Revisiting the Equivalence of Bayesian Neural Networks and Gaussian Processes: On the Importance of Learning Activations

Marcin Sendera[*1,2] **Amin Sorkhei** **Tomasz Kuśmierczyk**[*1]

[1]Jagiellonian University
[2]Mila, Université de Montréal

Abstract

Gaussian Processes (GPs) provide a convenient framework for specifying function-space priors, making them a natural choice for modeling uncertainty. In contrast, Bayesian Neural Networks (BNNs) offer greater scalability and extendability but lack the advantageous properties of GPs. This motivates the development of BNNs capable of replicating GP-like behavior. However, existing solutions are either limited to specific GP kernels or rely on heuristics.

We demonstrate that trainable activations are crucial for effective mapping of GP priors to wide BNNs. Specifically, we leverage the closed-form 2-Wasserstein distance for efficient gradient-based optimization of reparameterized priors and activations. Beyond learned activations, we also introduce trainable periodic activations that ensure global stationarity by design, and functional priors conditioned on GP hyperparameters to allow efficient model selection.

Empirically, our method consistently outperforms existing approaches or matches performance of the heuristic methods, while offering stronger theoretical foundations.

1 INTRODUCTION

Function-space priors for BNNs offer a better way of specifying beliefs on data modeled by the model. Unlike priors on model parameters (i.e. weights), which lack interpretability and are hard to specify, they ultimately lead to more intuitive and meaningful representation of prior knowledge [Sun et al., 2019]. Function-space priors can be conveniently specified in terms of GPs. Instead of specifying complex distributions over model parameters, GPs allow for defining priors over entire functions, through the choice of kernel capturing requirements such as smoothness or periodicity.

Finding a GP that matches a wide BNN is straightforward. A classic result by Neal [1996], Williams [1996], later extended to deep neural networks by Lee et al. [2017], Matthews et al. [2018], shows that wide layers in neural networks behave *a priori* like GPs. This is achieved by identifying GP kernels matching the covariances of (B)NNs.

In contrast, finding BNNs that exhibit the desired behavior of GPs is a notoriously difficult problem, with solutions or approximations existing for only a few GP kernels. For example, Meronen et al. [2020] recently found a solution for the popular Matérn kernel using the Wiener-Khinchin theorem. However, the proposed BNN activation assumes binary white noise priors on the model weights, which severely hinders posterior learning. Consequently, an approximate solution with Gaussian priors must be used, ultimately leading to suboptimal performance, as we demonstrate in Sec. 4.3.

The challenge of imposing function-space *a priori* behavior on BNNs previously has been addressed using gradient-based optimization. In particular, Flam-Shepherd et al. [2017], Flam-Shepherd et al. [2018], and especially Tran et al. [2022] attempted to solve this by learning priors on parameters. However, to achieve sufficient fidelity with simple priors (Gaussian or hierarchical), their approach requires deep networks, which presents a challenge for posterior learning due to the lack of effective algorithms for learning posteriors in deep and wide BNNs. On the other hand, by using normalizing flows to model weight priors they often can match target GP prior (see Sec. 4.2). Nevertheless, imposing different priors for each weight invalidates the theoretical assumptions regarding BNN convergence, rendering this approach merely a heuristic.

Fitting only weight (and biases) priors is insufficient for matching BNN and GP priors. Hence, we propose learning parametric activations as well. We draw inspiration from previous work demonstrating that function-space priors can be adjusted by altering activations [Neal, 1996] and also from

[*]Equal contribution to implementation.

research utilizing activations for improved uncertainty estimation [Morales-Alvarez et al., 2020, Postels et al., 2020]. In fact, the same GP-like behavior can often be realized by different combinations of parameter priors and activations, as seen in the BNN covariance formulation (Eq. 1) and learning both jointly offers a greater flexibility. In particular, our method (see Fig. 1) transfers priors from GPs to BNNs by matching their function-space distributions using the closed-form 2-Wasserstein loss and by learning activations in addition to priors on weights. It achieves faithful function-space priors using just shallow BNNs. In contrast to Tran et al. [2022], it is backed by theoretical results ensuring that the learned BNNs asymptotically converge to GPs, and compared to Meronen et al. [2020], it can work with arbitrary kernels. Finally, it relies on simple weight priors, enabling efficient posterior inference.

Within the proposed framework, we propose two novel extensions. First, by conditioning the BNN's functional priors (weights and activation) on GP hyperparameters – implemented via hypernetworks – we facilitate efficient model selection, e.g., allow prior hyperparameters optimization. Second, building on the result of Meronen et al. [2021] that periodic activations induce stationarity, we propose trainable periodic activations and show how such activations can be learned in practice, ensuring global stationarity in BNNs.

Our contributions extend the existing literature on BNN–GP equivalence in several ways: (1) we introduce a practical approach for imposing GP-like function-space priors on BNNs; (2) we employ hypernetworks to condition these priors on GP hyperparameters; (3) we introduce trainable periodic activations; and (4) we empirically demonstrate that even shallow BNNs can achieve the desired properties, offering an alternative to baselines using deep models. Note that although parametric activations and hypernetworks have been used previously, we are the first to employ and combine them for this particular task. Moreover, although our design of trainable periodic activations builds on the previous result that periodic activations induce stationarity, it is novel. Finally, despite our focus on the specification of priors, we also validate our pre-trained priors by learning posteriors, employing both Hamiltonian Monte Carlo (HMC) and scalable Stochastic Variational Inference (SVI) to demonstrate empirical improvements in predictive performance and uncertainty quantification.

In the Appendix, the results presented in the paper are supplemented with a description of related work, a discussion of computational advantages of our approach, a detailed overview of the experimental settings, and additional plots and numerical data.

Our code we made publicly available [1].

[1] https://github.com/gmum/bnn-functional-priors

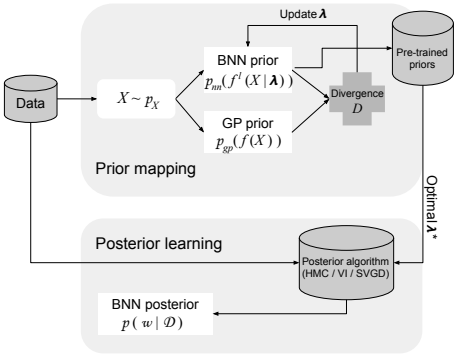


Figure 1: Schematic view of our approach.

2 PRELIMINARIES

Behavior akin to GPs has been identified in wide neural networks with a single hidden layer by Neal [1996], Williams [1996] and later generalized for deep architectures by Lee et al. [2017], Matthews et al. [2018] who considered stacking multiple such the wide layers.
Let's consider a network defined as:

$$f_i^0(x) = \sum_j^I w_{ij}^0 x_j + b_i^0, \quad i = 1, \ldots, H_0;$$

$$f^l(x) = \sum_j^{H_0} w_{ij}^l \phi(f_j^0(x)) + b^l$$

where x denotes an I-dimensional input, and $f^l(x)$ represents output at the l-th layer. In a BNN, $z_{ij}(x) = w_{ij}^l \phi(f_j^0(x))$ is a random variable. Assuming that the weights w_{ij} share a common prior and also the biases b_i have independent but common prior, the output $f^l(x)$ becomes a sum of independent and identically distributed (i.i.d.) random variables. By the Central Limit Theorem, this results in $f^l(x)$ converging to a Gaussian distribution, implying that the BNN converges to a GP in the limit of infinite width.

The covariance between two inputs, x and x', for the BNN at the (output) layer l is given by:

$$\mathrm{Cov}(f^l(x), f^l(x')) = {\sigma_b^l}^2 + {\sigma_w^l}^2 \mathbb{E}_{f_j^0}[\phi(f_j^0(x))\phi(f_j^0(x'))], \tag{1}$$

where σ_b^l and σ_w^l are respectively the variances of the biases and weights at layer l and the expectation is taken over distributions for w^0 and b^0. Hence, the BNN functional prior corresponds to a GP with kernel $\kappa_f^l(x, x') = \mathrm{Cov}(f^l(x), f^l(x'))$.

In this work, we assume zero-centered GPs. Therefore, for the distributions on weights and biases, we take $\mathbb{E}[w] = 0$ and $\mathbb{E}[b] = 0$. To ensure that the variance remains stable as the layer width increases we scale $\mathrm{Var}[w_{ij}^l] = \frac{{\sigma_w^l}^2}{H_0}$.

3 METHOD

3.1 TRANSFER OF PRIORS FROM GP TO BNN

Finding a GP equivalent to a pre-specified BNN can be easily done by a Monte Carlo estimate of Eq. 1. However, we focus on the significantly more challenging inverse problem of imposing behavior akin to a GP on a BNN. The task is to *identify appropriate priors on w^l, b^l, w^0, b^0, and the activation ϕ in Eq. 1 so that the covariance induced by the BNN aligns with the desired GP kernel κ*. Ideally, we aim to have $f^l \sim \text{GP}(0, \kappa)|_{\mathcal{X}}$, meaning that the distribution of the BNN outputs (pre-likelihood) would closely match that of the GP across the input space $\mathcal{X}$, i.e., $p_{nn}(f^l) = p_{gp}(f^l)$.

In practice, however, we settle for approximate matching over finite index sets [Shi et al., 2018, Sun et al., 2019], where the densities over BNN and GP outputs should approximately align as $p_{nn}(f^l(X)) \approx p_{gp}(f^l(X))$. $X \sim p_X$ ($X \in \mathcal{X}$) can be understood as sampling sets of inputs $\{x\}$ at which we observe the functions.

The matching between BNN and GP, we formulate as an optimization task:

$$p^*_{nn} = \text{argmin}_{p_{nn}} \frac{1}{S} \sum_{X \sim p_X} D(p_{nn}(f^l(X)), p_{gp}(f^l(X))),$$

where D is an *arbitrary*, differentiable divergence measure, and S is a number of samples. In practice, we perform gradient-based optimization with $S = 1$ sample used for each step.

3.2 DIFFERENTIABLE PRIORS AND ACTIVATIONS

The results presented in Sec. 2 hold for distributions with finite variances and light tails. We use the basic zero-centered factorized Gaussians with learned variances, e.g., $p(w|\sigma^2_w)$ and $p(b|\sigma^2_b)$, as prior distributions on model weights and biases. To enable gradient-based optimization, such as gradient propagation through weight and bias samples, we reparameterize the distributions using the reparameterization trick [Kingma and Welling, 2014]. Additionally, *we propose to employ parametric and differentiable activations $\phi(\cdot|\eta)$ to enhance the network's flexibility.* This previously overlooked idea allows us to model complex functional priors within a single hidden layer of a BNN, even with simple prior distributions on weights and biases.

Given the reparameterized distributions on weights and biases, as well as the parametric activation, p_{nn} prior is fully characterized by the parameters $\lambda = \{\sigma^0_b, \sigma^0_w, \sigma^l_b, \sigma^l_w, \eta\}$. The optimization objective can be then expressed as:

$$\lambda^* = \text{argmin}_{\lambda} \frac{1}{S} \sum_{X \sim p_X} D\left(p_{nn}\left(f^l(X|\lambda)\right), p_{gp}\left(f^l(X)\right)\right), \tag{2}$$

where S denotes the number of input samples. Eq. 2 is solved through gradient-based optimization w.r.t. λ.

Eq. 2 requires deciding on the divergence measure D and a model for the activation function ϕ. The problem of learning activations for neural networks involves designing functions that enable networks to capture complex and non-linear relationships effectively. In principle, any function $\phi : \mathcal{R} \to \mathcal{R}$ can serve as an activation, but the choice of ϕ significantly impacts both the network's expressiveness and its training dynamics.

We explore several models for ϕ, including Rational (Pade) activations [Molina et al., 2019], Piecewise Linear (PWL) activations[2], and activations implemented as a neural network with a single narrow (with only 5 neurons) hidden layer, using ReLU/SiLU own activations. These functions are computationally efficient and introduce desirable non-linear properties, striking a balance between efficiency and the capacity to model intricate patterns.

Our choice of a NN-based activation with a single hidden layer consisting of 5-10 neurons and using ReLU/SiLU activations was informed by empirical studies (see Fig. 2 and Tables 3 and 4 in Section B) comparing various learnable functions. These experiments, including ablations on the number of layers and width of the NN activations, show that this configuration offers a good balance of flexibility, fast convergence, and quality of prior fit. While deeper or wider NN activations can marginally improve fidelity in some cases, they also increase convergence times and can make optimization harder.

3.3 PERIODIC ACTIVATIONS FOR STATIONARY GPS

The optimization in Eq. 2 acts on range of inputs X, where the BNN's behavior increasingly resembles the GP as training progresses. However, there are no guarantees about the BNN's behavior outside this subset, particularly far from it. This limitation is exacerbated by *local stationarity* in BNNs: although GPs with stationary kernels are *globally* stationary, BNNs usually only exhibit stationarity within a limited input range. Outside this range, the covariance induced by the BNN may diverge from the desired GP behavior. This localized training can cause uncertainty quantification issues. In regions far from the data, a BNN may either become overly confident or overly uncertain, depending on the priors and the mismatch between the BNN's architecture and the GP's true behavior. It is challenging to enforce global properties, like stationarity or smoothness, across the entire input space, as the BNN's learned covariance structure is overly dependent on the inputs' range.

