Bayesian Deep Ensembles via the Neural Tangent Kernel

Bobby He
Department of Statistics
University of Oxford
bobby.he@stats.ox.ac.uk

Balaji Lakshminarayanan
Google Research
Brain team
balajiln@google.com

Yee Whye Teh
Department of Statistics
University of Oxford
y.w.teh@stats.ox.ac.uk

Abstract

We explore the link between deep ensembles and Gaussian processes (GPs) through the lens of the Neural Tangent Kernel (NTK): a recent development in understanding the training dynamics of wide neural networks (NNs). Previous work has shown that even in the infinite width limit, when NNs become GPs, there is no GP posterior interpretation to a deep ensemble trained with squared error loss. We introduce a simple modification to standard deep ensembles training, through addition of a computationally-tractable, randomised and untrainable function to each ensemble member, that enables a posterior interpretation in the infinite width limit. When ensembled together, our trained NNs give an approximation to a posterior predictive distribution, and we prove that our Bayesian deep ensembles make more conservative predictions than standard deep ensembles in the infinite width limit. Finally, using finite width NNs we demonstrate that our Bayesian deep ensembles faithfully emulate the analytic posterior predictive when available, and can outperform standard deep ensembles in various out-of-distribution settings, for both regression and classification tasks.

1 Introduction

Consider a training dataset $\mathcal{D}$ consisting of N i.i.d. data points $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, with $\boldsymbol{x} \in \mathbb{R}^d$ representing d-dimensional features and y representing C-dimensional targets. Given input features $\boldsymbol{x}$ and parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ we use the output, $f(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^C$, of a neural network (NN) to model the predictive distribution $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ over the targets. For univariate regression tasks, $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ will be Gaussian: $-\log p(y|\boldsymbol{x}, \boldsymbol{\theta})$ is the squared error $\frac{1}{2\sigma^2}(y - f(\boldsymbol{x}, \boldsymbol{\theta}))^2$ up to additive constant, for fixed observation noise $\sigma^2 \in \mathbb{R}_+$. For classification tasks, $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ will be a Categorical distribution.

Given a prior distribution $p(\boldsymbol{\theta})$ over the parameters, we can define the posterior over $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}|\mathcal{D})$, using Bayes' rule and subsequently the *posterior predictive* distribution at a test point $(\boldsymbol{x}^*, y^*)$:

$$p(y^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(y^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, d\boldsymbol{\theta} \tag{1}$$

The posterior predictive is appealing as it represents a marginalisation over $\boldsymbol{\theta}$ weighted by posterior probabilities, and has been shown to be optimal for minimising predictive risk under a well-specified model [1]. However, one issue with the posterior predictive for NNs is that it is computationally intensive to calculate the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ exactly. Several approximations to $p(\boldsymbol{\theta}|\mathcal{D})$ have been introduced for *Bayesian neural networks* (BNNs) including: Laplace approximation [2]; Markov chain Monte Carlo [3, 4]; variational inference [5–9]; and Monte-Carlo dropout [10].

Despite the recent interest in BNNs, it has been shown empirically that deep ensembles [11], which lack a principled Bayesian justification, outperform existing BNNs in terms of uncertainty quantification and out-of-distribution robustness, cf. [12]. Deep ensembles independently initialise and train individual NNs (referred to herein as *baselearners*) on the negative log-likelihood loss

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta}))$ with $\ell(y, f(\boldsymbol{x}, \boldsymbol{\theta})) = -\log p(y|\boldsymbol{x}, \boldsymbol{\theta})$, before aggregating predictions. Understanding the success of deep ensembles, particularly in relation to Bayesian inference, is a key question in the uncertainty quantification and Bayesian deep learning communities at present: Fort et al. [13] suggested that the empirical performance of deep ensembles is explained by their ability to explore different functional modes, while Wilson and Izmailov [14] argued that deep ensembles are actually approximating the posterior predictive.

In this work, we will relate deep ensembles to Bayesian inference, using recent developments connecting GPs and wide NNs, both before [15–21] and after [22, 23] training. Using these insights, we devise a modification to standard NN training that yields an exact posterior sample for $f(\cdot, \boldsymbol{\theta})$ in the infinite width limit. As a result, when ensembled together our modified baselearners give a posterior predictive approximation, and can thus be viewed as a *Bayesian deep ensemble*.

One concept that is related to our methods concerns ensembles trained with *Randomised Priors* to give an approximate posterior interpretation, which we will use when modelling observation noise in regression tasks. The idea behind randomised priors is that, under certain conditions, regularising baselearner NNs towards independently drawn "priors" during training produces exact posterior samples for $f(\cdot, \boldsymbol{\theta})$. Randomised priors recently appeared in machine learning applied to reinforcement learning [24] and uncertainty quantification [25, 26], like this work. To the best of our knowledge, related ideas first appeared in astrophysics where they were applied to Gaussian random fields [27]. However, one such condition for posterior exactness with randomised priors is that the model $f(\boldsymbol{x}, \boldsymbol{\theta})$ is linear in $\boldsymbol{\theta}$. This is not true in general for NNs, but has been shown to hold for wide NNs local to their parameter initialisation, in a recent line of work. In order to introduce our methods, we will first review this line of work, known as the *Neural Tangent Kernel* (NTK) [22].

2 NTK Background

Wide NNs, and their relation to GPs, have been a fruitful area recently for the theoretical study of NNs: we review only the most salient developments to this work, due to limited space.

First introduced by Jacot et al. [22], the *empirical NTK* of $f(\cdot, \boldsymbol{\theta}_t)$ is, for inputs $\boldsymbol{x}, \boldsymbol{x}'$, the kernel:

$$\hat{\Theta}_t(\boldsymbol{x}, \boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}', \boldsymbol{\theta}_t) \rangle \tag{2}$$

and describes the functional gradient of a NN in terms of the current loss incurred on the training set. Note that $\boldsymbol{\theta}_t$ depends on a random initialisation $\boldsymbol{\theta}_0$, thus the empirical NTK is random for all $t > 0$.

Jacot et al. [22] showed that for an MLP under a so-called NTK parameterisation, detailed in Appendix A, the empirical NTK converges in probability to a deterministic limit Θ, that stays constant during gradient training, as the hidden layer widths of the NN go to infinity sequentially. Later, Yang [28, 29] extended the NTK convergence result to convergence almost surely, which is proven rigorously for a variety of architectures and for widths (or channels in Convolutional NNs) of hidden layers going to infinity in unison. This limiting positive-definite (p.d.) kernel Θ, known as the NTK, depends only on certain NN architecture choices, including: activation, depth and variances for weight and bias parameters. Note that the NTK parameterisation can be thought of as akin to training under standard parameterisation with a learning rate that is inversely proportional to the width of the NN, which has been shown to be the largest scale for stable learning rates in wide NNs [30–32].

Lee et al. [23] built on the results of Jacot et al. [22], and studied the *linearised regime* of an NN. Specifically, if we denote as $f_t(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{\theta}_t)$ the network function at time t, we can define the first order Taylor expansion of the network function around randomly initialised parameters $\boldsymbol{\theta}_0$ to be:

$$f_t^{\text{lin}}(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \Delta\boldsymbol{\theta}_t \tag{3}$$

where $\Delta\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$ and $f_0 = f(\cdot, \boldsymbol{\theta}_0)$ is the randomly initialised NN function.

For notational clarity, whenever we evaluate a function at an arbitrary input set $\mathcal{X}'$ instead of a single point $\boldsymbol{x}'$, we suppose the function is vectorised. For example, $f_t(\mathcal{X}) \in \mathbb{R}^{NC}$ denotes the concatenated NN outputs on training set $\mathcal{X}$, whereas $\nabla_{\boldsymbol{\theta}} f_t(\mathcal{X}) = \nabla_{\boldsymbol{\theta}} f(\mathcal{X}, \boldsymbol{\theta}_t) \in \mathbb{R}^{NC \times p}$. In the interest of space, we will also sometimes use subscripts to signify kernel inputs, so for instance $\Theta_{\boldsymbol{x}'\mathcal{X}} = \Theta(\boldsymbol{x}', \mathcal{X}) \in \mathbb{R}^{C \times NC}$ and $\Theta_{\mathcal{X}\mathcal{X}} = \Theta(\mathcal{X}, \mathcal{X}) \in \mathbb{R}^{NC \times NC}$ throughout this work.

The results of Lee et al. [23] showed that in the infinite width limit, with NTK parameterisation and gradient flow under squared error loss, $f_t^{\text{lin}}(\boldsymbol{x})$ and $f_t(\boldsymbol{x})$ are equal for any $t \geq 0$, for a shared random initialisation $\boldsymbol{\theta}_0$. In particular, for the linearised network it can be shown, that as $t \to \infty$:

$$f^{\text{lin}}_{\infty}(\boldsymbol{x}) = f_0(\boldsymbol{x}) - \hat{\Theta}_0(\boldsymbol{x}, \mathcal{X})\hat{\Theta}_0(\mathcal{X}, \mathcal{X})^{-1}(f_0(\mathcal{X}) - \mathcal{Y}) \tag{4}$$

and thus as the hidden layer widths converge to infinity we have that:

$$f^{\text{lin}}_{\infty}(\boldsymbol{x}) = f_{\infty}(\boldsymbol{x}) = f_0(\boldsymbol{x}) - \Theta(\boldsymbol{x}, \mathcal{X})\Theta(\mathcal{X}, \mathcal{X})^{-1}(f_0(\mathcal{X}) - \mathcal{Y}) \tag{5}$$

We can replace $\Theta(\mathcal{X}, \mathcal{X})^{-1}$ with the generalised inverse when invertibility is an issue. However, this will not be a main concern of this work, as our methods will add regularisation that corresponds to modelling observation/output noise, which both ensures invertibility and alleviates any potential convergence issues due to fast decay of the NTK eigenspectrum [33].

From Eq. (5) we see that, conditional on the training data $\{\mathcal{X}, \mathcal{Y}\}$, we can decompose f_{∞} into $f_{\infty}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \gamma(\boldsymbol{x})$ where $\mu(\boldsymbol{x}) = \Theta(\boldsymbol{x}, \mathcal{X})\Theta(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y}$ is a deterministic mean and $\gamma(\boldsymbol{x}) = f_0(\boldsymbol{x}) - \Theta(\boldsymbol{x}, \mathcal{X})\Theta(\mathcal{X}, \mathcal{X})^{-1} f_0(\mathcal{X})$ captures predictive uncertainty, due to the randomness of f_0. Now, if we suppose that, at initialisation, $f_0 \overset{d}{\sim} \mathcal{GP}(0, k)$ for an arbitrary kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{C \times C}$, then we have $f_{\infty}(\cdot) \overset{d}{\sim} \mathcal{GP}(\mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}, \boldsymbol{x}'))$ for two inputs $\boldsymbol{x}, \boldsymbol{x}'$, where:[1]

$$\Sigma(\boldsymbol{x}, \boldsymbol{x}') = k_{\boldsymbol{x}\boldsymbol{x}'} + \Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}k_{\mathcal{X}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\Theta_{\mathcal{X}\boldsymbol{x}} - \left(\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}k_{\mathcal{X}\boldsymbol{x}'} + h.c.\right) \tag{6}$$

For a generic kernel k, Lee et al. [23] observed that this limiting distribution for f_{∞} does not have a posterior GP interpretation unless k and Θ are multiples of each other, .

