Discriminative Jackknife: Quantifying Uncertainty in Deep Learning via Higher-Order Influence Functions

Ahmed M. Alaa [1] **Mihaela van der Schaar** [1 2]

Abstract

Deep learning models achieve high predictive accuracy across a broad spectrum of tasks, but rigorously quantifying their predictive uncertainty remains challenging. Usable estimates of predictive uncertainty should (1) *cover* the true prediction targets with high probability, and (2) *discriminate* between high- and low-confidence prediction instances. Existing methods for uncertainty quantification are based predominantly on Bayesian neural networks; these may fall short of (1) and (2) — i.e., Bayesian credible intervals do not guarantee frequentist coverage, and approximate posterior inference undermines discriminative accuracy. In this paper, we develop the *discriminative jackknife* (DJ), a frequentist procedure that utilizes influence functions of a model's loss functional to construct a jackknife (or leave-one-out) estimator of predictive confidence intervals. The DJ satisfies (1) and (2), is applicable to a wide range of deep learning models, is easy to implement, and can be applied in a *post-hoc* fashion without interfering with model training or compromising its accuracy. Experiments demonstrate that DJ performs competitively compared to existing Bayesian and non-Bayesian regression baselines.

1. Introduction

Deep learning models have achieved state-of-the-art performance on a variety of learning tasks, and are becoming increasingly popular in various application domains (LeCun et al., 2015). A key question often asked of such models is "*Can we trust this particular model prediction?*" This question is highly relevant in high-stakes applications wherein predictions are used to inform critical decision-making — examples of such applications include: medical decision support (Alaa & van der Schaar, 2017), autonomous vehicles, and financial forecasts (Amodei et al., 2016). Despite their impressive accuracy, rigorously quantifying uncertainty in deep learning models is a challenging and yet an unresolved problem (Gal, 2016; Ovadia et al., 2019).

Actionable estimates of predictive uncertainty are ones that (1) *cover* the true prediction targets with a high probability, and (2) *discriminate* between high- and low-confidence predictions. (Figure 1 depicts a pictorial visualization for these coverage and discrimination requirements.) The coverage requirement is especially relevant in applications where predictive uncertainty is incorporated in a decision-theoretic framework (e.g., administering medical treatments (Dusenberry et al., 2019), or estimating value functions in model-free reinforcement learning (White & White, 2010)). The second requirement, discrimination, is crucial for auditing model reliability (Schulam & Saria, 2019), detecting dataset shifts and out-of-distribution samples (Barber et al., 2019a), and actively collecting new training examples for which the model is not confident (Cohn et al., 1996).

Existing methods for uncertainty estimation are based predominantly on Bayesian neural networks (BNNs), whereby predictive uncertainty is evaluated via posterior credible intervals (Welling & Teh, 2011; Hernández-Lobato & Adams, 2015; Ritter et al., 2018; Maddox et al., 2019). However, BNNs require significant modifications to the training procedure, and exact Bayesian inference is computationally prohibitive in practice. Approximate dropout-based inference schemes (e.g., Monte Carlo dropout (Gal & Ghahramani, 2016) and variational dropout (Kingma et al., 2015)) have been recently proposed as computationally efficient alternatives. However, Bayesian inference in dropout-based models has been shown to be ill-posed, since the induced posterior distributions in such models do not concentrate asymptotically (Osband, 2016; Hron et al., 2017), which jeopardizes both the coverage and discrimination performance of the resulting credible intervals. Moreover, even with exact inference, Bayesian credible intervals do not guarantee frequentist coverage (Bayarri & Berger, 2004). Non-Bayesian alternatives have been recently developed based on ad-hoc ensemble designs (Lakshminarayanan et al., 2017) — but formal and rigorous frequentist methods are still lacking.

[1]UCLA [2]Cambridge University. Correspondence to: Ahmed M. Alaa <ahmedmalaa@ucla.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

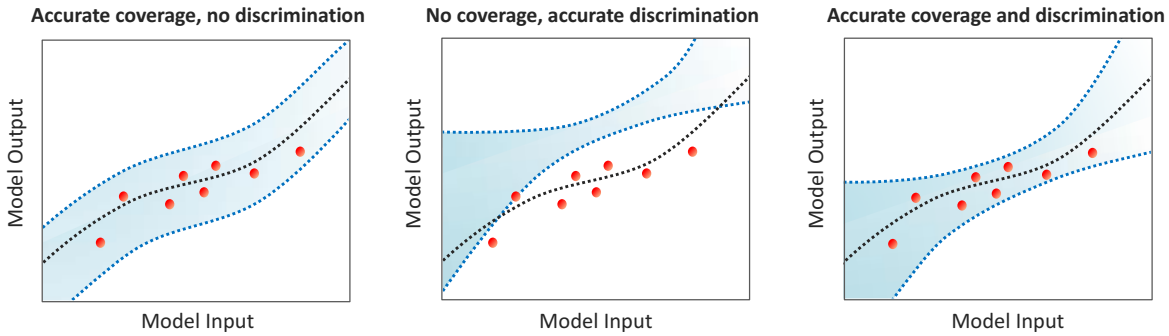

Figure 1. **Pictorial depiction for coverage and discrimination in uncertainty estimates.** Red dots correspond to training data and dotted black line corresponds to the target function. Confidence intervals are visualized as shaded blue regions, where dotted blue lines are the upper and lower confidence limits. The left panel shows a confidence interval that perfectly covers the data points, but does not discriminate high-confidence predictions (regions with dense training data) and low-confidence ones (regions with scarce training data). The middle panel shows a confidence interval with a width proportional to the density of training data (which determines model uncertainty), but does not cover any data point. The right panel shows a confidence interval that satisfies both coverage and discrimination requirements.

Summary of contributions. In this paper, we develop a formal procedure for constructing frequentist (pointwise) *confidence intervals* on the predictions of a broad class of deep learning models. Our method — which we call the *discriminative jackknife* (DJ) — is inspired by the classic jackknife leave-one-out (LOO) re-sampling procedure for estimating variability in statistical models (Miller, 1974; Efron, 1992). In order to ensure both frequentist coverage and discrimination, DJ constructs feature-dependent confidence intervals using the LOO local prediction variance at the input feature, and adjusts the interval width (for a given coverage probability) using the model's average LOO error residuals. Whereas the classic jackknife satisfies neither the coverage nor the discrimination requirements (Barber et al., 2019b), DJ satisfies both (i.e., DJ generates predictive confidence intervals resembling those in the rightmost panel of Figure 1 with high probability).

Central to our DJ procedure is the use of *influence functions* — a key concept in robust statistics and variational calculus (Cook & Weisberg, 1982; Efron, 1992) — in order to estimate the parameters of models trained on LOO versions of the training data, without exhaustively re-training the model for each held-out data point. That is, using the *von Mises* expansion (Fernholz, 2012) — a variant of Taylor series expansion for statistical functionals — we represent the (counter-factual) model parameters that would have been learned on LOO versions of the training data set in terms of an infinite series of higher-order influence functions (HOIFs) for the model parameters trained on the complete data. To compute the second-order von Mises expansion, we derive an approximate formula for evaluating second-order influence functions that extends on the formula for first-order influence in (Koh & Liang, 2017). We also propose a general procedure for computing HOIFs by *recursively* computing hessian-vector products between the Hessian and higher-order gradients of the model loss, without the need for explicitly inverting the Hessian matrix.

Comprehensive experimental evaluation demonstrates that the DJ performs competitively compared to both Bayesian and non-Bayesian methods with respect to both the coverage and discrimination criteria. Because of the *post-hoc* nature of the DJ, it is capable of improving coverage and discrimination without any modifications to the underlying predictive model. However, since computing influence functions entails at least linear complexity in both the number of training data points and the number of model parameters, a key limitation of our method is scalability. We identify computationally efficient methods for approximating HOIFs as an interesting direction for future research.

2. Preliminaries

2.1. Learning Setup

We consider a standard supervised learning setup with $(\boldsymbol{x}, y)$ being a feature-label pair, where the feature $\boldsymbol{x}$ belongs to a d-dimensional feature space $\mathcal{X} \subseteq \mathbb{R}^d$, and $y \in \mathcal{Y}$. A model is trained to predict y using a dataset $\mathcal{D}_n \triangleq \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ of n examples, which are drawn i.i.d from a distribution $\mathbb{P}$. Let $f(\boldsymbol{x}; \theta) : \mathcal{X} \to \mathcal{Y}$ be the prediction model, where $\theta \in \Theta$ are the model parameters, and Θ is the parameter space. The trained parameters $\hat{\theta} \in \Theta$ are obtained by solving the optimization problem $\hat{\theta} = \arg\min_{\theta \in \Theta} L(\mathcal{D}_n, \theta)$, for a loss

$$L(\mathcal{D}_n, \theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i; \theta)), \tag{1}$$

where we fold in any regularization terms into $\ell(.)$. We do not pose any assumptions on the specific architecture under-

lying the model $f(\boldsymbol{x};\theta)$; it can be any neural network variant, such as feed-forward or convolutional network.

