OPT2025: 17th Annual Workshop on Optimization for Machine Learning

Data-Aware Training Quality Monitoring and Certification for Deep Learning

Farhang Yeganegi FYEGAN2@UIC.EDU
Arian Eamaz AEAMAZ2@UIC.EDU
Mojtaba Soltanalian MSOL@UIC.EDU
University of Illinois Chicago, Chicago, Illinois, USA

Abstract

Deep learning models achieve remarkable representation power, yet their black-box nature raises concerns in high-stakes applications. While generalization analysis is well studied, less attention has been given to certifying the training process itself. We introduce the YES training quality bounds, a framework for real-time, data-aware certification and monitoring of neural network training. The bounds assess data utilization and optimization dynamics, revealing issues such as loss plateaus in suboptimal regions. Validated on synthetic and real data across classification and denoising tasks, YES bounds reliably certify training quality. Integrated with a color-coded cloud monitoring system, they provide a practical tool for real-time evaluation, setting a new standard for training quality assurance in deep learning.

1. Introduction

Deep learning models are central to solving complex computational problems due to the rich representations formed through layered structures and nonlinear transformations [2, 3]. Yet, their training processes remain opaque, raising concerns about transparency and reliability, particularly in high-stakes applications. Ensuring that neural networks (NNs) are effectively trained is therefore critical.

Existing approaches often rely on statistical analyses under restrictive assumptions. For instance, work on ReLU-based NNs has characterized memory capacity [9] and shown that, in the over-parameterized regime, stochastic gradient descent (SGD) converges to global minima [5]. Other studies focus on convergence diagnostics, typically by adapting solver parameters such as learning rates. Examples include step-size adjustment during stationary phases [6] and theoretical guarantees for SGD in over-parameterized NNs [1]. While these methods offer valuable insights, they are solver-specific, statistical in nature, and primarily address convergence speed rather than training quality.

A key challenge in optimization is the absence of prior knowledge about the true minimum of the training objective. This motivates the need for principled methods to assess training progress without relying on assumptions about the optimum. We introduce a framework for *data-aware monitoring* of deep neural network (DNN) training, called the *YES training bounds*. The central question is: **Is the network being properly trained by the data and optimizer (YES or NO)?** Unlike traditional diagnostics, YES bounds provide deterministic, solver-agnostic, and real-time evaluation by comparing optimizer trajectories against a simple linear baseline. This yields concrete, data-specific guarantees on loss reduction, enabling a principled test of whether meaningful learning is occurring.

Our approach complements, but is distinct from, cryptographic frameworks for reproducibility such as [7], which log training steps for external verification. While such methods ensure integrity, YES bounds certify progress by assessing optimization quality in real time. We further propose a cloud-based monitoring system that visualizes training effectiveness through a color-coded scheme—**red** (ineffective), **yellow** (non-optimal), and **green** (effective)—see Fig. 1.

In short, YES bounds act as a sanity check for optimizers, offering actionable, deterministic, and data-specific feedback during training. This moves beyond subjective heuristics like early learning curve inspection, providing a rigorous benchmark for convergence and solver performance.

The rest of the paper is organized as follows: Section 2 introduces preliminaries and model architecture; Section 3 develops the single-layer bound; Section 4 extends it to deep networks; Section A presents numerical experiments on synthetic and real data.

2. Preliminaries

Let $\mathbf{A}_k$ and $\mathbf{b}_k$ denote the weight matrix and bias vector for layer $k \in [K]$ of the network, respectively. Suppose the input and output data are represented by matrices $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{Y} \in \mathbb{R}^{m\times d}$, respectively, such that $\mathbf{X} = \left[\begin{array}{c|c|c|c} \mathbf{x}_1 & \cdots & \mathbf{x}_d \end{array}\right], \mathbf{Y} = \left[\begin{array}{c|c|c|c} \mathbf{y}_1 & \cdots & \mathbf{y}_d \end{array}\right]$, where $\{\mathbf{x}_i \in \mathbb{R}^n\}_{i=1}^d$ and $\{\mathbf{y}_i \in \mathbb{R}^m\}_{i=1}^d$ denote d observations of n-dimensional and m-dimensional input and output features, respectively. Define the matrices $\{\mathbf{B}_k\}_{k=1}^K$ as $\mathbf{B}_k = \left[\begin{array}{c|c|c|c} \mathbf{b}_k & \cdots & \mathbf{b}_k \end{array}\right] \in \mathbb{R}^{m\times d}, k \in [K]$. We consider the following training loss for such a DNN employing a nonlinear activation function $\Omega(.)$, namely

$$\mathcal{L}_0\left(\{\mathbf{A}_k, \mathbf{B}_k\}_{k=1}^K, \mathbf{X}, \mathbf{Y}\right) \triangleq \|\Omega(\mathbf{A}_K\Omega(\mathbf{A}_{K-1}\Omega(\cdots\Omega(\mathbf{A}_1\mathbf{X}+\mathbf{B}_1)\cdots+\mathbf{B}_{K-1})+\mathbf{B}_K)-\mathbf{Y}\|_{\mathrm{F}}^2. \quad (1)$$

For our purpose, we will use the following notations: $\tilde{\mathbf{A}}_k = \left[\begin{array}{c|c} \mathbf{A}_k & \mathbf{b}_k \end{array}\right], \tilde{\mathbf{Y}}_k = \left[\begin{array}{c|c} \mathbf{Y}_k^\top & \mathbf{1} \end{array}\right]^\top, k \in [K]$. Using these notations, the training loss expression in (1) can be reformulated as follows: $\mathcal{L}_0\left(\{\tilde{\mathbf{A}}_k\}_{k=1}^K, \mathbf{X}, \mathbf{Y}\right) = \|\mathbf{Y}_{K+1} - \mathbf{Y}\|_{\mathrm{F}}^2$, where $\mathbf{Y}_{k+1} = \Omega\left(\tilde{\mathbf{A}}_k\tilde{\mathbf{Y}}_k\right)$ for $k \in [K]$ with $\mathbf{Y}_1 = \mathbf{X}$. For the rest of the paper, we present our formulations without considering the effects of bias terms. However, these formulations can be easily extended to include bias terms by substituting $\{\mathbf{A}_k\}$ and $\{\mathbf{Y}_k\}$ with $\{\tilde{\mathbf{A}}_k\}$ and $\{\tilde{\mathbf{Y}}_k\}$, respectively.

3. The Training Bound for Single-Layer Neural Networks

We begin by considering a one-layer neural network where the goal is to approximate the function $f : \mathbb{R}^n \to \mathbb{R}_+^m$. Let $\mathbf{A}$ be the weight matrix that we aim to optimize, such that the objective $\|\mathbf{Y} - \Omega(\mathbf{A}\mathbf{X})\|_{\mathrm{F}}^2$ is minimized. Assume $\mathbf{Y}$ is in the feasible set, i.e. the range of a non-linear activation function $\Omega(.)$ of the layer. The weight matrix $\mathbf{A}$ that minimizes the alternative objective $\|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_{\mathrm{F}}^2$ can then be expressed as $\mathbf{A} = \mathbf{Y}\mathbf{X}^\dagger$. The minimal achievable loss of training for a one-layer neural network is thus bounded as $\|\mathbf{Y}-\Omega(\mathbf{A}^{\mathrm{opt}}\mathbf{X})\|_{\mathrm{F}}^2 \le \|\mathbf{Y}-\Omega(\mathbf{Y}\mathbf{X}^\dagger\mathbf{X})\|_{\mathrm{F}}^2$, where $\mathbf{A}^{\mathrm{opt}}$ is the optimal weight matrix that minimizes the objective $\|\mathbf{Y} - \Omega(\mathbf{A}\mathbf{X})\|_{\mathrm{F}}^2$. Since the solution $\mathbf{Y}\mathbf{X}^\dagger$ is feasible (not necessarily optimal, although meaningful) for the optimizer of $\|\mathbf{Y} - \Omega(\mathbf{A}\mathbf{X})\|_{\mathrm{F}}^2$, a well-designed training stage is generally expected to satisfy this bound.

4. The Training Performance Bounds for Multi-Layer Networks

Inspired by our observation in the single-layer case, in the following we introduce the YES training bounds for multi-layer NNs.

4.1. The YES-0 Bound

Assuming an initial value $\mathbf{Y}_1 = \mathbf{X}$, the network aims to transform $\mathbf{Y}_1$ through intermediate points $\mathbf{Y}_2, \mathbf{Y}_3, \ldots, \mathbf{Y}_K$, finally achieving $\mathbf{Y}_{K+1} = \mathbf{Y}$. A sensible[1] but sub-optimal approach will be to assume at each layer that we aim to project directly to $\mathbf{Y}$, instead of other useful intermediate points $\{\mathbf{Y}_k\}$. Considering our one-layer bound, this no-intermediate approach will be equivalent to setting $\mathbf{A}_k = \mathbf{Y}\mathbf{Y}_k^\dagger, k \in [K]$. In particular, when there are no intermediate points we are dealing with an order-0 (referred to as YES-0) bound: $\mathcal{L}_0\left(\{\mathbf{A}_k^{\text{opt}}\}_{k=1}^K, \mathbf{X}, \mathbf{Y}\right) \leq \text{B}_{\text{YES-0}} \triangleq \|\mathbf{Y} - \mathbf{Y}_{K+1}\|_{\text{F}}^2$, where $\mathbf{Y}_{k+1} = \Omega\left(\mathbf{Y}\mathbf{Y}_k^\dagger\mathbf{Y}_k\right), k \in [K]$.

Since the central idea behind the creation of the YES-0 bound, in essence, stems from sequential projections, one may readily expect a decreasing behavior from the bound as the number of layers grows large. This is theoretically and numerically verified in Appendix B.

