Generalized Boosting

Arun Sai Suggala, Bingbin Liu, Pradeep Ravikumar
Carnegie Mellon University
Pittsburgh, PA 15213
{asuggala,bingbinl,pradeepr}@cs.cmu.edu

Abstract

Boosting is a widely used learning technique in machine learning for solving classification problems. In boosting, one predicts the label of an example using an ensemble of weak classifiers. While boosting has shown tremendous success on many classification problems involving tabular data, it performs poorly on complex classification tasks involving low-level features such as image classification tasks. This drawback stems from the fact that boosting builds an additive model of weak classifiers, each of which has very little predictive power. Often, the resulting additive models are not powerful enough to approximate the complex decision boundaries of real-world classification problems. In this work, we present a general framework for boosting where, similar to traditional boosting, we aim to boost the performance of a weak learner and transform it into a strong learner. However, unlike traditional boosting, our framework allows for more complex forms of aggregation of weak learners. In this work, we specifically focus on one form of aggregation - *function composition*. We show that many popular greedy algorithms for learning deep neural networks (DNNs) can be derived from our framework using function compositions for aggregation. Moreover, we identify the drawbacks of these greedy algorithms and propose new algorithms that fix these issues. Using thorough empirical evaluation, we show that our learning algorithms have superior performance over traditional additive boosting algorithms, as well as existing greedy learning techniques for DNNs. An important feature of our algorithms is that they come with strong theoretical guarantees.

1 Introduction

Boosting is a widely used learning technique in machine learning for solving classification problems. Boosting aims to improve the performance of a weak learner by combining multiple weak classifiers to produce a strong classifier with good predictive performance. Since the seminal works of Freund [13], Schapire [32], a number of practical algorithms such as AdaBoost [16], gradient boosting [26], XGBoost [9], have been proposed for boosting. Over the years, boosting based methods such as XGBoost in particular, have shown tremendous success in many real-world classification problems, as well as competitive settings such as Kaggle competitions. However, this success is mostly limited to classification tasks involving structured or tabular data with hand-engineered features. On classification problems involving low-level features and complex decision boundaries, boosting tends to perform poorly [3, 30] (also see Section 5). One example where this is evident is the image classification task, where the decision boundaries are often complex and the features are low-level pixel intensities. This drawback stems from the fact that boosting builds an additive model of weak classifiers, each of which has very little predictive power. Since such additive models with any reasonable number of weak classifiers are usually not powerful enough to approximate complex decision boundaries, the models' output by boosting tend to have poor performance.

In this work, we aim to overcome this drawback of traditional boosting by considering a generalization of boosting which allows for more complex forms of aggregation than linear combinations of weak classifiers. To achieve this goal, we work in the feature representation space and boost the performance of *weak feature transformers*. Working in the representation space allows for more flexible combinations of weak feature transformers. This is unlike traditional boosting which works in the label space and builds an additive model on the predictions of the weak classifiers. The starting point for our approach is the greedy view of boosting, originally studied by Friedman et al. [18], Mason et al. [26]. Letting $\widehat{R}_S(f)$ be the risk of a classifier f on training samples S, boosting techniques aim to approximate the minimizer of $\widehat{R}_S$ in terms of linear combinations of elements from a set of weak classifiers $\mathcal{F}$. Many popular boosting algorithms including AdaBoost, XGBoost, rely on greedy techniques to find such an approximation. In our generalized framework for boosting, we take this greedy view, but differ in how we aggregate the weak learners. We approximate the minimizer of $\widehat{R}_S$ using models of the form $f_T = W\phi_T$, where $\phi_T = \sum_{t=0}^{T} g_t$, and $\{g_t\}_{t=0}^{T}$ are feature transformations learned in each iteration of the greedy algorithm, and W is the linear classifier on top of the feature transformation. Unlike additive boosting, where each g_t comes from a fixed weak feature transformer class $\mathcal{G}$, in our framework each g_t comes from a class $\mathcal{G}_t$ which evolves over time t and is allowed to depend on the past iterates $\{\phi_i\}_{i=0}^{t-1}$. Some potential choices for $\mathcal{G}_t$ that could be of interest are $\{g \circ \phi_{t-1} \text{ for } g \in \mathcal{G}\}$, $\{g \circ ([\phi_0, \dots, \phi_{t-1}]) \text{ for } g \in \mathcal{G}\}$, where $g \circ \phi(\mathbf{x}) = g(\phi(\mathbf{x}))$ denotes function composition of g and ϕ, and $\mathcal{G}$ is a weak feature transformer class. Note that the former choice of $\mathcal{G}_t$ is connected to layer-by-layer training of models with ResNet architecture [21].

As one particular instantiation of our framework, we consider weak feature transformers that are neural networks and use function compositions to combine them; that is, we use $\mathcal{G}_t$'s constructed using function compositions. We show that for certain choices of $\mathcal{G}_t$, our framework recovers the layer-by-layer training techniques developed in deep learning [6, 22]. Greedy layer-by-layer training techniques have seen a revival in recent years [5, 8, 22, 25, 29]. One reason for this revival is that greedy techniques consume less memory than end-to-end training of deep networks, as they do not perform end-to-end back-propagation. Consequently, they can accommodate much larger models in limited memory. As a primary contribution of the paper, we identify several drawbacks of existing layer-by-layer training techniques, and show that the choice of $\mathcal{G}_t$ used by these algorithms can lead to a drop in performance. We propose alternative choices for $\mathcal{G}_t$ which fix these issues and empirically demonstrate that the resulting algorithms have superior performance over existing layer-by-layer training techniques, and in some cases achieve performance close to that of end-to-end trained DNNs. Moreover, we show that the proposed algorithms perform much better than traditional additive boosting algorithms, on a variety of classification tasks.

As the second contribution of the paper, we provide excess risk bounds for models learned using our generalized boosting framework. Our results depend on a certain weak learning condition on feature transformer classes $\{\mathcal{G}_t\}_{t=1}^{T}$, which is a natural generalization of the weak learning condition that is typically imposed in traditional boosting. The resulting risk bounds are modular and depend on the generalization bounds of $\{\mathcal{G}_t\}_{t=1}^{T}$. An advantage of such modular bounds is that one can rely on the best-known generalization bounds for weak transformation classes $\{\mathcal{G}_t\}_{t=1}^{T}$ and obtain tight risk bounds for boosting. As an immediate consequence of this result, we obtain excess risk bounds for existing greedy layer-by-layer training techniques.

Related Work. Several works have proposed generalizations of traditional boosting [10, 11, 20, 22]. Cortes et al. [10] propose a boosting algorithm where the hypothesis set of weak classifiers is chosen adaptively. However, the resulting models are still additive models of weak classifiers and usually perform poorly on hard classification problems. Several recent works have attempted to learn neural networks greedily based on boosting theory. Cortes et al. [11] propose a boosting-style algorithm to learn both the structure and weights of neural networks in an adaptive way. However, the algorithms developed are restricted to feed forward neural networks and are mostly theoretical in nature. The experimental evidence in the paper is a proof-of-concept and only considers small scale binary classification tasks. Huang et al. [22], Nitanda and Suzuki [29] use ideas from classical boosting to learn neural networks in a layer-by-layer fashion. As we show later, these algorithms are specific instances of our generalized framework, and have certain drawbacks arising from the choice of $\mathcal{G}_t$ they use.

