Exploring the Uncertainty Properties of Neural Networks' Implicit Priors in the Infinite-Width Limit

Ben Adlam [*][1][2] Jaehoon Lee [*][1] Lechao Xiao [*][1] Jeffrey Pennington [1] Jasper Snoek [1]

Abstract

Modern deep learning models have achieved great success in predictive accuracy for many data modalities. However, their application to many real-world tasks is restricted by poor uncertainty estimates, such as overconfidence on out-of-distribution (OOD) data and ungraceful failing under distributional shift. Previous benchmarks have found that ensembles of neural networks (NNs) are typically the best calibrated models on OOD data. Inspired by this, we leverage recent theoretical advances that characterize the function-space prior of an ensemble of infinitely-wide NNs as a Gaussian process, termed the neural network Gaussian process (NNGP). We use the NNGP with a softmax function to build a probabilistic model for multi-class classification and marginalize over the latent Gaussian outputs to sample from the posterior. This gives us a better understanding of the prior NNs place on function space and allows a direct comparison of the calibration of the NNGP and its finite-width analogue. We also examine the calibration of previous approaches to classification with the NNGP, which treat classification problems as regression to the one-hot labels. In this case the Bayesian posterior is exact, and we compare several heuristics to generate a categorical distribution over classes. We find these methods are well calibrated under distributional shift.

1. Introduction

The representations learned by NNs are difficult to interpret and able to fit spurious correlations. Therefore if these models are ever to be widely applied in high-risk areas like medicine or autonomous driving, they must gain our trust in other ways. Specifically, we might ask that our models are calibrated, that is, rather than simply providing point estimates, they also report their confidence in the prediction. Over many predictions, well calibrated models report confidences that are consistent with measured performance. The Brier score (BS), expected calibration error (ECE), and negative log-likelihood (NLL) are common measurements of calibration (Brier, 1950; Naeini et al., 2015; Gneiting & Raftery, 2007).

Empirically, there are many concerning findings about the calibration of deep learning techniques, particularly on *out-of-distribution* (OOD) data whose distribution differs from that of the training data (MacKay, 1992; Hein et al., 2019). For in-distribution data, *post-hoc* calibration techniques such as temperature scaling tuned on a validation set (Platt et al., 1999; Guo et al., 2017) often give excellent results; however, such methods have not been found to be robust on shifted data and indeed sometimes even reduce calibration on such data (Ovadia et al., 2019). Thus finding ways to detect or build models that produce reliable signals when making predictions on OOD data is a key challenge.

Sometimes data can be only slightly OOD or can shift from the training distribution gradually over time. This is called *dataset shift* (Quionero-Candela et al., 2009) and is important in practice for models dealing with seasonality effects, for example. While perfect calibration under arbitrary distributional shift is impossible, simulating plausible kinds of distributional shift that may occur in practice at different intensities can be a useful tool for evaluating the calibration of existing models. A recently proposed benchmark takes this approach (Ovadia et al., 2019). Using several kinds of common image corruptions applied at various intensities, the authors observed the degradation in accuracy expected of models trained only on clean images (Hendrycks & Dietterich, 2019; Mu & Gilmer, 2019), but also saw very different levels of calibration, with deep ensembles (Lakshminarayanan et al., 2017) proving the best.

Bridging Bayesian Learning and Neural Networks In principle, Bayesian methods provide a promising way to tackle calibration, allowing us to define models with and

[*] Equal contribution [1] Google Brain, USA [2] Work done as a member of the Google AI Residency program. Correspondence to: Ben Adlam <adlam@google.com>, Jaehoon Lee <jaehlee@google.com>, Lechao Xiao <xlc@google.com>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning. Copyright 2020 by the author(s).

infer under specific aleatory and epistemic uncertainty. Typically, the datasets on which deep learning has proven successful have high SNR, meaning epistemic uncertainty is dominant and model averaging is crucial because our overparameterized models are not determined by the training data. Indeed Wilson (2020) argues that ensembles are a kind of Bayesian model average.

Ongoing theoretical work has built a bridge between NNs and Bayesian methods (Neal, 1994a; Lee et al., 2018; Matthews et al., 2018a), by identifying NNs with Gaussian processes as the width of the network becomes very large. Specifically, the neural network Gaussian process (NNGP) describes the prior on function space that is realized by an i.i.d. prior over the parameters. The function space prior is a GP with a specific kernel that is defined recursively with respect to the layers. While the many heuristics used in training NNs may obfuscate the issue, little is known theoretically about the uncertainty properties implied by even basic architectures and initializations of NNs. Indeed theoretically understanding overparameterized NNs is a major open problem. With the NNGP prior in hand, it is possible to disambiguate between the uncertainty properties of the NN prior and those due to the specific optimization decisions by performing Bayesian inference.

1.1. Summary of contributions

Unlike previous works, we construct a valid probabilistic model for classification tasks using the NNGP, that is, a label's prediction is always a categorical distribution. We perform neural network Gaussian process classification (NNGP-C) using a softmax link function to exactly mirror NNs used in practice. We perform a detailed comparison of NNGP-C against its corresponding NN on clean, OOD, and shifted test data and find NNGP-C to be significantly better calibrated and more performant than the NN.

Next, we evaluate the calibration of neural network Gaussian process regression (NNGP-R) on both UCI regression problems and classification on CIFAR10. As the posterior of NNGP-R is a multivariate normal and so not a categorical distribution, a heuristic must be used to calculate confidences for classification problems. On the full benchmark of Ovadia et al. (2019), we compare several such heuristics, and against the standard RBF kernel and ensemble methods. We find the calibration of NNGP-R to be competitive with the best results reported in (Ovadia et al., 2019).

2. Full Bayesian Treatment of Classification with Neural Kernels

It is common to interpret the logits of a NN once mapped through a softmax as a categorical distribution over the labels for each point. Indeed cross entropy loss is sometimes motivated as the KL divergence between the predicted distribution and the observed label. Similarly, while the initialization scheme used for a NN's parameters is often chosen for optimization reasons, it can also be thought of as a prior. This implicit prior over functions and over the distribution of labels has effects, despite the decision of most common training algorithms in deep learning to forgo explicitly trying to find its posterior. In this section, we take seriously this implicit prior and utilize the simple characterization it has over logits in the infinite-width limit to define a probabilistic model for mutli-class classification as

$$y \sim \mathrm{softmax}(f(x)), \text{ where } f \sim \mathcal{GP}(\mathbf{0}, \mathcal{K}), \tag{1}$$

where $\mathcal{K}$ is the NNGP kernel. If the prior is a correct model for the data generation process, then the posterior is optimal for inference. Thus, by avoiding heuristic approaches to inference, we are able to directly evaluate the prior. Then by comparing this to more standard gradient-based training methods, we can understand their effect on calibration.

For training data $(\mathcal{X}, \mathcal{Y})$, the posterior on a test point x can be found by marginalizing out the latent space. Denote $F := f(\mathcal{X})$, $f := f(x)$, and $\mathbf{F}$ as the concatenation of F and f, then

$$p(y|\mathcal{X}, \mathcal{Y}) = \int \mathrm{softmax}(f) p(f|\mathcal{Y}) \, \mathrm{d}f, \tag{2}$$

where

$$\begin{aligned} p(\mathbf{F}|\mathcal{Y}) &= p(\mathcal{Y}|F)p(\mathbf{F})/p(\mathcal{Y}) \\ &\propto \mathcal{N}(\mathbf{F}; \mathbf{0}, \mathcal{K}(\mathbf{F}, \mathbf{F})) \prod_i \mathrm{softmax}(F_i)_{\mathcal{Y}_i}. \end{aligned} \tag{3}$$

See (Williams & Rasmussen, 2006) for details. We generate samples from the joint posterior distribution of f and F using elliptical slice sampling (ESS) (Murray et al., 2010). Note that the latent space dimension in the datasets we consider is substantial, which makes inference with ESS computationally intensive, especially with hyperparameter tuning of the kernel. Therefore, we focus our attention on FC and CNN-Vec kernels.

Our main findings for NNGP-C, summarized in Fig. 1 and Table S1, show that NNGP-C is well calibrated and outperforms the corresponding NN. This indicates that the MAP-based training of the NN is partly responsible for its poor calibration and helps explain the success of ensembles.

