Lazy Estimation of Variable Importance for Large Neural Networks

Yue Gao [*1] **Abby Stevens** [*2] **Garvesh Raskutti** [1] **Rebecca Willett** [2 3]

Abstract

As opaque predictive models increasingly impact many areas of modern life, interest in quantifying the importance of a given input variable for making a specific prediction has grown. Recently, there has been a proliferation of model-agnostic methods to measure variable importance (VI) that analyze the difference in predictive power between a full model trained on all variables and a reduced model that excludes the variable(s) of interest. A bottleneck common to these methods is the estimation of the reduced model for each variable (or subset of variables), which is an expensive process that often does not come with theoretical guarantees. In this work, we propose a fast and flexible method for approximating the reduced model with important inferential guarantees. We replace the need for fully retraining a wide neural network by a linearization initialized at the full model parameters. By adding a ridge-like penalty to make the problem convex, we prove that when the ridge penalty parameter is sufficiently large, our method estimates the variable importance measure with an error rate of $O(\frac{1}{\sqrt{n}})$ where n is the number of training samples. We also show that our estimator is asymptotically normal, enabling us to provide confidence bounds for the VI estimates. We demonstrate through simulations that our method is fast and accurate under several data-generating regimes, and we demonstrate its real-world applicability on a seasonal climate forecasting example.

1. Introduction

As predictive modeling becomes ubiquitous across a wide swath of application areas, it is especially critical to understand which variables contribute most to making a particular prediction. Black-box machine learning methods are insufficient in the face of algorithmic decision-making about things like sentencing, healthcare, and education, and working toward developing more interpretable methods is becoming more and more relevant (Rudin & Radin, 2019; Guidotti et al., 2018).

Traditional statistical tools based on parametric models (e.g. p-values, ANOVA) for VI inference are dissatisfying due to restrictive assumptions often violated in modern datasets. Non-parametric extensions thus have been explored (Doksum & Samarov, 1995). In recent decades, many VI methods designed for modern deep learning models have been investigated; most of these methods are gradient-based and depend on the structure and the weights of nodes in a given specific neural network (Shrikumar et al., 2019; Sundararajan et al., 2017; Smilkov et al., 2017; Bach et al., 2015). Few statistically rigorous properties are provided for these methods, and the VI definition is always intimately attached to the network itself, making it hard to interpret in a model-agnostic setting.

In a model-agnostic setting, a natural definition of VI that is independent of the estimation procedure is to measure the loss of predictive power when the variables of interest are deleted. To estimate such model-agnostic VI, *retraining* is the most widely used type of method, which involves training separate models on the reduced data with the variables of interest deleted and assessing the predictive skill difference (Williamson et al., 2021; Lei et al., 2018; Sapp et al., 2014). Retraining often acts as the best benchmark to evaluate other VI estimation methods (Hooker et al., 2019) due to its accuracy, yet it is computationally infeasible in high-dimensional settings. Other methods for VI estimation include knockoff methods (Barber & Candes, 2018; Candes et al., 2017) and Floodgate (Zhang & Janson, 2021), which require the co-variate distribution to be known. An alternative approach is to use a *dropout*-type method (Chang et al., 2017). Dropout is best-suited for assessing how much a variable affects a predictive model, as opposed to our goal of assessing how much a variable affects the response. Despite the resulting issues with VI estimation accuracy, it is still widely used in practice as a proxy for VI due to its computational tractability.

[*]Equal contribution [1]Department of Statistics, University of Wisconsin, Madison [2]Department of Statistics, University of Chicago [3]Department of Computer Science, University of Chicago. Correspondence to: Rebecca Willett <willett@uchicago.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

In this work, we propose a computationally efficient variable importance estimation procedure for model-agnostic and distribution-free settings with theoretical guarantees that leverages a *lazy retraining* framework inspired by (Chizat et al., 2020). The key idea is to train a new model on the transformed training data, akin to retraining, but on a linearized version of the model centered around model parameters learned from the original (unreduced) training data. We perform ridge regression on this linearized model in the gradient feature space, meaning that our lazy retraining procedure can be computed very quickly. The resulting method, when applied to wide neural network models, admits error bounds that show it is nearly as accurate as full retraining, while computationally it is nearly as fast as dropout. Our theoretical bounds are complemented by a collection of simulations that explore the limitations of dropout and benefits of lazy retraining under a variety of conditions and an application to understanding the importance of various climate indices in a seasonal forecasting task.

In summary, the main contribution of this paper is a **new, computationally efficient VI estimation method with statistical performance guarantees** in a model-agnostic and distribution-free setting when using large neural networks. Our theoretical analysis facilitates statistical inference, and we illustrate our approach on both synthetic and real-world data to support the theoretical claims and demonstrate the utility of our method. Other empirically-driven VI estimation methods exhibit similarities to our approach; our theoretical analyses may provide new insights into those methods as well as the one we propose in this paper.

2. Notation and preliminaries

Suppose we have samples $Z_i = (\mathbf{X}_i, Y_i), i = 1, \ldots, n$ for data $Z = (X, Y) \sim P_0$, where $\mathbf{X}_i \in \mathbb{R}^p$ is the i-th p-dimensional feature vector and Y_i is the i-th observed response. X denotes the multi-variate random variable containing features, Y denotes the response random variable. Let $X_{-j} \in \mathbb{R}^{p-1}$ (resp. $\mathbf{X}_{i,-j}$) denote the features in X (resp. $\mathbf{X}_i$) with the j-th variable removed; on the other hand, if we replace the j-th random variable in X (resp. $\mathbf{X}_i$) by its marginal mean $\mu_j = \mathbb{E}(X_j)$, we denote it as $X^{(j)}$ (resp. $\mathbf{X}_i^{(j)}$), *i.e.,* $X^{(j)} = (X_1, \ldots, X_{j-1}, \mu_j, X_{j+1}, \ldots, X_p)$. Let $P_0, P_{0,-j}$ be the population distributions for X and X_{-j} and let $P_n, P_{n,-j}$ be the empirical distributions of X and X_{-j} for $j \in [p]$. δ_{Z_i} denotes the point mass probability measure at the i-th observation Z_i. We denote $\mathbb{E}_0, \mathbb{E}_{0,-j}$ as the expectations taken with respect to P_0 and $P_{0,-j}$.

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

$$f_0(X) := \mathbb{E}_0[Y|X]; \tag{1}$$

$$f_{0,-j}(X^{(j)}) := \mathbb{E}_{0,-j}[Y|X_{-j}]. \tag{2}$$

Let f_n be the empirical model trained using all p variables in X within a certain function class $\mathcal{F}$(we refer to this as the *full model*):

$$f_n \in \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} [Y_i - f(\mathbf{X}_i)]^2. \tag{3}$$

To measure the accuracy of an approximation $f_n(x)$ to its target function f_0, we use the $L_2(\mu)$-norm

$$\|f_n - f\|^2 = \int |f_n(x) - f(x)|^2 d\mu(x), \tag{4}$$

where μ is the probability measure for X.

Further, we use ϵ and $\epsilon^{(j)}$ to denote the respective remainder terms for any $j \in [p]$:

$$\epsilon := Y - \mathbb{E}_0[Y|X]; \;\; \epsilon^{(j)} := Y - \mathbb{E}_{0,-j}[Y|X_{-j}]. \tag{5}$$

We will define our measure of variable importance (VI) in terms of a *predictive skill measure* $V(f, P)$ (the same measure in (Williamson et al., 2021)). Larger values of $V(f, P)$ should indicate better predictive performance. For $Z = (X, Y)$, we denote $\dot{V}(f, P; \delta P)$ as the Gateaux derivative of $V(f, P)$ at P in the direction δP.

3. Estimating variable importance

The VI measure we consider, which makes no assumptions on the data generating mechanism, is

$$\text{VI}_j := V(f_0, P_0) - V(f_{0,-j}, P_{0,-j}). \tag{6}$$

VI_j quantifies the difference in predictive skill between the full model and the reduced model for any $j \in [p]$. Consider the following simple linear model example, where we take the negative MSE as the predictive skill measure.

Example 3.1. *Suppose $Y = \beta_1 X_1 + \beta_2 X_2 + \epsilon$, where $X_i \sim \mathcal{N}(0, \sigma^2), i = 1, 2$, $Cov(X_1, X_2) = \rho$, and ϵ is a $\mathcal{N}(0, \sigma_\epsilon^2)$ noise that is independent of the features. The variable importance of the first variable is*

$$\text{VI}_1 = \beta_1^2 \cdot Var(X_1|X_2) = \beta_1^2(1 - \rho^2)\sigma^2$$

due to the fact that $X_1|X_2 \sim \mathcal{N}(\rho X_2, (1 - \rho^2)\sigma^2)$ (see Appendix B.6)

In general, we see from this example that the variable importance measure is determined not only by the relationship

between X_j and Y, but also the covariance structure in the features.

Our goal is to estimate VI_j for any variable X_j from data $\{(\mathbf{X}_i, Y_i)\}_{i=1}^n$ with no assumptions on the relationship between X and Y. For empirical estimators f_n and $f_{n,-j}$ of f_0 and $f_{0,-j}$, a plug-in estimator of our VI measure is

$$\widehat{\text{VI}}_j = V(f_n, P_n) - V(f_{n,-j}, P_{n,-j}). \tag{7}$$

The key problem we are concerned with in this paper is how to estimate $f_{n,-j}$ in an accurate and computationally efficient way. Traditionally, people use the following two types of methods to do the estimation: *dropout* and *retraining*.

3.1. Dropout

The method we are calling *dropout* estimates $\mathbb{E}(Y|X_{-j})$ by plugging the dropout features $X^{(j)}$ into the full model f_n. In this case, the variable importance measure can be estimated by

$$\widehat{\text{VI}}_j^{(\text{DR})} = V(f_n, P_n) - V(f_n, P_{n,-j}). \tag{8}$$

For the negative MSE measure of predictive skill for instance, the dropout estimate measures the difference between the squared error on the *original* training set and the squared error on the training set *after replacing feature j with its mean*. Dropout is superior among all plug-in estimators in terms of computational cost – we only need to train the model once to get f_n. This is desirable, especially when the function class $\mathcal{F}$ is large and complicated, such as with neural networks, and the computational cost for training the model is high. Despite this benefit, dropout is unreliable in many settings, as we will revisit in Section 3.3.

3.2. Retrain

An alternative to dropout is what we call *retraining*. Given a function class $\mathcal{F}$, the retraining method estimates VI_j by training separate models

$$f_{n,-j} \in \underset{f\in\mathcal{F}}{\arg\min}[Y_i - f(\mathbf{X}_i^{(j)})]^2 \tag{9}$$

for each variable $j \in [p]$ to estimate $f_{0,-j}$. Hence, VI under this framework is estimated via

$$\widehat{\text{VI}}_j^{(\text{RT})} = V(f_n, P_n) - V(f_{n,-j}, P_{n,-j}). \tag{10}$$

When taking negative MSE as the predictive skill measure, the retraining estimate in this case measures the difference between the squared error of a model trained *without* feature j and the squared error of a model trained *with* feature j. Retraining is more accurate than dropout as long as the function class $\mathcal{F}$ is large enough, but requires training $p+1$ models, which can be prohibitively computationally expensive in many settings. In this paper, we are especially interested in the setting when the function class is as large as a wide neural network.

3.3. Dropout vs. Retrain for Linear Models

The dropout method is widely used to estimate variable importance due to its efficiency. However, in cases where variables in X are highly correlated, dropout behaves problematically. Below, we will illustrate and quantify the difference of the variable importance estimation in the random design linear model case, where we take the negative MSE as the $V(f, P)$ measure as in (**??**). For simplicity, we restrict the function space $\mathcal{F}$ to the linear function space here.

Suppose $X \in \mathbb{R}^p \sim \mathcal{N}(0, \Sigma)$, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. Assume Σ is positive definite. Let $\beta^* := \arg\min_{w\in\mathbb{R}^p} \mathbb{E}[Y - X^\top w]^2$, so $\beta^* = \Sigma^{-1}\mathbb{E}(XY)$. In the population version, the dropout method uses the predictor $X_{-j}^\top \beta_{-j}^*$ (where $\beta_{-j}^* \in \mathbb{R}^{p-1}$ is β^* with its j-th element removed) to estimate $\mathbb{E}(Y|X_{-j})$, while the retraining method uses the predictor $X_{-j}^\top \beta^{(j)}$, where $\beta^{(j)} \in \mathbb{R}^{p-1}$ is $\beta^{(j)} = \arg\min_{w\in\mathbb{R}^{p-1}} \mathbb{E}[Y - X_{-j}^\top w]^2$. The following proposition characterizes the difference between VI estimates corresponding to the retraining and dropout methods.