2 Preliminaries

In this section, we set up the notation and review the necessary background on additive boosting. A consolidated list of notation can be found in Appendix A.

Notation. Let $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ denote a feature-label pair following a probability distribution P. Let P^X, P^Y denote the marginal distributions of X and Y. In this work, we consider the multi-class classification problem where $\mathcal{Y} = \{0, \ldots K-1\}$, and assume $\mathcal{X} \subseteq \mathbb{R}^d$. Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be n i.i.d samples drawn from P. Let P_n be the empirical distribution of S and P_n^X, P_n^Y be the marginal distributions of $\{\mathbf{x}_i\}_{i=1}^n$, $\{y_i\}_{i=1}^n$.

In classification, our goal is to find a predictor that can well predict the label of any feature from just the samples S. Let $f : \mathcal{X} \to \mathbb{R}^K$ denote a score-based classifier which assigns X to class $\operatorname{argmax}_i f_i(X)$. The expected classification risk of f is defined as $\mathbb{E}_{X,Y}\left[\ell_{0-1}(f(X), Y)\right]$, where $\ell_{0-1}(f(X), Y) = 0$ if $\operatorname{argmax}_i f_i(X) = Y$, and 1 otherwise. Since optimizing $0/1$ risk is computationally intractable, we consider convex surrogates of $\ell_{0-1}(f(X), Y)$, which we denote by $\ell(f(X), Y)$; typical choices for ℓ include the logistic loss and the exponential loss. The population risk of f is then defined as $R(f) = \mathbb{E}_{X,Y}\left[\ell(f(X), Y)\right]$. Since directly optimizing the population risk is impossible, we approximate it with the empirical risk $\widehat{R}_S(f) = \frac{1}{n}\sum_{i=1}^n \ell(f(\mathbf{x}_i), y_i)$ and try to find its minimizer.

We consider classifiers of the form $f(X) = W\phi(X)$, where $\phi : \mathcal{X} \to \mathbb{R}^D$ is the feature transformer and $W \in \mathbb{R}^{K \times D}$ is the linear classifier on top. A popular choice for ϕ is a neural network. We denote the population and empirical risks of such an f as $R(W, \phi)$, $\widehat{R}_S(W, \phi)$. We usually work in the space of feature transforms. Let $L_2(P)$ denote the space of square integrable functions w.r.t P, and define the inner product between $\phi_1, \phi_2 \in L_2(P)$ as $\langle \phi_1, \phi_2 \rangle_P = \mathbb{E}_{X \sim P}\left[\langle \phi_1(X), \phi_2(X) \rangle\right]$. We denote with $\nabla_\phi R(W, \phi)$ the functional gradient of $R(W, \phi)$ w.r.t ϕ in the $L_2(P^X)$ space, which is defined as $\nabla_\phi R(W, \phi)(\mathbf{x}) = \mathbb{E}_{Y|\mathbf{x}}\left[W^T \nabla \ell(W\phi(\mathbf{x}), Y)\right]$, where $\nabla \ell(W\phi(\mathbf{x}), y)$ denotes the gradient of ℓ w.r.t its first argument, evaluated at $W\phi(\mathbf{x})$. Similarly, we let $\nabla_\phi \widehat{R}_S(W, \phi)$ denote the functional gradient of $\widehat{R}_S(W, \phi)$ in the $L_2(P_n^X)$ space

$$\nabla_\phi \widehat{R}_S(W, \phi)(\mathbf{x}) = \begin{cases} W^T \nabla \ell(W\phi(\mathbf{x}_i), y_i), & \text{if } \mathbf{x} = \mathbf{x}_i, \\ 0 & \text{otherwise} \end{cases}.$$

Additive Boosting. In this work, we refer to traditional boosting as additive boosting, as it constructs additive models of weak classifiers. Let $\mathcal{F}$ be a hypothesis class of weak classifiers, a typical example being decision trees of bounded depth. Additive boosting aims to find an element in the linear span of $\mathcal{F}$ which minimizes the empirical risk $\widehat{R}_S(f)$. As previously mentioned, there exists a duality between boosting and greedy algorithms [18, 19, 26]. Many popular boosting algorithms use a greedy forward stagewise approach to find a minimizer of $\widehat{R}_S(f)$, and solve the following in each iteration:

$$\eta_t, f_t = \operatorname{argmin}_{\eta \in \mathbb{R}, f \in \mathcal{F}} \widehat{R}_S\left(\sum_{i=1}^{t-1} \eta_i f_i + \eta f\right),$$

where η is the learning rate. Various algorithms differ in how they solve this optimization problem. In gradient boosting, one uses a linear approximation of $\widehat{R}_S$ around $\sum_{i=1}^{t-1} \eta_i f_i$ [26]. In this work, we take this greedy view of boosting to design the generalized boosting framework.

Additive Representation Boosting. In this work, we perform boosting in the representation space, contrasting with traditional boosting which works in the output space. Let $\mathcal{G}$ be a hypothesis class of *weak feature transformers*, whose examples include the set of one layer neural networks of bounded width and a set of vector-valued polynomials of bounded degree. More generally, $\mathcal{G}$ can be any set of non-linear transformations. In additive representation boosting, we aim to find a strong feature transform ϕ in the linear span of $\mathcal{G}$, and a linear predictor $W \in \mathcal{W} \subseteq \mathbb{R}^{K \times D}$ that minimizes $\widehat{R}_S(W, \phi)$. To this end, we consider greedy algorithms that solve the following problem each iteration:

$$W_t, g_t = \operatorname{argmin}_{W \in \mathcal{W}, g \in \mathcal{G}} \widehat{R}_S\left(W, \phi_{t-1} + \eta_t g\right), \tag{1}$$

where $\phi_t = \phi_0 + \sum_{i=1}^t \eta_i g_i$ with ϕ_0 being the initial feature transformation, and $\{\eta_i\}_{i=1}^\infty$ is a predefined learning rate schedule.

3 Generalized Boosting

The starting point for our generalized boosting framework is the additive representation boosting described in Section 2. Typically, linear combinations of weak feature transformations are not powerful enough to model complex decision boundaries. Consequently, the minimizer of $\widehat{R}_S(W, \phi)$ over the linear span of $\mathcal{G}$ tends to have a high risk. A simple workaround for this issue would be to perform additive boosting with a complex hypothesis class $\mathcal{G}$. For example, if the weak feature transformers are one layer neural networks, then one could increase the complexity of $\mathcal{G}$ by using deeper networks. However, such an alternative has several drawbacks both from an optimization and generalization perspective and defeats the purpose of boosting, which aims to convert weak learners into strong learners. From an optimization perspective, moving to complex $\mathcal{G}$ makes each greedy step harder to optimize. For example, compared to deep neural networks, shallow networks are easier to optimize, require fewer resources, and are easier to analyze or interpret [5]. From a generalization perspective, since the generalization bounds of boosting depend on the complexity of $\mathcal{G}$, larger hypothesis classes can lead to overfitting and poor performance on unseen data.