As mentioned in Section 1, previous work [15–21] has shown that there is a distinct but closely related kernel $\mathcal{K}$, known as the *Neural Network Gaussian Process* (NNGP) kernel, such that $f_0 \overset{d}{\to} \mathcal{GP}(0, \mathcal{K})$ at initialisation in the infinite width limit and $\mathcal{K} \neq \Theta$. Thus Eq. (6) with $k = \mathcal{K}$ tells us that, for wide NNs under squared error loss, there is no Bayesian posterior interpretation to a trained NN, nor is there an interpretation to a trained deep ensemble as a Bayesian posterior predictive approximation.

3 Proposed modification to obtain posterior samples in infinite width limit

Lee et al. [23] noted that one way to obtain a posterior interpretation to f_{∞} is by randomly initialising f_0 but only training the parameters in the final linear readout layer, as the contribution to the NTK Θ from the parameters in final hidden layer is exactly the NNGP kernel $\mathcal{K}$.[2] f_{∞} is then a sample from the GP posterior with prior kernel NNGP, $\mathcal{K}$, and noiseless observations in the infinite width limit i.e. $f_{\infty}(\mathcal{X}') \overset{d}{\sim} \mathcal{N}(\mathcal{K}_{\mathcal{X}'\mathcal{X}}\mathcal{K}^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{Y},\ \mathcal{K}_{\mathcal{X}'\mathcal{X}'} - \mathcal{K}_{\mathcal{X}'\mathcal{X}}\mathcal{K}^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{K}_{\mathcal{X}\mathcal{X}'})$. This is an example of the "sample-then-optimise" procedure of Matthews et al. [34], but, by only training the final layer this procedure limits the earlier layers of an NN solely to be random feature extractors.

We now introduce our modification to standard training that trains all layers of a finite width NN and obtains an exact posterior interpretation in the infinite width limit with NTK parameterisation and squared error loss. For notational purposes, let us suppose $\boldsymbol{\theta} = \texttt{concat}(\{\boldsymbol{\theta}^{\leq L}, \boldsymbol{\theta}^{L+1}\})$ with $\boldsymbol{\theta}^{\leq L} \in \mathbb{R}^{p - p_{L+1}}$ denoting L hidden layers, and $\boldsymbol{\theta}^{L+1} \in \mathbb{R}^{p_{L+1}}$ denoting final readout layer $L+1$. Moreover, define $\Theta^{\leq L} = \Theta - \mathcal{K}$ to be the p.d. kernel corresponding to contributions to the NTK from all parameters before the final layer, and $\hat{\Theta}^{\leq L}_t$ to be the empirical counterpart depending on $\boldsymbol{\theta}_t$. To motivate our modification, we reinterpret f^{lin}_t in Eq. (3) by splitting terms related to $\mathcal{K}$ and $\Theta^{\leq L}$:

$$f^{\text{lin}}_t(\boldsymbol{x}) = \underbrace{f_0(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}^{L+1}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\Delta\boldsymbol{\theta}^{L+1}_t}_{\mathcal{K}} + \underbrace{\mathbf{0}_C + \nabla_{\boldsymbol{\theta}^{\leq L}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\Delta\boldsymbol{\theta}^{\leq L}_t}_{\Theta - \mathcal{K}} \tag{7}$$

where $\mathbf{0}_C \in \mathbb{R}^C$ is the zero vector. As seen in Eq. (7), the distribution of $f^{\text{lin}}_0(\boldsymbol{x}) = f_0(\boldsymbol{x})$ lacks extra variance, $\Theta^{\leq L}(\boldsymbol{x}, \boldsymbol{x})$, that accounts for contributions to the NTK Θ from all parameters $\boldsymbol{\theta}^{\leq L}$ before the final layer. This is precisely why no Bayesian intepretation exists for a standard trained wide NN, as in Eq. (6) with $k = \mathcal{K}$. The motivation behind our modification is now very simple: we propose to manually add in this missing variance. Our modified NNs, $\tilde{f}(\cdot, \boldsymbol{\theta})$, will then have trained distribution:

$$\tilde{f}_{\infty}(\mathcal{X}') \overset{d}{\sim} \mathcal{N}(\Theta_{\mathcal{X}'\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{Y},\ \Theta_{\mathcal{X}'\mathcal{X}'} - \Theta_{\mathcal{X}'\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}'}) \tag{8}$$

[1] Throughout this work, the notation "$+h.c.$" means "plus the Hermitian conjugate", like Lee et al. [23]. For example: $\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}k_{\mathcal{X}\boldsymbol{x}'} + h.c. = \Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}k_{\mathcal{X}\boldsymbol{x}'} + \Theta_{\boldsymbol{x}'\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}k_{\mathcal{X}\boldsymbol{x}}$

[2] Up to a multiple of last layer width in standard parameterisation.

on a test set $\mathcal{X}'$, in the infinite width limit. Note that Eq. (8) is the GP posterior using prior kernel Θ and noiseless observations $\tilde{f}_\infty(\mathcal{X})=\mathcal{Y}$, which we will refer to as the *NTKGP* posterior predictive. We construct $\tilde{f}$ by sampling a random and untrainable function $\delta(\cdot)$ that is added to the standard forward pass $f(\cdot, \boldsymbol{\theta}_t)$, defining an augmented forward pass:

$$\tilde{f}(\cdot, \boldsymbol{\theta}_t) = f(\cdot, \boldsymbol{\theta}_t) + \delta(\cdot) \tag{9}$$

Given a parameter initialisation scheme $\texttt{init}(\cdot)$ and initial parameters $\boldsymbol{\theta}_0 \overset{d}{\sim} \texttt{init}(\cdot)$, our chosen formulation for $\delta(\cdot)$ is as follows: 1) sample $\tilde{\boldsymbol{\theta}} \overset{d}{\sim} \texttt{init}(\cdot)$ independently of $\boldsymbol{\theta}_0$; 2) denote $\tilde{\boldsymbol{\theta}} = \texttt{concat}(\{\tilde{\boldsymbol{\theta}}^{\leq L}, \tilde{\boldsymbol{\theta}}^{L+1}\})$; and 3) define $\boldsymbol{\theta}^* = \texttt{concat}(\{\tilde{\boldsymbol{\theta}}^{\leq L}, \mathbf{0}_{p_{L+1}}\})$. In words, we set the parameters in the final layer of an independently sampled $\tilde{\boldsymbol{\theta}}$ to zero to obtain $\boldsymbol{\theta}^*$. Now, we define:

$$\delta(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\boldsymbol{\theta}^* \tag{10}$$

There are a few important details to note about $\delta(\cdot)$ as defined in Eq. (10). First, $\delta(\cdot)$ has the same distribution in both NTK and standard parameterisations,[3] and also $\delta(\cdot) \mid \boldsymbol{\theta}_0 \overset{d}{\sim} \mathcal{GP}\big(0, \hat{\Theta}_0^{\leq L}\big)$ in the NTK parameterisation.[4] Moreover, Eq. (10) can be viewed as a single Jacobian-vector product (JVP), which packages that offer forward-mode autodifferentiation (AD), such as JAX [35], are efficient at computing for finite NNs. It is worth noting that our modification adds only negligible computational and memory requirements on top of standard deep ensembles [11]: a more nuanced comparison can be found in Appendix G. Alternative constructions of $\tilde{f}$ are presented in Appendix C.

To ascertain whether a trained $\tilde{f}_\infty$ constructed via Eqs. (9, 10) returns a sample from the GP posterior Eq. (8) for wide NNs, the following proposition, which we prove in Appendix B.1, will be useful:

Proposition 1. $\delta(\cdot) \xrightarrow{d} \mathcal{GP}(0, \Theta^{\leq L})$ *and is independent of* $f_0(\cdot)$ *in the infinite width limit. Thus,* $\tilde{f}_0(\cdot) = f_0(\cdot) + \delta(\cdot) \xrightarrow{d} \mathcal{GP}(0, \Theta)$.

Using Proposition 1, we now consider the linearisation of $\tilde{f}_t(\cdot)$, noting that $\nabla_{\boldsymbol{\theta}} \tilde{f}_0(\cdot) = \nabla_{\boldsymbol{\theta}} f_0(\cdot)$:

$$\tilde{f}_t^{\text{lin}}(\boldsymbol{x}) = \underbrace{f_0(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}^{L+1}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\Delta\boldsymbol{\theta}_t^{L+1}}_{\mathcal{K}} + \underbrace{\delta(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}^{\leq L}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\Delta\boldsymbol{\theta}_t^{\leq L}}_{\Theta - \mathcal{K}} \tag{11}$$

The fact that $\nabla_{\boldsymbol{\theta}} \tilde{f}_t^{\text{lin}}(\cdot) = \nabla_{\boldsymbol{\theta}} f_0(\cdot)$ is crucial in Eq. (11), as this initial Jacobian is the feature map of the linearised NN regime from Lee et al. [23]. As per Proposition 1 and Eq. (11), we see that $\delta(\boldsymbol{x})$ adds the extra randomness missing from $f_0^{\text{lin}}(\boldsymbol{x})$ in Eq. (7), and reinitialises $\tilde{f}_0$ as a sample from $\mathcal{GP}(0, \mathcal{K})$ to $\mathcal{GP}(0, \Theta)$ for wide NNs. This means we can set $k = \Theta$ in Eq. (6) and deduce:

Corollary 1. $\tilde{f}_\infty(\mathcal{X}') \overset{d}{\sim} \mathcal{N}(\Theta_{\mathcal{X}'\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}\mathcal{Y},\ \Theta_{\mathcal{X}'\mathcal{X}'} - \Theta_{\mathcal{X}'\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}\Theta_{\mathcal{X}\mathcal{X}'})$, *and hence a trained* $\tilde{f}_\infty$ *returns a sample from the posterior NTKGP in the infinite width limit.*

To summarise: we define our new NN forward pass to give $\tilde{f}_t(\boldsymbol{x}) = f_t(\boldsymbol{x}) + \delta(\boldsymbol{x})$ for standard forward pass $f_t(\boldsymbol{x})$, and an untrainable $\delta(\boldsymbol{x})$ defined as in Eq. (10). As given by Corollary 1, independently trained baselearners $\tilde{f}_\infty$ can then be ensembled to approximate the NTKGP posterior predictive. We will call $\tilde{f}_\infty$ trained in this section an NTKGP baselearner, regardless of parameterisation or width. We are aware that the name NTK-GP has been used previously to refer to Eq. (6) with NNGP kernel $\mathcal{K}$, which is what standard training under squared error with a wide NN yields. However, we believe GPs in machine learning are synonymous with probabilistic inference [36], which Eq. (6) has no connection to in general, so we feel the name NTKGP is more appropriate for our methods.

[3] In this work Θ always denotes the NTK under NTK parameterisation. It is also possible to model Θ to be the scaled NTK under standard parameterisation (which depends on layer widths) as in Sohl-Dickstein et al. [32] with minor reweightings to both $\delta(\cdot)$ and, when modelling observation noise, the L^2-regularisation described in Appendix D.

[4] With NTK parameterisation, it is easy to see that $\delta(\cdot) \mid \boldsymbol{\theta}_0 \overset{d}{\sim} \mathcal{GP}\big(0, \hat{\Theta}_0^{\leq L}\big)$, because $\tilde{\boldsymbol{\theta}}^{\leq L} \overset{d}{\sim} \mathcal{N}(0, I_{p - p_{L+1}})$. To extend this to standard parameterisation, note that Eq. (10) is just the first order term in the Taylor expansion of $f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \boldsymbol{\theta}^*)$, which has a parameterisation agnostic distribution, about $\boldsymbol{\theta}_0$.

