Reliable and Scalable Variable Importance Estimation via Warm-start and Early Stopping

Zexuan Sun
Department of Statistics
University of Wisconsin–Madison

Garvesh Raskutti
Department of Statistics
University of Wisconsin–Madison

Abstract

As opaque black-box predictive models such as neural networks become more prevalent, the need to develop interpretations for these models is of great interest. The concept of *variable importance* is an interpretability measure that applies to any predictive model and assesses how much a variable or set of variables improves prediction performance. When the number of variables is large, estimating variable importance presents a significant challenge because re-training neural networks or other black-box algorithms requires significant additional computation. In this paper, we address this challenge for algorithms using gradient descent and gradient boosting (e.g. neural networks, gradient-boosted decision trees). By using the ideas of early stopping of gradient-based methods in combination with warm-start using the *dropout* method, we develop a scalable method to estimate variable importance for any algorithm that can be expressed as an *iterative kernel update equation*. Importantly, we provide theoretical guarantees by using the theory for early stopping of kernel-based methods for neural networks with sufficient large width and gradient-boosting decision trees that use symmetric tree as a weaker learner. We also demonstrate the efficacy of our methods through simulations and a real data example which illustrates the computational benefit of early stopping rather than fully re-training the model as well as the increased accuracy of taking initial steps from the dropout solution.

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

1 INTRODUCTION

The widespread use of predictive modeling in crucial sectors like judiciary, healthcare, and education demands models that are both accurate and interpretable. Interpretability is essential for transparency and fairness in impactful decisions, especially given the limitations of black-box methods in sensitive applications (see e.g. (Rudin and Radin, 2019; Guidotti et al., 2018)).

Traditional statistical tools often fail with complex modern data, leading to a shift towards non-parametric and machine learning methods, particularly neural networks, for which interpretability is often derived from their architecture and weights. However, these methods are model-specific, limiting their general applicability (Shrikumar et al., 2017; Sundararajan et al., 2017). Model-agnostic variable importance (VI) techniques are increasingly important as they evaluate variable impact independent for any algorithm. Retraining models without certain variables can benchmark VI but struggles in high-dimensional settings due to computational demands (Feng et al., 2018). Alternative methods like *dropout*, which plugs in input data with dropped features to the full model trained with all features directly, offer computational feasibility but typically underfit the data (Chang et al., 2018).

A viable solution is to use a warm start with the dropout model and apply *early stopping* to improve fit while avoiding the high computational cost of full re-training. Early stopping has a long history and is widely employed in gradient-based algorithms like neural networks (Morgan and Bourlard, 1989), non-parametric regression Raskutti et al. (2014), and boosting (Bühlmann and Yu, 2003; Zhang and Yu, 2005; Wei et al., 2017). As an efficient regularization technique, early stopping has lower computational complexity and can improve dropout underfitting with proper stopping criteria, making it suitable for estimating VI. However, theoretical guarantees for early

stopping in the context of estimating VI remains an open challenge.

In this paper, we propose a general scalable strategy for any gradient-based iterative algorithm to estimate variable importance. Drawing ideas from (Raskutti et al., 2014; Gao et al., 2022), we combine warm-start initialization using the dropout method and early-stopping to estimate the VI efficiently. The key idea is to start the training of a new model without certain variables from a model learned from the original (full model) training data and then stop early to avoid overfitting and reduce computational costs. This idea regarding neural networks was first introduced in Gao et al. (2022); however, no theoretical guarantees were provided. We consider a more general setting and address this theoretical challenge.

1.1 Contributions

We emphasize that our main contributions lie in establishing **precise theoretical guarantees**, which are summarized as follows:

- We propose a general scalable framework with supporting theoretical guarantees to estimate VI efficiently for any iterative algorithm that can be expressed as an *iterative kernel update equation*. Importantly we provide theoretical guarantees for this method by leveraging theory for the early stopping of kernel-based methods (Theorem 4.1).
- Utilizing the *neural tangent kernel* for neural networks with sufficiently large width we apply our general theoretical bound to neural networks (Section 4.3). Further, we use a well-defined kernel to also adapt our bounds to gradient boosted decision trees (Section 4.4). Each of these theoretical results is of independent interest.
- The theoretical bounds are supported in a simulation. We then demonstrate the computational advantages over re-training and the the accuracy advantages over dropout for estimating both variable importance and Shapley values for a both simulated data and an application to understanding the variable importance of predicting flue gas emissions.

1.2 Related Work

Early stopping has a long history (Anderssen and Prenter, 1981; Strand, 1974), with theoretical developments for classification boosting (Jiang, 2004; Zhang and Yu, 2005; Yao et al., 2007), L2-boosting (Bühlmann and Yu, 2003; Bühlmann and Hothorn, 2007), and gradient algorithms in reproducing kernel Hilbert space (RKHS) (Vito et al., 2010; Yao et al., 2007; Raskutti et al., 2014; Wei et al., 2017). We adapt these methods to estimate variable importance (VI) for gradient-based approaches, offering the first theoretical guarantees for neural networks and gradient boosting.

Overparameterized models, including deep networks, are often studied via the Neural Tangent Kernel (NTK) (Jacot et al., 2018), which supports global convergence (Arora et al., 2019; Allen-Zhu et al., 2019; Du et al., 2019a,b) and linear approximation (Lee et al., 2019), with convolutional extensions (Yang, 2020). For gradient-boosted decision trees (GBDTs), kernels remain less explored; Ustimenko et al. (2023) proposed one for symmetric-tree GBDTs. These findings are exploited in this paper to study overparameterized neural networks and GBDTs through a RKHS lens, potentially linking them to theoretical results on early stopping (Raskutti et al., 2014).

Variable importance (VI) has been a fundamental topic in statistical modeling, with early work focusing on linear models (Nathans et al., 2012). Recent advances have extended VI to nonparametric settings (Williamson et al., 2021). Shapley values (Shapley, 1953) have become a prominent tool for feature importance due to their strong theoretical foundation, with development of stochastic estimators (Williamson and Feng, 2020) to reduce the inherent high computational cost. Alternative model-agnostic techniques have been proposed, including permutation (Smith et al., 2020), leave-one-covariate-out (LOCO) (Rinaldo et al., 2019), Our approach scales VI and Shapley estimation to large feature spaces.

Closest to this work is Gao et al. (2022), which estimates VI through a regularized local linear approximation in a scalable way. This work applied specifically to feed-forward neural networks with sufficiently large width. In this paper, our early stopping approaches to any gradient-based method (e.g. finite but large-width neural networks and gradient-boosted decision trees).

2 NOTATION AND PRELIMS

We adopt similar notation to what is used in Gao et al. (2022) but extend the framework to allow for the removal of a subset of features. Suppose we have data $(\mathbf{X}_i, Y_i), i = 1, \ldots, N$ for $(X, Y) \sim P_0$, where $\mathbf{X}_i \in \mathbb{R}^p$ is the i-th p-dimensional co-variate, and Y_i is the i-th observed predictor. Let $I \subseteq \{1, \ldots, p\}$, and $k = |I|$, let $X_{-I} \in \mathbb{R}^{p-k}$ denote the features with j-th co-variate, $j \in I$, dropped. For simplicity, we drop the j-th co-variate by replacing with its marginal mean $\mu_j = \mathbb{E}(X_j)$. Let $P_0, P_{0,-I}$ be the population distributions for X and X_{-I} and let $P_N, P_{N,-I}$ be the empir-

ical distributions of X and X_{-I}. We denote $\mathbb{E}_0, \mathbb{E}_{0,-I}$ as the expectation taken with respect to P_0 and $P_{0,-I}$ respectively. For notation simplicity, we use $\boldsymbol{X}$, $\boldsymbol{X}^{(I)}$ and $\boldsymbol{Y}$ to denote $(\mathbf{X}_1^T, \ldots, \mathbf{X}_N^T)^T$, $(\mathbf{X}_1^{(I)T}, \ldots, \mathbf{X}_N^{(I)T})^T$ and $(Y_1, \ldots, Y_N)^T$, respectively.

Let f_0 denote the true function mapping X to the expected value of Y conditional on X, and let $f_{0,-I}$ denote the function mapping X_{-I} to the expected value of Y conditional on X_{-I} :

$$\begin{aligned} f_0(X) &:= \mathbb{E}_0[Y \mid X]; \\ f_{0,-I}\left(X^{(I)}\right) &:= \mathbb{E}_{0,-I}\left[Y \mid X_{-I}\right]. \end{aligned} \tag{1}$$

We assume that the samples take the following form with respect to the X and $X^{(I)}$:

$$\begin{aligned} Y_i &= f_0(\mathbf{X}_i) + w_i \\ Y_i &= f_{0,-I}(\mathbf{X}_i^{(I)}) + w_i^{(I)} \end{aligned}$$

where w_i and $w_i^{(I)}$ are noise random variables independent of $\mathbf{X}_i$ and $\mathbf{X}_i^{(I)}$ respectively.

To measure the accuracy of an approximation $\hat{f}$ to its target function f^*, we use the $L^2(P_N)$-norm to measure the prediction error:

$$\|\hat{f} - f^*\|_N^2 = \frac{1}{N}\sum_{i=1}^{N}\left[\hat{f}(\mathbf{X}_i) - f^*\left(\mathbf{X}_i\right)\right]^2.$$

2.1 Variable Importance Estimation

Similar to (Williamson et al., 2023), the variable importance (VI) measure is defined in terms of a predictive skill metric. The primary predictive skill measure we consider is the negative mean squared error (MSE) as the loss we use is the least-squares loss:

$$V(f, P) = -\mathbb{E}_{(X,Y)\sim P}[Y - f(X)]^2. \tag{2}$$

And the VI measure is defined as

$$\mathrm{VI}_I := V\left(f_0, P_0\right) - V\left(f_{0,-I}, P_{0,-I}\right). \tag{3}$$

Typically, *dropout* and *retrain* are two methods to estimate VI. The dropout method estimates $f_{N,-I}$ by applying the full model f_N^c to the dropped data $\boldsymbol{X}^{(I)}$, calculating variable importance as

$$\widehat{\mathrm{VI}}_I^{(\mathrm{DR})} = V\left(f_N^c, P_N\right) - V\left(f_N^c, P_{N,-I}\right). \tag{4}$$

Retrain, on the other hand, estimates variable importance by training separate models for each subset I, with VI given by

$$\widehat{\mathrm{VI}}_I^{(\mathrm{RT})} = V\left(f_N, P_N\right) - V\left(f_{N,-I}, P_{N,-I}\right) \tag{5}$$

where $f_{N,-I}$ is derived by training separate models using $\boldsymbol{X}^{(I)}$ from scratch.

2.2 Reproducing Kernel Hilbert Spaces

Leveraging theory for the early stopping of kernel-based methods requires the use use of the machinery of reproducing kernel Hilbert spaces (Aronszajn, 1950; Wahba, 1990; Gu and Zhu, 2001). We consider a Hilbert space $\mathcal{H} \subset L^2(P_{0,-I})$, meaning a family of functions $g : \mathcal{X} \to \mathbb{R}$, with $\|g\|_{L^2(P_{0,-I})} < \infty$, and an associated inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ under which $\mathcal{H}$ is complete. The space $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) if there exists a symmetric kernel function $\mathbb{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$such that: (a) for each $x \in \mathcal{X}$, the function $\mathbb{K}(\cdot, x)$ belongs to the Hilbert space $\mathcal{H}$, and (b) we have the reproducing relation $f(x) = \langle f, \mathbb{K}(\cdot, x)\rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$. Any such kernel function must be positive semidefinite.

We define the associated *empirical kernel matrix* $K^{(I)} \in \mathbb{R}^{N\times N}$ for kernel $\mathbb{K}^{(I)}$ on X_{-I} with entries

$$K^{(I)}(i,j) = \frac{1}{N}\mathbb{K}^{(I)}(\mathbf{X}_i^{(I)}, \mathbf{X}_j^{(I)}). \tag{6}$$

The *empirical kernel matrix* plays a crucial role in our analysis, serving as the foundation for reparameterizing the gradient update equation and establishing the optimal stopping rule for the early stopping procedure.

3 ALGORITHM

Our primary focus is on estimating VI_I for general algorithms that employ gradient updates, such as neural networks and gradient-boosting decision trees.

Warm-start. The first step for estimating VI_I is to obtain an estimation of f_0. Consider a function class $\mathcal{H}$ that is a RKHS, we train the full model f_N^c from scratch using all the features $\boldsymbol{X}$ with respect to the least-square loss:

$$f_N^c \in \arg\min_{f\in\mathcal{H}} \frac{1}{2N}\|\boldsymbol{Y} - f(\boldsymbol{X})\|_2^2. \tag{7}$$

Given the subset of features to be dropped I, the next step is to estimate the *reduced* model $f_{0,-I}$. Instead of training a model from an arbitrary initialization, we start from the parameters of f_N^c and use gradient descent to optimize the least-square loss for the dropped data over the same function class $\mathcal{H}$. This loss given the dropped features becomes:

$$\mathcal{L}(f) := \frac{1}{2N}\|\boldsymbol{Y} - f(\boldsymbol{X}^{(I)})\|_2^2.$$

Assume a fixed step size ϵ for gradient descent algorithms, let the model parameters at iteration τ be denoted by θ_τ. Let $\nabla_\theta f(\theta_\tau) \in \mathbb{R}^{N\times|\theta|}$ represent the gradient of f_τ with respect to θ, evaluated at $\boldsymbol{X}^{(I)}$. We

initialize θ_0 with the model parameter of f_N^c, and the parameter update via gradient descent is given by:

$$\theta_{\tau+1} = \theta_\tau - \frac{\epsilon}{N}\nabla_\theta f(\theta_\tau)^T(f_\tau(\boldsymbol{X}^{(I)}) - \boldsymbol{Y}) \quad (8)$$

For gradient boosting algorithms, we consider the gradient descent in functional spaces, we update the function directly rather than the model parameters. We start from f_N^c and the update f_τ via:

$$f_{\tau+1} = f_\tau - \epsilon\nabla_{\mathcal{H}}\mathcal{L}(f_\tau) \quad (9)$$

where $\nabla_{\mathcal{H}}\mathcal{L}(f_\tau)$ is the functional gradient of functional $\mathcal{L}$ in the space $\mathcal{H}$, we can regard this as a weak learner since it is also a function from $\mathcal{H}$.

Early Stopping. The next important component of our algorithm is *early stopping*. It is well-known that running gradient descent for too many iterations can lead to overfitting, resulting in a poor approximation of the underlying reduced model, $f_{0,-I}$. Since we start from f_N^c, which is expected to be closer to $f_{0,-I}$ compared to a random initialization, we halt the gradient descent early at a specific iteration, denoted by $\widehat{T}$, based on certain criteria to prevent overfitting and reduce computational costs. Then combine these two steps, the estimated VI under this warm-start early stopping approach is:

$$\frac{1}{N}\left\{\|\boldsymbol{Y} - f_{\widehat{T}}\left(\boldsymbol{X}^{(I)}\right)\|_2^2 - \|\boldsymbol{Y} - f_N^c(\boldsymbol{X})\|_2^2\right\}.$$

4 THEORETICAL GUARANTEES

To accurately estimate VI_I, a reliable estimate for the reduced model $f_{0,-I}$ is crucial. We provide theoretical guarantees for our warm-start early stopping approach in terms of the estimation error between the early-stopped model $f_{\widehat{T}}$ and the target $f_{0,-I}$ under a fixed design, where $\boldsymbol{X}^{(I)}$ remain unchanged. We first rewrite the gradient update using the empirical kernel matrix and define the stopping rule. Then, we present a general convergence bound for kernel-based models trained via gradient descent and gradient boosting. We apply this approach to two examples: neural networks with large widths and gradient boosting decision trees using symmetric trees as weak learners with added noise. Detailed proofs are provided in the supplementary materials.

4.1 Kernel Gradient Update and Error Decomposition

Our theoretical analysis largely depends on rewriting the gradient update equations in (8) and (9) using the empirical kernel matrix. For both gradient descent and gradient boosting algorithms, let the model at each iteration τ, denoted by f_τ, induce a kernel $\mathbb{K}_\tau^{(I)}$. When updating the model using (8) and (9), suppose that for the model evaluated at $\boldsymbol{X}^{(I)}$, the updating rule can be expressed as follows:

$$f_{\tau+1}\left(\boldsymbol{X}^{(I)}\right) = (I - \epsilon K_\tau^{(I)})f_\tau\left(\boldsymbol{X}^{(I)}\right) + \epsilon K_\tau^{(I)}\boldsymbol{Y} \quad (10)$$

where $K_\tau^{(I)}$ is the associated empirical kernel matrix of kernel $\mathbb{K}_\tau^{(I)}$. We refer this equation as *iterative kernel update equation*. The recursion (10) is central to our analysis. In Sections 4.3 and 4.4 we derive the update equation for neural networks and gradient boosted decision trees.

