On Feature Collapse and Deep Kernel Learning for Single Forward Pass Uncertainty

Joost van Amersfoort[*]
OATML, University of Oxford

Lewis Smith
OATML, University of Oxford

Andrew Jesson
OATML, University of Oxford

Oscar Key
University College London

Yarin Gal
OATML, University of Oxford

Abstract

Inducing point Gaussian process approximations are often considered a gold standard in uncertainty estimation since they retain many of the properties of the exact GP and scale to large datasets. A major drawback is that they have difficulty scaling to high dimensional inputs. Deep Kernel Learning (DKL) promises a solution: a deep feature extractor transforms the inputs over which an inducing point Gaussian process is defined. However, DKL has been shown to provide unreliable uncertainty estimates in practice. We study why, and show that with no constraints, the DKL objective pushes "far-away" data points to be mapped to the same features as those of training-set points. With this insight we propose to constrain DKL's feature extractor to approximately preserve distances through a bi-Lipschitz constraint, resulting in a feature space favorable to DKL. We obtain a model, DUE, which demonstrates uncertainty quality outperforming previous DKL and other single forward pass uncertainty methods, while maintaining the speed and accuracy of standard neural networks.

1 Introduction

Deploying machine learning algorithms as part of automated decision making systems, such as self driving cars or medical diagnostics, requires implementing fail-safes. Whenever the model is presented with a novel or ambiguous situation, its predictions might not be trustworthy and the input should be deferred to an expert. A compelling way to handle this situation is to use a model that does not just achieve high accuracy, but is also able to quantify its uncertainty over predictions. Unfortunately, standard neural networks extrapolate overconfidently and are not able to express their uncertainty [8]. Alternatives, such as Deep Ensembles [21] and MC dropout [9], require multiple forward passes and are therefore computationally expensive. This leads to problems with using these models in situations that require real-time decision making, which is not feasible if many forward passes are necessary.

This work focuses on the problem of estimating uncertainty in deep learning in a *single* forward pass. This can in principle be achieved by using a distance-aware output function, such as a Gaussian process (GP) with a stationary kernel, which increases its uncertainty as a particular input gets further away from the training data. However on high dimensional structured data, such as images, computing a distance in input space is less meaningful. A practical solution is to transform the input data using a deep feature extractor and placing an approximate GP, that can scale to large datasets, on the computed features. Two key approaches to approximate the GP are Random Fourier Features (RFF) [31, 22] and variational inducing points, which is also known as Deep Kernel Learning (DKL)

---

[*]Correspondence to: `joost.van.amersfoort@cs.ox.ac.uk`

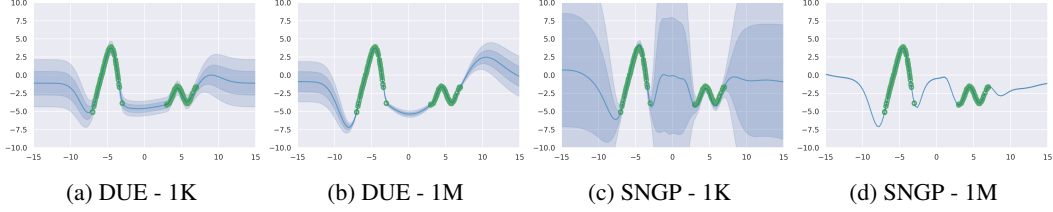| (a) DUE - 1K | (b) DUE - 1M | (c) SNGP - 1K | (d) SNGP - 1M |

Figure 1: In green 300 example training data points and in blue the prediction including uncertainty (one and two std). We see that DUE performs well when trained with 1 thousand (1K) datapoints and 1 million (1M) data points. Meanwhile, the RFF approximation in SNGP concentrates its uncertainty at 1M, and is very uncertain at 1K. This highlights a drawback of the parametric RFF approximation.

[41, 40]. The RFF approximation, while computationally fast, sacrifices the non-parametric property of the GP and is often avoided in the GP literature (Van der Wilk [39] and Figure 1 below). A variational inducing point approximation maintains the GP's non-parametric properties [14, 36], and can in combination with DKL obtain accuracies matching standard softmax neural networks [3, 40]. We discuss the differences between the RFF and inducing point approximation in detail in Section 4.1. However, maintaining good uncertainty estimates remains elusive: previous DKL works have mostly been evaluated in terms of accuracy and robustness to adversarial examples [3, 40] with recent research showing these to underperform in uncertainty estimation [28].

We study why this happens, and propose Deterministic Uncertainty Estimation (DUE - pronounced "Dewey"), which builds on DKL and addresses its limitations. Examining why DKL uncertainty underperforms, we find that for certain feature extractors, data points dissimilar to the training data (also called "out-of-distribution" or OoD data) might be mapped closed to feature representations of in-distribution points. This is called "feature collapse" (Figure 2), and suggests that a constraint must be placed on the deep feature extractor. To understand what constraints must be placed on the DKL feature extractor, we take inspiration from DUQ [37] and SNGP [22] which propose to use a bi-Lipschitz constraint on a feature extractor in the context of radial basis function (RBF) networks and an RFF GP. This constraint enforces the feature representation to be sensitive to changes in the input (lower Lipschitz, avoids feature collapse) but also generalize due to smoothness (upper Lipschitz) (see Figure 3).

Compared to alternative single forward pass uncertainty methods, DUE has a number of advantages. DUQ's one-vs-all loss cannot be extended to regression, while DUE is demonstrated to work well on both classification and regression. SNGP uses the RFF approximation, which means the uncertainty concentrates in the limit of data, even "far-away" from the training data. For example, in the region $-3 < x < 3$ of Figure 1 we are outside of the support of the training data. We can see for a training set size of 1K the uncertainty interval is wide for SNGP (Figure 1c). However, as we increase the training set size to 1M the uncertainty interval outside of the support of the training data becomes indistinguishable from that given for data inside the support (Figure 1b). In contrast to SNGP, DUE's inducing point GP preserves the exact GP's properties and has similar uncertainty outside the support of the training data when trained on small and very large datasets. In contrast to existing DKL methods, DUE avoids feature collapse (Table 1), and training DUE is substantially simplified: no pre-training is necessary and there is no computational overhead over a standard softmax model.

