Density Uncertainty Layers for Reliable Uncertainty Estimation

Yookoon Park
Department of Computer Science
Columbia University
New York, NY 10027, USA
yookoon.park@columbia.edu

David M. Blei
Department of Computer Science, Statistics
Columbia University
New York, NY 10027, USA
david.blei@columbia.edu

Abstract

Assessing the predictive uncertainty of deep neural networks is crucial for safety-related applications of deep learning. Although Bayesian deep learning offers a principled framework for estimating model uncertainty, the common approaches that approximate the parameter posterior often fail to deliver reliable estimates of predictive uncertainty. In this paper, we propose a novel criterion for reliable predictive uncertainty: a model's predictive variance should be grounded in the empirical density of the input. That is, the model should produce higher uncertainty for inputs that are improbable in the training data and lower uncertainty for inputs that are more probable. To operationalize this criterion, we develop the *density uncertainty layer*, a stochastic neural network architecture that satisfies the density uncertain criterion by design. We study density uncertainty layers on the UCI and CIFAR-10/100 uncertainty benchmarks. Compared to existing approaches, density uncertainty layers provide more reliable uncertainty estimates and robust out-of-distribution detection performance.

1 Introduction

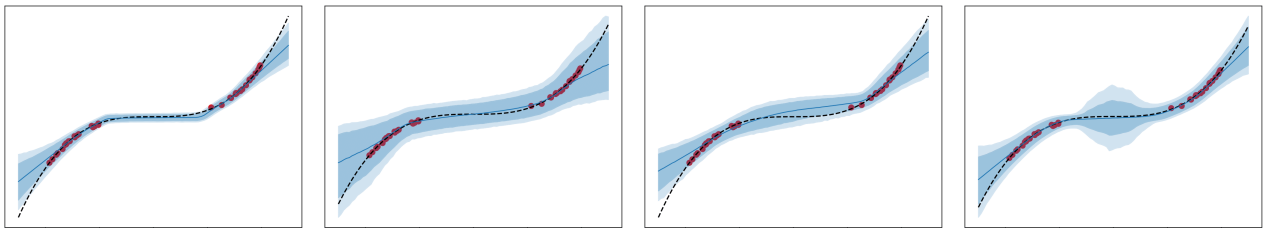
The success of deep learning models in a range of applications has spurred significant interest in deploying the for real-world predictions. But in high-stakes domains, such as healthcare, finance, and autonomous systems, both the model's prediction and its predictive uncertainty are crucial. A challenge is that conventional neural networks lack a robust mechanism for quantifying uncertainty, and they tend to produce overconfident predictions [Guo et al., 2017, Ovadia et al., 2019]. This paper is about how to produce reliable estimates of predictive uncertainty with deep neural networks.

Why is this a problem? Bayesian deep learning offers a principled framework for quantifying predictive uncertainty by incorporating uncertainty about the model parameters [Graves, 2011, Welling and Teh, 2011, Neal, 2012, Blundell et al., 2015]. However, the common variational inference (VI) approaches that approximate the parameter posterior in Bayesian deep learning [Graves, 2011, Blundell et al., 2015, Kingma et al., 2015, Dusenberry et al., 2020] often fall short of providing reliable predictive uncertainty [Foong et al., 2019b, Ober and Rasmussen, 2019]. For example, Figure 1a-c demonstrate that these methods produce either collapsed or flat predictive uncertainty around the origin, even though no training data were observed in that region.

In this work, we propose a novel criterion for reliable predictive uncertainty and a new stochastic neural network architecture to satisfy the criterion. The *density uncertainty criterion* posits that a model's predictive uncertainty should be grounded in the empirical density of the input. That is, we should see higher uncertainty for improbable inputs and lower for more probable ones. As motivation, we will show how Bayesian linear regression inherently adheres to the criterion.

We then develop the *density uncertainty layer*, a stochastic neural network architecture that is designed to satisfy the density uncertainty criterion. The idea is to fit an energy-based model of input and then satisfy the density uncertainty criterion via a constraint on the predictive variance produced by the approximate posterior. Density uncertainty layers serve as a flexible building block for density-aware deep neural networks. See Figure 1d for how Density Uncertainty produces high predictive uncertainty in the low-density region in the input space around the origin.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.



(a) MFVI (b) Variational Dropout (c) Rank 1 BNN (d) Density Uncertainty

Figure 1: Predictive uncertainty of a two-layer MLP on a toy regression problem. The red dots denote the training data and the blue shades mark the 95th and 99th percentiles of the predictive variance. All baselines other than Density Uncertainty fail to account for the in-between uncertainty in the low-density region around the origin

We study density uncertainty layers on the UCI and CIFAR-10/100 uncertainty benchmarks. We compare the proposed method to popular uncertainty estimation methods for neural networks [Blundell et al., 2015, Kingma et al., 2015, Gal and Ghahramani, 2016, Dusenberry et al., 2020]. We find that the proposed density-aware neural networks provide more reliable predictive uncertainty estimates and robust out-of-distribution (OOD) detection performance.

Contributions. In summary, this paper makes the following contributions.

1. We propose the *density uncertainty criterion* for reliable predictive uncertainty, asserting that a model's predictive uncertainty should be grounded in the training density of the input.
2. We present the *density uncertainty layer* whose predictive uncertainty adheres to the density uncertainty criterion and serves as a general building block for uncertainty-aware neural networks.
3. We study the proposed density uncertainty layers on the UCI and CIFAR-10/100 uncertainty estimation benchmarks. On both benchmarks, this method performs better than existing approaches.

2 Density Uncertainty Layers

2.1 Motivation: Bayesian Linear Regression

We first illustrate how Bayesian uncertainty for linear regression is grounded in the empirical density of the input. In the next section, this observation will motivate a novel criterion for reliable predictive uncertainty.

