Quantifying Epistemic Uncertainty in Deep Learning

Ziyi Huang[1]   **Henry Lam**[1]   **Haofeng Zhang**[1]

Abstract

Uncertainty quantification is at the core of the reliability and robustness of machine learning. In this paper, we provide a theoretical framework to dissect the uncertainty, especially the *epistemic* component, in deep learning into *procedural variability* (from the training procedure) and *data variability* (from the training data), which is the first such attempt in the literature to our best knowledge. We then propose two approaches to estimate these uncertainties, one based on influence function and one on batching. We demonstrate how our approaches overcome the computational difficulties in applying classical statistical methods. Experimental evaluations on multiple problem settings corroborate our theory and illustrate how our framework and estimation can provide direct guidance on modeling and data collection effort to improve deep learning performance.

1. Introduction

Uncertainty quantification is of major importance in machine learning, especially for risk-based decision making and safety-critical applications (Michelmore et al., 2018). It consists of two different components, often referred to as *aleatoric* and *epistemic* uncertainties. Aleatoric uncertainty refers to the intrinsic stochasticity of the problem while epistemic uncertainty refers to the errors coming from the inadequacy of the model or data noises. Measuring epistemic uncertainty is of importance in practice as it assesses the reliability of the predictive model, bearing the following immediate implications:

1. *Model selection:* Between models with similar predictive power, one with higher epistemic uncertainty is less favorable. That is, information on epistemic uncertainty gives a second dimension to predictive power for model selection.

2. *Effort planning:* A quantification of different sources of epistemic uncertainty guides which aspect of the uncertainty needs more attention and improvement. For instance, if more uncertainty appears due to the training noise than the data noise, then one should put more effort into training than collecting more data, and vice versa.

3. *Exploration guidance:* Epistemic uncertainty gives rise to upper confidence bound that has been widely used to tackle the exploration-exploitation dilemma in online learning.

Existing work on uncertainty measurement in deep learning models mainly focuses on *prediction sets* (*predictive uncertainty*), which captures the sum of epistemic and aleatoric uncertainties (Pearce et al., 2018; Romano et al., 2019; Barber et al., 2019b; Alaa & Van Der Schaar, 2020; Chen et al., 2021). Despite some high-level discussions (Pearce et al., 2018; Hüllermeier & Waegeman, 2021), few studies have been done in understanding epistemic uncertainty at a deeper level, namely differentiating and quantifying its different sources.

In this work, we fill in this gap by providing the first systematic study on the epistemic uncertainty in deep supervised learning. Epistemic uncertainty can be subdivided into three main sources: *Model uncertainty* signifies the discrepancy between the best model in the hypothesis class and the real data distribution, *procedural variability* refers to the noise stemming from the training procedure, and *data variability* arises from the uncertainty over how well the training data represents the real distribution. Our study in particular provides arguably the first dichotomy of procedural and data variabilities in deep learning. As an illustration of the strength of this dichotomy, we apply it to explain the benefit of deep

---

ensemble in reducing procedural rather than data variabilities, and demonstrate how this benefit scales with the ensemble size.

To execute our framework, we propose two approaches to estimate epistemic variances for well-trained neural networks in practice, which overcome some statistical and computational challenges faced by naive use of classical inference methods such as the bootstrap and jackknife. The first approach uses the delta method that involves computing the influence function, the latter viewed as the (functional) gradient of the predictor with respect to the data distribution. Our novelty lies in the derivation of this function using the theory of neural tangent kernel (NTK) (Jacot et al., 2018) that bypasses the convexity assumption violated by modern neural networks (Koh & Liang, 2017; Basu et al., 2020). The second approach uses the batching method that groups data into a small number of batches upon which re-training applies. This approach bypasses the time-consuming Gram matrix inversion in the influence function calculation, while advantageously expending a minimal re-training effort – In the case of deep ensemble, this effort is essentially the same as building the ensemble. While batching is long known in simulation analysis (Glynn & Iglehart, 1990; Schmeiser, 1982), its application and discovered advantages in deep learning are new to our best knowledge.

Our main contributions can be summarized as follows:
(1) We propose the first theoretical framework to dissect epistemic uncertainty in deep learning and give concrete justification and estimation methods for each component.
(2) We develop two approaches with theoretical justification to estimate the epistemic variances for deep neural networks, overcoming the statistical and computational challenges faced by classical statistical methods.
(3) We conduct comprehensive experiments to show the correctness of our theory and illustrate the practical use of our approaches on model training.

Related Work. Ronald A. Fisher (Fisher, 1930) was probably the first one to formally distinguish and build a connection between aleatory and epistemic uncertainty, according to Frank R. Hampel (Hampel, 2011). In modern machine learning, their distinction has been proposed in Senge et al. (2014) and then extended to deep learning (Kendall & Gal, 2017). The distinction between procedural and data variabilities in epistemic uncertainty is intuitively described in Pearce et al. (2018), but no estimation method or rigorous definition is given. We also remark that our work adopts the frequentist view, which is different from Bayesian uncertainty quantification (Kendall & Gal, 2017; Gal & Ghahramani, 2016; He et al., 2020). Further related work will be described in detail in the relevant subsections.

2. Statistical Framework of Uncertainty

In this section, we first review the standard statistical learning framework and then provide rigorous definitions for each type of uncertainty.

In supervised learning, we assume that the input-output pair (X, Y) is a random vector where $X \in \mathcal{X} \subset \mathbb{R}^d$ is an input and $Y \in \mathcal{Y}$ is the corresponding output. The random vector follows some unknown probability measure π on $\mathcal{X} \times \mathcal{Y}$. A learner is given access to a set of training data $\mathcal{D}_{tr} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, which is assumed to be independent and identically distributed (i.i.d.) according to the distribution π. Let the conditional distribution of Y given X be $\pi_{Y|X}(y|x)$. Let $\boldsymbol{x} = (x_1, ..., x_n)^T$ and $\boldsymbol{y} = (y_1, ..., y_n)^T$ for short. Let $\mathcal{H}$ denote a hypothesis class $\mathcal{H} = \{h_\theta : \mathcal{X} \to \mathcal{Y} | \theta \in \Theta\}$ where θ is the parameter (e.g., collection of network weights in deep learning) and Θ is the set of all possible parameters. Let $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be the loss function. For instance, $\mathcal{L}$ can be the square error in regression or cross-entropy loss in classification. The goal of the learner is to find a hypothesis $h_{\theta^*} \in \mathcal{H}$ that minimizes the following (population) risk function: $R(h) := \mathbb{E}_{(X,Y) \sim \pi}[\mathcal{L}(h(X), Y)]$. Let the true risk minimizer be

$$h_{\theta^*} := \arg \min_{\theta \in \Theta} R(h_\theta). \tag{1}$$

In practice, the population risk function is unknown (since π is unknown) so an empirical risk function is derived based on the training data $\mathcal{D}_{tr}$: $\hat{R}(h) := \mathbb{E}_{(X,Y) \sim \hat{\pi}_{\mathcal{D}_{tr}}}[\mathcal{L}(h(X), Y)] = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(h(x_i), y_i)$ where $\hat{\pi}_{\mathcal{D}_{tr}}$ is the empirical distribution on the training data. The learner aims to obtain

$$h_{\hat{\theta}^*} := \arg \min_{\theta \in \Theta} \hat{R}(h_\theta) \tag{2}$$

which is called empirical risk minimizer. In general, h_{θ^*} is not equal to $h_{\hat{\theta}^*}$. Note that in deep learning, $\hat{\theta}^*$ cannot be obtained precisely because of the non-convex nature of neural networks. Therefore, we usually approximate $\hat{\theta}^*$ by $\hat{\theta}$ using

gradient-based methods. We remark that h_{θ^*} is the best choice (in the sense of minimizing the risk) within the hypothesis set $\mathcal{H}$. When we are allowed to choose any possible functions, the best predictions (in the sense of minimizing the risk) are described by the *Bayes* predictor f^* (Hüllermeier & Waegeman, 2021):

$$f^*(X) := \arg \min_{\hat{y} \in \mathcal{Y}} \mathbb{E}_{Y \sim \pi_{Y|X}}[\mathcal{L}(\hat{y}, Y)|X]. \tag{3}$$

f^* cannot be obtained in practice since the conditional distribution $\pi_{Y|X}(y|x)$ is unknown.

2.1. Epistemic Uncertainty

We dissect epistemic uncertainty in deep supervised learning into three sources: model uncertainty, procedural variability, and data variability.

Model Uncertainty. Model uncertainty is the discrepancy between f^* (3) and h_{θ^*} (1) which is connected to the inadequacy regarding the choice of the hypothesis class $\mathcal{H}$:

$$\mathrm{UQ}_{MU} = h_{\theta^*} - f^*.$$

This error is sometimes referred to as the approximation error (Mohri et al., 2018). For over-parameterized neural networks where $\mathcal{H}$ is substantially rich, model uncertainty is usually negligible because of the impressive representation power of neural networks. For instance, a sufficiently wide network can universally approximate any continuous functions (Cybenko, 1989; Hornik, 1991; Hanin & Sellke, 2017) or any Lebesgue-integrable functions (Lu et al., 2017).

Proposition 2.1. *Suppose $f^* \in \mathcal{H}$. Then $UQ_{MU} = 0$.*

Procedural Variability. The empirical risk minimizer in (2) could not be obtained in deep learning because of the non-convex nature of neural networks. The standard way to solve (2) is to first randomly initialize the network parameters, which introduces the first additional epistemic uncertainty from *random initialization*. Then we do a mini-batch gradient descent on the shuffled training data to finalize the parameter $\hat{\theta}$, which introduces the second additional epistemic uncertainty due to *data ordering*. These lead to uncertainty in $\hat{\theta}$ associated with the training procedure.

We write Γ as the random variable corresponding to procedural variability. The trained parameter $\hat{\theta}$ is an instantiation of the outcome from one training run based on Γ and $\hat{\pi}_{\mathcal{D}_{tr}}$. Moreover, we reserve Γ_0 as the random variable that represents the random initialization of parameters. If the random initialization is the only uncertainty in procedural variability, then Γ is a deterministic function of Γ_0 and they represent the same procedural uncertainty. The uncertainty from procedural variability can be written as

$$\mathrm{UQ}_{PV} = h_{\hat{\theta}} - h_{\hat{\theta}^*}.$$

Data Variability. This is the most common epistemic uncertainty in classical statistics. The training dataset $\mathcal{D}_{tr}$ is a set of finite samples drawn from π, which cannot fully represent the population distribution π (i.e., $\hat{\pi}_{\mathcal{D}_{tr}} \neq \pi$). Data variability measures the representative level of the training dataset. Expressed at the hypothesis level, this uncertainty is written as

$$\mathrm{UQ}_{DV} = h_{\hat{\theta}^*} - h_{\theta^*}.$$

Summary. The three main sources of epistemic uncertainty add up to the total epistemic uncertainty:

$$h_{\hat{\theta}} - f^* =: \mathrm{UQ}_{EU} = \mathrm{UQ}_{MU} + \mathrm{UQ}_{PV} + \mathrm{UQ}_{DV}$$

which measures the discrepancy between $h_{\hat{\theta}}$ and f^*. We remark that $\mathrm{UQ}_{PV} + \mathrm{UQ}_{DV}$ is sometimes referred to as the estimation error in machine learning (Mohri et al., 2018). Lastly, we present additional discussions on aleatoric uncertainty and predictive uncertainty in the Supplementary Material for completeness; see Section A.

3. Quantifying Epistemic Uncertainty

Recall that the goal of supervised learning is to make prediction $h_{\hat{\theta}}$ on an unforeseen test point. Thus, quantifying epistemic uncertainty is important for risk-aware decision making, as a higher variance signifies a higher risk, and also for exploration in online learning as the variance is used in the principle of optimism in the face of uncertainty (Auer et al., 2002; Abbasi-Yadkori et al., 2011; Zhou et al., 2020). A natural approach to quantify epistemic uncertainty is to estimate the epistemic variance of $h_{\hat{\theta}}(x_0)$ on a given x_0. To this end, we have the following decomposition:

Theorem 3.1. *We have*

$$Var(UQ_{MU} + UQ_{PV} + UQ_{DV})$$
$$= Var(UQ_{PV} + UQ_{DV}) \tag{4}$$
$$= Var(\mathbb{E}[UQ_{DV}|\hat{\pi}_{\mathcal{D}_{tr}}]) + \mathbb{E}[Var(UQ_{PV}|\hat{\pi}_{\mathcal{D}_{tr}})] \tag{5}$$

Moreover, UQ_{MU}, UQ_{PV} and UQ_{DV} are uncorrelated.