We address the local stationarity challenge by *introducing trainable activations designed to exhibit periodic behavior.*

[2] `https://pypi.org/project/torchpwl/`

This approach is motivated by the result of Meronen et al. [2021], who showed that periodic activation functions induce stationarity in BNNs. Based on this theoretical result, we propose a practical solution. Our proposed activations

$$\phi(x|\psi, A) = \sum_{i=1}^{K} A_i \cos(2\pi\psi_i x) + \sum_{j=1}^{K} A_j \sin(2\pi\psi_j x)$$

are inspired by Fourier analysis and can be trained to fit a desired functional prior. They rely on the variational parameters $\eta = \{\psi_i, A_i, \psi_j, A_j\}$ steering respectively frequencies and amplitudes. For experiments, we used $K = 5$ components.

3.4 CONDITIONAL PRIORS AND ACTIVATIONS

Researchers previously, when fitting BNN priors, were limited to upfront fixing hyperparameters (such as length-scale) of target GPs or alternatively, they would employ workarounds like hierarchical kernels with hyperparameters sampled from hyperpriors (see, e.g., Tran et al. [2022]). Instead, *we propose to condition the priors on weights, biases, and activations in BNNs on the hyperparameters* of a GP kernel. By incorporating GP hyperparameters directly into BNNs, we *bridge the equivalence gap between GPs and BNNs*, finally enabling BNNs to fully replicate the behavior of GPs. In particular, integrating hyperparameters directly within BNNs allows for their optimization, for example, using marginal likelihood (evidence). This facilitates effective model selection, which is a significant novelty compared to previous approaches.

Although our work focuses on conditioning on GP hyperparameters, the proposed conditioning framework is however more general and could be extended to priors dependent on other factors. For example, conditioning on input data could enhance model robustness against input range shifts by allowing the BNN to adapt its priors to the input range. This adaptation could alleviate issues with the locality of the matching BNN to a GP, as discussed in Section3.3.

The conditioning we implemented using hypernetworks [Ha et al., 2017, Chauhan et al., 2024], as $[\sigma, \eta] := \text{hnet}(\gamma|\theta)$, where σ and η are the sets of parameters for the priors on weights and activations, respectively. Here, γ represents the set of conditioning hyperparameters, which are transformed by the hypernetwork hnet. The hypernetwork has its own parameters θ, which are now optimized in Eq. 2 as $\lambda = \{\theta\}$ instead of $\{\sigma, \eta\}$. For a fixed architecture of hnet, σ and η are now fully determined by γ along with θ.

We are the first to test hypernetworks for generating activation parameters and using them for conditioning with hyperparameters. This poses a technical challenge in network design. For example, no architectures other than those based on RBFs showed promising results. Ultimately, we employed an MLP with three hidden layers, each using RBF activations. On top of that, we applied separate linear layers to map to the appropriate outputs: one for each of the prior variances σ and one for the parameters η of the trainable activation.

3.5 LOSS

The loss D measures the divergence between two distributions over functions. While p_{gp} can be sampled and evaluated (for fixed inputs GPs act like Gaussians), p_{nn} is implicitly defined by a BNN, meaning it can be sampled from, but not evaluated directly. Due to the implicit nature of p_{nn}, D must be specified for samples $\{f^l\}$ and approximated using Monte Carlo.

We discovered that the standard losses fail for our task. For example, estimating the empirical entropy term $(\int p_{nn}(f) \log(p_{nn}(f))\, df)$ for KL presents a numerical challenge. Consequently, we followed Tran et al. [2022] and relied on the Wasserstein distance instead (see Tran et al. [2022] for details):

$$\begin{aligned} D &= \left(\inf_{\varsigma \in \Gamma(p_{nn}, p_{gp})} \int_{\mathcal{F}\times\mathcal{F}} d(f, f')^p \varsigma(f, f') df df' \right)^{1/p} \\ &= \sup_{|\Psi|_L \leq 1} \mathbb{E}_{p_{nn}}[\Psi(f)] - \mathbb{E}_{p_{gp}}[\Psi(f)] \end{aligned} \tag{3}$$

However, unlike Tran et al. [2022], we propose to use the 2-Wasserstein metric, which for Multivariate Gaussians (applicable in the case of GPs and wide BNNs – at least approximately) has a *closed-form solution* [Mallasto and Feragen, 2017]:

$$D = ||\mu_1 - \mu_2||_2^2 + \text{Tr}\left(\Sigma_1 + \Sigma_2 - 2\sqrt{\sqrt{\Sigma_1}\Sigma_2\sqrt{\Sigma_1}}\right),$$

where $\mu_{1/2}$, $\Sigma_{1/2}$ are respectively expectations and covariance matrices estimated for p_{nn} and p_{gp} from samples $\{f^l(X)\}$ (we used 512 or 1024 reparameterized samples) evaluated for inputs X. Not only we avoid the internal optimisation due to $\sup_{|\psi|}$, but additionally D can be efficiently computed based on results by Buzuti and Thomaz [2023]. For experiments, we report numerical values of D normalized by number of elements in X.

3.6 OUTPUT STRUCTURE

The optimization objective in Eq. 2 is defined over functions f. The functions are passed through likelihoods to form model outputs y. For regression tasks, a Gaussian likelihood is typically used; for binary classification, Bernoulli; and for multiclass classification, a Categorical likelihood with multivariate f transformed via softmax. Note that the *prior transfer is independent of a likelihood*, meaning it does not rely on how the latent functions f relate to the outputs y. Then, as explained in Sec. 3.5, the optimization can be performed efficiently between Gaussians. If these assumptions

were not satisfied—e.g., if the desired priors are not expressible via a GP – we can revert to the nested optimization for $|\Psi|_L$ as implied by Eq. 3. Overall, we explain our method in terms of GPs and for brevity, we denote the target functional priors by p_{gp}. However, it is important to note that the method is *applicable to arbitrarily specified functional priors*.

For completeness, we follow to briefly discuss also the transfer of priors from GPs to BNNs for multivariate/multi-class-output settings; however, such extensions fall outside of the scope of this paper. In particular, the multivariate models [Rasmussen and Williams, 2005, Alvarez et al., 2012] can be approached using various architectural strategies, depending on the nature of the dependencies among the output dimensions. One approach is to construct independent BNNs for each output dimension, *stacking them side-by-side*, where each BNN is pretrained separately with priors obtained from independent single-output GPs. This approach is appropriate when output dimensions are assumed to be independent, leading to an ensemble of independently trained BNNs. Alternatively, *shared hidden layer* BNN can be used, where a common hidden representation is shared across all output dimensions, thereby capturing potential dependencies between outputs. The shared hidden layer structure reflects the idea of a multi-output or multi-task GP, where dependencies among outputs can be modeled explicitly using coregionalization techniques [Goovaerts, 1997, Bonilla et al., 2008]. The shared hidden layer BNN can thus be endowed with priors derived from these multi-output GPs, ensuring that both spatial and output correlations are incorporated into the BNN model.

3.7 OPTIMISATION CHALLENGES

The problem of finding BNNs behaving like GPs (e.g. inverting covariance equation) in *unidenfiable*. There exist multiple solutions assuring similar quality of the final match. Activations ϕ solving Eq.(2) are not unique. For example:

- The solutions are symmetric w.r.t activity values (y-axis), i.e., for $\phi'(f) = -\phi(f)$, values of covariance given by Eq.(1) are not changed, simply because $(-1)^2 = 1$.
- For $p(f_i^0(x))$ symmetric around 0 (for example, Gaussians), activations with flipped arguments $\phi'(f) = \phi(-f)$ result in the same covariances.
- Scaling activations $\phi'(f) = \alpha\phi(f)$ leads to the same covariances as scaling variances of the output weights as $(\sigma_w^l)' = \alpha \cdot \sigma_w^l$

Given a sufficiently flexible model (like a neural network itself), one can learn to approximate any target activation function to an arbitrary degree of accuracy on a compact domain. This is in line with *the universal approximation theorem*. However, in practice, there are multiple limitations and challenges. A model may require an impractically large number of parameters to approximate certain complex functions to a desired level of accuracy. More complex models may be harder to fit and require more training data. Some functions might require high numerical precision to be approximated effectively, and even if a model can fit a target activation function on a compact set, it might not generalize well outside the training domain. Overall, multiple factors may lead gradient-based optimization to fail for our task. However, such the *poor outcomes will be reflected in low values of the final loss* (Eq. 3). A practical remedy is to rerun optimization, once poorly performing outliers are noticed.

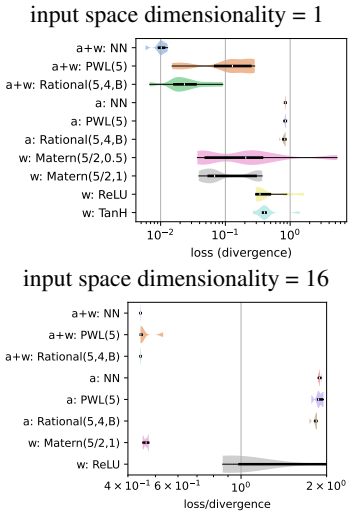


Figure 2: Quality of matching BNNs to the prior of a GP with a Matérn kernel ($\nu = 5/2$, $\ell = 1$) for 1D (top) and 16D inputs (bottom). We evaluate models with trained parameter priors (denoted by *w*), activations (denoted by *a*), and both (denoted by *a+w*). Each label specifies whether a fixed activation (e.g., ReLU) or a specific activation model (e.g., Rational) was employed. Gaussian parameter priors were used by default. If not trained, we set variances to 1., and for the hidden layer, we normalized the variance by its width. The label *w: Matérn* refers to a BNN with the closed-form (fixed) activation as derived by Meronen et al. [2020].

4 FINDINGS

4.1 DOES LEARNING ACTIVATIONS IMPROVE LEARNING OF FUNCTION-SPACE PRIORS?

To map GP function-space priors to a BNN, we can: 1) train its priors on parameters, 2) train both priors and activation, or 3) learn just the activation function. We empirically show that learning activations in addition to priors provide better solutions to the considered problem.

Fig. 2 illustrates the results of an extensive study in which we compare the quality of the GP prior fit for various mod-

Table 1: Results for UCI regression task (Boston dataset) with prior transferred from a GP; comparison between the baseline (using a deep BNN; [Tran et al., 2022]) and ours (single hidden layer BNN with varying widths). *Periodic* activation (Sec. 3.3) and an activation realized by a NN with SiLU own activation (*default*) were used.

Method →	our ($width = 128$, *Periodic act.*)		our ($width = 128$)		our ($width = 1000$)		Baseline
Metric ↓	- regularization	+ regularization	- regularization	+ regularization	- regularization	+ regularization	—
RMSE	2.9067±0.8257	2.8967±0.8258	2.8643±0.8386	**2.8348±0.8371**	2.9189±0.8408	3.0059±0.9068	2.8402±0.8986
NLL	2.5057±0.1870	2.5072±0.1904	2.4937±0.1798	2.4862±0.1763	2.5122±0.1871	2.5971±0.1206	**2.4778±0.1481**

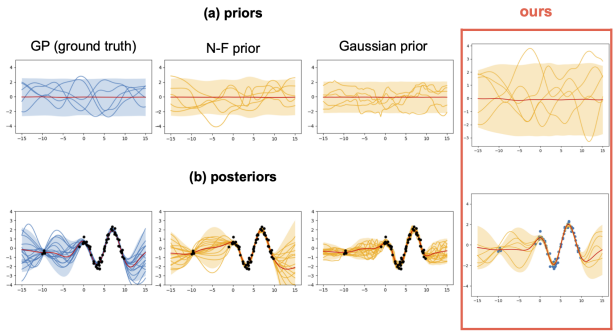


Figure 3: Prior **(a)** and posterior **(b)** predictive distributions for a BNN with trained parameters priors and activations (ours; 4th column), and for Tran et al. [2022] approach with different prior realizations (Gaussian (3 hidden layers; 3rd column) and NF (2 hidden layers; 2nd). The first column illustrates the ground truth (GP). Numerical results complementing the figures we provide in the Appendix.

els of parameter priors and activations. The target was $\mathrm{GP}(0, \text{Matérn}(\nu = 5/2, l = 1))$. For each configuration, we conducted several training iterations and measured the final (converged) loss multiple times. We observe that learning activations alone is not sufficient for good fits, but when combined with learning priors, it significantly improves the results.

Fig. 11 in the Appendix presents the results of a similar experiment conducted for a GP with the Periodic kernel ($\ell = 1, p = 1$). The best performance is observed for the activation introduced in Sec. 3.3, however, increasing the number of layers or neurons can also improve the fit for the activation modeled by an MLP.

We supplement the figure with additional plots showing samples from runs where poor convergence was observed. As explained in Sec. 3.7, due to unfortunate initialization or the insufficient modeling capacity of ϕ, gradient-based optimization may fail to capture a GP functional prior. In such cases, the optimization gets stuck in a local optimum, and the observed final loss differs significantly from the values obtained in other scenarios.

4.2 DO EXPRESSIVE FUNCTION-SPACE PRIORS REQUIRE NETWORKS TO BE DEEP?

Function-space priors require expressive models, which can be achieved either through a multi-layer neural network architecture or with a shallow BNN with more flexible (*e.g.,* learnable) activations. [Tran et al., 2022] explored the former, focusing on multi-layer networks. Instead of considering BNNs with wide layers – where the theoretical results in Sec. 2 apply – they *postulated* that a deep network can model a functional prior after tuning the network's parameter priors to match a target functional prior.