3. Regression with the NNGP

As observed in Sec. 2, inference with NNGP-C is challenging as the posterior is intractable. In this section, we consider Gaussian process regression using the NNGP (abbreviated as NNGP-R), which is defined by the model

$$y \sim f(x) + \varepsilon, \text{ where } f \sim \mathcal{GP}(\mathbf{0}, \mathcal{K}), \tag{4}$$

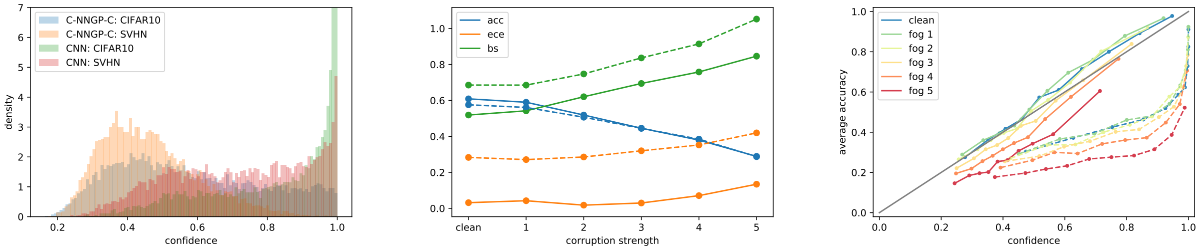

Figure 1. Investigating the calibration of Gaussian process classification with CNN-GP kernels. (**left**) Histogram of the confidence of the posterior distribution for each test point. We compare the C-NNGP-C and a finite width CNN on an identically distributed test set (CIFAR10) and an OOD test set (SVHN). C-NNGP-C shows lower confidence and higher entropy on both test sets compared to the CNN, which has very high confidence on many points as indicated by the spike around 1. On the clean data the CNN's overconfidence hurts its calibration, as it achieves worse ECE and BS than C-NNGP-C. (**middle**) The performance of both models, C-NNGP-C is solid and CNN dashed, under increasing distributional shift given by the CIFAR10 fog corruption. The accuracy of the CNN and C-NNGP-C are comparable as the shift intensity increases, but C-NNGP-C remains better calibrated throughout. (**right**) We bin the test set into ten bins sorted by confidence, and we plot mean confidence against mean accuracy. C-NNGP-C remains closer to $x = y$ than the CNN, which shifts toward increasing overconfidence.

where $\mathcal{K}$ is the NNGP kernel and $\varepsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is an independent noise term. One major advantage of NNGP-R is that the posterior is analytically tractable. The posterior at a test point x has a Gaussian distribution with mean and variance given by

$$\begin{aligned} \mu(x) &= \mathcal{K}(x, \mathcal{X})\mathcal{K}_\epsilon(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y} \\ \sigma^2(x) &= \mathcal{K}(x, x) - \mathcal{K}(x, \mathcal{X})\mathcal{K}_\epsilon(\mathcal{X}, \mathcal{X})^{-1}\mathcal{K}(\mathcal{X}, x) \end{aligned} \tag{5}$$

where $(\mathcal{X}, \mathcal{Y})$ is the training set of inputs and targets respectively and $\mathcal{K}_\epsilon \equiv \mathcal{K} + \sigma_\epsilon^2\mathbf{I}$. For regression problems where $y \in \mathbb{R}^d$, the variance describes the model's uncertainty about the test point. Note that while this avoids the difficulties of approximate inference methods like MCMC, computation of the kernel inverse means running time scales cubically with the dataset size.

3.1. Benchmark on UCI Dataset

We perform non-linear regression experiments proposed by Hernández-Lobato & Adams (2015), which is a standard benchmark for evaluating uncertainty of Bayesian NNs. We use all datasets except for `Protein` and `Year`. Each dataset is split into 20 train/test folds. We report our result in Table 1, comparing against the following strong baselines: Probabilistic BackPropagation with the Matrix Variate Gaussian distribution (PBP-MV) (Sun et al., 2017), Monte-Carlo Dropout (Gal & Ghahramani, 2016) evaluated with hyperparameter tuning as done in Mukhoti et al. (2018) and Deep Ensembles (Lakshminarayanan et al., 2017). We also evaluated a standard GP with RBF kernel $K_{\text{RBF}}(x, x') = \beta \exp\left(-\gamma||x - x'||^2\right)$ for comparison.

Instead of maximizing train NLL for model selection, we performed hyperparameter search on a validation set (we further split the training set so that overall train/valid/test split is 80/10/10), as commonly done in NN model selection and in the BNN context applied in (Mukhoti et al., 2018).

We found that NNGP-R can outperform and remain competitive with existing methods in terms of both root-mean-squared-error (RMSE) and negative-log-likelihood (NLL). In Table 1, we observe that NNGP-R achieves the lowest RMSE on the majority (5/8) of the datasets and competitive NLL.

Table 1. Result for regression benchmark on UCI Datasets with FC-NNGP-R. Note $\pm x$ reports the standard error around estimated mean for 20 splits. Average Test (top) RMSE (bottom) Negative Log-Likelihood Performance.

Dataset	(m, d)	PBP-MV	Dropout	Deep Ensembles	RBF	FC-NNGP-R
Boston Housing	(506, 13)	3.11 ± 0.15	**2.90 ± 0.18**	3.28 ± 1.00	3.24 ± 0.21	3.07 ± 0.24
Concrete Strength	(1030, 8)	5.08 ± 0.14	**4.82 ± 0.16**	6.03 ± 0.58	5.63 ± 0.24	5.25 ± 0.20
Energy Efficiency	(768, 8)	**0.45 ± 0.01**	0.54 ± 0.06	2.09 ± 0.29	0.50 ± 0.01	0.57 ± 0.02
Kin8nm	(8192, 8)	**0.07 ± 0.00**	0.08 ± 0.00	0.09 ± 0.00	**0.07 ± 0.00**	**0.07 ± 0.00**
Naval Propulsion	(11934, 16)	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
Power Plant	(9568, 4)	3.91 ± 0.04	4.01 ± 0.04	4.11 ± 0.17	3.82 ± 0.04	**3.61 ± 0.04**
Wine Quality Red	(1588, 11)	0.64 ± 0.01	0.62 ± 0.01	0.64 ± 0.04	0.64 ± 0.01	**0.57 ± 0.01**
Yacht Hydrodynamics	(308, 6)	0.81 ± 0.06	0.67 ± 0.05	1.58 ± 0.48	0.60 ± 0.07	**0.41 ± 0.04**
Boston Housing	(506, 13)	2.54 ± 0.08	**2.40 ± 0.04**	2.41 ± 0.25	2.63 ± 0.09	2.65 ± 0.13
Concrete Strength	(1030, 8)	3.04 ± 0.03	**2.93 ± 0.02**	3.06 ± 0.18	3.52 ± 0.11	3.19 ± 0.05
Energy Efficiency	(768, 8)	1.01 ± 0.01	1.21 ± 0.01	1.38 ± 0.22	**0.78 ± 0.06**	1.01 ± 0.04
Kin8nm	(8192, 8)	**-1.28 ± 0.01**	-1.14 ± 0.01	-1.20 ± 0.02	-1.11 ± 0.01	-1.15 ± 0.01
Naval Propulsion	(11934, 16)	-4.85 ± 0.06	-4.45 ± 0.00	-5.63 ± 0.05	**-10.07 ± 0.01**	-10.01 ± 0.01
Power Plant	(9568, 4)	2.78 ± 0.01	2.80 ± 0.01	2.79 ± 0.04	2.94 ± 0.01	**2.77 ± 0.02**
Wine Quality Red	(1588, 11)	0.97 ± 0.01	0.93 ± 0.01	0.94 ± 0.12	-0.78 ± 0.07	**-0.98 ± 0.06**
Yacht Hydrodynamics	(308, 6)	1.64 ± 0.02	1.25 ± 0.01	1.18 ± 0.21	**0.49 ± 0.06**	1.07 ± 0.27

4. Classification as Regression

Formulating classification as regression often leads to good results, despite being less principled (Rifkin et al., 2003; Rifkin & Klautau, 2004). By doing so, we can compare exact inference for GPs to trained NNs on well-studied image classification tasks. Recently, various studies of infinite NNs have considered classification as regression tasks, treating the one-hot labels as independent regression targets (e.g. (Lee et al., 2018; Novak et al., 2019b; Garriga-Alonso et al., 2019)). Predictions are then obtained as the argmax of the mean in Eq. (5), *i.e.* $\arg\max_k \mu(x)_k$.