DUE outperforms all alternative single forward pass uncertainty methods on the CIFAR-10 vs SVHN detection task. To evaluate DUE's uncertainty quality for regression on real world data, we use a recently introduced benchmark for predicting treatment effect and uncertainty based treatment deferral in causal models for personalized medicine [18]. This benchmark combines the need for accurate predictions and uncertainty in a regression task and we show that DUE outperforms all alternative methods. We make our code available at `https://github.com/y0ast/DUE`.

In summary, our contributions are as follows:

1. A single forward pass uncertainty model that works well for classification and regression.
2. Insight into DKL's failures and a principled solution that gives accurate uncertainty estimates.
3. A practical DKL training setup that trains from scratch with deep feature extractors on a variety of datasets.

2 Background

In DKL, a deep feature extractor is used to transform the inputs over which an inducing point GP is defined. DKL was originally introduced as a way to combine the expressiveness of deep neural networks with the probabilistic prediction ability of GPs [16, 5, 41]. The defining attribute of DKL is the use of a kernel which contains a deep feature extractor:

$$k_{l,\theta}(\boldsymbol{x}_i, \boldsymbol{x}_j) \rightarrow \bar{k}_l(f_\theta(\boldsymbol{x}_i), f_\theta(\boldsymbol{x}_j)), \tag{1}$$

where $f_\theta(\cdot)$ is a deep neural network, such as a Wide ResNet (WRN) [42] up to the last linear layer, parametrized by θ. The base kernel $\bar{k}_l(\cdot, \cdot)$ can be any of the standard kernels, such as the RBF or Matérn kernel. l represents the hyper-parameters of the base kernel, such as the length scale and output scale. DKL can be applied to both classification and regression problems.

Inference in an exact GP is bottlenecked by the inversion of an $n \times n$ kernel gram matrix with n the number of data points, which has a time complexity that scales cubically with n. In contrast, an inducing point GP is based on m inducing points (with $m << n$), which act as pseudo-input-points used to approximate the full dataset. The locations and values of the inducing points are variational parameters, and learned by maximizing a lower bound on the marginal likelihood known as the ELBO [36, 14]. This reduces the complexity of the matrix inversion from $\mathcal{O}(n^3)$ to $\mathcal{O}(m^2 n)$, thus models with fewer inducing points are faster to train. In practice, these inducing points are placed in feature space to exploit the clustering behavior of the deep feature extractor, we visualize this behavior in the Appendix Figure 6.

The two most prevalent instances of DKL are SV-DKL [40] and GPDNN [3]. In SV-DKL, an additional restriction is placed on the inducing points to lie on a grid. This enables faster matrix inversion algorithms, at the expense of a less flexible inducing point structure. In GPDNN, the inducing points are not constrained, but the feature dimensionality is reduced to just 25, leading to significantly increased risk of feature collapse (see Section 3.1) and poor uncertainty estimation. Both methods use a pre-training phase with a standard, non GP, output, and SV-DKL trains with a mini-batch size of 5,000. GPDNN trains with different optimizers for pre-training and training with GP output, and uses different learning rates for variational versus model parameters. In summary, these models are more cumbersome to use than alternative methods for uncertainty estimation, such as Deep Ensembles [21].

3 Method

In this section we explain how feature collapse affects DKL's uncertainty, give insight into why DKL's objective leads to feature collapse, and use that insight to mitigate feature collapse by introducing the DUE model.

3.1 Feature Collapse

When the deep feature extractor inside the kernel is unconstrained, it can map in- and out-of-distribution data to the same location in feature space. The GP assigns high confidence to inputs similar to the training data, so when in- and out-of-distribution data have the same feature representation, the GP will assign high confidence to out-of-distribution inputs. This is called *feature collapse* [37], and we visualize the pathology in Figure 2.

The objective in DKL directly encourages feature collapse: the ELBO (see Appendix equation 8) consists of an expected log-likelihood term and a KL term ("data fit" and "complexity penalty" respectively [32]). At convergence, the "data fit" term tends to zero, leaving only the "complexity penalty": $\log |K(f_\theta(X), f_\theta(X)) + \sigma^2 I|$. Since σ depends on the scale of observation noise and cannot usually be set to zero, to minimize the penalty term we must minimize the log determinant of the covariance matrix. It is this minimization of the term $|K(f(X), f(X))|$ which leads to feature collapse: the determinant tends to zero for feature representations of X that are collinear (intuitively, features which are mapped close together in feature space up to some constant scale). When optimising the feature extractor f_θ as part of the objective, this leads to feature extractors that collapse features. This is discussed in more detail in Ober et al. [28], who also point out that this behavior can lead to worse overfitting in DKL than standard maximum likelihood training.

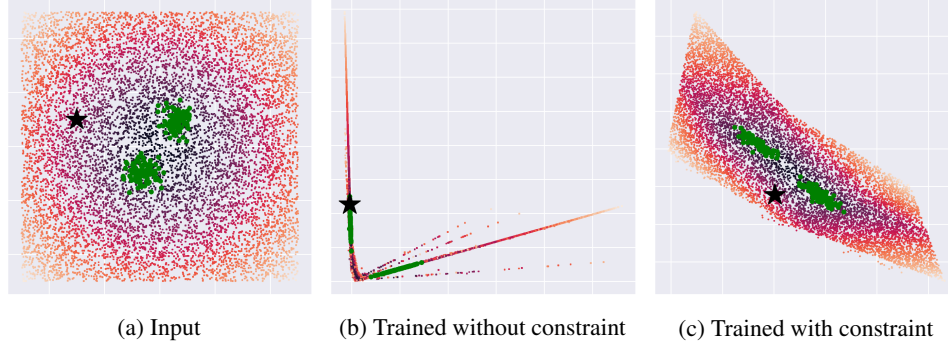| (a) Input | (b) Trained without constraint | (c) Trained with constraint |

Figure 2: A 2D classification task where the classes are two Gaussian blobs (drawn in green), and a grid of unrelated points (colored according to their log-probability under the data generating distribution). We additionally mark a specific point with a star. In (b), the features as computed by an unconstrained model. In (c), the features computed by a model with residual connections and spectral normalization. The objective for the unconstrained model introduces a large amount of distortion of the space, collapsing the input to a single line, making it almost impossible to use distance-sensitive measures on these features. In particular, the star moves from an unrelated area in input space on top of class data in feature space. In contrast, the constrained mapping maintains the relative distances of the other points.

