Journal of Machine Learning Research 23 (2022) 1-63 Submitted 5/22; Revised 12/22; Published 12/22

A Simple Approach to Improve Single-Model Deep Uncertainty via Distance-Awareness

Jeremiah Zhe Liu* JERELIU@GOOGLE.COM
Shreyas Padhy*† SP2058@ENG.CAM.AC.UK
Jie Ren* JJREN@GOOGLE.COM
Zi Lin† ZIL061@UCSD.EDU
Yeming Wen† YWEN@UTEXAS.EDU
Ghassen Jerfel† GHASSEN@WAYMO.COM
Zachary Nado ZNADO@GOOGLE.COM
Jasper Snoek JSNOEK@GOOGLE.COM
Dustin Tran TRANDUSTIN@GOOGLE.COM
Balaji Lakshminarayanan BALAJILN@GOOGLE.COM
Google Research, Mountain View, CA 94043, USA

Editor: Philipp Hennig

Abstract

Accurate uncertainty quantification is a major challenge in deep learning, as neural networks can make overconfident errors and assign high confidence predictions to out-of-distribution (OOD) inputs. The most popular approaches to estimate predictive uncertainty in deep learning are methods that combine predictions from multiple neural networks, such as Bayesian neural networks (BNNs) and deep ensembles. However their practicality in real-time, industrial-scale applications are limited due to the high memory and computational cost. Furthermore, ensembles and BNNs do not necessarily fix all the issues with the underlying member networks. In this work, we study principled approaches to improve uncertainty property of a single network, based on a single, deterministic representation. By formalizing the uncertainty quantification as a minimax learning problem, we first identify *distance awareness*, i.e., the model's ability to quantify the distance of a testing example from the training data, 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 with two simple changes: (1) applying spectral normalization to hidden weights to enforce bi-Lipschitz smoothness in representations and (2) replacing the last output layer with a Gaussian process layer. On a suite of vision and language understanding benchmarks and on modern architectures (Wide-ResNet and BERT), SNGP consistently outperforms other single-model approaches in prediction, calibration and out-of-domain detection. Furthermore, SNGP provides complementary benefits to popular techniques such as deep ensembles and data augmentation, making it a simple and scalable building block for probabilistic deep learning. Code is open-sourced at https://github.com/google/uncertainty-baselines.

Keywords: Single model uncertainty, Deterministic uncertainty quantification, Probabilistic neural networks, Calibration, Out-of-distribution detection

*Equal contribution.
†Work done at Google.

1. Introduction

In recent years, deep neural network (DNN) models have become ubiquitous in large-scale, real-world applications. The examples range from self-driving, image recognition, natural language understanding, to scientific applications including genomic sequence identification, genetic variant calling, drug discovery and protein design (Larson et al., 2019; Gupta et al., 2021; Jumper et al., 2021; Poplin et al., 2018; Ren et al., 2019; Han et al., 2021; Kivlichan et al., 2021; Roy et al., 2022). A key characteristic shared by these real-world tasks is their *risk sensitivity*: a confidently wrong decision from the deep learning model can lead to ethical violations, misleading scientific conclusions, and even fatal accidents (Amodei et al., 2016). Therefore, to ensure safe and responsible deployment of AI technologies to the real world, it is of utmost importance to develop efficient approaches that reliably improves a deep neural network's uncertainty quality without compromising its practical utility (i.e., in terms of accuracy and scalability).

It is well-known that a naively trained modern deep network tends to perform poorly in uncertainty tasks. They can be poorly calibrated (Guo et al., 2017) or assign high confidence predictions to out-of-domain (OOD) inputs (Nguyen et al., 2015; Hendrycks and Gimpel, 2017; Lakshminarayanan et al., 2017). This has led to the development of probabilistic methods tailored for deep neural networks, with examples including Bayesian neural networks (BNNs) (Blundell et al., 2010; Osawa et al., 2019; Wenzel et al., 2020), Monte Carlo (MC) Dropout (Gal and Ghahramani, 2016), and Deep Ensembles (Lakshminarayanan et al., 2017). A shared characteristic of these approaches is that they are *ensemble-based* methods: they need to maintain distribution samples over millions of model parameters, or require multiple forward passes to produce a final prediction. Consequently, despite their success in academic studies, these approaches can be computationally prohibitive for real-world usage (Ovadia et al., 2019; Wilson and Izmailov, 2020). Furthermore, ensembles and BNNs may not necessarily fix all the problems with the underlying neural network in the first place. For instance, if all the ensemble members consistently make same mistakes or produce high confidence predictions far away from the data, the ensemble would inherit this behavior and also produce high confidence predictions far away from the data.

In this work, we seek to address this gap by exploring an alternative direction to improve the uncertainty quality of a *single* DNN, so that the model achieves high-quality uncertainty with only a *single, deterministic representation*. That is, instead of improving uncertainty by ensembling over multiple representations, we focus on addressing the design choices in a single DNN that hinders its uncertainty performance. Specifically, we study principled inductive biases (i.e., mathematical conditions) that a model's hidden representation and output layer should satisfy to obtain good uncertainty performance. Then, we propose a simple, general-purpose algorithm to implement such inductive bias into a deep neural network. We expect this direction to be fruitful: an effective single-model uncertainty approach unlocks high-quality uncertainty estimation in resource-constrained, real-world settings where Monte Carlo sampling is not feasible (e.g., on-device, real-time prediction), and also can be combined with existing ensemble techniques to produce new probabilistic models that improves the state-of-the-art.

Specifically, we first propose *distance awareness*, i.e., a model's output prediction is aware of the distance between a new test example and the previously trained-upon examples, as an important property for the neural network to achieve high-quality uncertainty. Formally, this means that a predictive model $f(\mathbf{x})$ has the ability to output a scalar uncertainty estimate $u(\mathbf{x})$ that is monotonic with respect to an appropriate distance metric between $\mathbf{x}$ and the training examples (Definition

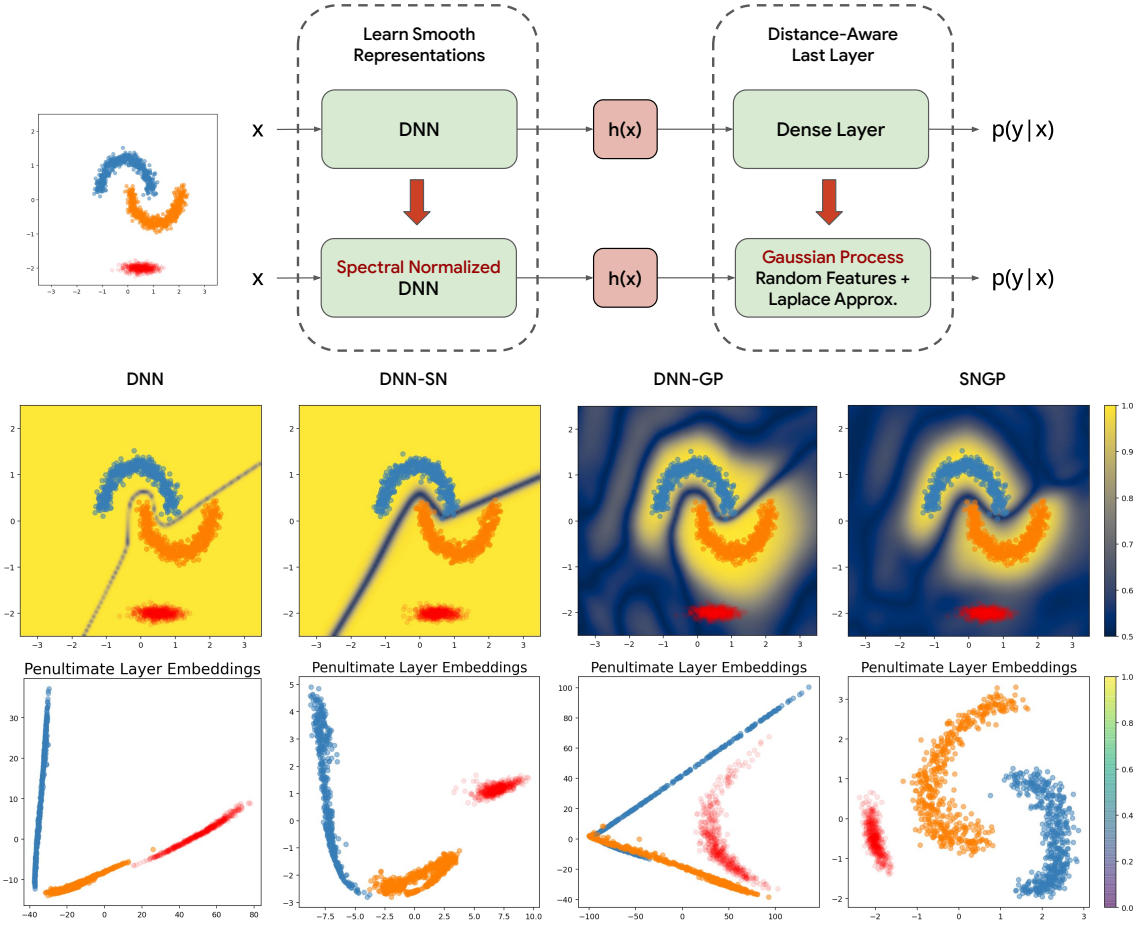


Figure 1: SNGP improves the standard Deep Learning framework by (1) learning smooth representations from input data using Spectral normalization, and (2) enforcing distance-awareness in the last layer of the neural network. In the *two moons* example, a regular deep neural network **(DNN)** learns hidden embeddings that are highly degenerate and map both in-distribution and OOD points close together. In **DNN-SN**, the application of Spectral Norm improves the quality of hidden representations, however a dense last layer still results in high confidences away from the training manifold. In **DNN-GP**, a Gaussian Process last layer learns distance-aware confidences, however due to degenerate hidden representations, there is still high confidence away from the training manifold, especially near the OOD class. Finally, Spectral Norm is combined with a last-layer Gaussian Process in **SNGP**, where the model learns smooth representations of the training manifold, and shows low confidence away from the training distribution. In the top row, we plot the model confidence surface given by $\max p(\mathbf{x})$ for all methods, and in the bottom row we plot the 2-dimensional TSNE projection of the penultimate layer embeddings for a 6-layer ResNet model with 128 hidden units per layer.

2). In the literature, several prior investigations have suggested the connection between a neural model's uncertainty quality and its ability to distinguish distances in the input space. For example, Kristiadi et al. (2020) shows that deep classifiers degrades in performance and becomes increasing overconfident when making predictions on examples that further from the support of the training set. On the other hand, van Amersfoort et al. (2021) and Behrmann et al. (2021) observe that deep models often collapse the representations of distinct examples onto the same location in the hidden

space (i.e., "feature collapse"), preventing the model from distinguishing between the familiar and unfamiliar examples and causing difficulty in detecting OOD inputs. In this work, we formalize the intuition behind these empirical findings by providing a precise definition of the *distance awareness*. We further provide rigorous arguments for the importance of the distance awareness property for a model's uncertainty performance, by casting uncertainty estimation as a minimax decision making problem and showing that distance awareness constitutes a necessary condition for obtaining the optimal solution (Section 2).

We then move to identify simple, general-purpose algorithm to implement the distance-awareness principle into a DNN model. Figure 1 gives an high level overview. We take inspiration from the classic probablistic learning literature, where a shallow Gaussian process (GP) model (when equipped with certain kernel) is known to perfectly achieve distance awareness (Rasmussen and Williams, 2006). However, the shallow GP model tends to generalize poorly for high-dimensional data, since the common choices of kernel function lack the ability to adaptively capture the intrinsic structure underlying the data's high surface dimension, leading to the issue of curse of dimensionality (Bach, 2017). Later works mitigate this issue by equipping the GP with a dimension-reduction feature extractor (e.g., implemented by a DNN) (Salakhutdinov and Mnih, 2008; Damianou and Lawrence, 2013; Wilson et al., 2016a,b; Calandra et al., 2016; Salimbeni and Deisenroth, 2017; Bradshaw et al., 2017). However, naively combining a DNN with a GP layer does not automatically guarantee distance awareness even with end-to-end training, as the hidden representation can still suffer the issue of feature collapse (as we will show in experiments, c.f. Figures 1 and 4).

To this end, we propose a simple and efficient algorithm for probabilistic deep learning which we term *Spectral-normalized Neural Gaussian Process* (SNGP). As shown in Figure 1, given an existing DNN architecture, SNGP makes two simple modifications to the neural network, namely

1. adding *spectral normalization* to the model's hidden layers, and

2. replacing the typical dense output layer with a distance-aware Gaussian Process.

We show that spectral normalization improves the distance-preservation property of learned representations by bounding the hidden-space distance $||h(\mathbf{x}) - h(\mathbf{x}')||_H$ with respect to $d_X(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are two inputs to the feature extractor of the DNNs $h(\mathbf{x})$, and d_X is a suitable distance metric defined for the data manifold (Proposition 3). We then build a GP output layer on these representations $h(\mathbf{x})$. To ensure computational scalability, we use a Laplace approximation to the random feature expansion of the GP. This results in a model posterior that can be learned scalably and in closed form, and lets us efficiently compute the predictive uncertainty on individual inputs without having to resort to computationally expensive methods such as multiple forward passes (Section 3).

We conduct a comprehensive study to investigate the behavior of SNGP model across data modalities. As we will show, the SNGP approach improves the calibration and OOD detection performance of a deterministic DNN, and outperforms or is competitive with other single model uncertainty approaches. We further illustrate the method's scalability by adapting it to large-scale recognition tasks (i.e., ImageNet), and illustrate the method's generality by extending it to additional tasks in natural language understanding and genomics sequence identification (Section 6). Finally, we show that SNGP can serve as a strong building block for probabilistic deep learning approaches and provides complementary benefits to other state-of-the-art techniques such as ensembling (e.g., MC Dropout, Deep Ensemble) and data augmentation (e.g., AugMix), providing orthogonal improvements in uncertainty quantification (Section 7).

2. Theoretical Motivation for Distance Awareness

Notation and Problem Setup Let us consider a data-generating distribution $p^*(y,\mathbf{x}) = p^*(y|\mathbf{x})p^*(\mathbf{x})$, where $y \in \{1,\dots,K\}$ is the K-dimensional simplex of labels, and $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ is the input data present on a manifold with a suitable metric $d_X : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In particular, such a metric is tailored toward the geometry of $\mathcal{X}$ such that, intuitively, the distance $d_X(\mathbf{x}_1,\mathbf{x}_2)$ between a pair of examples $\mathbf{x}_1,\mathbf{x}_2 \in \mathcal{X}$ reflects a meaningful difference in the input space (e.g., for a pair of sentences $\mathbf{x}_1,\mathbf{x}_2$, $d_X(\mathbf{x}_1,\mathbf{x}_2)$ describes their semantic similarity rather than the token-level edit distance (Cer et al., 2017)). In a supervised learning setup, the goal is often to learn the conditional distribution $p^*(y|\mathbf{x})$, and the training data $\mathcal{D} = \{y_i,\mathbf{x}_i\}_{i=1}^N$ is often a subset of the full input space $\mathcal{X}_{\texttt{IND}} \subset \mathcal{X}$. Due to this fact, we can represent the conditional data-generating distribution $p^*(y|\mathbf{x})$ as a mixture of an in-domain (IND) distribution $p_{\texttt{IND}}(y|\mathbf{x}) = p^*(y|\mathbf{x},\mathbf{x} \in \mathcal{X}_{\texttt{IND}})$ and an OOD distribution with non-overlapping support $p_{\texttt{OOD}}(y|\mathbf{x}) = p^*(y|\mathbf{x},\mathbf{x} \notin \mathcal{X}_{\texttt{IND}})$ (Meinke and Hein, 2020; Scheirer et al., 2014):

$$p^*(y|\mathbf{x}) = p^*(y|\mathbf{x},\mathbf{x} \in \mathcal{X}_{\texttt{IND}}) \times p^*(\mathbf{x} \in \mathcal{X}_{\texttt{IND}}) + p^*(y|\mathbf{x},\mathbf{x} \notin \mathcal{X}_{\texttt{IND}}) \times p^*(\mathbf{x} \notin \mathcal{X}_{\texttt{IND}}). \tag{1}$$

During the process of training, the model learns the in-domain predictive distribution $p^*(y|\mathbf{x},\mathbf{x} \in \mathcal{X}_{\texttt{IND}})$ from the training data $\mathcal{D}$, but does not have knowledge about $p^*(y|\mathbf{x},\mathbf{x} \notin \mathcal{X}_{\texttt{IND}})$.

In the example of an image classification model trained on MNIST, the out-of-domain space $\mathcal{X}_{\texttt{OOD}} = \mathcal{X}/\mathcal{X}_{\texttt{IND}}$ is the space of all images that do not contain handwritten characters with the classes 0 to 9, which can include other datasets such as CIFAR-10 or ImageNet, with the potential for some overlap whenever numbers are present in such image examples.

In the example of a weather-service chatbot, the out-of-domain space $\mathcal{X}_{\texttt{OOD}} = \mathcal{X}/\mathcal{X}_{\texttt{IND}}$ 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 $y \in \{1,\dots,K\}$. The out-of-domain distribution $p^*(y|\mathbf{x},\mathbf{x} \notin \mathcal{X}_{\texttt{IND}})$ can in general be very different from the in-domain distribution $p^*(y|\mathbf{x},\mathbf{x} \in \mathcal{X}_{\texttt{IND}})$, and it is usually expected that the model will only generalize well within $\mathcal{X}_{\texttt{IND}}$. However, during testing and deployment, the model is expected to construct a predictive distribution $p(y|\mathbf{x})$ for the entire input space, $\mathcal{X} = \mathcal{X}_{\texttt{IND}} \cup \mathcal{X}_{\texttt{OOD}}$, since the gamut of input data that the model can be deployed on can come from anywhere, and not just a curated training distribution.

2.1 Uncertainty Estimation as a Minimax Learning Problem

In order to formulate 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. One popular metric, the Expected Calibration Error (ECE), is defined as $C(p,p^*) = E\left[|E(y^* = \hat{y}|\hat{p} = p) - p|\right]$, and measures the difference in expectation between the model's predictive confidence (e.g., the maximum probability score) and its actual accuracy (Guo et al., 2017; Nixon et al., 2019). However, ECE is not suitable as a loss function, since it does not have a unique minimum at the true solution $p = p^*$. Using the ECE directly can result in trivial counterexamples where a predictor ignores the input example and achieve perfect calibration by predicting randomly according to the marginal distribution of the labels (Gneiting et al., 2007).

To this end, a more theoretically sound uncertainty metric needs to be obtained, which we accomplish by examining the rich literature of *strictly proper scoring rules* (Gneiting and Raftery, 2007) $s(.,p^*)$, which are loss functions that are uniquely minimized by the true distribution $p = p^*$. These family of loss functions include a lot of the more commonly used examples such as log-loss and Brier score. Another nice property of proper scoring rules is that they're related to ECE; both the

log-loss and the Brier score are upper bounds of the calibration error, and this can be shown by the classic calibration-refinement decomposition (Bröcker, 2009). Therefore, it follows that if a proper scoring rule is minimized, it implies that the calibration error of the model is also being minimized. We can now formalize the problem of uncertainty quantification as the problem of constructing an optimal predictive distribution $p(y|\mathbf{x})$ that minimizes the expected risk over all $\mathbf{x} \in \mathcal{X}$, i.e., an *Uncertainty Risk Minimization* problem[1]:

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

Unfortunately, we cannot minimize (2) over the entire input space $\mathcal{X}$, even with access to infinite amounts of in-domain data. This is due to the fact that during the training process, the data is collected only from $\mathcal{X}_{\texttt{IND}}$, and the true OOD distribution $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathcal{X}_{\texttt{IND}})$ can never be learned by the model, and therefore generalization is not guaranteed since we do not make the assumption that $p^*(y|\mathbf{x}, \mathbf{x} \in \mathcal{X}_{\texttt{IND}})$ and $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathcal{X}_{\texttt{IND}})$ are similar. In practice, using a model trained only with in-domain data to make predictions on OOD can lead to arbitrarily bad results, since nature can contain many OOD distributions $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathcal{X}_{\texttt{IND}})$ that are very dissimilar with the training data. This is clearly undesirable for safety-critical applications.

We therefore reformulate the problem using a more prudent strategy; minimize instead the *worst-case* risk with respect to all possible $p^* \in \mathcal{P}^*$, where $\mathcal{P}^*$ is the space of possible data-generating distributions whose in-domain component $p^*(y|\mathbf{x}, \mathbf{x} \in \mathcal{X}_{\texttt{IND}})$ generates the observational data, while whose out-of-domain component $p^*(y|\mathbf{x}, \mathbf{x} \notin \mathcal{X}_{\texttt{IND}})$ is unconstrained and can be arbitrary. That is, we seek to construct a $p(y|\mathbf{x})$ to minimize the *Minimax Uncertainty Risk*:

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

where $S(p, p^*)$ is the expected risk as defined in (2). This reformulation can be viewed from a game-theoretic lens; the uncertainty estimation task is acting as a two-player game with the model and 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 task of classification using the Brier score as a proper scoring rule, the solution to the minimax problem (3) adopts a simple and elegant form:

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

(4) has a very intuitive understanding; we should trust the model for an input point that lies in the training data domain, and otherwise make a maximum entropy (uniform) prediction.[2] For the practice

[1] It is interesting to note that, as a special case, (2) reduces back to the familiar maximum-likelihood (MLE) objective when s is the logarithm score. See Gneiting and Raftery (2007) for further detail. However, the analysis here goes beyond the scope of empirical risk parameter estimation, as it focuses on risk functional in terms of a infinite-dimensional parameter p and concerns the out-of-domain situations where the training data is not available.

[2] Noted that this uniform strategy is conservative and is derived under the minimax assumption that (1) the OOD distribution is adversarial, and (2) all the classes in y are semantically distinct. That is, in the OOD regions, $p^*(y|\mathbf{x})$ is always as far away from $p(y|\mathbf{x})$ as possible and puts its probability mass on an class that is the most different from the model prediction. For example, for a cancer prognosis model, the p^* puts the probability on a different cancer type that requires a completely different treatment. In practice, there exist situations where the OOD class is semantically related to an in-domain class (e.g., pickup truck v.s. truck for the CIFAR100 v.s. CIFAR10 OOD detection problem) such that the data is not completely adversarial. In this case, a non-uniform distribution is still sensible in the practical context. However that falls outside the scope of the minimax analysis that we consider here.

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 as a mixture of a discrete uniform distribution $p_{\texttt{uniform}}$ and the in-domain predictive distribution $p(y|\mathbf{x}, \mathbf{x} \in \mathcal{X}_{\texttt{IND}})$ that the model has already learned from data, *assuming one can quantify* $p^*(\mathbf{x} \in \mathcal{X}_{\texttt{IND}})$ *well*. In fact, the expression (4) can be shown to be optimal for a broad family of scoring rules known as the Bregman scores, which includes the Brier score and the widely used *log score* as the special cases.