2.2. Uncertainty Quantification

The predictions issued by the (trained) model are given by $f(\boldsymbol{x};\hat{\theta})$; our main goal is to obtain an estimate of uncertainty in the model's prediction, expressed through the pointwise confidence interval $\mathcal{C}(\boldsymbol{x};\hat{\theta})$, formally defined as follows:

$$\mathcal{C}(\boldsymbol{x};\hat{\theta}) \triangleq [\, f_-(\boldsymbol{x};\hat{\theta}),\, f_+(\boldsymbol{x};\hat{\theta})\,],\ \forall \boldsymbol{x} \in \mathcal{X}. \tag{2}$$

The degree of uncertainty in the model's prediction (for a data point with feature $\boldsymbol{x}$) is quantified by the *interval width* $W(.)$ of the confidence interval $\mathcal{C}(\boldsymbol{x};\hat{\theta})$, given by

$$W(\mathcal{C}(\boldsymbol{x};\hat{\theta})) \triangleq f_+(\boldsymbol{x};\hat{\theta}) - f_-(\boldsymbol{x};\hat{\theta}). \tag{3}$$

Wider intervals imply less confidence, and vice versa. For $\mathcal{C}(\boldsymbol{x};\hat{\theta})$ to be usable, it has to satisfy the following:

(i) *Frequentist coverage.* This is satisfied if the confidence interval $\mathcal{C}(\boldsymbol{x};\hat{\theta})$ covers the true target y with a prespecified coverage probability of $(1-\alpha)$, for $\alpha \in (0,1)$, i.e.,

$$\mathbb{P}\left\{ y \in \mathcal{C}(\boldsymbol{x};\hat{\theta}) \right\} \geq 1-\alpha,$$

where the probability is taken with respect to a (new) test point $(\boldsymbol{x}, y)$ as well as with respect to the training data $\mathcal{D}_n$ (Lawless & Fredette, 2005; Barber et al., 2019b).

(ii) *Discrimination.* This requirement is met when $\mathcal{C}(\boldsymbol{x};\hat{\theta})$ is wider for test points with less accurate predictions (Leonard et al., 1992), i.e., for the test points $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, we have

$$\mathbb{E}\big[\, W(\mathcal{C}(\boldsymbol{x};\hat{\theta}))\,\big] \geq \mathbb{E}\big[\, W(\mathcal{C}(\boldsymbol{x}';\hat{\theta}))\,\big] \Leftrightarrow$$
$$\mathbb{E}\big[\, \ell(y, f(\boldsymbol{x};\hat{\theta}))\,\big] \geq \mathbb{E}\big[\, \ell(y', f(\boldsymbol{x}';\hat{\theta}))\,\big],$$

where the expectation $\mathbb{E}[\,.\,]$ is taken with respect to the randomness of $\mathcal{D}_n$. In the next Section, we develop a post-hoc frequentist procedure for estimating $\widehat{\mathcal{C}}(\boldsymbol{x};\hat{\theta})$ that satisfies both of the requirements in (i) and (ii).

3. The Discriminative Jackknife

Before presenting our *discriminative jackknife* (DJ) procedure, we start with a brief recap of the classical jackknife. The jackknife quantifies predictive uncertainty in terms of the (average) prediction error, which is estimated with a leave-one-out (LOO) construction found by systematically leaving out each sample in $\mathcal{D}_n$, and evaluating the error of the re-trained model on the held-out sample, i.e., for a target coverage of $(1-\alpha)$, the naïve jackknife is (Efron, 1992):

$$\widehat{\mathcal{C}}^{J}_{\alpha}(\boldsymbol{x};\hat{\theta}) = f(\boldsymbol{x};\hat{\theta}) \pm \widehat{Q}^{+}_{\alpha}(\mathcal{R}), \tag{4}$$

with $\mathcal{R} = \{r_1, \ldots, r_n\}$, where $r_i = |\, y_i - f(\boldsymbol{x}_i;\hat{\theta}_{-i})\,|$ is the error residual on the i-th data point, $\hat{\theta}_{-i}$ are the parameters of the model re-trained on the dataset $\mathcal{D}_n \setminus \{(\boldsymbol{x}_i, y_i)\}$ (with the i-th point removed), and $\widehat{Q}^{+}_{\alpha}$ is the $(1-\alpha)$ empirical quantile of the set $\mathcal{R} = \{r_1, \ldots, r_n\}$, defined as

$$\widehat{Q}^{+}_{\alpha}(\mathcal{R}) \triangleq \text{ the } \lceil (1-\alpha)\,(n+1) \rceil\text{-th smallest value in } \mathcal{R},$$

where $\widehat{Q}^{-}_{\alpha}(\mathcal{R}) = \widehat{Q}^{+}_{1-\alpha}(-\mathcal{R})$. Albeit intuitive, the naïve jackknife is not guaranteed to achieve the target coverage (Barber et al., 2019b). More crucially, the interval width $W(\widehat{\mathcal{C}}^{J}_{\alpha}(\boldsymbol{x};\hat{\theta}))$ is a constant (independent of $\boldsymbol{x}$), which renders discrimination impossible, i.e., naïve jackknife would result in intervals resembling the leftmost panel in Figure 1.

3.1. Exact Construction of the DJ Confidence Intervals

We construct a generic ameliorated jackknife, the DJ, which addresses the shortcomings of naïve jackknife. We first define some notation. Let the set $\mathcal{V}(\boldsymbol{x})$ be defined as:

$$\mathcal{V}(\boldsymbol{x}) = \{\, v_i(\boldsymbol{x}) \mid \forall i,\, 1 \leq i \leq n \,\}, \tag{5}$$

where $v_i(\boldsymbol{x}) = f(\boldsymbol{x};\hat{\theta}) - f(\boldsymbol{x};\hat{\theta}_{-i})$. Our DJ procedure estimates the predictive confidence interval for a given test point $\boldsymbol{x}$ through the following steps:

$$\widehat{\mathcal{C}}^{DJ}_{\alpha}(\boldsymbol{x};\hat{\theta}) = [\, f_-(\boldsymbol{x};\hat{\theta}),\, f_+(\boldsymbol{x};\hat{\theta})\,],$$
$$f_\gamma(\boldsymbol{x};\hat{\theta}) = \mathcal{G}_{\alpha,\gamma}(\mathcal{R}, \mathcal{V}(\boldsymbol{x})),\ \gamma \in \{-1, +1\},$$

$$\mathcal{R} \Rightarrow \textbf{Marginal Error}, \mathcal{V}(\boldsymbol{x}) \Rightarrow \textbf{Local Variability}, \tag{6}$$

where $\mathcal{G}_{\alpha,\gamma}$ is a quantile function applied on the elements of the sets of *marginal prediction errors* $\mathcal{R}$ and *local prediction variability* $\mathcal{V}$. The marginal prediction error terms use the LOO residuals to estimate the model's generalization error, and the prediction variability term quantifies the extent to which each training data point impacts the value of the prediction at test point $\boldsymbol{x}$. The prediction error is constant, i.e., does not depend on $\boldsymbol{x}$, hence it only contributes to coverage but does not contribute to discrimination. On the contrary, the local variability term depends on $\boldsymbol{x}$, hence it fully determines the discrimination performance. The function $\mathcal{G}_{\alpha,\gamma}$ can be constructed in a variety of ways; here we follow the Jackknife+ construct in (Barber et al., 2019b)

$$\mathcal{G}_{\alpha,\gamma}(\mathcal{R}, \mathcal{V}(\boldsymbol{x})) = \widehat{Q}^{\gamma}_{\alpha}(\{f(\boldsymbol{x};\hat{\theta}) - v_i(\boldsymbol{x}) + \gamma \cdot r_i\}_i) \tag{7}$$

Figure 2 illustrates the construction of the DJ confidence intervals in (6). The confidence intervals are chosen so that the boundaries of the average error and local variability are exceeded by $\lceil (n+1)(1-\alpha) \rceil$ out of the n LOO samples — these are marked with a star. For the average prediction error term, the width of the resulting boundary is the same for any test data point $\boldsymbol{x} \in \mathcal{X}$. For the local prediction variability term, the width of the boundary depends on $\boldsymbol{x}$, and should be wider for less confident predictions, for which the model is vulnerable to the deletion of individual training points.

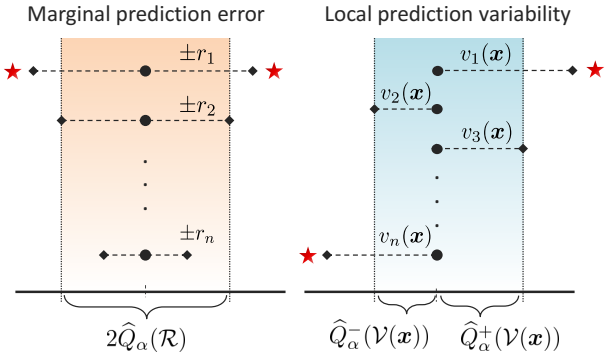

Figure 2. **Illustration of the discriminative jackknife.** Confidence intervals are constructed using the empirical quantiles of the LOO residuals (left) and input-dependent prediction variability (right). Here, we depict the (sorted) elements of $\mathcal{R}$ and $\mathcal{V}$ — elements marked with stars designate the boundaries of the $(1-\alpha)$ quantiles used to compute the DJ confidence intervals in (6).

