Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness

Jeremiah Zhe Liu[*]
Google Research & Harvard University
jereliu@google.com

Zi Lin[†]
Google Research
lzi@google.com

Shreyas Padhy[†]
Google Research
shreyaspadhy@google.com

Dustin Tran
Google Research
trandustin@google.com

Tania Bedrax-Weiss
Google Research
tbedrax@google.com

Balaji Lakshminarayanan
Google Research
balajiln@google.com

Abstract

Bayesian neural networks and deep ensembles are principled approaches to estimate the predictive uncertainty of a deep learning model. However their practicality in real-time, industrial-scale applications are limited due to their heavy memory and inference cost. This motivates us to study principled approaches to high-quality uncertainty estimation that require only a single deep neural network (DNN). By formalizing the uncertainty quantification as a minimax learning problem, we first identify *distance awareness*, i.e., the model's ability to properly quantify the distance of a testing example from the training data manifold, as a necessary condition for a DNN to achieve high-quality (i.e., minimax optimal) uncertainty estimation. We then propose *Spectral-normalized Neural Gaussian Process (SNGP)*, a simple method that improves the distance-awareness ability of modern DNNs, by adding a weight normalization step during training and replacing the output layer with a Gaussian Process. On a suite of vision and language understanding tasks and on modern architectures (Wide-ResNet and BERT), SNGP is competitive with deep ensembles in prediction, calibration and out-of-domain detection, and outperforms the other single-model approaches.[3]

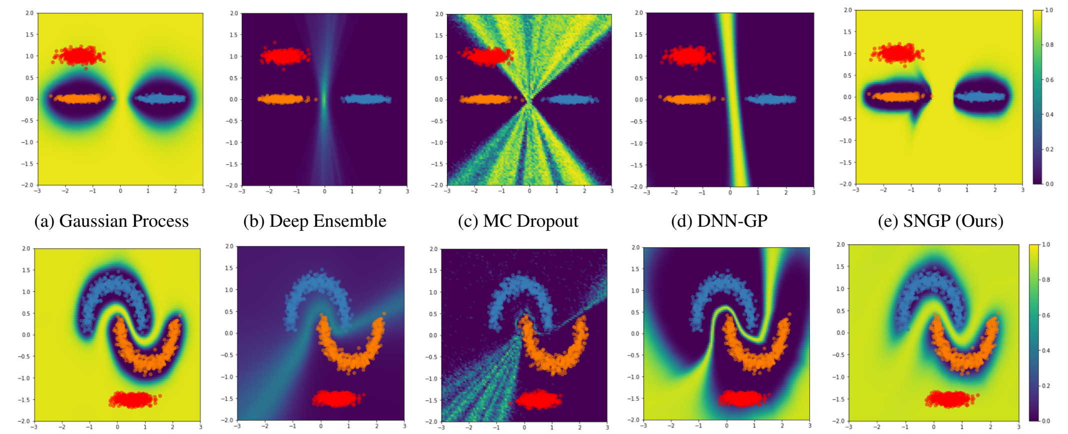
1 Introduction

Efficient methods that reliably quantify a deep neural network (DNN)'s predictive uncertainty are important for industrial-scale, real-world applications, which include examples such as object recognition in autonomous driving [22], ad click prediction in online advertising [76], and intent understanding in a conversational system [84]. For example, for a natural language understanding (NLU) model built for a domain-specific chatbot service (e.g, weather inquiry), the user's input utterance to the model can be of any topic, and the model needs to understand reliably and in real-time whether to abstain or to trigger one of its known APIs.

When deep classifiers make predictions on input examples that are far from the support of the training set, their performance can be arbitrarily bad [4, 14]. This motivates the need for methods that are aware of the distance between an input test example and previously seen training examples, so they can return a uniform (i.e., maximum entropy) distribution over output labels if the input is too far from the training set (i.e., the input is out-of-domain) [30]. Gaussian processes (GPs) with suitable kernels enjoy such a property. However, to apply Gaussian processes to a high-dimensional machine

[*]Work done at Google Research.
[†]Work done as an Google AI Resident.
[3]Code available at https://github.com/google/uncertainty-baselines/tree/master/baselines.

learning problem, it is usually necessary to perform some form of feature extraction or dimensionality reduction using a DNN. Ideally, the hidden representation of a DNN should reflect a meaningful distance in the data manifold (e.g., the semantic textual similarity between two sentences), such that this "distance aware" property is preserved. However, as we will show in the experiments, this is often not guaranteed for common deep learning models (cf. Figure 1).



(a) Gaussian Process (b) Deep Ensemble (c) MC Dropout (d) DNN-GP (e) SNGP (Ours)

(f) Gaussian Process (g) Deep Ensemble (h) MC Dropout (i) DNN-GP (j) SNGP (Ours)

Figure 1: The uncertainty surface of a GP and different DNN approaches on the *two ovals* (Top Row) and *two moons* (Bottom Row) 2D classification benchmarks. SNGP is the only DNN-based approach achieving a distance-aware uncertainty similar to the gold-standard GP. Training data for positive (**Orange**) and negative classes (**Blue**). OOD data (**Red**) not observed during training. Background color represents the estimated model uncertainty (See 1e and 1j for color map). See Section 5.1 for details.

We propose a simple solution to this problem, namely adding *spectral normalization* to the weights in each (residual) layer [54]. We refer to our method as "Spectral-normalized Neural Gaussian Processes" (SNGP). We show that this provides bounds on $||h(\mathbf{x}) - h(\mathbf{x}')||_H$ relative to $||\mathbf{x} - \mathbf{x}'||_X$, where $\mathbf{x}$ and $\mathbf{x}'$ are two inputs, $h(\mathbf{x})$ is a deep feature extractor, and $||.||_X$ a semantically meaningful distance for the data manifold. We can then safely pass $h(\mathbf{x})$ into a distance-aware GP output layer. To ensure computational scalability, we approximate the GP posterior using a Laplace approximation to the random feature expansion of the GP, which gives rise to a model posterior that can be learned scalably and in closed-form with minimal modification to the training pipeline of a deterministic DNN, and allows us to efficiently compute the predictive uncertainty on a per-input basis without Monte Carlo sampling.

In the rest of this paper, we first theoretically motivate the importance of *distance awareness* for a model's ability uncertainty estimation by studying it as a minimax learning problem (Section 2). We then introduce our SNGP method in detail in Section 3, and experimentally evaluate its performance against other single-model approaches as well as deep ensembles in Section 5 [42]. On two challenging real world problems, namely image classification (using a Wide Resnet model on CIFAR-10 and CIFAR-100) and conversational intent understanding (using a BERT model on CLINC out-of-scope (OOS) intent dataset), we show that the SNGP method attains an uncertainty performance (e.g., calibration and out-of-domain (OOD) detection) that is competitive with that of a deep ensemble, while maintaining the accuracy and latency of a single deterministic DNN.

2 Distance Awareness: An Important Condition for High-Quality Uncertainty Estimation

Notation and Problem Setup Consider a data-generation distribution $p^*(y|\mathbf{x})$, where $y \in \{1,\dots,K\}$ is the space of K-class labels, and $\mathbf{x} \in \mathscr{X} \subset \mathbb{R}^d$ is the input data manifold equipped with a suitable metric $||.||_X$. In practice, the training data $\mathscr{D} = \{y_i, \mathbf{x}_i\}_{i=1}^N$ is often collected from a subset of the full input space $\mathscr{X}_{\texttt{IND}} \subset \mathscr{X}$. As a result, the full data-generating distribution $p^*(y|\mathbf{x})$ is in fact a mixture of an in-domain (IND) distribution $p_{\texttt{IND}}(y|\mathbf{x}) = p^*(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}})$ and also an OOD distribution $p_{\texttt{OOD}}(y|\mathbf{x}) = p^*(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}})$ [52, 66]:

$$\begin{aligned} p^*(y|\mathbf{x}) &= \qquad p^*(y, \mathbf{x} \in \mathscr{X}_{\texttt{IND}}|\mathbf{x}) \qquad + \qquad p^*(y, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}}|\mathbf{x}) \\ &= p^*(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}}) * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + p^*(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}}) * p^*(\mathbf{x} \notin \mathscr{X}_{\texttt{IND}}). \end{aligned} \tag{1}$$

During training, the model learns the in-domain distribution $p^*(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}})$ from the data $\mathscr{D}$, but does not have knowledge about $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}})$. In the weather-service chatbot example, the out-of-domain space $\mathscr{X}_{\texttt{OOD}} = \mathscr{X} / \mathscr{X}_{\texttt{IND}}$ is the space of all natural utterances not related to weather queries, whose elements usually do not have a meaningful correspondence with the in-domain intent labels $y_k \in \{1, \dots, K\}$. Therefore, the out-of-domain distribution $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}})$ can be very different from the in-domain distribution $p^*(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}})$, and we only expect the model to generalize well within $\mathscr{X}_{\texttt{IND}}$. However, during testing, the model needs to construct a predictive distribution $p(y|\mathbf{x})$ for the entire input space $\mathscr{X} = \mathscr{X}_{\texttt{IND}} \cup \mathscr{X}_{\texttt{OOD}}$, since the users' utterances can be of any topic.

2.1 Uncertainty Estimation as a Minimax Learning Problem

To formulate the uncertainty estimation as a learning problem under (1), we need to define a loss function to measure a model $p(y|\mathbf{x})$'s quality of predictive uncertainty. A popular uncertainty metric is Expected Calibration Error (ECE), defined as $C(p, p^*) = E\big[|E(y^* = \hat{y}|\hat{p} = p) - p|\big]$, which measures the difference in expectation between the model's predictive confidence (e.g., the maximum probability score) and its actual accuracy [29, 56]. However, ECE is not suitable as a loss function, since it is not uniquely minimized at $p = p^*$. Specifically, there can exist a trivial predictor that ignores the input example and achieves perfect calibration by predicting randomly according to the marginal distribution of the labels [24].

To this end, a theoretically more well-founded uncertainty metric is to examine *strictly proper scoring rules* [25] $s(., p^*)$, which are uniquely minimized by the true distribution $p = p^*$. Examples include log-loss and Brier score. Proper scoring rules are related to ECE in that each is an upper bound of the calibration error by the classic calibration-refinement decomposition [10]. Therefore, minimizing a proper scoring rule implies minimizing the calibration error of the model. Consequently, we can formalize the problem of uncertainty quantification as the problem of constructing an optimal predictive distribution $p(y|\mathbf{x})$ to minimize the expected risk over the entire $\mathbf{x} \in \mathscr{X}$, i.e., an *Uncertainty Risk Minimization* problem:

$$\inf_{p \in \mathscr{P}} S(p, p^*) = \inf_{p \in \mathscr{P}} \underset{\mathbf{x} \in \mathscr{X}}{E} \big[s(p, p^*|\mathbf{x})\big]. \tag{2}$$

Unfortunately, directly minimizing (2) over the entire input space $\mathscr{X}$ is not possible even with infinite amounts of data. This is because since the data is collected only from $\mathscr{X}_{\texttt{IND}}$, the true OOD distribution $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}})$ is never learned by the model, and generalization is not guaranteed since $p^*(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}})$ and $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}})$ are not assumed to be similar. As a result, the naive practice of using a model trained only with in-domain data to generate OOD predictions can lead to arbitrarily bad results, since nature can happen to produce an OOD distribution $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}})$ that is at odds with the model prediction. This is clearly undesirable for safety-critical applications. To this end, a more prudent strategy is to instead minimize the *worst-case* risk with respect to all possible $p^* \in \mathscr{P}^*$, i.e., construct $p(y|x)$ to minimize the *Minimax Uncertainty Risk*:

$$\inf_{p \in \mathscr{P}} \Big[\sup_{p^* \in \mathscr{P}^*} S(p, p^*)\Big]. \tag{3}$$

In game-theoretic nomenclature, the uncertainty estimation problem acts as a two-player game of model v.s. nature, where the goal of the model is to produce a minimax strategy p that minimizes the risk $S(p, p^*)$ against all possible (even adversarial) moves p^* of nature. Under the classification task and for Brier score, the solution to the minimax problem (3) adopts a simple and elegant form:

$$p(y|\mathbf{x}) = p(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}}) * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + p_{\texttt{uniform}}(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}}) * p^*(\mathbf{x} \notin \mathscr{X}_{\texttt{IND}}). \tag{4}$$

This is very intuitive: if an input point is in the training data domain, trust the model, otherwise use a uniform (maximum entropy) prediction. For the practice of uncertainty estimation, (4) is conceptually important in that it verifies that there exists a unique optimal solution to the uncertainty estimation problem (3). Furthermore, this optimal solution can be constructed conveniently. Specifically, it can be constructed as a mixture of a discrete uniform distribution $p_{\texttt{uniform}}$ and the in-domain predictive distribution $p(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}})$ that the model has already learned from data, *assuming one can quantify* $p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}})$ *well*. In fact, the expression (4) can be shown to be optimal for a broad family of scoring rules known as the Bregman score, which includes the Brier score and the widely used *log score* as the special cases. We derive (4) in Appendix B.