Further suppose that for the kernel $\mathbb{K}_\tau^{(I)}$ converges to a stationary kernel $\mathbb{K}^{(I)}$ under some model specific conditions. Denote empirical kernel matrix of the stationary kernel $\mathbb{K}^{(I)}$ by $K^{(I)}$, by adding and subtracting $K^{(I)}$ in (10), we have

$$f_{\tau+1}\left(\boldsymbol{X}^{(I)}\right) = (I - \epsilon K^{(I)})f_\tau\left(\boldsymbol{X}^{(I)}\right) + \epsilon K^{(I)}\boldsymbol{Y} + \epsilon\delta_\tau \quad (11)$$

where $\delta_\tau = \left(K_\tau^{(I)} - K^{(I)}\right)(\boldsymbol{Y} - f_\tau(\boldsymbol{X}^{(I)}))$. This update equation is identical to Eq. (19) in Raskutti et al. (2014) with an additional δ_τ induced by the evolving kernel $\mathbb{K}_\tau^{(I)}$, which adds complexity to the overall analysis. Start from (11), and by some direct calculations, we are able to bound the $L^2(P_{N,-I})$ norm between f_τ and $f_{0,-I}$ as follows:

$$\|f_\tau - f_{0,-I}\|_N^2 \le B_\tau^2 + V_\tau + D_\tau^2 \quad (12)$$

where B_τ^2 is the bias term, V_τ is the variance term, and D_τ^2 is the additional difference term.

4.2 Stopping Rule and General Bound

The stopping threshold is closely related to a model complexity measure, known as the local empirical Rademacher complexity (Raskutti et al., 2014). In this paper, it takes the form

$$\widehat{\mathcal{R}}_K(r) := \left[\frac{1}{N}\sum_{i=1}^N \min\left\{\widehat{\lambda}_i, r^2\right\}\right]^{1/2}$$

where $\widehat{\lambda}_i$ is the eigenvalue of $K^{(I)}$.

Let $\mathcal{H}$ denote the RKHS induced by the stationary kernel $\mathbb{K}^{(I)}$, and denote the Hilbert norm in $\mathcal{H}$ by $\|\cdot\|_{\mathcal{H}}$. We make the following assumptions in this paper.

Assumption 4.1. *f_N^c and $f_{0,-I}$ belongs to $\mathcal{H}$, i.e., $f_N^c, f_{0,-I} \in span\{\mathbb{K}^{(I)}(\cdot, X_{-I})\}$.*

This assumption is made for purely theoretical convenience, avoiding the need to add additional misspecification error terms.

Assumption 4.2. *The data* $\{(\mathbf{X}_i, Y_i)\}_{i=1}^N$ *are contained in closed and bounded set in* $\mathbb{R}^{p+1}$.

Assumption 4.3. $w_i^{(I)}$ *are independent zero-mean random variables satisfying the following condition:*

$$\mathbb{E}\left[e^{tw_i^{(I)}}\right] \leq e^{t^2\sigma^2/2}, \text{ for all } t \in \mathbb{R}. \tag{13}$$

Assumption 4.4. *The dropout error does not explode satisfying* $\|\boldsymbol{Y} - f_N^c(\boldsymbol{X}^{(I)})\|_2 = O(\sqrt{N})$.

Assumption 4.5. *The trace of* $K^{(I)}$ *satisfies* $tr(K^{(I)}) = O(1)$.

When $N \to \infty$, $\widehat{\lambda}_i$ would be close to the population level eigenvalue λ_i. For certain kernel classes, such as kernels with polynomial eigendecay and finite rank kernels, this condition is satisfied.

For a given noise variance $\sigma > 0$, we define the critical empirical radius $\widehat{r}_N > 0$, to be the smallest positive the solution to the inequality

$$\widehat{\mathcal{R}}_K(r) \leq \frac{r^2 C_{\mathcal{H}}^2}{2e\sigma} \tag{14}$$

where $C_{\mathcal{H}}$ is $\|f_N^c - f_{0,-I}\|_{\mathcal{H}}$, the quantity to measure the difference between our start and the target.

Define running sum $\eta_\tau = \tau\epsilon$ to be the sum of step size over τ iterations. The stopping threshold $\widehat{T}_{max}$ is defined as

$$\widehat{T}_{\max} := \arg\min\left\{\tau \in \mathbb{N} \,\middle|\, \widehat{\mathcal{R}}_K\left(1/\sqrt{\eta_\tau}\right) > \frac{C_{\mathcal{H}}^2}{2e\sigma\eta_\tau}\right\} - 1$$

the integer $\widehat{T}_{\max}$ belongs to the interval $[0, \infty)$ and is unique in our setup. And $\widehat{T}_{\max}$ optimized the sum of B_τ^2 and V_τ in (12). According to Raskutti et al. (2014), for iteration $\tau \leq \widehat{T}_{\max}$, with certain probability, we can bound B_τ^2 and V_τ as

$$B_\tau^2 + V_\tau \leq \frac{C}{\eta_\tau}.$$

The difference term D_τ^2 is non-decreasing with τ. Suppose D_τ^2 is upper bounded by a function $g(\tau)$, which is also non-decreasing with τ. Finally, our proposed optimal stopping time $\widehat{T}_{\text{op}}$ is given by:

$$\widehat{T}_{\text{op}} := \arg\min\left\{\tau \leq \widehat{T}_{\max} \,\middle|\, \frac{C}{\eta_\tau} + g(\tau)\right\}.$$

The reason we should stop the gradient updates early is reflected in two aspects of the theory. First, the quantity $C_{\mathcal{H}}$ limits $\widehat{T}_{\max}$ from being too large, as we expect f_N^c to be reasonably close to $f_{0,-I}$. Second, the difference term D_τ^2 in the bound (12) favors a smaller number of iterations, since running for too many iterations would result in larger difference errors in the bound.

Theorem 4.1 (General convergence bound under fixed design)**.** *Under assumptions 4.1-4.5, consider a model that use gradient descent or gradient boosting with step size* $\epsilon \leq \min\{1, 1/\widehat{\lambda}_1\}$*, and under some additional model-specific conditions, start from the full model* f_N^c*, and stop the update early at iteration* $\widehat{T}_{op}$*, with probability at least* $1 - c_1\exp(-c_2 N\widehat{r}_N)$*, the following bound holds*

$$\|f_{\widehat{T}_{op}} - f_{0,-I}\|_N^2 \leq \mathcal{O}\left(N^{-\frac{1}{2}}\right).$$

where c_1 *and* c_2 *are some universal positive constants.*

4.3 Neural Networks

Neural networks, as powerful black-box models, are widely used for accurate predictions across various scenarios. They typically employ gradient-based methods to update parameters, making them ideal candidates for our general algorithm. With sufficiently large widths and the theoretical insights from the *neural tangent kernel*, we can analyze neural networks in the context of kernel-based methods and derive corresponding theoretical guarantees.

First introduced in Jacot et al. (2018), the Neural Tangent Kernel (NTK) provides a theoretical tool to study the neural network in the RKHS regime. Denote a neural network by $f(\theta, x)$, the corresponding *Neural Tangent Kernel* is defined as

$$\langle \nabla_\theta f(\theta, x), \nabla_\theta f(\theta, x')\rangle. \tag{15}$$

Consider a fully connected neural network with width m using gradient descent to update parameters. At each iteration, the network f_τ induces a corresponding NTK, denoted by $\mathbb{K}_\tau$. It is well-known that, under certain random initializations and gradient flow, for a network of infinite width, $\mathbb{K}_\tau$ remains constant during training, and moreover $\mathbb{K}_0$ converges to certain deterministic kernel $\mathbb{K}$ when widths goes to infinity (Jacot et al., 2018). And when we use least-squares loss, infinite-width networks behave as linearized networks (Lee et al., 2019).

In order to fit neural network into our general framework. We first need to verify assumption 4.5. As shown by Jacot et al. (2018), the NTK can be computed recursively using specific formulas. Under the assumption that the input data is bounded, we can ensure that $tr(K^{(I)})$ remains bounded as well.

Next, we show that the update can be expressed similarly to equation (10). Apply first-order Taylor's expansion and plug in the gradient update equation (8),

we have

$$f_{\tau+1}(\boldsymbol{X}^{(I)}) = f_\tau(\boldsymbol{X}^{(I)}) - \frac{\epsilon}{N}\nabla_\theta f(\theta_\tau)\nabla_\theta f(\theta_\tau)^T g_\tau(\boldsymbol{X}^{(I)}) \tag{16}$$

where $g_\tau(\boldsymbol{X}^{(I)}) := f(\boldsymbol{X}^{(I)}) - \boldsymbol{Y}$. By the definition of the NTK, we derive the same update equation as in (10). However, since neural networks are not linear models, there will be an additional linearization error. By extending Theorem 2.1 from Lee et al. (2019) to the warm-start initialization setting, the L_2 norm of this error is bounded by $\mathcal{O}(m^{-1/2})$. Moreover, the NTK stability result remains valid in our setup, allowing us to bound the difference between $K_\tau^{(I)}$ and $K^{(I)}$. As a result, we can bound the L_2 norm by $\mathcal{O}(m^{-1/2})$ with high probability when subtracting $K^{(I)}$, as shown in (11).

Corollary 4.1.1 (Error bound for neural network)**.** *For any $\gamma > 0$, there exists $M \in \mathbb{N}$, a full connected neural network with width $m \geq M$, the following holds with probability at least $1 - \gamma - c_1 \exp\left(-c_2 N \widehat{r}_N^2\right)$ over warm-start initialization when applying gradient descent with learning rate $\epsilon = \mathcal{O}\left(\frac{1}{m}\right)$*

$$\|f_{\widehat{T}_{op}} - f_{0,-I}\|_N^2 \leq \mathcal{O}\left(N^{-\frac{1}{2}}\right). \tag{17}$$

Remark 4.1. *The actual convergence rates hinge on the empirical kernel matrix's spectrum. This is a complex issue shaped by the network's structure, activation function, and depth. According to Bietti et al. (2019) and Bietti and Bach (2021), for bias-free ReLU networks trained on data normalized to the hypersphere (i.e., $x = \frac{x}{\|x\|_2} \in \mathbb{S}^{p-1}$), the eigenvalues λ_i of the empirical kernel matrix decay at a rate of $O(i^{-p})$. Consequently, we can achieve a convergence rate of $\mathcal{O}\left(N^{-\frac{p}{p+1}}\right)$. Although this may seem counterintuitive, it reflects the NTK's RKHS, which resembles the Laplace kernel (Chen and Xu, 2021): stronger smoothness in higher-dimensional spheres causes rapid eigenvalue decay, thereby tightening the RKHS constraints and amplifying smoothness relative to the growth of function classes.*

Remark 4.2. *When applied to neural networks, our method is theoretically equivalent to Gao et al. (2022). However, Gao et al. (2022) relies on k-fold cross-validation to select the ridge penalty λ, which becomes impractical for large sample sizes. In contrast, our implementation (Algorithm 1) is significantly more efficient in such settings. See Section D.5 in the supplementary materials for a detailed comparison.*

4.4 Gradient Boosting Decision Trees

A classic gradient boosting algorithm (Friedman, 2001) iteratively minimizes the expected loss $\mathcal{L}(f) = \mathbb{E}[L(f(x), y)]$ using weak learners. In each iteration τ, the model is updated as $f_\tau(x) = f_{\tau-1}(x) + \epsilon w_\tau(x)$, where the weak learner w_τ is selected from a function class $\mathcal{W}$ to approximate the negative gradient of the loss function. When $\mathcal{W}$ consists of decision trees, the algorithm is known as Gradient Boosting Decision Trees (GBDT). A decision tree recursively partitions the feature space into disjoint regions called leaves, with each leaf R_j assigned a value representing the estimated response y for that region (Ustimenko et al., 2023).

Given the complexity of standard GBDT, we explore a simpler GBDT algorithm that uses symmetric trees as weak learners and incorporates additional noise introduced in Ustimenko et al. (2023). Each weak learner, denoted by ν, is a symmetric, oblivious tree, i.e., all nodes at a given level share the same splitting criterion (feature and threshold). And the noise level is controlled by a parameter β, which is referred to as random strength. To limit the number of candidate splits, each feature is quantized into $n + 1$ bins. The maximum tree depth is limited by d.

Let $\mathcal{V}$ represent all tree structures. The RKHS structure for this specific GBDT is based on a kernel defined for each tree structure, i.e. weak learner, denoted by $k_\nu(\cdot, \cdot)$. At each iteration, because of the added noise, the model f_τ induces a distribution $p(\nu|f_\tau, \beta)$ over all tree structures. Then $\mathbb{K}_\tau^{(I)}$ is given by

$$\mathbb{K}_\tau(x, x') = \sum_{\nu \in \mathcal{V}} k_\nu(x, x')\, p(\nu \mid f_\tau, \beta). \tag{18}$$

The kernel during training $\mathbb{K}_\tau^{(I)}$ converges to a certain stationary kernel $\mathbb{K}^{(I)}$ when $\beta \to \infty$ or the model approaches the empirical minimizer when $\tau \to \infty$. The stationary kernel $\mathbb{K}$ is given by replace $p(\nu \mid f_\tau, \beta)$ with a uniform distribution over $\mathcal{V}$, denoted by $\pi(\nu)$. Since $\pi(\nu)$ does not involve any f_τ, $\mathbb{K}^{(I)}$ is independent with $\mathbb{K}_\tau^{(I)}$. It turns out that the kernel class for the GBDT belongs to the finite rank kernels, and we can bound $tr(K^{(I)})$ by 2^d, which satisfies the assumption 4.5.

In terms of the iterative kernel update equation, Lemma 3.7 in Ustimenko et al. (2023) allows us to express the model updates similarly to equation (10), but incorporating the distribution $p(\nu \mid f_\tau, \beta)$. To align with our general framework, we take the expectation with respect to the algorithm's randomness on both sides of the update equations, providing results for GBDT in terms of the model expectation $\mathbb{E}_u f_\tau$, where u represents the algorithm's randomness.

Corollary 4.1.2. *For the GBDT algorithm mentioned above with depth d, and random strength $\beta \geq N^{5/4}$, the following holds with probability at least $1 - c_3 \exp\left(-c_4 N \widehat{r}_N^2\right)$ over warm-start initialization when*

applying gradient boosting with learning rate $\epsilon = O(\frac{1}{N})$

$$\left\|\mathbb{E}_u f_{\widehat{T}_{op}} - f_{0,-I}\right\|_N^2 \leq \mathcal{O}\left(2^{\frac{d}{2}} N^{-\frac{1}{2}}\right). \tag{19}$$

Remark 4.3. *If we do not account for the randomness of the GBDT algorithm,* $\mathbb{E}\left\|f_{\widehat{T}_{op}} - f_{0,-I}\right\|_N^2$ *will include a variance component that does not converge to zero, even as* $N \to \infty$*, due to the algorithm's inherent randomness. However, this inherent variance is quite small as we display in the experimental results.*

4.5 Proof Overview

To bound the prediction error, it suffices to bound the terms on the RHS of (12). We start by considering a sub-optimal stopping time $\widehat{T}_{\max}$. At $\widehat{T}_{\max}$, we are able to bound the sum of bias and variance terms as

$$B^2_{\widehat{T}_{\max}} + V_{\widehat{T}_{\max}} \leq \frac{C}{\epsilon \widehat{T}_{\max}} \leq \mathcal{O}(\widehat{r}_N^2). \tag{20}$$

Then we bound $\widehat{r}_N^2$ as follows. In fact, $\widehat{r}_N^2$ is the solution when both sides of (14) are equal, by the assumption $tr(K^{(I)}) = O(1)$ and definition of Rademacher complexity, we have

$$\frac{\widehat{r}_N^2 C_{\mathcal{H}}^2}{2e\sigma} \leq \mathcal{O}(N^{-\frac{1}{2}}) \tag{21}$$

which gives $\widehat{r}_N^2 \leq \mathcal{O}(N^{-\frac{1}{2}})$.

We now need to bound the difference term $D^2_{\widehat{T}_{\max}}$, which arises from the discrepancy between $\mathbb{K}_\tau^{(I)}$ and $\mathbb{K}^{(I)}$. Model-specific conditions are required to control this error and ensure it is bounded by $\mathcal{O}(N^{-\frac{1}{2}})$. For neural networks, this is controlled by the width m, and for GBDT, by the random strength β. By constraining these hyperparameters within a certain range, we can bound the difference term. This leads to an error rate $f_{\widehat{T}_{\max}}$ of $\mathcal{O}(N^{-\frac{1}{2}})$.

