible function model classes.

In contrast to this explicit two-stage procedure, we propose leveraging NPs (Garnelo et al., 2018b) to directly learn an

estimator for the target interventional distribution conditional on the observational data $\mathcal{D}_{\text{obs}}$. Our approach aims to learn a mapping $\mathcal{D}_{\text{obs}} \mapsto p(y|\text{do}(x), \mathcal{D}_{\text{obs}})$ that does not require approximating potentially problematic intermediate quantities. This effectively amortises the complex inference and averaging procedure over the training of the NP. Our method thus seeks to mitigate the severe computational bottlenecks and avoid the compounding of approximation errors inherent in the standard two-stage pipeline for Bayesian causal inference. This is schematically shown in Figure 1.

There have been recent attempts at end-to-end or meta-learning approaches to estimating interventional distributions. While some methods assume knowledge of the ground truth causal graph (Bynum et al., 2025; Mahajan et al., 2024; Zhang et al., 2024), others offer a similar data driven approach as ours, but only in restricted settings. For example, Geffner et al. (2022) offer an end-to-end approach but restrict to additive noise models, and do not perform functional inference. Sauter et al. (2025) also use meta-learning to directly target the interventional distribution. Apart from differences in architecture and the loss used, their method is limited to discrete interventions. In contrast, we propose a general framework that is not restricted to types functional mechanisms or types of interventions. Further, by viewing meta-learning through a Bayesian lens, we provide insight into the role of the training data as encoding a prior distribution (Gordon et al., 2019; Müller et al., 2022; Hollmann et al., 2023). Tying our method to Bayesian inference also provides an understanding of the behaviour of our model under identifiability and non-identifiability of the causal model (Chickering, 2002; Dhir et al., 2024b;a).

B. Model architecture

This section provides the definitions of the model architecture, as well as motivation for the primary design choices.

Desirable properties As noted in the main text, given that we are interested in predicting $p_{\text{BCM}}(x_i|\text{do}(\mathbf{x}_j), \mathcal{D}_{\text{obs}})$, variables play distinct roles as either the outcome node X_i, the intervening node X_j, or the nodes that are being marginalised. Thus, properties of this distribution, and the role of the variables, guide our architecture choice. First, the interventional distribution remains invariant when the observational data samples are permuted or the nodes being marginalised over are permuted (*permutation-invariance* with respect to observational samples and to all nodes except the outcome X_i and intervention X_j). Second, permuting the interventional queries should permute the samples of the target distribution accordingly (*permutation-equivariance* with respect to interventional samples). Similarly, permuting any nodes involving the outcome or intervention nodes should yield the corresponding permuted interventional distribution (*permutation-equivariance* with respect to outcome and intervention nodes). For example, permutting the outcome and intervention nodes $i \leftrightarrow j$ should result in the permuted $p(\mathbf{x}_j|\text{do}(\mathbf{x}_i), \mathcal{D}_{\text{obs}})$.

Furthermore, we assume no correlations among the interventional samples and, as such, restrict our attention to the family of conditional neural processes (CNPs), where the predictive distributions factorises over the interventional samples $p_\theta(\mathbf{x}_i|\text{do}(\mathbf{x}_j), \mathcal{D}_{\text{obs}}) = \prod_{n=1}^{N_{\text{int}}} p_\theta(x_i^n|\text{do}(x_j^n), \mathcal{D}_{\text{obs}})$. In order to model non-Gaussian interventional distributions, we opt for a Mixture of Gaussians (MoG) representation of $p_\theta(x_i|\text{do}(x_j), \mathcal{D}_{\text{obs}})$ (Bishop, 1994).

An architecture that is flexible enough to satisfy these desiderata is the transformer (Vaswani et al., 2017; Lee et al., 2019). We provide a high-level schematic architecture for our proposed model, the *Model-Averaged Causal Estimation Transformer Neural Process* (MACE-TNP), in Figure 2. We next describe the model components in detail, beginning with the core operations used in the transformer blocks, and then explaining how these are integrated into an architecture tailored for estimating causal interventional effects.

B.1. Transformers

Transformers (Vaswani et al., 2017) can be viewed as general set functions (Lee et al., 2019), making them ideally suited for NPs, which must ingest datasets. We begin by briefly overviewing transformers, defining the attention operations and how we construct a transformer layer, followed by how we integrate transformers into the MACE-TNP architecture.

MHSA and MHCA Throughout this work we make use of two operations: multi-head self-attention (MHSA) and multi-head cross-attention (MHCA). Let $\mathbf{Z} \in \mathbb{R}^{N \times D_z}$ be a set of N tokens of dimensionality D_z. Then, for $\forall\, n = 1, \ldots, N$, the MHSA operation updates this set of tokens as follows

$$\mathbf{z}_n \leftarrow \text{cat}\Big(\Big\{ \sum_{m=1}^{N} \alpha_h(\mathbf{z}_n, \mathbf{z}_m) \mathbf{z}_m{}^T \mathbf{W}_{V,h} \Big\}_{h=1}^{H}\Big) \mathbf{W}_O, \tag{3}$$

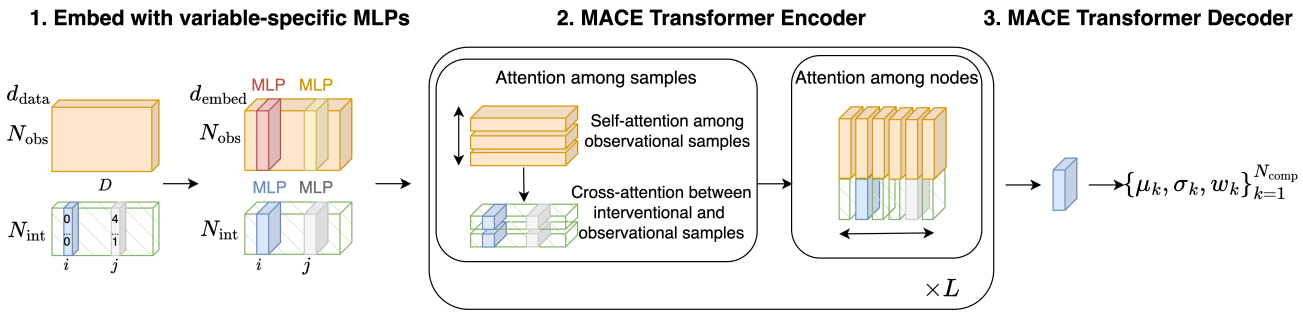

Figure 2. Overview of MACE-TNP yielding $p_\theta(\mathbf{x}_i|\text{do}(\mathbf{x}_j), \mathcal{D}_{\text{obs}})$. Inputs are 1) embedded via variable-specific MLPs, 2) fed into a transformer encoder that alternates sample-wise and node-wise attention. The resulting outcome node representation from the unknown interventional distribution is 3) decoded to obtain the parameters of the NP distribution.

