Improving Neural Network Generalization on Data-limited Regression with Doubly-Robust Boosting

Hao Wang

Zhejiang University
haohaow@zju.edu.cn

Abstract

Enhancing the generalization performance of neural networks given limited data availability remains a formidable challenge, due to the model selection trade-off between training error and generalization gap. To handle this challenge, we present a posterior optimization issue, specifically designed to reduce the generalization error of *trained neural networks*. To operationalize this concept, we propose a Doubly-Robust Boosting machine (DRBoost) which consists of a statistical learner and a zero-order optimizer. The statistical learner reduces the model capacity and thus the generalization gap; the zero-order optimizer minimizes the training error in a gradient-free manner. The two components cooperate to reduce the generalization error of a fully trained neural network in a doubly robust manner. Furthermore, the statistical learner alleviates the multicollinearity in the discriminative layer and enhances the generalization performance. The zero-order optimizer eliminates the reliance on gradient calculation and offers more flexibility in learning objective selection. Experiments demonstrate that DRBoost improves the generalization performance of various prevalent neural network backbones effectively.

1 Introduction

Since the development of backward propagation and stochastic gradient descent (SGD) methods (Schneider, Balles, and Hennig 2019; Kingma and Ba 2015), neural network training has greatly benefited from the efficiency due to the implicit dynamic programming mechanism. Despite its widespread success in various fundamental domains, recent work investigates the limitations and bottlenecks of SGD-based algorithm for neural network training (Hinton 2022; Ren et al. 2022).

Building upon these recent advances, the limitations of SGD manifest as three aspects. Firstly, multicollinearity becomes prominent when the parameter count surpasses the sample size. This issue is pertinent in neural networks trained with SGD, where the mini-batch size is often smaller than the parameter count. Such multicollinearity leads to less reliable parameter estimates and increases the network's sensitivity to data noise, ultimately hampering generalization (Alin 2010). Secondly, there is a notable disparity between ground-truth metrics and their surrogate counterparts. Metrics of primary interest in downstream tasks, such as the AUC in ranking tasks and the DCG in retrieval tasks (Li et al. 2023a; Zhou et al. 2022; Wang et al. 2022), are often non-differentiable and thus incompatible with SGD. This necessitates the use of surrogate losses like cross-entropy, which, despite advantages like strong convexity (Rosasco et al. 2004), introduces a surrogate gap and thereby an approximation error to the learning process (van der Hoeven 2020; Marcotte and Savard 1992). Lastly, the model selection trade-off poses a significant challenge, as elucidated by Vapnik (1999). This trade-off is characterized by a seesaw effect: simpler models have smaller generalization gaps but higher training errors, whereas larger models could achieve lower training errors at the cost of larger generalization gaps. This counterbalance complicates model selection and limits model generalization performance, especially in scenarios with low data availability (Wang et al. 2023a; Chu, Rathbun, and Li 2020; Chu et al. 2023b).

In response to the challenges presented above, in this work, we investigate an innovative post-optimization issue: optimizing networks that have been trained with SGD to boost generalization performance. To operationalize this concept, we construct **D**oubly **R**obust **Boost**ing machine (DRBoost), harmonizing statistical learning and zero-order optimization techniques to enhance generalization. The statistical learner is employed to mitigate multicollinearity and reduce the generalization gap by limiting model capacity. However, it inherently precludes gradient computation. Therefore, the zero-order optimizer is utilized as an alternative to SGD, to reduce training error without reliance on gradient calculations. The integration of the two components could mitigate the complexities associated with multicollinearity and surrogate gaps and improve the generalization performance of neural networks. To summarize our contributions:

- We introduce the concept of post-optimization for fully trained networks, a novel paradigm aimed at calibrating SGD-trained neural networks to enhance their capabilities.
- We develop DRBoost, a pioneering approach unifying zero-order optimization and statistical learning techniques to improve neural network generalization. The statistical learner also mitigates multicollinearity issues in neural networks. The zero-order optimizer bypasses the surrogate gap, offering more flexibility in the choice of metrics to optimize.
- We conduct extensive experiments on real-world datasets to verify the efficacy of DRBoost in diverse settings.

2 Preliminary

2.1 Multicollinearity

Multicollinearity, a pervasive issue in statistical modeling, refers to the problem that high correlations among input variables inflate the estimation variance of model parameters (Faraw 2015; Alin 2010). This phenomenon is particularly pronounced in linear models, as elaborated in Theorem 2.1. When the number of parameters exceeds that of input samples, leading to $R_k^2 = 1$ for $1 \leq k \leq \mathrm{K}$, the estimation of parameters is unstable, manifested as $\mathbb{V}(\widehat{w}_k) \to \infty$.

Theorem 2.1. *(Taboga 2021) Consider a linear model $y = \mathbf{x}^\top \mathbf{w} + \epsilon$, where $\mathbf{x} \in \mathbb{R}^{\mathrm{K}}$ represents input features, and $\epsilon \sim \mathcal{N}(0, \sigma)$. For training data $\mathbf{X} \in \mathbb{R}^{\mathrm{M} \times \mathrm{K}}$, if the k-th feature column $\mathbf{x}_k$ has a zero mean, the variance of the ordinary least squares (OLS) estimator for the coefficient is given by*

$$\mathbb{V}\left[\widehat{w}_k \mid \mathbf{X}\right] = \sigma^2 \left(\mathbf{x}_k^\top \mathbf{x}_k\right)^{-1} \frac{1}{1 - \mathrm{R}_k^2}, \tag{1}$$

where R_k^2 is the R-squared that measures the linear correlation between $\mathbf{x}_k$ and other features. Notably, if $\mathrm{M} < \mathrm{K}$, $\mathrm{R}_k^2 = 1$ holds since all feature columns can be well represented with linear combination of other feature columns.

Definition 2.1. *(Neural Network) Let $\mathcal{G}$ be a neural network with L layers, n_l be the number of units in the l-th layer, n_{L} be the number of outputs, $\mathbf{W}^{(\mathrm{L})}$ be the discriminant weights from the L layer to the outputs, $\mathcal{G}^\star$ be a pre-trained $\mathcal{G}$ with SGD. A visualization is provided in Figure 2.*

In neural networks, particularly those with a linear discriminant layer, multicollinearity could pose similar challenges. Considering a L-layer network as defined in Definition 2.1, multicollinearity can exist in the discriminant weights $\mathbf{W}^{(\mathrm{L})}$ when the number of input units in the discriminant layer $n_{\mathrm{L}-1}$ exceeds batch size. At each training step of SGD, such multicollinearity produces infinitely optimum solutions, which leads to insignificant estimation of $\mathbf{W}^{(\mathrm{L})}$ and therefore impeding convergence and reliability of the derived model.