However, this approach does not provide confidences corresponding to the predictions. Note that the posterior gives support to all of $\mathbb{R}^d$, including points that are known to be impossible. Thus, a heuristic is required to extract meaningful uncertainty estimates from the posterior Eq. (5), even though these confidences will not correspond to the Bayesian posterior of any model.

Following (Albert & Chib, 1993; Girolami & Rogers, 2006), we produce a categorical distribution for each test point x, denoted p_x, by defining

$$\begin{aligned} p_x(i) &:= \mathbb{P}[z_i = \max\{y_1, \ldots, y_d\}] \\ &= \int \mathbf{1}(i = \text{argmax}_j y_j) \prod_{k=1}^{K} p(y_k | x, \mathcal{X}, \mathcal{Y}) dy, \quad (6) \end{aligned}$$

where $(y_1, \ldots, y_d)$ is sampled from the posterior for (x, y) and we used the independence of the posterior for each class. Note that we also treat the predictions on different test points independently. In general, Eq. (6) does not have a analytic expression, and we resort to Monte-Carlo estimation. We refer readers to Sec. C for comparison to other heuristics (e.g. passing the mean predictor through a softmax function and pairwise comparison). While this is heuristic, we find it is well calibrated (see Fig. 2). This is perhaps because the posterior Eq. (5) still represents substantial model averaging, and most uncertainty in high SNR cases is epistemic rather than aleatory.

4.1. Benchmark on CIFAR10

We examine the calibration of NNGP-R on increasingly corrupted images of CIFAR10-C (Hendrycks & Dietterich, 2019) using the benchmark of (Ovadia et al., 2019). The results are displayed in Fig. 2. While FC-NNGP is similar to the standard RBF kernel, the C-NNGP is able to outperform both in terms of calibration and accuracy. Moreover, we find that at severe corruption levels, the CNN-GP actually outperforms all methods in (Ovadia et al., 2019) (compare against their Table G1) in BS and ECE.

5. Discussion

In this work, we explored several methods that exploit neural networks' implicit priors over functions in order to generate uncertainty estimates, using the corresponding Neural Network Gaussian Process (NNGP) as a means to harness the power of an infinite ensemble of NNs in the infinite-width limit. Using the NNGP, we performed fully Bayesian classification (NNGP-C) and regression (NNGP-R) and also examined heuristics for generating confidence estimates when classifying via regression. Across the board, we found that the NNGP provides good uncertainty estimates and generally delivers well-calibrated models even on OOD data. We found that NNGP-R is competitive with SOTA methods on the UCI regression task and remained calibrated even for severe levels of corruption. Despite their good calibration properties, as pure kernel methods, NNGP-C and NNGP-R cannot always compete with modern NNs in terms of accuracy.

In Appendix D, we show that adding an NNGP to the last-layer of a pre-trained model (NNGP-LL), allowed us to simultaneously obtain high accuracy and improved calibration. Moreover, we found NNGP-LL to be a simple and efficient way to generate uncertainty estimates with potentially very little data, and that it outperforms all other last-layer methods for generating uncertainties we studied.

Overall, we believe that the infinite-width limit provides a promising direction to improve and better understand uncertainty estimates for NNs.

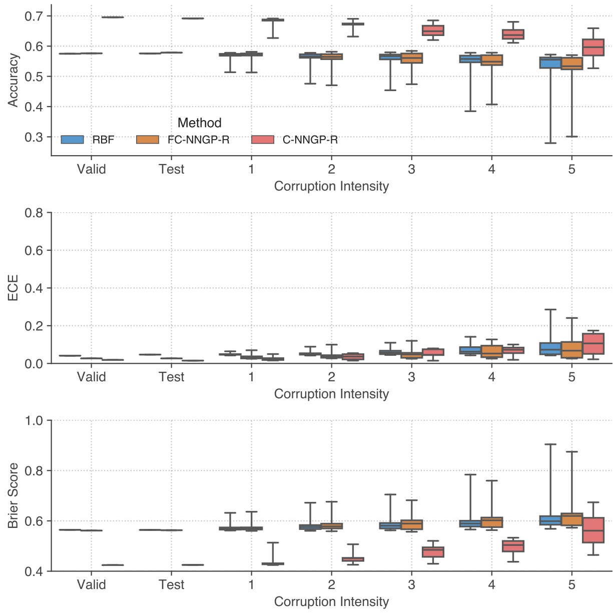

Figure 2. Uncertainty metrics across shift levels on CIFAR10 using NNGP-R. CNN kernels perform best as well as being more robust to corruption. See Table S3 for numerical values at each quartile. All methods remain well calibrated for all intensities of shifts, with C-NNGP-R performing best, and significantly better than methods in (Ovadia et al., 2019); contrast against their Figure 2 and S4.

References

Albert, J. H. and Chib, S. Bayesian analysis of binary and polychotomous response data. *Journal of the American statistical Association*, 88(422):669–679, 1993.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*. 2019.

Brier, G. W. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Dauphin, Y. N. and Schoenholz, S. S. Metainit: Initializing learning by learning to initialize. In *Advances in Neural Information Processing Systems*, 2019.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Garriga-Alonso, A., Aitchison, L., and Rasmussen, C. E. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019.

Girolami, M. and Rogers, S. Variational bayesian multinomial probit regression with gaussian process priors. *Neural Computation*, 18(8):1790–1817, 2006.

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1487–1495, 2017.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

Hastie, T. and Tibshirani, R. Classification by pairwise coupling. In *Advances in neural information processing systems*, pp. 507–513, 1998.

Hein, M., Andriushchenko, M., and Bitterwolf, J. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Hinton, G. E. and Salakhutdinov, R. R. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in neural information processing systems*, pp. 1249–1256, 2008.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2661–2671, 2019.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*. 2017.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S., Pennington, J., and Sohl-dickstein, J. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pp. 8570–8581, 2019.

Li, Z., Wang, R., Yu, D., Du, S. S., Hu, W., Salakhutdinov, R., and Arora, S. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.

MacKay, D. J. The evidence framework applied to classification networks. *NEURAL COMPUTATION*, 4:720–736, 1992.

Matthews, A., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 4 2018a. URL `https://openreview.net/forum?id=H1-nGgWC-`.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 9 2018b.

Mu, N. and Gilmer, J. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

Mukhoti, J., Stenetorp, P., and Gal, Y. On the importance of strong baselines in bayesian deep learning. *arXiv preprint arXiv:1811.09385*, 2018.

Murray, I., Prescott Adams, R., and MacKay, D. J. Elliptical slice sampling. 2010.

Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Neal, R. M. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994a.

Neal, R. M. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, Dept. of Computer Science, 1994b.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. Neural tangents: Fast and easy infinite neural networks in python. *arXiv preprint arXiv:1912.02803*, 2019a.

Novak, R., Xiao, L., Lee, J., Bahri, Y., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019b.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

Platt, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. In *Advances In Neural Information Processing Systems*, pp. 3360–3368, 2016.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN 0262170051.

Rifkin, R. and Klautau, A. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

Rifkin, R., Yeo, G., Poggio, T., et al. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *International Conference on Learning Representations*, 2017.

Shankar, V., Fang, A. C., Guo, W., Fridovich-Keil, S., Schmidt, L., Ragan-Kelley, J., and Recht, B. Neural kernels without tangents. *ArXiv*, abs/2003.02237, 2020.

Sun, S., Chen, C., and Carin, L. Learning structured weight uncertainty in bayesian neural networks. In *Artificial Intelligence and Statistics*, pp. 1283–1292, 2017.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Wilson, A. G. The case for bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial Intelligence and Statistics*, pp. 370–378, 2016.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., and Pennington, J. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, 2018.

Yang, G. and Schoenholz, S. Mean field residual networks: On the edge of chaos. In *Advances in Neural Information Processing Systems*. 2017.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference*, 2016.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Supplementary Material

A. Background

Neal (1994b) identified the connection between infinite-width NNs and Gaussian processes, showing that the outputs of a randomly initialized one-hidden layer NN converges to a Gaussian process as the number of neurons in the hidden layers approaches infinity. Let $z_i^l(x)$ describe the i^{th} pre-activation following a linear transformation in the l^{th} layer of a NN. At initialization, the parameters of the NN are independent and random, so the central-limit theorem can be used to show that the pre-activations become Gaussian with zero mean and a covariance matrix $\mathcal{K}(x, x') = \mathbb{E}[z_i^l(x) z_i^l(x')]$.

