CONSERVATIVE UNCERTAINTY ESTIMATION BY FITTING PRIOR NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Obtaining high-quality uncertainty estimates is essential for many applications of deep neural networks. In this paper, we theoretically justify a scheme for estimating uncertainties, based on sampling from a prior distribution. Crucially, the uncertainty estimates are shown to be conservative in the sense that they never underestimate a posterior uncertainty obtained by a hypothetical Bayesian algorithm. We also show concentration, implying that the uncertainty estimates converge to zero as we get more data. Uncertainty estimates obtained from random priors can be adapted to any deep network architecture and trained using standard supervised learning pipelines. We provide experimental evaluation of random priors on calibration and out-of-distribution detection on typical computer vision tasks, demonstrating that they outperform deep ensembles in practice.

1 INTRODUCTION

Deep learning has achieved huge success in many applications. In particular, increasingly often, it is used as a component in decision-making systems. In order to have confidence in decisions made by such systems, it is necessary to obtain good uncertainty estimates, which quantify how certain the network is about a given output. In particular, if the cost of failure is large, for example where the automated system has the capability to accidentally hurt humans, the availability and quality of uncertainty estimates can determine whether the system is safe to deploy at all (Carvalho, 2016; Leibig et al., 2017; Michelmore et al., 2018). Moreover, when decisions are made sequentially, good uncertainty estimates are crucial for achieving good performance quickly (Bellemare et al., 2016; Houthooft et al., 2016; Ostrovski et al., 2017; Burda et al., 2018).

Because any non-Bayesian inference process is potentially sub-optimal (De Finetti, 1937), an important theoretical desideratum for a method of obtaining uncertainty estimates is that it should be relatable to Bayesian inference with a useful prior. Deep ensembles (Lakshminarayanan et al., 2017) and Monte-Carlo dropout (Gal & Ghahramani, 2016), the two most popular methods available for uncertainty estimation in deep networks today, struggle with this requirement. While deep ensembles can be related (Rubin, 1981) to Bayesian inference in settings where the individual ensembles are trained on subsets of the data, this is not how they are used in practice. In order to improve data efficiency, all ensembles are typically trained using the same data (Lakshminarayanan et al., 2017), resulting in a method which does not have a theoretical justification. Moreover, deep ensembles can give overconfident uncertainty estimates in practice. On the other hand, while Monte-Carlo dropout can be interpreted as an approximation to a certain form of Bayesian inference (Gal & Ghahramani, 2016), questions have been raised over the use of singular KL-divergences in the original proof (Hron et al., 2018). In practice, MC dropout can give arbitrarily overconfident estimates (Foong et al., 2019). A third category of approaches, known as Bayesian Neural Networks (Blundell et al., 2015; Neal, 1996), maintains a distribution over the weights of the neural network. These methods typically have a Bayesian justification, but training them is both difficult and relies on procedures different from standard supervised learning, for which most Machine Learning pipelines are optimized in practice. While variants of BNNs that use an optimization process similar to standard supervised training have been proposed (Welling & Teh, 2011), tuning them is hard and achieving a good approximation to the posterior is difficult (Brosse et al., 2018).

In this work, we use another way of obtaining uncertainties for deep networks, based on fitting random priors (Osband et al., 2018; 2019; Burda et al., 2018). To obtain the uncertainty estimates, we first train a predictor network to fit a prior (Figure 1, top two plota). Faced with a novel input point, we obtain an uncertainty (Figure 1, bottom plot) by measuring the error of the predictor network against this pattern. Intuitively, these errors will be small close to the training points, but large far from them. The patterns themselves are drawn from randomly initialized (and therefore untrained) neural networks. While this way of estimating uncertainties was known before (Osband et al., 2019) and was used to great effect in the context of Reinforcement Learning (Burda et al., 2018), it did not have a theoretical justification beyond the Bayesian linear regression setting.

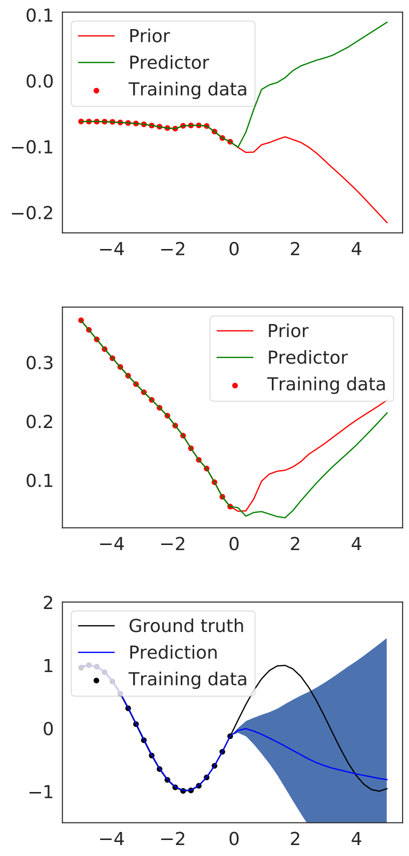

Figure 1: On top, two predictors (green) were trained to fit two randomly-generated priors (red). On the bottom, we obtain uncertainties from the difference between predictors and priors.

Contributions We justify how the process of fitting random priors can be used to produce estimates in the uncertainty of the output *of neural networks with any architecture*. In particular, we show in Lemma 1 and Proposition 1 that these uncertainty estimates are *conservative*, meaning they are never more certain than a Bayesian algorithm would be. Moreover, in Proposition 2 we show concentration, i.e. that the uncertainties become zero with infinite data. We use this theory to formulate practical insights about the training process, justifying the use of a predictor network with a more flexible architecture than the prior. Empirically, we evaluate the calibration and out-of-distribution performance of our uncertainty estimates on typical computer vision tasks, showing a practical benefit over deep ensembles.

2 Preliminaries

Stochastic processes We are going to reason about uncertainty within the formal framework of stochastic processes. We now introduce the required notations. A stochastic process is a collection of random variables $\{f(x)\}$. We consider processes where $x \in \mathbb{R}^K$ and the random-variable $f(x)$ takes values in $\mathbb{R}^M$. A stochastic process has exchangeable outputs if the distribution does not change when permuting the M entries in the output vector. Allowing a slight abuse of notation, we denote the finite-dimensional distribution of the process $\{f(x)\}$ for the set $X = \{x_i\}_{i=1,\dots,N}$ as $f(x_1, \dots, x_N) = f(X)$. In practice, the finite-dimensional distribution reflects the idea of restricting the process to points $x_1, \dots, x_N$ and marginalizing over all the other points. Inference can be performed on stochastic processes similarly to probability distributions. In particular, we can start with some prior process $\{f(x)\}$, observe a set of N training points $X = \{x_i\}_{i=1,\dots,N}$ and labels $y = \{y_i\}_{i=1,\dots,N}$ and then consider the posterior process $\{f_{Xy}(x)\}$, whose finite-dimensional distributions are given by $f_{Xy}(x_1^\star \dots x_{N'}^\star) = f(x_1^\star \dots x_{N'}^\star | x_1, \dots, x_N, y_1, \dots, y_N)$ for any set of testing points $x_1^\star \dots x_{N'}^\star$. We use subscripts to denote conditioning on the dataset throughout the paper.

