An Empirical Analysis of the Advantages of Finite v.s. Infinite Width Bayesian Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Understanding the relative advantages of finite versus infinite-width neural networks (NNs) is important for model selection. However, comparing NNs with different widths is challenging because, as the width increases, multiple model properties change simultaneously – model capacity increases while the model becomes less flexible in learning features from the data. Analyses of Bayesian neural networks (BNNs) is even more difficult because inference in the finite width case is intractable. In this work, we empirically compare finite and infinite width BNNs, and provide quantitative and qualitative explanations for their performance difference. We find that when the limiting model is mis-specified, increasing the width can reduce the generalization performance of BNNs. In these cases, we provide evidence that finite BNNs generalize better partially due to the properties of their frequency spectrum that allows them to adapt under model mismatch.

1 Introduction

Recent works in Bayesian deep learning note a counter-intuitive phenomenon – that larger architecture (specifically larger width) can hurt model performance. For example, Lee et al. observe that on image classification tasks, as the width of the Bayesian CNN increases, classification accuracy decreases; Pleiss and Cunningham make similar observations on regression tasks. However, posterior inference in finite width BNNs is challenging [Coker et al., 2022, Foong et al., 2020], whereas, in the infinite-width limit, BNNs converge to a well-behaved model, the Neural Network Gaussian Processes (NNGPs) (e.g. [Lee et al., 2017, Matthews et al., 2018, Neal, 2012]). Furthermore, infinite width BNNs easily admit theoretical analysis while finite width BNNs are difficult to analyze due to the non-Gaussian [Noci et al., 2021] processes they define in function space. Thus, to choose between finite versus infinite width BNNs, we need to know: when and why does solving the harder inference problem present a better trade-off for improved performance? In this paper, we provide empirical explanations for the counter-intuitive phenomenon that larger width hurts BNN performance, by comparing the inductive biases and frequency spectra of finite and infinite width BNNs.

Contributions We study one hidden layer BNNs with erf and ReLU activations, whose infinite-width counterparts we call limiting NNGPs. Our contribution is as follows:

1. We show that when there is model mismatch between the limiting NNGP and the data generating model, finite width BNNs can generalize better than infinite width BNNs;
2. To investigate the above phenomenon, we quantitatively show that finite width BNNs have different inductive biases from NNGPs (finite width BNNs can generate more diverse datasets than NNGPs);
3. We also qualitatively compare the inductive biases of finite and infinite width BNNs by comparing their frequency spectrum from spectral analysis. We demonstrate that, compared

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

to infinite width counterparts, prior predictive distributions of finite width BNNs define heavier-tailed distributions over the frequency spectrum (they place more mass on large coefficients). Additionally, after inference, the posterior predictive distributions of finite width BNNs define frequency spectra that are more similar to those defined by the data generating model;

4. We hypothesize that because finite width BNNs define frequency spectra with heavy-tailed coefficient distributions, they are more flexible in modeling the data using different frequencies. To test this, we design an inference procedure to filter out high-frequency components from the functional prior of BNNs. We see that generalization performance of BNNs decreases as high-frequency functions are removed, supporting our hypothesis.

In this work, we consider both erf and ReLU nonlinearity, since in these cases we obtain closed-form kernels for the limiting NNGPs, which allows for exact inference and theoretical analyses. Additionally, both activation functions are well studied in the literature (e.g. [Williams and Rasmussen, 2006, Cho and Saul, 2011]) and are suitable for modeling various types of functions [Lee et al., 2020].

2 Related Works

A large body of work exists regarding when and why finite (non-Bayesian) neural networks succeed. For example, Ghorbani et al. and Refinetti et al. empirically find that finite neural networks outperform kernel methods when the data contains low-dimensional structures. Aitchison [2020] show finite deep linear networks yield stochastic kernels, which are more flexible in learning representations from the data. However, it is not known whether these insights apply to BNNs.

In the context of BNNs, Pleiss and Cunningham study the performance degeneration of infinite width BNNs by studying deep GPs, which the authors consider to be a more generalized BNN model. The authors prove that as the width of a deep GP increases, the posterior collapses from a mixture function of data-dependent bases to that of data-independent bases. Thus, infinitely wide deep GPs are less capable of adapting to the data. We study BNNs from a spectral perspective and find that finite BNNs are more flexible in using functions of different frequencies.

Analogously to standard NNs [Rahaman et al., 2019], works have observed that—unlike their infinite counterparts—finite BNNs tend to be heavy-tailed in function space (e.g. [Noci et al., 2021, Pleiss and Cunningham, 2021]). We examine how and why this heavy-tailedness affects the prediction tasks under model mismatch.

Finally, a recent line of works provide theoretical and empirical analyses of how the power spectrum of standard, wide NNs changes during training (e.g. [Rahaman et al., 2019, Cao et al., 2019, Yang and Salman, 2019]). However, they do not consider finite-width, Bayesian settings nor the relationship between the power spectrum and performance.

3 Background

We consider a regression task with dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^{d_{\text{in}}}$, $y_i \in \mathbb{R}$. For a NN with one hidden-layer of width H, the output is given by, $f_{\text{NN}}(\mathbf{x}_i) = \frac{1}{\sqrt{H}}\mathbf{w}_1\phi(\mathbf{w}_0\mathbf{x}_i + \mathbf{b}_0) + b_1$, with $\mathbf{w}_0 \in \mathbb{R}^{H\times d_{\text{in}}}, \mathbf{b}_0 \in \mathbb{R}^{H\times 1},\ \mathbf{w}_1 \in \mathbb{R}^{1\times H}, b_1 \in \mathbb{R}$, and ϕ is a nonlinear activation function.

We denote neural network weights as $\mathbf{W}$ and biases as $\mathbf{b}$. For a Bayesian Neural Network, we specify prior distributions $p(\mathbf{W}) \sim \mathcal{N}(0, \sigma_{\mathbf{W}}^2\mathbf{I}),\ p(\mathbf{b}) \sim \mathcal{N}(0, \sigma_{\mathbf{b}}^2\mathbf{I})$. The posterior distribution of the weights is given by: $p(\mathbf{W}, \mathbf{b}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{W}, \mathbf{b})p(\mathbf{W})p(\mathbf{b})$, where $p(\mathcal{D}|\mathbf{W}, \mathbf{b}) = \prod_i p(y_i|\mathbf{x}_i, \mathbf{W}, \mathbf{b}),\ p(y_i|\mathbf{x}_i, \mathbf{W}, \mathbf{b}) = \mathcal{N}(y_i|f_{\text{NN}}(\mathbf{x}_i), \sigma_\epsilon^2)$ and σ_ϵ^2 is the variance of the output noise.

Given a set of new inputs $\mathcal{D}^* = \{\mathbf{x}_i^*\}_{i=1}^m$, the posterior predictive distribution of is given by

$$p(f^*|\mathbf{x}^*, \mathcal{D}) = \int_{\mathbf{W},\mathbf{b}} p(f^*|\mathbf{x}^*, \mathbf{W}, \mathbf{b}, \mathcal{D})p(\mathbf{W}, \mathbf{b}|\mathcal{D})d\mathbf{W}d\mathbf{b} \text{ where } f^* = f_{\text{NN}}(\mathbf{x}^*),$$

We use NUTS [Hoffman and Gelman, 2014] for sampling. Using erf, ReLU activations, a BNN will converge to a GP with the Arcsin, Arccos kernel, respectively (closed forms in Appendix A). In practice, the Arccos kernel has eigenvalues that decay slower than the Arcsin kernel and thus are more suitable for learning rough functions [Lee et al., 2020].

4 Generalization Performance of Finite and Infinite Width BNNs

We compare the regression performance of finite and infinite width BNNs under model mismatch – we train BNNs whose limiting NNGPs are different from the data generating model. Thus, the performance of the limiting NNGP is not optimal, leaving room for improvement. We want to see if, under this model mismatch, finite width BNNs perform differently from infinite width BNNs.