For the confidence interval's construction in (6), it follows that the DJ interval width can be bounded above by

$$W(\widehat{\mathcal{C}}^{DJ}_\alpha(\boldsymbol{x};\hat{\theta})) \;\leq\; 2\,\widehat{Q}_\alpha(\mathcal{R}) + \sum_\gamma |\widehat{Q}^\gamma_\alpha(\mathcal{V}(\boldsymbol{x}))|. \tag{8}$$

The marginal error and local variability terms in (8) jointly capture two types of uncertainty: *epistemic* and *aleatoric* uncertainties (Gal, 2016). Epistemic uncertainty measures how well the model fits the data, and is reducible as the size of training data n increases. On the contrary, aleatoric uncertainty is the irreducible variance arising from the inherent sources of ambiguity in the data, such as label noise or hidden features (Malinin & Gales, 2018). Consistency of the DJ confidence estimates requires that $W(\widehat{\mathcal{C}}^{DJ}_\alpha(\boldsymbol{x};\hat{\theta})) \to 0$, i.e., the interval width vanishes, as the size of the training data increases ($n \to \infty$). It follows from (8) that if there is no aleatoric uncertainty, and if the underlying model is stable (i.e., $\lim_{n\to\infty} v_i = 0$) and consistent (i.e., $\lim_{n\to\infty} r_i = 0$), then the interval width $W(\widehat{\mathcal{C}}^{DJ}_\alpha(\boldsymbol{x};\hat{\theta}))$ vanishes as n grows asymptotically (more training data is collected).

3.2. Efficient Implementation via Influence Functions

Exact computation of the DJ confidence intervals via (6) requires re-training the model n times in order to collect the "perturbed" LOO parameters $\{\hat{\theta}_{-i}\}_{i=1}^n$. This exhaustive procedure is infeasible for large datasets and complex models. To scale up the DJ, we use *influence functions* — a classic tool from robust statistics (Huber & Ronchetti, 1981; Hampel et al., 2011) — in order to recover the parameters $\{\hat{\theta}_{-i}\}_{i=1}^n$ on the basis of the trained model $f(\boldsymbol{x};\hat{\theta})$, without the need for explicit re-training. Through this implementation, the DJ can be applied in a *post-hoc* fashion, requiring only knowledge of the model loss gradients.

Influence functions enable efficient computation of the effect of a training data point $(\boldsymbol{x}_i, y_i)$ on $\hat{\theta}$. This is achieved by evaluating the change in $\hat{\theta}$ if $(\boldsymbol{x}_i, y_i)$ was up-weighted[1] by some small ϵ, resulting in a new parameter

$$\hat{\theta}_{i,\epsilon} \triangleq \arg\min_{\theta\in\Theta} L(\mathcal{D}_n,\theta) + \epsilon\cdot\ell(y_i, f(\boldsymbol{x}_i;\theta)).$$

The rate of change in $\hat{\theta}$ due to an infinitesimal perturbation ϵ in data point i is give by the (first-order) influence function

$$\mathcal{I}^{(1)}_\theta(\boldsymbol{x}_i, y_i) = \frac{\partial\,\hat{\theta}_{i,\epsilon}}{\partial\,\epsilon}\,\Big|_{\epsilon=0}. \tag{9}$$

Note that the model parameter $\hat{\theta}$ is a *statistical functional* of the data distribution $\mathbb{P}$. Perturbing the i-th training point is equivalent to perturbing $\mathbb{P}$ to create a new distribution $\mathbb{P}_{i,\epsilon} = (1-\epsilon)\,\mathbb{P} + \epsilon\Delta(\boldsymbol{x}_i, y_i)$, where $\Delta(\boldsymbol{x}_i, y_i)$ denotes the Dirac distribution in the point $(\boldsymbol{x}_i, y_i)$. In this sense, the influence function in (9) operationalizes the concept of derivatives to statistical functionals, i.e., the derivative of the parameters $\hat{\theta}$ with respect to the data distribution $\mathbb{P}$.

By recognizing that influence functions are the "derivatives" of $\hat{\theta}$ with respect to $\mathbb{P}$, we can use a Taylor-type expansion to represent the counter-factual model parameter $\hat{\theta}_{i,\epsilon}$ (that would have been learned from a dataset with the i-th data point up-weighted) in terms of the parameter $\hat{\theta}$ (learned from the complete $\mathcal{D}_n$) as follows (Robins et al., 2008):

$$\hat{\theta}_{i,\epsilon} = \hat{\theta} + \epsilon\cdot\mathcal{I}^{(1)}_\theta(\boldsymbol{x}_i, y_i) + \frac{\epsilon^2}{2!}\cdot\mathcal{I}^{(2)}_\theta(\boldsymbol{x}_i, y_i) + \;\ldots \tag{10}$$

where $\mathcal{I}^{(k)}_\theta(\boldsymbol{x}_i, y_i)$ is the k-th order influence function, defined as $\mathcal{I}^{(k)}_\theta(\boldsymbol{x}_i, y_i) = \partial^k\,\hat{\theta}_{i,\epsilon}/\partial\,\epsilon^k\,|_{\epsilon=0}$. The expansion in (10), known as the *von Mises* expansion (Fernholz, 2012), is a distributional analog of the Taylor expansion for statistical functionals. If all of the higher-order influence functions (HOIFs) in (10) exist, then we can recover $\hat{\theta}_{i,\epsilon}$ without re-training the model on the perturbed training dataset. Since exact reconstruction of $\hat{\theta}_{i,\epsilon}$ requires an infinite number of HOIFs, we can only approximate $\hat{\theta}_{i,\epsilon}$ by including a finite number of HOIF terms from the von Mises expansion.

The LOO model parameters $\{\hat{\theta}_{-i}\}_{i=1}^n$, required for the construction of the DJ confidence intervals, can be obtained by setting $\epsilon = -1/n$, i.e., $\hat{\theta}_{-i} = \hat{\theta}_{i,\frac{-1}{n}}$, since removing a training point is equivalent to up-weighting it by $-1/n$ in the loss function $L(\mathcal{D}_n;\theta)$. Thus, by setting $\epsilon = -1/n$ and selecting a prespecified number of HOIF terms m for obtaining the approximate LOO parameters $\hat{\theta}^{(m)}_{-i}$, the DJ confidence intervals can be computed using the steps in Algorithm 1.

3.3. Computing Influence Functions

The recent work on model interpretability in (Koh & Liang, 2017) has studied the usage of influence functions to quan-

[1] A detailed technical background on influence functions and its connection with the jackknife is provided in Appendix A.

tify the impact of individual data points on model training. There, first-order influence was computed, using a classical result in (Cook & Weisberg, 1982), as follows:

$$\mathcal{I}_\theta^{(1)}(\boldsymbol{x}, y) = -H_\theta^{-1} \cdot \nabla_\theta \, \ell(y, f(\boldsymbol{x}, \theta)), \qquad (11)$$

where $H_\theta \triangleq \nabla_\theta^2 \sum_i \ell(y_i, f(\boldsymbol{x}_i, \theta))$ is the Hessian of the loss function, which is assumed to be positive definite. We derive an approximate expression for the second order influence function in terms of the first order influence as follows:

$$\mathcal{I}_\theta^{(2)}(\boldsymbol{x}, y) \approx -2H_\theta^{-1} \cdot \mathcal{I}_\theta^{(1)}(\boldsymbol{x}, y) \cdot \nabla_\theta^2 \, \ell(y, f(\boldsymbol{x}, \theta)). \quad (12)$$

In general, it can be shown that HOIFs can be recursively represented in terms of lower-order influence and loss gradients as (Giordano et al., 2019; Debruyne et al., 2008)

$$\mathcal{I}_\theta^{(k+1)} = -H_\theta^{-1}\Big(\sum_{m=1}^{k} g_m\big(\{\mathcal{I}_\theta^{(j)}, \nabla_\theta^j \, \ell_\theta\}_{j=1}^m\big)\Big), \quad (13)$$

for some functions $\{g_m\}_m$. Here, we used short-hand notation for influence functions and loss gradients, dropping the dependency on $(\boldsymbol{x}, y)$. HOIFs exist if $\ell(.)$ is differentiable and locally convex in the neighborhood of θ, and $H_\theta \succeq 0$. In practice, we found that the second order terms are sufficient for obtaining an accurate estimate of the re-trained model parameters. The derivation of the second-order influence function in (12) is provided in Appendix B.

Computing HOIFs. On the positive side, (13) shows that we need to compute the inverse Hessian H_θ^{-1} only once for all HOIFs. However, for a model with p parameters, this is still a bottleneck operation with $\mathcal{O}(p^3)$ complexity. To address this hurdle, we capitalize on the recursive structure of (13) and the hessian-vector products approach in (Pearlmutter, 1994) to efficiently compute HOIFs as follows. To evaluate the $(k+1)$-th order influence given our estimate of k-th influence $\widetilde{\mathcal{I}}_k$, we execute the following steps:

(Step 1) Compute the k-th order loss gradient $\nabla_\theta^k \ell_\theta$.

(Step 2) Evaluate $w = \sum_{m=1}^k g_m\big(\{\widetilde{\mathcal{I}}_\theta^{(j)}, \nabla_\theta^j \, \ell_\theta\}_{j=1}^m\big)$.

(Step 3) Sample t data points $\{(\boldsymbol{x}_{s_i}, y_{s_i})\}_{i=1}^t$ from $\mathcal{D}_n$.

