MARS: Meta-learning as score matching in the function space

Anonymous Author(s)
Affiliation
Address
email

Abstract

We approach meta-learning through the lens of functional Bayesian neural network inference which views the prior as a stochastic process and performs inference in the function space. Specifically, we view the meta-training tasks as samples from the data-generating process and formalize meta-learning as empirically estimating the law of this stochastic process. Our approach can seamlessly acquire and represent complex prior knowledge by meta-learning the score function of the data-generating process marginals. In a comprehensive benchmark, we demonstrate that our method achieves state-of-the-art performance in terms of predictive accuracy and substantial improvements in the quality of uncertainty estimates.

1 Introduction

Meta-learning attempts to extract prior knowledge about the unknown data generation process from related tasks and embed it into the learner (Thrun & Pratt, 1998).We take a new approach to formulating the meta-learning problem and represent prior knowledge in a novel way, by building on recent advances in functional BNNs (Wang et al., 2018; Sun et al., 2019). When viewing the BNN prior and posterior as stochastic processes, the perfect Bayesian prior is the (true) data-generating stochastic process itself. Hence, we view the meta-training datasets as samples from the meta data-generating process and interpret meta-learning as empirically estimating the law of this *stochastic process*. More specifically, we *meta-learn* the *score function* of its *marginal distributions*, which can then directly be used as source of prior knowledge when performing approximate functional BNN inference on a new target task. This allows us to use flexible neural network models for learning the score and overcome the issues of meta-learning BNN priors in the parameter space.

In our experiments, we demonstrate that our proposed approach, *Meta-learning via Attention-based Regularised Score estimation (MARS)*, consistently outperforms previous meta-learners in predictive accuracy and yields significant improvements in the quality of uncertainty estimates. This promises fruitful future applications to domains like molecular biology or medicine, where meta-training data is scarce and reasoning about epistemic uncertainty is crucial.

2 Background

Bayesian Neural Networks. Consider a regression task with data $\mathcal{D} = \left(\mathbf{X}^{\mathcal{D}}, \mathbf{y}^{\mathcal{D}}\right)$ that consists of m i.i.d. noisy function evaluations $y_j = f\left(\mathbf{x}_j\right) + \epsilon_j$ of an unknown function $f : \mathcal{X} \mapsto \mathcal{Y}$. Here, $\mathbf{X}^{\mathcal{D}} = \{\mathbf{x}_j\}_{j=1}^m \in \mathcal{X}^m$ denotes training inputs and $\mathbf{y}^{\mathcal{D}} = \{y_j\}_{j=1}^m \in \mathcal{Y}^m$ the corresponding noisy function values. Let $h_\theta : \mathcal{X} \to \mathcal{Y}$ be a regresion function parametrized by a NN with weights $\theta \in \Theta$. Given a prior distribution $p(\theta)$ over the model parameters θ and a likelihood $p(y|\mathbf{x}, \theta)$, Bayes' theorem yields a posterior distribution $p(\theta|\mathbf{X}^{\mathcal{D}}, \mathbf{y}^{\mathcal{D}}) \propto p(\mathbf{y}^{\mathcal{D}}|\mathbf{X}^{\mathcal{D}}, \theta)p(\theta)$.

Submitted to the 6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans. Do not distribute.

BNN inference in the function space. BNN inference is difficult due to the high-dimensional parameter space Θ and the over-parameterized nature of the NN mapping $h_\theta(x)$. An alternative approach views BNN inference in the function space, i.e., the space of functions $h : \mathcal{X} \mapsto \mathcal{Y}$, yielding the Bayes rule[1] $p(h|\mathbf{X}^\mathcal{D}, \mathbf{y}^\mathcal{D}) \propto p(\mathbf{y}^\mathcal{D}|\mathbf{X}^\mathcal{D}, h)p(h)$ (Wang et al., 2019; Sun et al., 2019). Given finite measurement sets $\mathbf{X} := [\mathbf{x}_1, ..., \mathbf{x}_k] \in \mathcal{X}^k, k \in \mathbb{N}$, we can characterize the stochastic process by its marginal distributions of function values $\rho(\mathbf{h}^\mathbf{X}) := \rho(h(\mathbf{x}_1), ...h(\mathbf{x}_k))$ (Oksendal, 2013).

3 Meta-Learning as Score Estimation in the Function Space

3.1 Problem Statement: Meta-Learning

The meta-learner is given n datasets $\mathcal{D}_1, ..., \mathcal{D}_n$, where each dataset $\mathcal{D}_i = (\mathbf{X}_i^\mathcal{D}, \mathbf{y}_i^\mathcal{D})$ consists of m_i noisy function evaluations $y_{i,t} = f_i(\mathbf{x}_{i,t}) + \epsilon$ corresponding to a function $f_i : \mathcal{X} \mapsto \mathcal{Y} \subseteq \mathbb{R}$ and additive σ sub-Gaussian noise ϵ. In short, we write $\mathbf{X}_i^\mathcal{D} = (\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,m_i})^\top$ for the matrix of function inputs and $\mathbf{y}_i^\mathcal{D} = (y_{i,1}, ..., y_{i,m_i})^\top$ for the vector of corresponding observations. We assume that the functions $f_i \sim \mathcal{T}$ are sampled i.i.d. from a task distribution $\mathcal{T}$, which can be thought of as a stochastic process $p(f)$ that governs the random function $f : \mathcal{X} \mapsto \mathcal{Y}$. Our goal is to *extract knowledge from the observed datasets, which can then be used as a form of prior for learning a new, unknown target function* $f^* \sim \mathcal{T}$ from a corresponding dataset $\mathcal{D}^*$. By performing Bayesian inference with a meta-learned prior that is attuned to the task distribution $\mathcal{T}$, we hope to obtain posterior predicitons that generalize better.

3.2 Meta-Learning as Score Estimation on the Data-Generating Process

We view acquiring prior knowledge in a data-driven manner from a new perspective. Our novel approach to meta-learning hinges upon three key ideas:

First, we view BNN inference in the function space (see Sec. 2), i.e., as posterior inference $p(h|\mathbf{X}^\mathcal{D}, \mathbf{y}^\mathcal{D}) \propto p(\mathbf{y}^\mathcal{D}|\mathbf{X}^\mathcal{D}, h)p(h)$ over neural network mappings $h_\theta : \mathcal{X} \mapsto \mathcal{Y}$ instead of parameters θ. From this viewpoint, the prior which is the target of our meta-learning problem, is $p(h)$ a stochastic process. Second, from a Bayesian perspective, the best possible prior is the stochastic process of the task generating distribution $\mathcal{T}$ itself, i.e., $p(h) = p(f)$. Hence, we aim to meta-learn a prior that matches $p(f)$ as closely as possible. We treat the meta-training datasets $\mathcal{D}_1, ..., \mathcal{D}_n$ as noisy observations from $p(f)$ and use them to estimate the stochastic process marginals $p(\mathbf{f}^\mathbf{X})$. Third, we estimate the score of the marginals rather than the marginal distribution itself. Since the score does not have to integrate to 1 it allows for much more flexible neural network representations.

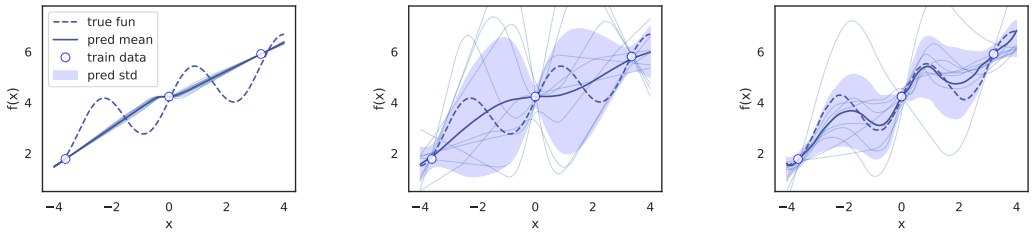
In summary, our high-level approach is to *meta-learn / estimate the prior score* $\nabla_{\mathbf{h}^\mathbf{X}} \ln p(\mathbf{h}^\mathbf{X})$ *that matches the data-generating stochastic process of the meta-training data* $\mathcal{D}_1, ..., \mathcal{D}_n$. At meta-test time, the meta-learned prior marginals are used in the approximate functional BNN inference on a target dataset $\mathcal{D}^*$, to infuse the acquired prior knowledge into the posterior predictions.

4 The MARS meta-learning algorithm

In the following, we discuss how to solve various challenges towards implementing our approach from Sec.3 and develop a practical algorithm for estimating the stochastic prior scores.