Datasets We generate data using GPs with three kernels, RBF, Arcsin and Arccos (GP-Arccos generates rougher functions that require larger a model capacity to learn). For GP-RBF, the length-scale l is chosen from $\{0.5, 1.0, 1.5, 2.0\}$. For GP-Arcsin/Arccos, the hyperparameters $\sigma_{\mathbf{W}}^2, \sigma_{\mathbf{b}}^2$ in Equation 6, 7, are chosen from $\{0.5, 1.0, 1.5, 2.0\}$. For each kernel setting, we draw 200 functions, each as a univariate datase (Details in Appendix B). We then fit each dataset using BNNs with erf and ReLU nonlinearity whose prior variances $\sigma_{\mathbf{W}}^2, \sigma_{\mathbf{b}}^2$ are chosen from $\{0.5, 1.0, 1.5, 2.0\}$ and hidden widths H vary from 2 to 200. For data drawn from GP-RBF, there is always model-mismatch between the limiting NNGP and the data generating model. For data drawn from GP-Arcsin/GP-Arccos, when the hyperparameters are equivalent to the prior variances of BNNs with erf/ ReLU nonlinearity, there is no model mismatch Appendix C).

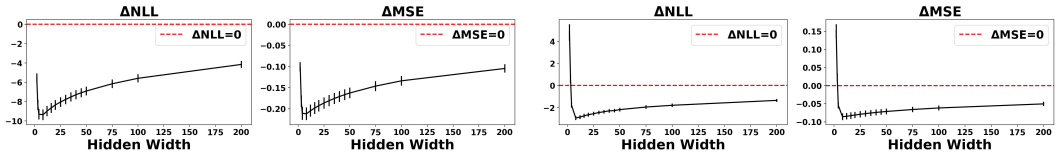
Evaluation Metrics To quantify the performance of finite width BNNs, we calculate the expected difference of negative loglikelihood (NLL) and test MSE. Specifically, for each kernel setting with dataset $\{\mathcal{D}^{(s)}\}_{s=1}^{S}$, we calculate

$$\Delta\text{NLL}_{\text{BNN}_H} = \frac{1}{S}\sum_{s=1}^{S}[\text{NLL}_{\text{BNN}_H}(\mathcal{D}^{(s)}) - \text{NLL}_{\text{NNGP}}(\mathcal{D}^{(s)})] \tag{1}$$

$$\Delta\text{MSE}_{\text{BNN}_H} = \frac{1}{S}\sum_{s=1}^{S}[\text{MSE}_{\text{BNN}_H}(\mathcal{D}^{(s)}) - \text{MSE}_{\text{NNGP}}(\mathcal{D}^{(s)})] \tag{2}$$

When there is no model mismatch, both terms will converge to 0 as hidden width H goes to infinity.

Results **When there is a model mismatch between the data generating model and the limiting NNGP, finite width BNNs can outperform NNGPs.** Previous works have observed similar phenomena [Pleiss and Cunningham, 2021] on the real datasets which inevitably have model mismatch. In Figure 1a, 1b, we see that finite width BNNs have better generalization performance (the black curves are below the red dashed line) and converge to the limiting NNGP when the hidden width increases (the black curve approaches the red dashed line). We see similar trends on large real datasets (Figure 8). Additionally, a larger model mismatch results in a larger difference in the generalization performance (Figure 6, 7).



(a) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$ (b) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$

Figure 1: Plots of the test performance of finite width BNNs: The data is generated from GP-RBF with lengthscale 0.5. We fit each dataset with BNNs with erf and ReLU nonlinearity. x-axis denotes the hidden-layer width of the BNN; y-axis denotes ΔNLL, ΔMSE (Equation 1,2). Being above/below the red dashed line means that finite width BNNs perform worse/better than the limiting NNGPs. We see that finite width BNNs can outperform the limiting NNGPs as the black curves are mostly below the red dashed line.

5 Inductive Biases of Finite and Infinite BNNs

Why do finite width BNNs outperform infinite width BNNs even though they have a smaller model capacity? To answer this question, we first empirically show that finite width BNNs have different

inductive biases from infinite width BNNs (finite and infinite width BNNs are two distinct model classes); thus, one may generalize better than the other given different situations.

5.1 Quantifying Differences in Inductive Bias Using the Data Likelihood

There are different ways to quantify inductive biases, one of which is to evaluate how much the model assumptions overlap with the true data generating process. In the Bayesian literature, log marginal likelihood (LML) is often used to quantify this overlap. The LML of generating dataset $\mathcal{D}$ under model $\mathcal{M}$ can be computed by marginalizing out the prior over the function values f,

$$\log p(\mathcal{D}|\mathcal{M}) = \log \int p(\mathcal{D}|\mathcal{M}, f)p(f|\mathcal{M})df. \tag{3}$$

In our setup, the dataset $\mathcal{D}$ is generated from a GP and f is modeled by either a BNN or a NNGP.

Equation 3 quantifies the inductive bias by answering the question that "*How likely is that the dataset can be generated under the model*?" The literature argues that a model has a good inductive bias when it can achieve high LML for various datasets (a model is too complex when achieving modest LML for many datasets and too simple when achieving high LML for limited datasets).

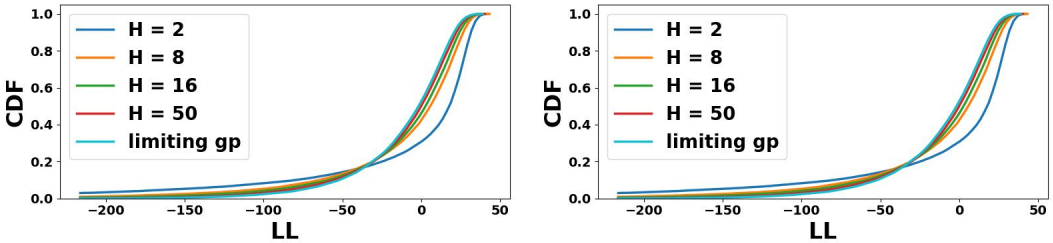
Despite many good properties of LML, Equation 3 is hard to compute (the Monte Carlo estimator exhibits large bias and variance given the large parameter space of BNNs).

In this work, we use an alternative way to quantify the overlap between the model assumptions and the data generating process: we still compute Equation 3 but now the datasets $\mathcal{D}$ are drawn from finite width BNNs and the limiting NNGPs while the model $\mathcal{M}$ is the data generating GP. A model has a desirable inductive bias when it can generate diverse types of datasets with high LML under the data generative model.

When the model $\mathcal{M}$ is a GP with prior GP$(m(\mathbf{x}), K)$, Equation 3 has an analytical form:

$$\log p(\mathcal{D}|\mathcal{M}) = -\frac{1}{2}(\mathbf{y}-m(\mathbf{x}))^{\intercal}\Sigma_f^{-1}(\mathbf{y}-m(\mathbf{x})) - \frac{1}{2}\log|\Sigma_f| - \frac{n}{2}\log 2\pi \text{ where} \Sigma_f = K + \sigma_\epsilon^2 I \tag{4}$$

Results **Finite and inifinite width BNNs have distinct inductive biases**. In Figure 2a, 2b, we see that BNNs with smaller hidden widths generate more diverse datasets than the limiting NNGP (the CDF of LML over datasets drawn from the data generative GP has more mass on the tail).



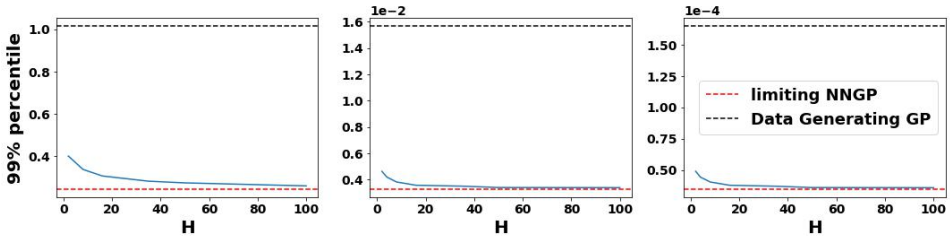
(a) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

(b) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

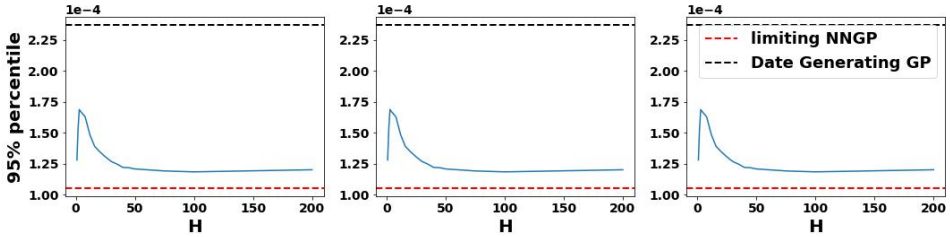
Figure 2: Cdf of LML (Eqn 4) with datasets $\mathcal{D}$ sampled from BNNs with different hidden widths evaluated under GP-RBF with lengthscale 0.5: x-axis denotes the values of LML while y-axis denotes cdf. Each color represents either a BNN with a specific hidden width or the limiting NNGP. BNNs with smaller hidden widths generate more datasets with higher LML and lower LML than the limiting GP at the same time.