Obtaining the YES-0 bound, which is evaluated for each layer and projects the input of each layer to the final output, can be viewed as a layer-wise optimization with a linear closed-form operator. Since the YES-0 bound can be computed prior to training, it can serve as an immediate benchmark for the assessment of training in deep learning: One can verify whether the training has a YES-0 bound *certificate*, meaning that they are achieving a training loss lower than YES-0. Otherwise, they can attest that the training is not *proper*.

4.2. Beyond the YES-0 Bound

In this section, we present more sophisticated bounds—specifically, higher-degree YES bounds—to further assess the quality of training. We begin this construction by exploring whether the bounds can be tightened through a more conducive route to the output $\mathbf{Y}$.

4.2.1. EXPLORING STRUCTURED TRANSITION THROUGH NONLINEARITY-COMPLIANT SPACES

Enhanced bounds can be established by defining a sequence of useful intermediate points $\{\mathbf{Y}_k\}$ that conform to the nonlinear activation function of the network, i.e. $\mathbf{Y}_k \in H_\Omega$. Let us assume we have such a useful sequence as the outcomes of layers $t_2, \cdots, t_\Sigma$ (where $t_\Sigma = K+1$) as $\mathbf{Y}_{t_2}^\star, \cdots, \mathbf{Y}_{t_\Sigma}^\star$ (where $\mathbf{Y}_{t_\Sigma}^\star = \mathbf{Y}$). This gives rise to the YES-Σ bound, i.e.,

$$\mathcal{L}_0\left(\{\mathbf{A}_k^{\text{opt}}\}_{k=1}^K, \mathbf{X}, \mathbf{Y}\right) \leq \text{B}_{\text{YES-}\Sigma} \triangleq \|\mathbf{Y} - \mathbf{Y}_{K+1}\|_{\text{F}}^2, \quad \mathbf{Y}_{k+1} = \Omega\left(\mathbf{Y}_{t_\sigma}^\star \mathbf{Y}_k^\dagger \mathbf{Y}_k\right), \tag{2}$$

where $t_{\sigma-1} \leq k \leq t_\sigma, \quad 2 \leq \sigma \leq \Sigma$, with $\mathbf{Y}_1 = \mathbf{X}$. Given a judicious selection of $\mathbf{Y}_{t_2}^\star, \cdots, \mathbf{Y}_{t_\Sigma}^\star$, the latter is expected to provide a tighter error bound compared to the YES-0 bounding approach.

While it is fair to say that we hope that by iterative mapping, we get closer and closer to the output of interest, it has also been observed in various machine learning problems that after extensive training (resembling what we can describe as optimal training), the output of some inner layers become something meaningful to domain experts. This observation closely associates with the vision put forth above on tightening the YES-0 bound by considering useful and meaningful intermediate points.

1. The reason that we call this approach sensible stems from the optimization perspective, where the goal is to iteratively decrease the error between the initial estimate and the desired output.

The key question in deriving the enhanced YES bounds is thus the judicious designation of intermediate points. One may suggest these two ways:

1. *Problem-specific construction-based sets of intermediate points:* As for the image processing example above, in specific applications, we might have prior knowledge of the structure of the intermediate points. However, this assumption conflicts with our broader goal of developing training bounds that are applicable across diverse input-output data structures.

2. *Training data-driven generation of the mapping points:* This approach takes advantage of the training data to enhance the YES bounds along with the training loss. This is going to be the focus of our proposed YES bounds of *higher degree* in the following.

4.2.2. THE YES-k TRAINING BOUNDS ($k \geq 1$)

Leveraging the training data, we utilize the output from a subset of layers acquired during the training process as the intermediate points and incorporate these points into the YES bounds by projecting the input of the layer onto these intermediate outputs rather than relying solely on $\mathbf{Y}$. Specifically, one can utilize $k \in [K-1]$ intermediate points in a K layer network to create associated YES bounds. To better illustrate this idea, we present the example of the higher-degree YES bounds framework for the case of $k = 1$:

1. We set $\mathbf{Y}_1 = \mathbf{X}$.

2. We choose $\mathbf{Y}_2^\star$ as the output of the first layer during the training stage.

3. Following Section 3, we aim to optimize the weight matrix of the first layer $\mathbf{A}_1$, such that the alternative objective $\|\mathbf{Y}_2^\star - \mathbf{A}_1\mathbf{Y}_1\|_{\mathrm{F}}^2$ is minimized. This is achieved by $\mathbf{A}_1 = \mathbf{Y}_2^\star\mathbf{Y}_1^\dagger$.

4. We then obtain the output of the first layer as $\mathbf{Y}_2 = \Omega\left(\mathbf{Y}_2^\star\mathbf{Y}_1^\dagger\mathbf{Y}_1\right)$.

5. Since we only chose $k = 1$ intermediate point, we then progressively project the input of each layer to the output $\mathbf{Y}$ as $\mathbf{Y}_{k+1} = \Omega\left(\mathbf{Y}\mathbf{Y}_k^\dagger\mathbf{Y}_k\right), k \in \{2, \cdots, K\}$.

6. We compute the resulting error of this process as $\|\mathbf{Y} - \mathbf{Y}_{K+1}\|_{\mathrm{F}}^2$.

7. We repeat steps (3-6) by choosing other intermediate points $\mathbf{Y}_3^\star, \cdots, \mathbf{Y}_K^\star$ in step 2.

8. Take the minimum of all errors obtained in Step 6, leading to the creation of the YES-1 training bound.

The name YES-1 bound takes into account the fact that we have only considered $k = 1$ intermediate point in the above process. Note that the above formulation can be easily extended to $k \in \{2, \cdots, K-1\}$ intermediate points, generating higher degree YES bounds, namely YES-k bounds for $k \in \{2, \cdots, K-1\}$. In contrast to the YES-0 training bound, the YES bounds of higher degree are *real-time*, i.e., they evolve alongside the training loss. It is important to note that YES-k bounds for larger k are not necessarily smaller than those for smaller k, particularly at the initial epochs where the training may not suggest excellent intermediate points. Higher degree bounds are, however, highly likely to perform better when the training is in good condition. We numerically validate this phenomenon in Appendix C. In Appendix D, we present a monotonic modification of

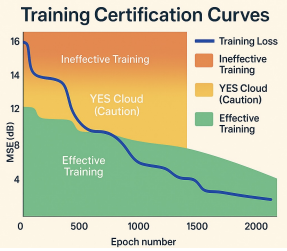


Figure 1: YES training cloud system for quality monitoring: A training loss that remains above the YES training cloud (**red** area) indicates ineffective training. When the loss penetrates the cloud (**yellow** area), it suggests that meaningful training has occurred or is in progress—network weights have been significantly influenced by the data. However, caution is advised, as the training is certainly not optimal. Dropping below the cloud (**green** area) signals effective training in progress and suggests potential for optimality. It may also indicate diminishing returns in the training process, where further gains could be incremental.

the YES-k bounds for $k \geq 1$, along with an important observation that increasing the degree, i.e., k, does not necessarily improve the YES bounds (for the monotonic case). In fact, all the bounds remain relatively close to each other. This observation can be beneficial from a computational aspect.

4.2.3. THE YES TRAINING CLOUD-SYSTEM FOR QUALITY MONITORING

We now illustrate the integration of the proposed YES bounds with the training process, culminating in the creation of an intuitive training cloud to monitor progress in real time (see Fig. 1). As the YES bounds evolve over epochs, similar to the training loss, they enable users to visually observe the interaction between training performance and the YES bounds. This visualization enables tracking of key moments during the training process, such as when the training loss surpasses the YES-0 bound, when it improves beyond the best YES-k bounds, the epochs at which these transitions occur, and how the bounds and training results interact throughout the process.

To help users navigate key transitions during the training process, the YES training cloud system employs a color-coding scheme. A training loss that remains above the YES training cloud—depicted as the red area—indicates ineffective training. When the loss reaches and enters the cloud—the yellow area—it signifies that meaningful training is taking place, with the network weights being substantially influenced by the data. However, caution is still advised at this stage, as the training is currently situated in a suboptimal region of the optimization landscape—evidenced by the fact that it has not yet surpassed the best achievable YES-k bounds. A loss that descends below the cloud into the green area denotes effective training, suggesting a potential for achieving optimal performance.

In Appendix A and its subsections, we present our numerical investigations of the proposed bounds' performance in training monitoring, considering both synthesized and real datasets. We also analyze the computational cost of our scheme and compare the effectiveness of different degrees of the YES bounds. In the subsequent appendices, we provide the theoretical foundations and detailed results underlying the proposed scheme.

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*, pages 242–252. PMLR, 2019.

[2] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[4] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.

[5] Samet Oymak and Mahdi Soltanolkotabi. Overparameterized nonlinear learning: Gradient descent takes the shortest path? In *International Conference on Machine Learning*, pages 4951–4960. PMLR, 2019.

[6] Scott Pesme, Aymeric Dieuleveut, and Nicolas Flammarion. On convergence-diagnostic based step sizes for stochastic gradient descent. In *International conference on machine learning*, pages 7641–7651. PMLR, 2020.

[7] Megha Srivastava, Simran Arora, and Dan Boneh. Optimistic verifiable training by controlling hardware nondeterminism. *arXiv preprint arXiv:2403.09603*, 2024.

[8] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[9] Roman Vershynin. Memory capacity of neural networks with threshold and rectified linear unit activations. *SIAM Journal on Mathematics of Data Science*, 2(4):1004–1033, 2020.