2.2 Input Distance Awareness as a Necessary Condition

In light of Equation (4), a key capacity for a deep learning model to reliably estimate predictive uncertainty is its ability to quantify, either explicitly or implicitly, the domain probability $p(\mathbf{x} \in \mathscr{X}_{\texttt{IND}})$.

This requires the model to have a good notion of the distance (or dissimilarity) between a testing example $\mathbf{x}$ and the training data $\mathscr{X}_{\texttt{IND}}$ with respect to a *meaningful* distance $||.||_X$ for the data manifold (e.g., semantic textual similarity [12] for language data). Definition 1 makes this notion more precise:

Definition 1 (Input Distance Awareness)**.** *Consider a predictive distribution $p(y|\mathbf{x})$ trained on a domain $\mathscr{X}_{\texttt{IND}} \subset \mathscr{X}$, where $(\mathscr{X}, ||.||_X)$ is the input data manifold equipped with a suitable metric $||.||_X$. We say $p(y|\mathbf{x})$ is input distance aware if there exists $u(\mathbf{x})$ a summary statistic of $p(y|\mathbf{x})$ that quantifies model uncertainty (e.g., entropy, predictive variance, etc) that reflects the distance between $\mathbf{x}$ and the training data with respect to $||.||_X$, i.e.,*

$$u(\mathbf{x}) = v\big(d(\mathbf{x}, \mathscr{X}_{\texttt{IND}})\big)$$

where v is a monotonic function and $d(\mathbf{x}, \mathscr{X}_{\texttt{IND}}) = E_{\mathbf{x}' \sim \mathscr{X}_{\texttt{IND}}} ||\mathbf{x} - \mathbf{x}'||^2_X$. is the distance between $\mathbf{x}$ and the training data domain.

A classic model that satisfies the *distance-awareness* property is a Gaussian process (GP) with a radial basis function (RBF) kernel. Its predictive distribution $p(y|\mathbf{x}) = softmax(g(\mathbf{x}))$ is a softmax transformation of the GP posterior $g \sim GP$ under the cross-entropy likelihood, and its predictive uncertainty can be expressed by the posterior variance $u(\mathbf{x}^*) = var(g(\mathbf{x}^*)) = 1 - \mathbf{k}^{*\top}\mathbf{V}\mathbf{k}^*$ for $\mathbf{k}^*_i = exp(-\frac{1}{2l}||\mathbf{x}^* - \mathbf{x}_i||^2_X)$ and $\mathbf{V}_{N\times N}$ a fixed matrix determined by data. Then $u(\mathbf{x}^*)$ increases monotonically toward 1 as $\mathbf{x}^*$ moves further away from $\mathscr{X}_{\texttt{IND}}$ [61]. In view of the expression (4), the *input distance awareness* property is important for both calibration and OOD detection. However, this property is not guaranteed for a typical deep learning model [33]. Consider a discriminative deep classifier with dense output layer $logit_k(\mathbf{x}) = h(\mathbf{x})^\top \beta_k$, whose model confidence (i.e., maximum predictive probability) is characterized by the magnitude of the class logits, which is defined by the inner product distances between the hidden representation $h(\mathbf{x})$ and the decision boundaries $\{\beta_k\}_{k=1}^K$ (see, e.g., Figure 1b-1c and 1g-1h). As a result, the model computes confidence for a $\mathbf{x}^*$ based not on its distance from the training data $\mathscr{X}_{\texttt{IND}}$, but based on its distance from the decision boundaries, i.e., the model uncertainty is not *input distance aware*.

Two Conditions for Input Distance Awareness in Deep Learning Notice that a deep learning model $logit(\mathbf{x}) = g \circ h(\mathbf{x})$ is commonly composed of a hidden mapping $h : \mathscr{X} \to \mathscr{H}$ that maps the input $\mathbf{x}$ into a hidden representation space $h(\mathbf{x}) \in \mathscr{H}$, and an output layer g that maps $h(\mathbf{x})$ to the label space. To this end, a DNN $logit(\mathbf{x}) = g \circ h(\mathbf{x})$ can be made *input distance aware* via a combination of two conditions: **(1)** make the output layer g *distance aware*, so it outputs an uncertainty metric reflecting distance in the hidden space $||h(\mathbf{x}) - h(\mathbf{x}')||_H$ (in practice, this can be achieved by using a GP with a shift-invariant kernel as the output layer), and **(2)** make the hidden mapping *distance preserving* (defined below), so that the distance in the hidden space $||h(\mathbf{x}) - h(\mathbf{x}')||_H$ has a meaningful correspondence to the distance $||\mathbf{x} - \mathbf{x}'||_X$ in the data manifold. From the mathematical point of view, this is equivalent to requiring h to satisfy the *bi-Lipschitz* condition [67]:

$$L_1 * ||\mathbf{x}_1 - \mathbf{x}_2||_X \leq ||h(\mathbf{x}_1) - h(\mathbf{x}_2)||_H \leq L_2 * ||\mathbf{x}_1 - \mathbf{x}_2||_X, \tag{5}$$

for positive and bounded constants $0 < L_1 < 1 < L_2$. It is worth noticing that for a deep learning model, the bi-Lipschitz condition (5) usually leads the model's hidden space to preserve a semantically meaningful distance in the input data manifold $\mathscr{X}$, rather than a naive metric such as the square distance in the pixel space. This is because that the upper Lipschitz bound $||h(\mathbf{x}_1) - h(\mathbf{x}_2)||_H \leq L_2 * ||\mathbf{x}_1 - \mathbf{x}_2||_X$ is an important condition for the adversarial robustness of a deep network, which prevents the hidden representations $h(\mathbf{x})$ from being overly sensitive to the semantically meaningless perturbations in the pixel space [65, 80, 75, 37, 71]. On the other hand, the lower Lipschitz bound $||h(\mathbf{x}_1) - h(\mathbf{x}_2)||_H \geq L_1 * ||\mathbf{x}_1 - \mathbf{x}_2||_X$ prevents the hidden representation from being unnecessarily invariant to the semantically meaningful changes in the input manifold [38, 77]. Combined together, the bi-Lipschitz condition essentially encourages h to be an approximately isometric mapping, thereby ensuring that the learned representation $h(\mathbf{x})$ has a robust and meaningful correspondence with the semantic properties of the input data $\mathbf{x}$. Although not stated explicitly, learning an approximately isometric and geometry-preserving mapping is a common goal in machine learning. For example, image classifiers strive to learn a mapping from image manifold to a hidden space that can be well-separated by a set of linear decision boundaries, and sentences encoders aim to project sentences into a vector space where the cosine distance reflects the semantic similarity in natural language. Finally, it is worth noting that preserving such approximate isometry in a neural network is possible even after significant dimensionality reduction [8, 32, 59, 64].

3 SNGP: A Simple Approach to Distance-aware Deep Learning

In this section we propose *Spectral-normalized Neural Gaussian Process (SNGP)*, a simple method to improve the *input distance awareness* ability of a modern residual-based DNN (e.g., ResNet, Transformer) by (1) making the output layer *distance aware* and (2) making the hidden layers *distance preserving*, as discussed in Section 2.2. Full method is summarized in Algorithms 1-2.

3.1 Distance-aware Output Layer via Laplace-approximated Neural Gaussian Process

To make the output layer $g : \mathcal{H} \to \mathcal{Y}$ distance aware, SNGP replaces the typical dense output layer with a Gaussian process (GP) with an RBF kernel, whose posterior variance at $\mathbf{x}^*$ is characterized by its L_2 distance from the training data in the hidden space. Specifically, given N training samples $\mathcal{D} = \{y_i, \mathbf{x}_i\}_{i=1}^N$ and denoting $h_i = h(\mathbf{x}_i)$, the Gaussian-process output layer $g_{N\times 1} = [g(h_1), \dots, g(h_N)]^\top$ follows a multivariate normal distribution *a priori*:

$$g_{N\times 1} \sim MVN(\mathbf{0}_{N\times 1}, \mathbf{K}_{N\times N}), \text{where } \mathbf{K}_{i,j} = \exp(-||h_i - h_j||_2^2/2), \tag{6}$$

and the posterior distribution is computed as $p(g|\mathcal{D}) \propto p(\mathcal{D}|g)p(g)$ where $p(g)$ is the GP prior in (6) and $p(\mathcal{D}|g)$ is the data likelihood for classification (i.e., the exponentiated cross-entropy loss). However, computing the exact Gaussian process posterior for a large-scale classification task is both analytically intractable and computationally expensive, In this work, we propose a simple approximation strategy for GP that is based on a Laplace approximation to the random Fourier feature (RFF) expansion of the GP posterior [61]. Our approach gives rise to a closed-form posterior that is end-to-end trainable with the rest of the neural network, and empirically leads to an improved quality in estimating the posterior uncertainty. Specifically, we first approximate the GP prior in (6) by deploying a low-rank approximation to the kernel matrix $\mathbf{K} = \Phi\Phi^\top$ using random features [60]:

$$g_{N\times 1} \sim MVN(\mathbf{0}_{N\times 1}, \Phi\Phi^\top_{N\times N}), \qquad \text{where} \qquad \Phi_{i,D_L\times 1} = \sqrt{2/D_L} * \cos(-\mathbf{W}_L h_i + \mathbf{b}_L), \tag{7}$$

where $h_i = h(\mathbf{x}_i)$ is the hidden representation in the penultimate layer with dimension D_{L-1}. Φ_i is the final layer with dimension D_L, it contains $\mathbf{W}_{L,D_L\times D_{L-1}}$ a fixed weight matrix whose entries are sampled i.i.d. from $N(0,1)$, and $\mathbf{b}_{L,D_L\times 1}$ a fixed bias term whose entries are sampled i.i.d. from $Uniform(0, 2\pi)$. As a result, for the k^{th} logit, the RFF approximation to the GP prior in (6) can be written as a neural network layer with fixed hidden weights $\mathbf{W}$ and learnable output weights β_k:

$$g_k(h_i) = \sqrt{2/D_L} * \cos(-\mathbf{W}_L h_i + \mathbf{b}_L)^\top \beta_k, \qquad \text{with prior} \qquad \beta_{k,D_L\times 1} \sim N(0, \mathbf{I}_{D_L\times D_L}). \tag{8}$$

Notice that conditional on h, $\beta = \{\beta_k\}_{k=1}^K$ is the only learnable parameter in the model. As a result, the RFF approximation in (8) reduces an infinite-dimensional GP to a standard Bayesian linear model, for which many posterior approximation methods (e.g., expectation propagation (EP)) can be applied [53]. In this work, we choose the Laplace method due to its simplicity and the fact that its posterior variance has a convenient closed form [61]. Briefly, the Laplace method approximates the RFF posterior $p(\beta|\mathcal{D})$ using a Gaussian likelihood centered around the maximum a posterior (MAP) estimate $\hat{\beta} = \text{argmax}_\beta\, p(\beta|\mathcal{D})$, such that $p(\beta_k|\mathcal{D}) \approx MVN(\hat{\beta}_k, \hat{\Sigma}_k = \hat{\mathbf{H}}_k^{-1})$, where $\hat{\mathbf{H}}_{k,(i,j)} = \frac{\partial^2}{\partial\beta_i\partial\beta_j} \log p(\beta_k|\mathcal{D})|_{\beta_k=\hat{\beta}_k}$ is the $D_L \times D_L$ Hessian matrix of the log posterior likelihood evaluated at the MAP estimates. Under the linear-model formulation of the RFF posterior, the posterior precision matrix (i.e., the inverse covariance matrix) adopts a simple expression $\hat{\Sigma}_k^{-1} = \mathbf{I} + \sum_{i=1}^N \hat{p}_{i,k}(1-\hat{p}_{i,k})\Phi_i\Phi_i^\top$, where $p_{i,k}$ is the model prediction $softmax(\hat{g}_i)$ under the MAP estimates $\hat{\beta} = \{\beta_k\}_{k=1}^K$ [61]. To summarize, the Laplace posterior for GP under the RFF approximation is:

$$\beta_k|\mathcal{D} \sim MVN(\hat{\beta}_k, \hat{\Sigma}_k), \quad \text{where} \quad \hat{\Sigma}_k^{-1} = \mathbf{I} + \sum_{i=1}^N \hat{p}_{i,k}(1-\hat{p}_{i,k})\Phi_i\Phi_i^\top. \tag{9}$$

During minibatch training, the posterior mean $\hat{\beta}$ is updated via regular stochastic gradient descent (SGD) with respect to the (unnormalized) log posterior $-\log p(\beta|\mathcal{D}) = -\log p(\mathcal{D}|\beta) + \frac{1}{2}||\beta||^2$ where $-\log p(\mathcal{D}|\beta)$ is the cross-entropy loss. The posterior precision matrix is updated cheaply as $\hat{\Sigma}_{k,t}^{-1} = (1-m) * \hat{\Sigma}_{k,t-1}^{-1} + m * \sum_{i=1}^M \hat{p}_{i,k}(1-\hat{p}_{i,k})\Phi_i\Phi_i^\top$ for a minibatch of size M and m a small scaling coefficient. This computation only needs to be performed by passing through training data once at the final epoch. As a result, the GP posterior (9) can be learned scalably and in closed-form with minimal modification to the training pipeline of a deterministic DNN. It is worth noting that the Laplace approximation to the RFF posterior is asymptotically exact by the virtue of the Bernstein-von Mises (BvM) theorem and the fact that (8) is a finite-rank model [16, 23, 46, 57].