[Tran et al., 2022] investigated several types of priors for BNN weights, including Gaussian, hierarchical, and Normalizing Flow (NF)-based priors. They argue that only the NF prior is flexible enough to capture complex distributions. While NFs [Rezende and Mohamed, 2015] can model intricate priors across all weights, by assigning a distinct prior to each weight they violate the assumptions of the Central Limit Theorem (CLT), preventing the BNN from converging to a GP (*see* Sec. 2). Although [Tran et al., 2022] achieved their best results using this heuristic NF prior, among the considered priors, the theoretical guarantees hold only for the Gaussian and hierarchical priors, and only in the context of wide BNNs.

Beyond the lack of theoretical justification, using complex priors such as NFs or requiring deep networks introduces additional challenges in posterior inference. MCMC-based methods [Chen et al., 2014, Del Moral et al., 2006] become computationally expensive, while approximate inference techniques [Hoffman et al., 2013, Ritter et al., 2018] may lack expressiveness.

In Fig. 4 we compare our approach against the behavior of their methods for varying numbers of layers and activations. Specifically, we evaluate posterior in a 1D regression task (Fig. 3), both on training data and in out-of-distribution (OOD) regions. The results show that multiple hidden layers are required for the Gaussian prior. In contrast, for the NF priors, increasing depth leads to worse performance in OOD regions – illustrating an example of the heuristic's failure. For both methods, we note high sensitivity to activation change. NFs perform best for one or two layers, but only with RBF activation. For Swish/ReLU they fail to capture the data (high RMSE/NLL). Please see also Sec. E.

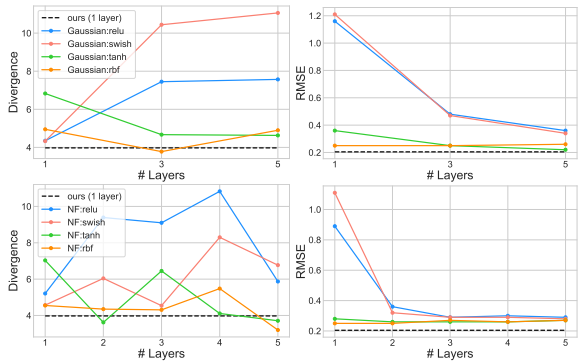


Figure 4: Comparison of posterior predictive quality across the methods from [Tran et al., 2022] for the problem from Fig. 3, considering different priors, activation functions, and numbers of hidden layers. We evaluate both *out-of-distribution* (in-between region) performance (**left**, Wasserstein divergence) and *in-distribution* performance (**right**, RMSE). For the Gaussian prior (**top**), the results align with the assumption by the authors – adding more layers improves the GP approximation. Contrary, for the Normalizing Flow prior (**bottom**), the best performance is observed with one or two hidden layers, while deeper networks suffer from instability, highlighting failure of the heuristic. Our method (**black line**) consistently outperforms the baselines.

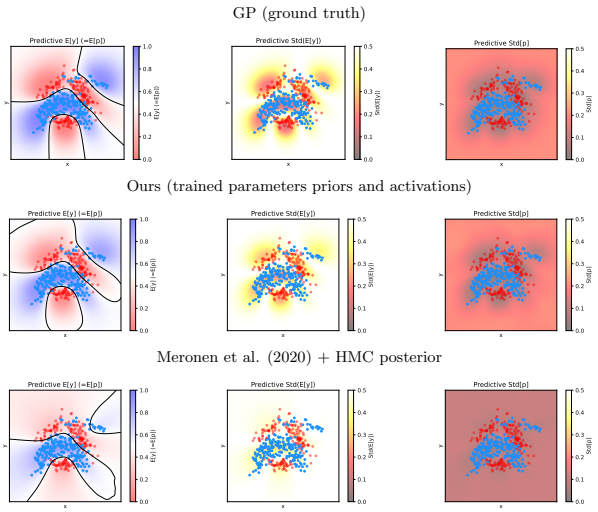


Figure 5: Posterior predictive distributions for a BNN with trained parameter priors and activations (ours; 2nd row) and for a BNN with analytically derived activations (3rd row). The first column illustrates class probabilities, the second column shows the total variance in class predictions, and the last column depicts the epistemic uncertainty component of the total uncertainty. Here, we show posteriors obtained using HMC, while in Sec. D.4 of the Appendix, we provide an extended version of the figure that includes results obtained using the original code from Meronen et al. [2020] (which uses MC-Dropout instead of MCMC), along with additional numerical results.

For further evaluation, we compare our approach with Tran et al. [2022] on a range of regression tasks, including both synthetic and real-world datasets (e.g., the Boston dataset). The results are shown in Fig.3 and Tab.1, respectively. Additional discussion and results for several other regression tasks are provided in the Appendix. We closely follow the settings picked by Tran et al. [2022] and still observe that our method performs comparably or better than the baseline, while requiring only a single-hidden-layer BNN.

Additionally, we checked the findings from [Wu et al., 2024] that moment matching helps biasing optimisation towards better solutions. For the UCI regression task, we included an additional moment matching regularization term: $\mathcal{R}(p_{nn}, p_{gp}) = \left(\mathbb{E}_{p_{nn}}[var(f)] - \mathbb{E}_{p_{gp}}[var(f)]\right)^2 + \left(\mathbb{E}_{p_{nn}}[kurtosis(f)] - \mathbb{E}_{p_{gp}}[kurtosis(f)]\right)^2 + \left(\mathbb{E}_{p_{nn}}[skeweness(f)] - \mathbb{E}_{p_{gp}}[skeweness(f)]\right)^2$. However, we have *not* observed any significant improvements.

4.3 CAN LEARNED ACTIVATIONS MATCH PERFORMANCE OF CLOSED-FORM ONES?

Deriving suitable neural activations to match BNNs to GPs is a formidable challenge. For example, Meronen et al. [2020] derived recently an analytical activation function for the popular Matérn kernel. However, their solution relies on the assumption that the priors on the BNN weights follow binary white noise. Such the prior hinders existing posterior learning algorithms, making the proposed solution impractical. Nevertheless, Meronen et al. [2020] shows that their solution can work with more convenient Gaussian priors, albeit as a suboptimal approximation. In this work, we propose a general gradient-based alternative that can also be applied to the GPs with the Matérn kernel and we demonstrate that our black-box approach can match or even surpass the performance of the aforementioned approximation.

The empirical evaluation we performed for the 2D data classification problem following the setting of Meronen et al. [2020]. For our method, we used a BNN with Gaussian priors with trained variances, where the activation function was modeled by an NN with a single hidden layer consisting of 5 neurons using SiLU activation. For the baseline, we trained just variances and used the fixed activation. Posterior distributions were generally obtained using a HMC sampler.

The results presented in Fig. 5 (and extended results provided in Sec. D.4 in the Appendix) demonstrate that our method enhances the match between the posteriors of a BNN and the desired target GP. Note that Meronen et al. [2020] originally used MC Dropout to obtain the posteriors, achieving much worse results than those obtained with HMC. For fairness in comparison, we tested both MC Dropout and HMC. In terms of performance, our model not only *captures class probabilities accurately but also*

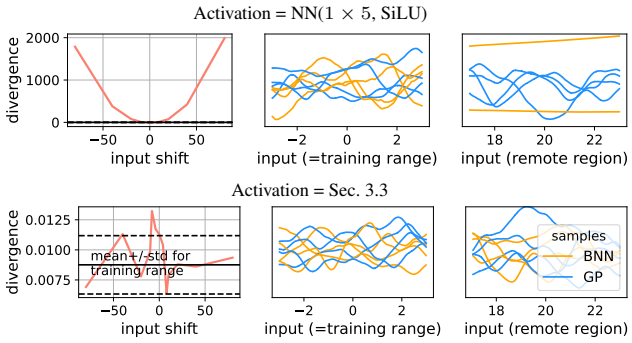


Figure 6: Sensitivity of priors transferred from a GP to a BNN with respect to shifts in the input range for learned activations: (top) activation realized by a NN with a single hidden layer containing 5 neurons and SiLU own activation; (bottom) periodic activation introduced in Sec. 3.3, which is robust to input range shifts. Priors were trained on random inputs $X \in [-3, 3]$. The first column shows the mismatch between the desired target GP and the trained BNN prior for shifted input ranges, measured by the 2-Wasserstein loss (lower is better). The remaining columns present samples from the priors for two ranges: one matching the training range and one far from it.

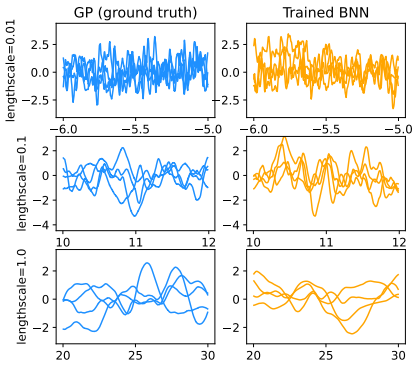


Figure 7: Comparison of samples from a GP (left) and a trained BNN (right) using priors and activations (periodic activation as in Sec. 3.3) conditioned on the GP's hyperparameter (e.g., lengthscale). The BNN was trained on random inputs $X \in [-3, 3]$. Regardless of input range and the hyperparameter's value, samples from both models appear *indistinguishable*, i.e., the BNN imitates the GP perfectly.

adeptly handles the total variance in class predictions and the epistemic uncertainty component, which are crucial for robust decision-making under uncertainty.

4.4 CAN STATIONARITY BE INDUCED WITH LEARNED PERIODIC ACTIVATIONS?

Trained priors for BNNs are inherently local, effectively mimicking GP behavior only within the range of training inputs X. As illustrated in Fig. 6 (top), outside this range, the BNN's learned prior may diverge from the desired behavior, and properties such as stationarity or smoothness may not be preserved beyond the observed input domain. We however can not only detect this issue by noting high values of the divergence D, but also mitigate it by using activations with appropriate structural biases. Specifically, the periodic activation introduced in Sec. 3.3 proves to be robust to input domain translations, as demonstrated in Fig. 6 (bottom).

4.5 IS MODEL SELECTION FOR BNNS WITH TRANSFERRED PRIORS POSSIBLE?

A BNN with a functional prior learned from a GP essentially inherits the properties dictated by the GP, including those determined by specific values of the GP's hyperparameters. While the hyperparameters of the source GP can be tuned and optimized, the BNN does not inherently possess this capability, requiring researchers to either pre-optimize the hyperparameters or look for suitable workarounds [Tran et al., 2022]. Conditioning the priors on weights, biases, and activation, as explained in Sec. 3.4, addresses this limitation, ultimately closing the BNN-GP equivalence gap. Fig. 7 shows results for a prior trained to adapt to a varying lengthscale of a Matérn kernel. The prior closely follows the behavior of the ground truth model.

To validate the usefulness of this method, we performed marginal likelihood maximization on the data from Fig. 3, using gradient-based optimization for both the GP with the Matérn kernel and our BNN. The optimal lengthscale found for the BNN ($\ell = 1.65$) closely matches the one found for the GP ($\ell = 1.84$).

4.6 DOES IT SCALE?

Learning posteriors for Gaussian Processes (GPs) is challenging, as exact inference scales cubically with the number of data points Rasmussen and Williams [2005]. To address this, scalable approximations such as Stochastic Variational Gaussian Processes (SVGP) [Hensman et al., 2015] have been proposed for practical applications. SVGP employs stochastic variational gradients to optimize a fixed set of inducing points, typically (and also here) 1024.

On the other hand, there exist a number of posterior inference methods for BNNs. For the experiments presented in the previous sections, we utilized HMC. However, Variational Inference (VI) with Normalizing Flows [Rezende and Mohamed, 2015] offers a scalable alternative, albeit with challenges related to both the accuracy of posterior approximation and selection of hyperparameters. However, the former limitation can be mitigated by employing RealNVPs [Dinh et al., 2016] with appropriate capacity for approximating posteriors. Agrawal and Domke [2024] showed

Table 2: Mean test-set NLL and RMSE for the HouseElectric dataset. The optimal prior lengthscale ℓ^* and additive Gaussian noise were *learned* by maximizing $\mathcal{L}$ with $\beta = 1$. Results for $\ell = 10\ell^*$ and $\ell = 0.1\ell^*$ highlight the importance of prior hyperparameter selection. Additionally, using a model of activation function with more neurons improves performance. The best results are achieved by combining these modifications with $\beta = 0.4$.

	Ours with $\beta = 1$				Ours with $\beta = 0.4$		$\pm$	SVGP	Exact GP
	learned	$\ell = 0.1 \times \ell^*$	$\ell = 10 \times \ell^*$	$\ell = \ell^*$ & improved activation	$\ell = \ell^*$	$\ell = \ell^*$ & improved activation	std.	(baseline)	(BBMM)
$\downarrow$ Test RMSE	0.0549	0.0550	0.0560	0.0550	0.0537	**0.0535**	≤ 0.0007	0.0566	0.055
$\downarrow$ Test NLL	-1.4925	-1.4850	-1.4550	-1.4980	-1.5107	**-1.5155**	≤ 0.0064	-1.4600	$-0.152^\dagger$

† Due to lack of available implementation, we failed to reproduce results from Wang et al. [2019] and provide here values copied directly from the paper.

that RealNVPs can exhibit sufficient fidelity to match posteriors for models with complex priors.