[10] Haoming Zhang, Mingqi Zhao, Chen Wei, Dante Mantini, Zherui Li, and Quanying Liu. Eegdenoisenet: a benchmark dataset for deep learning solutions of eeg denoising. *Journal of Neural Engineering*, 18(5):056057, 2021.

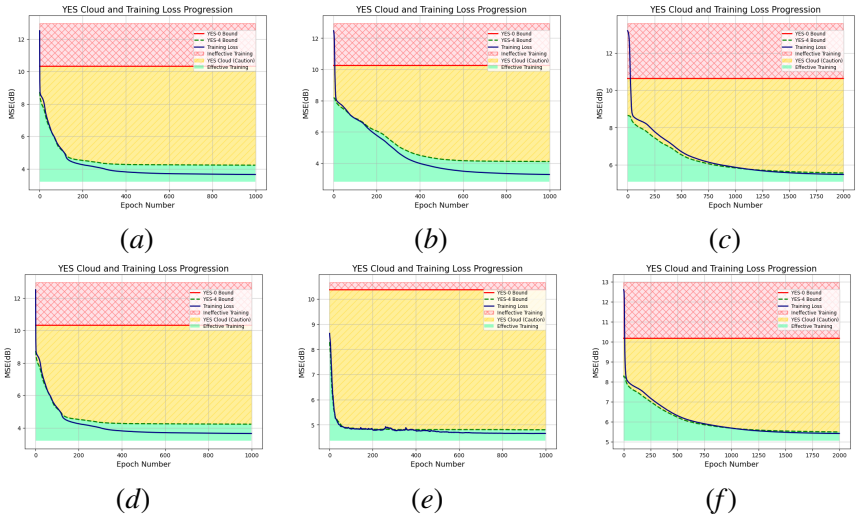


Figure 2: YES training clouds for the phase retrieval model. The clouds are shown for a fully connected network with 5 layers, each corresponding to different training parameter settings. Figs (a)-(c) illustrate the performance of the YES bounds for different batch sizes: 20, 100, and 500, respectively, with a learning rate of $1e-3$. Figs. (d)-(f) compare the YES bounds to the training process with different learning rates: $1e-3$, $1e-2$, and $1e-4$. As seen in Figs. (b) and (c), increasing the batch size slows the convergence rate, with the training loss entering the green region after more than 100 epochs. Interestingly, when adjusting the learning rate to $1e-2$ and $1e-4$, as shown in Figs. (e) and (f), the training struggles to reach the green region, suggesting that a learning rate of $1e-3$ is the proper parameter for this task. This observation is further supported by comparing the loss functions across Figs. (d)-(f). Another notable observation in Fig. (f) is that both the YES bound and the training loss converge relatively closely until the final convergence, indicating that the training solution behaves similarly to a linear projection.

Appendix A. Numerical Examples and Discussion

In this section, we numerically evaluate the effectiveness of the proposed YES bounds in assessing training performance across both synthetic and real-world data recovery tasks. For the synthetic experiments, datasets were generated based on the phase retrieval model. For real-world applications, we applied our bounds to three tasks: image classification on the MNIST dataset, EEG signal denoising using the EEGdenoiseNet dataset [10], and image denoising using the BSD500 dataset [4] (see Appendix A.3). All experiments were conducted using 2 vCPUs and 1 GPU.

- **Phase Retrieval:** The data is generated by the following model:

$$\mathbf{b}_i = |\mathbf{A}\mathbf{x}_i|, \quad i \in [d], \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{20\times 20}$ is a Gaussian sensing matrix with entries drawn from $\mathcal{N}(0, 1/20)$, the signal $\mathbf{x} \in \mathbb{R}^{20}$ is generated as $\mathcal{N}(0, 1)$, and $d = 1000$ samples are generated.

We trained 5-layer fully connected networks to approximate the phase retrieval model under different parameter configurations, utilizing the ADAM algorithm for optimization. In all experiments,

the learning rate was initialized at $\eta = \eta_0$ and reduced by a decay factor of 0.9 every 50 epochs. It is important to note that the bias term was excluded in the training process. We considered the following criterion for stopping the training: the rate of change in network weights is sufficiently low.

In Fig. 2, we present color-coded clouds illustrating the training process and corresponding bounds for the phase retrieval model under various conditions and parameters. Figs. 2(a)-(c) show the clouds for different batch sizes, while Figs. 2(d)-(f) illustrate the clouds for varying learning rates. Specifically, Fig. 2(a) corresponds to a batch size of 20, Fig. 2(b) to a batch size of 100, and Fig. 2(c) to a batch size of 500, all with a fixed learning rate of $1e-3$. Similarly, Fig. 2(d) uses a learning rate of $1e-3$, Fig. 2(e) uses $1e-2$, and Fig. 2(f) uses $1e-4$, all with a fixed batch size of 20.

As seen in Figs. 2(a)-(b), training enters the green region after approximately 100 epochs, signaling effective training is in progress. Notably, the convergence rate for the batch size of 20 is slightly faster than that of 100, as indicated by the earlier entry into the green region. In Fig. 2(c), the model struggles to reach the green region, only doing so after 1250 epochs, suggesting that batch sizes of 20 or 100 are more suitable for this task. Notably, by our real-time bounds, one can notice the ineffectiveness of the chosen parameters in a real-time manner instead of running the model for various parameter settings. In terms of learning rate, Fig. 2(d) shows that a rate of $1e-3$ leads to effective training, with the loss entering the green region after 100 epochs. In Fig. 2(e), increasing the learning rate to $1e-2$ does not alter the entry point into the green region, but the training loss plateaus closer to the YES bound, implying that the solution aligns with a linear projection—potentially suboptimal in this context. This pattern is consistent in Fig. 2(f) with a learning rate of $1e-4$, where the model enters the green region after 1250 epochs and similarly plateaus near the YES bound. Overall, these observations suggest that $1e-3$ is a better learning rate for this task compared to $1e-2$ and $1e-4$.

• **MNIST dataset:** To further assess the performance of our YES bounds in practical scenarios, we conducted experiments using the MNIST dataset, which was designed for classification tasks. We worked with 5000 training and 5000 test samples. Each image, representing a digit $i \in \{0, \cdots, 9\}$, was encoded by generating a zero matrix with the same dimension as the input image with a single 1 placed at $(i+1, i+1)$. A 5-layer fully connected network was trained with SGD, using an initial learning rate η_0 and a decay factor of 0.7 every 50 epochs. The classification was performed by minimizing the MSE between model outputs and encoded images, with the success rate determined by accurate classifications over the entire dataset.

As shown in Fig. 3(a), with a learning rate of $1e-4$, the training loss struggles to move beyond the caution region and remains close to the bottom of the YES clouds. In terms of success rates, Fig. 3(b) displays the training process, while Fig. 3(c) presents the test stage. Although the performance appears satisfactory, the YES cloud suggests that the model's solution is akin to a linear projection, indicating suboptimal training parameters. Adjusting these parameters could lead to improved model performance. In Fig. 3(d), we apply a learning rate of $5e-4$ for the solver. In this case, the training loss reaches the green region after approximately 30 epochs. The success rate for the training results, shown in Fig. 3(e), indicates that when the training loss enters the yellow region, the success rate is 85%. Once it enters the green region, the success rate increases to 95%, and at the convergence point, we achieve 100% performance. For the test results depicted in Fig. 3(f), the loss reaches the yellow region at 83%, and upon entering the green region, the success rate becomes 92%. At the convergence rate, the test results reach 95%.

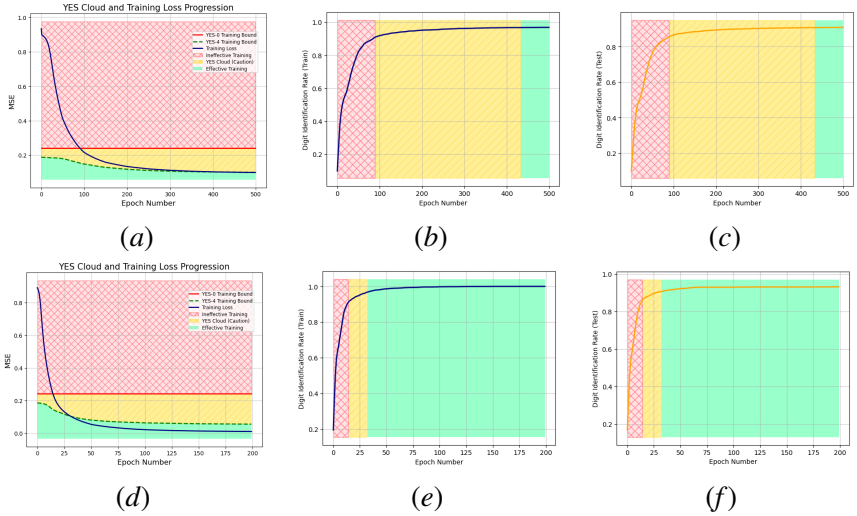


Figure 3: YES bounds cloud for the training process on the MNIST dataset is presented alongside the success rates for both the training and testing stages. Figs. (a) and (d) show the YES clouds for solvers with different learning rates: (a) $1e-4$ and (d) $5e-4$. Figs. (b) and (c) display the success rates during the training and testing phases, respectively, within the color-coded YES cloud regions. These figures demonstrate how effectively the YES bounds monitor solver performance using a learning rate of $1e-4$. Figs. (e) and (f) illustrate the success rates within the YES cloud regions for the solver using a learning rate of $5e-4$.

As discussed earlier, when the training loss plateaus in the green region, the model's solution can be a strong candidate for optimality. Figs. 3(e) and (f) illustrate the model's effectiveness, achieving a 100% success rate in training and 95% in testing. This demonstrates the model's high accuracy and generalization, indicating that it is well-tuned to the task at hand.