(Step 4) Initialize $\widetilde{H}_{0,\theta}^{-1} w = w$, and recursively compute:

$$\widetilde{H}_{j,\theta}^{-1} w = w + (\mathbf{I} - \nabla_\theta^2 \ell_\theta) \cdot \widetilde{H}_{j-1,\theta}^{-1} w,$$

for $j \in \{0, \ldots, t\}$, where $\widetilde{H}_{j,\theta}^{-1} \triangleq \sum_{i=o}^{j} (\mathbf{I} - \widetilde{H}_\theta)^i$, and $\widetilde{H}_\theta$ is the stochastic estimate of the Hessian computed over the sampled t data points in $\{(\boldsymbol{x}_{s_i}, y_{s_i})\}_{i=1}^t$.

(Step 5) Return $\widetilde{\mathcal{I}}_{k+1} = \widetilde{\mathcal{I}}_k - \widetilde{H}_{t,\theta}^{-1} w$.

As shown through the steps above, the recursive nature of HOIFs allow us to reuse much of the computations involved in evaluating lower-order influence in computing higher-order terms. The stochastic estimation process above is motivated by the power series expansion of matrix inversion, and converges if $H_\theta \succeq 1$, which can always be ensured via appropriate scaling of the loss. We approximate the higher order loss gradients in Step 1 using coordinate-wise gradients, thus, for computing m HOIFs, the overall complexity of the procedure above is linear in n, m and p, i.e., $\mathcal{O}(npm)$.

Despite the reduction in computational complexity, the proposed approximate procedure still entails a linear complexity in both the number of training data points n and the number of model parameters p. This computational bottleneck limits our post-hoc procedure to relatively small networks, hence we regard our method's inability to *scale* as its key limitation. Devising efficient methods for approximating the Hessian is an interesting direction for future research.

Algorithm 1 The Discriminative Jackknife

Input: Learned parameter $\hat{\theta}$, influence order m, coverage α, training data $\mathcal{D}_n$, test point $\boldsymbol{x}$.
Output: DJ confidence interval $\widehat{\mathcal{C}}_\alpha^{DJ}(\boldsymbol{x}; \hat{\theta}, m)$.

for $i = 1$ **to** n **do**
  $\hat{\theta}_{-i}^{(m)} \leftarrow \hat{\theta} - \sum_{k=1}^m (n^{-k}/k!) \cdot \mathcal{I}_\theta^{(k)}(\boldsymbol{x}_i, y_i)$.
  $r_i \leftarrow \big| \, y_i - f(\boldsymbol{x}_i; \hat{\theta}_{-i}^{(m)}) \, \big|$.
  $v_i(\boldsymbol{x}) \leftarrow f(\boldsymbol{x}; \hat{\theta}) - f(\boldsymbol{x}; \hat{\theta}_{-i}^{(m)})$.
end for
$f_-(\boldsymbol{x}; \hat{\theta}) \leftarrow \widehat{Q}_\alpha^-(\{f(\boldsymbol{x}; \hat{\theta}) - v_i(\boldsymbol{x}) + \gamma \cdot r_i\}_i)$.
$f_+(\boldsymbol{x}; \hat{\theta}) \leftarrow \widehat{Q}_\alpha^+(\{f(\boldsymbol{x}; \hat{\theta}) - v_i(\boldsymbol{x}) + \gamma \cdot r_i\}_i)$.
Return $\widehat{\mathcal{C}}_{\alpha,n}^{DJ}(\boldsymbol{x}; \hat{\theta}, m) \leftarrow [\, f_-(\boldsymbol{x}; \hat{\theta}), f_+(\boldsymbol{x}; \hat{\theta}) \,]$.

3.4. Theoretical Guarantees

We conclude this Section by revisiting the design requirements in Section 2. In the following Theorem, we show that the DJ provides a guarantee on the coverage condition.

Theorem 1. *For any model $f(\boldsymbol{x}; \hat{\theta})$, the coverage probability achieved by the approximate DJ with $m \to \infty$ is*

$$\mathbb{P}\Big\{ y \in \widehat{\mathcal{C}}_\alpha^{DJ}(\boldsymbol{x}; \hat{\theta}, \infty) \Big\} \;\geq\; (1 - 2\alpha). \quad \square$$

Theorem 1 provides a strong, model-independent guarantee on the frequentist coverage of the DJ confidence intervals. In Section 5, we show through empirical evaluation that in practice — even with second order influence terms only — the DJ intervals will achieve the target $(1 - \alpha)$ coverage. With further assumptions on the algorithmic stability of the underlying model, it can also be shown that the exact DJ satisfies the discrimination condition in Section 2.2 (See Appendix C of the supplementary material).

4. Related Work

Post-hoc methods for uncertainty quantification have been traditionally underexplored since existing approaches, such as calibration via temperature scaling (Platt et al., 1999), were known to under-perform compared to built-in methods (Ovadia et al., 2019). However, recent works have revived post-hoc approaches using ideas based on bootstrapping (Schulam & Saria, 2019), jackknife resampling (Barber et al., 2019b; Giordano et al., 2018) and cross-validation (Vovk et al., 2018; Barber et al., 2019a). An overview of the different classes of post-hoc and built-in methods proposed in recent literature is provided in Table 1.

Table 1. Overview of existing uncertainty quantification methods.

Method	**Bayesian / Frequentist**	**Post-hoc / Built-in**	**Coverage**
Bayesian neural nets	Bayesian	Built-in	None
Prob. backprop.	Bayesian	Built-in	None
Monte Carlo dropout	Bayesian	Built-in	None
Deep ensembles	Frequentist	Built-in	None
RUE	Frequentist	Post-hoc	None
DJ	Frequentist	Post-hoc	$1-2\alpha$

Bayesianism is the dominant approach to uncertainty quantification in deep learning (Welling & Teh, 2011; Hernández-Lobato & Adams, 2015; Ritter et al., 2018; Maddox et al., 2019). A post-hoc application of Bayesian methods is not possible since by their very nature, Bayesian models require specifying priors over model parameters, which leads to major modifications in the inference algorithms. While Bayesian models provide a formal framework for uncertainty estimation, posterior credible intervals do not guarantee frequentist coverage, and more crucially, the achieved coverage can be very sensitive to hyper-parameter tuning (Bayarri & Berger, 2004). Moreover, exact Bayesian inference is computationally prohibitive, and alternative approximations — e.g., (Gal & Ghahramani, 2016) — may induced posterior distributions that do not concentrate asymptotically (Osband, 2016; Hron et al., 2017).

Deep ensembles (Lakshminarayanan et al., 2017) are regarded as the most competitive (non-Bayesian) benchmark for uncertainty estimation (Ovadia et al., 2019). This method repeatedly re-trains the model on sub-samples of the data (using adversarial training), and then estimates uncertainty through the variance of the aggregate predictions. A similar bootstrapping approach developed in (Schulam & Saria, 2019), dubbed resampling uncertainty estimation (RUE), uses the model's Hessian and loss gradients to create an ensemble without re-training. While these methods may perform favorably in terms of discrimination, they are likely to undercover, since they only consider local variability terms akin to those in (6). Additionally, ensemble methods do not provide theoretical guarantees on their performance.

The (infinitesimal) jackknife method was previously used for quantifying the predictive uncertainty in random forests (Wager et al., 2014; Mentch & Hooker, 2016; Wager & Athey, 2018). In these works, however, the developed jackknife estimators are bespoke to bagging predictors, and cannot be straightforwardly extended to deep neural networks. More recently, general-purpose jackknife estimators were developed in (Barber et al., 2019b), where two exhaustive leave-one-out procedures: the *jackknife+* and the *jackknife-minmax* where shown to have assumption-free worst-case coverage guarantees of $(1-2\alpha)$ and $(1-\alpha)$, respectively. Our work improves on these results by alleviating the need for exhaustive leave-one-out re-training.

5. Experiments

In this Section, we evaluate the DJ using synthetic and real data, and compare its performance with various baselines. Further experimental details are deferred to Appendix D.

Baselines. We compared our DJ method with 4 state of the art baselines. These included 3 built-in Bayesian methods: (1) Monte Carlo Dropout (MCDP) (Gal & Ghahramani, 2016), (2) Probabilistic backpropagation (PBP) (Hernández-Lobato & Adams, 2015), and (3) Bayesian neural networks with inference via stochastic gradient Langevin dynamics (BNN-SGLD) (Welling & Teh, 2011). In addition, we considered deep ensembles (DE) (Lakshminarayanan et al., 2017), which is deemed the most competitive built-in frequentist method (Ovadia et al., 2019). For a target coverage of $(1-\alpha)$, uncertainty estimates were obtained by setting posterior quantile functions to $(1-\alpha)$ (for Bayesian methods), or obtaining the $(1-\alpha)$ percentile point of a normal distribution (for frequentist methods). Details on the implementation, hyper-parameter tuning and uncertainty calibration of all baselines are provided in Appendix D.