Gaussian processes (GPs) A stochastic process is Gaussian (Williams & Rasmussen, 2006), if all its finite-dimensional distributions are Gaussian. The main advantage of GPs is that the posterior process can be expressed in a tractable way. GPs are often used for regression, where we are learning an unknown function[1] $\phi : \mathbb{R}^K \rightarrow \mathbb{R}$ from noisy observations. Since a Gaussian distribution is com-

[1]We depart from standard notation, which uses f, because we will be using f to denote a sample from the prior process.

pletely identified by its first two moments, a GP can be defined by a mean function and a covariance function. Formally, the notation $\mathcal{GP}(\mu, k)$ refers to a GP with with mean function $\mu : \mathbb{R}^K \rightarrow \mathbb{R}$, a positive-definite kernel function $k : \mathbb{R}^K \times \mathbb{R}^K \rightarrow \mathbb{R}$. GPs can be used to model two kinds of uncertainty: epistemic uncertainty, which reflects lack of knowledge about unobserved values of ϕ and aleatoric uncertainty, which reflects measurement noise. When performing regression, we start with a zero-mean prior $\mathcal{GP}(0, k)$ and then observe N training points $X = \{x_i\}_{i=1,\dots,N}$ and labels $y = \{y_i\}_{i=1,\dots,N}$ where $y_i = \phi(x_i) + \epsilon_i$. Here, the i.i.d. random variables $\epsilon_i \sim \mathcal{N}(0, \sigma_A^2)$ model the aleatoric noise. We obtain the posterior process on $\mathcal{GP}(\mu_{Xy}, k_X)$. For GPs, the mean and covariance of the posterior GP on y evaluated at $x_\star$ can be expressed as

$$\mu_{Xy}(x_\star) = k_\star^\top (K + \sigma_A^2 I)^{-1} y \quad \text{and} \tag{1}$$

$$\sigma_X^2(x_\star) \triangleq k_X(x_\star, x_\star) + \sigma_A^2 = k_{\star\star} - k_\star^\top (K + \sigma_A^2 I)^{-1} k_\star + \sigma_A^2. \tag{2}$$

In particular, the posterior covariance does not depend on y. In the formula above, we use the kernel matrix $K \in \mathbb{R}^N \times \mathbb{R}^N$ defined as $K_{ij} = k(x_i, x_j)$, where x_i and x_j are in the training set. We also use the notation $k_\star : \mathbb{R}^N$ for the vector of train-test correlations $\{k_\star\}_i = k(x_i, x^\star)$, where x_i is in the training set and $k(x^\star, x^\star)$ is similarly defined. The shorthand $\sigma_X^2(x_\star)$ introduced in equation 2 denotes the posterior covariance at a single point.

Bayes Risk We are often interested in minimizing the Mean Squared Error $\mathrm{E}_f\left[(f(x_\star) - w)^2\right]$, where $x_\star$ is a given test point and w is a variable we are allowed to adjust. A known result of Bayesian decision theory (Robert, 2007; Murphy, 2012) is that the minimizer of the MSE is given by the expected value of f, i.e.

$$\arg\min_w \mathrm{E}_f\left[(f(x_\star) - w)^2\right] = \mathrm{E}_f\left[f(x_\star)\right]. \tag{3}$$

Equation 3 holds for any stochastic process f, including when f is a posterior process obtained by conditioning on some dataset. A consequence of equation 3 is that it is impossible to obtain a MSE lower than the one obtained by computing the posterior mean of f.

3 Estimating Uncertainty from Random Priors

Intuition Uncertainties obtained from random priors have an appealing intuitive justification. Consider the networks in the top part of Figure 1. We start with a randomly initialized prior network, shown in blue. Whenever we see a datapoint, we train the predictor network (green) to match this prior. Uncertainties can then be obtained by considering the squared error between the prior and the predictor at a given point. An example uncertainty estimate is shown as a shaded blue area in the bottom of Figure 1. While it may at first seem that the squared error is a poor measure of uncertainty because it can become very small by random chance, we show in Section 4.1 that this is very unlikely. In Section 4.2, we show that this error goes down to zero as we observe more data. Similarly to GP inference, uncertainty estimation in our framework does not depend on the regression label. The prediction mean (blue curve in the bottom part of Figure 1) is obtained by fitting a completely separate neural network. In section 6, we discuss how this framework differs from deep ensembles (Lakshminarayanan et al., 2017).

Algorithm 1 Training the predictors.

```
function TRAIN-UNCERTAINTIES($X$)
    for $i = 1 \dots B$ do
        $f^i \sim \{f(x)\}$                      ▷ random prior
        $h_{Xf^i} \leftarrow$ FIT($X, f^i(X)$)
    end for
    return $f_i, h_{Xf^i}$
end function

function FIT($X, f^i(X)$)
    $L(h) \triangleq \sum_{x \in X} \|f^i(x) - h(x)\|^2$
    $h_{Xf^i} \leftarrow$ OPTIMIZE($L$)          ▷ SGD or similar
    return $h_{Xf^i}$              ▷ return trained predictor
end function
```

Prior The process of obtaining the network uncertainties involves randomly initialized prior networks, *which are never trained*. While this may at first appear very different from they way deep learning is normally done, these random networks are a crucial component of our method. We show in Section 4.1 that the random process that corresponds to initializing these networks can be interpreted as a *prior* of a Bayesian inference procedure. A prior conveys the information about how the individual data points are related. The fact that we are using random networks is a very safe choice. In fact, whenever deep learning is used in practice, with or without uncertainty estimates, we are implicitly making the assumption that the network architecture is appropriate for the dataset anyway.

Algorithm The process of training the predictor networks is shown in Algorithm 1. The function TRAIN-UNCERTAINTIES first generates random priors, i.e. neural networks with random weights. In our notation, it corresponds to sampling functions from the prior process $\{f(x)\}$. These priors, evaluated at points from the dataset $X = \{x_i\}_{i=1,\dots,N}$ are then used as labels for supervised learning, performed by the function FIT. After training, when we want to obtain an uncertainty on ϕ at a given test point $x_\star$, we use the formula

$$\hat{\sigma}^2(x_\star) = \max(0, \hat{\sigma}^2_m(x_\star) + \beta\hat{\sigma}^2_s(x_\star) - \sigma^2_A). \tag{4}$$

Here, the quantity $\hat{\sigma}^2_m$ is the sample mean of the squared error. We will show in Section 4 that it is an unbiased estimator of a variable that models the uncertainty. On the other hand, $\hat{\sigma}^2_s$ is the sample-based estimate of the standard deviation of squared error across bootstraps, needed to quantify our uncertainty about *what the uncertainty is*. The hyper-parameter β controls the degree to which this uncertainty is taken into account. Formally, the quantities are defined as

$$\hat{\sigma}^2_m(x_\star) \triangleq \textstyle\sum_{i=1}^{B} \frac{1}{MB}\|f(x_\star) - h_{Xf_i}(x_\star)\|^2, \tag{5}$$

$$\hat{\sigma}^2_s(x_\star) \triangleq \sqrt{\textstyle\sum_{i=1}^{B} \frac{1}{B}(\hat{\sigma}^2_m(x_\star) - \|f(x_\star) - h_{Xf_i}(x_\star)\|^2)^2}. \tag{6}$$

In the above equations, B is the number of prior functions and each prior and predictor network has M outputs. We defer the discussion of details of network architecture to Section 5. Our experiments (Section 7) show that it is often sufficient to use $B = 1$ in practice.

4 THEORETICAL RESULTS

In Section 3, we introduced a process for obtaining uncertainties in deep learning. We now seek to provide a formal justification. We begin by setting the notation. We define the expected uncertainties as

$$\tilde{\sigma}^2_m(x_\star) \triangleq \mathrm{E}_f\left[\hat{\sigma}^2_m(x_\star)\right] = \mathrm{E}_f\left[\tfrac{1}{M}\|f(x_\star) - h_{Xf}(x_\star)\|^2\right]. \tag{7}$$

In other words, $\tilde{\sigma}^2_m$ is the expected version of the sample-based uncertainties $\hat{\sigma}^2_m(x_\star)$ introduced in equation 5. Since Bayesian inference is known to be optimal (De Finetti, 1937; Jaynes, 2003; Robert, 2007), the most appealing way of justifying uncertainty estimates $\tilde{\sigma}^2_m$ and $\hat{\sigma}^2_m$ is to relate them to a Bayesian posterior. We do this in two stages. First, in Section 4.1, we prove that the obtained uncertainties are larger than ones arrived at by Bayesian inference. This means that our uncertainties are *conservative*, ensuring that our algorithm is never more certain than it should be. Next, in Section 4.2, we show that uncertainties concentrate, i.e., they become small as we get more and more data. These two properties are sufficient to justify the use of our uncertainties in many applications.