• **EEGdenoiseNet dataset:** This dataset includes 4514 clean EEG segments, 3400 pure electrooculography (EOG) segments, and 5598 pure electromyography (EMG) artifact segments, enabling the synthesis of contaminated EEG signals with known ground-truth clean components [10]. We generated contaminated EEG signals using the model $\mathbf{x}_n = \mathbf{x} + \kappa\mathbf{n}$, where $\mathbf{x}$ is the clean EEG signal, $\mathbf{n}$ denotes ocular or myogenic artifacts, $\mathbf{x}_n$ is the resulting contaminated signal, and κ is a hyperparameter to control the SNR.

To perform the denoising task, we trained a 3-layer fully connected network using SGD, with a fixed batch size of 20 in all experiments. Fig. 4 illustrates the YES bound clouds alongside their corresponding test results. Specifically, Figs.4(a)-(c) show YES clouds for three learning rate schedules: (a) $2e-4$ with decay factor of 0.9 every 20 epochs, (b) $2e-4$ with decay factor of 0.7 every 20 epochs, and (c) $8e-5$ with decay factor of 0.9 every 20 epochs. As shown, case (a) successfully enters the green region of the YES cloud after approximately 200 epochs, while cases (b) and (c) fail to surpass the YES bound.

While the YES bounds can be used to correctly identify a better learning rate, one might argue that analyzing early learning curves and simply observing the training process could be sufficient

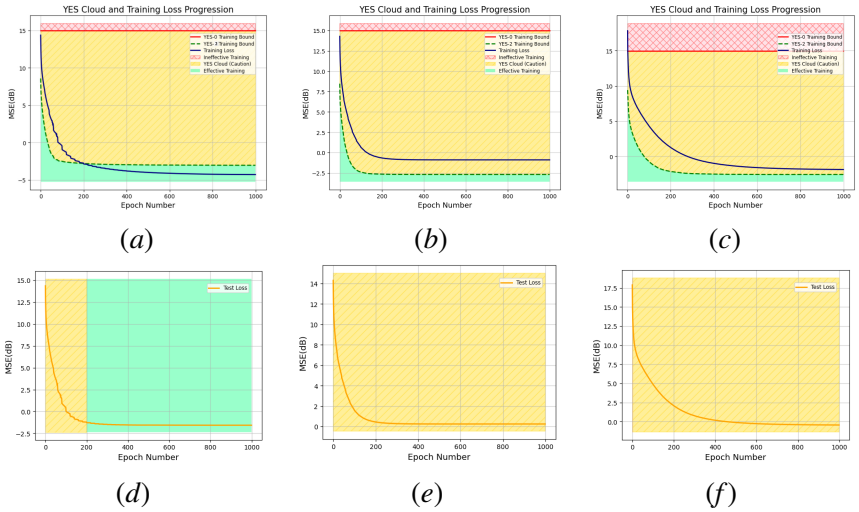


Figure 4: YES bound clouds for the training process on the EEGdenoiseNet dataset, shown alongside corresponding test performance. Figs. (a)-(c) depict YES clouds for solvers with different learning rate: (a) $2e-4$ with decay factor of 0.9 every 20 epochs, (b) $2e-4$ with decay factor of 0.7 every 20 epochs, and (c) $8e-5$ with decay factor of 0.9 every 20 epochs. Figs. (d)-(f) show the respective test results. These figures demonstrate how effectively the YES bounds monitor solver performance with respect to different learning rates.

to assess whether the learning rate is appropriate. However, as illustrated in Figs. 4(a) and (b), it appears impossible to conclude that the learning rate $2e-4$ with a decay factor of 0.9 is better than the learning rate $2e-4$ with a decay factor of 0.7 in the first 100 epochs without comparing against the proposed bound provided by the cloud system. Thanks to this comparison, we are able to clearly distinguish the better choice.

Another noteworthy observation is that although both (b) and (c) plateau in the suboptimal yellow region, case (c) lies closer to the bottom of the YES cloud than case (b). This also correlates with the test results: case (c) achieves better test performance than case (b). Additionally, when case (a) enters the green region at around epoch 200, the corresponding test performance reaches nearly -1.25 dB—outperforming both (b) and (c) even after 1000 epochs. This experimental observation suggests that the green region may indicate not only effective training performance but also the model's potential generalizability to unseen data. Appendix A.2 presents the computational analysis of the proposed YES bounds on the EEGdenoiseNet dataset.

A.1. Test Results For Training Process Monitored By YES Clouds

Beyond the training process, it is insightful to investigate how test results behave as the training progresses through different regions of the color-coded clouds. To explore this, we present both training and test outcomes for the phase retrieval model in Fig. 5. As observed, when the training loss decreases in the red region, the test loss similarly declines. When training plateaus in the yellow region, the test loss also plateaus. Interestingly, upon entering the green region, the training loss

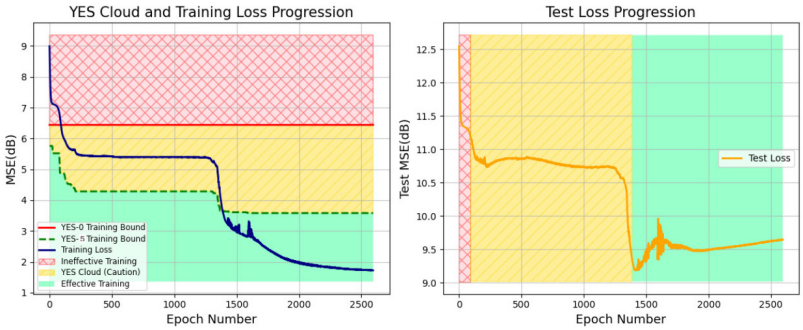
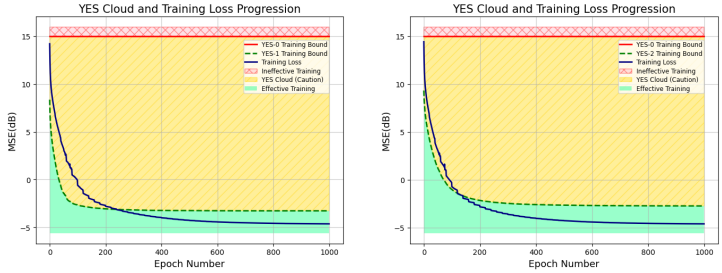


Figure 5: The YES bounds cloud for the training process is presented alongside the test stage for the same training results monitored by the YES training bounds.



(*a*) (*b*)

Figure 6: YES training bounds for the EEGdenoiseNet dataset, considering only (a) YES-1 and (b) YES-2. These results indicate that the training bound provided by only YES-1 is tighter compared to that of YES-2.

initially exhibits fluctuations, likely due to the learning rate, before plateauing—a pattern mirrored in the test loss. However, after approximately 2000 epochs, the test loss increases.

It is often a challenge to determine the optimal point to stop training (cost vs performance trade-off). A common approach is to monitor the rate of change in the loss function, waiting for the loss to plateau as a sign of potential convergence. However, as shown in Fig. 5, the training objective of neural networks can result in the loss appearing to plateau multiple times before quality training is achieved. This is where the YES clouds come to the rescue: *To know when the smaller rate of change in the loss or the weights of the neural network does not equate optimality, and when to act on it—i.e., when we are in the green.*

A.2. Computational Analysis of YES Bounds on EEGdenoiseNet dataset

To evaluate the computational cost of computing the YES training bounds on the EEGdenoiseNet dataset, we trained a 3-layer fully connected network, consistent with the setup described in Section A. The training was performed using SGD with a fixed batch size of 20 and an initial learning rate of $2e-4$, decayed by a factor of 0.9 every 20 epochs. Since a 3-layer fully connected network involves only the YES-1 and YES-2 bounds, we measured the CPU time required to compute each of them individually.

Table 1: CPU time for the training process, as well as the computation of the YES-1 and YES-2 bounds on the EEGdenoiseNet dataset.

	CPU Time (s)
Training	526.559
YES-1	550.268
YES-2	416.563

To visualize the corresponding YES training clouds, we generated two separate plots: one using only the YES-1 bound (excluding YES-2) and another using only the YES-2 bound (excluding YES-1). The resulting training clouds are shown in Fig. 6(a) and Fig. 6(b), respectively. As shown in the results, the YES-1 bound is more effective than the YES-2 bound in monitoring the training process. This is evident from the tighter training bound provided by YES-1 compared to YES-2. The corresponding CPU times are reported in Table 1. Training the model without incorporating the YES bounds takes 526.559 s, while computing the YES-1 and YES-2 bounds requires 550.268 s and 416 s, respectively. These results show that computing the YES-1 and YES-2 bounds requires nearly as much CPU time as the training process itself. The higher CPU time for YES-1 is due to the fact that, for a 3-layer network, it requires computing the associated bounds twice (as it considers only a single intermediate point), whereas YES-2 requires this computation only once. However, this relationship may change depending on the number of network layers.

A.3. BSD500 Dataset

To elucidate the practical significance of the YES bounds and their associated cloud system, we further examine an image recovery task for an image degradation process characterized by the model:

$$\mathbf{b}_i = |\mathbf{x}_i + \mathbf{n}_i|^2, \quad i \in [d], \tag{4}$$

where each $\mathbf{x}_i$ represents a distinct patch of the original image undergoing recovery. This model presents a degradation process that involves two primary challenges:

1. Additive Noise ($\mathbf{n}_i$): The term $\mathbf{n}_i$ introduces additive noise, demanding denoising strategies to mitigate its adverse effects on image quality.

2. Phase Loss ($|\cdot|^2$): The absolute value squared operation results in phase loss, requiring phase retrieval methods to restore essential phase information for accurate reconstruction.