For VI learning is performed using gradients of

$$\mathcal{L} = \mathbb{E}_{q(w|\zeta)} \left[\log p(\mathcal{D}|w)\right] - \beta \cdot \mathrm{KL}\left[q(w|\zeta) \,\|\, p(w)\right]$$
$$\approx \frac{1}{S} \sum_{w \sim q(w|\zeta)} \left(\frac{|\mathcal{D}|}{|\mathcal{B}|} \log p(\mathcal{B}|w) + \beta \cdot (p(w) - q(w|\zeta)) \right)$$

where we estimate the Evidence Lower Bound (ELBO) objective for Stochastic Variational Inference (SVI) [Hoffman et al., 2013] using minibatches $\mathcal{B} \subset \mathcal{D}$ with $|\mathcal{B}| = 10{,}240$. The expectation is approximated with $S = 128$ reparameterized Monte Carlo samples, where $q(w|\zeta)$ is the posterior approximation modeled by a RealNVP.

Tab. 2 summarizes the test set performance on the UCI Household Power Consumption (HouseElectric) regression dataset. This dataset contains approximately 2M rows, split into training (7/9) and test (2/9) subsets, and is the largest dataset used to evaluate GPs [Wang et al., 2019].

The regression model for the data has two hyperparameters: the prior *lengthscale* ℓ of the Matérn ($\nu = 3/2$) kernel and the additive *observation noise* scale in the Gaussian likelihood. We selected these hyperparameters by maximizing the ELBO $\mathcal{L}$, which for $\beta = 1$ provides a proper lower bound for the evidence. This was possible thanks to employing the conditional priors introduced in Sec. 3.4.

Variational Inference (VI) often performs better with $\beta < 1$ [Higgins et al., 2017], which also holds in our case - setting $\beta = 0.4$ (a typical value) improved both RMSE and NLL. However, as indicated by discrepancies with the Exact GP results, this improvement is likely due to the reduced influence of the learned GP prior, which may be suboptimal for this particular dataset.

To analyze the importance of prior selection, we also computed posteriors for lengthscales reduced by a factor of ten ($\ell = 0.1 \times \ell^*$) and increased tenfold ($\ell = 10 \times \ell^*$). In both cases, we observed a drop in NLL, highlighting the sensitivity of performance to choice of the prior.

We further tested a learned prior with slightly *improved activation* model. We used a neural network with two hidden layers of width 10, instead of the default single layer

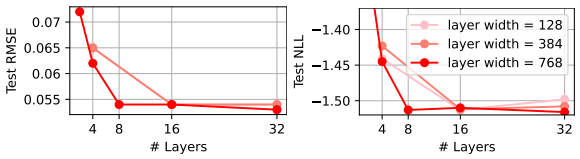


Figure 8: Impact of the architecture of $q(w|\zeta)$ parameterized by *RealNVP* [Dinh et al., 2016]: (1) although adding more layers improves performance, the gains become marginal for > 16 layers; (2) wider flows require fewer layers to achieve performance saturation.

with five neurons. This resulted in a 0.15% improvement in the 2-Wasserstein loss. Although this represents a minor enhancement in prior fidelity, it still led to noticeable improvements in posterior quality.

Finally, we investigated the capacity of the RealNVP used for posterior approximation. Fig. 8 compares test set performance for varying the number of flow layers and the width of each layer. We observed that increasing the number of layers improves performance, but the gains become marginal beyond 16 layers, and that wider flows require fewer layers to achieve performance saturation.

5 CONCLUSION

In this paper, we addressed the problem of transferring functional priors for wide Bayesian Neural Networks to replicate desired a priori properties of Gaussian Processes. Previous approaches typically focused on learning distributions over weights and biases, often requiring deep BNNs for sufficient flexibility. We proposed an alternative approach by also learning activations, providing greater adaptability for shallow models and eradicating the need for task-specific architectural designs. To the best of our knowledge, we are the first to explore learning activations in this context. Moreover, to further enhance adaptability of these transferred priors, we came up with the idea of conditioning them with hypernetworks, which opens a new interesting future research direction. To demonstrate the flexibility and effectiveness of the proposed methods, we conducted a comprehensive experimental study validating our ideas.

Author Contributions

Marcin Sendera actively participated in shaping the project across all its stages, contributing both during discussions and through hands-on development. He explored and assessed multiple variants of the Wasserstein loss, as well as a range of learnable activation functions proposed in the literature, including piecewise linear (PWL), Padé, and others. He proposed to use the closed-form solution for the Wasserstein metric. He was primarily responsible for implementing and executing the experiments reported in Figures 3 and 4, and in Table 1, ensuring the reproducibility and robustness of the results. He contributed to writing of the manuscript, particularly in sections related to empirical results.

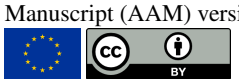
Amin Sorkhei participated in discussions at the initial stage of the project. He implemented and conducted the initial experiments involving the mapping of Gaussian Process priors to Bayesian Neural Networks using the Kullback-Leibler divergence and a learnable activation parameterized by a neural network.

Tomasz Kuśmierczyk conceived the project and supervised its development across all stages, including the core theoretical contributions and experimental design. He introduced the idea of learning activations and proposed the use of Wasserstein-based loss and regularizations. He proposed to use neural networks to parameterize learnable activations. He developed and implemented the periodic (Section 3.3) and conditioned activations (Section 3.4). He was responsible for implementing and executing the experiments presented in Figures 2, 5, 6, 7, 8, and Table 2. He was responsible for the majority of the manuscript writing as well as for shaping its final form.

Acknowledgements

We thank Nikolay Malkin and Jacek Tabor for helpful discussions at early stages of this project and revising the manuscript.

This research is part of the project No. **2022/45/P/ST6/02969** co-funded by the National Science Centre and the European Union Framework Programme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie grant agreement No. 945339. For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.



This research was in part funded by National Science Centre, Poland, **2022/45/N/ST6/03374**.

We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. **PLG/2023/016302**.

References

Abhinav Agrawal and Justin Domke. Disentangling impact of capacity, objective, batchsize, estimators, and step-size on flow VI. *arXiv preprint arXiv:2412.08824*, 2024.

Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4 (3):195–266, 2012. doi: 10.1561/2200000036.

Edwin V Bonilla, Ying Chai, and Christopher K Williams. Multi-task Gaussian Process prediction. In *Advances in Neural Information Processing Systems*, pages 153–160, 2008.

David R Burt, Sebastian W Ober, Adrià Garriga-Alonso, and Mark van der Wilk. Understanding variational inference in function-space. *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

Lucas F Buzuti and Carlos E Thomaz. Fréchet autoencoder distance: a new approach for evaluation of generative adversarial networks. *Computer Vision and Image Understanding*, 235:103768, 2023.

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57 (9):1–29, 2024.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, pages 1683–1691. PMLR, 2014.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo Samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68 (3):411–436, 05 2006. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2006.00553.x.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Daniel Flam-Shepherd, James Requeima, and David Duvenaud. Mapping Gaussian Process priors to Bayesian Neural Networks. In *NIPS Bayesian deep learning workshop*, volume 3, 2017.

Daniel Flam-Shepherd, James Requeima, and David Duvenaud. Characterizing and warping the function space of Bayesian Neural Networks. In *NeurIPS Workshop on Bayesian Deep Learning*, page 18, 2018.

Vincent Fortuin. Priors in bayesian deep learning: A review. *International Statistical Review*, 90(3):563–591, 2022. doi: https://doi.org/10.1111/insr.12502.

Pierre Goovaerts. *Geostatistics for natural resources evaluation*. Oxford University Press on Demand, 1997. ISBN 978-0195115383.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. In *ICLR*. International Conference on Representation Learning, 2017.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable Variational Gaussian Process Classification. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 351–360, San Diego, California, USA, 09–12 May 2015. PMLR.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR2017, Conference Track Proceedings*, 2017.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John William Paisley. Stochastic variational inference. *Journal of Machine Learning Research (JMLR)*, 14:1303–1347, 2013.

Jiri Hron, Yasaman Bahri, Roman Novak, Jeffrey Pennington, and Jascha Sohl-Dickstein. Exact posterior distributions of wide Bayesian Neural Networks. *arXiv preprint arXiv:2006.10541*, 2020.

Jiri Hron, Roman Novak, Jeffrey Pennington, and Jascha Sohl-Dickstein. Wide Bayesian neural networks have a simple weight posterior: theory and accelerated sampling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8926–8945. PMLR, 17–23 Jul 2022.

Theofanis Karaletsos and Thang D Bui. Hierarchical Gaussian Process priors for Bayesian Neural Network weights. *Advances in Neural Information Processing Systems*, 33: 17141–17152, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian Processes. *arXiv preprint arXiv:1711.00165*, 2017.

Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Anton Mallasto and Aasa Feragen. Learning from uncertain curves: The 2-Wasserstein metric for Gaussian Processes. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Takuo Matsubara, Chris J Oates, and François-Xavier Briol. The ridgelet prior: A covariance function approach to prior specification for Bayesian Neural Networks. *The Journal of Machine Learning Research*, 22(1):7045–7101, 2021.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian Process behaviour in wide deep Neural Networks. *ICLR*, 2018.

Lassi Meronen, Christabella Irwanto, and Arno Solin. Stationary activations for uncertainty calibration in deep learning. *Advances in Neural Information Processing Systems*, 33:2338–2350, 2020.

Lassi Meronen, Martin Trapp, and Arno Solin. Periodic activation functions induce stationarity. *Advances in Neural Information Processing Systems*, 34:1673–1685, 2021.

Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *International Conference on Learning Representations*, 2019.

Pablo Morales-Alvarez, Daniel Hernández-Lobato, Rafael Molina, and José Miguel Hernández-Lobato. Activation-level uncertainty in deep neural networks. In *International Conference on Learning Representations*, 2020.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0387947248.

Tim Pearce, Russell Tsuchida, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. Expressive priors in Bayesian Neural Networks: Kernel combinations and periodic functions. In *Uncertainty in Artificial Intelligence*, pages 134–144. PMLR, 2020.

Janis Postels, Hermann Blum, Yannick Strümpler, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. The hidden uncertainty in a neural network's activations. In *Proceedings of the Bayesian Deep Learning Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, 2015.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *6th international conference on learning representations, ICLR 2018-conference track proceedings*, volume 6. International Conference on Representation Learning, 2018.

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

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 5690–5701, 2019.

Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. All you need is a good functional prior for bayesian deep learning. *Journal of Machine Learning Research*, 23(74):1–56, 2022.

Russell Tsuchida, Fred Roosta, and Marcus Gallagher. Richer priors for infinitely wide multi-layer perceptrons. *arXiv preprint arXiv:1911.12927*, 2019.

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact Gaussian Processes on a million data points. *Advances in Neural Information Processing Systems*, 32, 2019.

Veit David Wild, Robert Hu, and Dino Sejdinovic. Generalized variational inference in function spaces: Gaussian measures meet bayesian deep learning. *Advances in Neural Information Processing Systems*, 35:3716–3730, 2022.

Christopher Williams. Computing with infinite networks. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

Mengjing Wu, Junyu Xuan, and Jie Lu. Functional wasserstein bridge inference for bayesian deep learning. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

Revisiting the Equivalence of Bayesian Neural Networks and Gaussian Processes: On the Importance of Learning Activations (Supplementary Material)

Marcin Sendera[*1,2] **Amin Sorkhei** **Tomasz Kuśmierczyk**[*1]

[1]Jagiellonian University
[2]Mila, Université de Montréal

We supplement the main text here with a description of the related work, a discussion of computational costs, and additional details of the experimental setup, followed by additional numerical results. Our code we made publicly available [1].

A RELATED WORK

The relationship between (B)NNs and GPs has been a subject of significant research interest. Much of this work has been motivated by the desire to better understand NNs through their connection to GPs. A seminal result by Neal [1996], Williams [1996], later extended to deep NNs by Lee et al. [2017], Matthews et al. [2018], shows that infinitely wide layers in NNs exhibit behavior akin to GPs. While we explore a similar setting, our focus is the inverse: implementing function-space priors, specifically GP-like, within BNNs.

This problem was previously addressed by Meronen et al. [2020], who proposed an activation function for BNNs that corresponds to the popular Matérn kernel. Due to the complexity of deriving analytical solutions, other works, such as Flam-Shepherd et al. [2017, 2018], Tran et al. [2022], have explored gradient-based optimization to learn priors on weights and biases. However, their approach requires deep networks with complex distributions and additionally, presents a challenge in learning posterior. In contrast, we achieve high-fidelity functional priors in shallow BNNs by matching *both* priors on parameters and activations.

Finding accurate posteriors for deep and complex models such as BNNs is notoriously challenging due to their high-dimensional parameter spaces and complex likelihood surfaces. However, for single-hidden-layer wide BNNs, it has been recently shown that posterior sampling via Markov Chain Monte Carlo can be performed efficiently [Hron et al., 2022]. Moreover, research has demonstrated that the exact posterior of a wide BNN weakly converges to the posterior corresponding to the GP that matches the BNN's prior [Hron et al., 2020]. This allows BNNs to inherit GP-like properties while preserving their advantages.