In Theorem 3.1, we have assumed we plug in the test point x_0 into h when computing UQ, and the Var there is thus taken with respect to both the procedural and data randomness while fixing x_0. (4) follows since UQ_{MU} is deterministic. (5) follows from conditioning and noting that $\mathbb{E}[UQ_{PV}|\hat{\pi}_{\mathcal{D}_{tr}}]$ is constant and UQ_{DV} given $\hat{\pi}_{\mathcal{D}_{tr}}$ is also constant. By the same argument to obtain (5) we also conclude the uncorrelation among the uncertainties, similar to a mixed-effect model in classical statistics.

For convenience in the subsequent discussion, we express $h_{\hat{\theta}}(\cdot)$ as $h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; \cdot)$ where Γ corresponds to the procedural randomness and $\hat{\pi}_{\mathcal{D}_{tr}}$ encodes the data noise. We also denote the conditional population mean with respect to the procedural variability as

$$\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0) := \mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}]$$

The above variance decomposition can be equivalently rewritten as

$$Var(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0))$$
$$= Var(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0)) + \mathbb{E}[Var(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})] \tag{6}$$

In either (5) or (6), the overall epistemic variance consists of two terms, where the first term can be interpreted as coming from data variability and the second from procedural variability. In the following, we illustrate a direct application of our decomposition to explain the statistical power of deep ensemble (Lee et al., 2015; Lakshminarayanan et al., 2017; Fort et al., 2019).

3.1. Deep Ensemble Reduces Procedural Variability

Compared with other ensemble methods, deep ensemble has empirically demonstrated its state-of-the-art performance on model improvement (Ashukha et al., 2020). However, few theoretical justification has been developed despite its superior performance. In this section, we argue that the deep ensemble predictor improves the statistical profile over a single model by reducing its procedural variability. The *deep ensemble predictor* $\bar{h}_m(x_0)$ at x_0 is defined as:

$$\bar{h}_{m'}(x_0) := \frac{1}{m'} \sum_{i=1}^{m'} h_{\hat{\theta}_i}(x_0) \tag{7}$$

where m' is the number of independent training times (with the same training data) and $\hat{\theta}_i$, $i \in [m']$ represent the finalized network parameters. Our theory is formally stated as:

Theorem 3.2. *We have*

$$Var(\bar{h}_{m'}(x_0))$$
$$= Var(\mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}]) + \frac{1}{m'}\mathbb{E}[Var(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})] \tag{8}$$
$$\leq Var(\mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}]) + \mathbb{E}[Var(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})]$$
$$= Var(h_{\hat{\theta}_1}(x_0)).$$

Therefore, a deep ensemble predictor has less epistemic variance than a single model predictor.

In (8), by averaging m' training runs with independent procedural randomness, the second term of the variance decomposition of $\bar{h}_{m'}(x_0)$, which corresponds to the procedural variability, is reduced by m' folds compared to $\bar{h}_{m'}(x_0)$. If we could use a very large m', then this procedural variability can be completely removed, which resembles the conditional Monte Carlo method (Asmussen & Glynn, 2007) in variance reduction. Nonetheless, as long as $m' > 1$, variance reduction for $\bar{h}_{m'}(x_0)$ over $h_{\hat{\theta}_1}(x_0)$ is already present, and indeed a small number of single model predictors are used in the ensemble in practice.

We also contrast our explanation on deep ensemble to bagging (Breiman, 1996). The former averages over training runs while the latter over data resample. Bagging is known to improve prediction accuracy by smoothing base predictors (Bühlmann & Yu, 2002), and is more effective when these base predictors are non-smooth such as decision trees. This is very different from the procedural variability reduction we have asserted for deep ensemble.

3.2. Challenges of Existing Inference Approaches

In the rest of this paper, we focus on the estimation of epistemic variance, namely the two terms in (6) or their sum, or (8) in the deep ensemble case.

Borrowed from classical statistics, two high-level approaches may be considered to estimate these variances. First is an analytical approach using the delta method, which involves computing the influence function that acts as the functional gradient of the predictor with respect to the data distribution. Second is to use resampling, such as the bootstrap or jackknife. Unfortunately, each approach runs into its respective barrier. The performance of the delta method highly depends on the quality of the first-order approximation. However, neural networks are in general "excessively" non-convex and the M-estimation underpinning the delta method breaks down. Resampling requires enough resample replications and thus demands the effort of re-training. Even worse, in estimating the data variability (the first term in (6) or (8)), in fact we would need a "nested" sampling, i.e., first resample the data many times and for each fixed resample, run and average an inner number of re-training with different procedural randomness in order to compute the involved conditional expectation, which is more demanding than a standard bootstrap (or jackknife).

In the next two sections, we present two approaches to bypass each respective challenge. Our first approach applies the theory of NTK to derive an influence function that does not utilize convexity assumptions. Our second approach uses batching that divides data into a small number of batches upon whom re-training applies, which does not rely on many resample replications nor a nested sampling. We note that the NTK approach requires the evaluation of the Gram matrix and its inversion. Batching avoids this computation, but at the expense of a looser estimator. After presenting these approaches individually in Sections 4 and 5, we provide a detailed discussion in Section 6 on how to combine them judiciously.

4. Influence Function Method via NTK

4.1. Estimating Influence Function

The theory of NTK implies that the neural network (NN) predictor is essentially a kernel regression (KR) predictor whose kernel is the NTK when the network is sufficiently wide; see detailed discussion in Section B. Therefore, one direction to estimate the epistemic variance is to analyze the variance of its corresponding KR predictor, via its influence function. The influence function has a long history in statistics (Hampel, 1974), and is used as a nonparametric building block in the delta method for variance estimation (Fernholz, 2012). It was introduced in modern machine learning initially for understanding the effect of a training point on a model's prediction (Koh & Liang, 2017; Koh et al., 2019) but could be fragile for modern NNs (Basu et al., 2020) partially because the convexity assumption in deep learning is violated. We remark that our work focuses on applying the influence function to compute the epistemic variance. Moreover, our work calculates the influence function for the NN predictor, equivalently the KR predictor using the NTK, which is different from the influence function for network weights (Koh & Liang, 2017). Our approach can bypass the convexity assumption and cast another direction to study neural networks by influence functions.

We estimate the first component in (6). Let $\boldsymbol{K} = (K(x_i, x_j))_{i,j=1,...,n} \in \mathbb{R}^{n \times n}$ be the NTK Gram matrix evaluated on training data. For a test point $x_0 \in \mathbb{R}^d$, we let $K(x_0, \boldsymbol{x}) \in \mathbb{R}^n$ be the kernel value evaluated between the testing point and all training points, i.e., $K(x_0, \boldsymbol{x}) := (K(x_0, x_1), K(x_0, x_2), ..., K(x_0, x_n))^T$. We first obtain the inner expectation based on (22) and (24):

$$\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0) = h_0(x_0) + K(x_0, \boldsymbol{x})^T (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1} (\boldsymbol{y} - h_0(\boldsymbol{x})) \tag{9}$$

and $h_0(x_0) = \mathbb{E}_{\theta_0}[h(\theta_0; x_0)]$ where the expectation is with respect to the random initialization (h_0 does not depend on the data). There are two ways to approximate $\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0)$. It can be approximated by the *ensemble average* $\bar{h}_m(x_0)$ for sufficiently large m. Or it can be obtained by computing the NTK K and estimating $h_0(x_0)$ in (9) without training the network at all. In practice, we notice that both methods give us satisfactory performance (Lee et al., 2019).

Proposition 4.1. $\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; \cdot)$ *is the solution to the following kernel ridge regression problem*

$$\bar{h} - h_0 = \arg\min_{g \in \bar{\mathcal{H}}} \frac{1}{n} \sum_{i=1}^{n} (y_i - h_0(x_i) - g(x_i))^2 + \lambda \|g\|_{\mathcal{H}}^2. \tag{10}$$

We recall the definition of the influence function. Let P be a distribution on the domain $\mathcal{X} \times \mathcal{Y}$. Let $z = (z_x, z_y) \in \mathcal{X} \times \mathcal{Y}$. Suppose $P_{\varepsilon,z} = (1 - \varepsilon)P + \varepsilon\delta_z$ where δ_z denotes the point mass distribution in z. Let T be a statistical functional $T : P \to T(P)$. Then the influence function of T at P at the point z is defined as

$$IF(z; T, P) = \lim_{\varepsilon \to 0} \frac{T(P_{\varepsilon,z}) - T(P)}{\varepsilon}.$$

Let $T(P)$ be the predictor $\bar{h}(P; x_0)$ based on the data distribution P. Note that $\bar{h}(P; x_0)$ has removed the procedural variability. If $P = \hat{\pi}_{\mathcal{D}_{tr}}$, then $T(\hat{\pi}_{\mathcal{D}_{tr}})$ is $\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0)$. If $P = \pi$, then $T(\pi)$ is $h_{\theta^*}(x_0)$. Under some mild conditions (e.g., T is ρ_∞-Hadamard differentiable; see Fernholz (2012)), the central limit theorem holds: $\sqrt{n}(T(\hat{\pi}_{\mathcal{D}_{tr}}) - T(\pi))$ is asymptotically normal with mean 0 and variance

$$\sigma^2 = \int_z IF^2(z; T, \pi)d\pi(z). \tag{11}$$

In other words, $T(\hat{\pi}_{\mathcal{D}_{tr}}) \approx \mathcal{N}(T(\pi), \sigma^2/n)$. Therefore, we have approximately $\text{Var}(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0)) \approx \frac{\sigma^2}{n} \approx \frac{1}{n^2} \sum_{i=1}^{n} IF^2(z_i; T, \pi)$ where $z_i = (x_i, y_i)$ is from the training dataset. This formula gives us the approximation of the first component in (6). Hence our main target is to compute the influence function. In fact, we have

Theorem 4.2. *The influence function is given by*

$$IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}}) = K(x_0, \boldsymbol{x})^T (\boldsymbol{K} + \lambda n\boldsymbol{I})^{-1} M_z(\boldsymbol{x}) - M_z(x_0) \tag{12}$$

where $M_z(\boldsymbol{x}) := (M_z(x_1), ..., M_z(x_n))^T$ *and* $M_z(x)$ *is*

$$M_z(x) := \bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x) - h_0(x) - \frac{1}{\lambda}(z_y - \bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; z_x))K(z_x, x).$$

Theorem 4.3. *Suppose that T is continuously ρ_∞-Hadamard differentiable at π and $\hat{\pi}_{\mathcal{D}_{tr}}$. Suppose that $n(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0) - h_{\theta^*}(x_0))^2$ is uniformly integrable. Then we have the strong consistency*

$$\lim_{n \to \infty} \left| \frac{1}{n} \sum_{i=1}^{n} IF^2((x_i, y_i); T, \hat{\pi}_{\mathcal{D}_{tr}}) \right.$$

$$\left. - n \, Var(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0)) \right| = 0, \quad a.s.$$

Theorem 4.2 extends the influence function derivation of kernel ridge regression (Christmann & Steinwart, 2007a; Debruyne et al., 2008) to provide $IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}})$ at any test point x_0. Note that the asymptotic variance (11) is expressed in terms of $IF(z; T, \pi)$, instead of $IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}})$. Therefore, to fill this gap, Theorem 4.3 proves the convergence of $IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}})$ to $IF(z; T, \pi)$ with the aid of continuous differentiability. To conclude, the estimator of the data variability using the influence function is given by

$$\text{Var}(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0)) \approx \frac{1}{n^2} \sum_{i=1}^{n} IF^2((x_i, y_i); T, \hat{\pi}_{\mathcal{D}_{tr}})$$

4.2. Estimating Procedural Variance

For the procedural variance (the second component in (6)), we estimate with the *ensemble variance*:

$$\begin{aligned}
\tau^2 &:= \mathbb{E}[\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})] \\
&\approx \text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}) \\
&\approx \frac{1}{m-1} \sum_{i=1}^{m} \left(h_{\hat{\theta}_i}(x_0) - \bar{h}_m(x_0) \right)^2 := \text{EV}.
\end{aligned} \tag{13}$$

The first approximation above follows since, when viewing $\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0) | \hat{\pi}_{\mathcal{D}_{tr}})$ as a smooth function of $\hat{\pi}_{\mathcal{D}_{tr}}$, this quantity follows a central limit theorem which gives a fluctuation of order $1/\sqrt{n}$. This results in an approximation error $O_p(1/\sqrt{n})$ which is negligible compared to the magnitude of $\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0) | \hat{\pi}_{\mathcal{D}_{tr}})$ itself. The second approximation holds readily when m is large enough. Otherwise, even for small m, we could still use

$$\left[\frac{\text{EV}}{\chi^2_{m-1,1-\alpha/2}/(m-1)}, \frac{\text{EV}}{\chi^2_{m-1,\alpha/2}/(m-1)} \right]$$

as a $(1-\alpha)$-level confidence interval (CI) for $\mathbb{E}[\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0) | \hat{\pi}_{\mathcal{D}_{tr}})]$, where $\chi^2_{m-1,\kappa}$ denotes the κ-th quantile of the χ^2-distribution with degree of freedom $m-1$. This is assuming that the procedural variability is Gaussian and thus we can use the χ^2-distribution for inference on variance. The latter is related to the batching method that we discuss next.