Parametric Score Matching for Stochastic Process Marginals. Performing functional approximate posterior inference with fSVGD updates as in (3) requires estimation of the prior marginal scores $\nabla_{\mathbf{h}^\mathbf{X}} \ln p(\mathbf{h}^\mathbf{X})$ for arbitrary measurement sets $\mathbf{X}$. As parametric model of the prior marginal scores, we use a transformer encoder architecture (Vaswani et al., 2017), takes as input a concatenation of measurement set $\mathbf{X} \in \mathbb{R}^{\dim(\mathcal{X}) \times k}$ of k points and corresponding query function values $\mathbf{h}^\mathbf{X} \in \mathbb{R}^{\dim(\mathcal{Y}) \times k}$. The network performs attention over the second dimension, i.e., the k columns corresponding to measurement points. Our score network model is denoted as $\mathbf{s}_\phi(\mathbf{h}^\mathbf{X}, \mathbf{X})$ with trainable parameters ϕ. The model is permutation equivariant w.r.t. the columns that correspond to measurement points and, thus, reflects a key property of stochastic process marginal scores.

[1] Here, $p(h)$ is a stochastic process prior.



(a) Vanilla BNN (b) fBNN with GP prior (c) MARS (ours)

Figure 1: BNN posterior predictions with the corresponding prior distributions/processes.

We train the score network with a modified version of the score matching loss (Hyvärinen & Dayan, 2005) which is based on samples $\mathbf{f}_1^{\mathbf{X}}, ..., \mathbf{f}_n^{\mathbf{X}}$ from the data-generating process marginals. In contrast to the normal score matching loss, our modified loss takes the dependence of marginals on their measurements sets into account and uses an expectation over random measurement sets:

$$\mathcal{L}(\phi) := \mathbb{E}_{\mathbf{X}} \left[\frac{1}{n} \sum_{i=1}^{n} \left(\mathrm{tr}(\nabla_{\mathbf{f}_i^{\mathbf{X}}} \mathbf{s}_\phi(\mathbf{f}_i^{\mathbf{X}}, \mathbf{X})) + \frac{1}{2} ||\mathbf{s}_\phi(\mathbf{f}_i^{\mathbf{X}}, \mathbf{X})||_2^2 \right) \right] \tag{1}$$

Interpolating the Datasets across $\mathcal{X}$. In our meta-learning setup (see Section 3.1), we only receive noisy function evaluations $\mathbf{y}_i^{\mathcal{D}}$ in m_i locations $\mathbf{X}_i^{\mathcal{D}}$ for each meta-training task. However, to minimize the score matching loss in (1), we need the corresponding function values in arbitrary measurement locations of the domain $\mathcal{X}$. To solve this issue, we fit a Gaussian Process (GP) on each dataset $\mathcal{D}_i$ which gives us a GP posterior $p(\mathbf{f}_i^{\mathbf{X}} | \mathbf{X}, \mathbf{X}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}})$ over function values $\mathbf{f}_i^{\mathbf{X}}$ for arbitrary measurement sets $\mathbf{X}$. To account for the epistemic uncertainty of the GP interpolation, for each task, we sample function values $\tilde{\mathbf{f}}_i^{\mathbf{X}} \sim p(\mathbf{f}_i^{\mathbf{X}} | \mathbf{X}, \mathbf{X}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}})$ from the corresponding GP posterior and use them as an input to the score matching loss in (1). Correspondingly, the score matching loss modifies to

$$\tilde{\mathcal{L}}(\phi) := \mathbb{E}_{\mathbf{X}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{p(\tilde{\mathbf{f}}_i^{\mathbf{X}} | \mathbf{X}, \mathbf{X}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}})} \left[\mathrm{tr}(\nabla_{\tilde{\mathbf{f}}_i^{\mathbf{X}}} \mathbf{s}_\phi(\tilde{\mathbf{f}}_i^{\mathbf{X}}, \mathbf{X})) + \frac{1}{2} ||\mathbf{s}_\phi(\tilde{\mathbf{f}}_i^{\mathbf{X}}, \mathbf{X})||_2^2 \right] \right] . \tag{2}$$

Preventing Meta-Overfitting of the Prior Score Network. We only have meta-training data $\{\mathcal{D}_i\}_{i=1}^n$, corresponding to n functions f_i drawn from the data-generating stochastic process, i.e., we only have n samples for estimation of marginal scores $\nabla_{\mathbf{h}^{\mathbf{X}}} \ln p(\mathbf{h}^{\mathbf{X}})$. This makes us prone to overfitting of the prior score network. To counteract the tendency to overfit, we regularize the score network via spectral normalization (Miyato et al., 2018) of the linear layers in the transformer encoder blocks. This biases the score estimates towards higher entropy.

The full MARS algorithm. Overall, the MARS meta-learning algorithm consists of two stages:

Stage 1: Meta-Learning the Prior Score Network. We fit a GP to each of the n meta-training datasets $\mathcal{D}_i$. Then, we train the score network by stochastic gradient descent on the modified score matching loss $\tilde{\mathcal{L}}(\phi)$ in (2). In each iteration, we first sample a measurement set $\mathbf{X}$ i.i.d. from the measurement distribution $\nu = \mathcal{U}(\tilde{\mathcal{X}})$, with $\tilde{\mathcal{X}} \subseteq \mathcal{X}$. For the measurement set, we sample a vector of functions values $\tilde{\mathbf{f}}_i^{\mathbf{X}}$ from the corresponding GP posterior marginals $p(\mathbf{f}_i^{\mathbf{X}} | \mathbf{X}, \mathbf{X}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}}), i = 1, ..., n$. Based on these samples, we form an empirical estimate $\hat{\mathcal{L}}(\phi)$ of the score matching loss and perform a gradient update step on ϕ. This is repeated till convergence and summarized in Alg. 1.

Stage 2: Functional BNN Inference with the Prior Score Network. When concerned with a target learning task with training dataset $\mathcal{D}^* = (\mathbf{X}_*^{\mathcal{D}}, \mathbf{y}_*^{\mathcal{D}})$, we infuse the meta-learned inductive bias into the funcional BNN inference by using the score network $\mathbf{s}_\phi(\mathbf{h}^{\mathbf{X}}, \mathbf{X})$ predictions as swap-in replacement for the marginal scores $\nabla_{\mathbf{h}^{\mathbf{X}}} \ln p(\mathbf{h}^{\mathbf{X}})$. In our experiments, we use fSVGD (see Section 2) as functional approximate inference method (see Alg. 2 in Appx. B).

5 Experiments

We provide a detailed benchmark comparison, demonstrating that MARS: *(i)* achieves state-of-the-art performance in terms of predictive accuracy, *(ii)* yields well-calibrated uncertainty estimates.

	SwissFEL	Physionet-GCS	Physionet-HCT	Berkeley-Sensor	Argus-Control
Vanilla GP	0.876 ± 0.000	2.240 ± 0.000	2.768 ± 0.000	0.258 ± 0.000	0.026 ± 0.000
Vanilla BNN	0.529 ± 0.022	2.664 ± 0.274	3.938 ± 0.869	0.151 ± 0.018	0.016 ± 0.002
MAML	0.730 ± 0.057	1.895 ± 0.141	$\mathbf{2.413 \pm 0.113}$	$\mathbf{0.121 \pm 0.027}$	0.017 ± 0.001
BMAML	0.577 ± 0.044	1.894 ± 0.062	2.500 ± 0.002	0.222 ± 0.032	0.037 ± 0.003
NP	0.471 ± 0.053	2.056 ± 0.209	2.594 ± 0.107	0.173 ± 0.018	0.020 ± 0.001
MLAP	0.486 ± 0.026	2.009 ± 0.248	2.470 ± 0.039	0.348 ± 0.034	0.030 ± 0.003
PACOH-NN	0.437 ± 0.021	1.623 ± 0.057	2.405 ± 0.017	0.160 ± 0.070	0.018 ± 0.002
MARS (ours)	$\mathbf{0.391 \pm 0.011}$	$\mathbf{1.471 \pm 0.083}$	$\mathbf{2.309 \pm 0.041}$	$\mathbf{0.116 \pm 0.024}$	$\mathbf{0.013 \pm 0.001}$

Table 1: Meta-Learning benchmark results in terms of the test RMSE.

	SwissFEL	Physionet-GCS	Physionet-HCT	Berkeley-Sensor	Argus-Control
Vanilla GP	0.135 ± 0.000	0.268 ± 0.000	0.277 ± 0.000	0.119 ± 0.000	0.090 ± 0.000
Vanilla BNN	0.085 ± 0.008	0.277 ± 0.013	0.307 ± 0.009	0.206 ± 0.025	0.104 ± 0.005
BMAML	0.115 ± 0.036	0.279 ± 0.010	0.423 ± 0.106	0.154 ± 0.021	0.068 ± 0.005
NP	0.131 ± 0.056	0.299 ± 0.012	0.319 ± 0.004	0.140 ± 0.035	0.094 ± 0.015
MLAP	0.090 ± 0.021	0.343 ± 0.017	0.344 ± 0.016	0.183 ± 0.017	0.168 ± 0.009
PACOH-NN	$\mathbf{0.037 \pm 0.005}$	$\mathbf{0.267 \pm 0.005}$	0.302 ± 0.003	0.223 ± 0.012	0.119 ± 0.005
MARS (ours)	$\mathbf{0.035 \pm 0.002}$	$\mathbf{0.263 \pm 0.001}$	$\mathbf{0.136 \pm 0.007}$	$\mathbf{0.080 \pm 0.005}$	$\mathbf{0.055 \pm 0.002}$

Table 2: Meta-learning benchmark results in terms of the calibration error.