An alternative line of work touching efficient learning for BNNs in function space has explored variational inference. Sun et al. [2019] introduced functional variational BNNs (fBNNs) that maximize an Evidence Lower Bound defined over functions. However, Burt et al. [2020] highlighted issues with using the Kullback-Leibler divergence in function space, leading to ill-defined objectives. To overcome this, methods like Gaussian Wasserstein inference [Wild et al., 2022] and Functional Wasserstein Bridge Inference [Wu et al., 2024] leverage the Wasserstein distance to define well-behaved variational objectives. These approaches try to incorporate functional priors into variational objective for deep networks. In contrast, in our method transfer of priors to shallow BNNs remains independent and fully separated from an inference method.

Finally, other works worth mentioning include Meronen et al. [2021], who explored periodic activation functions in BNNs to connect network weight priors with translation-invariant GP priors. Pearce et al. [2020] derived BNN architectures to mirror GP kernel combinations, showcasing how BNNs can produce periodic kernels. Karaletsos and Bui [2020] introduced a hierarchical model using GP for weights to encode correlated weight structures and input-dependent weight priors, aimed at

[1]https://github.com/gmum/bnn-functional-priors

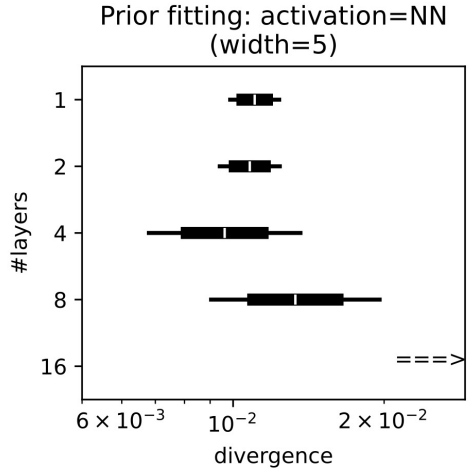


Figure 9: Prior matching quality: fixed width.

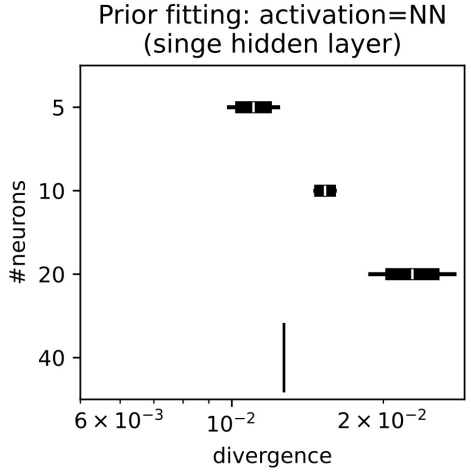


Figure 10: Prior matching quality: fixed depth.

regularizing the function space. Matsubara et al. [2021] proposed using ridgelet transforms to approximate GP function-space distributions with BNN weight-space distributions, providing a non-asymptotic analysis with finite sample-size error bounds. Finally, Tsuchida et al. [2019] extended the convergence of NN function distributions to GPs under broader conditions, including partially exchangeable priors. A more detailed discussion of related topics can be found for example, in Sections 2.3 and 4.2 of [Fortuin, 2022].

B COMPUTATIONAL AND ARCHITECTURAL ADVANTAGES

Our method involves an initial prior fitting step, which precedes the posterior inference step. The prior fitting step takes approximately one hour for the HouseElectric dataset (Section 4.6); however, one can alternatively use a prior from a library of pre-trained priors, thereby amortizing the cost of this step.

Our experiments show that substituting fixed activations (ReLU/TanH, ~ 0.5s/iteration) with learnable NN activations (single hidden layer, 5 neurons, ~ 2.3s/iteration) or periodic activations (~ 2.7s/iteration) increases the time per iteration by approximately a factor of 4 on our 16-core CPU setup. On the other hand, learning parameter priors alongside activations has only a marginal effect on computation time compared to fixing priors and only learning activations.

Additionally, we conducted a study measuring both the quality of prior fit and the runtime for a prior modeled by a neural network with varying numbers of layers and widths. The respective results can be found in Figures 9, 10 and Tables 3, 4. These results suggest that increasing the number of layers up to a certain depth (e.g., 4 layers in this case) may marginally improve the fidelity of the solutions, whereas increasing the network width tends to make convergence harder. In either case, using a larger neural network consistently leads to longer convergence times.

Table 3: Prior matching time
(activation=NN, width=5)

#layers	iteration time [s]
1	2.34
2	3.10
4	4.57
8	7.38
16	15.03

Table 4: Prior matching time
(activation=NN, single hidden layer)

#neurons	iteration time [s]
5	2.34
10	2.59
20	3.95
40	7.59

Once the prior is established (or a pre-trained one is used), posterior inference can be performed. Our approach offers here computational advantages, particularly for large datasets where exact GP inference is prohibitive. For example, RealNVP-based variational inference (Section 4.6) converges in about 2 hours on a single GPU for HouseElectric, while exact GP

inference can take over three days on a single GPU, or 1.5 hours when parallelized on 8 GPUs using methods such as BBMM [Wang et al., 2019]. Furthermore, during posterior inference using HMC for the data in Fig. 3, we observed no significant differences in step time ($\sim$ 1s) between fixed and learnable activations, though this may vary with implementation and hardware.

Finally, beyond computational performance, BNNs provide greater modularity and extensibility. Single-layer BNNs can serve as components in larger neural architectures, for example, as Bayesian last layers (with remaining layers trained point-wise). This enables leveraging architectural innovations from deep learning in ways that GPs cannot easily accommodate. Our method could replace existing approaches in models like SNGP [Liu et al., 2020], potentially offering improved prior control (as random Fourier feature GPs are limited to specific stationary kernels), more off-the-shelf end-to-end training options (e.g., SVGD), and potentially enhanced performance, as suggested by our comparisons against SVGP in Section 4.6.

C LEARNED GP KERNELS

In this paper, we show that our approach allows for inducing a GP-like behavior onto single-layer BNNs without being restricted to a family of GP kernels. In particular, we induced the following GP kernel behavior, while performing the mentioned experiments:

- Matérn (3/2) - house electric regression;
- Matérn (5/2) - inducing stationarity, ablation on learning activations and priors, 2d classification;
- RBF - 1d regression;
- RBF + ARD - UCI regression;
- Periodic - ablation on learning activations and priors.

D EXPERIMENTAL DETAILS

D.1 CAN LEARNED ACTIVATIONS ACHIEVE MORE FAITHFUL FUNCTION-SPACE PRIORS? – ADDITIONAL FIGURES

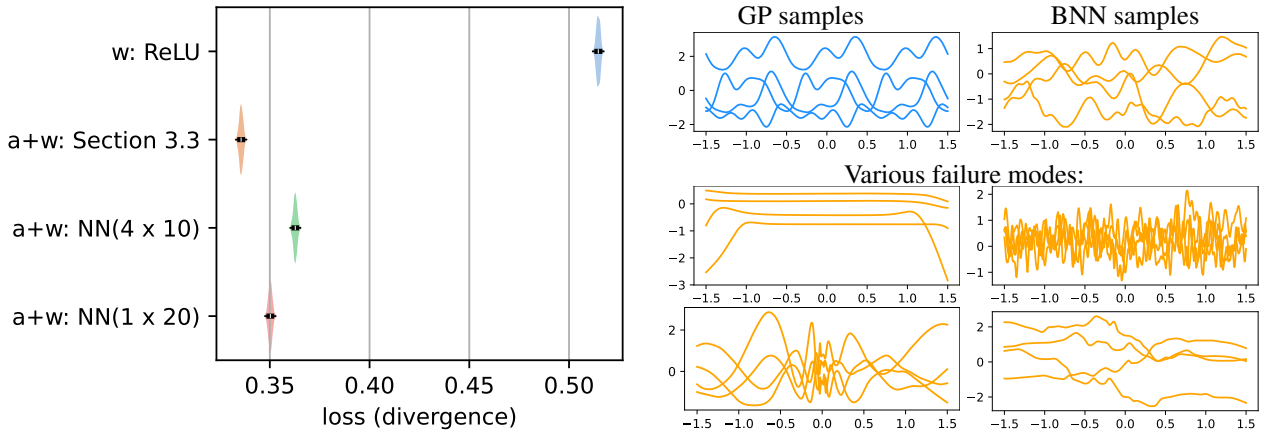


Figure 11: Quality of matching BNNs to the prior of a GP with the Periodic kernel ($\ell = 1$, $p = 1$). (Left): Evaluation of models with trained parameter priors (denoted by w) and with trained both weights priors and activations (denoted by a+w). (Right): Samples from the ground truth GP, a well-fitted BNN and BNN outliers with poor final loss. As explained in Sec. 3.7, or because of the insufficient modeling capacity of ϕ, gradient-based optimization may fail to capture a GP functional prior.

D.2 1-DIMENSIONAL REGRESSION - COMPARISON WITH Tran et al. [2022]

In order to illustrate the abilities of our approach on a standard regression task and for a fair comparison with another method optimizing the Wasserstein distance, we followed the exact experimental setting presented in Tran et al. [2022]. We used a GP with an RBF kernel (*length scale* ℓ=0.6, *amplitude*=1.0) and a Gaussian likelihood (*noise variance*=0.1) as the ground truth.

For a baseline, we utilized the approach by Tran et al. [2022], consisting of a BNN with 3 layers and 50 neurons each, using TanH activation function. We considered all three prior realizations presented in that work: a simple Gaussian prior, a hierarchical prior, and a prior given by a Normalizing Flow. On the other hand, our model configuration consisted of a BNN with Gaussian priors centered at **0** and trained variances, where the activation function was modeled by a neural network with a single hidden layer with 5 neurons and SiLU activation. Posterior distributions were obtained using an HMC sampler.

We found that our approach allows for achieving the same or better results than the baseline by Tran et al. [2022], both in visual and numerical comparisons. We obtained better results in terms of distributional metrics as presented in Tab. 5 and Tab. 6 and getting better (visually) posterior distributions (see Fig. 3) without utilizing any computationally heavy prior realization (like, *e.g.,* Normalizing Flow).

Table 5: Comparison of trained (function-space) priors in a 1D regression task: [Tran et al., 2022] vs. our method. Whereas Tran et al. [2022] considered three different priors on parameters (including Gaussian, Hierarchical, and Normalizing Flow) for deep BNNs, our implementation consists of a single-hidden-layer BNN with standard Gaussian priors with learned variances and trained activation. The methods were compared using a set of distributional metrics against the ground truth provided by a GP. We compared prior distributions of functions over a range of X covering regions with and without data (to account also for overconfidence far from data). The lower, the better.

Setting	1-Wasserstein	2-Wasserstein	Linear-MMD	Poly-MMD $\times 10^3$	RBF-MMD	Mean-MSE	Mean-L2	Mean-L1	Median-MSE	Median-L2	Median-L1
Priors:											
Gaussian prior	7.52	7.77	-6.08	1.158	0.036	0.003	0.058	0.045	0.004	0.063	0.052
Hierarchical prior	8.14	8.37	0.22	1.421	0.027	0.004	0.064	0.059	0.008	0.090	0.074
Normalizing Flow prior	7.61	8.09	-8.192	0.393	0.019	0.004	0.061	0.051	0.004	0.066	0.052
ours	**6.86**	**7.06**	**-9.40**	**-0.016**	**0.008**	**0.001**	**0.032**	**0.027**	**0.002**	**0.048**	**0.041**

Table 6: Comparison of trained (function-space) posteriors in a 1D regression task: [Tran et al., 2022] vs. our method. Whereas Tran et al. [2022] considered three different priors on parameters (including Gaussian, Hierarchical, and Normalizing Flow) for deep BNNs, our implementation consists of a single-hidden-layer BNN with standard Gaussian priors with learned variances and trained activation. The methods were compared on RMSE and NLL of test data.

Setting	RMSE	NLL
Posteriors:		
Gaussian prior	0.2559	**0.2456**
Hierarchical prior	0.2461	0.2613
Normalizing Flow prior	0.2534	0.2550
ours	**0.2045**	0.3044

D.3 UCI REGRESSION - COMPARISON WITH Tran et al. [2022]

In addition, we compare against Tran et al. [2022] also on UCI regression tasks, which have more input dimensions d – between 8 and 12, while having 1-dimensional output. We use 10-split of training datasets as in typical in this direction of research.

For the baseline, we followed the exact experimental setting presented in Tran et al. [2022]. We used a GP with an RBF kernel (*length scale* $\ell = \sqrt{2.0 * input\_dim}$ and *amplitude*=1.0, ARD) and a Gaussian likelihood as the ground truth. We set the noise variance value and architectures for the specific datasets as in the mentioned baseline paper.

On the other hand, our model configuration consisted of a BNN with Gaussian priors centered at $\mathbf{0}$ and trained variances, where the activation function was modeled by a neural network with a single hidden layer with 5 neurons and SiLU activation. Posterior distributions were obtained using an HMC sampler.

We found that our approach allows for achieving the same or better results than the baseline, in terms of numerical values (*RMSE* and *NLL*) as presented in Tab. 7.