2.2 Model Selection Trade-Off

The model selection trade-off is a pivotal concept in machine learning, which have been examined from various theoretical aspects. In this section, we conceptualize it from the Vapnik–Chervonenkis (VC) perspective, as formulated by Vapnik (1999). We begin by establishing fundamental notations in Definition 2.2.

Definition 2.2. *Consider g as the model selected by a learning algorithm for a statistically large dataset $\mathcal{D}$. Let $E_{\mathrm{in}}(g)$ represent the training error within $\mathcal{D}$, $E_{\mathrm{out}}(g)$ denote the generalization error outside of $\mathcal{D}$, and $\delta_\tau(g) := |E_{\mathrm{out}}(g) - E_{\mathrm{in}}(g)|$ signify the generalization gap that holds with probability $1 - \tau$.*

The generalization error can be dissected into two components: the training error (E_{in}) and the generalization gap (δ). Complex models, characterized by a high VC dimension (d_{vc}), typically exhibit lower training error due to their capacity to fit the data. However, this advantage could be offset by an increased generalization gap due to high d_{vc}. Conversely, simpler models often show larger E_{in} and smaller δ. Consequently, the model selection trade-off can be described as a seesaw effect, where reductions in one aspect (either E_{in} or δ) are counterbalanced by the other, illustrating the delicate trade-off in determining model complexity.

Lemma 2.1. *(Abu-Mostafa et al. 2012) The generalization error of g is bounded as:*

$$P_{\mathcal{D}}[\underbrace{|E_{\mathrm{out}}(g) - E_{\mathrm{in}}(g)|}_{\delta(g)} > \epsilon] \leq 4(2\mathrm{M})^{d_{\mathrm{vc}}} \exp\left(-\frac{1}{8}\epsilon^2 \mathrm{M}\right), \tag{2}$$

where M is the sample size of the dataset $\mathcal{D}$, d_{vc} is the VC-dimension that measures the complexity of the model, and δ is the generalization gap in Definition 2.2. If δ is significantly large, overfitting happens and learning fails.

The seesaw effect challenges existing regularization techniques such as dropout and weight decay. These techniques are designed to mitigate overfitting by reducing the generalization gap δ, but they struggle to simultaneously address the enlarging of the training error E_{in}. Consequently, the reduction of E_{out} cannot be reliably achieved using these regularization methods alone (Zhang et al. 2021).

2.3 Post-Optimization

In this section, we introduce a post-optimization problem, uniquely crafted to explicitly minimize the generalization error of a *pretrained neural network*. Drawing upon the foundations laid out in Definition 2.1 and Definition 2.2, let us consider $\mathcal{G}^\star$ as a pretrained neural network comprising L layers, where E_{in} represents its training error. The objective of the post-optimization problem is to concurrently reduce both the training error E_{in} and the generalization gap $|E_{\mathrm{out}} - E_{\mathrm{in}}|$ of $\mathcal{G}^\star$, ultimately achieving a reduction of the generalization error E_{out} with high probability.

In handling overfitting and the model selection trade-off, contemporary strategies mainly involve lightweight neural architectures and effective regularizers (Zhang et al. 2021). While these strategies effectively reduce model complexity and the generalization gap (δ), they often inadvertently elevate E_{in}. In contrast, the post-optimization issue aims for a simultaneous reduction of both δ and E_{in}, facilitating a doubly robust reduction of E_{out}. Moreover, unlike traditional methods that are integrated with the model training process, post-optimization operates in parallel to model training, which makes it a versatile and flexible practice.

3 Proposed Method

In this section, we develop DRBoost to operationalize the post-optimization issue. Figure 1 illustrates the architecture of DRBoost, which comprises three key components: a pretrained neural network $\mathcal{G}^\star$, a statistical learner $\mathcal{A}_{\mathrm{s}}$, and a zero-order optimizer $\mathcal{A}_{\mathrm{z}}$. The statistical learner and zero-order optimizer collaboratively minimize the E_{out} of neural networks, as depicted by the internal arrows in Figure 1. Furthermore, they cooperate to mitigate critical challenges in neural networks such as multicollinearity, the surrogate gap, and local minima, as shown by the external arrows.

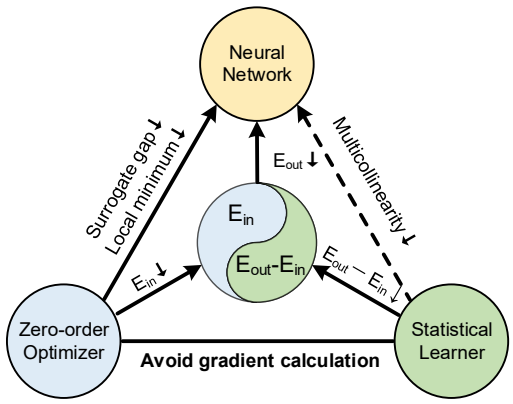

Figure 1: Composition of DRBoost.

3.1 Statistical Learner

Lemma 2.1 conceptualizes the dilemma of model selection and implicitly gives an upper bound of E_{out}. Moving forward, such dilemma also exists in deep neural networks. It manifests itself as a probably approximately correct bound of E_{out} in Theorem 3.1, including the training error E_{in} and the generalization gap δ_τ. In a nutshell, Theorem 3.1 can be proved given Lemma 2.1 by setting $\tau = 4(2\text{M})^{d_{\text{vc}}} \exp\left(-\epsilon^2 \text{M}/8\right)$ and then eliminating τ with ϵ.

Theorem 3.1. *Given $\mathcal{G}$ as a neural network selected by learning the algorithm with the statistical large dataset $\mathcal{D}$, for any tolerance $\tau > 0$, the generalization error of $\mathcal{G}$ is bounded with probability $\geq 1 - \tau$ as*

$$E_{\text{out}}(\mathcal{G}) \leq E_{\text{in}}(\mathcal{G}) + \sqrt{\frac{8}{\text{M}} \ln\left(\frac{4}{\tau}(2\text{M})^{d_{\text{vc}}}\right)} := E_{\text{in}}(\mathcal{G}) + \delta_\tau(\mathcal{G}), \tag{3}$$

where the existence of neural network's VC-dimension has been proved by Baum and Haussler (1988), with $d_{\text{vc}} = \mathcal{O}(\text{N}\log\text{N})$ being the VC-dimension of $\mathcal{G}$; M is the number of samples, N is the number of parameters, δ_τ is the maximum generalization gap in Definition 2.2.