In this work, we are interested in other approaches for increasing the complexity of models produced by boosting, while ensuring the boosting/greedy steps are easy to implement. One way to achieve this is by considering more complex combinations of weak feature transformers than the linear combinations considered in additive representation boosting. Formally, let $\mathcal{G}_t$ denote the hypothesis class of feature transformations used in the t^{th} iteration of boosting. In additive boosting, $\mathcal{G}_t = \mathcal{G}$ for all t. In our generalized boosting framework, we increase the complexity of $\mathcal{G}_t$ by letting it depend on the past iterates $\{\phi_i\}_{i=0}^{t-1}$. Here are some potential choices for $\mathcal{G}_t$, other than the ones stated in the introduction: $\{g \circ (\sum_{i=0}^{t-1} \alpha_i \phi_i),\ \text{for } g \in \mathcal{G}, \alpha_i \in \mathbb{R}\}$, $\{g \circ \phi_{t-1} \circ \phi_{t-2} \cdots \circ \phi_0,\ \text{for } g \in \mathcal{G}\}$. Depending on the problem domain, one could consider several other ways of constructing $\mathcal{G}_t$ using the past iterates. Note that even with these complex choices of $\mathcal{G}_t$, the greedy steps are easy to implement and only need a weak learner which can identify an element in $\mathcal{G}$ that best fits the data. As a result, this remains in the spirit of boosting and at the same time ensures the models learned are complex enough for real world problems.

We now present our algorithm for generalized boosting (see Algorithm 1). Similar to additive representation boosting, our algorithm proceeds in a greedy fashion. In the t^{th} iteration of the algorithm, we aim to solve the following optimization problem:

$$W_t, g_t = \underset{W \in \mathcal{W}, g \in \mathcal{G}_t}{\operatorname{argmin}}\ \widehat{R}_S\left(W, \phi_{t-1} + \eta_t g\right). \tag{2}$$

We provide two approaches for solving this problem. One is the *exact greedy approach*, which directly solves the optimization problem (Algorithm 2). For problems where direct optimization of Equation (2) is difficult[1], we provide an approximate technique which performs functional gradient descent on the objective. In this approach, which we call *gradient greedy approach*, we approximate the objective with the linear approximation of $\widehat{R}_S$ around ϕ_{t-1} (Algorithm 3):

$$\widehat{R}_S\left(W, \phi_{t-1} + \eta_t g\right) \approx \widehat{R}_S\left(W, \phi_{t-1}\right) + \eta_t \left\langle \nabla_\phi \widehat{R}_S(W, \phi_{t-1}), g \right\rangle_{P_n^X}.$$

To optimize the linear approximation, we first fix W to W_{t-1} and find a minimizing $g_t \in \mathcal{G}_t$. Intuitively, this step can be seen as finding a g which best aligns with the negative functional gradient of empirical risk at the current iterate. For appropriate choice of learning rate η, moving along g_t results in reduction of $\widehat{R}_S$. Next, we fix g_t and find a linear predictor W which minimizes the empirical risk $\widehat{R}_S(W, \phi_t)$. This alternating optimization of g and W makes the algorithm easy to implement in practice. Moreover, this algorithm is more stable than joint optimization of g and W. We note that such gradient greedy approaches have been developed for traditional boosting [26].

3.1 Compositional Boosting

As one particular instantiation of our framework, we consider $\mathcal{G}_t$'s constructed by composing elements from a weak feature transformer class $\mathcal{G}$ with the past iterates $\{\phi_i\}_{i=0}^{t-1}$ and study the resulting boosting algorithms. We refer to such boosting algorithms as *compositional boosting* algorithms since the strong feature transformer is constructed from weak feature transformer via function composition. When $\mathcal{G}_t = \{g \circ \phi_{t-1} \text{ for } g \in \mathcal{G}\}$, the models in our framework have the ResNet architecture and can be defined recurrently as $\phi_t = \phi_{t-1} + \eta_t g_t \circ \phi_{t-1}$. Moreover, Algorithm 1 with this choice of $\mathcal{G}_t$ and Algorithm 2 as update routine give us the greedy layer-wise supervised training technique proposed by Bengio et al. [6] and recently revisited by Belilovsky et al. [5]. In another recent work, Huang et al. [22] propose a boosting-based algorithm for learning ResNets (see Algorithm 4 in Appendix). We now show that their approach is equivalent to the greedy technique of Bengio et al. [6], and thus can be seen as an instance of our general framework. We note that such a connection is not known previously.

Proposition 3.1. *Suppose the classification loss ℓ is the exponential loss. Then the greedy technique of Huang et al. [22] for learning ResNets is equivalent to the greedy layer-wise supervised training technique of Bengio et al. [6].*

[1]Such scenarios can potentially arise if the feature transformations are non-differentiable functions.

In another recent work, Nitanda and Suzuki [29] propose a gradient boosting technique to greedily learn a ResNet. This algorithm is closely related to the gradient greedy approach described in Algorithm 3, with $\mathcal{G}_t = \{g \circ \phi_{t-1} \text{ for } g \in \mathcal{G}\}$.

Algorithm 1 Generalized Boosting

```
1: Input: Training data $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, iterations $T$, initial linear predictor $W_0$, initial feature transformer
   $\phi_0$, learning rates $\{\eta_i\}_{i=1}^T$, Update-routine: UPDATE
2: $t \leftarrow 1$
3: while $t \leqslant T$ do
4:     Construct feature transformer class $\mathcal{G}_t$ based on past iterates $\{(W_i, \phi_i)\}_{i=0}^{t-1}$
5:     $W_t, \phi_t, g_t \leftarrow$ UPDATE $(S, W_{t-1}, \phi_{t-1}, \eta_t, \mathcal{G}_t)$
6:     $t \leftarrow t + 1$
7: end while
8: Return: $W_T, \phi_T$
```

Algorithm 2 Exact Greedy Update

```
1: Input: Training data $S$, previous iterate $(W, \phi)$, learning rate $\eta$, feature transformer class $\mathcal{G}$
2:
   $$W^+, g^+ \leftarrow \underset{\widetilde{W} \in \mathcal{W}, \tilde{g} \in \mathcal{G}}{\operatorname{argmin}} \widehat{R}_S(\widetilde{W}, \phi + \eta\tilde{g})$$
3: $\phi^+ \leftarrow \phi + \eta g^+$
4: Return: $W^+, \phi^+, g^+$
```

Algorithm 3 Gradient Greedy Update

```
1: Input: Training data $S$, previous iterate $(W, \phi)$, learning rate $\eta$, feature transformer class $\mathcal{G}$
2: // Pick a descent direction
3: $g^+ \leftarrow \operatorname{argmin}_{\tilde{g} \in \mathcal{G}} \left\langle \nabla_\phi \widehat{R}_S(W, \phi), \tilde{g} \right\rangle_{P_n^X}$
4: $\phi^+ \leftarrow \phi + \eta g^+$
5: // Update the linear predictor
6: $W^+ \leftarrow \operatorname{argmin}_{\widetilde{W} \in \mathcal{W}} \widehat{R}_S(\widetilde{W}, \phi^+)$
7: Return: $W^+, \phi^+, g^+$
```