Consider a Bayesian linear regression model with input $\mathbf{X} \in \mathbb{R}^{N \times D}$, target $\mathbf{y} \in \mathbb{R}^N$, and weight $w \in \mathbb{R}^D$:

$$p(w) = \mathcal{N}(w|0, \alpha^{-1}I), \tag{1}$$

$$p(\mathbf{y}|\mathbf{X}, w) = \prod_{i=1}^{N} \mathcal{N}(y_i|w^T x_i, \beta^{-1}), \tag{2}$$

The posterior distribution of the weight w given the observations $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ is

$$p(w|\mathcal{D}) = \mathcal{N}(w|\mu, \Lambda^{-1}), \tag{3}$$

$$\text{where } \mu = \beta\Lambda^{-1}\mathbf{X}^T\mathbf{y} \text{ and } \Lambda = \beta\mathbf{X}^T\mathbf{X} + \alpha I. \tag{4}$$

The posterior predictive distribution for a test input x_* is obtained by marginalizing out the weight from the posterior joint

$$p(y_*|x_*, \mathcal{D}) = \int p(y_*, w|x_*, \mathcal{D})dw \tag{5}$$

$$= \mathcal{N}(y_*|\mu^T x_*, \beta^{-1} + x_*^T\Lambda^{-1}x_*). \tag{6}$$

Equation (6) helps establish the connection between the Bayesian predictive uncertainty for linear regression and the empirical input density. Rewriting the predictive variance in Equation (6),

$$\text{Var}[y_*|x_*, \mathcal{D}] = \frac{1}{\beta} + \frac{2}{\beta N}E(x_*), \tag{7}$$

$$\text{where } E(x_*) = \frac{1}{2}x_*^T\Sigma^{-1}x_*, \tag{8}$$

$$\Sigma = \frac{1}{N}(\mathbf{X}^T\mathbf{X} + \frac{\alpha}{\beta}I). \tag{9}$$

Here $E(x)$ is an *energy function*, an unnormalized negative log density of x such that

$$p_{\text{energy}}(x) \propto \exp(-E(x)) \tag{10}$$

Specifically, Equations (8) to (9) describe an instance of Gaussian energy model $\mathcal{N}(0, \Sigma)$, where Σ is the MAP estimate of the input covariance matrix given the observations $\mathbf{X}$. This shows that Bayesian linear model derives its predictive uncertainty from the Gaussian energy model of the input that is fitted to the training data. Consequently, predictive uncertainty will be high for test inputs that are improbable in the training density and low for those that are more probable, providing intuitive and reliable predictive uncertainty. In the following sections, we build upon this principle.

2.2 Variational Inference for BNNs

A Bayesian Neural Network (BNN) $\Omega(x;\theta)$ treats its parameters θ of neural networks as random variables from a prior $p(\theta)$. Combined with a likelihood function $p(y|\Omega(x;\theta))$ and the observations $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}$, the posterior distribution of the parameters is defined by the Bayes' rule:

$$p(\theta|\mathcal{D}) \propto p(\theta) \prod_{i=1}^{N} p(y_i|\Omega(x_i;\theta)) \tag{11}$$

However, the posterior is highly complex and intractable in BNNs. Variational inference (VI) circumvents this problem by optimizing a tractable approximation to the posterior. For example, the approximate distribution $q(\theta)$ commonly takes the form of fully-factorized Gaussians whose parameters are denoted by ϕ. VI fits the approximate posterior by maximizing the evidence lower-bound (ELBO) [Blundell et al., 2015]:

$$\mathcal{L}_\phi = \mathbb{E}_{q(\theta)}\Big[\sum_{i=1}^{N} \log p(y_i|\Omega(x_i;\theta))\Big] - D(q(\theta)\|p(\theta)),$$

where D is the Kullback-Leibler divergence. At test time, the posterior predictive distribution is approximated with M Monte Carlo samples from $q(\theta)$:

$$p(y_*|x_*, \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} p(y_*|\Omega(x_*;\theta_m)), \tag{12}$$

2.3 The Density Uncertainty Criterion

We have shown that Bayesian predictive uncertainty has desirable properties for a linear model. However, VI-based BNNs often fails to provide reliable uncertainty estimates [Foong et al., 2019a, Ober and Rasmussen, 2019] as demonstrated in Figure 1 where the VI baselines fail to accurately capture the density of the data in their predictive uncertainty.

To address this, we propose a novel uncertainty criterion that asserts model's predictive uncertainty should be grounded in the training density of the input. To formalize the concept, we first introduce an auxiliary energy-based generative model of the input:

$$p_{\text{energy}}(x;\omega) \propto \exp(-E(x;\omega)), \tag{13}$$

where the generative parameter ω is fitted to the training data $x_1, ..., x_N$. We omit ω for simplicity hereafter.

We now establish the *density uncertainty criterion*.

Definition 1 (Density uncertainty criterion). For a parameterized model $f(x;\theta)$ with a distribution on the parameters $q(\theta)$ and an non-negative energy model $E(x) \geq 0$ fitted to the training data, the model $f(x;\theta)$ adheres to the density uncertainty criterion if

$$\text{Var}_{q(\theta)}[f(x;\theta)] \propto E(x) \text{ for all } x \in \mathcal{X}. \tag{14}$$

The criterion posits that the predictive uncertainty should be proportional to the energy of the input, producing high predictive uncertainty for inputs that are improbable in the training distribution and low uncertainty for inputs that are more probable. As illustrated in Section 2.1, Bayesian linear regression inherently satisfies the density uncertainty criterion of Equation (14).

Incorporating the density uncertainty criterion into the ELBO objective yields

$$\begin{aligned}\arg\max_{q\in\mathcal{Q}} \mathbb{E}_{q(\theta)}\Big[\sum_{i=1}^{N} \log p(y_i|f(x_i;\theta))\Big] - D(q(\theta)\|p(\theta)) \\ \text{s.t. } \text{Var}_{q(\theta)}[f(x;\theta)] \propto E(x) \text{ for all } x \in \mathcal{X},\end{aligned} \tag{15}$$

where D is the Kullback-Leibler divergence and p is a prior. The constraint in Equation (15) ensures the predictive uncertainty adheres to the density uncertainty criterion, so that its predictive uncertainty is consistently derived from the training density of the input, yielding reliable uncertainty estimates.

Example: Bayesian linear regression Revisiting the Bayesian linear regression example (Section 2.1), the family of parameter distribution $q(w)$ that satisfies the density uncertainty criterion is

$$q(w) = \mathcal{N}(\mu, \gamma\Sigma^{-1}), \tag{16}$$

$$\text{where } \Sigma = \frac{1}{N}(\mathbf{X}^T\mathbf{X} + \frac{\alpha}{\beta}I), \tag{17}$$

where γ is a scaling scalar and μ is a trainable parameter. The posterior precision of the weight is now tied to the training covariance estimate of the input. This results in the predictive variance of

$$\text{Var}_{q(w)}[f(x_*;w)] \propto E(x_*) = \frac{1}{2} x_* \widehat{\Sigma}^{-1} x_*, \tag{18}$$

satisfying the density uncertainty criterion for a Gaussian energy model E. Notably, the constrained parameter posterior (Equation (16)) recovers the true posterior in Bayesian linear regression (Equation (3)).