5 EXPERIMENTS

As shown in the theoretical results, the exact stopping time $\widehat{T}_{\text{op}}$ is impractical to compute due to the lack of knowledge of $f_{0,-I}$ and the noise level σ. In practice, we recommend using the hold-out method, where the training data is split into a validation set, and updates are halted when the validation loss shows no improvement over several iterations. Paritcularly, we use data splitting for training and estimating VI, see Algorithm 1 for details. Further discussion is provided in the supplementary material.

In our experiments, we majorly compare our method with two baseline approaches *dropout* and *retrain* discussed above (Section 2.1). We empirically verify the theoretical bounds and evaluate our algorithm's performance on both simulated data and a real-world example, demonstrating its practical effectiveness in estimating variable importance. Additional experiments, including comparisons with Gao et al. (2022) in a high-dimensional regression setting (Section D.5) and the full details of all experiments, are provided in the supplementary materials due to space constraints.

For these experiments, unless otherwise specified, we use the following model structures: for neural network, we train a three-layer fully connected neural network with ReLU activation, where the hidden layer has a width of 2048; for GBDT, the random strength β we use is 10,000 and tree depth is 2. And instead of computing $\mathbb{E}_u f_{\widehat{T}}$, we simply use $f_{\widehat{T}}$ in our experiments. We use 3/4 of all the data to train the model, while the remaining samples are employed to estimate the VI. Specifically, we set $\alpha = 0.75$ in Algorithm 1. The implementation of the proposed algorithm and all the experiments are available at `https://github.com/ZexuanSun/Early-stopping-VI`.

Algorithm 1 Early stopping training for VI_I

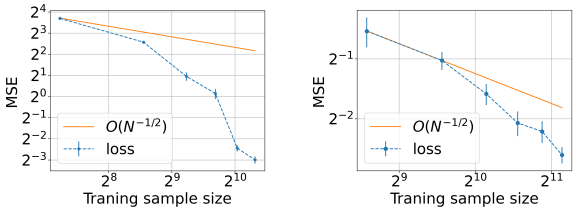
1: Input Data $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^N$, training size $N_1 = \alpha N$; $N_2 = (1-\alpha)N$; Kernel based model: $f_\theta(\cdot)$; Drop features set $I \subseteq \{1, \ldots, p\}$; Patience P;
2: Train full model $f^c_{N_1}$ using training data $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^{N_1}$;
3: Replace feature $j \in I$ with its empirical mean to get $\boldsymbol{X}_i^{(I)}$;
4: Split $\left\{(\boldsymbol{X}_i^{(I)}, Y_i)\right\}_{i=1}^{N_1}$ into a training set $\mathcal{D}_1$ of sample size αN_1 and a validation set $\mathcal{D}_2$ of sample size $(1-\alpha)N_1$;
5: Initialize model with $f^c_{N_1}$, for each epoch τ, train on $\mathcal{D}_1$, evaluate on $\mathcal{D}_2$;
6: If the loss evaluated on $\mathcal{D}_2$ at epoch $\widehat{T}$ has no improvement after P epochs, stop and return $f_{\widehat{T}}$ as the estimator for $f_{0,-I}$;
7: Use remaining N_2 instances to construct $\widehat{VI}_I$

$$\widehat{\text{VI}}_I = \frac{1}{N_2}\sum_{i=1}^{N_2}\left[Y_i - f_{\widehat{T}}(\mathbf{X}_i^{(I)})\right]^2 - \left[Y_i - f^c_{N_1}(\mathbf{X}_i)\right]^2.$$

5.1 Verifying Theoretical Bounds

To verify the theoretical bounds, we consider independent features, dropping only the first feature. Construct f_0 as $f(\boldsymbol{X}^{(1)}) + \beta_1 X_1$. For neural networks, f is a two-layer fully connected neural network with ReLU activation, where the hidden layer has a width of 2048. For GBDT, we use a model with a random strength β of 10,000 and a depth of 2. This design

is to satisfy assumption 4.1 for both models. We use shallow neural network and GDBT here to reduce the cost of compute the empirical kernel matrix. Note that our theory does not assume feature independence, we use independent data for implementation convenience. Since computing the optimal stopping time $\widehat{T}_{\text{op}}$ is difficult, we use $\widehat{T}_{\max}$, which also achieves convergence rate $\mathcal{O}(N^{-1/2})$. We repeat the experiment 10 times, and compared the $L^2(P_{N,-I})$ norm with the upper bound $\mathcal{O}(N^{-1/2})$. Results are shown in Figure 1.



(a) Neural Networks (b) GBDT

Figure 1: Log–log plot comparing empirical bounds to the theoretical $\mathcal{O}(N^{-\frac{1}{2}})$ rate. The dotted blue line shows the mean prediction error, and the error bars represent one standard deviation across 10 repetitions.

5.2 Linear Models with Correlation

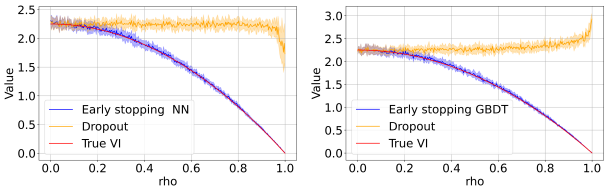
According to Example 3.1 in Gao et al. (2022), for a linear model $Y = \beta_1 X_1 + \beta_2 X_2 + \epsilon$, where $X_i \sim \mathcal{N}\left(0, \sigma^2\right), i = 1, 2, \text{Cov}\left(X_1, X_2\right) = \rho$, and ϵ is a $\mathcal{N}\left(0, \sigma_\epsilon^2\right)$ noise that is independent of the features. Then VI for X_1 can be computed analytically as

$$\text{VI}_1 = \beta_1^2 \cdot \text{Var}\left(X_1 \mid X_2\right) = \beta_1^2 \left(1 - \rho^2\right) \sigma^2. \tag{22}$$

We consider a higher-dimensional example with $\beta = (1.5, 1.2, 1, 0, 0, 0)^T$ and generate $Y_i = \boldsymbol{X}_i^T \beta + \epsilon$. We drop the first feature and estimate its VI using our method and *dropout* method, comparing against the known ground truth. A dataset of size 5000 is generated and repeated 10 times for each ρ. Figure 2 shows that for neural networks and GBDT, dropout's VI estimation becomes increasingly unreliable as ρ grows, overestimating VI, while our early stopping approach remains accurate.

5.3 High-dimensional Regression

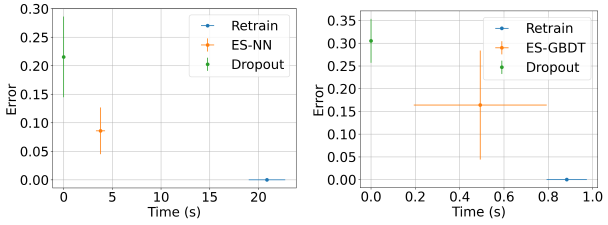
The computational burden of *retrain* is most pronounced in high-dimensional settings, as it needs fitting at least p models to estimate VI (Gao et al., 2022). For this simulation, we generate various high-dimensional datasets for neural networks and GBDT to evaluate their performance in high-dimensional settings. The different datasets are designed to better align the experiments with assumption 4.1.



(a) Neural Networks (b) GBDT

Figure 2: VI estimation comparison for correlated linear model. Solid lines and shaded areas represent the average and std. over 10 repetitions, respectively.

We generate $X \sim N\left(0, \Sigma_{100\times 100}\right)$, where variables are independent except $\text{Corr}\left(X_1, X_2\right) = 0.5$. For neural networks, we borrow the settings from Gao et al. (2022). Let $\boldsymbol{\beta} = (5, 4, 3, 2, 1, 0, \ldots, 0)^\top \in \mathbb{R}^{100}$, we construct a weight matrix $W \in \mathbb{R}^{m\times p}$ such that the $W_{:,j} \sim \mathcal{N}\left(\beta_j, \sigma^2\right)$. Let $V \sim \mathcal{N}(0, 1)$, we generate the response $Y_i = V\sigma\left(W\mathbf{X}_i\right) + \epsilon_i$ where σ is the ReLU function. For GDBT, we generate the response $Y_i = \mathbf{X}_i\boldsymbol{\beta} + \epsilon_i$. As the true VI is unknown, we take retrain estimation results as ground truth and use the normalized estimation error relative to retrain as metric, i.e., we compute the error as $(\widehat{\text{VI}} - \widehat{\text{VI}}^{(\text{RT})})/\widehat{\text{VI}}^{(\text{RT})}$. We assessed VI for X_1 on 10 simulated datasets with sample size 5000. As shown in Figure 3, our method's error is roughly half that of *dropout* while using significantly less time than *retrain*, which is a clear advantage in high-dimensional contexts.



(a) Neural Networks (b) GBDT

Figure 3: Distribution of computation time vs. normalized estimation error (relative to retraining) for the VI of X_1. Error bars show the standard deviation over 10 runs and ES is the shorthand for early stopping.

5.4 Shapley Values

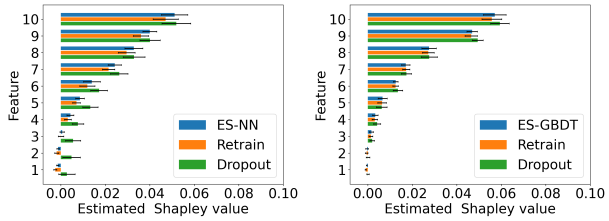
When variables are correlated, the defined VI in (3) approaches zero (Gao et al., 2022). Recent studies suggest using Shapley values for variable importance due to their effective management of correlated variables, assigning similar weights to correlated significant variables (Owen and Prieur, 2017; Williamson and Feng, 2020). These studies also highlight the high computational cost of Shapley values, requiring a model fit

for each subset of variables. However, our algorithm may speed up computing Shapley values by efficiently calculating the quantity defined in (3). The Shapley value is defined via a value function *val* of features in I. The Shapley value of a feature value is its contribution to the payout, weighted and summed over all possible feature value combinations:

$$\sum_{I\subseteq\{1,\ldots,p\}\setminus\{j\}} w_j \left(\text{val}(I\cup\{j\}) - \text{val}(I)\right) \tag{23}$$

where $w_j = \frac{|I|!(p-|I|-1)!}{p!}$ (Molnar, 2022). In our paper, we consider the *val* function to be the negative MSE.

Consider a Logistic Model, we generate $X \sim \mathcal{N}(0, \Sigma_{10\times 10})$, where the variables are independent except $\text{Corr}(X_1, X_2) = 0.5$. The responses are generated from a logistic model: $\log\frac{\mathbb{P}(Y=1)}{1-\mathbb{P}(Y=1)} = X\boldsymbol{\beta}$, where $\boldsymbol{\beta} = 10\times(0,1,2,\ldots,9)^\top \in \mathbb{R}^{10}$. The dataset we generate is of size $N = 5000$. We use the subset sampling scheme proposed by (Williamson and Feng, 2020) when calculating Shapley values. We can see that our proposed approach is closer to the *retrain* compared with *dropout* method, as shown in Figure 4. The runtime for Shapley values estimates is 2-3 times faster using our early stopping approach compared to retraining.

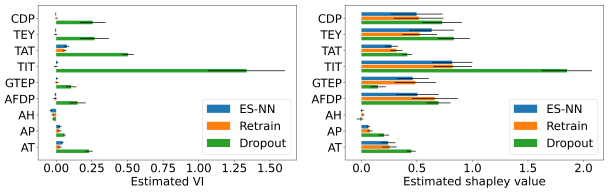


(a) Neural Networks (b) GBDT

Figure 4: Shapley value estimation for logisitic model using our method compared with *retrain* and *dropout* over 10 repetitions.

5.5 Predicting Flue Gas Emissions

Assessing variable importance in predicting CO and NOx emissions from gas turbines is crucial for optimizing efficiency, reducing environmental impact, and ensuring compliance. It helps operators adjust turbine settings and prioritize maintenance on key variables. To demonstrate the value of our framework, we used data from a gas turbine in northwestern Turkey, collected in 2015, to study NOx emissions (Gas Turbine Dataset, 2019). The dataset includes 7,384 instances of 9 sensor measures, such as turbine inlet temperature and compressor discharge pressure, aggregated hourly. Initial variable importance using negative MSE estimates were low, as shown in Figure 5(a) due to high feature correlation, prompting us to apply Shapley value estimation using our proposed methods. According to the Shapley value estimates using neural networks in Figure 5(b), Ambient Humidity (AH) and Ambient Pressure (AP) are not significant features, while Turbine Inlet Temperature (TIT) is the most important. Other features have similar importance and contribute to predicting NOx emissions. These findings align with previous studies (Kochueva and Nikolskii, 2021; Hoque et al., 2024), which highlight the significance of temperature-related variables, such as TIT, in predicting NOx emissions, with ambient conditions like humidity and pressure having a lesser impact. Once again we observe how our early stopping approach closely mimics the estimates for re-training while dropout tends to often over-estimate both VI and Shapley values. The feature correlation heat map and estimation results using GBDT are available in supplementary materials (Section D.6).



(a) VI estimation (b) Shapley value

Figure 5: VI and Shapley value estimation using neural networks, compared with *retrain* and *dropout* on the flue gas emissions data. Mean estimates and standard deviations over 10 repetitions are shown.

6 SUMMARY AND OUTLOOK

In this paper, we introduce a general algorithm designed for any gradient-based iterative process to efficiently estimate variable importance (VI). Our method based on warm-start using a dropout estimator with early stopping accurately estimates VI and matches the precision of more computationally intensive retraining methods. We demonstrate the effectiveness of the warm-start early-stopping approach and provide practical guidelines for its use.

The theory presented marks a significant step toward improving computational efficiency in machine learning interpretability. We believe this framework can be extended in several ways: (1) in the finite sample case, deriving the exact or approximate distribution of the VI estimator could enable hypothesis testing for reliability; and (2) applying the lazy training concept to bagging algorithms like random forests is another promising direction.

Acknowledgments

Support for this research was provided by American Family Insurance through a research partnership with the University of Wisconsin–Madison's American Family Insurance Data Science Institute.

References

Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR.

Anderssen, R. S. and Prenter, P. M. (1981). A formal comparison of methods proposed for the numerical solution of first kind integral equations. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 22(4):488–500.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. (2019). On exact computation with an infinitely wide neural net. In *33rd Conference on Neural Information Processing Systems*, Vancouver, Canada.

Bietti, Alberto, Mairal, and Julien (2019). On the inductive bias of neural tangent kernels. In *Advances in Neural Information Processing Systems*, volume 32.

Bietti, A. and Bach, F. (2021). Deep equals shallow for relu networks in kernel regimes. In *9th Conference on International Conference on Learning Representations*, Virtual.

Bühlmann, P. and Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, 22(4):477 – 505.

Bühlmann, P. and Yu, B. (2003). Boosting with the l2 loss. *Journal of the American Statistical Association*, 98(462):324–339.

Chang, C.-H., Rampasek, L., and Goldenberg, A. (2018). Dropout feature ranking for deep learning models.

Chen, L. and Xu, S. (2021). Deep neural tangent kernel and laplace kernel have the same rkhs. In *9th Conference on International Conference on Learning Representations*, Virtual.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. (2019a). Gradient descent finds global minima of deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2019b). Gradient descent provably optimizes over-parameterized neural networks. In *In Proceedings of the International Conference on Learning Representations (ICLR)*.

Falcon, W. and the PyTorch Lightning team (2019). Pytorch lightning.

Feng, J., Williamson, B., Simon, N., and Carone, M. (2018). Nonparametric variable importance using an augmented neural network with multi-task learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1496–1505. PMLR.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232.

Funahashi and Ken-Ichi (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183 – 192.

Gao, Y., Stevens, A., Raskutti, G., and Willett, R. (2022). Lazy Estimation of VI for Large NNs. In *Proceedings of the 39th International Conference on Machine Learning*, Baltimore, Maryland, USA.

Gas Turbine Dataset (2019). Gas Turbine CO and NOx Emission Data Set. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5WC95.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5).

Hoque, K. E., Hossain, T., Haque, A. M., Miah, M. A. K., and Haque, M. A. (2024). NOx Emission Predictions in Gas Turbines Through Integrated Data-Driven Machine Learning Approaches. *Journal of Energy Resources Technology*, 146(7):071201.

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

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *32nd Conference on Neural Information Processing Systems*, Montréal , Canada.

Jiang, W. (2004). Process consistency for adaboost. *The Annals of Statistics*, 32(1):13–29.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation. In *6th Conference on International Conference on Learning Representations*, Vancouver, Canada.

Kochueva, O. and Nikolskii, K. (2021). Data analysis and symbolic regression models for predicting co and nox emissions from gas turbines. *Computation*, 9(12).