5. Batching

In this section, we propose an alternative approach, based on *batching* (Glynn & Iglehart, 1990; Schmeiser, 1982; Schruben, 1983), to avoid the Gram matrix inversion in the influence function calculation. Originating from simulation analysis, the key idea of batching is to construct a self-normalizing t-statistic that "cancels out" the unknown variance, leading to a valid confidence interval without explicitly needing to compute the variance. It can be used to conduct inference on serially dependent simulation outputs where the standard error is difficult to analytically compute. Previous studies have demonstrated the effectiveness of batching on the use of inference for Markov chain Monte Carlo (Geyer, 1992; Flegal et al., 2010; Jones et al., 2006) and also the so-called input uncertainty problem (Glynn & Lam, 2018).

Our approach leverages the self-normalizing idea in batching to construct an epistemic variance estimator via the χ^2-statistic, especially in the context of deep ensemble. We divide the data into several batches of equal size, say $\mathcal{D}_1, ..., \mathcal{D}_K$, where K is a constant and $|\mathcal{D}_i| = l = \frac{n}{K}$, $i \in [K]$. With a random initialization Γ_i, we train the neural network on batch $\mathcal{D}_i$ once, and obtain the output $\hat{\psi}_i(x_0) := h(\Gamma_i, \hat{\pi}_{\mathcal{D}_i}; x_0)$ where $i \in [K]$. We show that, perhaps surprisingly, it is easy to obtain the epistemic variance of a deep ensemble predictor by using the K batched predictors.

Recall in Section 4.1 we have argued that $T(\hat{\pi}_{\mathcal{D}_{tr}}) \approx \mathcal{N}(T(\pi), \sigma^2/n)$. Suppose further that the training noise is also approximate Gaussian, independent of $\hat{\pi}_{\mathcal{D}_{tr}}$ (recall from Theorem 3.1 that the uncertainty sources are all uncorrelated). In other words, we assume that

$$h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0) \stackrel{d}{=} h_{\theta^*}(x_0) + \left(\frac{\sigma}{\sqrt{n}} Z + \tau W \right) (1 + o_p(1)) \tag{14}$$

where Z, W are two independent $\mathcal{N}(0, 1)$ and the $o_p(1)$ term is as the data size n and potentially a parameter controlling the procedural noise go to ∞. To enhance credibility on this assumption, we have empirically verified it with our experiments; see Section 7. Thus we would obtain, for a deep ensemble predictor $\bar{h}_{m'}(x_0)$,

$$\bar{h}_{m'}(x_0) \stackrel{d}{=} h_{\theta^*}(x_0) + \left(\frac{\sigma}{\sqrt{n}} Z + \frac{\tau}{\sqrt{m'}} W \right) (1 + o_p(1))$$
$$= h_{\theta^*}(x_0) + \nu Y (1 + o_p(1)) \tag{15}$$

where $Y \sim \mathcal{N}(0, 1)$ and $\nu = \sqrt{\sigma^2/n + \tau^2/m'}$. This matches Theorem 3.2 which gives

$$\text{Var}(\bar{h}_{m'}(x_0)) = \left(\frac{\sigma^2}{n} + \frac{\tau^2}{m'} \right) (1 + o(1))$$

where we would have

$$\mathbb{E}[\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0) | \hat{\pi}_{\mathcal{D}_{tr}})] \approx \tau^2,$$

$$\text{Var}(\mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0) | \hat{\pi}_{\mathcal{D}_{tr}}]) \approx \frac{\sigma^2}{n}.$$

Now for the batched predictors $\hat{\psi}_i(x_0)$, we have similarly

$$\hat{\psi}_i(x_0) \stackrel{d}{=} h_{\theta^*}(x_0) + \left(\frac{\sigma}{\sqrt{l}} Z_i + \tau W_i \right) (1 + o_p(1)) \tag{16}$$

where $Z_i, W_i, i \in [K]$ are independent $\mathcal{N}(0, 1)$. To estimate $\text{Var}(\bar{h}_{m'}(x_0))$, we set $K = m'$ so that each batched predictor satisfies

$$\hat{\psi}_i(x_0) \overset{d}{=} h_{\theta^*}(x_0) + \sqrt{m'}\nu Y_i(1 + o_p(1)) \tag{17}$$

Note that the RHS in (17) differs from that of (15) only by the $\sqrt{m'}$ factor in front of the Y. This turns out to enable the construction of a self-normalizing statistic and in turn an estimate for $\sigma^2/n + \tau^2/m'$, as detailed below:

Theorem 5.1. *Suppose a single model predictor $h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)$ satisfies (14) and the batched predictors satisfy (16). Then, choosing the number of batches $K = m'$ (assumed fixed), and letting $\bar{\psi}(x_0) = \frac{1}{m'}\sum_{i=1}^{m'} \hat{\psi}_i(x_0)$ and $S^2 = \frac{1}{m'-1}\sum_{i=1}^{m'}(\hat{\psi}_i(x_0) - \bar{\psi}(x_0))^2$ be the sample mean and variance of the batched predictors, we have*

$$\frac{S^2/m'}{\sigma^2/n + \tau^2/m'} \Rightarrow \frac{\chi^2_{m'-1}}{m'-1}$$

where $\Rightarrow$ denotes convergence in distribution. Therefore,

$$\left[\frac{S^2/m'}{\chi^2_{m'-1,1-\alpha/2}/(m'-1)}, \frac{S^2/m'}{\chi^2_{m'-1,\alpha/2}/(m'-1)} \right] \tag{18}$$

is an asymptotically exact $(1 - \alpha)$-level confidence interval (CI) for $\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$, the asymptotic epistemic variance of the deep ensemble estimator.

Theorem 5.1 shows that with the same number of batched predictors as the training runs used in the deep ensemble, we can construct an accurate-coverage CI for its epistemic variance. Note that by virtue of (18) we could also simply use $\frac{S^2}{m'}$ as a rough estimate of $\text{Var}(\bar{h}_{m'}(x_0))$, which is accurate when m' is moderate. We remark that Theorem 5.1 indicates a non-trivial, possibly surprising result: The correct approach to estimate the variance of deep ensemble predictor is to train a **single, rather than ensemble** neural network predictor in each batch. If we train an ensemble predictor in each batch, we would actually *under-estimate* the uncertainty of the original deep ensemble. Moreover, this result also suggests that batching has an advantageously light computation load, as it only requires the same number of training runs as the original deep ensemble predictor and does not involve any matrix calculation.

6. Summary and Combining Methods

In this section, we provide a summary of our proposals on epistemic variance estimation in Equation (6):

a) $\frac{\sigma^2}{n}$ (11) can be estimated by the influence function, which does not require network training but needs to evaluate the NTK Gram matrix:

$$\text{Var}(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0)) \approx \frac{1}{n^2}\sum_{i=1}^{n} IF^2((x_i, y_i); T, \hat{\pi}_{\mathcal{D}_{tr}}).$$

b) τ^2 (13) can be estimated by the ensemble variance of m networks:

$$\tau^2 \approx \frac{1}{m-1}\sum_{i=1}^{m} \left(h_{\hat{\theta}_j}(x_0) - \bar{h}_m(x_0) \right)^2.$$

c) $\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ can be estimated by the batched predictors which only requires m' times of network training:

$$\text{Var}(\bar{h}_{m'}(x_0)) \approx \frac{S^2}{m'}.$$

Any two estimators from a)-c) can give rise to separate estimates of both components in (6): The use of a)+b) or a)+c) both involves matrix calculation and either the m training runs in building an ensemble or m' batched training, while the use of b)+c) can avoid the matrix calculation. Here, a small m or m' produces less accurate point estimates of the variances (but accurate-coverage CIs). On the other hand, if consistent point estimates of the variances are desired, we can increase m to a large number, but not m' since doing so would lead to a small batch size l which in turn deteriorates the required Gaussian approximation underpinning the batching method. By doing this, we can use a)+b) with matrix calculation and m training runs.

7. Experiments

In this section, we conduct evaluations to demonstrate the effectiveness of our approaches: ensemble variance, influence function, and batching on the epistemic variance estimation (6). Since this is the first study that aims to quantify different sources of epistemic uncertainty, we do not find any off-the-shelf algorithms for baseline comparisons. Thus, to fully evaluate our model, we conduct comprehensive experiments on both synthetic datasets and benchmark real-world datasets with different problem settings. We will make our code publicly available.

Our approach does not require *any* prior knowledge of the underlying generative process and this knowledge is only used for ground-truth calculation on synthetic datasets. In each experiment on synthetic datasets, the ground-truth variance is empirically calculated with $J = 100$, which is a sufficient number of repeated trials leading to stable estimations. In each repetition $j \in [J]$, we generate a new training dataset from the same synthetic distribution and re-train the single predictor/deep ensemble predictor to obtain the prediction $h_j(x_0)$. Then, the ground-truth variance is calculated by the following equation: $\frac{1}{J-1} \sum_{j=1}^{J} \left(h_j(x_0) - \frac{1}{J} \sum_{i=1}^{J} h_i(x_0) \right)^2$. By doing this, we can obtain the ground-truth data variance and procedural variance separately. See Section D.1 for more details.

7.1. Synthetic Datasets

	τ^2 GT	τ^2 EV	τ^2 Diff
$d = 2$	$0.5 * 10^{-4}$	$0.4 * 10^{-4}$	$-0.1 * 10^{-4}$
$d = 4$	$1.6 * 10^{-4}$	$1.5 * 10^{-4}$	$-0.1 * 10^{-4}$
$d = 8$	$9.3 * 10^{-4}$	$9.1 * 10^{-4}$	$-0.2 * 10^{-4}$
$d = 16$	$2.1 * 10^{-3}$	$2.0 * 10^{-3}$	$-0.1 * 10^{-3}$
$d = 32$	$3.7 * 10^{-3}$	$3.5 * 10^{-3}$	$-0.2 * 10^{-3}$
	$\frac{\sigma^2}{n}$ GT	$\frac{\sigma^2}{n}$ IF	$\frac{\sigma^2}{n}$ Diff
$d = 2$	$1.6 * 10^{-4}$	$1.8 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 4$	$3.3 * 10^{-4}$	$3.5 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 8$	$4.7 * 10^{-4}$	$4.8 * 10^{-4}$	$+0.1 * 10^{-4}$
$d = 16$	$0.7 * 10^{-3}$	$0.8 * 10^{-3}$	$+0.1 * 10^{-3}$
$d = 32$	$1.5 * 10^{-3}$	$1.7 * 10^{-3}$	$+0.2 * 10^{-3}$
	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ GT	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ BA	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ Diff
$d = 2$	$1.7 * 10^{-4}$	$1.7 * 10^{-4}$	$+0.0 * 10^{-4}$
$d = 4$	$3.6 * 10^{-4}$	$3.8 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 8$	$6.6 * 10^{-4}$	$6.8 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 16$	$1.1 * 10^{-3}$	$1.2 * 10^{-3}$	$+0.1 * 10^{-3}$
$d = 32$	$2.2 * 10^{-3}$	$2.3 * 10^{-3}$	$+0.1 * 10^{-3}$

Table 1. Epistemic Variance Estimation on Synthetic Datasets with $n = 200$ with Different Dimensions.

We conduct experiments on the following synthetic distribution: $X \sim \text{Unif}([0, 0.2]^d)$ and $Y \sim \sum_{i=1}^{d} \sin(X^{(i)}) + \mathcal{N}(0, 0.1^2)$ with multiple dimension settings $d = 2, 4, 8, 16, 32$ to study the dimensionality effect on the epistemic uncertainty. The training set $\mathcal{D}_{tr} = \{(x_i, y_i) : i = 1, ..., n\}$ is formed by i.i.d. samples of (X, Y) with sample size n taking values in $200, 500, 1000$. We use $x_0 = (0.1, ..., 0.1)$ as the fixed test point and set $K = m' = 5$, $m = 50$. We implement a two-layer fully-connected NN with 1024 hidden neurons as the baseline predictor according to the NTK specification in Section B and it is optimized by the regularized square loss (20).