3.2 Distance-preserving Hidden Mapping via Spectral Normalization

Replacing the output layer g with a Gaussian process only allows the model $logit(\mathbf{x}) = g \circ h(\mathbf{x})$ to be aware of the distance in the hidden space $||h(\mathbf{x}_1) - h(\mathbf{x}_2)||_H$. It is also important to ensure the hidden mapping h is *distance preserving* so that the distance in the hidden space $||h(\mathbf{x}) - h(\mathbf{x}')||_H$ has a meaningful correspondence to the distance in the input space $||\mathbf{x} - \mathbf{x}'||_X$. To this end, we notice that modern deep learning models (e.g., ResNets, Transformers) are commonly composed of residual blocks, i.e., $h(\mathbf{x}) = h_{L-1} \circ \cdots \circ h_2 \circ h_1(\mathbf{x})$ where $h_l(\mathbf{x}) = \mathbf{x} + g_l(\mathbf{x})$. For such models, there exists a simple method to ensure h is *distance preserving*: by bounding the Lipschitz constants of all nonlinear residual mappings $\{g_l\}_{l=1}^{L-1}$ to be less than 1. We state this result formally below:

Proposition 1 (Lipschitz-bounded residual block is distance preserving [3])**.** *Consider a hidden mapping $h : \mathscr{X} \to \mathscr{H}$ with residual architecture $h = h_{L-1} \circ \ldots h_2 \circ h_1$ where $h_l(\mathbf{x}) = \mathbf{x} + g_l(\mathbf{x})$. If for $0 < \alpha \leq 1$, all g_l's are α-Lipschitz, i.e., $||g_l(\mathbf{x}) - g_l(\mathbf{x}')||_H \leq \alpha||\mathbf{x} - \mathbf{x}'||_X \quad \forall(\mathbf{x}, \mathbf{x}') \in \mathscr{X}$. Then:*

$$L_1 * ||\mathbf{x} - \mathbf{x}'||_X \leq ||h(\mathbf{x}) - h(\mathbf{x}')||_H \leq L_2 * ||\mathbf{x} - \mathbf{x}'||_X,$$

where $L_1 = (1-\alpha)^{L-1}$ and $L_2 = (1+\alpha)^{L-1}$, i.e., h is distance preserving.

Proof is in Appendix E.1. The ability of a residual network to construct a geometry-preserving metric transform between the input space $\mathscr{X}$ and the hidden space $\mathscr{H}$ is well-established in learning theory and generative modeling literature, but the application of these results in the context of uncertainty estimation for DNN appears to be new [3, 5, 32, 64].

Consequently, to ensure the hidden mapping h is distance preserving, it is sufficient to ensure that the weight matrices for the nonlinear residual block $g_l(\mathbf{x}) = \sigma(\mathbf{W}_l\mathbf{x} + \mathbf{b}_l)$ to have spectral norm (i.e., the largest singular value) less than 1, since $||g_l||_{Lip} \leq ||\mathbf{W}_l\mathbf{x} + \mathbf{b}_l||_{Lip} \leq ||\mathbf{W}_l||_2 \leq 1$. In this work, we enforce the aforementioned Lipschitz constraint on g_l's by applying the *spectral normalization (SN)* on the weight matrices $\{\mathbf{W}_l\}_{l=1}^{L-1}$ as recommended in [5]. Briefly, at every training step, the SN method first estimate the spectral norm $\hat{\lambda} \approx ||\mathbf{W}_l||_2$ using the power iteration method [26, 54], and then normalizes the weights as:

$$\mathbf{W}_l = \begin{cases} c * \mathbf{W}_l / \hat{\lambda} & \text{if } c < \hat{\lambda} \\ \mathbf{W}_l & \text{otherwise} \end{cases} \tag{10}$$

where $c > 0$ is a hyperparameter used to adjust the exact spectral norm upper bound on $||\mathbf{W}_l||_2$ (so that $||\mathbf{W}_l||_2 \leq c$). This hyperparameter is useful in practice since the other regularization mechanisms (e.g., Dropout, Batch Normalization) in the hidden layers can rescale the Lipschitz constant of the original residual mapping [26]. Therefore, (10) allows us more flexibility in controlling the spectral norm of the neural network weights so it is the most compatible with the architecture at hand.
Method Summary We summarize the method in Algorithms 1-2. As shown, for every minibatch step, the model first updates the hidden-layer weights $\{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L-1}$ and the trainable output weights $\beta = \{\beta_k\}_{k=1}^{K}$ via SGD, then performs spectral normalization, and finally (if in the final epoch) performs precision matrix update (Equation (9). We discuss further details (e.g. computational complexity) in Appendix A.

Algorithm 1 SNGP Training

```
1:  Input:
    Minibatches $\{D_i\}_{i=1}^N$ for $D_i = \{y_m, \mathbf{x}_m\}_{m=1}^M$.
2:  Initialize:
        $\hat{\Sigma} = \mathbf{I}, \mathbf{W}_L \overset{iid}{\sim} N(0,1), \mathbf{b}_L \overset{iid}{\sim} U(0, 2\pi)$
3:  for train_step = 1 to max_step do
4:      SGD update $\left\{\beta, \{\mathbf{W}_l\}_{l=1}^{L-1}, \{\mathbf{b}_l\}_{l=1}^{L-1}\right\}$
5:      Spectral Normalization $\{\mathbf{W}_l\}_{l=1}^{L-1}$ (10).
6:      if final_epoch then
7:          Update precision matrix $\{\hat{\Sigma}_k^{-1}\}_{k=1}^K$ (9).
8:      end if
9:  end for
10: Compute posterior covariance $\hat{\Sigma}_k = inv(\hat{\Sigma}_k^{-1})$.
```

Algorithm 2 SNGP Prediction

```
1: Input: Testing example $\mathbf{x}$.
2: Compute Feature:
       $\Phi_{D_L \times 1} = \sqrt{2/D_L} * \cos(\mathbf{W}_L h(\mathbf{x}) + \mathbf{b}_L)$,
3: Compute Posterior Mean:
       $\text{logit}_k(\mathbf{x}) = \Phi^\top \beta_k$
4: Compute Posterior Variance:
       $\text{var}_k(\mathbf{x}) = \Phi^\top \hat{\Sigma}_k \Phi$.
5: Compute Predictive Distribution:
       $p(y|\mathbf{x}) = \int_{m \sim N(\text{logit}(\mathbf{x}), \text{var}(\mathbf{x}))} \text{softmax}(m)$
```

4 Related Work

Single-model approaches to deep classifier uncertainty Recent work examines uncertainty methods that add few additional parameters or runtime cost to the base model. The state-of-the-art on large-scale tasks are efficient ensemble methods [79, 21], which cast a set of models under a single one, encouraging independent member predictions using low-rank perturbations. These methods are parameter-efficient but still require multiple forward passes from the model. SNGP investigates an orthogonal approach that improves the uncertainty quantification by imposing suitable regularization on a single model, and therefore requires only a single forward pass during inference. There exists other runtime-efficient, single-model approaches to estimate predictive uncertainty, achieved by either replacing the loss function [33, 50, 51, 68, 69], the output layer [6, 72, 11, 48], or computing a closed-form posterior for the output layer [62, 70, 41]. SNGP builds on these approaches by also considering the intermediate representations which are necessary for good uncertainty estimation, and proposes a simple method (spectral normalization) to achieve it. A recent method named Deterministic Uncertainty Quantification (**DUQ**) also regulates the neural network mapping but uses a two-sided gradient penalty [77]. The two-sided gradient penalty can be undesirable for a residual network, since imposing $||\nabla f|| = 1$ onto a residual connection $f(\mathbf{x}) = \mathbf{x} + g(\mathbf{x})$ can force $g(\mathbf{x})$ toward 0, leading to an identity mapping. We compare with DUQ in our experiments.

Laplace approximation and GP inference with DNN Laplace approximation has a long history in GP and NN literature [73, 17, 61, 49, 63], and the theoretical connection between a Laplace-approximated DNN and GP has being explored recently [40]. Differing from these works, SNGP applies the Laplace approximation to the posterior of a neural GP, rather than to a shallow GP or a dense-output-layer DNN. Earlier works that combine a GP with a DNN usually perform MAP estimation [11] or structured Variational Inference (VI) [9, 81]. These approaches were shown to lead to poor calibration by recent work [74], which proposed a simple fix by combing Monte Carlo Dropout (MC Dropout) with random Fourier features, which we term Calibrated Deep Gaussian Process (**MCD-GP**). SNGP differs from MCD-GP in that it considered a different regularization approach (spectral normalization) and can compute its posterior uncertainty more efficiently in a single forward pass. We compare with MCD-GP in our experiments. Appendix D contains further related work on distance-preserving neural networks and open-set classification.

5 Experiments

5.1 2D Synthetic Benchmark

We first study the behavior of the uncertainty surface of a SNGP model under a suite of 2D classification benchmarks. Specifically, we consider the *two ovals* benchmark (Figure 1, row 1) and the *two moons* benchmark (Figure 1, row 2). The *two ovals* benchmark consists of two near-flat Gaussian distributions, which represent the two in-domain classes (orange and blue) that are separable by a linear decision boundary. There also exists an OOD distribution (red) that the model doesn't observe during training. Similarly, the *two moons* dataset consists of two banana-shaped distributions separable by a nonlinear decision boundary. We consider a 12-layer, 128-unit deep architecture ResFFN-12-128. The full experimental details are in Appendix C.

Figure 1 shows the results, where the background color visualizes the uncertainty surface output by each model. We first notice that the shallow Gaussian process models (Figures 1a and 1f) exhibit an expected behavior for high-quality predictive uncertainty: it generates low uncertainty in $\mathscr{X}_{\texttt{IND}}$ that is supported by the training data (purple color), and generates high uncertainty when $\mathbf{x}$ is far from $\mathscr{X}_{\texttt{IND}}$ (yellow color), i.e., *input distance awareness*. As a result, the shallow GP model is able to assign low confidence to the OOD data (colored in red), indicating reliable uncertainty quantification. On the other hand, deep ensembles (Figures 1b, 1g) and MC Dropout (Figures 1c, 1h) are based on dense output layers that are not distance aware. As a result, both methods quantify their predictive uncertainty based on the distance from the decision boundaries, assigning low uncertainty to OOD examples even if they are far from the data. Finally, the DNN-GP (Figures 1d and1i) and SNGP (Figures 1e and1j) both use GP as their output layers, but with SNGP additionally imposing the spectral normalization on its hidden mapping $h(.)$. As a result, the DNN-GP's uncertainty surfaces are still strongly impacted by the distance from decision boundary, likely caused by the fact that the un-regularized hidden mapping $h(\mathbf{x})$ is free to discard information that is not relevant for prediction. On the other hand, the SNGP is able to maintain the *input distance awareness* property via its bi-Lipschitz constraint, and exhibits a uncertainty surface that is analogous to the gold-standard model (shallow GP) despite the fact that SNGP is based on a 12-layer network.

5.2 Vision and Language Understanding

Baseline Methods All methods included in the vision and language understanding experiments are summarized in Table 1. Specifically, we evaluate SNGP on a Wide ResNet 28-10 [83] for image classification, and BERT$_{\texttt{base}}$ [18] for language understanding. We compare against a deterministic baseline and two ensemble approaches: **MC Dropout** (with 10 dropout samples) and **deep ensembles** (with 10 models), all trained with a dense output layer and no spectral regularization. We consider three single-model approaches: **MCD-GP** (with 10 samples), **Deterministic Uncertainty Quantification (DUQ)** (see Section 4). For all models that use GP layer, we keep $D_L = 1024$ and compute predictive distribution by performing Monte Carlo averaging with 10 samples. We also include two ablated version of SNGP: **DNN-SN** which uses spectral normalization on its hidden weights and a dense output layer (i.e. distance preserving hidden mapping without distance-aware output layer), and **DNN-GP** which uses the GP as output layer but without spectral normalization on its hidden layers (i.e., distance-aware output layer without distance-preserving hidden mapping). Further experiment details and recommendations for practical implementation are in Appendix C. All baselines are built on the `uncertainty_baselines` framework.