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. In *33rd Conference on Neural Information Processing Systems*, Vancouver, Canada.

Molnar, C. (2022). *Interpretable Machine Learning*. 2 edition.

Morgan, N. and Bourlard, H. (1989). Generalization and parameter estimation in feedforward nets: Some experiments. In *Proceedings of Neural Information Processing Systems*.

Nathans, L. L., Oswald, F. L., and Nimon, K. (2012). Interpreting multiple linear regression: A guidebook of variable importance. *Practical Assessment, Research & Evaluation*, 17(9).

Owen, A. B. and Prieur, C. (2017). On shapley value for measuring importance of dependent inputs. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):986–1002.

Park, D. S., Sohl-Dickstein, J., Le, Q. V., and Smith., S. L. (2019). The effect of network width on stochastic gradient descent and generalization: an empirical study. In *International Conference on Machine Learning*, CA, USA.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: gradient boosting with categorical features support.

Raskutti, G., Wainwright, M. J., and Yu, B. (2014). Early stopping and non-parametric regression: An optimal data-dependent stopping rule. *Journal of Machine Learning Research*, 15:335–366.

Rinaldo, A., Wasserman, L., and G'Sell, M. (2019). Bootstrapping and sample splitting for high-dimensional, assumption-lean inference. *The Annals of Statistics*, 47(6):3438 – 3469.

Rudin, C. and Radin, J. (2019). Why Are We Using Black Box Models in AI When We Don't Need To? A Lesson From an Explainable AI Competition. *Harvard Data Science Review*, 1(2).

Shapley, L. S. (1953). *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, Princeton.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*.

Smith, G., Mansilla, R., and Goulding, J. (2020). Model class reliance for random forests. In *Advances in Neural Information Processing Systems*, volume 33.

Strand, O. N. (1974). Theory and methods related to the singular-function expansion and landweber's iteration for integral equations of the first kind. *SIAM Journal on Numerical Analysis*, 11(4):798–825.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*.

Ustimenko, A., Beliakov, A., and Prokhorenkova, L. (2023). Gradient boosting performs gaussian process inference. In *11th Conference on International Conference on Learning Representations*, Kigali, Rwanda.

Vito, E. D., Pereverzyev, S., and Rosasco, L. (2010). Adaptive kernel methods using the balancing principle. *Foundations of Computational Mathematics*, 10:455–479.

Wei, Y., Yang, F., and Wainwright1, M. J. (2017). Early stopping for kernel boosting algorithms: A general analysis with localized complexities. In *31st Conference on Neural Information Processing Systems*, Long Beach, CA, USA.

Williamson, B. D. and Feng, J. (2020). Efficient nonparametric statistical inference on population feature importance using shapley values. In *Proceedings of the 37th International Conference on Machine Learning*, Online.

Williamson, B. D., Gilbert, P. B., Carone, M., and Simon, N. (2021). Nonparametric variable importance assessment using machine learning techniques. *Biometrics*, 77(1):9–22.

Williamson, B. D., Gilbert, P. B., Simon, N. R., and Carone, M. (2023). A general framework for inference on algorithm-agnostic variable importance. *Journal of the American Statistical Association*, 118(543):1645–1658.

Yang, G. (2020). Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation.

Yao, Y., Rosasco, L., and Caponnetto, A. (2007). On early stopping in gradient descent learning. *Constructive Approximation*, 26(2).

Zhang, T. and Yu, B. (2005). Boosting with early stopping: Convergence and consistency. *The Annals of Statistics*, 33(4):1538 – 1579.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Material: Reliable and Scalable Variable Importance Estimation via Warm-start and Early Stopping

A GENERAL ALGORITHM

We start by proving the error decomposition lemma and then prove the general convergence results.

A.1 Error Decomposition

The error decomposition builds on Lemma 6 from Raskutti et al. (2014). Since Lemma 6 from Raskutti et al. (2014) strongly depends on zero initialization, we start by analyzing the dropout error, defined as $\boldsymbol{e}^{(I)} := \boldsymbol{Y} - f_N^c(\boldsymbol{X}^{(I)})$. We then subtract f_N^c from f_τ to apply similar proof techniques. For models using gradient boosting algorithms, this process is straightforward. However, for models based on gradient descent, such as neural networks, we need to linearize the model, which introduces an additional error (discussed further in Section B). And this is why the difference term δ_τ for neural networks is more complex. Importantly, at each step, we use the kernel induced by f_τ rather than $f_\tau - f_N^c$. This implies that at step 0, we use the kernel induced by f_N^c, not the zero function. When analyzing $\boldsymbol{e}^{(I)}$, the underlying true function becomes $f_{0,-I} - f_N^c$. Thus, the training process can be interpreted approximately as starting from 0 and using functions from span $\left\{\mathbb{K}^{(I)}(\cdot, X_{-I})\right\}$ to approximate $f_{0,-I} - f_N^c$.

Let $r = \text{rank}(K^{(I)})$, decompose $K^{(I)}$ using eigendecomposition $K^{(I)} = U\Lambda U^T$, where $U \in \mathbb{R}^{N\times N}$ is an orthonormal matrix and

$$\Lambda := \text{diag}\left(\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_r, 0, 0, \ldots, 0\right) \tag{1}$$

is the diagonal matrix of eigenvalues. We then define a sequence of diagonal shrinkage matrices as follows:

$$S^\tau := (I - \epsilon\Lambda)^\tau \tag{2}$$

Then we have the following lemma shows that the prediction error can be bounded in terms of the eigendecomposition and these shrinkage matrices:

Lemma A.1. *At each iteration $\tau = 0, 1, 2, \ldots$,*

$$\begin{aligned}\|f_\tau - f_{0,-I}\|_N^2 \leq & \underbrace{2\sum_{j=1}^{r}[S^\tau]_{jj}^2\left(\zeta_{jj}^*\right)^2 + 2\sum_{j=r+1}^{n}\left(\zeta_{jj}^*\right)^2}_{\textit{Bias } B_\tau^2} + \underbrace{\frac{4}{N}\sum_{j=1}^{r}\left(1 - S_{jj}^\tau\right)^2\left[U^T w^{(I)}\right]_j^2}_{\textit{Variance } V_\tau} \\ & + \underbrace{\frac{4\epsilon^2}{N}\|\sum_{i=0}^{\tau-1} S^{\tau-1-i}\tilde{\delta}_i\|_2^2}_{\textit{Difference } D_\tau^2}\end{aligned} \tag{3}$$

where $\zeta_{jj}^* = \frac{1}{\sqrt{N}}\left[U^T\left(f_{0,-I}\left(\boldsymbol{X}^{(I)}\right) - f_N^c\left(\boldsymbol{X}^{(I)}\right)\right)\right]_j$, and $\tilde{\delta}_\tau = U^T\delta_\tau$.

Proof. Starting from equation (16) in the main text, we have

$$f_{\tau+1}(\boldsymbol{X}^{(I)}) = (I - \epsilon K^{(I)})f_\tau(\boldsymbol{X}^{(I)}) + \epsilon K^{(I)}\boldsymbol{Y} + \epsilon\delta_\tau \tag{4}$$

where $\delta_\tau = \left(K^{(I)} - K_\tau^{(I)}\right)\left[f_\tau\left(\boldsymbol{X}^{(I)}\right) - \boldsymbol{Y}\right]$.

As discussed above, we should analyse $\boldsymbol{e}^{(I)}$. Let $f'_\tau = f_\tau - f_N^c$. Then we can write the above equation as:

$$f'_{\tau+1}(\boldsymbol{X}^{(I)}) = (I - \epsilon K^{(I)}) f'_\tau(\boldsymbol{X}^{(I)}) + \epsilon K^{(I)} \boldsymbol{e}^{(I)} + \epsilon \delta'_\tau \tag{5}$$

where $\delta'_\tau = \left(K^{(I)} - K^{(I)}_\tau\right)\left[f'_\tau\left(\boldsymbol{X}^{(I)}\right) - \boldsymbol{e}^{(I)}\right]$.

Let $\zeta^\tau = \frac{1}{\sqrt{N}} U^T f'_\tau(\boldsymbol{X}^{(I)})$ and $\zeta^* = \frac{1}{\sqrt{N}} U^T \left[f_{0,-I}(\boldsymbol{X}^{(I)}) - f^c_N(\boldsymbol{X}^{(I)})\right]$, we can write

$$\zeta^{\tau+1} = \zeta^\tau + \epsilon\Lambda \frac{\tilde{w}}{\sqrt{N}} - \epsilon\Lambda(\zeta^\tau - \zeta^*) + \frac{\epsilon U^T \delta_\tau}{\sqrt{N}} \tag{6}$$

where $\tilde{w} = U^T[\boldsymbol{e}^{(I)} - f_{0,-I}(\boldsymbol{X}^{(I)}) + f^c_N(\boldsymbol{X}^{(I)})] = U^T w^{(I)}$.

Rearranging, we have

$$\zeta^{\tau+1} - \zeta^* = (I - \epsilon\Lambda)(\zeta^\tau - \zeta^*) + \epsilon\Lambda \frac{\tilde{w}}{\sqrt{N}} + \frac{\epsilon\tilde{\delta}_\tau}{\sqrt{N}} \tag{7}$$

where $\tilde{\delta}_\tau = U^T \delta_\tau$.

Unwrapping, we have

$$\zeta^\tau - \zeta^* = (I - S^\tau)\frac{\tilde{w}}{\sqrt{N}} - S^\tau \zeta^* + \sum_{i=0}^{\tau-1} S^{\tau-i-1} \frac{\epsilon\tilde{\delta}_i}{\sqrt{N}}. \tag{8}$$

The zero initialization condition is important, otherwise we cannot derive the above equation. Then we have

$$\|\zeta^\tau - \zeta^*\|_2^2 \le \frac{4}{N}\|(I - S^\tau)\tilde{w}\|_2^2 + \frac{4\epsilon^2}{N}\|\sum_{i=0}^{\tau-1} S^{\tau-i-1}\tilde{\delta}_i\|_2^2 + 2\|S^\tau\zeta^*\|_2^2 \tag{9}$$

where we use the inequality $\|a+b\|_2^2 \le 2\left(\|a\|_2^2 + \|b\|_2^2\right)$ twice. Plug in $\tilde{w}, \tilde{\delta}_i$ and ζ^*, we can the desired results. Note that we subtract f^c_N from both f_τ and $f_{0,-I}$, so it simply cancels out. □

The difference term arises due to the evolving kernel during training. If the kernel were constant, the difference term D_τ^2 in the bound would vanish, and our algorithm would reduce to the early stopping framework proposed in Raskutti et al. (2014).

A.2 General Convergence Bound

We first show that within $\widehat{T}_{\max}$, the $L^2(P_{N,-I})$ norm satisfies a bound, and then the optimal stopping time is obtained by choosing the optimal value within $[0, \widehat{T}_{\max}]$.

Lemma A.2. *Consider a fixed design case, where $\boldsymbol{X}^{(I)}$ is considered unchanged, under assumptions 4.1-4.5 in the main text, consider a model that use gradient descent or gradient boosting with step size $\epsilon \le \min\left\{1, 1/\hat{\lambda}_1\right\}$, start from the full model f^c_N trained using $\boldsymbol{X}$, with probability at least $1 - c_1\exp\left(-c_2 N\widehat{r}_N^2\right)$ for $\tau = 1, 2, \ldots, \widehat{T}_{max}$, the following holds*

$$\left[\|f_\tau - f_{0,-I}\|_N^2\right] \le \frac{C}{\epsilon\tau} + g(\boldsymbol{X}^{(I)}, \tau) \tag{10}$$

where $g(\boldsymbol{X}^{(I)}, \tau)$ is a non-decreasing function depend on τ, c_1, c_2 and C are some universal positive constants.

Proof. By lemma 7 in Raskutti et al. (2014), for $\tau = 1, \ldots, \widehat{T}$, the squared bias is upper bounded as

$$B_\tau^2 \le \frac{C_1}{e\eta_\tau} \tag{11}$$

Moreover, because we have the assumption 4.3, according to the proof of lemma 7 in Raskutti et al. (2014), with probability at least $1 - c_1\exp\left(-c_2 N\widehat{r}_N^2\right)$, the variance term is upper bounded as

$$V_\tau \le \frac{C_2}{e\eta_\tau}. \tag{12}$$

Then it suffices to show that the difference term is upper bounded by a non-decreasing $g(\tau)$. Recall that

$$S^\tau := (I - \epsilon\Lambda)^\tau \tag{13}$$

so we have for the difference term D_τ

$$\frac{2\epsilon}{\sqrt{N}}\|\sum_{i=0}^{\tau-1} S^{\tau-1-i}\tilde{\delta}_i\|_2 \leq \frac{2\epsilon}{\sqrt{N}}\sum_{i=0}^{\tau-1}\|S^{\tau-1-i}\|_2\|\tilde{\delta}_i\| \leq \frac{2\epsilon}{\sqrt{N}}\sum_{i=0}^{\tau-1}\|\tilde{\delta}_i\|_2 = \frac{2\epsilon}{\sqrt{N}}\sum_{i=0}^{\tau-1}\|\delta_i\|_2 := \left(g(\boldsymbol{X}^{(I)},\tau)\right)^{1/2} \tag{14}$$

where we use the condition that $\epsilon \leq \min\left\{1, 1/\hat{\lambda}_1\right\}$. Observe that $g(\boldsymbol{X}^{(I)},\tau)$ is a non-decreasing function. This concludes our proof. □

Theorem A.1. *Same as lemma A.2, further suppose under some model-specific conditions, we have $g(\boldsymbol{X}^{(I)},\tau) \in \mathcal{O}(\frac{1}{\sqrt{N}})$, then for a model start from f_N^c, with probability at least $1 - c_1\exp\left(-c_2 N\widehat{r}_N^2\right)$, the following holds*

$$\|f_{\widehat{T}_{op}} - f_{0,-I}\|_N^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{N}}\right) \tag{15}$$

where $\widehat{T}_{op}$ is the optimal stopping time obtained by optimizing the RHS of (10) within $\widehat{T}_{max}$.

Proof. We show that for $\tau = \widehat{T}_{\max}$, $f_{\widehat{T}_{\max}}$ satisfies the desired bound. Then since $\widehat{T}_{\text{op}}$ is the optimal stopping time within $[0, \widehat{T}_{\max}]$, it should at least have the same convergence bound. And because we have the condition that $g(\boldsymbol{X}^{(I)},\tau) \in \mathcal{O}(\frac{1}{\sqrt{N}})$, we only need to show that

$$\frac{C}{\eta_{\widehat{T}_{\max}}} \leq \mathcal{O}\left(\frac{1}{\sqrt{N}}\right). \tag{16}$$

According to the proof of theorem 1 in Raskutti et al. (2014)

$$\frac{C}{\eta_{\widehat{T}_{\max}}} \leq C'\widehat{r}_N^2. \tag{17}$$

And by the properties of the empirical Rademacher complexity (Appendix D in Raskutti et al. (2014)), the critical radius satisfies

$$\widehat{\mathcal{R}}_K^{(I)}(\widehat{r}_N) = \frac{\widehat{r}_N^2 C_{\mathcal{H}}^2}{2e\sigma}. \tag{18}$$

Because we have $\text{tr}(K^{(I)}) = O(1)$ by assumption 4.5, then

$$\frac{\widehat{r}_N^2 C_{\mathcal{H}}^2}{2e\sigma} = \widehat{\mathcal{R}}_K(\widehat{r}_N) = \left[\frac{1}{N}\sum_{i=1}^N \min\left\{\widehat{\lambda}_i, \widehat{r}_N\right\}\right]^{1/2} \leq \left[\frac{\text{tr}(K^{(I)})}{N}\right]^{1/2} = \mathcal{O}\left(\frac{1}{\sqrt{N}}\right). \tag{19}$$

This shows that

$$\frac{\widehat{r}_N^2 C_{\mathcal{H}}^2}{2e\sigma} \leq \mathcal{O}\left(\frac{1}{\sqrt{N}}\right). \tag{20}$$

Combined with equation (17), the proof is done. □

Theorem A.1 aims to provide a general understanding of the convergence rate under some regularity conditions. For certain models, however, adjustments to the theoretical results may be required. In the Section B and Section C, we will illustrate how to adapt neural networks and GBDT to this general framework and apply the necessary modifications.

B NEURAL NETWORKS

We aim to give all the theoretical results for neural networks in this section. As stated in the main text, to fit neural networks, the key is to extend Theorem 2.1 in Lee et al. (2019) to our warm-start initialization settings. We adopt the same notations and assumptions as those Lee et al. (2019) to facilitate the proof in this section. Note that this set of notations is a little bit different from that used in the main text.