Reparametrized objective More generally, we introduce a reparametrized version of the objective in Equation (15). Consider a stochastic function $f(x, \epsilon; \phi)$ but now with a deterministic parameter ϕ, exogenous noise variable ϵ, and a noise distribution $q(\epsilon)$:

$$\begin{aligned}\arg\max_{\phi} \mathbb{E}_{q(\epsilon)}\Big[\sum_{i=1}^{N} \log p(y_i|f(x_i, \epsilon; \phi))\Big] - D(q(\epsilon)\|p(\epsilon)) \\ \text{s.t. } \text{Var}_{q(\epsilon)}[f(x, \epsilon; \phi)] \propto E(x) \text{ for all } x \in \mathcal{X},\end{aligned} \tag{19}$$

A broad class of $q(\theta)$ such as normal distributions admits this kind of reparametrization. The reparametrized objective offers more flexibility in how we incorporate noise into the model while adhering to the density uncertainty criterion. For example, we may choose to directly inject noise into the low-dimensional function output instead of sampling the high-dimensional parameters, thereby improving the computational efficiency and reducing the gradient variance (e.g. Kingma et al. [2015]). Therefore, we default to this form in the remainder of the paper.

2.4 The Density Uncertainty Layer

We now select an appropriate energy model $E(x)$ and the structure of a stochastic function $\Omega(x, \epsilon; \phi)$ for deep neural networks. In this work, we consider a popular form of residual network architecture

$$a_1 = f_1(x, \epsilon_1; \phi_1), \tag{20}$$
$$h_\ell = \pi(a_\ell), \tag{21}$$
$$a_{\ell+1} = a_\ell + f_\ell(h_\ell, \epsilon_\ell; \phi_\ell), \text{ for } l = 1, ..., L \tag{22}$$

where f_l is a stochastic linear layer with deterministic parameter ϕ_ℓ and exogenous noise variable ϵ_ℓ. π is a nonlinear activation function such as ReLU.

Section 2.1 shows that Bayesian linear regression derives its predictive uncertainty from a Gaussian energy model. Motivated by this insight, we pair each linear layer f_ℓ with a Gaussian energy model E_ℓ and make the individual linear layers $f_1, f_2, ..., f_L$ adhere to the density uncertainty criterion layer-wise:

$$\arg\max_{\phi} \mathbb{E}_q \Big[\sum_{i=1}^{N} \log p(y_i | \Omega(x_i, \boldsymbol{\epsilon}; \boldsymbol{\phi})) \Big] - D(q(\boldsymbol{\epsilon}) \| p(\boldsymbol{\epsilon}))$$
$$\text{s.t. } \mathrm{Var}_{q(\epsilon_\ell)}[f_\ell^j(h_\ell, \epsilon_\ell; \phi_\ell)] \propto E_\ell(h_\ell) \tag{23}$$
$$\text{for all } h_\ell \in \mathcal{H},\ j = 1, ..., D, \text{ and } \ell = 1, ..., L, \tag{24}$$

where j indexes the hidden units and D is the width of the layers. $\boldsymbol{\phi}$ encompasses the deterministic weights of the neural network and the posterior parameters of the noise distributions, and $\boldsymbol{\epsilon}$ involves all noise variables.

This design is based on several key observations: First, the Gaussian energy facilitates efficient training and energy evaluation. Second, making the individual layers stochastic fosters functional diversity and stochastic regularization effect. Third, the complexity of the uncertainty landscape grows naturally with the model complexity with size of in the neural network. Lastly, a relaxed version of the density uncertainty criterion holds at the network level which we describe next.

Proposition 1 (Density Uncertainty Criterion for Residual Networks). *Define the total energy* $E(x, h_1, ..., h_\ell) = \sum_{\ell=1}^{L} E_\ell(h_\ell)$. *Assume the followings:*

1. *There exist* $\alpha, \beta \in \mathbb{R}_+$ *such that for all* j, ℓ
$$\alpha E_\ell(h_\ell) \leq Var_{q(\epsilon_\ell)}[f_\ell^j(h_\ell, \epsilon_\ell; \phi_\ell)] \leq \beta E_\ell(h_\ell)$$
2. *There exists* $M \in \mathbb{R}_+$ *such that* $\|w_\ell^j\|_2^2 \leq M$ *for all* j, ℓ, *where* w_ℓ^j *is the deterministic weight for* j*-th hidden unit at layer* l.
3. ϕ *is a 1-Lipschitz activation fucntion (e.g. ReLU).*

Then the dimension-wise variance of the network output a_{L+1} *is bounded from below by the expected total energy:*

$$Var_{q(\theta)}[a_{L+1}^j | x] \geq C\, \mathbb{E}_{q(\theta)}[E(x, h_1, ..., h_\ell)] \tag{25}$$

for some constant C. *Proof is in the appendix.*

We now introduce the *density uncertainty layer*, a stochastic architecture that by design satisfies the density uncertainty criterion layer-wise (Equation (23)):

$$f_\ell^j(h_\ell, \epsilon_\ell; w_\ell^j) = w_\ell^j \cdot h_\ell + \epsilon_\ell^j \sqrt{E(h_\ell)} + \eta_\ell^j,$$
$$\text{where } E_\ell(h_\ell) = \frac{1}{2} h_\ell^T \Sigma_\ell^{-1} h_\ell, \tag{26}$$
$$q(\epsilon_\ell^j) = \mathcal{N}(0, \gamma_\ell^j),\ q(\eta_\ell^j) = \mathcal{N}(0, \beta_\ell^j), \tag{27}$$

where $\cdot$ denotes dot product, w_ℓ^j is the deterministic weight for the j-th hidden unit at layer ℓ, and $E_\ell(h_\ell)$ is the Gaussian energy model with the covariance parameter Σ_ℓ. The layer has two noise components ϵ_ℓ^j and η_ℓ^j, and the predictive variance of

$$\mathrm{Var}_{q(\epsilon_\ell)}[f_\ell^j(h_\ell, \epsilon_\ell; \phi_\ell)] = \gamma_\ell^j E_\ell(h_\ell) + \beta_\ell^j, \tag{28}$$

which satisfies the density uncertainty criterion as the bias term β_ℓ^j can be always absorbed into the energy function. The posterior noise variance parameters $\gamma_\ell^j, \beta_\ell^j$ are optimized using the ELBO, granting the model the flexibility to adjust the noise scale for individual hidden units in the layer. While we assume Gaussian noise for simplicity, we may potentially incorporate other noise distributions such as heavy-tailed ones [Dusenberry et al., 2020] for enhanced robustness.

Gaussian energy We adopt the LDL parametrization of the precision matrix $\Sigma_\ell^{-1} = L_\ell D_\ell L_\ell^T$ for the layer-wise Gaussian energy models, where L_ℓ is a lower unit triangular matrix, and D_ℓ is a non-negative diagonal matrix. This admits efficient energy evaluation

$$E_\ell(h_\ell) = \frac{1}{2} h_\ell^T \Sigma_\ell^{-1} h_\ell = \frac{1}{2} \| D_\ell^{\frac{1}{2}} L_\ell^T h_\ell \|_2^2,$$

without computing the inverse of the covariance matrix and simplifies the log determinant of the covariance matrix as $\log |\Sigma_\ell| = -\sum_j \log D_\ell^{jj}$. For convolutional architectures, we replace the matrix-vector product

$L_\ell^T h_\ell$ with a masked convolution [Van den Oord et al., 2016] to build a convolutional energy model. The main computational overhead of density uncertainty layers comes from the matrix-vector product $L_\ell^T h_\ell$ with $O(D^2)$ complexity, same as regular linear layers.