Proposition 3.2. *In the linear function space, the difference between the variable importance estimates for variable j from the population version of the dropout and retraining methods is:*

$$\begin{aligned}&\widehat{\text{VI}}_j^{(\text{DR})} - \widehat{\text{VI}}_j^{(\text{RT})} \\ =&\frac{\vec{\gamma_j}^\top \Sigma_{(j)}^{-1} \vec{\gamma_j}}{(\Sigma_{jj} - \vec{\gamma_j}^\top \Sigma_{(j)}^{-1}\vec{\gamma_j})^2}\left[\mathbb{E}(X_j Y) - \vec{\gamma_j}^\top \Sigma_{(j)}^{-1}\mathbb{E}(X_{-j}Y)\right]^2,\end{aligned}$$

where $\vec{\gamma_j} = \mathbb{E}(X_j X_{-j}) \in \mathbb{R}^{p-1}$.

If the true model between Y and X is linear, *i.e.,* $Y = X^\top\beta^* + \epsilon$, and $X \perp\!\!\!\perp \epsilon$, the variable importance estimated by retraining linear regression is:

$$\widehat{\text{VI}}_j^{(\text{RT})} = \beta_j^{*2}(\Sigma_{jj} - \vec{\gamma_j}^\top \Sigma_{(j)}^{-1}\vec{\gamma_j}); \tag{11}$$

furthermore, in this setting $\widehat{\text{VI}}_j^{(\text{RT})}$ is exactly the true variable importance defined in (6). In contrast, the dropout framework will give

$$\widehat{\text{VI}}_j^{(\text{DR})} = \beta_j^{*2} \cdot \Sigma_{jj}. \tag{12}$$

If feature j is important and highly correlated with feature k (but independent of all other features), then $\vec{\gamma_j}^\top \Sigma_{(j)}^{-1}\vec{\gamma_j}$ may be very large, making the difference between $\widehat{\text{VI}}_j^{(\text{DR})}$ and $\widehat{\text{VI}}_j^{(\text{RT})}$ similarly large. This example illustrates how dropout can significantly overestimate variable importance, even in simple settings.

4. Lazy Training

Our central interest is in inferring VI using complex models that are time-consuming to train, making the baseline retraining method described above computationally infeasible.

With this in mind, we turn our attention to neural network (NN) models, a setting in which dropout is widely used.

Motivated by the need for faster and more accurate methods for estimating VI with NN, we propose a computationally efficient VI estimate inspired by the lazy training framework of (Chizat et al., 2020) that estimates the difference between the full model parameters and the model parameters when the j-th variable is removed. Like dropout, our procedure only requires us to train the NN once on the full data, and then we solve a linear system to update the full model parameters for each variable $j \in [p]$.

Given the training data $\{(\mathbf{X}_i^{(j)}, Y_i)\}$ sampled from $(X, Y) \sim P_0$ for $i = 1, \ldots, n$ and the underlying function $f_0(X) = \mathbb{E}_{P_0}[Y|X]$, there exists a a neural network function class $\{h_\theta(x) : \mathbb{R}^p \mapsto \mathbb{R} | \theta \in \mathbb{R}^M\}$ that is parameterized by a vector θ, such that when we train the model parameters over this class by

$$\theta_f = \underset{\theta \in \mathbb{R}^M}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} [Y_i - h_\theta(\mathbf{X}_i)]^2, \tag{13}$$

the estimation error can be bounded by $\|h_{\theta_f}(x) - f_0(x)\| = O(n^{-1/2})$ up to some log terms (Barron, 1994). To achieve this, the scale of the number of parameters M depends on the complexity of the target function. For very complex functions, we can still achieve this accuracy with $M = O(\sqrt{n})$.

In order to estimate VI_j, we need an estimate of what we are calling the *reduced* model $h_{\theta_{-j}}$, where

$$\theta_{-j} = \underset{\theta \in \mathbb{R}^M}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} [Y_i - h_\theta(\mathbf{X}_i^{(j)})]^2. \tag{14}$$

Instead of retraining a NN to estimate θ_{-j}, we can instead estimate the difference between the full model parameters θ_f and θ_{-j} using this linear approximation, and simply update the full model parameters with this correction to estimate $h_{\theta_{-j}}$. We are essentially regressing the error resulting from the dropout estimation against the gradient to estimate this correction, and to do so we solve the following convex problem based on the training data $\{(\mathbf{X}_i^{(j)}, Y_i)\}$ for $i = 1, \ldots, n$ and a 2-norm penalty on the parameters:

$$\Delta\theta_j(\lambda, n) = \underset{\omega \in \mathbb{R}^M}{\arg\min} \Big\{ \frac{1}{n} \sum_{i=1}^{n} \big[Y_i - h_{\theta_f}(\mathbf{X}_i^{(j)}) - \omega^\top \nabla_\theta h_\theta(\mathbf{X}_i^{(j)})|_{\theta=\theta_f} \big]^2 + \lambda \|\omega\|_2^2 \Big\}, \tag{15}$$

where $\lambda > 0$ is the penalty parameter.

Accordingly, the reduced neural network parameters are $\Delta\theta_j(\lambda, n) + \theta_f$. For the simplicity of notation, we write $\Delta\theta_j(\lambda, n)$ as $\Delta\theta_j$ for short. Then the reduced model approximation without the j-th feature is $\mathbb{R}^p \mapsto \mathbb{R} : x \mapsto h_{\theta_f + \Delta\theta_j}(x)$.

Hence, the variable importance measure under lazy training is

$$\widehat{\text{VI}}_j^{(\text{LAZY})} = V(h_{\theta_f}, P_n) - V(h_{\theta_f + \Delta\theta_j}, P_{n,-j}). \tag{16}$$

Under the negative MSE measure $V(f, P)$, we have

$$\widehat{\text{VI}}_j^{(\text{LAZY})} = \frac{1}{n} \sum_{i=1}^{n} \{ [Y_i - h_{\theta_f + \Delta\theta_j}(\mathbf{X}_i^{(j)})]^2 - [Y_i - h_{\theta_f}(\mathbf{X}_i)]^2 \}.$$

(More precisely, we use data splitting for training and estimating VI as detailed in Algorithm 1.) Essentially, the linearized approximation of the NN is linear in the gradient feature map $x \mapsto \nabla_\theta h_\theta(x)|_{\theta_f}$. In fact, this gradient feature map induces the Neural Tangent Kernel (NTK, (Jacot et al., 2020)): for any $x, x' \in \mathbb{R}^p$,

$$\ker_{\theta_f}(x, x') := \langle \nabla_\theta h_\theta(x)|_{\theta_f}, \nabla_\theta h_\theta(x')|_{\theta_f} \rangle. \tag{17}$$

Thus $\Delta\theta_j$ can be viewed as the solution for a kernel ridge regression problem with kernel $\ker_{\theta_f}$.

4.1. Theoretical Guarantee

By (Williamson et al., 2021), when the empirical estimates for $\mathbb{E}(Y|\mathbf{X})$ and $\mathbb{E}(Y|\mathbf{X}_{-j})$ converge to the target functions f_0 and $f_{0,-j}$ at the rate of $O_p(n^{-1/4})$ in function norm, we achieve an asymptotically normal and efficient estimator for the VI measure. In this section, we give a theoretical guarantee to show that the lazy prediction $h_{\theta_f + \Delta\theta_j}(X^{(j)})$ for the reduced model achieves such convergence rate, so that the lazy training procedure gives an accurate estimate of VI with an error in the order of $O(\frac{1}{\sqrt{n}})$ and we can make inference accordingly.

Let $\mathbf{e}^{(j)}$ denote the difference between the true reduced function $f_{0,-j}(X^{(j)})$ and the corresponding dropout estimation:

$$\mathbf{e}^{(j)} := f_{0,-j}(\mathbf{X}^{(j)}) - h_{\theta_f}(\mathbf{X}^{(j)}) \in \mathbb{R}^n. \tag{18}$$

Further, we denote the kernel matrix on $X^{(j)}$ induced by the gradient feature map as $\mathbb{K}^{(j)} \in \mathbb{R}^{n \times n}$, whose elements are defined as:

$$\mathbb{K}_{ik}^{(j)} := \ker_{\theta_f}(\mathbf{X}_i^{(j)}, \mathbf{X}_k^{(j)}),\ i, k \in [n]. \tag{19}$$

Before diving into the main results, we first clarify two types of notation for order of approximation:

- $f(n) = O(g(n))$ if there exists $M > 0$ and $N > 0$, such that $|f(n)| \leq M g(n)$ for all $n > N$.
- $X_n = O_p(a_n)$ as $n \to \infty$ if for any $\epsilon > 0$, there exists $M > 0$ and $N > 0$, such that $\mathbb{P}(|\frac{X_n}{a_n}| > M) < \epsilon$ for any $n > N$.

Assumption 4.1. For any $j \in [p]$ and the regularization parameter $\lambda = O(\sqrt{n})$, we assume:

(a) $\|[\mathbb{K}^{(j)} + \lambda I_n]^{-1}\mathbf{e}^{(j)}\|^2 = O_p(1/\sqrt{n})$;

(b) $\text{tr}(\mathbb{K}^{(j)}) = O_p(n)$.

The above assumption (b) is commonly used in NTK literature (see e.g. Hu et al. (2019)). For a two-layer neural network, we can verify this numerically (see Appendix C.1). For the assumption (a), by the fact that $\mathbb{K}^{(j)}$ is positive semi-definite, this assumption can be satisfied when we have a large regularization $\lambda = O(\sqrt{n})$.

Assumption 4.2. For the noise term $\epsilon^{(j)}$, we have the following assumption on its conditional tail probability: there exists σ such that for any $j \in [p]$,

$$\mathbb{E}\left[e^{\lambda\epsilon^{(j)}}|X^{(j)}\right] \leq e^{\sigma^2\lambda^2/2}, \text{ for all } \lambda \in \mathbb{R}. \tag{20}$$

Assumption 4.3. Denote the gradient feature matrix as $\Phi \in \mathbb{R}^{n\times M} = (\nabla_\theta h_\theta(\mathbf{X}_1)|_{\theta=\theta_f}, \ldots, \nabla_\theta h_\theta(\mathbf{X}_n)|_{\theta=\theta_f})^\top$. We assume $\|\Phi^\top \mathbf{e}^{(j)}\|_2 \leq O_p(1)$.

This assumption essentially requires that the linear space of neural tangent kernels can well represent $\mathbf{e}^{(j)}$. We know that $\mathbf{e}_i^{(j)}$ is a function of $\mathbf{X}_i^{(i)}$, thus as long as the neural network function class is large enough, this can be satisfied with respect to the sample size n.

Theorem 4.4. *Suppose Assumption 4.1, 4.2 and 4.3 hold, then for a neural network structure $h_\theta(\cdot)$ which is L-smooth with respect to its parameters θ, as long as we take the ridge penalty parameter in the order $\lambda = O(n^{1/2})$, then the lazy training method can accurately predict the reduced model without the j-th covariate,* i.e.,

$$\|h_{\theta_f+\Delta\theta_j}(x) - \mathbb{E}(Y|X^{(j)})\|_2 = O_p(n^{-1/4}). \tag{21}$$

Therefore our variable importance estimator $\widehat{\text{VI}}_j^{(\text{LAZY})}$ is asymptotically normal and has an error rate $O_p(n^{-1/2})$:

$$\widehat{\text{VI}}_j^{(\text{LAZY})} - \text{VI}_j = \Delta_{n,j} + O_p(n^{-1/2}); \tag{22}$$

where

$$\Delta_{n,j} = \frac{1}{n}\sum_{i=1}^{n}\big[\dot{V}(f_0, P_0; \delta_{Z_i} - P_0) - \dot{V}(f_{0,-j}, P_{0,-j}; \delta_{Z_i} - P_{0,-j})\big] \to_d \mathcal{N}(0, \tau_{n,j}^2); \tag{23}$$

here the variance is $\tau_{n,j}^2 = Var({\epsilon^{(j)}}^2 - \epsilon^2)/n$, where ϵ and $\epsilon^{(j)}$ is defined in Equation (5).

This result enables us to construct Wald-type confidence intervals around our LazyVI estimates. In particular, the $\alpha-$level confidence intervals are given by

$$\widehat{\text{VI}}_j^{(\text{LAZY})} \pm z_{\frac{\alpha}{2}}\hat{\tau}_{n,j} \tag{24}$$

where $\hat{\tau}_{n,j}$ is the plug-in estimate of $\tau_{n,j}$ in (23) and $z_{\frac{\alpha}{2}}$ is the $\alpha/2$ quantile of the standard normal distribution.