4.1 UNCERTAINTIES FROM RANDOM PRIORS ARE CONSERVATIVE

From the point of view of safety, it is preferable to overestimate the ground truth uncertainty than to underestimate it. We now show that this property holds for uncertainties obtained from random priors. First, we justify conservatism for the expected uncertainty $\tilde{\sigma}^2_m$ defined in equation 7 and then for the sampled uncertainty $\hat{\sigma}^2_m$ defined in equation 5.

Amortized Conservatism We first consider a weak form of this conservatism, which we call amortized. It guarantees that $\tilde{\sigma}^2_m$ is never smaller than the average posterior uncertainty across labels sampled from the prior. Formally, amortized conservatism holds if for any test point $x_\star$ we have

$$\tilde{\sigma}^2_m(x_\star) \geq \mathrm{E}_{f(X)}\left[\sigma^2_{Xf}(x_\star)\right]. \tag{8}$$

Here $\sigma^2_{Xf}(x_\star)$ corresponds to the second moment of the posterior process $\{f_{Xf}(x)\}$. We will introduce a stronger version of conservatism, which does not have an expectation on the right-hand side, later in this section (eq. 12) . For now, we concentrate on amortized conservatism, which holds under very general conditions.

Lemma 1. *For any function $h : \mathbb{R}^{N\times(K+1)} \to \mathbb{R}^M$, for any test point $x_\star \in \mathbb{R}^K$ and for any stochastic process $\{f(x)\}_{x\in\mathbb{R}^K}$ with all second moments finite and exchangeable outputs, it holds that*

$$\tilde{\sigma}^2_m(x_\star) = \mathrm{E}_{f(X)}\left[\sigma^2_{Xf}(x_\star) + \tfrac{1}{M}\|\mu_{Xf}(x_\star) - h_{Xf}(x_\star)\|^2\right]. \tag{9}$$

Strict Conservatism Lemma 1 holds for any prior process $\{f(x)\}$. However, the prior process used by Algorithm 1 is not completely arbitrary. The fact that prior samples are obtained by initializing neural networks with independently sampled weights gives us additional structure. In fact, it can be shown that randomly initialized neural networks become close to GPs as the width of the layers increases. While the original result due to Neal (1996) held for a simple network with one hidden layer, it has been extended to a wide class of popular architectures, including to CNNs and RNNs of arbitrary depth (Matthews et al., 2018; Lee et al., 2018; Novak et al., 2019; Williams, 1997; Le Roux & Bengio, 2007; Hazan & Jaakkola, 2015; Daniely et al., 2016; Garriga-Alonso et al., 2019). Recently, it has been shown to hold for a broad class of functions trainable by gradient descent (Yang, 2019). While the precise statement of these results involves technicalities which fall beyond the scope of this paper, we recall the upshot. For a family of neural networks $\{f^W(x)\}$, where the weights are sampled independently and W is the width of the hidden layers, there exist a limiting kernel function k_∞ such that

$$\lim_{W\to\infty}[\{f^W(x)\}] = \mathcal{GP}(0, k_\infty, 0). \tag{10}$$

In other words, as the size of the hidden layers increases, the stochastic process obtained by initializing networks randomly converges in distribution to a GP. In the context of our uncertainty estimates, this makes it is reasonable for W large enough to consider the prior to be a GP. We stress that the GP assumption has to hold *only for the prior network*, which is never trained. We do not make any assumptions about connections between the predictor training process and GPs. This allows us to state 1, which is proved in the appendix.

Proposition 1 (Strict Conservatism in Expectation)**.** *Assume that f is a GP. Then for any function $h : \mathbb{R}^{N\times K} \to \mathbb{R}^M$, we have*

$$\tilde{\sigma}^2_m(x_\star) = \sigma^2_X(x_\star) + \underbrace{\mathrm{E}_{f(X)}\left[\tfrac{1}{M}\|\mu_{Xf}(x_\star) - h_{Xf}(x_\star)\|^2\right]}_{\geq 0}. \tag{11}$$

Moreover, equality holds if and only if $h_{Xf}(x_\star) = \mu_{Xf}(x_\star)$.

Proposition 1 is useful because it implies that uncertainty estimates are strictly conservative in the sense of

$$\hat{\sigma}^2(x_\star) \geq \sigma^2_X(x_\star). \tag{12}$$

This statement is stronger than the amortized conservatism in equation 8. Intuitively, equation 12 can be interpreted as saying that our uncertainty estimates are never too small. This confirms the intuition expressed by Burda et al. (2018) that random priors do not overfit.

Finite Bootstraps Lemma 1 above shows conservatism for expected uncertainties, i.e. $\tilde{\sigma}^2_m$ introduced in equation 8. However, in practice we have to estimate this expectation using a finite number of samples, and use the sampled uncertainties $\hat{\sigma}^2_m$ defined in equation 5. We now state a conservatism guarantee that holds even in the case of finite samples.

Lemma 2. *Assume that the random variable $\hat{\sigma}^2_m(x_\star)$ has finite variance upper bounded by σ^2_S. With probability $1-\delta$, we have $\hat{\sigma}^2_m(x_\star) + \frac{1}{\sqrt{\delta}}\sigma_S \geq \tilde{\sigma}^2_m(x_\star)$.*

In practice, Lemma 2 says that the uncertainty estimates $\hat{\sigma}^2_m(x_\star) + \frac{1}{\sqrt{\delta}}\sigma_S$ are conservative with high probability. However, applying the Lemma requires the knowledge of σ_S. We now provide an upper bound.

Lemma 3. *Assume that the GP $\{f(x)\}$ is zero mean with exchangeable outputs and the function h_{Xf} takes values in $[-U, U]^M$. Assume that permuting the outputs of f produces the same permutation in the outputs of h_{Xf}. With probability $1-\delta$, we have*

$$\sigma_S^2 \leq \frac{1}{B}\left(4U^4 + (\hat{\sigma}_0^2(x_\star))^2 C\right). \tag{13}$$

Here, $\hat{\sigma}_0^2(x_\star)$ is a sample-based estimate of the prior variance obtained with B_0 samples and $C = \left(\frac{B_0-1}{\chi_I^2(\delta)}\right)^2$, where χ_I^2 denotes the inverse CDF of the Chi-Squared distribution with $B_0 - 1$ degrees of freedom.

Combining Lemmas 2 and 3 gives us a formal conservatism guarantee for the setting of finite bootstraps. In cases where conservatism is desired, but not absolutely essential, we can avoid the torturous calculation of Lemma 3 and replace σ_S^2 with its sample-based estimate $\hat{\sigma}_s^2(x_\star)$, defined in equation 5. In this case, the conservatism guarantee is only approximate. We have done this to obtain equation 4, used by the algorithm in practice.

4.2 Uncertainties from Random Priors Concentrate

While the conservatism property in Proposition 1 is appealing, in order for the uncertainty estimates to be useful, we need to make sure that they concentrate, i.e. that the uncertainties $\hat{\sigma}^2$ become small with more data. We obtain this result by assuming that the class of neural networks being fitted is Lipschitz-continuous and bounded. Intuitively, By assumption of Lipschitz continuity, the predictors h_{Xf} cannot behave very differently on points from the training and test sets, since both come from the same data distribution. To show this formally, we use standard Rademacher tools to obtain a bound on the expected uncertainty in terms of the squared error on the training set. This process is formalized in Proposition 2.

Proposition 2. *If the training converges, i.e. the training loss $\frac{1}{MN}\sum_{i=1}^{N}\|f(x_i) - h_{Xf}(x_i)\|^2 = \sigma_A^2$ for arbitrarily large training sets, then assuming the predictors h_{Xf} are bounded and Lipschitz continuous with constant L, then under technical conditions the uncertainties concentrate, i.e. $\hat{\sigma}^2(x_\star) \to 0$ as $N \to \infty$ and $B \to \infty$ with probability 1.*

The proof and the technical conditions are given in Appendix C. Proposition 2 assumes that the training error is zero for arbitrarily large training sets, which might at first seem unrealistic. We argue that this assumption is in fact reasonable. The architecture of our predictor networks (Figure 2, right diagram) is a superset of the prior architecture (Figure 2, left diagram), guaranteeing the existence of weight settings for the predictor that make the training loss zero. Recent results on deep learning optimization (Du et al., 2019; Allen-Zhu et al., 2019) have shown that stochastic gradient descent can in general be expected to find representable functions.