Proof Sketch: The proof for (4) relies on the following key lemma (proved in Appendix E.2):

Lemma 1 ($p_{\texttt{uniform}}$ is Optimal for Minimax Bregman Score in $\mathbf{x} \notin \mathcal{X}_{IND}$)**.**
Consider the Bregman score in (24). At a location $\mathbf{x} \notin \mathcal{X}_{IND}$ *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*

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

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

Using Lemma 1, (4) can be proved easily by decomposing p^* into its in-domain and out-of-domain components and apply standard minimax arguments. A proof sketch is included below:
Proof Sketch. Denote $\mathcal{X}_{\texttt{OOD}} = \mathcal{X} / \mathcal{X}_{\texttt{IND}}$. Decompose the overall Bregman risk by domain:

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

where we have denoted $S_{\texttt{IND}}(p, p^*) = E_{\mathbf{x} \in \mathcal{X}_{\texttt{IND}}}\big(s(p, p^*|\mathbf{x})\big)$ and $S_{\texttt{OOD}}(p, p^*) = E_{\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}}\big(s(p, p^*|\mathbf{x})\big)$. Noting that $S_{\texttt{IND}}(p, p^*)$ and $S_{\texttt{OOD}}(p, p^*)$ have disjoint support, we can decompose the minimax risk as follows:

$$\inf_{p} \sup_{p^*} S(p, p^*) = \inf_{p} \sup_{p^*} \big[S_{\texttt{IND}}(p, p^*)\big] * p^*(\mathbf{x} \in \mathcal{X}_{\texttt{IND}}) + \inf_{p} \sup_{p^*} \big[S_{\texttt{OOD}}(p, p^*)\big] * p^*(\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}), \quad (5)$$

Since the model's predictive distribution is learned from data, the in-domain minimax risk is fixed. Therefore, we only need to show $p_{\texttt{uniform}}$ is the optimal and unique solution to the out-of-domain minimax risk $\inf_p \sup_{p^*} \big[S_{\texttt{OOD}}(p, p^*)\big]$. To this end, notice that for a given p:

$$\sup_{p^* \in \mathcal{P}^*} \big[S_{\texttt{OOD}}(p, p^*)\big] = \int_{\mathcal{X}_{\texttt{OOD}}} \sup_{p^*} [s(p, p^*|\mathbf{x})] p(\mathbf{x}|\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}) d\mathbf{x}, \quad (6)$$

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 $\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}$). Furthermore, there exists p that minimize $\sup_{p^*} s(p, p^*|\mathbf{x})$ at every location of $\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}$, then it minimizes the integral (Berger, 1985). By Lemma 1, such p exists and is unique, i.e.:

$$p_{\texttt{uniform}} = \operatorname*{arginf}_{p \in \mathcal{P}} \sup_{p^* \in \mathcal{P}^*} S_{\texttt{OOD}}(p, p^*).$$

In conclusion, we have shown that $p_{\texttt{uniform}}$ is the unique solution to $\inf_p \sup_{p^*} S_{\texttt{OOD}}(p, p^*)$, and therefore that the unique solution to (5) is (4). ■

A complete version of the proof is available in Appendix B.

2.2 Distance Awareness as a Necessary Condition

Following from Equation (4), it stands to reason that a key recipe for a deep learning model to be able to reliably estimate predictive uncertainty is its ability to quantify (explicitly or implicitly) the domain probability $p(\mathbf{x} \in \mathcal{X}_{\texttt{IND}})$. This requires that the model have a good notion of the distance (or dissimilarity) between a testing example $\mathbf{x}$ and the training data $\mathcal{X}_{\texttt{IND}}$ with respect to a *suitable* metric d_X for the data manifold (e.g., *semantic textual similarity* (Cer et al., 2017) for language data). Definition 2 makes this notion more precise:

Definition 2 (Distance Awareness)**.** *Consider a predictive distribution $p(y|\mathbf{x})$ trained on a domain $\mathcal{X}_{\texttt{IND}} \subset \mathcal{X}$, where $(\mathcal{X}, d_X)$ is the input data manifold equipped with a suitable metric d_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) and reflects the distance between $\mathbf{x}$ and the training data with respect to d_X, i.e.,*

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

Here, v is a monotonic function and $d(\mathbf{x}, \mathcal{X}_{\texttt{IND}}) = E_{\mathbf{x}' \sim \mathcal{X}_{\texttt{IND}}} d_X(\mathbf{x}, \mathbf{x}')$ is the distance between $\mathbf{x}$ and the training data domain.

In the literature, there have been many models that attempt to enforce *distance-awareness*, and here we consider a classic model that satisfies this property: a Gaussian process (GP) with a radial basis function (RBF) kernel. For a classification task, the predicted class probability $r(g(\mathbf{x}))$ is a nonlinear transformation of the GP posterior $g \sim GP$, i.e., $r(.) = \text{sigmoid}(.)$ for binary classification. The predictive uncertainty can then be expressed by the posterior variance $u(\mathbf{z}_{test}) = \text{var}(g(\mathbf{z}_{test})) = 1 - \mathbf{k}_{test}^{\top} \mathbf{V} \mathbf{k}_{test}$ for $\mathbf{k}_{test,i} = \exp(-\frac{1}{2}||\mathbf{z}_{test} - \mathbf{z}_i||^2)$ 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 $\mathcal{X}_{\texttt{IND}}$ (Rasmussen and Williams (2006), Chapter 3.4). It follows from the expression (4) that the *distance awareness* property is important for both calibration and OOD detection. However, there is no guarantee of this property for a typical deep learning model (Hein et al., 2019b). For example, consider a discriminative deep classifier with a dense output layer $\text{logit}(\mathbf{x}) = h(\mathbf{x})^{\top} \beta$, where the 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 1). As a result of this formulation, the model computes confidence for a point $\mathbf{x}^*$ based not on its distance from the training data $\mathcal{X}_{\texttt{IND}}$, but based on its distance from the decision boundaries, i.e., the model uncertainty is not *distance aware*. Section 8.1 provides further discussion.

Two Conditions for Distance Awareness in Deep Learning. The output of deep learning classifiers can be written as $\text{logit}(\mathbf{x}) = g \circ h(\mathbf{x})$, which is composed of a hidden mapping $h : \mathcal{X} \to \mathcal{H}$ that map the input $\mathbf{x}$ into a hidden representation space $h(\mathbf{x}) \in \mathcal{H}$, and an output layer g that maps $h(\mathbf{x})$ to the label space. As shown in the previous section, this formulation is not traditionally *input distance aware*, but it can be made to be so by imposing 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 $d_X(\mathbf{x}, \mathbf{x}')$ in the data manifold. From a mathematical point of view, this is equivalent to requiring h to

satisfy the *bi-Lipschitz* condition (Searcod, 2006):

$$L_1 \times d_X(\mathbf{x}_1, \mathbf{x}_2) \leq ||h(\mathbf{x}_1) - h(\mathbf{x}_2)||_H \leq L_2 \times d_X(\mathbf{x}_1, \mathbf{x}_2), \tag{7}$$

for positive and bounded constants $0 < L_1 < L_2$. For a deep learning model, the bi-Lipschitz condition (7) usually leads the model's hidden space to preserve a meaningful distance in the input data manifold $\mathcal{X}$ that is, e.g., effective for determining if an observation is in-distribution. This is due to the fact that the upper Lipschitz bound $||h(\mathbf{x}_1) - h(\mathbf{x}_2)||_H \leq L_2 \times d_X(\mathbf{x}_1, \mathbf{x}_2)$ 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 meaningless perturbations in the pixel space (e.g., Gaussian noise) (Ruan et al., 2018; Weng et al., 2018; Tsuzuku et al., 2018; Jacobsen et al., 2019a; Sokolic et al., 2017). On the other hand, the lower Lipschitz bound $||h(\mathbf{x}_1) - h(\mathbf{x}_2)||_H \geq L_1 \times d_X(\mathbf{x}_1, \mathbf{x}_2)$ prevents the hidden representation from collapsing the representations of distinct examples together, which otherwise leads to undesired invariance to the meaningful differences between examples (i.e., the concept of *feature collapse*) (Jacobsen et al., 2019b; Van Amersfoort et al., 2020). When we combine these two inequalities and the properties they enforce, the bi-Lipschitz condition essentially encourages h to be an *approximately* isometric mapping, thereby ensuring that the learned representation space H has a robust and meaningful correspondence with the geometry of the input data manifold $\mathcal{X}$. Heuristically as well, machine learning methods usually tend to attempt to learn an approximately isometric and geometry-preserving mapping (Hauser and Ray, 2017b; Perrault-Joncas and Meila, 2012; Rousseau et al., 2020). For example, deep image classifiers usually strive to learn a mapping from the image manifold to a hidden representation space where the input data is easily separable using a set of linear decision boundaries. Similarly, sentence encoders aim to project sentences into a vector space where the cosine distance can be used to measure the *semantic textual similarity* between natural language sentences (Cer et al., 2018). It has also been shown that preserving such *approximate* isometry in a neural network is possible even after significant dimensionality reduction (Blum, 2006).

3. Our Proposed Method: Spectral-normalized Neural Gaussian Process (SNGP)

In this section we formalize the definition of the *Spectral-normalized Neural Gaussian Process (SNGP)* algorithm for modern residual-based DNN (e.g., ResNet, Transformer). The SNGP algorithm provides a simple approach to encode distance awareness into the output layer, and distance preservation into the hidden layers (i.e., the two properties introduced in Section 2.2). We summarize the method in Algorithms 1-2.

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

SNGP achieves distance-awareness in the output layer $g : \mathcal{H} \to \mathcal{Y}$ by replacing a dense output layer with an approximate Gaussian process (GP) conditioned on the learned hidden representations of the DNN, where the posterior variance of a test input $\mathbf{x}^*$ is proportional to its L_2 distance from the training datapoints in the hidden space[3].

Given a training dataset $\mathcal{D} = \{y_i, \mathbf{x}_i\}_{i=1}^N$ with N data points, we denote the hidden representation of the DNN defined for each training point as $h_i = h(\mathbf{x}_i)$, and denote the Gaussian Process conditioned

[3] In this work, we focus on the RBF kernel for its simplicity, however it is easy to extend our framework to other kernels (e.g., Matérn, MLP, or arccos kernel, etc) by modifying the activation function and the distribution of the random-feature mapping in Equation (10) (Choromanski et al., 2018; Liu et al., 2021).

on the hidden representation as $g_{N\times 1} = [g(h_1), \ldots, g(h_N)]^\top$. Then, the prior distribution of a GP model equipped with an RBF kernel is a multivariate normal:

$$g_{N\times 1} \sim N(\mathbf{0}_{N\times 1}, \sigma^2 * \mathbf{K}_{N\times N}), \quad \text{where } \mathbf{K}_{i,j} = \exp(-||h_i - h_j||_2^2/2), \tag{8}$$

where σ^2 is the kernel amplitude (Rasmussen and Williams, 2006). The model's posterior distribution is then calculated using Bayes Rule:

$$p(g|\mathcal{D}) \propto p(\mathcal{D}|g) \times p(g), \tag{9}$$

where $p(g)$ is the GP prior as given by (8). $p(\mathcal{D}|g)$ is the data likelihood for the task, e.g., for a regression task, it can be the exponentiated squared loss $p(\mathcal{D}|g) = \exp(\sum_{i=1}^N (y_i - g_i)^2)$. For a binary classification task, it can be the exponentiated sigmoid cross-entropy loss $p(\mathcal{D}|g) = \exp\left(\sum_{i=1}^N I(y_i = 1)\log p_i + I(y_i = 0)\log(1-p_i)\right)$ where $p_i = \text{sigmoid}(g_i)$.

However, in the context of large-scale neural modeling, performing exact GP inference (8)-(9) is difficult for two reasons: (i) inference with the exact prior (8) is *computationally intractable*, due to the need of inverting a $N \times N$ kernel matrix $\mathbf{K}$ which has a cubic time complexity $\mathcal{O}(N^3)$. (ii) Inference with a non-Gaussian likelihood in (9) is *analytically intractable*, due to the difficulty of deriving the analytical integration over a non-conjugate likelihood. The existing literature commonly handles (i)-(ii) by deploying sophisticated inference algorithms such as structured Variational Inference (VI) or Markov chain Monte Carlo (MCMC) (see Section 5 for a full review). However, this invariably imposes nontrivial engineering and computational complexity to an existing deep learning system, rendering itself infeasible in many practical scenarios where the system scalability and maintainability is of high importance.

In this work, we propose to tackle the above challenge by performing Laplace-approximation inference to the random Fourier features (RFFs) expansion of the GP model (Rasmussen and Williams, 2006), by (i) converting the GP model (8) into a featurized Bayesian linear model via the random-feature expansion (Rahimi and Recht, 2008b). Then (ii) approximate the intractable posterior (9) using Laplace approximation (Gelman et al., 2013). Both are well-established methods in the statistical machine learning literature with rigorous theoretical guarantees (Rahimi and Recht, 2008a; DeGroot and Schervish, 2012). We re-iterate that the goal here is *not* to develop another algorithm to approximate the exact GP posterior, but to identify a simple, practical baseline approach to implement the distance-awareness property (Definition 2) into a neural model with minimal modification to the training pipeline for a deterministic DNN. To this end, as we will show, the proposed approach leads to a closed-form posterior that can be trained end-to-end using the same pipeline for a deterministic DNN and with comparable time complexity. Empirically, the trained model maintains the generalization performance of a deterministic DNN while illustrating improved uncertainty performance in calibration and in OOD detection.

Random-feature Expansion of GP Model. First, we approximate the prior defined in (8) by defining a low-rank approximation of the kernel matrix $\mathbf{K} = \Phi\Phi^\top$ by using random features (Rahimi and Recht, 2008b), which gives us a *random-feature Gaussian process*:

$$g_{N\times 1} \sim MVN(\mathbf{0}_{N\times 1}, \Phi\Phi^\top_{N\times N}), \quad \text{where} \quad \Phi_{i,D_L\times 1} = \sqrt{2\sigma^2/D_L} * \cos(-\mathbf{W}_L h_i + \mathbf{b}_L). \tag{10}$$

Here, $h_i = h(\mathbf{x}_i)$ represents the hidden representations of the penultimate layer with a dimensionality of D_{L-1}. $\Phi_i = \phi(\mathbf{x}_i)$ is the final layer with dimension D_L, which is represented by (1) a fixed weight

matrix $\mathbf{W}_{L,D_L \times D_{L-1}}$ whose entries are sampled i.i.d from $N(0,1)$, and (2) a fixed bias term $\mathbf{b}_{L,D_L \times 1}$ whose entries are sampled i.i.d from $Uniform(0, 2\pi)$. We can therefore write the logits using the RFF expansion to the GP prior in (8) as a neural network layer consisting of fixed hidden weights $\mathbf{W}$ and learnable output weights β:

$$g(h_i) = \sqrt{2\sigma^2/D_L} * \cos(-\mathbf{W}_L h_i + \mathbf{b}_L)^\top \beta, \quad \text{with prior} \qquad \beta_{D_L \times 1} \sim N(0, \tau * \mathbf{I}_{D_L \times D_L}). \tag{11}$$

Laplace Approximation for Posterior Uncertainty. Notice that conditional on h, the output weights β are the only learnable parameters in the model. As a result, the random Fourier feature (RFF) approximation in (11) 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 (Minka, 2001). In this work, we choose the Laplace method due to its simplicity and the fact that its posterior variance has a convenient closed form (Rasmussen and Williams, 2006). Briefly, the Laplace method approximates the model posterior $p(\beta|\mathcal{D})$ using a Gaussian distribution that is centered around the maximum a posterior (MAP) estimate $\hat{\beta} = \text{argmax}_\beta \, p(\beta|\mathcal{D})$, such that

$$p(\beta|\mathcal{D}) \approx MVN(\hat{\beta}, \hat{\Sigma} = \widehat{\mathbf{H}}^{-1}), \quad \text{where} \quad \widehat{\mathbf{H}}_{(i,j)} = -\frac{\partial^2}{\partial \beta_i \partial \beta_j} \log p(\beta|\mathcal{D})|_{\beta=\hat{\beta}} \tag{12}$$

is the $D_L \times D_L$ Hessian matrix of the log posterior likelihood evaluated at the MAP estimates. For a binary classification task, the posterior precision matrix (i.e., the inverse covariance matrix) adopts a simple expression $\hat{\Sigma}_k^{-1} = \mathbf{I} + \sum_{i=1}^N \hat{p}_i(1-\hat{p}_i)\Phi_i\Phi_i^\top$, where p_i is the model prediction $p_i = sigmoid(\hat{g}_i)$ under the MAP estimates $\hat{g}_i = \Phi_i^\top \hat{\beta}$ (Rasmussen and Williams, 2006). We introduce the extensions to regression and multi-class classification in Appendix A.1.

To summarize, for a classification task, the Laplace posterior for an approximate GP under the RFF expansion is:

$$\beta|\mathcal{D} \sim MVN(\hat{\beta}, \hat{\Sigma}), \quad \text{where} \quad \hat{\Sigma}^{-1} = \tau * \mathbf{I} + \sum_{i=1}^N \hat{p}_i(1-\hat{p}_i)\Phi_i\Phi_i^\top, \tag{13}$$

where $\hat{\beta}$ is the model's MAP estimate conditioned on the RFF hidden representation Φ in (10), and τ is the prior variance. To obtain the posterior distribution (13), one only need to first obtain the MAP estimate by training the entire network with respect to the MAP objective using stochastic gradient descent (SGD):

$$-\log p(\beta, \{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L-1}|\mathcal{D}) = -\log p(\mathcal{D}|\beta, \{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L-1}) + \frac{1}{2\tau}||\beta||^2, \tag{14}$$

where $\{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L-1}$ are the hidden weights of the network and $-\log p(\mathcal{D}|\beta, \{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L-1})$ is the negative log likelihood for the task (e.g., the cross entropy loss for a classification task). Then, in the final epoch, we can compute this covariance in an incremental fashion by initializing covariance with I and accumulating covariance contributions from each batch as in (13) [4] [5]. As a result, the

[4] i.e., to update the posterior precision matrix as $\hat{\Sigma}_t^{-1} = \hat{\Sigma}_{t-1}^{-1} + \sum_{i=1}^M \hat{p}_i(1-\hat{p}_i)\Phi_i\Phi_i^\top$ for minibatches of size M. Notice that to obtain a MAP estimate $\hat{\beta}$ that is strictly conditioned on the hidden representations Φ, one should freeze the hidden representations in the final epoch and only update the output weights β. However in practice, we find this has no significant impact on the final performance. This is most likely due to the fact that the hidden representation has already become stabilized in the final epochs.

[5] In the online learning setting where the model is processing a infinite stream of data, this expression can be modified to $\hat{\Sigma}_t^{-1} = (1-m) * \hat{\Sigma}_{t-1}^{-1} + m * \sum_{i=1}^M \hat{p}_i(1-\hat{p}_i)\Phi_i\Phi_i^\top$ where m is a small scaling coefficient. This is analogous to the exponential moving average estimator for batch variance as used in batch normalization (Ioffe and Szegedy, 2015)

approximate GP posterior (13) can be learned scalably and in closed-form with minimal modification to the training pipeline of a deterministic DNN. It is worth noting that under the RFF posterior, the Laplace approximation is in fact asymptotically exact by the virtue of the Bernstein-von Mises (BvM) theorem and the fact that (11) is a finite-rank model (Freedman, 1999; LeCam, 1973; Panov and Spokoiny, 2015; Dehaene, 2019).

3.2 Approximately Distance-preserving Hidden Mapping via Spectral Normalization

Replacing the output layer g with an approximate Gaussian process only allows the model $\text{logit}(\mathbf{x}) = g \circ h(\mathbf{x})$ 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 *approximately 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 data manifold $d_X(\mathbf{x}, \mathbf{x}')$. 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 residual mappings $\{g_l\}_{l=1}^{L-1}$ in the residual blocks $h_l(\mathbf{x}) = \mathbf{x} + g_l(\mathbf{x})$. Specifically, it can be shown that, under suitable regularity conditions, a *strict* distance preservation can be guaranteed by restricting the Lipschitz constants to be less than 1. We state this result formally below:

Proposition 3 (Lipschitz-bounded residual block is distance preserving (Bartlett et al., 2018))**.** *Consider a hidden mapping $h : \mathcal{X} \to \mathcal{H}$ with residual architecture $h = h_{L-1} \circ \ldots h_2 \circ h_1$ where $h_l(\mathbf{x}) = \mathbf{x} + g_l(\mathbf{x})$ and $\mathcal{X}$ and $\mathcal{H}$ are of equal dimension. If for $0 < \alpha \leq 1$, all g_l's are α-Lipschitz, i.e., $||g_l(\mathbf{x}) - g_l(\mathbf{x}')||_H \leq \alpha d_X(\mathbf{x}, \mathbf{x}') \quad \forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}$. Then:*

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

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 $\mathcal{X}$ and the hidden space $\mathcal{H}$ 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 (Bartlett et al., 2018; Behrmann et al., 2019; Hauser and Ray, 2017b; Rousseau et al., 2020).

However in practice, a *strict* preservation of distance is both impossible and likely undesirable. The reason is that per common neural network practice, the model often projects a high-dimensional example into a lower-dimensional representation (i.e., $|\mathcal{H}| < |\mathcal{X}|$ for the hidden mapping $h : \mathcal{X} \to \mathcal{H}$). This necessitates information loss and precludes the possibility of exact isometry (i.e., invertibility) (Smith et al., 2021). Furthermore, an overly strict bound on the Lipschitz constant of g_l can push the model toward identity mapping, greatly restricting the expressiveness of the network and leading to suboptimal generalization (Behrmann et al., 2019). To this end, a more prudent strategy would be to pursue *approximate* distance preservation, i.e., by finetuning the magnitude of Lipschitz constant to a more relaxed extent, so as to balance the tradeoff between the expressiveness of the network and its distance-preservation ability (see Section 8.1 for further discussion).

Algorithm-wise, to ensure the hidden mapping h is approximately distance preserving, it is sufficient to ensure that the weight matrices for the nonlinear residual block $g_l(\mathbf{x}) = a(\mathbf{W}_l\mathbf{x} + \mathbf{b}_l)$ to have spectral norm (i.e., the largest singular value) to be upper-bounded, since $||g_l||_{Lip} \leq ||\mathbf{W}_l\mathbf{x} +$

$\mathbf{b}_l||_{Lip} \leq ||\mathbf{W}_l||_2$. 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 Behrmann et al. (2019). Briefly, at every training step, the SN method first estimates the spectral norm $\hat{\lambda} \approx ||\mathbf{W}_l||_2$ using the power iteration method (Gouk et al., 2021; Miyato et al., 2018), and then normalizes the weights as:

$$\mathbf{W}_l = \begin{cases} c * \mathbf{W}_l / \hat{\lambda} & \text{if } c < \hat{\lambda} \\ \mathbf{W}_l & \text{otherwise} \end{cases} \tag{15}$$

where $c > 0$ is a hyperparameter used to adjust the exact spectral norm upper bound on $||\mathbf{W}_l||_2$ (so that $||\mathbf{W}_l||_2 \leq c$). Therefore, (15) 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. In this work, we treat c as a tunable hyperparameter which can be selected based on the validation data (Appendix A.2).