4.2. Proof Overview

The challenge of proving Theorem 4.4 is to bound the error of the lazy neural network trained using data without a certain variable – note that we are bounding the estimation error ($\|h_{\theta_f+\Delta\theta_j} - f_{0,-j}\|$) instead of the prediction error ($\|h_{\theta_f+\Delta\theta_j} - f_0\|$) that is the focus of much of the deep learning community, since the predictive skill of the reduced model is expected to decrease when an important variable is removed. At a high level, our proof reduces the estimation error of the neural network from lazy training to the error between the NTK estimation and the target function, where we use techniques from kernel ridge regression. The difference here is that most NTK papers (see e.g. (Jacot et al., 2020)) use random initialization for the parameters and optimization without penalty, while our method starts from a specific initialization (the full model), and requires the penalty parameter λ to be large ($\lambda = O(n^{1/2})$) to ensure convergence.

The following two lemmas give some intuition on how the neural network trained by the lazy procedure can accurately estimate the reduced model. Basically, the bound for the error consists of two parts: the error from the kernel ridge regression (discussed in Lemma 4.5), and the error from the linear approximation of the neural network (in Lemma 4.6). More proof details are deferred to the Appendix.

Denote the linear approximation of the network as

$$\tilde{h}_{\theta_f+\Delta\theta_j}(x) := h_{\theta_f}(x) + \langle\nabla_\theta h_\theta(x)|_{\theta=\theta_f}, \Delta\theta_j\rangle. \tag{25}$$

Lemma 4.5. *Let λ be penalty parameter in Equation (15), we have with probability at least $1-\delta$,*

$$\|\tilde{h}_{\theta_f+\Delta\theta_j}(X^{(j)}) - f_{0,-j}(X^{(j)})\|_n \leq \frac{\lambda\|[\mathbb{K}^{(j)} + \lambda I_n]^{-1}\mathbf{e}^{(j)}\|}{\sqrt{n}} + \sigma\sqrt{\frac{tr[\mathbb{K}^{(j)}]}{4\lambda n}} + \sigma\sqrt{\frac{2\log(1/\delta)}{n}}. \tag{26}$$

Lemma 4.5 combined with Assumption 4.1 when the penalty parameter is $\lambda = O(n^{1/2})$, yields a bound on the empirical error of the kernel ridge regression component of $O_p(n^{-1/4})$. Based on this empirical bound, we could then further bound the generalization error of the estimated function using function complexity (See Appendix B.3).

Lemma 4.6. *For a large neural network with width in the order $O(\sqrt{n})$, with high probability we have for all $j \in [p]$,*

$$\|\tilde{h}_{\theta_f+\Delta\theta_j}(x) - h_{\theta_f+\Delta\theta_j}(x)\| \leq O(n^{-1/4}). \tag{27}$$

Lemma 4.6 shows that as long as the neural network is sufficiently large, the neural network with updated parameters $\theta_f + \Delta_j$ is close to its linear approximation.

Algorithm 1 Lazy training for VI

Require: Data: $\{\mathbf{X}_i, Y_i\}_{i=1}^n$; $\lambda > 0$; training size: $0 < n_1 < n$; $n_2 \leftarrow n - n_1$; NN structure: $\theta \in \mathbb{R}^M \mapsto h_\theta(\cdot)$
LazyVI$\{\mathbf{X}_i, Y_i\}_{i=1}^n$; λ, n_1
$\theta_f \leftarrow \arg\min_{\theta \in \mathbb{R}^M} \frac{1}{n_1} \sum_{i=1}^{n_1} [Y_i - h_\theta(\mathbf{X}_i)]^2$
$v_n \leftarrow -\frac{1}{n_2} \sum_{i=n_1+1}^{n} [Y_i - h_{\theta_f}(\mathbf{X}_i)]^2$
for $j \in [p]$ **do**
  $\mathbf{X}_i^{(j)} \leftarrow \mathbf{X}_i$; $\mathbf{X}_{ij}^{(j)} \leftarrow \frac{1}{n_1} \sum_{i=1}^{n_1} \mathbf{X}_{ij}$
  $\mathbf{e}_i^{(j)} \leftarrow Y_i - h_{\theta_f}(\mathbf{X}_i^{(j)})$, $i = 1, \ldots, n_1$
  $\Phi_i^{(j)} \leftarrow \nabla_\theta h_\theta(\mathbf{X}_i^{(j)})|_{\theta=\theta_f}$, $i = 1, \ldots, n_1$
  $\Delta\theta_j \leftarrow \arg\min_{\omega \in \mathbb{R}^M} \frac{1}{n_1} \sum_{i=1}^{n_1} [\mathbf{e}_i^{(j)} - \omega^\top \Phi_i^{(j)}]^2 + \lambda \|\omega\|_2^2$
  $v_{n,-j} \leftarrow -\frac{1}{n_2} \sum_{i=n_1+1}^{n} [Y_i - h_{\theta_f + \Delta\theta_j}(\mathbf{X}_i^{(j)})]^2$
  $\widehat{\text{VI}}_j \leftarrow v_n - v_{n,-j}$
  $t_{i,j} \leftarrow (Y_i - h_{\theta_f + \Delta\theta_j}(\mathbf{X}_i^{(j)}))^2 - (Y_i - h_{\theta_f}(\mathbf{X}_i))^2$
  $\hat{\tau}_j \leftarrow \frac{1}{n_2} \sum_{i=1}^{n_2} (t_{i,j} - \bar{t}_j)^2 / n_2$
end for
Ensure: $\widehat{\text{VI}}_j$, $j = 1, \ldots, p$.

4.3. Implementation

We estimate h_{θ_f} and $h_{\theta_{-j}}$ using $n_1 < n$ samples as training data, and use the remaining $n_2 = n - n_1$ samples to estimate VI. For the dropout method, VI is estimated simply by plugging the modified testing data $\{\mathbf{X}_i^{(j)}\}_{i=n_1+1}^n$ into h_{θ_f}. For the retraining method, first $h_{\theta_{-j}}$ is estimated by retraining the NN h with $\{\mathbf{X}_i^{(j)}\}_{i=1}^{n_1}$, and then VI is estimated by plugging the modified testing data into this retrained estimate.

For the lazy training method, which we call LazyVI, we use the training data to estimate the full model parameters, compute the gradient of the network with respect to each model parameter for each training sample, and then regress these gradients against the difference between Y the dropout estimates from the training data to estimate the parameter correction $\Delta\theta_j$ for variable j. We then update the full model parameters using this learned correction to compute the VI estimate and its associated standard errors. See Algorithm 1 for full details.

Theorem 4.4 makes the assumption that the ridge parameter λ from Equation (15) is large. Since we are ultimately interested in estimating $h_{\theta_{-j}}$ and not $\Delta\theta_j$, we evaluate $h_{\theta_f + \Delta\theta_j}(\cdot)$ through K-fold CV to choose $\hat{\lambda}_j$ for each variable (Algorithm 2 in Appendix C.2). Our implementation is available at `https://github.com/Willett-Group/lazyvi`.

5. Simulations

We first assess the performance of LazyVI on simulated data to highlight key theoretical claims and assumptions and show that our method is empirically practical. For these experiments, we train a wide, fully connected two-layer neural network with ReLU activation for all simulations. Unless otherwise specified, the width of the hidden layer in the training network is $m = 50$.

5.1. Impact of correlation in linear systems

Our first set of simulations serve to support key details of our theoretical analysis. We consider data generated from the linear model $f(X) = 1.5X_1 + 1.2X_2 + X_3 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.1)$ and $X \sim \mathcal{N}(0, \Sigma_{6\times 6})$, so the response only depends on the first three of the six variables. All variables are independent except for X_1 and X_2, whose correlation is ρ. As discussed in Example 3.1, the true VI of X_1, X_2, and X_3 are given by $(1.5)^2(1-\rho^2)$, $(1.2)^2(1-\rho^2)$, and 1, respectively, and the VI of the remaining 3 variables is zero. In this simple setting, we find that LazyVI approximates the true VI well with desirable coverage and a considerable speed-up relative to retraining (Appendix C.3).

We show in Prop. 3.2 that, when data are generated from a linear model, the difference between the dropout and retraining variable importance estimates is a function of the covariance of X. After training the full model, we use both the dropout and our lazy procedure to estimate VI for increasing values of ρ. In Figure 1, we show the difference between the dropout and LazyVI estimates for variables X_1 and X_2 alongside the analytic difference between $\widehat{\text{VI}}^{(\text{DR})}$ and VI (dotted line). We see that the gap between LazyVI and dropout evolves with ρ according to the theoretical analysis, providing evidence that LazyVI behaves as expected.

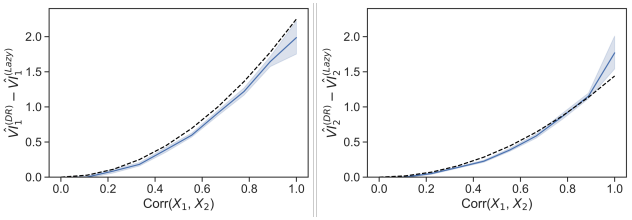

Figure 1. Difference between the dropout and LazyVI estimates for X_1 and X_2. Dotted line is theoretical gap and shading shows std. across 10 repetitions.

We use this simple linear setting to explore two additional assumptions from our theoretical resuts. First, the linearization in (25) is a first order Taylor approximation and assumes the full model parameters are close to the reduced model parameters. If we try to linearize a neural network around a random initialization, our LazyVI estimates are much less accurate and more highly variable (Appendix C.4).

Next, our theory assumes that our training network is over-parameterized and sufficiently wide. We compute empirical confidence intervals for LazyVI for increasing network widths and find that coverage increases as the width increases, but at a computational cost (Appendix C.5).

5.2. Binary classification

Because we borrow much of our theoretical framework from (Williamson et al., 2021), we also leverage their simulation framework as a useful point of comparison. We draw independent samples $X \sim \mathcal{N}(0, I_{4\times 4})$ and generate a binary outcome $Y \sim \text{Bernoulli}(\Phi(X\beta))$ where $\beta = (2.5, 3.5, 0, 0)$. Because the outcome is binary, we use accuracy as our predictive skill measure, and the true VI values are given by (0.136, 0.236, 0, 0), respectively. We first directly compare the LazyVI and retrain estimators by estimating VI across 100 simulated datasets of sample size $n = 1000$ and computing the empirical 95% confidence intervals. In Figure 2, we see that the LazyVI and retrain estimates both achieve the desired level of coverage with low bias. In this simulation, LazyVI took on average 0.6 seconds (including cross-validating to find the optimal ridge parameter), while retraining took 7.5 seconds. In this setting, LazyVI is just as accurate as retraining with a more than 10x speed-up.

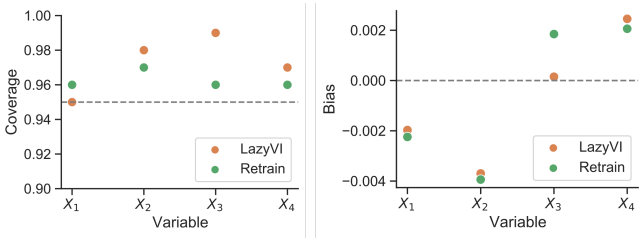

Figure 2. Left: Average coverage of empirical 95% confidence intervals from the LazyVI and retrain estimates across 100 simulations. Right: Average empirical bias (VI − $\widehat{\text{VI}}$) of LazyVI and retrain estimates.

5.3. Nonlinear, high-dimensional regression

The computational burden of retraining is most pronounced in high-dimensional settings, since estimating $\widehat{\text{VI}}^{(\text{RT})}$ for all variables requires refitting at least p models. For this simulation, we have data $X \sim N(0, \Sigma_{100\times 100})$, where variables are independent except $\text{Corr}(X_1, X_2) = 0.5$. Letting $\beta = (5, 4, 3, 2, 1, 0, \ldots, 0)^\top \in \mathbb{R}^{100}$, we construct a weight matrix $W \in \mathbb{R}^{m\times p}$ such that the $W_{:,j} \sim \mathcal{N}(\beta_j, \sigma^2)$ (i.e. the weights associated with variable j are centered at β_j). Letting $V \sim \mathcal{N}(0, 1)$, we generate the response $Y_i = V\sigma(W\mathbf{X}_i) + \epsilon_i$ where σ is the ReLU function. Because the "true" VI values are unknown and difficult to estimate, we present the accuracy of different estimation methods relative to the retraining estimates, which we take as ground truth. We estimate VI for X_1 across 10 simulated datasets ($n = 1000$) and benchmark against retraining using both a linear regression (OLS) and random forest (RF).In Figure 3, we show the spread of both the computation time and normalized error (relative to retrain) for all methods. We see that LazyVI is the most accurate method and is substantially faster than retraining, which is especially beneficial in this high-dimensional setting.