5.1 Experiment Setup

Baselines. We consider five realistic meta-learning *regression* environments with detailed descriptions provided in Appendix C.2. As non-meta-learning baselines, we use a *Vanilla GP* with RBF kernel and a *Vanilla BNN* with a zero-centered, spherical Gaussian prior and SVGD posterior inference (Liu & Wang, 2016). For meta-learning baselines, we compare to model agnostic meta-learning (*MAML*) (Finn et al., 2017), Bayesian MAML (*BMAML*) (Yoon et al., 2018), neural processes (*NPs*) (Garnelo et al., 2018), *MLAP* (Amit & Meir, 2018) and *PACOH-NN* (Rothfuss et al., 2021a).

5.2 Empirical Benchmark Study

Qualitative illustration. Fig. 1 illustrates the superiority of MARS in (c) to posterior predictions of a) a BNN with standard Gaussian Prior in the parameter space b) a functional BNN with GP prior (Wang et al., 2019). For the meta-training we use $n = 20$ tasks with each $m = 8$ datapoints, generated from a student-t process sinusoidal mean function $2x + 5\sin(2x)$.

MARS provides accurate predictions. We perform a comprehensive benchmark study with the baselines introduced in Sec. 5.1. Table 1 shows that MARS yields substantial RMSE improvements on the unseen meta-data when compared to the non-meta-learning baselines and significantly outperforms all other meta-learning baselines in the majority of the environments.

MARS yields well-calibrated uncertainty estimates. In order to compare the quality of uncertainty estimates of MARS to other meta-learning methods, we compute the *calibration error* which measures how much the predicted confidence regions deviate from the actual frequencies of test data in the respective regions. The results are listed in Table 2, and the metric is detailed Appendix D.2. We can see that MARS consistently yields the best-calibrated uncertainty estimates.

6 Conclusion

We have introduced a novel meta-learning approach in the function space that estimates the score of the data-generating process marginals from a set related datasets. When we face a new learning task, the meta-learned score network is used as a source of prior for functional approximate BNN inference. By representing inductive bias as the score of of stochastic process, our approach is versatile and can seamlessly acquire/represent complex prior knowledge. Empirically, this translates into strong performance when compared to previous meta-learning methods. The substantial improvements of MARS in terms of the quality of uncertainty estimates opens up many potential extensions towards interactive machine learning where exploration based on epistemic uncertainty is key.

References

Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended pac-bayes theory. In *International Conference on Machine Learning*, pp. 205–214. PMLR, 2018.

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 2000.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Stéphane Canu and Alex Smola. Kernel methods and the exponential family. *Neurocomputing*, 69 (7-9):714–720, 2006.

Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In *International Conference on Machine Learning*, 2016.

Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi. The born supremacy: quantum advantage and training of an ising born machine. *npj Quantum Information*, 6(1):1–11, 2020.

Charles-Alban Deledalle, Samuel Vaiter, Jalal M. Fadili, and Gabriel Peyré. Stein unbiased gradient estimator of the risk (sugar) for multiple parameter selection, 2014. URL `https://arxiv.org/abs/1405.1164`.

Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. Tensorflow distributions, 2017. URL `https://arxiv.org/abs/1711.10604`.

Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Vincent Fortuin and Gunnar Rätsch. Deep mean functions for meta-learning in gaussian processes. *arXiv preprint arXiv:1901.08098*, 2019.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL `http://github.com/deepmind/dm-haiku`.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Johannes Kirschner, Mojmir Mutný, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces. In *International Conference on Machine Learning*, 2019a.

Johannes Kirschner, Manuel Nonnenmacher, Mojmir Mutnỳ, Andreas Krause, Nicole Hiller, Rasmus Ischebeck, and Andreas Adelmann. Bayesian optimisation for fast and safe parameter tuning of swissfel. In *FEL2019, Proceedings of the 39th International Free-Electron Laser Conference*, pp. 707–710. JACoW Publishing, 2019b.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pp. 2796–2804. PMLR, 2018.

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*, pp. 3744–3753. PMLR, 2019.

Qiang Liu and Dilin Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *Advances in Neural Information Processing Systems*, 2016.

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*. PMLR, 2016.

Lars Lorch, Scott Sussex, Jonas Rothfuss, Andreas Krause, and Bernhard Schölkopf. Amortized inference for causal structure learning. *arXiv preprint arXiv:2205.12934*, 2022.

Samuel Madden. Intel lab data. `http://db.csail.mit.edu/labdata/labdata.html`, 2004. Accessed: Sep 8, 2020.

Christopher J Milne, Thomas Schietinger, Masamitsu Aiba, Arturo Alarcon, Jürgen Alex, Alexander Anghel, Vladimir Arsov, Carl Beard, Paul Beaud, Simona Bettoni, et al. Swissfel: the swiss x-ray free electron laser. *Applied Sciences*, 7(7):720, 2017.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A Simple Neural Attentive Meta-Learner. In *International Conference on Learning Representations*, 7 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Tianyu Pang, Kun Xu, Chongxuan Li, Yang Song, Stefano Ermon, and Jun Zhu. Efficient learning of generative models via finite-difference score matching. *Advances in Neural Information Processing Systems*, 33:19175–19188, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Anastasia Pentina and Christoph Lampert. A PAC-Bayesian bound for lifelong learning. In *International Conference on Machine Learning*, 2014.

Thomas Pinder and Daniel Dodd. Gpjax: A gaussian process framework in jax. *Journal of Open Source misc*, 7(75):4455, 2022. doi: 10.21105/joss.04455. URL `https://doi.org/10.21105/joss.04455`.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003.

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2018.

Jonas Rothfuss, Fabio Ferreira, Simon Boehm, Simon Walther, Maxim Ulrich, Tamim Asfour, and Andreas Krause. Noise Regularization for Conditional Density Estimation. *arXiv preprint arXiv:1907.08982*, 2019a.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. ProMP: Proximal Meta-Policy Search. In *International Conference on Learning Representations*, 2019b.

Jonas Rothfuss, Vincent Fortuin, Martin Josifoski, and Andreas Krause. Pacoh: Bayes-optimal meta-learning with pac-guarantees. In *International Conference on Machine Learning*, pp. 9116–9126. PMLR, 2021a.

Jonas Rothfuss, Dominique Heyn, Jinfan Chen, and Andreas Krause. Meta-learning reliable priors in the function space. *Advances in Neural Information Processing Systems*, 34, 2021b.

Jonas Rothfuss, Christopher Koenig, Alisa Rupenyan, and Andreas Krause. Meta-learning priors for safe bayesian optimization. In *Conference on Robot Learning*, 2022.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pp. 1842–1850, 2016.

Amar Shah, Andrew Wilson, and Zoubin Ghahramani. Student-t processes as alternatives to gaussian processes. In *Artificial intelligence and statistics*, pp. 877–885. PMLR, 2014.

Jiaxin Shi, Shengyang Sun, and Jun Zhu. A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pp. 4644–4653. PMLR, 2018.

Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *Computing in Cardiology*, 2012.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pp. 574–584. PMLR, 2020.

Ingo Steinwart. A sober look at neural network initializations. *arXiv preprint arXiv:1903.11482*, 2019.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger B. Grosse. Functional variational bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Danica J Sutherland, Heiko Strathmann, Michael Arbel, and Arthur Gretton. Efficient and principled score estimation with nyström kernel exponential families. In *International Conference on Artificial Intelligence and Statistics*, pp. 652–660. PMLR, 2018.

Sebastian Thrun and Lorien Pratt (eds.). *Learning to Learn*. Kluwer Academic Publishers, 1998.

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.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

Dilin Wang, Zhe Zeng, and Qiang Liu. Stein variational message passing for continuous graphical models. In *International Conference on Machine Learning*, pp. 5219–5227. PMLR, 2018.

Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

Yuhao Zhou, Jiaxin Shi, and Jun Zhu. Nonparametric score estimators. In *International Conference on Machine Learning*, pp. 11513–11522. PMLR, 2020.