Evaluation metrics. In all experiments, we used the mean squared error (MSE) as the loss $L(\mathcal{D}_n, \hat{\theta})$ for training the model $f(\boldsymbol{x}; \hat{\theta})$. To ensure fair comparisons, the hyperparameters of the underlying neural network $f(\boldsymbol{x}; \hat{\theta})$ were fixed for all baselines. In each experiment, the uncertainty estimate $\widehat{\mathcal{C}}_\alpha(\boldsymbol{x}; \hat{\theta})$ is obtained from a training sample, and then coverage and discrimination are evaluated on a test sample. To evaluate empirical coverage probability, we compute the fraction of test samples for which y resides in $\widehat{\mathcal{C}}_\alpha(\boldsymbol{x}; \hat{\theta})$. Discrimination was evaluated as follows. For each baseline method, we evaluate the interval width $W(\widehat{\mathcal{C}}_\alpha(\boldsymbol{x}; \hat{\theta}))$ for all test points. For a given error threshold $\mathcal{E}$, we use the interval width to detect whether the test prediction $f(\boldsymbol{x}; \hat{\theta})$ is a high-confidence, i.e., $\ell(y, f(\boldsymbol{x}; \theta)) \leq \mathcal{E}$, or low-confidence prediction, i.e., $\ell(y, f(\boldsymbol{x}; \theta)) > \mathcal{E}$. We use the area under the precision-recall curve (AUPRC) — also known as the average precision score — in order to evaluate the accuracy of this confidence classification task.

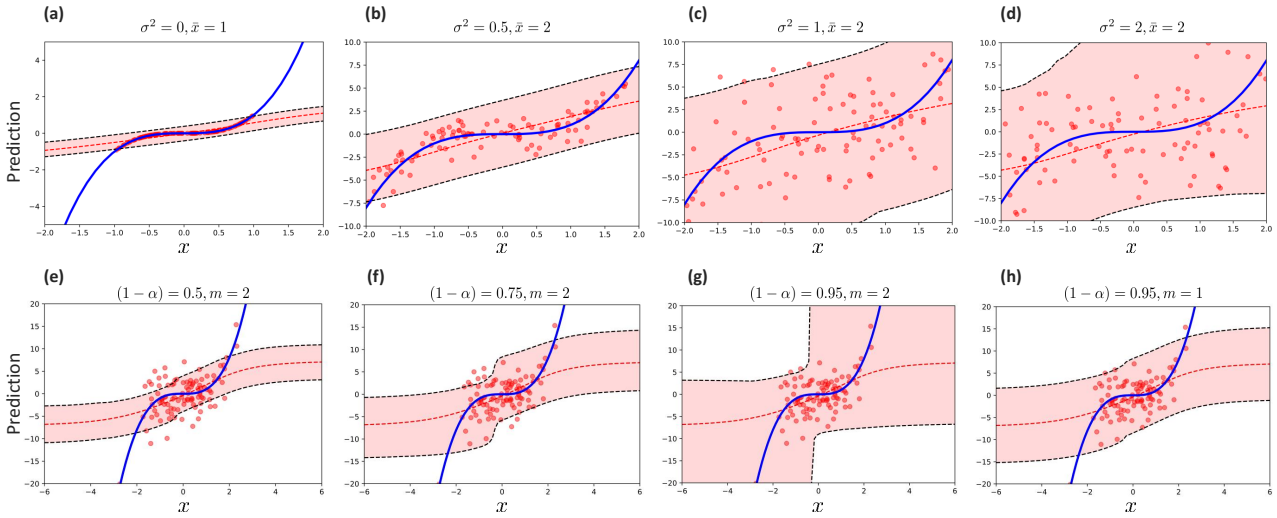

Figure 3. **DJ confidence intervals in a one-dimensional feature space. (a)** Uniform feature distribution with $\bar{x} = 1$ and no aleatoric noise. **(b)** For a Uniform feature distribution with $\bar{x} = 2$ and noise variance $\sigma^2 = 0.5$, the DJ confidence intervals are wider than those in (a). The confidence intervals are of a fixed width for all x because the training points are uniformly distributed everywhere in the feature space. **(c)** For a Uniform feature distribution with $\bar{x} = 2$ and noise variance $\sigma^2 = 1$, the DJ confidence intervals are wider than (b) and some of the training data points are not covered because of the high noise variance. **(d)** Uniform feature distribution with $\bar{x} = 2$ and noise variance $\sigma^2 = 2$. **(e)** Normal feature distribution with noise variance $\sigma^2 = 1$ and target coverage $(1 - \alpha) = 0.5$. **(f)** Normal feature distribution with noise variance $\sigma^2 = 1$ and target coverage $(1 - \alpha) = 0.75$. Because of the stricter coverage target, the DJ confidence intervals are wider than those in (e). **(g)** Normal feature distribution with noise variance $\sigma^2 = 1$ and target coverage $(1 - \alpha) = 0.95$. Because the normal feature distribution is associated with epistemic uncertainty, the width of the confidence interval is not uniform for the different values of x. **(h)** DJ confidence intervals with first-order influence functions ($m = 1$) for the same setting in (g). Here, we can see that the DJ confidence intervals exhibit a fixed width for all values of the feature x, and do not capture epistemic uncertainty as in (g).

5.1. Synthetic Data

We start off by illustrating the DJ confidence intervals using data generated from the following synthetic model. In particular, we use the synthetic model introduced in (Hernández-Lobato & Adams, 2015), defined as follows:

$$y = x^3 + \epsilon, \tag{14}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and σ^2 is the noise variance. We consider two possible feature distributions:

$$\begin{aligned} &\text{Uniform feature distribution} : x \sim \mathrm{U}([-\bar{x}, \bar{x}]), \\ &\text{Normal feature distribution} : x \sim \mathcal{N}(0, \bar{\sigma}_x^2). \end{aligned} \tag{15}$$

Under the uniform distribution, the model will be equally uncertain about its predictions for any x since all feature instances have the same amount of noise and frequency of observations. In this case, the main source of uncertainty is the aleatoric uncertainty resulting from the noise ϵ. On the contrary, under the normal distribution, the model will be more uncertain in predictions made for values of x that deviate from 0. This is because most of the training data will be concentrated around 0, leading to an increased epistemic uncertainty for very large or very small values of x. In all experiments, we fit a 2-layer feed-forward neural network with 100 hidden units and compute the DJ confidence intervals using the post-hoc procedure in Algorithm 1.

Results. In Figure 3, we depict various samples of the DJ confidence intervals for different feature distributions, target coverage levels, and noise variances. In Figures 3 (a) to (d), we show the confidence intervals issued by a model trained under the uniform feature distribution: here, we can see that the interval width does not vary significantly for the different values of x, because the training points are uniformly distributed everywhere in the feature space. Moreover, the interval width increases as the noise variance increases.

In Figures 3 (e) to (h), we show the DJ confidence intervals issued by a model trained under the normal feature distribution. Here, we see that the interval width is narrowest around $x = 0$, i.e., the point around which most training points are concentrated. We also see that the inclusion of the second-order influence terms enriches the shape of the confidence intervals in a way that reflects the ground-truth epistemic uncertainty (see Figures 3 (g) and (h)).

Finally, in Figure 4, we show the average width of the DJ confidence intervals (averaged over 100 test points across 10 simulations) and the achieved coverage for a neural network model trained using $n = 100$ training points with varying

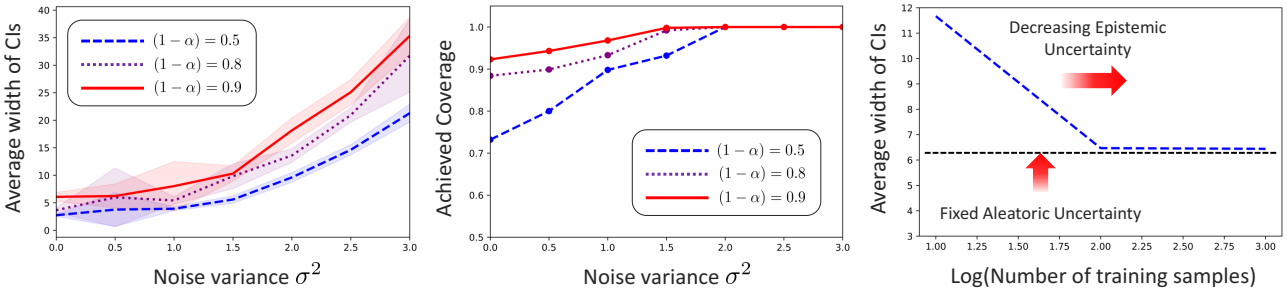

Figure 4. Average width of the DJ confidence intervals and achieved coverage at varying levels of aleatoric and epistemic uncertainty.

levels of ground-truth aleatoric uncertainty (i.e., varying noise variance σ^2). As we can see, the width of the confidence intervals increase for larger noise variances (reflecting higher levels of reported uncertainty) and for more strict target coverage $(1-\alpha)$ (Figure 4, left). For all noise variances, the DJ confidence intervals manage to achieve the target coverage levels (Figure 4, middle). By changing the number of training points with a fixed noise variance, we control for the amount of epistemic uncertainty: as we can see in Figure 4 (right), the width of the confidence intervals decrease as the training data increases, until it hits a floor dictated by the inherent aleatoric uncertainty in the labels.

5.2. Real Data: Auditing Model Reliability

In this Section, we conduct a series of experiments on real-world datasets in order to evaluate the accuracy of uncertainty estimates issued by the DJ procedure. In particular, we show how uncertainty estimates can be used to audit the reliability of a model using experiments on 4 UCI benchmark datasets for regression: yacht hydrodynamics (Yacht), Boston housing (Housing), energy efficiency (Energy) and naval propulsion (Naval) (Dua & Graff, 2017).