where $\mathbf{W}_{V,h} \in \mathbb{R}^{D_z \times D_V}$ and $\mathbf{W}_O \in \mathbb{R}^{HD_V \times D_z}$ are the value and projection weight matrices, H denotes the number of heads, and α_h is the attention mechanism. We opt for the most widely used softmax formulation

$$\alpha_h(\mathbf{z}_n, \mathbf{z}_m) = \text{softmax}(\{\mathbf{z}_n^T \mathbf{W}_{Q,h} \mathbf{W}_{K,h}^T \mathbf{z}_m\}_{m=1}^N)_m, \tag{4}$$

where $\mathbf{W}_{Q,h} \in \mathbb{R}^{D_z \times D_{QK}}$ and $\mathbf{W}_{K,h} \in \mathbb{R}^{D_z \times D_{QK}}$ are the query and key matrices.

The MHCA operation performs attention between two *different* sets of tokens $\mathbf{Z}_1 \in \mathbb{R}^{N_1 \times D_z}$ and $\mathbf{Z}_2 \in \mathbb{R}^{N_2 \times D_z}$. For $\forall\, n = 1, \ldots, N_1$, the following update on $\mathbf{z}_{1,n}$ is performed:

$$\mathbf{z}_{1,n} \leftarrow \text{cat}\Big(\Big\{ \sum_{m=1}^{N_2} \alpha_h(\mathbf{z}_{1,n}, \mathbf{z}_{2,m}) \mathbf{z}_{2,m}{}^T \mathbf{W}_{V,h} \Big\}_{h=1}^H \Big) \mathbf{W}_O. \tag{5}$$

In order to obtain the attention blocks used within the transformer, these operations are typically combined with residual connections, layer-isations and point-wise MLPs.

More specifically, we define the MHSA operation as follows:

$$\begin{aligned} \widetilde{\mathbf{Z}} &\leftarrow \mathbf{Z} + \text{MHSA}(\text{layer-norm}_1(\mathbf{Z})) \\ \mathbf{Z} &\leftarrow \widetilde{\mathbf{Z}} + \text{MLP}(\text{layer-norm}_2(\widetilde{\mathbf{Z}})). \end{aligned} \tag{6}$$

Similarly, the MHCA operation is defined as:

$$\begin{aligned} \widetilde{\mathbf{Z}}_1 &\leftarrow \mathbf{Z}_1 + \text{MHCA}(\text{layer-norm}_1(\mathbf{Z}_1), \text{layer-norm}_1(\mathbf{Z}_2)) \\ \mathbf{Z}_1 &\leftarrow \widetilde{\mathbf{Z}}_1 + \text{MLP}(\text{layer-norm}_2(\widetilde{\mathbf{Z}}_1)). \end{aligned} \tag{7}$$

Masked-MHSA Consider the general case in which we want to update N token $\mathbf{Z} \in \mathbb{R}^{N \times D_z}$. There might be some situations where we want to make the update of a certain token $\mathbf{z}_n \in \mathbf{Z}$ independent of some other tokens. In that case, we can specify a set $M_n \subseteq \mathbb{N}_{\leq N}^+$ containing the indices of the tokens we want to make the update of $\mathbf{z}_n$ independent of. Then, we can modify the pre-softmax activations within the attention mechanism $\tilde{\alpha}_h(\mathbf{z}_n, \mathbf{z}_m)$, where $\alpha_h(\mathbf{z}_n, \mathbf{z}_m) = \text{softmax}(\tilde{\alpha}_h(\mathbf{z}_n, \mathbf{z}_m))$ as follows:

$$\tilde{\alpha}_h(\mathbf{z}_n, \mathbf{z}_m) = \begin{cases} -\infty & \text{if } m \in M_n \\ \mathbf{z}_n^T \mathbf{W}_{Q,h} \mathbf{W}_{K,h}^T \mathbf{z}_m & \text{otherwise} \end{cases} \tag{8}$$

From the indices of M_n we can construct a binary masking matrix $\mathbf{M} \in \{0,1\}^{N \times N}$:

$$\mathbf{M}_{n,m} = \begin{cases} 0 & \text{if } m \in M_n \\ 1 & \text{otherwise} \end{cases}$$

When used in the context of MHSA, we refer to this operation as masked-MHSA and represent it as $\mathbf{Z} = \text{masked-MHSA}(\mathbf{Z}, \mathbf{M})$.

B.2. Model-Averaged Causal Estimation Transformer Neural Processes (MACE-TNPs)

We refer to (Nguyen and Grover, 2022; Ashman et al., 2024) for a complete description of standard TNP architectures, and focus on describing the architecture of the MACE-TNP in more detail. Our proposed architecture is conceptually similar to the standard TNP architectures, but incorporates specific design choices and inductive biases that make it suitable for causal estimation.

We assume we have access to N_{obs} observational samples and want to predict the distribution of N_{int} interventional samples. The inputs to the MACE-TNP are: the observational dataset $\mathcal{D}_{\text{obs}} \in \mathbb{R}^{N_{\text{obs}} \times D \times d_{\text{data}}}$, the values of the node we intervene upon $\mathbf{x}_j \in \mathbb{R}^{N_{\text{int}}}$ (implying we intervene on node index j), and the outcome node index i. Let $\mathcal{D}_{\text{obs},i} \in \mathbb{R}^{N_{\text{obs}} \times d_{\text{data}}}$ denote the observational data at node i. We omit the batch dimension for notational convenience.

Data pre-processing The model takes as input a matrix of N_{obs} observational samples of D nodes and an intervention matrix of N_{int} queries for a node of interest X_j, with the rest of the $D-1$ nodes masked out (by zeroing them out). Let $\mathcal{D}_{\text{int},i} \in \mathbb{R}^{N_{\text{int}} \times d_{\text{data}}}$ denote the interventional data at node i. In the following we use $\mathcal{D}_{\text{obs},\{k \in [D] \setminus \{i,j\}\}}$ to denote nodes in the observational dataset that are being marginalised over.