A Related Work

Meta-Learning. Common approaches in meta-learning amortize the entire inference process (Santoro et al., 2016; Mishra et al., 2018; Ravi & Beatson, 2018; Garnelo et al., 2018), learn a good neural network initialization (Finn et al., 2017; Rothfuss et al., 2019b; Nichol et al., 2018; Kim et al., 2018) or a shared embedding space (Baxter, 2000; Vinyals et al., 2016; Snell et al., 2017). Although these approaches can meta-learn complex inference patterns, the require a large amount of meta-training data and often perform poorly in settings where data is scarce. Another line of work uses a hierarchical Bayesian approach to meta-learn priors over the NN parameters (Pentina & Lampert, 2014; Amit & Meir, 2018; Rothfuss et al., 2021a). Such methods perform much better on small data. However, they suffer form the lack of expressive families of priors for the high-dimensional and complex parameters space of NNs, making too restrictive assumptions to represent complex inductive biases. Our approach overcomes these issues by viewing the problem in the function space and directly learning the score, which can easily represented by a NN, instead of a prior distribution. Also related to our stochastic process approach are methods that who meta-learn Gaussian Process (GP) priors (Fortuin & Rätsch, 2019; Rothfuss et al., 2021b, 2022). However, the GP assumption is quite limiting, while MARS can, in principle, match the marginals of any data-generating process.

Score estimation. We use score estimation as central element of our meta-learning method. In particular, we use a parametric approach to score matching and employ an extended version of the score matching objective of Hyvärinen & Dayan (2005). For high-dimensional problems, Song et al. (2020); Pang et al. (2020) propose randomly sliced variations of the score matching loss. Alternatively, there is a body of work on nonparametric score estimation (Canu & Smola, 2006; Liu et al., 2016; Shi et al., 2018; Engl et al., 1996; Zhou et al., 2020). Among those, the Spectral Stein Gradient Estimator (Shi et al., 2018) has been used for estimating the stochastic process marginals for functional BNN inference in a setting where the stochastic prior is user-defined and allows for generating arbitrarily many samples (Sun et al., 2019). However, compared to our score network, it is harder to prevent meta-overfitting via regularization and to add an explicit dependence on the measurement sets in such nonparametric estimators, making them less suited to our problem.

B MARS implementation details

In this section, we focus on the detailed explanation of the proposed framework. We start by elaborating on the fSVGD inference process, after which we provide information regarding the network architecture and the hyperparameter configuration.

B.1 fSVGD inference

We now provide detailed explanation of the fSVGD implementation, following Wang et al. (2019). The fSVGD algorithm approximates the posterior using a set of L NN parameter particles $\{\theta_1, ..., \theta_L\}$. To optimize posterior approximation, fSVGD iteratively re-samples measurement sets $\mathbf{X}$ from a measurement distribution ν, supported on $\mathcal{X}$, computes SVGD updates (Liu & Wang, 2016) in the function space and projects them back into the parameter space, in order to update the parameter-space particle configuration approximating the posterior. To achieve this, fSVGD uses the functional posterior score, i.e., the gradient of the log-density $\nabla_{\mathbf{h}^\mathbf{X}} \ln p(\mathbf{h}^\mathbf{X}|\mathbf{X}, \mathbf{X}^\mathcal{D}, \mathbf{y}^\mathcal{D}) = \nabla_{\mathbf{h}^\mathbf{X}} \ln p(\mathbf{y}^\mathcal{D}|\mathbf{h}^{\mathbf{X}^\mathcal{D}}) + \nabla_{\mathbf{h}^\mathbf{X}} \ln p(\mathbf{h}^\mathbf{X})$ to guide the particles towards areas of high posterior probability. Formally, for all $l = 1, ..., L$ the particle updates are computed as

$$\theta^l \leftarrow \theta^l - \gamma \left(\nabla_{\theta^l} \mathbf{h}^\mathbf{X}_{\theta^l}\right)^\top \underbrace{\left(\frac{1}{L} \sum_{i=1}^{L} \mathbf{K}_{li} \nabla_{\mathbf{h}^\mathbf{X}_{\theta^i}} \ln p(\mathbf{h}^\mathbf{X}_{\theta^l}|\mathbf{X}, \mathbf{X}^\mathcal{D}, \mathbf{y}^\mathcal{D}) + \nabla_{\mathbf{h}^\mathbf{X}_{\theta^l}} \mathbf{K}_{li} \right)}_{\text{SVGD update in the function space}} \tag{3}$$

where $K = [k(\mathbf{h}^\mathbf{X}_{\theta^l}, \mathbf{h}^\mathbf{X}_{\theta^i})]_{li}$ is the kernel matrix between the function values in the measurement points based on a positive semi-definite kernel function $k(\cdot, \cdot)$. A key insight which we will later draw upon in the paper is that such functional approximate inference techniques only make use

Algorithm 1 MARS: Meta-Learning the Data-Generating Process Score

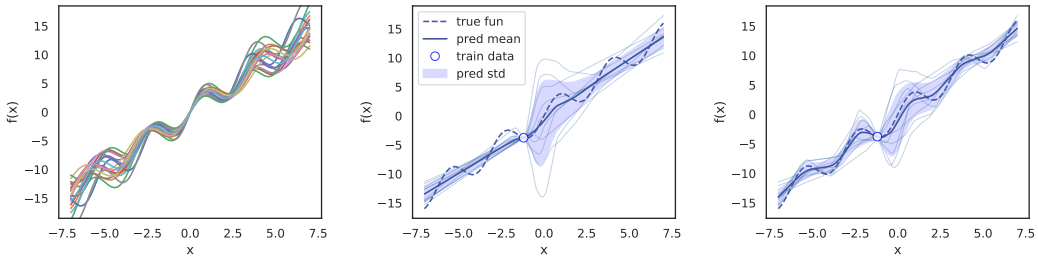
Input: datasets $\mathcal{D}_1, ..., \mathcal{D}_n$, measurement point distribution ν, step size η
Initialize score network parameters ϕ
for $n = 1, ..., n$ **do**
  fit GP on $\mathcal{D}_i$, obtain GP posterior $p(f_i|\mathbf{X}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}})$
while not converged **do**
  $\mathbf{X} \overset{iid}{\sim} \nu$ // sample measurement set
  for $i = 1, ..., n$ **do**
    $\tilde{\mathbf{f}}_i^{\mathbf{X}} \sim p(\mathbf{f}_i^{\mathbf{X}}|\mathbf{X}, \mathbf{X}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}})$ // sample function values from posterior marginal
  $\hat{\mathcal{L}}(\phi) \leftarrow \frac{1}{n}\sum_{i=1}^{n}\left(\text{tr}(\nabla_{\tilde{\mathbf{f}}_i^{\mathbf{X}}}\mathbf{s}_\phi(\tilde{\mathbf{f}}_i^{\mathbf{X}}, \mathbf{X})) + \frac{1}{2}||\mathbf{s}_\phi(\tilde{\mathbf{f}}_i^{\mathbf{X}}, \mathbf{X})||_2^2\right)$ // score matching loss
  $\phi \leftarrow \phi + \eta\nabla_\phi\hat{\mathcal{L}}(\phi)$ // score network gradient update
Output: trained score network $\mathbf{s}_\phi$

of the prior scores and, in principle, do not require a full prior density which integrates to 1. To ingest the meta-learned prior knowledge into the posterior inference, we swap in our MARS score network $\mathbf{s}_\phi(\cdot,\cdot)$ for the prior scores $\nabla_{\mathbf{h}^{\mathbf{X}}} \ln p(\mathbf{h}^{\mathbf{X}})$ which are typically based on a known, user-defined stochastic process prior, and estimated with inifinite sample access (Wang et al., 2018). We summarize the fSVGD without score network in Algorithm 2.

Algorithm 2 Approximate BNN Inference with fSVGD (Wang et al., 2019)

Input: SVGD kernel function $k(\cdot,\cdot)$, step size γ
Input: dataset $\mathcal{D}^* = (\mathbf{X}_*^{\mathcal{D}}, \mathbf{y}_*^{\mathcal{D}})$ for target task, trained score network $\mathbf{s}_\phi(\cdot,\cdot)$
Initialize set of BNN particles $\{\theta_1, ..., \theta_L\}$
while not converged **do**
  $\mathbf{X} \overset{iid}{\sim} \nu$ // sample measurement set
  for $l = 1, ..., L$ **do**
    $\mathbf{h}_{\theta^l}^{\mathbf{X}} \leftarrow (h_\theta^l(x_1), ..., h_\theta^l(x_k))$ where $\mathbf{X} = (x_1, ..., x_k)$ // compute NN function values in $\mathbf{X}$
    $\nabla_{\mathbf{h}_{\theta^l}^{\mathbf{X}}} \ln p(\mathbf{h}_{\theta^l}^{\mathbf{X}}|\mathbf{X}, \mathbf{y}^{\mathcal{D}}) \leftarrow \nabla_{\mathbf{h}_{\theta^l}^{\mathbf{X}}} \ln p(\mathbf{y}^{\mathcal{D}}|\mathbf{h}_{\theta^l}^{\mathbf{X}^{\mathcal{D}}}) + \mathbf{s}_\phi(\mathbf{h}_{\theta^l}^{\mathbf{X}}, \mathbf{X})$ // posterior marginal score
    $\theta^l \leftarrow \theta^l - \gamma\left(\nabla_{\theta^l}\mathbf{h}_{\theta^l}^{\mathbf{X}}\right)^\top\left(\frac{1}{L}\sum_{i=1}^{L}\mathbf{K}_{li}\nabla_{\mathbf{h}_{\theta^i}^{\mathbf{X}}}\ln p(\mathbf{h}_{\theta^l}^{\mathbf{X}}|\mathbf{X}, \mathbf{y}^{\mathcal{D}}) + \nabla_{\mathbf{h}_{\theta^l}^{\mathbf{X}}}\mathbf{K}_{li}\right)$ // fSVGD update
Output: Set of BNN particles $\{\theta_1, ..., \theta_L\}$ that approximate the BNN posterior process