5 Practical Conclusions from the Theory

We now re-visit the algorithm we defined in Section 3, with the aim of using the theory above to obtain practical improvements in the quality of the uncertainty estimates.

Architecture and Choosing the Number of Bootstraps Our conservatism guarantee in Proposition 1 holds for *any* architecture for the predictor h_{Xf}. In theory, the predictor could be completely arbitrary and does not even have to be a deep network. In particular, there is no formal requirement for the predictor architecture to be the same as the prior. On the other hand, to show concentration in Proposition 2, we had to ensure that the prior networks are representable by the predictor. In practice, we use the architecture shown in Figure 2, where the predictor mirrors the prior, but has additional layers, giving it more representational power. Moreover, the architecture requires choosing the number of bootstraps B. Our experiments in Section 7 show that even using $B = 1$, i.e. one bootstrap, produces uncertainty estimates of high quality in practice.

Modeling Epistemic and Aleatoric Uncertainty Corollary 1 and Proposition 2 hold for *any* Gaussian Process. When choosing the process appropriately, the random prior framework can be used to model both epistemic and aleatoric uncertainty. The amount of aleatoric uncertainty can

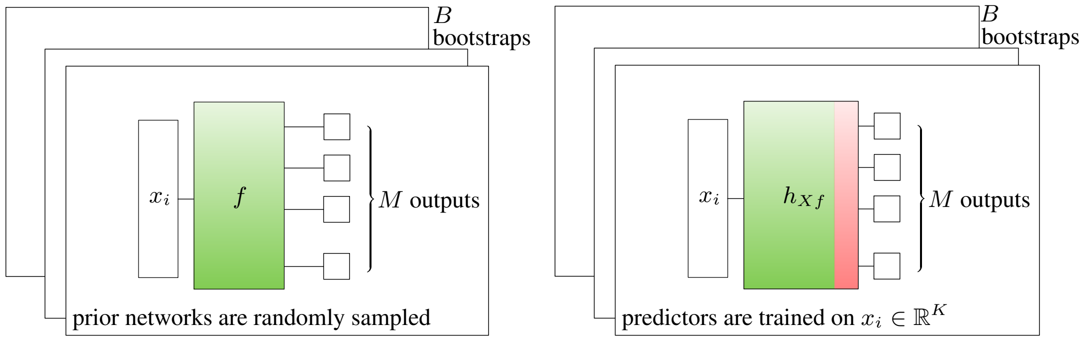
B bootstraps
x_i
f
M outputs
prior networks are randomly sampled
B bootstraps
x_i
h_{Xf}
M outputs
predictors are trained on $x_i \in \mathbb{R}^K$

Figure 2: Architecture of the random prior networks f and predictor networks h_{Xf}. The predictor networks h_{Xf} typically share the same architectural core, but have additional layers relative to the prior networks.

be adjusted by choosing the right prior. For example, we can use the prior process $\{f(x)\} = \{n(x) + \epsilon(x)\sigma_A^2\}$. Samples from the epistemic component $\{n(x)\}$ are obtained by randomly initializing neural networks, while samples from $\{\epsilon(x)\sigma_A^2\}$ are obtained by sampling from $\mathcal{N}(0, \sigma_A^2)$ at each x independently.

6 Prior Work

Randomized Prior Functions (RPFs) Our work was inspired by, an builds on, Randomised Prior Functions (Osband et al., 2019; 2018), but it is different in two important respects. First, the existing theoretical justification for RPFs only holds for Bayesian linear regression (Osband et al., 2018, equation 3) with non-zero noise[2] added to the priors. In contrast, our results are much more general and hold *for any deep network* with or without added aleatoric noise. Second, we are targeting a slightly different setting. While RPFs were designed as a way of sampling functions from the posterior, we provide estimates of posterior uncertainty at a given test point. Our algorithm is based on the work by Burda et al. (2018), who applied RPFs to exploration in MDPs, obtaining state-of-the art results, but without justifying their uncertainty estimates formally. Our paper provides this missing justification, while also introducing a way of quantifying our error in estimating the uncertainty itself. Moreover, since Burda et al. (2018) focused on the application of RPFs to Reinforcement Learning, they only performed out-of-distribution evaluation on the relatively easy MNIST dataset (LeCun, 1998). In contrast, in Section 7 we evaluate the uncertainties on more complex vision tasks.

Deep Ensembles The main competing approach for obtaining uncertainties in deep learning are deep ensembles (Lakshminarayanan et al., 2017). Building on the bootstrap (Efron & Tibshirani, 1994), deep ensembles maintain several bootstraps of a deep network and quantify epistemic uncertainty by measuring how their outputs vary. Deep ensembles have two major shortcomings. First, they do not have theoretical support in the case when all the members of the ensemble are trained on the same data, which is how they are used in practice (Lakshminarayanan et al., 2017). Second, because they use representations trained on regression labels, they tend to learn similar representations for different inputs with similar labels, which can lead to over-fitting the uncertainty estimates. We show empirically in section 7 that deep ensembles can give overconfident uncertainty estimates, particularly on points that have the same label as points in the training set. On the other hand, our method does not exhibit such overconfidence, while taking a similar time to train.

Dropout In cases where it is not economical to train more than one network, uncertainties can be obtained with dropout (Srivastava et al., 2014; Gal & Ghahramani, 2016). Dropout has been interpreted as an approximation to a certain form of Bayesian inference (Gal & Ghahramani, 2016). However, the justification of dropout as a variational approximation has been recently called into question due to the existence of undefined KL-divergences in the original proof (Hron et al., 2018).

[2]The existing justification of RPFs (Osband et al., 2019, Section 5.3.1) involves a division by the noise variance.

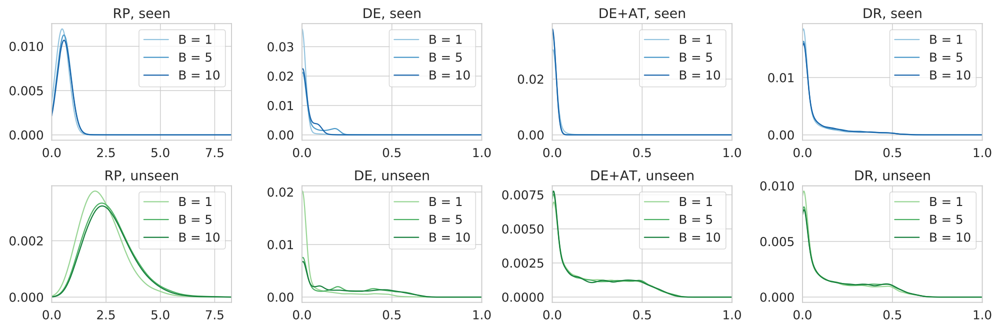

Figure 3: Distribution of uncertainty estimates for various algorithms. Top row shows seen data, bottom row shows unseen data from CIFAR-10.

Moreover, since dropout implicitly approximates non-Gaussian weight distribution with Gaussians, it exhibits spurious patterns in the obtained uncertainties, which can lead to arbitrarily overconfident estimates (Foong et al., 2019). In contrast, due to the conservatism property, random priors avoid such overconfidence.

7 Experiments

Encouraged by the huge empirical success of random priors in Reinforcement Learning (Burda et al., 2018), we wanted to provide an evaluation in a more typical supervised learning setting. We tested the uncertainties in two ways. First, we investigated *calibration*, i.e. whether we can expect a higher accuracy for more confident estimates. Next, we checked whether the uncertainties can be used for out-of-distribution detection. We compared to two competing approaches for uncertainty detection: deep ensembles (Lakshminarayanan et al., 2017) and spatial concrete dropout (Gal et al., 2017). The same ResNet architecture served as a basis for all methods. Details of the implementation are provided in Appendix A.