Embedding To differentiate between the different types of variables, we employ six different types of encodings, depending on the source of the data (observational (obs) or interventional (int)), and the type of the node (node we intervene upon (j), outcome node (i), or node we marginalise over). These are all performed using 2-layer MLPs of dimension d_{embed}.

$$
\begin{aligned}
\text{observational, intervention node:} \quad & \mathbf{Z}_{\text{obs},j} = \text{MLP}_{\text{obs},j}(\mathcal{D}_{\text{obs},j}) \\
\text{observational, outcome node:} \quad & \mathbf{Z}_{\text{obs},i} = \text{MLP}_{\text{obs},i}(\mathcal{D}_{\text{obs}}, i) \\
\text{observational, marginal nodes:} \quad & \mathbf{Z}_{\text{obs},\{k \in [D] \setminus \{i,j\}\}} = \text{MLP}_{\text{obs}}(\mathcal{D}_{\text{obs},\{k \in [D] \setminus \{i,j\}\}}) \\
\text{interventional, intervention node:} \quad & \mathbf{Z}_{\text{int},j} = \text{MLP}_{\text{int},j}(\mathcal{D}_{\text{int},j}) \\
\text{interventional, outcome node:} \quad & \mathbf{Z}_{\text{int},i} = \text{MLP}_{\text{int},i}(\mathcal{D}_{\text{int},i}) \\
\text{interventional, marginal nodes:} \quad & \mathbf{Z}_{\text{int},\{k \in [D] \setminus \{i,j\}\}} = \text{MLP}_{\text{int}}(\mathcal{D}_{\text{int},\{k \in [D] \setminus \{i,j\}\}}),
\end{aligned}
\tag{9}
$$

where $\{k \in [D] \setminus \{i,j\}\}$ represents the set of indices from $\{1, \dots, D\}$ excluding i and j. The representations are then concatenated back together in the original node order:

$$
\mathbf{Z}_{\text{obs}} = \text{concat}\left([\mathbf{Z}_{\text{obs},k}]_{k \in [D]}\right), \quad \text{where } \mathcal{D}_k = \begin{cases} \mathbf{Z}_{\text{obs},i} & \text{if } k = i \\ \mathbf{Z}_{\text{obs},j} & \text{if } k = j \\ \mathbf{Z}_{\text{obs},k} & \text{otherwise} \end{cases}
$$

$$
\mathbf{Z}_{\text{int}} = \text{concat}\left([\mathbf{Z}_{\text{int},k}]_{k \in [D]}\right), \quad \text{where } \mathbf{Z}_k = \begin{cases} \mathbf{Z}_{\text{int},i} & \text{if } k = i \\ \mathbf{Z}_{\text{int},j} & \text{if } k = j \\ \mathbf{Z}_{\text{int},k} & \text{otherwise} \end{cases}
$$

After the embedding stage, we obtain the representation of the observational dataset $\mathbf{Z}_{\text{obs}} \in \mathbb{R}^{N_{\text{obs}} \times D \times d_{\text{embed}}}$, and the representation of the interventional one $\mathbf{Z}_{\text{int}} \in \mathbb{R}^{N_{\text{int}} \times D \times d_{\text{embed}}}$.

MACE Transformer Encoder We utilise a transformer-based architecture composed of L layers, where we alternate between attention among samples, followed by attention among nodes. This choice preserves 1) permutation-invariance with respect to the obervational samples, 2) permutation-equivariance with respect to the interventional samples, 3) permutation-invariance with respect to the nodes we marginalise over, and 4) permutation-equivariance with respect to the outcome and interventional nodes. Although we generally omit the batch dimension for convenience, we include it in this subsection to accurately reflect our implementation. Thus, the input to the MACE transformer encoder are the observational data representation $\mathbf{Z}_{\text{obs}} \in \mathbb{R}^{B \times N_{\text{obs}} \times D \times d_{\text{embed}}}$ and interventional data representation $\mathbf{Z}_{\text{int}} \in \mathbb{R}^{B \times N_{\text{int}} \times D \times d_{\text{embed}}}$, with B the batch size.

Attention among samples We propose two variants to perform attention among samples. We use the less costly MHSA + MHCA variant for the experiments in the main paper and show that it performs better in Appendix E.1.

1. **Masked-MHSA** among the observational and interventional samples: At each layer l, we first move the node dimension to the batch dimension for efficient batched attention: $\mathbf{Z}^l_{\text{obs}} \in \mathbb{R}^{B\times N_{\text{obs}}\times D\times d_{\text{embed}}} \rightarrow \mathbb{R}^{(B\times D)\times N_{\text{obs}}\times d_{\text{embed}}}$ and $\mathbf{Z}^l_{\text{int}} \in \mathbb{R}^{B\times N_{\text{int}}\times D\times d_{\text{embed}}} \rightarrow \mathbb{R}^{(B\times D)\times N_{\text{int}}\times d_{\text{embed}}}$. We then concatenate the two representations $\mathbf{Z}^l \in \mathbb{R}^{(B\times D)\times(N_{\text{obs}}+N_{\text{int}})\times d_{\text{embed}}} = [\mathbf{Z}^l_{\text{obs}}, \mathbf{Z}^l_{\text{int}}]$, and construct a mask $\mathbf{M} \in \mathbb{R}^{N_{\text{obs}}+N_{\text{int}}}$ that only allows interventional tokens to attend to observational ones.

$$\mathbf{M}_{n,m} = \begin{cases} 1 & \text{if } m < N_{\text{obs}} \\ 0 & \text{otherwise} \end{cases}$$

   We then perform masked-MHSA: $\mathbf{Z}^l = \text{masked-MHSA}(\mathbf{Z}^l, \mathbf{M})$. This strategy has a computational complexity $\mathcal{O}((N_{\text{obs}} + N_{\text{int}})^2)$.

2. **MHSA + MHCA**: An alternative, less costly strategy, is to perform MHSA on the observational data, followed by MHCA between the interventional and observational data. More specifically, as in the previous case we move the node dimension to the batch dimension and then perform:

$$\mathbf{Z}^l_{\text{obs}} = \text{MHSA}(\mathbf{Z}^l_{\text{obs}})$$
$$\mathbf{Z}^l_{\text{int}} = \text{MHCA}(\mathbf{Z}^l_{\text{int}}, \mathbf{Z}^l_{\text{obs}}).$$

   We then concatenate the two representations into $\mathbf{Z}^l \in \mathbb{R}^{(B\times D)\times(N_{\text{obs}}+N_{\text{int}})\times d_{\text{embed}}} = [\mathbf{Z}^l_{\text{obs}}, \mathbf{Z}^l_{\text{int}}]$. This strategy has a reduced computational cost of $\mathcal{O}(N^2_{\text{obs}} + N_{\text{obs}}N_{\text{int}})$ and is the strategy we use for the results in the main paper.