B.1 Notation and Preliminaries

Consider a fully-connected feed-forward network with L hidden layers with widths $n_l = m$, for $l = 1, \ldots, L$ and a readout layer with $n_{L+1} = 1$.[1] For each $x \in \mathbb{R}^{n_0}$, we use $h^l(x), x^l(x) \in \mathbb{R}^{n_l}$ to represent the pre- and post-activation functions at layer l with input x. The recurrence relation for a feed-forward network is defined as

$$\begin{cases} h^{l+1} = x^l W^{l+1} + b^{l+1} \\ x^{l+1} = \phi(h^{l+1}) \end{cases} \text{ and } \begin{cases} W^l_{i,j} &= \frac{\sigma_\omega}{\sqrt{n_l}} \omega^l_{ij} \\ b^l_j &= \sigma_b \beta^l_j \end{cases} \tag{21}$$

where ϕ is a point-wise activation function, $W^{l+1} \in \mathbb{R}^{n_l \times n_{l+1}}$ and $b^{l+1} \in \mathbb{R}^{n_{l+1}}$ are the weights and biases, ω^l_{ij} and b^l_j are the trainable variables, drawn i.i.d. from a standard Gaussian $\omega^l_{ij}, \beta^l_j \sim \mathcal{N}(0,1)$ at initialization, and σ^2_ω and σ^2_b are weight and bias variances. We refer to it as the NTK parameterization. This is a widely used parameterization in NTK literature (Jacot et al., 2018; Karras et al., 2018; Park et al., 2019) for theoretical convenience.

Besides the NTK parameterization, there is another parameterization called standard parameterization, which is more commonly used for training neural networks. A network with standard parameterization is defined as:

$$\begin{cases} h^{l+1} &= x^l W^{l+1} + b^{l+1} \\ x^{l+1} &= \phi(h^{l+1}) \end{cases} \text{ and } \begin{cases} W^l_{i,j} &= \omega^l_{ij} \sim \mathcal{N}\left(0, \frac{\sigma^2_\omega}{n_l}\right) \\ b^l_j &= \beta^l_j \sim \mathcal{N}\left(0, \sigma^2_b\right) \end{cases} . \tag{22}$$

Define $\theta^l \equiv \text{vec}\left(\left\{W^l, b^l\right\}\right)$, the $((n_{l-1}+1)\, n_l) \times 1$ vector of all parameters for layer l.$\theta = \text{vec}\left(\cup_{l=1}^{L+1} \theta^l\right)$ then indicates the vector of all network parameters, with similar definitions for $\theta \leq l$ and $\theta^{>l}$. Denote by θ_t the time-dependence of the parameters and by θ_0 their initial values. We use $f_\tau(x) \equiv h^{L+1}(x) \in \mathbb{R}$ to denote the output of the neural network at iteration τ. Since the standard parameterization does not have the scaling factor $\frac{1}{\sqrt{n_L}}$ in the output layer, when we define the NTK for standard parameterization, we should normalize it with the width m as follows:

$$\frac{1}{m} \left\langle \nabla_\theta f\left(\theta(t), x\right), \nabla_\theta f\left(\theta(t), x'\right) \right\rangle . \tag{23}$$

The corresponding *empirical kernel matrix* $K^{(I)}$ under standard parameterization and warm-start initialization has entries:

$$K^{(I)}(i,j) = \lim_{m \to \infty} \frac{1}{N} \cdot \frac{1}{m} \left\langle \nabla_\theta f\left(\theta(0), \mathbf{X}^{(I)}_i\right), \nabla_\theta f\left(\theta(0), \mathbf{X}^{(I)}_j\right) \right\rangle . \tag{24}$$

We define the below short-hand:

$$\begin{aligned} f\left(\theta_\tau\right) &= f\left(\boldsymbol{X}^{(I)}, \theta_\tau\right) \in \mathbb{R}^N \\ g\left(\theta_\tau\right) &= f\left(\boldsymbol{X}^{(I)}, \theta_\tau\right) - \boldsymbol{Y} \in \mathbb{R}^N \\ J\left(\theta_\tau\right) &= \nabla_\theta f\left(\theta_\tau\right) \in \mathbb{R}^{N \times |\theta|} \end{aligned} \tag{25}$$

where N is the number of training samples. Then the least-square loss we consider becomes

$$\mathcal{L}(t) = \frac{1}{2N} \left\| g\left(\theta_\tau\right) \right\|_2^2 . \tag{26}$$

The linearized network is defined as:

$$f^{\text{lin}}_\tau(x) \equiv f_0(x) + \nabla_\theta f_0(x)|_{\theta=\theta_0}\, \omega_\tau \tag{27}$$

where $\omega_\tau \equiv \theta_\tau - \theta_0$ is the change in the parameters from their initial values. The linearized network is simply first order Taylor expansion. Andthe change of w_τ for $f^{\text{lin}}_\tau(x)$ is given by

$$w_{\tau+1} - w_\tau = -\frac{\epsilon}{N} J(\theta_0)^T \left(f^{\text{lin}}_\tau(\boldsymbol{X}^{(I)}) - \boldsymbol{Y}\right). \tag{28}$$

[1] We set the output dimension to 1 for illustration purposes. The theoretical results can be easily extended to cases where $n_{L+1} = k$.

B.2 Neural Network Linearization

Same as Lee et al. (2019), we present the proof for standard parameterization, the proof can also apply to the NTK parameterization with some minor changes. We adopt the same assumptions from (Lee et al., 2019) and add additional requirement on the initialization of the full model f_N^c.

Assumption B.1. *The empirical kernel matrix induced by the stationary NTK , as width $m \to \infty$, $K^{(I)}$ is full rank, i.e. $0 < \lambda_{\min} := \lambda_{\min}(K^{(I)}) \leq \lambda_{\max} := \lambda_{\max}(K^{(I)}) < \infty$. Let $\eta_{critical} = 2\left(\lambda_{\min} + \lambda_{\max}\right)^{-1}$.*

Note that the analytic NTK matrix in Lee et al. (2019) is different than the empirical kernel matrix we consider. As ours is normalized by the sample size N, and since the loss we use is also normalized by N, the effects cancel out. This is why we can make assumption about $K^{(I)}$ directly here.

Assumption B.2. *The activation function ϕ satisfies*

$$|\phi(0)|, \quad \|\phi'\|_\infty, \quad \sup_{x \neq \tilde{x}} |\phi'(x) - \phi'(\tilde{x})| / |x - \tilde{x}| < \infty. \tag{29}$$

Assumption B.3. *The full model f_N^c has same structure as the reduced model, i.e. width, layer, dimension, and is trained under normal random initialization as in (22) using complete features $\boldsymbol{X}$ using gradient descent with learning rate $\epsilon = \frac{\eta_0}{m}$.*

We denote the parameter of f_N^c as θ_0, and let θ_0' be the parameter of the initial network used to train f_N^c. Follow the same proof strategy of Lee et al. (2019), we prove convergence of neural network training and the stability of NTK for discrete gradient descent under our special warm-start initialization. We first prove the local Lipschitzness of the Jacobian, and then prove the global convergence of the NTK. Finally, with both of these two results we are able to show that the linearization of neural network under our setup is valid.

Lemma B.1 (Local Lipschitzness of the Jacobian under warm-start initialization)**.** *Under assumptions B.1-B.3, there is a $L > 0$ such that for every $C > 0$, for $\gamma > 0$ and $\eta_0 < \eta_{critical}$, there exists $M \in \mathbb{N}$, for $m \geq M$, the following holds with probability at least $(1 - \gamma)$ over warm-start initialization*

$$\begin{cases} \frac{1}{\sqrt{m}}\|J(\theta) - J(\tilde{\theta})\|_F & \leq L\|\theta - \tilde{\theta}\|_2 \\ \frac{1}{\sqrt{m}}\|J(\theta)\|_F & \leq L \end{cases}, \quad \forall \theta, \tilde{\theta} \in B\left(\theta_0, Cm^{-\frac{1}{2}}\right) \tag{30}$$

where

$$B\left(\theta_0, R\right) := \{\theta : \|\theta - \theta_0\|_2 < R\} \tag{31}$$

and θ_0 is the parameter of f_N^c.

Proof. Since the full model f_N^c is training from random normal initialization, we can use the theoretical results in Lee et al. (2019) directly to prove this result. By theorem G.1 in Lee et al. (2019), $\exists M \in \mathbb{N}$, for $m \geq M$, the following holds with probability at least $1 - \frac{\gamma}{2}$:

$$\|\theta_0 - \theta_0'\|_2 \leq C_1 m^{-\frac{1}{2}} \tag{32}$$

where θ_0' is the parameter of a network with initialization in (22). Then consider any $\theta, \tilde{\theta} \in B\left(\theta_0, Cm^{-\frac{1}{2}}\right)$

$$\|\theta - \theta_0'\|_2 \leq \|\theta - \theta_0'\|_2 + \|\theta_0 - \theta_0'\|_2 \leq (C + C_1)m^{-\frac{1}{2}} \tag{33}$$

same argument holds for $\tilde{\theta}$. Thus we have $\theta, \tilde{\theta} \in B\left(\theta_0', (C + C_1)m^{-\frac{1}{2}}\right)$. Apply lemma 1 in Lee et al. (2019) with probability $1 - \frac{\gamma}{2}$ we can get the desired result. □

Remark B.1. *Note that the width m requirement here is for the full model f_N^c, because we used Theorem G.1 in Lee et al. (2019) w.r.t. f_N^c.*

Theorem B.1. *Under assumptions B.1-B.3, for $\gamma > 0$ and $\eta_0 < \eta_{critical}$, there exist $R_0 > 0, M \in \mathbb{N}$ and $L > 1$, such that for every $m \geq M$, the following holds with probability at least $(1 - \gamma)$ over warm-start initialization when applying gradient descent with learning rate $\epsilon = \frac{\eta_0}{m}$,*

$$\begin{cases} \|g\left(\theta_\tau\right)\|_2 \leq \left(1 - \frac{\eta_0 N \lambda_{\min}}{3}\right)^\tau R_0 \\ \sum_{j=1}^\tau \|\theta_j - \theta_{j-1}\|_2 \leq \frac{\eta_0 K^{(I)} R_0}{\sqrt{n}} \sum_{j=1}^\tau \left(1 - \frac{\eta_0 N \lambda_{\min}}{3}\right)^{j-1} \leq \frac{3LR_0}{N\lambda_{\min}} m^{-\frac{1}{2}} \end{cases} \tag{34}$$

and

$$\sup_{\tau} \left\| K_0^{(I)} - K_\tau^{(I)} \right\|_F \leq \frac{6L^3 R_0}{N^2 \lambda_{\min}} m^{-\frac{1}{2}} \tag{35}$$

where $K_0^{(I)}$ is the empirical kernel matrix induced by f_N^c and $K_\tau^{(I)}$ is the one induced by f_τ using dropping features $\boldsymbol{X}^{(I)}$.

Proof. Follow the same proof techniques as theorem G.1 in Lee et al. (2019), we first need to have

$$\|g(\theta_0)\|_2 < R_0. \tag{36}$$

This is ensured by assumption 4.4 in the main text. Note that in Lee et al. (2019), the input data is considered to be fixed, so the sample size is just a constant. But in our setup, we need to be more careful with the constants that are dependent with the training size N as this is crucial when we obtain the prediction error bound in terms of N. Here the R_0 is a constant of order $O(\sqrt{N})$. We do not write the dependence of R_0 on N here, but we will be careful about the effects of R_0 in the proof of prediction error bound.

The next condition we need to satisfy is something similar to equation (S61) in Lee et al. (2019). Specifically, we should have with high probability

$$\|K^{(I)} - K_0^{(I)}\|_F \leq \frac{\eta_0 \lambda_{min}}{3}. \tag{37}$$

Note that the sample size N on both sides cancel out, and here the empirical kernel matrix is calculated using dropped data $\boldsymbol{X}^{(I)}$. The key difference in our setup is that we drop features in set I, the data becomes different. A fact about NTK is that, the convergence of NTK when width goes to infinity is independent of data. So even though we change the input data, this result still holds.

The full model is trained using complete data $\boldsymbol{X}$ and random normal initialization, so theorem G.1 in Lee et al. (2019), the difference between θ_0 and some random initialization θ_0' is still within $O(m^{-\frac{1}{2}})$. Then by Lemma 1 in Lee et al. (2019), we can show that the convergence of NTK still holds. This argument is essentially the same as the reasoning as equation (S217) in Lee et al. (2019). The reason why we can do this is because in the proof of Lemma 1 in Lee et al. (2019), the constant depends on the training sample size N rather than specific data we use. Notice that in the detailed proof of *lemma* 1 in Lee et al. (2019), in equation (S85) and (S86), we can set the constant large enough so that for any input data, the local Lipschitz condition holds. This is ensured by the assumption that the input data lies in a closed set.

Finally, combined with lemma B.1, we are able to prove this result using the same proof strategy of theorem G.1 in Lee et al. (2019). □

Remark B.2. *The assumption 4.4 seems too strong in the case of neural network. But we can formally show that the dropout error is at least $O_p(\sqrt{N})$. First when we train the full model $\|g(\theta_0', \boldsymbol{X})\|_2$ is $O_p(\sqrt{N})$, where θ_0' is the parameter with random normal initialization (see proof in Lee et al. (2019) (S44)). Then since the loss is decreasing over training $\|g(\theta_0, \boldsymbol{X})\|_2$, where θ_0 is the parameter of f_N^c. Remember we dropped data, so we need to show that $\|g(\theta_0, \boldsymbol{X}^{(I)})\|_2$ is $O_p(\sqrt{N})$. This is sufficient to show that $f(\theta)$ is lipschitz continuous w.r.t. input data with high probability. This is not hard to show as θ_0 and θ_0' are close and $\|W_0\|_{op}$ is bounded w.h.p. similar to arguments as (S84) in Lee et al. (2019) .*

Theorem B.2. *Under assumptions B.1-B.3, for $\gamma > 0$ arbitrarily small and $\eta_0 < \eta_{critical}$, there exist $R_0 > 0$ and $M \in \mathbb{N}$ such that for every $m \geq M$, with probability at least $(1-\gamma)$ over warm-start initialization when applying gradient descent with learning rate $\epsilon = \frac{\eta_0}{m}$*

$$\sup_{\tau} \left\| f_\tau^{lin}(\boldsymbol{X}^{(I)}) - f_\tau(\boldsymbol{X}^{(I)}) \right\|_2 \lesssim m^{-\frac{1}{2}} R_0^2 \tag{38}$$

where $\lesssim$ is to hide the dependence on some uninteresting constants.

Proof. With lemma B.1 and theorem B.1, follow the same proof techniques as Theorem H.1 in Lee et al. (2019) we can prove this result. Note that Theorem H.1 in Lee et al. (2019) is for gradient flow. For the gradient descent case, the update of linear model is given by:

$$\begin{aligned} &\omega_{t+1} - \omega_t = -\epsilon \nabla_\theta f_0(\mathcal{X})^T \nabla_{f_t^{\text{lin}}(\mathcal{X})} \mathcal{L} \\ &f_{\tau+1}^{\text{lin}}(x) - f_\tau^{\text{lin}}(x) = -\epsilon J(\theta_0, x) J(\theta_0, \boldsymbol{X}^{(I)})^T \nabla_{f_t^{\text{lin}}(\mathcal{X})} \mathcal{L}. \end{aligned} \tag{39}$$

Note that the update for function value is exact, because the linear model only contains linear terms. Then by induction, it is not hard to show that

$$f_\tau^{\text{lin}}(\boldsymbol{X}^{(I)}) = \left(I - \left(I - \epsilon K_0^{(I)}\right)^\tau\right)\boldsymbol{Y} + \left(I - \eta K_0^{(I)}\right)^\tau f_0(\boldsymbol{X}^{(I)}) \tag{40}$$

For the nonlinear counterpart, we need to use mean value theorem, at each update we have

$$f_{\tau+1}(x) - f_{\tau+1}(x) = J(\tilde{\theta}_\tau)\Delta\theta \tag{41}$$

where $\tilde{\theta}_\tau$ is some some linear interpolation between θ_τ and $\theta_{\tau+1}$. If we use $J(\theta_\tau)$, the expression is not exact and there will be an second order Taylor expansion remainder term. Then use lemma B.1 and theorem B.1, it is not hard to prove the results (apply similar arguments as in the proof of Lemma B.2)

□

Remark B.3. *For NTK parameterization, analogues of these theoretical results still hold. For the local Lipschitzness, we just drop the scaling factor $\frac{1}{\sqrt{m}}$. And for the stability and linearization results, we replace learning rate with $\epsilon = \eta_0$. And our requirement on f_N^c becomes initializing all the parameters with $\mathcal{N}(0,1)$. The proof are almost the same with some minor changes.*

B.3 Kernel Gradient Update

In this subsection, we show how to write the gradient update similar to the form as equation (20) in the main text. By using the results proved in previous subsection, we have the following result for neural network for standard parameterization. For NTK parameterization, replace the learning rate ϵ with η_0.