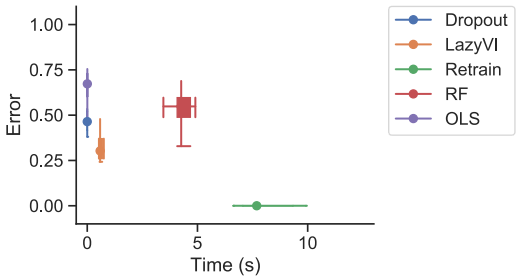

Figure 3. Distribution of computation time vs. normalized estimation error relative to retrain for the VI of X_1 $((\hat{\text{VI}} - \widehat{\text{VI}}^{(\text{RT})})/\widehat{\text{VI}}^{(\text{RT})})$ across 10 repetitions.

6. Predicting seasonal precipitation

Extreme precipitation events have become more and more common in recent years, and are expected to intensify with climate change (Tabari, 2020; Li et al., 2019). Early and reliable precipitation forecasting is thus critical for regional water resource management, which increasingly impacts large swaths of the population (AghaKouchak et al., 2015). Many studies have shown that the sea surface temperature (SST) over various regions of the ocean, such as the El Niño-Southern Oscillation (ENSO), are predictive of precipitation in the United States (Mamalakis et al., 2018; Dai, 2013; Lenssen et al., 2020). Understanding which ocean regions are most predictive is challenging, however, due to a short observational record and strong correlations among SSTs (Stevens et al., 2021).

6.1. Importance of Ocean Climate Indices

We estimate the importance of different ocean regions for seasonal precipitation forecasting using our lazy training method. The response is the average winter precipitation over the Southwestern US, and as predictors we use 10 ocean climate indices (OCIs), which are defined as the average detrended SST anomalies over different ocean regions (Chen et al., 2016). As data, we use simulations from the Community Earth System Model-Large Ensemble project (CESM-LENS; Kay et al. (2015); de La Beaujardière et al. (2019)). Details about data processing can be found in Appendix C.6.

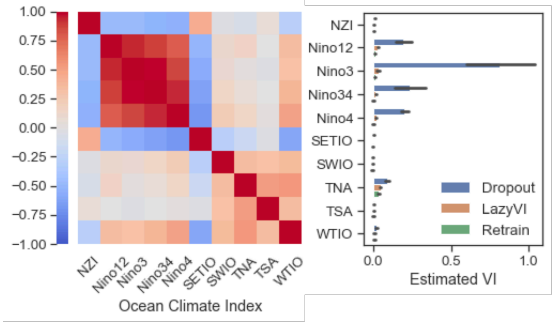

Figure 4. Left: sample covariance matrix of the OCIs across the 40 LENS ensemble members; Right: estimated VI for each OCI across 10 different train/test splits.

There are strong correlations among the various OCIs (Figure 4)) — in particular, the various Niño indices appear to be nearly collinear. Because of this, we would expect methods like linear regression to inaccurately estimate coefficients and their importance (see appendix for more discussion).

We apply LazyVI to this problem by first training a two-layer neural network with a hidden width of 50 and then removing each climate index and linearly estimating the correction. When comparing with the dropout and retraining VI estimates, we see that dropout drastically overestimates VI of Niño 3 and Niño 3.4 relative to retraining, and that LazyVI results in estimates much closer to the retraining estimates. These results are consistent with recent literature indicating that the predictive ability of Niño is often overstated relative to other OCIs (Mamalakis et al., 2018), suggesting that LazyVI could potentially help us better understand the relative importance of different climate mechanisms.

6.2. High-dimensional seasonal forecasting

Aggregating climate regions into OCIs is standard in the climate literature and a critical tool for understanding climate dynamics. However, while more difficult to interpret and estimate, disaggregating OCIs and investigating individual SST locations offers important insights into the rapidly changing climate system (Stevens et al., 2021). Neural networks have increasingly been used to make these types of high-dimensional forecasts, and with that comes an increased interest in explainability (Mamalakis et al., 2022b). However, standard gradient-based attribution/saliency methods used to interpret NNs, while powerful for particular networks, are often subjective and difficult to interpret (Mamalakis et al., 2022a).

The ROAR (RemOve and Retrain) framework introduced by (Hooker et al., 2019) offers a helpful way to evaluate such importance measures. This work provides a retraining-based benchmark for evaluating NN attribution/saliency methods by removing variables in order of estimated importance and measuring the drop in predictive power. This work finds that many common attribution methods are no more informative than a random baseline, and aruges that retraining the network after dropping out variables is key in understanding this behavior.

Using all summer SSTs across the Pacific basin on a $10^\circ \times 10^\circ$ grid, for a total of 220 predictors, we show that LazyVI can achieve similar results to retraining in the ROAR framework at a computational speed-up. We train fully-connected three-layer neural network of widths (100, 50) on all variables, and from this trained network, we estimate feature importance using the baseline Gradients importance method (GRAD, (Simonyan et al., 2013)). We then remove $t = (.1, .25, .5, .75, .9, .99)$ proportion of variables by removing them in order of GRAD importance (in addition to a random ordering as a baseline, see Figure 12 in Appendix C.6 for a visualization of this procedure). We estimate model performance on these modified datasets using the Dropout, Retraining (ROAR) and LazyVI approaches and find that LazyVI closely approximates ROAR in nearly half the time (Figure 5) and the ordering of variables removed does not matter much; Dropout, on the other hand, vastly overestimates the degradation of the model performance, even with a small number of variables removed.

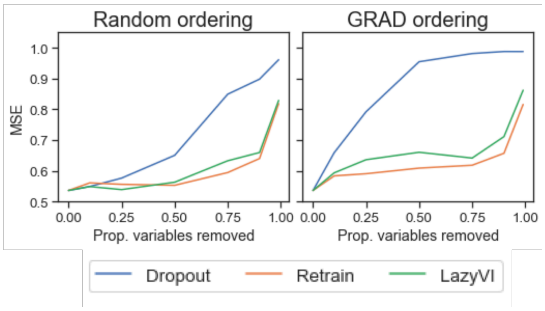

Figure 5. Average MSE across 5 runs after removing increasing proportions of variables with the specified importance orderings. On average, LazyVI took 1.8s and Retrain took 3.2s.

7. Discussion and extended applications

Assessing variable importance in machine learning is a vital task as learning-based tools are increasingly integrated into societally-impactful systems, including autonomous vehicles, financial and healthcare decision-making, and social and criminal justice. In this work, we propose a method, LazyVI, for efficiently estimating variable importance based on a linearization of a fully trained neural network. We prove that our method provides an accurate estimate of VI and can achieve the same rate of accuracy as a computationally expensive retraining method nearly as quickly as the inaccurate dropout method. We further show how to

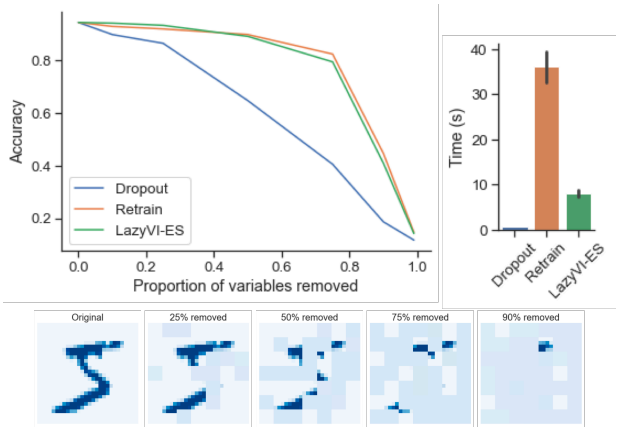

Figure 6. (Left) Average accuracy of estimator across 5 repetitions. (Right) Average computation time for a single run of each method. (Bottom) Example MNIST image with different proportions of variables removed.

construct confidence intervals around these estimates.

The theory developed in this paper provides an important step toward making interpretability in neural networks more computationally efficient, and we suspect this theoretical framework can extend to other settings, which we discuss here.

7.1. Early stopping and regularization

A potential alternative to our proposed LazyVI method is to first train a full model (as we do) and then train the reduced model using a gradient-based method initialized with the full model parameters and stopped early. Empirical evidence suggests that this approach would have similar speed and accuracy to our LazyVI approach due to the implicit regularization associated with early stopping. This approach has the potential to extend LazyVI to far more complicated architectures than the standard feedforward networks we have experimented with thus far. As a proof of concept, we train a convolutional neural network on the MNIST benchark dataset, and then follow the ROAR procedure with random ordering. We see in Figure 6 that Dropout results in a consistent decline in predictive performance when variables are removed, while the accuracy remains relatively high when the network is retrained up until around 75% of variables are removed - remarkable, given how relatively uninformative the image appears (Figure 6, bottom). Importantly, we see that taking a single step from a model initialized at the full model parameters (LazyVI-ES) results in nearly identical performance to the full retraining at a $5\times$ speed-up.

While these results are promising, we currently lack theoretical guarantees for early stopping in this setting. It is possible that our theoretical results could lead to new insights into early stopping for assessing VI due to the intimate connection between kernel ridge regression and early stopping algorithms (Raskutti et al., 2014). In fact, if the eigenvalues of the NTK matrix at the full model initialization decay in a sufficiently fast rate, early stopping of the reduced model training should give an as good estimate of the reduced model. However, analyzing early stopping in this setting requires characterizing the spectrum of the NTK with *the full model initialization*, whereas most spectral properties of the NTK have been developed under the assumption of a *random initialization* (Nguyen et al., 2021; Montanari & Zhong, 2020). Better understanding the NTK spectrum after full-model initialization in the future could provide new insights into fast algorithms for VI estimation.

7.2. Shapley values

When features are correlated, the quantity VI defined in (6) tends to zero. Recent work proposes using Shapley values to measure variable importance, arguing that their handling of correlated variables, which assigns similar positive weights to correlated important variables, is desirable (Owen & Prieur, 2016; Williamson & Feng, 2020) in some settings. These papers also note that Shapley values are prohibitively expensive to compute, as they require fitting a new model for each of the 2^p possible subsets of variables. However, we note that computing the Shapley values requires many calculations of the quantity in (6); an important avenue is investigating the use of our LazyVI framework to accelerate the computation of Shapley values. We perform a preliminary experiment on calculating Shapley values using our LazyVI framework, and compare it with the retraining method used in (Williamson & Feng, 2020). We estimate the Shapley values for a sparse high-dimensional data generated by a logistic model, and perform the retraining/lazyVI method on a two-layer neural network. When using LazyVI training, the computation is roughly 5 times faster and the estimated Shapley values are close to retraining. Moreover, when the sample size is relatively small with respective to the dimension, we observe Lazy training has a smaller variance of estimated Shapley values on non-important variables than retraining method, due to the regularization proposed in our method. See Appendix C.7 for our initial exploration into this line of work.

Acknowledgements

This work was supported by AFOSR FA9550-18-1-0166, DOE DE-AC02-06CH113575, NSF OAC-1934637, NSF DMS-1930049 and NSF DMS-2023109.

References

AghaKouchak, A., Feldman, D., Hoerling, M., Huxman, T., and Lund, J. Water and climate: Recognize anthropogenic drought. *Nature*, 524(7566), August 2015. ISSN 1476-4687. doi: 10.1038/524409a. URL https://www.nature.com/articles/524409a.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Barber, R. F. and Candes, E. J. A knockoff filter for high-dimensional selective inference, 2018.

Barron, A. R. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Candes, E., Fan, Y., Janson, L., and Lv, J. Panning for Gold: Model-X Knockoffs for High-dimensional Controlled Variable Selection. *arXiv:1610.02351 [math, stat]*, December 2017. URL http://arxiv.org/abs/1610.02351. arXiv: 1610.02351.

Chang, C.-H., Rampasek, L., and Goldenberg, A. Dropout feature ranking for deep learning models. *arXiv preprint arXiv:1712.08645*, 2017.

Chen, Y., Morton, D. C., Andela, N., Giglio, L., and Randerson, J. T. How much global burned area can be forecast on seasonal time scales using sea surface temperatures? *Environmental Research Letters*, 11(4):045001, March 2016. ISSN 1748-9326. doi: 10.1088/1748-9326/11/4/045001. URL https://doi.org/10.1088/1748-9326/11/4/045001.

Chizat, L., Oyallon, E., and Bach, F. On Lazy Training in Differentiable Programming. *arXiv:1812.07956*, January 2020.

Dai, A. The influence of the inter-decadal Pacific oscillation on US precipitation during 1923–2010. *Climate Dynamics*, 41(3), August 2013. ISSN 1432-0894. doi: 10.1007/s00382-012-1446-5. URL https://doi.org/10.1007/s00382-012-1446-5.

de La Beaujardière, J., Banihirwe, A., Shih, C. F. G., Paul, K., and Hamman, J. Ncar cesm lens cloud-optimized subset. *UCAR/NCAR Computational and Informations Systems Lab*, 2019.