In each experiment, we use 80% of the data for training and 20% for testing. We use a 2-layer neural network model with 100 hidden units, Tanh activation functions, MSE loss, and a single set of learning hyper-parameters for all baselines (1000 epochs with 100 samples per minibatch, and an Adam optimizer with default settings). We set the target coverage to $(1-\alpha) = 0.9$. On each test set, we evaluate the model's MSE, achieved coverage rate and AUPRC in predicting whether the model's test error exceeds a threshold $\mathcal{E}$ that is set to be 90% percentile of the empirical distribution over test errors. Results are provided in Table 2.

We observe that, by virtue of its post-hoc nature, the DJ procedure yields the best predictive performance (MSE) on almost all baselines. This is because the DJ does not interfere with the model training or compromise its accuracy, and is only applied on an already trained model that is optimized to minimize the MSE. The post-hoc nature of our method does not compromise the accuracy of its uncertainty intervals. Across all data sets, the DJ achieves the desired target coverage, whereas other Bayesian methods (e.g., BNN and PBP) tend to under-cover the true labels. Moreover, DJ provides high AUPRC scores on all data sets, and on data sets were its AUPRC scores are lower than the other coverage-achieving baselines (e.g., MCDP), it offers a much better predictive accuracy in terms of the MSE.

	Dataset			
Method	**Yacht**	**Housing**	**Energy**	**Kin8nm**
DJ	0.87 ± 0.05 $(95.9\%)^*$ 26.55	0.80 ± 0.04 $(99.8\%)^*$ 33.87	0.77 ± 0.08 $(98.11\%)^*$ 11.06	0.88 ± 0.01 $(93.77\%)^*$ 0.00
MCDP	0.67 ± 0.06 $(100.0\%)^*$ 150.93	0.86 ± 0.00 $(99.6\%)^*$ 113.04	0.84 ± 0.03 $(100.0\%)^*$ 92.57	0.83 ± 0.03 (100.0%) 0.05
PBP	0.66 ± 0.06 (70.4%) 22.21	0.85 ± 0.03 (5.0%) 221.11	0.84 ± 0.04 (10.1%) 201.73	0.82 ± 0.04 (89.9%) 0.62
DE	0.87 ± 0.04 (0.0%) 327.74	0.62 ± 0.04 (19.4%) 61.82	0.80 ± 0.09 (23.6%) 21.53	0.82 ± 0.02 (21.83%) 0.03
BNN	0.81 ± 0.05 (82.3%) 317.94	0.88 ± 0.00 (89.0%) 118.89	—	0.89 ± 0.00 (100.0%) 0.68

Table 2. **Auditing predictive model reliability.** AUPRC performance ($\pm$ 95% confidence intervals) of all baselines on the real-world UCI regression datasets. In each cell, the first line contains the AUPRC score, the second line contains the achieved (empirical) coverage and the third line lists the MSE loss. Coverage rates marked with an asterisk achieve the desired 90% target rate. Blank entries correspond to models with confidence intervals that perform no better than random guessing with respect to the AUPRC.

6. Conclusion

Uncertainty quantification is a crucial requirement in various high-stakes applications, wherein deep learning can inform critical decision-making. In this paper, we introduced a rig-

orous frequentist procedure for quantifying the uncertainty in predictions issued by deep learning models in a post-hoc fashion. Our procedure, which is inspired by classical jackknife estimators, does not require any modifications in the underlying deep learning model, and provides theoretical guarantees on its achieved performance. Because of its post-hoc and model-agnostic nature, this procedure can be applied to a wide variety of models ranging from feed-forward networks to convolutional and recurrent networks. While our focus was mainly on deep learning models, our procedure can also be applied to general machine learning models.

A key ingredient of our procedure is the usage of influence functions to reconstruct leave-one-out model parameters without the need for explicit re-training. Influence functions provide a powerful tool for constructing ensembles of models in a post-hoc fashion that can be used to assess model variability without the need for built-in designs for uncertainty intervals. While we present an algorithm that recursively computes influence functions with a complexity that is linear in the number of model parameters and size of training data, our procedure is still limited to relatively small networks or small data sets. Developing methods for fast computation of the Hessian matrix that would scale up our method to more complex networks and larger data sets is a valuable direction for future research.

Acknowledgments

The authors would like to thank the reviewers for their helpful comments. This work was supported by the US Office of Naval Research (ONR) and the National Science Foundation (NSF grants 1524417 and 1722516).

References

Alaa, A. M. and van der Schaar, M. Bayesian inference of individualized treatment effects using multi-task gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 3424–3432, 2017.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Barber, R. F., Candes, E. J., Ramdas, A., and Tibshirani, R. J. Conformal prediction under covariate shift. *arXiv preprint arXiv:1904.06019*, 2019a.

Barber, R. F., Candes, E. J., Ramdas, A., and Tibshirani, R. J. Predictive inference with the jackknife+. *arXiv preprint arXiv:1905.02928*, 2019b.

Bayarri, M. J. and Berger, J. O. The interplay of bayesian and frequentist analysis. *Statistical Science*, pp. 58–80, 2004.

Bousquet, O. and Elisseeff, A. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

Cook, R. D. and Weisberg, S. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

Debruyne, M., Hubert, M., and Suykens, J. A. Model selection in kernel based regression using the influence function. *Journal of Machine Learning Research*, 9(Oct): 2377–2400, 2008.

Devroye, L. and Wagner, T. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25(2):202–207, 1979.

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

Dusenberry, M. W., Tran, D., Choi, E., Kemp, J., Nixon, J., Jerfel, G., Heller, K., and Dai, A. M. Analyzing the role of model uncertainty for electronic health records. *arXiv preprint arXiv:1906.03842*, 2019.

Efron, B. Jackknife-after-bootstrap standard errors and influence functions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 54(1):83–111, 1992.

Fernholz, L. T. *Von Mises calculus for statistical functionals*, volume 19. Springer Science & Business Media, 2012.

Gal, Y. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pp. 1050–1059, 2016.

Giordano, R., Stephenson, W., Liu, R., Jordan, M. I., and Broderick, T. A swiss army infinitesimal jackknife. *arXiv preprint arXiv:1806.00550*, 2018.

Giordano, R., Jordan, M. I., and Broderick, T. A higher-order swiss army infinitesimal jackknife. *arXiv preprint arXiv:1907.12116*, 2019.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons, 2011.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning (ICML)*, pp. 1861–1869, 2015.

Hron, J., Matthews, A. G. d. G., and Ghahramani, Z. Variational gaussian dropout is not bayesian. *arXiv preprint arXiv:1711.02989*, 2017.

Huber, P. J. and Ronchetti, E. M. Robust statistics john wiley & sons. *New York*, 1(1), 1981.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1885–1894. JMLR. org, 2017.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6402–6413, 2017.

Lawless, J. and Fredette, M. Frequentist prediction intervals and predictive distributions. *Biometrika*, 92(3):529–542, 2005.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436, 2015.

Leonard, J., Kramer, M. A., and Ungar, L. A neural network architecture that computes its own reliability. *Computers & chemical engineering*, 16(9):819–835, 1992.

Maddox, W., Garipov, T., Izmailov, P., Vetrov, D., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. *arXiv preprint arXiv:1902.02476*, 2019.

Malinin, A. and Gales, M. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pp. 7047–7058, 2018.

Mentch, L. and Hooker, G. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research (JMLR)*, 17(1):841–881, 2016.

Miller, R. G. The jackknife-a review. *Biometrika*, 61(1): 1–15, 1974.

Osband, I. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Pearlmutter, B. A. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Platt, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. 2018.

Robins, J., Li, L., Tchetgen, E., van der Vaart, A., et al. Higher order influence functions and minimax estimation of nonlinear functionals. In *Probability and statistics: essays in honor of David A. Freedman*, pp. 335–421. Institute of Mathematical Statistics, 2008.

Schulam, P. and Saria, S. Can you trust this prediction? auditing pointwise reliability after learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1022–1031, 2019.

Vovk, V., Nouretdinov, I., Manokhin, V., and Gammerman, A. Cross-conformal predictive distributions. In *The Journal of Machine Learning Research (JMLR)*, pp. 37–51, 2018.

Wager, S. and Athey, S. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Wager, S., Hastie, T., and Efron, B. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *The Journal of Machine Learning Research (JMLR)*, 15(1):1625–1651, 2014.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

White, M. and White, A. Interval estimation for reinforcement-learning algorithms in continuous-state domains. In *Advances in Neural Information Processing Systems*, pp. 2433–2441, 2010.

Appendix

A. Influence Functions: Background & Key Concepts

A.1. Formal Definition

Robust statistics is the branch of statistics concerned with the detection of outlying observations. An estimator is deemed *robust* if it produces similar results as the majority of observations indicates, regardless of how a minority of other observations is perturbed ((Huber & Ronchetti, 1981)). The influence function measures these effects in statistical functionals by analyzing the behavior of a functional not only at the distribution of interest, but also in an entire neighborhood of distributions around it. Lack of model robustness is a clear indicator of model uncertainty, and hence influence functions arise naturally in our method as a (pointwise) surrogate measure of model uncertainty. In this section we formally define influence functions and discuss its properties.