5.2 Qualitatively Comparing Inductive Biases in Function Space

While the LML allows us to quantitatively compare the inductive biases of two Bayesian model classes, we also want to qualitatively describe the differences between them. For the latter, we want to compare the differences between the prior predictive distributions of finite and infinite width BNNs in the function space.



(a) Prior predictives of BNNs with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$



(b) Posterior predictives of BNNs with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$

Figure 3: Qualitative analyses of DCT coefficients: Each plot here represents a different frequency. x-axis denotes the hidden width while y-axis denotes the 99% percentile for the DCT coefficient distributions. The red, black dashed lines represent the limiting and the data generating GP, respectively. A smaller percentile implies more mass on the large coefficients.

Previous works prove that finite BNNs yield heavy-tailed priors in the function space [Noci et al., 2021, Zavatone-Veth and Pehlevan, 2021]. However, they do not discuss how the heavy-tailed priors affect the generalization performance.

One common approach to quantify heavy-tailedness is to compute the higher-order moments of the distribution; however, this is not easy because the function space is large and requires a prohibitively large number of samples. We instead analyze the samples drawn from the prior and posterior predictive distributions using Discrete Cosine Transform (DCT). DCT is a linear transformation that expresses a function as a weighted sum of cosines of different frequencies.

Formally, given a function $\mathbf{f} = [f_0, \ldots, f_{N-1}]^\intercal$, DCT is a linear and invertible function, $\mathbf{T}_{\text{DCT}} : \mathbb{R}^N \to \mathbb{R}^N$, that transforms $\mathbf{f}$ into a set of DCT coefficients, denoted as $\mathbf{a} = [a_0, \ldots, a_{N-1}]^\intercal$:

$$\mathbf{a} = \mathbf{T}_{\text{DCT}}\mathbf{f}, \quad \text{where} \quad \mathbf{T}_{\text{DCT}_{ij}} = \frac{\sqrt{2}}{\sqrt{2^{\mathbb{I}[i=0]}N}} \cos\left(\frac{\pi i(2j+1)}{2N}\right). \tag{5}$$

We qualitatively compare the DCT coefficient distributions of finite and infinite width BNNs.

Results Figure 3a shows that comparing to the NNGP, the function priors of **finite width BNNs place more mass on larger coefficients** (the 99% percentile is smaller). Figure 3b shows similar trends after inference. Additionally, **the frequency spectrum defined by finite width BNNs behaves more similar to that of the data generating model** which also puts more mass on the larger coefficients (the black dashed line is higher).

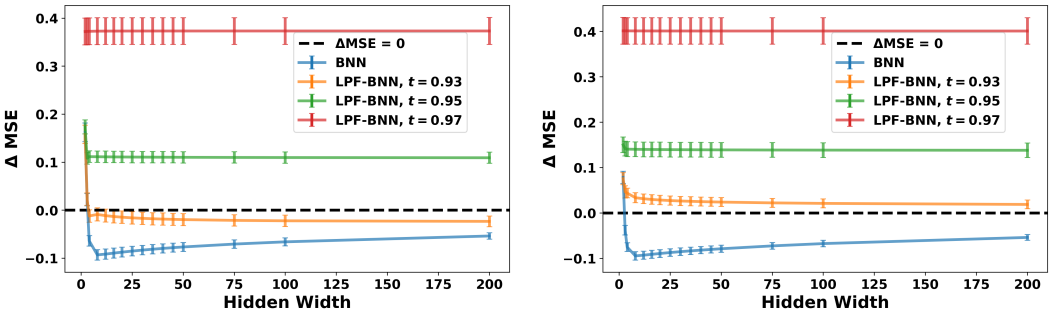
5.3 Connecting Inductive Bias to Performance Difference Through Frequency Filtering

In the previous section, we have shown that the frequency spectrum defined by prior predictive distributions of finite BNN places more mass on large coefficients than that of the limiting NNGP, which makes them more flexible in using functions of different frequencies. In this section, we go a step further: we demonstrate that removing high-frequency components from the prior predictive distributions of finite width BNNs hurt the model performance as now finite width BNNs are constrained to the low-frequency regime.

To do this, we compare the generalization performance of a finite-width BNN with that of a "lowpass-filtered" BNN—a BNN in which we filter out high-frequency components from the prior predictive distributions (i.e. we "smooth" the functions drawn from the function prior).

At a high level, given a function $\mathbf{f}$ drawn from a BNN prior, we first compute its DCT coefficients via Equation 5, and reconstruct that function from the coefficient via $\mathbf{f} = \mathbf{T}_{\text{DCT}}^{\top}\mathbf{a}$. We then remove high-frequency components from the function by setting the corresponding DCT coefficients to 0: $\tilde{\mathbf{f}} = \mathbf{T}_{\text{DCT}}^{\top}\mathbf{I}_N^{(k)}\mathbf{T}_{\text{DCT}}\mathbf{f} := \mathbf{C}\mathbf{f}$, where $k = tN$, $t \in [0, 1]$, $\mathbf{I}_N^{(k)}$ is an $N \times N$ identity matrix with the last k diagonal elements set to 0 (algorithm details in Appendix D). A larger t suggests more high-frequency components being removed with $t = 0$ recovering the original BNN.

Results **Removing high-frequency components hurt the generalization performance of finite BNNs.** In Figure 4, we see that as t increases (i.e. removing more high-frequency components), the generalization performance decreases (ΔMSE increases). Given different BNN architecutre, removing the same proportion of the high frequency components has different effect (when $t = 0.93$, lowpass-filtered BNNs with ReLU outperform the limiting NNGP while low-pass filtered BNNs with erf fail.) (Additional results in Appendix E.3



(a) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$

(b) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$

Figure 4: Plots of the test performance of low-pass filtered BNNs: The data is generated from GP-RBF with lengthscale 0.5. x-axis denotes the hidden-layer width of the BNN while y-axis denotes ΔMSE. Higher thresholds t implies more high frequencies being removed. When t increases, the generalizationg performance decreases.

6 Discussion

In this paper, we quantitatively and qualitatively compare the behaviors of finite and infinite width BNNs. We empirically show that finite width BNNs define a frequency spectrum with more mass on large coefficients, which allows them to adapt to the data faster under model mismatch. Our experiments suggest that there are advantages of using non-Gaussian models such as finite width BNNs or non-Gaussian approximation of the true posterior even though the inference is challenging.

References

Laurence Aitchison. Why bigger is not always better: on finite and infinite neural networks. In *International Conference on Machine Learning*, pages 156–164. PMLR, 2020.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. *arXiv preprint arXiv:1912.01198*, 2019.

Youngmin Cho and Lawrence K Saul. Analysis and extension of arc-cosine kernels for large margin classification. *arXiv preprint arXiv:1112.3712*, 2011.

Beau Coker, Wessel P Bruinsma, David R Burt, Weiwei Pan, and Finale Doshi-Velez. Wide mean-field bayesian neural networks ignore the data. In *International Conference on Artificial Intelligence and Statistics*, pages 5276–5333. PMLR, 2022.

Andrew Foong, David Burt, Yingzhen Li, and Richard Turner. On the expressiveness of approximate inference in bayesian neural networks. *Advances in Neural Information Processing Systems*, 33: 15897–15908, 2020.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *Advances in Neural Information Processing Systems*, 33: 14820–14830, 2020.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014.