Doksum, K. and Samarov, A. Nonparametric estimation of global functionals and a measure of the explanatory power of covariates in regression. *The Annals of Statistics*, pp. 1443–1473, 1995.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), August 2018. ISSN 0360-0300. doi: 10.1145/3236009. URL https://doi.org/10.1145/3236009.

Hooker, S. et al. A benchmark for interpretability methods in deep neural networks. In *NeurIPS*, 2019.

Hsu, D., Kakade, S., and Zhang, T. A tail inequality for quadratic forms of subgaussian random vectors. *Electronic Communications in Probability*, 17:1–6, 2012.

Hu, W., Li, Z., and Yu, D. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. *arXiv preprint arXiv:1905.11368*, 2019.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks, 2020.

Kay, J. E., Deser, C., Phillips, A., Mai, A., Hannay, C., Strand, G., Arblaster, J. M., Bates, S. C., Danabasoglu, G., Edwards, J., Holland, M., Kushner, P., Lamarque, J.-F., Lawrence, D., Lindsay, K., Middleton, A., Munoz, E., Neale, R., Oleson, K., Polvani, L., and Vertenstein, M. The Community Earth System Model (CESM) Large Ensemble Project: A Community Resource for Studying Climate Change in the Presence of Internal Climate Variability. *Bulletin of the American Meteorological Society*, 96(8):1333–1349, August 2015. ISSN 0003-0007, 1520-0477. doi: 10.1175/BAMS-D-13-00255.1. URL https://journals.ametsoc.org/view/journals/bams/96/8/bams-d-13-00255.1.xml.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124002, Dec 2020. ISSN 1742-5468. doi: 10.1088/1742-5468/abc62b. URL http://dx.doi.org/10.1088/1742-5468/abc62b.

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

Lenssen, N. J. L., Goddard, L., and Mason, S. Seasonal Forecast Skill of ENSO Teleconnection Maps. *Weather and Forecasting*, 35(6):2387–2406, December 2020. ISSN 1520-0434, 0882-8156. doi: 10.1175/WAF-D-19-0235.1.

Publisher: American Meteorological Society Section: Weather and Forecasting.

Li, C., Zwiers, F., Zhang, X., Chen, G., Lu, J., Li, G., Norris, J., Tan, Y., Sun, Y., and Liu, M. Larger Increases in More Extreme Local Precipitation Events as Climate Warms. *Geophysical Research Letters*, 46(12):6885–6891, 2019. ISSN 1944-8007. doi: 10.1029/2019GL082908.

Mamalakis, A., Yu, J.-Y., Randerson, J. T., AghaKouchak, A., and Foufoula-Georgiou, E. A new interhemispheric teleconnection increases predictability of winter precipitation in southwestern US. *Nature Communications*, 9 (1):2332, June 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-04722-7. URL https://www.nature.com/articles/s41467-018-04722-7.

Mamalakis, A., Barnes, E. A., and Ebert-Uphoff, I. Investigating the fidelity of explainable artificial intelligence methods for applications of convolutional neural networks in geoscience, 2022a. URL https://arxiv.org/abs/2202.03407.

Mamalakis, A., Ebert-Uphoff, I., and Barnes, E. A. Neural network attribution methods for problems in geoscience: A novel synthetic benchmark dataset. *Environmental Data Science*, 1:e8, 2022b. doi: 10.1017/eds.2022.7.

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

Montanari, A. and Zhong, Y. The interpolation phase transition in neural networks: Memorization and generalization under lazy training. *arXiv preprint arXiv:2007.12826*, 2020.

Nguyen, Q., Mondelli, M., and Montufar, G. F. Tight bounds on the smallest eigenvalue of the neural tangent kernel for deep relu networks. In *International Conference on Machine Learning*, pp. 8119–8129. PMLR, 2021.

Owen, A. B. and Prieur, C. On Shapley value for measuring importance of dependent inputs, October 2016. URL https://arxiv.org/abs/1610.02080v3.

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

Rudin, C. and Radin, J. 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), 11 2019. doi: 10.1162/99608f92.5a8a3a3d. URL https://hdsr.mitpress.mit.edu/pub/f9kuryi8. https://hdsr.mitpress.mit.edu/pub/f9kuryi8.

Sapp, S., van der Laan, M. J., and Page, K. Targeted estimation of binary variable importance measures with interval-censored outcomes. *The international journal of biostatistics*, 10(1):77–97, 2014.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning Important Features Through Propagating Activation Differences. *arXiv:1704.02685 [cs]*, October 2019. URL http://arxiv.org/abs/1704.02685. arXiv: 1704.02685.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013. URL https://arxiv.org/abs/1312.6034.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Stevens, A., Willett, R., Mamalakis, A., Foufoula-Georgiou, E., Tejedor, A., Randerson, J. T., Smyth, P., and Wright, S. Graph-Guided Regularized Regression of Pacific Ocean Climate Variables to Increase Predictive Skill of Southwestern U.S. Winter Precipitation. *Journal of Climate*, 34 (2):737–754, January 2021. ISSN 0894-8755, 1520-0442. doi: 10.1175/JCLI-D-20-0079.1.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Tabari, H. Climate change impact on flood and extreme precipitation increases with water availability. *Scientific Reports*, 10(1), August 2020. ISSN 2045-2322. doi: 10.1038/s41598-020-70816-2. URL https://www.nature.com/articles/s41598-020-70816-2.

Williamson, B. D. and Feng, J. Efficient nonparametric statistical inference on population feature importance using Shapley values. *arXiv:2006.09481 [stat]*, June 2020. URL http://arxiv.org/abs/2006.09481. arXiv: 2006.09481.

Williamson, B. D., Gilbert, P. B., Simon, N. R., and Carone, M. A general framework for inference on algorithm-agnostic variable importance. *arXiv:2004.03683*, 2021.

Zhang, L. and Janson, L. Floodgate: inference for model-free variable importance. *arXiv:2007.01283 [stat]*, April 2021. URL http://arxiv.org/abs/2007.01283. arXiv: 2007.01283.

A. SUPPORTING LEMMAS

Conditions

(A1) There exists some constant $C > 0$ such that, for each sequence $f_1, f_2, \cdots \in \mathcal{F}$ such that $\|f_i - f_0\|_{\mathcal{F}} \to 0$, $|V(f_j, P_0) - V(f_0, P_0)| \leq C\|f_j - f_0\|_{\mathcal{F}}^2$ for each j large enough;

(A2) There exists some constant $\delta > 0$ such that for each sequence $\epsilon_1, \epsilon_2, \cdots \in \mathbb{R}$ and $h, h_1, h_2, \cdots \in \mathbb{R}$ satisfying that $\epsilon_j \to 0$ and $\|h_j - h\|_\infty \to 0$, it holds that

$$\sup_{f \in \mathcal{F}: \|f - f_0\|_{\mathcal{F}} < \delta} |\frac{V(f, P_0 + \epsilon_j h_j) - V(f, P_0)}{\epsilon_j} - \dot{V}(f, P_0; h_j)| \to 0;$$

(B2) $\int [g_n(z)]^2 dP_0(z) = o_P(1)$;

Lemma A.1. *((Williamson et al., 2021)) Suppose (A1-A2, B2) regularity conditions hold. Denote $f_n(X)$ and $f_{n,-j}(X_{-j})$ as the estimate for f_0 and $f_{0,-j}$, Then for a predictive skill measure $V(f, P)$ satisfying conditions (A1)-(A2), (B2) in Appendix, as long as the estimators satisfy the following condition:*

$$\|f_n - f_0\|_{\mathcal{F}} = O_p(n^{-\frac{1}{4}}), \;\; \|f_{n,-j} - f_{0,-j}\|_{\mathcal{F}} = O_p(n^{-\frac{1}{4}}), \tag{28}$$

for all $j \in [p]$, then we have

$$\begin{aligned} v_n - v_0 =& \frac{1}{n}\sum_{i=1}^{n} \dot{V}(f_0, P_0; \delta_{Z_i} - P_0) + O_p(\frac{1}{\sqrt{n}}), \\ v_{n,-j} - v_{0,-j} =& \frac{1}{n}\sum_{i=1}^{n} \dot{V}(f_{0,-j}, P_{0,-j}; \delta_{Z_i} - P_{0,-j}) + O_p(\frac{1}{\sqrt{n}}), \end{aligned} \tag{29}$$

where $v_n = V(f_n, P_n)$ and $v_{n,-j} = V(f_{n,-j}, P_{n,-j})$, $\forall j \in [p]$.

B. MISSING PROOFS

B.1. Proof of Lemma 4.5

In this section, we present the detailed proof of Lemma 4.5, which gives the empirical estimation error bounds for the NTK kernel ridge regression estimation. The proof follows the proof framework provided in (Hu et al., 2019).

[Proof.]

First of all, according to kernel ridge regression, denote

- $\mathbf{Y} = (Y_1, \ldots, Y_n)^\top$;
- $\mathbf{h}_{\theta_f}^{(j)} = (h_{\theta_f}(\mathbf{X}_1^{(j)}), \ldots, h_{\theta_f}(\mathbf{X}_n^{(j)}))^\top$;
- $\mathbf{f}_{0,-j} = (f_{0,-j}(\mathbf{X}_1^{(j)}), \ldots, f_{0,-j}(\mathbf{X}_n^{(j)}))^\top$.

we have

$$(\tilde{h}_{\theta_f + \Delta\theta_j}(\mathbf{X}_1^{(j)}), \ldots, \tilde{h}_{\theta_f + \Delta\theta_j}(\mathbf{X}_n^{(j)}))^\top = \mathbb{K}^{(j)}(\mathbb{K}^{(j)} + \lambda I_n)^{-1}(\mathbf{Y} - \mathbf{h}_{\theta_f}^{(j)}) + \mathbf{h}_{\theta_f}^{(j)}. \tag{30}$$

Recall that $\epsilon^{(j)} = Y - \mathbb{E}(Y|X_{-j}) = Y - f_{0,-j}(\mathbf{X}^{(j)})$, we define its observed samples as

$$\boldsymbol{\epsilon}^{(j)} = \left(Y_1 - f_{0,-j}(\mathbf{X}_1^{(j)}), \ldots, Y_n - f_{0,-j}(\mathbf{X}_n^{(j)})\right)^\top = \mathbf{Y} - \mathbf{f}_{0,-j};$$

Recall the definition of $\mathbf{e}^{(j)}$, we have

$$\mathbf{e}^{(j)} = \mathbf{f}_{0,-j} - \mathbf{h}_{\theta_f}^{(j)}.$$

Hence we have

$$\begin{aligned}
&\sqrt{n}\|\tilde{h}_{\theta_f+\Delta\theta_j}(X^{(j)}) - f_{0,-j}(X^{(j)})\|_n \\
=&\sqrt{\sum_{i=1}^{n}\left[\tilde{h}_{\theta_f+\Delta\theta_j}(\mathbf{X}_i^{(j)}) - f_{0,-j}(\mathbf{X}_i^{(j)})\right]^2} \\
=&\|\mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-1}(\mathbf{Y}-\mathbf{h}_{\theta_f}^{(j)}) + \mathbf{h}_{\theta_f}^{(j)} - \mathbf{f}_{0,-j}\| \\
=&\|\mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-1}(\mathbf{f}_{0,-j}+\boldsymbol{\epsilon}^{(j)}-\mathbf{h}_{\theta_f}^{(j)}) + \mathbf{h}_{\theta_f}^{(j)} - \mathbf{f}_{0,-j}\| \\
=&\|\mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-1}(\mathbf{e}^{(j)}+\boldsymbol{\epsilon}^{(j)}) - \mathbf{e}^{(j)}\| \\
=&\|\mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-1}\boldsymbol{\epsilon}^{(j)} - \lambda(\mathbb{K}^{(j)}+\lambda I_n)^{-1}\mathbf{e}^{(j)}\| \\
\leq&\|\mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-1}\boldsymbol{\epsilon}^{(j)}\| + \|\lambda(\mathbb{K}^{(j)}+\lambda I_n)^{-1}\mathbf{e}^{(j)}\|.
\end{aligned} \tag{31}$$

According to Assumption 4.2 and (Hsu et al., 2012), we have

$$P\left(\|A\boldsymbol{\epsilon}^{(j)}\|^2/\sigma^2 > \mathrm{tr}(\Sigma) + 2\sqrt{\mathrm{tr}(\Sigma^2)t} + 2\|\Sigma\|t \mid \mathbf{X}^{(j)}\right) \leq e^{-t}, \tag{32}$$

where $A = \mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-1}$, and $\Sigma = A^\top A$. Hence we have with probability at least $1-\delta$ for any $\delta > 0$, we have

$$\|\mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-1}\boldsymbol{\epsilon}^{(j)}\| \leq \sigma\sqrt{\mathrm{tr}(\Sigma) + 2\sqrt{\mathrm{tr}(\Sigma^2)\log(\frac{1}{\delta})} + 2\|\Sigma\|\log(\frac{1}{\delta})}. \tag{33}$$