Theorem 3.1 provides a crucial insight into the relationship between the number of parameters N and the VC dimension (d_{vc}). It posits that a reduction in the number of parameters correlates with a lower VC dimension, which in turn leads to a diminished generalization gap. This understanding serves as a foundational principle for leveraging statistical learners with built-in regularization to decrease the model complexity, thereby reducing the generalization gap. Specifically, within a L-layer neural network $\mathcal{G}$, a strategic approach is utilizing a statistical learner to replace the discriminant weights $\mathbf{W}^{(\text{L})}$. It not only leads to a reduction in model complexity, owing to the inherent regularization of the statistical learner, but also effectively tackles the issue of multicollinearity, which is often prevalent in the discriminant layer of neural networks.

One representative statistical learner in regression tasks is the partial least square regression (PLSR) (Rosipal and Krämer 2005), which operates by projecting inputs and labels onto a lower-dimensional subspace where their covariance is maximized. These projections are computed using the NIPALS algorithm (Del Zotto 2013) with complexity of $\mathcal{O}(\text{M} \times n_{\text{L}})$. In the context of DRBoost, PLSR is applied to replace the weight $\mathbf{W}^{(\text{L})}$, projecting the n_{L}-dimensional input to the discriminant layer and the labels onto a subspace with $d_{\text{s}} < n_{\text{L}}$ latent bases. This process reduces the d_{vc} of $\mathcal{G}$, and thus the generalization gap $\delta_\tau(\mathcal{G})$. Moreover, since the projections of PLSR are computed with the full training set, it receives more samples and bypasses multicollinearity.

As posited in Theorem 3.1, if *both* $E_{\text{in}}(\mathcal{G})$ and $\delta_\tau(\mathcal{G})$ are reduced, the upper bound of the generalization error E_{out} will consequently decrease. With the incorporation of PLSR, we have effectively reduced $\delta_\tau(\mathcal{G})$. The remaining challenge is to minimize $E_{\text{in}}(\mathcal{G})$. However, such statistical learners impedes gradient propagation to shallower layers of the neural network, thus obstructing the gradient-based optimization of $E_{\text{in}}(\mathcal{G})$. Therefore, the implementation of gradient-free optimization techniques becomes imperative.

3.2 Zero-Order Optimizer

Zero-order optimization, recognized for its gradient-free nature, has exhibited impressive results across various practical applications (Zhang et al. 2017; Zhu and Jin 2020). Among diverse zero-order optimization techniques, we select metaheuristic optimizers due to their simplicity and parallelization (Yang and Deb 2009). The meta-heuristic optimization can be viewed as an adaptive search for the optimum. It maintains a population $\mathbf{Z}$ of P individuals. In each step, it determines the search direction and step size based on the dynamically evaluated goodness value I_p of each individual.

Optimizing the entire network from scratch using metaheuristic optimizers is computationally prohibitive, given the exponentially growing computational cost with increasing parameters (Gong et al. 2021). Nevertheless, we notice that a wide-enough neural layer has infinite capacity (Yarotsky 2017), and optimizing neural networks layer-by-layer has been a well-established practice (Hinton 2022). These observations motivate our strategy in DRBoost to concentrate on optimizing the weights $\mathbf{W}^{(\text{L}-1)}$ before the discriminant layer, as depicted in Figure 2, which suffices to reduce the training error while keeping computational costs manageable.

Apart from reducing the training error devoid of gradients, metaheuristic optimizers offer two additional benefits for neural network generalization. (1) Their randomness helps in escape from local minima and facilitates a more thorough exploration of the solution space. (2) They provide increased flexibility in the design of optimization metrics. Given independence from gradient calculations, they can directly address non-differentiable and even black-box metrics without introducing surrogate gap.

3.3 Implementation of DRBoost

In this section, we elaborate on the implementation of DRBoost, which is structured around three principal components: (1) Neural network $\mathcal{G}^\star$, pretrained by SGD using a surrogate metric $\hat{\mathcal{L}}$; (2) Metaheuristic optimizer $\mathcal{A}_z$, tasked

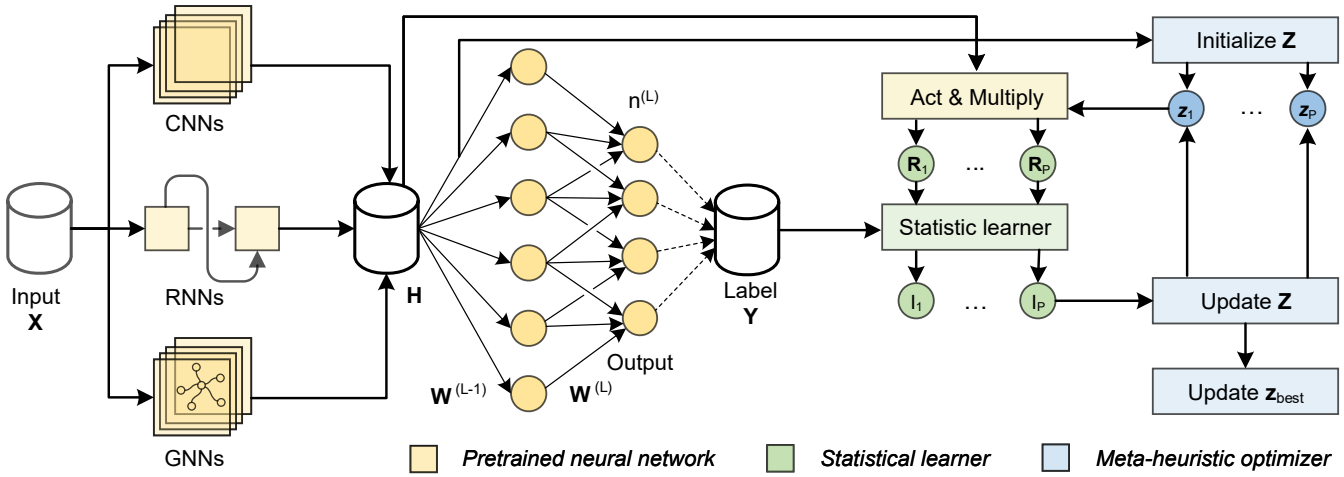

Figure 2: An overview of DRBoost workflow. The intermediate representation of the pretrained network is exported and saved as $\mathbf{H}$. The population $\mathbf{Z}$ in metaheuristic optimizer $\mathcal{A}_{\mathrm{z}}$ is initialized with the parameters $\mathbf{W}^{(\mathrm{L}-1)}$ in the pretrained network. For the p-th individual with state $\mathbf{z}_p$, the statistical learner with d_{s} bases takes $\mathbf{R}_p := \sigma\left(\mathbf{H} * \mathbf{z}_p\right)$ as inputs to forecast the label $\mathbf{Y}$. The forecast quality I_p acts as the individual's goodness value, determining the subsequent update of the population's states $\mathbf{Z}$ and the best individual's state $\mathbf{z}_{\mathrm{best}}$.