Out-Of-Distribution Detection We evaluated the uncertainty estimates on out-of-distribution detection. To quantify the results, we evaluated the area under the ROC curve (AUROC) for the task of deciding whether a given image comes from the same distribution or not. All methods were trained on four classes from the CIFAR-10 (Krizhevsky et al., 2009) dataset. We then tested the resulting networks on images from withheld classes and on the SVHN dataset (Netzer et al., 2011), which contains completely different images. Results are shown in Table 1. Considering the statistical errors (see Appendix B), random priors performed slightly better than deep ensembles with adversarial training for $B = 1$ and about the same for $B = 10$. Dropout performed worse, but was cheaper to train. In order to gain a more finely-grained insight into the quality of the uncertainties, we also show uncertainty histograms in Figure 3. The figure shows the distribution of uncertainty estimates for seen data (top row) vs. unseen data (bottom row) for bootstrap sizes $B = \{1, 5, 10\}$. The main conclusion is that uncertainties obtained from random priors are already well-separated with $B = 1$, while deep ensembles need more bootstraps to achieve the full separation between test and train examples.

	RP	DE	DE +AT	DR
B=1				
Train v. cat/deer	**0.99**	0.83	0.96	0.81
Train v. vehicles	**1.00**	0.82	0.96	0.76
Train v. excluded	**1.00**	0.82	0.96	0.77
Train v. SVHN	**0.95**	0.88	**0.96**	0.86
B=10				
Train v. cat/deer	**1.00**	0.95	**0.99**	0.82
Train v. vehicles	**1.00**	0.92	0.98	0.78
Train v. excluded	**1.00**	0.93	0.98	0.79
Train v. SVHN	0.97	0.94	**0.99**	0.87

Table 1: Out-of-distribution AUROC for random priors (RP), deep ensembles (DE), deep ensembles with adversarial training (DE+AT) and spatial concrete dropout (DR). Estimated confidence intervals are provided in Appendix B.

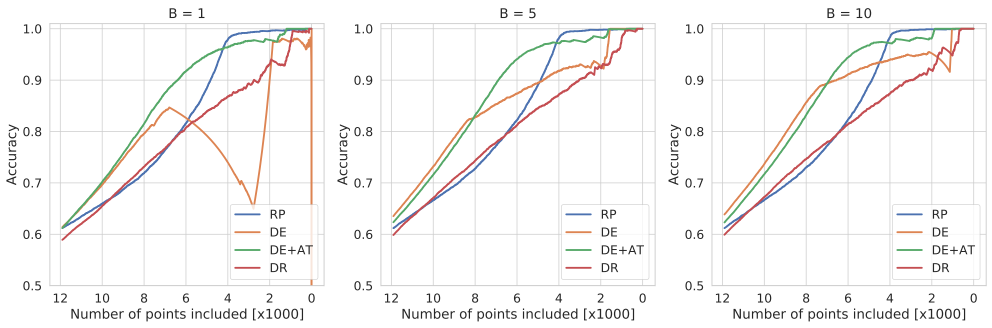

Figure 4: Calibration curves showing the relationship between uncertainty (horizontal axis) and accuracy (vertical axis) for $B = 1, 5, 10$ on CIFAR-10.

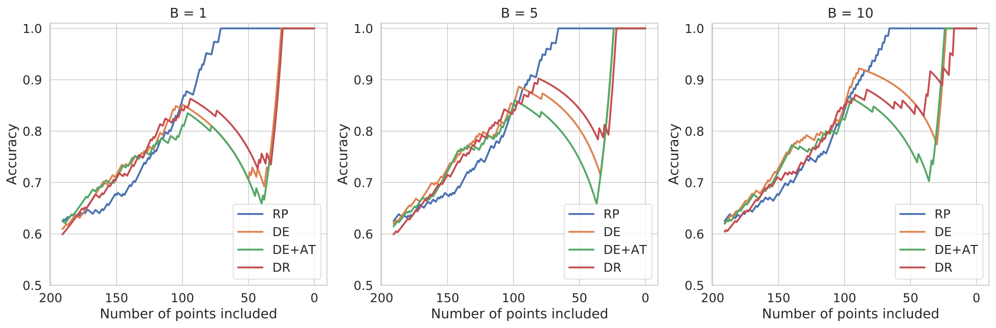

Figure 5: The relationship between uncertainty (horizontal axis) and accuracy (vertical axis) for $B = 1, 5, 10$ on a subset of 75 samples from CIFAR-10. In well-calibrated models, accuracy increases as uncertainty declines.

Calibration Good uncertainty estimates have the property that accuracy increases as we become more certain, a property known as *calibration*. We measured it by evaluating average accuracy on the subset of images with uncertainty smaller than a given value. We trained on four classes from the CIFAR-10 (Krizhevsky et al., 2009) dataset. We then tested the resulting networks on the whole dataset, which included both the seen and unseen classes. Results are shown in Figure 4. Ideally, in a calibrated method, these curves should be increasing, indicating that a method always becomes more accurate as it becomes more confident. In coarse terms, Figure 4 confirms that all methods except a degenerate deep ensemble with only one bootstrap are roughly monotonic. However, uncertainty estimates from random priors are more stable, showing montonicity on a finer scale as well as on a large scale. Interestingly, calibration improved only slightly when increasing the number of bootstraps B.

Subsampling Ablation In the previous experiment, we kept the architectural and optimization choices fixed across algorithms. This ensured a level playing field, but meant that we were not able to obtain zero training error on the predictor networks used by random priors. However, we also wanted to evaluate random priors in the setting of zero training error, which is assumed by our concentration result in Section 4.2. To do this, we used a smaller set of training images, while still keeping the network architecture the same. This allowed us to obtain complete convergence. Results of this ablation

	RP	DE	DE +AT	DR
B=1				
Train v. excluded	**1.00**	0.90	0.89	0.91
Train v. SVHN	**1.00**	0.95	0.94	0.97
B=10				
Train v. excluded	**1.00**	0.94	0.90	0.92
Train v. SVHN	**1.00**	0.97	0.95	0.97

Table 2: Out-of-distribution AUROC for the same models as above (see Tab. 1) on subsampled data. Numbers are accurate up to ± 0.01.

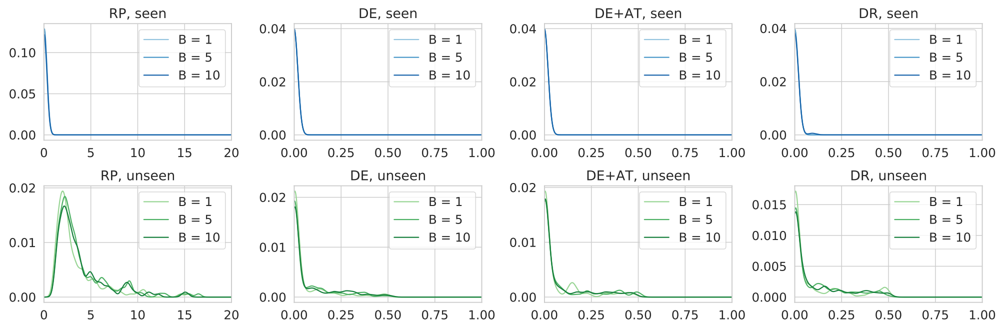

Figure 6: Distribution of uncertainty estimates for various algorithms. Top row shows seen data, bottom row shows unseen data from CIFAR-10, where we trained on a sample of 75 images from the training set.

are shown in Figures 5 and 6, as well as Table 2, analogous to our results on the full dataset presented above. In this sub-sampled regime, the random prior method easily outperformed competing approaches, showing better calibration (Fig. 5). The histograms in Figure 6 also demonstrate good separation between seen and unseen data. In the out-of-distribution benchmarks reported in Table 2, the random prior method has comfortably outperformed the baselines. While this training regime is not practical for real-life tasks, it demonstrates the potential performance of random priors when trained to full convergence.

Summary of experiments We have shown that uncertainties obtained from random priors achieve competitive performance with fewer bootstraps in a regime where the network architecture is typical for standard supervised learning workloads. Random priors showed superior performance in a regime where the predictors can be trained to zero loss.