Knowing the distributions of the outputs, one can apply Bayes theorem to compute the posterior distribution for new observations, which we detail in Sec. 2 for classification and Sec. 3 for regression.

In this work, we focus on FC and CNN-Vec NNGPs, whose kernels are derived from fully-connected networks and convolution networks without pooling respectively. When it is required, we prepend FC- or C- to the NNGP to distinguish between these two variants. We use the Neural Tangents library of Novak et al. (2019a) to automate the transformation of finite-width NNs to their corresponding Gaussian processes.

A.1. Metrics

We use **Negative Log-Likelihood (NLL)**, **Brier Score** and **Expected Calibration Error (ECE)** for metrics for calibration following definition used in Ovadia et al. (2019). The first two are proper scoring rules, where an optimal score corresponds to perfect prediction. While ECE is not a proper scoring rule it is commonly used for its intuitive definition.

A.2. Further Detailed Description of the NNGP

In this section, we describe the FC-NNGP and the C-NNGP. Most of the contents are adopted from (Lee et al., 2018; Novak et al., 2019b; Lee et al., 2019), which we refer readers to for more technical details.

NNGP : Let $\mathcal{D} \subseteq \mathbb{R}^{n_0} \times \mathbb{R}^K$ denote the training set and $\mathcal{X} = \{x : (x, y) \in \mathcal{D}\}$ and $\mathcal{Y} = \{y : (x, y) \in \mathcal{D}\}$ denote the inputs and labels, respectively. Consider a fully-connected feed-forward network with L hidden layers with widths n_l, for $l = 1, ..., L$ and a readout layer with $n_{L+1} = K$. For each $x \in \mathbb{R}^{n_0}$, we use $h^l(x), x^l(x) \in \mathbb{R}^{n_l}$ to represent the pre- and post-activation functions at layer l with input x. The recurrence relation for a feed-forward network is defined as

$$\begin{cases} h^{l+1} & = x^l W^{l+1} + b^{l+1} \\ x^{l+1} & = \phi\left(h^{l+1}\right) \end{cases} \quad \text{and} \quad \begin{cases} W_{i,j}^l & = \frac{\sigma_\omega}{\sqrt{n_l}} \omega_{ij}^l \\ b_j^l & = \sigma_b \beta_j^l \end{cases}, \tag{S1}$$

where ϕ is a point-wise activation function, $W^{l+1} \in \mathbb{R}^{n_l \times n_{l+1}}$ and $b^{l+1} \in \mathbb{R}^{n_{l+1}}$ are the weights and biases, ω_{ij}^l and b_j^l are the trainable variables, drawn i.i.d. from a standard Gaussian $\omega_{ij}^l, \beta_j^l \sim \mathcal{N}(0, 1)$ at initialization, and σ_ω^2 and σ_b^2 are weight and bias variances.

As the width of the hidden layers approaches infinity, the Central Limit Theorem (CLT) implies that the outputs at initialization $\{f(x)\}_{x \in \mathcal{X}}$ converge to a multivariate Gaussian in distribution. Informally, this occurs because the pre-activations at each layer are a sum of Gaussian random variables (the weights and bias), and thus become a Gaussian random variable themselves. See (Poole et al., 2016; Schoenholz et al., 2017; Lee et al., 2018; Xiao et al., 2018; Yang & Schoenholz, 2017) for more details, and (Matthews et al., 2018b; Novak et al., 2019b) for a formal treatment.

Therefore, randomly initialized neural networks are in correspondence with a certain class of GPs (hereinafter referred to as NNGPs), which facilitates a fully Bayesian treatment of neural networks (Lee et al., 2018; Matthews et al., 2018a). More precisely, let f^i denote the i-th output dimension and $\mathcal{K}$ denote the sample-to-sample kernel function (of the pre-activation) of the outputs in the infinite width setting,

$$\mathcal{K}^{i,j}(x, x') = \lim_{\min(n_1, \ldots, n_L) \to \infty} \mathbb{E}\left[f^i(x) \cdot f^j(x')\right], \tag{S2}$$

then $f(\mathcal{X}) \sim \mathcal{N}(0, \mathcal{K}(\mathcal{X}, \mathcal{X}))$, where $\mathcal{K}^{i,j}(x, x')$ denotes the covariance between the i-th output of x and j-th output of x', which can be computed recursively (see Lee et al. (2018, §2.3). For a test input $x \in \mathcal{X}_T$, the joint output distribution

$f\left([x, \mathcal{X}]\right)$ is also multivariate Gaussian. Conditioning on the training samples, $f(\mathcal{X}) = \mathcal{Y}$, the distribution of $f(x)|\, \mathcal{X}, \mathcal{Y}$ is also a Gaussian $\mathcal{N}\left(\mu(x), \sigma^2(x)\right)$,

$$\mu(x) = \mathcal{K}(x, \mathcal{X})\mathcal{K}^{-1}\mathcal{Y}, \quad \sigma^2(x) = \mathcal{K}(x, x) - \mathcal{K}(x, \mathcal{X})\mathcal{K}^{-1}\mathcal{K}(x, \mathcal{X})^T, \tag{S3}$$

and where $\mathcal{K} = \mathcal{K}(\mathcal{X}, \mathcal{X})$. This is the posterior predictive distribution resulting from exact Bayesian inference in an infinitely-wide neural network.

C-NNGP : The above arguments can be extended to convolutional architectures (Novak et al., 2019b). By taking the number of channels in the hidden layers to infinity simultaneously, the outputs of CNNs also converge weakly to a Gaussian process (C-NNGP). The kernel of the C-NNGP takes into account the correlation between pixels in different spatial location and can also be computed exactly via a recursively formula; *e.g.*, see Eq. (7) in (Novak et al., 2019b, §2.2). Note that for convolutional architectures, there are two canonical ways of collapsing image-shaped data into logits. One is to vectorlize the image to a one-dimensional vector (CNN-Vec) and the other is to apply a global average pooling to the spatial dimensions (CNN-GAP). The kernels induced by these two approaches are very different and so are the C-NNGPs. We refer the readers to Section 3.2 of (Novak et al., 2019b, §3) for more details. In this paper, we have focused entirely on vectorization since it is more efficient to compute.

B. Additional Figures for NNGP-C

In this section, we show some additional plots and results comparing NNGP-C against standard NNs. Mainly, we address the method of hyperparameter tuning considered in the main text, where we fixed the hyperparameters that are common to both the NNGP and the NN, then only tuned the additional NN hyperparameters. Here, we show results for tuning all of the NN's hyperparameter from scratch.

Additional Tuning details. We found the performance of NNGP-C to be sensitive to the kernel hyperparameters. To tune these parameters we used the Google Vizier service (Golovin et al., 2017) with a budget of 250 trials and selected the setting with the best log-likelihood on a validation set. We use the same hyperparamters for the NN to make a direct comparison of the prior. The additional hyperparameters required for the NN, like width and training time, were also tuned using Vizier. We also compared against the NN performance when all its hyperparameters are tuned, and found the accuracy of the NN improved but the calibration results were broadly similar.

For any tuning of hyperparameters, we split the original training set of CIFAR10 into a 45K training set and a 5K validation set. All models were trained using the 45K points, and we then selected the hyperparameters from the validation set performance. We introduced a constant that multiples the whole NNGP kernel, or equivalently scales the whole latent space vector or the last layer bias and weight standard deviations—we called this constant the kernel scale. For FC-NNGP-C, the activation function was tuned over $\{\mathrm{ReLU}, \mathrm{erf}\}$, the weight standard deviation was tuned over $[0.1, 2.0]$ on a linear scale, the bias standard deviation was tuned over $[0.1, 0.5]$ on a linear scale, the kernel scale was tuned over $[10^{-2}, 100]$ on a logarithmic scale, the depth was tuned over $\{1, 2, 3, 4, 5\}$, and the diagonal regularizer was tuned over $[0., 0.01]$ on a linear scale.

For the FC-NN, there are additional hyperparameters: the learning rate, training steps, and width. For the NN, we considered two types of tuning. Either, as in the main text, the hyperparameters that are shared with the NNGP are fixed and the additional hyperparameters are tuned, or, as we present in Fig. S1 and Table S2, all of the NN's hyperparameters are tuned from scratch. In either case, the activation, the weight standard deviation, the bias standard deviation, the kernel scale, and the depth were tuned as above. The learning rate was tuned over $[10^{-4}, 0.1]$ on a logarithmic scale, the total training steps was tuned over $[10^5, 10^7]$ on a logarithmic, and the width was tuned over $\{64, 128, 256, 512\}$.