Cheolhei Lee, Jianguo Wu, Wenjia Wang, and Xiaowei Yue. Neural network gaussian process considering input uncertainty for composite structures assembly. *IEEE/ASME Transactions on Mechatronics*, 2020.

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.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Lorenzo Noci, Gregor Bachmann, Kevin Roth, Sebastian Nowozin, and Thomas Hofmann. Precise characterization of the prior predictive distribution of deep relu networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*, 2019.

Geoff Pleiss and John P Cunningham. The limitations of large width in neural networks: A deep gaussian process perspective. *Advances in Neural Information Processing Systems*, 34, 2021.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

Maria Refinetti, Sebastian Goldt, Florent Krzakala, and Lenka Zdeborová. Classifying high-dimensional gaussian mixtures: Where kernel methods fail and neural networks succeed. In *International Conference on Machine Learning*, pages 8936–8947. PMLR, 2021.

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

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. *arXiv preprint arXiv:1907.10599*, 2019.

Jacob Zavatone-Veth and Cengiz Pehlevan. Exact marginal prior distributions of finite bayesian neural networks. *Advances in Neural Information Processing Systems*, 34:3364–3375, 2021.

A Analytical Form for NNGP Kernels

Define the following terms,

$$\tilde{\mathbf{x}}^\intercal = [x,\ 1],\ \Sigma = \mathrm{Diag}(\sigma_{\mathbf{W}}^2, \sigma_{\mathbf{b}}^2),\ \|\tilde{\mathbf{x}}\|_\Sigma = \sqrt{\tilde{\mathbf{x}}^\intercal \Sigma \tilde{\mathbf{x}}}.$$

For a one hidden-layer BNN with erf nonlinearity, the BNN will converge to a GP with Arcsin kernel defined as,

$$K_{\arcsin}(x, x') = \frac{2\sigma_{\mathbf{W}}^2}{\pi}\theta + \sigma_{\mathbf{b}}^2 \quad \text{where} \quad \theta = \sin^{-1}\Big(\frac{2\tilde{\mathbf{x}}^\intercal \Sigma \tilde{\mathbf{x}}'}{\sqrt{1 + 2\|\tilde{\mathbf{x}}\|_\Sigma^2}\sqrt{1 + \|\tilde{\mathbf{x}}'\|_\Sigma^2}}\Big) \tag{6}$$

For a one hidden-layer BNN with ReLU nonlinearity, the BNN will converge to a GP with Arccos kernel defined as,

$$K_{\arccos}(x, x') = \frac{\sigma_{\mathbf{W}}^2}{2\pi}\|\tilde{\mathbf{x}}\|_\Sigma \|\tilde{\mathbf{x}}'\|_\Sigma (\sin\theta + (\pi - \theta)\cos\theta) + \sigma_{\mathbf{b}}^2 \quad \text{where} \quad \theta = \cos^{-1}\Big(\frac{\tilde{\mathbf{x}}^\intercal \Sigma \tilde{\mathbf{x}}'}{\|\tilde{\mathbf{x}}\|_\Sigma \|\tilde{\mathbf{x}}'\|_\Sigma}\Big) \tag{7}$$

B Datasets

Synthetic Datasets We generate data using GPs with three kernels, RBF, Arcsin and Arccos (GP-Arccos generates rougher functions that require larger model capacity to learn). For GP-RBF, the lengthscale l is chosen from $\{0.5, 1.0, 1.5, 2.0\}$. For GP-Arcsin/Arccos, the hyperparameters $\sigma^2_{\mathbf{W}}, \sigma^2_{\mathbf{b}}$ in Equation 6, 7, are chosen from $\{0.5, 1.0, 1.5, 2.0\}$. For each kernel setting, we draw 200 functions, each as a univariate dataset. For each dataset, we uniformly sample 20 training points from $[-3, -1] \bigcup [1, 3]$ aggregate with $x = 0$ to facilitate learning ($N_{\text{train}} = 21$) and $N_{\text{test}} = 100$ test point from $[-3, 3]$. We add Gaussian noise with $\sigma_\epsilon = 0.1$ to the training data.

Real Datasets We test 3 real datasets: UCI boston, UCI Energy (the output dimension are fit individually), Sic97 [Shah et al., 2014]. We split the data into train, test, valid with the ratio $80 - 10 - 10$. We perform cross validation on NNGPs and choose the GP hyper-parameter with the highest validation MSE.

C Models and Inference

We fit each dataset using BNNs with erf and ReLU nonlinearity whose prior variances $\sigma^2_{\mathbf{W}}, \sigma^2_{\mathbf{b}}$ are chosen from $\{0.5, 1.0, 1.5, 2.0\}$ and hidden widths H vary from 2 to 200. We use NUTS implemented in NumPyro [Phan et al., 2019] to obtain posterior samples from $p(\mathbf{W}, \mathbf{b}|\mathcal{D})$.

We used 4 parallel chains, each with $10,000$ samples during burn-in, $40,000$ MCMC samples and a thinning of 5.

D Lowpass-filtered BNNs

We modify the original BNN model,

$$\begin{aligned} \mathbf{W}, \mathbf{b} &\sim p(\mathbf{W}, \mathbf{b}) \\ y|\mathbf{x}, \mathbf{W}, \mathbf{b} &\sim \mathcal{N}(y|f_{\text{NN}}(\mathbf{x}; \mathbf{W}, \mathbf{b}), \sigma^2_\epsilon), \end{aligned} \tag{8}$$

so that the f_{NN} is filtered to include only its low-frequency components. We denote this lowpass filtered function by $\tilde{f}_{\text{NN}}$, and describe a general algorithm for computing $\tilde{f}_{\text{NN}}$ and incorporating it into the model, allowing it to be paired with *any* inference algorithm (e.g. NUTS, which requires $\tilde{f}_{\text{NN}}$ to be differentiable). For simplicity, we assume the inputs are 1-dimensional ($d_{\text{in}} = 1$).

Step 1: From weight-space to function space. To filter out high-frequency components of a *function* f_{NN}, we cannot simply operate on the corresponding neural network *weights* $\mathbf{W}, \mathbf{b}$. Instead, we decompose the function (evaluated on a grid of points) into a weighted sum of cosines (using the DCT) and then remove the highest frequency terms in the sum. Specifically, we define a grid of N^{grid} uniformly spaced inputs $\mathbf{x}^{\text{grid}}_n \in \mathbb{R}^{d_{\text{in}}}$, denoted jointly by $\mathbf{X}^{\text{grid}} \in \mathbb{R}^{N^{\text{grid}} \times d_{\text{in}}}$. We perform the DCT on $f_{\text{NN}}(\cdot; \mathbf{W}, \mathbf{b})$ evaluated on this grid. We further ensure that our training and test sets lie on this grid by mapping them to their nearest neighbors on the grid (a fine grid will only negligibly modify the training and test sets).

Step 2: Performing the DCT. We compute the type-II orthonormal DCT, $\mathbf{a}_k(\mathbf{W}, \mathbf{b}) \in \mathbb{R}^{N^{\text{grid}}}$, of $f_{\text{NN}}(\mathbf{X}^{\text{grid}}; \mathbf{W}, \mathbf{b})$ as follows:

$$\mathbf{a}_k(\mathbf{W}, \mathbf{b}) = \frac{\sqrt{2}}{\sqrt{2^{\mathbb{I}[k=0]} N^{\text{grid}}}} \cdot \sum_{n=0}^{N^{\text{grid}}-1} f_{\text{NN}}(\mathbf{x}^{\text{grid}}_n; \mathbf{W}, \mathbf{b}) \cdot \cos\left(\frac{\pi k(2n+1)}{2N^{\text{grid}}}\right), \tag{9}$$

for $k = 0, \ldots, N^{\text{grid}} - 1$.