8 Conclusions

We provided a theoretical justification for the use of random priors for obtaining uncertainty estimates in the context of deep learning. We have shown that the obtained uncertainties are conservative and that they concentrate for any neural network architecture. We performed an extensive empirical comparison, showing that random priors perform similarly to deep ensembles in a typical supervised training setting, while outperforming them in a regime where we are able to accomplish zero training loss for the predictors.

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 242–252, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/allen-zhu19a.html.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Nicolas Brosse, Alain Durmus, and Eric Moulines. The promises and pitfalls of stochastic gradient langevin dynamics. In *Advances in Neural Information Processing Systems*, pp. 8268–8278, 2018.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Ashwin Mark Carvalho. *Predictive Control under Uncertainty for Safe Autonomous Driving: Integrating Data-Driven Forecasts with Control Design*. PhD thesis, UC Berkeley, 2016.

Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pp. 2253–2261, 2016.

Bruno De Finetti. La prévision: ses lois logiques, ses sources subjectives. In *Annales de l'institut Henri Poincaré*, pp. 1–68, 1937.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pp. 1675–1685, 2019.

John Duchi. Probability bounds, 2009.

Bradley Efron and Robert J. Tibshirani. An Introduction to the Bootstrap. *SIAM Review*, 36(4):677–678, 1994. doi: 10.1137/1036171.

Andrew YK Foong, David R Burt, Yingzhen Li, and Richard E Turner. Pathologies of factorised gaussian and mc dropout posteriors in bayesian neural networks. *arXiv preprint arXiv:1909.00719*, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 1050–1059, 2016. URL `http://proceedings.mlr.press/v48/gal16.html`.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 3581–3590, 2017.

Adri Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bklfsi0cKm`.

Tamir Hazan and Tommi Jaakkola. Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*, 2015.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

Jiri Hron, Alexander G. de G. Matthews, and Zoubin Ghahramani. Variational bayesian dropout: pitfalls and fixes. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 2024–2033, 2018. URL `http://proceedings.mlr.press/v80/hron18a.html`.

Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

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

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Nicolas Le Roux and Yoshua Bengio. Continuous neural networks. In *Artificial Intelligence and Statistics*, pp. 404–411, 2007.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

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. URL `https://openreview.net/forum?id=B1EA-M-0Z`.

Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):17816, 2017.

Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5(Jun):669–695, 2004.

AGDG Matthews, M Rowland, J Hron, RE Turner, and Z Ghahramani. Gaussian process behaviour in wide deep neural networks. In *Proceedings of the 6th International Conference on Learning Representations.*, 2018.

Rhiannon Michelmore, Marta Kwiatkowska, and Yarin Gal. Evaluating uncertainty quantification in end-to-end autonomous driving control. *arXiv preprint arXiv:1811.06817*, 2018.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Radford M Neal. Bayesian learning for neural networks. *Phd Thesis*, 1996.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

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. URL `https://openreview.net/forum?id=B1g30j0qF7`.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 8617–8629, 2018.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. URL `http://jmlr.org/papers/v20/18-339.html`.

Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2721–2730. JMLR. org, 2017.

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

Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

Donald B Rubin. The bayesian bootstrap. *The annals of statistics*, pp. 130–134, 1981.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.

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)*, pp. 681–688, 2011.

Christopher KI Williams. Computing with infinite networks. In *Advances in neural information processing systems*, pp. 295–301, 1997.

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

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *Neural Information Processing Systems (NeurIPS)*, 2019.

APPENDICES

APPENDIX A REPRODUCIBILITY AND DETAILS OF EXPERIMENTAL SETUP

APPENDIX A.1 SYNTHETIC DATA

For the 1D regression experiment on synthetic data (Fig 1), we used feed-forward neural networks with 2 layers of 128 units each and a 1-dimensional output layer. We used an ensemble size of 5. The network was trained on 20 points sampled from the negative domain of a sigmoid function and tested on 20 points sampled from the positive domain.

APPENDIX A.2 EXPERIMENTAL SETUP

Model architecture For the CIFAR-10 experiments, we adapted the setup from the `cifar10-fast` model[3]. For the supervised networks and the prior networks in our uncertainty estimators, we used the exact same architecture as in this model. For the predictor networks in the uncertainty estimators, we added two additional layers at the end to make sure the prior functions are learnable (see Fig. 2).

We followed Burda et al. (2018) in choosing the output size to be $M = 512$ and using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.0001. We optimized the initialization scale of our networks as a hyperparameter on the grid $\{0.01, 0.1, 1.0, 2.0, 10.0\}$ and chose 2.0, which achieved the best performance. We chose a scaling factor of $\beta = 1.0$ for the uncertainty bonus of the random priors and fixed it for all experiments.

Data For the CIFAR-10 experiment, we trained on the classes {bird, dog, frog, horse} and excluded {cat, deer, airplane, automobile, ship, truck}. For the small CIFAR-10 ablation experiment, we trained on 75 images sampled from the classes {ship, truck} and excluded the remaining classes.

Out-of-distribution classification For computing the areas under the receiver-operator characteristic curves (AUROC) in the OOD classification tables, we used the `roc_auc_score` function from the Python package `sklearn` (Pedregosa et al., 2011), using the predicted uncertainties as predicted label scores and binary labels for whether or not the samples were from the training set.

APPENDIX B ADDITIONAL RESULTS

	RP	DE	DE +AT	DR
B=1				
Train v. cat/deer	**0.99 ± 0.002**	0.83 ± 0.065	0.96 ± 0.008	0.81 ± 0.001
Train v. vehicles	**1.00 ± 0.000**	0.82 ± 0.070	0.96 ± 0.007	0.76 ± 0.001
Train v. excluded	**1.00 ± 0.001**	0.82 ± 0.069	0.96 ± 0.007	0.77 ± 0.002
Train v. SVHN	**0.95 ± 0.013**	0.88 ± 0.101	**0.96 ± 0.009**	0.86 ± 0.002

Table 3: Out-of-distribution AUROC for random priors (RP), deep ensembles (DE), deep ensembles with adversarial training (DE+AT) and spatial concrete dropout (DR). The errors are computed from ten samples each in the $B = 1$ case.

APPENDIX C PROOFS

We now give formal proofs for the results in the paper.

[3] `https://github.com/davidcpage/cifar10-fast`

Appendix C.1 Proofs relating to Conservatism

Proposition 1 (Strict Conservatism in Expectation)**.** *Assume that f is a GP. Then for any function $h : \mathbb{R}^{N\times K} \to \mathbb{R}^M$, we have*

$$\tilde{\sigma}^2_m(x_\star) = \sigma^2_X(x_\star) + \underbrace{\mathrm{E}_{f(X)}\left[\tfrac{1}{M}\|\mu_{Xf}(x_\star) - h_{Xf}(x_\star)\|^2\right]}_{\geq 0}. \tag{11}$$

Moreover, equality holds if and only if $h_{Xf}(x_\star) = \mu_{Xf}(x_\star)$.