fSVGD hyperparameter selection. For the fBNN training using fSVGD, among other parameters, we need to choose the learning rate, kernel bandwidth and likelihood standard deviation, as we found these three to have strongest impact on training dynamics. Across all hyperparameter configurations, we fixed the number of particles approximating the posterior distribution to 10, trained the network for 10000 training steps, normalize both input and output data and as fSVGD particles, used NN with three hidden layers, each of size 32. As the activation of the networks, we use the Leaky Rectified Linear Unit (leaky ReLU); as the weight initializer we use He initialization (He et al., 2015), and when comparing to the original fSVGD implementation with SSGE, we fix the bandwidth value to 0.2. Note that we also experimented with using the median heuristic for choosing the bandwidth (as proposed by Shi et al. (2018)), however this yielded inferior performance. Finally, for the fSVGD kernel, we use a radial basis function kernel with a standard deviation of one throughout our experiments. For the kernel lengthscale, we perform a random search over values in the range of $[0.1, 10.]$. As mentioned, the final performance heavily depends on selecting an appropriate combination of the kernel lengthscale and likelihood standard deviation, whereas the remaining hyperparameters of the fSVGD algorithm yielded minor differences in the overall performance. Finally, the bias initialization is done according to Steinwart (2019), detailed in the following section.



(a) Data generation process (b) Constant bias initializer (c) *Steinwart* bias initializer

Figure 2: Accelerating the non-linear behaviour of the fSVGD network particles

B.2 Bias initialization

In order to promote non-linearity outside of the meta-data domain, we initialize each bias b_i according to Steinwart (2019). This is performed in the following way:

$$b_i := -\langle w_i, x_i^\star \rangle ,$$

where w_i is uniformly sampled from a sphere by taking $w_i = \frac{a_i}{||a_i||}$, $a_i \sim U(0,1)$. x_i^*'s are sampled uniformly: $x_i^* \sim U(min_x, max_x)$, where min_x and max_x are the minimum and maximum points in the input domain respectively.

Accelerating non-linear behaviour. By applying *Steinwart* initialization, we can significantly speed up the learning of non-linear functional relationship, as showcased in Figure 2: Figure 2a displays the underlying data generation process, Figure 2b displays BNN predictions after 2000 iterations with a constant bias initializer (where the constant equals 0.01), and Figure 2c shows the result of same training dynamics, using *Steinwart's* bias initializer instead of the constant initializer, clearly indicating the benefit of using the latter. Note that although *Steinwart* bias initializer significatly speeds up the non-linear behaviour of the network, if trained for a large enough number of iterations, the performance of the two networks with different bias initializations becomes much less distinct. For further details on the implications of the *Steinwart* initializer on the training dynamics, we refer to Steinwart (2019).

B.3 Score estimation network

We now give an overview of the score estimation network. We start by providing motivation for the required architecture, detailing on the permutation equivariance properties of the network. We then construct the proposed architecture step-by-step, after which, we give the architectural details and mention additional architectural designs we experimented with. Finally, we comment on the optimization method and acknowledge the libraries we used in our implementations.

Incorporating task invariances. The proposed network is permutation equivariant across the k dimension: reordering the measurement inputs would result in reordering the network's prediction in the same manner: for any permutation π, $\mathbf{s}_\phi(\mathbf{h}^{\mathbf{X}_{\pi(1:d)}}, \mathbf{X}_{\pi(1:d)})_{i,j} \mapsto \tilde{\nabla}_{\mathbf{h}^{\mathbf{X}_{1:d}}} p(\mathbf{h}^{\mathbf{X}_{1:d}})_{\pi(i),\pi(j)}$. In order to impose permutation equivariance, we use the self-attention mechanism (Vaswani et al., 2017). Permutation equivariance is obtained by concatenating inputs (measurement points) and the corresponding functional evaluations (i.e., the concatenation constructs an object corresponding to Transformer *tokens*), which are then embedded and inputted to the attention mechanism. The architecture is displayed in Figure 3.

Constructing the network architecture. The core of our model is composed of *two* identical blocks, each block consisting of two residual layers, first one applying multi-head self-attention and

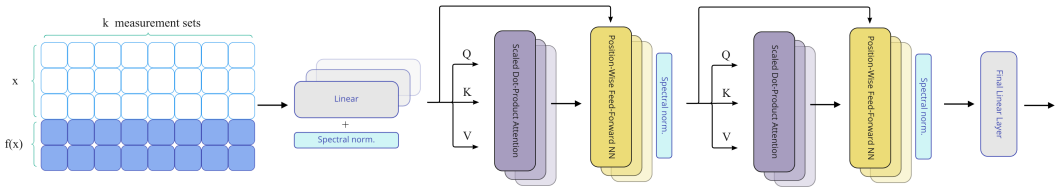


Figure 3: Architecture of MARS score estimation network. From left to right: k measurement sets consisting of input-output concatenations with $x_i \in \mathbb{R}^3, f(x_i) \in \mathbb{R}^2, i = 1, ..., k$; inputs/outputs embeddings using spectrally normalized linear layers; two identical blocks of scaled dot product attention, residual layers and feed-forward position-wise NN with spectrally normalized linear layers; the final linear layer, *not* spectrally normalized.

the second one position-wise feed-forward neural network, akin to the vanilla Transformer encoder. Since multi-head attention is permutation equivariant over the measurement point, (i.e., token) dimension, the representation is permutation equivariant at all times (Lee et al., 2019). Finally, to minimize training time, we select attention embedding dimensions proportional to the data dimension of the environments; higher-dimensional environments (e.g., SwissFEL, detailed in Appendix C) correspond to higher number of attention parameters/embedding dimensions and lower environments (e.g., the Sinusoid and Berkeley environment) to lower embeddings. Furthermore, we tune for the learning rate: the chosen configurations are reported at dropbox.com/MARS_hparams, and the corresponding implementation details described in the following subsection.

Constructing the score network. The score network is trained on the input/output pair concatenations, which are then embedded onto some higher-dimensional space. After this, the self-attention is performed. Specifically, we fix the initialization scale of self-attention weights is set to 2.0. As mentioned, the size of the model and embeddings varies across meta-learning environments. Next follows a position-wise feed-forward neural network, for which we use Exponential Linear-Unit (ELU) as the activation, as implemented in Haiku's vanilla Transformer encoder (Hennigan et al., 2020). Afterwards, the self-attention mechanism is once again applied, following position-wise feed-forward neural network, after which the final linear layer is applied.

Variants of the attention mechanism. We experimented with multiple variations of the attention mechanism, all being permutation equivariant in the k dimension similar to the work of Lorch et al. (2022). We mention two other architectural designs: in the first design, rather than using the embeddings of the concatenation of input-output points as keys, queries and values (as performed in the vanilla Transformer encoder), we experimented with using the emeddings of the inputs as the keys and queries and the embeddings of functional outputs. This however, yielded varying performance: on several low-dimensional tasks the performance was slightly increased, however performance on tasks with higher-dimensional inputs substantially decreased. In another attempt, we experimented with learning different embeddings for inputs and functional outputs, which we then concatenated and used as keys, queries and values. The difference in performance in this method was marginal and for simplicity purposes, we decided against it.

Optimizer setup and employed libraries. Finally, across all experiments, we use gradient clipping of the prior score together with the ADAM optimizer (Kingma & Ba, 2014), with the default values set in Jax (Bradbury et al., 2018) and Haiku (Hennigan et al., 2020). For the gradient clipping, we use values of 1., 10., or 100., depending on the task and the underlying properties of the data generating process. For example, for the first experiment in Appendix E, larger clipping values (50, or 100) performed better for the heavier tailed Student't-t process, whereas clipping at 10. resulted in decent performance of the GP task. For the GP and Student-t process implementations, we use GP-Jax (Pinder & Dodd, 2022), scikit-learn (Pedregosa et al., 2011) and TensorFlow Distributions packages (Dillon et al., 2017).

C Meta-Learning Environments

For each of the meta-environments and algorithms, we ran a separate hyper-parameter search to select the hyper-parameters. In particular, we use 30 randomly sampled hyperparameter configurations across 5 randomly selected seeds and select the best performing one in terms of either RMSE or calibration error. For the reported results, we provide the chosen hyperparameters and detailed evaluation results in `https://tinyurl.com/376wp8xe`.