Tables 1, 2, 3 report the ground-truth (GT) variance and estimates from our proposed ensemble variance (EV), influence function (IF), and batching (BA). As a primary observation, the estimated variances from our methods appear highly consistent with the ground truth on all experiments with different settings, which evidently demonstrates the robustness and effectiveness of our proposals. In addition, we observe that:
1) The procedural variance seems almost uncorrelated with the size of training data, while the data variance reduces roughly at the rate n^{-1} when the size of training data increases. This observation is consistent with our (14).
2) Both procedural variance and data variance increase with the dimension of input data. In particular, the procedural variance increases much faster than the data variance. This is probably caused by the increasing number of minima in the

	τ^2 GT	τ^2 EV	τ^2 Diff
$d = 2$	$0.4 * 10^{-4}$	$0.3 * 10^{-4}$	$-0.1 * 10^{-4}$
$d = 4$	$1.6 * 10^{-4}$	$1.5 * 10^{-4}$	$-0.1 * 10^{-4}$
$d = 8$	$9.2 * 10^{-4}$	$9.0 * 10^{-4}$	$-0.2 * 10^{-4}$
$d = 16$	$2.1 * 10^{-3}$	$2.0 * 10^{-3}$	$-0.1 * 10^{-3}$
$d = 32$	$3.7 * 10^{-3}$	$3.6 * 10^{-3}$	$-0.1 * 10^{-3}$
	$\frac{\sigma^2}{n}$ GT	$\frac{\sigma^2}{n}$ IF	$\frac{\sigma^2}{n}$ Diff
$d = 2$	$0.7 * 10^{-4}$	$0.9 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 4$	$1.4 * 10^{-4}$	$1.8 * 10^{-4}$	$+0.4 * 10^{-4}$
$d = 8$	$2.1 * 10^{-4}$	$2.4 * 10^{-4}$	$+0.3 * 10^{-4}$
$d = 16$	$0.3 * 10^{-3}$	$0.4 * 10^{-3}$	$+0.1 * 10^{-3}$
$d = 32$	$0.7 * 10^{-3}$	$0.8 * 10^{-3}$	$+0.1 * 10^{-3}$
	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ GT	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ BA	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ Diff
$d = 2$	$0.8 * 10^{-4}$	$1.0 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 4$	$1.7 * 10^{-4}$	$1.9 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 8$	$3.9 * 10^{-4}$	$4.1 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 16$	$0.7 * 10^{-3}$	$0.8 * 10^{-3}$	$+0.1 * 10^{-3}$
$d = 32$	$1.4 * 10^{-3}$	$1.6 * 10^{-3}$	$+0.2 * 10^{-3}$

Table 2. Epistemic Variance Estimation on Synthetic Datasets with $n = 500$ with Different Dimensions.

	τ^2 GT	τ^2 EV	τ^2 Diff
$d = 2$	$0.4 * 10^{-4}$	$0.3 * 10^{-4}$	$-0.1 * 10^{-4}$
$d = 4$	$1.5 * 10^{-4}$	$1.4 * 10^{-4}$	$-0.1 * 10^{-4}$
$d = 8$	$9.2 * 10^{-4}$	$8.9 * 10^{-4}$	$-0.3 * 10^{-4}$
$d = 16$	$2.0 * 10^{-3}$	$1.9 * 10^{-3}$	$-0.1 * 10^{-3}$
$d = 32$	$3.6 * 10^{-3}$	$3.3 * 10^{-3}$	$-0.3 * 10^{-3}$
	$\frac{\sigma^2}{n}$ GT	$\frac{\sigma^2}{n}$ IF	$\frac{\sigma^2}{n}$ Diff
$d = 2$	$0.4 * 10^{-4}$	$0.6 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 4$	$0.8 * 10^{-4}$	$1.0 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 8$	$1.3 * 10^{-4}$	$1.6 * 10^{-4}$	$+0.3 * 10^{-4}$
$d = 16$	$0.2 * 10^{-3}$	$0.3 * 10^{-3}$	$+0.1 * 10^{-3}$
$d = 32$	$0.4 * 10^{-3}$	$0.4 * 10^{-3}$	$+0.0 * 10^{-3}$
	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ GT	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ BA	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ Diff
$d = 2$	$0.5 * 10^{-4}$	$0.5 * 10^{-4}$	$+0.0 * 10^{-4}$
$d = 4$	$1.1 * 10^{-4}$	$1.4 * 10^{-4}$	$+0.3 * 10^{-4}$
$d = 8$	$3.1 * 10^{-4}$	$3.3 * 10^{-4}$	$+0.2 * 10^{-4}$
$d = 16$	$0.6 * 10^{-3}$	$0.7 * 10^{-3}$	$+0.1 * 10^{-3}$
$d = 32$	$1.1 * 10^{-3}$	$1.2 * 10^{-3}$	$+0.1 * 10^{-3}$

Table 3. Epistemic Variance Estimation on Synthetic Datasets with $n = 1000$ with Different Dimensions.

	τ^2 EV	$\frac{\sigma^2}{n}$ IF	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ BA
Boston	$9.1 * 10^{-3}$	$0.2 * 10^{-3}$	$2.3 * 10^{-3}$
Concrete	$1.3 * 10^{-3}$	$0.4 * 10^{-3}$	$0.7 * 10^{-3}$
Energy	$10.8 * 10^{-3}$	$0.2 * 10^{-3}$	$2.5 * 10^{-3}$
Wine	$2.0 * 10^{-3}$	$0.1 * 10^{-3}$	$0.6 * 10^{-3}$
Yacht	$1.3 * 10^{-3}$	$0.1 * 10^{-3}$	$0.5 * 10^{-3}$

Table 4. Epistemic Variance Estimation on Real-World Datasets.

networks along with the increasing number of parameters. This suggests that for large dataset, deep ensemble should be applied to reduce the procedural variance to boost performance.

7.2. Real-World Datasets

Finally, we conduct a series of experiments on the following widely-used benchmark real-world datasets: Boston, Concrete, Energy, Wine, and Yacht (Hernández-Lobato & Adams, 2015; Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017; Rosenfeld et al., 2018; Pearce et al., 2018; Zhu et al., 2019). The test point is chosen to be the mean feature vector. It is impossible to obtain the ground truth of epistemic variance on real-world datasets and hence, in this section, we focus on the implications of our estimates.

Table 4 presents the results of our proposed approaches on the benchmark datasets. As shown, on real-world datasets, the procedural variance is the predominant component in epistemic uncertainty. Compared with the synthetic datasets in Section 7.1 where we add additional data noise on the labels, these benchmark datasets seem to have less noise on the ground-truth labels, and thus have less data uncertainty. Hence, techniques which efficiently reduce the procedural variance (such as deep ensemble), can bring considerable improvements on the model performance. This also provides a theoretical justification of the findings that deep ensemble performs similarly as combining with Bagging methods in previous work (Lakshminarayanan et al., 2017).

8. Concluding Remarks

In this paper, we propose the first theoretical framework to dissect epistemic uncertainty in deep learning, with concrete justification and estimation methods for each component. By quantifying different constituents of epistemic uncertainty, our approaches can provide guidance on modeling and data collection to improve training. The evaluations on synthetic datasets fully corroborate our theory. Moreover, from the experiments in real-world datasets, we observe that the procedural variability is most likely to be the predominant component, which suggests the use of ensemble methods to boost model performance. Finally, quantifying epistemic uncertainty can also be very useful for real-world applications such as online learning problems, which is left as an interesting direction of future work.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems*, 24:2312–2320, 2011.

Alaa, A. and Van Der Schaar, M. Frequentist uncertainty in recurrent neural networks via blockwise influence functions. In *International Conference on Machine Learning*, pp. 175–190. PMLR, 2020.

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in Neural Information Processing Systems*, 2019a.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pp. 242–252. PMLR, 2019b.

Allen-Zhu, Z., Li, Y., and Song, Z. On the convergence rate of training recurrent neural networks. *Advances in Neural Information Processing Systems*, 32:6676–6688, 2019c.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32:8141–8150, 2019.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470*, 2020.

Asmussen, S. and Glynn, P. W. *Stochastic simulation: algorithms and analysis*, volume 57. Springer Science & Business Media, 2007.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47 (2):235–256, 2002.

Bai, Y., Mei, S., Wang, H., and Xiong, C. Understanding the under-coverage bias in uncertainty estimation. *arXiv preprint arXiv:2106.05515*, 2021.

Barber, R. F., Candes, E. J., Ramdas, A., and Tibshirani, R. J. The limits of distribution-free conditional predictive inference. *arXiv preprint arXiv:1903.04684*, 2019a.

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

Basu, S., Pope, P., and Feizi, S. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

Berlinet, A. and Thomas-Agnan, C. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

Breiman, L. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Bühlmann, P. and Yu, B. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.

Cao, Y. and Gu, Q. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *Advances in Neural Information Processing Systems*, 32:10836–10846, 2019.

Cao, Y. and Gu, Q. Generalization error bounds of gradient descent for learning over-parameterized deep relu networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3349–3356, 2020.

Chen, H., Huang, Z., Lam, H., Qian, H., and Zhang, H. Learning prediction intervals for regression: Generalization and calibration. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Christmann, A. and Steinwart, I. Consistency and robustness of kernel-based regression in convex risk minimization. *Bernoulli*, 13(3):799–819, 2007a.

Christmann, A. and Steinwart, I. How svms can estimate quantiles and the median. *Advances in Neural Information Processing Systems*, 20:305–312, 2007b.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2 (4):303–314, 1989.

Dalmasso, N., Pospisil, T., Lee, A. B., Izbicki, R., Freeman, P. E., and Malz, A. I. Conditional density estimation tools in python and r with applications to photometric redshifts and likelihood-free cosmological inference. *Astronomy and Computing*, 30:100362, 2020.

Debruyne, M., Hubert, M., and Suykens, J. A. Model selection in kernel based regression using the influence function. *Journal of Machine Learning Research*, 9(10), 2008.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pp. 1675–1685. PMLR, 2019.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2018.

Dutordoir, V., Salimbeni, H., Hensman, J., and Deisenroth, M. Gaussian process conditional density estimation. In *Advances in Neural Information Processing Systems*, pp. 2385–2395, 2018.

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

Fisher, R. A. Inverse probability. *Mathematical Proceedings of the Cambridge Philosophical Society*, 26(4):528–535, 1930.

Flegal, J. M., Jones, G. L., et al. Batch means and spectral variance estimators in Markov chain Monte Carlo. *The Annals of Statistics*, 38(2):1034–1070, 2010.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Freeman, P. E., Izbicki, R., and Lee, A. B. A unified framework for constructing, tuning and assessing photometric redshift density estimates in a selection bias setting. *Monthly Notices of the Royal Astronomical Society*, 468(4):4556–4565, 2017.

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

Geyer, C. J. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

Glynn, P. W. and Iglehart, D. L. Simulation output analysis using standardized time series. *Mathematics of Operations Research*, 15(1):1–16, 1990. ISSN 0364765X, 15265471.

Glynn, P. W. and Lam, H. Constructing simulation output intervals under input uncertainty via data sectioning. In *2018 Winter Simulation Conference (WSC)*, pp. 1551–1562. IEEE, 2018.

Hampel, F. R. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69 (346):383–393, 1974.

Hampel, F. R. Potential surprises. *International Symposium on Imprecise Probability: Theories and Applications (ISIPTA)*, pp. 209, 2011.

Hanin, B. and Sellke, M. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

He, B., Lakshminarayanan, B., and Teh, Y. W. Bayesian deep ensembles via the neural tangent kernel. *arXiv preprint arXiv:2007.05864*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

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

Holmes, M. P., Gray, A. G., and Isbell Jr, C. L. Fast nonparametric conditional density estimation. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pp. 175–182, 2007.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Hu, W., Li, Z., and Yu, D. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. In *International Conference on Learning Representations*, 2020.

Hüllermeier, E. and Waegeman, W. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

Izbicki, R. and Lee, A. B. Nonparametric conditional density estimation in a high-dimensional regression setting. *Journal of Computational and Graphical Statistics*, 25(4):1297–1316, 2016.