Mixture of rank-1 Gaussians When the computational overhead is a major consideration, we propose to use a mixture of rank-1 Gaussians as an alternative layer-wise energy model. More specifically, the rank-1 Gaussian has the covariance matrix of the form:

$$\Sigma = vv^T + D, \tag{29}$$

where v is the rank-1 factor and D is a non-negative diagonal matrix. Using the rank-1 construction, the inverse can be computed as

$$\Sigma^{-1} = D^{-1} - \frac{D^{-1}vv^TD^{-1}}{1 + v^TD^{-1}v}, \tag{30}$$

and the log determinant is also simplified as

$$\log|\Sigma| = \log(1 + v^TD^{-1}v) + \sum \log D_{jj}, \tag{31}$$

reducing the computational overhead to $O(D)$. However, as the rank-1 construction may be overly restrictive in practice, we choose a K mixture of rank-1 Gaussians as our layer-wise generative model, with the computational complexity of $O(KD)$. We set K as a fraction of the layer's width and show that it can be as small as $\approx 1\%$ with negligible performance drop.

Optimization The construction of the density uncertainty layer (Equation (27)) inherently satisfies the density uncertainty constraint, simplifying the constrained optimization objective (Equation (23)) to

$$\mathcal{L}_{\boldsymbol{\phi}} = \mathbb{E}_{q(\boldsymbol{\epsilon})}\Big[\sum_{i=1}^{N} \log p(y_i|\Omega(x_i, \boldsymbol{\epsilon}; \boldsymbol{\phi}))\Big] + D(q(\boldsymbol{\epsilon})\|p(\boldsymbol{\epsilon}))$$

We assume Gaussian priors with a shared variance for the noise variables. Concurrently, the layer-wise energy models are optimized using the generative objective:

$$\mathcal{L}_{\boldsymbol{\omega}} = \sum_{i=1}^{N}\sum_{\ell=1}^{L} \mathbb{E}_{q(\boldsymbol{\epsilon})}[\log p_{\text{energy}}(h_\ell; \omega_\ell)] + \log p(\omega_\ell),$$

where $p_{\text{energy}}(h_\ell; \omega_\ell) = \exp(-E(h_\ell))$ is the Gaussian energy distribution for the input at layer ℓ and $p(\omega_\ell)$ is a prior on the generative parameter. The two objectives are jointly optimized during training.

3 Related Work

This paper contributes to Bayesian deep learning and uncertainty estimation for deep learning.

Bayesian Neural Networks and Uncertainty Bayesian Neural Networks (BNNs) establish a principled framework for estimating the uncertainty of neural networks by assuming their parameters are latent variables that follow a prior distribution. Bayes' rule, combined with a likelihood function and observations, defines the posterior distribution of the parameters. However, as exact posterior inference in BNNs is intractable, the problem boils down to approximating the parameter posterior distribution. For example, Markov Chain Monte Carlo (MCMC) [Welling and Teh, 2011, Chen et al., 2014] simulates samples from the posterior distribution, using Langevin [Welling and Teh, 2011] or Hamiltonian [Chen et al., 2014] dynamics. On the other hand, the Laplace approximation [Ritter et al., 2018] applies a second-order approximation at a mode of the posterior distribution.

Variational inference (VI) is a popular approach that reformulates inference as an optimization problem. It seeks the best approximating distribution within a distribution family that minimizes a discrepancy metric to the true posterior, such as the Kullback-Leibler (KL) divergence. Graves [2011] adopt fully-factorized Gaussian posteriors for the network's parameters, and Blundell et al. [2015] further incorporate the reparametrization trick [Kingma and Welling, 2014, Rezende et al., 2014] to obtain unbiased, low-variance gradient estimates with automatic differentiation. Louizos and Welling [2016] enhance the expressiveness of the posteriors by utilizing matrix Gaussian distributions for structured modeling of parameter correlations within each layer. More recently, Ritter et al. [2021] propose sparse representations of matrix Gaussian posteriors using inducing points [Snelson and Ghahramani, 2005, Titsias, 2009].

However, VI often fails to provide reliable uncertainty estimates for neural networks in practice [Foong et al., 2019b, Ober and Rasmussen, 2019]. This failure might stem from the disconnect between Bayesian parameter uncertainty and predictive uncertainty—while Bayesian methods focus on the posterior parameter uncertainty, the practical interest often lies in estimating the model's predictive uncertainty. This gap may be further exacerbated by the the restrictive independence assumptions [Trippe and Turner, 2017, Foong et al., 2019a, 2020] and the mode-seeking behavior of the evidence lower-bound (ELBO) [Bishop, 2006]. Sun et al. [2019] tries to bridge the gap by performing variational inference in the function space, albeit this requires additional approximations to the intractable functional KL divergence. In contrast, the density uncertainty criterion directly imposes a constraint on the model's predictive uncertainty, so that the predictive uncertainty is always grounded in the training density of the input and provides reliable estimates of predictive uncertainty.

Uncertainty Estimation for Deep Learning This paper proposes a new methodology for estimating uncertainty in deep learning models. Other alternatives for uncertainty estimation include Monte Carlo dropout [Gal and Ghahramani, 2016], which interprets dropout regularization as approximate Bayesian inference and estimates predictive uncertainty by performing Monte Carlo integration using dropout at test time. Variational Gaussian dropout Kingma et al. [2015], motivated by a continuous approximation to dropout [Wang and Manning, 2013], applies multiplicative noise to the preactivations. The authors show this corresponds to assuming a degenerate parameter posterior distribution and propose a variational inference method for adapting the dropout rates. Recently, Dusenberry et al. [2020] propose to model only the uncertainty of rank-1 factors in the network's parameters by introducing dimension-wise multiplicative noise to both the input and the output at each layer.

On the other hand, deep kernel learning methods (DKL) [Snoek et al., 2015, Wilson et al., 2016, Liu et al., 2020, van Amersfoort et al., 2021] combine the expressive power of neural networks with the uncertainty estimation capability of Gaussian Processes (GPs) by using a deterministic neural network as a feature extractor and applying the GP in the resulting feature space. However, Ober et al. [2021] show that such construction is prone to overfitting as it ignores any uncertainty associated with the neural network feature extractor. To alleviate the problem, Liu et al. [2020], van Amersfoort et al. [2021] limit the expressive power of the neural network feature extractor using spectral normalization [Miyato et al., 2018] and apply approximations to the GP posterior to circumvent the cubic complexity of GP inference. While DKL methods provide interesting alternatives for uncertainty estimation, in this work we primarily focus on the BNN and the related methods [Blundell et al., 2015, Kingma et al., 2015, Gal and Ghahramani, 2016, Dusenberry et al., 2020] that model the uncertainty of the neural network as a whole.