We now highlight certain drawbacks of the existing greedy layer-wise training techniques, which arise from the particular choice of $\mathcal{G}_t$ used by these algorithms. Since $\{g \circ \phi_{t-1} \text{ for } g \in \mathcal{G}\}$ is constructed solely based on the past iterate ϕ_{t-1}, any mistake in ϕ_{t-1} is propagated to all the future iterates. As a result, these algorithms can not recover from their past mistakes. As an example, consider the following scenario where two points $\mathbf{x}_1, \mathbf{x}_2$ belonging to two different classes are placed close to each other in the feature space, after 1^{st} iteration of greedy; that is $\phi_1(\mathbf{x}_1) \approx \phi_1(\mathbf{x}_2)$. In such a scenario, the future iterates $\{\phi_t\}_{t=2}^\infty$ generated by existing greedy algorithms will always place $\mathbf{x}_1, \mathbf{x}_2$ close to each other in the representation space. As a result, the algorithm will always misclassify at least one of $\mathbf{x}_1, \mathbf{x}_2$. Another issue with existing greedy techniques is that they do not guarantee that the complexity of $\mathcal{G}_t$ increases with time t. In such scenarios, Algorithm 1 doesn't make much progress in each iteration and can result in poor models. As an example, consider the setting where $\mathcal{G}$ is the set of all linear transformations. Suppose ϕ_0 is the identity transform and ϕ_1 is such that its range lies in a low dimensional subspace. Then, it is evident that $\mathcal{G}_1 \supseteq \mathcal{G}_t$ for all $t \geqslant 2$.

To fix these issues, we propose two new compositional boosting algorithms obtained with a more careful choice of $\mathcal{G}_t$. In our first algorithm, which we call DenseCompBoost, we choose $\mathcal{G}_t$ as follows

$$\mathcal{G}_t = \left\{ g \circ \left(\mathrm{Id} + \sum_{i=0}^{t-1} \alpha_i \phi_i \right), \text{ for } g \in \mathcal{G}, \alpha_i \in \mathbb{R} \right\}, \tag{3}$$

where $\mathrm{Id}(\cdot)$ is the identify function. Such a choice of $\mathcal{G}_t$ helps us recover from the past mistakes. For example, if ϕ_1 is a constant function, then the algorithm can still learn a good feature transformer by relying on the input $\mathbf{x}$ and the initial feature transform ϕ_0. Moreover, our choice of $\mathcal{G}_t$ ensures its complexity grows with t and satisfies: $\mathcal{G}_{t-1} \subseteq \mathcal{G}_t$, for all t. We call our algorithm DenseCompBoost, since the resulting model for this choice of $\mathcal{G}_t$ resembles a DenseNet [23], where each layer is allowed to be connected to all the previous layers. That being said, the models output by DenseCompBoost differ from DenseNet in how they aggregate the previous layers. DenseNet concatenates the features from previous layers, whereas DenseCompBoost adds the features. Our second algorithm, which we call CmplxCompBoost, tries to increase the complexity of $\mathcal{G}_t$ in each iteration as follows

$$\mathcal{G}_t = \left\{ g \circ \phi_{t-1}, \text{ for } g \in \widetilde{\mathcal{G}}_t \right\}, \tag{4}$$

where $\widetilde{\mathcal{G}}_t$ is a weak feature transformer class and satisfies $\widetilde{\mathcal{G}}_{t-1} \subset \widetilde{\mathcal{G}}_t$ for all t. In the case of one layer neural networks, such $\widetilde{\mathcal{G}}_t$'s can be constructed by increasing the layer width with t. We note that the $\widetilde{\mathcal{G}}_t$ in this algorithm is independent of the past iterates. By increasing the complexity of $\widetilde{\mathcal{G}}_t$ with t, we expect the complexity of $\mathcal{G}_t$ to increase and Algorithm 1 to make more progress in each iteration. While not immediately evident, we note that this technique can also fix the mistakes made by past iterates. For example, suppose ϕ_1 is such that it places two points $\mathbf{x}_1, \mathbf{x}_2$ from different classes, close to each other in the feature space. Then having a more complex $\widetilde{\mathcal{G}}_2$ can help recover from this mistake, as one can potentially find a $g \in \widetilde{\mathcal{G}}_2$ which can separate these two points. In Section 5, we present empirical evidence showing that our new boosting algorithms have superior performance over existing additive and compositional boosting algorithms. Further empirical evidence corroborating the issues we identified with existing layer-wise training techniques can be found in Appendix J.1.

4 Excess Risk Bounds

In this section, we provide excess risk bounds for the models' output by the generalized boosting framework. Our results depend on a *weak learning condition* on the hypothesis class $\mathcal{G}_t$ used in the t^{th} iteration of Algorithm 1. This condition is a way to quantify the relative strength of $\mathcal{G}_t$ and roughly says that there always exists an element in $\mathcal{G}_t$ which has an acute angle with the negative functional gradient at the current iterate. Such a condition ensures progress in each iteration of boosting.

Definition 4.1. *Let $\beta \in (0,1], \epsilon \geqslant 0$ be constants. $\mathcal{G}_{t+1}$ is said to satisfy the (β, ϵ)-weak learning condition for a dataset S, if there exists a $g \in \mathcal{G}_{t+1}$ such that*

$$\left\langle g, -\nabla_\phi \widehat{R}_S(W_t, \phi_t) \right\rangle_{P_n^X} \geqslant \beta B(\mathcal{G}_{t+1}) \|\nabla_\phi \widehat{R}_S(W_t, \phi_t)\|_{P_n^X} - \epsilon,$$

where $B(\mathcal{G}_{t+1}) = \sup_{g \in \mathcal{G}_{t+1}} \|g\|_{P_n^X}$, and P_n is the empirical distribution of S.

In traditional boosting, such conditions are typically referred to as the *edge* of a weak learner and play a crucial role in the convergence analysis. For example, Freund and Schapire [14] assume that for any set of weights over the training set S, there exists a classifier in the hypothesis class of weak classifiers which has better than random accuracy on the weighted samples. The following proposition shows that their condition is closely related to Definition 4.1.

Proposition 4.1. *For binary classification, the weak learning condition of Freund and Schapire [14] satisfies the empirical weak learning condition in Definition 4.1, albeit in the label space.*

For binary classification problems, it is well known that the weak learning condition of [14] is the weakest condition under which boosting is possible [15, 31]. This, together with the above proposition, suggests that our weak learning condition in Definition 4.1 cannot be weakened for binary classification problems.