For the C-NNGP-C, all hypermaramters were treated as for the FC-NNGP-C case, except depth which was limited to $\{1, 2\}$. For the CNN, we again considered the two types of tuning: either fixing common hyperparameters or retuning all hyperparameters from scratch. The CNN's learning rate was tuned over $[10^{-4}, 0.1]$ on a logarithmic scale, the total training steps was tuned over $[2^{17}, 2^{22}]$ on a logarthimic scale, and the width was tuned over $\{64, 128, 256, 512\}$.

Table S1. Comparing NNs and against the equivalent NNGP-C on CIFAR-10 and evaluated on several test sets. We observe that the NNGP-C outperforms its parametric NN counterpart on *every* metric. We see particularly significant improvements in ECE and NLL, implying that NNGP-C is considerably better calibrated.

Data	Metric	FC-NN	FC-NNGP-C	CNN	C-NNGP-C
CIFAR10	ECE	0.209	**0.072**	0.283	**0.031**
	Brier Score	0.711	**0.629**	0.685	**0.519**
	Accuracy	0.487	**0.518**	0.576	**0.609**
	NLL	17383	**14007**	21786	**11215**
Fog 1	ECE	0.178	**0.098**	0.271	**0.042**
	Brier Score	0.707	**0.655**	0.685	**0.542**
	Accuracy	0.471	**0.496**	0.561	**0.590**
	NLL	16702	**14671**	19716	**11755**
Fog 2	ECE	0.177	**0.073**	0.286	**0.018**
	Brier Score	0.758	**0.711**	0.747	**0.620**
	Accuracy	0.416	**0.425**	0.507	**0.519**
	NLL	18026	**16233**	20822	**13819**
Fog 3	ECE	0.202	**0.039**	0.320	**0.03**
	Brier Score	0.822	**0.759**	0.836	**0.694**
	Accuracy	0.358	**0.363**	0.445	**0.445**
	NLL	20092	**17638**	23752	**16019**
Fog 4	ECE	0.236	**0.026**	0.352	**0.071**
	Brier Score	0.881	**0.797**	0.914	**0.758**
	Accuracy	0.307	**0.311**	0.385	**0.380**
	NLL	22589	**18867**	27055	**18353**
Fog 5	ECE	0.279	**0.057**	0.420	**0.134**
	Brier Score	0.961	**0.847**	1.052	**0.846**
	Accuracy	0.241	**0.251**	0.287	**0.289**
	NLL	26345	**20694**	33891	**22014**
SVHN	Mean Confidence	0.537	0.335	0.718	0.463
	Entropy	1.230	1.840	0.733	1.403
CIFAR100	Mean Confidence	0.651	0.398	0.812	0.474
	Entropy	0.944	1.663	0.493	1.420

Table S2. Performance of NNs, where all of the NN's hyperparameters are tuned independently with Vizier. The NaN entropy measurement is due to the confidence on a specific test point being 1.0 to machine precision.

Data	Metric	FC-NN (all tuned)	CNN (all tuned)
CIFAR10	ECE	0.217	0.265
	Brier Score	0.721	0.605
	Accuracy	0.478	0.632
	NLL	17967	21160
Fog 1	ECE	0.187	0.258
	Brier Score	0.726	0.618
	Accuracy	0.458	0.617
	NLL	17372	19484
Fog 2	ECE	0.189	0.287
	Brier Score	0.781	0.712
	Accuracy	0.396	0.552
	NLL	18930	21657
Fog 3	ECE	0.215	0.333
	Brier Score	0.846	0.816
	Accuracy	0.336	0.480
	NLL	21247	25703
Fog 4	ECE	0.241	0.381
	Brier Score	0.902	0.924
	Accuracy	0.292	0.409
	NLL	23746	30844
Fog 5	ECE	0.282	0.472
	Brier Score	0.975	1.104
	Accuracy	0.232	0.301
	NLL	27493	41058
SVHN	Mean Confidence	0.542	0.794
	Entropy	1.208	0.524
CIFAR100	Mean Confidence	0.654	0.847
	Entropy	0.930	NaN

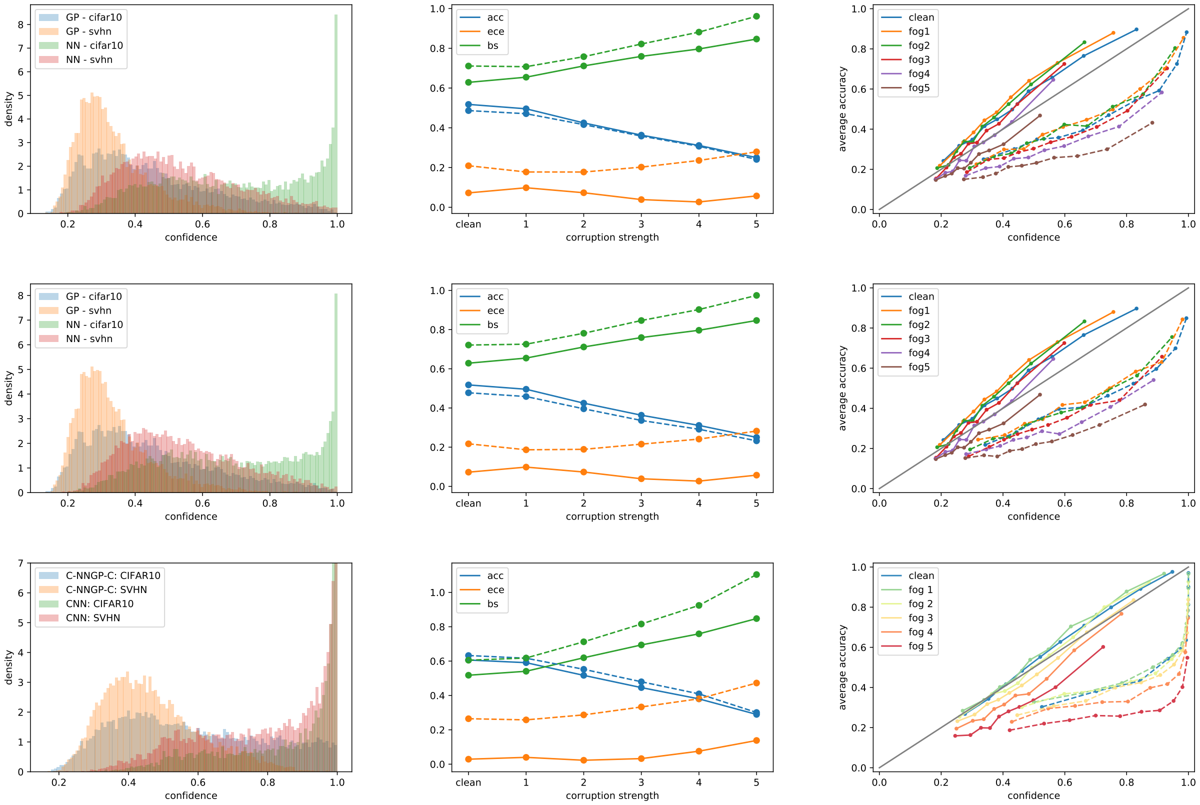

Figure S1. Investigating the calibration of Gaussian process classification with NNGP kernels as in Fig. 1, where we find similar results. (**left column**) Histogram of the confidence of the posterior distribution for each test point. We compare the NNGP-C and a finite width NN on an identically distributed test set (CIFAR10) and an OOD test set (SVHN). (**middle column**) Performance, NNGP-C is solid and NN dashed, under increasing distributional shift given by the CIFAR10 fog corruption. (**right column**) We bin the test set into ten bins sorted by confidence, and we plot mean confidence against mean accuracy. (**top row**) We compare FC-NNGP-C against a FC-NN with the same hyperparameters. (**middle row**) We compare FC-NNGP-C against a FC-NN, where all of the NN's hyperparameters are tuned independently with Vizier. (**bottom row**) We compare C-NNGP-C against a CNN, where all of the CNN's hyperparameters are tuned independently with Vizier.