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:**

$$\hat{\Sigma} = \tau * \mathbf{I}, \mathbf{W}_L \overset{iid}{\sim} N(0,1), \mathbf{b}_L \overset{iid}{\sim} U(0, 2\pi)$$

3: **for** train_step $= 1$ **to** max_step **do**
4: SGD update $\left\{\beta, \{\mathbf{W}_l\}_{l=1}^{L-1}, \{\mathbf{b}_l\}_{l=1}^{L-1}\right\}$ (14)
5: **if** final_epoch **then**
6: Update precision matrix $\hat{\Sigma}^{-1}$ (13).
7: **end if**
8: **end for**
9: Compute posterior covariance $\hat{\Sigma} = \text{inv}(\hat{\Sigma}^{-1})$.

Algorithm 2 SNGP Prediction

1: **Input:** Testing example $\mathbf{x}$.
2: Compute Features:

$$\Phi_{D_L \times 1} = \sqrt{2\sigma^2 / D_L} * \cos(\mathbf{W}_L h(\mathbf{x}) + \mathbf{b}_L)$$

3: Compute Posterior Mean:

$$\text{logit}(\mathbf{x}) = \Phi^\top \beta$$

4: Compute Posterior Variance:

$$\text{var}(\mathbf{x}) = \Phi^\top \hat{\Sigma} \Phi.$$

5: Compute Predictive posterior distribution:

$$p(y|\mathbf{x}) = \int_{g \sim N(\text{logit}(\mathbf{x}), \text{var}(\mathbf{x}))} \text{sigmoid}(g) dg$$

Method Summary & Practical Implementation. We summarize the method in Algorithms 1-2. As shown, during training, the model updates the hidden-layer weights $\{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L-1}$ and the trainable output weights β via minibatch SGD (i.e., exactly the same way as a deterministic DNN). Then, in the final epoch, it also performs an update of the precision matrix using Equation (13). During inference, the model first performs the conventional forward pass to compute the final hidden features $\phi(\mathbf{x})_{D_L \times 1}$, and then compute the posterior mean $m(\mathbf{x}) = \phi(\mathbf{x})^\top \beta$ (time complexity $\mathcal{O}(D_L)$) and the predictive variance $v(\mathbf{x})^2 = \phi(\mathbf{x})^\top \hat{\Sigma} \phi(\mathbf{x})$ (time complexity $\mathcal{O}(D_L^2)$)[6]. To see how this predictive distribution of SNGP reflects the distance-awareness property, notice that conditional on the penultimate embedding h and assuming squared loss, the predictive logit of the SNGP follows a Gaussian process:

$$\begin{aligned} g(\mathbf{x}) &\sim N\big(\text{logit}(\mathbf{x}), \text{var}(\mathbf{x})\big) \quad \text{where} \\ \text{logit}(\mathbf{x}) &= \phi(\mathbf{x})^\top (\Phi^\top \Phi + \tau \mathbf{I})^{-1} \Phi^\top \mathbf{y}, \\ \text{var}(\mathbf{x}) &= \phi(\mathbf{x})^\top (\Phi^\top \Phi + \tau \mathbf{I})^{-1} \phi(\mathbf{x}), \end{aligned} \tag{16}$$

[6]To extend Algorithm 1-2 to regression or multi-class classification task, one just need to use the appropriate precision matrix update as introduced in A.1, and replace the sigmoid output activation to identity or softmax.

where Φ is the $N \times D_L$ random feature embedding of the training data. Then, straightforward application of Woodbury matrix identity reveals that (16) can equivalently be expressed in the familiar dual form of the Gaussian process (Rasmussen and Williams, 2006):

$$\text{logit}(\mathbf{x}) = \mathbf{k}^*(\mathbf{x})^\top (\mathbf{K} + \tau \mathbf{I})^{-1} \mathbf{y}, \tag{17}$$

$$\text{var}(\mathbf{x}) = \tau^{-1} * \left[k(\mathbf{x},\mathbf{x}) - \mathbf{k}^*(\mathbf{x})^\top (\mathbf{K} + \tau \mathbf{I})^{-1} \mathbf{k}^*(\mathbf{x})\right], \tag{18}$$

where $k(\mathbf{x},\mathbf{x})_{1\times 1} = \phi(\mathbf{x})^\top \phi(\mathbf{x})$, $\mathbf{k}^*(\mathbf{x})_{N\times 1} = \phi(\mathbf{x})^\top \Phi^\top$ and $\mathbf{K}_{N\times N} = \Phi\Phi^\top$ are kernel matrices approximating those under the RBF kernel $k(\mathbf{x},\mathbf{x}') \propto \exp(-||h(\mathbf{x}) - h(\mathbf{x}')||_2^2)$. Consequently, under a distance-preserving mapping h, for a test example $\mathbf{x}$ that is moving away from the training-data manifold, its predictive kernel matrix $\mathbf{k}^*(\mathbf{x}) = [k(\mathbf{x},\mathbf{x}_i)]_{i=1}^N$ systematically approaches 0, while the testing-data kernel $k(\mathbf{x},\mathbf{x})$ and the training-data kernel $\mathbf{K}$ remain in a constant range. This causes the predictive mean $m(\mathbf{x})$ to approach zero, and the predictive variance $v(\mathbf{x})$ to approach its maximum [7]. As a result, the SNGP model achieves the behavior as motivated in (4), i.e., generating a maximum-entropy predictive distribution for OOD inputs that are far outside the training domain. We include a proof of this behavior in Appendix C.3.

To estimate the predictive distribution $p(y|\mathbf{x}) = \int_{g\sim N(\text{logit}(\mathbf{x}),\text{var}(\mathbf{x}))} \text{softmax}(g) dg$, we can use either of two approaches: Monte Carlo averaging or mean-field approximation. Specifically, Monte Carlo averaging randomly generates K i.i.d samples from the distribution $N(\text{logit}(\mathbf{x}), \text{var}(\mathbf{x}))$, compute the softmax, and take the average over the K samples, $p(y|\mathbf{x}) = \frac{1}{K}\sum_{k=1}^K \text{softmax}(g_k)$. Note that this Monte Carlo approximation is applied just in last layer, so it is very fast and does not require multiple forward passes of the network. However, in ML systems where Monte Carlo sampling is difficult or considered too expensive (e.g., the on-device settings), we can alternatively consider mean-field approximation to the softmax Gaussian posterior (Daunizeau, 2017; Lu et al., 2020):

$$p(y|\mathbf{x}) = \text{softmax}\Big(\frac{\text{logit}(\mathbf{x})}{\sqrt{1 + \lambda * \text{var}(\mathbf{x})}}\Big), \tag{19}$$

where λ is a scaling factor that is commonly set to $\pi/8$ (Lu et al., 2020). In our experiments, we have found the two approaches to perform similarly. Therefore we report the results from mean-field approximation in our experiments due to its wider applicability.

Interestingly, Equation (19) highlights the role of the kernel amplitude σ in improving a classifier's calibration performance: as the predictive variance $var(\mathbf{x})$ is proportional to σ (see Equation (18)), it effectively re-scales the magnitude of the logits and functions similarly to the temperature parameter in the temperature scaling technique (Guo et al., 2017). We also note that in our experiment, the value of σ generally does not have a significant impact on model performance in OOD detection. This is likely due to the fact that scaling the model uncertainty by a constant σ does not change the ranking of the uncertainty scores. In practice, we recommend estimating σ on a small amount of in-distribution validation data by minimizing with respect to a proper scoring rule. This is because an overparameterized model such as a DNN is known to overfit the training data, rendering the in-sample estimate of the second-order statistics not reliable (Wahba, 1990). In all experiments, we estimate σ by minimizing it against the logarithm score (i.e., the marginalized negative log likelihood (Gneiting

[7] This conclusion also holds for non-Gaussian outcome. Where the predictive mean and variance can be expressed as $m(\mathbf{x}) = \mathbf{k}^*(\mathbf{x})^\top (\mathbf{K} + \tau \mathbf{I})^{-1} \hat{\mathbf{y}}$ and $v(\mathbf{x}) = \tau^{-1} k(\mathbf{x},\mathbf{x}) - \mathbf{k}^*(\mathbf{x})^\top \hat{\mathbf{V}} \mathbf{k}^*(\mathbf{x})$, where $\hat{\mathbf{y}}$ is the model prediction on the training data, and $\hat{\mathbf{V}}$ is the inverse covariance matrix calculated using the Laplace method (i.e., via Equation (13)). (Rasmussen and Williams (2006), Chapter 3.6)

et al., 2007)) on a small held-out validation dataset. We summarize all other model hyperparameters (e.g., the spectral norm upper bound c) and provide practical recommendations in Appendix A.2.

4. Where does SNGP fit in the current landscape of methods?

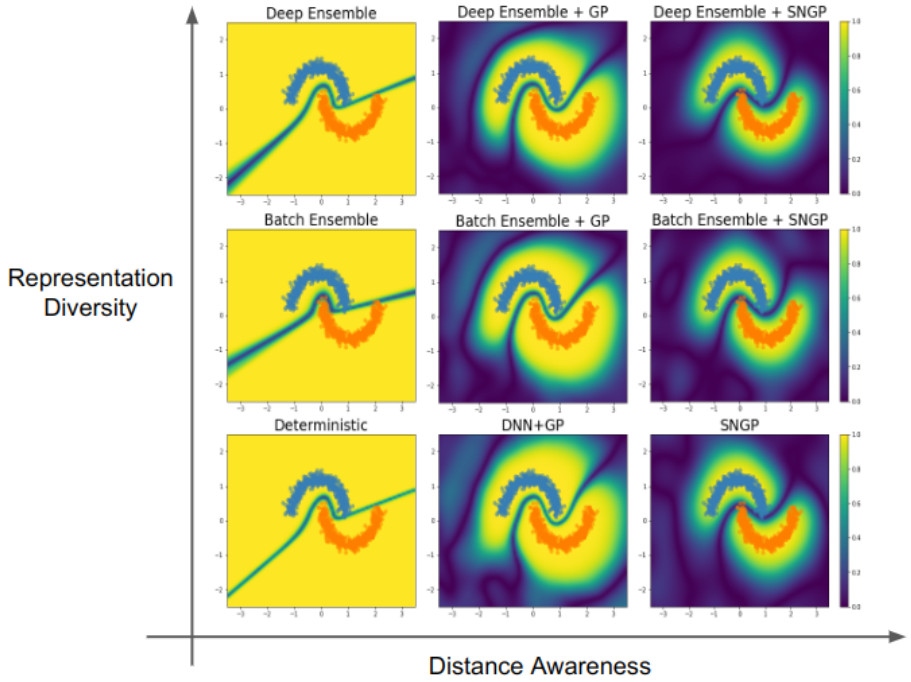


Figure 2: We highlight complementary ways of improving uncertainty of a vanilla DNN. **The x-axis highlights improvements to single model uncertainty $p_\theta(y|x)$ via distance-awareness.** Note that in this setting, we're focused on improving quality of uncertainty for a fixed deterministic representation. **The y-axis shows an orthogonal direction of improvement by combining multiple models $\theta_1, \theta_2, \ldots, \theta_M$.** Note that ensembles work primarily by leveraging multiple representations (as opposed to the deterministic uncertainty quantification setting). This reveals some interesting differences: for instance, the left most column which uses vanilla DNN as the base model to construct deep ensembles (top-left) and efficient ensembles (top-middle), shows that ensembling strategies provide qualitatively different benefits as they primarily add uncertainty near the boundary and do not necessarily change the confidence landscape far away from the data. On the other hand, the bottom row shows that methods such as DNN-GP (bottom-middle) and SNGP (bottom-right) improve distance awareness in the classification boundary. **Finally, these two axes are not really competing approaches; they actually provide complementary benefits** and can be combined to further improve performance. See Section 7.1.

We have introduced SNGP as a method to improve single model uncertainty. Before describing the experimental setup, we first explain where SNGP fits in the current landscape of uncertainty methods in deep learning.

SNGP is a method for *deterministic* uncertainty quantification, that is, given a single, non-random representation, SNGP improves the base model by enhancing its distance-awareness property. This is a orthogonal direction to what was taken by popular ensemble-based approaches (e.g., Monte-Carlo dropout, BatchEnsemble (Wen et al., 2020) or Deep Ensemble), who improve performance primarily through integrating over multiple diverse representations (Fort et al., 2019). However, when base models consistently make overconfident predictions far away from the data, their ensemble can inherit such behavior as well. As a result, while ensembling vanilla DNNs are effective in improving

accuracy and calibration under shift, they can be lacking in providing as significant a boost in OOD detection (e.g., due to the lack of distance awareness).

To see this point visually, in Figure 2 where we highlight the two orthogonal axes of improvement. The y-axis denotes better ensembling strategies: the left column shows that ensembling vanilla DNNs does not necessarily change the confidence landscape far away from the inputs. Even though efficient ensembles reduce compute and memory, they are still solving a different problem from SNGP by improving representation diversity but in an efficient manner. The x-axis denotes better single model uncertainty; the bottom row shows how DNN-GP and SNGP improve deterministic uncertainty quantification by improving distance preservation and distance awareness.

Finally, we note that these two axes provide complementary benefits, so they can be combined to further improve performance (see SNGP ensemble in top right). Since SNGP primarily improves single model uncertainty, we focus our comparisons with other methods for deterministic uncertainty quantification in the experiments (Section 6). Whenever possible, we recommend to ensemble SNGP models build toward a strong probabilistic DNN model that simultaneously quantifies representation uncertainty and ensures distance awareness. We explore ensembles of SNGP in Section 7.1. As an aside, data augmentation is another orthogonal axis of improvement (Hendrycks* et al., 2020). We also explore SNGP as a building block for data augmentation methods in Section 7.2.

5. 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 (Wen et al., 2020; Dusenberry et al., 2020), 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 (Hein et al., 2019b; Malinin and Gales, 2018a,b; Sensoy et al., 2018b; Shu et al., 2017), the output layer (Bendale and Boult, 2016; Tagasovska and Lopez-Paz, 2019; Calandra et al., 2016; Macedo et al., 2020; Macêdo and Ludermir, 2021; Padhy et al., 2020), computing a closed-form posterior for the output layer (Riquelme et al., 2018; Snoek et al., 2015; Kristiadi et al., 2020) , or predicting a-priori uncertainty away from training data (Skafte et al., 2019). 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 (Van Amersfoort et al., 2020). However, empirically, the two-sided gradient penalty can lead to unstable training dynamics for a deep residual network, and is observed to over-constrain the model capacity in more difficulty tasks (e.g., CIFAR-100) in our experiments (Section 6.2.1). Following the initial conference publication of this work (Liu et al., 2020a) and in a similar vein, some later work also investigated building other class of probabilistic models on top of a spectral-regularized network, e.g., variational Gaussian process, Gaussian mixture model, or stochastic differential equations (Mukhoti et al., 2021; van Amersfoort et al., 2021; Cui et al., 2021). While obtaining encouraging results on small-scale benchmarks (e.g., CIFAR), these approaches are still computationally prohibitive

for large-scale tasks with high number of output classes (e.g., ImageNet) which SNGP can easily scale to (Section 6.2.2).

Laplace approximation and GP inference with DNN. Laplace approximation has a long history in GP and (Bayesian) NN literature (Tierney et al., 1989; Denker and LeCun, 1991; Rasmussen and Williams, 2006; MacKay, 1992; Ritter et al., 2018; Hobbhahn et al., 2022; Kristiadi et al., 2022, 2021; Eschenhagen et al., 2021; Daxberger et al., 2021), and the theoretical connection between a Laplace-approximated DNN and GP has being explored recently (Khan et al., 2019). 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 learns the hidden representation separately Salakhutdinov and Hinton (2007), or performs end-to-end learning via MAP estimation (Calandra et al., 2016) or structured VI (Hensman et al., 2015; Bradshaw et al., 2017; Wilson et al., 2016b). These approaches were shown to lead to poor calibration by recent work (Tran et al., 2019), which proposed a simple fix by combing Monte Carlo Dropout (MC Dropout) with random Fourier features, which we term MC Dropout Gaurssian 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 (i.e., the DNN-GP + Dropout method in Section 7).

Distance-preserving neural networks and bi-Lipschitz condition. The theoretic connection between distance preservation and the bi-Lipschitz condition is well-established (Searcod, 2006), and learning an approximately isometric, distance-preserving transform has been an important goal in the fields of dimensionality reduction (Blum, 2006; Perrault-Joncas and Meila, 2012), generative modeling (Lawrence and Quinonero-Candela, 2006; Dinh et al., 2014, 2016; Jacobsen et al., 2018), and adversarial robustness (Jacobsen et al., 2019a; Ruan et al., 2018; Sokolic et al., 2017; Tsuzuku et al., 2018; Weng et al., 2018). This work is a novel application of the distance preservation property for uncertainty quantification. There are several methods for controlling the Lipschitz constant of a DNN (e.g., gradient penalty or norm-preserving activation (An et al., 2015; Anil et al., 2019; Chernodub and Nowicki, 2017; Gulrajani et al., 2017)), 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 (Bartlett et al., 2018; Behrmann et al., 2019; Rousseau et al., 2020; Behrmann et al., 2021). Finally, Smith et al. (2021) studied the effect of spectral regularization on the residual networks in the context of image recognition, and concluded that, under mild assumptions, residual networks will be approximately distance preserving on the low-passed portion of their input.

Open Set Classification. The uncertainty risk minimization problem in Section 2 assumes a data-generation mechanism similar to the *open set recognition* problem (Scheirer et al., 2014), 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 (Grünwald and Dawid, 2004; Landes, 2015). 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 (Hafner et al., 2020; Harang and Rudd, 2018; Lee et al., 2018a; Malinin and Gales, 2018b; Meinke and Hein, 2020; Hendrycks et al., 2018). Our analysis provides 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 (*distance awareness*) that enables strong OOD performance in discriminative classifiers without the need of explicit generative modeling.

6. Benchmarking Experiments

In this section, we benchmark the performance of the SNGP model by applying it on a variety of both toy datasets and real-world tasks across different modalities Specifically, we first benchmark the behavior of the approximate GP layer versus an exact GP, and illustrate the impact of spectral normalization on toy regression and classification tasks (Section 6.1). We then conduct a thorough benchmark study to compare the performance of SNGP against the other state-of-the-art methods on popular benchmarks such as CIFAR-10 and CIFAR-100 (Section 6.2.1. Finally, we illustrate the scalability of and the generality of the SNGP approach by applying it to a large-scale image recognition task (ImageNet, Section 6.2.2), and highlight the broad usefulness by applying SNGP to uncertainty tasks in two other data modalities, namely conversational intent understanding and genomics sequence identification (Section 6.3).

6.1 Low-dimensional Experiments

6.1.1 REGRESSION

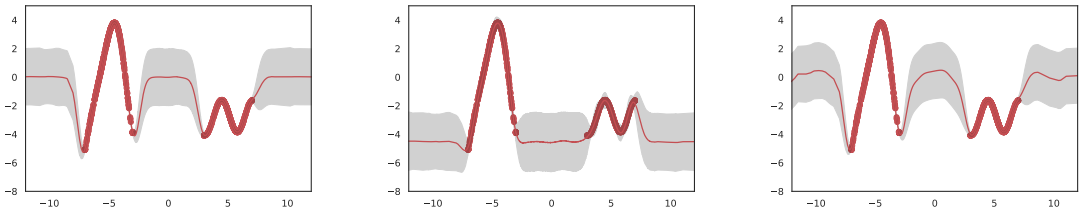
We first benchmark the performance of the *Random-Feature Gaussian process* layer (RFGP), which form a key component of the SNGP model, versus both an exact GP formulation and a variational Gaussian process (VGP) (Titsias, 2009) on a 1D toy regression task. We consider the bimodal toy regression example from van Amersfoort et al. (2021), and use use the RBF kernel for all three models . As seen from Figure 3a, the GP posterior mean for the exact GP tends to zero in the absence of training data, with low variance near training points and high variance otherwise. The RFGP mimics this behavior, with the addition of some periodic artifacts, which arises naturally due to the periodic nature of the approximation. In comparison, the prediction from the VGP model at locations that are further away from data tend to depend on the location of the inducing points, and does not revert back to the original zero-mean Gaussian process prior (e.g, between the two modes and at two ends of Figure 3). Although a more modern treatment of VGP may address this issue (Dutordoir et al., 2020) [8].

6.1.2 CLASSIFICATION

After having benchmarked the predictive and uncertainty behavior of the RFF-GP algorithm compared to an exact GP formulation, in the subsequent subsections we use it as a subcomponent in the SNGP algorithm.

In this subsection, we compare the behavior of SNGP on a suite of 2D classification tasks. Specifically, we consider the *two ovals* benchmark (Figure 4, row 1) and the *two moons* benchmark (Figure 4, 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.

[8] Note that the goal of this toy example is not to claim RFGP outperforms VGP, but just to illustrate that the RFGP, despite its simplicity, indeed returns a valid uncertainty surface that is distance-aware. To this end, van Amersfoort et al. (2021) studied the extension of SNGP method under VGP models, and obtained promising result on toy datasets.



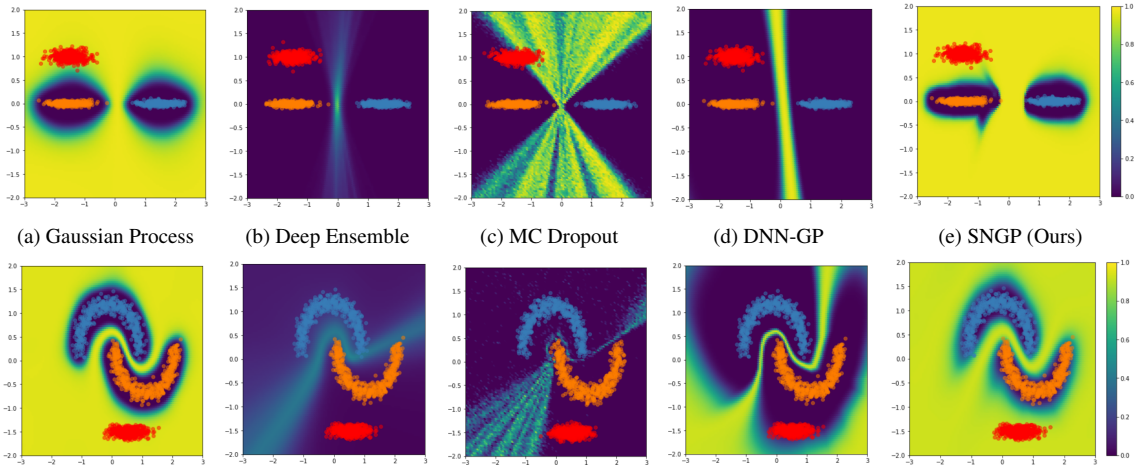
(a) Exact GP (b) Variational GP (c) Random Fourier Features

Figure 3: 1D toy Regression Task: The models are trained using the training datapoints sampled from a bimodal Gaussian distribution (shown as blue dots), and then return both a posterior mean and variance for the entire test domain $[-12, 12]$. Exact GP (left) returns predictions with zero mean and high posterior variance on test points away from the training data distribution, and low posterior variance on the training points, whereas Variational GP (middle) predicts a mean that is negative on unseen points. RFF-GP (right) closely mimics the behavior of Exact GP, with the addition of some periodic fluctuations arising due to the use of Fourier features to approximate the covariance matrix.