Lemma B.2. *Under assumptions B.1-B.3, for a neural network applying warm-start initialization and use gradient descent with learning rate $\epsilon = \frac{\eta_0}{m}$ for standard parameterization, and $\eta_0 < \eta_{critical}$, the iterative kernel update equation for neural network can be written as*

$$f_{\tau+1}(\boldsymbol{X}^{(I)}) = (I - \eta_0 K^{(I)}) f_\tau(\boldsymbol{X}^{(I)}) + \eta_0 K^{(I)}\boldsymbol{Y} + \eta_0\delta_\tau \tag{42}$$

where δ_τ is a term containing errors caused by the changing kernel and linearlization of neural networks. And for any $\gamma > 0$, there exists $M \in \mathbb{N}$, for a network described above with width $m \geq M$, the following holds with probability at least $1 - \gamma$

$$\|\delta_\tau\|_2 \leq \mathcal{O}\left(m^{-\frac{1}{2}}\right). \tag{43}$$

Proof. By theorem B.2, we have for any $\gamma > 0$, there exists $M_1 \in \mathbb{N}$, a full connected neural network with width $m \geq M_1$, with probability at least $1 - \frac{\gamma}{3}$

$$\begin{aligned} f_\tau(\boldsymbol{X}^{(I)}) &= f_0(\boldsymbol{X}^{(I)}) + \nabla_\theta f_0(\boldsymbol{X}^{(I)})\Big|_{\theta=\theta_0}\omega_\tau + e_\tau \\ f_{\tau+1}(\boldsymbol{X}^{(I)}) &= f_0(\boldsymbol{X}^{(I)}) + \nabla_\theta f_0(\boldsymbol{X}^{(I)})\Big|_{\theta=\theta_0}\omega_{\tau+1} + e_{\tau+1} \end{aligned} \tag{44}$$

where for e_τ and $e_{\tau+1}$ we have

$$\|e_\tau\|_2, \|e_{\tau+1}\|_2 \lesssim m^{-\frac{1}{2}} R_0^2. \tag{45}$$

Subtract these two equations, we have

$$\begin{aligned} f_{\tau+1}(\boldsymbol{X}^{(I)}) &= f_\tau(\boldsymbol{X}^{(I)}) + \nabla_\theta f_0(\boldsymbol{X}^{(I)})\Big|_{\theta=\theta_0}(\omega_{\tau+1} - \omega_\tau) + e_{\tau+1} - e_\tau \\ &= f_\tau(\boldsymbol{X}^{(I)}) - \frac{\eta_0}{mN} J(\theta_0)J(\theta_0)^T g(\theta_\tau) + \frac{\eta_0}{mN} J(\theta_0)J(\theta_0)^T e_\tau + e_{\tau+1} - e_\tau \\ &= f_\tau(\boldsymbol{X}^{(I)}) - \eta_0 K^{(I)} g(\theta_\tau) + \eta_0(K^{(I)} - K_0^{(I)})g(\theta_\tau) + \eta_0 K^{(I)} e_\tau + e_{\tau+1} - e_\tau \\ &= f_\tau(\boldsymbol{X}^{(I)}) - \eta_0 K^{(I)}(f_\tau(\boldsymbol{X}^{(I)}) - \boldsymbol{Y}) + \eta_0\delta_\tau \end{aligned} \tag{46}$$

where δ_τ is

$$(K^{(I)} - K_0^{(I)})g(\theta_\tau) + K^{(I)} e_\tau + \frac{e_{\tau+1} - e_\tau}{\eta_0}. \tag{47}$$

Then apply lemma B.1 and theorem B.1 with probability $1-\frac{\gamma}{3}$, there exists M_2, s.t. for any $m \geq M_2$, we can bound the L_2 norm of $\eta_0\delta_\tau$ with probability at least $1-\gamma$

$$\|\delta_\tau\|_2 \leq \mathcal{O}\left(\frac{R_0^2}{m^{\frac{1}{2}}}\right). \tag{48}$$

Choose $M = \max\{M_1, M_2\}$, then we finish our proof. □

We can see that here since neural network has an additional linearlization error, the term δ_τ is more complicated than the noraml one in the general theretical framework.

B.4 Error Bound

Corollary B.2.1. *Same as lemma B.2, and under assumptions 4.1-4.4 in the main text, for any $\gamma > 0$, there exists $M \in \mathbb{N}$, a full connected neural network with width $m \geq M$, the following holds with probability at least $1-\gamma-c_1\exp\left(-c_2 N\widehat{r}_N^2\right)$ over warm-start initialization when applying gradient descent with learning rate $\epsilon = \frac{\eta_0}{m}$, and η_0 for stanrdard parameterization and NTK parameterization, respectively*

$$\|f_{\widehat{T}_{op}} - f_{0,-I}\|_N^2 \leq \mathcal{O}\left(N^{-\frac{1}{2}}\right). \tag{49}$$

Proof. First, we need to show that $\mathrm{tr}(K^{(I)}) = O(1)$ in case of neural networks. For NTK parameterization, by definition

$$\mathrm{tr}(K^{(I)}) = \lim_{m\to\infty}\frac{1}{N}\sum_{i=1}^{N}\left\langle \nabla_\theta f\left(\theta(0), \mathbf{X}_i^{(I)}\right), \nabla_\theta f\left(\theta(0), \mathbf{X}_i^{(I)}\right)\right\rangle .. \tag{50}$$

Observed that in our setting, the NTK expression is the same as the one stated in theorem 1 in Jacot et al. (2018). So for each entry, it is $O(1)$ because of assumption 4.2 in the main text. This implies that $\mathrm{tr}(K^{(I)})$ is also $O(1)$. We can also interpret this as a result of CLT. Note that for standard parameterization, there would be a scaling factor $\frac{1}{m}$, and by Lee et al. (2019), we have the same results.

Next, we need to bound the difference term D_τ^2. By the proof of lemma A.2 , we know that

$$D_\tau^2 \leq \frac{4\eta_0^2}{N}\left(\sum_{i=0}^{\tau-1}\|\delta_i\|_2\right)^2. \tag{51}$$

Then apply lemma B.2, there exists $M_1 \in \mathbb{N}$, for any $m \geq M_1$, we have with probability at least $1-\gamma$

$$D_\tau^2 \leq \mathcal{O}\left(\frac{\tau^2 R_0^4}{Nm}\right) \tag{52}$$

then we can choose a large $M_2 \in \mathbb{R}$, for any $m \geq M_2$, D_τ^2 is bounded by $\mathcal{O}(N^{-\frac{1}{2}})$. We set $M = \max\{M_1, M_2\}$, and this concludes the proof.

□

In equation (51), we use learning rate $\epsilon = \eta_0$ for both parameterizations. This is becasue for both parameterization, lemma B.2 has the same resulting equation. The $\frac{1}{m}$ scaling factor for standard parameterization disappears when we use lemma B.1 and theorem B.1.

We do not provide a specific rate for m because, first, we do not know how many iterations are required. The suboptimal stopping time $\widehat{T}_{\max}$ could also depend on N, and we cannot assume it is bounded by a fixed value. Second, several constants in the bound, such as R_0^2, depend on N, and we have hidden some constants in previous proofs, like L, which is also of order $O(N)$. Third, in the proof of Theorem B.1, same as the proof of Theorem G.1 in Lee et al. (2019), the exact rate of M is hard to figure out, see equation (S68) in Lee et al. (2019). In summary, we present the results in terms of the existence of $M \in \mathbb{N}$ for m to ensure theoretical precision.

B.5 Discussion

One major advantage of using neural networks is that, according to the universal approximation theory for arbitrary width and bounded depth (Funahashi and Ken-Ichi, 1989; Hornik, 1991), we can approximate any continuous target function. This makes our approach a model-agnostic VI estimation method, as it doesn't rely on strong assumptions about $f_{0,-I}$. In this regard, our framework is equivalent to the lazy estimation method using neural networks proposed in Gao et al. (2022).

C GRADIENT BOOSTED DECISION TREES

This section is devoted to show the theoretical results for GBDT. In this paper, we used the tree and RKHS structures defined in Ustimenko et al. (2023). We also define a new set of necessary notations here to facilitate the presentation of proof. This set of notations is also independent of the notations used in the main text.

C.1 Notations

We list the used notations in this section below, some of them are the same as the main text:

- $\|\cdot\|_{\mathbb{R}^N}$ — $\|\cdot\|_2$ of $f(\boldsymbol{X}^{(I)})$;
- ρ — distribution of features;
- N — number of samples;
- $\mathcal{V}$ — set of all possible tree structures;
- $L_\nu : \mathcal{V} \to \mathbb{N}$ — number of leaves for $\nu \in \mathcal{V}$
- $D(\nu, z)$ — score used to choose a split (3);
- $\mathcal{S}$ — indices of all possible splits;
- n — number of borders in our implementation of SampleTree;
- d — depth of the tree in our implementation of SampleTree;
- β — random strength;
- ϵ — learning rate;
- $\mathcal{F}$ — space of all possible ensembles of trees from $\mathcal{V}$;
- $\phi_\nu^{(j)}$ — indicator of j-th leaf;
- $k_\nu(\cdot, \cdot)$ — single tree kernel;
- $L(f)$ — $\frac{1}{2N}\left\|\boldsymbol{Y} - f\left(\boldsymbol{X}^{(I)}\right)\right\|_{\mathbb{R}^N}^2$, empirical error of a model f;
- f_* — $\arg\min_{f\in\mathcal{F}} L(f)$, empirical minimizer ;
- $V(f)$ — $L(f) - L(f_*)$, excess risk;
- R — $\|f_*\|_{\mathbb{R}^N}$;
- $\mathbb{K}(\cdot, \cdot)$ — stationary kernel of the GBDT;
- $p(\cdot \mid f, \beta)$ — distribution of trees, $f \in \mathcal{F}$;
- $\pi(\cdot) = \lim_{\beta\to\infty} p(\cdot \mid f, \beta) = p\left(\cdot \mid f_*, \beta\right)$ — stationary distribution of trees;
- $\boldsymbol{e}^{(I)}$ — dropout error $\boldsymbol{Y} - f_N^c(\boldsymbol{X}^{(I)})$;
- M_β — $e^{\frac{dR^2}{N\beta}}$.

C.2 Tree Structure

Assume that given a finite set $\mathcal{V}$ of weak learners used for gradient boosting. Each ν corresponds to a decision tree. Let $\phi_\nu : X \to \{0,1\}^{L_\nu}$. Having ϕ_ν, define a weak learner associated with it as $x \mapsto \langle \theta, \phi_\nu(x) \rangle_{\mathbb{R}^{L_\nu}}$, for $\theta \in \mathbb{R}^{L_\nu}$, which is leaf values. Define a linear space $\mathcal{F} \subset L_2(\rho)$ of all possible ensembles of trees from $\mathcal{V}$:

$$\mathcal{F} = \operatorname{span}\left\{ \phi_\nu^{(j)}(\cdot) : X \to \{0,1\} | \nu \in \mathcal{V}, j \in \{1, \ldots, L_\nu\} \right\}. \tag{53}$$

Since $\mathcal{V}$ is finite and therefore $\mathcal{F}$ is finite-dimensional and thus topologically closed.

The weak learner *SampleTree* here is oblivious decision tree, all nodes at a given level share the same splitting criterion (feature and threshold). The tree is built in a top-down greedy manner. To reduce the possible number of splits for each feature, we discretize each feature into $n+1$ bins. This means that for each feature, there are n potential thresholds that can be selected freely. A common method involves quantizing the feature so that each of the $n+1$ buckets contains roughly an equal number of training samples. The depth of the tree is constrained to a maximum of d. Remember, we represent the collection of all potential tree structures as $\mathcal{V}$. At each step, the algorithm chooses one split among all the remaining candidates based on the following *score* defined for $\nu \in \mathcal{V}$ and residuals z :

$$D(\nu, z) := \frac{1}{N} \sum_{j=1}^{L_v} \frac{\left(\sum_{i=1}^N \phi_\nu^{(j)}(x_i)\, z_i \right)^2}{\sum_{i=1}^N \phi_\nu^{(j)}(x_i)}. \tag{54}$$

Algorithm 1 SampleTree$(z; d, n, \beta)$

input: residuals $z = (z_i)_{i=1}^N$
output: oblivious tree structures $\nu \in \mathcal{V}$
hyper-parameters: number of feature splits n, max tree depth d, random strength $\beta \in [0, \infty)$
definitions:
$S = \{(j,k) | j \in \{1, \ldots, d\}, k \in \{1, \ldots, n\}\}$ — indices of all possible splits
instructions:
initialize $i = 0, \nu_0 = \emptyset, S^{(0)} = S$
repeat
  sample $(u_i(s))_{s \in S^i} \sim U\left([0,1]^{nd-i}\right)$
  choose the next split as $\{s_{i+1}\} =$
  $\arg\max_{s \in S^{(i)}} \left(D\left((\nu_i, s), z\right) - \beta \log(-\log u_i(s)) \right)$
  update tree: $\nu_{i+1} = (\nu_i, s_{i+1})$
  update candidate splits: $S^{(i+1)} = S^{(i)} \setminus \{s_{i+1}\}$
  $i = i + 1$
until $i \geq d$ **or** $S^{(i)} = \emptyset$
return: ν_i

Algorithm 2 TrainGBDT$(\mathcal{D}; \epsilon, T, d, n, \beta)$

input: dataset $\mathcal{D} = (\boldsymbol{X}^{(I)}, \boldsymbol{Y})$
hyper-parameters: learning rate $\epsilon > 0$, regularization $\lambda > 0$, iteration of boosting T, parameters of SampleTree d, n, β

instructions:
initialize $\tau = 0, f_0(\cdot) = 0$

repeat
  $z_\tau = \boldsymbol{Y} - f_\tau(\boldsymbol{X}^{(I)})$
  $\nu_\tau = \text{SampleTree}(z_\tau; d, n, \beta)$
  $\theta_\tau = \left(\frac{\sum_{i=1}^N \phi_{\nu_\tau}^{(j)}(x_i) z_\tau^{(i)}}{\sum_{i=1}^N \phi_{\nu_\tau}^{(j)}(x_i)} \right)_{j=1}^{L_{\nu_\tau}}$
  $f_{\tau+1}(\cdot) = (1 - \frac{\lambda\epsilon}{N}) f_\tau(\cdot) + \epsilon \langle \phi_{\nu_\tau}(\cdot), \theta_\tau \rangle_{\mathbb{R}^{L_{\nu_\tau}}}$
  $\tau = \tau + 1$
until $\tau \geq T$
return: $f_T(\cdot)$

The *SampleTree* algorithm induces a local family of distribution $p(\nu | f, \beta)$ for each $f \in \mathcal{F}$:

$$p(\nu \mid f, \beta) = \sum_{\varsigma \in \mathcal{P}_d} \prod_{i=1}^d \frac{e^{\frac{D(\nu_{\varsigma,i}, z)}{\beta}}}{\sum_{s \in \mathcal{S} \setminus \nu_{\varsigma,i-1}} e^{\frac{D((\nu_{\varsigma,i-1}, s), z)}{\beta}}} \tag{55}$$

where the sum is over all permutations $\varsigma \in \mathcal{P}_d, \nu_{\varsigma,i} = (s_{\varsigma(1)}, \ldots, s_{\varsigma(i)}), \nu = (s_1, \ldots, s_d)$.

C.3 RKHS Structure

We define necessary kernels in this subsection. These kernels are adopted directly from Ustimenko et al. (2023).

Definition C.1. *A weak learner's kernel $k_\nu(\cdot, \cdot)$ is a kernel function associated with a tree structure $\nu \in \mathcal{V}$ which can be defined as:*

$$k_\nu(x, x') = \sum_{j=1}^{L_\nu} w_\nu^{(j)} \phi_\nu^{(j)}(x) \phi_\nu^{(j)}(x'), \text{ where } w_\nu^{(j)} = \frac{N}{\max\left\{N_\nu^{(j)}, 1\right\}}, N_\nu^{(j)} = \sum_{i=1}^N \phi_\nu^{(j)}(x_i). \tag{56}$$

This weak learner's kernel is a building block for any other possible kernel in boosting and is used to define the iterations of the boosting algorithm analytically.

Definition C.2. *We also define a greedy kernel of the gradient-boosting algorithm as follows:*

$$\mathbb{K}_f\left(x, x'\right) = \sum_{\nu \in \mathcal{V}} k_\nu\left(x, x'\right) p(\nu \mid f, \beta). \tag{57}$$

This greedy kernel is a kernel that guides the GBDT iterations.