Proof. We instantiate Lemma 1 by setting f to be a GP. By equation 2, the posterior covariance of a GP does not depend on the target values, i.e. $\sigma^2_{Xf}(x_\star) = \sigma^2_X(x_\star)$. The first part of the result can be shown by pulling $\sigma^2_X(x_\star)$ out of the expectation. Moreover, since $\|\cdot\|$ is a norm and hence positive semi-definite, equality holds if and only if $h_{Xf}(x_\star) = \mu_{Xf}(x_\star)$. □

Lemma 1. *For any function $h : \mathbb{R}^{N\times(K+1)} \to \mathbb{R}^M$, for any test point $x_\star \in \mathbb{R}^K$ and for any stochastic process $\{f(x)\}_{x\in\mathbb{R}^K}$ with all second moments finite and exchangeable outputs, it holds that*

$$\tilde{\sigma}^2_m(x_\star) = \mathrm{E}_{f(X)}\left[\sigma^2_{Xf}(x_\star) + \tfrac{1}{M}\|\mu_{Xf}(x_\star) - h_{Xf}(x_\star)\|^2\right]. \tag{9}$$

Proof. We prove the statement by re-writing the expression on the left.

$$\begin{aligned}
\tilde{\sigma}^2_m(x_\star) &= \tfrac{1}{M}\,\mathrm{E}_{f(X),f(x_\star)}\left[\|f(x_\star) - h_{Xf}(x_\star)\|^2\right] && (14)\\
&= \tfrac{1}{M}\,\mathrm{E}_{f(X)}\left[\mathrm{E}_{f(x_\star|f(X))}\left[\|f(x_\star) - h_{Xf}(x_\star)\|^2\right]\right] && (15)\\
&= \tfrac{1}{M}\,\mathrm{E}_{f(X)}\left[\mathrm{E}_{f(x_\star|f(X))}\left[\textstyle\sum_{m=1}^M (f^m(x_\star) - h^m_{Xf}(x_\star))^2\right]\right] && (16)\\
&= \tfrac{1}{M}\,\mathrm{E}_{f(X)}\left[\mathrm{E}_{f(x_\star|f(X))}\left[\textstyle\sum_{m=1}^M (f^m(x_\star))^2 - 2f^m(x_\star)h^m_{Xf}(x_\star) + (h^m_{Xf}(x_\star))^2\right]\right] && (17)\\
&= \tfrac{1}{M}\,\mathrm{E}_{f(X)}\left[\textstyle\sum_{m=1}^M \sigma^2_{Xf^m}(x_\star) + (\mu_{Xf^m}(x_\star))^2 - 2f^m(x_\star)\mu_{Xf^m}(x_\star) + (h^m_{Xf}(x_\star))^2\right] && (18)\\
&= \tfrac{1}{M}\,\mathrm{E}_{f(X)}\left[\textstyle\sum_{m=1}^M \sigma^2_{Xf^m}(x_\star) + (\mu_{Xf^m}(x_\star) - h^m_{Xf}(x_\star))^2\right] && (19)\\
&= \mathrm{E}_{f(X)}\left[\sigma^2_{Xf}(x_\star) + \tfrac{1}{M}\|\mu_{Xf}(x_\star) - h_{Xf}(x_\star)\|^2\right] && (20)
\end{aligned}$$

Here, the equality in (15) holds by definition of conditional probability. The equality in (18) holds by definition of posterior mean and the equality 20 follows by assumption that the process has exchangeable outputs. While this argument follows a similar pattern to a standard result about Bayesian Risk (equation 3), it is not identical because the function h_{Xf} depends on f. □

Lemma 2. *Assume that the random variable $\hat{\sigma}^2_m(x_\star)$ has finite variance upper bounded by σ^2_S. With probability $1-\delta$, we have $\hat{\sigma}^2_m(x_\star) + \frac{1}{\sqrt{\delta}}\sigma_S \geq \tilde{\sigma}^2_m(x_\star)$.*

Proof. The proof is standard, but we state it in our notation for completeness. Applying Chebyshev's inequality to the random variable $\hat{\sigma}^2_m(x_\star)$, we have that Prob $\left(|\tilde{\sigma}^2_m(x_\star) - \hat{\sigma}^2_m(x_\star)| \geq \frac{1}{\sqrt{\delta}}\sigma_S\right) \leq \delta$, implying the statement. □

Lemma 3. *Assume that the GP $\{f(x)\}$ is zero mean with exchangeable outputs and the function h_{Xf} takes values in $[-U, U]^M$. Assume that permuting the outputs of f produces the same permutation in the outputs of h_{Xf}. With probability $1-\delta$, we have*

$$\sigma^2_S \leq \tfrac{1}{B}\left(4U^4 + (\hat{\sigma}^2_0(x_\star))^2 C\right). \tag{13}$$

Here, $\hat{\sigma}^2_0(x_\star)$ is a sample-based estimate of the prior variance obtained with B_0 samples and $C = \left(\frac{B_0-1}{\chi^2_I(\delta)}\right)^2$, where χ^2_I denotes the inverse CDF of the Chi-Squared distribution with $B_0 - 1$ degrees of freedom.

Proof. We seek to decompose the variance of $\hat{\sigma}^2_m(x_\star)$ into the part that comes from the prior and the part that comes from the fitted function h_{Xf^m}.

$$
\begin{aligned}
\sigma_S^2 &= \mathrm{Var}_{f_1,\dots,f_B}\left[\hat{\sigma}^2_m(x_\star)\right] && (21)\\
&= \mathrm{Var}_{f_1,\dots,f_B}\left[\textstyle\sum_{i=1}^{B} \frac{1}{MB}\|f(x_\star) - h_{Xf_i}(x_\star)\|^2\right] && (22)\\
&= \tfrac{1}{B}\,\mathrm{Var}_f\left[\tfrac{1}{M}\|f(x_\star) - h_{Xf_i}(x_\star)\|^2\right] + && (23)\\
&= \tfrac{1}{B}\tfrac{1}{M^2}\,\mathrm{Var}_f\left[(\textstyle\sum_{m=1}^{M}(f^m(x_\star) - h_{Xf^m}(x_\star))^2\right] + && (24)\\
&= \tfrac{1}{B}\tfrac{1}{M^2}\textstyle\sum_{m=1}^{M}\mathrm{Var}_f\left[(f^m(x_\star) - h_{Xf^m}(x_\star))^2\right] + && (25)\\
&\qquad \textstyle\sum_{m=1}^{M}\sum_{l=1}^{M}\mathrm{Cov}_f\left[(f^m(x_\star) - h_{Xf^m}(x_\star))^2), (f^l(x_\star) - h_{Xf^l}(x_\star))^2)\right] && (26)\\
&\le \tfrac{1}{B}\tfrac{1}{M^2}M^2\,\mathrm{Var}_f\left[(f^m(x_\star) - h_{Xf^m}(x_\star))^2\right] && (27)\\
&= \tfrac{1}{B}\,\mathrm{Var}_f\left[(f^m(x_\star) - h_{Xf^m}(x_\star))^2\right] && (28)\\
&= \tfrac{1}{B}\left(\mathrm{Var}_f\left[(f^m(x_\star))^2\right] + \mathrm{Var}_f\left[(h_{Xf^m}(x_\star))^2\right] - \mathrm{Cov}_f\left[(f^m(x_\star))^2, (h_{Xf^m}(x_\star))^2\right]\right) && (29)\\
&\le \tfrac{1}{B}\left(\mathrm{Var}_f\left[(f^m(x_\star))^2\right] + \mathrm{Var}_f\left[(h_{Xf^m}(x_\star))^2\right]\right) && (30)
\end{aligned}
$$

We now bound the two variances in equation 30 separately. Since $f(x_\star)$ is Gaussian, we can use a sample-based estimate of the prior variance and obtain an probabilistic confidence interval. In particular, we know that $\tilde{\sigma}^2_0(x_\star) \le \hat{\sigma}^2_0(x_\star)\frac{B_0-1}{\chi^2_I(\delta)}$ with probability $1-\delta$, where χ^2_I denotes the inverse CDF of the Chi-Squared distribution with $B_0 - 1$ degrees of freedom. Using the assumption that f is zero-mean, we now have

$$
\mathrm{Var}_f\left[(f^m(x_\star))^2)\right] = 2\tilde{\sigma}^4_0(x_\star) \le 2(\hat{\sigma}^2_0(x_\star))^2 \underbrace{\left(\frac{B_0-1}{\chi^2_I(\delta)}\right)^2}_{C}. \tag{31}
$$

Moreover, since $h_{Xf^m}(x_\star)$ is bounded by U, we have

$$
\mathrm{Var}_f\left[(h_{Xf^m}(x_\star))^2)\right] \le 4U^4. \tag{32}
$$

The bound is obtained by combining equation 30 with equations 32 and 31. □

Appendix C.2 Proofs relating to concentration

We now proceed to the proofs showing concentration. We begin by formally defining a class of predictor networks.