Table 7: Extended results for a set of UCI regression tasks (various input dimensionality d) with prior transferred from a GP; comparison between the baseline (using a deep BNN; [Tran et al., 2022]) and ours (single hidden layer BNN with 128 neurons; w/o regularization) and an activation realized by a NN with SiLU own activation were used.

Dataset →	Boston ($d = 12$)		Concrete ($d = 8$)		Energy ($d = 8$)		Protein ($d = 9$)		Wine ($d = 11$)	
Method ↓ Metric →	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*
baseline	$2.8402_{\pm 0.8986}$	$2.4778_{\pm 0.1481}$	$4.4628_{\pm 0.7511}$	$2.9728_{\pm 0.0876}$	$0.3431_{\pm 0.0613}$	$0.3482_{\pm 0.1607}$	$0.5038_{\pm 0.0093}$	$0.7447_{\pm 0.0142}$	$0.4736_{\pm 0.0420}$	$0.8725_{\pm 0.0285}$
ours	$2.8643_{\pm 0.8386}$	$2.4937_{\pm 0.1798}$	$4.9143_{\pm 0.7528}$	$3.0201_{\pm 0.1011}$	$0.3692_{\pm 0.0566}$	$0.4064_{\pm 0.1347}$	$0.4957_{\pm 0.0058}$	$0.7251_{\pm 0.0094}$	$0.4833_{\pm 0.0398}$	$0.8673_{\pm 0.0255}$

D.4 CAN LEARNED ACTIVATIONS MATCH PERFORMANCE OF CLOSED-FORM ONES? – COMPARISON WITH Meronen et al. [2020]

For the comparison against Meronen et al. [2020], we closely followed their experimental setting and used a GP with a Matérn kernel ($\nu = 5/2$, ℓ=1) as the ground truth. For a baseline, we utilized the approach by Meronen et al. [2020], where the authors derived analytical activations to match the Matérn kernel. Our model configuration consists of a BNN with Gaussian priors and trained variances, where the activation function was modeled by a neural network with a single hidden layer consisting of 5 neurons using SiLU activation. Posterior distributions were generally obtained using an HMC sampler, except in the case of Meronen et al. [2020], where the original implementation employed MC Dropout to approximate the posterior. However, for completeness and fairness in comparison, we also generated results using an HMC-derived posterior for the model by Meronen et al. [2020]. Additional tests (see Tab. 8) were conducted on BNNs with fixed parameter priors (*Default*=Gaussian priors with a variance of 1, and *Normal*=Gaussian priors normalized by the hidden layer width) and with activation functions including ReLU and TanH.

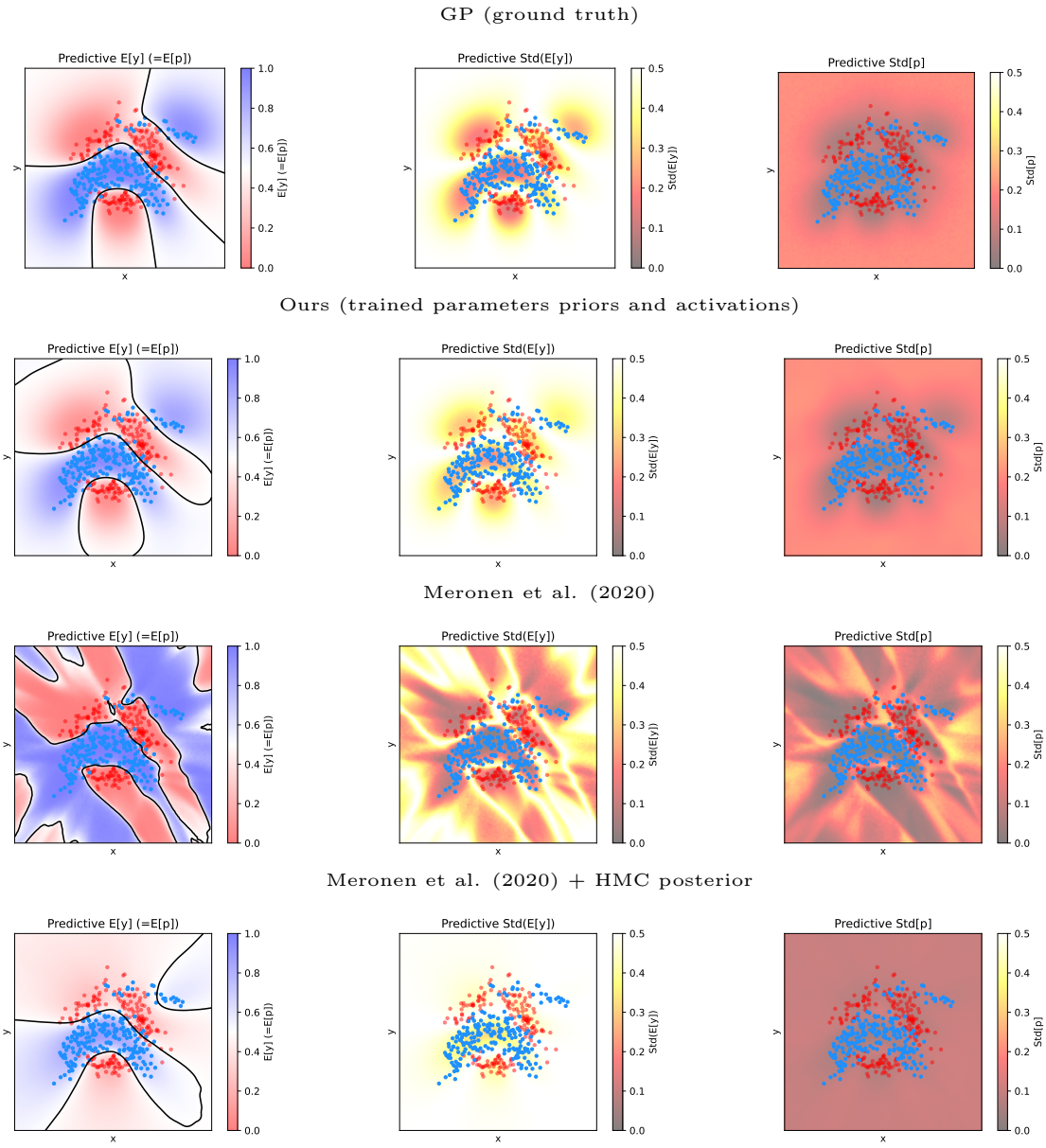


Figure 12: Posterior predictive distributions for a BNN with trained parameters priors and activations (ours; 2nd row), and for a BNN with analytically derived activation (3rd and 4th row). The first column illustrates class probabilities, the second column shows the total variance in class predictions, and the last column depicts the epistemic uncertainty component of the total uncertainty.

Table 8: Similarity of the posterior predictive distributions of BNNs with a single wide hidden layer to the posterior of a GP, considered as the ground truth. Calculations were performed for a 2D test grid X (100×100) spanning the area visible in Fig. 12, which includes regions both with and without training data to account for overconfidence far from the data. The posteriors were obtained using the HMC sampler in all cases, except for [Meronen et al., 2020], where the original code was used (however, results for this model with an HMC-derived posterior are also included below). Lower values indicate better performance. For our method, we present results using weights and priors pretrained for 1D inputs, as well as those trained on functional priors for 1D/2D inputs matched to the task on X.

Weights and activation	1-Wasserstein	2-Wasserstein	Linear-MMD	Mean-L1	Mean-L2	Mean-MSE	Median-L1	Median-L2	Median-MSE	Poly-MMD $\times 10^3$	RBF-MMD
Default with ReLU	39	39	667	0.23	0.26	0.067	0.3	0.33	0.11	4518	0.16
Default with TanH	38	38	595	0.22	0.24	0.058	0.3	0.32	0.11	4181	0.15
Normal with ReLU	31	32	603	0.21	0.25	0.061	0.22	0.26	0.069	3093	0.47
Normal with TanH	25	25	314	0.14	0.18	0.031	0.15	0.18	0.034	1721	0.55
[Meronen et al., 2020]	36	36	777	0.24	0.28	0.078	0.28	0.32	0.1	7311	0.24
Normal + HMC	21	21	72	0.07	0.09	0.007	0.071	0.094	0.009	435	0.24
Default + HMC	42	42	284	0.14	0.16	0.026	0.32	0.34	0.12	1867	0.11
Ours: pretrained 1D	25	25	73	0.06	0.09	0.008	0.076	0.1	0.011	771	0.07
Ours: trained 1D	23	23	6	0.03	0.03	0.001	0.029	0.035	0.001	45	0.03
Ours: trained 2D	23	23	13	0.03	0.03	0.001	0.028	0.035	0.001	74	0.02

D.5 PERIODIC AND CONDITIONAL ACTIVATIONS

BNNs with periodic activations were trained on functions sampled from a GP with a Matérn kernel with $\nu = \frac{5}{2}$ and lengthscales ℓ of values ranging from 0.01 to 10.0. The inputs were randomly sampled from the interval $[-3\ell, 3\ell]$. Each input batch comprised 8 sets X, each containing 512 values x. For each set, 128 functions were sampled. Adam optimizer, with a learning rate of 0.01, was employed over 4000 iterations.

For the conditioning of priors and activations, a MLP hypernetwork was utilized. This hypernetwork consisted of 3 hidden layers with respective widths of 128, 32, and 8, and RBF activations. On top of the hidden layers, separate dense heads were deployed for each output—a 4-dimensional head for producing variances and two 10-dimensional heads for generating $\{\psi\}$ and $\{A\}$. Each head projected the output of the last hidden layer to the requested dimensions.

E EXTENSIVE COMPARISON AGAINST Tran et al. [2022]

As is well known, a single-hidden-layer BNN with infinite width approximates a Gaussian Process (GP). We leverage this fundamental result in the context of optimal transport (minimizing the Wasserstein metric) to induce a GP with a desired kernel behavior onto a BNN.

A related approach, formulated as the dual problem of optimal transport, was explored in [Tran et al., 2022]. However, their method utilizes the complex Normalizing Flow priors as default, and as such, does not incorporate any mathematical property that guarantees the resulting model is a GP. From this perspective, the approach in [Tran et al., 2022] can be seen as a heuristic that yields favorable performance – measured by RMSE and NLL – within the training data range, but without any formal assurance that the model ultimately corresponds to a GP. Their method relies on a multiple-layer BNN with narrow layers, where the number of layers is chosen empirically based on the specific problem. Although employing a Gaussian prior may encourage GP-like behavior, this property does not necessarily hold when using Normalizing Flow priors.

In the following experiments, we empirically demonstrate that this approach generally fails to converge to a GP or satisfy the theoretical requirements due to the use of narrow layers and Normalizing Flow priors.

E.1 USAGE OF WIDER BNNS

First, we investigated the effect of increasing the width of a single-hidden-layer BNN from 50 neurons – following the typical choice in [Tran et al., 2022] – to 1024 neurons. Specifically, we compared two types of priors: Gaussian priors and Normalizing Flow priors. Additionally, we examined four different activation functions commonly used in BNNs and evaluated these settings on a 1D regression task.

Numerical results for RMSE, NLL, and the 1-Wasserstein ($\mathcal{W}_1^1$) and 2-Wasserstein ($\mathcal{W}_2^2$) metrics are presented in Tab.9 and Tab.10 for Gaussian and Normalizing Flow priors, respectively. We observe that increasing the layer width generally improves the Wasserstein metrics while maintaining similar or slightly worse RMSE and NLL when using Gaussian priors. This improvement is likely due to better adherence to theoretical assumptions (*see* Tab. 9). However, even with a wider layer and a fixed activation function, achieving true GP behavior remains elusive.

In contrast, using wider layers with Normalizing Flow priors significantly degrades performance across all considered metrics (*see* Tab.10). We hypothesize that this is due to the lack of theoretical foundations for such priors. Furthermore, we visualize the posterior distributions of narrow and wide BNN layers for two activation functions: ***ReLU***, which performs poorly even with a Gaussian prior, and *TanH* (*see* Fig.13). These results indicate that Normalizing Flow priors – despite being uniquely assigned to each neuron – fail to capture the increased capacity of wider networks, regardless of the activation function used. This serves as a clear example of where the heuristic proposed by [Tran et al., 2022] collapses.

Table 9: Comparison of Tran et al. [2022] using a narrow and a wide hidden layer in a BNN with a set Gaussian prior. We observe that the wider hidden layer usually helps in achieving better Wasserstein metrics and similar or slightly worse RMSE and NLL when using Gaussian priors.

$width \rightarrow$	50				1024			
$activation \downarrow metric \rightarrow$	*RMSE*	*NLL*	$\mathcal{W}_1^1$	$\mathcal{W}_2^2$	*RMSE*	*NLL*	$\mathcal{W}_1^1$	$\mathcal{W}_2^2$
TanH	0.36	0.42	7.57	7.83	0.53	0.86	8.04	8.26
RBF	0.25	0.24	7.27	7.61	0.24	0.25	6.96	7.16
ReLU	1.16	4.10	13.67	13.82	1.19	4.35	12.45	12.62
Swish	1.21	4.52	13.18	13.34	1.22	4.59	12.61	12.78

Table 10: Comparison of Tran et al. [2022] using a narrow and a wide hidden layer in a BNN with a set Normalizing Flow prior. We notice that using wider layers with Normalizing Flow priors significantly lowers the result on each of the considered metrics.