C.1 Toy experiment

We now provide detailed information regarding the meta-learning environments investigated in Section 5. We start by presenting the number of tasks and samples in each environment, as shown in Table 3.

	Sinusoid	SwissFEL	Physionet	Berkeley	Berkeley*	Argus-Control
n	20	5	100	10	36	20
m_i	8	200	4 - 24	30	288	500

Table 3: Number of tasks n and samples per task m_i for the different meta-learning environments.

Note that the original Berkeley dataset (Berkeley*) has 36 tasks with 288 samples per task, however the unmodified dataset presents a poor environment from probabilistic meta-learning. In order to create a suitable environment for testing prediction uncertainty, we randomly select 30 evaluations of 10 randomly chosen functions. We further elaborate on this in C.2.3.

C.1.1 Sinusoids

Each task of the sinusoid environment corresponds to a parametric function

$$f_{a,b,c,\beta}(x) = \beta * x + a * \sin(1.5 * (x - b)) + c \ , \tag{4}$$

yielding a sum of an affine and a sinusoid function. Tasks differ in the function parameters (a, b, c, β) that are sampled from the task environment $\mathcal{T}$ as follows:

$$a \sim \mathcal{U}(0.7, 1.3), \quad b \sim \mathcal{N}(0, 0.1^2), \quad c \sim \mathcal{N}(5.0, 0.1^2), \quad \beta \sim \mathcal{N}(0.5, 0.2^2) \ . \tag{5}$$

Figure 2a displays functions $f_{a,b,c,\beta}$ with parameters sampled according to (5). To draw training samples from each task, we uniformly sample x from $\mathcal{U}(-5, 5)$ and add Gaussian noise with standard deviation 0.1 to the function values $f(x)$:

$$x \sim \mathcal{U}(-5, 5) \ , \qquad y \sim \mathcal{N}(f_{a,b,c,\beta}(x), 0.1^2) \ . \tag{6}$$

C.2 Real-world data

We consider five realistic meta-learning *regression* environments. The first environment corresponds to data of different calibration sessions of the Swiss Free Electron Laser (SwissFEL) (Milne et al., 2017). Here, a task requires predicting the laser's beam intensity based on parameters of its undulators. We also consider data from the *PhysioNet* 2012 challenge, consisting of electronic health measurements from intensive care patients (Silva et al., 2012), in particular the Glasgow Coma Scale (*GCS*) and the hematocrit value (*HCT*). Here, the different tasks correspond to different patients. In addition, we use the Intel Berkeley Lab temperature sensor dataset (*Berkeley-Sensor*) (Madden, 2004), requiring auto-regressive prediction of temperature measurements corresponding to sensors in different locations of the building. Finally, the *Argus-Control* environment requires predicting of the total variation of a robot from its target position based on its PID controller parameters (Rothfuss et al., 2022). Here, different tasks correspond to different step sizes between source and target location.

	RMSE		Calib. error	
	Full dataset	Partial dataset	Full dataset	Partial dataset
Vanilla GP	0.276 ± 0.000	0.258 ± 0.000	$\mathbf{0.119 \pm 0.000}$	0.109 ± 0.000
Vanilla BNN	0.109 ± 0.004	0.124 ± 0.018	0.179 ± 0.002	0.100 ± 0.125
MAML	$\mathbf{0.045 \pm 0.003}$	0.121 ± 0.027	/	/
BMAML	0.073 ± 0.014	0.222 ± 0.032	0.161 ± 0.013	0.154 ± 0.021
PACOH-NN	0.130 ± 0.009	0.160 ± 0.070	0.167 ± 0.005	0.223 ± 0.012
MLAP	0.050 ± 0.034	0.348 ± 0.034	$\mathbf{0.108 \pm 0.024}$	0.183 ± 0.017
NP	0.079 ± 0.014	0.173 ± 0.000	0.210 ± 0.000	0.140 ± 0.035
MARS	0.093 ± 0.002	$\mathbf{0.116 \pm 0.024}$	0.140 ± 0.002	$\mathbf{0.080 \pm 0.005}$

Table 4: Prediction accuracy and uncertainty calibration on full and partial Berkeley-Sensor dataset

C.2.1 SwissFEL

Free-electron lasers (FELs) accelerate electrons to very high speed in order to generate shortly pulsed laser beams with wavelengths in the X-ray spectrum. The X-ray pulses from the accelerator can be used to map nanometer scale structures, thus facilitating experiments in molecular biology and material science. The accelerator and the electron beam line of a FEL consist of multiple magnets and other adjustable components, which have several parameters that experts adjust in order to maximize the pulse energy (Kirschner et al., 2019a). Due do different operational modes, parameter drift, and changing (latent) conditions, the laser's pulse energy function, in response to its parameters, changes across time. As a result, optimizing the laser's parameters is a recurrent task.

Meta-learning setup. The meta-learning environment represents different parameter optimization runs (i.e., tasks) on SwissFEL, an 800 meter long laser located in Switzerland (Milne et al., 2017). The input space is 12 dimensional and corresponds to the laser parameters, whereas the regression target corresponds to one dimensional pulse energy. For details on the individual parameters, we refer to Kirschner et al. (2019b). Each optimization run consists of roughly 2000 data points, generated with online optimization methods, yielding non-i.i.d. data which becomes successively less diverse throughout the optimization. In order to avoid issues with highly dependent data points, we take the first 400 data points per run and split them into training and test subsets of size 200. As we have a total of 9 runs (tasks) available, we use 5 of them for meta-training and the remaining 4 for meta-testing.

C.2.2 PhysioNet

The 2012 Physionet competition (Silva et al., 2012) published an open-access dataset of patient stays on the intensive care unit (ICU). The dataset consists of measurement taken during the patient stays, where up to 37 clinical variables are measured in a time-series fashion over the span of 48 hours. The intended task for the competition was binary classification of patient mortality, but due to the large number of missing values (around 80 % across all features), the dataset is also popular for testing time series prediction methods.

Meta-learning setup. In order to set up the meta-learning environment, we treat each patient as a separate task and the different clinical variables as different environments. Out of the 37 variables, we pick the Glasgow coma scale (GCS) and hematocrit value (HCT) as environments for our study, since they are among the most frequently measured variables in this dataset. From the dataset, we remove all patients where less than four measurements of CGS (and HCT respectively) are available. From the remaining patients we use 100 patients for meta-training and 500 patients for meta-validation and meta-testing. Since the number of available measurements differs across patients, the number of training points m_i ranges between 4 and 24.

C.2.3 Berkeley-Sensor

The Berkeley dataset contains data from 46 sensors deployed in different locations at the Intel Research lab in Berkeley (Madden, 2004). The measurements are taken over the span of 4 days, sampled at 10 minute intervals. Each task corresponds to one of the 46 sensors and requires auto-regressive prediction, in particular, predicting the next temperature measurement given the last 10 measurement values. There is an abundance of data and the measurement are taken at very close intervals, so the unmodified dataset does not present a good test-bed for probabilistic meta-learning. More specifically, the Bayesian methods are implicitly compelled to put much larger importance on the densely-populated context dataset, making Berkeley-Sensor an unwelcome environment for probabilistic meta-learning. In the following, we compare of the performance of all meta-learning methods mentioned in Section 5 on both the full dataset to its subset where 10 tasks and 30 measurement points are subsampled. The performance on the subsampled dataset is reported in the main section.

Meta-learning setup. When comparing performance on the full dataset, we use 36 sensors (tasks) with data for the first two days for meta-training and the last 10 for meta-testing. We separate the meta-training and meta-testing both temporally and spatially since the data is non-i.i.d. In the partial dataset, instead of using 36 sensor datasets for training we randomly select 10 training tasks and instead of using full measurements, we randomly sample 30 of them. In Table 4, we report performance on both datasets in terms of RMSE and calibration error.

C.2.4 Argus-Control

The final experiment consists of a robotic case study and regards tuning the controller of the Argus linear motion system by Schneeberger Linear Technology. The tuning is performed by optimizing over three gain parameters of the cascaded PI controller in order to minimize the total variation (TV) of the position error signal. This is done in order to achieve minimal position error signal, while simultaneously constraining its maximum frequency in the FFT which measures (potentially damaging) instabilities/vibrations in the system.

Meta-learning setup. In order to place the dataset in the meta-learning setting, we treat the gain parameters as input and the total variation as the target variable. The dataset consists of 24 tasks, out of which 20 are used for meta-training and the remaining 4 for meta-testing. The tasks correspond to optimization runs with different step sizes, ranging from $10\mu m$ to $10mm$. At different scales, the robot behaves differently in response to the controller parameters, resulting in different target and constraint functions, presenting an ideal environment for attempting to learn the underlying functional relationship governed by the step size.

D Experimental methodology

In the following, we provide further description of the inference process and empirical methodology reported in Section 5.