Attention among nodes The output of the attention among samples at layer l $\mathbf{Z}^l \in \mathbb{R}^{(B\times D)\times(N_{\text{obs}}+N_{\text{int}})\times d_{\text{embed}}}$ is then fed into the next stage: attention among nodes. We first reshape the data $\mathbf{Z}^l \in \mathbb{R}^{(B\times D)\times(N_{\text{obs}}+N_{\text{int}})\times d_{\text{embed}}} \rightarrow \mathbf{Z}^{l'} \in \mathbb{R}^{(B\times(N_{\text{obs}}+N_{\text{int}}))\times D\times d_{\text{embed}}}$, and then perform MHSA between the nodes:

$$\mathbf{Z}^{l+1} = \text{MHSA}(\mathbf{Z}^{l'})$$

This is then reshaped back into $\mathbf{Z}^{l+1} \in \mathbb{R}^{B\times(N_{\text{obs}}+N_{\text{int}})\times D\times d_{\text{embed}}}$, and then split into the observational and intervational data representations that are fed into layer $l+1$: $\mathbf{Z}^{l+1}_{\text{obs}} \in \mathbb{R}^{B\times N_{\text{obs}}\times D\times d_{\text{embed}}}$ and $\mathbf{Z}^{l+1}_{\text{int}} \in \mathbb{R}^{B\times N_{\text{int}}\times D\times d_{\text{embed}}}$.

MACE Decoder We parameterise the output distribution of the NP as a Mixture of Gaussians (MoG) with N_{comp} components. The NP outputs the mean, standard deviation and weight corresponding to each component for each interventional query $\{x^n_j\}^{N_{\text{int}}}_{n=1}$: $\{\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{w}\}(x^n_j) := \{\mu_k(x^n_j), \sigma_k(x^n_j), w_k(x^n_j)\}^{N_{\text{comp}}}_{k=1}$. These are computed based on the outcome interventional representation from the final layer of the MACE Transformer Encoder. More specifically, the input to the decoder is $\mathbf{Z}^L_{\text{int},i} \in \mathbb{R}^{N_{\text{int}}\times d_{\text{embed}}}$. This is then passed through a two-layer MLP of hidden size d_{emb}, followed by an activation function

$$\mathbf{z}_{\text{out}} = \text{activation}(\text{MLP}(\mathbf{Z}^L_{\text{int},i}))$$

Finally, we use linear layers to project the embedding $\mathbf{z}_{\text{out}} \in \mathbb{R}^{N_{\text{int}}\times d_{\text{embed}}}$ to the parameters of a mixture of N_{comp} Gaussian components:

$$\boldsymbol{\mu} = \text{Linear}_{\text{mean}}(\mathbf{z}_{\text{out}}) \in \mathbb{R}^{N_{\text{int}}\times N_{\text{comp}}}$$
$$\text{pre-}\boldsymbol{\sigma} = \text{Linear}_{\text{std}}(\mathbf{z}_{\text{out}}) \in \mathbb{R}^{N_{\text{int}}\times N_{\text{comp}}}$$
$$\text{pre-}\mathbf{w} = \text{Linear}_{\text{weight}}(\mathbf{z}_{\text{out}}) \in \mathbb{R}^{N_{\text{int}}\times N_{\text{comp}}}.$$

We then apply element-wise transforms to obtain valid parameters:

$$\boldsymbol{\sigma} = \text{softplus}(\text{pre-}\boldsymbol{\sigma}) \qquad \mathbf{w} = \text{softmax}(\text{pre-}\mathbf{w}),$$

with the softmax being applied along the component dimension.

Loss The output parameters are then used to evaluate the per-dataset loss of the MACE-TNP, which, as shown in Section 3 requires the evaluation of the log-posterior interventional distribution of the MoG:

$$\mathcal{L}_\theta(\mathbf{x}_i, \{\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{w}\}(\mathbf{x}_j)) = \sum_{n=1}^{N_{\text{int}}} \log p_\theta(x_i^n | \operatorname{do}(x_j^n), \mathcal{D}) = \sum_{n=1}^{N_{\text{int}}} \log \left(\sum_{k=1}^{N_{\text{comp}}} w_k(x_j^n) \cdot \mathcal{N}(x_i^n \mid \mu_k(x_j^n), \sigma_k^2(x_j^n)) \right) \quad (10)$$

where $\mathcal{N}(x|\mu, \sigma)$ represents the Gaussian distribution with mean μ and standard deviation σ.

C. Data Generation

We provide in Figure 3 a diagram showing how we sample data from the specified Bayesian Causal Model.

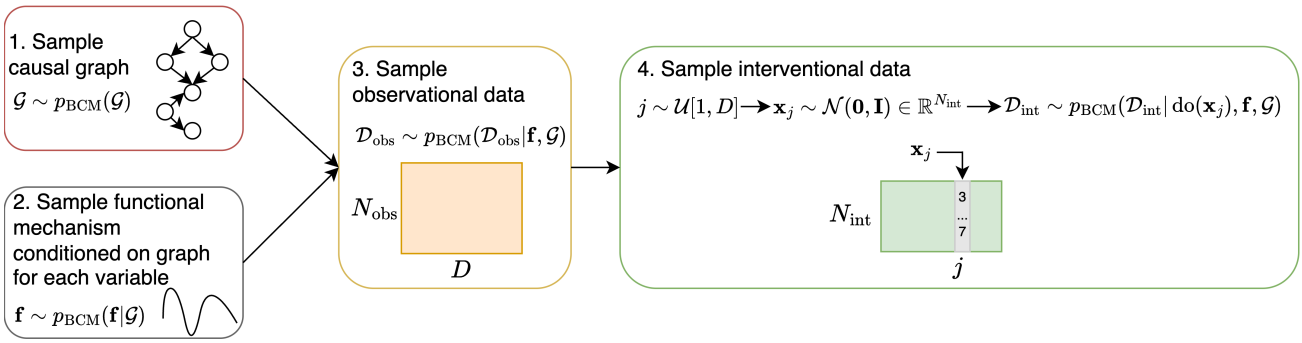

Figure 3. Overview of the data generation process. We first sample a graph $\mathcal{G}$, and a functional mechanism (conditioned on the sampled graph) for each of the D nodes in the dataset. These are then used to draw N_{obs} observational samples. To construct the interventional dataset, we first randomly sample a node to intervene upon j, draw N_{int} intervention values $\mathbf{x}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and set the values of node j to be $\mathbf{x}_j$. We then drawn N_{int} samples of each node to form an interventional dataset $\mathcal{D}_{\text{int}}$.

C.1. Three-node Experiments

In the three-node experiments (Section 4.1) we use two datasets with two different functional mechanisms $f_i(\cdot)$: one sample from a GP prior, and one based on neural networks. In both cases, we sample Erdős–Rényi graphs with graph degree chosen uniformly from $\{1, 2, 3\}$. Following Ormaniec et al. (2025), we standardise all variables upon generation.