$width \rightarrow$	50				1024			
$activation \downarrow metric \rightarrow$	*RMSE*	*NLL*	$\mathcal{W}_1^1$	$\mathcal{W}_2^2$	*RMSE*	*NLL*	$\mathcal{W}_1^1$	$\mathcal{W}_2^2$
TanH	0.28	0.27	6.22	6.53	0.89	2.32	10.41	10.52
RBF	0.25	0.24	5.48	5.75	0.35	0.41	6.00	6.20
ReLU	0.89	2.34	9.53	9.67	1.10	3.67	12.29	12.44
Swish	1.11	3.64	13.67	13.82	1.18	4.19	13.71	13.86

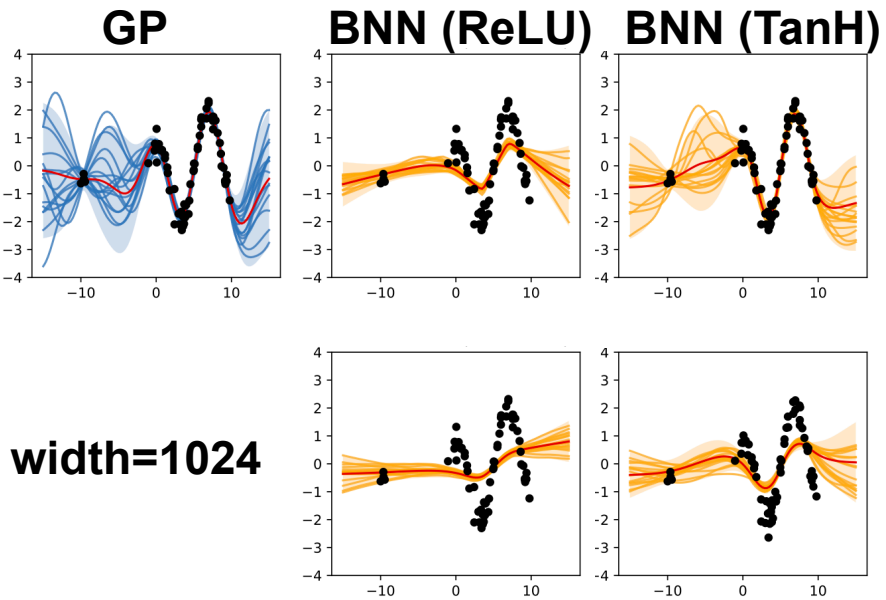


Figure 13: Comparison of Normalizing Flow prior with a one-layer BNN having a varying number of neurons and ReLU (**middle column**) or TanH (**3rd rcolumn**) activation function. We provide a BNN with a narrow hidden layer (50 neurons) as the baseline and compare it against a BNN with a wider hidden layer (1024 neurons). We observe that even wider hidden layer does not provide a better GP approximation due to the lack of theoretical guarantees when using non-Gaussian priors.

E.2 INFLUENCE OF NUMBER OF HIDDEN LAYERS IN BNNS

Next, we examine whether deeper BNNs within the [Tran et al., 2022] framework lead to improved results.

Specifically, we compare BNNs with multiple hidden layers (*i.e.,* $\{2, 3, 4, 5\}$) against single-hidden-layer baselines, evaluating four distinct activation functions – ReLU, TanH, Swish, and RBF – on RMSE and NLL, as shown in Fig. 14.

Additionally, in Fig. 15, we visualize the posterior distributions for two activation functions, ReLU and TanH, under the Normalizing Flow prior to provide further qualitative insights.

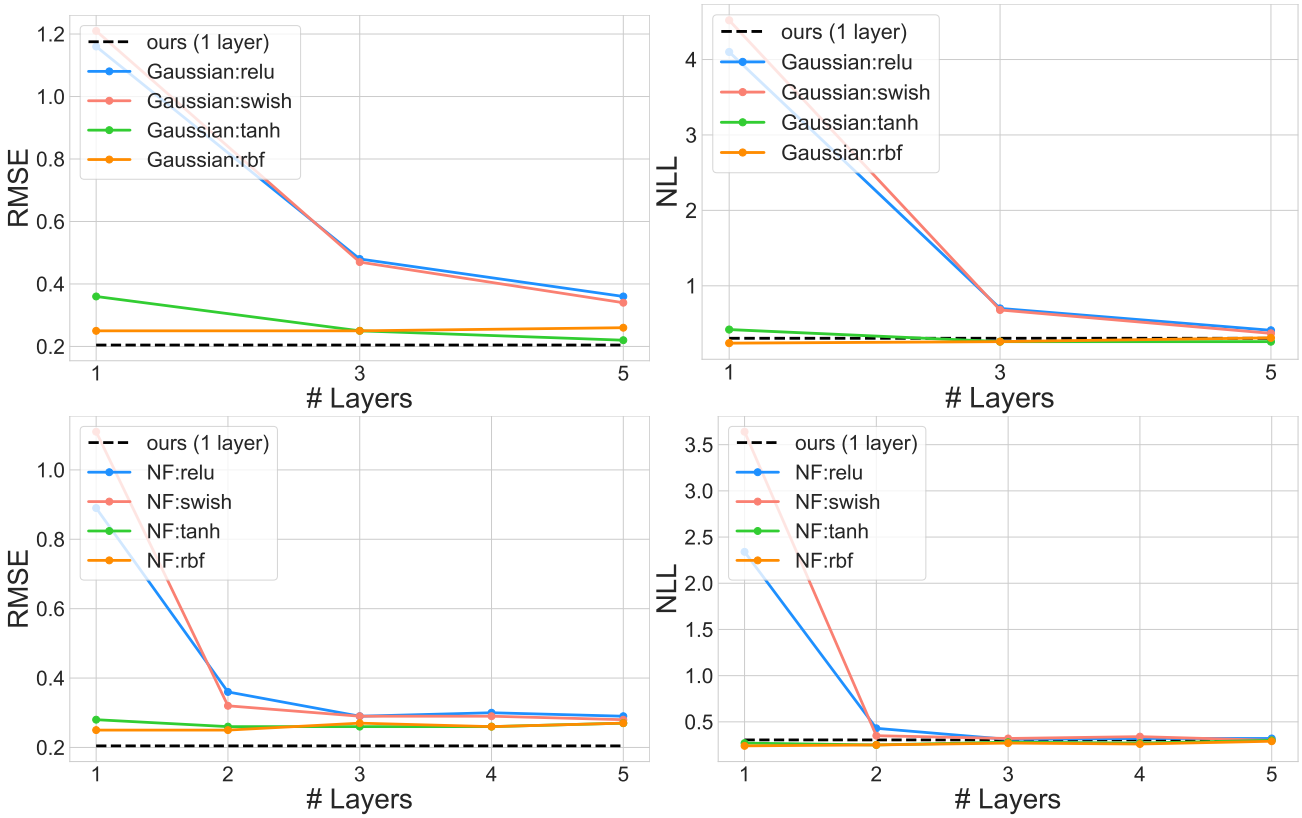


Figure 14: Comparison of different priors, activation functions and BNNs depth in Tran et al. [2022] method in terms of RMSE (**left column**) and NLL **right column** on test data. We observe that this method with Gaussian prior (**top row**) usually get better results with a larger number of hidden layers. Contrary, using Normalizing Flow prior get better results only for some (*worse*) activations. The largest increase of quality might be observed for the worst activation functions. We find that our approach (**dashed black line**) has better, or similar results in terms of both NLL and RMSE.

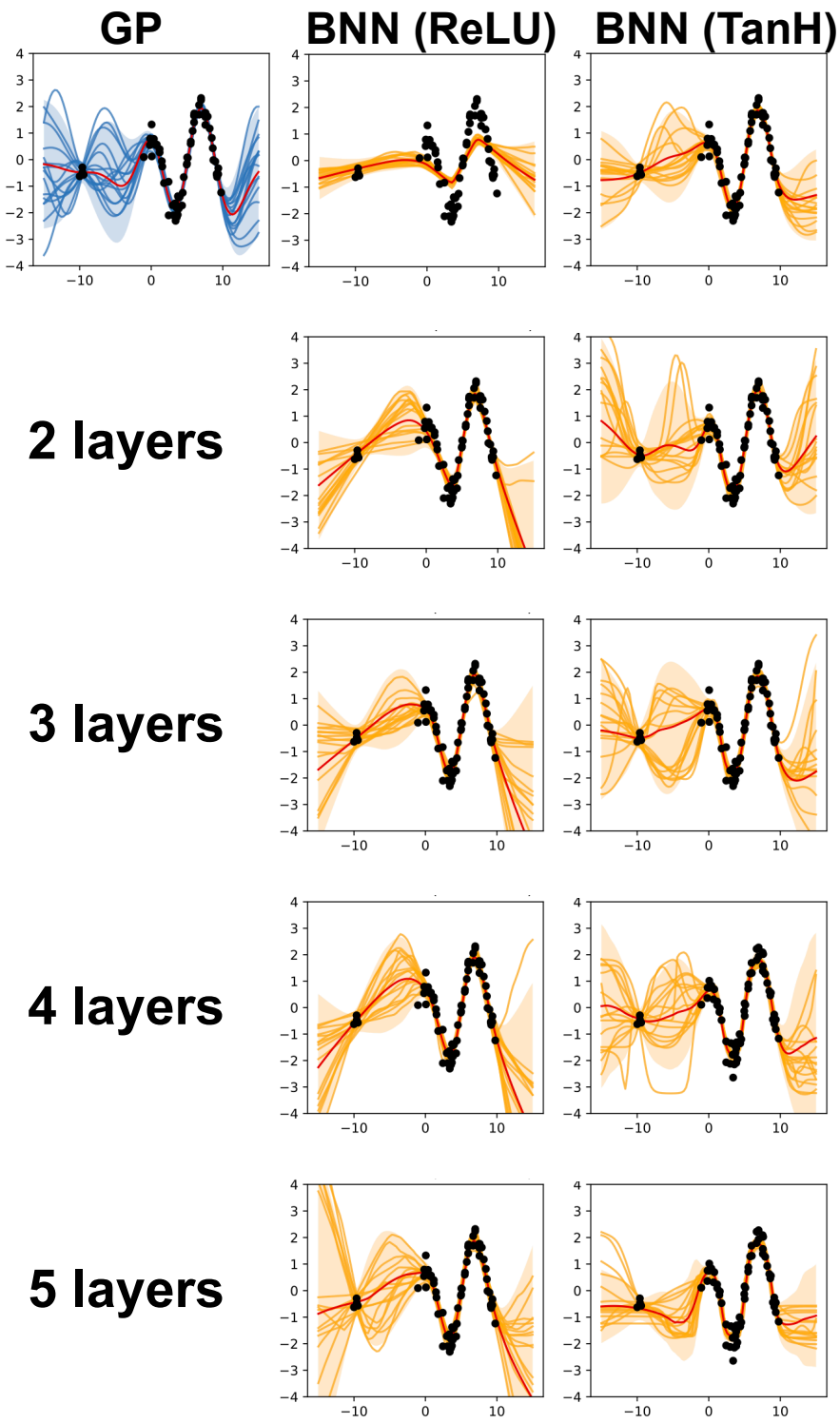


Figure 15: Comparison of Normalizing Flow prior with a BNN having a varying number of hidden layers and ReLU (**middle column**) or TanH (**3rd column**) activation function. We select a single hidden layer (50 neurons) BNN as a baseline (**top row**), which we compare against a multiple hidden layers BNNs (2, 3, 4, or 5 layers). We notice that using a larger number of hidden layers destroys the posteriors for a BNN with TanH activation function, but for some limit helps when using poor ReLU activation.

E.3 OUT OF DISTRIBUTION EVALUATION

Finally, we evaluate the [Tran et al., 2022] method with different priors and varying numbers of hidden layers on an out-of-distribution (OOD) task. Since the true data distribution in OOD regions is unknown, we rely on optimal transport metrics—1-Wasserstein ($\mathcal{W}_1^1$) and 2-Wasserstein ($\mathcal{W}_2^2$). Additionally, we compare all these configurations against our single-hidden-layer method.

To ensure a comprehensive comparison across various OOD scenarios, we define multiple distinct regions of x, where different methods may exhibit different behaviors. These regions are illustrated in Fig. 16.

The results, presented in Fig. 17, 18, 19, and 20, lead to several key observations regarding the performance on OOD tasks:

- **Gaussian prior:** A greater number of hidden layers generally improves performance when using appropriate activation functions (*TanH, RBF*) but degrades it for others (*ReLU, Swish*).
- **Normalizing Flow prior:** In many cases, the best results are achieved with single- or two-hidden-layer BNNs, while deeper architectures significantly degrade performance.
- **Sensitivity to depth:** The Normalizing Flow prior is highly sensitive to the number of hidden layers, providing clear evidence of the heuristic's instability.
- **Our method:** Due to its grounding in theoretical properties, our approach consistently outperforms all configurations of the [Tran et al., 2022] method.

These findings highlight the limitations of heuristic-based approaches and reinforce the importance of theoretical guarantees in achieving robust generalization.