To begin with, we derive excess risk bounds for the gradient greedy approach. Our analysis crucially relies on the observation that it can be viewed as performing inexact gradient descent on the population risk R. Several recent works have analyzed inexact gradient descent on convex objectives [2, 12, 33, 34]. However, the condition on the inexact gradient imposed by these works is different from ours and in many cases is stronger than our condition. For example, the condition of Balakrishnan et al. [2] translates to $\|g + \nabla_\phi R(W, \phi)\|_{P^X} \leqslant \epsilon$ in our setting, which is stronger than our weak learning condition. So the core of our analysis focuses on understanding inexact gradient descent with descent steps satisfying the weak learning condition in Definition 4.1. In our analysis, we consider a sample-splitting variant of the algorithm, where in each iteration we use a fresh batch of samples. This is mainly done to simplify the analysis by avoiding complex statistical dependencies between the iterates of the algorithm. Let $\tilde{n} = \lfloor \frac{n}{T} \rfloor$, we split the training dataset S into T subsets $\{S_t\}_{t=1}^T$ of size $\tilde{n}$, where $S_t = \{(\mathbf{x}_{t,i}, y_{t,i})\}_{i=1}^{\tilde{n}}$. We work with the subset S_t in the t^{th} iteration of Algorithm 1. We are now ready to state our main result on the excess risk bounds of the iterates of Algorithm 3. Our results depend on the Rademacher complexity terms related to the hypothesis sets $\mathcal{W}, \mathcal{G}_t$

$$\mathcal{R}(\mathcal{W}, \mathcal{G}_t) = \mathbb{E}\left[\sup_{\substack{W \in \mathcal{W}, \\ g \in \mathcal{G}_t}} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{k=1}^{K} \rho_{ik} [W g(\mathbf{x}_{t,i})]_k\right], \ \mathcal{R}(\mathcal{G}_t) = \mathbb{E}\left[\sup_{g \in \mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{j=1}^{D} \rho_{ij} [g(\mathbf{x}_{t,i})]_j\right],$$

where $[\mathbf{u}]_k$ denotes the k^{th} entry of a vector $\mathbf{u}$, and the expectation is taken w.r.t the randomness from S_t and the Rademacher random variables ρ_{ij}'s.

Theorem 4.1 (Gradient Greedy). *Suppose the classification loss ℓ is L-Lipschitz and M-smooth w.r.t the first argument. Let the hypothesis set of linear predictors $\mathcal{W}$ be s.t. any $W \in \mathcal{W}$ satisfies $\lambda_{min}\left(WW^T\right) \geqslant \sigma_{min}^2 > 0$ and $\lambda_{max}\left(WW^T\right) \leqslant \sigma_{max}^2$. Moreover, suppose for all t, $\mathcal{G}_t$ satisfies the (β, ϵ_t)-weak learning condition of Definition 4.1 for any dataset S_t. Finally, suppose any $g \in \mathcal{G}_t$ is bounded with $\sup_X \|g(X)\|_2 \leqslant B$. Let the learning rates $\{\eta_t\}_{t=1}^\infty$ be chosen as $\eta_t = ct^{-s}$, for some $s \in \left(\frac{\beta+1}{\beta+2}, 1\right)$ and positive constant c. If Algorithm 1 is run for T iterations with Algorithm 3 as* *update routine, then (W_T, ϕ_T), the T^{th} iterate output by the algorithm, satisfies the following risk bound for any W^*, ϕ^* and $\alpha \in (0, \beta(1-s))$, with probability at least $1-\delta$ over datasets of size n*

$$R(W_T, \phi_T) \leqslant R(W^*, \phi^*) + O\left(\frac{1}{T^\alpha} + T^{2-s}\sqrt{\frac{\log\frac{T}{\delta}}{\tilde{n}}}\right) + 2\sum_{t=1}^{T} \eta_t \left(L\mathcal{R}\left(\mathcal{W}, \mathcal{G}_t\right) + L\mathcal{R}\left(\mathcal{G}_t\right) + \epsilon_t\right).$$

Proof Sketch. We first show that Algorithm 3 can be viewed as performing inexact gradient descent on the population risk R. Specifically, we show that with high probability, the t^{th} iterate g_t satisfies

$$\langle g_t, -\nabla_\phi R(W_{t-1}, \phi_{t-1})\rangle_P \geqslant \beta B \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P - \epsilon_t - \zeta_t,$$

for some $\zeta_t > 0$. This follows from the weak learning condition satisfied by $\mathcal{G}_t$. Ignoring ϵ_t, ζ_t, the above equation shows that g_t makes acute angle with the population functional gradient at ϕ_{t-1}. Consequently, we would expect the population risk to decrease, if we move along g_t. This is indeed the case, and the final step in the proof formalizes this intuition. □

Remarks: We now briefly discuss the above result. See Appendix D for more discussion.

- The reference classifier (W^*, ϕ^*) in the above bound can be any classifier, as long as $\|W^*\|_2 < \infty, \|\phi^*\|_{P^X} < \infty$. In particular, if there exists a Bayes optimal classifier satisfying this condition, then the above Theorem provides an excess risk bound w.r.t the Bayes optimal classifier.
- The $T^{-\alpha}$ term in the bound corresponds to the *optimization error*. The $\eta_t\epsilon_t$ term corresponds to the *approximation error* and the rest of the terms correspond to the *generalization error*. As T increases, the optimization error goes down, and as $\tilde{n}$ increases, the generalization error goes down. If there is no approximation error, that is $\epsilon_t = 0$ for all t, then the excess risk goes down to 0 as $\tilde{n}, T \to \infty$ at appropriate rate.
- If $\beta = 1$, then for appropriate choice of step size the optimization error goes down as $O\left(T^{-1/3+\gamma}\right)$, for some arbitrarily small $\gamma > 0$. This rate is slower than the $O(T^{-1})$ rates for inexact gradient descent obtained by Devolder et al. [12], Schmidt et al. [33]. However, we note that unlike our work, these works assume that the level sets of the objective are bounded. Under the assumption that the level sets of population risk are bounded, the optimization error in Theorem 4.1 can be improved to $O(T^{-1})$. However, such a condition need not hold in the our setting.
- Note that the risk bounds are modular and only depend on the Rademacher complexity terms $\mathcal{R}(\mathcal{W}, \mathcal{G}_t), \mathcal{R}(\mathcal{G}_t)$ which capture the complexity of $\mathcal{G}_t$. To instantiate Theorem 4.1 for specific choices of $\mathcal{G}_t$, we need to bound these two complexity terms.

We now extend the analysis of Theorem 4.1 to the exact greedy approach.

Corollary 4.1 (Exact Greedy)**.** *Consider the setting of Theorem 4.1. Suppose Algorithm 1 is run with Algorithm 2 as update routine. Then (W_T, ϕ_T), the T^{th} iterate output by the algorithm, satisfies the same risk bounds as gradient greedy algorithm in Theorem 4.1.*

In the rest of the section, we instantiate Theorem 4.1 for specific choices of $\mathcal{G}_t$. We first consider the additive representation boosting algorithm.