GP functional mechanism To model $f_i(\cdot)$ we use a GP with a squared exponential kernel, with a randomly sampled lengthscale for each parent set PA_i of size $|\text{PA}_i|$. More specifically, we sample the lengthscale from a log- distribution $\{\lambda_p\}_{p=1}^{|\text{PA}_i|} \sim \text{Log}(-1, 1)$, followed by clipping between $\lambda_p = \text{clip}(\lambda_p, 0.1, 5)$ to ensure that a too long lengthscale does not result in independence of the variable from a parent. This defines the kernel matrix between the n-th and m-th samples as:

$$\mathbf{K}_{nm} = \exp(-(\text{PA}_i^n - \text{PA}_i^m)^T \boldsymbol{\Lambda}^{-1} (\text{PA}_i^n - \text{PA}_i^m)),$$

with $\boldsymbol{\Lambda} := \text{Diag}(\lambda_1, \ldots, \lambda_{|\text{PA}_i|})$. We then add noise with variance $\sigma^2 \sim \text{Gamma}(1, 5)$ and sample the variables as follows

$$X_i \sim \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I})$$

Neural network-based functional mechanism We sample each variable as follows

$$\eta_i \sim \mathcal{N}(0, 1)$$
$$X_i^n \sim \text{ResNet}_\theta([\text{PA}_i^n, \eta_i]) + \sigma\epsilon,$$

where $\sigma^2 \sim \text{Gamma}(1, 10)$, $\epsilon \sim \mathcal{N}(0, 1)$. ResNet_θ is a residual neural network with a randomly sampled number of blocks $N_{\text{blocks}} \sim \mathcal{U}\{1, \ldots, 8\}$ and randomly sampled hidden dimension $d_{\text{hidden}} \sim \mathcal{U}\{2^5, 2^6, 2^7, 2^8\}$. We use the GELU (Hendrycks and Gimpel, 2016) activation function.

C.2. Higher-dimensional experiments

For the higher dimension experiments in Section 4.2, we generate the data as follows:

- We sample number of variables $D \sim \mathcal{U}[5, 40]$.
- We sample a type of graph, either an Erdős–Rényi graph or a scale-free graph (Barabási and Albert, 1999).
- The density of the graph (number of edges) is sampled from $\mathcal{U}\left[\frac{D}{2}, 6D\right]$.
- For each node, we sample a functional mechanism randomly from either a GP with an additional latent variable input, or a Neural network with an additional latent variable input:
  - GP with latent: We sample a latent $\eta_i \sim \mathcal{N}(0, 1)$, and lengthscales $\{\lambda_p\}_{p=1}^{|\mathrm{PA}_i|+1} \sim \mathrm{Log}(-0.5, 1)$, where PA_i denotes the set of parents of node index i. Functions are sampled from a squared exponential kernel with η_i included as an input and Gaussian noise added with variance $\sigma^2 \sim \mathrm{Gamma}(1, 5)$.
  - NN with latent:

$$\eta_i \sim \mathcal{N}(0, 1)$$
$$X_i^n \sim \mathrm{NN}_\theta([\mathrm{PA}_i^n, \eta_i]) + \sigma\epsilon,$$

    where $\sigma^2 \sim \mathrm{Gamma}(1, 10)$, $\epsilon \sim \mathcal{N}(0, 1)$. NN_θ denotes a randomly initialised neural network with 128 hidden dimensions and one hidden layer. We use the GELU activation function.

Using a latent as an input ensures that the final distribution is not Gaussian. Following Ormaniec et al. (2025), we standardise all variables during the data generation process.

For testing in each variable size in Table 2, we only generate Erdős–Rényi graphs with density $4D$. This is to test the performance of the baselines and our method in the difficult dense graph case. The rest of the data generation process is the same as the training data.

D. Experimental Details

This section provides additional details and results for the experiments presented in Section 4.

D.1. Architecture, training details and hardware

Throughout our experiments we use $H = 8$ attention heads, each of dimension $D_Q = D_{KV} = d_{\text{model}}/8$. The MLPs used in the encoding use two layers and a hidden dimension of $d_{\text{embed}} = d_{\text{model}}$. Unless otherwise specified, we use a learning rate of 5×10^{-4} with a linear warmup of 2% of the total iterations, and a batch size of 32.

To train MACE-TNP, we randomise the number of observational samples $N_{\text{obs}} \sim \mathcal{U}\{50, 750\}$, and set $N_{\text{int}} = 1000 - N_{\text{obs}}$. The training loss is evaluated on these N_{int} samples. For testing, we sample 500 observation points and compute the loss against 500 intervention points.

Two-node linear Gaussian model We use $L = 2$ transformer encoder layers, where each transformer encoder layer involves the attention over samples, followed by attention over nodes. The model dimension is $d_{\text{model}} = 128$, and feedforward width $d_{\text{ff}} = 128$. We train the model for 1 epoch on 50.000 datasets and test on 100 datasets. Training takes roughly 60 minutes on a single NVIDIA GeForce RTX 2080 Ti GPU 11GB, and testing is performed in less than 5 seconds.

Three-node experiments For the experiment in the main paper, we use $L = 2$ transformer encoder layers, a model dimension $d_{\text{model}} = 128$, and feedforward width $d_{\text{ff}} = 128$. We train the model for 2 epochs on 50.000 datasets for the GP experiment and 100.000 datasets for the NN one, and test on 100 datasets in both cases. When testing the OOD performance, we train on the union of the two datasets for 2 epochs. Training the models described in the main text required roughly $4 - 6$ hours of GPU time; however, because we ran them on a shared cluster, actual runtimes may vary with cluster utilization.

Higher dimensional and Sachs experiments For the higher dimensional experiments we use $L = 4$ encoder layers. The model dimension is $d_{\text{model}} = 256$ with feedforward dimension $d_{\text{ff}} = 1024$. We train the model on data generated as listed in Appendix C.2, with $2,500,000$ datasets in total. The model was trained on an NVIDAI A100 80GB GPU for 2 epochs which took roughly 20 hours. We use the model trained for the higher dimensional experiment for the Sachs experiment.

Hardware For the two- and three-node experiments, we ran both training and inference on a single NVIDIA GeForce RTX 2080 Ti (11 GB) with 20 CPU cores on a shared cluster. The only exception was for our largest three-node GP and NN models (with $d_{\text{model}} = 1024$), where we used a single NVIDIA RTX 6000 Ada Generation (50 GB) paired with 56 CPU cores; those models required roughly 25 GB of GPU memory. For the higher-node experiments, we used a single NVIDIA A100 80GB GPU, as well as an RTX 4090 24GB GPU.

E. Additional Results

E.1. Three-node Experiments

In this section we provide additional results on the three-node experiments. First of all, we investigate the performance of MACE-TNP when tested on out-of-distribution data, and discuss how its generalisation capabilities can be enhanced. We then provide results for a range of architectural choices, including varying the number of components in the MoG, the attention mechanism, the model dimension d_{model}, the feedforward width d_{ff}, and the number of layers L.