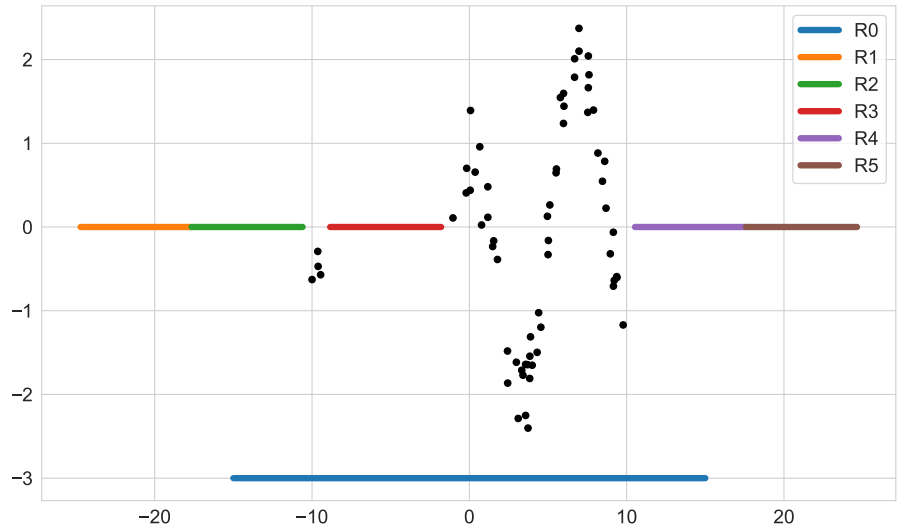


Figure 16: We compare our method with Tran et al. [2022] on the out of distribution data. For this aim, we provide a wide experimental setting comparing both methods on a few different ranges of data (x). We present these ranges as different colour lines in this plot. Firstly, we consider the original test range (**R0**), then we focus on the data between the given blobs of points (**R3**). Finally, we consider the out of distribution ranges – ones being closer to the train data (**R2**, **R4**, and **R2** $\bigcup$ **R4**) or further (**R1**, **R5**, and **R1** $\bigcup$ **R5**). The densities of evaluation points meets the density of the original data on each range despite **R0**.

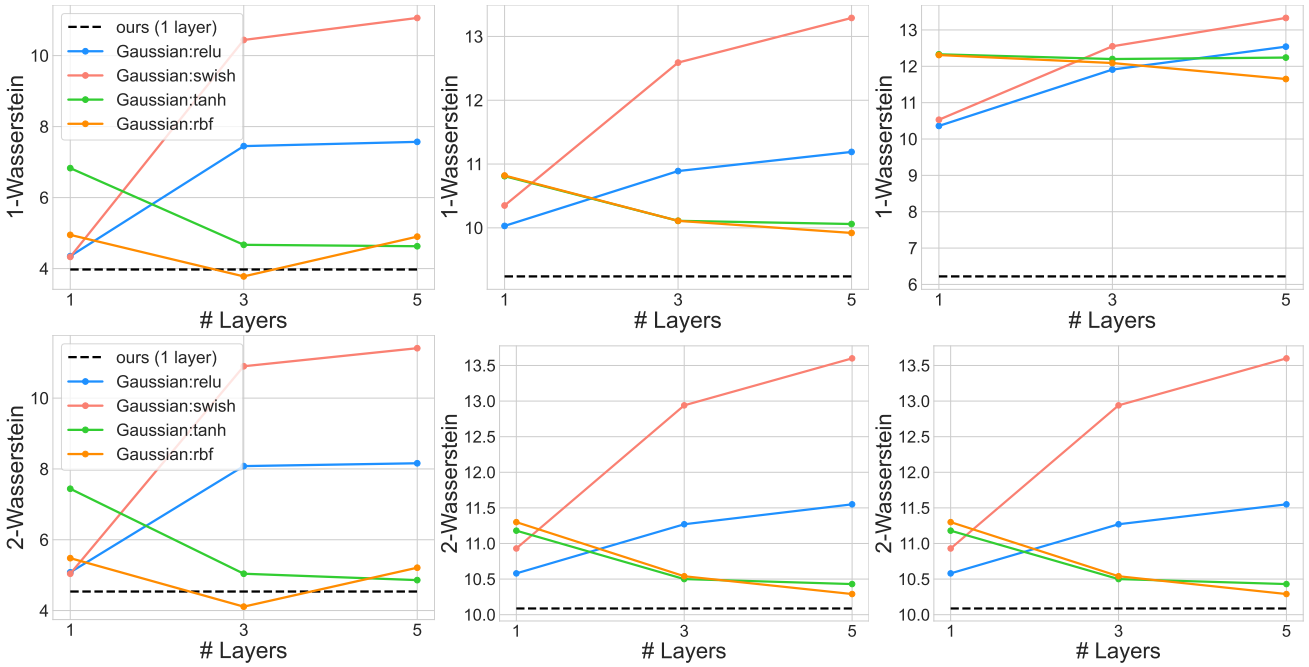


Figure 17: Comparison of Tran et al. [2022] method with a Gaussian prior and different number of hidden layers. In this plot, we present $\mathcal{W}_1^1$ (**top row**) and $\mathcal{W}_2^2$ (**bottom row**) for the regions **R3**, **R4** $\bigcup$ **R2**, and **R1** $\bigcup$ **R5** from left to right side. The appropriate regions might be found in Fig. 16.

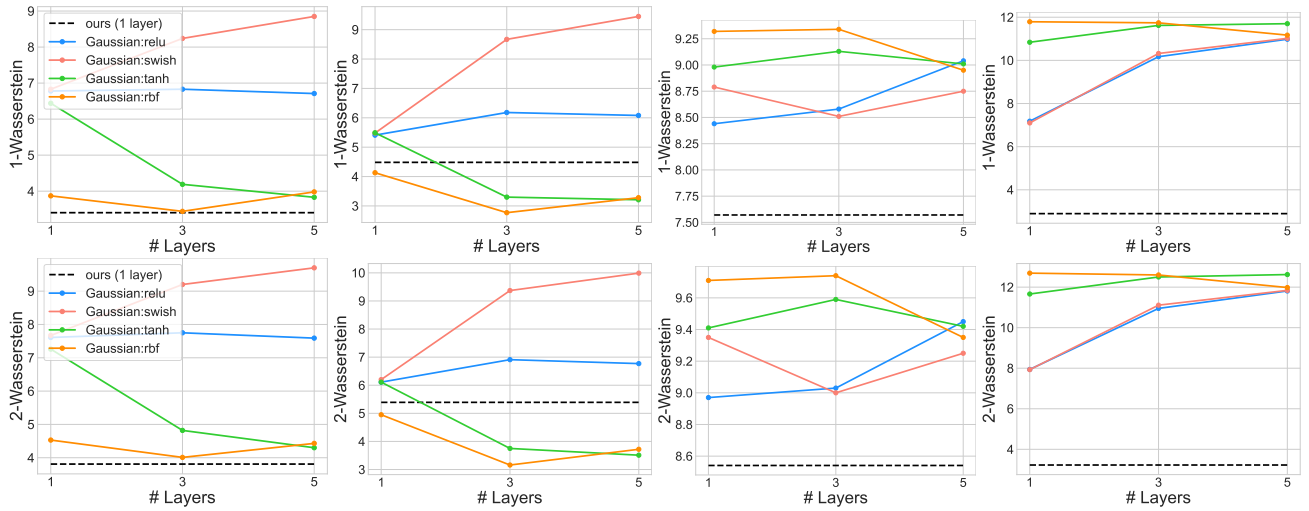


Figure 18: Comparison of Tran et al. [2022] method with a Gaussian prior and different number of hidden layers. In this plot, we present $\mathcal{W}_1^1$ (**top row**) and $\mathcal{W}_2^2$ (**bottom row**) for the regions **R1**, **R2**, **R4**, and **R5** from left to right side. The appropriate regions might be found in Fig. 16.

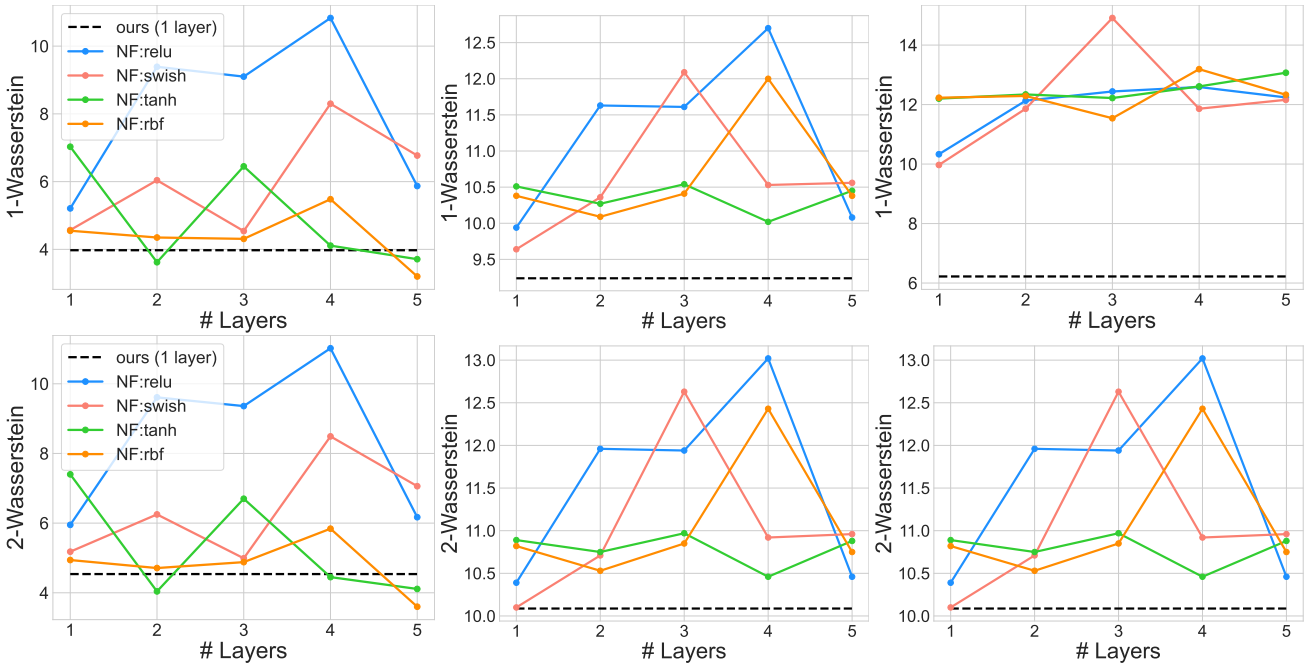


Figure 19: Comparison of Tran et al. [2022] method with a Normalizing Flow prior and different number of hidden layers. In this plot, we present $\mathcal{W}_1^1$ (**top row**) and $\mathcal{W}_2^2$ (**bottom row**) for the regions **R3**, **R4** $\bigcup$ **R2**, and **R1** $\bigcup$ **R5** from left to right side. The appropriate regions might be found in Fig. 16.

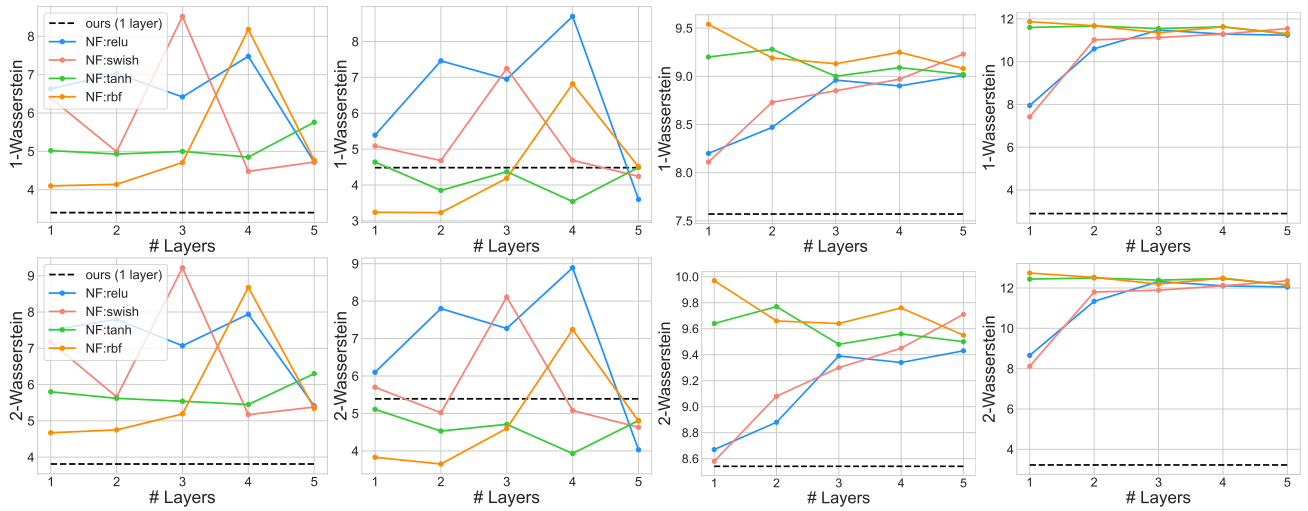


Figure 20: Comparison of Tran et al. [2022] method with a Normalizing Flow prior and different number of hidden layers. In this plot, we present $\mathcal{W}_1^1$ (**top row**) and $\mathcal{W}_2^2$ (**bottom row**) for the regions **R1**, **R2**, **R4**, and **R5** from left to right side. The appropriate regions might be found in Fig. 16.