Corollary 4.2. *Consider the setting of Theorem 4.1 and consider the additive representation boosting algorithm, where $\mathcal{G}_t = \mathcal{G}$ for all t. Suppose $\mathcal{G}$ is the set of one layer neural networks with sigmoid activation functions: $\mathcal{G} = \left\{\sigma(C\mathbf{x}), \text{ for } C \in \mathbb{R}^{D\times d}, \|C_{i,*}\|_1 \leqslant \Lambda, \forall i\right\}$. Moreover, suppose the feature domain $\mathcal{X}$ is a subset of $[0,1]^d$. Then the T^{th} iterate output by Algorithm 1, with Algorithm 2 or 3 as update routine, satisfies the following risk bound for any (W^*, ϕ^*), with probability at least $1-\delta$*

$$R(W_T, \phi_T) \leqslant R(W^*, \phi^*) + O\left(\frac{1}{T^\alpha}\right) + 2\sum_{t=1}^{T}\eta_t\epsilon_t + O\left(\frac{KD\Lambda T^{1-s}\log D}{\sqrt{\tilde{n}}} + T^{2-s}\sqrt{\frac{\log\frac{T}{\delta}}{\tilde{n}}}\right).$$

Next, we consider the layer-by-layer fitting technique of Bengio et al. [6].

Corollary 4.3. *Consider the setting of Corollary 4.2 and consider the layer-by-layer training technique of Bengio et al. [6], where $\mathcal{G}_t = \{g \circ \phi_{t-1} \text{ for } g \in \mathcal{G}\}$. Suppose $\mathcal{G}$ is the set of one layer neural networks with sigmoid activation functions: $\mathcal{G} = \left\{\sigma(C\mathbf{x}), \text{ for } C \in \mathbb{R}^{D\times D}, \|C_{i,*}\|_1 \leqslant \Lambda, \forall i\right\}$. Then the T^{th} iterate output by Algorithm 1, with Algorithm 2 or 3 as update routine, satisfies the following risk bound for any (W^*, ϕ^*) with probability at least $1-\delta$*

$$R(W_T, \phi_T) \leqslant R(W^*, \phi^*) + O\left(\frac{1}{T^\alpha}\right) + 2\sum_{t=1}^{T}\eta_t\epsilon_t + O\left(\frac{KD\Lambda T^{2-2s}\log D}{\sqrt{\tilde{n}}} + T^{2-s}\sqrt{\frac{\log\frac{T}{\delta}}{\tilde{n}}}\right).$$

Note that the generalization and optimization errors for both additive feature boosting and layer-by-layer fitting have similar dependence on $T, \tilde{n}$. However, the latter tends to have a smaller approximation error (ϵ_t) as it is able to build complex $\mathcal{G}_t$'s over time. So one would expect layer-by-layer fitting to output models with a better population risk, which our empirical results in fact verify.

5 Experiments

In this section, we present experiments comparing the performance of various boosting techniques on both simulated and benchmark datasets.

Baselines. We compare our proposed boosting techniques with XGBoost, AdaBoost, additive representation boosting (discussed in Corollary 4.2) and greedy layer-by-layer training technique of Bengio et al. [6] (Corollary 4.3). XGBoost uses decision trees as weak classifiers. For AdaBoost, we use 1 hidden layer neural networks as weak classifiers. We use two kinds of neural networks, based on the dataset. For tabular datasets, we use fully connected networks and for image datasets, we use convolutional networks (CNN) with the convolution block made up of *Convolution, BatchNorm, ReLU* layers arranged sequentially. For additive representation boosting (Additive Feature Boost from now on) and layer-by-layer fitting (StdCompBoost from now on), the weak feature transformer class $\mathcal{G}$ consists of one layer neural network transformations. Similar to AdaBoost, we use two kinds of transformations: a) fully connected transformations of the form $g(\mathbf{x}) = \mathrm{ReLU}(C\mathbf{x} + \mathbf{d})$, and b) convolutional transformations with *Convolution, BatchNorm, ReLU* blocks arranged sequentially. Finally, we also compare against end-to-end training of ResNets.

Proposed Techniques. For DenseCompBoost, we use a slight variant of $\mathcal{G}_t$ defined in Equation (3) : $\mathcal{G}_t = \{h + g \circ (\sum_{i=0}^{t-1} \alpha_i \phi_i), \text{ for } h \in \mathcal{H}, g \in \mathcal{G}, \alpha_i \in \mathbb{R}\}$, where $\mathcal{H}, \mathcal{G}$ are weak feature transformer classes. We use this variant because the dimensions of the input feature space and the representation space need not be the same, and as a consequence $\mathcal{G}_t$ in Equation (3) can not always be used. Similar to StdCompBoost, we consider two choices for $\mathcal{H}, \mathcal{G}$: one based on fully connected blocks and the other based on convolution blocks. For CmplxCompBoost, we again consider two choices for the weak transformer class $\tilde{\mathcal{G}}_t$ in Equation (4): a) $\mathrm{ReLU}(C\mathbf{x} + \mathbf{d})$ with $C \in \mathbb{R}^{D_t \times D_{t-1}}$, where $D_t = D_{t-1} + \Delta$ for some positive constant Δ, and b) convolution blocks with number of output channels equal to the number of input channels plus a constant Δ. This choice of feature transformers ensures the complexity of $\tilde{\mathcal{G}}_t$ increases with t. We use exact greedy updates (Algorithm 2) for both of our proposed methods and set learning rate η_t to 1. We do not present experimental results for Algorithm 3, which we noticed has marginally worse performance than Algorithm 2.

5.1 Simulated Datasets

Datasets. In this section we compare the techniques described above on simulated datasets. We generated 3 synthetic binary classification datasets in $\mathbb{R}^{32}$. Simulation 1 is a concentric ellipsoids dataset, where a point $\mathbf{x}$ is classified based on $\mathbf{x}^T A \mathbf{x}$, for some randomly generated positive semidefinite matrix A. Simulations 2, 3 are datasets whose classification boundaries are polynomials of degrees 8 and 9 respectively. For each of these datasets, we generated 10^6 samples for training and testing.

Hyper-parameters. We used hold-out set validation to pick the best hyper-parameters for all the methods. We used 20% of the training data as validation data and picked the best parameters using grid search, based on validation accuracy. After picking the best parameters, we train on the entire training data and report performance on the test data. For all the greedy techniques based on neural networks, we used fully connected blocks and tuned the following parameters: weight decay, width of weak feature transformers, number of boosting iterations T, which we upper bound by 15. For CmplxCompBoost, we set $\Delta = D_0/5$. For end-to-end training, we tuned weight decay, width of layers, depth. We used SGD for optimization of all these techniques. The number of epochs and step size schedule of SGD are chosen to ensure convergence. For XGBoost, we tuned the number of trees, depth of each tree, learning rate. The exact values of hyper-parameters tuned for each of the methods can be found in Appendix J.

Results. Table 1 presents the results from our experiments. Both CmplxCompBoost and StdCompBoost largely outperform the additive boosting methods, with CmplxCompBoost being slightly better due to the increasing complexity in $\tilde{G}_t$. Notably, DenseCompBoost performs significantly better than the rest and is able to bridge the gap between StdCompBoost and End-to-End. We attribute its success to its ability to recover from earlier mistakes: while StdCompBoost or CmplxCompBoost necessarily accumulate errors at each layer, DenseCompBoost is further connected to earlier layers, allowing it to undo its past mistakes.