Algorithm 1: The workflow of DRBoost.

Input: $\mathbf{X}$: input features; $\mathbf{Y}$: labels; $\mathcal{G}^*$: pretrained neural network with activation function σ.
Parameter: $\mathcal{L}$: metric of interest; d_{s}: number of latent bases for the statistical learner $\mathcal{A}_{\mathrm{s}}$; $\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_{\mathrm{P}}]$: initialized population with P individuals for the heuristic optimizer $\mathcal{A}_{\mathrm{z}}$; T: number of iterations.
Output: boosted model $\mathcal{G}^\dagger$.

1: $\mathbf{H} \leftarrow \mathrm{generate}\left(\mathcal{G}^\star, \mathbf{X}\right)$.
2: $\mathbf{Z} \leftarrow \mathrm{initialize}(\mathbf{W}^{(\mathrm{L}-1)})$
3: **for** $t \in [0, \mathrm{T}]$ **do**
4: **for** $p \in [0, \mathrm{P}]$ **do**
5: $\mathbf{R}_p \leftarrow \sigma\left(\mathbf{H} * \mathbf{z}_p\right)$.
6: $\mathbf{s}_p \leftarrow \mathcal{A}_{\mathrm{s}}.\mathrm{fit}\left(\mathbf{R}_p, \mathbf{Y}, d_{\mathrm{s}}\right)$.
7: $I_p \leftarrow \mathcal{L}\left(\mathcal{A}_{\mathrm{s}}.\mathrm{predict}\left(\mathbf{R}_p, \mathbf{s}_p\right), \mathbf{Y}\right)$.
8: **end for**
9: $\mathbf{Z} \leftarrow \mathcal{A}_{\mathrm{z}}.\mathrm{update}\left(\{I_p\}_{p=1}^{\mathrm{P}}, \mathbf{Z}\right)$
10: $\mathbf{z}_{\mathrm{best}} \leftarrow \mathcal{A}_{\mathrm{z}}.\mathrm{updateBest}\left(\{I_p\}_{p=1}^{\mathrm{P}}, \mathbf{Z}, \mathbf{z}_{\mathrm{best}}\right)$
11: **end for**
12: $\mathbf{s} \leftarrow \mathcal{A}_{\mathrm{s}}.\mathrm{fit}\left(\sigma(\mathbf{H} * \mathbf{z}_{\mathrm{best}}), \mathbf{Y}, d_{\mathrm{s}}\right)$.
13: $\mathcal{G}^\dagger \leftarrow \mathrm{derive}\left(\mathcal{G}^\star, \mathbf{z}_{\mathrm{best}}, \mathbf{s}\right)$.

with directly optimizing the target metric $\mathcal{L}$; (3) Statistical learner $\mathcal{A}_{\mathrm{s}}$, equipped with d_{s} latent bases for regularization.

The optimization procedure of DRBoost is outlined in Algorithm 1. Initially, the intermediate representation of $\mathcal{G}^\star$, as shown in Figure 2, is denoted as $\mathbf{H}$ (steps 1). A population $\mathbf{Z}$ comprising P individuals is then randomly initialized, with one individual set as the pretrained weights to facilitate fast convergence, i.e., $\mathbf{z}_1 = \mathbf{W}^{(\mathrm{L}-1)}$ (step 2). Subsequently, the metaheuristic optimizer $\mathcal{A}_{\mathrm{z}}$ optimizes this population with respect to the metric $\mathcal{L}$ directly on the training set (steps 3-10). For each individual $\mathbf{z}_p$, the weight $\mathbf{s}_p$ calculated by $\mathcal{A}_{\mathrm{s}}$ maps $\mathbf{R}_p$ to $\mathbf{Y}$ (step 5-6), and its mapping quality is assessed using $\mathcal{L}$ on the training data, yielding a goodness value I_p (step 7). The population $\mathbf{z}_p$ and the best-performing individual $\mathbf{z}_{\mathrm{best}}$ are updated by $\mathcal{A}_{\mathrm{z}}$ based on calculated goodness values $\{I_p\}_{p=1}^{\mathrm{P}}$ (steps 9-10). Finally, the weights $\mathbf{W}^{(\mathrm{L})}$ and $\mathbf{W}^{(\mathrm{L}-1)}$ in $\mathcal{G}^\star$ are replaced with $\mathbf{s}$ and $\mathbf{z}_{\mathrm{best}}$, respectively, resulting in the boosted neural network $\mathcal{G}^\dagger$ (steps 12-13).

A crucial hyperparameter in this process is the capacity of the statistical learner ($\mathcal{A}_{\mathrm{s}}$), e.g., the number of latent bases d_{s} in PLSR. A large d_{s} may leave the generalization gap δ inadequately controlled, whereas a small d_{s} might result in a huge training error E_{in}. thereby making it challenging for $\mathcal{A}_{\mathrm{z}}$ to identify a model that surpasses $\mathcal{G}^\star$ in training performance.

DRBoost is a model-agnostic framework that supports most neural backbones. The primary requirement for integrating DRBoost is the ability to substitute the discriminant layer of these architectures with appropriate statistical learners. For instance, an affine layer in the network can be replaced with a linear regressor, and a softmax layer can be substituted with a support vector machine. Such flexible compatibility facilitates its seamless integration into various applications.

3.4 Connections to Current Methods

Fine-tuning. Fine-tuning, a pivotal technique in transfer learning (Yosinski et al. 2014), is employed to adapt models trained on a source domain for effective application in a target domain (Liu et al. 2021, 2022). One significant challenge with conventional fine-tuning is the propensity for overfitting, especially when the target domain has a limited number of samples (Mou et al. 2016). DRBoost introduces an innovative approach to fine-tuning uniquely designed to address these issues. Moreover, it does not rely on gradient calculation

Dataset	# variables	# samples
METR-LA	207	34,272
PEMSBAY	325	52,116
PEMS04	304	16,992
TRAFFIC	862	17,544
ELECTRICITY	321	26,304

Table 1: Dataset description.

for optimization, which allows it to finetune models directly towards the specific metrics that are crucial in downstream tasks, such as AUC in ranking tasks and DCG in retrieval tasks (Li, Zheng, and Wu 2023; Zhou et al. 2022). Consequently, it eliminates the need for differentiable surrogate objectives, effectively avoiding the surrogate gap issue.