3.1 Modelling observation noise

So far, we have used squared loss $\ell(y, \tilde{f}(\boldsymbol{x}, \boldsymbol{\theta})) = \frac{1}{2\sigma^2}(y - \tilde{f}(\boldsymbol{x}, \boldsymbol{\theta}))^2$ for $\sigma^2{=}1$, and seen how our NTKGP training scheme for $\tilde{f}$ gives a Bayesian interpretation to trained networks when we assume noiseless observations. Lemma 3 of Osband et al. [24] shows us how to draw a posterior sample for linear $\tilde{f}$ if we wish to model Gaussian observation noise $y \overset{d}{\sim} \mathcal{N}(\tilde{f}(\boldsymbol{x}, \boldsymbol{\theta}), \sigma^2)$ for $\sigma^2{>}0$: by adding i.i.d. noise to targets $y'_n \overset{d}{\sim} \mathcal{N}(y_n, \sigma^2)$ and regularising $\mathcal{L}(\boldsymbol{\theta})$ with a weighted L^2 term, either $\|\boldsymbol{\theta}\|^2_\Lambda$ or $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2_\Lambda$, depending on if you regularise in function space or parameter space. The weighting Λ is detailed in Appendix D. These methods were introduced by Osband et al. [24] for the application of Q-learning in deep reinforcement learning, and are known as *Randomised Prior parameter* (RP-param) and *Randomised Prior function* (RP-fn) respectively. The randomised prior (RP) methods were motivated by a Bayesian linear regression approximation of the NN, but they do not take into account the difference between the NNGP and the NTK. Our NTKGP methods can be viewed as a way to fix this for both the parameter space or function space methods, which we will name NTKGP-param and NTKGP-fn respectively. Similar regularisation ideas were explored in connection to the NTK by Hu et al. [37], when the NN function is initialised from the origin, akin to kernel ridge regression.

3.2 Comparison of predictive distributions in infinite width

Having introduced the different ensemble training methods considered in this paper: NNGP; deep ensembles; randomised prior; and NTKGP, we will now compare their predictive distributions in the infinite width limit with squared error loss. Table 1 displays these limiting distributions, $f_\infty(\cdot) \overset{d}{\sim} \mathcal{GP}(\mu, \Sigma)$, and should be viewed as an extension to Equation (16) of Lee et al. [23].

Table 1: Predictive distributions of wide ensembles for various training methods. *std* denotes standard training with $f(\boldsymbol{x}, \boldsymbol{\theta})$, and *ours* denotes training using our additive $\delta(\boldsymbol{x})$ to make $\tilde{f}(\boldsymbol{x}, \boldsymbol{\theta})$.

Method	Layers trained	Output Noise	$\mu(\boldsymbol{x})$	$\Sigma(\boldsymbol{x}, \boldsymbol{x}')$
NNGP	Final	$\sigma^2 \geq 0$	$\mathcal{K}_{\boldsymbol{x}\mathcal{X}}(\mathcal{K}_{\mathcal{X}\mathcal{X}} + \sigma^2 I)^{-1}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - \mathcal{K}_{\boldsymbol{x}\mathcal{X}}(\mathcal{K}_{\mathcal{X}\mathcal{X}} + \sigma^2 I)^{-1}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'}$
Deep Ensembles	All (*std*)	$\sigma^2 = 0$	$\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - (\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'} + h.c.)$ $\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{K}_{\mathcal{X}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\Theta_{\mathcal{X}\boldsymbol{x}'}$
Randomised Prior	All (*std*)	$\sigma^2 > 0$	$\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}} + \sigma^2 I)^{-1}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - (\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}} + \sigma^2 I)^{-1}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'} + h.c.)$ $+\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}} + \sigma^2 I)^{-1}(\mathcal{K}_{\mathcal{X}\mathcal{X}} + \sigma^2 I)(\Theta_{\mathcal{X}\mathcal{X}} + \sigma^2 I)^{-1}\Theta_{\mathcal{X}\boldsymbol{x}'}$
NTKGP	All (*ours*)	$\sigma^2 \geq 0$	$\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}} + \sigma^2 I)^{-1}\mathcal{Y}$	$\Theta_{\boldsymbol{x}\boldsymbol{x}'} - \Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}} + \sigma^2 I)^{-1}\Theta_{\mathcal{X}\boldsymbol{x}'}$

In order to parse Table 1, let us denote $\mu_{\text{NNGP}}, \mu_{\text{DE}}, \mu_{\text{RP}}, \mu_{\text{NTKGP}}$ and $\Sigma_{\text{NNGP}}, \Sigma_{\text{DE}}, \Sigma_{\text{RP}}, \Sigma_{\text{NTKGP}}$ to be the entries in the $\mu(\boldsymbol{x})$ and $\Sigma(\boldsymbol{x}, \boldsymbol{x}')$ columns of Table 1 respectively, read from top to bottom. We see that $\mu_{\text{DE}}(\boldsymbol{x}) = \mu_{\text{NTKGP}}(\boldsymbol{x})$ if $\sigma^2{=}0$, and $\mu_{\text{RP}}(\boldsymbol{x}) = \mu_{\text{NTKGP}}(\boldsymbol{x})$ if $\sigma^2{>}0$. In words: the predictive mean of a trained ensemble is the same when training all layers, both with standard training and our NTKGP training. This holds because both f_0 and $\tilde{f}_0$ are zero mean. It is also possible to compare the predictive covariances as the following proposition, proven in Appendix B.2, shows:

Proposition 2. *For $\sigma^2{=}0$, $\Sigma_{NTKGP} \succeq \Sigma_{DE} \succeq \Sigma_{NNGP}$. Similarly, for $\sigma^2{>}0$, $\Sigma_{NTKGP} \succeq \Sigma_{RP} \succeq \Sigma_{NNGP}$.*

Here, when we write $k_1 \succeq k_2$ for p.d. kernels k_1, k_2, we mean that $k_1 - k_2$ is also a p.d. kernel. One consequence of Proposition 2 is that the predictive distribution of an ensemble of NNs trained via our NTKGP methods is always more conservative than a standard deep ensemble, in the linearised NN regime, when the ensemble size $K{\to}\infty$. It is not possible to say in general when this will be beneficial, because in practice our models will always be misspecified. However, Proposition 2 suggests that in situations where we suspect standard deep ensembles might be overconfident, such as in situations where we expect some dataset shift at test time, our methods should hold an advantage.

3.3 Modelling heteroscedasticity

Following Lakshminarayanan et al. [11], if we wish to model heteroscedasticity in a univariate regression setting such that each training point, $(\boldsymbol{x}_n, y_n)$, has an individual observation noise $\sigma^2(\boldsymbol{x}_n)$

then we use the heteroscedastic Gaussian NLL loss (up to additive constant):

$$\ell(y_n', \tilde{f}(\boldsymbol{x}_n, \boldsymbol{\theta})) = \frac{(y_n' - \tilde{f}(\boldsymbol{x}_n, \boldsymbol{\theta}))^2}{2\sigma^2(\boldsymbol{x}_n)} + \frac{\log \sigma^2(\boldsymbol{x}_n)}{2} \tag{12}$$

where $y_n' = y_n + \sigma(\boldsymbol{x}_n)\epsilon_n$ and $\epsilon_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$. It is easy to see that for fixed $\sigma^2(\boldsymbol{x}_n)$, our NTKGP trained baselearners will still have a Bayesian interpretation: $\mathcal{Y}' \leftarrow \Sigma^{-\frac{1}{2}}\mathcal{Y}'$ and $\tilde{f}(\mathcal{X}, \boldsymbol{\theta}) \leftarrow \Sigma^{-\frac{1}{2}}\tilde{f}(\mathcal{X}, \boldsymbol{\theta})$ returns us to the homoscedastic case, where $\Sigma = \text{diag}(\sigma^2(\mathcal{X})) \in \mathbb{R}^{N \times N}$. We will follow Lakshminarayanan et al. [11] and parameterise $\sigma^2(\boldsymbol{x}) = \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})$ by an extra output head of the NN, that is trainable alongside the mean function $\mu_{\boldsymbol{\theta}}(\boldsymbol{x})$ when modelling heteroscedasticity.[5]

3.4 NTKGP Ensemble Algorithms

We now proceed to train an ensemble of K NTKGP baselearners. Like previous work [11, 24], we independently initialise baselearners, and also use a fixed, independently sampled training set noise $\boldsymbol{\epsilon}_k \in \mathbb{R}^{NC}$ if modelling output noise. These implementation details are all designed to encourage diversity among baselearners, with the goal of approximating the NTKGP posterior predictive for our Bayesian deep ensembles. Appendix F details how to aggregate predictions from trained baselearners. In Algorithm 1, we outline our NTKGP-param method: `data_noise` adds observation noise to targets; `concat` denotes a concatenation operation; and `init`$(\cdot)$ will be standard parameterisation initialisation in the JAX library Neural Tangents [38] unless stated otherwise. As discussed by Pearce et al. [25], there is a choice between "anchoring"/regularising parameters towards their initialisation or an independently sampled parameter set when modelling observation noise. We anchor at initialisation as the linearised NN regime only holds local to parameter initialisation [23], and also this reduces the memory cost of sampling parameters sets. Appendix E details our NTKGP-fn method.

Algorithm 1 NTKGP-param ensemble

```
Require: Data $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, loss function $\mathcal{L}$, NN model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, Ensemble size $K \in \mathbb{N}$, noise
  procedure: data_noise, NN parameter initialisation scheme: init$(\cdot)$
  for $k = 1, \dots, K$ do
    Form $\{\mathcal{X}_k, \mathcal{Y}_k\} =$ data_noise$(\mathcal{D})$
    Initialise $\boldsymbol{\theta}_k \overset{d}{\sim}$ init$(\cdot)$
    Initialise $\tilde{\boldsymbol{\theta}}_k \overset{d}{\sim}$ init$(\cdot)$ and denote $\tilde{\boldsymbol{\theta}}_k =$ concat$(\{\tilde{\boldsymbol{\theta}}_k^{\leq L}, \tilde{\boldsymbol{\theta}}_k^{L+1}\})$
    Set $\boldsymbol{\theta}_k^* =$ concat$(\{\tilde{\boldsymbol{\theta}}_k^{\leq L}, \mathbf{0}_{p_{L+1}}\})$
    Define $\delta(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_k)\boldsymbol{\theta}_k^*$
    Define $\tilde{f}_k(\boldsymbol{x}, \boldsymbol{\theta}_t) = f(\boldsymbol{x}, \boldsymbol{\theta}_t) + \delta(\boldsymbol{x})$ and set $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_k$
    Optimise $\mathcal{L}(\tilde{f}_k(\mathcal{X}_k, \boldsymbol{\theta}_t), \mathcal{Y}_k) + \frac{1}{2}\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_k\|_\Lambda^2$ for $\boldsymbol{\theta}_t$ to obtain $\hat{\boldsymbol{\theta}}_k$
  end for
  return ensemble $\{\tilde{f}_k(\cdot, \hat{\boldsymbol{\theta}}_k)\}_{k=1}^K$
```

3.5 Classification methodology

For classification, we follow recent works [23, 39, 40] which treat classification as a regression task with one-hot regression targets. In order to obtain probabilistic predictions, we temperature scale our trained ensemble predictions with cross-entropy loss on a held-out validation set, noting that Fong and Holmes [41] established a connection between marginal likelihood maximisation and cross-validation.