Table 1: Test accuracy of various boosting techniques on synthetic datasets. Numbers in bold indicate the best performance among various greedy techniques.

Technique	Simulation 1	Simulation 2	Simulation 3
XGBoost (Trees)	84.40	97.59	50.10
AdaBoost (1 NN)	67.90	93.73	72.64
Additive Feature Boost	88.49	93.91	73.13
StdCompBoost	91.53	96.95	82.49
DenseCompBoost	**93.55**	**98.35**	**95.70**
CmplxCompBoost	91.97	97.22	82.52
End-to-End	93.88	98.35	99.09

Table 2: Test accuracy of various boosting techniques on benchmark datasets. We use convolution blocks for the first 5 datasets and fully connected blocks for the other datasets.

Technique	SVHN	FashionMNIST	CIFAR10	Convex	MNIST-rot-back-image	MNIST	Letter	CovType	Connect4
XGBoost (Trees)	77.72	90.34	58.34	82.29	53.89	97.96	96.16	**97.46**	86.63
AdaBoost (1 NN)	82.88	88	72.78	86.17	50.02	98.27	92.08	90.95	86.39
Additive Feature Boost	83.36	89.95	74.33	89.30	54.31	98.27	90.86	93.12	86.58
StdCompBoost	90.81	92.77	81.93	98.19	73.17	98.37	96.43	95.61	86.33
DenseCompBoost	91.03	**93.17**	82.31	**98.6**	73.1	98.34	**96.96**	96.28	**86.85**
CmplxCompBoost	**91.25**	**93.18**	**82.43**	98.52	**74.32**	98.34	96.66	95.92	86.49
End-to-End	94.82	93.49	86.88	98.81	82.69	98.95	97.67	96.86	87.37

5.2 Benchmark Datasets

Datasets. In this section, we compare various techniques on the following image datasets: CIFAR10, MNIST, FashionMNIST [35], MNIST-rot-back-image [24], convex [35], SVHN [28], and the following tabular datasets from UCI repository [7]: letter recognition [17], forest cover type (covtype), connect4. The convex dataset involves classifying shapes in images as either convex or non-convex.

Hyper-parameters. For covtype dataset, which doesn't come with a test set, we randomly sample 20% of the original data and use it as the test set. We use a similar hyper-parameter selection technique as above and tune the same set of hyper-parameters as described above. We use convolution blocks for CIFAR10, SVHN, FashionMNIST, convex, MNIST-rot-back-image and fully connected blocks for the rest. We limit the width of fully connected blocks to 4096, and the number of output channels in convolution blocks to 128 while tuning the hyper-parameters for the composition boosting techniques and end-to-end training. For AdaBoost and additive representation boosting, we set these limits to 16000 and 350 respectively. For CmplxCompBoost with convolution blocks, we set $\Delta = D_0/8$. We *do not* use data augmentation in our experiments.

Results. Table 2 presents the results from our experiments. It can be seen that on image classification tasks, additive boosting techniques have poor performance. Among compositional boosting methods, StdCompBoost performs the worst. While DenseCompBoost performs comparably to CmplxCompBoost on image datasets, it is better on tabular data. We believe a hybrid of DenseCompBoost and CmplxCompBoost algorithms can achieve better performance than either of the algorithms.

6 Conclusion

We proposed a generalized framework for boosting, which allows for more complex forms of aggregation of weak learners than traditional boosting. Our generalized framework allows to derive learning algorithms that (a) have performance close to that of end-to-end trained DNNs, and (b) come with strong theoretical guarantees. Additive boosting algorithms do not satisfy property (a), while DNNs do not satisfy property (b). In particular, additive boosting algorithms, even with small neural networks as their weak classifiers, do not have the strong performance of end-to-end trained DNNs. Improving their performance requires the hypothesis space to increase in complexity while not increasing sample complexity of each boosting step too greatly, which can be achieved by our generalized boosting framework. One particular instantiation of our framework is aggregation using function compositions. A number of existing greedy techniques for learning neural networks fall into our framework, and our analysis allowed us to delineate some of their key flaws, then consequently, propose new techniques which improve upon them. We believe our work opens up a new line of inquiry for greedy learning of highly flexible models with rigorous theoretical guarantees, by leveraging the theory of boosting and generalized greedy algorithms in function spaces. We moreover believe our work has the potential to bridge the gap in performance between existing greedy layer-by-layer training techniques and end-to-end training of deep networks.

Broader Impact

Deep learning has been tremendously successful over the past decade in many application areas such as computer vision, image recognition, speech recognition, and natural language processing. Despite this success, deep neural networks have largely remained a mystery. With millions of parameters, these models are blackboxes to humans, making it harder to diagnose errors. This also makes it harder to adopt these models in critical applications such as healthcare, law and finance. Consequently, it is crucial to come up with techniques that make neural networks transparent and easy to understand. We take a step towards this goal by drawing inspiration from classical boosting. Similar to classical boosting, our generalized boosting framework builds complex models greedily. But unlike classical boosting, it allows us to derive learning algorithms that have performance close to that of end-to-end trained DNNs. Moreover, models built using our framework are easy to understand and come with strong theoretical guarantees.

Acknowledgement

We acknowledge the support of NSF via IIS-1909816, OAC-1934584 and ONR via N000141812861.

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6155–6166, 2019.

[2] Sivaraman Balakrishnan, Martin J Wainwright, Bin Yu, et al. Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1):77–120, 2017.

[3] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.

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

[5] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. *arXiv preprint arXiv:1812.11446*, 2018.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[7] Catherine L Blake and Christopher J Merz. Uci repository of machine learning databases, 1998, 1998.

[8] Chang Chen, Zhiwei Xiong, Xinmei Tian, and Feng Wu. Deep boosting for image denoising. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[9] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[10] Corinna Cortes, Mehryar Mohri, and Umar Syed. Deep boosting. volume 32 of *Proceedings of Machine Learning Research*, pages 1179–1187, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/cortesb14.html.

[11] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 874–883. JMLR. org, 2017.

[12] Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

[13] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

[14] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[15] Yoav Freund and Robert E Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the ninth annual conference on Computational learning theory*, pages 325–332, 1996.

[16] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[17] Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.

[18] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[19] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[20] Alexander Grubb and J Andrew Bagnell. Generalized boosting algorithms for convex optimization. *arXiv preprint arXiv:1105.2054*, 2011.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Furong Huang, Jordan Ash, John Langford, and Robert Schapire. Learning deep resnet blocks sequentially using boosting theory. *arXiv preprint arXiv:1706.04964*, 2017.

[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[24] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480, 2007.

[25] Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *Advances in Neural Information Processing Systems*, pages 3033–3045, 2019.

[26] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000.

[27] Andreas Maurer. A vector-contraction inequality for rademacher complexities. In *International Conference on Algorithmic Learning Theory*, pages 3–17. Springer, 2016.

[28] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[29] Atsushi Nitanda and Taiji Suzuki. Functional gradient boosting based on residual network perception. *arXiv preprint arXiv:1802.09031*, 2018.