Ensemble Learning. According to Section 3.3, massive experiments are required to find the best d_{s}. The candidate models in this process are wasted, and we cannot theoretically guarantee that the final selected candidate has the best generalization than the other candidates. Nevertheless, ensemble learning can unify these candidates to further improve generalization with limited computational effort.

4 Experiments

To demonstrate the efficacy of DRBoost, a plugin for improving the generalization performance of pretrained neural networks, the aspects below deserve empirical investigation.

- **Performance:** *Does DRBoost work?* We report the performance gain brought by DRBoost to MLP in Section 4.2.
- **Gains:** *Why does it work?* We deconstruct various aspects of DRBoost to identify the gain sources in Section 4.3.
- **Sensitivity:** *Is it sensitive to hyperparameters and specification of zero-order optimizers?* We report performance given different number of latent features in Section 4.4 and various metaheuristic optimizers in Section 4.2.
- **Generality:** *Does it support other backbones?* We verify the efficacy of DRBoost for mainstream backbones, such as GNNs, RNNs, and CNNs in Section 4.5.

4.1 Setup

Datasets. In this study, we validate the efficacy of DRBoost in regression tasks. Time series forecasting, distinct from other regression tasks, presents a suitable setting for this validation, due to its moderate sample scale, diverse developed neural architectures (Wu et al. 2019; Bai, Kolter, and Koltun 2018), and wide applications (Wang et al. 2023b; Chen et al. 2023). We utilize well-known public benchmarks in this domain (Chen et al. 2001; Li et al. 2018; Lai et al. 2018), with relevant dataset statistics provided in Table 1. The datasets are partitioned into training, validation, and testing sets in a 0.7:0.15:0.15 ratio. Prior to model training, we employ a min-max scaling technique to normalize the data, ensuring consistent data range across different datasets.

Pretraining protocol. We train all neural models for 200 epochs with Adam optimizer, using mean squared error (MSE) as a surrogate loss functions. All models are trained with the same set of hyper parameters to make the results comparable. Specifically, we set learning rate to 0.001 and other hyperparameters consistent with Kingma and Ba (2015). We checkpoint the performance on validation data every 5 epochs and save the best model $\mathcal{G}^\star$ for post-optimization.

Post-optimization protocol. The statistical learner is implemented with PLSR (Del Zotto 2013) to reduce the model complexity and thus the generalization gap δ_τ. E_{in} is measured with the R-squared (R^2), which is the metric that we really care about. To avoid overclaiming, we perform Algorithm 1 with fixed $d_{\mathrm{s}} = 48$ and disable the ensemble trick in Section 3.4. For zero-order optimization, we implement it using metaheuristic optimizers, with hyperparameters tuned to optimize the performance on the validation set.

4.2 Overall Performance

To evaluate the efficacy in standard regression task, we select a multilayer perceptron (MLP) with 64-64-64-1 neurons as the backbone model and report the results in Table 2. We implement the statistical learner with PLSR, the zero-order optimization with particle swarm optimization (PSO) (Esmin, Coelho, and Matwin 2015). We also explore the efficacy of other zero-order optimizers: cuckoo search (CS) (Yang and Deb 2009) and modified beetle antennae search (BAS) (Jiang and Li 2017). To summarize the main observations:

- DRBoost offers notable performance enhancement, especially in datasets that are considered "difficult", as exemplified by the TRAFFIC dataset (with the most variables but limited samples). This indicates DRBoost's efficacy in complex, data-scarce regression scenarios. Conversely, for "easy" datasets where baseline models like the MLP already show strong performance, DRBoost's improvements are comparatively modest. An exemplar "easy" dataset is PEMSBAY (with the largest sample size).
- DRBoost's efficacy transcends dependence on specific zero-order optimizers due to its "doubly robustness". As delineated in Theorem 3.1, a reduction in either the training error (E_{in}) or the generalization gap (δ_τ), without increasing the other, leads to a probable reduction in the generalization error (E_{out}). Consequently, given that statistical learners are adept at narrowing the generalization gap, the role of the zero-order optimizer is simplified to ensuring the training error remains at or below that of the pre-trained network ($\mathcal{G}^\star$), a target attainable without the necessity for intricate implementation or tuning of zero-order optimizers.

4.3 Ablation Study

In this section, we ablate individual components of DRBoost: the statistical learner $\mathcal{A}_{\mathrm{s}}$ and the zero-order optimizer $\mathcal{A}_{\mathrm{z}}$, to discern their contributions to overall performance. The results are detailed in Table 4. To summarize the main observations:

- The exclusion of $\mathcal{A}_{\mathrm{s}}$ leads to obvious performance drop. This decline is attributed to the inability to reduce the generalization gap δ_τ in its absence, resulting in an uncontrolled upper bound of E_{out}, as discussed in Section 3.1.

Methods	Base			PSO			BAS			CS		
Datasets	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE
METR-LA	0.945	0.048	0.079	0.955	0.041	0.071	0.951	0.044	0.075	0.956	0.041	0.071
PEMSBAY	0.949	0.016	0.032	0.951	0.016	0.031	0.951	0.016	0.031	0.950	0.016	0.032
PEMS04	0.889	0.043	0.057	0.897	0.040	0.054	0.893	0.041	0.056	0.893	0.041	0.056
TRAFFIC	0.835	0.040	0.062	0.849	0.037	0.059	0.845	0.038	0.060	0.853	0.037	0.059
ELECTRICITY	0.828	0.032	0.041	0.845	0.030	0.038	0.840	0.031	0.039	0.852	0.029	0.038

Table 2: Performance of DRBoost based on pretrained MLPs (Base). PSO, CS and BAS refer to metaheuristic optimizers.

Backbone		FC-LSTM		GRU		LSTNET		TCN		WaveNet		GraphWaveNet	
Datasets	DRBoost	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2
METR-LA	✗	0.041	0.958	0.040	0.958	0.041	0.957	0.030	0.975	0.043	0.949	0.030	0.966
	✓	0.037	0.961	0.035	0.961	0.037	0.960	0.030	0.964↓	0.039	0.952	0.030	0.966
PEMSBAY	✗	0.019	0.938	0.019	0.933	0.021	0.906	0.013	0.962	0.020	0.921	0.013	0.957
	✓	0.018	0.943	0.017	0.944	0.021	0.907	0.014↓	0.962	0.021↓	0.901	0.013	0.958
PEMS04	✗	0.042	0.876	0.041	0.887	0.043	0.875	0.060	0.786	0.039	0.892	0.202	-1.241
	✓	0.042	0.879	0.041	0.889	0.041	0.886	0.040	0.897	0.039	0.895	0.045	0.868

Table 3: Performance improvements of DRBoost for other network architectures. "↓" highlights performance drop.