4 Empirical Studies

In this section, we empirically demonstrate that the density uncertainty layer delivers reliable predictive uncertainty estimates compared to the existing approaches. The empirical studies are structured as follows:

1. We visualize the predictive uncertainty landscape of different uncertainty estimation methods and their failure modes, on a toy regression problem.
2. We evaluate the uncertainty estimation performance on CIFAR-10/100 classification benchmarks [Krizhevsky and Hinton, 2009] using the ResNet-14 [He et al., 2016] and the Wide ResNet-28 (WRN-28) [Zagoruyko and Komodakis, 2016] architecture.
3. We evaluate the out-of-distribution (OOD) detection performance on SVHN [Netzer et al., 2011] using the models trained on CIFAR-10/100.

In the appendix, we include the results on the UCI regression benchmarks using a MLP architecture.

We compare our method to the following popular uncertainty methods for deep learning:

1. **Mean-field Variational Inference** (MFVI) [Blundell et al., 2015] assumes fully factorized normal posteriors on the neural network parameters.
2. **Monte Carlo Dropout** (MCDropout) [Gal and Ghahramani, 2016] views dropout as approximate Bayesian inference and applies dropout at test time in order to estimate the predictive uncertainty.
3. **Variational Dropout** (VDropout) [Kingma et al., 2015] applies Gaussian multiplicative noise $\epsilon_\ell^j \sim \mathcal{N}(1, \alpha)$ to the output of linear layers.
4. **Rank-1 BNN** [Dusenberry et al., 2020] further extends Variational Dropout by introducing multiplicative noise for both the layer's input and output, dimension-wise. We assume Gaussian noise.

In addition, we include the last-layer GP methods of SNGP [Liu et al., 2020] and DUE [van Amersfoort et al., 2021] for the WRN-28 experiments. We use the DUE authors' implementation for SNGP and DUE.

Experimental details for CIFAR-10/100 For the CIFAR-10/100 experiments, we use the standard convolutional ResNet-14 architecture [He et al., 2016] and also the significantly larger WRN-28 architecture [Zagoruyko and Komodakis, 2016] with 2× depth and 10× width. We default to the full-rank Gaussian energy models using LDL parametrization but also include the efficient rank-1 mixture Gaussians for the larger WRN-28 experiments. We use the ADAM optimizer with learning rate 0.1 with batch size of 128 except for MFVI where the learning rate is reduced to 0.01 as higher learning rate led to divergence. We train the models for 200 epochs with cosine learning rate schedule without restarts [Loshchilov and Hutter, 2017]. During training, we apply random cropping and padding, and horizontal flipping data augmentations. The input pixel values are normalized using the training pixel means and standard deviations, channel-wise. We do not employ KL annealing or posterior tempering but initialize the posterior standard deviation to a sufficiently small value (e.g. 10^{-3}) to stabilize training [Dusenberry et al., 2020]. The weight decay is set to

Table 1: CIFAR-10 classification using the ResNet-14 architecture and 25 posterior samples. The average and the standard deviation over 3 random seeds are shown. Density Uncertainty significantly reduces the calibration error

Method	Accuracy (↑)	ECE (↓)	NLL (↓)
MFVI	91.9 ± 0.1	0.080 ± 0.002	0.291 ± 0.004
MCDropout	91.0 ± 0.1	0.016 ± 0.001	0.261 ± 0.001
VDropout	92.2 ± 0.0	0.013 ± 0.001	0.233 ± 0.002
Rank-1 BNN	92.2 ± 0.0	0.017 ± 0.002	0.231 ± 0.002
Density Uncertainty	92.2 ± 0.3	**0.004 ± 0.000**	**0.226 ± 0.003**

Table 2: CIFAR-100 classification using the ResNet-14 architecture and 25 posterior samples. The average and the standard deviation over 3 random seeds are shown. Density Uncertainty improves all three metrics

Method	Accuracy (↑)	ECE (↓)	NLL (↓)
MFVI	67.9 ± 0.4	0.132 ± 0.002	1.212 ± 0.017
MCDropout	66.3 ± 0.3	0.066 ± 0.001	1.205 ± 0.005
VDropout	67.8 ± 0.1	0.051 ± 0.002	1.167 ± 0.002
Rank-1 BNN	68.0 ± 0.5	0.041 ± 0.003	1.132 ± 0.002
Density Uncertainty	**68.8 ± 0.2**	**0.011 ± 0.003**	**1.110 ± 0.006**

0.0001 for CIFAR-10 and 0.0002 for CIFAR-100 experiments. For WRN-28, we use weight decay of 0.0005, learning rate of 0.05, and batch size of 64.

Model-specific hyperparameters are searched on a grid on a randomly sampled held-out set of CIFAR-10. The dropout rate for Monte Carlo Dropout is searched over $\{0.1, 0.2, 0.3\}$ and set to 0.1. For Variational Dropout, we find that adapting the noise variance leads to underfitting and thus fix the noise variance. The multiplicative noise variance is searched over $\{0.1, 0.25, 0.5\}$ and set to 0.1. For Rank-1 BNN, the prior standard deviation for the multiplicative noise distributions is searched over $\{0.01, 0.1, 1\}$ and set to 0.1. For Density Uncertainty, the prior noise standard deviation is searched over $\{0.1, 1, 10\}$ and set to 1.

All experiments are implemented in PyTorch [Paszke et al., 2019] and executed on a single Titan X GPU. The code is available at `https://github.com/yookoon/density_uncertainty_layers`

4.1 Toy Regression

We generate a toy regression problem in 1-D to illustrate the predictive uncertainty landscapes of different uncertainty estimation methods. We uniformly sample x_i from $[-4, -2] \cup [2, 4]$ and generate the target as $y_i = x_i^3 + \epsilon$. This leaves a in-between low density region in $[-2, 2]$. We normalize the input and the target to have zero-mean and unit variance. We use a MLP with one hidden layer of width 50 as the base architecture.

The predictive uncertainty landscapes on a toy regression problem are visualized in Figure 1. The baselines fail to produce reliable in-between uncertainty in the low-density region around the origin. For example, MFVI gives collapsed in-between uncertainty while Variational Dropout and Rank 1 BNNs produce flat uncertainty in the low-density region. Theses two methods both apply input-independent noise to the layers and lack a robust mechanism for adjusting their uncertainty depending on the empirical density of the input. In contrast, Density Uncertainty captures the density of the training data and produces higher predictive uncertainty in the low-density regions. This is because it derives its uncertainty from the energy model of the input, adhering to the density uncertainty criterion.