Out-of-distribution testing: A natural question is: how does the model perform when tested on out-of-distribution (OOD) data? To probe this, we evaluate MACE-TNP (GP) on NN-generated data and MACE-TNP (NN) on GP-generated data. As expected, performance degrades when the test mechanism differs from training, since our model lacks built-in inductive bias for unseen mechanisms: MACE-TNP (GP) tested on NN achieves 608.3 ± 17.3, compared to MACE-TNP (NN) with 527.9 ± 19.8. Similarly, MACE-TNP (NN) tested on GP achieves 678.0 ± 10.0, compared to MACE-TNP (GP) with 563.9 ± 23.4. However, NPs trivially support additional training on any data likely to be informative. Indeed, training MACE-TNP on the combined GP+NN data nearly recovers in-distribution accuracy, achieving 531.0 ± 19.4 on NN data and 583.9 ± 21.5 on GP data (see Table 3).

Table 3. Results for the OOD two-node experiment. We show the NLPID (↓) and report the mean ± the error of the mean over 100 datasets. Each row corresponds to a different functional mechanism used in the test set (GP / NN).

	Training →		
Test ↓	GP	NN	GP+NN
GP	$\mathbf{563.9 \pm 23.4}$	678.0 ± 10.0	$\mathbf{583.9 \pm 21.5}$
NN	608.3 ± 17.3	$\mathbf{527.9 \pm 19.8}$	$\mathbf{531.0 \pm 19.4}$

Results for different architectural choices: Next, we aim to address three questions: 1) between the MHSA and MHCA schemes for sample attention introduced in Appendix B.2, which one performs better? 2) Does increasing the number of MoG components improve performance, and 3) How does the model performance vary with the size of the architecture?

Table 4 shows the results for the two functional mechanisms used in the three-node experiments: GP and NN-based. For each model configuration, we present four sets of results: for a model trained on GP and tested on GP (GP / GP), a model trained on NN and tested on NN (NN / NN), and for a model trained on the combination between the two datasets and tested on each of them (GP+NN / GP and GP+NN / NN). These results allow us to assess whether the influence of model architecture is consistent across the functional mechanisms. Notably, models trained on the combined GP+NN dataset are able to match—within error—the performance of models trained specifically on either GP or NN data. This highlights the strength of the meta-learning approach: even when trained on data generated from diverse functional mechanisms, a single model can generalise effectively across both, achieving performance comparable to specialised models while also benefiting from broader prior coverage. We summarise the findings from Table 4:

1. MHSA + MHCA outperforms the masked-MHSA strategy for attention over samples.
2. Increasing the number of MoG components increases the performance of MACE-TNP. There is a larger gap in

performance when going from 1 to 3 mixture components, indicating the importance of allowing the model to output non-Gaussian marginal predictions. Increasing the number of components to 10 further improves performance, but the gains are not as significant.

3. Scaling up the model architecture generally leads to decreased NLPID.

4. Training a model on the combination of the two datasets (GP+NN) is able to recover—within error—the performance on both datasets.

Table 4. Results of MACE-TNP under different architectural configurations. M-SA stands for Masked-MHSA, while SA+CA indicates the MHSA+MHCA attention mechanism. For each model, the column name under NLPID indicates the training set / test set (i.e. GP+NN / GP indicates we trained the model on the GP+NN dataset and tested it on the GP one). We report the mean $\pm$ the error of the mean of the NLPID over 100 datasets.

					NLPID (↓)			
MoG	Attention	d_{model}	d_{ff}	L	GP / GP	NN / NN	GP+NN / GP	GP+NN / NN
1	M-SA	128	128	4	629.0 ± 20.0	664.1 ± 16.0	640.1 ± 17.3	668.3 ± 17.2
1	SA+CA	128	128	4	617.4 ± 20.1	664.9 ± 16.4	629.8 ± 17.5	688.0 ± 31.5
3	M-SA	128	128	4	581.8 ± 21.8	538.9 ± 19.1	597.6 ± 19.9	547.1 ± 17.8
3	SA+CA	128	128	4	569.3 ± 23.1	540.5 ± 17.1	582.1 ± 21.6	540.6 ± 18.8
10	M-SA	128	128	4	572.1 ± 21.9	533.2 ± 18.3	599.4 ± 20.3	531.5 ± 19.6
10	SA+CA	128	128	4	563.9 ± 23.4	527.9 ± 19.8	583.9 ± 21.5	531.0 ± 19.4
10	SA+CA	512	256	8	555.7 ± 24.6	527.0 ± 19.1	564.6 ± 23.6	532.1 ± 18.4
10	SA+CA	1024	256	8	558.0 ± 23.9	518.2 ± 19.7	565.6 ± 22.3	521.1 ± 20.8

E.2. Sachs Experiment

In Table 5, we show the performance of all the baselines, and the MACE-TNP for the Sachs dataset (Section 4.3). The MACE-TNP performs competitively along with DECI, and they both outperform the other methods.

Table 5. Shows the NLPID (↓) for the Sachs dataset (Sachs et al., 2005). The numbers reported are the mean of the NLPID across 10 nodes across 5 interventions $\pm$ the error of the mean.

	Sachs
MACE-TNP	$\mathbf{998.9 \pm 104.9}$
DiBS-GP	1417.5 ± 186.7
ARCO-GP	1400.7 ± 208.7
DECI	$\mathbf{1000.9 \pm 133.5}$
NOGAM+GP	1763.7 ± 297.4

F. Two-node Linear Gaussian model

We consider the simple case of n independent and identically distributed (i.i.d.) random vectors with 2 nodes of the form $X^i := [X_1^i, X_2^i]^T$ with $i \in \{1, 2, \dots n\}$. To simplify notation, we omit the subscript BCM from p_{BCM} throughout this section. In this setting, there are three distinct possible additive noise linear structural causal models:

$$\mathcal{G}_1 := \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} : X_1 = wX_2 + U_1 \text{ and } X_2 = U_2 \tag{11}$$

$$\mathcal{G}_2 := \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} : X_1 = U_1 \text{ and } X_2 = wX_1 + U_2 \tag{12}$$

$$\mathcal{G}_3 := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} : X_1 = U_1 \text{ and } X_2 = U_2 \tag{13}$$

Identifiable case: We begin with the case where the noise variances of U_1 and U_2 are equal and known—a setting shown to be identifiable in Peters and Bühlmann (2013). Fixing $\sigma^2, \sigma_w^2 \in \mathbb{R}^+$, we consider the following hierarchical model:

$$\begin{gathered} \mathcal{G} \sim \mathcal{U}\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}, \quad U_i \sim \mathcal{N}(0, \sigma^2) \quad \text{for } i = 1, 2 \\ w \sim \mathcal{N}(0, \sigma_w^2) \quad \text{if} \quad \mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2\}. \end{gathered}$$