- Omitting $\mathcal{A}_z$ also hampers model generalization, underscoring the role of zero-order optimizers to minimize E_{in}, as elaborated in Section 3.2. Notably, ablating $\mathcal{A}_z$ removes the randomness brought by metaheuristic optimization, which makes the results identical in different runs.

4.4 Additional Discussion on Statistical Learner

The statistical learner plays a critical role in DRBoost to control generalzation gap. To further verify its efficacy, we select PLSR as statistical learner, performing DRBoost on a pretrained MLP with different regularization strengths (i.e., number of latent bases d_{s}). The results are summarized in Figure 3. To summarize the main observations:

- DRBoost significantly improves the performance of the original model $\mathcal{G}^\star$ in a wide range of d_{s}. Models with best performance have similar d_{s}. For example, on the TRAFFIC dataset, the numbers of latent bases where $R^2 > 0.85$ holds are concentrated in the interval $30 \leq d_{\text{s}} \leq 46$.
- Small d_{s} makes poor performance, evidenced by the performance drop when $d_{\text{s}} \leq 4$. This is because strong regularization hinders the fitness on the training data. Although the generalization gap is narrowed, $E_{\text{in}}(\mathcal{G}^\dagger)$ is increased, and the generalization error $E_{\text{out}}(\mathcal{G}^\dagger)$ is thus uncontrolled. Similarly, large d_{s} also leads to suboptimal performance. For example, the median values of R^2 is less than 0.85 for most models with $d_{\text{s}} \geq 48$. This is because d_{s} is positively monotone with the VC-dimension d_{vc} of $\mathcal{G}^\dagger$ in Theorem 3.1. Therefore, large d_{s} enlarges the generalization gap $\delta_\tau(\mathcal{G})$ and thus $E_{\text{out}}(\mathcal{G}^\dagger)$. Experiments on the ELECTRICITY dataset show similar patterns.

4.5 Efficacy on Neural Backbones

In contrast to canonical regression tasks considering independent and identically distributed (i.i.d.) samples, real-world tasks further consider inter-sample dependencies. In particular, time series illusrates high inter-sample dependencies, which motivates diverse neural backbones to encode such dependencies: RNNs (Sutskever, Vinyals, and Le 2014; Lai et al. 2018; Cho et al. 2014), CNNs (Bai, Kolter, and Koltun 2018; van den Oord et al. 2016), and GNNs (Wu et al. 2019). In this section, we evaluate the efficacy of DRBoost for supporting them. Following Lai et al. (2018), we employ a 168-length input sequence for one-step single variable prediction.

According to Table 3, DRBoost improves performance of diverse neural backbones. Notably, it effectively calibrates the networks that initially exhibit poor performance due to overfitting (e.g., GraphWaveNet on PEMS04). However, it is important to acknowledge the presence of some instances without performance improvements, as seen in the case of WaveNet's underperformance on PEMSBAY. These exceptions are consistent with the theoretical underpinnings of DRBoost in Theorem 3.1. The theorem posits that DRBoost reduces the upper bound of the generalization error which is *probably approximately correct*, i.e., while DRBoost is generally effective, it does not guarantee that the boosted models always outperform their original counterparts.

5 Related Works

While the specific problem of post-optimization in neural networks remains relatively unexplored, the broader concept of integrating zero-order optimizers and statistical learners into deep learning has garnered considerable interest in the AI community. We offer an overview and discern with them.

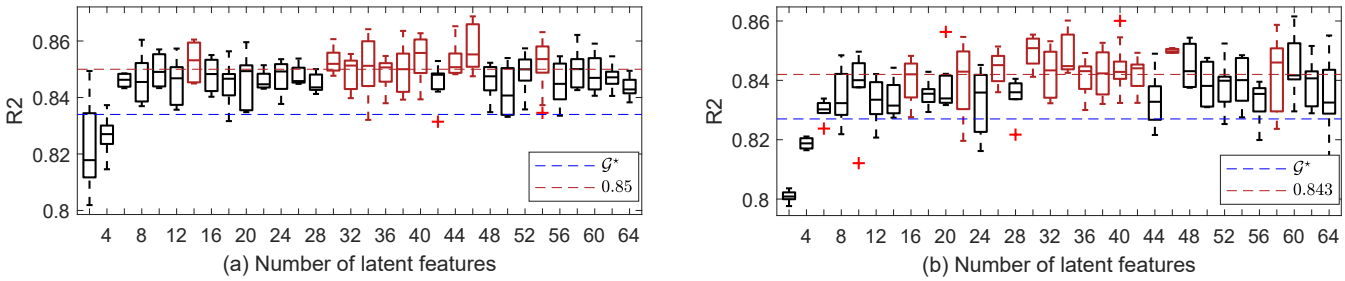

Figure 3: Performance (R^2) of DRBoost with different numbers of latent bases on TRAFFIC (a) and ELECTRICITY (b). Blue lines indicate the R^2 of the model without DRBoost. Models with relatively high performance are marked in red.

		ELECTRICITY		METR-LA		PEMS04		PEMSBAY	
$\mathcal{A}_s$	$\mathcal{A}_z$	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2
✗	✓	$0.031_{\pm 0.001}$	$0.836_{\pm 0.009}$	$0.043_{\pm 0.003}$	$0.953_{\pm 0.004}$	$0.041_{\pm 0.000}$	$0.893_{\pm 0.001}$	$0.016_{\pm 0.000}$	$0.947_{\pm 0.002}$
✓	✗	$0.031_{\pm 0.000}$	$0.838_{\pm 0.000}$	$0.050_{\pm 0.000}$	$0.944_{\pm 0.000}$	$0.041_{\pm 0.000}$	$0.893_{\pm 0.000}$	$0.016_{\pm 0.000}$	$0.949_{\pm 0.000}$
✓	✓	$\mathbf{0.030}_{\pm 0.000}$	$\mathbf{0.844}_{\pm 0.007}$	$\mathbf{0.040}_{\pm 0.001}$	$\mathbf{0.955}_{\pm 0.001}$	$\mathbf{0.040}_{\pm 0.001}$	$\mathbf{0.897}_{\pm 0.000}$	$\mathbf{0.016}_{\pm 0.001}$	$\mathbf{0.950}_{\pm 0.001}$

Table 4: Ablation study of DRBoost based on the MLP neural backbone. The best results are bolded.