4.2 CIFAR-10 and CIFAR-100

For the CIFAR-10/100 experiments, we report accuracy, Expected Calibration Error (ECE) [Naeini et al., 2015] and Negative Log-Likelihood (NLL). ECE assesses the quality of the model's predictive uncertainty estimates by measuring how well-calibrated the model's predictions are. NLL is a proper scoring rule [Gneiting and Raftery, 2007] which favors predictions that are both accurate and well-calibrated.

The results using ResNet-14 are summarized in Table 1 and Table 2. On CIFAR-10, Density Uncertainty significantly reduces ECE compared to the baselines while maintaining comparable accuracy and slightly improving NLL. This indicates that Density Uncertainty provides more precise uncertainty estimates with lower calibration error. On CIFAR-100, Density Uncertainty

Table 3: CIFAR-10 classification using the WRN28 architecture and 25 posterior samples. The average and the standard deviation over 3 random seeds are shown. Density Uncertainty improves all three metrics

Method	Accuracy (↑)	ECE (↓)	NLL (↓)
MFVI	89.3 ± 1.4	0.310 ± 0.011	0.668 ± 0.024
MCDropout	96.0 ± 0.2	0.013 ± 0.001	0.135 ± 0.002
VDropout	96.2 ± 0.2	0.012 ± 0.001	0.128 ± 0.002
Rank-1 BNN	95.8 ± 0.1	0.011 ± 0.002	0.143 ± 0.004
SNGP	95.9 ± 0.1	0.015 ± 0.001	0.143 ± 0.002
DUE	95.6 ± 0.1	0.014 ± 0.006	0.170 ± 0.005
Density Uncertainty			
Full-rank LDL	96.4 ± 0.1	**0.010 ± 0.001**	0.119 ± 0.003
Rank-1 Mixture (1.25%)	**96.5 ± 0.0**	0.011 ± 0.001	**0.118 ± 0.001**

Table 4: CIFAR-100 classification using the WRN28 architecture and 25 posterior samples. The average and the standard deviation over 3 random seeds are shown. Density Uncertainty improves all three metrics

Method	Accuracy (↑)	ECE (↓)	NLL (↓)
MFVI	74.2 ± 0.8	0.260 ± 0.006	1.197 ± 0.023
MCDropout	80.2 ± 0.3	0.033 ± 0.003	0.766 ± 0.005
VDropout	81.1 ± 0.3	0.036 ± 0.002	0.756 ± 0.011
Rank-1 BNN	79.7 ± 0.2	0.032 ± 0.001	0.815 ± 0.008
SNGP	80.5 ± 0.3	0.035 ± 0.006	0.782 ± 0.017
DUE	Out of memory error		
Density Uncertainty			
Full-rank LDL	**82.3 ± 0.2**	**0.029 ± 0.003**	**0.684 ± 0.005**
Rank-1 Mixture (1.25%)	**82.3 ± 0.0**	0.032 ± 0.003	0.692 ± 0.001

improves the performance on metrics. Notably, Density Uncertainty again reduces ECE significantly. These results show that Density Uncertainty delivers the most reliable uncertainty estimates among the baselines.

Table 3 and Table 4 present the results using the larger WRN-28 architecture, including the additional last layer GP baselines of SNGP [Liu et al., 2020] and DUE [van Amersfoort et al., 2021]. For Density Uncertainty, we also experiment with the efficient rank-1 mixture Gaussians energy model to test the scalability of the method in addition to the default LDL parametrization. We set the number of mixture components to only 1.25% of the layers' width, adding only negligible amount of parameter and computational overhead to the neural network. Density Uncertainty brings the best performance in all three metrics. Surprisingly, the rank-1 mixture with only 1.25% number of components achieves performance comparable with the full-rank energy model, demonstrating the effectiveness of low-rank approximations in neural networks [Maddox et al., 2019, Dusenberry et al., 2020] and the scalability of the method. Table 5 shows the results on CIFAR-10 classification using the WRN28 architecture with varying number of rank-1 mixture components. We find that the performance is robust for a wide range of values, ranging from 1.25% to 20%, and is comparable to the full-rank LDL model.

4.3 Out-of-Distribution Detection on SVHN

Can we deploy the energy models of the density uncertainty layers for detecting out-of-distribution (OOD) inputs? Table 6 summarizes the the OOD detection performance on SVHN, using the models trained on CIFAR-10/100. We report the area under the precision-recall curve (AUPRC) and the receiver operator characteristic (AUROC). Following the previous work [Ritter et al., 2021], the baselines use the maximum predicted probability heuristic for OOD detection. On the other hand, Density Uncertainty, equipped with layer-wise generative energy models, can inherently perform OOD detection using the energy statistics. Specifically, the energy at layer ℓ can be represented as the squared

Table 5: Varying the number of components for rank-1 Gaussian mixture energy on CIFAR-10 classification using the WRN28 architecture. The average and the standard deviation over 3 random seeds are shown. The performance of rank-1 Gaussian mixure is robust for a wide range of values

Method	Accuracy (↑)	ECE (↓)	NLL (↓)
Rank-1 Mixture (1.25%)	96.5 ± 0.0	0.011 ± 0.001	0.118 ± 0.001
Rank-1 Mixture (2.5%)	96.4 ± 0.1	0.012 ± 0.001	0.124 ± 0.002
Rank-1 Mixture (5%)	96.4 ± 0.1	0.012 ± 0.001	0.122 ± 0.001
Rank-1 Mixture (10%)	96.5± 0.0	0.012 ± 0.001	0.122 ± 0.001
Rank-1 Mixture (20%)	96.5± 0.0	0.013 ± 0.000	0.121 ± 0.003
Full-rank LDL	96.4 ± 0.1	0.010 ± 0.001	0.119 ± 0.003

Table 6: Out-of-distribution detection performance on SVHN using ResNet-14 models trained on CIFAR-10/100. The average and the standard deviation over 3 random seeds are shown. Using the energy statistics, Density Uncertainty gives the most robust out-of-distribution detection performance