Because $\delta(\cdot)$ is untrainable in our NTKGP methods, it is important to match the scale of the NTK Θ to the scale of the one-hot targets in multi-class classification settings. One can do this either by introducing a scaling factor $\kappa > 0$ such that we scale either: 1) $\tilde{f} \leftarrow \frac{1}{\kappa}\tilde{f}$ so that $\Theta \leftarrow \frac{1}{\kappa^2}\Theta$, or 2) $e_c \leftarrow \kappa e_c$ where $e_c \in \mathbb{R}^C$ is the one-hot vector denoting class $c \leq C$. We choose option 2) for our implementation, tuning κ on a small set of values chosen to match the second moments of the randomly initialised baselearners, in logit space, of each ensemble method on the training set. We found κ to be an important hyperparameter that can determine a trade-off between in-distribution and out-of-distribution performance: see Appendix H for further details.

[5] We use the *sigmoid* function, instead of *softplus* [11], to enforce positivity on $\sigma^2_{\boldsymbol{\theta}}(\cdot)$, because our data will be standardised.

4 Experiments

Due to limited space, Appendix I will contain all experimental details not discussed in this section.

Toy 1D regression task We begin with a toy 1D example $y = x\text{sin}(x) + \epsilon$, using homoscedastic $\epsilon \overset{d}{\sim} \mathcal{N}(0, 0.1^2)$. We use a training set of 20 points partitioned into two clusters, in order to detail uncertainty on out-of-distribution test data. For each ensemble method, we use MLP baselearners with two hidden layers of width 512, and erf activation. The choice of erf activation means that both the NTK Θ and NNGP kernel $\mathcal{K}$ are analytically available [23, 42]. We compare ensemble methods to the analytic GP posterior using either Θ or $\mathcal{K}$ as prior covariance function using the Neural Tangents library [38].

Figure 1 compares the analytic NTKGP posterior predictive with the analytic NNGP posterior predictive, as well as three different ensemble methods: deep ensembles, RP-param and NTKGP-param. We plot 95% predictive confidence intervals, treating ensembles as one Gaussian predictive distribution with matched moments like Lakshminarayanan et al. [11]. As expected, both NTKGP-param and RP-param ensembles have similar predictive means to the analytic NTKGP posterior. Likewise, we see that only our NTKGP-param ensemble predictive variances match the analytic NTKGP posterior. As foreseen in Proposition 2, the analytic NNGP posterior and other ensemble methods make more confident predictions than the NTKGP posterior, which in this example results in overconfidence on out-of-distribution data.[6]

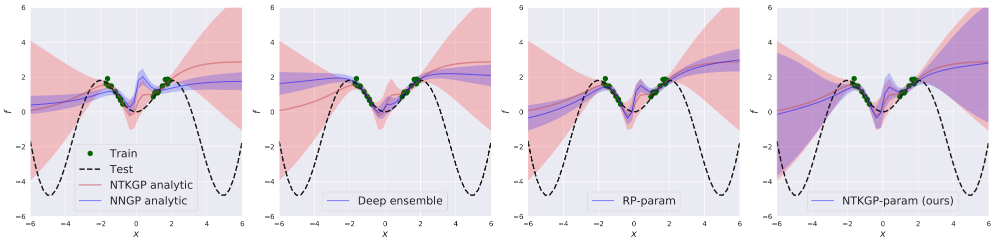

Figure 1: All subplots plot the analytic NTKGP posterior (in red). From left to right, (in blue): analytic NNGP posterior; deep ensembles; RP-param; and NTKGP-param (ours). For each method we plot the mean prediction and 95% predictive confidence interval. Green points denote the training data, and the black dotted line is the true test function $y = x\text{sin}(x)$.

Flight Delays We now compare different ensemble methods on a large scale regression problem using the Flight Delays dataset [43], which is known to contain dataset shift. We train heteroscedastic baselearners on the first 700k data points and test on the next 100k test points at 5 different starting points: 700k, 2m (million), 3m, 4m and 5m. The dataset is ordered chronologically in date through the year 2008, so we expect the NTKGP methods to outperform standard deep ensembles for the later starting points. Figure 2 (Left) confirms our hypothesis. Interestingly, there seems to be a seasonal effect between the 3m and 4m test set that results in stronger performance in the 4m test set than the 3m test set, for ensembles trained on the first 700k data points. We see that our Bayesian deep ensembles perform slightly worse than standard deep ensembles when there is little or no test data shift, but fail more gracefully as the level of dataset shift increases.

Figure 2 (Right) plots confidence versus error for different ensemble methods on the combined test set of 5×100k points. For each precision threshold τ, we plot root-mean-squared error (RMSE) on examples where predictive precision is larger than τ, indicating confidence. As we can see, our NTKGP methods incur lower error over all precision thresholds, and this contrast in performance is magnified for more confident predictions.

MNIST vs NotMNIST We next move onto classification experiments, comparing ensembles trained on MNIST and tested on both MNIST and NotMNIST.[7] Our baselearners are MLPs with 2-hidden layers, 200 hidden units per layer and ReLU activation. The weight parameter initialisation variance σ_W^2 is tuned using the validation accuracy on a small set of values around the He initialisation, σ_W^2=2, [44] for all classification experiments. Figure 3 shows both in-distribution and out-of-distribution performance across different ensemble methods. In Figure 3 (left), we see that our

[6] Code for this experiment is available at: `https://github.com/bobby-he/bayesian-ntk`.
[7] Available at `http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html`

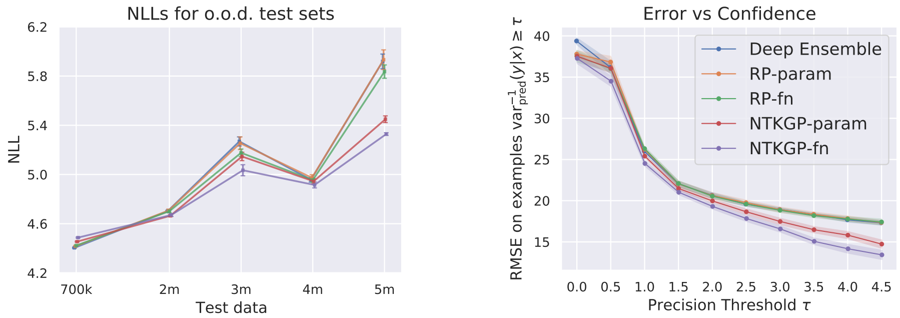

Figure 2: (Left) Flight Delays NLLs for ensemble methods trained on first 700k points of the dataset and tested on various out-of-distribution test sets, with time shift between training set and test set increasing along the x-axis. (Right) Error vs Confidence curves for ensembles tested on all 5×100k test points combined. Both plots include 95% CIs corresponding to 10 independent ensembles.

NTKGP methods suffer from slightly worse in-distribution test performance, with around 0.2% increased error for ensemble size 10. However, in Figure 3 (right), we plot error versus confidence on the combined MNIST and NotMNIST test sets: for each test point $(\boldsymbol{x}, y)$, we calculate the ensemble prediction $p(y = k|\boldsymbol{x})$ and define the predicted label as $\hat{y} = \text{argmax}_k p(y = k|\boldsymbol{x})$, with confidence $p(y = \hat{y}|\boldsymbol{x})$. Like Lakshminarayanan et al. [11], for each confidence threshold $0 \leq \tau \leq 1$, we plot the average error for all test points that are more confident than τ. We count all predictions on the NotMNIST test set to be incorrect. We see in Figure 3 (right) that the NTKGP methods vastly outperform both deep ensembles and RP methods, obtaining over 15% lower error on test points that have confidence τ=0.6, compared to all baselines. This is because our methods correctly make much more conservative predictions on the out-of-distribution NotMNIST test set, as can be seen by Figure 4, which plots histograms of predictive entropies. Due to the simple MLP architecture and ReLU activation, we can compare ensemble methods to analytic NTKGP results in Figures 3 & 4, where we see a close match between the NTKGP ensemble methods (at larger ensemble sizes) and the analytic predictions, both on in-distribution and out-of-distribution performance.

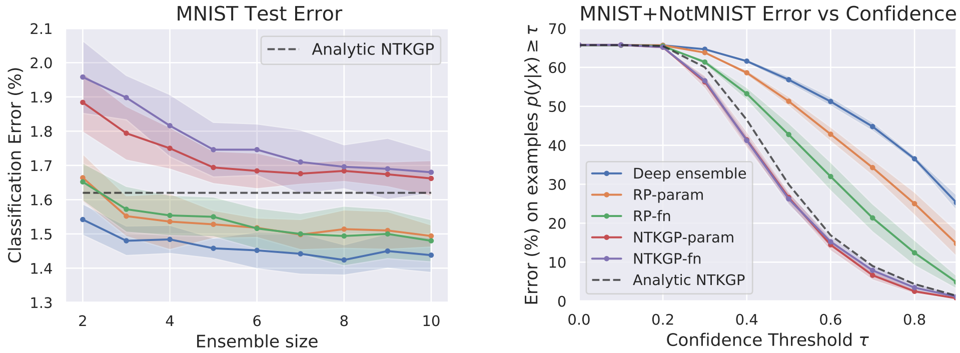

Figure 3: (Left) Classification error on MNIST test set for different ensemble sizes. (Right) Error versus Confidence plots for ensembles, of size 10, trained on MNIST and tested on both MNIST and NotMNIST. CIs correspond to 5 independent runs.

CIFAR-10 vs SVHN Finally, we present results on a larger-scale image classification task: ensembles are trained on CIFAR-10 and tested on both CIFAR-10 and SVHN. We conduct the same setup as for the MNIST vs NotMNIST experiment, with baselearners taking the Myrtle-10 CNN architecture [40] of channel-width 100. Figure 5 compares in distribution and out-of-distribution performance: we see that our NTKGP methods and RP-fn perform best on in-distribution test error. Unlike on the simpler MNIST task, there is no clear difference on the corresponding error versus confidence plot, and this is also reflected in the entropy histograms, which can be found in Figure 8 of Appendix I.

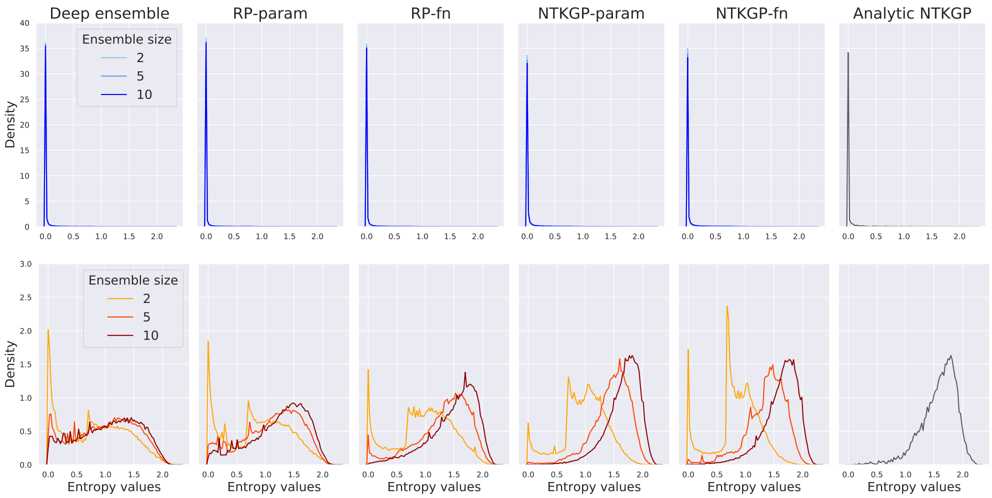

Figure 4: Histograms of predictive entropy on MNIST (top) and NotMNIST (bottom) test sets for different ensemble methods of different ensemble sizes, and also for Analytic NTKGP.