There also exists an OOD distribution (red) that the model doesn't observe during training. Similarly, the *two moons* dataset consists of two moon-shaped distributions separable by a non-linear decision boundary. 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 input and was trained with Hamiltonian Monte Carlo (HMC). 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 a single ResFFN-12-128 model with dense output layer and 10 dropout samples at test time. **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 spectral normalization. The full experimental details for these algorihms are in Appendix C.

Figure 4 shows the results of training these algorithms to achieve high (close to 100 % accuracy) on the test data, and visualizes the uncertainty surface output by each model. Each algorithm returns a predictive distribution of the form $p(y|\mathbf{x})$, following which the confidence of the prediction can be written as $\hat{p} = \max_y p(y|\mathbf{x})$. We plot the uncertainty surface by defining $u(x) = \hat{p}(1-\hat{p})$, and normalize it to the range of $[0, 1]$ by dividing it by 0.25. Firstly, we notice that the exact Gaussian process models (without using a DNN, Figures 4a, 4f) exhibit the expected behavior for high-quality predictive uncertainty: they predict low uncertainty in the region $\mathcal{X}_{\texttt{IND}}$ supported by the training data (blue color), and predict high uncertainty when $\mathbf{x}$ is far from $\mathcal{X}_{\texttt{IND}}$ (yellow color), i.e., *distance-awareness*. As a result, the exact GP model is able to assign low confidence to the OOD data (colored in red), indicating reliable uncertainty quantification. On the other hand, Deep Ensemble (Figures 4b, 4g) and MC Dropout (Figures 4c, 4h) 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, due to a pathological behavior where the uncertainty is only high near the decision boundary (linear



(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 4: 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 4e and 4j for color map). See Section 5.1 for details. Visualization of additional methods (e.g., a single deterministic DNN and ensemble of SNGPs) is shown in Figure 2.

for *two ovals* and non-linear for *two moons*). Finally, the DNN-GP (Figures 4d and 4i) and SNGP (Figures 4e and4j) 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, SNGP is able to maintain the *distance-awareness* property via its bi-Lipschitz constraint, and exhibits a uncertainty surface that is analogous to the gold-standard model (exact GP) despite the fact that SNGP is based on a deep 12-layer network.

6.2 Image Classification

Baseline Methods All methods included in the vision and language understanding experiments are summarized in Table 1. We compare SNGP against a suite of algorithms; a deterministic baseline, two single-model approaches: **Deterministic Uncertainty Quantification (DUQ)**, and **Enhanced Isotropic Maximization (IsoMax+)**. The original DUQ consists of two novel components, one is a RBF kernel based loss function which computes the distance between the feature vector and the class centroids, and the other is adding gradient penalty to avoid feature collapse. As pointed in the paper, RBF networks prove difficult to optimize and scale to large number of output classes (e.g., CIFAR-100), so we instead replace the RBF layer with our GP layer. Thus we call this variant as **Deterministic Uncertainty Quantification with GP Last Layer (DUQ-GP)**. **IsoMax+** replaces the softmax cross-entropy loss with a IsoMax+ loss which minimizes the distance to the correct class prototype based on the normalized feature embeddings in the last layer (Macêdo and Ludermir, 2021). We also include two ablations 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). Finally, to evaluate the SNGP's efficacy as a base model for ensemble approaches, we also train ensemble of SNGP and compare it with two popular approaches that quantifies representation diversity: **MC Dropout** (with 10 dropout samples) and **Deep Ensemble** (with 10 models), both are trained with a dense output layer and no spectral regularization. Section 7 further explores the interplay between SNGP and various ensemble approaches.

For all models that use GP layer, we keep $D_L = 1024$ and compute predictive distribution by performing mean-field approximation to the softmax Gaussian posterior. Further experiment details and recommendations for practical implementation are in Appendix C. All experiments are built on the `uncertainty_baselines` framework [9] (Nado et al., 2021). We open-source our code there.

Methods	Additional Regularization	Output Layer	Representation Uncertainty	Multi-pass Inference
DNN	-	Dense	-	-
DNN-IsoMax+	-	Dense	-	-
DUQ-GP	Gradient Penalty	GP	-	-
DNN-SN	Spec Norm	Dense	-	-
DNN-GP	-	GP	-	-
SNGP	Spec Norm	GP	-	-
MC Dropout	Dropout	Dense	Yes	Yes
Deep Ensemble	-	Dense	Yes	Yes
SNGP Ensemble	Spec Norm	GP	Yes	Yes

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.

6.2.1 CIFAR-10 AND CIFAR-100

For the CIFAR-10 and CIFAR-100 image classification benchmarks, we use a Wide ResNet 28-10 model as the base for all methods (Zagoruyko and Komodakis, 2017). Following the benchmarking setup first suggested in Ovadia et al. (2019), we evaluate the model's predictive accuracy, negative log-likelihood (NLL), and expected calibration error (ECE) under both clean CIFAR testing data and its corrupted versions termed CIFAR-*-C (Hendrycks and Dietterich, 2018). To evaluate the model's OOD detection performance, we consider two tasks: a standard *far*-OOD task using SVHN as the OOD dataset for a model trained on CIFAR-10/-100, and a difficult *near*-OOD task using CIFAR-100 as the OOD dataset for a model trained on CIFAR-10, and vice versa. We use the in-distribution training data statistics to preprocess and normalize the OOD datasets. In Tables 2 and 3, we use the maximum softmax probability (MSP) as the uncertainty score while performing OOD evaluation, and report the Area Under the Reciever Operator Curve (AUROC) metric.

Tables 2-3 report the results. As shown, for predictive accuracy, SNGP is competitive with other single-model approaches. For calibration error, GP-based models clearly outperform the other single-model approaches and are competitive with Deep Ensemble and MC Dropout approaches. We also observe that under MSP metric, IsoMax+ method produces underconfident predictions, leading to the highest calibration error. Furthermore, its performance degrades quickly on more

[9] https://github.com/google/uncertainty-baselines

complex tasks (e.g., CIFAR-100) both in terms of accuracy and in uncertainty quality[10]. That is consistent with the findings of Padhy et al. (2020). For OOD detection, SNGP mostly outperforms other single-model approaches that are based on a dense output layer (except for CIFAR-10 vs. CIFAR-100 where IsoMax+ does best), and is competitive with deep ensembles, MC Dropout approaches and DUQ-GP. Finally, ensemble using SNGP as base model strongly outperforms all the other approaches, illustrating the importance of the *distance-awareness* property for high-quality performance in uncertainty quantification and the composability of SNGP as a building block toward the state-of-the-art probabilistic deep models.

Method	Accuracy (↑)		ECE (↓)		NLL (↓)		OOD AUROC (↑)	
	Clean	Corrupted	Clean	Corrupted	Clean	Corrupted	SVHN	CIFAR-100
Single Model								
DNN	95.8 ± 0.190	79.0 ± 0.350	0.028 ± 0.002	0.153 ± 0.005	0.183 ± 0.007	1.042 ± 0.038	0.946 ± 0.005	0.893 ± 0.001
DNN-IsoMax+	95.6 ± 0.190	79.1 ± 0.230	0.525 ± 0.003	0.439 ± 0.003	0.214 ± 0.008	1.276 ± 0.041	**0.959 ± 0.006**	**0.918 ± 0.002**
DUQ-GP	95.8 ± 0.120	78.2 ± 0.460	0.027 ± 0.001	0.149 ± 0.005	0.186 ± 0.008	1.160 ± 0.049	0.939 ± 0.007	0.900 ± 0.003
DNN-SN	**96.0 ± 0.170**	79.1 ± 0.290	0.027 ± 0.002	0.150 ± 0.004	0.179 ± 0.006	1.019 ± 0.027	0.945 ± 0.005	0.894 ± 0.002
DNN-GP	95.8 ± 0.150	79.1 ± 0.430	**0.017 ± 0.003**	0.100 ± 0.009	**0.149 ± 0.006**	0.761 ± 0.039	**0.964 ± 0.006**	0.902 ± 0.002
SNGP (Ours)	95.7 ± 0.140	**79.3 ± 0.340**	**0.017 ± 0.003**	**0.099± 0.008**	**0.149 ± 0.005**	**0.745 ± 0.026**	**0.960 ± 0.004**	0.902 ± 0.003
Ensemble Model								
MC Dropout	95.7 ± 0.130	78.6 ± 0.430	0.013 ± 0.002	0.099 ± 0.005	0.145 ± 0.004	0.829 ± 0.021	0.934 ± 0.004	0.903 ± 0.001
Deep Ensemble	**96.4 ± 0.090**	80.4 ± 0.150	0.011 ± 0.001	0.092 ± 0.003	0.124 ± 0.001	0.768 ± 0.009	0.947 ± 0.002	0.914 ± 0.000
SNGP Ensemble (Ours)	**96.4 ± 0.040**	**81.1 ± 0.130**	**0.009 ± 0.001**	**0.042 ± 0.002**	**0.114 ± 0.001**	**0.590 ± 0.005**	**0.967 ± 0.002**	**0.920 ± 0.001**

Table 2: Results for Wide ResNet-28-10 on CIFAR-10, averaged over 10 seeds. Bold indicates the on-average best-performing methods among the single-model methods and the ensemble methods. SNGP almost always outperform other single model variants, and SNGP ensemble outperform other ensemble methods.

Method	Accuracy (↑)		ECE (↓)		NLL (↓)		OOD AUROC (↑)	
	Clean	Corrupted	Clean	Corrupted	Clean	Corrupted	SVHN	CIFAR-10
Single Model								
DNN	80.4 ± 0.290	55.0 ± 0.180	0.107 ± 0.004	0.258 ± 0.004	0.941 ± 0.016	2.663 ± 0.045	0.799 ± 0.020	0.795 ± 0.001
DNN+IsoMax+	77.3 ± 0.330	52.9 ± 0.190	0.663 ± 0.004	0.455 ± 0.002	1.289 ± 0.014	3.800 ± 0.069	0.770 ± 0.026	0.786 ± 0.002
DUQ-GP	79.7 ± 0.200	54.0 ± 0.280	0.112 ± 0.002	0.269 ± 0.006	0.988 ± 0.013	2.998 ± 0.100	0.777 ± 0.026	0.789 ± 0.002
DNN-SN	**80.5 ± 0.300**	55.0 ± 0.210	0.111± 0.002	0.268± 0.005	0.951 ± 0.008	2.727± 0.055	0.798± 0.023	0.793± 0.003
DNN-GP	80.3 ± 0.380	**55.3 ± 0.250**	0.034± 0.005	**0.059± 0.002**	0.767± 0.008	**1.918± 0.016**	0.835± 0.021	0.797± 0.001
SNGP (Ours)	80.3 ± 0.230	**55.3 ± 0.190**	**0.030 ± 0.004**	0.060 ± 0.004	**0.761 ± 0.007**	1.919 ± 0.013	**0.846 ± 0.019**	**0.798 ± 0.001**
Ensemble Model								
MC Dropout	80.2 ± 0.220	54.6 ± 0.002	0.031 ± 0.002	0.136 ± 0.005	0.762 ± 0.008	2.198 ± 0.027	0.800 ± 0.014	0.797 ± 0.002
Deep Ensemble	**82.5 ± 0.190**	57.6 ± 0.110	0.041 ± 0.002	0.146± 0.003	0.674 ± 0.004	2.078 ± 0.015	0.812 ± 0.007	0.814 ± 0.001
SNGP Ensemble (Ours)	**82.5 ± 0.160**	**58.2 ± 0.130**	**0.028 ± 0.002**	**0.064 ± 0.001**	**0.635 ± 0.003**	**1.752 ± 0.007**	**0.838 ± 0.008**	**0.819 ± 0.001**

Table 3: Results for Wide ResNet-28-10 on CIFAR-100, averaged over 10 seeds. Bold indicates the on-average best-performing methods among the single-model methods and the ensemble methods. SNGP almost always outperform other single model variants, and SNGP ensemble outperform other ensemble methods.

In Appendix C.2 we studied the performance of other uncertainty scores including Dempster-Shafer (Sensoy et al., 2018a), Mahalanobis distance (Lee et al., 2018c), and Relative Mahalanobis distance (Ren et al., 2021) for OOD detection. We found in general the Dempster Shafer metric attains a better OOD performance (e.g., 0.984 AUROC for CIFAR-10 v.s. SVHN) than the widely-used MSP metric reported in the main text.

[10]Contemporary to our work, Macêdo et al. (2022) shows the performance of IsoMax++ in OOD detection can be improved by using more specialized metrics (e.g., minimum distance score (MDS)) instead of MSP.

6.2.2 IMAGENET

We illustrate the scalability of SNGP by experimenting on the large-scale ImageNet dataset (Russakovsky et al., 2015) using a ResNet-50 model as the base for all methods. Similar to the CIFAR benchmarking setup, we follow the procedure from (Ovadia et al., 2019) and evaluate the model's predictive accuracy and calibration error under both clean ImageNet testing data and its corrupted versions termed ImageNet-C. For this complex, large-scale task with high-dimensional output, we find it difficult to scale some of the other single-model methods (e.g., IsoMax or DUQ), which tend to over-constrain the model expressiveness and lead to lower accuracy than a baseline DNN (a phenomenon we already observe in the CIFAR-100 experiment). Therefore we omit them from results. Table 4 reports the results of methods that achieve competitive performance on ImageNet. As shown, for predictive accuracy, SNGP is competitive with that of a deterministic network, despite using a last-layer GP and spectral normalization on the weights of the residual network. For calibration error, SNGP not only outperforms the other single-model approaches and also, interestingly, the Deep Ensemble. Finally, the ensemble using SNGP as the base model attains the strongest predictive accuracy and NLL across all methods. These results illustrate the importance of the *distance-awareness* property for high-quality performance in uncertainty quantification, and the proposed components (i.e., spectral normalization and the GP-layer) perform well in complex, large-scale tasks by maintaining model accuracy while improving the quality of its uncertainty quantification.

Method	**Accuracy** (↑)		ECE (↓)		NLL (↓)	
	Clean	Corrupted	Clean	Corrupted	Clean	Corrupted
Single Model						
DNN	76.2 ± 0.01	40.5 ± 0.01	0.032 ± 0.002	0.103 ± 0.011	0.939 ± 0.01	3.21 ± 0.02
DNN-SN	$\mathbf{76.4 \pm 0.01}$	40.6 ± 0.01	0.079 ± 0.001	0.074 ± 0.001	0.96 ± 0.01	3.14 ± 0.02
DNN-GP	76.0 ± 0.01	$\mathbf{41.3 \pm 0.01}$	0.017 ± 0.001	0.049 ± 0.001	0.93 ± 0.01	3.06 ± 0.02
SNGP (Ours)	76.1 ± 0.01	41.1 ± 0.01	$\mathbf{0.013 \pm 0.001}$	$\mathbf{0.045 \pm 0.012}$	$\mathbf{0.93 \pm 0.01}$	$\mathbf{3.03 \pm 0.01}$
Ensemble Model						
MC Dropout	76.6 ± 0.01	42.4 ± 0.02	0.026 ± 0.002	$\mathbf{0.046 \pm 0.009}$	0.919 ± 0.01	2.96 ± 0.01
Deep Ensemble	77.9 ± 0.01	$\mathbf{44.9 \pm 0.01}$	$\mathbf{0.017 \pm 0.001}$	0.047 ± 0.004	0.857 ± 0.01	2.82 ± 0.01
SNGP Ensemble (Ours)	$\mathbf{78.1 \pm 0.01}$	$\mathbf{44.9 \pm 0.01}$	0.039 ± 0.001	0.050 ± 0.002	$\mathbf{0.851 \pm 0.01}$	$\mathbf{2.77 \pm 0.01}$

Table 4: Results for ResNet 50 on ImageNet, averaged over 10 seeds. Bold indicates the on-average best-performing methods among the single-model methods and the ensemble methods. SNGP outperforms other single model variants, and SNGP ensemble outperform other ensemble methods in generalization and NLL.

6.3 Generalization to other data modalities

6.3.1 CONVERSATIONAL LANGUAGE UNDERSTANDING

To validate the hypothesis that *distance awareness* is a crucial component for good predictive uncertainty on data modalities beyond images, we also evaluate SNGP on a practical language understanding task where uncertainty quantification is of natural importance: dialog intent detection (Larson et al., 2019; Vedula et al., 2019; Yaghoub-Zadeh-Fard et al., 2020; Zheng et al., 2020). 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 a user is in-scope (so it can activate 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 out-of-scope (OOS) intent detection benchmark dataset (Larson et al., 2019). 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 a $BERT_{\texttt{base}}$ model 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. Due the lack of empirically validated implementations of other single-model methods for non-image modalities, we focus on comparing ablated versions of SNGP and the two general-purpose methods: MC Dropout and Deep Ensemble. The results are in Table 5. As shown, consistent with the previous vision experiments, SNGP outperforms other single model approaches. It is competitive with Deep Ensemble in predictive accuracy and calibration, and outperforms all the approaches in OOD detection.

Method	Accuracy (↑)	ECE (↓)	NLL (↓)	OOD	
				AUROC (↑)	AUPR (↑)
Single Model					
DNN	96.5 ± 0.11	0.024 ± 0.002	3.559 ± 0.11	0.897 ± 0.01	0.757 ± 0.02
DNN-SN	95.4 ± 0.10	0.037 ± 0.004	3.565 ± 0.03	0.922 ± 0.02	0.733 ± 0.01
DNN-GP	95.9 ± 0.07	0.075 ± 0.003	3.594 ± 0.02	0.941 ± 0.01	0.831 ± 0.01
SNGP (Ours)	**96.6 ± 0.05**	**0.014 ± 0.005**	**1.218 ± 0.03**	**0.969 ± 0.01**	**0.880 ± 0.01**
Ensemble Model					
MC Dropout	96.1 ± 0.10	0.021 ± 0.001	1.658 ± 0.05	0.938 ± 0.01	0.799 ± 0.01
Deep Ensemble	**97.5 ± 0.03**	0.013 ± 0.002	1.062 ± 0.02	0.964 ± 0.01	0.862 ± 0.01
SNGP Ensemble (Ours)	97.4 ± 0.02	**0.009 ± 0.002**	**0.918 ± 0.03**	**0.973 ± 0.02**	**0.910 ± 0.01**

Table 5: Results for $\text{BERT}_{\texttt{Base}}$ on CLINC OOS, averaged over 10 seeds. Bold indicates the on-average best-performing methods among the single-model methods and the ensemble methods. SNGP outperforms other single model variants, and SNGP ensemble outperform other ensemble methods in uncertainty performance.

6.3.2 BACTERIA GENOMICS SEQUENCE IDENTIFICATION

Here we apply SNGP to a genomic sequence prediction task as another data modality. Ren et al. (2019) proposed the genomics OOD benchmark dataset motivated by the real-world problem of bacteria identification based on genomic sequences, which can be useful for diagnosis and treatment of infectious diseases. A classification model can be trained for classifying known bacteria species with decent test accuracy. However, when deploying the model to real data, the model will be inevitably exposed to the genomic sequences from unknown bacteria species. In fact, the real data can contain approximately $60-80\%$ of sequences from unknown classes that have not been studied before. The model needs to be able to detect those out-of-distribution inputs from unknown species, and abstain from making predictions for them. The genomic OOD benchmark dataset contains 10 bacteria classes as in-distribution for training, and 60 bacteria classes as out-of-distribution for testing. The in-distribution and out-of-distribution bacteria classes were chosen and separated naturally by the year of discovery, to mimic the real scenario that new bacteria species are discovered gradually over the years. Following (Ren et al., 2019), we train a 1D CNN but with SN and GP components added on the 10 in-domain classes and evaluate the models' accuracy, ECE and NLL on the in-domain test data, and also evaluate the model's OOD performance for detecting the 60 OOD classes. In addition to SNGP, we also consider Monte Carlo dropout and Deep Ensemble as another two baselines for comparison. We also study the effect of each component of SN and GP on the model's calibration and OOD detection performance. Section C.1 contains further model detail.

The results are shown in Table 6. Comparing with all the other baseline models, SNGP model achieves the best OOD performance and best in-distribution accuracy, ECE, and NLL, among the

Method	Accuracy (↑)	ECE (↓)	NLL (↓)	OOD AUROC (↑)	OOD AUPR (↑)
Single Model					
DNN	84.40 ± 0.390	0.049 ± 0.007	0.487 ± 0.007	0.640 ± 0.005	0.609 ± 0.005
DNN-SN	85.56 ± 0.150	0.025 ± 0.003	0.422 ± 0.004	0.658 ± 0.006	0.625 ± 0.005
DNN-GP	85.23 ± 0.200	0.041 ± 0.006	0.457 ± 0.005	0.654 ± 0.006	0.629 ± 0.003
SNGP (Ours)	**85.71 ± 0.100**	**0.019 ± 0.004**	**0.417 ± 0.004**	**0.672 ± 0.011**	**0.637 ± 0.009**
Ensemble Model					
MC Dropout	84.16 ± 0.370	0.033 ± 0.003	0.480 ± 0.003	0.641 ± 0.004	0.609 ± 0.004
Deep Ensemble	87.21 ± 0.630	**0.014 ± 0.006**	0.373 ± 0.012	0.671 ± 0.005	0.640 ± 0.005
SNGP Ensemble (Ours)	**88.19 ± 0.560**	0.049 ± 0.004	**0.357 ± 0.012**	**0.687 ± 0.008**	**0.656 ± 0.007**

Table 6: Results for Genomics sequence prediction and OOD detection, average over 10 seeds. Bold indicates the on-average best-performing methods among the single-model methods and the ensemble methods.

single models. Each of the two components, SN and GP helps to reduce ECE and NLL, and improve OOD detection over the baseline DNN model. SNGP Ensemble is better than DNN Ensemble and MC Dropout in terms of in-distribution accuracy and NLL, and OOD detection.

7. SNGP as a Building Block for Probabilistic Deep Learning

From a probabilistic machine learning perspective, the SNGP algorithm provides an efficient way to learn a single high-quality deterministic model $p_\theta(y|\mathbf{x})$ by, 1) improving the representation learning quality of the underlying neural network model through spectral normalization, and 2) introducing *distance awareness* in conjunction with an approximate GP random feature layer. However, in the literature, there exist other competitive and state-of-the-art probabilistic methods that improve a deep learning model's predictive uncertainty via different approaches.

The first class of such approaches is ***ensembling*** that quantifies a model's uncertainty in the hidden layers by marginalizing over an (implicit) probabilistic distribution of model parameters. The most notable examples of this class is Deep Ensemble that averages parallel-trained deep learning models that are initialized from distinct random seeds (Lakshminarayanan et al., 2017; Fort et al., 2019), and also MC Dropout (Gal and Ghahramani, 2016) that samples from a functional-space model distribution that is generated by perturbing the model's dropout masks. As these approaches are uniquely capable of quantifying hidden-representation uncertainty, we hypothesize that they would synergize very well with SNGP, which is a single-model, last-layer-oriented approach.