Step 3: Lowpass Filtering. We select a filtering threshold $t \in [0, 1]$, where $t = 0$ recovers the original f_{NN}, and as t increases, frequencies are filtered out from highest to lowest. We then zero out all DCT coefficients $\mathbf{a}_k(\mathbf{W}, \mathbf{b})$ for which $k \geq (1 - t)N^{\text{grid}}$, and construct the lowpass filtered

function using the type-II inverse orthonormal discrete cosine transformation:

$$\tilde{f}_{\text{NN}}(\mathbf{x}_n^{\text{grid}}; \mathbf{W}, \mathbf{b}) = \frac{1}{\sqrt{N^{\text{grid}}}} \cdot \mathbb{I}[t < 1] \cdot \mathbf{a}_0(\mathbf{W}, \mathbf{b}) + \frac{\sqrt{2}}{\sqrt{N^{\text{grid}}}} \cdot \sum_{k=1}^{N^{\text{grid}}-1} \mathbb{I}[k < (1-t)N^{\text{grid}}] \cdot \mathbf{a}_n(\mathbf{W}, \mathbf{b}) \cdot \cos\left(\frac{\pi k(2n+1)}{2N^{\text{grid}}}\right), \tag{10}$$

for $n = 0, \ldots, N^{\text{grid}} - 1$.

Step 4: Inference. Lastly, we incorporate the lowpass filtered function into the BNN model by replacing f_{NN} with $\tilde{f}_{\text{NN}}$ in the model in Equation 8 as follows:

$$\begin{aligned} \mathbf{W}, \mathbf{b} &\sim p(\mathbf{W}, \mathbf{b}) \\ y|\mathbf{x}, \mathbf{W}, \mathbf{b} &\sim \mathcal{N}(y|\mathbf{0}_{\mathbf{x}}^{\intercal} \cdot \tilde{f}_{\text{NN}}(\mathbf{X}^{\text{grid}}; \mathbf{W}, \mathbf{b}), \sigma_\epsilon^2), \end{aligned} \tag{11}$$

where $\mathbf{0}_{\mathbf{x}}$ is a vector of 0s with a 1 corresponding to the row of $\mathbf{X}^{\text{grid}}$ closest to $\mathbf{x}$. Since the lowpass filter is a linear operation, it is differentiable and enables pairing with any inference method available in standard probabilistic programming languages.

Computational Efficiency. We can vectorize the above algorithm as follows:

1. Let $\mathbf{T} \in \mathbb{R}^{N^{\text{grid}} \times N^{\text{grid}}}$ denote the matrix corresponding to the type-II orthonormal discrete cosine transform on a grid of points $\mathbf{X}^{\text{grid}} \in \mathbb{R}^{N^{\text{grid}}}$:

$$\mathbf{T}_{\text{DCT}_{kn}} = \frac{\sqrt{2}}{\sqrt{2^{\mathbb{I}[k=0]} N^{\text{grid}}}} \cos\left(\frac{\pi k(2n+1)}{2N^{\text{grid}}}\right) \tag{12}$$

   So, $\mathbf{a} = \mathbf{T}_{\text{DCT}}\mathbf{f}$ and $\mathbf{T}_{\text{DCT}}^{\top}$ is the inverse transform.
2. Let $\mathbf{I}_{N^{\text{grid}},t}$ denote an identity matrix of size N^{grid} with $(\mathbf{I}_{N^{\text{grid}},t})_{kk} = 0$ for $k \geq (1-t)N^{\text{grid}}$.
3. Let $\tilde{f}_{\text{NN}}(\mathbf{X}^{\text{grid}}; \mathbf{W}, \mathbf{b}) = \mathbf{T}_{\text{DCT}}^{\top}\mathbf{I}_{N^{\text{grid}},t}\mathbf{T}_{\text{DCT}} \cdot f_{\text{NN}}(\mathbf{X}^{\text{grid}}; \mathbf{W}, \mathbf{b})$.
4. For a batch of inputs $\mathbf{X} \in \mathbb{R}^{N \times N^{\text{grid}}}$, we can write the likelihood as

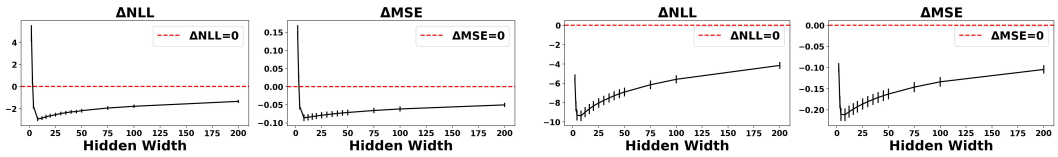
$$p(\mathbf{y}|\mathbf{X}, \mathbf{W}) = \mathcal{N}(\mathbf{y}|\mathbf{0}_{\mathbf{X}}\tilde{f}_{\text{NN}}(\mathbf{X}^{\text{grid}}; \mathbf{W}, \mathbf{b}), \sigma_\epsilon^2), \tag{13}$$

   where $\mathbf{0}_{\mathbf{X}} = (\mathbf{0}_{\mathbf{x}_1}^{\intercal}, \ldots, \mathbf{0}_{\mathbf{x}_N}^{\intercal})^{\intercal} \in \mathbb{R}^{N \times N^{\text{grid}}}$ is a matrix of 0s and a single 1 in each row that functions to select the elements of $\tilde{f}_{\text{NN}}(\mathbf{X}^{\text{grid}}; \mathbf{W}, \mathbf{b})$ corresponding to the training inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$.

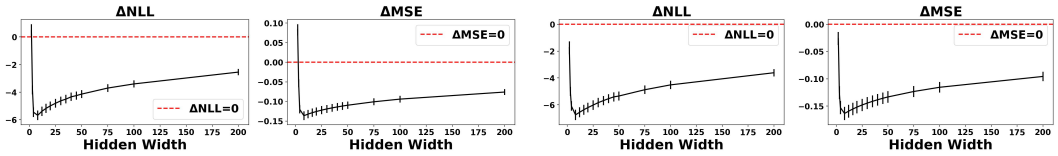
E Additional Results

E.1 Generalization Performance of Finite and Infinite Width BNNs

We show additional results comparing the generalization performance of finite and infinite width BNNs. Under all model mismatch (the limiting NNGP is not equivalent to the data generating model) we design, finite width BNNs can outperform infinite width BNNs. Additionally, we observe that larger model mismatch results in larger performance gap (Figure 6, 7). We also provide empirical results on the real dataset (Figure 8) and we see that finite width BNNs perform better on large datasets (Figure 8b, 8c, 8d).



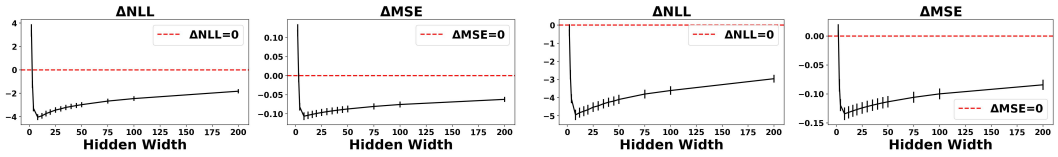
(a) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$



(b) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$

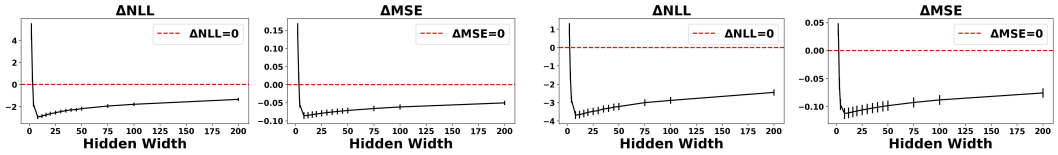
(c) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.0$

(d) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.0$



(e) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.5$

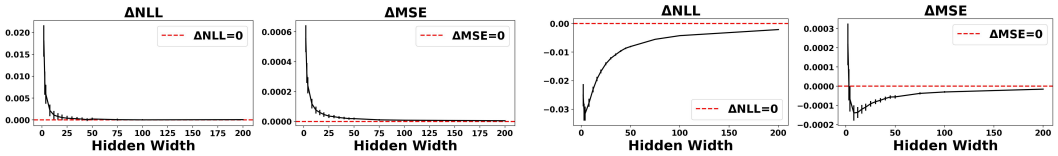
(f) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.5$



(g) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

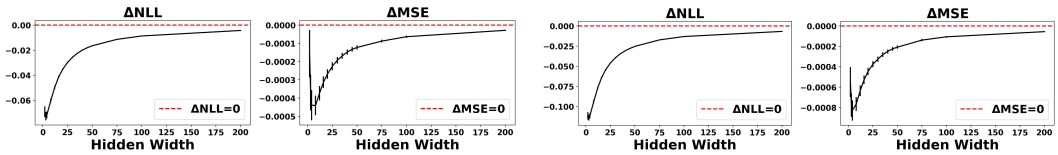
(h) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

Figure 5: Plots of the test performance of finite width BNNs: The data is generated from GP-RBF with lengthscale 0.5. We fit each dataset with BNNs with erf and ReLU nonlinearity. x-axis denotes the hidden-layer width of the BNN; y-axis denotes ΔNLL, ΔMSE (Equation 1,2). Being above/below the red dashed line means that finite width BNNs perform worse/better than the limiting NNGPs. We see that finite width BNNs can outperform the limiting NNGPs as the black curves are mostly below the red dashed line.