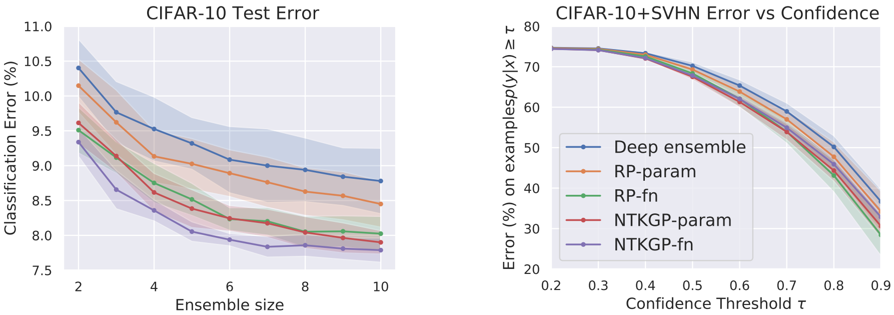

Figure 5: (Left) Classification error on CIFAR-10 test set for different ensemble sizes. (Right) Error versus Confidence plots of ensembles trained on CIFAR-10 and tested on both CIFAR-10 and SVHN. CIs correspond to 5 independent runs.

5 Discussion

We built on existing work regarding the Neural Tangent Kernel (NTK), which showed that there is no posterior predictive interpretation to a standard deep ensemble in the infinite width limit. We introduced a simple modification to training that enables a GP posterior predictive interpretation for a wide ensemble, and showed empirically that our Bayesian deep ensembles emulate the analytic posterior predictive when it is available. In addition, we demonstrated that our Bayesian deep ensembles often outperform standard deep ensembles in out-of-distribution settings for both regression and classification tasks.

In terms of limitations, our methods may perform worse than standard deep ensembles [11] when confident predictions are not detrimental, though this can be alleviated via NTK hyperparameter tuning. Moreover, our analyses are planted in the “lazy learning” regime [45, 46], and we have not considered finite-width corrections to the NTK during training [47–49]. In spite of these limitations, the search for a Bayesian interpretation to deep ensembles [11] is of particular relevance to the Bayesian deep learning community, and we believe our contributions provide useful new insights to resolving this problem by examining the limit of infinite-width.

A natural question that emerges from our work is how to tune hyperparameters of the NTK to best capture inductive biases or prior beliefs about the data. Possible lines of enquiry include: the large-depth limit [50], the choice of architecture [51], and the choice of activation [52]. Finally, we would like to assess our Bayesian deep ensembles in non-supervised learning settings, such as active learning or reinforcement learning.

Acknowledgments and Disclosure of Funding

We thank Arnaud Doucet, Edwin Fong, Michael Hutchinson, Lewis Smith, Jasper Snoek, Jascha Sohl-Dickstein and Sheheryar Zaidi, as well as the anonymous reviewers, for helpful discussions and feedback. We also thank the JAX and Neural Tangents teams for their open-source software. BH is supported by the EPSRC and MRC through the OxWaSP CDT programme (EP/L016710/1).

References

[1] James Aitchison. Goodness of Prediction Fit. *Biometrika*, 62(3):547–554, 1975.

[2] David JC MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1992.

[3] Radford M Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media, 2012.

[4] Max Welling and Yee W Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

[5] Alex Graves. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[6] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

[7] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org, 2017.

[8] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. *arXiv preprint arXiv:1803.04386*, 2018.

[9] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks. *arXiv preprint arXiv:1903.05779*, 2019.

[10] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[11] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[12] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *NeurIPS*, 2019.

[13] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep Ensembles: A Loss Landscape Perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[14] Andrew Gordon Wilson and Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. *arXiv preprint arXiv:2002.08791*, 2020.

[15] Radford M Neal. Priors for Infinite Networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

[16] Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep Neural Networks as Gaussian Processes. In *International Conference on Learning Representations*, 2018.

[17] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian Process Behaviour in Wide Deep Neural Networks. In *International Conference on Learning Representations*, volume 4, 2018.

[18] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep Convolutional Networks as shallow Gaussian Processes. In *International Conference on Learning Representations*, 2019.

[19] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes. In *International Conference on Learning Representations*, 2019.

[20] Greg Yang. Tensor Programs I: Wide Feedforward or Recurrent Neural Networks of Any Architecture are Gaussian Processes. *arXiv preprint arXiv:1910.12478*, 2019.

[21] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: NNGP and NTK for deep attention networks. *arXiv preprint arXiv:2006.10540*, 2020.

[22] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, pages 8571–8580, 2018.

[23] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models under gradient descent. In *Advances in Neural Information Processing Systems*, pages 8570–8581, 2019.

[24] Ian Osband, John Aslanides, and Albin Cassirer. Randomized Prior Functions for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

[25] Tim Pearce, Mohamed Zaki, Alexandra Brintrup, Nicolas Anastassacos, and Andy Neely. Uncertainty in Neural Networks: Bayesian Ensembling. *arXiv preprint arXiv:1810.05546*, 2018.

[26] Kamil Ciosek, Vincent Fortuin, Ryota Tomioka, Katja Hofmann, and Richard Turner. Conservative Uncertainty Estimation By Fitting Prior Networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJlahxHYDS`.

[27] Yehuda Hoffman and Erez Ribak. Constrained Realizations of Gaussian Fields: A Simple Algorithm. *The Astrophysical Journal*, 380:L5–L8, 1991.

[28] Greg Yang. Scaling Limits of Wide Neural Networks with Weight Sharing: Gaussian Process Behavior, Gradient Independence, and Neural Tangent Kernel Derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[29] Greg Yang. Tensor Programs II: Neural Tangent kernel for Any Architecture. *arXiv preprint arXiv:2006.14548*, 2020.

[30] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1032–1041, 2019.

[31] Daniel Park, Jascha Sohl-Dickstein, Quoc Le, and Samuel Smith. The Effect of Network Width on Stochastic Gradient Descent and Generalization: an Empirical Study. In *International Conference on Machine Learning*, pages 5042–5051, 2019.

[32] Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020.

[33] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies. In *Advances in Neural Information Processing Systems*, pages 4761–4771, 2019.

[34] Alexander G de G Matthews, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Sample-then-optimize posterior sampling for Bayesian linear models. In *NeurIPS Workshop on Advances in Approximate Bayesian Inference*, 2017.

[35] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[36] Carl Edward Rasmussen. Gaussian Processes in Machine Learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[37] Wei Hu, Zhiyuan Li, and Dingli Yu. Simple and Effective Regularization Methods for Training on Noisily Labeled Data with Generalization Guarantee. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hke3gyHYwH.

[38] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural Tangents: Fast and Easy Infinite Neural Networks in Python. In *International Conference on Learning Representations*, 2020. URL https://github.com/google/neural-tangents.

[39] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On Exact Computation with an Infinitely Wide Neural Net. In *Advances in Neural Information Processing Systems*, pages 8139–8148, 2019.

[40] Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Ludwig Schmidt, Jonathan Ragan-Kelley, and Benjamin Recht. Neural Kernels Without Tangents. *arXiv preprint arXiv:2003.02237*, 2020.

[41] E Fong and C C Holmes. On the marginal likelihood and cross-validation. *Biometrika*, 107(2):489–496, 01 2020. ISSN 0006-3444. doi: 10.1093/biomet/asz077. URL https://doi.org/10.1093/biomet/asz077.

[42] Christopher KI Williams. Computing with Infinite Networks. In *Advances in neural information processing systems*, pages 295–301, 1997.

[43] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian Processes for Big Data. In *Uncertainty in Artificial Intelligence*, page 282. Citeseer, 2013.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[45] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947, 2019.

[46] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and Rich Regimes in Overparametrized Models. *arXiv preprint arXiv:2002.09277*, 2020.

[47] Ethan Dyer and Guy Gur-Ari. Asymptotics of Wide Networks from Feynman Diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

[48] Jiaoyang Huang and Horng-Tzer Yau. Dynamics of Deep Neural Networks and Neural Tangent Hierarchy. *arXiv preprint arXiv:1909.08156*, 2019.

[49] Boris Hanin and Mihai Nica. Finite Depth and Width Corrections to the Neural Tangent Kernel. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgndT4KwB.

[50] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. Mean-field Behaviour of Neural Tangent Kernel for Deep Neural Networks. *arXiv preprint arXiv:1905.13654*, 2019.

[51] Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris Holmes, Frank Hutter, and Yee Whye Teh. Neural Ensemble Search for Performant and Calibrated Predictions. *arXiv preprint arXiv:2006.08573*, 2020.

[52] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *arXiv preprint arXiv:2006.10739*, 2020.

[53] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient BackProp. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[54] David Williams. *Probability with Martingales*. Cambridge University Press, 1991. doi: 10.1017/CBO9780511813658.

[55] Patrick Billingsley. *Probability and Measure*. John Wiley and Sons, second edition, 1986.

[56] Linh Tran, Bastiaan S Veeling, Kevin Roth, Jakub Swiatkowski, Joshua V Dillon, Jasper Snoek, Stephan Mandt, Tim Salimans, Sebastian Nowozin, and Rodolphe Jenatton. Hydra: Preserving Ensemble Diversity for Model Distillation. *arXiv preprint arXiv:2001.04694*, 2020.

[57] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The Reversible Residual Network: Backpropagation Without Storing Activations. In *Advances in Neural Information Processing Systems 30*, pages 2214–2224, 2017.

[58] Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty Estimation Using a Single Deep Deterministic Neural Network. In *International Conference on Machine Learning*, 2020.

[59] Michael W Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-an Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors. *arXiv preprint arXiv:2005.07186*, 2020.

[60] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Recap of standard and NTK parameterisations

For completeness, we recap the difference between standard and NTK parameterisations & initialisations [22, 23] for an MLP in this section.

Consider an MLP with L hidden layers of widths from $n_0{=}d$ to n_L respectively, and final readout layer with $n_{L+1} = C$. For a given $\boldsymbol{x} \in \mathbb{R}^d$, under the *NTK* parameterisation the recurrence relation that constitutes the forward pass of the NN is then:

$$\alpha^{(0)}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{x} \tag{13}$$

$$\tilde{\alpha}^{(l+1)}(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\sigma_W}{\sqrt{n_l}} W^{(l)} \alpha^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}) + \sigma_b b^{(l)} \tag{14}$$

$$\alpha^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}) = \phi(\tilde{\alpha}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta})) \tag{15}$$

for $l \leq L$ where $\tilde{\alpha}^{(l)}$ and $\alpha^{(l)}$ are the preactivations and activations respectively at layer l, with entrywise nonlinearity $\phi(\cdot)$. In the NTK parameteriation, all parameters $W^{(l)} \in \mathbb{R}^{n_{l+1} \times n_l}$ and $b^{(l)} \in \mathbb{R}^{n_{l+1}}$ for all layers l are initialised as i.i.d. standard normal $\mathcal{N}(0, 1)$. The hyperparameters σ_W and σ_b are known as the weight and bias variances respectively, and are hyperparameters of the infinite width limit NTK Θ.

On the other hand, under *standard* parameterisation, the recurrence relation of the NN is:

$$\alpha^{(0)}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{x} \tag{16}$$

$$\tilde{\alpha}^{(l+1)}(\boldsymbol{x}, \boldsymbol{\theta}) = W^{(l)} \alpha^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}) + b^{(l)} \tag{17}$$

$$\alpha^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}) = \phi(\tilde{\alpha}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta})) \tag{18}$$

with $W^{(l)}_{i,j} \sim \mathcal{N}(0, \frac{1}{n_l}\sigma_W^2)$ and $b^{(l)}_j \sim \mathcal{N}(0, \sigma_b^2)$ at initialisation. Commonly used initialisation schemes like LeCun [53] or He [44] fall into this category.