Statistical Learner and Neural Network Methods integrating statistical learners with neural networks can be categorized based on the positioning of the statistical learners. The first approach applies statistical learners prior to neural network processing to reduce noise and redundancy of data. In this approach, raw features are transformed into latent variables by statistical learners such as principal component analysis (Varkeshi et al. 2020), partial least squares regression (Putro, Saputro, and Imawan 2018), and independent component analysis (Alzoubi et al. 2018), before being input into the neural network. The second approach utilizes statistical learning after neural network processing, employing neural networks for feature extraction and statistical models for discrimination (Chen et al. 2018; Sun et al. 2017). While these methods have shown promise in practice, they overlook the role of statistical learners in handling challenges of multicollinearity and generalization in neural networks.

Zero-order Optimizer and Neural Network Zero-order optimizers, known for their gradient-free approach, offer an alternative optimization strategy. This approach bypasses several drawbacks associated with SGD, such as susceptibility to local minima and issues related to the surrogate gap. Despite their potential, the application of zero-order optimizers remains mainly in the context of simple and shallow neural network structures (Sun, Yen, and Yi 2019). A notable implementation in this domain is the evolutionary extreme learning machine (Zhu et al. 2005), essentially a two-layer neural network. In this model, the first layer undergoes optimization via a zero-order method, while the second layer is conventionally trained. This approach has demonstrated promising results in specific benchmarks (Wong et al. 2018; Deng, Han, and Zhao 2020). However, its efficacy on larger datasets is questioned due to limited model capacity (Delgado et al. 2014).

Applying zero-order optimization to deep neural networks presents a significant challenge, primarily due to the vast parameter space inherent in these networks (Gong et al. 2021). Blocked to this limitation, it is more prevalent to use zero-order optimizers for hyperparameter search (Zhu and Jin 2020; Zhang et al. 2017; Sun, Yen, and Yi 2019) and neural architecture design (Real et al. 2019; Liu et al. 2018a,b; Real et al. 2017; Xie and Yuille 2017). Nevertheless, employing zero-order optimization for network training could provide additional benefits, e.g., flexibility in metric design and handling of local minima, which shows considerable promise.

6 Conclusion

This paper introduces an innovative post-optimization problem, aimed at explicitly enhancing the generalization of neural networks while addressing the limitations inherent in standard SGDs. To operationalize this concept, we develop DRBoost, a framework that integrates statistical learners and metaheuristic optimizers to increase deep neural network performance. DRBoost effectively mitigates limitations of pure SGD such as multicollinearity, surrogate gap and local minimum, and improves generalization of neural networks. The efficacy is verified in time-series forecasting applications.

Limitation & Future works. One significant limitation is the complexity of the statistical learner with respect to the size of the dataset, which poses challenges in deploying DRBoost in large-scale data. However, this limitation can be alleviated by involving parallel computation technologies, a prospect that we aim to explore in future research. Meanwhile, it is essential to extend the application of DRBoost to more real-world scenarios characterized by low data availability, such as policy evaluation (Li et al. 2023b), treatment effect estimation (Chu et al. 2023a) and drug discovery, where the model selection trade-off is quite critical yet delicate.

Acknowledgements

This work is supported by National Key R&D Program of China (Grant No. 2021YFC2101100), National Natural Science Foundation of China (62073288, 12075212, 12105246, 11975207) and Zhejiang University NGICS Platform.

References

Abu-Mostafa; S, Y.; Magdon-Ismail; Malik; and Lin, H.-T. 2012. *Learning from data*, volume 4. AMLBook New York, NY, USA:.

Alin, A. 2010. Multicollinearity. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(3): 370–374.

Alzoubi, I.; Delavar, M. R.; Mirzaei, F.; and Nadjar Arrabi, B. 2018. Comparing ANFIS and integrating algorithm models for prediction of energy consumption for irrigation land leveling. *Geosystem Engineering*, 21(2): 81–94.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271.

Baum, E. B.; and Haussler, D. 1988. What Size Net Gives Valid Generalization? In *NIPS*, 81–90.

Chen, C.; Petty, K.; Skabardonis, A.; Varaiya, P.; and Jia, Z. 2001. Freeway performance measurement system: mining loop detector data. *Transp. Res. Rec.*, 1748(1): 96–102.

Chen, Y.; Tao, G.; Ren, H.; Lin, X.; and Zhang, L. 2018. Accurate seat belt detection in road surveillance images based on CNN and SVM. *Neurocomputing*, 274: 80–87.

Chen, Z.; Ding, L.; Chu, Z.; Qi, Y.; Huang, J.; and Wang, H. 2023. Monotonic Neural Ordinary Differential Equation: Time-series Forecasting for Cumulative Data. In *CIKM*, 4523–4529. ACM.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 1724–1734.

Chu, Z.; Huang, J.; Li, R.; Chu, W.; and Li, S. 2023a. Causal Effect Estimation: Recent Advances, Challenges, and Opportunities. *CoRR*, abs/2302.00848.

Chu, Z.; Li, R.; Rathbun, S. L.; and Li, S. 2023b. Continual Causal Inference with Incremental Observational Data. In *ICDE*, 3430–3439. IEEE.

Chu, Z.; Rathbun, S. L.; and Li, S. 2020. Matching in Selective and Balanced Representation Space for Treatment Effects Estimation. In *CIKM*, 205–214. ACM.

Del Zotto, S. 2013. *The PLS regression model: algorithms and application to chemometric data*. Ph.D. thesis, University of Udine.

Delgado, M. F.; Cernadas, E.; Barro, S.; and Amorim, D. G. 2014. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1): 3133–3181.

Deng, C.; Han, Y.; and Zhao, B. 2020. High-Performance Visual Tracking With Extreme Learning Machine Framework. *IEEE Trans. Cybern.*, 50(6): 2781–2792.

Esmin, A. A. A.; Coelho, R. A.; and Matwin, S. 2015. A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. *Artif. Intell. Rev.*, 44(1): 23–45.

Faraw, J. J. 2015. *Practical Regression and ANOVA using R*.

Gong, M.; Liu, J.; Qin, A. K.; Zhao, K.; and Tan, K. C. 2021. Evolving Deep Neural Networks via Cooperative Coevolution With Backpropagation. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1): 420–434.

Hinton, G. 2022. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*.

Jiang, X.; and Li, S. 2017. BAS: Beetle Antennae Search Algorithm for Optimization Problems. arXiv:1710.10724.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Lai, G.; Chang, W.; Yang, Y.; and Liu, H. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *SIGIR*, 95–104.