Let $\lambda_1, \ldots, \lambda_n > 0$ be the eigenvalues of $\mathbb{K}^{(j)}$, we then have

$$\begin{aligned}
&\mathrm{tr}[\Sigma] = \mathrm{tr}[A^\top A] = \sum_{i=1}^{n}\frac{\lambda_i^2}{(\lambda_i+\lambda)^2} \leq \sum_{i=1}^{n}\frac{\lambda_i^2}{4\lambda_i\cdot\lambda} = \frac{\mathrm{tr}[\mathbb{K}^{(j)}]}{4\lambda}, \\
&\mathrm{tr}[\Sigma^2] = \mathrm{tr}[A^\top A^2 A^\top] = \sum_{i=1}^{n}\frac{\lambda_i^4}{(\lambda_i+\lambda)^4} \leq \sum_{i=1}^{n}\frac{\lambda_i^4}{4^4\lambda(\frac{\lambda_i}{3})^3} = \frac{3^3\mathrm{tr}[\mathbb{K}^{(j)}]}{4^4\lambda} \leq \frac{\mathrm{tr}[\mathbb{K}^{(j)}]}{4\lambda}, \\
&\|\Sigma\| = \|\mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-2}\mathbb{K}^{(j)}\| \leq 1.
\end{aligned} \tag{34}$$

Hence we have with probability at least $1-\delta$,

$$\begin{aligned}
&\|\mathbb{K}^{(j)}(\mathbb{K}^{(j)}+\lambda I_n)^{-1}\boldsymbol{\epsilon}^{(j)}\| \\
\leq&\sigma\sqrt{\frac{\mathrm{tr}[\mathbb{K}^{(j)}]}{4\lambda} + 2\sqrt{\frac{\mathrm{tr}[\mathbb{K}^{(j)}]}{4\lambda}} + 2\log(\frac{1}{\delta})} \\
\leq&\sigma\sqrt{\frac{\mathrm{tr}[\mathbb{K}^{(j)}]}{4\lambda} + 2\sqrt{\frac{\mathrm{tr}[\mathbb{K}^{(j)}]}{2\lambda}} + 2\log(\frac{1}{\delta})} \\
=&\sigma\sqrt{\frac{\mathrm{tr}[\mathbb{K}^{(j)}]}{4\lambda}} + \sigma\sqrt{2\log(\frac{1}{\delta})}.
\end{aligned} \tag{35}$$

By the fact that

$$\|\lambda(\mathbb{K}^{(j)}+\lambda I_n)^{-1}\mathbf{e}^{(j)}\| = \lambda\sqrt{(\mathbf{e}^{(j)})^\top(\mathbb{K}^{(j)}+\lambda I_n)^{-2}\mathbf{e}^{(j)}}, \tag{36}$$

we have

$$\|\tilde{h}_{\theta_f+\Delta\theta_j}(X^{(j)}) - f_{0,-j}(X^{(j)})\|_n \leq \frac{\lambda}{\sqrt{n}}\sqrt{(\mathbf{e}^{(j)})^\top[\mathbb{K}^{(j)}+\lambda I_n]^{-2}\mathbf{e}^{(j)}} + \sigma\sqrt{\frac{\mathrm{tr}[\mathbb{K}^{(j)}]}{4n\lambda}} + \sigma\sqrt{\frac{2}{n}\log(\frac{1}{\delta})}. \tag{37}$$

□

B.2. Lemma B.1 and its proof

Define the Hilbert norm for a function $f(x) = \alpha^\top K(x, \mathbf{X}^{(j)}),\ \forall \alpha \in \mathbb{R}^n$ in the NTK kernel space is: $\|f\|_{\mathcal{H}} = \sqrt{\alpha^\top \mathbb{K}^{(j)} \alpha}$. The following lemma is to bound the Hilbert norm for $\tilde{h}_{\theta_f + \Delta\theta_j} - h_{\theta_f}$ so that we could bound the complexity of the function class it lies in.

Lemma B.1. *With probability at least $1-\delta$, for any $j \in [p]$ we have*

$$\|\tilde{h}_{\theta_f+\Delta\theta_j}(x) - h_{\theta_f}(x)\|_{\mathcal{H}} \leq \sqrt{(\mathbf{e}^{(j)})^\top (\mathbb{K}^{(j)} + \lambda I_n)^{-1} \mathbf{e}^{(j)}} + \frac{\sigma}{\sqrt{\lambda}} \left(\sqrt{n} + \sqrt{2\log(\frac{1}{\delta})} \right). \tag{38}$$

[Proof.] Recall that $\tilde{h}_{\theta_f+\Delta\theta_j}(x) = \text{ker}_{\theta_f}(x, \mathbf{X}^{(j)})(\mathbb{K}^{(j)} + \lambda I_n)^{-1}(\mathbf{Y} - \mathbf{h}_{\theta_f}^{(j)}) + h_{\theta_f}(x)$. Based on the fact that $\mathbf{Y} - \mathbf{h}_{\theta_f}^{(j)} = \mathbf{e}^{(j)} + \boldsymbol{\epsilon}^{(j)}$, we have

$$\begin{aligned}
&\|\tilde{h}_{\theta_f+\Delta\theta_j}(x) - h_{\theta_f}(x)\|_{\mathcal{H}} \\
=&\|(\mathbf{Y} - \mathbf{h}_{\theta_f}^{(j)})^\top (\mathbb{K}^{(j)} + \lambda I_n)^{-1} \text{ker}_{\theta_f}(\mathbf{X}^{(j)}, x)\|_{\mathcal{H}} \\
=&\sqrt{(\mathbf{e}^{(j)} + \boldsymbol{\epsilon}^{(j)})^\top (\mathbb{K}^{(j)} + \lambda I_n)^{-1} \mathbb{K}^{(j)} (\mathbb{K}^{(j)} + \lambda I_n)^{-1} (\mathbf{e}^{(j)} + \boldsymbol{\epsilon}^{(j)})} \\
\leq&\sqrt{(\mathbf{e}^{(j)} + \boldsymbol{\epsilon}^{(j)})^\top (\mathbb{K}^{(j)} + \lambda I_n)^{-1} (\mathbf{e}^{(j)} + \boldsymbol{\epsilon}^{(j)})} \\
\leq&\sqrt{(\mathbf{e}^{(j)})^\top (\mathbb{K}^{(j)} + \lambda I_n)^{-1} \mathbf{e}^{(j)}} + \sqrt{(\boldsymbol{\epsilon}^{(j)})^\top (\mathbb{K}^{(j)} + \lambda I_n)^{-1} \boldsymbol{\epsilon}^{(j)}} \\
\leq&\sqrt{(\mathbf{e}^{(j)})^\top (\mathbb{K}^{(j)} + \lambda I_n)^{-1} \mathbf{e}^{(j)}} + \sqrt{\frac{(\boldsymbol{\epsilon}^{(j)})^\top \boldsymbol{\epsilon}^{(j)}}{\lambda}}.
\end{aligned} \tag{39}$$

Using the concentration inequality in (Hsu et al., 2012) again, we have with probability at least $1-\delta$, we have

$$\sqrt{(\boldsymbol{\epsilon}^{(j)})^\top \boldsymbol{\epsilon}^{(j)}} \leq \sigma \sqrt{n + 2\sqrt{n\log(\frac{1}{\delta})} + 2\log(\frac{1}{\delta})} \leq \sigma \left(\sqrt{n} + \sqrt{2\log(\frac{1}{\delta})} \right). \tag{40}$$

Hence we prove Lemma B.1 by combining Equation (39) and Equation (40).

B.3. Generalization error bound and its proof

In the following, we will bound the generalization error based on the above empirical error bound.

Lemma B.2. *For any $j \in [p]$, let $\|\cdot\|$ be the $L_2(P_0)$ norm defined as $\|f\| = \sqrt{\int |f(x^{(j)})| dP_0(x)}$, then we have with probability at least $1-\delta$ for any $\delta > 0$,*

$$\begin{aligned}
\|\tilde{h}_{\theta_f+\Delta\theta_j} - f_{0,-j}\| \leq& \left\{ \frac{\lambda}{\sqrt{n}} \sqrt{(\mathbf{e}^{(j)})^\top [\mathbb{K}^{(j)} + \lambda I_n]^{-2} \mathbf{e}^{(j)}} + \sigma \sqrt{\frac{tr[\mathbb{K}^{(j)}]}{4n\lambda}} + \sigma \sqrt{\frac{2}{n}\log(\frac{3}{\delta})} \right\} \\
&+ \frac{2\sqrt{tr[\mathbb{K}^{(j)}]}}{n} \left[O(1) + \frac{\sigma}{\sqrt{\lambda}} (\sqrt{n} + \sqrt{2\log(3/\delta)}) \right] + \sqrt{\frac{\log(3/\delta)}{2n}}.
\end{aligned} \tag{41}$$

Under Assumption 4.1 and Assumption 4.2, when we take the penalty parameter in the rate $\lambda = O(\sqrt{n})$, we have with high probability that $\|\tilde{h}_{\theta_f+\Delta\theta_j} - f_{0,-j}\| \leq O_p(n^{-1/4})$.

[Proof.] According to Lemma 4.5, we know that with probability at least $1 - \delta/3$,

$$\|\tilde{h}_{\theta_f+\Delta\theta_j} - f_{0,-j}\|_n \leq \frac{\lambda}{\sqrt{n}} \sqrt{(\mathbf{e}^{(j)})^\top [\mathbb{K}^{(j)} + \lambda I_n]^{-2} \mathbf{e}^{(j)}} + \sigma \sqrt{\frac{\text{tr}[\mathbb{K}^{(j)}]}{4n\lambda}} + \sigma \sqrt{\frac{2}{n}\log(\frac{3}{\delta})}. \tag{42}$$

By (Bartlett & Mendelson, 2002), we know that the empirical Rademacher complexity for a function class $\mathcal{F}_B = \{f(x) = \alpha^\top \ker_{\theta_f}(\mathbf{X}^{(j)}, x) : \|f\|_{\mathcal{H}} \leq B\}$ is bounded as

$$\hat{\mathcal{R}}_S(\mathcal{F}_B) \leq \frac{B\sqrt{\text{tr}[\mathbb{K}^{(j)}]}}{n}.$$

According to (Mohri et al., 2018), with probability at least $1-\delta/3$, we have

$$\begin{aligned}
&\sup_{\tilde{h}_{\theta_f+\Delta\theta_j} - h_{\theta_f} \in \mathcal{F}} \left\{ \left\| \tilde{h}_{\theta_f+\Delta\theta_j}(x^{(j)}) - h_{\theta_f}(x^{(j)}) - \left(f_{0,-j}(x^{(j)}) - h_{\theta_f}(x^{(j)}) \right) \right\| - \|\tilde{h}_{\theta_f+\Delta\theta_j}(x) - f_{0,-j}(x^{(j)})\|_n \right\} \\
\leq & 2\hat{\mathcal{R}}_S(\mathcal{F}) + \sqrt{\frac{\log(3/\delta)}{2n}}.
\end{aligned} \tag{43}$$

From Assumption 4.1 and Lemma B.1, we have with probability $1-\delta/3$

$$\|\tilde{h}_{\theta_f+\Delta\theta_j}(x) - h_{\theta_f}(x)\|_{\mathcal{H}} := B' \leq O(1) + \frac{\sigma}{\sqrt{\lambda}} \left(\sqrt{n} + \sqrt{2\log(\frac{3}{\delta})} \right). \tag{44}$$

Then we have with probability $1-\delta$,

$$\begin{aligned}
&\|\tilde{h}_{\theta_f+\Delta\theta_j} - f_{0,-j}\| \\
\leq & \|\tilde{h}_{\theta_f+\Delta\theta_j} - f_{0,-j}\|_n + 2\hat{\mathcal{R}}_S(\mathcal{F}) + \sqrt{\frac{\log(3/\delta)}{2n}} \\
\leq & \left\{ \frac{\lambda}{\sqrt{n}} \sqrt{(\mathbf{e}^{(j)})^\top [\mathbb{K}^{(j)} + \lambda I_n]^{-2} \mathbf{e}^{(j)}} + \sigma\sqrt{\frac{\text{tr}[\mathbb{K}^{(j)}]}{4n\lambda}} + \sigma\sqrt{\frac{2}{n}\log(\frac{3}{\delta})} \right\} + \frac{2B'\sqrt{\text{tr}[\mathbb{K}^{(j)}]}}{n} + \sqrt{\frac{\log(3/\delta)}{2n}} \\
\leq & \left\{ \frac{\lambda}{\sqrt{n}} \sqrt{(\mathbf{e}^{(j)})^\top [\mathbb{K}^{(j)} + \lambda I_n]^{-2} \mathbf{e}^{(j)}} + \sigma\sqrt{\frac{\text{tr}[\mathbb{K}^{(j)}]}{4n\lambda}} + \sigma\sqrt{\frac{2}{n}\log(\frac{3}{\delta})} \right\} \\
& + \frac{2\sqrt{\text{tr}[\mathbb{K}^{(j)}]}}{n} \left[O_p(1) + \frac{\sigma}{\sqrt{\lambda}}(\sqrt{n} + \sqrt{2\log(3/\delta)}) \right] + \sqrt{\frac{\log(3/\delta)}{2n}}
\end{aligned} \tag{45}$$