Regardless of parameterisation, our notation from Sections 2 & 3 corresponds to $f(\boldsymbol{x}, \boldsymbol{\theta}) = \tilde{\alpha}^{(L+1)}(\boldsymbol{x}, \boldsymbol{\theta})$, with $\boldsymbol{\theta} = \{W^{(l)}, b^{(l)}\}_{l=0}^{L}$, $\boldsymbol{\theta}^{\leq L} = \{W^{(l)}, b^{(l)}\}_{l=0}^{L-1}$ and $\boldsymbol{\theta}^{L+1} = \{W^{(L)}, b^{(L)}\}$.

We see that the different parameterisations yield the same distribution for the functional output $f(\cdot, \boldsymbol{\theta})$ at initialisation, but give different scalings to the parameter gradients in the backward pass. Sohl-Dickstein et al. [32] have recently explored further variants of these parameterisations.

B Proofs

B.1 Proof of Proposition 1

Proposition 1. $\delta(\cdot) \xrightarrow{d} \mathcal{GP}(0, \Theta^{\leq L})$ *and is independent of* $f_0(\cdot)$ *in the infinite width limit. Thus,* $\tilde{f}_0(\cdot) = f_0(\cdot) + \delta(\cdot) \xrightarrow{d} \mathcal{GP}(0, \Theta)$.

Proof. For notational ease, let us define two jointly independent GPs $g(\cdot) \overset{d}{\sim} \mathcal{GP}(0, \Theta^{\leq L})$ & $h(\cdot) \overset{d}{\sim} \mathcal{GP}(0, \mathcal{K})$. By independence, we have $g(\cdot) + h(\cdot) \overset{d}{\sim} \mathcal{GP}(0, \Theta)$. Moreover, let $\delta_m(\cdot)$, $f^0_m(\cdot)$ and $\boldsymbol{\theta}^0_m$ denote $\delta(\cdot)$, $f_0(\cdot)$ and $\boldsymbol{\theta}_0$ respectively at width parameter $m \in \mathbb{N}$. The infinite width limit thus corresponds to $m \to \infty$.

For our purposes, it will be sufficient to prove convergence of finite-dimensional marginals, $(\delta_m(\mathcal{X}), f^0_m(\mathcal{X}')) \xrightarrow{d} (g(\mathcal{X}), h(\mathcal{X}'))$ jointly, for arbitrary sets of inputs $\mathcal{X}, \mathcal{X}'$. Note that previous work [16, 17] has already shown that $f^0_m(\mathcal{X}') \xrightarrow{d} h(\mathcal{X}')$.

The proof that $(\delta_m(\mathcal{X}), f^0_m(\mathcal{X}')) \xrightarrow{d} (g(\mathcal{X}), h(\mathcal{X}'))$ relies on Lévy's Convergence theorem [54] and the Cramér-Wold device (Theorem 29.4 of [55]). Using these results it is sufficient to show, denoting φ_X as the characteristic function of a random variable X, that:

$$\varphi_{Y_m}(t) \to \varphi_Y(t) \tag{19}$$

where $Y_m = u^\top \delta_m(\mathcal{X}) + v^\top f_m^0(\mathcal{X}')$ and $Y = u^\top g(\mathcal{X}) + v^\top h(\mathcal{X}')$, for all $t \in \mathbb{R}, u \in \mathbb{R}^{|\mathcal{X}|C}$ and $v \in \mathbb{R}^{|\mathcal{X}'|C}$. But:

$$\begin{aligned}
\varphi_{Y_m}(t) =& \mathbb{E}\big[\exp(itY_m)\big] && (20)\\
=& \mathbb{E}\Big[\mathbb{E}\big[\exp(itY_m) \mid \boldsymbol{\theta}_{0,m}\big]\Big] && (21)\\
=& \mathbb{E}_{\boldsymbol{\theta}_{0,m}}\big[\exp\big(-t^2u^\top \hat{\Theta}^{\leq L}_{0,m}(\mathcal{X},\mathcal{X})u + itv^\top f_m^0(\mathcal{X}')\big)\big] && (22)\\
=& \exp\big(-t^2u^\top \Theta^{\leq L}(\mathcal{X},\mathcal{X})u\big)\mathbb{E}_{\boldsymbol{\theta}_{0,m}}\big[\exp\big(itv^\top f_m^0(\mathcal{X}')\big)\big] + r_m && (23)\\
\to& \mathbb{E}\big[\exp(itY)\big] && (24)
\end{aligned}$$

where r_m, defined as the difference between Eqs. (23) & (22), can be shown to be $o_m(1)$ using the Bounded Convergence theorem and the empirical NTK convergence results, and by noting that proofs of NTK convergence [22, 23, 28, 29] are all done on a layer-by-layer basis.

The claim that $\tilde{f}_0(\cdot) = f_0(\cdot) + \delta(x) \xrightarrow{d} \mathcal{GP}(0, \Theta)$ then follows by setting $\mathcal{X} = \mathcal{X}'$ and $v = u$.

□

B.2 Proof of Proposition 2

Proposition 2. *For $\sigma^2{=}0$, $\Sigma_{NTKGP} \succeq \Sigma_{DE} \succeq \Sigma_{NNGP}$. Similarly, for $\sigma^2{>}0$, $\Sigma_{NTKGP} \succeq \Sigma_{RP} \succeq \Sigma_{NNGP}$.*

Proof. We will prove the case for $\sigma^2 > 0$ as the case for $\sigma^2 = 0$ is similar, and one can replace inversions of $\Theta(\mathcal{X},\mathcal{X})$ and $\mathcal{K}(\mathcal{X},\mathcal{X})$ with generalised inverses if need be.

Let $\mathcal{X}'$ be an arbitrary test set. We will first show $\Sigma_{\text{RP}} \succeq \Sigma_{\text{NNGP}}$. It will suffice to show that $\Sigma_{\text{RP}}(\mathcal{X}',\mathcal{X}') - \Sigma_{\text{NNGP}}(\mathcal{X}',\mathcal{X}') \succeq 0$ is a p.s.d. matrix. But it is not hard to check that:

$$\Sigma_{\text{RP}}(\mathcal{X}',\mathcal{X}') - \Sigma_{\text{NNGP}}(\mathcal{X}',\mathcal{X}') = U(\mathcal{K}(\mathcal{X},\mathcal{X}) + \sigma^2 I)U^\top \qquad (25)$$

which is clearly p.s.d, where $U = \Theta(\mathcal{X}',\mathcal{X})(\Theta(\mathcal{X},\mathcal{X}) + \sigma^2 I)^{-1} - \mathcal{K}(\mathcal{X}',\mathcal{X})(\mathcal{K}(\mathcal{X},\mathcal{X}) + \sigma^2 I)^{-1} \in \mathbb{R}^{|\mathcal{X}'|\times|\mathcal{X}|}$

Likewise, to show $\Sigma_{\text{NTK}} \succeq \Sigma_{\text{RP}}$ we can check that:

$$\Sigma_{\text{NTK}}(\mathcal{X}',\mathcal{X}') - \Sigma_{\text{RP}}(\mathcal{X}',\mathcal{X}') = U_1 + U_2\Delta(\mathcal{X},\mathcal{X})U_2^\top \succeq 0 \qquad (26)$$

where

$$U_1 = \Delta(\mathcal{X}',\mathcal{X}') - \Delta(\mathcal{X}',\mathcal{X})\Delta^g(\mathcal{X},\mathcal{X})\Delta(\mathcal{X},\mathcal{X}') \qquad (27)$$

and $\Delta = \Theta^{\leq L} \succeq 0$ is the contributions to the NTK from parameters before the final layer as before. Finally, we need to define U_2 as:

$$U_2 = \Theta(\mathcal{X}',\mathcal{X})(\Theta(\mathcal{X},\mathcal{X}) + \sigma^2 I)^{-1} - \Delta(\mathcal{X}',\mathcal{X})\Delta^g(\mathcal{X},\mathcal{X}) \qquad (28)$$

The notation $\Delta^g(\mathcal{X},\mathcal{X})$ denotes the generalised inverse. $U_1 \succeq 0$ follows from standard properties of generalised Schur complements, as does the fact that $\Delta(\mathcal{X}',\mathcal{X})\Delta^g(\mathcal{X},\mathcal{X})\Delta(\mathcal{X},\mathcal{X}) = \Delta(\mathcal{X}',\mathcal{X})$, which is required for Eq. (26) to hold.

□

C Alternative constructions of NTKGP baselearners

To summarise the analysis in Section 3, the criteria for an NTKGP baselearner $\tilde{f}(\cdot,\boldsymbol{\theta})$ is that: 1) $\tilde{f}(\cdot,\boldsymbol{\theta}_0) \xrightarrow{d} \mathcal{GP}(0,\Theta)$ as width increases, while 2) preserving the initial Jacobian $\nabla_{\boldsymbol{\theta}} f_0(\cdot) = \nabla_{\boldsymbol{\theta}} \tilde{f}_0(\cdot)$.

A possible alternative construction would be if one could (approximately) sample a fixed $f^* \overset{d}{\sim} \mathcal{GP}(0,\Theta)$, and set:

$$\tilde{f}_t(\cdot) = f_t(\cdot) + f^*(\cdot) - f_0(\cdot) \qquad (29)$$

It is easy to approximately sample f^* for finite width NNs using a single JVP, under either standard or NTK parameterisation, by sampling $\tilde{\boldsymbol{\theta}}$ independent of $\boldsymbol{\theta}_0$ and setting:

$$f^*(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\tilde{\boldsymbol{\theta}} \tag{30}$$

Note that Eq. (29) requires computation of two forward passes f_t and f_0 in addition to a JVP $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\tilde{\boldsymbol{\theta}}$. For some implementations of JVPs, such as in JAX [35], the computation of f_0 will come essentially for free alongside the computation of $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\tilde{\boldsymbol{\theta}}$, because the JVP is centered about the same "primal" parameters $\boldsymbol{\theta}_0$ that are used for f_0. Hence, this alternative $\tilde{f}$ presented in Eq. (29) may have similar costs to our main construction in Section 3, for certain AD packages.

A second valid alternative to $\tilde{f}_t$ would be to replace f_t with f_t^{lin}, which would give $\tilde{f}^{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \tilde{\boldsymbol{\theta}})\boldsymbol{\theta}_t$ (where we swap $\tilde{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}_0$ for notational consistency with other NTKGP methods, and initialise at $\boldsymbol{\theta}_0$). Because $\tilde{\boldsymbol{\theta}}$ is fixed, we see that $\tilde{f}^{\text{lin}}(\cdot, \boldsymbol{\theta}_t)$ is linear in $\boldsymbol{\theta}_t$. This gives a realisation of the "sample-then-optimize" approach [34] to give posterior samples from randomly initialised linear models, and ensures that $\tilde{f}_{\infty}^{\text{lin}}(\cdot)$ is an exact posterior sample (using the empirical NTK $\hat{\Theta}_0$ as prior kernel) irrespective of parameterisation or width. Note though, of course, the linearised regime holds for $\tilde{f}_t^{\text{lin}}$ throughout parameter space, hence for strongly convex optimisation problems like regression tasks with observation noise, the initialisation is irrelevant. We will call $\tilde{f}_{\infty}^{\text{lin}}$ trained in such a way an *NTKGP-Lin* baselearner.