We conducted our experiments on natural image patches using the BSD500 dataset [4], selecting a total of 200 images. From each image, we randomly extracted 25 patches of size 8×8, resulting in 5000 patches in total. Of these, $d = 4000$ were used for training and the remaining 1000 were reserved for the test stage. Following the model in (4), we added additive Gaussian noise with a standard deviation of 0.05 to the patches (with pixel values normalized to the range $[0, 1]$), and subsequently generated $\mathbf{b}_i$ for $i \in [d]$.

We trained a 2-layer fully connected neural network using SGD with a batch size of 20. Two initial learning rates were considered: (i) $1e-4$ and (ii) $3e-4$. In both settings, we applied a learning rate scheduler with a decay factor of 0.9 every 20 epochs. The training performance, as monitored by our proposed YES cloud framework, is presented in Figs.7(a) and (b), with corresponding test performance shown in Figs.7(c) and (d). As illustrated, our YES bounds continue to serve as effective indicators of training quality across different learning configurations.

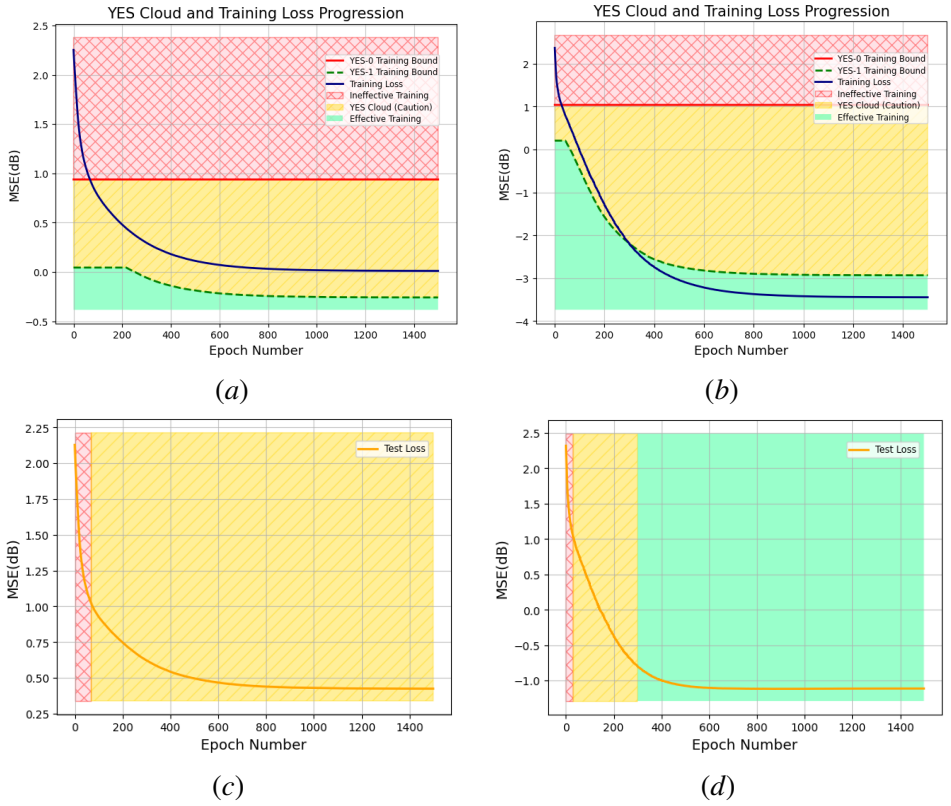


Figure 7: YES bound clouds for the training process on the BSD500 dataset, shown alongside corresponding test performance. Figs. (a) and (b) depict YES clouds for solvers with different learning rate: (a) $1e-4$ with decay factor of 0.9 every 20 epochs, and (b) $3e-4$ with decay factor of 0.9 every 20 epochs. Figs. (c) and (d) show the respective test results. These figures demonstrate how effectively the YES bounds monitor solver performance with respect to different learning rates.

Appendix B. The Decreasing Behaviour of YES-0 Bound

Theorem 1 *Let Ω be an activation function in a deep neural network. If Ω is applied in an element-wise manner and satisfies the following conditions:*

- 1-Lipschitz Condition*:*

$$\|\Omega(\mathbf{x}_1) - \Omega(\mathbf{x}_2)\| \leq \|\mathbf{x}_1 - \mathbf{x}_2\|, \quad \forall\, \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n, \tag{5}$$

- Projection Property*:*

$$\Omega(\mathbf{Y}) = \mathbf{Y}, \quad \text{if} \quad \mathbf{Y} \in H_\Omega, \tag{6}$$

then the YES-0 bound is monotonically decreasing with respect to the depth of the network. That is, for each layer k:

$$\|\mathbf{Y} - \mathbf{Y}_{k+1}\|_{\mathrm{F}}^2 \leq \|\mathbf{Y} - \mathbf{Y}_k\|_{\mathrm{F}}^2. \tag{7}$$

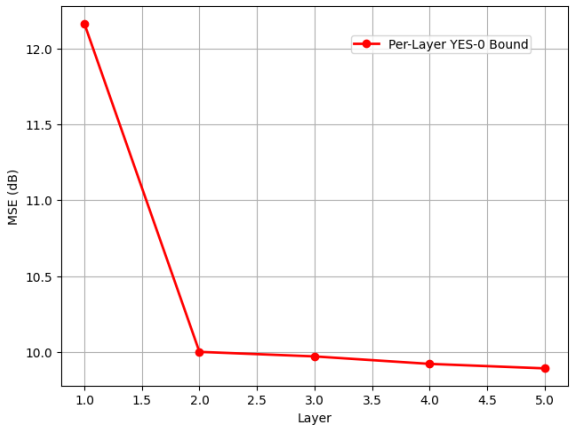


Figure 8: YES-0 decay with respect to the number of layers.

Proof Following our formulations, the error at layer $(k-1)$ is

$$\mathbf{E}_{k-1} = \mathbf{Y} - \mathbf{Y}_k, \tag{8}$$

where $\mathbf{Y}$ is the target output, and $\mathbf{Y}_k$ is the network output after $(k-1)$ layers. At each layer, the network updates its output via:

$$\mathbf{Y}_{k+1} = \Omega(\mathbf{A}_k \mathbf{Y}_k), \tag{9}$$

with $\mathbf{A}_k$ representing the weight matrix associated with the YES-0 bound at layer k. The error at layer k is thus:

$$\mathbf{E}_k = \mathbf{Y} - \Omega(\mathbf{A}_k \mathbf{Y}_k). \tag{10}$$

By considering the 1-Lipschitz and projection properties of the activation function Ω, we have:

$$\begin{aligned} \|\mathbf{E}_k\|_{\mathrm{F}}^2 &= \|\mathbf{Y} - \Omega(\mathbf{A}_k \mathbf{Y}_k)\|_{\mathrm{F}}^2 \\ &= \|\Omega(\mathbf{Y}) - \Omega(\mathbf{A}_k \mathbf{Y}_k)\|_{\mathrm{F}}^2 \\ &\leq \|\mathbf{Y} - \mathbf{A}_k \mathbf{Y}_k\|_{\mathrm{F}}^2. \end{aligned} \tag{11}$$

Since $\mathbf{A}_k$ is the minimizer of the quadratic criterion $\|\mathbf{Y} - \mathbf{A}_k \mathbf{Y}_k\|_{\mathrm{F}}^2$, we have:

$$\|\mathbf{Y} - \mathbf{A}_k \mathbf{Y}_k\|_{\mathrm{F}}^2 \leq \|\mathbf{Y} - \mathbf{Y}_k\|_{\mathrm{F}}^2 = \|\mathbf{E}_{k-1}\|_{\mathrm{F}}^2. \tag{12}$$

Combining (12) with (11) completes the proof. ■

In Fig. 8, we validate Theorem 1 by demonstrating the decay of the YES-0 bound across layers. This result is based on the phase retrieval model.

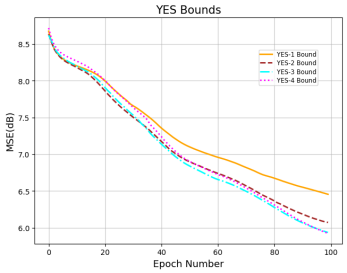
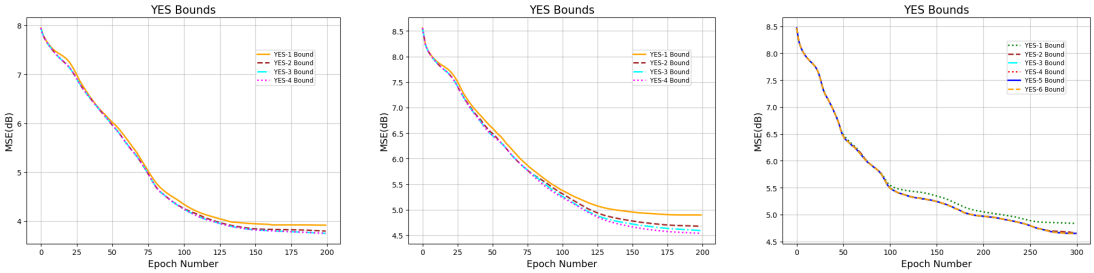


Figure 9: YES training bounds with varying degrees, without imposing monotonicity, are presented. As can be observed, increasing the degree of the bound does not necessarily improve it, as the bounds remain closely aligned with each other.



(*a*) (*b*) (*c*)

Figure 10: YES training bounds with varying degrees, this time incorporating monotonicity across different initializations, are presented. Similar to the non-monotonic case, increasing the degree of the bound does not necessarily enhance it, as the bounds remain close to each other.