The pioneering works in ((Hampel et al., 2011)) and ((Huber & Ronchetti, 1981)) coined the notion of influence functions to assess the robustness of statistical functionals to perturbations in the underlying distributions. Consider a statistical functional $T : \mathcal{P} \to \mathbb{R}$, defined on a probability space $\mathcal{P}$, and a probability distribution $\mathbb{P} \in \mathcal{P}$. Consider distributions of the form $\mathbb{P}_{\varepsilon,z} = (1-\varepsilon)\mathbb{P} + \varepsilon\Delta z$ where Δz denotes the Dirac distribution in the point $z = (\boldsymbol{x}, y)$, representing the contaminated part of the data. For the functional T to be considered robust, $T(\mathbb{P}_{\varepsilon,z})$ should not be too far away from $T(\mathbb{P})$ for any possible z and any small ε. The limiting case of $\varepsilon \to 0$ defines the influence function. That is, Then the influence function of T at $\mathbb{P}$ in the point z is defined as

$$\mathcal{I}(z;\mathbb{P}) = \lim_{\varepsilon\to 0} \frac{T(\mathbb{P}_{\varepsilon,z}) - T(\mathbb{P})}{\varepsilon} \triangleq \frac{\partial}{\partial\varepsilon} T(\mathbb{P}_{\varepsilon,z}) \bigg|_{\varepsilon=0}, \tag{1}$$

The influence function measures the robustness of T by quantifying the effect on the estimator T when adding an infinitesimally small amount of contamination at the point z. If the supremum of $\mathcal{I}(.)$ over z is bounded, then an infinitesimally small amount of perturbation cannot cause arbitrary large changes in the estimate. Then small amounts of perturbation cannot completely change the estimate which ensures the robustness of the estimator.

A.2. The von Mises Expansion

The von Mises expansion is a distributional analog of the Taylor expansion applied for a functional instead of a function. For two distributions $\mathbb{P}$ and $\mathbb{Q}$, the Von Mises expansion is ((Fernholz, 2012)):

$$T(\mathbb{Q}) = T(\mathbb{P}) + \int \mathcal{I}^{(1)}(z;\mathbb{P})\, d(\mathbb{Q}-\mathbb{P}) + \frac{1}{2}\int \mathcal{I}^{(2)}(z;\mathbb{P})\, d(\mathbb{Q}-\mathbb{P}) + \ldots, \tag{2}$$

where $\mathcal{I}^{(k)}(z;\mathbb{P})$ is the k^{th} order influence function. By setting $\mathbb{Q}$ to be a perturbed version of $\mathbb{P}$, i.e., $\mathbb{Q} = \mathbb{P}_\varepsilon$, the von Mises expansion at point z reduces to:

$$T(\mathbb{P}_{\varepsilon,z}) = T(\mathbb{P}) + \varepsilon\,\mathcal{I}^{(1)}(z;\mathbb{P}) + \frac{\varepsilon^2}{2}\mathcal{I}^{(2)}(z;\mathbb{P}) + \ldots, \tag{3}$$

and so the k^{th} order influence function is operationalized through the derivative

$$\mathcal{I}^{(k)}(z;\mathbb{P}) \triangleq \frac{\partial}{\partial^k\varepsilon} T(\mathbb{P}_{\varepsilon,z}) \bigg|_{\varepsilon=0}. \tag{4}$$

A.3. Influence Function of Model Loss

Now we apply the mathematical definitions in Sections A.1 and A.2 to our learning setup. In our setting, the functional $T(.)$ corresponds to the (trained) model parameters $\hat{\theta}$ and the distribution $\mathbb{P}$. In this case, influence functions of $\hat{\theta}$ computes how much the model parameters would change if the underlying data distribution was perturbed infinitesimally.

$$\mathcal{I}^{(1)}_\theta(z) = \frac{\partial\,\hat{\theta}_{\varepsilon,z}}{\partial\,\varepsilon}\bigg|_{\varepsilon=0}, \quad \hat{\theta}_{\varepsilon,z} \triangleq \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \ell(z_i;\theta) + \varepsilon\,\ell(z;\theta). \tag{5}$$

Recall that in the definition of the influence function $\mathbb{P}_{\varepsilon,z} = (1-\varepsilon)\mathbb{P} + \varepsilon\Delta z$ where Δz denotes the Dirac distribution in the point $z = (\boldsymbol{x}, y)$. Thus, the (first-order) influence function in (5) corresponds to perturbing a training data point z by an infinitesimally small change ε and evaluating the corresponding change in the learned model parameters $\hat{\theta}$. More generally, the k^{th} order influence function of $\hat{\theta}$ is defined as follows:

$$\mathcal{I}_{\theta}^{(k)}(z) = \left. \frac{\partial^k \hat{\theta}_{\varepsilon,z}}{\partial\, \varepsilon^k} \right|_{\varepsilon=0}. \tag{6}$$

By applying the von Mises expansion, we can approximate the parameter of a model trained on the training dataset with perturbed data point z as follows:

$$\hat{\theta}_{\varepsilon,z} \approx \hat{\theta} + \varepsilon\, \mathcal{I}_{\theta}^{(1)}(z) + \frac{\varepsilon^2}{2}\, \mathcal{I}_{\theta}^{(2)}(z) \; + \; \ldots \; + \frac{\varepsilon^m}{m!}\, \mathcal{I}_{\theta}^{(m)}(z), \tag{7}$$

where m is the number of terms included in the truncated expansion. When $m = \infty$, the exact parameter $\hat{\theta}_{\varepsilon,z}$ without the need to re-train the model.

A.4. Connection to leave-one-out estimators

Our uncertainty estimator depends on perturbing the model parameters by removing a single training point at a time. Note that removing a point z is the same as perturbing z by $\varepsilon = \frac{-1}{n}$, hence we obtain an ($m^{th}$ order) approximation of the parameter change due to removing the point z as follows:

$$\hat{\theta}_{-z} - \hat{\theta} \approx \frac{-1}{n}\, \mathcal{I}_{\theta}^{(1)}(z) + \frac{1}{2n^2}\, \mathcal{I}_{\theta}^{(2)}(z) + \; \ldots \; + \frac{(-1)^m}{n^m \cdot m!}\, \mathcal{I}_{\theta}^{(m)}(z), \tag{8}$$

where $\hat{\theta}_{-z}$ is the model parameter learned by removing the data point z from the training data.

B. Derivation of Influence Functions

Recall that the LOO parameter $\hat{\theta}_{i,\epsilon}$ is obtained by solving the optimization problem:

$$\hat{\theta}_{i,\epsilon} = \arg\min_{\theta\in\Theta} L(\mathcal{D}, \theta) + \epsilon \cdot \ell(y_i, f(\boldsymbol{x}_i; \theta)). \tag{9}$$

Let us first derive the first order influence function $\mathcal{I}^{(1)}(\boldsymbol{x}_i, y_i)$. Let us first define $\Delta_{i,\epsilon} \triangleq \hat{\theta}_{i,\epsilon} - \hat{\theta}$. The first order influence function is given by:

$$\mathcal{I}^{(1)}(\boldsymbol{x}_i, y_i) = \frac{\partial \hat{\theta}_{i,\epsilon}}{\partial \epsilon} = \frac{\partial \Delta_{i,\epsilon}}{\partial \epsilon}. \tag{10}$$

Note that, since $\hat{\theta}_{i,\epsilon}$ is the minimizer of (9), then the perturbed loss has to satisfy the following (first order) optimality condition:

$$\nabla_{\theta} \left\{ L(\mathcal{D}, \theta) + \epsilon \cdot \ell(y_i, f(\boldsymbol{x}_i; \theta)) \right\} \big|_{\theta=\hat{\theta}_{i,\epsilon}} = 0. \tag{11}$$

Since $\lim_{\epsilon\to 0} \hat{\theta}_{i,\epsilon} = \hat{\theta}$, then we can write the following Taylor expansion:

$$\nabla_{\theta} \sum_{k=0}^{\infty} \frac{\Delta_{i,\epsilon}^k}{k!} \cdot \nabla_{\theta}^k \left\{ L(\mathcal{D}, \hat{\theta}) + \epsilon \cdot \ell(y_i, f(\boldsymbol{x}_i; \hat{\theta})) \right\} = 0. \tag{12}$$

Now by dropping the $o(\|\Delta_{i,\epsilon}\|)$ terms, we have:

$$\nabla_{\theta} \left(\left\{ L(\mathcal{D}, \hat{\theta}) + \epsilon \cdot \ell(y_i, f(\boldsymbol{x}_i; \hat{\theta})) \right\} + \Delta_{i,\epsilon} \cdot \nabla_{\theta} \left\{ L(\mathcal{D}, \hat{\theta}) + \epsilon \cdot \ell(y_i, f(\boldsymbol{x}_i; \hat{\theta})) \right\} \right) = 0. \tag{13}$$

Since $\hat{\theta}$ is a indeed a minimizer of the loss function $\ell(.)$, then we have $\nabla_{\theta}\ell(.) = 0$. Thus, (13) reduces to the following condition:

$$\left\{ \epsilon \cdot \nabla_{\theta}\; \ell(y_i, f(\boldsymbol{x}_i; \hat{\theta})) \right\} + \Delta_{i,\epsilon} \cdot \left\{ \nabla_{\theta}^2\; L(\mathcal{D}, \hat{\theta}) + \epsilon \cdot \nabla_{\theta}^2\; \ell(y_i, f(\boldsymbol{x}_i; \hat{\theta})) \right\} = 0. \tag{14}$$