Methods	Additional Regularization	Output Layer	Ensemble Training	Multi-pass Inference
Deterministic	-	Dense	-	-
MC Dropout	Dropout	Dense	-	Yes
Deep Ensemble	-	Dense	Yes	Yes
MCD-GP	Dropout	GP	-	Yes
DUQ	Gradient Penalty	RBF	-	-
DNN-SN	Spec Norm	Dense	-	-
DNN-GP	-	GP	-	-
SNGP	Spec Norm	GP	-	-

Table 1: Summary of methods used in experiments. Multi-pass Inference refers to whether the method needs to perform multiple forward passes to generate the predictive distribution.

CIFAR-10 and CIFAR-100 We evaluate the model's predictive accuracy and calibration error under both clean CIFAR testing data and its corrupted versions termed CIFAR-*-C [34]. To evaluate the model's OOD detection performance, we consider two tasks: a standard OOD task using SVHN as the OOD dataset for a model trained on CIFAR-10/-100, and a difficult OOD task using CIFAR-100 as the OOD dataset for a model trained on CIFAR-10, and vice versa. We compute the uncertainty score for OOD using the Dempster-Shafer metric as introduced in [68], which empirically leads to better performance for distance-aware models (see Appendix C). Table 2-3 reports the results. As shown, for predictive accuracy, SNGP is competitive with that of a deterministic network, and outperforms the other single-model approaches. For calibration error, SNGP clearly outperforms the other single-model approaches and is competitive with the deep ensemble. Finally, for OOD detection, SNGP outperforms not only the deep ensembles and MC Dropout approaches that are based on a dense output layer, but also the MCD-GP and DUQ that are based on the GP layer, illustrating the importance of the *input distance awareness* property for high-quality performance in uncertainty quantification.

	Accuracy (↑)		ECE (↓)		NLL (↓)		OOD AUPR (↑)		Latency (↓)
Method	Clean	Corrupted	Clean	Corrupted	Clean	Corrupted	SVHN	CIFAR-100	(ms / example)
Deterministic	96.0 ± 0.01	72.9 ± 0.01	0.023 ± 0.002	0.153 ± 0.011	0.158 ± 0.01	1.059 ± 0.02	0.781 ± 0.01	0.835 ± 0.01	**3.91**
MC Dropout	96.0 ± 0.01	70.0 ± 0.02	0.021 ± 0.002	0.116 ± 0.009	0.173 ± 0.01	1.152 ± 0.01	0.971 ± 0.01	0.832 ± 0.01	27.10
Deep Ensembles	**96.6** ± **0.01**	**77.9** ± **0.01**	**0.010** ± **0.001**	**0.087** ± **0.004**	**0.114** ± **0.01**	**0.815** ± **0.01**	0.964 ± 0.01	0.888 ± 0.01	38.10
MCD-GP	95.5 ± 0.02	70.0 ± 0.01	0.024 ± 0.004	0.100 ± 0.007	0.172 ± 0.01	1.157 ± 0.01	0.960 ± 0.01	0.863 ± 0.01	29.53
DUQ	94.7 ± 0.02	71.6 ± 0.02	0.034 ± 0.002	0.183 ± 0.011	0.239 ± 0.02	1.348 ± 0.01	0.973 ± 0.01	0.854 ± 0.01	8.68
DNN-SN	96.0 ± 0.01	72.5 ± 0.01	0.025 ± 0.004	0.178 ± 0.013	0.171 ± 0.01	1.306 ± 0.01	0.974 ± 0.01	0.859 ± 0.01	5.20
DNN-GP	95.9 ± 0.01	71.7 ± 0.01	0.029 ± 0.002	0.175 ± 0.008	0.221 ± 0.02	1.380 ± 0.01	0.976 ± 0.01	0.887 ± 0.01	5.58
SNGP (Ours)	95.9 ± 0.01	74.6 ± 0.01	0.018 ± 0.001	0.090± 0.012	0.138 ± 0.01	0.935 ± 0.01	**0.990** ± **0.01**	**0.905** ± **0.01**	6.25

Table 2: Results for Wide ResNet-28-10 on CIFAR-10, averaged over 10 seeds.

Detecting Out-of-Scope Intent in Conversational Language Understanding To validate the method beyond image modalities, we also evaluate SNGP on a practical language understanding task where uncertainty quantification is of natural importance: dialog intent detection [44, 78, 82, 84]. In a goal-oriented dialog system (e.g. chatbot) built for a collection of in-domain services, it is important for the model to understand if an input natural utterance from an user is in-scope (so it can activate

Method	Accuracy (↑)		ECE (↓)		NLL (↓)		OOD AUPR (↑)		Latency (↓)
	Clean	Corrupted	Clean	Corrupted	Clean	Corrupted	SVHN	CIFAR-10	(ms / example)
Deterministic	79.8 ± 0.02	50.5 ± 0.04	0.085 ± 0.004	0.239 ± 0.020	0.872 ± 0.01	2.756 ± 0.03	0.882 ± 0.01	0.745 ± 0.01	**5.20**
MC Dropout	79.6 ± 0.02	42.6 ± 0.08	0.050 ± 0.003	0.202 ± 0.010	0.825 ± 0.01	2.881 ± 0.01	0.832 ± 0.01	0.757 ± 0.01	46.79
Deep Ensemble	**80.2** ± **0.01**	**54.1** ± **0.04**	**0.021** ± **0.004**	0.138± 0.013	**0.666** ± **0.02**	**2.281** ± **0.03**	0.888 ± 0.01	0.780 ± 0.01	42.06
MCD-GP	79.5± 0.04	45.0 ± 0.05	0.085 ± 0.005	0.159 ± 0.009	0.937 ± 0.01	2.584 ± 0.02	0.873 ± 0.01	0.754 ± 0.01	44.20
DUQ	78.5 ± 0.02	50.4 ± 0.02	0.119 ± 0.001	0.281 ± 0.012	0.980 ± 0.02	2.841 ± 0.01	0.878 ± 0.01	0.732 ± 0.01	6.51
DNN-SN	79.9 ± 0.02	48.6 ± 0.02	0.098± 0.004	0.272± 0.011	0.918 ± 0.01	3.013± 0.01	0.879± 0.03	0.745± 0.01	6.20
DNN-GP	79.2 ± 0.03	47.7 ± 0.03	0.064± 0.005	0.166± 0.003	0.885± 0.009	2.629± 0.01	0.876± 0.01	0.746± 0.02	6.82
SNGP (Ours)	79.9 ± 0.03	49.0 ± 0.02	0.025 ± 0.012	**0.117** ± **0.014**	0.847 ± 0.01	2.626 ± 0.01	**0.923** ± **0.01**	**0.801** ± **0.01**	6.94

Table 3: Results for Wide ResNet-28-10 on CIFAR-100, averaged over 10 seeds.

one of the in-domain services) or out-of-scope (where the model should abstain). To this end, we consider training an intent understanding model using the CLINC OOS intent detection benchmark dataset [44]. Briefly, the OOS dataset contains data for 150 in-domain services with 150 training sentences in each domain, and also 1500 natural out-of-domain utterances. We train the models only on in-domain data, and evaluate their predictive accuracy on the in-domain test data, their calibration and OOD detection performance on the combined in-domain and out-of-domain data. The results are in Table 4. As shown, consistent with the previous vision experiments, SNGP is competitive in predictive accuracy when compared to a deterministic baseline, and outperforms other approaches in calibration and OOD detection.

Method	Accuracy (↑)	ECE (↓)	NLL (↓)	OOD		Latency (↓)
				AUROC (↑)	AUPR (↑)	(ms / example)
Deterministic	96.5 ± 0.11	0.024 ± 0.002	3.559 ± 0.11	0.897 ± 0.01	0.757 ± 0.02	**10.42**
MC Dropout	96.1 ± 0.10	0.021 ± 0.001	1.658 ± 0.05	0.938 ± 0.01	0.799 ± 0.01	85.62
Deep Ensemble	**97.5** ± **0.03**	**0.013** ± **0.002**	**1.062** ± **0.02**	0.964 ± 0.01	0.862 ± 0.01	84.46
MCD-GP	95.9 ± 0.05	0.015 ± 0.003	1.664 ± 0.04	0.906 ± 0.02	0.803 ± 0.01	88.38
DUQ	96.0 ± 0.04	0.059 ± 0.002	4.015 ± 0.08	0.917 ± 0.01	0.806 ± 0.01	15.60
DNN-SN	95.4 ± 0.10	0.037 ± 0.004	3.565 ± 0.03	0.922 ± 0.02	0.733 ± 0.01	17.36
DNN-GP	95.9 ± 0.07	0.075 ± 0.003	3.594 ± 0.02	0.941 ± 0.01	0.831 ± 0.01	18.93
SNGP	96.6 ± 0.05	0.014 ± 0.005	1.218 ± 0.03	**0.969** ± **0.01**	**0.880** ± **0.01**	17.36

Table 4: Results for $\text{BERT}_{\texttt{Base}}$ on CLINC OOS, averaged over 10 seeds.

6 Conclusion

We propose SNGP, a simple approach to improve a single deterministic DNN's ability in predictive uncertainty estimation. It makes minimal changes to the architecture and training/prediction pipeline of a deterministic DNN, only adding spectral normalization to the hidden mapping, and replacing the dense output layer with a random feature layer that approximates a GP. We theoretically motivate *input distance awareness*, the key design principle behind SNGP, via a learning-theoretic analysis of the uncertainty estimation problem. We also propose a closed-form approximation method to make the GP posterior end-to-end trainable in linear time with the rest of the neural network. On a suite of vision and language understanding tasks and on modern architectures (ResNet and BERT), SNGP is competitive with a deep ensemble in prediction, calibration and out-of-domain detection, and outperforms other single-model approaches.

A central observation we made in this work is that *good representational learning is important for good uncertainty quantification*. In particular, we highlighted *bi-Lipschitz* (Equation (5)) as an important condition for the learned representation of a DNN to attain high-quality uncertainty performance, and proposed spectral normalization as a simple approach to ensure such property in practice. However, it is worth noting that there exists other representation learning techniques, e.g., data augmentation or unsupervised pretraining, that are known to also improve a network's uncertainty performance [35, 36]. Analyzing whether and how these approaches contribute to improve a DNN *bi-Lipschitz* condition, and whether the *bi-Lipschitz* condition is sufficient in explaining these methods' success, are interesting avenues of future work. Furthermore, we note that the spectral norm bound $\alpha < 1$ in Proposition 1 forms only a sufficient condition for ensuring bi-Lipschitz [5]. In practice, we observed that for convolutional layers, a looser norm bound is needed for state-of-the-art performance (see Section C), raising questions of whether the current regularization approach is precise enough in controlling the spectral norm of a convolutional kernel, or if there is an alternative mechanism at play in ensuring the bi-Lipschitz criterion. Finally, from a probabilistic learning perspective, SNGP focuses on learning a single high-quality model $p_\theta(y|\mathbf{x})$ for a deterministic representation. Therefore we expect it to provide complementary benefits to approaches such as (efficient) ensembles and Bayesian neural networks [21, 42, 79] which marginalize over the representation parameters as well.

Acknowledgements We would like to thank Kevin Murphy, Deepak Ramachandran, Jasper Snoek, and Timothy Nguyen at Google Research for the insightful comments and fruitful discussion.

Broader Impact

This work proposed a simple and practical methodology to improve the uncertainty estimation performance of a deterministic deep learning model. Experiment results showcased the method's ability in improving model performance in calibration and OOD detection while maintaining similar level of accuracy and latency, therefore illustrating its feasibility for industrial-scale applications. We hope the proposed approach can be used to bring concrete improvements to AI-driven, socially-relevant services where uncertainty is of natural importance. Examples include medical and policy decision making, online toxic comment management, fairness-aware recommendation systems, etc.

Nonetheless, we do not claim that the improvement illustrated in this paper solve the problem of model uncertainty entirely. This is because the analysis and experiments in this study may not capture the full complexity of the real-world use cases, and there will always be room for improvement. Designers of machine learning systems are encouraged to proactively confront the shortcomings of model uncertainty and the underlying models that generate these confidences. Even with a proper user interface, there is always room to misinterpret model outputs and probabilities, such as with nuanced applications such as election predictions, and users of these models should to be properly trained to take these factors into account.