(a) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$

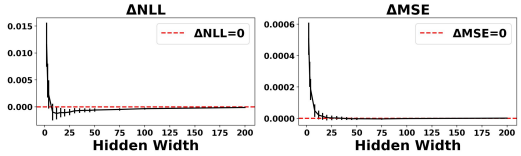
(b) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.0$



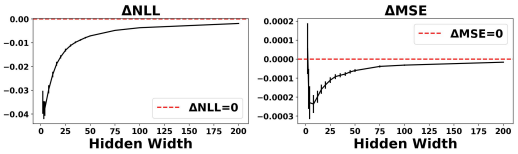
(c) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.5$

(d) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

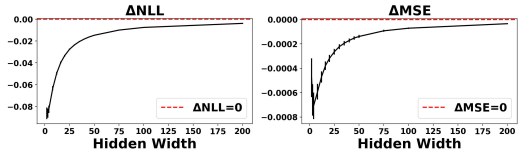
Figure 6: Plots of the test performance of finite width BNNs: The data is generated from GP-Arccos with $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$ and fit with BNNs with ReLU. There is no model mismatch in Figure 6a. The performance gap increases as the model mismatch increases (From Figure 6b) to 6d, the lowest point of the black curves gets smaller.



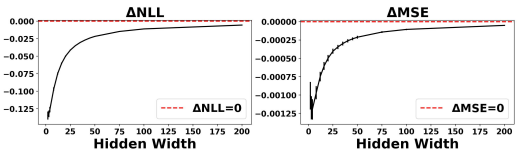
(a) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$



(b) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.0$

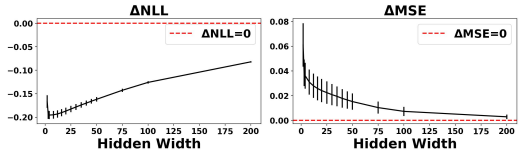


(c) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.5$

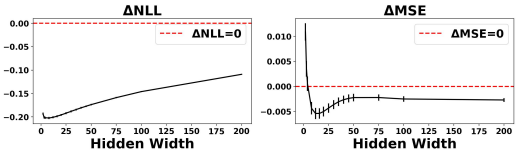


(d) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

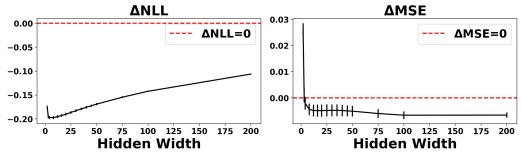
Figure 7: Plots of the test performance of finite width BNNs: The data is generated from GP-Arcsin with $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$ and fit with BNNs with erf. There is no model mismatch in Figure 7a. The performance gap increases as the model mismatch increases (From Figure 7b) to 7d, the lowest point of the black curves gets smaller.



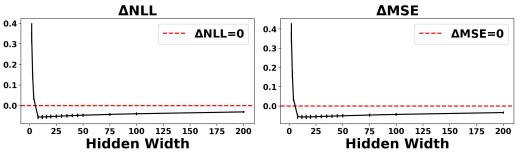
(a) UCI Boston: BNNs with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 3.0,\ \sigma_\epsilon = 0.5$



(b) UCI Energy (first output dimension): BNNs with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 5.0,\ \sigma_\epsilon = 0.5$



(c) UCI Energy (second output dimension): BNNs with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 5.0,\ \sigma_\epsilon = 0.5$

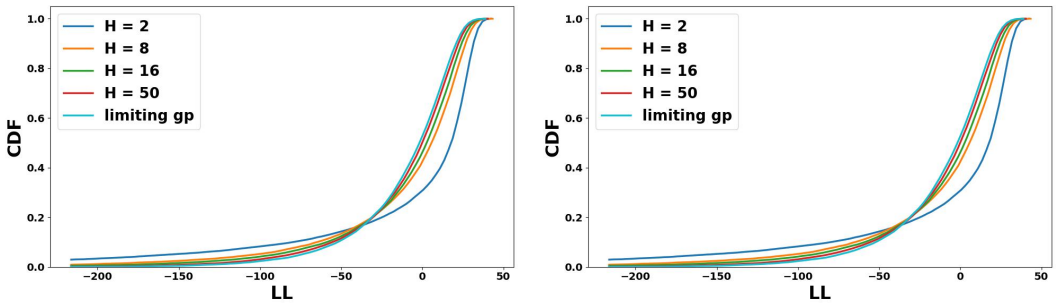


(d) Sic-97: BNNs with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 5.0,\ \sigma_\epsilon = 0.5$

Figure 8: Plots of the test performance of finite width BNNs on real datasets: There is often model mismatch in real life. We see that except for UCI Boston, finite width BNNs have better generalization performance.

E.2 Log Marginal Likelihood (LML)

We provide additional results of the CDF of LML. Similar to Figure 2a, 2b, we see that under different types of data generating models, BNNs with smaller hidden widths generate more diverse datasets than the limiting NNGP (the CDF of LML over datasets drawn from the data generative GP has more mass on the tail).



(a) $\mathcal{D} \sim$ GP-Arccos(0.5); BNN with ReLU, $\sigma^2_{\mathbf{W}} = \sigma^2_{\mathbf{b}} = 2.0$

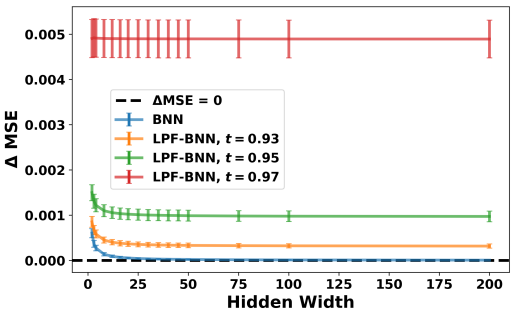
(b) $\mathcal{D} \sim$ GP-Arcsin(0.5); BNN with erf, $\sigma^2_{\mathbf{W}} = \sigma^2_{\mathbf{b}} = 2.0$

Figure 9: CDF of LML (Eqn 4) with datasets $\mathcal{D}$ sampled from BNNs with different hidden widths: 9a $\mathcal{D} \sim$GP-Arccos(0.5); 9b $\mathcal{D} \sim$GP-Arcsin(0.5). x-axis denotes the values of LML while y-axis denotes cdf. Each color represents either a BNN with a specific hidden width or the limiting NNGP. BNNs with smaller hidden widths generate more datasets with higher LML and lower LML than the limiting GP at the same time.