	CIFAR-10 → SVHN		CIFAR-100 → SVHN	
Method	AUPRC (↑)	AUROC (↑)	AUPRC (↑)	AUROC (↑)
MFVI	0.903 ± 0.009	0.830 ± 0.014	0.803 ± 0.008	0.640 ± 0.017
MCDropout	0.892 ± 0.006	0.832 ± 0.010	0.817 ± 0.005	0.666 ± 0.009
VDropout	0.917 ± 0.011	0.866 ± 0.018	0.822 ± 0.028	0.677 ± 0.052
Rank-1 BNN	0.925 ± 0.003	0.880 ± 0.001	0.822 ± 0.027	0.681 ± 0.047
Density Uncertainty	**0.952 ± 0.026**	**0.893 ± 0.056**	**0.908 ± 0.015**	**0.800 ± 0.024**

sum of random variables:

$$E_\ell(h_\ell) = \sum_{j=1}^{D} (z_\ell^j)^2 \text{ where } z_\ell = \Sigma_\ell^{-\frac{1}{2}} h_\ell, \tag{32}$$

where z_ℓ can be thought of as whitened input with $\Sigma_\ell^{-1/2}$ decorrelating the layer's input h_ℓ. Based on the NN-GP equivalence in the infinite width limit, we expect the energy to be approximately normally-distributed. Therefore, we use the squared deviation of energy from the in-distribution average

$$|E_\ell(h_\ell) - \mu_\ell|^2 \tag{33}$$

as a test statistic for OOD detection where the in-distribution average energy μ_ℓ is estimated using a in-distribution held-out set. We use the energy of the last convolutional layer as it reflects the most high-level, semantic information of the input. Table 6 demonstrates that Density Uncertainty can most robustly detect OOD input among the baselines.

5 Conclusion

We proposed a novel density criterion for reliable uncertainty estimation, asserting that the predictive uncertainty of a model should be grounded in the empirical density of the input. A model that adheres to the criterion will produce higher uncertainty for inputs that are improbable in the training data, and lower uncertainty for those inputs that are more probable. We formalized the concept as a constraint on the predictive variance of a stochastic function and developed the density uncertainty layer as a flexible building block for uncertainty-aware deep learning. Through the empirical studies, we demonstrated that the proposed method provides the most reliable uncertainty estimates and robust out-of-distribution detection performance among the baselines. This could have practical applications in various fields where robust uncertainty estimation is crucial, such as medical diagnosis, autonomous driving, and financial forecasting.

References

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer-Verlag New York, Inc., Secaucus, NJ, 2006.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *ICML*, 2015.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *ICML*, 2014.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *ICML*, 2020.

Andrew YK Foong, David R Burt, Yingzhen Li, and Richard E Turner. Pathologies of factorised gaussian and mc dropout posteriors in bayesian neural networks. In *NeurIPS 4th Workshop on Bayesian Deep Learning*, 2019a.

Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 'in-between'uncertainty in bayesian neural networks. In *ICML Uncertainty and Robustness in Deep Learning Workshop*, 2019b.

Andrew YK Foong, David R Burt, Yingzhen Li, and Richard E Turner. On the expressiveness of approximate inference in bayesian neural networks. In *NeurIPS*, 2020.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

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

Alex Graves. Practical variational inference for neural networks. In *NeurIPS*, 2011.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *NeurIPS*, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto, 2009.

Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *NeurIPS*, 2020.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *ICML*, 2016.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. 2019.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. 2018.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015.

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

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

Sebastian W Ober and Carl Edward Rasmussen. Benchmarking the neural linear model for regression. In *AABI*, 2019.

Sebastian W Ober, Carl E Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. In *UAI*, 2021.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *ICLR*, 2018.

Hippolyt Ritter, Martin Kukla, Cheng Zhang, and Yingzhen Li. Sparse uncertainty representation in deep learning with inducing weights. In *NeurIPS*, 2021.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *NeurIPS*, 2005.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *ICML*, 2015.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. In *ICLR*, 2019.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *AISTATS*, 2009.

Brian Trippe and Richard Turner. Overpruning in variational bayesian neural networks. In *NeurIPS Approximate Bayesian Inference Workshop*, 2017.

Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *NeurIPS*, 2016.

Sida Wang and Christopher Manning. Fast dropout training. In *ICML*, 2013.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *AISTATS*, 2016.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [No]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [No]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A UCI Regression Results

We perform additional experiments on the UCI regression benchmarks [Dua and Graff, 2017], one of the standard uncertainty estimation benchmarks for Bayesian neural networks. We follow the experiment protocol of

Table 7: Test NLL on UCI Regression. Lower is better. The average and the standard deviation over the 20 random train-test splits are shown

Dataset	MCDropout	VDropout	Rank1-BNN	Density Uncertainty
Boston Housing	2.794 ± 0.141	2.815 ± 0.092	2.588 ± 0.236	**2.523** ± 0.205
Concrete Strength	3.533 ± 0.037	3.497 ± 0.050	3.112 ± 0.081	**3.093** ± 0.116
Energy Efficiency	2.604 ± 0.065	2.525 ± 0.057	2.044 ± 0.099	**2.034** ± 0.087
Kin8nm	-0.590 ± 0.012	-0.792 ± 0.015	-1.194 ± 0.028	**-1.234** ± 0.036
Naval Propulsion	-3.437 ± 0.048	-4.019 ± 0.008	-4.956 ± 0.033	**-5.274** ± 0.036
Protein Structure	2.935 ± 0.006	2.852 ± 0.005	**2.771** ± 0.010	2.821 ± 0.001
Wine Quality Red	0.952 ± 0.062	**0.950** ± 0.067	0.954 ± 0.108	0.981 ± 0.109
Yacht Hydrodynamics	3.205 ± 0.062	3.206 ± 0.071	2.712 ± 0.084	**2.593** ± 0.067
Year Prediction MSD	3.568 ± NA	**3.558** ± NA	3.566 ± NA	3.570 ± NA

Table 8: Test RMSE on UCI Regression. Lower is better. The average and the standard deviation over the 20 random train-test splits are shown

Dataset	MCDropout	VDropout	Rank1-BNN	Density Uncertainty
Boston Housing	3.715 ± 0.952	3.710 ± 0.729	3.212 ± 0.758	**2.957** ± 0.606
Concrete Strength	7.589 ± 0.630	7.154 ± 0.672	5.348 ± 0.552	**5.290** ± 0.639
Energy Efficiency	3.073 ± 0.360	3.209 ± 0.429	1.718 ± 0.246	**1.690** ± 0.268
Kin8nm	0.121 ± 0.003	0.094 ± 0.003	0.073 ± 0.003	**0.070** ± 0.002
Naval Propulsion	0.007 ± 0.000	0.003 ± 0.000	0.010 ± 0.000	**0.001** ± 0.000
Protein Structure	4.549 ± 0.033	4.123 ± 0.028	**3.894** ± 0.046	4.077 ± 0.043
Wine Quality Red	0.626 ± 0.046	0.626 ± 0.049	**0.623** ± 0.051	0.630 ± 0.050
Yacht Hydrodynamics	5.060 ± 1.328	4.820 ± 1.175	3.070 ± 0.082	**2.505** ± 0.060
Year Prediction MSD	8.726 ± NA	**8.700** ± NA	8.712 ± NA	8.710 ± NA

Gal and Ghahramani [2016]. The benchmark includes regression datasets of varying size ranging from 300 to 515K, and input dimensions ranging from 4 to 90, as summarized in Table 9.