By solving for ∇_θ, we have

$$\Delta_{i,\epsilon} = -\left\{\nabla_\theta^2\ L(\mathcal{D},\hat{\theta}) + \epsilon\cdot\nabla_\theta^2\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta}))\right\}^{-1}\cdot\left\{\epsilon\cdot\nabla_\theta\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta}))\right\}, \tag{15}$$

which can be approximated by keeping only the $O(\epsilon)$ terms as follows:

$$\Delta_{i,\epsilon} = -\left\{\nabla_\theta^2\ L(\mathcal{D},\hat{\theta})\right\}^{-1}\cdot\left\{\epsilon\cdot\nabla_\theta\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta}))\right\}. \tag{16}$$

Noting that $\nabla_\theta^2\ L(\mathcal{D},\hat{\theta})$ is the Hessian matrix $H_{\hat{\theta}}$, we have:

$$\Delta_{i,\epsilon} = -H_{\hat{\theta}}^{-1}\cdot\left\{\epsilon\cdot\nabla_\theta\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta}))\right\}. \tag{17}$$

By taking the derivative with respect to ϵ, we arrive at the expression for first order influence functions:

$$\mathcal{I}^{(1)}(\boldsymbol{x}_i, y_i) = \frac{\Delta_{i,\epsilon}}{\epsilon}\Big|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\cdot\nabla_\theta\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta})). \tag{18}$$

Now let us examine the second order influence functions. In order to obtain $\mathcal{I}^{(2)}(\boldsymbol{x}_i, y_i)$, we need to differentiate (14) after omitting the $O(\epsilon)$ once again as follows:

$$\left\{2\Delta_{i,\epsilon}\cdot\epsilon\cdot\nabla_\theta^2\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta}))\right\} + \left\{\Delta_{i,\epsilon}^2\cdot\nabla_\theta^2\ L(\mathcal{D},\hat{\theta}) + \Delta_{i,\epsilon}\cdot\Delta_{i,\epsilon}\cdot\nabla_\theta^3\ L(\mathcal{D},\hat{\theta})\right\} = 0. \tag{19}$$

Where we have applied the chain rule to obtain the above. By substituting $\nabla_\theta^2\ L(\mathcal{D},\hat{\theta}) = H_\theta$ and dividing both sides of (19) by ϵ^2, we have

$$\left\{2\,\frac{\Delta_{i,\epsilon}}{\epsilon}\cdot\nabla_\theta^2\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta}))\right\} + \left\{\frac{\Delta_{i,\epsilon}^2}{\epsilon^2}\cdot H_\theta + \left(\frac{\Delta_{i,\epsilon}}{\epsilon}\right)^2\cdot\nabla_\theta^3\ L(\mathcal{D},\hat{\theta})\right\} = 0. \tag{20}$$

Thus, by re-arranging (19), we can obtain $\mathcal{I}^{(2)}(\boldsymbol{x}_i, y_i)$ in terms of $\mathcal{I}^{(1)}(\boldsymbol{x}_i, y_i)$ as follows:

$$\mathcal{I}^{(2)}(\boldsymbol{x}_i, y_i) = -H_\theta^{-1}\left(\left(\mathcal{I}^{(1)}(\boldsymbol{x}_i, y_i)\right)^2\cdot\nabla_\theta^3\ L(\mathcal{D},\hat{\theta}) + 2\,\mathcal{I}^{(1)}(\boldsymbol{x}_i, y_i)\cdot\nabla_\theta^2\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta}))\right).$$

Similarly, we can obtain the k^{th} order influence function, for any $k > 1$, by repeatedly differentiating equation (14) k times, i.e.,

$$\frac{\partial}{\partial\epsilon^k}\left\{\epsilon\cdot\nabla_\theta\ \ell(y_i, f(\boldsymbol{x}_i;\hat{\theta})) + \Delta_{i,\epsilon}\cdot\nabla_\theta^2\ L(\mathcal{D},\hat{\theta})\right\} = 0. \tag{21}$$

and solving for $\partial\Delta_{i,\epsilon}^k/\partial\epsilon^k$. By applying the higher-order chain rule to (21) (or equivalently, take the derivative of $\mathcal{I}^{(2)}(\boldsymbol{x}_i, y_i)$ for $k-2$ times), we recover the expressions in Definition 2 and Lemma 3 in (Giordano et al., 2019).

C. Theorem 1

Theorem 1 follows from Theorem 1 in (Barber et al., 2019b) for $m \to \infty$ when all HOIFs exist.

Recall that the exact DJ interval width is bounded above by:

$$W(\widehat{\mathcal{C}}^{(\infty)}_{\alpha,n}(\boldsymbol{x};\hat{\theta})) \leq 2\,\widehat{Q}_{\alpha,n}(\mathcal{R}_n) + 2\,\widehat{Q}_{\alpha,n}(\mathcal{V}_n(\boldsymbol{x})). \tag{22}$$

Since the term $\widehat{Q}_{\alpha,n}(\mathcal{R}_n)$ is constant for any $\boldsymbol{x}$, discrimination boils down to the following condition:

$$\mathbb{E}[\,\widehat{Q}_{\alpha,n}(\mathcal{V}_n(\boldsymbol{x}))\,] \geq \mathbb{E}[\,\widehat{Q}_{\alpha,n}(\mathcal{V}_n(\boldsymbol{x}'))\,] \Leftrightarrow \mathbb{E}[\,\ell(y, f(\boldsymbol{x};\hat{\theta}))\,] \geq \mathbb{E}[\,\ell(y', f(\boldsymbol{x}';\hat{\theta}))\,]. \tag{23}$$

Note that to prove the above, it suffices to prove the following:

$$\mathbb{E}[\,v_i(\boldsymbol{x})\,] \geq \mathbb{E}[\,v_i(\boldsymbol{x}')\,] \Leftrightarrow \mathbb{E}[\,\ell(y, f(\boldsymbol{x};\hat{\theta}))\,] \geq \mathbb{E}[\,\ell(y', f(\boldsymbol{x}';\hat{\theta}))\,]. \tag{24}$$

If the model is stable (based on the definition in (Bousquet & Elisseeff, 2002)), then a classical result by (Devroye & Wagner, 1979) states that:

$$\mathbb{E}[\,|\ell(y, f(\boldsymbol{x};\hat{\theta})) - \ell_n(y, f(\boldsymbol{x};\hat{\theta}))|^2\,] \approx \mathbb{E}[\,|\ell(y, f(\boldsymbol{x};\hat{\theta})) - \ell(y, f(\boldsymbol{x};\hat{\theta}_{-i}))|^2\,] + Const., \tag{25}$$

as $n \to \infty$, where $\ell_n(.)$ is the empirical risk on the training sample, and the expectation above is taken over $y \,|\, \boldsymbol{x}$. From (25), we can see that an increase in the LOO risk $\ell(y, f(\boldsymbol{x};\hat{\theta}_{-i}))$ implies an increase in the empirical risk $\ell_n(y, f(\boldsymbol{x};\hat{\theta}))$, and vice versa. Thus, for any two feature points $\boldsymbol{x}$ and $\boldsymbol{x}'$, if $v(\boldsymbol{x})$ is greater than $v(\boldsymbol{x}')$, then on average, the empirical risk at $\boldsymbol{x}$ is greater than that at $\boldsymbol{x}'$.

D. Experimental Details

D.1. Implementation of Baselines

In what follows, we provide details for the implementation and hyper-parameter settings for all baseline methods involved in Section 5.

Probabilistic backpropagation (PBP). We implemented the PBP method proposed in ((Hernández-Lobato & Adams, 2015)) with inference via expectation propagation using the `theano` code provided by the authors in (`github.com/HIPS/Probabilistic-Backpropagation`). Training was conducted via 1000 epochs.

Monte Carlo Dropout (MCDP). We implemented a `Pytorch` version of the MCDP method proposed in ((Gal & Ghahramani, 2016)). In all experiments, we tuned the dropout probability using Bayesian optimization to optimize the AUC-ROC performance on the training sample. We used 1000 samples at inference time to compute the mean and variance of the predictions. The credible intervals were constructed as the $(1-\alpha)$ quantile function of a posterior Gaussian distribution defined by the predicted mean and variance estimated through the Monte Carlo outputs. Similar to the other baselines, we conducted training via 1000 epochs for the SGD algorithm.

Bayesian neural networks (BNN). We implemented BNNs with inference via stochastic gradient Langevin dynamics (SGLD) ((Welling & Teh, 2011)). We initialized the prior weights and biases through a uniform distribution over $[-0.01, 0.01]$. We run 1000 epochs of the SGLD inference procedure and collect the posterior distributions to construct the credible intervals.

Deep ensembles (DE). We implemented a `Pytorch` version of the DE metho (without adversarial training)d proposed in ((Lakshminarayanan et al., 2017)). We used the number of ensemble members $M = 5$ as recommended in the recent study in ((Ovadia et al., 2019)). Predictions of the different ensembles were averaged and the confidence interval was estimated as 1.645 multiplied by the empirical variance for a target coverage of 90%. We trained the model through 1000 epochs.