Inference process and evaluation. Evaluating the performance of a meta-learner consists of two phases, *meta-training* and *meta-testing*. The latter phase, *meta-testing*, can be further split into *target training* and *target testing*. In particular, the MARS setting would consist of the following: *(i) meta-training*, where the related tasks are used to train the score estimator network. Then, equipped with the knowledge about the underlying data-generation process, the score estimation network is applied to fSVGD training. Starting the fSVGD training marks the beginning of the second phase, i.e., the *target training*. During this phase, the obtained knowledge (i.e., the inductive bias) is combined with the new information about the task at hand in a, the *context data*, to produce an approximate posterior distribution using the fSVGD algorithm. After the fSVGD training has finished, the remaining tasks/data are used to perform *target-testing*. In this stage, we evaluate the final performance using

two metrics, residual mean squared error (RMSE) and the calibration error. This completes the inference process.

D.1 Residual mean squared error (RMSE)

Evaluating prediction accuracy. The outcome of the training procedure is the particle approximation for the functional SVGD posterior, concerning the unseen task $f \sim \mathcal{T}$ from which we observe a dataset $\mathcal{D}$. During *target-testing*, we evaluate its predictions on a held-out test dataset $\tilde{S}^* \sim \mathcal{D}$ from the same task. Among the meta-learning methods mentioned in Section 5, the predictors of MARS, PACOH-NN, NPs and MLAP bring forward a probability distribution $\hat{p}(y^*|x^*, \tilde{S})$ for the x^* in $\tilde{S}^*$. The respective mean prediction corresponds to the expectation of $\hat{p}$, that is $\hat{y} = \hat{\mathbb{E}}(y^*|x^*, \tilde{S})$. In the case of MAML, only the mean prediction is available. Based on the mean predictions, we compute the *root mean-squared error (RMSE)*:

$$\text{RMSE} = \sqrt{\frac{1}{|\tilde{S}^*|} \sum_{(x^*,y^*)\in S^*} (y^* - \hat{y})^2} \tag{7}$$

and the *calibration error* (see Appendix D.2). Note that, unlike e.g. Rothfuss et al. (2019a) who report the test log-likelihood, we aim to measure the quality of mean prediction and the quality of uncertainty estimation separately, thus reporting both RMSE and calibration error.

The described meta-training and meta-testing procedure is repeated for five random seeds that influence both the initialization and gradient-estimates of the concerned algorithms. The reported averages and standard deviations are based on the results obtained for different seeds.

D.2 Calibration error

Evaluating uncertainty calibration. Consider a new target input x_i over predicted target values y_i and the corresponding conditional distribution $p\left(y_i \mid x_i\right)$. As fBNNs and other probabilistic meta-learning methods produce probability distributions $\hat{p}\left(y_i \mid x_i\right)$, our aim is to compare the similarity between the predicted and the true conditional distribution.

Let us denote the predictor's cumulative density function (CDF) as $\hat{F}\left(y_j \mid x_j\right) = \int_{-\infty}^{y_j} \hat{p}\left(y \mid x_i\right) dy$, where $\hat{p}\left(y \mid x\right)$ corresponds to the produced probability distribution. For confidence levels $0 \leq q_h < \ldots < q_H \leq 1$, we compute the corresponding empirical frequency

$$\hat{q}_h = \frac{\left|\left\{y_j \mid \hat{F}\left(y_j \mid x_j\right) < q_h, j = 1, \ldots, m\right\}\right|}{m},$$

based on some dataset $S = \{(x_i, y_i)\}_{i=1}^m$ containing m samples. As an intuition-based remark; uncertainty estimation should, in principle, result in $\hat{q}_h \to q_h$ as $m \to \infty$. Following Kuleshov et al. (2018), we define the calibration error metric as a function of the residuals $\hat{q}_h - q_h$:

$$\text{calib-err} = \frac{1}{H} \sum_{h=1}^{H} \left|\hat{q}_h - q_h\right|.$$

Note that we report the average of absolute residuals $|\hat{q}_h - q_h|$, rather than reporting the average of squared residuals $|\hat{q}_h - q_h|^2$, as done by Kuleshov et al. (2018). This is done in order to preserve the units and keep the calibration error easier to interpret. In our experiments, we compute the empirical frequency using $M = 20$ equally spaced confidence levels between 0 and 1.

E Further experiments

We provide further empirical evidence for the proposed method. In particular, we showcase that:

1. parametric approach to **MARS performs favourably to many nonparametric estimators**

Estimator	SwissFEL	Physionet-GCS	Physionet-HCT	Berkeley-Sensor	Argus-Control
MARS	$\mathbf{0.391 \pm 0.011}$	$\mathbf{1.471 \pm 0.083}$	$\mathbf{2.309 \pm 0.041}$	$\mathbf{0.116 \pm 0.024}$	$\mathbf{0.013 \pm 0.001}$
SSGE score estimates	0.449 ± 0.027	3.292 ± 0.562	2.784 ± 0.257	1.105 ± 0.562	0.030 ± 0.003
No spectral reg.	0.420 ± 0.060	2.208 ± 0.338	$\mathbf{2.560 \pm 0.341}$	1.734 ± 0.169	$\mathbf{0.014 \pm 0.001}$
No GP sampling	0.471 ± 0.059	2.994 ± 0.363	5.995 ± 1.108	1.253 ± 0.112	0.073 ± 0.003

Table 5: Ablation study results for MARS components in terms of the RMSE.

Estimator	SwissFEL	Physionet-GCS	Physionet-HCT	Berkeley-Sensor	Argus-Control
MARS	$\mathbf{0.035 \pm 0.002}$	0.263 ± 0.001	$\mathbf{0.136 \pm 0.007}$	$\mathbf{0.080 \pm 0.005}$	$\mathbf{0.055 \pm 0.002}$
SSGE score estimates	0.151 ± 0.001	0.249 ± 0.002	0.246 ± 0.007	0.232 ± 0.010	0.210 ± 0.011
No spectral reg.	0.233 ± 0.041	0.265 ± 0.012	0.244 ± 0.009	0.244 ± 0.009	0.187 ± 0.028
No GP sampling	0.204 ± 0.013	$\mathbf{0.225 \pm 0.021}$	0.237 ± 0.018	0.141 ± 0.029	0.216 ± 0.066

Table 6: Ablation study results for MARS components in terms of the calibration error.

2. regularization by **spectral normalization does not hinder flexibility** of the network
3. **GP interpolation successfully incorporates uncertainty** in the scarce parts of the input domain

E.1 Ablation Study

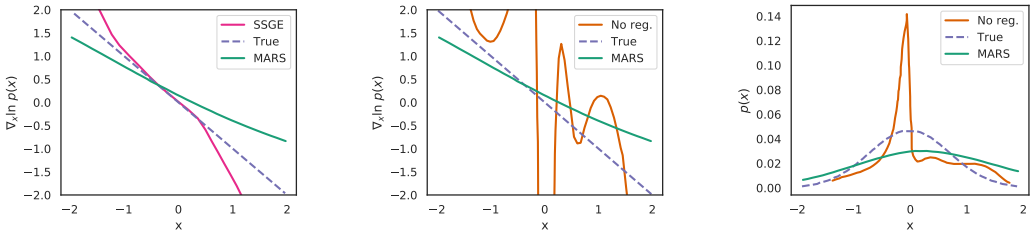
In this section, we empirically investigate the algorithm components that were introduced in Sec. 4 and provide supporting empirical evidence for our design decisions.

First, we perform a quantitative ablation study where we vary/remove components of our MARS algorithm. We consider MARS 1) with nonparametric score estimator SSGE (Shi et al., 2018) instead of the parametric score network + score matching from Sec. **??**, 2) without spectral regularization, and 3) without GP posterior sampling, i.e., using the GP posterior mean instead of samples for the score matching. Table 5 and 6 report the quantitative results of this ablation experiment in terms of the RMSE and the calibration error. In the following, we discuss the three aspects separately:

Parametric score matching outperforms nonparametric score estimation. In Sec. **??** we have introduced a parametric score network which we train with the score matching loss in 1. Alternatively, for each measurement set $\mathbf{X}$, we could apply the nonparametric score estimator SSGE (Shi et al., 2018) to the function values $\tilde{\mathbf{f}}_1^{\mathbf{X}}, ..., \tilde{\mathbf{f}}_n^{\mathbf{X}}$, sampled from the GP posteriors, and directly use the resulting score estimates as prior scores during the fSVGD posterior inference in (3). This alternative approach produces score estimates 'ad-hoc' and does not require us to train an explicit score network. In contrast, our score network is global function which explicitly takes into account the measurement set and thus can exploit similarity structure across $\mathcal{X}$ which SSGE cannot. Additionally, we face the issue that we cannot simply add inductive bias towards higher entropy as we do through the spectral normalization. Overall, the experiment results in Table 5 and 6 suggest that instantiating our general approach with SSGE performs worse than MARS. We can also observe this visually in Fig. 4a where we plot the score estimate of MARS and SSGE on a GP marginal where the true score is known. While the MARS score network slightly over-estimates the variance of true generative-process marginal, SSGE implicitely under-estimates the prior variance which would lead to over-confident predictions. Finally, Table 7 in Appx. E quantitatively shows that, when compared to a variety of nonparametric estimators, our score networks estimates are the most accurate.