References

[1] S. An, F. Boussaid, and M. Bennamoun. How Can Deep Rectifier Networks Achieve Linear Separability and Preserve Distances? In *International Conference on Machine Learning*, pages 514–523, June 2015. ISSN: 1938-7228 Section: Machine Learning.

[2] C. Anil, J. Lucas, and R. Grosse. Sorting Out Lipschitz Function Approximation. In *International Conference on Machine Learning*, pages 291–301, May 2019. ISSN: 1938-7228 Section: Machine Learning.

[3] P. Bartlett, S. Evans, and P. Long. Representing smooth functions as compositions of near-identity functions with implications for deep network optimization. *arXiv*, 2018.

[4] P. L. Bartlett and M. H. Wegkamp. Classification with a Reject Option using a Hinge Loss. *Journal of Machine Learning Research*, 9(Aug):1823–1840, 2008.

[5] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible Residual Networks. In *International Conference on Machine Learning*, pages 573–582, May 2019. ISSN: 1938-7228 Section: Machine Learning.

[6] A. Bendale and T. E. Boult. Towards Open Set Deep Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2 edition, 1985.

[8] A. Blum. Random Projection, Margins, Kernels, and Feature-Selection. In C. Saunders, M. Grobelnik, S. Gunn, and J. Shawe-Taylor, editors, *Subspace, Latent Structure and Feature Selection*, Lecture Notes in Computer Science, pages 52–68, Berlin, Heidelberg, 2006. Springer.

[9] J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani. Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks. *arXiv:1707.02476 [stat]*, July 2017. arXiv: 1707.02476.

[10] J. Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 135(643):1512–1519, 2009.

[11] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian Processes for regression. *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016.

[12] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, Aug. 2017. Association for Computational Linguistics.

[13] A. Chernodub and D. Nowicki. Norm-preserving Orthogonal Permutation Linear Unit Activation Functions (OPLU). *arXiv:1604.02313 [cs]*, Jan. 2017. arXiv: 1604.02313.

[14] C. Cortes, M. Mohri, and A. Rostamizadeh. Learning Non-Linear Combinations of Kernels. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 396–404. Curran Associates, Inc., 2009.

[15] J. Daunizeau. Semi-analytical approximations to statistical moments of sigmoid and softmax mappings of normal variables. Feb. 2017.

[16] G. P. Dehaene. A deterministic and computable Bernstein-von Mises theorem. *ArXiv*, 2019.

[17] J. S. Denker and Y. LeCun. Transforming Neural-Net Output Levels to Probability Distributions. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 853–859. Morgan-Kaufmann, 1991.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, Oct. 2018. arXiv: 1810.04805.

[19] L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear Independent Components Estimation. *arXiv:1410.8516 [cs]*, Oct. 2014. arXiv: 1410.8516.

[20] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv:1605.08803 [cs, stat]*, May 2016. arXiv: 1605.08803.

[21] M. Dusenberry, G. Jerfel, Y. Wen, Y. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, and D. Tran. Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors. *Proceedings of the International Conference on Machine Learning*, 1, 2020.

[22] D. Feng, L. Rosenbaum, and K. Dietmayer. Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network For Lidar 3D Vehicle Detection. Apr. 2018.

[23] D. Freedman. Wald Lecture: On the Bernstein-von Mises theorem with infinite-dimensional parameters. *The Annals of Statistics*, 27(4):1119–1141, Aug. 1999.

[24] T. Gneiting, F. Balabdaoui, and A. E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, Apr. 2007.

[25] T. Gneiting and A. E. Raftery. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477):359–378, Mar. 2007.

[26] H. Gouk, E. Frank, B. Pfahringer, and M. Cree. Regularisation of Neural Networks by Enforcing Lipschitz Continuity. Apr. 2018.

[27] P. D. GrÃŒnwald and A. P. Dawid. Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory. *Annals of Statistics*, 32(4):1367–1433, Aug. 2004. Publisher: Institute of Mathematical Statistics.

[28] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein GANs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5769–5779, Long Beach, California, USA, Dec. 2017. Curran Associates Inc.

[29] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. In *International Conference on Machine Learning*, pages 1321–1330, July 2017. ISSN: 1938-7228 Section: Machine Learning.

[30] D. Hafner, D. Tran, T. Lillicrap, A. Irpan, and J. Davidson. Reliable Uncertainty Estimates in Deep Neural Networks using Noise Contrastive Priors. July 2018.

[31] R. E. Harang and E. M. Rudd. Principled Uncertainty Estimation for Deep Neural Networks, 2018. Library Catalog: www.semanticscholar.org.

[32] M. Hauser and A. Ray. Principles of Riemannian Geometry in Neural Networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2807–2816. Curran Associates, Inc., 2017.

[33] M. Hein, M. Andriushchenko, and J. Bitterwolf. Why ReLU Networks Yield High-Confidence Predictions Far Away From the Training Data and How to Mitigate the Problem. pages 41–50, 2019.

[34] D. Hendrycks and T. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. Sept. 2018.

[35] D. Hendrycks, K. Lee, and M. Mazeika. Using Pre-Training Can Improve Model Robustness and Uncertainty. In *International Conference on Machine Learning*, pages 2712–2721, May 2019. ISSN: 1938-7228 Section: Machine Learning.

[36] D. Hendrycks*, N. Mu*, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. AugMix: A Simple Method to Improve Robustness and Uncertainty under Data Shift. In *International Conference on Learning Representations*, 2020.

[37] J.-H. Jacobsen, J. Behrmann, R. Zemel, and M. Bethge. Excessive Invariance Causes Adversarial Vulnerability. Sept. 2018.

[38] J.-H. Jacobsen, J. Behrmannn, N. Carlini, F. TramÃšr, and N. Papernot. Exploiting Excessive Invariance caused by Norm-Bounded Adversarial Robustness. Mar. 2019.

[39] r.-H. Jacobsen, A. W. M. Smeulders, and E. Oyallon. i-RevNet: Deep Invertible Networks. Feb. 2018.

[40] M. E. E. Khan, A. Immer, E. Abedi, and M. Korzepa. Approximate Inference Turns Deep Networks into Gaussian Processes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. AlchÃ©-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3094–3104. Curran Associates, Inc., 2019.

[41] A. Kristiadi, M. Hein, and P. Hennig. Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks. *arXiv:2002.10118 [cs, stat]*, Feb. 2020. arXiv: 2002.10118.

[42] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017.

[43] J. Landes. Probabilism, entropies and strictly proper scoring rules. *International Journal of Approximate Reasoning*, 63:1–21, Aug. 2015.

[44] S. Larson, A. Mahendran, J. J. Peper, C. Clarke, A. Lee, P. Hill, J. K. Kummerfeld, K. Leach, M. A. Laurenzano, L. Tang, and J. Mars. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. *arXiv:1909.02027 [cs]*, Sept. 2019. arXiv: 1909.02027.

[45] N. D. Lawrence and J. Q. Candela. Local Distance Preservation in the GP-LVM Through Back Constraints. Jan. 2006.

[46] L. LeCam. Convergence of Estimates Under Dimensionality Restrictions. *The Annals of Statistics*, 1(1):38–53, Jan. 1973.

[47] K. Lee, H. Lee, K. Lee, and J. Shin. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. In *International Conference on Learning Representations*, 2018.

[48] D. Macedo, T. I. Ren, C. Zanchettin, A. L. I. Oliveira, A. Tapp, and T. Ludermir. Isotropic Maximization Loss and Entropic Score: Fast, Accurate, Scalable, Unexposed, Turnkey, and Native Neural Networks Out-of-Distribution Detection. *arXiv:1908.05569 [cs, stat]*, Feb. 2020. arXiv: 1908.05569.

[49] D. J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, May 1992. Number: 3 Publisher: MIT Press.

[50] A. Malinin and M. Gales. Predictive Uncertainty Estimation via Prior Networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7047–7058. Curran Associates, Inc., 2018.

[51] A. Malinin and M. Gales. Prior Networks for Detection of Adversarial Attacks. *arXiv:1812.02575 [cs, stat]*, Dec. 2018. arXiv: 1812.02575.

[52] A. Meinke and M. Hein. Towards neural networks that provably know when they don't know. In *International Conference on Learning Representations*, 2020.

[53] T. P. Minka. *A family of algorithms for approximate bayesian inference*. phd, Massachusetts Institute of Technology, USA, 2001. AAI0803033.

[54] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations*, 2018.

[55] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[56] J. Nixon, M. W. Dusenberry, L. Zhang, G. Jerfel, and D. Tran. Measuring calibration in deep learning. In *CVPR Workshop*, 2019.

[57] M. Panov and V. Spokoiny. Finite Sample Bernstein von Mises Theorem for Semiparametric Problems. *Bayesian Analysis*, 10(3):665–710, Sept. 2015.

[58] M. Parry, A. P. Dawid, and S. Lauritzen. Proper local scoring rules. *Annals of Statistics*, 40(1):561–592, Feb. 2012. Publisher: Institute of Mathematical Statistics.

[59] D. C. Perrault-Joncas. Metric Learning and Manifolds: Preserving the Intrinsic Geometry. 2017.

[60] A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008.

[61] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. University Press Group Limited, Jan. 2006. Google-Books-ID: vWtwQgAACAAJ.

[62] C. Riquelme, G. Tucker, and J. Snoek. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. In *International Conference on Learning Representations*, 2018.

[63] H. Ritter, A. Botev, and D. Barber. A Scalable Laplace Approximation for Neural Networks. In *International Conference on Learning Representations*, 2018.

[64] F. Rousseau, L. Drumetz, and R. Fablet. Residual Networks as Flows of Diffeomorphisms. *Journal of Mathematical Imaging and Vision*, 62(3):365–375, Apr. 2020.

[65] W. Ruan, X. Huang, and M. Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 2651–2659, Stockholm, Sweden, July 2018. AAAI Press.

[66] W. J. Scheirer, L. P. Jain, and T. E. Boult. Probability Models for Open Set Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2317–2324, Nov. 2014. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[67] M. O. Searcod. *Metric Spaces*. Springer London, London, 2007 edition edition, Aug. 2006.

[68] M. Sensoy, L. Kaplan, and M. Kandemir. Evidential Deep Learning to Quantify Classification Uncertainty. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3179–3189. Curran Associates, Inc., 2018.

[69] L. Shu, H. Xu, and B. Liu. DOC: Deep Open Classification of Text Documents. *arXiv:1709.08716 [cs]*, Sept. 2017. arXiv: 1709.08716.

[70] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams. Scalable Bayesian Optimization Using Deep Neural Networks. *arXiv:1502.05700 [stat]*, Feb. 2015. arXiv: 1502.05700.

[71] J. Sokolic, R. Giryes, G. Sapiro, and M. R. D. Rodrigues. Robust Large Margin Deep Neural Networks. *IEEE Transactions on Signal Processing*, 2017.

[72] N. Tagasovska and D. Lopez-Paz. Single-Model Uncertainties for Deep Learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. AlchÃ©-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 6417–6428. Curran Associates, Inc., 2019.

[73] L. Tierney, R. E. Kass, and J. B. Kadane. Approximate Marginal Densities of Nonlinear Functions. *Biometrika*, 76(3):425–433, 1989. Publisher: [Oxford University Press, Biometrika Trust].

[74] G.-L. Tran, E. V. Bonilla, J. Cunningham, P. Michiardi, and M. Filippone. Calibrating Deep Convolutional Gaussian Processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1554–1563, Apr. 2019. ISSN: 1938-7228 Section: Machine Learning.

[75] Y. Tsuzuku, I. Sato, and M. Sugiyama. Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6541–6550. Curran Associates, Inc., 2018.

[76] B. van Aken, J. Risch, R. Krestel, and A. Loser. Challenges for Toxic Comment Classification: An In-Depth Error Analysis. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 33–42, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[77] J. van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal. Simple and Scalable Epistemic Uncertainty Estimation Using a Single Deep Deterministic Neural Network. *arXiv:2003.02037 [cs, stat]*, Mar. 2020. arXiv: 2003.02037.

[78] N. Vedula, N. Lipka, P. Maneriker, and S. Parthasarathy. Towards Open Intent Discovery for Conversational Text. *arXiv:1904.08524 [cs]*, Apr. 2019. arXiv: 1904.08524.

[79] Y. Wen, D. Tran, and J. Ba. BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning. In *International Conference on Learning Representations*, 2020.

[80] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. In *International Conference on Learning Representations*, 2018.

[81] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Stochastic Variational Deep Kernel Learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2594–2602, USA, 2016. Curran Associates Inc.

[82] M.-A. Yaghoub-Zadeh-Fard, B. Benatallah, F. Casati, M. Chai Barukh, and S. Zamanirad. User Utterance Acquisition for Training Task-Oriented Bots: A Review of Challenges, Techniques and Opportunities. *IEEE Internet Computing*, pages 1–1, 2020. Conference Name: IEEE Internet Computing.