Appendix C. Non-Decreasing Behaviour of YES-k Bounds Without Monotonicity

YES training bounds with different degrees, without imposing monotonicity, are shown in Fig. 9 for the phase retrieval model. An interesting observation from this figure is that increasing the degree does not necessarily improve the YES bounds. In fact, all the bounds remain relatively close to each other.

Appendix D. The YES-k Training Bounds ($k \geq 1$) With Monotonicity

In Algorithm 1, we reformulate the YES-k bounds for $k \geq 1$, incorporating a monotonic modification through the inclusion of YES-k subsets to ensure the bounds remain monotonic.

For the YES bounds with monotonicity, as illustrated in Fig. 10 with various initializations, it is evident that the YES bounds are closely grouped. We investigated this observation using fully connected networks with both 5-layer and 7-layer architectures, conducted this experiment 1000 times, and consistently observed similar results. This observation suggests that we may leverage the

Algorithm 1: Generation of the YES Training Bounds ($k \geq 1$).

Data: $(\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{Y} \in \mathbb{R}^{m \times d})$ are training data matrices with d denoting the number of training samples.
Result: YES training bounds.
$\mathcal{I} \leftarrow \{2, \cdots, K\}$;
$\mathbf{e} \leftarrow \mathbf{0}_{K-1}$;
for $k = 1 : (K-1)$ **do**
  $\mathcal{H} \leftarrow \mathrm{combination}(\mathcal{I}, k)$ ▷ $\mathrm{combination}(\mathcal{I}, k)$ is the combination operator that selects k items from the set $\mathcal{I}$;
  $\mathbf{u} \leftarrow \mathbf{0}_{|\mathcal{H}|}$ ▷ $\mathbf{0}_{|\mathcal{H}|}$ denotes a zero vector with the length $|\mathcal{H}|$;
  for $i = 0 : |\mathcal{H}| - 1$ **do**
    $\mathcal{H}_i \leftarrow \mathcal{H}[i]$ ▷ $\mathcal{H}[i]$ denotes the i-th combination item of $\mathcal{H}$;
    $l \leftarrow 0$;
    $\mathbf{Y}^\star \leftarrow [\,]$ ▷ $[\,]$ denotes an empty tensor;
    for $j = 0 : (k-1)$ **do**
      $\mathbf{Y}^\star.\mathrm{append}(\mathrm{model}_{\mathcal{H}_i}(\mathbf{X}))$ ▷ $\mathbf{Y}^\star.\mathrm{append}(\mathbf{T})$ denotes appending the matrix $\mathbf{T}$ in an empty tensor $\mathbf{Y}^\star$, $\mathrm{model}_{\mathcal{H}_i}(\mathbf{X})$ denotes the output of the training model at specific layers specified by the elements in $\mathcal{H}_i$;
    end
    $\mathbf{Y}_t \leftarrow \mathbf{X}$;
    for $j = 0 : (k-1)$ **do**
      while $l \leq \mathcal{H}_i[j]$ **do**
        $\mathbf{A}_t \leftarrow \mathbf{Y}^\star[j]\mathbf{Y}_t^\dagger$;
        $\mathbf{Y}_t \leftarrow \Omega\left(\mathbf{A}_t\mathbf{Y}_t\right)$;
        $l \leftarrow l + 1$;
      end
    end
    for $z = 1 : K - l - 1$ **do**
      $\mathbf{A}_t \leftarrow \mathbf{Y}\mathbf{Y}_t^\dagger$;
      $\mathbf{Y}_t \leftarrow \Omega\left(\mathbf{A}_t\mathbf{Y}_t\right)$;
    end
    $\mathbf{u}[i] \leftarrow \|\mathbf{Y} - \mathbf{Y}_t\|_{\mathrm{F}}^2/d$;
  end
  $\mathbf{e}[k-1] \leftarrow \min\ \mathbf{u}$;
end
YES bound $\leftarrow \min\ \mathbf{e}$;
return YES bound

advantages of higher-degree YES bounds by calculating only the first few YES-k bounds, which could be beneficial from a computational standpoint.

Appendix E. Theoretical Analysis of Linear Projection Method

In this section, we aim to theoretically analyze the linear projection method used in deriving our proposed YES bounds. To this end, we focus on the *ReLU-based architecture*. Within the YES framework, each step involves solving a one-layer ReLU equation. The linear projection method offers a simple, closed-form solution to this problem. Therefore, it is important to assess whether this method yields a reasonable approximation—specifically, one that is close to the true weight matrix in terms of the norm-2 error, for a single-layer ReLU equation. To address this, we consider a random setting and compare the solution obtained from the linear projection method with the corresponding information-theoretic (IT) bound.

E.1. Formulation and Guarantees

Consider the following ReLU equation:

$$y_j = \mathrm{ReLU}(\mathbf{x}_j^\top \boldsymbol{\omega}^{\mathrm{opt}}), \quad j \in [d], \tag{13}$$

where $\mathrm{ReLU}(\cdot)$ denotes the ReLU activation function, $\boldsymbol{\omega}^{\mathrm{opt}} \in \mathcal{S}^{n-1} \triangleq \{\boldsymbol{\omega} \in \mathbb{R}^n : \|\boldsymbol{\omega}\|_2 = 1\}$ is the target weight vector, and each $\mathbf{x}_j$ is drawn independently from the standard multivariate normal distribution, i.e., $\mathbf{x}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$ for $j \in [d]$. To recover $\boldsymbol{\omega}^{\mathrm{opt}}$ from the observations in (13), one can formulate the following optimization problem:

$$\mathcal{P}: \quad \underset{\boldsymbol{\omega} \in \mathcal{S}^{n-1}}{\text{minimize}} \quad \mathcal{L}(\boldsymbol{\omega}) \triangleq \frac{1}{d} \sum_{j=1}^{d} \left| y_j - \mathrm{ReLU}\left(\mathbf{x}_j^\top \boldsymbol{\omega}\right) \right|^2. \tag{14}$$

Since the labels $\{y_j\}$ are generated according to (13), the minimum achievable loss in the optimization problem $\mathcal{P}$ defined in (14) is zero. Therefore, any estimator $\bar{\boldsymbol{\omega}} \in \mathcal{S}^{n-1}$ that satisfies $\mathcal{L}(\bar{\boldsymbol{\omega}}) = 0$ is a global minimizer of (14). To characterize such estimators, we introduce the following notion of consistency:

Definition 2 (Estimator Consistency (EC)) *Let $\bar{\boldsymbol{\omega}} \in \mathcal{S}^{n-1}$ be an estimator obtained by any arbitrary algorithm addressing the optimization problem $\mathcal{P}$ in (14). We say that $\bar{\boldsymbol{\omega}}$ is consistent if it satisfies the following condition:*

$$\mathrm{sgn}\left(\mathbf{x}_j^\top \boldsymbol{\omega}^{\mathrm{opt}}\right) = \mathrm{sgn}\left(\mathbf{x}_j^\top \bar{\boldsymbol{\omega}}\right), \quad j \in [d], \tag{15}$$

where $\mathrm{sgn}(\cdot)$ denotes the sign function.

Based on the definition of EC, it is evident that any global minimizer of (14) must satisfy the EC condition. With this in mind, our objective is to derive an upper bound on the estimation error of any consistent estimator, with high probability. To facilitate this analysis, we introduce the following operator:

Definition 3 (Sample Average Operator) *For any weight vector $\boldsymbol{\omega} \in \mathcal{S}^{n-1}$, we define the sample average operator $T(\boldsymbol{\omega})$ as:*

$$T(\boldsymbol{\omega}) = \frac{1}{d}\sum_{j=1}^{d}\left|\mathbf{x}_j^\top \boldsymbol{\omega}\right|. \tag{16}$$

Based on the definition of the sample average operator $T(\boldsymbol{\omega})$, the following theorem establishes a concentration bound for this operator. As we will demonstrate in Theorem 6, this result serves as a key step toward our main objective—deriving a high-probability upper bound on the estimation error of any consistent estimator.

Theorem 4 *Let the sample average operator $T(\boldsymbol{\omega})$ be defined as in Definition 3, where each $\mathbf{x}_j$ is independently drawn from the standard multivariate normal distribution, i.e., $\mathbf{x}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$ for all $j \in [d]$. Then, there exist universal constants $c_1, c_2, c_3, c_4 > 0$ such that, for any $\boldsymbol{\omega} \in \mathcal{S}^{n-1}$, the following concentration bound holds:*

$$\mathbb{P}\left(\sup_{\boldsymbol{\omega}\in\mathcal{S}^{n-1}}\left|T(\boldsymbol{\omega}) - \sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right| \le \delta\|\boldsymbol{\omega}\|_2\right) \ge 1 - 2e^{-c_1\delta^2 d}, \tag{17}$$

provided that the number of samples d satisfies:

$$d \ge c_2\delta^{-2} n \log\left(c_3 + c_4\delta^{-1}\right). \tag{18}$$

Proof For a fixed $\boldsymbol{\omega} \in \mathcal{S}^{n-1}$, it is easy to notice that[2]

$$\mathbb{E}T(\boldsymbol{\omega}) = \sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2. \tag{19}$$

In the following lemma, we present the general Hoeffding's inequality:

Lemma 5 *[8, Theroem 2.6.2] Let $X_1, \cdots, X_N$ be independent, mean zero, sub-gaussian random variables. Then, for every $t \ge 0$, we have*

$$\mathbb{P}\left(\left|\sum_{i=1}^{N} X_i\right| \ge t\right) \le 2e^{-\frac{ct^2}{\sum_{i=1}^{N}\|X_i\|_{\psi_2}^2}}, \tag{20}$$

where c is a positive constant.