By the assumptions that $\|[\mathbb{K}^{(j)} + \lambda I_n]^{-1}\mathbf{e}^{(j)}\|^2 = O_p(1/\sqrt{n})$ and $\text{tr}[\mathbb{K}^{(j)}] = O_p(n)$ in Assumption 4.1 (a) (b), when we take $\lambda = O(\sqrt{n})$, we have

$$\|\tilde{h}_{\theta_f+\Delta\theta_j} - f_{0,-j}\| \leq O_p(n^{-1/4}). \tag{46}$$

B.4. Proof of Lemma 4.6

Lemma 4.6 *For a large neural network whose width is in the order of $O(\sqrt{n})$ where n is the training sample size, our lazy trained neural network is close to its linearization with high probability:*

$$\|\tilde{h}_{\theta_f+\Delta\theta_j}(x) - h_{\theta_f+\Delta\theta_j}(x)\| \leq O(n^{-1/4}). \tag{47}$$

[Proof.] Since $\tilde{h}_{\theta_f+\Delta\theta_j}(x) = h_{\theta_f} + \Delta\theta_j^\top \nabla_\theta h_\theta(x)|_{\theta=\theta_f}$ is a linearization of $h_{\theta_f+\Delta\theta_j}(x)$ around the initialization θ_f, according to Theorem 2.1 in (Lee et al., 2020), when the neural network has a width M, the neural network is close to its linearization with probability arbitrarily close to 1:

$$\|\tilde{h}_{\theta_f+\Delta\theta_j}(x) - h_{\theta_f+\Delta\theta_j}(x)\|_2 = O(\frac{1}{\sqrt{M}}). \tag{48}$$

Specifically, when the neural network M takes the order of $O(\sqrt{n})$, we have $\|\tilde{h}_{\theta_f+\Delta\theta_j}(x) - h_{\theta_f+\Delta\theta_j}(x)\|_2 = O(n^{-1/4})$.

B.5. Proof of the Main Theorem (Theorem 4.4)

Based on Lemma B.2 and Lemma 4.6 , for a neural network with width at least $M = O(\sqrt{n})$ when the assumptions hold true, by triangular inequality we have

$$\|h_{\theta_f+\Delta\theta_j} - f_{0,-j}\| \leq \|h_{\theta_f+\Delta\theta_j} - \tilde{h}_{\theta_f+\Delta\theta_j}\| + \|\tilde{h}_{\theta_f+\Delta\theta_j} - f_{0,-j}\| = O_p(n^{-1/4}). \tag{49}$$

This holds true for any $j \in [p]$. Then by Lemma A.1, we finish the proof for Theorem 4.4.

B.6. Proof of Example 3.1

The density of X_1 given X_2 in the setting of Example 3.1 is:

$$f(x_1|x_2;\rho,\sigma) = \frac{1}{2\pi\sigma^2\sqrt{1-\rho^2}} \exp\left\{-\frac{1}{2(1-\rho^2)\sigma^2}(x_1^2 - 2\rho x_1 x_2 + x_2^2)\right\}, \tag{50}$$

thus we have $X_1|X_2 \sim \mathcal{N}(\rho X_2, (1-\rho^2)\sigma^2)$.

C. ADDITIONAL EXPERIMENTS

C.1. Trace divergence rate of the Neural Tangent Kernel Matrix

In Assumption 4.1(b), we assume the trace of the neural tangent kernel matrix with full-model parameters as initialization diverges in the order of n in probability: $\text{tr}(\mathbb{K}^{(j)}) = O_p(n)$. In the following experiment, we'll verify this through a simulation.

We consider a two-layer neural network with 128 nodes in the hidden layer. The data is generated from a sparse linear model $Y = 1.5X_1 + 1.2X_2 + X_3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.1^2)$ and we have 6 predictors $X_1, X_2, \ldots, X_6$ generated from a normal distribution $\mathcal{N}(0, I_6 + C)$ with $C_{1,2} = C_{2,1} = 0.5$ and $C_{i,j} = 0$ for $(i,j) \notin \{(1,2),(2,1)\}$ (this is to add some correlation to the predictors); The total sample size of the data varies in the set $\{1000, 1100, \ldots, 4000\}$; Among these simulated samples at each sample size, $2/3$ data are sampled into the training set and $1/3$ samples fall in the testing set.

We'll first train the full NN model on the training set and get inferred parameters $\hat{\theta}_f$ in the neural network from the full training data. Then we use $\hat{\theta}_f$ as initialization for the reduced neural network. Then we could calculate the neural tangent kernel matrix and its corresponding trace the testing data with one feature (*e.g.*the first feature) dropped. We will repeat this process at each sample size level five times (with different random seed to generate data each time), and record the traces with respect to the test data sample size. As shown in Figure 7, there is a clear trend that the trace diverges linearly as the sample size, which numerically verifies of Assumption 4.1 (b).

C.2. Choosing the regularization parameter

LazyVI involves solving a ridge regression to estimate the difference between the full and reduced model parameters. For variable j, we choose the regularization parameter λ_j through K-fold cross validation on the prediction made using the estimated $\Delta\theta_j^{\lambda}$. Algorithm 2 below shows the entire procedure.

C.3. Full linear experiment

Figure 8 shows the distribution of computation time vs. VI estimation accuracy for three different groups of variables (important and correlated, important and uncorrelated, unimportant and uncorrelated). We see that LazyVI and retrain are both accurate across all groups of variables, but LazyVI is much faster. Dropout is consistently the fastest method, but is highly inaccurate in estimating VI for the first group of variables due to their strong correlations. Also in Figure 8 we show the empirical coverage of the LazyVI and retrain 95% confidence intervals. We see that both retrain and LazyVI achieve desirable coverage for the three important variables; poor coverage of unimportant variables is expected and possibly remedied with a sample-splitting procedure (Williamson et al., 2021).

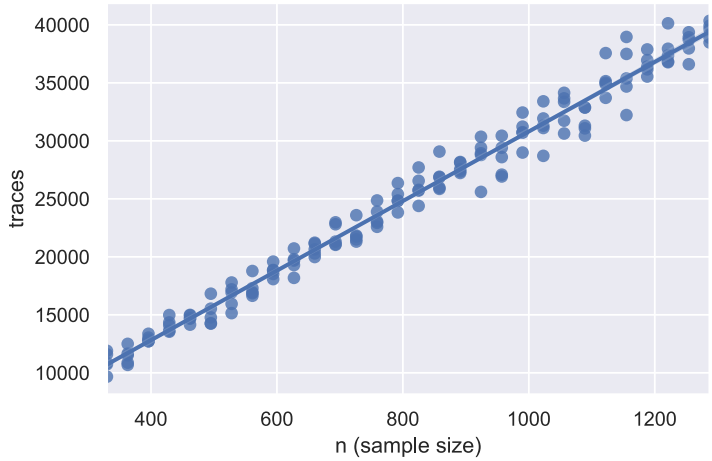

Figure 7. The divergence of the NTK trace with respect to the sample size

Algorithm 2 K-Fold CV for λ_j

Require: $\{\mathbf{X}_i^{(j)}, Y_i, \mathbf{e}_i^{(j)}, \Phi_i^{(j)}\}_{i=1}^{n_1}$ and θ_f from Algorithm 1 in main paper; candidate λ values Λ
  Partition $[n_1]$ into K subsets, each denoted S_t
  for $\lambda \in \Lambda$ **do**
    for $k = 1, \ldots, K$ **do**
      $\Delta\theta_j^\lambda = \arg\min_{\omega \in \mathbb{R}^M} \frac{1}{n_1 - |S_k|} \sum_{i \notin S_k} [\mathbf{e}_i^{(j)} - \langle \omega, \Phi_i^{(j)} \rangle]^2 + \lambda \|\omega\|_2^2$
      $\hat{Y}_i = h_{\theta_f + \Delta\theta_j^\lambda}(\mathbf{X}_i^{(j)})$ for $i \in S_k$
      $\epsilon_{\lambda,k} = \frac{1}{|S_k|} \sum_{i \in S_k} (Y_i - \hat{Y}_i)^2$
    end for
    $\epsilon_\lambda = \frac{1}{K} \sum_{k=1}^K \epsilon_{\lambda,k}$
  end for
Ensure: $\hat{\lambda}_j = \arg\min_\lambda \{\epsilon_\lambda\}_{\lambda \in \Lambda}$

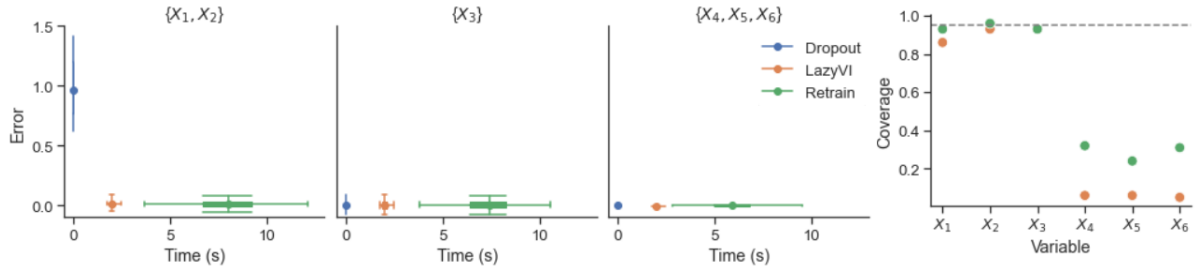

Figure 8. Distribution of computation time vs. estimation error relative to retrain ($\hat{\text{VI}} - \hat{\text{VI}}^{(\text{RT})}$) for three different groups of variables: important, correlated ($\{X_1, X_2\}$); important, uncorrelated (X_3); and unimportant, uncorrelated ($\{X_4, X_5, X_6\}$). 2D box plots show quantiles across 10 repetitions.

C.4. Impact of lazy initialization

As discussed in the main paper, the initialization of the LazyVI procedure plays a significant role in the accuracy of its estimates. Figure 9 shows the distribution of the VI error for dropout, LazyVI with a good initialization, and LazyVI with a random initialization across 10 repetitions. We see that the random initialization results not only in less accurate but also

much more highly variable estimates.

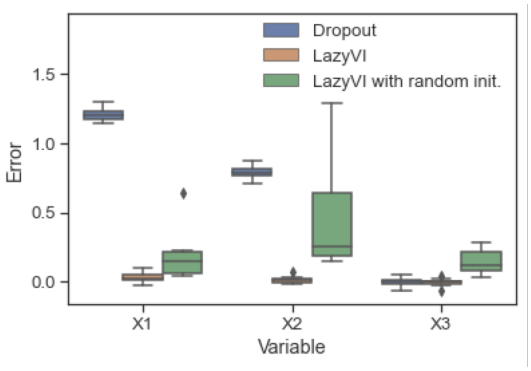

Figure 9. Distribution of VI $-$ $\hat{\text{VI}}$ for the first 3 variables for dropout, LazyVI initialized with the parameters from the full model, and LazyVI with a random initialization.

C.5. Width of training network

Theorem 4.4 implies that LazyVI will perform well when the training network is sufficiently wide. Figure 10 shows the empirical coverage of the 95% confidence intervals defined in (24) (across 40 repetitions) for increasing hidden layer widths. We see that coverage increases as the width of the network increases, but the trade-off is that the computation time for LazyVI also increases with the network width (although remains much faster than retraining).

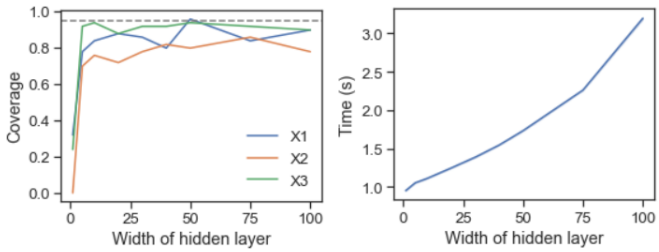

Figure 10. Left: empirical coverage of 95% confidence intervals (across 50 repetitions) of LazyVI estimates for increasing widths of the training network for the important variables. Dotted line shows 95% coverage; Right: average computation time for LazyVI with increasing network widths.