D Regularisation in the NTKGP and RP training procedures

As stated in Lemma 3 of Osband et al. [24], suppose we are in a Bayesian linear regression setting with linear map $g_{\boldsymbol{\theta}}(\boldsymbol{z}) = \boldsymbol{z}^\top \boldsymbol{\theta}$, model $y = g_{\boldsymbol{\theta}}(\boldsymbol{z}) + \epsilon$ for $\epsilon \sim \mathcal{N}(0, \sigma^2)$ i.i.d., and parameter prior $\boldsymbol{\theta} \sim N(0, \lambda I_p)$. Then, having observed training data $\{(\boldsymbol{z}_i, y_i)\}_{i=1}^n$, solving the following optimisation problem returns a posterior sample $\boldsymbol{\theta}$:

$$\tilde{\boldsymbol{\theta}} + \operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \frac{1}{2\sigma^2} \left\| \tilde{y}_i - (g_{\boldsymbol{\theta}} + g_{\tilde{\boldsymbol{\theta}}})(\boldsymbol{z}_i) \right\|_2^2 + \frac{1}{2\lambda} \left\| \boldsymbol{\theta} \right\|_2^2 \tag{31}$$

where $\tilde{y}_i \sim \mathcal{N}(y_i, \sigma^2)$ and $\tilde{\boldsymbol{\theta}} \sim \mathcal{N}(0, \lambda I_p)$.

We see that when there is a homoscedastic prior $\mathcal{N}(0, \lambda I_p)$ for $\boldsymbol{\theta}$ that the correct weighting of L^2 regularisation is $\|\boldsymbol{\theta}\|_{\Lambda}^2 = \frac{1}{\lambda}\boldsymbol{\theta}^\top \boldsymbol{\theta}$. In fact, even with a heteroscedastic prior $\boldsymbol{\theta} \sim \mathcal{N}(0, \Lambda)$ with a diagonal matrix $\Lambda \in \mathbb{R}_+^{p \times p}$ and diagonal entries $\{\lambda_j\}_{j=1}^p$, it is straightforward to show that the correct setting of regularisation is $\|\boldsymbol{\theta}\|_{\Lambda}^2 = \boldsymbol{\theta}^\top \Lambda^{-1} \boldsymbol{\theta}$ in order to obtain a posterior sample of $\boldsymbol{\theta}$. For RP-param or NTKGP-param methods, with initial parameters $\boldsymbol{\theta}_0$, we have regularisation $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_{\Lambda}^2 = (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \Lambda^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)$, which can be seen as a Mahalanobis distance.

For an NN in the linearised regime [23], this is related to the fact that the NTK and standard parameterisations initialise parameters differently, yet yield the same functional distribution for a randomly initialised NN. In the standard parameterisation, λ_j will be a factor of the NN width smaller than in the NTK parameterisation, but the corresponding feature map $\boldsymbol{z}$ will be a square root factor of the NN width larger. Thus, solving Eq. (31) will lead to the same functional outputs in both parameterisations, if the NN remains in the linearised regime. However, only with our NTKGP trained baselearners $\tilde{f}$ do you get a posterior interpretation to the trained NN because of the difference between the NNGP and the NTK that standard training does not account for, and because the linearised regime only holds locally to the parameter initialisation.

E Additional ensemble algorithms

Here, we present our ensemble algorithms for NTKGP-Lin (Algorithm 2) and NTKGP-fn (Algorithm 3), to complement the NTKGP-param algorithm that was presented in Section 3.4.

Algorithm 2 NTKGP-Lin ensemble

```
Require: Data $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, loss function $\mathcal{L}$, NN model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, Ensemble size $K \in \mathbb{N}$, noise procedure: data_noise, NN parameter initialisation scheme: init(·)
  for $k = 1, \dots, K$ do
    Form $\{\mathcal{X}_k, \mathcal{Y}_k\}$ = data_noise($\mathcal{D}$)
    Initialise $\boldsymbol{\theta}_k \overset{d}{\sim}$ init(·)
    Initialise $\tilde{\boldsymbol{\theta}}_k \overset{d}{\sim}$ init(·)
    Define $\tilde{f}^{\text{lin}}_k(\boldsymbol{x}, \boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \tilde{\boldsymbol{\theta}}_k)\boldsymbol{\theta}_t$ and set $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_k$
    Optimise $\mathcal{L}(\tilde{f}^{\text{lin}}_k(\mathcal{X}_k, \boldsymbol{\theta}_t), \mathcal{Y}_k) + \frac{1}{2}\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_k\right\|^2_\Lambda$ for $\boldsymbol{\theta}_t$ to obtain $\hat{\boldsymbol{\theta}}_k$
  end for
  return ensemble $\{\tilde{f}^{\text{lin}}_k(\cdot, \hat{\boldsymbol{\theta}}_k)\}_{k=1}^K$
```

Algorithm 3 NTKGP-fn ensemble

```
Require: Data $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, loss function $\mathcal{L}$, NN model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, Ensemble size $K \in \mathbb{N}$, noise procedure: data_noise, NN parameter initialisation scheme: init(·)
  for $k = 1, \dots, K$ do
    Form $\{\mathcal{X}_k, \mathcal{Y}_k\}$ = data_noise($\mathcal{D}$)
    Initialise $\boldsymbol{\theta}_k \overset{d}{\sim}$ init(·)
    Initialise $\tilde{\boldsymbol{\theta}}_k \overset{d}{\sim}$ init(·) and denote $\tilde{\boldsymbol{\theta}}_k$ = concat($\{\tilde{\boldsymbol{\theta}}^{\leq L}_k, \tilde{\boldsymbol{\theta}}^{L+1}_k\}$)
    Set $\boldsymbol{\theta}^*_k$ = concat($\{\sqrt{2}\tilde{\boldsymbol{\theta}}^{\leq L}_k, \tilde{\boldsymbol{\theta}}^{L+1}_k\}$)
    Define $\delta(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \tilde{\boldsymbol{\theta}}_k)\boldsymbol{\theta}^*_k$
    Define $\tilde{f}_k(\boldsymbol{x}, \boldsymbol{\theta}_t) = f(\boldsymbol{x}, \boldsymbol{\theta}_t) + \delta(\boldsymbol{x})$ and set $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_k$
    Optimise $\mathcal{L}(\tilde{f}_k(\mathcal{X}_k, \boldsymbol{\theta}_t), \mathcal{Y}_k) + \frac{1}{2}\left\|\boldsymbol{\theta}_t\right\|^2_\Lambda$ for $\boldsymbol{\theta}_t$ to obtain $\hat{\boldsymbol{\theta}}_k$
  end for
  return ensemble $\{\tilde{f}_k(\cdot, \hat{\boldsymbol{\theta}}_k)\}_{k=1}^K$
```

In Algorithm 3 we seek to reinitialise $\tilde{f}_k(\boldsymbol{x}, \boldsymbol{\theta}_0)$ from $\mathcal{GP}(0, \mathcal{K})$ to $\mathcal{GP}(0, 2\Theta)$ in the infinite width limit, following the randomised prior function method of Osband et al. [24]. While there are many ways to do this we choose to use only one JVP, with a reweighted tangent vector, for $\delta(\cdot)$ in order to reduce extra computational costs. It would be similarly possible to model a scaling factor β for the prior function, like [24], using a single JVP with a differently reweighted tangent vector.

Note also that for the NTKGP-fn it is unreasonable to assume that the linearised NN dynamics will hold true for the duration of training because, unlike in NTKGP-param (Algorithm 1) we regularise towards the origin not the initialised parameters.

F Aggregating predictions from ensemble members

For completeness, we now describe how to aggregate predictions from ensemble members. Given a test point $(\boldsymbol{x}, y)$, for each baselearner NN $k \leq K$, we suppose we have a probabilistic prediction $p_k(y|\boldsymbol{x})$ obtained from the NN output. We then treat the ensemble as a uniformly-weighted mixture model over baselearners and combine predictions as $p(y|\boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^K p_k(y|\boldsymbol{x})$. For our Bayesian deep ensembles, we can view this aggregation as a Monte Carlo approximation of the GP posterior predictive with NTK prior.

For classification tasks, this aggregation is exactly an average of predicted probabilities. For regression tasks, the prediction is a mixture of normal distributions, and we follow Lakshminarayanan et al. [11] by approximating the ensembled prediction as a single Gaussian with matched moments. That is to say, if $p_k(y|\boldsymbol{x}) \sim \mathcal{N}(\mu_k(\boldsymbol{x}), \sigma^2_k(\boldsymbol{x}))$, then we approximate $p(y|\boldsymbol{x})$ by $\mathcal{N}(\mu_*(\boldsymbol{x}), \sigma^2_*(\boldsymbol{x}))$ for $\mu_*(\boldsymbol{x}) = \frac{1}{K}\sum_k \mu_k(\boldsymbol{x})$ and $\sigma^2_*(\boldsymbol{x}) = \frac{1}{K}\sum_k(\mu^2_k(\boldsymbol{x}) - \mu^2_*(\boldsymbol{x})) + \sigma^2_k(\boldsymbol{x})$.

G Comparison of memory and computation costs for ensemble methods

There is only a negligible training-time computational overhead for our NTKGP methods compared to other ensemble methods [11, 24], for a training set of fixed size (e.g. MNIST, CIFAR-10). This is because one can obtain and store our fixed additive JVPs δ in a single pass over the training data. For test-time constrained applications, one can employ ensemble distillation [56] for our NTKGP ensembles as one would for standard deep ensembles.

For completeness, we include in Table 2 (left) the computational cost of different ensemble methods when the modified forward pass $\tilde{f}$ needs to be computed on the fly for new data, though we again stress that this is not necessary for train nor test time, as described in the paragraph above. A rule of thumb for a library offering forward-mode AD, like JAX [35], is that a JVP costs on the order of three standard forward passes in terms of FLOPs. We use forward-mode AD to compute JVPs as this is known to be more memory-efficient than reverse-mode AD for JVP computation. It is worth pointing out that our methods share the same trainable parameters as standard deep ensembles, and so do not incur any additional computational cost in the backward pass.

Table 2: Comparison of computational and memory costs of different ensemble methods per ensemble member. Computational costs are specified per (modified) forward pass and represent a naive worst-case scenario (presented for completeness); a more astute approach renders only a negligible difference between ensemble methods, as discussed in this section.

Method	Computational cost		Parameter sets to store	
	Forward passes	JVPs	Train time	Test time
Deep ensembles	1	0	1	1
RP-param	1	0	2	1
RP-fn	2	0	2	2
NTKGP-param	1	1	3	3
NTKGP-fn	1	1	3	3

In terms of memory, both NTKGP and RP methods require storage of extra sets of parameters in order to compute the untrainable additive functions $\delta(\cdot)$ and regularise in parameter space, displayed in Table 2 (right). However, the activations of the extra forward pass in the Randomised prior function method need not be stored. And moreover, forward mode JVPs are composed alongside the primitive operations that comprise the forward pass, so the memory requirements incurred by the extra JVP are independent of the NN depth for our NTKGP methods. Note that the memory bottleneck for large NNs is most often from the need to store activations for the backward pass [57] and not from storing parameter sets, hence our NTKGP ensembles are not affected by the main memory bottleneck for large NNs, relative to standard deep ensembles.

It is worth noting that our Bayesian deep ensembles still retain the distributability of standard deep ensembles. Moreover, our computational and memory costs still scale linearly in dataset size and parameter space dimension, enabling us to work with large scale datasets like Flight Delays [43].

Finally, in this section we only compare the costs associated to different ensemble methods. Ensembles methods are known to be computationally expensive and there has been recent interest in the community to derive new methods [58, 59] that reduce such costs. However, at the time of writing, deep ensembles [11] are state-of-the-art for uncertainty quantification tasks [12], and hence we believe a comparison of costs between ensemble methods is most appropriate for this work.