C. Comparison of Heuristics for Generating Confidences from NNGP-R

In Secs. 3 and D, we utilized a heuristic to generate confidence from exact GPR posterior distribution. Here we denote the heuristic described in Eq. (6) as *exact* confidence, which is the probability of a class probit being maximal under an independent multivariate Gaussian distribution. We consider two more heuristics. One is denoted *pairwise*, where we take confidence to be proportional to the probability that the i-th class probit is larger than other probits in pairwise fashion, *i.e.*

$$p_x(i) \propto \mathbb{P}[z_i > z_j, \forall j \neq i] = \prod_{j\neq i} p(y_i > y_j | x, \mathcal{X}, \mathcal{Y}) = \prod_{j \neq i} \Phi\left(\frac{\mu_i - \mu_j}{\sqrt{\sigma_i^2 + \sigma_j^2}}\right), \tag{S4}$$

where $\Phi(\cdot)$ is Gaussian cumulative distribution function. In order to obtain confidence, we normalize by the sum so that the heuristic confidence sums up to 1. This is following the one-vs-one multiclass classification strategy (Hastie & Tibshirani, 1998).

We note that, we introduce temperature scaling with temperature T by replacing posterior variances as

$$\sigma_T^2 = T\sigma^2 \,. \tag{S5}$$

Another heuristic is denoted *softmax*, where we apply the softmax function to the posterior mean:

$$p_x(i) := \sigma(\mu)_i = \frac{e^{\mu_i/\sqrt{T}}}{\sum_j e^{\mu_j/\sqrt{T}}} \,. \tag{S6}$$

In this case, the posterior variance is not used to construct the heuristic confidences.

A comparison for these three-different heuristics for C-NNGP-R is shown in Fig. S2 with and without temperature scaling. We note that *exact* and *pairwise* heuristics remain well calibrated without temperature scaling. However with temperature scaling the *softmax* heuristic can be competitive to other heuristics. In Sec. 3 and D, we focused on the *exact* heuristic.

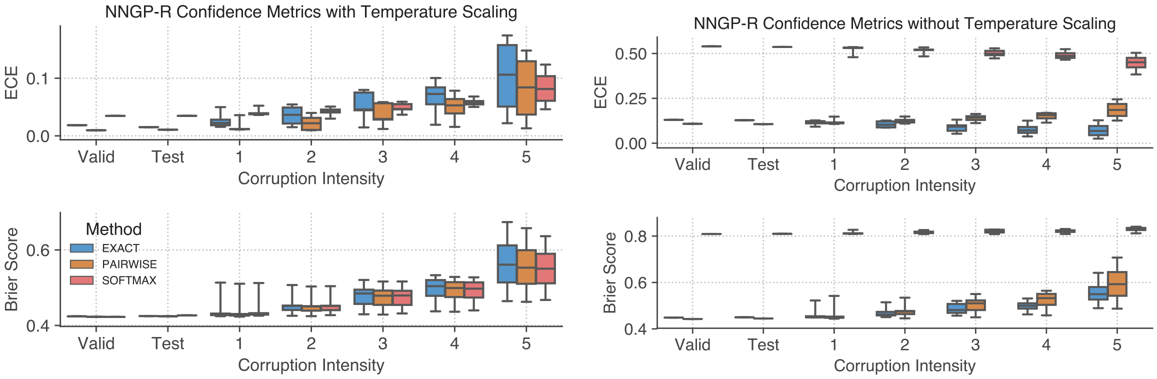

Figure S2. Comparison of NNGP-R based confidence measures on CIFAR10 corruptions using C-NNGP-R. Left column uses temperature scaling based on a validation set whereas right column uses $T = 1$. We see that softmax confidence requires adjusting temperature which is equivalent to modifying prior variance to be calibrated whereas exact and pairwise heuristic confidence is calibrated without needing to modify the prior variance.

D. Bayesian or Infinite-Width Last Layer

As we have seen NNGP-C and NNGP-R are remarkably well-calibrated. However, obtaining high performing models can be computationally intensive, especially for large datasets. NNGP-C and NNGP-R have running times that are cubic in the dataset size, due to computation of the kernel's Cholesky decomposition, with NNGP-C suffering additionally from potentially slow convergence of MCMC. Moreover, performant NNGP kernels require substantial compute to obtain (Novak et al., 2019b; Arora et al., 2019; Novak et al., 2019a; Li et al., 2019) in contrast to training a NN to similar accuracies.

Table S3. Quartiles of Brier score, negative-log-likelihood and ECE over all CIFAR10 corruptions for methods in Figure 2.

Method/Metric	RBF	FC-NNGP-R	C-NNGP-R
Brier Score (25th)	0.568	0.569	**0.435**
Brier Score (50th)	0.580	0.586	**0.464**
Brier Score (75th)	0.599	0.613	**0.515**
Gaussian NLL (25th)	0.147	0.612	**0.108**
Gaussian NLL (50th)	**0.270**	0.830	0.457
Gaussian NLL (75th)	**0.447**	1.099	1.027
NLL (25th)	1.331	1.351	**1.002**
NLL (50th)	1.363	1.398	**1.079**
NLL (75th)	1.415	1.466	**1.178**
ECE (25th)	0.048	0.030	**0.022**
ECE (50th)	0.052	**0.039**	0.046
ECE (75th)	0.069	**0.065**	0.071
Accuracy (75th)	0.573	0.574	**0.683**
Accuracy (50th)	0.566	0.561	**0.659**
Accuracy (25th)	0.549	0.541	**0.628**

Moreover, even though the most performant NNGP kernels are SotA for a kernel method (Shankar et al., 2020), they still under-perform SotA NNs by a large margin.

To combine the benefits of the NNGP and NNs, obtaining models that are both performant and well calibrated, we propose stacking an infinite-width sub-network on top of a pre-trained NN. More precisely, we use features obtained from a pre-trained model as inputs for the NNGP. As such, the outputs of the combined network are drawn from a GP and we may use Eqs. (5) and (6) for inference. We refer to this as Gaussian process last-layer (NNGP-LL). Mathematically, the model is

$$y \sim f(g(x)) + \varepsilon, \text{ where } f \sim \mathcal{GP}(\mathbf{0}, \mathcal{K}), \tag{S7}$$

where g is a pre-trained embedding, $\mathcal{K}$ is the NNGP kernel, and $\varepsilon \sim \mathcal{N}(0, \sigma_\epsilon)$ is an independent noise term. This draws inspiration from (Hinton & Salakhutdinov, 2008; Wilson et al., 2016), but with a *multi-layer* NNGP kernel. Note we are specifically interested in the calibration properties and so the innovations in computational efficiency obtained in (Wilson et al., 2016) are complementary to our work.

This setup mirrors an important use case in practice, e.g. for customers of cloud ML services, who may use embeddings trained on vast amounts of non-domain-specific data, and then fine-tune this model to their specific use case. This fine tuning consists of either fitting a logisitic regression layer or deeper NNs using the embeddings obtained from their data, or perhaps training the whole NN generating the embedding by simply initialzing with the pre-trained weights (Kornblith et al., 2019). These strategies allow practitioners to obtain highly accurate models without substantial data or computation. However, little is understood about the calibration of these transfer learning approaches.

We consider the EfficientNet-B3 (Tan & Le, 2019) embedding from TF-Hub[1] that is trained on ImageNet (Deng et al., 2009), and perform our evaluations on CIFAR10 and its corruptions. For our experiments, we use a multi-layer FC-NNGP as the top sub-network since its kernel is very fast to compute and the final FC layer of EfficientNet-B3 removes any spacial structure that might be exploited by convolutions. We compare our method with other popular last-layer methods for generating uncertainty estimates (Vanilla logisitic regression, using a deep NN for the last layers, temperature scaling Platt et al. (1999); Guo et al. (2017), MC-Dropout (Gal & Ghahramani, 2016), ensembles of several last-layer deep NNs). Below we further investigate these results with a WideResNet (Zagoruyko & Komodakis, 2016) that we can train from scratch, using the initialization method in (Dauphin & Schoenholz, 2019), which achieves good test performance on CIFAR-10 without BatchNorm (Ioffe & Szegedy, 2015). This allows us to compare against the gold standard ensemble method.