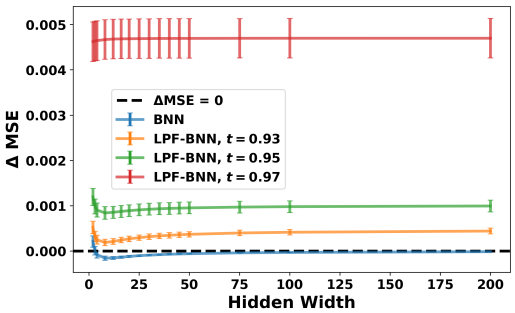
E.3 Lowpass-filtered BNNs

Removing high-frequency components from the BNN hurts its ability to adapt to model mismatch. We fix the data-generating process to be that of a GP. Across varying widths, we then compare the Δ_{MSE} (Equation 2) of a BNN with that of lowpass-filtered BNNs with different amounts of filtering, $t \in [0, 1]$. As expected, as the width increases, Δ_{MSE} of the BNN converges to 0. Moreover, under no model mismatch (i.e. when the limiting NNGP model matches the data generating process, as in Figure 10a), the BNN always under-performs the limiting NNGP. In contrast, under model mismatch (remainder of Figure 10, as well as Figures 11 and 12), small-width BNNs always outperform the NNGP (i.e. $\Delta\text{MSE} < 0$). As we hypothesized, removing the high-frequency components from prior functions hinders the BNN's ability to adapt to model mismatch.

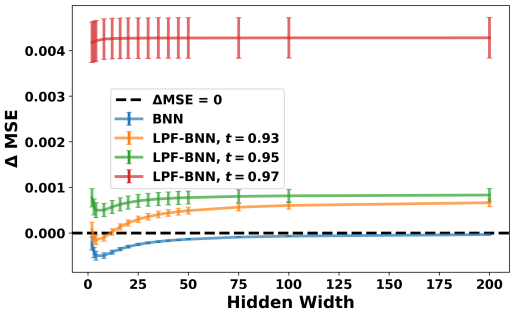
A qualitative comparison of a BNN vs. a Lowpass-filtered BNN. Figures 13 and 14 show how a BNN ($t = 0$) differs from a lowpass filtered BNN ($t > 0$), both in its prior and in its posterior predictive distributions. Specifically, as t increases, the functions being drawn become more smooth.



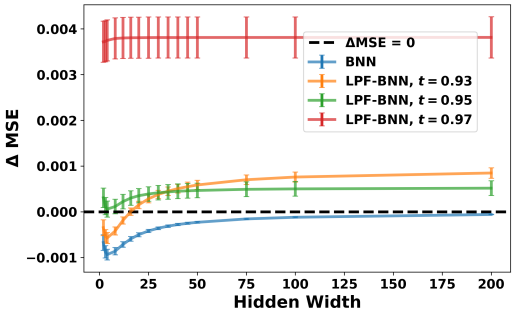
(a) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$



(b) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.0$

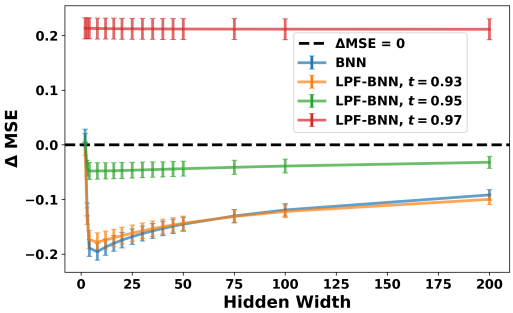


(c) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.5$

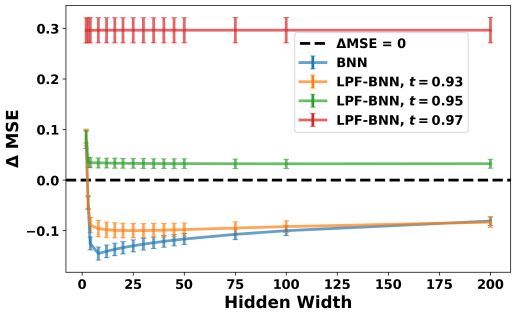


(d) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

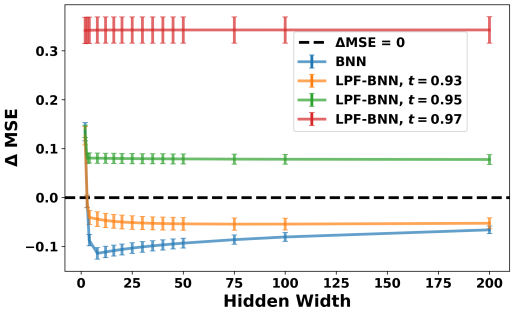
Figure 10: Δ_{MSE} of BNNs and lowepass-filtered BNNs, as a function of hidden width H (ΔMSE < 0 means a worse performance than the limiting NNGP). Higher thresholds t correspond to more high frequencies removed. The performance is averaged over 200 datasets drawn from a GP-Arccos(0.5). Error bars represent standard errors.



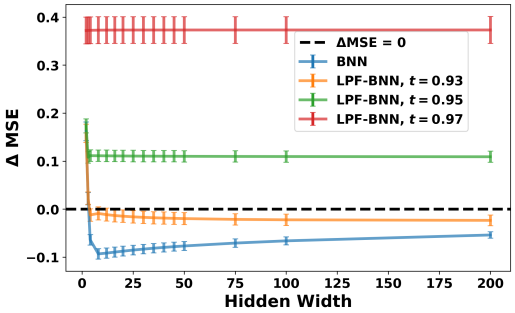
(a) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$



(b) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.0$

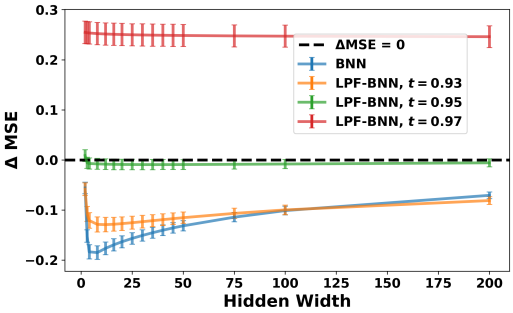


(c) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.5$

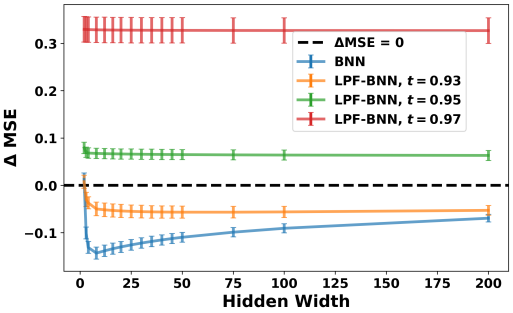


(d) BNN with ReLU, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

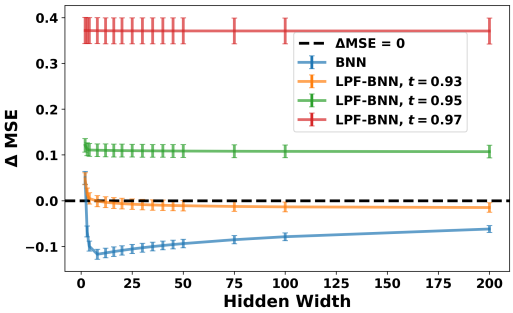
Figure 11: Analogous to Figure 10 but with data generated from a GP with an RBF kernel, $\mathcal{D} \sim$ GP-RBF(0.5) (so the limiting NNGP has an Arccos kernel).



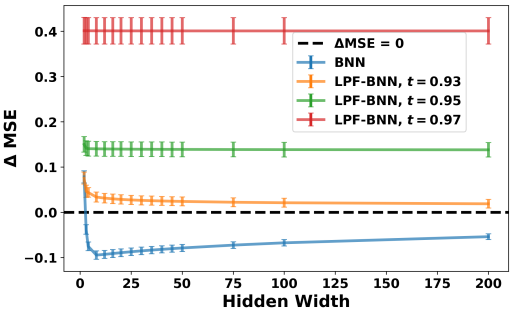
(a) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 0.5$



(b) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.0$



(c) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 1.5$



(d) BNN with erf, $\sigma_{\mathbf{W}}^2 = \sigma_{\mathbf{b}}^2 = 2.0$

Figure 12: Analogous to Figure 10 but with data generated from a GP with an RBF kernel, $\mathcal{D} \sim$ GP-RBF(0.5), and a BNN with erf activation (so the limiting NNGP has an Arcsin kernel).