Definition C.3. *Finally, there is a stationary kernel $\mathbb{K}\left(x, x'\right)$ that is independent from f :*

$$\mathbb{K}\left(x, x'\right) = \sum_{\nu \in \mathcal{V}} k_\nu\left(x, x'\right) \pi(\nu). \tag{58}$$

The greedy kernel converges to the stationary kernel as the training progresses (Ustimenko et al., 2023). In other words, $\mathbb{K}_{f_*} = \mathbb{K}$.

C.4 Kernel Gradient Update

As mentioned in the main text, because of the randomness of the *SampleTree* algorithm, we need to analyze $\mathbb{E}f_\tau$ to be consistent with our general framework. In this subsection, we show the necessary modifications of the general framework to fit the GBDT algorithm.

Because of the additive structure of the boosting algorithm, we can subtract $f_N^c(\boldsymbol{X}^{(I)})$ from $\boldsymbol{Y}$. We can then analyze from the dropout error $\boldsymbol{e}^{(I)}$ and regard this as a GBDT with initialization 0. Because $f_N^c(\boldsymbol{X}^{(I)})$ is also generated by *SampleTree*, it is also random, so $\boldsymbol{e}^{(I)}$ is random.

Lemma C.1. *The iterative kernel update equation for the particular GBDT we consider is:*

$$\mathbb{E}f_{\tau+1}(\boldsymbol{X}^{(I)}) = (I - \epsilon\mathbb{E}K^{(I)})\mathbb{E}f_\tau(\boldsymbol{X}^{(I)}) + \epsilon\mathbb{E}K^{(I)}\mathbb{E}\boldsymbol{e}^{(I)} + \epsilon\delta_\tau \tag{59}$$

where $\delta_\tau = \mathbb{E}(K^{(I)} - K_\tau^{(I)})[f_\tau(\boldsymbol{X}^{(I)}) - \boldsymbol{e}^{(I)}]$ and the expectation is taken w.r.t. the randomness of the SampleTree algorithm.

The corresponding error decomposition then becomes:

$$\begin{aligned}\|\mathbb{E}_u f_\tau - f_{0,-I}\|_N^2 \leq & \underbrace{2\sum_{j=1}^{r}[S^\tau]_{jj}^2\left(\zeta_{jj}^*\right)^2 + 2\sum_{j=r+1}^{n}\left(\zeta_{jj}^*\right)^2}_{Bias\ B_\tau^2} + \underbrace{\frac{4}{N}\sum_{j=1}^{r}\left(1 - S_{jj}^\tau\right)^2\left[U^T w^{(I)}\right]_j^2}_{Variance\ V_\tau} \\ & + \underbrace{\frac{4\epsilon^2}{N}\|\sum_{i=0}^{\tau-1} S^{\tau-1-i}\tilde{\delta}_i\|_2^2}_{Difference\ D_\tau^2}\end{aligned} \tag{60}$$

where $\zeta_{jj}^* = \left[\frac{1}{\sqrt{N}}U^T\left(f_{0,-I}(\boldsymbol{X}^{(I)}) - \mathbb{E}_u f_N^c(\boldsymbol{X}^{(I)})\right)\right]_j$, f_τ here has absorbed f_N^c and $\mathbb{E}_u$ here means taking expectation w.r.t. the randomness of the SampleTree algorithm.

Proof. By lemma 3.7 in Ustimenko et al. (2023), the iteration of GBDT can be written as

$$f_{\tau+1} = f_\tau + \frac{\epsilon}{N}k_{\nu_\tau}(\cdot, \boldsymbol{X}^{(I)})[\boldsymbol{e}^{(I)} - f_\tau(\boldsymbol{X}^{(I)})],\ \nu_\tau \sim p(\nu|f_\tau, \beta) \tag{61}$$

The empirical kernel matrix at each iteration is

$$K_\tau^{(I)} = \frac{1}{N}k_\nu\left(\boldsymbol{X}^{(I)}, \boldsymbol{X}^{(I)}\right) = \frac{1}{N}\oplus_{i=1}^{L_\nu} w_{\nu_\tau}^{(i)}\mathbf{1}_{N_\nu^i \times N_\nu^i} \tag{62}$$

The eigenvalues of $K_\tau^{(I)}$ are $(1,\dots,1,0,\dots,0)$, with first L_{ν_τ} eigenvalues being 1. Take the expectations w.r.t. the randomness of *SampleTree* algorithm, we have

$$\mathbb{E}K_\tau^{(I)} = \frac{1}{N}\mathbb{K}_f^{(I)}\left(\boldsymbol{X}^{(I)},\boldsymbol{X}^{(I)}\right) = \sum_{\nu\in\mathcal{V}}\frac{1}{N}k_\nu\left(\boldsymbol{X}^{(I)},\boldsymbol{X}^{(I)}\right)p(\nu|f_\tau,\beta) \tag{63}$$

By the iteration update rule, we can write

$$\begin{aligned} f_{\tau+1}(\boldsymbol{X}^{(I)}) &= f_\tau(\boldsymbol{X}^{(I)}) + \frac{\epsilon}{N}k_{\nu_\tau}(\boldsymbol{X}^{(I)},\boldsymbol{X}^{(I)})[e^{(I)} - f_\tau(\boldsymbol{X}^{(I)})] \\ &= f_\tau(\boldsymbol{X}^{(I)}) - \epsilon K_\tau^{(I)}[f_\tau(\boldsymbol{X}^{(I)}) - \boldsymbol{e}^{(I)}] \\ &= (I-\epsilon K_\tau^{(I)})f_\tau(\boldsymbol{X}^{(I)}) + \epsilon K_\tau^{(I)}\boldsymbol{e}^{(I)} \end{aligned} \tag{64}$$

where $\nu_\tau \sim p(\nu|f_\tau,\beta)$ and $K_\tau^{(I)} = \frac{1}{N}k_{\nu_\tau}(\boldsymbol{X}^{(I)},\boldsymbol{X}^{(I)})$.

For the stationary kernel $\mathbb{K}^{(I)}$, we have

$$\mathbb{E}K^{(I)} = \frac{1}{N}\sum_\nu k_\nu(\boldsymbol{X}^{(I)},\boldsymbol{X}^{(I)})\pi(\nu) \tag{65}$$

Apply SVD, we have $\mathbb{E}K^{(I)} = U\Lambda U^T$.

Since $K^{(I)}$ is independent of f_τ and $\boldsymbol{e}^{(I)}$, we can write

$$\mathbb{E}f_{\tau+1}(\boldsymbol{X}^{(I)}) = (I-\epsilon\mathbb{E}K^{(I)})\mathbb{E}f_\tau(\boldsymbol{X}^{(I)}) + \epsilon\mathbb{E}K^{(I)}\mathbb{E}\boldsymbol{e}^{(I)} + \epsilon\delta_\tau \tag{66}$$

where $\delta_\tau = \mathbb{E}(K^{(I)} - K_\tau^{(I)})[f_\tau(\boldsymbol{X}^{(I)}) - \boldsymbol{e}^{(I)}]$.

Let $\zeta^\tau = \frac{1}{\sqrt{N}}U^T\mathbb{E}f_\tau(\boldsymbol{X}^{(I)})$ and $\zeta^* = \frac{1}{\sqrt{N}}U^T\mathbb{E}[f_{0,-I}(\boldsymbol{X}^{(I)}) - f_N^c(\boldsymbol{X}^{(I)})]$, we can write

$$\zeta^{\tau+1} = \zeta^\tau + \epsilon\Lambda\frac{\tilde{w}}{\sqrt{N}} - \epsilon\Lambda(\zeta^\tau - \zeta^*) + \frac{\epsilon U^T\delta_\tau}{\sqrt{N}} \tag{67}$$

where $\tilde{w} = U^T[\mathbb{E}e^{(I)} - f_{0,-I}(\boldsymbol{X}^{(I)}) + \mathbb{E}f_N^c(\boldsymbol{X}^{(I)})]$. Here we can regard the response as $\mathbb{E}e^{(I)}$ and its true generating function is $f_{0,-I}(\boldsymbol{X}^{(I)}) - \mathbb{E}f_N^c(\boldsymbol{X}^{(I)})$ under reduced data. Since $\mathbb{E}e^{(I)} = f_{0,-I}(\boldsymbol{X}^{(I)}) - \mathbb{E}f_N^c(\boldsymbol{X}^{(I)})] + w^{(I)}$, $\tilde{w} = U^T w^{(I)}$.

Rearranging, we have

$$\zeta^{\tau+1} - \zeta^* = (I-\epsilon\Lambda)(\zeta^\tau - \zeta^*) + \epsilon\Lambda\frac{\tilde{w}}{\sqrt{N}} + \frac{\epsilon\tilde{\delta}_\tau}{\sqrt{N}} \tag{68}$$

where $\tilde{\delta}_\tau = U^T\delta_\tau$.

Unwrapping, we have

$$\zeta^\tau - \zeta^* = (I-S^\tau)\frac{\tilde{w}}{\sqrt{N}} - S^\tau\zeta^* + \sum_{i=0}^{\tau-1}S^{\tau-i-1}\frac{\epsilon\tilde{\delta}_i}{\sqrt{N}} \tag{69}$$

Then we have

$$\|\zeta^\tau - \zeta^*\|_2^2 \le \frac{4}{N}\|(I-S^\tau)\tilde{w}\|_2^2 + \frac{4\epsilon^2}{N}\|\sum_{i=0}^{\tau-1}S^{\tau-i-1}\tilde{\delta}_i\|_2^2 + 2\|S^\tau\zeta^*\|_2^2. \tag{70}$$

Plug in $\tilde{w}, \tilde{\delta}_i$ and ζ^*, we then get the error decomposition result for GBDT as follows:

$$\begin{aligned} \|\mathbb{E}_u f_\tau - f_{0,-I}\|_N^2 \le & \underbrace{2\sum_{j=1}^{r}[S^\tau]_{jj}^2\left(\zeta_{jj}^*\right)^2 + 2\sum_{j=r+1}^{n}\left(\zeta_{jj}^*\right)^2}_{\text{Bias } B_\tau^2} + \underbrace{\frac{4}{N}\sum_{j=1}^{r}\left(1-S_{jj}^\tau\right)^2\left[U^T w^{(I)}\right]_j^2}_{\text{Variance } V_\tau} \\ & + \underbrace{\frac{4\epsilon^2}{N}\|\sum_{i=0}^{\tau-1}S^{\tau-1-i}\tilde{\delta}_i\|_2^2}_{\text{Difference } D_\tau^2} \end{aligned} \tag{71}$$

where f_τ here has absorbed f_N^c. □

C.5 Error Bound

To prove corollary in the main text, we need to bound the term δ_τ first to get a final bound for the difference term. We need to use the properties of the specifically defined GBDT algorithm. The key lemma we use is shown below.

Lemma C.2. *Let the step size satisies $\epsilon, 0 < \epsilon < 1$ and $\frac{1}{4M_\beta N} \geq \epsilon$. Then the following inequality holds:*

$$\mathbb{E}V\left(f_T\right) \leq \frac{R^2}{2N} e^{-\frac{T\epsilon}{2M_\beta N}}. \tag{72}$$

where $M_\beta = e^{\frac{mR^2}{N\beta}}$.

Proof. By theorem G.22 in Ustimenko et al. (2023), let $\lambda \to 0$, we get the desired result. □

By assumption 4.4 in the mian text, the dropout error $e^{(I)}$ is bounded. We denote the bound for $e^{(I)}$ as C_0, i.e., $\|\boldsymbol{e}^{(I)}\|_2 \leq C_0$, and C_0 is of order $O(\sqrt{N})$. We hide the dependence of N here, but it will be considered when we prove the error bound. Define f_* as the empirical minimizer, which is different to our target function $f_{0,-I}$. We further denote $\|f_*\|_{\mathbb{R}^N} = R$.

Corollary C.0.1. *Under assumption 4.1-4.4 in the main text, suppose the step size satisfies $\epsilon \leq \frac{1}{4M_\beta N}$, for all iterations $\tau = 1, 2, \ldots, \widehat{T}_{max}$, with probability at least $1 - c_1 \exp\left(-c_2 N \hat{r}_N^2\right)$, the following holds*

$$\left\|\mathbb{E}_u f_\tau - f_{0,-I}\right\|_N^2 \leq \frac{C}{\epsilon\tau} + \frac{4\epsilon^2}{N} C'^2 (M_\beta - 1)^2 \left(\frac{1 - e^{-\frac{\tau\epsilon}{2M_\beta N}}}{1 - e^{-\frac{\epsilon}{2M_\beta N}}}\right)^2 \tag{73}$$

where $C' = 2R + C_0$, and $\mathbb{E}_u$ is taking expectation w.r.t. the randomness of the SampleTree algorithm.

Further, if $\beta \geq N^{5/4}$, with probability at least $1 - c_1 \exp\left(-c_2 N \hat{r}_N^2\right)$ over warm-start initialization, the following holds

$$\left\|\mathbb{E}_u f_{\widehat{T}_{op}} - f_{0,-I}\right\|_N^2 \leq \mathcal{O}\left(\sqrt{\frac{2^d}{N}}\right), \tag{74}$$

where d is the tree depth and $\widehat{T}_{op}$ is obtained by optimizing the RHS of (73) within $[0, \widehat{T}_{max}]$.

Proof. Note that

$$\begin{aligned} \|\delta_\tau\|_2 &= \|\mathbb{E}[(K^{(I)} - K_\tau^{(I)}) f_\tau(\boldsymbol{X}^{(I)})] - \mathbb{E}[(K^{(I)} - K_\tau^{(I)})]\boldsymbol{e}^{(I)}\|_2 \\ &\leq \|\mathbb{E}[(K^{(I)} - K_\tau^{(I)}) f_\tau(\boldsymbol{X}^{(I)})]\|_2 + \|\mathbb{E}[(K^{(I)} - K_\tau^{(I)})]\boldsymbol{e}^{(I)}\|_2 \end{aligned} \tag{75}$$

By lemma A.2, we just need to show $g(\boldsymbol{X}^{(I)}, \tau)$ has the same form as the last term of the RHS of (74).

We leave the second term right now and start from the first term of the above equation.

For the first term, we have

$$\begin{aligned} \|\mathbb{E}[(K^{(I)} - K_\tau^{(I)}) f_\tau(\boldsymbol{X}^{(I)})]\|_2 &= \|\mathbb{E}K^{(I)}\mathbb{E}f_\tau(\boldsymbol{X}^{(I)}) - \mathbb{E}[\mathbb{E}K_\tau^{(I)} f_\tau(\boldsymbol{X}^{(I)})|f_\tau]\|_2 \\ &= \left\| \frac{1}{N} \sum_\nu k_\nu(\boldsymbol{X}^{(I)}, \boldsymbol{X}^{(I)}) \pi(\nu) \mathbb{E}\left[\left(1 - \frac{p(\nu|f_\tau, \beta)}{\pi(\nu)}\right) f_\tau(\boldsymbol{X}^{(I)})\right] \right\|_2 \\ &\leq \mathbb{E}\left[\max_{\nu \in \mathcal{V}} \left|1 - \frac{p(\nu|f_\tau, \beta)}{\pi(\nu)}\right| \left\| f_\tau(\boldsymbol{X}^{(I)}) \right\|_2\right] \qquad (\star) \end{aligned} \tag{76}$$

In the last inequality, we use the fact that the max eigenvalues of $\frac{1}{N} k_\nu\left(\boldsymbol{X}^{(I)}, \boldsymbol{X}^{(I)}\right)$ is 1. Then by lemma F.2 in Ustimenko et al. (2023), we can bound $\max_{\nu \in \mathcal{V}} \left|1 - \frac{p(\nu|f_\tau, \beta)}{\pi(\nu)}\right|$ as follows

$$\begin{aligned} \max_{\nu \in \mathcal{V}} \left|1 - \frac{p(\nu|f_\tau, \beta)}{\pi(\nu)}\right| &\leq \max\left\{1 - e^{-\frac{2mV(f_\tau)}{\beta}}, e^{\frac{2mV(f_\tau)}{\beta}} - 1\right\} \\ &\leq CV(f_\tau) \end{aligned} \tag{77}$$

the second inequality is because $V(f_\tau) \leq \frac{R^2}{2N}$ according to lemma G.20 in Ustimenko et al. (2023), we have $\|f_\tau - f_*\|_{\mathbb{R}^N} \leq \|f_*\|_{\mathbb{R}^N}$. We then can find $CV(f_\tau)$ to bound both the exponential terms. It is easy to see this from graphs. The constant C here can be

$$\frac{2N}{R^2}\left(e^{\frac{mR^2}{N\beta}} - 1\right) = \frac{2N}{R^2}\left(M_\beta - 1\right). \tag{78}$$

Then by Corollary G.21 in Ustimenko et al. (2023), we have $\|f_\tau\|_{\mathbb{R}^N} \leq 2\|f_*\|_{\mathbb{R}^N}$, so we can bound $(\star)$ as

$$\mathbb{E}\left[\max_{\nu\in\mathcal{V}}\left|1 - \frac{p(\nu|f_\tau,\beta)}{\pi(\nu)}\right|\left\|f_\tau(\boldsymbol{X}^{(I)})\right\|_2\right] \leq 2CR\mathbb{E}V(f_\tau) \leq 2R(M_\beta - 1)e^{-\frac{\tau\epsilon}{2M_\beta N}}. \tag{79}$$

For the second term in (75) involving $e^{(I)}$, by assumption 4.4, we can bound it similarly as:

$$\|\mathbb{E}[(K^{(I)} - K_\tau^{(I)})]e^{(I)}\|_2 \leq C_0(M_\beta - 1)e^{-\frac{\tau\epsilon}{2M_\beta N}}. \tag{80}$$

Combining all of this, we can bound the difference term D_τ as

$$D_\tau \leq \frac{2\epsilon}{\sqrt{N}}\sum_{i=0}^{\tau-1}(C_0 + 2R)(M_\beta - 1)e^{-\frac{i\epsilon}{2M_\beta N}} = \frac{2\epsilon}{\sqrt{N}}(C_0 + 2R)(M_\beta - 1)\frac{1 - e^{-\frac{\tau\epsilon}{2M_\beta N}}}{1 - e^{-\frac{\epsilon}{2M_\beta N}}}. \tag{81}$$

Let $C' = 2R + C_0$, and square the RHS of the above equation, we get the same difference term as stated in the corollary. Note that for the stopping threshold for GBDT, we need to use the spectrum of $\mathbb{E}K^{(I)}$ instead of $K^{(I)}$ as illustrated in the proof. For now, we finish the proof of the first part of this corollary.