We find that in the transfer learning case, ensembles are quite ineffective alone. The best previous method is given by

[1] `https://www.tensorflow.org/hub`

Table S4. NNGP-LL with EfficientNet-B3 embedding fine tuned on CIFAR10 and evaluated over all corruptions and intensities. We show the quartiles for these evaluations for several different last-layer methods of obtaining uncertainties. Ensembles refers to (Lakshminarayanan et al., 2017) and Ens/Drp/T refers to combining Ensembles, MC-Dropout (Gal & Ghahramani, 2016) and temperature scaling (Guo et al., 2017). The rightmost columns show that NNGP-LL can be well-calibrated with very little training data. See the Fig. S3 for a fine-grained box plots for each corruption level.

Method	Vanilla	Ensembles	Ens/Drp/T	100	1K	5K	10K	NNGP-LL
Brier Score (25th)	0.230	0.218	0.182	0.363	0.256	0.218	0.203	**0.173**
Brier Score (50th)	0.351	0.331	**0.265**	0.448	0.346	0.308	0.288	0.271
Brier Score (75th)	0.521	0.511	0.410	0.572	0.478	0.436	0.409	**0.397**
NLL (25th)	0.913	0.823	0.382	0.797	0.562	0.474	0.455	**0.367**
NLL (50th)	1.517	1.411	**0.569**	0.991	0.746	0.682	0.655	0.586
NLL (75th)	2.662	2.492	0.932	1.326	1.075	0.972	0.921	**0.905**
ECE (25th)	0.104	0.098	**0.016**	0.023	0.044	0.018	0.019	0.017
ECE (50th)	0.160	0.154	0.028	0.040	0.062	**0.023**	0.027	0.025
ECE (75th)	0.247	0.243	0.079	0.081	0.104	**0.042**	0.057	0.044
Accuracy (75th)	0.869	0.875	0.875	0.742	0.825	0.851	0.860	**0.884**
Accuracy (50th)	0.802	0.812	**0.813**	0.674	0.758	0.784	0.798	**0.813**
Accuracy (25th)	0.714	0.719	0.718	0.582	0.663	0.689	0.704	**0.722**

combining MC-Dropout, temperature scaling, and ensembles. However, this is still bested by NNGP-LL (see Table S4). We also examine the effect of tuning dataset size on calibration and performance. Remarkably, NNGP-LL is able to achieve accuracies and calibration comparable to ensembles with as few as 1000 training points.

E. Additional Figures for NNGP-LL

The above results focused on a fixed embedding (see Table S4) and we show additional results for this as a box-plot in Fig. S3 here. However, it is also common in practice to tune all of the embedding's weights and simply initialize at their pre-triained values. We explore this setting in Table S5 and Fig. S4 by considering the EfficientNet-B3 embedding and fine tuning it on CIFAR10. We also show a results comparing the FC-NNGP-LL with using the standard RBF kernel for the same purpose (see Fig. S5).

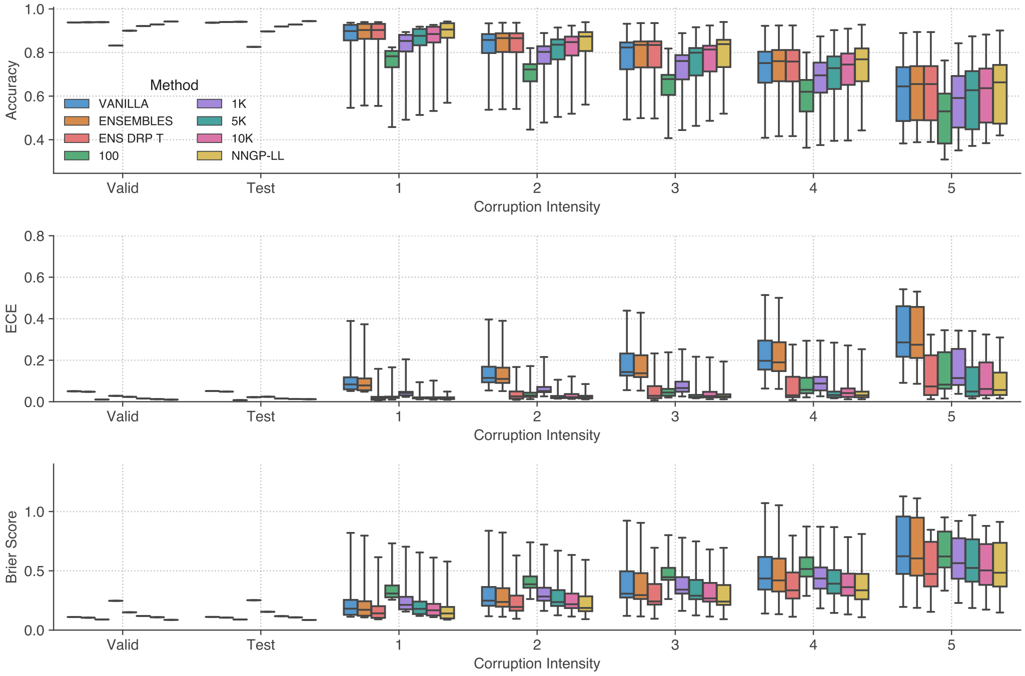

Figure S3. Uncertainty metrics across corruption levels on CIFAR10 using NNGP-LL with EfficientNet-B3 embedding. Baseline NNs are trained on CIFAR10 with parameters of body network fixed. See Table S4 for quartile comparison and Figure S4 and Table S5 for comparison with fine tuning of all the embedding's weights. Figure S5 compares use of NNGP and RBF kernel for NNGP-LL settings.

Table S5. Quartiles of Brier score, negative-log-likelihood, and ECE over all CIFAR10 corruptions for methods in Figure S4 using EfficientNet-B3 embedding. The first five columns are the same as the methods in Table S4, whereas the last two columns use the same embedding architecture but where all the weights are fine tuned (FT) on CIFAR10. See Fig. S4 for the corresponding box-plot.

	LL	LL T Scaling	LL Ens	LL Ens Drp T	NNGP-LL	FT-NN	FT-NNGP-LL
Brier Score (25th)	0.230	0.191	0.218	0.182	0.173	0.083	**0.081**
Brier Score (50th)	0.351	0.281	0.331	0.265	0.271	0.155	**0.153**
Brier Score (75th)	0.521	0.422	0.511	0.410	0.397	**0.341**	0.361
NLL (25th)	0.913	0.406	0.823	0.382	0.367	**0.166**	0.196
NLL (50th)	1.517	0.612	1.411	0.569	0.586	**0.320**	0.374
NLL (75th)	2.662	0.988	2.492	0.932	0.905	**0.730**	0.868
ECE (25th)	0.104	0.016	0.098	0.016	0.017	**0.005**	0.012
ECE (50th)	0.160	0.030	0.154	0.028	0.025	**0.015**	0.022
ECE (75th)	0.247	0.092	0.243	0.079	0.044	0.051	**0.046**
Accuracy (75th)	0.869	0.869	0.875	0.875	0.884	0.945	**0.947**
Accuracy (50th)	0.802	0.802	0.812	0.813	0.813	0.894	**0.896**
Accuracy (25th)	0.714	0.714	0.719	0.718	0.722	**0.758**	0.748

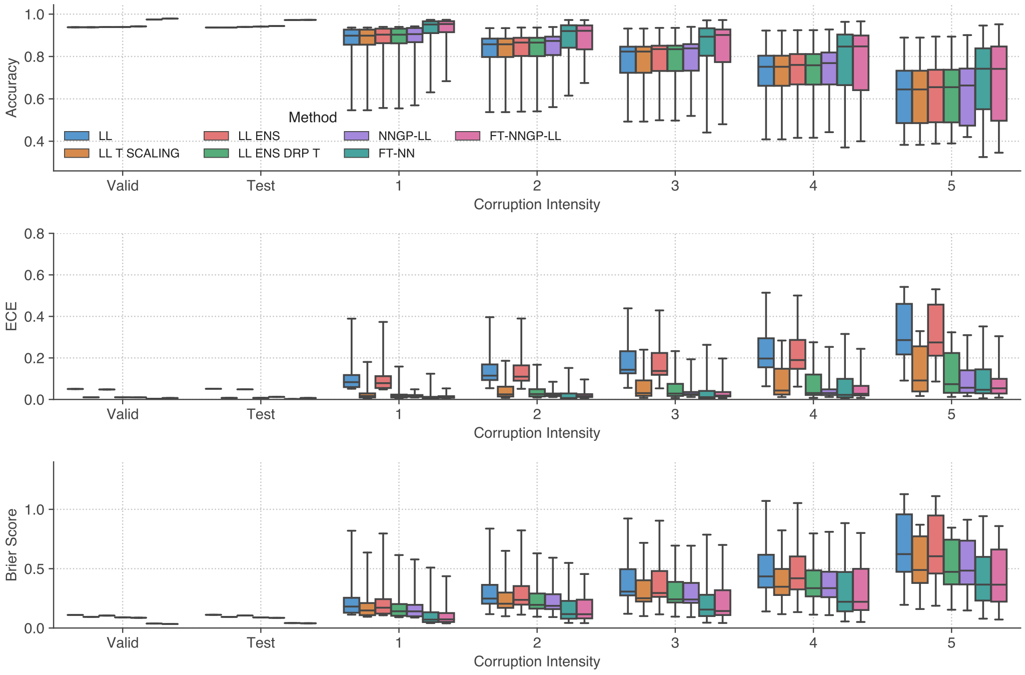

Figure S4. Uncertainty metrics across corruption levels on CIFAR10 using NNGP-LL with EfficientNet-B3 embedding. Baseline NNs are either last layer (LL) trained on CIFAR10 with parameters for body networks fixed or where all the weights are fine tuned (FT). Quartile comparisons can be found in Table S5.