Non-identifiable case: Second, we consider the errors' variances to be unknown and hence, place priors on these extra parameters as well. We propose the following hierarchical model for fixed $\alpha > \frac{1}{2}$,

$$\begin{gathered} \mathcal{G} \sim \mathcal{U}\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\} \\ \text{If } \mathcal{G} = \mathcal{G}_1, \text{ then } \tau_1^2 \sim InvGamma(\alpha, 0.5), \tau_2^2 \sim InvGamma(\alpha - 0.5, 0.5), w \sim \mathcal{N}(0, \tau_1^2) \\ \text{If } \mathcal{G} = \mathcal{G}_2, \text{ then } \tau_1^2 \sim InvGamma(\alpha - 0.5, 0.5), \tau_2^2 \sim InvGamma(\alpha, 0.5), w \sim \mathcal{N}(0, \tau_2^2) \\ \text{If } \mathcal{G} = \mathcal{G}_3, \text{ then } \tau_1^2 \sim InvGamma(\alpha - 0.5, 0.5), \tau_2^2 \sim InvGamma(\alpha, 0.5). \end{gathered}$$

We use asymmetric priors for τ_1^2 and τ_2^2 to ensure that the posterior assigns equal weight to the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, which are in the same Markov equivalence class. This setup corresponds to the prior structure proposed by Geiger and Heckerman (2002, Equation 12), obtained by setting the precision matrix T in the Wishart distribution to the identity. As noted in their Geiger and Heckerman (2002, Section 4), a change of variables transforms the Wishart prior on the covariance of X into the prior used here for the weights w and error variances τ_1^2 and τ_2^2.

F.1. Experiments

We study the performance of MACE-TNP in both identifiable and non-identifiable causal settings by generating data according to the models described above. For all experiments, we set $\sigma = \sigma_w = 1$ in the identifiable case and $\alpha = 3$ in the non-identifiable case. Closed-form expressions are available for both the posterior over graphs $p(\mathcal{G} \mid \mathcal{D}_{\text{obs}})$ (Geiger and Heckerman, 2002) and the posterior interventional distribution $p(x_i \mid \text{do}(x_j), \mathcal{D}_{\text{obs}})$. We define the average KL between the ground-truth interventional posterior and MACE-TNP's output:

$$\mathbb{E}_{X_j \sim \mathcal{N}(0,1)} \left[\text{KL}(p_{BCM}(x_i | \text{do}(X_j), \mathcal{D}_{\text{obs}}) \| p_\theta(x_i | \text{do}(X_j), \mathcal{D}_{\text{obs}})) \right]. \tag{14}$$

Additionally, given that we work with synthetic data, in this experiment we also have access to the following quantity, where $\{\mathbf{f}^*, \mathcal{G}^*\}$ characterise the true data-generating mechanism:

$$\mathbb{E}_{X_j \sim \mathcal{N}(0,1)} \left[\text{KL}(p_{BCM}(x_i | \text{do}(X_j), \mathbf{f}^*, \mathcal{G}^*) \| p_\theta(x_i | \text{do}(X_j), \mathcal{D}_{\text{obs}})) \right]. \tag{15}$$

This measures the distance between the interventional posterior conditioned on the true data-generating mechanism and MACE-TNP's output. This is different to Equation (14)—in the non-identifiable case, $p_{BCM}(x_i | \text{do}(X_j), \mathcal{D}_{\text{obs}})$ does not converge necessarily to $p_{BCM}(x_i | \text{do}(X_j), \mathbf{f}^*, \mathcal{G}^*)$ even in the limit of infinite observational data.

The results are shown in Figure 4 for the identifiable (left) and non-identifiable (right) cases. They confirm that the output of MACE-TNP does indeed converge to the Bayesian optimal posterior, as the dark blue lines indicating the average KL between the ground-truth posterior distribution and MACE-TNP's output (Equation (14)) go to 0 with increasing sample size in both cases. This is also the case in the identifiable scenario for the average KL between the posterior distribution conditioned on the true data-generating mechanism and MACE-TNP's output (Equation (15)), as shown with the red line in the left plot. However, as expected, in the right plot, this quantity no longer goes to 0 due to the non-identifiability of

the causal graph. This indicates that in the identifiable case, our model recovers the true posterior as well as the correct interventional distribution (conditioned on the true function and graph), whereas in the non-identifiable case it recovers the true posterior but remains uncertain about the correct interventional distribution (conditioned on the true function and graph).

The flexibility of our architecture also allows for conditional queries, multiple interventions, as well as easily incorporating interventional data to help identify causal relations. Hence, we investigate here whether providing a small number $M_{\text{int}} = 5$ of true interventional samples, alongside the observational data, resolves identifiability challenges in the non-identifiable case. As shown in Figure 4 (right) with the green line, we find that this does indeed lower the $\text{KL}(p_{BCM}(x_i|\text{do}(x_j), \mathbf{f}^*, \mathcal{G}^*) \| p_\theta(x_i|\text{do}(x_j), \mathcal{D}_{\text{obs}}, \{x_i^n\}_{n=1}^{M_{\text{int}}}))$, suggesting that even limited interventions can enhance identifiability. We also test this with an increasing number of interventional samples in Figure 5. As soon as the interventional information is rich enough ($M_{\text{int}} \in \{50, 300\}$), the NP recovers the interventional disribution of the true data-generating mechanism even with little to no observational data, as indicated by the near-flat KL curves.

Finally, we show in Figure 6 two examples where the intervention is made at $x = 1$ for the identifiable model and at $x = 2$ for the non-identifiable model. A clear distinction is observed between the two settings: for the identifiable case, the analytical posterior interventional distribution is a mixture of two Gaussian distributions, which, at high observational sample sizes, converges to a single Gaussian (i.e. because the observational data gives information regarding the causal structure, the weight corresponding to one mode collapses to 0). In contrast, for the non-identifiable case, the posterior places equal mass on both $\mathcal{G}_1$ and $\mathcal{G}_2$, and therefore, the mixture structure persists across both regimes. In both settings, the NP-predicted distributions closely match the correct interventional distributions, with accuracy improving as the number of observational samples increases. This improvement is due to two factors. First, larger sample sizes provide the NP with more information about the underlying causal model, allowing for enhanced inference. Second, in the non-identifiable case, the posterior interventional distribution is a mixture of two Student-t distributions with a number of degrees of freedom proportional to the number of observational samples. Thus, in the high sample regime, the mixture distribution converges to a mixture of Gaussians, which is the class that parameterises the output of the NP model.