Definition 1 (Class $\mathcal{H}_U$ of Lipschitz networks)**.** *Consider functions $h : \mathbb{R}^K \to \mathbb{R}^M$. Let $j, j' = 1, \dots, M$, index the outputs of the function. We define $\mathcal{H}_U$ so that each $h \in \mathcal{H}_U$ has the following properties for each j, j'.* **(P1)** *h_j is Lipschitz continuous with constant L, i.e. $\|h_j(x) - h_j(x')\|_2 \le L\|x - x`\|_2$ for all x, x' with $\|x\|_\infty \le 1$ and $\|x'\|_\infty \le 1$,* **(P2)** *outputs are exchangeable, i.e. $\{h_j : h \in \mathcal{H}_U\} = \{h_{j'} : h \in \mathcal{H}_U\}$,* **(P3)** *the class is symmetric around zero, i.e. $h_j \in \{h_j : h \in \mathcal{H}_U\}$ implies $-h_j \in \{h_j : h \in \mathcal{H}_U\}$.* **(P4)** *$h_j$ is bounded, i.e. $\max_{\|x\|_\infty \le 1}|h_j(x)| \le U$.*

While the conditions in Definition 1 look complicated, they are in fact easy to check for predictor networks that follow the architecture in Figure 2. In particular, Lipschitz continuity **(P1)** has to hold in practice because its absence would indicate extreme sensitivity to input perturbations. Output exchangeability **(P2)** holds since reordering the outputs doesn't change our architecture. Symmetry around zero **(P3)** holds by flipping the sign in the last network layer. Boundedness **(P4)** is easy to ensure by clipping outputs. In the following Lemma, we obtain a bound on the expected uncertainty.

Lemma 4. *Consider a target function $f : \mathbb{R}^K \to \mathbb{R}^M$, where $j = 1, \dots, M$, with the domain restricted to $\|x\|_\infty \le 1$. Introduce a constant U such that $\max_{\|x\|_\infty \le 1}|f_j(x)| \le U$. Denote the data distribution with support on $\{x : \|x\|_\infty \le 1\}$ as $\mathcal{D}$. Moreover, assume $K \ge 3$. For $h_{Xf} \in \mathcal{H}_U$, with probability $1 - \delta$ we have*

$$
\mathrm{E}_{x_\star \sim \mathcal{D}}\left[\tfrac{1}{M}\|f(x_\star) - h_{Xf}(x_\star)\|^2\right] \le \tfrac{1}{MN}\textstyle\sum_{i=1}^{N}\|f(x_i) - h_{Xf}(x_i)\|^2 + LU\, O\Big(\tfrac{1}{\sqrt[K]{N}}\sqrt{\tfrac{\log(1/\delta)}{N}}\Big). \tag{33}
$$

Proof. The proof uses standard Rademacher tools. To avoid confusion across several conventions, we explicitly define the Rademacher complexity of a function class $\mathcal{G}$ as:

$$\hat{\mathfrak{R}}_N(\mathcal{G}) \triangleq \mathrm{E}_{u_i}\left[\sup_{g\in\mathcal{G}} \frac{1}{N}\sum_{i=1}^N u_i g(x_i)\right] = \mathrm{E}_{u_i}\left[\sup_{g\in\mathcal{G}} \frac{1}{N}\left|\sum_{i=1}^N u_i^j g(x_i)\right|\right]. \tag{34}$$

Here, the random variables u_i are sampled i.i.d. using a discrete distribution with $\mathrm{Prob}(u_i = -1) = \mathrm{Prob}(u_i = 1) = \frac{1}{2}$ and the the second equality follows by using property **(P3)**. We start by applying the generic Rademacher bound (Mohri et al., 2018) to the function class $\mathcal{M} = \{x_1, \dots, x_N, t_1 \dots, t_N \to \frac{1}{U^2}\frac{1}{M}\|t_i - h(x_i)\|^2, h \in \mathcal{H}_U\}$, which contains the possible errors of the predictor.

$$\mathrm{E}_{x_\star\sim\mathcal{D}}[\tfrac{1}{B^2}\tfrac{1}{M}\|f(x_\star) - h_{Xf}(x_\star)\|^2] \tag{35}$$

$$\leq \frac{1}{MN}\frac{1}{B^2}\sum_{i=1}^N \|f(x_i) - h_{Xf}(x_i)\|^2 + \hat{\mathfrak{R}}_N(\mathcal{M}) + O\left(\sqrt{\frac{\log(1/\delta)}{N}}\right). \tag{36}$$

We now introduce the function class $\mathcal{M}' = \{x_1, \dots, x_N, t_1 \dots, t_N \to \frac{1}{B^2}(t_i^j - h^j(x_i))^2, h \in \mathcal{H}_U\}$, which models the per-output squared error. Because of property **(P2)**, $\mathcal{M}'$ does not depend on the output index j. By pulling out the sum outside the supremum in equation 34, we get

$$\hat{\mathfrak{R}}_N(\mathcal{M}) \leq \hat{\mathfrak{R}}_N(\mathcal{M}'). \tag{37}$$

by Talagrand's Lemma (Mohri et al., 2018; Duchi, 2009), we also have

$$\hat{\mathfrak{R}}_N(\mathcal{M}') \leq 4\hat{\mathfrak{R}}_N(\mathcal{H}_1). \tag{38}$$

Here, $\mathcal{H}_1 = \{\frac{1}{U}h^j : h \in \mathcal{H}_U\}$. By property **(P1)**, functions in $\mathcal{H}_1$ are Lipschitz continuous with constant L/U. Instantiating a known bound for Lipschitz-continuous functions (Luxburg & Bousquet, 2004, Theorem 18 and Example 4), and using the assumption $K \geq 3$, we get $\hat{\mathfrak{R}}_N(\mathcal{H}_1) \leq \frac{L}{U}\, O\left(\frac{1}{\sqrt[K]{N}}\right)$. The Lemma follows by combining this with equation 37 and equation 38, plugging into equation 35 and re-scaling by U^2. □

Lemma 4 allowed us to relate the error on the training set to the expected error on the test set. It also shows that the two will be closer for small values of the Lipschitz constant L. We now use this Lemma to show our main concentration result (Proposition 2).

Proposition 2. *If the training converges, i.e. the training loss* $\frac{1}{MN}\sum_{i=1}^N \|f(x_i) - h_{Xf}(x_i)\|^2 = \sigma_A^2$ *for arbitrarily large training sets, then assuming the predictors* h_{Xf} *are bounded and Lipschitz continuous with constant L, then under technical conditions the uncertainties concentrate, i.e.* $\hat{\sigma}^2(x_\star) \to 0$ *as* $N \to \infty$ *and* $B \to \infty$ *with probability 1.*

Proof. We are assuming the technical conditions of Lemma 4. Instantiating Lemma 4, setting the training loss to zero in the RHS of equation 33 and letting $N \to \infty$, we obtain the following with probability 1:

$$\lim_{N\to\infty} \mathrm{E}_{x_\star\sim\mathcal{D}}[\hat{\sigma}_m^2(x_\star)] = \sigma_A^2. \tag{39}$$

From the continuity of f and h_{Xf} we have that $\hat{\sigma}_m^2$ is continuous in $x_\star$. Together with the property that $\hat{\sigma}_m^2$ is non-negative, this gives that for every $x_\star$.

$$\lim_{N\to\infty} \hat{\sigma}_m^2(x_\star) = \sigma_A^2. \tag{40}$$

Since σ_A^2 does not depend on B, we also have:

$$\lim_{B\to\infty}\lim_{N\to\infty} \hat{\sigma}_m^2(x_\star) = \sigma_A^2. \tag{41}$$

From the definition of $\hat{\sigma}_s^2$, we have that

$$\lim_{B\to\infty}\lim_{N\to\infty} \hat{\sigma}_s^2 = 0. \tag{42}$$

Combining equation 41 and equation 42 with equation 4, we get:

$$\lim_{B\to\infty}\lim_{N\to\infty} \hat{\sigma}_m^2(x_\star) + \beta\hat{\sigma}_s^2 - \sigma_A^2 = 0 \tag{43}$$

We obtain the Lemma by comparing with equation 4. □