Experiment details We follow the experiment protocol of Gal and Ghahramani [2016]. We report the negative log-likelihood (NLL) and the root mean squared error (RMSE) on the test split using 10 posterior samples. For each dataset, the results are averaged over 20 random train-test splits of the data (except for Protein which uses 5 splits and Year which uses a single split). We normalize the input using the mean and the standard deviation of the training split. We use MLPs with two hidden layers of width 50. We increase the width to 100 for the larger datasets of Protein and Year. All models are trained for 100 epochs with learning rate of 0.01 using momentum optimizer. We use batch size of 128 and weight decay of 0.0001. We treat the output variance as a trainable parameter. In order to minimize hyperparameter tuning, we use the same hyperparameters used in the CIFAR-10/100 experiments except that we initialize the posterior standard deviations to a higher value of 0.1 as lower values led to overfitting.

Results Table 7 and Table 8 present the test NLL and the test RMSE on the datasets. Overall, Density Uncertainty gives the best results, outperforming the baselines on 6 of 9 datasets in both NLL and RMSE. These results demonstrate that density uncertainty layers can also bring benefits for regression problems potentially in low-data regimes besides the natural image classification problems studied in the paper. However, a caveat is that the variance of results are high on some datasets due to their small sizes and the performance on these datasets can be sensitive to the hyperparameters.

B Proof of Proposition 1

For brevity, we omit the noise variable ϵ_ℓ and the parameter ϕ_ℓ from the layer $f_\ell(h_\ell; \epsilon_\ell; \phi_\ell)$. First decompose the network output variance as

$$\begin{aligned}
&\mathrm{Var}[a^j_{L+1}|x] && (34)\\
&= \mathrm{Var}[a^j_L + f^j_L(h_L)|x] && (35)\\
&= \mathrm{Var}[a^j_L|x] + \mathrm{Var}[f^j_L(h_L)|x] + 2\mathrm{Cov}[a^j_L, f^j_L(h_L)|x]
\end{aligned}$$

The second term can be decomposed as

$$\begin{aligned}
&\mathrm{Var}[f^j_L(h_L)|x] && (36)\\
&= \mathbb{E}[\mathrm{Var}[f^j_L(h_L)|h_L]] + \mathrm{Var}[\mathbb{E}[f^j_L(h_L)|h_L]]. && (37)
\end{aligned}$$

Each of these components are bounded as

$$\alpha\mathbb{E}[E_L(h_L)] \le \mathbb{E}[\mathrm{Var}[f^j_L(h_L)|h_L]] \le \beta\mathbb{E}[E_L(h_L)],$$

by assumption, and

$$0 \le \mathrm{Var}[\mathbb{E}[f^j_L(h_L)|h_L]] \le MD \max_k \mathrm{Var}[a^k_L|x] \quad (38)$$

Table 9: The size and the dimensionality of the UCI regression datasets

Dataset	Size	Dimension
Boston Housing	506	13
Concrete Strength	1,030	8
Energy Efficiency	768	8
Kin8nm	8,192	8
Naval Propulsion	11,934	16
Protein Structure	45,730	9
Wine Quality Red	1,599	11
Yacht Hydrodynamics	308	6
Year Prediction MSD	515,345	90

On the other hand, the covariance term is bounded as

$$\left|\mathrm{Cov}[a_L^j, f_L^j(h_L)|x]\right| \leq \sqrt{M} D \max_k \mathrm{Var}[a_L^k|x] \tag{39}$$

If there exist C_L, C_L' such that for all j,

$$C_L \mathbb{E}[E(x_1, ..., h_{L-1})] \leq \mathrm{Var}[a_L^j|x] \tag{40}$$

and

$$\mathrm{Var}[a_L^j|x] \leq C_L' \, \mathbb{E}[E(x_1, ..., h_{L-1})]. \tag{41}$$

Combining the above, we have

$$\mathrm{Var}[a_{L+1}^j|x] \geq C \, \mathbb{E}[E(x_1, ..., h_L)], \tag{42}$$

for some constant C by induction.

C Bayesian Uncertainty in Linear Classification

In the paper, we demonstrated that the Bayesian uncertainty for linear regression is grounded in the density estimate of input. But how about in classification? We show that the Bayesian uncertainty in classification is also based on a density estimate but with a weighted generative objective.

Consider a logistic classification problem, with the input $\mathbf{X} \in \mathbb{R}^{N \times D}$, the target $\mathbf{y} \in \mathbb{R}^N$, and the weight $w \in \mathbb{R}^D$:

$$p(w) = \mathcal{N}(w|0, \alpha^{-1} I), \tag{43}$$

$$p(\mathbf{y}|\mathbf{X}, w) = \prod_{i=1}^N y_i^{\sigma(w^T x_i)} (1 - y_i)^{1 - \sigma(w^T x_i)}, \tag{44}$$

where σ is the logistic sigmoid function: $\sigma(x) = 1/(1 + e^{-x})$. Although the exact posterior distribution is intractable in this case, we can obtain a Gaussian approximation using the Laplace's approximation [Bishop, 2006]:

$$q(w) = \mathcal{N}(w|\mu_{\mathrm{MAP}}, \Lambda^{-1}), \tag{45}$$

$$\Lambda = \sum_{i=1}^N \sigma(w^T x_i)(1 - \sigma(w^T x_i)) x_i x_i^T + \alpha I, \tag{46}$$

and μ_{MAP} is the MAP estimate of the weight. Comparing the posterior precision (Equation (46)) to that of the regression in the paper $\Lambda = \beta \sum_i x_i x_i^T + \alpha I$, we find that the precision matrix in classification is a *weighted* estimate of the input covariance. Noting that the weight $\sigma(w^T x_i)(1 - \sigma(w^T x_i))$ is higher for inputs with more uncertain predictions (i.e., $\sigma(w^T x_i)$ is closer to 0.5), Bayesian classification prioritizes inputs that give higher prediction uncertainty, in contrast to the regression case where all inputs were weighted uniformly. Recall that the posterior precision essentially serves as a Gaussian density estimate of the input. Equation (46) shows that Bayesian logistic classification performs weighted generative modeling of the input, and by prioritizing inputs that are more informative, it can potentially better utilize the capacity of the Gaussian energy model.

Despite this finding, in this work we choose to use the unweighted generative objective for density uncertainty layers because (1) the weighted objective can lead to an unfaithful estimate of the input density and (2) the capacity of the generative model is not a major concern for density uncertainty layers, as the complexity of the uncertainty landscape naturally grows with the number of layers in the network.