Feature collapse can be reduced by enforcing two constraints on the model: sensitivity and smoothness [37]. Sensitivity implies that when the input changes the feature representation also changes. Thus the model cannot simply collapse feature representations arbitrarily. Smoothness implies small changes in the input cannot cause massive shifts in the output.

These constraints can be achieved by enforcing the feature extractor to be bi-Lipschitz, which has also been shown to help in a large number of other contexts [33]. A bi-Lipschitz feature extractor can be obtained in different ways and each comes with different trade-offs:

- **Two-sided gradient penalty:** this penalty was introduced in the context of GANs [13]. It regularizes using a two-sided gradient penalty, penalizing the squared distance of the gradient from a fixed value at every input point. It is easy to implement, but only a soft constraint. In practice the training stability and regularization effectiveness is sensitive to the relative weight of the penalty in the loss. This method is used in DUQ [37].
- **Direct spectral normalization and residual connections:** spectral normalization [25, 12] on the weights leads to smoothness and it is possible to combine this with an architecture that contains residual connections for the sensitivity constraint. This method is faster than the gradient penalty, and in practice offers a more effective way of mitigating feature collapse. This method is used in SNGP [22].
- **Reversible model:** A reversible model is constructed by using reversible layers and avoiding any down scaling operations [17, 2]. This approach can guarantee that the overall function is bi-Lipschitz, but it consumes considerably more memory and can be difficult to train.

In this work, we use direct spectral normalization and residual connections, as we find it to be more stable than a direct gradient penalty and significantly more computationally efficient than reversible models. In Figure 2, we show that a constrained model does not collapse points on top of each other in feature space, enabling the GP to correctly quantify uncertainty. Using a stationary kernel, the model reverts back to the prior away from the training data just like a GP defined over the input space.

3.2 Model

DUE builds upon the foundation of GPDNN [3]. To enforce the bi-Lipschitz constraint and enable high quality uncertainty, we restrict the deep feature extractor to have residual connections in combination with spectral normalization as described in the previous section. We further make several simplifications to the training process to make GPDNN more practical to use. Instead of an additional downsampling layer to 25 dimensions which increases the risk of feature collapse, the GP output is placed directly on top of the last convolutional layer of a large model, which is 640

Algorithm 1 Algorithm for training DUE

1: **Definitions:**
   - Residual NN $f_\theta : x \to \mathbb{R}^J$ with feature space dimensionality J and parameters θ.
   - Approximate GP with parameters $\phi = \{l, s, \omega\}$, where l length scale and s output scale of $\bar{k}$, ω
     GP variational parameters (including m inducing point locations Z)
   - Learning rate η, loss function $\mathcal{L}$
2: Using a random subset of p points of our training data, $X^{\text{init}} \subset X$, compute:
   Initial inducing points: K-means on $f_\theta(X^{\text{init}})$ with $K = m$. Use found centroids as initial
   inducing point locations Z in GP.
   Initial length scale: $l = \frac{1}{\binom{p}{2}} \sum_{i=0}^{p} \sum_{j=i+1}^{p} |f(X_i^{\text{init}}) - f(X_j^{\text{init}})|_2$.
3: **for** minibatch $\boldsymbol{x}_b, \boldsymbol{y}_b \subset X, Y$ **do**
4:     $\theta' \leftarrow$ spectral_normalization(θ)
5:     $p(\boldsymbol{y}_b'|\boldsymbol{x}_b) \leftarrow$ evaluate_GP$_\phi(f_{\theta'}(\boldsymbol{x}_b))$
6:     $\mathcal{L} \leftarrow$ ELBO$_\phi(p(\boldsymbol{y}_b'|\boldsymbol{x}_b), \boldsymbol{y}_b)$
7:     $(\phi, \theta) \leftarrow (\phi, \theta) + \eta * \nabla_{\phi,\theta}\mathcal{L}$
8: **end for**

---

dimensional in the case of the WRN. In practice, DUE trains well with just 10 inducing points on CIFAR-10 which leads to a runtime that is only 3% slower than a softmax model. No pre-training is necessary and training is stable with a single set of hyper-parameters for both the variational and model parameters. The inference procedure is described in detail in Appendix A. The training procedure is described in Algorithm 1.

Online power iteration for spectral normalization can be implemented exactly for fully connected layers and 1x1 convolutions. For convolutions larger than 1x1, we use an approximate method that lower bounds the exact method, as proposed in Gouk et al. [12] and implemented by Behrmann et al. [2].

Batch Normalization In SNGP, spectral normalization is applied only on the convolution operations. However batch normalization, a crucial component of training deep models, has a non-trivial Lipschitz constant. Batch normalization transforms the input following:

$$x_{out} = \text{diag}\left(\frac{\gamma}{\sqrt{\text{Var}(x)}}\right)(x - \mathbb{E}[x]) + \beta, \tag{2}$$

with γ and β the learnable scale and shift parameters. It has a Lipschitz constant of $\max_i |\frac{\gamma_i}{\sqrt{\text{Var}(x)_i}}|$ [12]. Using the above equation, we can extend spectral normalization to batch normalization by dividing the weight γ of the batch normalization by the (scaled) Lipschitz constant. In practice, we find that batch normalization layers in trained ResNets have a relatively high Lipschitz constant, up to around 12, with 95% of the channel-wise Lipschitz constants being greater than one (see also Figure 5 in the Appendix). Adding spectral normalization to batch normalization layers ensures the entire network's upper Lipschitz constant is bounded.

4 Related Work

The single forward pass uncertainty methods most similar to DUE are DUQ [37] and SNGP [22], as discussed in the introduction and in the context of enforcing the bi-Lipschitz constraint. Compared to DUQ, DUE can readily be applied to regression problems, and the gradient penalty in DUQ is difficult to optimize and slow to compute. SNGP [22] consists of a deep feature extractor and an output GP, which is approximated using the parametric Random Fourier Features (RFF) approximation [31] in combination with a Laplace approximation for the non-conjugate Softmax likelihood [32]. We discuss the trade-offs between the RFF and variational inducing point approximations in the next subsection.