Spectral normalization prevents meta-overfitting. In Sec. **??**, we added spectral normalization to our score network. Here, we investigate empirically what happens when we remove the spectral normalization from MARS. Figure 4b and 4c illustrate how our score network over-fits and under-estimates the true data-generating variance when we do not use spectral regularization. Quantitatively, we observe substantial increases in the calibration-errors and a consistent worsening of the predictive accuracy. Overall, this highlights the empirical importance of spectral normalization for preventing meta-overfitting and biasing the score estimates towards higher entropy.

Accounting for epistemic uncertainty of the GP interpolators is crucial. In MARS we interpolate each dataset with a GP and, when performing score estimation, use samples from the GP



(a) SSGE vs. score network (b) spectral regularization (c) Corresponding density

Figure 4: (a) Underconfident MARS and overconfident SSGE score predictions on 10 samples from a Gaussian distribution. (b) Score estimates of MARS with and without regularization on 10 function samples from a zero mean GP with SE kernel. (c) Numerically integrated score predictions corresponding to the scores in (b). Overall, SSGE and MARS w/o regularization overfit and underestimate the variance, whereas spectral normalization biases MARS towards higher entropy.

posterior marginals to account for the epistemic uncertainty of the interpolation. Here, we empirically study what happens if we ignore the epistemic uncertainty and just take the each GP's mean predictions. When doing so we observe detrimental effects on the RMSE and calibration error in the majority of the environments in Table 5 and 6. This affirms that propagating the epistemic uncertainty of the interpolation into the score estimates is a critical component of MARS.

E.2 Parametric vs nonparametric methods

We start by comparing the performance of MARS to that of several nonparametric score estimators, described in Zhou et al. (2020).

Nonparametric models. There are several theoretically-motivated nonparametric score estimation methods, with well-understood properties and straightforward implementations. Their simplicity and flexibility makes them a popular choice (Coyle et al., 2020; Wang et al., 2019; Deledalle et al., 2014). KEF (Canu & Smola, 2006) performs regularized score matching inside a kernel exponential family; this can further be efficiently approximated by the Nyström method (Sutherland et al., 2018). Liu et al. (2016); Chwialkowski et al. (2016) propose a Stein estimator, and Shi et al. (2018) propose Spectral Stein Gradient Estimator (SSGE) by expanding the score function in terms of the spectral eigenbasis. Together with iterative methods (the ν-method and Landweber iteration (Engl et al., 1996)), these approaches can be naturally unified through a regularized, nonparametric regression framework (Zhou et al., 2020).

	RMSE			
	GP-2D	GP-3D	TP-2D	TP-3D
KEF	15.836 ± 0.000	30.625 ± 0.000	6.665 ± 0.000	63.123 ± 0.000
NKEF	8.322 ± 0.000	38.726 ± 0.000	13.479 ± 0.000	30.642 ± 0.000
nu-method	8.602 ± 0.000	36.833 ± 0.000	25.105 ± 0.000	70.227 ± 0.000
SSGE	8.566 ± 0.000	20.991 ± 0.000	6.638 ± 0.000	66.101 ± 0.000
Stein estimator	3.472 ± 0.000	27.188 ± 0.000	7.155 ± 0.000	52.247 ± 0.000
MARS network	$\mathbf{1.629 \pm 0.381}$	$\mathbf{16.397 \pm 1.559}$	$\mathbf{3.733 \pm 0.632}$	$\mathbf{20.837 \pm 1.199}$

Table 7: Prediction accuracy in terms of RMSE on samples from a GP and TP

MARS vs nonparametric estimators. In order to compare the performance of MARS to nonparametric methods, we consider score estimation of marginal distributions of Gaussian Processes (GP) (Rasmussen, 2003) and Student's-t processes (TP) (Shah et al., 2014). More specifically, we randomly sample two/three-dimensional inputs from $[-5, 5]$ and call the underlying process at those inputs 50 (200) times respectively. As the stochastic process mean function we use sinusoidal mean function $2x + 5sin(2x)$, and for the covariance kernel we choose the radial basis function kernel. As nonparametric score estimators cannot incorporate the underlying functional relationship, every

	Cosine similarity			
	GP-2D	GP-3D	TP-2D	TP-3D
KEF	0.475 ± 0.000	0.410 ± 0.000	0.532 ± 0.000	0.485 ± 0.000
NKEF	0.556 ± 0.000	0.434 ± 0.000	0.384 ± 0.000	0.318 ± 0.000
nu-method	0.435 ± 0.000	0.206 ± 0.000	0.444 ± 0.000	0.494 ± 0.000
SSGE	$\mathbf{0.943 \pm 0.000}$	0.601 ± 0.000	$\mathbf{0.661 \pm 0.000}$	0.521 ± 0.000
Stein estimator	0.778 ± 0.000	$\mathbf{0.629 \pm 0.000}$	0.626 ± 0.000	0.590 ± 0.000
MARS network	$\mathbf{0.956 \pm 0.110}$	$\mathbf{0.670 \pm 0.048}$	$\mathbf{0.737 \pm 0.097}$	$\mathbf{0.7061 \pm 0.061}$

Table 8: Cosine similarity between predicted and true scores from a GP and TP

time a new input set is selected, the nonparametric estimators have to be refitted. For two dimensional experiments, we sample each conditional distribution 50 times, and for three dimensional experiments we sample it 200 times. When evaluating the performance of MARS, we take the average performance over 5 different seeds, after training the network for 2000 (5000) iterations for the two (three) dimensional problem. The findings are reported in Tables 7-8 in terms of RMSE and cosine similarity. We see that MARS outperforms all nonparametric score estimators in terms of RMSE and is among the best performing estimators in terms of cosine similarity.

Selecting the comparison metrics. The arguments for reporting the chosen metrics are as follows: RMSE provides a comprehensible, commonly used metric in regression tasks, whereas the cosine similarity plays an important part in ensuring that the direction of the function-space particle optimization path in fSVGD is correct, i.e., the vector field implied by the score network has a high similarity to the true score vector field implied by the underlying stochastic process. Note that by re-writing the Hyvärinen's score matching loss (Hyvärinen & Dayan, 2005), one can easily observe that both metrics are incorporated in the Hyvärinen score matching objective.

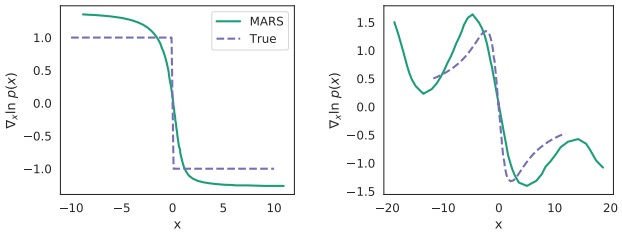


Figure 5: Prediction and actual score of Student's-t (left) and Laplace distribution (right)

E.3 Flexibility of the regularized network

In order to investigate whether spectral normalization hinders the flexibility of the network's outputs, we examine the network's predictions on the following two distribution estimation tasks. In the following experiment, the network is trained on: *(i)* 45 samples from a Student's-t distribution, and *(ii)* 45 samples from a Laplace distribution. The respective predictions are displayed in Figure 5, where one can observe that the network successfully learned curvature of non-differentiable and discontinuous score functions respectively.

E.4 Incorporating uncertainty through GP interpolation

As another empirical addition to the ablation study performed in Section E.1, we further investigate the implications of GP interpolation method, particularly using the Sinusoid environment, detailed in Appendix C.1.1.

Setting the experiment. In this experiment, we sample 10 functions from the environment and evaluate each function at 5 randomly selected inputs in the $[-5, -2] \cup [2, 5]$ range. We do this with

Figure 6: Sampling from the GP successfully incorporates uncertainty about theinput domain.

the aim of investigating whether the proposed GP interpolation method promotes uncertainty in the $[-2, 2]$ part of the domain. In order to provide the analysis, we compare training the score estimation network on the samples from the fitted GPS to the mean predictions of those GPs. More specifically, we run the fSVGD training algorithm using the same data and hyperparameter configuration, once using the mean prediction and once using the GP samples, as performed in the main section.

Empirical findings. In order to compare the two approaches, we plot the corresponding network predictions on four input points sampled uniformly from the whole input domain, i.e. $[-5, 5]$, as shown in Figure 6. We observe a clear difference between the two approaches: the first two plots correspond to two randomly choosen GPs that are fitted to the scarce data and their corresponding functional samples. The middle plot corresponds to the fSVGD trained using only mean GP predictions and the last plot corresponds to the combination of both - the overall framework has successfully learned about the scarcity in the input domain. Note that the amount of underconfidence of the network can be controlled by either decreasing/increasing the likelihood standard deviation parameter, or the kernel bandwidth of the fSVGD algorithm.