[83] S. Zagoruyko and N. Komodakis. Wide Residual Networks. *arXiv:1605.07146 [cs]*, June 2017. arXiv: 1605.07146.

[84] Y. Zheng, G. Chen, and M. Huang. Out-of-Domain Detection for Natural Language Understanding in Dialog Systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1198–1209, 2020. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

Supplementary Material: Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness

Contents

A Method Summary

Architecture Given a deep learning model $\text{logit}(\mathbf{x}) = g \circ h(\mathbf{x})$ with $L-1$ hidden layers of size $\{D_l\}_{l=1}^{L}$, SNGP makes two changes to the model:

1. adding spectral normalization to the hidden weights $\{\mathbf{W}_l\}_{l=1}^{L}$, and
2. replacing the dense output layer $g(h) = h^\top \beta$ with a GP layer. Under the RFF approximation, the GP layer is simply a one-layer network with D_L hidden units $g(h) \propto \cos(-\mathbf{W}_L h_i + \mathbf{b}_L)^\top \beta$, where $\{\mathbf{W}_L, \mathbf{b}_L\}$ are frozen weights that are initialized from a Gaussian and a uniform distribution, respectively (as described in Equation (8)).

Training Algorithm 1 summarizes the training step. As shown, for every minibatch step, the model first updates the hidden-layer weights $\{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L-1}$ and the trainable output weights $\beta = \{\beta_k\}_{k=1}^{K}$ via SGD, then performs spectral normalization using power iteration method ([26] which has time complexity $O(\sum_{l=1}^{L-1} D_l)$), and finally performs precision matrix update (Equation (9), time complexity $O(D_L^2)$). Since $\{D_l\}_{l=1}^{L-1}$ are fixed for a given architecture and usually $D_L \leq 1024$, the computation scales linearly with respect to the sample size. We use $D_L = 1024$ in the experiments.

Prediction Algorithm 2 summarizes the prediction step. The model first performs the conventional forward pass, which involves (i) computing the final hidden feature $\Phi(\mathbf{x})_{D_L \times 1}$, and (ii) computing the posterior mean $\hat{m}_k(\mathbf{x}) = \Phi^\top \beta_k$ (time complexity $O(D_L)$). Then, using the posterior variance matrices $\{\hat{\Sigma}_k\}_{k=1}^{K}$, the model computes the predictive variance $\hat{\sigma}_k(\mathbf{x})^2 = \Phi(\mathbf{x})^\top \hat{\Sigma} \Phi(\mathbf{x})$ (time complexity $O(D_L^2)$).

To estimate the predictive distribution $p_k = \exp(m_k)/\sum_k \exp(m_k)$ where $m_k \sim N\big(\hat{m}_k(\mathbf{x}), \hat{\sigma}_k^2(\mathbf{x})\big)$, we calculate its posterior mean using Monte Carlo averaging. Notice that this Monte Carlo averaging is computationally very cheap since it only involves sampling from a closed-form distribution whose parameters $(\hat{m}, \hat{\sigma}^2)$ are already computed by the single feed-forward pass (i.e., a single call to `tf.random.normal`). This is different from the full Monte Carlo sampling used by MC Dropout or deep ensembles which require multiple forward passes and are computationally expensive. As shown in the latency results in experiments (Section 5.2), the extra variance-related computation only adds a small overhead to the inference time of a deterministic DNN. In the experiments, we use 10 samples to compute the mean predictive distribution.

In applications where the inference latency is of high priority (e.g., real-time pCTR prediction for online advertising), we can reduce the computational overhead further by replacing the Monte Carlo averaging with the mean-field approximation [15]. We leave this for future work.

B Formal Statements

Minimax Solution to Uncertainty Risk Minimization The expression in (4) seeks to answer the following question: *assuming we know the true domain probability $p^*(\mathbf{x} \in \mathscr{X}_{IND})$, and given a model $p(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{IND})$ that we have already learned from data, what is the best solution we can construct to minimize the minimax objective (3)?* The interest of this conclusion is not to construct a practical algorithm, but to highlight the theoretical necessity of taking into account the domain probability in constructing a good solution for uncertainty quantification. If the domain probability is not necessary, then the expression of the unique and optimal solution to the minimax probability should not contain $p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}})$ even if it is available. However the expression of (4) shows this is not the case.

To make the presentation clear, we formalize the statement about (4) into the below proposition:

Proposition 2 (Minimax Solution to Uncertainty Risk Minimization)**.**
Given:

(a) $p(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}})$ *the model's predictive distribution learned from data* $\mathscr{D} = \{y_i, \mathbf{x}_i\}_{i=1}^N$

(b) $p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}})$ *the true domain probability,*

then there exists an unique optimal solution to the minimax problem (3), and it can be constructed using (a) and (b) as:

$$p(y|\mathbf{x}) = p(y|\mathbf{x}, \mathbf{x} \in \mathscr{X}_{\texttt{IND}}) * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + p_{\texttt{uniform}}(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}}) * p^*(\mathbf{x} \notin \mathscr{X}_{\texttt{IND}}) \tag{11}$$

where $p_{\texttt{uniform}}(y|\mathbf{x}, \mathbf{x} \notin \mathscr{X}_{\texttt{IND}}) = \frac{1}{K}$ *is a discrete uniform distribution for K classes.*

As discussed in Section 2.1, the solution (11) is not only optimal for the minimax Brier risk, but is in fact optimal for a wide family of strictly proper scoring rules known as the (separable) *Bregman score* [58]:

$$s(p, p^*|\mathbf{x}) = \sum_{k=1}^{K} \Big\{ [p^*(y_k|\mathbf{x}) - p(y_k|\mathbf{x})]\, \psi'(p^*(y_k|\mathbf{x})) - \psi(p^*(y_k|\mathbf{x})) \Big\} \tag{12}$$

where ψ is a strictly concave and differentiable function. Bregman score reduces to the log score when $\psi(p) = p * \log(p)$, and reduces to the Brier score when $\psi(p) = p^2 - \frac{1}{K}$.

Therefore we will show (11) for the Bregman score. The proof relies on the following key lemma:

Lemma 1 ($p_{\texttt{uniform}}$ is Optimal for Minimax Bregman Score in $\mathbf{x} \notin \mathscr{X}_{IND}$)**.**
Consider the Bregman score in (12). At a location $\mathbf{x} \notin \mathscr{X}_{IND}$ where the model has no information about p^* other than $\sum_{k=1}^K p(y_k|\mathbf{x}) = 1$, the solution to the minimax problem

$$\inf_{p \in \mathscr{P}} \sup_{p^* \in \mathscr{P}^*} s(p, p^*|\mathbf{x})$$

is the discrete uniform distribution, i.e., $p_{uniform}(y_k|\mathbf{x}) = \frac{1}{K} \forall k \in \{1, \dots, K\}$.

The proof for Lemma 1 is in Section E.2. It is worth noting that Lemma 1 only holds for a *strictly* proper scoring rule [24]. For a non-strict proper scoring rule (e.g., the ECE), there can exist infinitely many optimal solutions, making the minimax problem ill-posed.

We are now ready to prove Proposition 2:

Proof. Denote $\mathscr{X}_{\texttt{OOD}} = \mathscr{X} / \mathscr{X}_{\texttt{IND}}$. Decompose the overall Bregman risk by domain:

$$\begin{aligned}
S(p, p^*) &= E_{x \in \mathscr{X}}\big(s(p, p^*|\mathbf{x})\big) = \int_{\mathscr{X}} s\big(p, p^*|\mathbf{x}\big) p^*(\mathbf{x}) d\mathbf{x} \\
&= \int_{\mathscr{X}} s\big(p, p^*|\mathbf{x}\big) * \big[p^*(\mathbf{x}|\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + p^*(\mathbf{x}|\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}) p^*(\mathbf{x} \in \mathscr{X}_{\texttt{OOD}})\big] d\mathbf{x} \\
&= E_{\mathbf{x} \in \mathscr{X}_{\texttt{IND}}}\big(s(p, p^*|\mathbf{x})\big) p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + E_{\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}}\big(s(p, p^*|\mathbf{x})\big) p^*(\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}) \\
&= S_{\texttt{IND}}(p, p^*) * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + S_{\texttt{OOD}}(p, p^*) * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}).
\end{aligned}$$

where we have denoted $S_{\texttt{IND}}(p, p^*) = E_{\mathbf{x} \in \mathscr{X}_{\texttt{IND}}}\big(s(p, p^*|\mathbf{x})\big)$ and $S_{\texttt{OOD}}(p, p^*) = E_{\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}}\big(s(p, p^*|\mathbf{x})\big)$.

Now consider decomposing the sup risk $\sup_{p^*} S(p,p^*)$ for a given p. Notice that sup risk $\sup_{p^*} S(p,p^*)$ is separable by domain for any $p \in \mathscr{P}$. This is because $S_{\texttt{IND}}(p,p^*)$ and $S_{\texttt{OOD}}(p,p^*)$ has disjoint support, and we do not impose assumption on p^*:

$$\sup_{p^*} S(p,p^*) = \sup_{p^*} \big[S_{\texttt{IND}}(p,p^*)\big] * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + \sup_{p^*} \big[S_{\texttt{OOD}}(p,p^*)\big] * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{OOD}})$$

We are now ready to decompose the minimax risk $\inf_p \sup_{p^*} S(p,p^*)$. Notice that the minimax risk is also separable by domain due to the disjoint in support:

$$\begin{aligned}\inf_p \sup_{p^*} S(p,p^*) &= \inf_p \Big[\sup_{p^*} \big[S_{\texttt{IND}}(p,p^*)\big] * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + \sup_{p^*} \big[S_{\texttt{OOD}}(p,p^*)\big] * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{OOD}})\Big] \\ &= \inf_p \sup_{p^*} \big[S_{\texttt{IND}}(p,p^*)\big] * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{IND}}) + \inf_p \sup_{p^*} \big[S_{\texttt{OOD}}(p,p^*)\big] * p^*(\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}), \quad (13)\end{aligned}$$

also notice that the in-domain minimax risk $\inf_p \sup_{p^*} \big[S_{\texttt{IND}}(p,p^*)\big]$ is fixed due to condition (a).

Therefore, to show that (11) is the optimal and unique solution to (13), we only need to show $p_{uniform}$ is the optimal and unique solution to $\inf_p \sup_{p^*} \big[S_{\texttt{OOD}}(p,p^*)\big]$. To this end, notice that for a given p:

$$\sup_{p^* \in \mathscr{P}^*} \big[S_{\texttt{OOD}}(p,p^*)\big] = \int_{\mathscr{X}_{\texttt{OOD}}} \sup_{p^*} [s(p,p^*|\mathbf{x})] p(\mathbf{x}|\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}) d\mathbf{x}, \quad (14)$$

due to the fact that we don't impose assumption on p^* (therefore p^* is free to attain the global supreme by maximizing $s(p,p^*|\mathbf{x})$ at every single location $\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}$). Furthermore, there exists p that minimize $\sup_{p^*} s(p,p^*|\mathbf{x})$ at every location of $\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}$, then it minimizes the integral [7]. By Lemma 1, such p exists and is unique, i.e.:

$$p_{uniform} = \underset{p \in \mathscr{P}}{\text{arginf}} \sup_{p^* \in \mathscr{P}^*} S_{\texttt{OOD}}(p,p^*).$$

In conclusion, we have shown that $p_{uniform}$ is the unique solution to $\inf_p \sup_{p^*} S_{\texttt{OOD}}(p,p^*)$. Combining with condition (a)-(b), we have shown that the unique solution to (13) is (11). □

C Experiment Details and Further Results

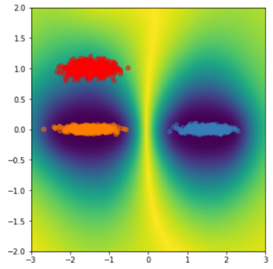
C.1 2D Synthetic Benchmark

For both benchmarks, we sample 500 observations $\mathbf{x}_i = (x_{1i}, x_{2i})$ from each of the two in-domain classes (orange and blue), and consider a deep architecture ResFFN-12-128, which contains 12 residual feedforward layers with 128 hidden units and dropout rate 0.01. The input dimension is projected from 2 dimensions to the 128 dimensions using a dense layer.