4.1 Inducing Points versus RFF Approximation

Inducing point GP approximations maintain the non-parametric properties of the full GP, while the RFF approximation sacrifices this. With the RFF approximation, for any *finite* number of features,

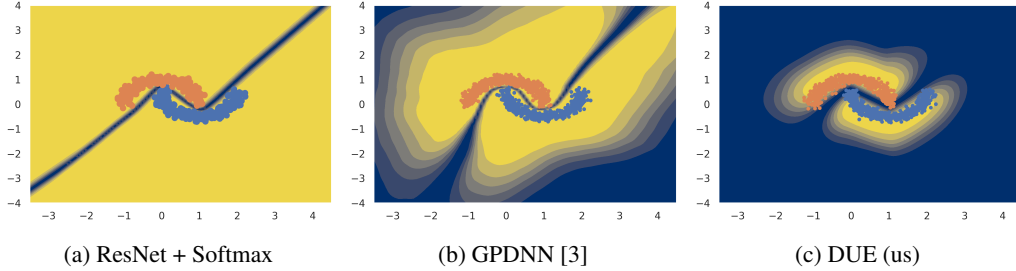(a) ResNet + Softmax   (b) GPDNN [3]   (c) DUE (us)

Figure 3: We show uncertainty results on the two moons dataset. Yellow indicates high confidence, while blue indicates uncertainty. In Figure 3a, a simple feed-forward ResNet with a softmax output is certain everywhere except on the decision boundary. In Figure 3b, we see that GPDNN, which uses a simple Feed-Forward Network as feature extractor, is certain even far away from the training data. In Figure 3c, we show DUE, which has the appropriate restrictions on the feature extractor (residual connections and spectral normalization) and obtains close to ideal uncertainty on this dataset.

the kernel is approximated as a linear model on a finite number of features. Because of this, the RFF GP's uncertainty will erroneously concentrate to zero as the number of training points increases even in areas where there is no training data. Various extensions of RFF (and its close relative, the sparse spectrum GP approximation) have attempted to fix this problem, see for example Gal et al. [10].

Inducing point GP approximations, on the other hand, use as the approximation a standard GP which is defined over a set of *inducing inputs* instead of over the entire training set (which is computationally prohibitive). These do not change the kernel definition (unlike the RFF approximation), and the GP is still non-parametric. The inducing point approximation then tries to move the inducing input locations to minimize the KL between this inducing point GP and the full GP [36], giving a tractable objective which also preserves the full GP's non-parametric properties. The approximation will become exact when the number of inducing points matches the number of training points (at which point the ELBO can match the GP marginal likelihood by placing the inducing inputs on the training input locations). This is in contrast to RFF which will only become exact at the infinite limit of the number of random features. Inducing point GPs also have a tight bound on the marginal log-likelihood, which means that we can do model selection by optimising the ELBO with respect to various hyper-parameters [4].

In Figure 1, we show the effects of the approximations in DUE and SNGP in practice by training on a small and large dataset sampled from the same distribution. SNGP's uncertainty interval is dependent on the number of training points: it is wide in areas where there is no training data at 1K datapoints, but very narrow in the same regions at 1M datapoints. While DUE's inducing point GP has similar uncertainty outside the support of the training data when trained on either dataset size. SNGP uncertainty was estimated using the exact method with a ridge penalty of 1. All other hyper-parameters were shared between the two methods, and are listed in Appendix B.

5 Experiments

5.1 Feature Collapse in DKL

We analyze the problem of feature collapse in DKL in Figure 3 on the Two Moons dataset [30]. We show results for three different models: a standard softmax model, DUE, and a variation where the spectral normalized ResNet is replaced by a fully connected model (similar to Bradshaw et al. [3]), further details are provided in Appendix B. The uncertainty is computed using the predictive entropy of the class prediction; we model the problem as a two class classification problem.

The standard softmax output model is certain everywhere except on the decision boundary. DUE (Figure 3c) quantifies uncertainty as expected for the two moons dataset: certain on the training data, uncertain away from it and in-between the two moons. Figure 3b highlights the importance of our contribution, because the uncertainty estimation of a standard Feed-Forward Network in combination with DKL suffers from feature collapse and is certain away from the training data.

6

Table 1: AUROC on CIFAR10 vs SVHN for two DKL methods trained with feature extractors with and without a bi-Lipschitz constraint. Non bi-Lipschitz uses VGG-19 [35]; bi-Lipschitz uses a WRN with spectral normalization. Both models obtain high accuracy, matching standard softmax models. SV-DKL without bi-Lipschitz obtains poor uncertainty: distinguishing in- and out-of-distribution data is no better than chance. GPDNN without bi-Lipschitz obtains better but still poor uncertainty. The cell highlighted in gray is DUE.

| | bi-Lipschitz | | non bi-Lipschitz | |
	AUROC	Accuracy	AUROC	Accuracy
SV-DKL [40]	**0.959±.001**	95.7±0.06	0.498±.001	93.6±0.05
GPDNN [3]	**0.958±.005**	95.6±0.04	0.876±.004	93.7±0.10

Table 2: Results on the CIFAR-10 dataset, and distinguishing between CIFAR-10 and SVHN by uncertainty. All results use a WRN as feature extrator and are the average and standard error of 5 runs. The runtime is relative to the "Standard WRN" row. In bold are top results (within standard error), and the horizontal line separates ensembles from single forward pass methods.

Method	Runtime ↓	Accuracy (%) ↑	AUROC ↑
Ensemble of 5 [21]	5x	**96.6±0.03**	**0.967±0.005**
Standard WRN [42]	**1x**	**96.2±0.01**	0.932±0.008
DUQ [37]	3.5x	94.9±0.04	0.940±0.003
SNGP [22]	**1x**	96.0±0.04	0.940±0.006
SV-DKL (with constraints) [40]	2x	95.7±0.06	**0.959±0.001**
DUE	**1x**	95.6±0.04	**0.958±0.005**

5.2 Feature Collapse in CIFAR-10 versus SVHN

We now consider training end-to end with a large feature extractor, the Wide Residual Network (WRN), on CIFAR-10 [20]. We follow Zagoruyko and Komodakis [42] and use a 28 layer model with BasicBlocks and dropout. Importantly, we can use their hyper-parameters (such as dropout rate, learning rate, and optimizer) when training DUE, and no further tuning is necessary. We remove the final linear layer of the original WRN and the 640-dim feature vector is directly used in the GP. We use 10 inducing points for DUE on CIFAR-10, which leads to a runtime of 1m47s for one epoch on an Nvidia GTX1080 Ti, while a standard WRN with a linear+softmax output takes 1m43s. An ablation were we look at the learned location of the 10 inducing points is available in the Appendix Figure 6, and using more inducing points is evaluated in Table 4.