Data augmentation (Thulasidasan et al., 2019; Hendrycks* et al., 2020; Cubuk et al., 2020; Chen et al., 2020a,b) represents a second class of approach that improves the representation learning of neural networks by explicitly injecting expert knowledge about the types of surface-form perturbations that a model should be sensitive to or invariance against, thereby encouraging the model to learn a meaningful distance in its representation space. Compared to the spectral normalization technique which provides a *global* guarantee in distance preservation that applies to the entire input space (Proposition 3), the data augmentation methods provides a *local* guarantee in the neighborhood of the training data. However, this guarantee is also expected to be *stronger* (i.e., a better correspondence between the hidden-space distance $||.||_H$ and a suitable metric d_X for the input data manifold) since we have explicitly instructed the model representation to be sensitive to or invariant against the semantically meaningful or meaningless directions as specified by human experts, respectively (see

Section 8.1 for further discussion). Consequently, in the case where the coverage of the training data is sufficiently large, and a well-designed augmentation library is available for the task, data augmentation is expected to complement the spectral-normalization technique to provide a even stronger guarantee in distance-awareness.

In this section, we investigate the efficacy of the SNGP model as a building block for other uncertainty estimation techniques, and show that it leads to improvements that are complementary to those achieved by the other methods. Specifically, we consider the following methods from literature:

- **Ensemble Methods**:
  - MC Dropout (Gal and Ghahramani, 2016) uses dropout regularization at test-time as a way to obtain samples from a predictive posterior distribution. SNGP can be easily combined with MC Dropout by enabling dropout at the inference time.
  - Deep Ensemble (Lakshminarayanan et al., 2017) train multiple SNGP models with different random seed initializations and then average the predictions during test-time.
- **Data Augmentation** We consider AugMix (Hendrycks* et al., 2020), a data augmentation algorithm that is known to improve the robustness and uncertainty estimation properties of a network by improving the learned representations of data through a combination of expert-designed data augmentations.It is easy to augment the SNGP training procedure by simply plugging in the Augmix augmentations into the data processing pipeline before feeding the data into the SNGP model [11].

Method	CIFAR-10		
	Acc / cAcc (↑)	ECE / cECE (↓)	AUROC SVHN / CIFAR-100 (↑)
DNN	95.8 ± 0.190 / 79.0 ± 0.350	0.029 ± 0.002 / 0.153 ± 0.005	0.946 ± 0.005 / 0.893 ± 0.001
DNN-SN	**96.0 ± 0.170** / 79.1 ± 0.290	0.027 ± 0.002 / 0.150 ± 0.004	0.945 ± 0.005 / 0.894 ± 0.002
DNN-GP	95.8 ± 0.150 / 79.1 ± 0.430	**0.017 ± 0.003** / 0.100 ± 0.009	**0.964 ± 0.006 / 0.902 ± 0.002**
SNGP	95.7 ± 0.140 / **79.3 ± 0.340**	**0.017 ± 0.003 / 0.099 ± 0.008**	0.960 ± 0.004 / **0.902 ± 0.003**
DNN + Dropout	**95.7 ± 0.130** / 78.6 ± 0.430	0.013 ± 0.002 / 0.099 ± 0.005	0.934 ± 0.004 / 0.903 ± 0.001
DNN-SN + Dropout	**95.7 ± 0.100** / 78.6 ± 0.200	0.014 ± 0.001 / 0.097 ± 0.003	0.935 ± 0.006 / 0.902 ± 0.001
DNN-GP + Dropout	**95.7 ± 0.140** / 78.8 ± 0.330	**0.009 ± 0.002 / 0.048 ± 0.004**	**0.960 ± 0.006 / 0.909 ± 0.002**
SNGP + Dropout	95.5 ± 0.130 / **78.9 ± 0.260**	**0.009 ± 0.003** / 0.055 ± 0.007	0.953 ± 0.007 / 0.908 ± 0.002
DNN + Ensemble	96.4 ± 0.090 / 80.4 ± 0.150	0.011 ± 0.001 / 0.092 ± 0.003	0.947 ± 0.002 / 0.914 ± 0.000
DNN-SN + Ensemble	**96.5 ± 0.050** / 80.5 ± 0.090	0.011 ± 0.001 / 0.090 ± 0.002	0.952 ± 0.002 / 0.914 ± 0.001
DNN-GP + Ensemble	96.4 ± 0.060 / 80.8 ± 0.110	0.009 ± 0.001 / **0.041 ± 0.002**	**0.970 ± 0.002 / 0.920 ± 0.001**
SNGP + Ensemble	96.4 ± 0.040 / **81.1 ± 0.130**	**0.008 ± 0.001** / 0.042 ± 0.002	0.967 ± 0.002 / **0.920 ± 0.000**
DNN + AugMix	**96.6 ± 0.150 / 87.5 ± 0.003**	0.014 ± 0.002 / 0.035 ± 0.002	0.958 ± 0.007 / 0.923 ± 0.002
DNN-SN + AugMix	**96.6 ± 0.110 / 87.5 ± 0.300**	0.013 ± 0.001 / 0.035 ± 0.002	0.958 ± 0.005 / 0.923 ± 0.001
DNN-GP + AugMix	96.4 ± 0.240 / 87.2 ± 0.330	0.010 ± 0.002 / 0.028 ± 0.004	**0.978 ± 0.004 / 0.927 ± 0.002**
SNGP + AugMix	96.2 ± 0.200 / 87.1 ± 0.330	**0.009 ± 0.003 / 0.026 ± 0.005**	0.976 ± 0.005 / 0.925 ± 0.003

Table 7: Results for CIFAR-10 when SNGP is used as a building block combined with MC Dropout, Ensemble, and Augmix. Best result in each class of method highlighted in bold.

[11] For ease of composability, we did not add the further consistency regularization through the Jensen-Shannon divergence as done in the original AugMix paper, as we do not notice further significant improvements on adding the loss on top of the Augmix augmentations.

Method	CIFAR-100		
	Acc / cAcc (↑)	ECE / cECE (↓)	AUROC SVHN / CIFAR-10 (↑)
DNN	80.4 ± 0.290 / 55.0 ± 0.180	0.107 ± 0.004 / 0.258 ± 0.004	0.799 ± 0.020 / 0.795 ± 0.001
DNN-SN	**80.5 ± 0.300** / 55.0 ± 0.210	0.111 ± 0.002 / 0.268 ± 0.005	0.798 ± 0.022 / 0.793 ± 0.003
DNN-GP	80.3 ± 0.400 / **55.3 ± 0.300**	0.034 ± 0.005 / **0.059 ± 0.002**	0.835 ± 0.021 / 0.797 ± 0.001
SNGP	80.3 ± 0.230 / **55.3 ± 0.190**	**0.030 ± 0.004** / 0.060 ± 0.004	**0.846± 0.019 / 0.798 ± 0.001**
DNN + Dropout	80.2 ± 0.220 / 54.6 ± 0.160	**0.031 ± 0.002** / 0.136 ± 0.005	0.800 ± 0.014 / 0.797 ± 0.002
DNN-SN + Dropout	**80.5 ± 0.340** / 54.6 ± 0.260	0.035 ± 0.003 / 0.143 ± 0.004	0.781 ± 0.018 / 0.797 ± 0.001
DNN-GP + Dropout	80.2 ± 0.360 / **54.9 ± 0.230**	0.126 ± 0.007 / **0.116 ± 0.004**	0.825 ± 0.020 / 0.802 ± 0.003
SNGP + Dropout	80.3 ± 0.370 / **54.9 ± 0.240**	0.128 ± 0.008 / **0.116 ± 0.006**	**0.840 ± 0.021 / 0.804 ± 0.003**
DNN + Ensemble	82.5 ± 0.190 / 57.6 ± 0.110	0.041 ± 0.002 / 0.146 ± 0.003	0.812 ± 0.007 / 0.814 ± 0.001
DNN-SN + Ensemble	**82.7 ± 0.120** / 57.7 ± 0.120	0.044 ± 0.002 / 0.151 ± 0.003	0.804 ± 0.005 / 0.814 ± 0.001
DNN-GP + Ensemble	82.4 ± 0.120 / **58.2 ± 0.110**	0.030 ± 0.002 / 0.065 ± 0.001	0.828 ± 0.008 / 0.817 ± 0.001
SNGP + Ensemble	82.5 ± 0.160 / **58.2 ± 0.130**	**0.028 ± 0.002 / 0.064 ± 0.001**	**0.838 ± 0.008 / 0.819 ± 0.001**
DNN + AugMix	81.6 ± 0.003 / **66.4 ± 0.280**	0.082 ± 0.003 / 0.131 ± 0.005	0.814 ± 0.025 / **0.798 ± 0.003**
DNN-SN + AugMix	81.9 ± 0.280 / **66.4 ± 0.240**	0.080 ± 0.002 / 0.133 ± 0.004	0.824 ± 0.021 / 0.796 ± 0.002
DNN-GP + AugMix	81.6 ± 0.350 / 66.2 ± 0.220	**0.042 ± 0.004** / 0.066 ± 0.002	0.855 ± 0.019 / 0.797 ± 0.002
SNGP + AugMix	81.6 ± 0.240 / **66.4 ± 0.190**	**0.042 ± 0.004 / 0.064 ± 0.002**	**0.870 ± 0.024 / 0.798 ± 0.001**

Table 8: Results for CIFAR-100 when SNGP is used as a building block combined with MC Dropout, Ensemble, and Augmix. Best result in each class of method highlighted in bold.

7.1 Ensembling Approaches for Hidden Representation Uncertainty

We find that the SNGP approach complements extremely well with the ensemble methods in practice. From Tables 7 and 8, we can see that ensembling SNGP members either through MC Dropout or through Deep Ensemble results in improved performance on all metrics when compared to a single model, and in particular improves both the in-distribution calibration and the OOD detection for both CIFAR-10 and CIFAR-100 datasets. Though there are some benefits to adding either spectral normalization or a last-layer GP separately to ensemble members, the combined effect of both parts of the SNGP algorithm consistently results in the best calibration and OOD performance, especially in CIFAR-100 (which is a harder task). This behavior suggests that ensembling models does not inherently address the caveats in the representations of the underlying members, and further improvement can be obtained by imposing spectral normalization and the GP layer to the base models to improve their distance awareness property. To put it another way, SNGP offers an orthogonal improvement to the model's uncertainty quality that cannot be obtained from ensembling, and combining these two approaches can result in the best overall performance. Therefore, wherever resource permits, we suggest using an ensemble of SNGP base models to compound the benefits of *distance awareness* and *representation diversity* in improving the quality of uncertainty quantification (Section 4). To conclude, even though ensembling different neural network instantiations improves epistemic uncertainty quantification by marginalizing from different points in the posterior, improving the quality of each member of the ensemble is crucial to further improve the predictive uncertainty.

7.2 Data Augmentation for Improved Distance Awareness

As shown in the introduction (Figure 1), applying spectral normalization to residual neural networks encourages the model representation to be distance preserving and prevents it from feature collapse. Interestingly, this overlaps with the design goal of another well-known uncertainty technique: *data augmentation* (e.g., AugMix (Hendrycks* et al., 2020)). Specifically, data augmentation

improves the distance preservation ability of the representation by forcing the model to be invariant against the semantic-preserving perturbations (e.g., rotating an image), and also being sensitive to semantic-modifying perturbations (e.g., adding a word "not" to a natural language sentence). To empirically investigate the composability of these two techniques, we perform ablation experiments of training SNGP models on the CIFAR-10 and CIFAR-100 datasets with AugMix, and show that the improvements in representations learnt with AugMix complement SNGP. Results are shown in Table 7-8. If data augmentation improves distance awareness, we would expect to see DNN-GP + AugMix significantly improve over DNN-GP; this is indeed the case, which confirms our hypothesis that increasing distance awareness (either through smoothness or data augmentation) improve DNN-GP. It is interesting to observe that in this task, while the SNGP + AugMix on average outperforms its ablated counterparts, the gap between DNN-GP + AugMix vs. SNGP + AugMix is lower than DNN-GP vs. SNGP, illustrating the effectiveness of AugMix in improving model uncertainty via enhancing the distance awareness. Therefore, for domains where a suite of well-designed augmentation is available for the task, and when the test data is in a neighborhood of the augmented training examples (which is the case for CIFAR-C), the data augmentation method can provide a strong guarantee in preserving a meaningful distance in the input space.

In summary, these experiments illustrate the importance of the representation's distance-awareness quality in improving the model's uncertainty quality. Consequently, for datasets and modalities where a well-designed augmentation library is available (e.g., AugMix for image modality), it is advantageous to complement SNGP with the data augmentation approaches to obtain a stronger guarantee in preserving a semantically meaningful distance in its representation space. However, when it is difficult to obtain high-quality augmentations (e.g. the genomics example), or it is computationally intractable to sufficiently augment the training data, the SNGP alone provides a cheap, modality-invariant approach to improve trained representations that provides a *global* guarantee for distance preservation (see Section 8.1 for further discussion).

8. Conclusions and Discussion

We propose SNGP, a simple approach to improve the predictive uncertainty estimation of a single deterministic DNN. 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 *distance awareness*, the key design principle behind SNGP, via a decision-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 in prediction, calibration and out-of-domain detection, outperforms other single-model approaches, and combines well with other state-of-the-art uncertainty techniques.

8.1 Further Discussions

A central goal of this work is to provide theoretical and empirical evidences for the importance of incorporating *distance awareness* (i.e., the distance of a test example from the training data) into a model's uncertainty estimate. This provides a complementary view to the classic approaches to deep learning uncertainty, where the model uncertainty is primarily quantified by a test example's distance from the decision boundary (e.g., Figure 1). Indeed, both the *distance to training data* and the

distance to decision boundary are reasonable quantifiers of model uncertainty, and should be treated as equally important components of a practitioner's uncertainty quantification toolbox. In practice, the choice between the two distances should be made based on the nature of the data generating mechanism and the optimality criteria that the practitioner wish to pursue. For example, under the classic i.i.d. assumption where the test examples always stays in-domain (i.e., identically distributed as the training data) and one wish to use model uncertainty to detect ambiguous examples, then *distance to the decision boundary* is a suitable choice. On the other hand, in a safe-critical setting where it is important to guard against worst-case risk (i.e., Section 2.1) in the presence of likely distributional shift, *distance to the training data* would be a more appropriate choice. Alternatively, one may consider selecting a uncertainty metric that incorporates both types of distances. For example, as revealed by Equation (19), the posterior predictive mean of SNGP in fact incorporates both the magnitude of the predictive logit (i.e., distance to the decision boundary) and predictive variance (i.e., distance to the training data).

Furthermore, an important observation we made in this work is that *learning smooth representations is important for good uncertainty quantification*. In particular, we highlighted *bi-Lipschitz* (Equation (7)) as an important condition for the learned representation of a DNN to attain high-quality uncertainty performance. We proposed spectral normalization as a simple approach to ensure such property in practice, and illustrated its practical effectiveness across a wide range of data modalities. Perhaps surprisingly, the improvement is observed even in the high-dimensional regime (e.g., image and text), where the true "semantic" distance seemingly diverges from that based on the surface-form representation (Section 6), and the benefit of SNGP seems to be not fully overlapping with those provided by the other state-of-the-art uncertainty techniques (Section 7). To this end, we find it theoretically relevant to initiate a discussion about the role that the bi-Lipschitz condition may play in high-dimensional and overparameterized learning, and how it compares with those of the other state-of-the-art approaches. The following discussion is in no way comprehensive, and is intended to serve as potential starting points for future work.

Preservation of "semantic" distance. At a first glance, the technique proposed in Section 3.2 seems to act on a naive, surface-level distance in the input space (e.g., L_2 distance in the pixel space), hence raising the question whether the "semantic" distance d_X (i.e., a meaningful distance metric that is appropriate for the input data manifold) is being preserved as well. Indeed, in common machine learning applications, the data points reside on an underlying low-dimensional manifold behind their high-dimensional surface representation [12]. Here, the correspondence between the true manifold distance d_X (i.e., the "semantic distance") and the surface-level distance (e.g., L_2 distance in the pixel space) is complex and dynamic, and can be described by the concept of *metric distortion* from metric embedding theory (Abraham et al., 2011; Matoušek, 2013; Chennuru Vankadara and von Luxburg, 2018). Formally, consider $m : (\mathcal{X}, d_S) \to (\mathcal{X}, d_X)$ a mapping between the metric space $(\mathcal{X}, d_S)$ equipped with surface-level distance to that equipped with the true distance $(\mathcal{X}, d_X)$. Then, $d_X(\mathbf{x}_1, \mathbf{x}_2)$ can be understood as a *distorted* version of the surface-level distance $d_S(\mathbf{x}_1, \mathbf{x}_2)$ as implemented by m, with the type and degree of the distortion changes depending on the location in the product feature space $(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{X} \times \mathcal{X}$. More specifically, defining $\rho(\mathbf{x}_1, \mathbf{x}_2) = \frac{d_X(\mathbf{x}_1, \mathbf{x}_2)}{d_S(\mathbf{x}_1, \mathbf{x}_2)}$ the distance ratio that measures the degree of local distortion at $(\mathbf{x}_1, \mathbf{x}_2)$, two types of distortions may occur:

[12] Appendix D provides an example mathematical formalization that further elaborates this idea.

Definition 4 (Distance Distortion)**.**

- ***(Distance Contraction)*** *$d_X(\mathbf{x}_1,\mathbf{x}_2) < d_S(\mathbf{x}_1,\mathbf{x}_2)$, such that the magnitude of $\rho(x_1,x_2)$ is high.*
- ***(Distance Expansion)*** *$d_X(\mathbf{x}_1,\mathbf{x}_2) > d_S(\mathbf{x}_1,\mathbf{x}_2)$, such that the magnitude of $1/\rho(x_1,x_2)$ is high.*

The first scenario (distance contraction) is common in both image and text modalities, where the underlying content of an image / text is invariant to the noisy movements in the pixel / token space. The second scenario (distance expansion) occurs often in language understanding, where the addition of a single token (e.g., "not") drastically changes the meaning of a sentence. Furthermore, the type and degree of distortion is location- and direction-dependent. For example, in the language domain, a sentence's meaning is generally invariant to its surface-level syntactic forms (i.e., distance contraction), however some sentences can be drastically changed by the modification of a few key tokens (i.e., distance expansion).

Interestingly, despite this complicated and dynamic distortion of distance, it is possible to show that, as long as the "semantic" distance d_X is well defined (i.e., $0 \leq d_X(\mathbf{x}_1,\mathbf{x}_2) < \infty, \forall \mathbf{x}_1,\mathbf{x}_2 \in \mathcal{X}$), the global bound on the surface-level distance still provides a basic guarantee in preserving the local true distances at every $(\mathbf{x}_1,\mathbf{x}_2)$ (albeit at a looser degree):

Proposition 5 (Preserving "semantic" distance under metric distortion)**.** *Assume that the mapping $h : \mathcal{X} \to \mathcal{H}$ preserves the surface-form distance d_S, i.e., there exist non-negative constants $L_1 < L_2$ such that:*

$$L_1 \times d_S(x_1,x_2) \leq ||h(x_1) - h(x_2)||_H \leq L_2 \times d_S(x_1,x_2), \tag{20}$$

then, for a well-defined surface-form distance $d_S \in [0,\infty)$ that is a distortion of d_X, we have:

- ***(Distance Contraction)*** *For a local region $(\mathbf{x}_1,\mathbf{x}_2)$ where the distance contraction occurs with magnitude $\rho(\mathbf{x}_1,\mathbf{x}_2) = l'$, there exists a L_2' such that $L_2' > l' * L_2$ and*

$$L_1 \times d_X(x_1,x_2) \leq ||h(x_1) - h(x_2)||_H \leq L_2 \times d_S(x_1,x_2) \leq L_2' \times d_X(x_1,x_2),$$

- ***(Distance Expansion)*** *For a local region $(\mathbf{x}_1,\mathbf{x}_2)$ where the distance expansion occurs with magnitude $1/\rho(\mathbf{x}_1,\mathbf{x}_2) = l'$, there exists a L_1' such that $L_1' \leq L_1/l'$ and*

$$L_1' \times d_X(x_1,x_2) \leq L_1 \times d_S(x_1,x_2) \leq ||h(x_1) - h(x_2)||_H \leq L_2 \times d_X(x_1,x_2).$$

As a result, by bounding the neural network model from warping the representation space distance $||.||_H$ to an extreme degree, the Lipschitz bound (20) provides a basic guarantee in the preservation of the "semantic" distance that is appropriate for the data manifold. Consequently, the SNGP approach concretely improves the unregularized training of the overparameterized network by helping it to reach a better balance between *dimension reduction* and *information preservation*, which we discuss the next.

Balancing the information tradeoff in high-dimensional learning. As the information of machine learning data tends to concentrate on a low-dimensional manifold, a high-dimensional learning model is often confronted with a tradeoff between the need of *dimension reduction* and the need of *information preservation*: dimension reduction is necessary for reducing the statistical complexity of the learning problem and for ensuring superior generalization under finite data. However, an excessive reduction of information leads the model to ignore semantically meaningful features that are less correlated with the training label, creating a challenge for uncertainty quantification.

In the context of modern overparameterized networks, naive training without regularization is often observed to lead to one extreme end of this tradeoff (i.e., excessive dimension reduction). Specifically, by minimizing the training cross entropy toward zero, the model pushes its logits $logit(\mathbf{x}) = h(\mathbf{x})^T\beta$ to a large magnitude at the location of training labels y, leading to a high alignment between training logits and the corresponding labels. Viewing from the representation space $h(\mathbf{x}) \in H$, this manifests into the training data representations $h(x)$ being pushed far away from the decision boundary β, which is often achieved by stretching the hidden-space distance $||.||_H$ along the task-relevant directions (i.e., those orthogonal to the decision boundaries) to an extreme degree. This essentially leads to a warped hidden-space geometry with extremely high weighting on a handful of task-relevant principal directions (see, e.g., Figure 1 top), implying low effective degrees of freedom (Papyan, 2020; Hauser and Ray, 2017a). Although not detrimental to the in-domain generalization, this warped geometry leads the model to be overly sensitive to the hidden-space movements along the label-relevant direction, while being insensitive to semantic change in the directions that are less correlated with the training labels, thereby creating challenges for uncertainty tasks such as out-of-domain detection (Hein et al., 2019a).

To this end, the SNGP approach helps the model to strike a better balance between dimension reduction v.s. information preservation by modulating the degree of geometry distortion via the Lipschitz bound. As a result, the model is still able to disentangle input features through a cascade of layer-wise transformations, but is protected from extreme distortion in H to a degree that causes feature collapse (see, e.g., Figure 4 bottom). As a result, the model is still capable of learning abstract and task-relevant features by exploiting the expressive power of overparameterization, while not blind to semantically meaningful movements that are less correlated with training labels, thereby leading to a better balance between generalization and uncertainty performance.