H Scaling for one-hot targets in classification

As discussed in Section 3.5 and repeated here for completeness: because $\delta(\cdot)$ is untrainable in our NTKGP methods, it is important to match the scale of the NTK Θ to the scale of the one-hot targets in multi-class classification settings. One can do this either by introducing a scaling factor $\kappa > 0$ such that we scale either: 1) $\tilde{f} \leftarrow \frac{1}{\kappa}\tilde{f}$ and so $\Theta \leftarrow \frac{1}{\kappa^2}\Theta$, or 2) $e_c \leftarrow \kappa e_c$ where $e_c \in \mathbb{R}^C$ is the one-hot vector denoting class $c \leq C$. We choose option 2) for our implementation.

To set κ, for each ensemble method we calculated the mean squared values of baselearner outputs at initialisation, which we define for convenience as ζ_0, on the training set for that particular ensemble method, and tuned κ^2 (based on validation accuracy) on a small linear scale centered around $C\zeta_0$, where C is the number of classes. This is in order to match the second moments of the random NNs at initialisation with the scaled one-hot targets across the C classes. For example, for NTKGP-param, we set $\zeta_0 = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}} \Theta(\boldsymbol{x}, \boldsymbol{x}) \in \mathbb{R}^+$.

To illustrate the importance of κ, in Figure 6 we present the corresponding results to Figure 5 where instead of setting κ dependent on the scale of each ensemble methods' initialised baselearners, as above, we set $\kappa = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}}\Theta(\boldsymbol{x},\boldsymbol{x})\in\mathbb{R}$ for all ensemble methods. This is the base κ value for NTKGP-param at initialisation, but note that we did not tune neither κ (around this base value) nor weight variance (set at $\sigma_W^2 = 2$ like He initialisation [44], which has been optimised for standard NNs and hence standard deep ensembles) for Figure 6.

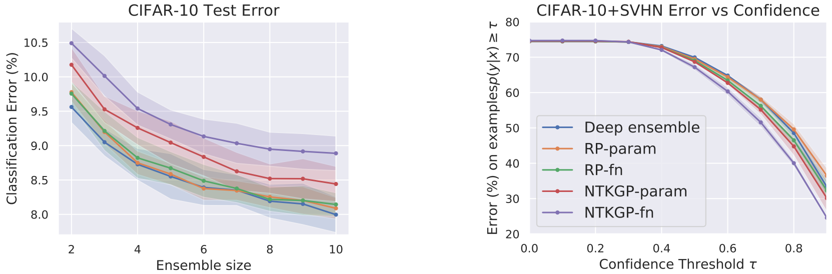

Figure 6: Figure 5 but where regression target scale κ is constant across ensemble methods and set to match the second moment of the NTK on the training set at initialisation. Error bars correspond to 5 independent runs.

In Figure 6 we see a different results to Figure 5, as here our NTKGP methods suffer slightly on in-distribution performance but also outperform the baselines methods on out-of-distribution detection. This highlights the importance of the regression target scale when considering classification tasks, and moreover reflects a general theme in our experiments of the trade-off between more aggressive predictions (that tend to perform better on in-distribution) and more conservative predictions (that tend to perform better on out-of-distribution). In our classification methodology, larger κ values tend to lead to more confident predictions. We point out that this is an issue that affects all ensemble methods and is not limited to our Bayesian ensembles.

I Experimental Details & additional plots

I.1 Toy 1d example

We set ensemble size $K = 20$, and train on full batch GD with learning rate 0.001 for 50,000 iterations under standard parameterisation in Neural Tangents [38], with $\sigma_W = 1.5$ & $\sigma_b = 0.05$, for σ_W, σ_b defined as in Appendix A. In Figure 7 we evaluate the impact of the ensemble size on this toy problem for different ensemble methods. We find that, of the two methods that approximate the analytic NTKGP mean predictor (c.f. Table 1), the approximation of the analytic mean predictor for NTKGP-param degrades compared to RP-param at small ensemble sizes, although the predictive uncertainties are well matched even at small ensemble sizes. The degradation in mean predictor is unsurprising as there is more (untrainable) noise in the initialised NTKGP baselearners. One simple possible solution to this problem, which we leave for future work, is to use separate baselearners for the mean and uncertainty predictions, like in Ciosek et al. [26].

I.2 Flight Delays

Our baselearners are MLPs with 4 hidden layers, 100 hidden units per layer and ReLU activations, and we use standard parameterisation with $\sigma_W = 1$ & $\sigma_b = 0.05$, and choose ensemble size $K = 5$. We train for 10 epochs with learning rate 0.001, batch size 100 and Adam [60]. For all experiments, all ensemble methods apart from standard deep ensembles [11] are L^2 regularised according to Appendix D, with weight decay strength set to 10^{-4} for standard deep ensembles.

We use a validation set of size 50k that is sampled uniformly from the training set of size 700k, and early stop baselearner NNs based on validation set loss. Inputs and targets are standardised so that the training data is zero mean and unit variance.

I.3 MNIST vs. NotMNIST

For all image classification experiments, we use a $90-10\%$ split for the train-validation sets needed for temperature scaling.

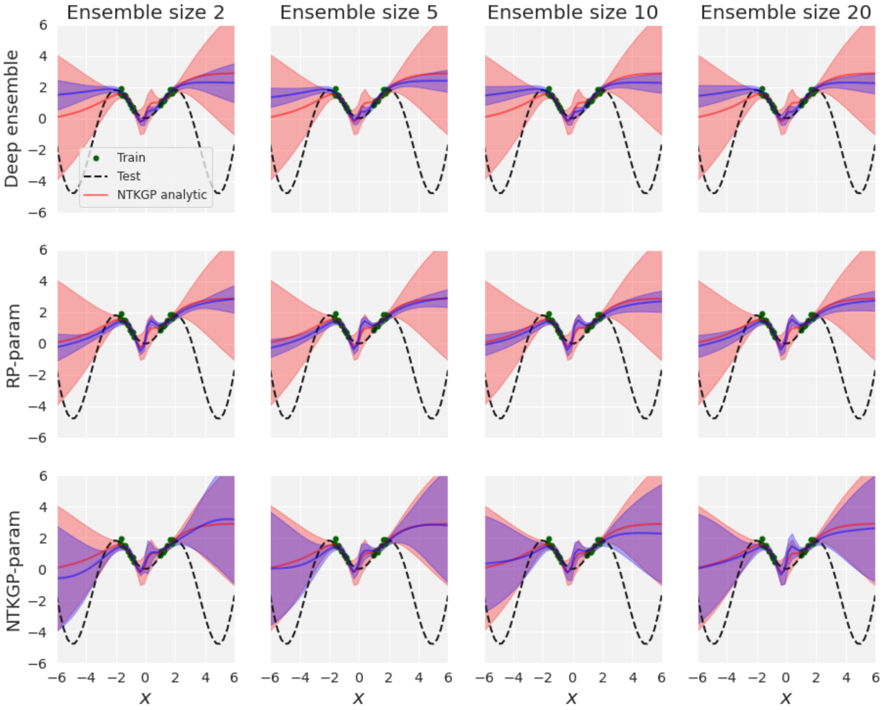

Figure 7: Comparison between ensemble methods (blue) and the analytic NTKGP posterior as the ensemble size is varied on toy example.

Baselearners are MLPs with 2-hidden layers, 200 hidden units per layer and ReLU activations. We standardise data to have mean 0 and standard deviation 1 across flattened pixels.

For all ensemble methods, we use standard parameterisation with fixed bias standard deviation $\sigma_b = 0.05$, observation noise $\sigma = 0.1$ and tune weight variance σ_W^2 on a small linear scale around $\sigma_W^2 = 2$. We set observation noise $\sigma = 0.1$ for . We train for 20 epochs with batch size 100, learning rate 0.001 and Adam [60]. We do not early stop for any classification experiment, and use the final trained baselearners throughout.

For the analytic NTKGP results, we use the NTK in NTK parameterisation, and use the same observation noise and bias variance as for ensemble methods. However, we fix $\sigma_W^2 = 2$ and also do not tune target scale κ (set to the base value described in Appendix A) due to computational resources. We also use only half the test sets both for MNIST and NotMNIST due to resource requirements, keeping the ratios of test sizes consistent in order for the error versus confidence plot Figure 3 (right) to be comparable. To compute test and out-of-distribution predictions, having obtained the optimal temperature scale T^* and analytic NTKGP predictions in logit space, $p(\cdot|\mathcal{X}, \mathcal{Y})$, we approximate the softmax class probability predictions: $\int \text{softmax}(z/T^*)p(dz|\mathcal{X}, \mathcal{Y})$, by a Monte Carlo ensemble approximation with 100 samples.

For all classification ensemble methods, we temperature scale on validation cross entropy for 5 epochs with batch size 100 and learning rate 0.1, whereas for analytic NTKGP we temperature scale for 1000 epochs on full batch size 6000. Like above, we approximate the analytic NTKGP validation predictions (for temperature scaling) by a Monte Carlo ensemble, this time of size 10. We found the various temperature scaling training hyperparameter considerations here to be unimportant to achieve convergence, due to the fact that the temperature scale is a scalar value.

I.4 CIFAR-10 vs SVHN

Baselearners are Myrtle-10 CNNs [40] with 100 channel width and ReLU activations. We use $\sigma_b = 0.01$ and set observation noise $\sigma = 0.1$. Like for MNIST we tune σ_W^2 on a small linear scale around $\sigma_W^2 = 2$. We train using SGD, with momentum parameter 0.9, for 100 epochs and learning rate 0.001, which is decayed to 0.0002 after 80 epochs. In the first 5 epochs we raise the learning rate in linear increments from 0.0001 to 0.001. We use batch size 125. During training we apply random crops and horizontal flips before standardisation. We do not compare to the analytic NTKGP for the Myrtle-10 CNN due to resource requirements.

Figure 8 displays entropy histograms for ensembles trained on CIFAR-10 and tested on in distribution CIFAR-10 test data and out-of-distribution SVHN test data, corresponding to the same experiments as in Figure 5. As we can see, there is a much less noticeable difference between ensemble methods compared to the simpler MNIST vs NotMNIST case.

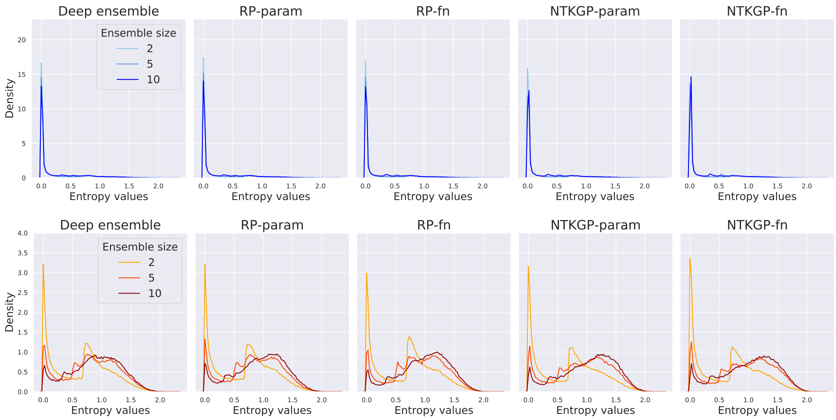

Figure 8: Histograms of predictive entropy on CIFAR-10 (top) and SVHN (bottom) test sets for different ensemble methods and for different ensemble sizes.