Uncertainty. To compare the uncertainty quality between different models, we use the experiment of distinguishing between the test sets of CIFAR-10 and SVHN [27] – a notably difficult dataset pair [26] – using each model's uncertainty metric. In Table 1, we compare SV-DKL and GPDNN with two different feature extractors: with and without the DUE constraints. In contrast to the original SV-DKL and GPDNN, these models were trained from scratch using the same hyper parameters (including mini-batch-size) as DUE. Note that GPDNN with constraints and the improved training setup is conceptually the same as DUE. We see that in both cases the accuracy is high, but the uncertainty is very poor when no constraints are placed. This highlights the importance of our contribution: additional constraints on the feature extractor are crucial for good uncertainty performance in DKL.

Table 2 compares DUE to several baselines. We train DUE using 10 inducing points and spectral normalization constant 3, set in similar fashion to Liu et al. [22] by choosing the lowest value that does not significantly affect accuracy. All methods compute uncertainty using the predictive entropy, except DUQ which uses the closest kernel distance [37]. We see that DUE and SV-DKL, with DUE feature extractor, outperform all competing single forward pass uncertainty methods. DUE is competitive with Deep Ensembles uncertainty, but only requires a single forward pass and is therefore five times faster to train. SV-DKL with the DUE feature extractor performs similar to DUE, but takes 2x longer to train, since it needs to invert a larger covariance matrix. The DUE and SV-DKL result shows that the theoretically preferable inducing point variational approximation also leads to practical improvements, as both outperform SNGP in terms of uncertainty estimation.

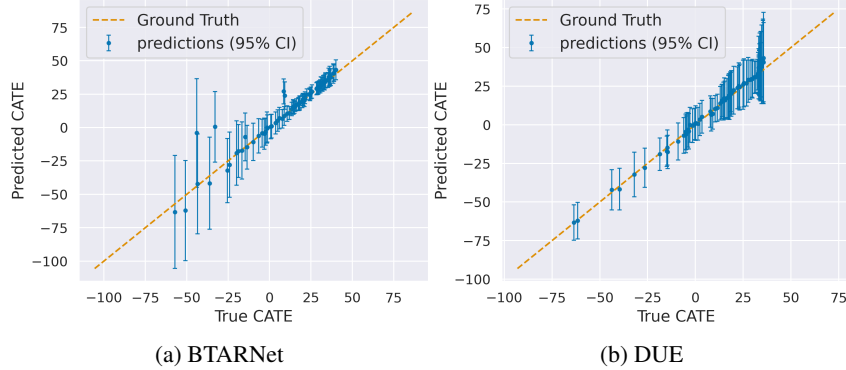|	
(a) BTARNet	(b) DUE

Figure 4: Predicted CATE versus true CATE with 95% confidence intervals for a randomly chosen cross-validation run. DUE is confident (without converging to no uncertainty) and correct, while BTARNet [34] is wrong in 2 instances and the true CATE is not within the confidence interval.

5.3 Uncertainty in Regression for Personalized Healthcare

To demonstrate the performance of DUE on regression, we focus on a new benchmark on personalized healthcare [18]. The task is to predict responses of individuals to a particular treatment, which is only possible if there is sufficient data available to assess how they might respond. This is also called the *overlap assumption*, which states that to predict the effect of treatment for a particular input x (an individual described by features), we need to have seen similar data points that have received treatment as well as points that have not received treatment. We can use uncertainty to assess when this assumption is violated and the patient should be referred to an expert instead [18]. It is important that the uncertainty estimates are accurate, otherwise an individual could receive treatment even if their response to treatment is not known, which can result in undue stress, financial burdens, or worse.

This benchmark assesses both predictive performance and uncertainty estimation. Other benchmarks which only evaluate test log-likelihood, such as UCI [7], only capture uncertainty on the test set (in-distribution), which is insufficient to evaluate uncertainty on out-of-distribution data. In the case of feature collapse, the test log-likelihood is therefore not a useful comparison tool.

We train DUE on the input features of the training set, appending a 0 or 1 to represent no treatment and treatment. The model is then used to predict the Conditional Average Treatment Effect (CATE) [1], computed as the difference between the expected effect of treatment and no treatment. The CATE estimate and its uncertainty are the expectation and variance over the difference in joint predictions with the two inputs:

$$[y_0, y_1] \sim \text{DUE}([\boldsymbol{x}, t = 0], [\boldsymbol{x}, t = 1])$$
$$\text{CATE}(x) = \mathbb{E}[y_1 - y_0] \qquad \text{CATE}_{\text{u}}(x) = \text{Var}[y_1 - y_0].$$

The CATE can be computed exactly, using the mean of the GP posterior, while we use Monte Carlo sampling of the joint posterior for the uncertainty (note that this requires only a single forward pass through the model, and sampling from the GP is fast). We use the uncertainty to decide which predictions will be deferred to an expert. This process allows us to make a *causal* statement on the effect of treatment, under the assumptions in Jesson et al. [18].

We use the IHDP [15] and IHDP Covariate shift (referred to as IHDP Cov.) datasets. IHDP is a regression datasets with ∼750 data points, and IHDP Cov. is a variant with additional covariate shift to increase the difficulty of the task. These are real world datasets derived from the Infant Health and Development Program. The details of the covariate shift are discussed in Jesson et al. [18]. In the experiments, treatment-effect recommendations are deferred to an expert if the CATE estimate has high uncertainty. We include a baseline that defers at random. We run IHDP and its covariate shift variant for 1,000 cross-validation trials.