For the second part,to satisfy assumption 4.5, we need to show that $\mathrm{tr}(\mathbb{E}K^{(I)}) \leq 2^d$ and the difference term is $\mathcal{O}(1/\sqrt{N})$. The trace is bounded as follows:

$$\sum_i \hat{\lambda}_i = \frac{1}{N}\sum_\nu \sum_{i=1}^{L_\nu}\frac{N}{N_\nu^{(i)}}N_\nu^{(i)}\pi(v) = \sum_\nu L_\nu\pi(v) \leq 2^d. \tag{82}$$

Next, we show how to bound D_τ^2. It suffices to show that in the limit of $N \to \infty$, the order of convergence for D_τ^2 is at least $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$.

Observe that

$$D_\tau^2 \leq \frac{4\epsilon^2}{N}C'^2(M_\beta - 1)^2\frac{1}{\left(1 - e^{\frac{\epsilon}{-2M_\beta N}}\right)^2} \tag{83}$$

When $N \to \infty$, $\frac{C'^2}{N}$ tends to a constant by LLN, and $M_\beta - 1 \sim \mathcal{O}(\frac{1}{\beta})$. Then apply $(1 - e^x)^2 \sim x^2$, when $x \to 0$, we have

$$\frac{4\epsilon^2}{N}C'^2(M_\beta - 1)^2\frac{1}{\left(1 - e^{-\frac{\epsilon}{2M_\beta N}}\right)^2} \sim \mathcal{O}\left(\frac{N^2}{\beta^2}\right). \tag{84}$$

To have at least order $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$, we need $\beta \geq N^{5/4}$. And this concludes our proof. □

Observe in the RHS of (73), the difference term would favor smaller iterations especially when the sample size is not too large. In the limiting setting, the first term might dominate the whole error term, we can simply set $\widehat{T}_{op} = \widehat{T}_{max}$ as they should be very close.

The condition that $\beta \geq N^{5/4}$ ensures that when the sample size is large, the kernel for each iteration would be really close to the stationary kernel. Because when $\beta \to \infty$, we also have $\pi(\cdot) = \lim_{\beta\to\infty} p(\cdot \mid f, \beta)$, i.e. the induced distribution of f will converge to the stationary distribution which leads to the stationary kernel.

C.6 Discussion

Unlike neural networks, Gradient Boosted Decision Trees (GBDT) do not have a universal approximation theorem. However, since decision trees partition the feature space into disjoint regions, we can imagine that with a tree of infinite depth, it would be possible to approximate any continuous function. Nonetheless, increasing tree depth would significantly raise computational costs and could lead to overfitting.

D IMPLEMENTATION AND EXPERIMENT DETAILS

D.1 Practical Guidance

As stated in Section 5 of the main text, the proposed optimal stopping rule is not feasible in practice. In addition to the method described in the main text, we could also use cross-validation. However, this would increase the computational cost of our algorithm without necessarily ensuring better fitting results. It is important to note that the primary computational expense comes from the gradient calculations.

An alternative approach is to perform a single update with a larger step size. Cross-validation could also be used to select an appropriate step size for this one-iteration update. In this scenario, the method becomes similar to kernel ridge regression. According to Raskutti et al. (2014), kernel ridge regression and early stopping are equivalent, with the running sum η_τ functioning similarly to the regularization parameter λ in kernel ridge regression. In the kernel ridge regression framework of lazy training, Gao et al. (2022) uses cross-validation to select an optimal λ. Hence, we can view early stopping as a comparable regularization technique.

D.2 Implementation

We implemented the neural networks and related experiments using PyTorch Lightning (Falcon and the PyTorch Lightning team, 2019). Our neural network structure follows the design described in Jacot et al. (2018), which includes a scaling factor of $\frac{1}{\sqrt{m}}$. For the GBDT experiments, we utilized the standard CatBoost library (Prokhorenkova et al., 2018). The experiments were conducted on a high-performance computing (HPC) system equipped with an Intel Xeon E5-2680 v3 @ 2.50GHz CPU and an NVIDIA RTX 2080Ti GPU. The running time for the neural networks was measured using the NVIDIA RTX 2080Ti, while the GBDT experiments were timed using the CPU.

D.3 Verifying Theoretical Bounds

The features for the neural networks are generated as $\boldsymbol{X} \sim \mathcal{N}(0, I_{3\times 3})$, while the features for the GBDT are drawn from a discrete uniform distribution, $X_i \sim \text{Uniform}(i-1, i+2)$. We use a discrete uniform distribution for GBDT because, if we were to use a continuous distribution, the CatBoost library would partition the feature space into too many bins. This would significantly increase the computational cost when calculating the exact empirical kernel matrix. By using a discrete uniform distribution, the number of bins is limited to a manageable size. And we use $\beta_1 = 3$ in the construction of $f_0 := f(\boldsymbol{X}^{(1)}) + \beta_1 X_1$ defined in Section 5.1 of the main text.

The reason why we use independent features in this simulation is because if X_1 is correlated with some other features, when we drop X_1, $f_{0,-1}$ would not simply be f as constructed in Section 5.1 in the main text, and it is complex to derive the true $f_{0,-1}$ in this setting.

As for the use of shallow networks and GBDT, in neural networks, the calculation of the Neural Tangent Kernel (NTK) is implemented recursively. For deeper networks, this significantly increases computational cost. In the case of GBDT, by the definition of the weak learner's kernel in equation (56), deeper tree structures would lead to an increased number of trees. To compute the empirical kernel matrix, we would need to perform calculations for each possible tree, making the implementation for a tree of depth 2 the simplest.

To compute the constant $C_{\mathcal{H}}$ in the stopping rule, we assume that $f_N^c, f_{0,-I} \in \text{span}\left\{\mathbb{K}^{(I)}(\cdot, \mathbf{X}_i^{(I)}), i = 1, \ldots, N\right\}$. When samples are large enough, this is fine. Note that this is simply an approximation. Then we can write

$$f_N^c(\cdot) - f_{0,-I}(\cdot) = \sum_{i=1}^{N} \alpha_i \mathbb{K}^{(I)}(\cdot, \mathbf{X}_i^{(I)}). \tag{85}$$

Plug into the data, we can solve $\boldsymbol{\alpha}$ as

$$\boldsymbol{\alpha} = \frac{1}{N} {K^{(I)}}^{-1} \left[f_N^c(\boldsymbol{X}^{(I)}) - f_{0,-I}(\boldsymbol{X}^{(I)}) \right] \tag{86}$$

where we use pesudo-inverse if $K^{(I)}$ is not ivnertible. By property of the RKHS, the Hilbert norm of $f_N^c(\cdot) - f_{0,-I}(\cdot)$ can be computed using the emprical kernel matrix as:

$$C_{\mathcal{H}}^2 = N \boldsymbol{\alpha}^T K^{(I)} \boldsymbol{\alpha} = \frac{1}{N} \boldsymbol{c}^T {K^{(I)}}^{-1} \boldsymbol{c} \tag{87}$$

where $\boldsymbol{c} := f_N^c(\boldsymbol{X}^{(I)}) - f_{0,-I}(\boldsymbol{X}^{(I)})$.

Even though we provide the optimal stopping time $\widehat{T}_{\text{op}}$ in the main text, the upper bound for the difference term $g(\tau)$ is still quite complex. As a result, computing the exact $\widehat{T}_{\text{op}}$ is infeasible. This is why we use $\widehat{T}_{\max}$ in this simulation. As shown in the proof, $\widehat{T}_{\max}$ also achieves the desired convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$. The advantage of $\widehat{T}_{\max}$ is that it can be calculated exactly using the straightforward formula:

$$\widehat{T}_{\max} := \arg\min \left\{ \tau \in \mathbb{N} \,\middle|\, \widehat{\mathcal{R}}_K \left(1/\sqrt{\eta_\tau}\right) > \frac{C_{\mathcal{H}}^2}{2e\sigma\eta_\tau} \right\} - 1. \tag{88}$$

In this simulation, we made several simplifications to ease the implementation and employed approximations at various steps. Even under these simplified conditions, and with knowledge of the true data-generating functions, computing the theoretical optimal stopping time $\widehat{T}_{\text{op}}$ remains challenging. The main goal of this simulation is to empirically validate our theoretical results and provide readers with a clearer understanding of the theoretical framework proposed in this paper.

D.4 Shapley Value Estimation

The Shapley value is defined via a value function *val* of features in I. The Shapley value of a feature value is its contribution to the payout, weighted and summed over all possible feature value combinations:

$$\phi_j(val) = \sum_{I \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|I|!(p-|I|-1)!}{p!} \left(\text{val}(I \cup \{j\}) - \text{val}(I)\right) \tag{89}$$

where I is a subset of the features used in the model (Molnar, 2022). For the value function, we use negative MSE in this experiment. By definition, we need to fit 2^p models to compute $\phi_j(val)$, which has a really high computation cost. So people normally use Monte-Carlo-based methods to reduce the cost. We use the sampling scheme proposed in Williamson and Feng (2020). Specifically, we sample each $I \subseteq \{1, \ldots, p\}$ with probability $\frac{|I|!(p-|I|-1)!}{p!}$.

We generate 50 samples to estimate each $\phi_j(\text{val})$, repeating this process 10 times. The running times for estimating all 10 Shapley values using both the neural network and GBDT models are listed in Table 1. Our proposed approach is approximately twice as fast as the retrain method in both models. The running time for the neural network is quite long, largely due to our unoptimized implementation and limited computing infrastructure. Parallelization could certainly be applied to estimate Shapley values, as the calculation for each value is independent, though we did not implement it here. The running times should be interpreted in the context of consistent implementation and infrastructure, our proposed method demonstrates significant computational savings compared to the retrain approach. Given that GBDT runs much faster than neural networks while providing similar estimation results, it is recommended to use GBDT in practice when advanced computing infrastructure and optimization techniques are not available.

D.5 Comparison with LazyVI

We compare our method with the LazyVI method (Gao et al., 2022) using the same high-dimensional regression simulation described in Section 5.3 of the main text. The results are shown in Figure 1. While the LazyVI method for neural networks achieves similar precision to our approach, it requires significantly more computation time. The primary issue with LazyVI is that it relies on k-fold cross-validation to select the ridge penalty parameter λ, which becomes highly time-consuming when the sample size is large. In our case, with $N = 5000$, the process takes an exceptionally long time to run.

Model	Method	Time (Mean ± Std)
Neural network	Early stopping	2.98 ± 0.24 hours
	Retrain	5.44 ± 0.37 hours
GBDT	Early stopping	56.94 ± 1.59 seconds
	Retrain	154.12 ± 4.27 seconds

Table 1: Shapley value etimation running time comparison.

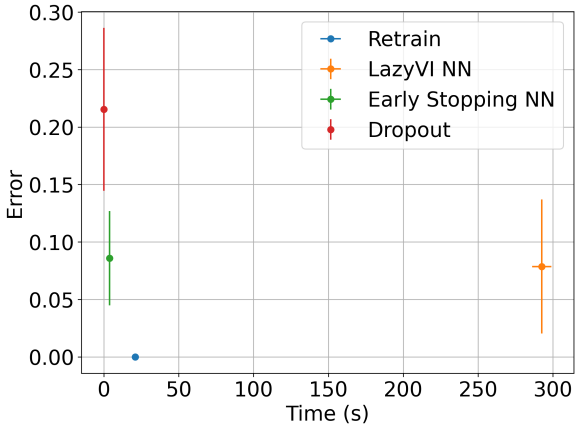


Figure 1: Distribution of computation time vs. normalized estimation error relative to retrain for the VI of X_1.

D.6 Predicting Flue Gas Emissions

The feature correlation heat map of the NOx emissions data, used in Section 5.4 of the main text, is depicted in Figure 2. The variable importance (VI) estimation results using GBDT are shown in Figure 3. Note that we use decision trees of depth 8 here to enable a better fitting of the data. It is evident that the features are highly correlated. The GBDT estimation results are similar to those from the neural network. For VI, defined using negative MSE, all values are relatively small. Regarding Shapley values, the neural network estimates tend to be higher than those from GBDT. Similar to the neural network, we observe that our early stopping approach closely matches the estimates obtained through re-training, while the dropout method tends to overestimate both VI and Shapley values.

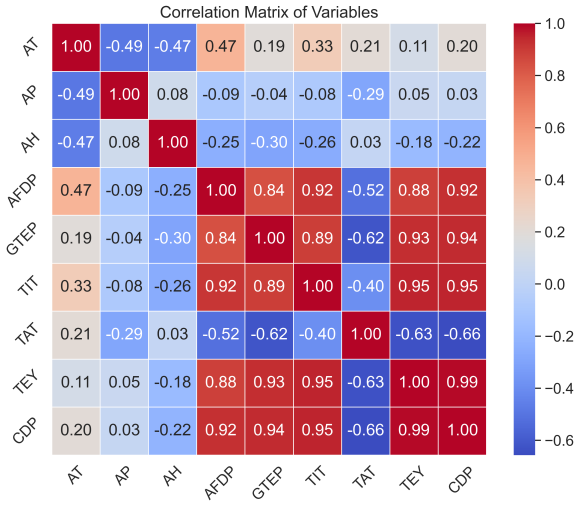
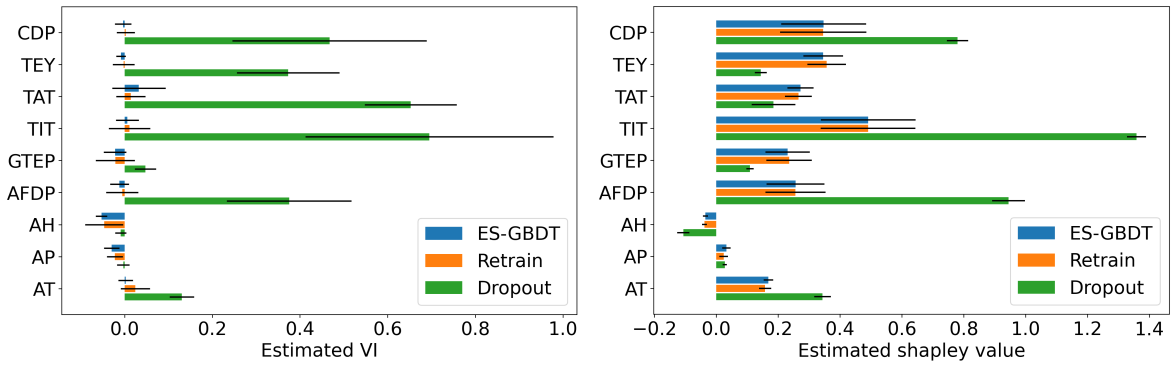


Figure 2: Correlation heat map.



(a) VI estimation

(b) Shapley value estimation

Figure 3: Flue gas emissions variable importance estimation results using GBDT.