Izbicki, R., Lee, A. B., and Freeman, P. E. Photo-z estimation: An example of nonparametric conditional density estimation under selection bias. *Ann. Appl. Stat.*, 11(2):698–724, 06 2017. doi: 10.1214/16-AOAS1013.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

Jones, G. L., Haran, M., Caffo, B. S., and Neath, R. Fixed-width output analysis for Markov chain Monte Carlo. *Journal of the American Statistical Association*, 101(476):1537–1547, 2006.

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE Transactions on Neural Networks*, 22(3):337–346, 2010.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on Neural Networks*, 22(9):1341–1356, 2011.

Koenker, R. and Hallock, K. F. Quantile regression. *Journal of Economic Perspectives*, 15(4):143–156, 2001.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pp. 1885–1894. PMLR, 2017.

Koh, P. W., Ang, K.-S., Teo, H. H., and Liang, P. On the accuracy of influence functions for measuring group effects. *arXiv preprint arXiv:1905.13289*, 2019.

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

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in Neural Information Processing Systems*, 32: 8572–8583, 2019.

Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D., and Batra, D. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

Lei, J. and Wasserman, L. Distribution-free prediction bands for non-parametric regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1):71–96, 2014.

Lei, J., Rinaldo, A., and Wasserman, L. A conformal prediction approach to explore functional data. *Annals of Mathematics and Artificial Intelligence*, 74(1-2):29–43, 2015.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., and Wasserman, L. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in Neural Information Processing Systems*, 2018.

Li, Z., Wang, R., Yu, D., Du, S. S., Hu, W., Salakhutdinov, R., and Arora, S. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. The expressive power of neural networks: A view from the width. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6232–6240, 2017.

Meinshausen, N. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun):983–999, 2006.

Michelmore, R., Kwiatkowska, M., and Gal, Y. Evaluating uncertainty quantification in end-to-end autonomous driving control. *arXiv preprint arXiv:1811.06817*, 2018.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2018.

Pearce, T., Zaki, M., Brintrup, A., and Neely, A. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *International Conference on Machine Learning, PMLR: Volume 80*, 2018.

Romano, Y., Patterson, E., and Candes, E. Conformalized quantile regression. In *Advances in Neural Information Processing Systems*, pp. 3543–3553, 2019.

Rosenfeld, N., Mansour, Y., and Yom-Tov, E. Discriminative learning of prediction intervals. In *International Conference on Artificial Intelligence and Statistics*, pp. 347–355, 2018.

Schmeiser, B. Batch size effects in the analysis of simulation output. *Operations Research*, 30(3):556–568, 1982. ISSN 0030364X, 15265463.

Schruben, L. Confidence interval estimation using standardized time series. *Operations Research*, 31(6):1090–1108, 1983.

Senge, R., Bösner, S., Dembczyński, K., Haasenritter, J., Hirsch, O., Donner-Banzhoff, N., and Hüllermeier, E. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences*, 255:16–29, 2014.

Shao, J. and Tu, D. *The jackknife and bootstrap*. Springer Science & Business Media, 2012.

Smale, S. and Zhou, D.-X. Shannon sampling ii: Connections to learning theory. *Applied and Computational Harmonic Analysis*, 19(3):285–302, 2005.

Steinwart, I., Christmann, A., et al. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1): 211–225, 2011.

Vovk, V., Gammerman, A., and Shafer, G. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

Yang, G. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

Yang, G. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020.

Zhang, H., Zimmerman, J., Nettleton, D., and Nordman, D. J. Random forest prediction intervals. *The American Statistician*, pp. 1–15, 2019.

Zhang, Y., Xu, Z.-Q. J., Luo, T., and Ma, Z. A type of generalization error induced by initialization in deep neural networks. In *Mathematical and Scientific Machine Learning*, pp. 144–164. PMLR, 2020.

Zhou, D., Li, L., and Gu, Q. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

Zhu, L., Lu, J., and Chen, Y. HDI-forest: highest density interval regression forest. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 4468–4474. AAAI Press, 2019.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, 2020.

We provide further results and discussions in this supplementary material. Section A discusses aleatoric uncertainty and predictive uncertainty. Section B discusses recent NTK work regrading the equivalence between the NN predictor and KR predictor. Section C presents the missing proofs for results in the main paper. Section D presents experimental details and more experimental results.

A. Other Types of Uncertainty

We present an additional discussion on other types of uncertainty for the sake of completeness. Section A.1 presents aleatoric uncertainty and Section A.2 presents predictive uncertainty.

A.1. Aleatoric Uncertainty

We note that if we could remove all the epistemic uncertainty, i.e., letting $\text{UQ}_{EU} = 0$, the best predictor we could get is the Bayes predictor f^*. However, f^*, as a point estimator, is not able to capture the randomness in $\pi_{Y|X}$ if it is not a point mass distribution.

The easiest way to think of aleatoric uncertainty is that it is captured by $\pi_{Y|X}$. At the level of the realized response or label value, this uncertainty is represented by

$$\text{UQ}_{AU} = y - f^*(x)$$

If the connection between X and Y is non-deterministic, the description of a new prediction problem involves a conditional probability distribution $\pi_{Y|X}$. Standard neural network predictors can only provide a single output y. Thus, even given full information of the distribution π, the uncertainty in the prediction of a single output y remains. This uncertainty cannot be removed by better modeling or more data.

There are multiple work aiming to estimate the aleatoric uncertainty, i.e., to learn $\pi_{Y|X}$. For instance, conditional quantile regression aims to learn each quantile of the distribution $\pi_{Y|X}$ (Koenker & Hallock, 2001; Meinshausen, 2006; Steinwart et al., 2011). Conditional density estimation aims to approximately describe the density of $\pi_{Y|X}$ (Holmes et al., 2007; Dutordoir et al., 2018; Izbicki & Lee, 2016; Dalmasso et al., 2020; Freeman et al., 2017; Izbicki et al., 2017). However, we remark that these approaches also face their own epistemic uncertainty in the estimation. See (Christmann & Steinwart, 2007b; Steinwart et al., 2011; Bai et al., 2021) for recent studies on the epistemic bias in quantile regression.

A.2. Predictive Uncertainty

In certain scenarios, it is not necessary to estimate each uncertainty separately. A distinction between aleatoric and epistemic uncertainties might appear less significant. The user may only concern about the overall uncertainty related to the prediction, which is called predictive uncertainty and can be thought as the summation of the epistemic and aleatoric uncertainties.

The most common way to quantify predictive uncertainty is the *prediction set*: We aim to find a map $\hat{C} : \mathcal{X} \to 2^{\mathcal{Y}}$ which maps an input to a subset of the output space so that for each test point $x_0 \in \mathcal{X}$, the prediction set $\hat{C}(x_0)$ is likely to cover the true outcome y_0 (Vovk et al., 2005; Barber et al., 2019a;b; Lei & Wasserman, 2014; Lei et al., 2015; 2018). This prediction set is also called a prediction interval in the case of regression (Khosravi et al., 2010; 2011). The prediction set communicates predictive uncertainty via a statistical guarantee on the marginal coverage, i.e.,

$$\mathbb{P}(y_0 \in \hat{C}(x_0)) \geq 1 - \delta$$

for a small threshold $\delta > 0$ where the probability is taken with respect to both training data $\mathcal{D}_{tr}$ (epistemic uncertainty) for learning $\hat{C}$ and the test data (x_0, y_0) (aleatoric uncertainty). It is more tempting to obtain a statistical guarantee with only aleatoric uncertainty by considering the probablity conditional on the prediction set, i.e.,

$$\mathbb{P}(y_0 \in \hat{C}(x_0)|\hat{C}) \geq 1 - \delta. \tag{19}$$

However, this guarantee is in general very hard to achieve in the finite-sample case. Even asymptotically, (19) is not easy to achieve unless we have a very simple structure on the data distribution (Rosenfeld et al., 2018; Zhang et al., 2019). A recent study show that (19) could hold in the finite-sample sense if we could leverage a set of validation data (Chen et al., 2021).

B. Theory of Neural Tangent Kernel

In this section, we provide a brief review on the theory of neural tangent kernel and discuss one of its results employed by our work.

NTK (Jacot et al., 2018) has attracted a lot of attention recently since it provides a new perspective on training dynamics, generalization, and expressibility of over-parameterized neural networks. Recent papers show that when training an over-parametrized neural network, the weight matrix at each layer is close to its initialization (Li & Liang, 2018; Du et al., 2018). An over-parameterized neural network can rapidly reduce training error to zero via gradient descent, and thus finds a global minimum despite the objective function being non-convex (Du et al., 2019; Allen-Zhu et al., 2019a;b;c; Zou et al., 2020). Moreover, the trained network also exhibits good generalization property (Cao & Gu, 2019; 2020). These observations are implicitly described by a notion, NTK, suggested by Jacot et al. (2018). This kernel is able to characterize the training behavior of sufficiently wide fully-connected neural networks and build a new connection between neural networks and kernel methods. Another line of work is to extend NTK for fully-connected neural networks to CNTK for convolutional neural networks (Arora et al., 2019; Yang, 2019; Li et al., 2019). NTK is a fruitful and rapidly growing area.

We focus on the work that is relevant to our epistemic uncertainty quantification: Since the gradient of over-parameterized neural networks is nearly constant and close to its initialization, training networks is similar to kernel regression where the feature map of the kernel is the gradient of networks (Lee et al., 2019). This observation helps us to establish the equivalence between the NN predictor and the KR predictor. We provide more details about this point.

We consider a fully-connected neural network defined formally as follows. Denote $g^{(0)}(x) = x$ and $d_0 = d$. Let

$$f^{(l)}(x) = W^{(l)}g^{(l-1)}(x), \ g^{(l)}(x) = \sqrt{\frac{2}{d_l}}\sigma(f^{(l)}(x))$$

where $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix in the l-th layer, σ is the coordinate-wise ReLu activation and $l \in [L]$. The output of the neural network is

$$h_\theta(x) = f^{(L+1)}(x) = W^{(L+1)}g^{(L)}(x)$$

where $W^{(L+1)} \in \mathbb{R}^{1 \times d_L}$, and $\theta = (W^{(1)}, ..., W^{(L+1)})$ represents all the parameters in the network. Suppose the dimension of θ is p. We randomly initialize all the weights to be i.i.d. $\mathcal{N}(0,1)$ random variables. In other words, we let $\Gamma_0 \sim \mathcal{N}(0, \boldsymbol{I}_p)$ and let $\theta_0 = \theta(0)$ be an instantiation of Γ_0. This initialization method is essentially the He initialization (He et al., 2015). The NTK matrix is defined recursively as (Jacot et al., 2018; Arora et al., 2019): For $l \in [L]$,

$$\Sigma^{(0)}(x, x') = x^T x' \in \mathbb{R},$$

$$\Lambda^{(l)}(x, x') = \begin{pmatrix} \Sigma^{(l-1)}(x, x) & \Sigma^{(l-1)}(x, x') \\ \Sigma^{(l-1)}(x, x') & \Sigma^{(l-1)}(x', x') \end{pmatrix} \in \mathbb{R}^{2 \times 2},$$

$$\Sigma^{(l)}(x, x') = 2\mathbb{E}_{(u,v) \sim \mathcal{N}(0, \Lambda(l))}[\sigma(u)\sigma(v)] \in \mathbb{R}.$$

We also define a derivative covariance:

$$\Sigma^{(l)\prime}(x, x') = 2\mathbb{E}_{(u,v) \sim \mathcal{N}(0, \Lambda(l))}[\sigma'(u)\sigma'(v)] \in \mathbb{R},$$

The final (population) NTK matrix is defined as

$$K(x, x') = \sum_{l=1}^{L+1}\left(\Sigma^{(l-1)}(x, x')\prod_{s=l}^{L+1}\Sigma^{(s)\prime}(x, x')\right)$$

where $\Sigma^{(L+1)\prime}(x, x') = 1$ for convenience. Let $\langle \cdot, \cdot \rangle$ be the standard inner product in $\mathbb{R}^p$. Denote $J(\theta; x) := \nabla_\theta h_\theta(x)$. The empirical (since the initialization is random) NTK matrix is defined as $K_\theta(x, x') = \langle J(\theta; x); J(\theta; x') \rangle$. One of the most important statement in the NTK theory is that the empirical NTK matrix converges to the population NTK matrix as the width of the network increases (Jacot et al., 2018; Yang, 2019; 2020; Arora et al., 2019). In practice we can use the empirical NTK matrix to approximate the population NTK matrix.