In Figure 4, we compare DUE with Bayesian TARNet, which is the standard TARNet [34] extended with MC dropout [9] for uncertainty quantification. The TARNet is a commonly used deep learning baseline in the causality field. The results show that DUE is more accurate than BTARNet for most CATE values, and that the BTARNet makes predictions for which the ground truth is not within the

Table 3: Comparing the performance in terms of RMSE of several treatment effect and uncertainty estimation models under "random" and "uncertainty" based deferral to an expert. The first three rows were obtained from Jesson et al. [18], DKLITE was evaluated using the author's open source implementation and the ensemble of TARNet was reimplemented. DUE outperforms all alternative methods, while being 5x faster to train and evaluate than the ensemble.

| Method | IHDP Cov. (50% def.) | | IHDP (10% def.) | |
	random	*uncertainty*	*random*	*uncertainty*
BART [6]	2.6±.2	1.8±.2	1.90±.20	1.60±.10
BTARNet [34, 18]	2.2±.3	1.2±.1	1.10±.03	0.76±.03
BCEVAE [23, 18]	2.5±.2	1.7±.1	1.80±.06	1.47±.08
DKLITE [43]	2.6±.7	1.8±.5	1.74±.53	1.34±.41
Ensemble of 5 TARNet	1.74±.1	1.19±.03	1.14±.04	0.76±.01
DUE	**1.63±.06**	**1.05±.05**	**0.91±.04**	**0.48±.02**

confidence interval. In Table 3, we compare across several models on the two datasets. The additional baselines are: BART which is based on decision trees [6], BCEVAE which is the MC Dropout extension of CEVAE [23], and DKLITE [43] which is a recent method also based on DKL. DKLITE works by jointly training a VAE [19] with a GP on the latent dimensions of the VAE. BTARNet, ensemble of TARNet and DUE all use the same feature extractor, as detailed in Appendix B. We report the root mean squared error between the predicted CATE and the true CATE (which is given in the dataset) to assess the error on the held out data set (lower is better). Table 3 summarizes our results and shows that DUE has improved performance and uncertainty estimates better suited to rejection policies than other uncertainty-aware methods.

6   Conclusion and Limitations

We demonstrated that DUE outperforms alternative single forward pass methods on CIFAR-10, and obtains SotA performance in a personalized medicine regression benchmark. These results show that DUE overcomes the previous problems with uncertainty in DKL, and makes DKL a viable method for uncertainty estimation. The main limitation of DUE is that despite the empirical improvements, the uncertainty estimation is not guaranteed to be correct. In particular, the downsampling operations are not bi-Lipschitz and future work is necessary to assess the practical implications of this. A potential negative societal impact of this work is that if DUE is used in sensitive applications and incorrectly estimates its uncertainty, then decisions might be made in an automated fashion that should have been deferred to an expert.

References

[1] Jason Abrevaya, Yu-Chin Hsu, and Robert P Lieli. Estimating conditional average treatment effects. *Journal of Business & Economic Statistics*, 33(4):485–505, 2015.

[2] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582, 2019.

[3] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

[4] David Burt, Carl Edward Rasmussen, and Mark Van Der Wilk. Rates of convergence for sparse variational gaussian process regression. In *International Conference on Machine Learning*, pages 862–871. PMLR, 2019.

[5] Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3338–3345. IEEE, 2016.

[6] Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

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

[8] Yarin Gal. Uncertainty in deep learning. *University of Cambridge*, 1(3), 2016.

[9] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[10] Yarin Gal, Yutian Chen, and Zoubin Ghahramani. Latent gaussian processes for distribution estimation of multivariate categorical data. In *International Conference on Machine Learning*, pages 645–654. PMLR, 2015.

[11] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[12] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael Cree. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[14] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. *JMLR*, 2015.

[15] Jennifer L Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011.

[16] Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in neural information processing systems*, pages 1249–1256, 2008.

[17] Jörn-Henrik Jacobsen, Arnold WM Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. In *International Conference on Learning Representations*, 2018.

[18] Andrew Jesson, Sören Mindermann, Uri Shalit, and Yarin Gal. Identifying causal-effect inference failure with uncertainty-aware models. *Advances in Neural Information Processing Systems*, 33, 2020.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

[21] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[22] Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *Advances in Neural Information Processing Systems 33*, 2020.

[23] Christos Louizos, Uri Shalit, Joris Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6449–6459, 2017.

[24] Alexander Graeme de Garis Matthews. *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge, 2017.

[25] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[26] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2018.

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

[28] Sebastian W Ober, Carl E Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. *arXiv preprint arXiv:2102.12108*, 2021.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 2019.

[30] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[31] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[32] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited, 2006. ISBN 9780262182539. URL https://books.google.co.uk/books?id=vWtwQgAACAAJ.

[33] Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. A case for new neural network smoothness constraints. *arXiv preprint arXiv:2012.07969*, 2020.

[34] Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, pages 3076–3085. PMLR, 2017.

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[36] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[37] Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, 2020.

[38] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[39] Mark Van der Wilk. *Sparse Gaussian process approximations and applications*. PhD thesis, University of Cambridge, 2019.

[40] Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016.

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

[42] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[43] Yao Zhang, Alexis Bellot, and Mihaela Schaar. Learning overlapping representations for the estimation of individualized treatment effects. In *International Conference on Artificial Intelligence and Statistics*, pages 1005–1014. PMLR, 2020.
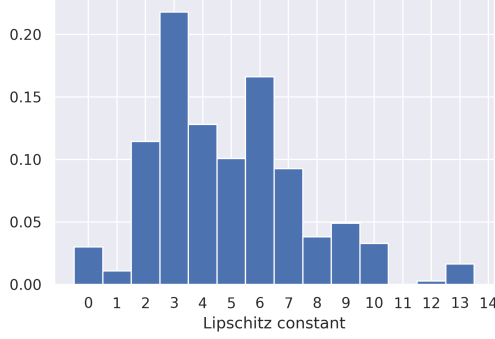
Figure 5: A density of the Lipschitz values in batch normalization layers, averaged across 15 WRN models that were trained with Softmax output and without spectral normalization (exactly following Zagoruyko and Komodakis [42]). We see that many of the constants are significantly above 1, highlighting that batch normalization has significant impact on the Lipschitz constant of the network.
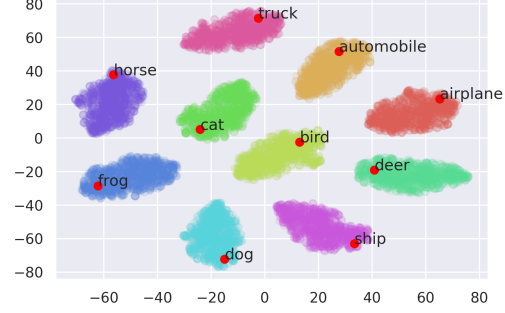

Figure 6: Visualization of the learned feature representations of the training set of CIFAR-10. We use T-SNE [38] to reduce the dimensionality down to 2D and we color the points by their class label. We overlay (in red) the inducing points location, which we label by computing the closest class average T-SNE representation. The features and inducing point locations are obtained from a DUE model trained with 10 inducing points and spectral normalization.