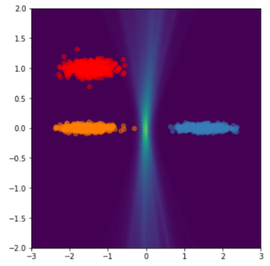
In addition to SNGP, we also visualize the uncertainty surface of the below approaches: **Gaussian process (GP)** is a standard Gaussian process directly taking $\mathbf{x}_i$ as in input. In low-dimensional datasets, GP is often considered the gold standard for uncertainty quantification. **Deep Ensemble** is an ensemble of 10 ResFFN-12-128 models with dense output layers, **MC Dropout** uses single ResFFN-12-128 model with dense output layer and 10 dropout samples. **DNN-GP** uses a single ResFFN-12-128 model with the GP Layer (described in Section 3.1) without spectral normalization. Finally, **SNGP** uses a single ResFFN-12-128 model with the GP layer and with the spectral normalization.

For these two binary classification tasks, in Figure 1 we plot the predictive uncertainty for **GP**, **DNN-GP** and **SNGP** as the posterior predictive variance of the logits, i.e., $u(\mathbf{x}) = var(logit(\mathbf{x}))$ which ranges between $[0,1]$ under the RBF kernel. For **MC Dropout** and **Deep Ensemble**, since these two methods don't provide an convenient expression of predictive variance, we plot their predictive uncertainty as the distance of the maximum predictive probability from 0.5, i.e., $u(\mathbf{x}) = 1 - 2 * |p(\mathbf{x}) - 0.5|$, so that $u(\mathbf{x}) \in [0,1]$.

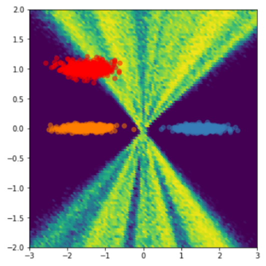
Figure 2-3 compares the aforementioned methods in terms of the same metric based on the predictive probability introduced in the last paragraph: $u(\mathbf{x}) = 1 - 2 * |p - 0.5|$. We also included **DNN-SN** (a ResFFN-12-128 model with spectral normalization but no GP layer) into comparison. As shown, compared to the uncertainty surface based on predictive variance (Figure 1), the uncertainty surface based on predictive probability shows stronger influence from the model's decision boundary. This empirical observation seems to suggest that the predictive uncertainty from the GP logits can be a better metric for calibration and OOD detection. We will explore the performance difference of different uncertainty metrics in calibration and OOD performance in the future work.



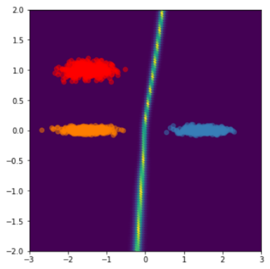
(a) Gaussian Process



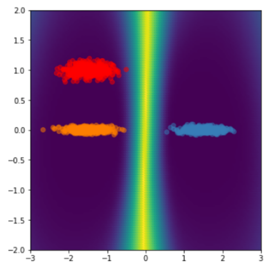
(b) Deep Ensemble



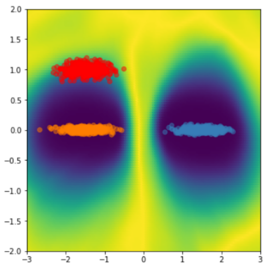
(c) MC Dropout



(d) DNN-SN

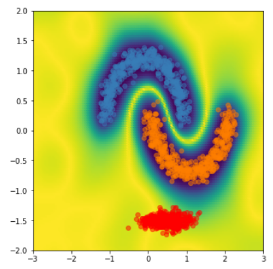


(e) DNN-GP

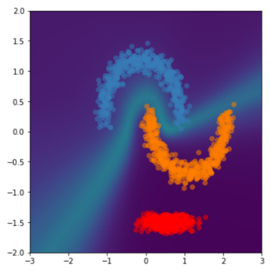


(f) SNGP (Ours)

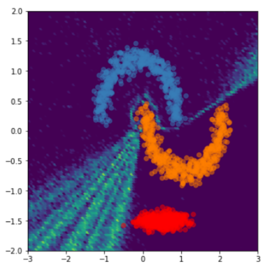
Figure 2: The uncertainty surface of a GP and different DNN approaches on the *two ovals* 2D classification benchmarks. The uncertainty is computed in terms of the distance of the maximum predictive probability from 0.5, i.e. $u(\mathbf{x}) = 1 - 2 * |p(\mathbf{x}) - 0.5|$. Background color represents the estimated model uncertainty (See 1e for color map).



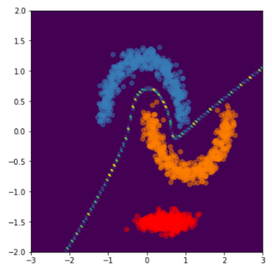
(a) Gaussian Process



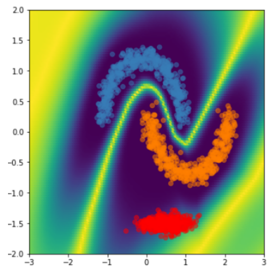
(b) Deep Ensemble



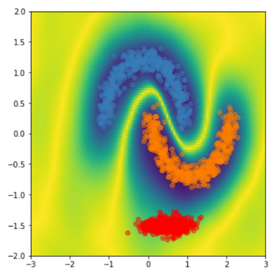
(c) MC Dropout



(d) DNN-SN



(e) DNN-GP



(f) SNGP (Ours)

Figure 3: The uncertainty surface of a GP and different DNN approaches on the *two moons* 2D classification benchmarks. The uncertainty is computed in terms of the distance of the maximum predictive probability from 0.5, i.e. $u(\mathbf{x}) = 1 - 2 * |p(\mathbf{x}) - 0.5|$. Background color represents the estimated model uncertainty (See Figures 1j and 1e for color map).

C.2 Vision and Language Understanding

Hyperparameter Configuration SNGP is composed of two components: Spectral Normalization (SN) and Gaussian Process (GP) layer, both are available at the open-source Edward2 probabilistic programming library [4].

Spectral normalization contains two hyperparameters: the number of power iterations and the upper bound for spectral norm (i.e., c in Equation (10)). In our experiments, we find it is sufficient to fix power iteration to 1. The value for the spectral norm bound c controls the trade-off between the expressiveness and the distance-awareness of the residual block, where a small value of c may shrink the residual block back to identity mapping hence harming the expressiveness, while a large value of c may lead to the loss of bi-Lipschitz property (Proposition 1). Furthermore, the proper range of c depends on the layer type: for dense layers (e.g., the intermediate and the output dense layers of a Transformer), it is sufficient to set c to a value between $(0.95, 1)$. For the convolutional layers, the norm bound needs to be set to a larger value to not impact the model's predictive performance. This is likely caused by the fact that the current spectral normalization technique does not have a precise control of the true spectral norm of the convolutional kernel, in conjuction with the fact that the other regularization mechanisms (e.g., BatchNorm and Dropout) may rescale a layer's spectral norm in unexpected ways [26, 54]. In general, we recommend performing a grid search for $c \in \{0.9, 1, 2, ...\}$ to identify the smallest possible values of c that still retains the predictive performance of the original model. In the experiments, we set the norm bound to $c = 6$ for a WideResNet model.

The GP layer contains 3 hyperparameters for the main layer (Equation (8)), and 2 hyperparameters for its covariance module (Equation (9). The three hyperparameter for the main layers are the *hidden dimension* (D_L, i.e., the number of random features), the *length-scale parameter l* for the RBF kernel, and the strength of L_2 regularization on output weights β_k. In the experiments, we find the model's performance to be not very sensitive to these parameters. Setting $D_L = 1024$ or 2048, $l = 2.0$ and L_2 regularization to 0 are sufficient in most cases. The two hyperparameters for the covariance module is the ridge factor s and the discount factor m, they come into the update rule of the precision matrix as:

$$\Sigma_{k,0}^{-1} = s * \mathbf{I}, \quad \Sigma_{k,t}^{-1} = m * \Sigma_{k,t-1}^{-1} + (1-m) * \sum_{i=1}^{M} \hat{p}_{ik}(1-\hat{p}_{ik})\Phi_i\Phi_i^\top,$$

i.e., the ridge factor s serves to control the stability of matrix inverse (if the number of sample size n is small), and m controls how fast the moving average update converges to the population value $\Sigma_k = s\mathbf{I} + \sum_{i=1}^{n} \hat{p}_{ik}(1-\hat{p}_{ik})\Phi_i\Phi_i^\top$. Similar to other moving-average update method, these two parameters can impact the quality of learned covariance matrix in non-trivial ways. In general, we recommend conducting some small scale experiments on the data to validate the learning quality of the moving average update in approximating the population covariance. Alternatively, the covariance update can be computed exactly at the final epoch by initialize $\Sigma_{k,0}^{-1} = \mathbf{0}$ and simply using the update formula $\Sigma_{k,t}^{-1} = \Sigma_{k,t-1}^{-1} + \sum_{i=1}^{M} \hat{p}_{ik}(1-\hat{p}_{ik})\Phi_i\Phi_i^\top$. In the experiments, we set $s = 0.001$ and $m = 0.999$, which is sufficient for our setting where the number of minibatch steps per epoch is large.

We also implemented two additional functionalities for GP layers: *input dimension projection* and *input layer normalization*. The input dimension project serves to project the hidden dimension of the penultimate layer D_{L-1} to a lower value D'_{L-1} (using a random Gaussian matrix $\mathbf{W}_{D_{L-1} \times D'_{L-1}}$), it can be projected down to a smaller dimension. *Input layer normalization* applies Layer Normalization to the input hidden features, which is akin to performing automatic relevance determination (ARD)-style variable selection to the input features. In the experiments, we always turn on the *input layer normalization* and set input layer normalization to 128. Although later ablation studies revealed that the model performance is not sensitive to these values.

Data Preparation and Computing Infrastructure For CIFAR-10 and CIFAR-100, we followed the original Wide ResNet work to apply the standard data augmentation (horizontal flips and random crop-ping with 4x4 padding) and used the same hyperparameter and training setup [83]. The only exception is the learning rate and training epochs, where we find a smaller learning rate (0.04 for CIFAR-10 and 0.08 for CIFAR100, v.s. 0.1 for the original WRN model) and longer epochs (250 for SNGP v.s. 200 for the original WRN model) leads to better performance.

[4]https://github.com/google/edward2

Spectral Normalization		Gaussian Process Layer	
Power Iteration	1	Hidden Dimension	1024 (WRN), 2048 (BERT)
Spectral Norm Bound	6.0 (WRN), 0.95 (BERT)	Length-scale Parameter	2.0
		L_2 Regularization	0.0
		Covariance Ridge Factor	0.001
		Covariance Discount Factor	0.999
		Projected Input Dimension	128 (WRN) None (BERT)
		Input Layer Normalization	True

Table 5: Hyperparameters of SNGP used in the experiments, where important hyperparameters are highlighted in bold.

For CLINC OOS intent understanding data, we pre-tokenized the sentences using the standard BERT tokenizer[5] with maximum sequence length 32, and created standard binary input mask for the BERT model that returns 1 for valid tokens and 0 otherwise. Following the original BERT work, we used the Adam optimizer with weight decay rate 0.01 and warmup proportion 0.1. We initialize the model from the official $\text{BERT}_{\texttt{Base}}$ checkpoint[6]. For this fine-tuning task, we using a smaller step size ($5e-5$ for SNGP .v.s. $1e-4$ for the original BERT model) but shorter epochs (40 for SNGP v.s. 150 for the original BERT model) leads to better performance. When using spectral normalization, we set the hyperparameter $c = 0.95$ and apply it to the pooler dense layer of the classification token. We do not spectral normalization to the hidden transformer layers, as we find the pre-trained BERT representation is already competent in preserving input distance due to the masked language modeling training, and further regularization may in fact harm its predictive and calibration performance.

All models are implemented in TensorFlow and are trained on 8-core Cloud TPU v2 with 8 GiB of high-bandwidth memory (HBM) for each TPU core. We use batch size 32 per core.

Evaluation For CIFAR-10 and CIFAR-100, we evaluate the model's predictive accuracy and calibration error under both clean and corrupted versions of the CIFAR testing data. The corrupted data, termed CIFAR10-C, includes 15 types of corruptions, e.g., noise, blurring, pixelation, etc, over 5 levels of corruption intensity [34]. We also evaluate the model performance in OOD detection by using the CIFAR-10/CIFAR-100 model's uncertainty estimate as a predictive score for OOD classification, where we consider a standard OOD task by testing CIFAR-10/CIFAR-100 model's ability in detecting samples from the Street View House Numbers (SVHN) dataset [55], and a more difficult OOD task by testing CIFAR-10's ability in detecting samples from the CIFAR-100 dataset, and vice versa. Specifically, for all models, we compute the OOD uncertainty score using the so-called *Dempster-Shafer metric* [68], which empirically leads to better performance for a distance-aware model. Given logits for K classes $\{h_k(\mathbf{x}_i)\}_{k=1}^K$, this metric computes its uncertainty for a test example $\mathbf{x}_i$ as:

$$u(\mathbf{x}_i) = \frac{K}{K + \sum_{k=1}^K \exp\big(h_k(\mathbf{x}_i)\big)}. \tag{15}$$

As shown, for a distance-aware model where the magnitude of the logits reflects the distance from the observed data manifold, $u(\mathbf{x}_i)$ can be a more effective metric since it is monotonic to the magnitude of the logits. On the other hand, the maximum probability $p_{\texttt{max}} = \arg\max_k \exp(h_k)/\sum_{k=1}^K \exp\big(h_k(\mathbf{x}_i)\big)$ does not take advantage of this information since it normalizes over the exponentiated logits.