Based on NTK, we can derive that the neural network (NN) predictor is essentially a kernel regression (KR) predictor whose kernel is the NTK kernel. More precisely, we focus on the regression problem and consider the following regularized square loss function

$$\hat{R}(h_\theta) = \frac{1}{n} \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2 + \lambda \|\theta\|_2^2. \tag{20}$$

We remark that in the theory of NTK, the training manner is typically the standard gradient descent/gradient flow (feeding the network with the entire training data) (Du et al., 2019; Lee et al., 2019; Arora et al., 2019; Zou et al., 2020) so the uncertainty from the ordering of the data is of less importance. Therefore, we regard the random initialization Γ_0 as the only uncertainty from the procedural variability Γ.

Suppose the network is trained under the gradient flow: We minimize the empirical risk $\hat{R}(h_\theta)$ by gradient descent with infinitesimally small learning rate: $\frac{d\theta(t)}{dt} = -\nabla \hat{R}(h_\theta)$.

Let $h(\theta(t); x)$ be the network output at time t. Then after training, the final output of the network is

$$h_{nn}(\hat{\theta}; x) = \lim_{t \to \infty} h(\theta(t); x). \tag{21}$$

Note that there is procedural variability in $\hat{\theta}$ because $\theta_0 = \theta(0)$ is random.

On the other hand, we build a kernel regression based on the NTK. Recall that we let $\boldsymbol{K} = (K(x_i, x_j))_{i,j=1,...,n} \in \mathbb{R}^{n \times n}$ be the NTK Gram matrix evaluated on these training data. For a test point $x_0 \in \mathbb{R}^d$, we let $K(x_0, \boldsymbol{x}) \in \mathbb{R}^n$ be the kernel value evaluated between the testing point and all training points, i.e., $K(x_0, \boldsymbol{x}) := (K(x_0, x_1), K(x_0, x_2), ..., K(x_0, x_n))^T$. The prediction of the kernel regression using NTK with respect to an initial $h(\theta_0; \cdot)$ is

$$h_{kr}(x_0) = h(\theta_0; x_0) + K(x_0, \boldsymbol{x})^T (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1} (\boldsymbol{y} - h(\theta_0; \boldsymbol{x})) \tag{22}$$

In other words, $h_{kr}(x) - h(\theta_0; x)$ is the solution to the kernel ridge regression problem:

$$\arg \min_{g \in \bar{\mathcal{H}}} \frac{1}{n} \sum_{i=1}^{n} (y_i - h(\theta_0; x_i) - g(x_i))^2 + \lambda \|g\|_{\bar{\mathcal{H}}}^2 \tag{23}$$

where λ is a regularization hyper-parameter and $\bar{\mathcal{H}}$ is the reproducing kernel Hilbert space constructed from the NTK kernel K (Berlinet & Thomas-Agnan, 2011).

Note that there is procedural variability in $h_{kr}(x)$ because θ_0 is random.

Previous work has established some equivalence between the NN predictor $h_{nn}(x)$ and the KR predictor $h_{kr}(x)$ based on a wide variety of assumptions; See below. We do not intend to dig into those detailed theorems and assume for simplicity that the following result holds:

Assumption B.1. Let $d_{\min} = \min\{d_1, ..., d_L\}$. We have

$$\lim_{d_{\min} \to \infty} \mathbb{E}_{\Gamma_0}[h_{nn}(\hat{\theta}; x)] = \mathbb{E}_{\Gamma_0}[h_{kr}(x)]. \tag{24}$$

where $\mathbb{E}_{\Gamma_0}$ is taken with respect to the random initialization Γ_0.

In the case of $\lambda = 0$ (no regularization term in the loss), Arora et al. (2019); Lee et al. (2019) have established the equivalence between the NN predictor and the KR predictor, as described below.

In Arora et al. (2019), their KR predictor is given by

$$h_{kr}(x_0) = K(x_0, \boldsymbol{x})^T (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1} \boldsymbol{y} \tag{25}$$

without the initial term $h(\theta_0; x_0)$. This is the standard kernel ridge regressor, which corresponds to kernel regression under gradient flow with zero initialization. Then they apply a small multiplier $\kappa > 0$, and modify the final output of the neural network to be

$$h'_{nn}(x) = \kappa h_{nn}(\hat{\theta}; x). \tag{26}$$

Their Theorem 3.2 rigorously establishes the equivalence between the (modified) NN predictor h'_{nn} (26) and the KR predictor h_{kr} (25) using the NTK. Informally, with high probability with respect to the random initialization Γ_0, we have $h_{kr}(x) \approx h'_{nn}(x)$.

Simultaneously, Lee et al. (2019) introduces the *linearized neural network*, i.e, they replace the outputs of the neural network by their first order Taylor expansion. They obtain the final output of the linearized neural network (meaning $t \to \infty$ in the gradient descent) as

$$h^{lin}(\hat{\theta}; x) := h(\theta_0; x) + K(x_0, \boldsymbol{x})^T \boldsymbol{K}^{-1}(\boldsymbol{y} - h(\theta_0; \boldsymbol{x})) \tag{27}$$

which corresponds to our (22) when $\lambda = 0$. They show that in the infinite width limit (which is less realistic than Arora et al. (2019)), we have informally

$$h^{lin}(\hat{\theta}; x) \approx h_{nn}(\hat{\theta}; x).$$

In the case of $\lambda \geq 0$, He et al. (2020); Hu et al. (2020) consider to add the weight regularization in the both loss functions, i.e., the λ in (20) and (23). These papers have shown certain equivalence between the NN predictor $h_{nn}(x)$ and the KR predictor $h_{kr}(x)$, which is similar to (24). They adopt a wide variety of assumptions, such as infinite width of networks, or zero-valued initialization $h(\theta_0; \cdot)$, or a small multiplier κ, or normalized training inputs $\|x_i\|_2 = 1$. Their results are also of different forms. The one we adopt in this paper is close to (Lee et al., 2019) but with a weight regularization (He et al., 2020). Since a uniform statement is not available, we do not intend to write down a specific theorem for (24) in this paper. Interested readers can refer to the above papers.

C. Proofs

In this section, we provide detailed proofs for results in the main paper.

Proof of Proposition 2.1. For any $h \in \mathcal{H}$, we have by the definition of f^*

$$\mathbb{E}_{Y \sim \pi_{Y|X}}[\mathcal{L}(f^*(X), Y)|X] \leq \mathbb{E}_{Y \sim \pi_{Y|X}}[\mathcal{L}(h(X), Y)|X]$$

Taking expectation with respect to X, we obtain

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(f^*(X), Y)] \\
=& \mathbb{E}[\mathbb{E}_{Y \sim \pi_{Y|X}}[\mathcal{L}(f^*(X), Y)|X]] \\
\leq& \mathbb{E}[\mathbb{E}_{Y \sim \pi_{Y|X}}[\mathcal{L}(h(X), Y)|X]] \\
=& \mathbb{E}[\mathcal{L}(h(X), Y)].
\end{aligned}$$

So f^* is the true risk minimizer in (1), i.e., $h_{\theta^*} = f^*$. $\qquad\square$

Proof of Theorem 3.1. We first note that

$$\text{UQ}_{MU} = h_{\theta^*} - f^*$$

which is a deterministic value. So UQ_{MU} is uncorrelated with the UQ_{PV} and UQ_{DV} and

$$\text{Var}(\text{UQ}_{MU} + \text{UQ}_{PV} + \text{UQ}_{DV}) = \text{Var}(\text{UQ}_{PV} + \text{UQ}_{DV}).$$

Furthermore, we have

$$\begin{aligned}
\text{Cov}(\text{UQ}_{PV}, \text{UQ}_{DV}) &= \text{Cov}(\mathbb{E}[\text{UQ}_{PV}|\hat{\pi}_{\mathcal{D}_{tr}}], \mathbb{E}[\text{UQ}_{DV}|\hat{\pi}_{\mathcal{D}_{tr}}]) + \mathbb{E}[\text{Cov}(\text{UQ}_{PV}, \text{UQ}_{DV}|\hat{\pi}_{\mathcal{D}_{tr}})] \\
&= 0 + 0 = 0
\end{aligned}$$

where the first equality uses conditioning, and the second equality is because $\mathbb{E}[\text{UQ}_{PV}|\hat{\pi}_{\mathcal{D}_{tr}}]$, and also UQ_{DV} given $\hat{\pi}_{\mathcal{D}_{tr}}$, are constant. This shows that UQ_{PV} and UQ_{DV} are uncorrelated, so

$$\begin{aligned}
&\text{Var}(\text{UQ}_{PV} + \text{UQ}_{DV}) = \text{Var}(\text{UQ}_{PV}) + \text{Var}(\text{UQ}_{DV}) \\
=& \text{Var}(\mathbb{E}[\text{UQ}_{PV}|\hat{\pi}_{\mathcal{D}_{tr}}]) + \mathbb{E}[\text{Var}(\text{UQ}_{PV}|\hat{\pi}_{\mathcal{D}_{tr}})] + \text{Var}(\mathbb{E}[\text{UQ}_{DV}|\hat{\pi}_{\mathcal{D}_{tr}}]) + \mathbb{E}[\text{Var}(\text{UQ}_{DV}|\hat{\pi}_{\mathcal{D}_{tr}})] \\
=& 0 + \mathbb{E}[\text{Var}(\text{UQ}_{PV}|\hat{\pi}_{\mathcal{D}_{tr}})] + \text{Var}(\mathbb{E}[\text{UQ}_{DV}|\hat{\pi}_{\mathcal{D}_{tr}}]) + 0 \\
=& \mathbb{E}[\text{Var}(\text{UQ}_{PV}|\hat{\pi}_{\mathcal{D}_{tr}})] + \text{Var}(\mathbb{E}[\text{UQ}_{DV}|\hat{\pi}_{\mathcal{D}_{tr}}])
\end{aligned}$$

where the second equality follows from conditioning, and the third equality is again because $\mathbb{E}[UQ_{PV}|\hat{\pi}_{\mathcal{D}_{tr}}]$, and UQ_{DV} given $\hat{\pi}_{\mathcal{D}_{tr}}$, are constant.

$\square$

Proof of Theorem 3.2. In terms of estimation bias, we note that

$$\mathbb{E}[h_{\hat{\theta}_1}(x_0)|\hat{\pi}_{\mathcal{D}_{tr}}] = \mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}].$$

$$\mathbb{E}[\bar{h}_{m'}(x_0)|\hat{\pi}_{\mathcal{D}_{tr}}] = \frac{1}{m'}\sum_{i=1}^{m'}\mathbb{E}[h_{\hat{\theta}_i}(x_0)|\hat{\pi}_{\mathcal{D}_{tr}}] = \mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}].$$

This implies that the deep ensemble prediction and the single model prediction are the same. In terms of variance, note that $h_{\hat{\theta}_1}(x_0), ..., h_{\hat{\theta}_{m'}}(x_0)$ are conditionally independent given $\hat{\pi}_{\mathcal{D}_{tr}}$. Therefore deep ensemble gives

$$\text{Var}(\bar{h}_{m'}(x_0)|\hat{\pi}_{\mathcal{D}_{tr}}) = \frac{1}{(m')^2}\sum_{i=1}^{m'}\text{Var}(h_{\hat{\theta}_i}(x_0)|\hat{\pi}_{\mathcal{D}_{tr}}) = \frac{1}{m'}\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})$$

while the single model prediction gives $\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})$. Now using conditioning, we have

$$
\begin{aligned}
&\text{Var}(\bar{h}_{m'}(x_0)) \\
=&\text{Var}(\mathbb{E}[\bar{h}_{m'}(x_0))|\hat{\pi}_{\mathcal{D}_{tr}}]) + \mathbb{E}[\text{Var}(\bar{h}_{m'}(x_0)|\hat{\pi}_{\mathcal{D}_{tr}})] \\
=&\text{Var}(\mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}]) + \mathbb{E}[\frac{1}{m'}\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})] \\
\leq&\text{Var}(\mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}]) + \mathbb{E}[\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})] \\
=&\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)) \\
=&\text{Var}(h_{\hat{\theta}_1}(x_0))
\end{aligned}
$$

as desired.