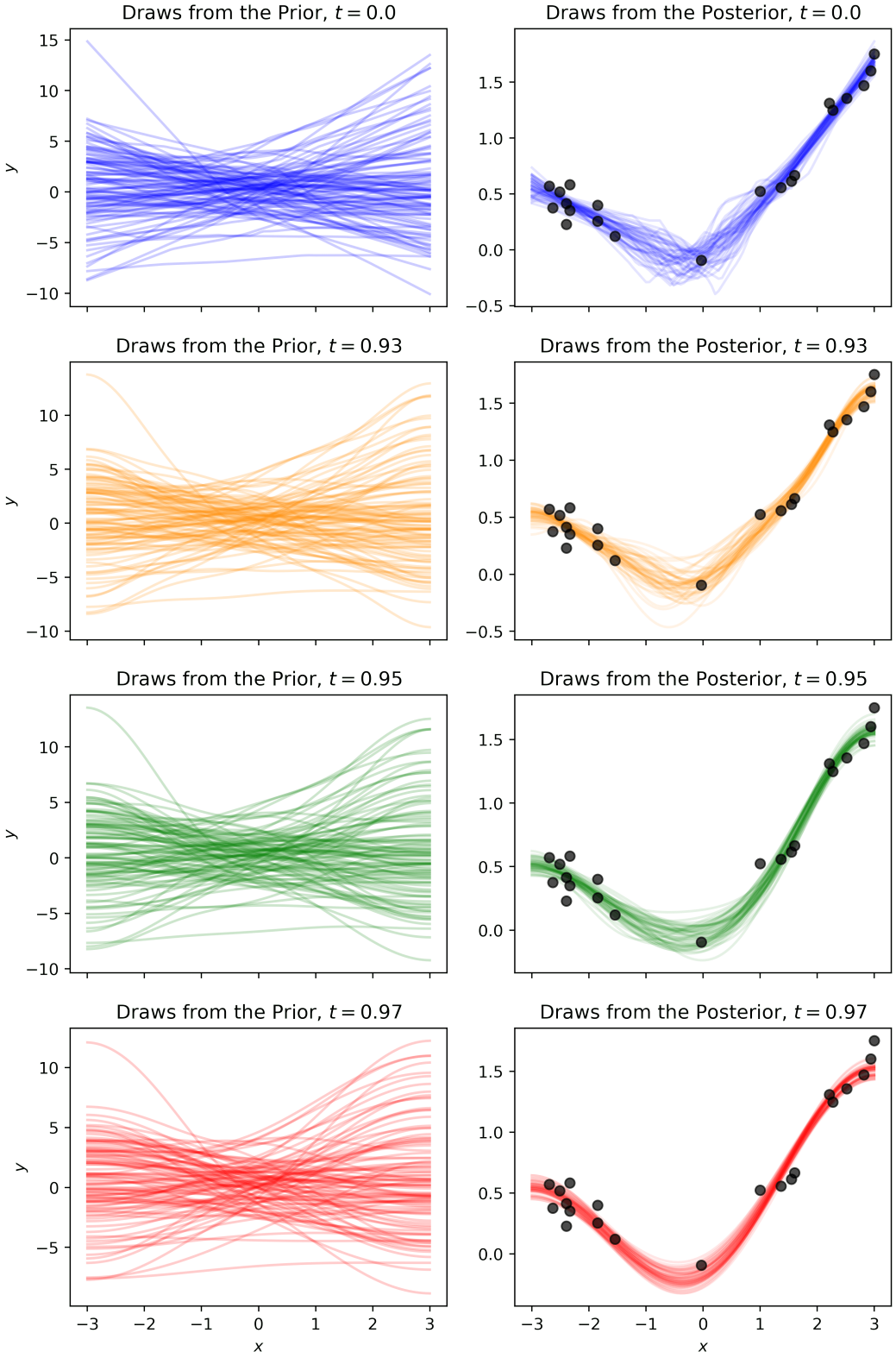


Figure 13: A BNN with a width of 15, a prior weight variance of 2.0 and a ReLU activation, trained on data drawn from a GP-Arccos(0.5). As the filtering threshold t increases, the functions under both the prior and posterior become less wiggly.

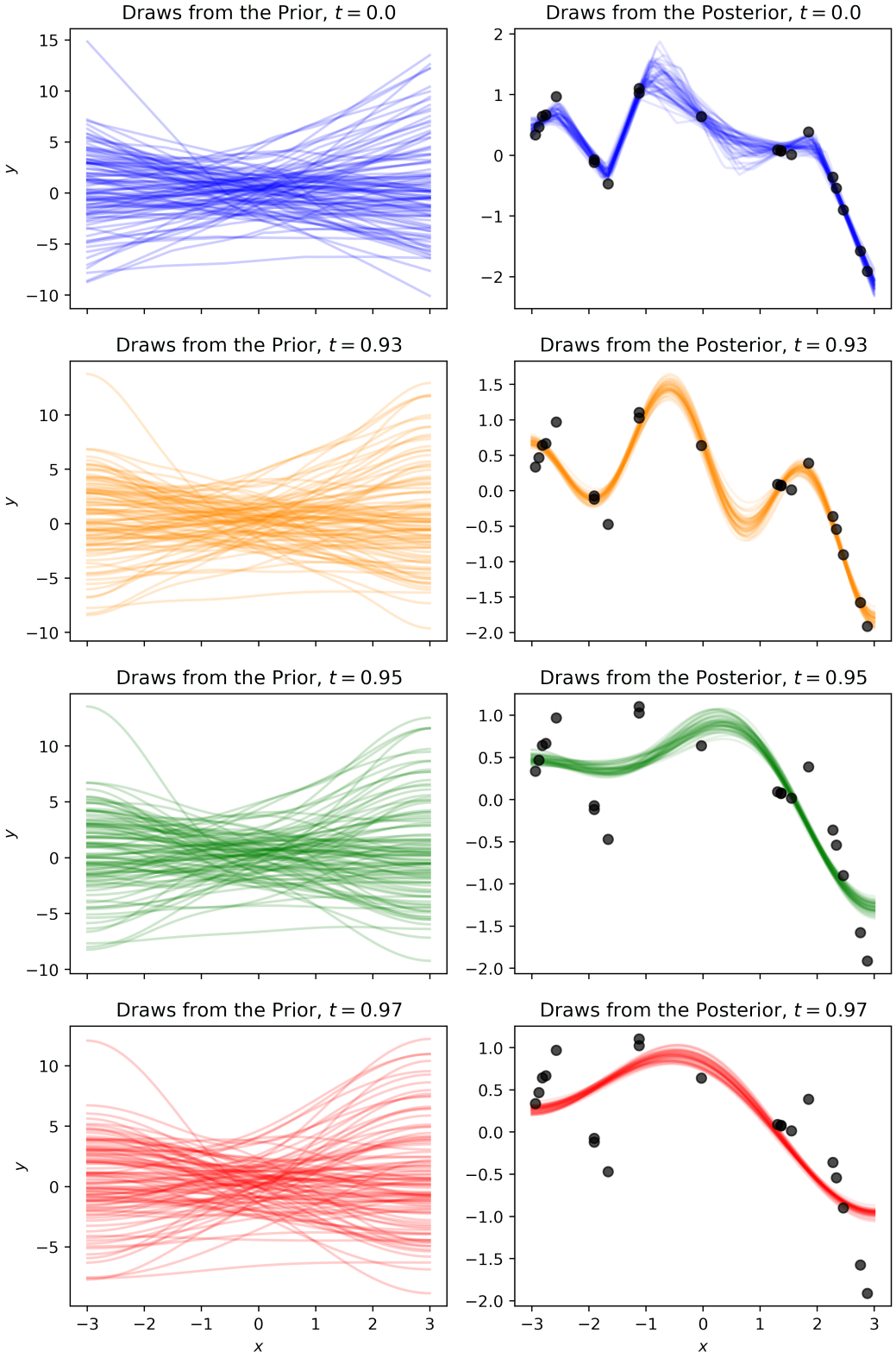


Figure 14: A BNN with a width of 15, a prior weight variance of 2.0 and a ReLU activation, trained on data drawn from a GP-RBF(0.5). As the filtering threshold t increases, the functions under both the prior and posterior become less wiggly.