A Model details

DUE is an instance of DKL [41] and uses the sparse GP of Titsias [36] and the variational approximation of Hensman et al. [14]. In this section we give a complete description of the model.

A.1 Full model definition

Let $X \in \mathbb{R}^{N \times D}$ and $Y \in \mathbb{R}^{N \times T}$ be a dataset of N points with input dimensionality D and output dimensionality T. In addition to regression, we also consider classification tasks. For classification tasks, T is the number of classes, with a single instance $\boldsymbol{y} \in Y$ being a T dimensional vector of class probabilities. Let $F \in \mathbb{R}^{N \times T}$ be the value of a T dimensional latent function at each input. For regression tasks F is the values of the underlying noiseless function the GP is modelling. The model is formed of an independent GP for each output dimension. The joint distribution over Y and F, evaluated at inputs X, is

$$p(Y, F; X) = p(Y \mid F) \prod_{t=1}^{T} p(F_{[:,t]}; X) \tag{3}$$

$$p(F_{[:,t]}; X) = \mathcal{GP}(\mu_t(X), k_{l_t,\theta}(X, X)). \tag{4}$$

Here $F_{[:,t]}$ refers to the tth column vector of the matrix F. $\mu_t(\cdot)$ is a mean function for output dimension t. For both regression and classification we use a constant mean function $\mu_t(X) = \mu_t$, where μ_t is a hyperparameter. $k_{l,\theta}(\cdot, \cdot)$ is a deep kernel function with feature extractor parameters θ, as we discuss in Section 3.2, and base kernel $\bar{k}_{l_t}(\cdot, \cdot)$ with hyperparameters l_t specific to each output dimension (but shared for each input dimension). $p(Y \mid F)$ is the likelihood function. For regression tasks this is defined $p(Y \mid F) = \prod_{i=1}^{N} \mathcal{N}(Y_{[i,:]} \mid F_{[i,:]}, \sigma^2 I_T)$, where σ^2 is a variance hyparameter and I is the identity matrix. For classification tasks,

$$p(Y \mid F) = \prod_{i=1}^{N} p(Y_{[i,:]} \mid F_{[i,:]}) \tag{5}$$

$$p(Y_{[i,:]} \mid F_{[i,:]}) = \text{softmax}(F_{[i,:]})_{(\arg\max_c Y_{[i,c]})}. \tag{6}$$

Note that while μ_t, σ^2, and l_t are described as hyperparameters, we do not specify them manually but instead learn them alongside the other model parameters, as below.

13

Table 4: Test accuracy and Negative Log-Likelihood (NLL) on CIFAR-10 of DUE with WRN feature extractor for increasing number of inducing points (m). As the number of inducing points increases, both the NLL and accuracy remain constant with no statistically significant difference. This shows that is not necessary to have a large number of inducing points to obtain strong performance.

m	Accuracy (%) ↑	NLL ↓
10	95.56±0.04	0.187±0.004
50	95.54±0.06	0.182±0.002
100	95.35±0.06	0.183±0.002
1000	95.49±0.05	0.180±0.001

Table 5: ECE calibration results on CIFAR-10 with 15 bins and no post-hoc scaling. DUQ and SNGP results obtained from Liu et al. [22].

Method	ECE
DUE	0.01795 +/- 0.0015
DUQ	0.034 +/- 0.002
SNGP	0.018 +/- 0.001

A.2 Sparse GP approximation and variational inference

Exact inference for the classification likelihood is not tractable because the softmax function is not a conjugate likelihood to the GP prior. Additionally, while exact inference is possible for the regression case, the computational complexity scales cubically with the number of data points, thus it is not suitable for large datasets. Thus, for both regression and classification we use a sparse GP approximation and variational inference.

We use the sparse GP approximation of Titsias [36] which augments the model with M inducing inputs, $Z \in \mathbb{R}^{M \times J}$, where J is the dimensionality of the feature space. The associated inducing variables, $U \in \mathbb{R}^{M \times T}$, give the function value at each inducing input. Together, the inducing inputs and inducing variables approximate the full dataset. To perform inference in this model we use the variational approximation introduced by Hensman et al. [14]. Here Z are treated as variational parameters. U are random variables with prior $p(U) = \prod_{t=1}^{T} \mathcal{N}(U_{[:,t]} \mid \mu_t(Z), \bar{k}(Z, Z))$, and variational posterior $q(U) = \prod_{t=1}^{T} \mathcal{N}(U_{[:,t]} \mid \boldsymbol{m}_t, S_t)$, where $\boldsymbol{m}_t \in \mathbb{R}^M$ and $S_t \in \mathbb{R}^{M \times M}$ are variational parameters and initialized at the zero vector and the identity matrix respectively. The approximate predictive posterior distribution at test points X^* is then

$$q(F^* \mid Y; X, X^*) = \int p(Y \mid F^*)p(F^* \mid U; X^*, Z) \prod_{t=1}^{T} q(U_{[:,t]} \mid \boldsymbol{m}_t, S_t) \, dU \, dF^*. \quad (7)$$

Here $p(F^* \mid U; X^*, Z)$ is a Gaussian distribution for which we have an analytic expression, see Hensman et al. [14] for details. Note that we deviate from Hensman et al. [14] in that our input points x are mapped into feature space just before computing the base kernel, while inducing points are used as is (they are defined in feature space).

The variational parameters Z, $\boldsymbol{m}_t$, and S_t, alongside the feature extractor parameters θ and model hyparparameters μ_t, l_t, and σ^2, are all learned by maximizing a lower bound on the log marginal likelihood, known as the ELBO, $\mathcal{L}$. For the variational approximation above, this is defined as

$$p(Y; X) \geq \mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_{q(F_{[i,:]} \mid \boldsymbol{m}_1, \ldots, \boldsymbol{m}_T, S_1, \ldots, S_T; \boldsymbol{x}_i, Z)} \left[\log p(Y_{[i,:]} \mid F_{[i,:]}) \right] - D_{\mathrm{KL}}(q(U) || p(U)).$$
$$(8)$$

Both terms can be computed analytically when the likelihood is Gaussian and for classification (i.e. a non Gaussian likelihood) we do MC sampling. Armed with this objective function, we can learn the model parameters and hyperparameters, and variational parameters, using stochastic gradient descent. To accelerate optimization we additionally use the whitening procedure of Matthews [24]. We specify the precise optimizer configuration for each experiment later in this section.