$\square$

Proof of Proposition 4.1. We first let $\tilde{y}_i = y_i - h_0(x_i)$ be the shifted label. With a little abuse of notation, $\hat{\pi}_{\mathcal{D}_{tr}}$ should be understood as the empirical distribution on the shifted training data $\{(x_i, \tilde{y}_i) : i = 1, ..., n\}$. Now consider the kernel ridge regression problem

$$\arg\min_{g\in\bar{\mathcal{H}}} \frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - g(x_i))^2 + \lambda\|g\|_{\mathcal{H}}^2.$$

Standard theory in kernel ridge regression (e.g., Theorem 1 in Smale & Zhou (2005)) shows that the closed form of the solution to the above problem is

$$g^*(x_0) = K(x_0, \boldsymbol{x})^T(\boldsymbol{K} + \lambda n\boldsymbol{I})^{-1}\tilde{\boldsymbol{y}} = K(x_0, \boldsymbol{x})^T(\boldsymbol{K} + \lambda n\boldsymbol{I})^{-1}(\boldsymbol{y} - h_0(\boldsymbol{x}))$$

which corresponds to the definition of $\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x_0) - h_0(x_0)$.

$\square$

Proof of Theorem 4.2. Let $\bar{h}(x) = \bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}; x)$ for short. We first note that $\text{Var}(\bar{h}(x_0) - h_0(x_0)) = \text{Var}(\bar{h}(x_0))$ where the variance is only respect to the training data. Hence it is sufficient to study the variance of $\bar{h}(x_0) - h_0(x_0)$ which is the solution of the problem (10). Let Φ be the feature map associated with the NTK kernel. We apply Proposition 5 in Debruyne et al. (2008) (see also Christmann & Steinwart (2007a)) to calculate the influence function. With a slight abuse of notation, we use $IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}})(x_0)$ to represent the influence function at the fixed test point x_0 in the main paper, and reserve $IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}})$ as a function only in this proof to be consistent with Debruyne et al. (2008).

$$
\begin{aligned}
IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}}) &= -S^{-1}(2\lambda(\bar{h} - h_0)) + \mathcal{L}'(z_y - h_0(z_x) - (\bar{h}(z_x) - h_0(z_x)))S^{-1}\Phi(z_x) \\
&= -S^{-1}(2\lambda(\bar{h} - h_0)) + 2(z_y - \bar{h}(z_x))S^{-1}\Phi(z_x)
\end{aligned}
$$

where $S : \mathcal{H} \to H$ is defined by

$$S(f) = 2\lambda f + \mathbb{E}_{\hat{\pi}_{\mathcal{D}_{tr}}}[\mathcal{L}''(Y - \bar{h}(X))\langle\Phi(X), f\rangle\Phi(X)]$$

To do this, we need to obtain S and S^{-1}. Since the loss function is $\mathcal{L}(\hat{y} - y) = (\hat{y} - y)^2$, we have

$$S(f) = 2\lambda f + \mathbb{E}_{\hat{\pi}_{\mathcal{D}_{tr}}}[\mathcal{L}''(Y - \bar{h}(X))\langle\Phi(X), f\rangle\Phi(X)] = 2\lambda f + \frac{2}{n}\sum_{j=1}^{n}f(x_j)\Phi(x_j).$$

Suppose $S^{-1}(2\lambda(\bar{h} - h_0)) = g_1$. Then at x_0, we have

$$2\lambda(\bar{h}(x_0) - h_0(x_0)) = S(g_1(x_0)) = 2\lambda g_1(x_0) + \frac{2}{n}\sum_{j=1}^{n}g_1(x_j)K(x_0, x_j).$$

Hence

$$g_1(x_0) = \bar{h}(x_0) - h_0(x_0) - \frac{1}{\lambda n}\sum_{j=1}^{n}g_1(x_j)K(x_0, x_j)$$

This implies that we need to evaluate $g_1(x_j)$ on training data first, which is straightforward by letting $x_0 = x_1, ..., x_n$:

$$g_1(\boldsymbol{x}) = \bar{h}(\boldsymbol{x}) - h_0(\boldsymbol{x}) - \frac{1}{\lambda n}\boldsymbol{K}g_1(\boldsymbol{x})$$

so

$$g_1(\boldsymbol{x}) = (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1}(\lambda n)\left(\bar{h}(\boldsymbol{x}) - h_0(\boldsymbol{x})\right)$$

and

$$\begin{aligned} g_1(x_0) &= \bar{h}(x_0) - h_0(x_0) - \frac{1}{\lambda n}K(x_0, \boldsymbol{x})^T g_1(\boldsymbol{x}) \\ &= \bar{h}(x_0) - h_0(x_0) - K(x_0, \boldsymbol{x})^T(\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1}\left(\bar{h}(\boldsymbol{x}) - h_0(\boldsymbol{x})\right) \end{aligned}$$

Next we compute $S^{-1}\Phi(z_x) = g_2$. At x_0, we have

$$K(z_x, x_0) = \Phi(z_x)(x_0) = S(g_2(x_0)) = 2\lambda g_2(x_0) + \frac{2}{n}\sum_{j=1}^{n}g_2(x_j)K(x_0, x_j)$$

Hence

$$g_2(x_0) = \frac{1}{2\lambda}K(z_x, x_0) - \frac{1}{\lambda n}\sum_{j=1}^{n}g_2(x_j)K(x_0, x_j)$$

This implies that we need to evaluate $g_2(x_j)$ on training data first, which is straightforward by letting $x_0 = x_1, ..., x_n$:

$$g_2(\boldsymbol{x}) = \frac{1}{2\lambda}K(z_x, \boldsymbol{x}) - \frac{1}{\lambda n}\boldsymbol{K}g_2(\boldsymbol{x})$$

so

$$g_2(\boldsymbol{x}) = (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1}(\frac{n}{2})K(z_x, \boldsymbol{x})$$

and

$$\begin{aligned} g_2(x_0) &= \frac{1}{2\lambda}K(z_x, x_0) - \frac{1}{\lambda n}K(x_0, \boldsymbol{x})^T g_2(\boldsymbol{x}) \\ &= \frac{1}{2\lambda}K(z_x, x_0) - \frac{1}{2\lambda}K(x_0, \boldsymbol{x})^T(\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1}K(z_x, \boldsymbol{x}) \end{aligned}$$

Combing previous results, we obtain

$$
\begin{aligned}
IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}})(x_0) = & -\bar{h}(x_0) + h_0(x_0) + K(x_0, \boldsymbol{x})^T (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1} \left(\bar{h}(\boldsymbol{x}) - h_0(\boldsymbol{x}) \right) \\
& + 2(z_y - \bar{h}(z_x)) \left(\frac{1}{2\lambda} K(z_x, x_0) - \frac{1}{2\lambda} K(x_0, \boldsymbol{x})^T (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1} K(z_x, \boldsymbol{x}) \right) \\
= & K(x_0, \boldsymbol{x})^T (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1} \left(\bar{h}(\boldsymbol{x}) - h_0(\boldsymbol{x}) - \frac{1}{\lambda} (z_y - \bar{h}(z_x)) K(z_x, \boldsymbol{x}) \right) \\
& - \bar{h}(x_0) + h_0(x_0) + \frac{1}{\lambda} (z_y - \bar{h}(z_x)) K(z_x, x_0) \\
= & K(x_0, \boldsymbol{x})^T (\boldsymbol{K} + \lambda n \boldsymbol{I})^{-1} M_z(\boldsymbol{x}) - M_z(x_0)
\end{aligned}
$$

as desired. □

Proof of Theorem 4.3. In this proof, we write $\hat{\pi}_{\mathcal{D}_{tr}}^n$ rather than $\hat{\pi}_{\mathcal{D}_{tr}}$ to emphsize the size of training dataset n. Since $T(P)$ is ρ_∞-Hadamard differentiable at π, we have the following asymptotic normality by the central limit theorem (Fernholz, 2012; Shao & Tu, 2012):

$$
\sqrt{n}(T(\hat{\pi}_{\mathcal{D}_{tr}}^n) - T(\pi)) \xrightarrow{d} \mathcal{N}(0, \sigma^2)
$$

where $\sigma^2 = \mathbb{E}_{z \sim \pi}[IF^2(z; T, \pi)]$. In other words,

$$
\sqrt{n}(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}^n; x_0) - h_{\theta^*}(x_0)) \xrightarrow{d} \mathcal{N}(0, \sigma^2)
$$

The uniform integrability implies (by Skorokhod's theorem and Vitali convergence theorem) that

$$
\mathbb{E}\left[\left(\sqrt{n}(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}^n; x_0) - h_{\theta^*}(x_0)) \right)^2 \right] \to \mathbb{E}\left[\mathcal{N}(0, \sigma^2)^2 \right] = \sigma^2
$$

which gives that

$$
\lim_{n \to \infty} |n \mathrm{Var}(\bar{h}(\hat{\pi}_{\mathcal{D}_{tr}}^n; x_0)) - \sigma^2| = 0 \tag{28}
$$

Since $T(P)$ is continuously ρ_∞-Hadamard differentiable (which implies continuously Gâteaux differentiable) at $\hat{\pi}_{\mathcal{D}_{tr}}^n$, we have that

$$
\lim_{t \to 0} \left[\frac{T(\hat{\pi}_{\mathcal{D}_{tr}}^n + t(\delta_z - T(\hat{\pi}_{\mathcal{D}_{tr}}^n))) - T(\hat{\pi}_{\mathcal{D}_{tr}}^n)}{t} - IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}}^n) \right] = 0, \quad \text{uniformly in } z
$$

where implies that for every n, there exists a t_n (depending on $\hat{\pi}_{\mathcal{D}_{tr}}^n$), such that

$$
\left| \frac{T(\hat{\pi}_{\mathcal{D}_{tr}}^n + t_n(\delta_z - T(\hat{\pi}_{\mathcal{D}_{tr}}^n))) - T(\hat{\pi}_{\mathcal{D}_{tr}}^n)}{t_n} - IF(z; T, \hat{\pi}_{\mathcal{D}_{tr}}^n) \right| \leq \epsilon, \quad \text{uniformly in } z. \tag{29}
$$

Without loss of generality, we assume $t_n \to 0$ as $n \to \infty$.

Again, note that $T(P)$ is continuously ρ_∞-Hadamard differentiable (which implies continuously Gâteaux differentiable) at π. Then since $t_n \to 0$ and $\rho_\infty(\hat{\pi}_{\mathcal{D}_{tr}}^n, \pi) \xrightarrow{a.s.} 0$, the definition implies that

$$
\lim_{n \to \infty} \left[\frac{T(\hat{\pi}_{\mathcal{D}_{tr}}^n + t_n(\delta_z - T(\hat{\pi}_{\mathcal{D}_{tr}}^n))) - T(\hat{\pi}_{\mathcal{D}_{tr}}^n)}{t_n} - L_\pi(\delta_z - \hat{\pi}_{\mathcal{D}_{tr}}^n) \right] = 0, \quad a.s. \text{ uniformly in } z
$$

where $L_\pi(\delta_z - \hat{\pi}_{\mathcal{D}_{tr}}^n)$ is the Gâteaux differential of T at π. Therefore, there exists a N_1, such that for $n \geq N_1$,

$$
\left| \frac{T(\hat{\pi}_{\mathcal{D}_{tr}}^n + t_n(\delta_z - T(\hat{\pi}_{\mathcal{D}_{tr}}^n))) - T(\hat{\pi}_{\mathcal{D}_{tr}}^n)}{t_n} - L_\pi(\delta_z - \hat{\pi}_{\mathcal{D}_{tr}}^n) \right| \leq \epsilon, \quad a.s. \text{ uniformly in } z. \tag{30}
$$

As Gâteaux differential is a linear functional, we have that

$$
\begin{aligned}
L_\pi(\delta_z - \hat{\pi}_{\mathcal{D}_{tr}}^n) &= L_\pi(\delta_z - \pi) + L_\pi(\pi - \hat{\pi}_{\mathcal{D}_{tr}}^n) \\
&= L_\pi(\delta_z - \pi) + \frac{1}{n} \sum_{i=1}^n L_\pi(\pi - \delta_{z_i}) \\
&= IF(z; T, \pi) - \frac{1}{n} \sum_{i=1}^n IF(z_i; T, \pi)
\end{aligned} \tag{31}
$$

Therefore, combing (29), (30) and (31), we have that for $n \geq N_1$,

$$\left| IF(z;T,\hat{\pi}^n_{\mathcal{D}_{tr}}) - \left(IF(z;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right) \right| \leq 2\epsilon, \quad \text{uniformly in } z.$$