In terms of evaluation metrics, we assess the model's calibration performance using the empirical estimate of ECE: $E\hat{C}E = \sum_{m=1}^M \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$ which estimates the difference in model's accuracy and confidence by partitioning model prediction into M bins $\{B_m\}_{m=1}^M$ [29]. In this work, we choose $M = 15$. We assess the model's OOD performance using Area Under Precision-Recall (AUPR). Finally, we measure each method's inference latency by millisecond per image.

For CLINC OOS intent detection data, we evaluate the predictive accuracy on the in-domain test data, evaluate the ECE and OOD detection performance on the combined in-domain and out-of-domain testing data, and we measure inference latency by millisecond per sentence.

[5]`https://github.com/google-research/bert`

[6]`https://storage.googleapis.com/bert_models/2020_02_20/uncased_L-12_H-768_A-12.zip`

D Additional Related Work

Distance-preserving neural networks and bi-Lipschitz condition The theoretic connection between distance preservation and the bi-Lipschitz condition is well-established [67], and learning an approximately isometric, distance-preserving transform has been an important goal in the fields of dimensionality reduction [8, 59], generative modeling [45, 19, 20, 39], and adversarial robustness [37, 65, 71, 75, 80]. This work is a novel application of the distance preservation property for uncertainty quantification. There existing several methods for controlling the Lipschitz constant of a DNN (e.g., gradient penalty or norm-preserving activation [1, 2, 13, 28]), and we chose spectral normalization in this work due to its simplicity and its minimal impact on a DNN's architecture and the optimization dynamics [3, 5, 64].

Open Set Classification The uncertainty risk minimization problem in Section 2 assumes a data-generation mechanism similar to the *open set recognition* problem [66], where the whole input space is partitioned into known and unknown domains. However, our analysis is unique in that it focuses on measuring a model's behavior in uncertainty quantification and takes a rigorous, decision-theoretic approach to the problem. As a result, our analysis works with a special family of risk functions (i.e., *the strictly proper scoring rule*) that measure a model's performance in uncertainty calibration. Furthermore, it handles the existence of unknown domain via a minimax formulation, and derives the solution by using a generalized version of maximum entropy theorem for the Bregman scores [27, 43]. The form of the optimal solution we derived in (4) takes an intuitive form, and has been used by many empirical work as a training objective to leverage adversarial training and generative modeling to detect OOD examples [30, 31, 47, 51, 52]. Our analysis provide strong theoretical support for these practices in verifying rigorously the uniqueness and optimality of this solution, and also provides a conceptual unification of the notion of calibration and the notion of OOD generalization. Furthermore, it is used in this work to motivate a design principle (*input distance awareness*) that enables strong OOD performance in discriminative classifiers without the need of explicit generative modeling.

E Proof

E.1 Proof of Proposition 1

The proof for Proposition 1 is an adaptation of the classic result of [3] to our current context:

Proof. First establish some notations. We denote $I(\mathbf{x}) = \mathbf{x}$ the identity function such that for $h(\mathbf{x}) = \mathbf{x} + g(\mathbf{x})$, we can write $g = h - I$. For $h : \mathscr{X} \to \mathscr{H}$, denote $||h|| = \sup\left\{\frac{||f(\mathbf{x})||_H}{||\mathbf{x}||_X} \text{ for } \mathbf{x} \in \mathscr{X}, ||\mathbf{x}|| > 0\right\}$. Also denote the Lipschitz seminorm for a function h as:

$$||h||_L = \sup\left\{\frac{||h(\mathbf{x}) - h(\mathbf{x}')||_H}{||\mathbf{x} - \mathbf{x}'||_X} \quad \text{for} \quad \mathbf{x}, \mathbf{x}' \in \mathscr{X}, \mathbf{x} \neq \mathbf{x}'\right\} \tag{16}$$

It is worth noting that by the above definitions, for two functions $(\mathbf{x}' - \mathbf{x}) : \mathscr{X} \times \mathscr{X} \to \mathscr{X}$ and $(h(\mathbf{x}) - h(\mathbf{x}')) : \mathscr{X} \times \mathscr{X} \to \mathscr{H}$ who shares the same input space, the Lipschitz inequality can be expressed using the $||.||$ norm, i.e., $||h(\mathbf{x}) - h(\mathbf{x}')||_H \leq \alpha||\mathbf{x} - \mathbf{x}'||_X$ implies $||h(\mathbf{x}') - h(\mathbf{x})|| \leq \alpha||\mathbf{x} - \mathbf{x}'||$, and vice versa.

Now assume $\forall l$, $||g_l||_L = ||h_l - I||_L \leq \alpha < 1$. We will show Proposition 1 by first showing:

$$(1-\alpha)||\mathbf{x} - \mathbf{x}'|| \leq ||h_l(\mathbf{x}) - h_l(\mathbf{x}')|| \leq (1+\alpha)||\mathbf{x} - \mathbf{x}'||, \tag{17}$$

which is the bi-Lipschitz condition for a single residual block.

First show the left hand side:

$$\begin{aligned}
||\mathbf{x} - \mathbf{x}'|| &\leq ||\mathbf{x} - \mathbf{x}' - (h_l(\mathbf{x}) - h_l(\mathbf{x}')) + (h_l(\mathbf{x}) - h_l(\mathbf{x}'))|| \\
&\leq ||(h_l(\mathbf{x}') - \mathbf{x}') - (h_l(\mathbf{x}) - \mathbf{x})|| + ||h_l(\mathbf{x}) - h_l(\mathbf{x}')|| \\
&\leq ||g_l(\mathbf{x}') - g_l(\mathbf{x})|| + ||h_l(\mathbf{x}) - h_l(\mathbf{x}')|| \\
&\leq \alpha||\mathbf{x}' - \mathbf{x}|| + ||h_l(\mathbf{x}) - h_l(\mathbf{x}')||,
\end{aligned}$$

where the last line follows by the assumption $||g_l||_L \leq \alpha$. Rearranging, we get:

$$(1-\alpha)||\mathbf{x} - \mathbf{x}'|| \leq ||h_l(\mathbf{x}) - h_l(\mathbf{x}')||. \tag{18}$$

Now show the right hand side:

$$||h_l(\mathbf{x}) - h_l(\mathbf{x}')|| = ||\mathbf{x} + g_l(\mathbf{x}) - (\mathbf{x}' + g_l(\mathbf{x}'))|| \leq ||\mathbf{x} - \mathbf{x}'|| + ||g_l(\mathbf{x}) - g_l(\mathbf{x}')|| \leq (1+\alpha)||\mathbf{x} - \mathbf{x}'||.$$

Combining (18)-(19), we have shown (17), which also implies:

$$(1-\alpha)||\mathbf{x} - \mathbf{x}'||_X \leq ||h_l(\mathbf{x}) - h_l(\mathbf{x}')||_H \leq (1+\alpha)||\mathbf{x} - \mathbf{x}'||_X \tag{19}$$

Now show the bi-Lipschitz condition for a L-layer residual network $h = h_L \circ h_{L-1} \circ \cdots \circ h_1$. It is easy to see that by induction:

$$(1-\alpha)^L||\mathbf{x} - \mathbf{x}'||_X \leq ||h(\mathbf{x}) - h(\mathbf{x}')||_H \leq (1+\alpha)^L||\mathbf{x} - \mathbf{x}'||_X \tag{20}$$

Denoting $L_1 = (1-\alpha)^L$ and $L_2 = (1+\alpha)^L$, we have arrived at expression in Proposition 1. □

E.2 Proof of Lemma 1

Proof. This proof is an application of the generalized maximum entropy theorem to the case of Bregman score. We shall first state the generalized maximum entropy theorem to make sure the proof is self-contained. Briefly, the generalized maximum entropy theorem verifies that for a general scoring function $s(p, p^*|\mathbf{x})$ with entropy function $H(p|\mathbf{x})$, the maximum-entropy distribution $p' = \underset{p}{argsup} H(p|\mathbf{x})$ attains the minimax optimality :

Theorem 1 (Maximum Entropy Theorem for General Loss [27]). *Let $\mathscr{P}$ be a convex, weakly closed and tight set of distributions. Consider a general score function $s(p, p^*|\mathbf{x})$ with an associated entropy function defined as $H(p|\mathbf{x}) = \inf_{p^* \in \mathscr{P}^*} s(p, p^*|\mathbf{x})$. Assume below conditions on $H(p|\mathbf{x})$ hold:*

- *(Well-defined) For any $p \in \mathscr{P}$, $H(p|\mathbf{x})$ exists and is finite.*

- *(Lower-semicontinous) For a weakly converging sequence $p_n \to p_0 \in \mathscr{P}$ where $H(p_n|\mathbf{x})$ is bounded below, we have $s(p, p_0|\mathbf{x}) \leq \liminf_{n\to\infty} s(p, p_n|\mathbf{x})$ for all $p \in \mathscr{P}$.*

Then there exists an maximum-entropy distribution p' such that

$$p' = \sup_{p\in\mathscr{P}} H(p) = \sup_{p\in\mathscr{P}} \inf_{p^*\in\mathscr{P}^*} s(p, p^*|\mathbf{x}) = \inf_{p\in\mathscr{P}} \sup_{p^*\in\mathscr{P}^*} s(p, p^*|\mathbf{x}).$$

Above theorem states that the maximum-entropy distribution attains the minimax optimality for a scoring function $s(p, p^*|\mathbf{x})$, assuming its entropy function satisfying certain regularity conditions. Authors of [27] showed that the entropy function of a Bregman score satisfies conditions in Theorem 1. Consequently, to show that the discrete uniform distribution is minimax optimal for Bregman score at $\mathbf{x} \notin \mathscr{X}_{\texttt{IND}}$, we only need to show discrete uniform distribution is the maximum-entropy distribution.

Recall the definition of the *strictly* proper Bregman score [58]:

$$s(p, p^*|\mathbf{x}) = \sum_{k=1}^{K} \Big\{ [p^*(y_k|\mathbf{x}) - p(y_k|\mathbf{x})]\,\psi'(p^*(y_k|\mathbf{x})) - \psi(p^*(y_k|\mathbf{x})) \Big\} \tag{21}$$

where ψ is differentiable and *strictly* concave. Moreover, its entropy function is:

$$H(p|\mathbf{x}) = -\sum_{k=1}^{K} \psi(p(y_k|\mathbf{x})) \tag{22}$$

Our interest is to show that for $\mathbf{x} \in \mathscr{X}_{\texttt{OOD}}$, the maximum-entropy distribution for the Bregman score is the discrete uniform distribution $p(y_k|\mathbf{x}) = \frac{1}{K}$. To this end, we notice that in the absence of any information, the only constraint on the predictive distribution is that $\sum_k p(y_k|\mathbf{x}) = 1$. Therefore, denoting $p(y_k|\mathbf{x}) = p_k$, we can set up the optimization problem with respect to Bregman entropy (22) using the Langrangian form below:

$$L(p|\mathbf{x}) = H(p|\mathbf{x}) + \lambda * (\sum_k p_k - 1) = -\sum_{k=1}^{K} \psi(p_k) + \lambda * (\sum_k p_k - 1) \tag{23}$$

Taking derivative with respect to p_k and λ:

$$\frac{\partial}{\partial p_k} L = -\psi'(p_k) + \lambda = 0 \tag{24}$$

$$\frac{\partial}{\partial \lambda} L = \sum_{k=1}^{K} p_k - 1 = 0 \tag{25}$$

Notice that since $\psi(p)$ is *strictly* concave, the function $\psi'(p)$ is monotonically decreasing and therefore invertible. As a result, to solve the maximum entropy problem, we can solve the above systems of equation by finding a inverse function $\psi'^{-1}(p)$, which lead to the simplification:

$$p_k = \psi'^{-1}(\lambda); \qquad \sum_{k=1}^{K} p_k = 1. \tag{26}$$

Above expression essentially states that all p_k's should be equal and sum to 1. The only distribution satisfying the above is the discrete uniform distribution, i.e., $p_k = \frac{1}{K}\ \forall k$. □