Alternative approaches, limitations, and future work. It is worth noting that there exist other representation learning techniques, e.g., data augmentation, constrastive learning or unsupervised pretraining, that are known to also improve a network's uncertainty performance (Hendrycks et al., 2019; Hendrycks* et al., 2020). From the perspective of distance awareness, these methods help the model representation to preserve a semantically meaningful distance by injecting external knowledge into the learning pipeline. For example, data augmentation and constrastive learning instructs the model to learn distance contraction or expansion by creating semantically similar or dissimilar pairs $(\mathbf{x}_1, \mathbf{x}_2)$ using expert knowledge, so that the DNN representation is invariant against the semantic-preserving perturbations (e.g., rotating an image), and is sensitive to semantic-modifying perturbations (e.g., adding a word "not" to a natural language sentence). On the other hand, pretraining injects the model with a prior about the similarity or dissimilarity between examples that was learned (e.g., via contrastive masked language modeling) from a large external corpus. Compared to these data-intensive approaches, spectral normalization provides a global guarantee in distance preservation and does not require external resources such as pretraining datasets or an understanding

of which augmentations are semantically meaningful and useful. However, the guarantee is also looser since it only places a uniform upper and lower bound on $||.||_H$ along all directions of the perturbation, rather than explicitly training $||.||_H$ to be contractive or expansive with respect to the direction of semantic perturbation. Interestingly, as shown in Sections 6 - 7, SNGP combines well with these additional techniques, indicating that the benefits they provide to the model do not conflict with each other.

Finally, we note that the Gaussian process algorithm presented this work is optimized for practicality (i.e., scalability to extremely large datasets, and seamless integration into minibatch SGD-based neural training pipeline). Such design goal invariably necessitates several approximations for the exact Gaussian process posterior: (1) random-feature expansion for the kernel function, (2) Laplace approximation for the posterior variance, and (3) mean-field approximation for the softmax posterior mean. In our preliminary experiments, we found that a higher-quality approximation to the kernel function (e.g., using orthogonal random feature (Yu et al., 2016) rather than random Fourier feature) and an near-exact Monte Carlo approximation to the softmax posterior mean did not lead to a meaningful improvement to model performance. However, this does not preclude the theoretical possibility that an exact Gaussian process posterior can further improve uncertainty quality when compared to the current algorithm choice. Therefore, identifying GP algorithms that can better implement the distance-awareness principle without sacrificing practicality is an important direction for future work.

Acknowledgments

We would like to thank Rodolphe Jenatton, D. Sculley, Kevin Murphy, Deepak Ramachandran for the insightful comments and fruitful discussions.

References

Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. *Advances in Mathematics*, 228(6):3026–3126, 2011.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *arXiv:1606.06565 [cs]*, June 2016.

Senjian An, Farid Boussaid, and Mohammed 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.

Cem Anil, James Lucas, and Roger Grosse. Sorting Out Lipschitz Function Approximation. In *International Conference on Machine Learning*, pages 291–301, May 2019. ISSN: 1938-7228 Section: Machine Learning.

Francis Bach. Breaking the Curse of Dimensionality with Convex Neural Networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017. ISSN 1533-7928.

Peter Bartlett, Steven Evans, and Phil Long. Representing smooth functions as compositions of near-identity functions with implications for deep network optimization. *arXiv*, 2018.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen. Invertible Residual Networks. In *International Conference on Machine Learning*, pages 573–582, May 2019. ISSN: 1938-7228 Section: Machine Learning.

Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Joern-Henrik Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR, 2021.

Abhijit Bendale and Terrance E. Boult. Towards Open Set Deep Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2 edition, 1985. ISBN 978-0-387-96098-2.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, April 2011. ISBN 978-0-387-31073-2.

Avrim Blum. Random Projection, Margins, Kernels, and Feature-Selection. In *Subspace, Latent Structure and Feature Selection*, Lecture Notes in Computer Science, pages 52–68, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-34138-3. doi: 10.1007/11752790_3.

Charles Blundell, Yee Whye Teh, and Katherine A Heller. Bayesian rose trees. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 65–72, 2010.

John Bradshaw, Alexander G. de G. Matthews, and Zoubin Ghahramani. Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks. *arXiv:1707.02476 [stat]*, July 2017. arXiv: 1707.02476.

Jochen 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.

Roberto Calandra, Jan Peters, Carl E. Rasmussen, and Marc Peter Deisenroth. Manifold Gaussian Processes for regression. *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016. doi: 10.1109/IJCNN.2016.7727626.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia 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, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for english. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174, 2018.

Dhivya Chandrasekaran and Vijay Mago. Evolution of semantic similarity—a survey. *ACM Computing Surveys (CSUR)*, 54(2):1–37, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020a.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Leena Chennuru Vankadara and Ulrike von Luxburg. Measures of distortion for machine learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Artem Chernodub and Dimitri Nowicki. Norm-preserving Orthogonal Permutation Linear Unit Activation Functions (OPLU). *arXiv:1604.02313 [cs]*, January 2017. arXiv: 1604.02313.

Krzysztof Choromanski, Mark Rowland, Tamas Sarlos, Vikas Sindhwani, Richard Turner, and Adrian Weller. The Geometry of Random Features. In *International Conference on Artificial Intelligence and Statistics*, pages 1–9, March 2018.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

Peng Cui, Zhijie Deng, Wenbo Hu, and Jun Zhu. Accurate and reliable forecasting using stochastic differential equations. *arXiv preprint arXiv:2103.15041*, 2021.

Andreas Damianou and Neil Lawrence. Deep Gaussian Processes. In *Artificial Intelligence and Statistics*, pages 207–215, April 2013.

Jean Daunizeau. Semi-analytical approximations to statistical moments of sigmoid and softmax mappings of normal variables. *arXiv preprint arXiv:1703.00091*, 2017.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless Bayesian deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Morris H DeGroot and Mark J Schervish. *Probability and statistics*. Pearson Education, 2012.

Guillaume P. Dehaene. A deterministic and computable Bernstein-von Mises theorem. *ArXiv*, 2019.

John S. Denker and Yann LeCun. Transforming Neural-Net Output Levels to Probability Distributions. In *Advances in Neural Information Processing Systems 3*, pages 853–859. Morgan-Kaufmann, 1991.

Thomas Deselaers and Vittorio Ferrari. Visual and semantic similarity in imagenet. In *CVPR 2011*, pages 1777–1784. IEEE, 2011.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. *arXiv:1410.8516 [cs]*, October 2014. arXiv: 1410.8516.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv:1605.08803 [cs, stat]*, May 2016. arXiv: 1605.08803.

Mike Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors. *Proceedings of the International Conference on Machine Learning*, 1, 2020.

Vincent Dutordoir, Nicolas Durrande, and James Hensman. Sparse gaussian processes with spherical harmonic features. In *International Conference on Machine Learning*, pages 2793–2802. PMLR, 2020.

Runa Eschenhagen, Erik Daxberger, Philipp Hennig, and Agustinus Kristiadi. Mixtures of laplace approximations for improved post-hoc uncertainty in deep learning. *CoRR*, abs/2111.03577, 2021.

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

David Freedman. Wald Lecture: On the Bernstein-von Mises theorem with infinite-dimensional parameters. *The Annals of Statistics*, 27(4):1119–1141, August 1999. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1017938917.

Yarin Gal and Zoubin Ghahramani. Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059, New York, NY, USA, 2016. JMLR.org.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, Boca Raton, 3 edition edition, November 2013. ISBN 978-1-4398-4095-5.

Tilmann Gneiting and Adrian E Raftery. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477):359–378, March 2007. ISSN 0162-1459. doi: 10.1198/016214506000001437.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2): 243–268, April 2007. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2007.00587.x.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

Peter D. Grünwald and A. Philip Dawid. Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory. *Annals of Statistics*, 32(4):1367–1433, August 2004. ISSN 0090-5364, 2168-8966. doi: 10.1214/009053604000000553. Publisher: Institute of Mathematical Statistics.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein GANs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NeurIPS'17, pages 5769–5779, Long Beach, California, USA, December 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian 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.

Abhishek Gupta, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021. ISSN 2590-0056. doi: https://doi.org/10.1016/j.array.2021.100057.

Danijar Hafner, Dustin Tran, Timothy Lillicrap, Alex Irpan, and James Davidson. Noise contrastive priors for functional uncertainty. In *Uncertainty in Artificial Intelligence*, pages 905–914. PMLR, 2020.

Kehang Han, Balaji Lakshminarayanan, and Jeremiah Zhe Liu. Reliable graph neural networks for drug discovery under distributional shift. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

Richard Harang and Ethan M Rudd. Towards principled uncertainty estimation for deep neural networks. *arXiv preprint arXiv:1810.12278*, 2018.

Tatsunori B Hashimoto, David Alvarez-Melis, and Tommi S Jaakkola. Word embeddings as metric recovery in semantic spaces. *Transactions of the Association for Computational Linguistics*, 4: 273–286, 2016.

Michael Hauser and Asok Ray. Principles of Riemannian geometry in neural networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017a.

Michael Hauser and Asok Ray. Principles of Riemannian Geometry in Neural Networks. In *Advances in Neural Information Processing Systems 30*, pages 2807–2816. Curran Associates, Inc., 2017b.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019a.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why ReLU Networks Yield High-Confidence Predictions Far Away From the Training Data and How to Mitigate the Problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019b.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *International Conference on Learning Representations*, Toulon, France, April 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2018.

Dan Hendrycks, Kimin Lee, and Mantas 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.

Dan Hendrycks*, Norman Mu*, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A Simple Method to Improve Robustness and Uncertainty under Data Shift. In *International Conference on Learning Representations*, 2020.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

Marius Hobbhahn, Agustinus Kristiadi, and Philipp Hennig. Fast predictive uncertainty for classification with bayesian deep networks. In *Uncertainty in Artificial Intelligence*, pages 822–832. PMLR, 2022.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Joern-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *International Conference on Learning Representations*, 2019a.

Jörn-Henrik Jacobsen, Jens Behrmannn, Nicholas Carlini, Florian Tramer, and Nicolas Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness. *arXiv preprint arXiv:1903.10484*, 2019b.

Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. i-RevNet: Deep invertible networks. In *International Conference on Learning Representations*, 2018.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Mohammad Emtiyaz E Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate Inference Turns Deep Networks into Gaussian Processes. In *Advances in Neural Information Processing Systems 32*, pages 3094–3104. Curran Associates, Inc., 2019.

Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR, 2020.

Ian Kivlichan, Zi Lin, Jeremiah Liu, and Lucy Vasserman. Measuring and improving model-moderator collaboration using uncertainty estimation. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 36–53, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.woah-1.5.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International conference on machine learning*, pages 5436–5446. PMLR, 2020.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Learnable uncertainty under Laplace approximations. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 344–353. PMLR, 27–30 Jul 2021.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being a bit frequentist improves bayesian neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 529–545. PMLR, 2022.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017.

Jurgen Landes. Probabilism, entropies and strictly proper scoring rules. *International Journal of Approximate Reasoning*, 63:1–21, August 2015. ISSN 0888-613X.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, 2019.

Neil D Lawrence and Joaquin Quinonero-Candela. Local distance preservation in the GP-LVM through back constraints. In *Proceedings of the 23rd international conference on Machine learning*, pages 513–520, 2006.

L. LeCam. Convergence of Estimates Under Dimensionality Restrictions. *The Annals of Statistics*, 1 (1):38–53, January 1973. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1193342380.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. In *International Conference on Learning Representations*, 2018a.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018b.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Advances in Neural Information Processing Systems 31*, pages 7167–7177. Curran Associates, Inc., 2018c.

Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018.

Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan AK Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7128–7148, 2021.

Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in Neural Information Processing Systems*, 33:7498–7512, 2020a.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33:21464–21475, 2020b.

Zhiyun Lu, Eugene Ie, and Fei Sha. Mean-field approximation to Gaussian-softmax integral with application to uncertainty estimation. *arXiv preprint arXiv:2006.07584*, 2020.

David Macêdo and Teresa Ludermir. Enhanced isotropy maximization loss: Seamless and high-performance out-of-distribution detection simply replacing the softmax loss. *arXiv preprint arXiv:2105.14399*, 2021.

David Macedo, Tsang Ing Ren, Cleber Zanchettin, Adriano L. I. Oliveira, Alain Tapp, and Teresa 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]*, February 2020. arXiv: 1908.05569.

David Macêdo, Cleber Zanchettin, and Teresa Ludermir. Distinction maximization loss: Efficiently improving classification accuracy, uncertainty estimation, and out-of-distribution detection simply replacing the loss and calibrating. *arXiv preprint arXiv:2205.05874*, 2022.

David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, May 1992. ISSN 0899-7667. Number: 3 Publisher: MIT Press.

Andrey Malinin and Mark Gales. Predictive Uncertainty Estimation via Prior Networks. In *Advances in Neural Information Processing Systems 31*, pages 7047–7058. Curran Associates, Inc., 2018a.

Andrey Malinin and Mark Gales. Prior Networks for Detection of Adversarial Attacks. *arXiv:1812.02575 [cs, stat]*, December 2018b. arXiv: 1812.02575.

Jirı Matoušek. Lecture notes on metric embeddings. Technical report, Technical report, ETH Zürich, 2013.

Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know. In *International Conference on Learning Representations*, 2020.

Thomas P. Minka. *A family of algorithms for approximate Bayesian inference*. phd, Massachusetts Institute of Technology, USA, 2001. AAI0803033.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations*, 2018.

Saif Mohammad and Graeme Hirst. Distributional measures as proxies for semantic distance: A survey. *Computational Linguistics*, 1(1), 2006.

Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*, 2021.

Zachary Nado, Neil Band, Mark Collier, Josip Djolonga, Michael W Dusenberry, Sebastian Farquhar, Angelos Filos, Marton Havasi, Rodolphe Jenatton, Ghassen Jerfel, et al. Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning. *arXiv preprint arXiv:2106.04015*, 2021.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPR Workshop*, 2019.

Kazuki Osawa, Siddharth Swaroop, Mohammad Emtiyaz E Khan, Anirudh Jain, Runa Eschenhagen, Richard E Turner, and Rio Yokota. Practical Deep Learning with Bayesian Principles. In *Advances in Neural Information Processing Systems 32*, pages 4287–4299. Curran Associates, Inc., 2019.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model' s uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Shreyas Padhy, Zachary Nado, Jie Ren, Jeremiah Liu, Jasper Snoek, and Balaji Lakshminarayanan. Revisiting one-vs-all classifiers for predictive uncertainty and out-of-distribution detection in neural networks. *arXiv preprint arXiv:2007.05134*, 2020.

Maxim Panov and Vladimir Spokoiny. Finite Sample Bernstein von Mises Theorem for Semiparametric Problems. *Bayesian Analysis*, 10(3):665–710, September 2015. ISSN 1936-0975, 1931-6690. doi: 10.1214/14-BA926.

Vardan Papyan. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(252):1–64, 2020.

Matthew Parry, A. Philip Dawid, and Steffen Lauritzen. Proper local scoring rules. *Annals of Statistics*, 40(1):561–592, February 2012. ISSN 0090-5364, 2168-8966. doi: 10.1214/12-AOS971. Publisher: Institute of Mathematical Statistics.

Dominique Perrault-Joncas and Marina Meila. Metric learning and manifolds: Preserving the intrinsic geometry. *Preprint Department of Statistics, University of Washington*, 2012.

Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T Afshar, et al. A universal SNP and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10):983–987, 2018.

Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008a.

Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008b.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. University Press Group Limited, January 2006. ISBN 978-0-262-18253-9. Google-Books-ID: vWtwQgAACAAJ.

Jamie Reilly, Bonnie Zuckerman, Ann Marie Finley, Celia Paula Litovsky, and Yoed Kenett. What is semantic distance? a review and proposed method for modeling conceptual transitions in natural language. 2022.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in neural information processing systems*, 32, 2019.

Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. A simple fix to Mahalanobis distance for improving near-OOD detection. *arXiv preprint arXiv:2106.09022*, 2021.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. In *International Conference on Learning Representations*, 2018.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A Scalable Laplace Approximation for Neural Networks. In *International Conference on Learning Representations*, 2018.

Francois Rousseau, Lucas Drumetz, and Ronan Fablet. Residual Networks as Flows of Diffeomorphisms. *Journal of Mathematical Imaging and Vision*, 62(3):365–375, April 2020. ISSN 1573-7683. doi: 10.1007/s10851-019-00890-3.

Abhijit Guha Roy, Jie Ren, Shekoofeh Azizi, Aaron Loh, Vivek Natarajan, Basil Mustafa, Nick Pawlowski, Jan Freyberg, Yuan Liu, Zach Beaver, et al. Does your dermatology classifier know what it doesn't know? detecting the long-tail of unseen conditions. *Medical Image Analysis*, 75: 102274, 2022.

Wenjie Ruan, Xiaowei Huang, and Marta 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. ISBN 978-0-9992411-2-7.

Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211–252, December 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y.

Ruslan Salakhutdinov and Andriy Mnih. Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 880–887, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390267.

Russ R Salakhutdinov and Geoffrey E Hinton. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in Neural Information Processing Systems*, 2007.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. *Advances in neural information processing systems*, 30, 2017.

Walter J. Scheirer, Lalit P. Jain, and Terrance E. Boult. Probability Models for Open Set Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2317–2324, November 2014. ISSN 1939-3539. doi: 10.1109/TPAMI.2014.2321392. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Micheal O. Searcod. *Metric Spaces*. Springer London, London, 2007 edition edition, August 2006. ISBN 978-1-84628-369-7.

Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *Advances in Neural Information Processing Systems*, 31, 2018a.

Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential Deep Learning to Quantify Classification Uncertainty. In *Advances in Neural Information Processing Systems 31*, pages 3179–3189. Curran Associates, Inc., 2018b.

Lei Shu, Hu Xu, and Bing Liu. Doc: Deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2911–2916, 2017.

Nicki Skafte, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Lewis Smith, Joost van Amersfoort, Haiwen Huang, Stephen Roberts, and Yarin Gal. Can convolutional resnets approximately preserve input distances? a frequency analysis perspective. *arXiv preprint arXiv:2106.02469*, 2021.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. Robust Large Margin Deep Neural Networks. *IEEE Transactions on Signal Processing*, 2017. doi: 10.1109/TSP.2017.2708039.

Natasa Tagasovska and David Lopez-Paz. Single-Model Uncertainties for Deep Learning. In *Advances in Neural Information Processing Systems 32*, pages 6417–6428. Curran Associates, Inc., 2019.

Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks. In *Advances in Neural Information Processing Systems 32*, pages 13888–13899. Curran Associates, Inc., 2019.

Luke Tierney, Robert E. Kass, and Joseph B. Kadane. Approximate Marginal Densities of Nonlinear Functions. *Biometrika*, 76(3):425–433, 1989. ISSN 0006-3444. doi: 10.2307/2336109. Publisher: [Oxford University Press, Biometrika Trust].

Michalis Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Artificial Intelligence and Statistics*, pages 567–574, April 2009.

Gia-Lac Tran, Edwin V. Bonilla, John Cunningham, Pietro Michiardi, and Maurizio Filippone. Calibrating Deep Convolutional Gaussian Processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1554–1563, April 2019. ISSN: 1938-7228 Section: Machine Learning.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks. In *Advances in Neural Information Processing Systems 31*, pages 6541–6550. Curran Associates, Inc., 2018.

Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pages 9690–9700. PMLR, 2020.

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

Nikhita Vedula, Nedim Lipka, Pranav Maneriker, and Srinivasan Parthasarathy. Towards Open Intent Discovery for Conversational Text. *arXiv:1904.08524 [cs]*, April 2019. arXiv: 1904.08524.

Grace Wahba. *Spline Models for Observational Data*. SIAM, September 1990. ISBN 978-0-89871-244-5.

Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning. In *International Conference on Learning Representations*, 2020.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. In *International Conference on Learning Representations*, 2018.

Florian Wenzel, Kevin Roth, Bastiaan Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the Bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10248–10259. PMLR, 13–18 Jul 2020.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016a.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Stochastic Variational Deep Kernel Learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NeurIPS'16, pages 2594–2602, USA, 2016b. Curran Associates Inc. ISBN 978-1-5108-3881-9.

Mohammad-Ali Yaghoub-Zadeh-Fard, Boualem Benatallah, Fabio Casati, Moshe Chai Barukh, and Shayan Zamanirad. User Utterance Acquisition for Training Task-Oriented Bots: A Review of Challenges, Techniques and Opportunities. *IEEE Internet Computing*, pages 1–1, 2020. ISSN 1941-0131. doi: 10.1109/MIC.2020.2978157. Conference Name: IEEE Internet Computing.

Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal Random Features. In *Advances in Neural Information Processing Systems 29*, pages 1975–1983. Curran Associates, Inc., 2016.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv:1605.07146 [cs]*, June 2017. arXiv: 1605.07146.

Yinhe Zheng, Guanyi Chen, and Minlie 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. ISSN 2329-9304. doi: 10.1109/TASLP.2020.2983593. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

Appendix 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 $g(h) \propto \cos(-\mathbf{W}_L h_i + \mathbf{b}_L)^\top \beta$. Here $\{\mathbf{W}_L, \mathbf{b}_L\}$ are frozen weights that are initialized from a Gaussian and a uniform distribution, respectively (as described in Equation (11)).

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 ((Gouk et al., 2021) which has time complexity $O(\sum_{l=1}^{L-1} D_l)$), and finally performs precision matrix update (Equation (13), 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 to compute the final hidden feature $\Phi(\mathbf{x})_{D_L \times 1}$, and then compute the posterior mean $\hat{m}_k(\mathbf{x}) = \Phi^\top \beta_k$ (time complexity $O(D_L)$) and 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 either Monte Carlo averaging or mean-field approaximation. Notice that this Monte Carlo averaging is computationally 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.

However, in applications where the inference latency is of high priority (e.g., in the on-device settings), we can reduce the computational overhead further by replacing the Monte Carlo averaging with the mean-field approximation as described in the main text (Equation 19) (Daunizeau, 2017).

A.1 Extension to Regression and Multi-class Classification

As introduced in the main text (12), for an arbitrary model posterior $p(\beta|\mathcal{D})$, its corresponding Laplace posterior is:

$$p(\beta|\mathcal{D}) \approx MVN(\hat{\beta}, \hat{\Sigma} = \hat{\mathbf{H}}^{-1}), \quad \text{where} \quad \hat{\mathbf{H}}_{(i,j)} = -\frac{\partial^2}{\partial \beta_i \partial \beta_j} \log p(\beta|\mathcal{D})|_{\beta=\hat{\beta}}.$$

Consequently, to compute the Laplace posterior for any data likelihood $p(\beta|\mathcal{D})$, it is sufficient to first compute the MAP estimate $\hat{\beta}$ as done in (14), and then derive the model Hessian $\hat{\Sigma} = \frac{\partial^2}{\partial \beta_i \partial \beta_j} \log p(\beta|\mathcal{D})|_{\beta=\hat{\beta}}$.

In the main text, we introduced the expression for $\hat{\Sigma}$ for a sigmoid cross entropy likelihood $\log p(\beta|\mathcal{D}) = -\sum_{i=1}^{n} \big[I(y_i = 1) \log p_i + I(y_i = 0) \log(1 - p_i)\big] + \frac{1}{2}||\beta||_2^2$ based on the classic result from Rasmussen and Williams (2006). In this section, we show how to apply (12) to other common

likelihood functions such as regression (using a squared loss) and K-class classification (using a softmax cross entropy loss) by deriving their Hessian.