C.6. Additional details for seasonal forecasting experiment

For our real data seasonal precipitation forecasting experiment, we use simulations from the Community Earth System Model-Large Ensemble project (CESM-LENS; Kay et al. (2015); de La Beaujardière et al. (2019)). CESM-LENS is a 40-member ensemble of climate simulations, where the ensemble members all have the same physics but different initial conditions. From this dataset we extracted monthly sea surface temperature (SST) records from 1940-2005 on a $1.25° \times 0.9°$ grid. We compute SST anomalies at each grid point relative to the time period 1950-1989[1] by subtracting the monthly mean and dividing by the monthly standard deviation, and then we linearly detrend each time series.

To compute the 10 ocean climate indices (OCI) used in our experiment, we find the average summer (July-October) monthly SST values of these detrended SST anomalies over specified ocean regions. These regions are well established in the literature; we refer to the supplement from (Chen et al., 2016) to define the boundaries of all OCIs besides NZI, for which we use (Mamalakis et al., 2018). See 1 for the specific boundaries. As a response, we use the average winter (November-March) precipitation over part of the southwestern US (see (Stevens et al., 2021)). We are interested in predicting winter precipitation from the previous summer's SSTs.

[1] https://climatedataguide.ucar.edu/climate-data/nino-sst-indices-nino-12-3-34-4-oni-and-tni

Ocean	OCI	Latitude	Longitude
Pacific	Niño1+2	10°S - 0°	90°W - 80°W
	Niño3	5°S - 5°N	150°W - 90°W
	Niño3.4	5°S - 5°N	170°W - 120°W
	Niño4	5°S - 5°N	160°E - 150°W
	NZI	40°S - 25°S	170°E - 160°W
Atlantic	TNA	5°N - 25°N	55°W - 15°W
	TSA	20°S- 0°	30°W - 10°E
Indian	SWIO	32°S - 25°S	31°E - 45°E
	WTIO	10°S- 10°N	50°E - 70°E
	SETIO	10°S- 0°	90°E - 110°E

Table 1. Ocean climate indices (OCIs) are defined as the average of the detrended SST anomalies across the regions indicated above.

To build more intuition as to why dropout behaves problematically when trying to predict precipitation, we attempt this analysis using linear regression and find that the coefficient estimates are highly unstable. Figure 11 shows the estimated coefficients with their 95% confidence intervals from a multiple linear regression including all variables, along with estimated coefficients from separate simple linear regressions. Note that the coefficient for Niño 3 is highly negative in the full regression, which is offsetting the highly positive coefficient on Niño 3.4; when separated, both of these indices receive much smaller positive coefficients.

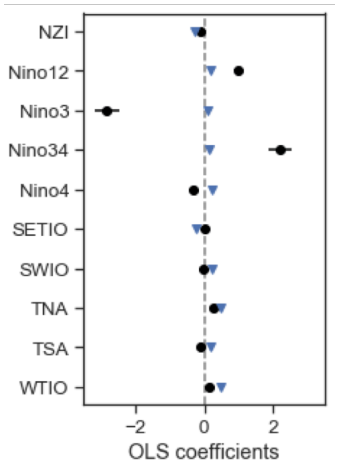

Figure 11. OLS coefficients and standard errors for multiple linear regression with all OCIs (black dots) and simple linear regression with each OCI separately (blue triangles).

	Retrain	Lazy
Time	272.75s	50.27s
Std.	10.8s	1.6s

Table 2. Average time to compute the Shapley values in one data

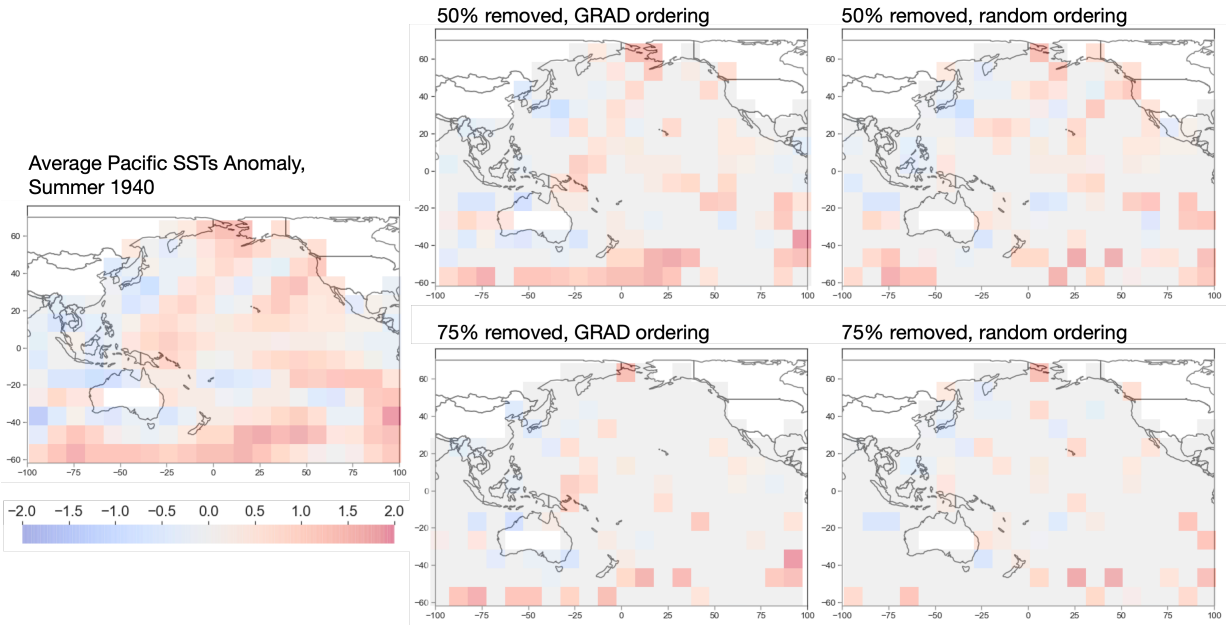

Figure 12. For an example year (1940), fig:saliency shows the original image of SST anomalies, and then the modified images after dropping out 50% and 75% of the data according to the GRAD importance measure and a baseline random order.

C.7. Shapley Value Calculation Using Lazy Training Method

As we have discussed in the paper, our method may also provide a faster alternative when calculating Shapley values using large neural networks. We'll indicate this in the following experiment.

We use the same definition and subset sampling scheme as (Williamson & Feng, 2020). However, instead of retraining the model for each subset of features, we apply our proposed lazy training method to make the computation more scalable and much faster.

We are dealing with a Logistic Model with high dimensional sparse features. Specifically, we have 100 features from a $\mathcal{N}(0, \Sigma_{100\times 100})$, where the variables are independent except $\text{Corr}(X_1, X_2) = 0.75$. The responses are binary, generated from a logistic model: $\log \frac{\mathbb{P}(Y=1)}{1-\mathbb{P}(Y=1)} = X\beta$, where $\beta = (5, 4, 3, 2, 1, 0, \ldots, 0)^\top \in \mathbb{R}^{100}$.

We use a two-layer neural network (with 128 hidden nodes) and the subset sampling scheme proposed by (Williamson & Feng, 2020) when calculating Shapley values. We compare the estimated Shapley values and the computing times when we use the retraining method (as is used in (Williamson & Feng, 2020)) and the lazy training method we proposed respectively. We calculate the Shapley values on 20 simulated datasets, each with a sample size 750. We split each dataset into a training set and test set with sample size 500 and 250 respectively. We train all the models on the training set, then evaluate the predictiveness loss and calculate the Shapley values on the test set. We set the penalty parameter as 100 in accordance to the assumption $\lambda = O(\sqrt{n})$.

In the retraining method we'll reconstruct a two-layer neural network with 128 hidden nodes for each subset of features; In the lazy training method however, after training a two-layer neural network with all the features, we use Algorithm 1 to train the new model on each subset of features (with all features not included in the subset be replaced by their means).

Table 2 gives the average time to calculate the Shapley values for one data set. Lazy training speeds up the calculation by

more than 5 times. In the meanwhile, we don't sacrifice too much on the Shapley value estimation performance. As shown in Figure 13, the Shapley value calculated by lazy training is generally close to the retraining results; for those unimportant variables (Feature id ≥ 6), Shapley values estimated from retraining method has a larger variance, as the sample size is relatively small (training size $n = 500$ while $p = 100$), due to the benefit of the regularization step we have in the lazyVI method.

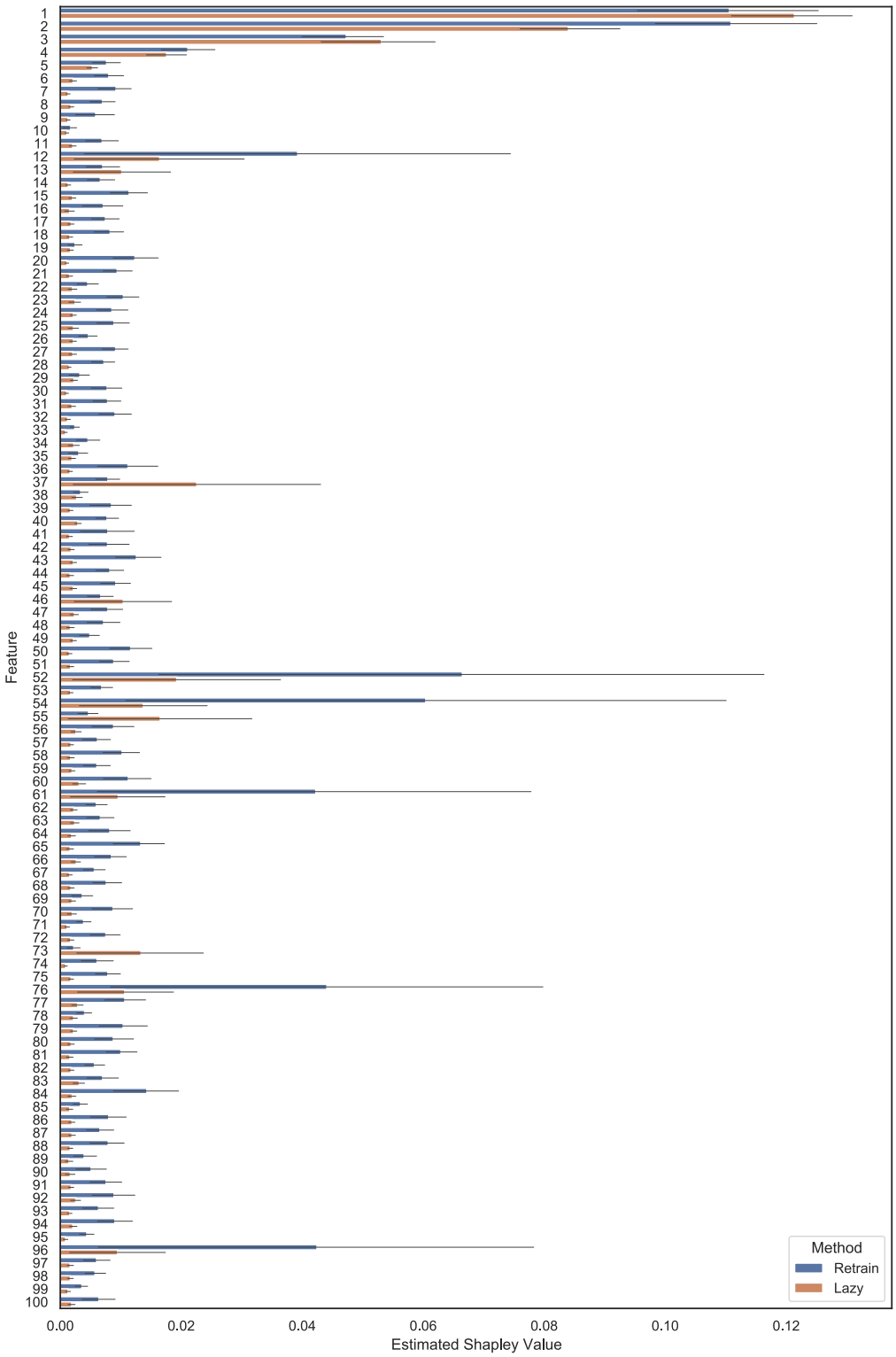

Figure 13. Shapley Values calculated By Retraining vs LazyVI Method using a two-layer neural network. The data is generated from a logistic model, where only the top 5 features are involved and all the left features are non-importance (The corresponding Shapley values are zero). The experiment is repeated 20 times. The colored bars are the averaged estimated Shapley value of each feature using different methods, and the grey lines indicate the standard deviations.