A.3 Making predictions and measuring uncertainty

For regression tasks we directly use the function values F^* above as the predictions. We use the mean of $q(F^* \mid Y; X, X^*)$ as the prediction, and the variance as the uncertainty.

For classification tasks we need the posterior over the class probabilities, $q(Y^* \mid Y; X, X^*)$, rather than the latent function values. Thus, we approximate the integral

$$\bar{Y}^* = \int \text{softmax}(F^*) q(F^* \mid Y; X, X^*) \, dF^*, \tag{9}$$

using Monte Carlo samples (32 in practice), which are very fast to compute and do not require additional forward passes. Note that we consider the inputs *independent* at test time (i.e. we only take the diagonal of the posterior covariance) which is especially important when detecting out of distribution data.

The predicted class for input i is then the most likely class in $\bar{Y}^*_{[i,:]}$. To estimate the uncertainty of the prediction we compute the entropy of $\bar{Y}^*_{[i,:]}$:

$$\text{entropy}(\bar{Y}^*_{[i,:]}) = -\sum_c \bar{Y}^*_{[i,c]} \log \bar{Y}^*_{[i,c]}. \tag{10}$$

A.4 Implementation and Compute

The inducing points locations are initialized using centroids obtained from performing k-means on the feature representation of 1,000 points randomly chosen from the training set. The initial length scales are computed by taking the average pairwise euclidean distance between feature representation of the 1,000 points. The models are implemented using GPyTorch [11] and PyTorch [29] and use their default values if not otherwise specified.

B Experimental Details

B.1 1D and 2D experiments

We perform these experiments using a simple feed-forward ResNet, similar to Liu et al. [22]. The first linear layer maps from the input to the initial feature representation and does not have an activation function. After which the model is a fully connected ResNet consisting of blocks computing $x' = x + f(x)$, with $f(\cdot)$ a combination of a linear mapping and a ReLU activation function. Spectral normalization is applied to the linear mapping for which we use the implementation of Behrmann et al. [2]. We use the Adam optimizer for regression and SGD for the two moons classification with learning rate 0.01. We use 4 layers with 128 features, a Lipschitz constant of 0.95, a single power iteration, an RBF kernel and additive Gaussian noise with standard deviation 0.1. For toy regression, we use 20 inducing points and for two moons we use four. For the Deep Ensemble in Figure 7 we train 10 separate fully connected ResNet models, using a different initialization and data order for each and train to maximize the Gaussian log-likelihood as described in Section 2.4 of Lakshminarayanan et al. [21].

B.2 CIFAR-10

For the WRN, we follow the experimental setup and implementation of Zagoruyko and Komodakis [42]. This means that for CIFAR-10, we use depth 28 with widen factor 10 and BasicBlocks with dropout. We train for the prescribed 200 epochs (no early stopping) with batch size 128, starting with learning rate 0.1 and dropping with a factor of 0.2 at epoch 60, 120 and 160. We use momentum 0.9 and weight decay 5e-4. We select 20% of the training data at random as a validation set for hyper-parameter tuning, but obtain the final model by using the full training set with the final set of hyper-parameters. SV-DKL was trained using the suggested values in the GPyTorch [11] tutorial: grid size set to 64 and grid bounds between -10 and 10.

For spectral normalization, we use the implementation of Behrmann et al. [2], and also constrain batch normalization as described in [12]. We use 1 power iteration and use the lowest Lipschitz constant that still allows for good accuracy, which we found to be around 3 in practice. We set the momentum of spectral batch normalization to 0.99 to reduce the variance of the running average estimator of the empirical feature variance, which can be a source of instability.

In the out-of-distribution detection experiment, it is crucial to preprocess the SVHN images in the same way as CIFAR-10, as we cannot know at test time from which dataset the image comes and the only sensible procedure is to preprocess as if it was coming from the training dataset.

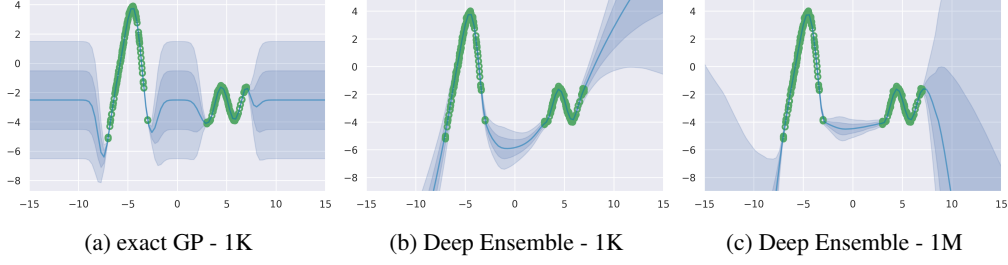(a) exact GP - 1K   (b) Deep Ensemble - 1K   (c) Deep Ensemble - 1M

Figure 7: In green 300 example training data points and in blue the prediction including uncertainty (one and two std). The exact GP could only be fit on 1K data points, so the 1M version is omitted. The Deep Ensemble does not revert to any prior, but also does not concentrate fully in the 1M case.

B.3 Regression Experiment

Following Shalit et al. [34], we use 63%/27%/10% train / validation / test splits and report the RMSE evaluated on the test set over 1000 trials. For each trial, we train for a maximum of 750 epochs with batch size 100 and report results on the model with the lowest negative log-likelihood evaluated on the validation set. We employ Adam optimization with a learning rate of 0.001 and batch size of 100.

The feature extractor uses a feed-forward ResNet architecture with 3 layers, 200 hidden units per layer, and ELU activations. Dropout is applied after each activation at a rate of 0.1. The feature extractor takes the individual x as input, and treatment t is appended to the output of the feature extractor, in similar fashion to the TARNet architecture. Spectral normalization with value 0.95 is used on all layers of the feature extractor. The output GP uses a Matérn kernel with $\nu = \frac{3}{2}$ and 100 inducing points.

For the DKLITE experiments we use the open source code from the authors[2]. We write a custom loop over the IHDP dataset to follow the above protocol.

---

[2]https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/master/alg/dklite/