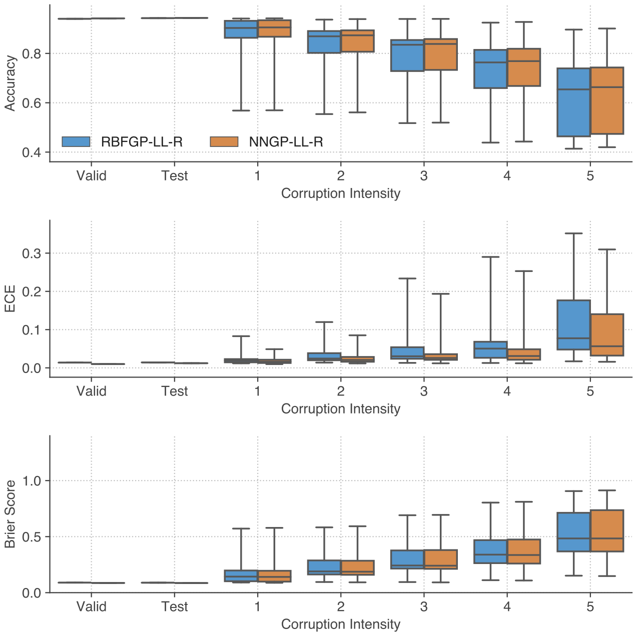

Figure S5. Comparison of NNGP-LL using NNGP head vs RBF GP head on EfficientNet-B3 embedding. While there are slight advantage of using NNGP head over RBF-GP overall they provide very similar benefits as corruption intensity increase.

E.1. A comparison of Ensemble and NNGP-LL on WideResnet

To compare the NNGP-LL method against the gold standard ensemble method, we train a WideResnet 28-10 on CIFAR-10 from scratch with 5 different random initialization. The model is trained using MetaInit (Dauphin & Schoenholz, 2019), Delta-Orthogonal (Xiao et al., 2018), mixup (Zhang et al., 2017) and without BatchNorm (Ioffe & Szegedy, 2015). The model achieves about 94% accuracy on the clean test set. See Table S6 and Fig. S6 for the comparison. We find the ECE for NNGP-LL is very competitive, even with small dataset size, compared to baseline methods including ensembles.

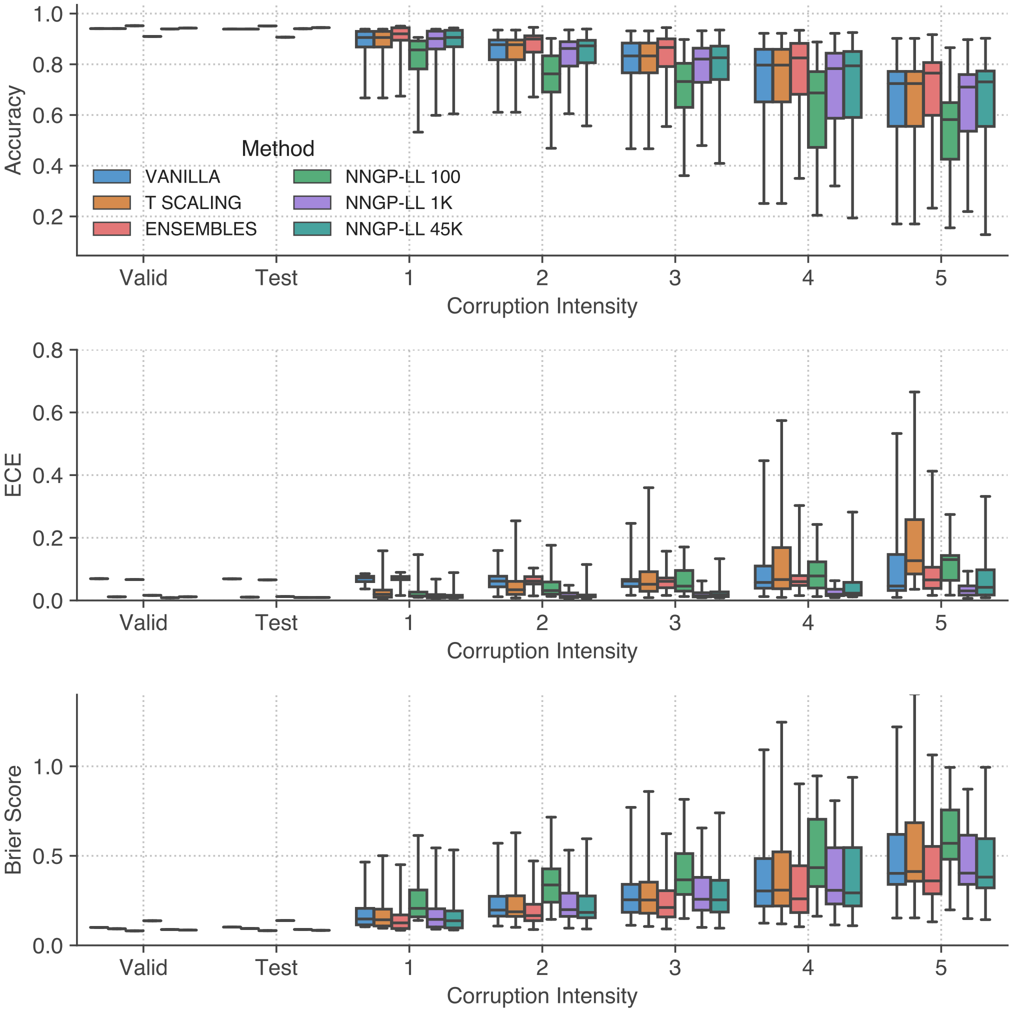

Figure S6. Uncertainty metrics across corruption levels on CIFAR10 using NNGP-LL with MetaInit embedding. Baseline NNs are compared with vanilla training, temperature scaling and ensembles.

Table S6. Quartiles of Brier score, negative-log-likelihood, ECE and accuray over all CIFAR10 corruptions for MetaInit Embedding NNGP-LL and MetaInit trained networks in Figure S6.

Method/Metric	Vanilla	T Scaling	Ensembles	100	1K	NNGP-LL
Brier Score (25th)	0.165	0.164	**0.141**	0.256	0.162	0.152
Brier Score (50th)	0.247	0.244	**0.210**	0.366	0.257	0.241
Brier Score (75th)	0.397	0.410	**0.356**	0.567	0.415	0.399
NLL (25th)	0.382	0.391	**0.328**	0.562	0.381	0.360
NLL (50th)	0.553	0.576	**0.483**	0.799	0.599	0.570
NLL (75th)	0.864	1.040	**0.781**	1.214	0.967	0.925
ECE (25th)	0.046	0.025	0.044	0.018	**0.011**	0.012
ECE (50th)	0.062	0.051	0.066	0.044	**0.017**	**0.017**
ECE (75th)	0.079	0.126	0.077	0.101	**0.030**	0.039
Accuracy (75th)	0.895	0.895	**0.916**	0.824	0.889	0.896
Accuracy (50th)	0.840	0.840	**0.866**	0.738	0.821	0.834
Accuracy (25th)	0.726	0.726	**0.761**	0.586	0.703	0.716