Denote $d_j = \left|\mathbf{x}_j^\top \boldsymbol{\omega}\right|$ for all $j \in [d]$. Since each $\mathbf{x}_j$ is a standard Gaussian random vector, we can write $\|\mathbf{x}_j\|_{\psi_2} \le K$ for some universal constant K, where $\|\cdot\|_{\psi_2}$ denotes the sub-gaussian norm. Then, the sub-gaussian norm of each d_j can be bounded as follows:

$$\|d_j\|_{\psi_2} = \left\|\left|\mathbf{x}_j^\top \boldsymbol{\omega}\right|\right\|_{\psi_2} = \left\|\mathbf{x}_j^\top \boldsymbol{\omega}\right\|_{\psi_2} \le \|\mathbf{x}_j\|_{\psi_2}\|\boldsymbol{\omega}\|_2 \le K\|\boldsymbol{\omega}\|_2. \tag{21}$$

Following Lemma 5, we obtain the following concentration bound for a fixed $\boldsymbol{\omega} \in \mathcal{S}^{n-1}$:

$$\mathbb{P}\left(\left|T(\boldsymbol{\omega}) - \sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right| \ge \epsilon\|\boldsymbol{\omega}\|_2\right) \le 2e^{-\frac{c\epsilon^2 d}{K^2}}. \tag{22}$$

2. The sample average operator $T(\boldsymbol{\omega})$ is the empirical mean of d i.i.d. folded normal random variables.

To extend this bound uniformly over all $\boldsymbol{\omega} \in \mathcal{S}^{n-1}$, we begin by approximating the set $\mathcal{B}_2^n \triangleq \{\boldsymbol{\omega} \in \mathbb{R}^n : \|\boldsymbol{\omega}\|_2 \leq 1\}$[3] using a net $\mathcal{N}$. Leveraging the concentration bound (22), we then apply a union bound over all $\boldsymbol{\omega} \in \mathcal{N}$. For the approximation step, we employ a ρ-net $\mathcal{N}$ of the set $\mathcal{B}_2^n$. Consequently, for any $\boldsymbol{\omega} \in \mathcal{B}_2^n$, one can find $\boldsymbol{\omega}' \in \mathcal{N}$ such that $\|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2 \leq \rho$. This allows us to express the following:

$$\begin{aligned}
\left|T(\boldsymbol{\omega}) - \sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right| &\leq \left|T(\boldsymbol{\omega}') - \sqrt{\frac{2}{\pi}}\left\|\boldsymbol{\omega}'\right\|_2\right| + \left|T(\boldsymbol{\omega}) - T(\boldsymbol{\omega}')\right| + \sqrt{\frac{2}{\pi}}\left|\|\boldsymbol{\omega}\|_2 - \left\|\boldsymbol{\omega}'\right\|_2\right| \\
&\leq \left|T(\boldsymbol{\omega}') - \sqrt{\frac{2}{\pi}}\left\|\boldsymbol{\omega}'\right\|_2\right| + \left|T(\boldsymbol{\omega}) - T(\boldsymbol{\omega}')\right| + \sqrt{\frac{2}{\pi}}\left\|\boldsymbol{\omega} - \boldsymbol{\omega}'\right\|_2 \\
&\leq \left|T(\boldsymbol{\omega}') - \sqrt{\frac{2}{\pi}}\left\|\boldsymbol{\omega}'\right\|_2\right| + \underbrace{\left|T(\boldsymbol{\omega}) - T(\boldsymbol{\omega}')\right|}_{\text{Term } \star} + \sqrt{\frac{2}{\pi}}\rho.
\end{aligned} \tag{23}$$

To upper bound Term $\star$ in (23), we can write

$$\begin{aligned}
\left|T(\boldsymbol{\omega}) - T(\boldsymbol{\omega}')\right| &= \frac{1}{d}\left|\sum_{j=1}^{d}\left|\mathbf{x}_j^\top\boldsymbol{\omega}\right| - \left|\mathbf{x}_j^\top\boldsymbol{\omega}'\right|\right| \\
&\leq \frac{1}{d}\sum_{j=1}^{d}\left|\left|\mathbf{x}_j^\top\boldsymbol{\omega}\right| - \left|\mathbf{x}_j^\top\boldsymbol{\omega}'\right|\right| \\
&\leq \frac{1}{d}\sum_{j=1}^{d}\left|\mathbf{x}_j^\top\left(\boldsymbol{\omega} - \boldsymbol{\omega}'\right)\right| \\
&= \frac{2}{d}\sum_{j=1}^{d}\left|\mathbf{x}_j^\top\left(\frac{\boldsymbol{\omega} - \boldsymbol{\omega}'}{2}\right)\right| \\
&= 2T\left(\frac{\boldsymbol{\omega} - \boldsymbol{\omega}'}{2}\right).
\end{aligned} \tag{24}$$

Let us introduce the smallest constant $\delta \geq 0$ such that

$$\left|T(\boldsymbol{\omega}) - \sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right| \leq \delta\|\boldsymbol{\omega}\|_2, \quad \forall \boldsymbol{\omega} \in \mathcal{B}_2^n. \tag{25}$$

Note that $\frac{1}{2}\left(\boldsymbol{\omega} - \boldsymbol{\omega}'\right) \in \frac{\rho}{2}\mathcal{B}_2^n \subset \mathcal{B}_2^n$. Therefore, using (25), we obtain

$$T\left(\frac{\boldsymbol{\omega} - \boldsymbol{\omega}'}{2}\right) \leq \frac{1}{2}\left(\sqrt{\frac{2}{\pi}} + \delta\right)\left\|\boldsymbol{\omega} - \boldsymbol{\omega}'\right\|_2 \leq \left(\sqrt{\frac{2}{\pi}} + \delta\right)\frac{\rho}{2}. \tag{26}$$

Combining (26) with (24) and substituting into (23) yields

$$\left|T(\boldsymbol{\omega}) - \sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right| \leq \left|T(\boldsymbol{\omega}') - \sqrt{\frac{2}{\pi}}\left\|\boldsymbol{\omega}'\right\|_2\right| + \left(2\sqrt{\frac{2}{\pi}} + \delta\right)\rho. \tag{27}$$

3. We approximate the set $\mathcal{B}_2^n$ to facilitate a tight upper bound on Term $\star$ in (23).

Taking the supremum over $\boldsymbol{\omega} \in \mathcal{B}_2^n$, we obtain

$$\sup_{\boldsymbol{\omega}\in\mathcal{B}_2^n}\left|T(\boldsymbol{\omega})-\sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right| \leq \underbrace{\sup_{\boldsymbol{\omega}'\in\mathcal{N}}\left|T(\boldsymbol{\omega}')-\sqrt{\frac{2}{\pi}}\left\|\boldsymbol{\omega}'\right\|_2\right|}_{\text{Term }\star\star}+\left(2\sqrt{\frac{2}{\pi}}+\delta\right)\rho. \tag{28}$$

To control Term $\star\star$, we apply the concentration bound from (22) via a union bound:

$$\mathbb{P}\left(\sup_{\boldsymbol{\omega}'\in\mathcal{N}}\left|T(\boldsymbol{\omega}')-\sqrt{\frac{2}{\pi}}\left\|\boldsymbol{\omega}'\right\|_2\right|\geq\epsilon\|\boldsymbol{\omega}'\|_2\right)\leq 2\mathcal{N}\left(\mathcal{B}_2^n,\|\cdot\|_2,\rho\right)e^{-\frac{c\epsilon^2 d}{K^2}}, \tag{29}$$

where $\mathcal{N}\left(\mathcal{B}_2^n,\|\cdot\|_2,\rho\right)$ denotes the covering number of $\mathcal{B}_2^n$ with respect to the Euclidean norm. It is well known that

$$\mathcal{N}\left(\mathcal{B}_2^n,\|\cdot\|_2,\rho\right)\leq\left(1+\frac{2}{\rho}\right)^n=e^{n\log\left(1+\frac{2}{\rho}\right)}. \tag{30}$$

By combining (29) with (28), the following holds with probability at least $1-2e^{n\log\left(1+\frac{2}{\rho}\right)-\frac{c\epsilon^2 d}{K^2}}$:

$$\sup_{\boldsymbol{\omega}\in\mathcal{B}_2^n}\left|T(\boldsymbol{\omega})-\sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right|\leq\epsilon+\left(2\sqrt{\frac{2}{\pi}}+\delta\right)\rho. \tag{31}$$

Following the assumption in (25), we require that

$$\delta\leq\epsilon+\left(2\sqrt{\frac{2}{\pi}}+\delta\right)\rho,\quad\text{or}\quad\delta\leq\frac{\epsilon+2\sqrt{\frac{2}{\pi}}\rho}{1-\rho}. \tag{32}$$

To ensure that this condition holds, we choose $\epsilon=\frac{\delta}{2}$ and $\rho=\frac{\delta}{2\delta+4\sqrt{\frac{2}{\pi}}}$. Substituting these values into (31) guarantees the following bound:

$$\sup_{\boldsymbol{\omega}\in\mathcal{B}_2^n}\left|T(\boldsymbol{\omega})-\sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right|\leq\delta, \tag{33}$$

with probability at least $1-2e^{n\log\left(5+4\sqrt{\frac{2}{\pi}}\delta^{-1}\right)-\frac{c\delta^2 d}{4K^2}}$. To ensure this probability is at least $1-2e^{-\frac{c\delta^2 d}{8K^2}}$, it suffices that

$$e^{n\log\left(5+4\sqrt{\frac{2}{\pi}}\delta^{-1}\right)}\leq e^{\frac{c\delta^2 d}{8K^2}}, \tag{34}$$

which leads to

$$d\geq\frac{8K^2}{c}\delta^{-2}n\log\left(5+4\sqrt{\frac{2}{\pi}}\delta^{-1}\right). \tag{35}$$