Regression. Under squared loss, The model posterior likelihood is:

$$-\log p(\beta_{D_L\times 1}|\mathcal{D}) = \sum_{i=1}^{n}\Big[\frac{1}{2}(y_i - g_i)^2\Big] + \frac{1}{2}||\beta||^2,$$

where $g_i = \Phi_i\beta^\top$. Then the model Hessian is:

$$\hat{\mathbf{H}}_{D_L\times D_L} = \sum_{i=1}^{n}\Phi_i\Phi_i^\top + \mathbf{I}_{D_L}$$

Multi-class Classification. Under a K-class softmax cross entropy loss, the model posterior likelihood is:

$$-\log p(\beta|\mathcal{D}) = \Big[\sum_{i=1}^{n}\sum_{k=1}^{K} y_{i,k} * \log p_{i,k}\Big] + \frac{1}{2}||\beta||^2,$$

where $\beta_{D_L\times K} = \{\beta_k\}_{k=1}^K$ is the output weight for the K classes, $p_i = \text{softmax}(g_i)$ is the length-K vector of class probability, and g_i is the multi-class logits defined as $g_i = \beta_{D_L\times K}\Phi_{i,1\times D_L}^\top$.

Then, computing the Hessian with respect to $-\log p(\beta|\mathcal{D})$ yields (Kunstner et al., 2019):

$$\begin{aligned}
\hat{\mathbf{H}}_{KD_L\times KD_L} &= \sum_{i=1}^{n}[\nabla_\beta g_i]_{KD_L\times K}^\top[\text{diag}(p_i) - p_i p_i^\top]_{K\times K}[\nabla_\beta g_i]_{K\times KD_L} + \mathbf{I}_{KD_L} \\
&= [\Phi_i\otimes\mathbf{I}_K][\text{diag}(p_i) - p_i p_i^\top][\Phi_i^\top\otimes\mathbf{I}_K] + \mathbf{I}_{KD_L} \\
&= \sum_{i=1}^{n}\begin{bmatrix}\Phi_i & \dots & 0\\ & \ddots & \\ 0 & \dots & \Phi_i\end{bmatrix}_{KD_L\times K}[\text{diag}(p_i) - p_i p_i^\top]_{K\times K}\begin{bmatrix}\Phi_i^\top & \dots & 0\\ & \ddots & \\ 0 & \dots & \Phi_i^\top\end{bmatrix}_{K\times KD_L} + \mathbf{I}_{KD_L}.
\end{aligned}$$

This leads to a $KD_L\times KD_L$ matrix with $D_L\times D_L$ diagonal and off-diagonal blocks as:

$$\hat{\mathbf{H}}_{k,k'} = \begin{cases}\sum_{i=1}^{n} p_{i,k}(1-p_{i,k})\Phi_i\Phi_i^\top + \mathbf{I}_{D_L} & \text{if } k = k'\\ \sum_{i=1}^{n} -p_{i,k}p_{i,k'}\Phi_i\Phi_i^\top & \text{otherwise.}\end{cases}$$

Consequently, the Laplace posterior for each class is:

$$\beta_k \sim MVN(\hat{\beta}_k, \hat{\Sigma}_k = \hat{\mathbf{H}}_k^{-1}), \quad \text{where} \quad \hat{\mathbf{H}}_k = \sum_{i=1}^{n} p_{i,k}(1-p_{i,k})\Phi_i\Phi_i^\top + \mathbf{I}_{D_L}. \tag{21}$$

In the case where computing the class-specific covariance matrix is either infeasible (e.g., the number of output classes is simply too large) or not of interest (e.g., we are just interested in computing an scalar uncertainty statistic for an input example), one can consider computing an upper-bound of the covariance matrices $\hat{\Sigma}_k$ in (21) as:

$$\hat{\Sigma} = \hat{\mathbf{H}}^{-1} \quad \text{where} \quad \hat{\mathbf{H}} = \sum_{i=1}^{n} p_i^*(1-p_i^*)\Phi_i\Phi_i^\top + \mathbf{I}_{D_L}, \tag{22}$$

where $p_i^* = \max_k(p_{i,1}, \ldots, p_{i,k})$ is the maximum class probability. From an information theoretic perspective, quantifying model uncertainty using (22) can be understood adopting the maximum entropy distribution among the family of all class-specific distributions. To ensure computational feasibility for tasks with high-dimensional output (e.g., CIFAR100, CLINC, and ImageNet which correspond to 100, 150 and 1000 classes), we use (22) in our experiments.

A.2 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 [13].

1. **Spectral normalization** contains two hyperparameters: the number of power iterations and the upper bound for spectral norm (i.e., c in Equation (15)). 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 3). 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 (Gouk et al., 2021; Miyato et al., 2018). In general, we recommend performing a grid search for $c \in \{0.9, 0.95, 1, 2, ...\}$ 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.

2. **The Gaussian process layer** (Equation 11) contains 3 hyperparameters, which are (1) the *hidden dimension* (D_L, i.e., the number of random features), (2) the *length-scale parameter* l for the RBF kernel, and (3) the kernel amplitude σ. In the experiments, we find the model's performance to be not very sensitive to the hidden dimension or the length-scale parameter. Setting D_L in a range between $[512, 2048]$ and the length-scale $l = 2.0$ are sufficient in most cases. This is likely due to the fact that, contrary to the classic GP case without DNN, the DNN hidden mapping in SNGP is capable of adjust itself to adapt to the kernel parameters during SGD learning. Finally, as discussed in the main text, the model's calibration performance is sensitive to the kernel amplitude σ, since it functions in a manner that is similar to the temperature parameter in temperature scaling (Guo et al., 2017).

3. **The posterior covariance** (Equation 13) does not contain hyperparameter. However, if one wishes to use a moving average estimator for the covariance matrix, i.e.,

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

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

Then there will be two additional hyperparameters: the ridge factor s and the discount factor m. 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. 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 use the exact update formula (13) in the CIFAR and Genomics experiments, and use the moving average update as described above for larger tasks such as ImageNet.

In summary, when applying SNGP method to a new dataset it is sufficient to sweep the spectral norm bound c and kernel amplitude parameter σ on a holdout validation dataset, and fix all other parameters to their default values (Table 9). We report the values (c, σ) for each experiment in Section C.1.

Spectral Normalization		Gaussian Process Layer		Covariance Matrix	
Power Iteration	1	Hidden Dimension	2048 (BERT) / 1024 (Otherwise)	(Optional)	
Spectral Norm Bound	Model dependent	Length-scale Parameter	2.0	**Ridge Factor**	0.001
		Kernel amplitude	Task dependent	**Discount Factor**	0.999

Table 9: Hyperparameters of SNGP used in the experiments, where important hyperparameters are highlighted in bold.

As an aside, 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. Ablation studies revealed that the model performance is not sensitive to these changes.

Appendix 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 $p^*(\mathbf{x} \in \mathcal{X}_{IND})$, and given a model $p(y|\mathbf{x}, \mathbf{x} \in \mathcal{X}_{IND})$ 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 $p^*(\mathbf{x} \in \mathcal{X}_{\texttt{IND}})$ 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 6 (Minimax Solution to Uncertainty Risk Minimization)**.**
Given:

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

(b) $p^*(\mathbf{x} \in \mathcal{X}_{\texttt{IND}})$ *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:

$$p(y|\mathbf{x}) = p(y|\mathbf{x}, \mathbf{x} \in \mathcal{X}_{\texttt{IND}}) \times p^*(\mathbf{x} \in \mathcal{X}_{\texttt{IND}}) + p_{\texttt{uniform}}(y|\mathbf{x}, \mathbf{x} \notin \mathcal{X}_{\texttt{IND}}) \times p^*(\mathbf{x} \notin \mathcal{X}_{\texttt{IND}}) \tag{23}$$

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

As discussed in Section 2.1, the solution (23) 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* (Parry et al., 2012):

$$s(p, p^*|\mathbf{x}) = \sum_{k=1}^{K} \Big\{ [p^*(y_k|\mathbf{x}) - p(y_k|\mathbf{x})] \psi'(p^*(y_k|\mathbf{x})) - \psi(p^*(y_k|\mathbf{x})) \Big\} \tag{24}$$

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 (23) for the Bregman score. The proof relies on the following key lemma:

Lemma 7 ($p_{\texttt{uniform}}$ is Optimal for Minimax Bregman Score in $\mathbf{x} \notin \mathcal{X}_{IND}$)**.**
Consider the Bregman score in (24). At a location $\mathbf{x} \notin \mathcal{X}_{IND}$ *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*

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

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

The proof for Lemma 7 is in Section E.2. It is worth noting that Lemma 7 only holds for a *strictly* proper scoring rule (Gneiting et al., 2007). 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 6: *Proof.* Denote $\mathcal{X}_{\texttt{OOD}} = \mathcal{X} / \mathcal{X}_{\texttt{IND}}$. Decompose the overall Bregman risk by domain:

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

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

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 $p \in \mathcal{P}$. 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^*:

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

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:

$$\begin{aligned}\inf_p \sup_{p^*} S(p,p^*) &= \inf_p \Big[\sup_{p^*} \left[S_{\texttt{IND}}(p,p^*)\right] * p^*(\mathbf{x} \in \mathcal{X}_{\texttt{IND}}) + \sup_{p^*} \left[S_{\texttt{OOD}}(p,p^*)\right] * p^*(\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}) \Big] \\ &= \inf_p \sup_{p^*} \left[S_{\texttt{IND}}(p,p^*)\right] * p^*(\mathbf{x} \in \mathcal{X}_{\texttt{IND}}) + \inf_p \sup_{p^*} \left[S_{\texttt{OOD}}(p,p^*)\right] * p^*(\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}), \quad (25)\end{aligned}$$

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

Therefore, to show that (23) is the optimal and unique solution to (25), we only need to show $p_{\texttt{uniform}}$ is the optimal and unique solution to $\inf_p \sup_{p^*} \left[S_{\texttt{OOD}}(p,p^*)\right]$. To this end, notice that for a given p:

$$\sup_{p^* \in \mathcal{P}^*} \left[S_{\texttt{OOD}}(p,p^*)\right] = \int_{\mathcal{X}_{\texttt{OOD}}} \sup_{p^*} [s(p,p^*|\mathbf{x})] p(\mathbf{x}|\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}) d\mathbf{x}, \quad (26)$$

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 $\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}$). Furthermore, there exists p that minimize $\sup_{p^*} s(p,p^*|\mathbf{x})$ at every location of $\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}$, then it minimizes the integral (Berger, 1985). By Lemma 7, such p exists and is unique, i.e.:

$$p_{\texttt{uniform}} = \operatorname*{arginf}_{p \in \mathcal{P}} \sup_{p^* \in \mathcal{P}^*} S_{\texttt{OOD}}(p,p^*).$$

In conclusion, we have shown that $p_{\texttt{uniform}}$ 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 (25) is (23). ■

Appendix C. Experiment Details

C.1 Model Configuration

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 (Zagoruyko and Komodakis, 2017). 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.

For CLINC OOS intent understanding data, we pre-tokenized the sentences using the standard BERT tokenizer[14] 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[15]. 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

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

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

classification token. We do not apply 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.

For Genomics sequence data, we consider a 1D CNN model following the prior work (Ren et al., 2019). Specifically, the model is composed by one convolutional layer of 2,000 filters of length 20, one max-pooling layer, one dense layer of 2,000 units, and a final dense layer with softmax activation for predicting class probabilities. To build the SNGP model, we add spectral normalization to both the convolutional layer and the dense layer, and we replace the last layer with Gaussian process layer. For MC Dropout, we add filter-wise dropout before the convolutional layer with dropout rate 0.1. For Deep Ensemble, we ensemble 5 models trained based on random initialization of network parameters and random shuffling of training inputs. The model is trained using the batch size 128, the learning rate 1e-4, and Adam optimizer. The training step is set for 1 million, but we choose the best step when validation loss is at the lowest value. Due to the large size of test OOD dataset, we randomly select 100,000 OOD samples to pair with the same number of in-distribution samples.

For each of the experiment, we fix the SNGP hyperparameters to their recommended values as in Table 9, and sweep the Spectral Norm Bound m and the kernel amplitude σ with respect to the negative log likelihood on the validation data. The final values found for each experiment is summarized at Table 10. As shown, the optimal value for spectral norm bound usually depends on the layer type (1D Convolution v.s. 2D Convolution v.s. Dense), while the optimal value of kernel amplitude seems to be sensitive to the data modality, the model type, and also the type of covariance estimator (i.e., exact estimator v.s. moving average estimator) being used (also see earlier discussion in Section A.2).

Task	Spectral Norm Bound	Kernel Amplitude
CIFAR-10	6.0	20 / 30 (+Ensemble) / 7.5 (+AugMix)
CIFAR-100	6.0	7.5 / 5.0 (+Ensemble) / 1 (+AugMix)
ImageNet	6.0	1.
CLINC	0.95	0.1
Genomics	10.0	0.001

Table 10: Hyperparameters of SNGP used in the experiments.

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.

C.2 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 (Hendrycks and Dietterich, 2018).

We assess the model's calibration performance using the empirical estimate of ECE: $\hat{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$ which estimates the difference in model's accuracy and confidence by partitioning model prediction into M bins $\{B_m\}_{m=1}^{M}$ (Guo et al., 2017). In this work, we choose $M = 15$.

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 (Netzer et al., 2011), and a more difficult OOD task by testing CIFAR-10's ability in detecting samples from the CIFAR-100 dataset, and vice versa. For CIFAR dataset, since the input data is normalized by the sample mean and variance, we perform the same normalization for the test OOD data. We evaluate the OOD uncertainty scores for the inputs in in-distribution test set, and the inputs in OOD test set, and we use AUROC to measure the separation between the two sets based on the OOD uncertainty score.

For all models, we compute the confidence score (or reverse equivalently OOD uncertainty score) for an input $\mathbf{x}$ as the Maximum of Softmax Probability (MSP) (Hendrycks and Gimpel, 2017), which is $\max_k(\text{softmax}(g_k(\mathbf{x}))), k = 1, 2, , K$, given logits for K classes $\{g_k(\mathbf{x})\}_{k=1}^K$. The results presented in the main text are based on MSP. We study other commonly used uncertainty scores, including the so-called *Dempster-Shafer metric* (Sensoy et al., 2018b), the Mahalanobis distance Lee et al. (2018b), and the relative Mahalanobis distance Ren et al. (2021), to compare with the MSP.

The *Dempster-Shafer metric* computes its uncertainty for a test example $\mathbf{x}_i$ as:

$$u(\mathbf{x}) = \frac{K}{K + \sum_{k=1}^K \exp\left(h_k(\mathbf{x})\right)}, \tag{27}$$

and the confidence score is defined as $\text{score}_{\text{DS}}(\mathbf{x}) = 1 - u(\mathbf{x})$, 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 with respect to the magnitude of the logits, similar to the energy-based OOD score in Liu et al. (2020b). On the other hand, the maximum probability $p_{\texttt{max}} = \max_k \exp(g_k) / \sum_{k=1}^K \exp\left(g_k(\mathbf{x}_i)\right)$ does not take advantage of this information since it normalizes over the exponentiated logits.

Mahalanobis distance (MD) method was proposed by Lee et al. (2018b) to fit a Gaussian distribution to the class-conditional embeddings and use the Mahalanobis distance for OOD detection. Let $h(\mathbf{x})$ denote the embedding (e.g. the penultimate layer before computing the logits) of an input $\mathbf{x}$. We fit a Gaussian distribution to the embeddings of the training data, computing per-class mean $\mu_k = \frac{1}{N_k} \sum_{i:y_j=k} h(\mathbf{x}_j)$ and a shared covariance matrix $\Sigma = \frac{1}{N} \sum_{k=1}^K \sum_{i:y_j=k} \left(h(\mathbf{x}_j) - \mu_k\right)\left(h(\mathbf{x}_j) - \mu_k\right)^\top$. The confidence score (negative of the Mahalanobis distance) is then computed as: $\text{score}_{\text{Maha}}(\mathbf{x}) = -\min_k \left(\left(h(\mathbf{x}) - \mu_k\right)\Sigma^{-1}\left(h(\mathbf{x}) - \mu_k\right)^\top\right) = -\min_k \left(\text{MD}_k(\mathbf{x})\right)$.

Relative Mahalanobis distance (RMD) was proposed by Ren et al. (2021) to fix the degradation of the raw Mahalanobis distance in near-OOD scenarios. The confidence score $\text{score}_{\text{RMaha}}(\mathbf{x}) = -\min_k \left(\text{MD}_k(\mathbf{x}) - \text{MD}_0(\mathbf{x})\right)$, where $\text{MD}_0(\mathbf{x})$ is the Mahalanobis distance to a background Gaussian distribution fitted to the entire training data not considering the class labels.

Table 11 shows the OOD performance for CIFAR dataset based on the above four different confidence scores. First, it shows that Dempster Shafer score in general has better performance than MSP. Second, we noticed that Mahalanobis distance has much worse performance on GP based models, while Relative Mahalanobis distance corrects for the degradation. For example, for CIFAR-10 vs. CIFAR-100 task, Mahalanobis distance based on SNGP model's embeddings has only 0.742 AUROC, while other OOD methods achieve around 0.90 AUROC based on SNGP model, including Relative Mahalanobis distance. This suggests that the GP based models probably preserved background features which confounds the raw Mahalanobis distance score, and the relative

Mahalanobis distance corrects for the background features and fix the performance. It is interesting to see that the Relative Mahalanobis distance has the best performance for the most challenging near-OOD task CIFAR-100 vs. CIFAR-10 for all models.

We also additionally evaluate another two simple OOD datasets, random Gaussian noise and texture dataset (i.e. DTD), used in the prior work (Hendrycks et al., 2018; Liang et al., 2018; Liu et al., 2020b). The results are included in Table 11. As shown, SNGP provides the best performance when using the default MSP method for OOD detection: 0.999 AUROC for detecting random Gaussian noise, and 0.959 AUROC for detecting DTD. When combined with more advanced OOD detection signals, such as Dempster Shafer, the performance can be further improved to 1.000 for random Gaussian and 0.988 for DTD. Furthermore, for the uncertainty metrics whose performance hinders on the qualities of both the hidden representation and the last-layer (i.e., MSP and Dempster-Shafer), the SNGP model attains the strongest performance when compared to its ablated counterparts. Finally, we observed some vanilla methods (e.g., Mahalanobis distance based on vanilla DNN) also achieves strong performance for these simple datasets. However, this advantage starts to break down on the more difficult datasets (e.g., CIFAR100 v.s. SVHN).

In conclusion, when comparing to the baseline approaches, SNGP provides the best out-of-box performance when using the default MSP method for OOD detection (Section 6.2.1), and it can be combined with more advanced OOD detection signals to further improve performance.

	MSP	Dempster Shafer	Mahalanobis	Relative Mahalanobis
		Near-OOD		
		CIFAR-10 vs. CIFAR-100		
DNN	0.893 ± 0.003	**0.894 ± 0.004**	0.891 ± 0.002	0.890 ± 0.002
DNN-SN	**0.893 ± 0.002**	**0.893 ± 0.003**	0.892 ± 0.002	0.889 ± 0.002
DNN-GP	**0.903 ± 0.003**	0.902 ± 0.005	0.767 ± 0.016	0.899 ± 0.002
SNGP	**0.905 ± 0.002**	0.903 ± 0.005	0.742 ± 0.011	0.899 ± 0.002
		CIFAR-100 vs. CIFAR-10		
DNN	0.795 ± 0.001	0.804 ± 0.002	0.780 ± 0.003	**0.809 ± 0.002**
DNN-SN	0.793 ± 0.003	0.801 ± 0.002	0.785 ± 0.003	**0.809 ± 0.002**
DNN-GP	0.797 ± 0.001	0.782 ± 0.004	0.559 ± 0.005	**0.804 ± 0.003**
SNGP	0.798 ± 0.001	0.784 ± 0.003	0.576 ± 0.006	**0.804 ± 0.002**
		Far-OOD		
		CIFAR-10 vs. SVHN		
DNN	0.946 ± 0.005	0.954 ± 0.008	**0.959 ± 0.006**	0.924 ± 0.011
DNN-SN	0.945 ± 0.005	0.958 ± 0.006	**0.960 ± 0.005**	0.929 ± 0.010
DNN-GP	0.964 ± 0.006	**0.984 ± 0.004**	0.914 ± 0.012	0.932 ± 0.005
SNGP	0.960 ± 0.004	**0.982 ± 0.004**	0.915 ± 0.014	0.931 ± 0.016
		CIFAR-100 vs. SVHN		
DNN	0.799 ± 0.020	**0.841 ± 0.019**	0.764 ± 0.020	0.755 ± 0.022
DNN-SN	0.798 ± 0.022	**0.832 ± 0.021**	0.752 ± 0.025	0.727 ± 0.025
DNN-GP	0.835 ± 0.021	**0.887 ± 0.025**	0.550 ± 0.073	0.724 ± 0.033
SNGP	0.846 ± 0.019	**0.894 ± 0.020**	0.533 ± 0.039	0.712 ± 0.021
		CIFAR-10 vs Random Gaussian		
DNN	0.981 ± 0.014	0.990 ± 0.008	**1.000 ± 0.000**	0.990 ± 0.007
DNN-SN	0.976 ± 0.013	0.996 ± 0.003	**1.000 ± 0.000**	0.983 ± 0.013
DNN-GP	0.997 ± 0.003	**1.000 ± 0.000**	0.999 ± 0.000	0.968 ± 0.032
SNGP	0.999 ± 0.002	**1.000 ± 0.000**	0.999 ± 0.000	0.982 ± 0.013
		CIFAR-10 vs DTD		
DNN	0.902 ± 0.020	0.915 ± 0.011	**0.994 ± 0.001**	0.906 ± 0.022
DNN-SN	0.909 ± 0.016	0.909 ± 0.024	**0.994 ± 0.001**	0.901 ± 0.028
DNN-GP	0.946 ± 0.008	0.982 ± 0.008	**0.987 ± 0.002**	0.925 ± 0.015
SNGP	0.959 ± 0.009	**0.988 ± 0.004**	0.985 ± 0.003	0.938 ± 0.004

Table 11: OOD detection performance (AUROC) based on different confidence scores, including MSP, Dempster Shafer, Mahalanobis distance, and Relative Mahalanobis distance. The highest values in each row are highlighted.

C.3 Theoretical Convergence to Optimal Behaviour

In this section, we discuss the asymptotic behaviour of the SNGP algorithm on OOD datapoints far from the training distribution, and show how the predictive distribution converges to the optimal distribution suggested by Equation (4). We formalize this in the following proposition:

Proposition 8 (SNGP Convergence to Optimal Behaviour)**.**
Given:

(a) A distance-preserving hidden mapping $h : \mathcal{X} \to \mathcal{H}$, so that

$$L_1 * d_X(\mathbf{x}, \mathbf{x}^*) \leq \|h(\mathbf{x}) - h(\mathbf{x}^*)\|_2^2 \leq L_2 * d_X(\mathbf{x}, \mathbf{x}^*),$$

for $0 < L_1 < L_2$ two positive constants.