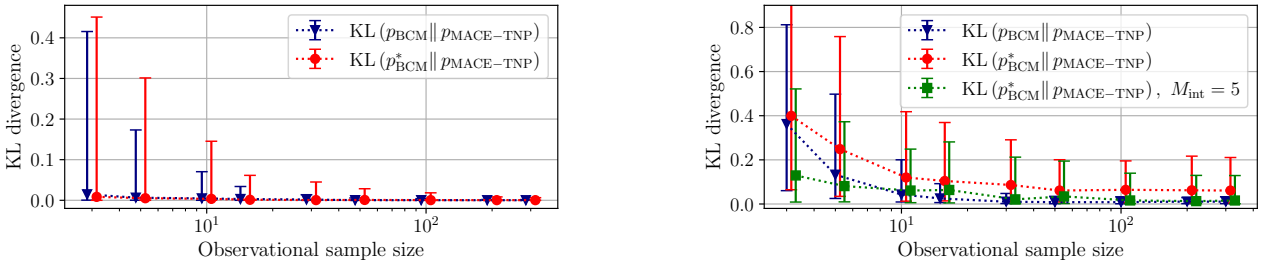

Figure 4. KL divergences as a function of the observational sample size, for the identifiable case (left) and the non-identifiable one (right). Dark blue denotes p_{BCM}—the posterior interventional distribution defined in Equation (1), red and green use p^*_{BCM}—the interventional distribution conditioned on $\{\mathbf{f}^*, \mathcal{G}^*\}$. We additionally provide MACE-TNP with $M_{\text{int}} = 5$ interventional samples. We indicate the median and the 10-90% quantiles.

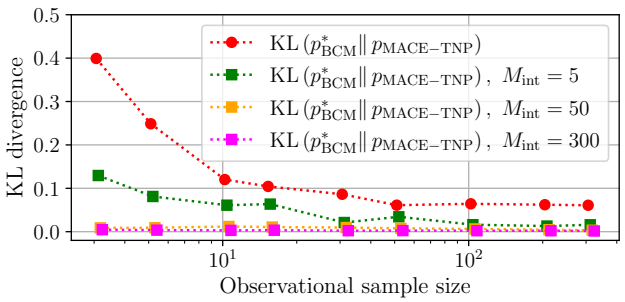

Figure 5. Average KL divergence between the interventional distribution of the true generating mechanism, $\{\mathcal{G}^*, \mathbf{f}^*\}$, and the NP-predicted distribution shown as a function of the observational sample size for the non-identifiable setting. Results are shown for various interventional sample sizes. For simplicity, we only report the medians.

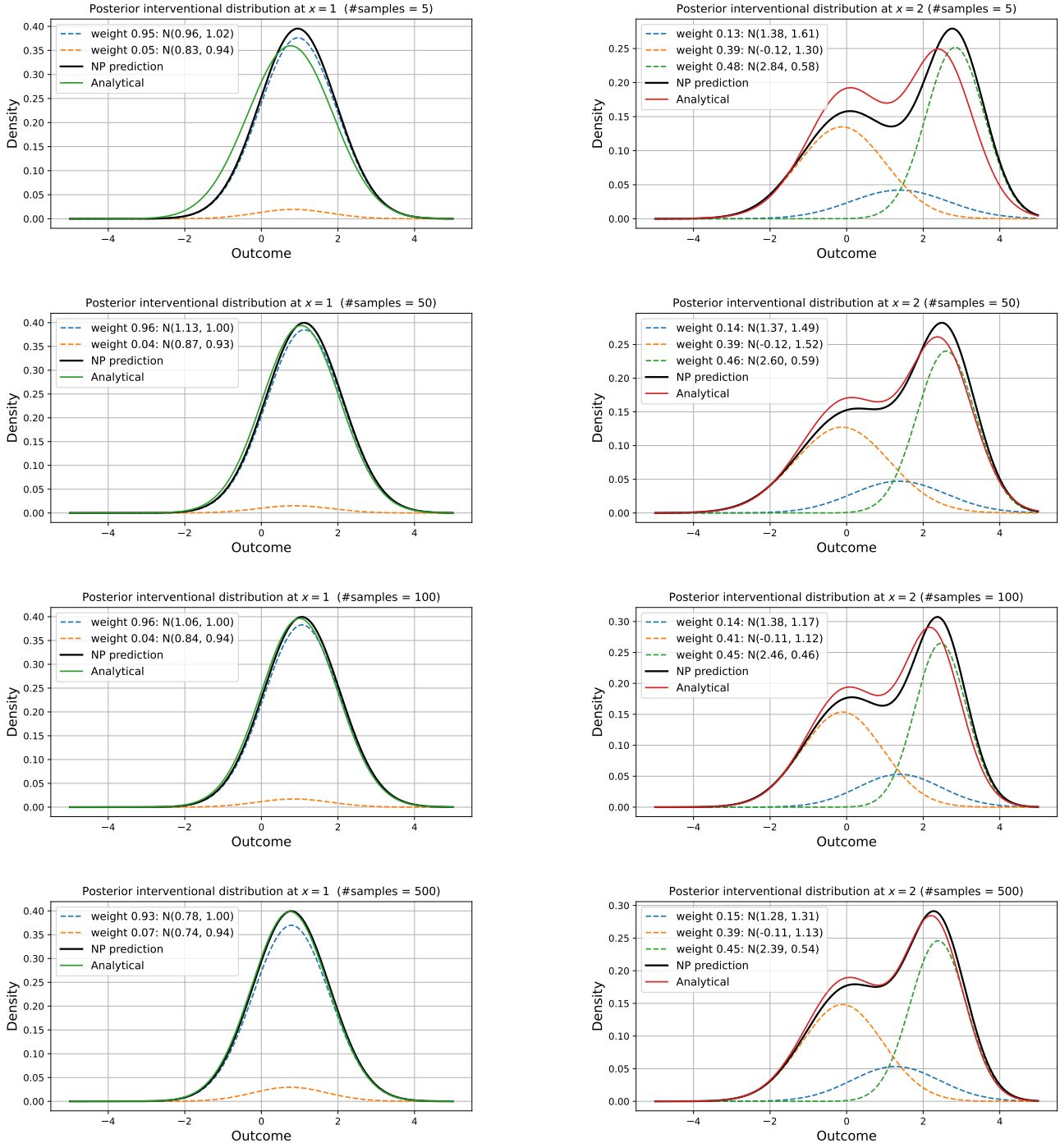

Figure 6. Fitted NP posterior interventional distributions vs. true posterior interventional distributions for identifiable (left) and non-identifiable models (right) at increasing observational sample sizes $\{5, 50, 100, 500\}$.