Since $\mathcal{S}^{n-1}\subset\mathcal{B}_2^n$, we have

$$\sup_{\boldsymbol{\omega}\in\mathcal{S}^{n-1}}\left|T(\boldsymbol{\omega})-\sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right|\leq\sup_{\boldsymbol{\omega}\in\mathcal{B}_2^n}\left|T(\boldsymbol{\omega})-\sqrt{\frac{2}{\pi}}\|\boldsymbol{\omega}\|_2\right|, \tag{36}$$

which completes the proof. ■

In the following theorem, we provide an upper bound on the estimation error for any estimator that satisfies the EC:

Theorem 6 *Let $\bar{\boldsymbol{\omega}} \in \mathcal{S}^{n-1}$ be an estimator obtained by any arbitrary algorithm addressing the optimization problem $\mathcal{P}$ in (14), and let $\boldsymbol{\omega}^{\text{opt}}$ denote the global minimizer of $\mathcal{P}$. Suppose that $\bar{\boldsymbol{\omega}}$ satisfies the EC in Definition 2. Then, with probability at least $1 - 2e^{-c_1\delta^2 d}$, the estimation error is bounded as:*

$$\left\|\bar{\boldsymbol{\omega}} - \boldsymbol{\omega}^{\text{opt}}\right\|_2 \leq c_2\sqrt{\delta}, \tag{37}$$

provided that the number of samples d satisfies $d \geq c_3\delta^{-2}n\log\left(c_4 + c_5\delta^{-1}\right)$ for some universal constants $c_1, \cdots, c_5 > 0$.

Proof The proof builds on the result of Theorem 4. Define $\boldsymbol{\zeta} = \frac{1}{2}\left(\bar{\boldsymbol{\omega}} + \boldsymbol{\omega}^{\text{opt}}\right)$. Then, for any $j \in [d]$, multiplying both sides by $\mathbf{x}_j^\top$ gives $\mathbf{x}_j^\top\boldsymbol{\zeta} = \frac{1}{2}\left(\mathbf{x}_j^\top\bar{\boldsymbol{\omega}} + \mathbf{x}_j^\top\boldsymbol{\omega}^{\text{opt}}\right)$. Under the assumption that $\bar{\boldsymbol{\omega}}$ satisfies the EC, we have

$$\left|\mathbf{x}_j^\top\boldsymbol{\zeta}\right| = \frac{1}{2}\left(\left|\mathbf{x}_j^\top\bar{\boldsymbol{\omega}}\right| + \left|\mathbf{x}_j^\top\boldsymbol{\omega}^{\text{opt}}\right|\right), \quad j \in [d]. \tag{38}$$

Taking the empirical average over $j \in [d]$, this yields

$$T(\boldsymbol{\zeta}) = \frac{1}{2}\left(T\left(\bar{\boldsymbol{\omega}}\right) + T\left(\boldsymbol{\omega}^{\text{opt}}\right)\right). \tag{39}$$

Since $\bar{\boldsymbol{\omega}}, \boldsymbol{\omega}^{\text{opt}} \in \mathcal{S}^{n-1}$, then $\boldsymbol{\zeta} \in \frac{1}{2}\left(\mathcal{S}^{n-1} + \mathcal{S}^{n-1}\right)$, where $+$ denotes the Minkowski sum. Consequently, $\boldsymbol{\zeta}$ does not necessarily belong to the unit sphere. However, we can still apply the result of Theorem 4 to $\boldsymbol{\zeta}$, since this theorem is valid not only over $\mathcal{S}^{n-1}$ but also over $\mathcal{B}_2^n$[4]. Therefore, by Theorem 4, with probability at least $1 - 2e^{-c_1\delta^2 d}$, we have

$$\sqrt{\frac{2}{\pi}}\|\boldsymbol{\zeta}\|_2 \geq T(\boldsymbol{\zeta}) - \delta\|\boldsymbol{\zeta}\|_2, \tag{40}$$

provided that the number of samples d satisfies the condition in (18). Substituting (39) into (40), we obtain

$$\begin{aligned}\sqrt{\frac{2}{\pi}}\|\boldsymbol{\zeta}\|_2 &\geq \frac{1}{2}\left[T\left(\bar{\boldsymbol{\omega}}\right) + T\left(\boldsymbol{\omega}^{\text{opt}}\right)\right] - \delta\|\boldsymbol{\zeta}\|_2 \\ &\geq \frac{1}{2}\left[\left(\sqrt{\frac{2}{\pi}} - \delta\right)\|\bar{\boldsymbol{\omega}}\|_2 + \left(\sqrt{\frac{2}{\pi}} - \delta\right)\left\|\boldsymbol{\omega}^{\text{opt}}\right\|_2\right] - \delta\|\boldsymbol{\zeta}\|_2,\end{aligned} \tag{41}$$

where the second inequality uses the result of Theorem 4. Based on the definition of $\boldsymbol{\zeta}$, we then obtain

$$\begin{aligned}\left\|\bar{\boldsymbol{\omega}} + \boldsymbol{\omega}^{\text{opt}}\right\|_2 &\geq \|\bar{\boldsymbol{\omega}}\|_2 + \left\|\boldsymbol{\omega}^{\text{opt}}\right\|_2 - \delta\sqrt{2\pi}\left(\|\bar{\boldsymbol{\omega}}\|_2 + \left\|\boldsymbol{\omega}^{\text{opt}}\right\|_2\right) \\ &= 2 - 2\sqrt{2\pi}\delta.\end{aligned} \tag{42}$$

By the parallelogram law, we conclude that

$$\begin{aligned}\left\|\bar{\boldsymbol{\omega}} - \boldsymbol{\omega}^{\text{opt}}\right\|_2^2 &= 4 - \left\|\bar{\boldsymbol{\omega}} + \boldsymbol{\omega}^{\text{opt}}\right\|_2^2 \\ &\leq 4 - (2 - 2\sqrt{2\pi}\delta)^2 \\ &\leq 8\sqrt{2\pi}\delta,\end{aligned} \tag{43}$$

4. This extension is justified by the covering argument presented in the proof of Theorem 4, which establishes the bound uniformly over $\mathcal{B}_2^n$.

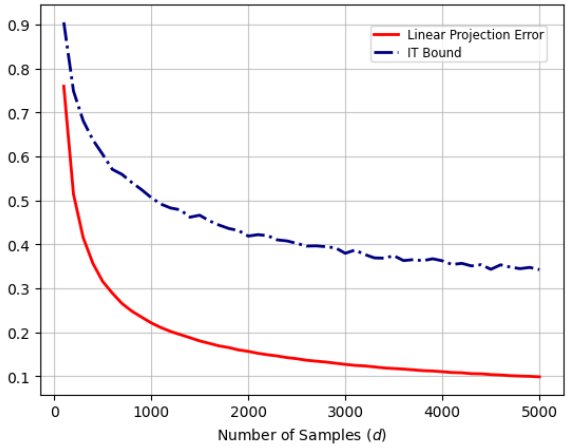


Figure 11: Comparison between the estimation error of the linear projection method and the IT bound from Theorem 6. Notably, the empirical error of the linear projection method consistently remains below the theoretical bound, demonstrating its effectiveness in this setting.

which completes the proof. ■

Based on the result of Theorem 6, one can simply observe that the upper bound on the estimation error in (37) decreases as the number of samples d increases (this decrease is at a rate of $d^{-\frac{1}{4}}$).

E.2. Numerical Examples

In this section, we numerically compared the estimation error of the linear projection method with the theoretical upper bound established in Theorem 6 (referred to as the IT bound). To conduct this comparison, we generated data $\{\mathbf{x}_j\}_{j=1}^d \in \mathbb{R}^n$ with $n = 50$, and let the number of samples d vary from 100 to 5000 in increments of 100. Each data $\mathbf{x}_j$ was sampled independently from the standard multivariate normal distribution, consistent with the assumptions in Section E.1. Likewise, the ground-truth weight vector $\boldsymbol{\omega}^{\text{opt}} \in \mathbb{R}^n$ was sampled from the standard multivariate normal distribution and subsequently normalized to lie on the unit sphere, again in accordance with the assumptions in Section E.1. The corresponding labels $\{y_j\}_{j=1}^d$ were then generated according to the model specified in (13).

For each value of d, we we repeated the following process 2000 times: (i) the estimation error of the linear projection method, i.e., $\left\|\boldsymbol{\omega}^\ell - \boldsymbol{\omega}^{\text{opt}}\right\|_2$, where $\boldsymbol{\omega}^\ell$ denotes the solution obtained by the linear projection estimator, and (ii) the theoretical upper bound on the estimation error as given in Theorem 6. Since $\boldsymbol{\omega}^\ell$ does not necessarily belong to the unit sphere, we applied normalization to it when computing the error in case (i). Finally, we took the empirical average to report the final error associated with each case. Fig. 11 illustrates the results of this process and compares the estimation errors from case (i) with the IT bound from case (ii) as a function of d. Note that the result of Theorem 6 holds under the assumption that the estimator $\bar{\boldsymbol{\omega}}$ satisfies the EC. In contrast, the linear projection method does not guarantee this condition, as discussed in Section 3. Interestingly, as shown in Fig. 11, the estimation error of the linear projection method consistently falls below

the theoretical bound. While the IT bound is a universal upper bound that must be satisfied by any global minimizer of (14), the results in Fig. 11 suggest that the simple linear projection approach can serve as an effective solution to the optimization problem in (14) (at least in this setting). Therefore, incorporating this method into our proposed YES bound framework appears to be a promising and justified direction.