Li, H.; Wu, K.; Zheng, C.; Xiao, Y.; Wang, H.; Geng, Z.; Feng, F.; He, X.; and Wu, P. 2023a. Removing Hidden Confounding in Recommendation: A Unified Multi-Task Learning Approach. In *NeurIPS*.

Li, H.; Zheng, C.; Cao, Y.; Geng, Z.; Liu, Y.; and Wu, P. 2023b. Trustworthy Policy Learning under the Counterfactual No-Harm Criterion. In *ICML*, volume 202, 20575–20598.

Li, H.; Zheng, C.; and Wu, P. 2023. StableDR: Stabilized Doubly Robust Learning for Recommendation on Data Missing Not at Random. In *ICLR*.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR*.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2018a. Hierarchical Representations for Efficient Architecture Search. In *ICLR*.

Liu, J.; Gong, M.; Miao, Q.; Wang, X.; and Li, H. 2018b. Structure Learning for Deep Neural Networks Based on Multiobjective Optimization. *IEEE Trans. Neural Networks Learn. Syst.*, 29(6): 2450–2463.

Liu, W.; Su, J.; Chen, C.; and Zheng, X. 2021. Leveraging distribution alignment via stein path for cross-domain cold-start recommendation. *NeurIPS*, 19223–19234.

Liu, W.; Zheng, X.; Hu, M.; and Chen, C. 2022. Collaborative Filtering with Attribution Alignment for Review-based Non-overlapped Cross Domain Recommendation. In *WWW*, 1181–1190. ACM.

Marcotte, P.; and Savard, G. 1992. Novel approaches to the discrimination problem. *ZOR Methods Model. Oper. Res.*, 36(6): 517–545.

Mou, L.; Meng, Z.; Yan, R.; Li, G.; Xu, Y.; Zhang, L.; and Jin, Z. 2016. How Transferable are Neural Networks in NLP Applications? In *EMNLP*, 479–489.

Putro, I. S.; Saputro, A. H.; and Imawan, C. 2018. Electrical Conductivity Prediction System of Honey using Hyperspectral Imaging. In *ISRITI*, 487–491.

Real, E.; Aggarwal, A.; Huang, Y.; and V.Le, Q. 2019. Regularized evolution for image classifier architecture search. In *AAAI*, 4780–4789.

Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y. L.; Tan, J.; Le, Q. V.; and Kurakin, A. 2017. Large-scale evolution of image classifiers. In *ICML*, 2902–2911.

Ren, M.; Kornblith, S.; Liao, R.; and Hinton, G. 2022. Scaling forward gradient with local losses. *arXiv preprint arXiv:2210.03310*.

Rosasco, L.; Vito, E. D.; Caponnetto, A.; Piana, M.; and Verri, A. 2004. Are Loss Functions All the Same? *Neural Comput.*, 16(5): 1063–1076.

Rosipal, R.; and Krämer, N. 2005. Overview and recent advances in partial least squares. In *SLSFS*, 34–51.

Schneider, F.; Balles, L.; and Hennig, P. 2019. DeepOBS: A Deep Learning Optimizer Benchmark Suite. arXiv:1903.05499.

Sun, X.; Park, J.; Kang, K.; and Hur, J. 2017. Novel hybrid CNN-SVM model for recognition of functional magnetic resonance images. In *SMC*, 1001–1006.

Sun, Y.; Yen, G. G.; and Yi, Z. 2019. Evolving Unsupervised Deep Neural Networks for Learning Meaningful Representations. *IEEE Trans. Evol. Comput.*, 23(1): 89–103.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*, 3104–3112.

Taboga, M. 2021. *Lectures on probability theory and mathematical statistics*. Kindle Direct Publishing.

van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. W.; and Kavukcuoglu, K. 2016. WaveNet: A Generative Model for Raw Audio. In *ISCA*, 125.

van der Hoeven, D. 2020. Exploiting the Surrogate Gap in Online Multiclass Classification. In *NeurIPS*.

Vapnik, V. 1999. *The nature of statistical learning theory*. Springer science & business media.

Varkeshi, M. B.; Mohammadi, K.; Kisi, Ö.; and Fasihi, R. 2020. A new wavelet conjunction approach for estimation of relative humidity: wavelet principal component analysis combined with ANN. *Neural Comput. Appl.*, 32(9): 4989–5000.

Wang, H.; Chang, T.; Liu, T.; Huang, J.; Chen, Z.; Yu, C.; Li, R.; and Chu, W. 2022. ESCM2: Entire Space Counterfactual Multi-Task Model for Post-Click Conversion Rate Estimation. In *SIGIR*, 363–372.

Wang, H.; Chen, Z.; Fan, J.; Li, H.; Liu, T.; Liu, W.; Dai, Q.; Wang, Y.; Dong, Z.; and Tang, R. 2023a. Optimal Transport for Treatment Effect Estimation. In *NeurIPS*.

Wang, H.; Wang, Z.; Niu, Y.; Liu, Z.; Li, H.; Liao, Y.; Huang, Y.; and Liu, X. 2023b. An Accurate and Interpretable Framework for Trustworthy Process Monitoring. *IEEE Trans. Artif. Intell.*

Wong, C.; Vong, C.; Wong, P.; and Cao, J. 2018. Kernel-Based Multilayer Extreme Learning Machines for Representation Learning. *IEEE Trans. Neural Networks Learn. Syst.*, 29(3): 757–762.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI*, 1907–1913.

Xie, L.; and Yuille, A. 2017. Genetic CNN. In *ICCV*, 1379–1388.

Yang, X.; and Deb, S. 2009. Cuckoo search via Levy flights. In *NABIC*, 210–214.

Yarotsky, D. 2017. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94: 103–114.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *NIPS*, 3320–3328.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2021. Understanding Deep Learning (Still) Requires Rethinking Generalization. *Commun. ACM*, 64(3): 107–115.

Zhang, C.; Lim, P.; Qin, A. K.; and Tan, K. C. 2017. Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. *IEEE Trans. Neural Networks Learn. Syst.*, 28(10): 2306–2318.

Zhou, K.; Yu, H.; Zhao, W. X.; and Wen, J. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *WWW*, 2388–2399. ACM.

Zhu, H.; and Jin, Y. 2020. Multi-Objective Evolutionary Federated Learning. *IEEE Trans. Neural Networks Learn. Syst.*, 31(4): 1310–1322.

Zhu, Q.; Qin, A. K.; Suganthan, P. N.; and Huang, G. 2005. Evolutionary extreme learning machine. *Pattern recognit.*, 38(10): 1759–1763.