(a) A dual form formulation of the SNGP model assuming a Laplace approximation over the posterior, with posterior mean and variance given by

$$m(\mathbf{x}) = \mathbf{k}^*(\mathbf{x})^\top (\mathbf{K} + \tau \mathbf{I})^{-1} \mathbf{y}$$
$$v(\mathbf{x}) = \tau^{-1} * \left[k(\mathbf{x}, \mathbf{x}) - \mathbf{k}^*(\mathbf{x})^\top (\mathbf{K} + \tau \mathbf{I})^{-1} \mathbf{k}^*(\mathbf{x}) \right]$$

where $k(\mathbf{x}, \mathbf{x})_{1\times 1} = \phi(\mathbf{x})^\top \phi(\mathbf{x}), \mathbf{k}^*(\mathbf{x})_{N\times 1} = \phi(\mathbf{x})^\top \Phi^\top$ and $\mathbf{K}_{N\times N} = \Phi\Phi^\top$ are kernel matrices approximating those under the RBF kernel $k(\mathbf{x}, \mathbf{x}') \propto \exp\left(-\|h(\mathbf{x}) - h(\mathbf{x}')\|_2^2\right)$.

(b) The predictive distribution of the SNGP model for classification $p(y|\mathbf{x})$ given by

$$p(y \mid \mathbf{x}) = \int_{g \sim N(m(\mathbf{x}), v(\mathbf{x}))} \text{softmax}(g) dg$$

then as test points $\mathbf{x}^*$ tend away from the training manifold $\mathbf{x}$ (i.e $d_X(\mathbf{x}, \mathbf{x}^*) \to \infty$), the limit of the predictive distribution $p(y|\mathbf{x})$ is either exact equal to the optimal distribution $p_{uniform}$ as given by (4) (under the mean-field approximation), or can be closely approximated by $p_{\texttt{uniform}}$ (under the Monte Carlo approximation).

The proof for Proposition 8 follows: *Proof.* Given condition (a), h is a bi-Lipschitz, distance-preserving function, and therefore we can write down the asymptotic convergence of the RBF kernel $k(\mathbf{x}, \mathbf{x}^*)$ as follows

$$\begin{aligned}\lim_{d_X(\mathbf{x}, \mathbf{x}^*) \to \infty} k(\mathbf{x}, \mathbf{x}^*) &= \lim_{d_X(\mathbf{x}, \mathbf{x}^*) \to \infty} C \exp\left(-\|h(\mathbf{x}) - h(\mathbf{x}^*)\|_2^2\right) \\ &\leq \lim_{d_X(\mathbf{x}, \mathbf{x}^*) \to \infty} C \exp\left(-L_1 * d_X(\mathbf{x}, \mathbf{x}^*)\right) = 0,\end{aligned}$$

where L_1 and C are constants, and the convergence is guaranteed by the sandwich theorem, since by the distance preservation property, $\|h(\mathbf{x}) - h(\mathbf{x}^*)\|_2^2 \geq L_1 \times d_X(\mathbf{x}, \mathbf{x}^*)$.

Leveraging the above result, we can now reason about the asymptotic behaviour of the posterior moments $(m(\mathbf{x}^*), v(\mathbf{x}^*))$:

$$\lim_{d_X(\mathbf{x}, \mathbf{x}^*) \to \infty} m(\mathbf{x}^*) = \lim_{d_X(\mathbf{x}, \mathbf{x}^*) \to \infty} \mathbf{k}^*(\mathbf{x}^*)^\top (\mathbf{K} + \tau \mathbf{I})^{-1} \mathbf{y} = 0$$

$$\begin{aligned}\lim_{d_X(\mathbf{x}, \mathbf{x}^*) \to \infty} v(\mathbf{x}^*) &= \lim_{d_X(\mathbf{x}, \mathbf{x}^*) \to \infty} \tau^{-1} * \left[k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^*(\mathbf{x}^*)^\top (\mathbf{K} + \tau \mathbf{I})^{-1} \mathbf{k}^*(\mathbf{x}) \right] \\ &= \tau^{-1} * k(\mathbf{x}^*, \mathbf{x}^*) = c\end{aligned}$$

for some data-independent constant c.

As a result, we can now consider the asymptotic behaviour of the predictive distribution $p(y|\mathbf{x})$ by leveraging the asymptotic behaviour of the moments.

Mean-field Approximation. First notice that by mean-field theorem, the sigmoid and softmax transformation of Gaussian is approximated as below (Bishop, 2011; Lu et al., 2020):

$$p(y \mid \mathbf{x}) = \sigma\left(\frac{m(\mathbf{x})}{\sqrt{1+\lambda * v(\mathbf{x})}}\right).$$

Then, since $m(\mathbf{x}^*) \to \mathbf{0}$ and $v(\mathbf{x}^*) \to c$ as $d(\mathbf{x}, \mathbf{x}^*) \to \infty$, the predictive distribution $p(y|\mathbf{x}^*)$ converges to the uniform distribution:

$$\lim_{d_X(\mathbf{x},\mathbf{x}^*)\to\infty} p(y \mid \mathbf{x}^*) = \sigma\left(\frac{m(\mathbf{x}^*)}{\sqrt{1+\lambda * v(\mathbf{x}^*)}}\right) = \sigma(0) = p_{\texttt{uniform}}, \tag{28}$$

where the second equality follows from the continuous mapping theorem.

Monte Carlo Approximation. Notice that since the Monte Carlo approximation of $p(y \mid \mathbf{x}^*)$ is an unbiased estimator of the true integral, it is sufficient to investigate the asymptotic behavior of the integral:

$$\begin{aligned}
\lim_{d_X(\mathbf{x},\mathbf{x}^*)\to\infty} p(y \mid \mathbf{x}^*) &= \lim_{d_X(\mathbf{x},\mathbf{x}^*)\to\infty} \int_{g\sim N(m(\mathbf{x}^*),v(\mathbf{x}^*))} \sigma(g) dg \\
&= \int \lim_{d_X(\mathbf{x},\mathbf{x}^*)\to\infty} \sigma(g) * f_N(g|m(\mathbf{x}^*), v(\mathbf{x}^*)) dg \\
&= \int \sigma(g) * f_N(g|0,c) dg
\end{aligned} \tag{29}$$

where $f_N(g|m,v)$ denotes the probability density function of a Gaussian-distributed random variable g with mean m and standard deviation v. In the above, the second equality (i.e., switching of limit and integral) follows by the bounded convergence theorem since the integrand $\sigma(g) * f_N(g)$ is a bounded function. Intuitively, Equation (29) is again a Gaussian-softmax integral whose expectation can be approximated closely via the mean-field approximation (Bishop, 2011; Lu et al., 2020). That is, by following the same line of argument as above, we now that the limiting distribution $\lim_{d_X(\mathbf{x},\mathbf{x}^*)\to\infty} p(y \mid \mathbf{x}^*)$ of the Monte Carlo approximation can be closely approximated by a uniform distribution, i.e.,

$$\lim_{d_X(\mathbf{x},\mathbf{x}^*)\to\infty} p(y \mid \mathbf{x}^*) = E_{g\sim N(0,c)}\left[\sigma(g)\right] \approx \sigma(0) = p_{\texttt{uniform}}.$$

Alternatively, we can argue about convergence of Equation (29) by appealing to the multivariate continuous mapping theorem. Specifically, since $(m(\mathbf{x}^*), v(\mathbf{x}^*)) \to (\mathbf{0}, c)$, we have $g \to g' \overset{d}{\sim} N(\mathbf{0}, c\mathbf{I})$ and $\sigma(g) \overset{d}{\sim} \sigma(g')$, where $\overset{d}{\sim}$ denotes the convergence in distribution. Consequently, $E(g) \to E(g')$. Now, for a K-dimensional g', consider the j^{th} coordinate of $log\sigma(g')$:

$$\sigma(g')_j = \frac{exp(g'_j)}{\sum_{k=1}^{K} exp(g'_k)},$$

As shown, since $g'_k \overset{i.i.d.}{\sim} N(0,c)$, we expect $E(\sigma(g')_j)$ to be equal $\forall j \in \{1,\ldots,K\}$. Consequently, denoting $E(\sigma(g')_k) = p' \; \forall k$ and notice $\sigma(g')$ sum to 1, we have:

$$\begin{aligned} E(\sigma(g')_k) &= p' \quad \forall k = 1,\ldots,K, \\ \sum_k E(\sigma(g')_k) &= 1, \end{aligned}$$

which leads to the unique solution of $p' = \frac{1}{K}$, i.e., $E(\sigma(g')) = p_{\texttt{uniform}}$, which again shows:

$$\lim_{d_X(\mathbf{x},\mathbf{x}^*)\to\infty} p(y \mid \mathbf{x}^*) = E(\sigma(g)) \to E(\sigma(g')) = p_{\texttt{uniform}}.$$

■

Appendix D. An Example Formalization of "Semantic Distance".

In this section, we develop an example formalization of the intuition notion of *semantic distance* using languages from the metric embedding theory (Abraham et al., 2011; Matoušek, 2013; Chennuru Vankadara and von Luxburg, 2018). Here, our goal is to supply an example formalization of this often intuitive notion, with then goal of facilitating a rigorous understanding of Section 8.1 (Abraham et al., 2011; Matoušek, 2013; Chennuru Vankadara and von Luxburg, 2018). Indeed, the term "semantic distance" has a long history in the literature, and it is out of the scope of this work to provide an authoritative, all-encompassing mathematical construction that unifies its diverse usages across many fields such as manifold learning, representation learning, natural language processing, and cognitivle psychology (Tenenbaum et al., 2000; Mohammad and Hirst, 2006; Deselaers and Ferrari, 2011; Hashimoto et al., 2016; Higgins et al., 2018; Khemakhem et al., 2020; Chandrasekaran and Mago, 2021; Reilly et al., 2022).

Defining "semantic space" in terms of a metric space. We consider the setting where all examples $\mathbf{x}$ within a problem domain $\mathcal{X}$ can be sufficiently described by a large collection of attributes, with each attribute being either discrete (e.g., types of entities that appeared in a image and their relationships) or continuous (e.g., color intensity or camera angle). The number of attributes D is finite but allowed to be very large. Furthermore, we assume different attribute impacts the semantic similarity between examples to a different degree, so that the variations along only a subset of attributes constitute a meaningful difference between the examples.

We can formalize the above intuition terms of metric space (Rudin et al., 1976). Specifically, we can assume the examples $\mathbf{x}$ has a semantic representation $\mathbf{x}$ that reside in a *semantic space* $\mathcal{X}$, which is a P-dimensional metric space with its dimensions correspond to the discrete and continuous attributes. Specifically, $\mathcal{X}$ can be expressed as a product of D *attribute subspaces* $\{\mathcal{A}_j\}_{j=1}^{D}$:

$$\mathcal{X} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_D, \tag{30}$$

where each attribute subspace $(\mathcal{A}_j, d_j)$ is a metric space that corresponds to a continuous or discrete attribute, and is equipped with a well-defined metric d_j. For example, $\mathcal{A}_j$ can represent a *continuous* attribute such as the color, which implies $\mathcal{A}_j = \mathbb{R}^3$ and d_j is the standard Euclidean metric for the RGB space. $\mathcal{A}_j$ can also represent a *discrete* attribute such as entity types, which implies $\mathcal{A}_j$ is a discrete space that is supported on a large amount of candidate entities (e.g., from a knowledge graph), and the metric between entities d_j can be defined by a certain graph metric with respect to a pre-established concept hierarchy (e.g., WordNet) (Chandrasekaran and Mago, 2021). As a result, we can write the semantic representation of an example $\mathbf{x} \in S$ as $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_D] \in S$ where $\mathbf{x}_j \in \mathcal{A}_j$, and the differences between two examples $(\mathbf{x}, \mathbf{x}')$ in the j^{th} attribute can be described as $d_j(\mathbf{x}_j, \mathbf{x}'_j)$.

Consequently, a "semantic distance" can be defined in terms of a metric function $d_X : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for the product space $\mathcal{X}$, so that $(\mathcal{X}, d_X)$ is a valid metric space. To this end, a proper choice of d_X should satisfy the metric axioms (positivity, symmetry, and triangle inequality) while aligns well with the intuitive notion of "semantic similarity" between examples. For example, consider the below definition of d_X:

$$d_X(\mathbf{x}, \mathbf{x}') = \sum_j w_j d_j(\mathbf{x}_j, \mathbf{x}'_j) + \sum_j \sum_k w_{jk} d_{j,k}(\mathbf{x}_j, \mathbf{x}'_j) d_k(\mathbf{x}_k, \mathbf{x}'_k), \tag{31}$$

i.e., d_X is a weighted sum of attribute-specific metrics d_j and their pairwise products[16]. Here $\mathbf{w} = [w_1, w_2, \ldots, w_D, w_{11}, w_{12}, \ldots, w_{DD}]$ is the set of positive weights that sum to 1. We see the definition of semantic distance d_X in Equation (31) is flexible, as it not only allows the domain expert to define what constitutes a "semantically-meaningful difference" by assigning different weights among attributes, but also allows the attributes to interact to define the overall metric. We also see that d_X is a valid metric, as the positive $\mathbf{w}$ guarantees positivity, and triangle inequality is closed under the summation and multiplication of positive terms[17].

From semantic space to data space via metric embedding. So far, we have described a formal definition of *semantic space* $\mathcal{X}$ and the associated *semantic distance* d_X in terms of a metric space $(\mathcal{X}, d_X)$. Further, $(\mathcal{X}, d_X)$ is constructed as the product of a collection of attribute metric spaces $\{(\mathcal{A}_j, d_j)\}_{j=1}^D$, so that the coordinates of $\mathcal{X}$ adopt meaningful interpretations in terms of well-defined attributes. In practice, we often do not have direct access to $(\mathcal{X}, d_X)$, and can only work with its surface-form data representation $(\mathcal{S}, d_S)$, where the coordinates of $\mathcal{S}$ and the surface-form distance d_S is less meaningful. However, the elements in $(\mathcal{X}, d_X)$ still has a unique (i.e., one-to-one) correspondence with respect to $\mathbf{x}$'s in the semantic space. For example, an image is often represented as a tensor of dimension (W, H, C) (i.e., $\mathcal{X} = \mathbb{R}^{W \times H \times C}$). However, by visually inspecting this tensor, a human can still discern the various attributes underlying the image, implying that an inverse mapping exists there exists from the surface-form data space $\mathcal{S}$ to the semantic space $\mathcal{X}$.

Formally, this means we can define an *embedding function* $\psi : \mathcal{X} \to \mathcal{S}$, which is a mapping from the semantic space $(\mathcal{X}, d_X)$ to the data space $(\mathcal{S}, d_S)$. As a result, every example $\mathbf{x} \in \mathcal{X}$ adopts a surface-form representation $\mathbf{s} = \psi(\mathbf{x}) \in \mathcal{S}$, which means when measuring based on the data space, the distance between a pair of examples $(\mathbf{x}, \mathbf{x}')$ becomes:

$$d_S(\mathbf{x}, \mathbf{x}') := d_S(\mathbf{s}, \mathbf{s}') = d_S(\psi(\mathbf{x}), \psi(\mathbf{x}')).$$

Consequently, the *distortion* (introduced in the Section 8.1 in the discussion section) between the semantic distance d_X and the surface-form distance d_S can be expressed as:

$$\rho(\mathbf{x}, \mathbf{x}') = \frac{d_X(\mathbf{x}, \mathbf{x}')}{d_S(\mathbf{s}, \mathbf{s}')} = \frac{d_X(\mathbf{x}, \mathbf{x}')}{d_S(\psi(\mathbf{x}), \psi(\mathbf{x}'))}.$$

As shown, the distortion is induced by both the embedding function ψ and the difference in the metrics (i.e., d_X *v.s.* d_S). From this perspective, when learning a neural network model $logit(\mathbf{x}) = h(\mathbf{x})^\top \beta$ based on the observed data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, a semantic *distance preserving* representation $h : \mathcal{X} \to \mathcal{H}$ should ideally satisfying the *bi-Lipschitz* condition (Equation (7)) with respect to d_X:

$$L_1 \times d_X(\mathbf{x}_1, \mathbf{x}_2) \leq ||h(\mathbf{x}_1) - h(\mathbf{x}_2)||_H \leq L_2 \times d_X(\mathbf{x}_1, \mathbf{x}_2),$$

so that $||.||_H$ has a reasonable correspondence with the semantic distance.

[16] It is also possible to define d_X with even higher-order products, e.g., $d_i(\mathbf{x}_i, \mathbf{x}_i') d_j(\mathbf{x}_j, \mathbf{x}_j') d_k(\mathbf{x}_k, \mathbf{x}_k')$, which we don't explore here for the simplicity of exposition.

[17] That is, if the attribute-specific metrics satisfy positivity and triangle inequality, then their sum and product also satisfy triangle inequality. For example, $d_i(\mathbf{x}_i, \mathbf{x}_i'') + d_j(\mathbf{x}_j, \mathbf{x}_j'') \leq (d_i(\mathbf{x}_i, \mathbf{x}_i') + d_i(\mathbf{x}_i', \mathbf{x}_i'')) + (d_j(\mathbf{x}_j, \mathbf{x}_j') + d_j(\mathbf{x}_j', \mathbf{x}_j'')) = (d_i(\mathbf{x}_i, \mathbf{x}_i') + d_j(\mathbf{x}_i, \mathbf{x}_i')) + (d_i(\mathbf{x}_i', \mathbf{x}_i'') + d_j(\mathbf{x}_j', \mathbf{x}_j''))$. and $d_i(\mathbf{x}_i, \mathbf{x}_i'') d_j(\mathbf{x}_j, \mathbf{x}_j'') = (d_i(\mathbf{x}_i, \mathbf{x}_i') + d_i(\mathbf{x}_i', \mathbf{x}_i''))(d_j(\mathbf{x}_j, \mathbf{x}_j') + d_j(\mathbf{x}_j', \mathbf{x}_j'')) \leq d_i(\mathbf{x}_i, \mathbf{x}_i') d_j(\mathbf{x}_j, \mathbf{x}_j') + d_i(\mathbf{x}_i', \mathbf{x}_i'') d_j(\mathbf{x}_j', \mathbf{x}_j'')$.

Appendix E. Proof

E.1 Proof of Proposition 3

The proof for Proposition 3 is an adaptation of the classic result of (Bartlett et al., 2018) 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 $h : \mathcal{X} \to \mathcal{H}$, denote $||h|| = \sup\left\{ \frac{||f(\mathbf{x})||_H}{||\mathbf{x}||_X} \text{ for } \mathbf{x} \in \mathcal{X}, ||\mathbf{x}|| > 0 \right\}$. Also denote the Lipschitz seminorm for a function h as:

$$||h||_L = \sup\left\{ \frac{||h(\mathbf{x}) - h(\mathbf{x}')||_H}{d_X(\mathbf{x}, \mathbf{x}')} \quad \text{for} \quad \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \mathbf{x} \neq \mathbf{x}' \right\} \tag{32}$$

It is worth noting that by the above definitions, for two functions $(\mathbf{x}' - \mathbf{x}) : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ and $(h(\mathbf{x}) - h(\mathbf{x}')) : \mathcal{X} \times \mathcal{X} \to \mathcal{H}$ who shares the same input space, the Lipschitz inequality can be expressed using the $||.||$ norm, i.e., $||h(\mathbf{x}) - h(\mathbf{x}')||_H \leq \alpha d_X(\mathbf{x}, \mathbf{x}')$ 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 3 by first showing:

$$(1 - \alpha)||\mathbf{x} - \mathbf{x}'|| \leq ||h_l(\mathbf{x}) - h_l(\mathbf{x}')|| \leq (1 + \alpha)||\mathbf{x} - \mathbf{x}'||, \tag{33}$$

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:

$$(1 - \alpha)||\mathbf{x} - \mathbf{x}'|| \leq ||h_l(\mathbf{x}) - h_l(\mathbf{x}')||. \tag{34}$$

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 (34)-(35), we have shown (33), which also implies:

$$(1 - \alpha)||\mathbf{x} - \mathbf{x}'||_X \leq ||h_l(\mathbf{x}) - h_l(\mathbf{x}')||_H \leq (1 + \alpha)||\mathbf{x} - \mathbf{x}'||_X \tag{35}$$

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:

$$(1 - \alpha)^L ||\mathbf{x} - \mathbf{x}'||_X \leq ||h(\mathbf{x}) - h(\mathbf{x}')||_H \leq (1 + \alpha)^L ||\mathbf{x} - \mathbf{x}'||_X \tag{36}$$

Denoting $L_1 = (1 - \alpha)^L$ and $L_2 = (1 + \alpha)^L$, we have arrived at expression in Proposition 3. ■

E.2 Proof of Lemma 7

Proof Sketch. 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 9 (Maximum Entropy Theorem for General Loss (Grünwald and Dawid, 2004))**.** *Let $\mathcal{P}$ 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 $H(p|\mathbf{x}) = \inf_{p^* \in \mathcal{P}^*} s(p, p^*|\mathbf{x})$. Assume below conditions on $H(p|\mathbf{x})$ hold:*

- *(Well-defined) For any $p \in \mathcal{P}$, $H(p|\mathbf{x})$ exists and is finite.*
- *(Lower-semicontinous) For a weakly converging sequence $p_n \to p_0 \in \mathcal{P}$ 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 $p \in \mathcal{P}$.*

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

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

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 (Grünwald and Dawid, 2004) 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 $\mathbf{x} \notin \mathcal{X}_{\texttt{IND}}$, we only need to show discrete uniform distribution is the maximum-entropy distribution.

Recall the definition of the *strictly* proper Bregman score (Parry et al., 2012):

$$s(p, p^*|\mathbf{x}) = \sum_{k=1}^{K} \Big\{ [p^*(y_k|\mathbf{x}) - p(y_k|\mathbf{x})]\psi'(p^*(y_k|\mathbf{x})) - \psi(p^*(y_k|\mathbf{x})) \Big\} \tag{37}$$

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

$$H(p|\mathbf{x}) = -\sum_{k=1}^{K} \psi(p(y_k|\mathbf{x})) \tag{38}$$

Our interest is to show that for $\mathbf{x} \in \mathcal{X}_{\texttt{OOD}}$, 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 (38) using the Langrangian form below:

$$L(p|\mathbf{x}) = H(p|\mathbf{x}) + \lambda * (\sum_k p_k - 1) = -\sum_{k=1}^{K} \psi(p_k) + \lambda * (\sum_k p_k - 1) \tag{39}$$

Taking derivative with respect to p_k and λ:

$$\frac{\partial}{\partial p_k} L = -\psi'(p_k) + \lambda = 0 \tag{40}$$

$$\frac{\partial}{\partial \lambda} L = \sum_{k=1}^{K} p_k - 1 = 0 \tag{41}$$

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:

$$p_k = \psi'^{-1}(\lambda); \qquad \sum_{k=1}^{K} p_k = 1. \tag{42}$$

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., $p_k = \frac{1}{K}\ \forall k$. ■