Hence

$$\left| IF^2(z;T,\hat{\pi}^n_{\mathcal{D}_{tr}}) - \left(IF(z;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right)^2 \right|$$

$$= \left| IF(z;T,\hat{\pi}^n_{\mathcal{D}_{tr}}) + \left(IF(z;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right) \right| \left| IF(z;T,\hat{\pi}^n_{\mathcal{D}_{tr}}) - \left(IF(z;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right) \right|$$

$$\leq \epsilon \left(2\left| \left(IF(z;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right) \right| + \epsilon \right)$$

$$= 2\epsilon \left| \left(IF(z;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right) \right| + \epsilon^2.$$

Since this inequality holds for uniformly in z, take $z = z_j$ where $j \in [n]$, we obtain

$$\left| \frac{1}{n}\sum_{j=1}^{n} IF^2(z_j;T,\hat{\pi}^n_{\mathcal{D}_{tr}}) - \frac{1}{n}\sum_{j=1}^{n}\left(IF(z_j;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right)^2 \right|$$

$$\leq 2\epsilon\frac{1}{n}\sum_{j=1}^{n}\left| \left(IF(z_j;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right) \right| + \epsilon^2$$

$$\leq 2\epsilon\sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(IF(z_j;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right)^2} + \epsilon^2 \tag{32}$$

Note that by the strong law of large number, we have that

$$\frac{1}{n}\sum_{j=1}^{n}\left(IF(z_j;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right)^2$$

$$= \frac{1}{n}\sum_{j=1}^{n} IF^2(z_j;T,\pi) - \left(\frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right)^2$$

$$\xrightarrow{a.s.} \sigma^2$$

since $\mathbb{E}_{z\sim\pi}[IF^2(z;T,\pi)] = \sigma^2$ and $\mathbb{E}_{z\sim\pi}[IF(z;T,\pi)] = 0$. Therefore (32) implies that

$$\limsup_{n\to\infty} \left| \frac{1}{n}\sum_{j=1}^{n} IF^2(z_j;T,\hat{\pi}^n_{\mathcal{D}_{tr}}) - \frac{1}{n}\sum_{j=1}^{n}\left(IF(z_j;T,\pi) - \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,\pi) \right)^2 \right| \leq 2\epsilon\sigma + \epsilon^2, \quad a.s.$$

and thus

$$\lim_{n\to\infty} \left| \frac{1}{n}\sum_{j=1}^{n} IF^2(z_j;T,\hat{\pi}^n_{\mathcal{D}_{tr}}) - \sigma^2 \right| = 0, \quad a.s.$$

Combing it with (28), we obtain

$$\lim_{n\to\infty} \left| \frac{1}{n}\sum_{j=1}^{n} IF^2(z_j;T,\hat{\pi}^n_{\mathcal{D}_{tr}}) - n\text{Var}(\bar{h}(\hat{\pi}^n_{\mathcal{D}_{tr}};x_0)) \right| = 0, \quad a.s.$$

$\square$

Proof of Theorem 5.1. We have, by Assumptions (14) and (16), that

$$\hat{\psi}_i(x_0) - h_{\theta^*}(x_0) = \sqrt{K}\nu Y_i(1 + o_p(1))$$

$$\bar{\psi}(x_0) - h_{\theta^*}(x_0) = \sqrt{K}\nu \bar{Y}(1 + o_p(1))$$

where Y_i are i.i.d. $\mathcal{N}(0,1)$, $\bar{Y} = \frac{1}{K}\sum_{i=1}^{K} Y_i$ and $\nu = \sqrt{\sigma^2/n + \tau^2/m'} = \sqrt{\sigma^2/n + \tau^2/K}$. Then, we note some basic facts that $\bar{Y} \sim \frac{1}{\sqrt{K}}\mathcal{N}(0,1)$, and $\sum_{i=1}^{K}(Y_i - \bar{Y})^2 \sim \chi^2_{K-1}$, and these two variables are independent of each other. Therefore by the continuous mapping theorem, we have

$$\frac{S}{\sqrt{K}} = \sqrt{\frac{\sum_{i=1}^{K}(\hat{\psi}_i(x_0) - \bar{\psi}(x_0))^2}{K(K-1)}} \Rightarrow \sqrt{\frac{K\sum_{i=1}^{K}(\nu Y_i - \nu\bar{Y})^2}{K(K-1)}} = \sqrt{\frac{\sum_{i=1}^{K}(\nu Y_i - \nu\bar{Y})^2}{K-1}},$$

and thus

$$\frac{S^2/K}{\nu^2} \Rightarrow \frac{\sum_{i=1}^{K}(\nu Y_i - \nu\bar{Y})^2}{\nu^2(K-1)} = \frac{1}{K-1}\sum_{i=1}^{K}(Y_i - \bar{Y})^2 = \frac{\chi^2_{K-1}}{K-1}.$$

The CI (18) then follows immediately from the above result, since it implies

$$P\left(\frac{\chi^2_{K-1,\alpha/2}}{K-1} \leq \frac{S^2/K}{\nu^2} \leq \frac{\chi^2_{K-1,1-\alpha/2}}{K-1}\right) \to 1-\alpha$$

or equivalently

$$P\left(\frac{S^2/K}{\chi^2_{K-1,1-\alpha/2}/(K-1)} \leq \nu^2 \leq \frac{S^2/K}{\chi^2_{K-1,\alpha/2}/(K-1)}\right) \to 1-\alpha$$

which concludes the result. $\square$

D. Experiments: Details and More Results

D.1. Experimental Details

We provide more details about obtaining the ground-truth data variance and procedural variance separately. As we discussed in Section 7, we have employed that

$$\text{Var}(h(x_0)) \approx \frac{1}{J-1}\sum_{j=1}^{J}\left(h_j(x_0) - \frac{1}{J}\sum_{i=1}^{J}h_i(x_0)\right)^2 := \tilde{\text{Var}}(h(x_0))$$

where J is the number of experimental repetitions (typically vary large). Note that we can allow $h(x_0)$ to be a single model predictor $h_{\hat{\theta}_1}(x_0)$ or a deep ensemble predictor $\bar{h}_{m'}(x_0)$. It follows from Theorem 3.2 in Section 3.1 that

$$\mathbb{E}[\text{Var}(h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}})] = \frac{m'}{m'-1}\left(\text{Var}(h_{\hat{\theta}_1}(x_0)) - \text{Var}(\bar{h}_{m'}(x_0))\right),$$

$$\text{Var}(\mathbb{E}[h(\Gamma, \hat{\pi}_{\mathcal{D}_{tr}}; x_0)|\hat{\pi}_{\mathcal{D}_{tr}}]) = \frac{m'}{m'-1}\text{Var}(\bar{h}_{m'}(x_0)) - \frac{1}{m'-1}\text{Var}(h_{\hat{\theta}_1}(x_0)).$$

Hence, by replacing $\text{Var}(h_{\hat{\theta}_1}(x_0))$, $\text{Var}(\bar{h}_{m'}(x_0))$ with $\tilde{\text{Var}}(h_{\hat{\theta}_1}(x_0))$, $\tilde{\text{Var}}(\bar{h}_{m'}(x_0))$ respectively in the above equations, we obtain the ground truth data variance and procedural variance.

Throughout our experiments, we use a two-layer fully-connected NN with 1024 hidden neurons as the base predictor based on the NTK specification in Section B and it is trained using the regularized square loss (20) with regularization hyperparameter $\lambda = 0.1^3$. The learning rate is properly tuned based on the specific dataset and it is generally small to make the training procedure operate in the NTK regime (Jacot et al., 2018; Du et al., 2019; Lee et al., 2019; Zhang et al., 2020).

D.2. Additional Experiments

We conduct experiments on more datasets.

Synthetic Datasets 2: $X \sim \mathcal{N}(0, 0.1^2 \boldsymbol{I}_d)$ and $Y \sim \sum_{i=1}^{d} \exp(X^{(i)}) + (X^{(i)})^2 + \mathcal{N}(0, (0.1d)^2)$. The training set $\mathcal{D}_{tr} = \{(x_i, y_i) : i = 1, ..., n\}$ are formed by i.i.d. samples of (X, Y) with sample size $n = 200$. We use $x_0 = (0.1, 0.1, ..., 0.1)$ as the fixed test point and set $K = m' = 5$, $m = 50$. The results are displayed in Table 5.

Synthetic Datasets 3: $X \sim \mathcal{N}(0, 0.1^2 \boldsymbol{I}_d)$ and $Y \sim \sum_{i=1}^{d} \cos(X^{(i)}) + (X^{(i)})^3 + \mathcal{N}(0, (0.1d)^2)$. The training set $\mathcal{D}_{tr} = \{(x_i, y_i) : i = 1, ..., n\}$ are formed by i.i.d. samples of (X, Y) with sample size $n = 500$. We use $x_0 = (0.1, 0.1, ..., 0.1)$ as the fixed test point and set $K = m' = 5$, $m = 50$. The results are displayed in Table 6.

	τ^2 GT	τ^2 EV	τ^2 Diff
$d = 2$	$0.1 * 10^{-3}$	$0.1 * 10^{-3}$	$-0.0 * 10^{-3}$
$d = 4$	$1.7 * 10^{-3}$	$1.4 * 10^{-3}$	$-0.3 * 10^{-3}$
$d = 8$	$1.8 * 10^{-2}$	$1.3 * 10^{-2}$	$-0.5 * 10^{-2}$
$d = 16$	$1.4 * 10^{-1}$	$0.9 * 10^{-1}$	$-0.5 * 10^{-1}$
	$\frac{\sigma^2}{n}$ GT	$\frac{\sigma^2}{n}$ IF	$\frac{\sigma^2}{n}$ Diff
$d = 2$	$1.1 * 10^{-3}$	$1.5 * 10^{-3}$	$+0.4 * 10^{-3}$
$d = 4$	$8.1 * 10^{-3}$	$8.7 * 10^{-3}$	$+0.6 * 10^{-3}$
$d = 8$	$4.8 * 10^{-2}$	$5.0 * 10^{-2}$	$+0.2 * 10^{-2}$
$d = 16$	$3.0 * 10^{-1}$	$3.4 * 10^{-1}$	$+0.4 * 10^{-1}$
	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ GT	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ BA	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ Diff
$d = 2$	$1.1 * 10^{-3}$	$1.5 * 10^{-3}$	$+0.4 * 10^{-3}$
$d = 4$	$8.4 * 10^{-3}$	$8.6 * 10^{-3}$	$+0.2 * 10^{-3}$
$d = 8$	$5.2 * 10^{-2}$	$5.6 * 10^{-2}$	$+0.4 * 10^{-2}$
$d = 16$	$3.3 * 10^{-1}$	$3.6 * 10^{-1}$	$+0.3 * 10^{-1}$

Table 5. Epistemic Variance Estimation on Synthetic Datasets 2 with Different Dimensions.

	τ^2 GT	τ^2 EV	τ^2 Diff
$d = 2$	$4.5 * 10^{-4}$	$4.3 * 10^{-4}$	$-0.2 * 10^{-4}$
$d = 4$	$1.1 * 10^{-3}$	$0.8 * 10^{-3}$	$-0.3 * 10^{-3}$
$d = 8$	$1.0 * 10^{-2}$	$0.7 * 10^{-2}$	$-0.3 * 10^{-2}$
$d = 16$	$0.4 * 10^{-1}$	$0.2 * 10^{-1}$	$-0.2 * 10^{-1}$
	$\frac{\sigma^2}{n}$ GT	$\frac{\sigma^2}{n}$ IF	$\frac{\sigma^2}{n}$ Diff
$d = 2$	$5.7 * 10^{-4}$	$6.3 * 10^{-4}$	$+0.6 * 10^{-4}$
$d = 4$	$3.9 * 10^{-3}$	$4.5 * 10^{-3}$	$+0.6 * 10^{-3}$
$d = 8$	$2.1 * 10^{-2}$	$2.4 * 10^{-2}$	$+0.3 * 10^{-2}$
$d = 16$	$0.6 * 10^{-1}$	$0.8 * 10^{-1}$	$+0.2 * 10^{-1}$
	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ GT	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ BA	$\frac{\sigma^2}{n} + \frac{\tau^2}{m'}$ Diff
$d = 2$	$6.6 * 10^{-4}$	$6.9 * 10^{-4}$	$+0.3 * 10^{-4}$
$d = 4$	$4.1 * 10^{-3}$	$4.3 * 10^{-3}$	$+0.2 * 10^{-3}$
$d = 8$	$2.3 * 10^{-2}$	$2.5 * 10^{-2}$	$+0.2 * 10^{-2}$
$d = 16$	$0.7 * 10^{-1}$	$0.8 * 10^{-1}$	$+0.1 * 10^{-1}$

Table 6. Epistemic Variance Estimation on Synthetic Datasets 3 with Different Dimensions.