[30] Natalia Ponomareva, Thomas Colthurst, Gilbert Hendry, Salem Haykal, and Soroush Radpour. Compact multi-class boosted trees. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 47–56. IEEE, 2017.

[31] Gunnar Rätsch and Manfred K Warmuth. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6(Dec):2131–2152, 2005.

[32] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[33] Mark Schmidt, Nicolas L Roux, and Francis R Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems*, pages 1458–1466, 2011.

[34] Vladimir Nikolaevich Temlyakov. Greedy expansions in convex optimization. *Proceedings of the Steklov Institute of Mathematics*, 284(1):244–262, 2014.

[35] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

A Notation and Terminology

Notation

Symbol	Description
X	feature vector
Y	label
$\mathcal{X}$	domain of feature vector
$\mathcal{Y}$	domain of the label
K	number of classes in multi-class classification problem
S	data set
P	true data distribution
P^X, P^Y	marginal distributions of X, Y
P_n	empirical distribution
P_n^X, P_n^Y	empirical marginal distributions of X, Y in data set S
$f : \mathcal{X} \to \mathbb{R}^K$	score based classifier
ϕ	feature transformer
W	linear classifier on top of feature transformer
ℓ_{0-1}	0/1 classification loss
ℓ	convex surrogate of ℓ_{0-1}
$R(f)$	population risk of classifier f, measured w.r.t ℓ
$\widehat{R}_S(f)$	empirical risk of classifier f, measured w.r.t ℓ
$R(W, \phi)$	population risk of classifier $f = W\phi$, measured w.r.t ℓ
$\widehat{R}_S(W, \phi)$	empirical risk of classifier $f = W\phi$, measured w.r.t ℓ
$L_2(P)$	set of square integrable functions w.r.t P
$f \circ g(\mathbf{x})$	denotes function composition $f(g(\mathbf{x}))$
$[\phi_0, \ldots, \phi_t](\mathbf{x})$	denotes concatenation of vectors $\phi_0(\mathbf{x}) \ldots \phi_t(\mathbf{x})$
$\mathcal{F}$	hypothesis class of weak classifiers
$\mathcal{G}$	hypothesis class of weak feature transformers
$\mathcal{G}_t$	hypothesis class of weak feature transformers used in the t^{th} iteration of greedy
$\mathcal{W}$	hypothesis class of linear classifiers on top of feature transformers

Terminology

Term	Description
Additive Boosting	Classical boosting framework which constructs a strong classifier using additive combinations of weak classifiers
Additive Feature Boosting	Feature boosting framework which constructs a strong classifier using additive combinations of weak feature transformers with a linear classifier on top of the feature transformer
Weak classifier	Any classifier which by itself doesn't achieve good performance on a given classification task and whose performance we wish to boost
Weak feature transformer	Any feature transformation which by itself doesn't provide good performance on a given classification task and whose performance we wish to boost

B Proof of Proposition 3.1

Notation. We use the notation of Huang et al. [22] in this proof. We note that this notation will only be used in this section. Later sections use the notation introduced in Section 2. We let $g_t(\mathbf{x})$ be the output of the t^{th} residual block, which is given by the following recursion

$$g_t(\mathbf{x}) = f_{t-1} \circ g_{t-1}(\mathbf{x}) + g_{t-1}(\mathbf{x}) = \sum_{i=0}^{t-1} f_i \circ g_i(\mathbf{x}),$$

with g_0, f_0 equal to identity functions. The final output of a depth-T ResNet, given input $\mathbf{x}$, is rendered after a linear classifier $W \in \mathbb{R}^{K \times D}$ on representation $g_{T+1}(\mathbf{x})$. Let W_t be the auxiliary linear classifier on top of the residual block g_t. Define $o_t(\mathbf{x})$ as

$$o_t(\mathbf{x}) \overset{def}{=} W_t g_t(\mathbf{x}).$$

Note that $o_t(\mathbf{x}) = \sum_{i=0}^{t} W_t f_i \circ g_i(\mathbf{x})$. Define $h_t(\mathbf{x})$ as $h_t(\mathbf{x}) \overset{def}{=} \alpha_{t+1} o_{t+1}(\mathbf{x}) - \alpha_t o_t(\mathbf{x})$, where α_t is a scalar. Huang et al. [22] consider exponential loss in their work, which is defined as

$$\ell(o(\mathbf{x}), y) = \sum_{k \neq y} \exp\left([o(\mathbf{x})]_k - [o(\mathbf{x})]_y\right).$$

Algorithm of Bengio et al. [6]. Using this notation, the greedy layer-by-layer training technique of Bengio et al. [6] for learning ResNets is given by the following update rule

$$W_{t+1}, f_t \leftarrow \underset{W,f}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(W\left[f \circ g_t(\mathbf{x}_i) + g_t(\mathbf{x}_i)\right], y_i\right). \tag{5}$$

Algorithm of Huang et al. [22]. The algorithm of Huang et al. [22] for greedy learning of ResNets is given in Algorithm 4, which is a reproduction of Algorithm 3 of Huang et al. [22]. Note that the key update step is given in step 2 of Algorithm 5

$$f_t, \alpha_{t+1}, W_{t+1} \leftarrow \underset{f,\alpha,W}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(\alpha W[f \circ g_t(\mathbf{x}_i) + g_t(\mathbf{x}_i)], y_i). \tag{6}$$

Since α is a scalar, it can be consumed into the linear classifier W. This shows that the update step of Huang et al. [22] is equivalent to Equation (5).

Algorithm 4 Greedy algorithm of Huang et al. [22] for learning ResNets

```
1: Input: Training data $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, iterations $T$, threshold $\gamma$
2: Initialize $t \leftarrow 0, \tilde{\gamma}_0 \leftarrow 0, \alpha_0 \leftarrow 0, o_0 \leftarrow \mathbf{0} \in \mathbb{R}^K, s_0(\mathbf{x}_i) = \mathbf{0} \in \mathbb{R}^K, \forall i \in [n]$
3: Initialize cost function $[C_0(i)]_k \leftarrow \begin{cases} 1 & \text{if } k \neq y_i \\ 1-K & \text{if } k = y_i \end{cases}, \forall i \in [n], k \in [K]$
4: while $\gamma_t > \gamma$ do
5:     $f_t, \alpha_{t+1}, W_{t+1}, o_{t+1} \leftarrow$ Algorithm 5$(g_t)$
6:     Compute $\gamma_t \leftarrow \sqrt{\frac{\tilde{\gamma}_{t+1}^2 - \tilde{\gamma}_t^2}{1-\tilde{\gamma}_t^2}}$, where $\tilde{\gamma}_{t+1} = \frac{-\sum_{i=1}^n C_t(i)^T o_{t+1}(\mathbf{x}_i)}{\sum_{i=1}^n \sum_{k \neq y_i} [C_t(i)]_k}$
7:     Update $s_{t+1}(\mathbf{x}_i) \leftarrow s_t(\mathbf{x}_i) + h_t(\mathbf{x}_i)$, where $h_t(\mathbf{x}_i) = \alpha_{t+1} o_{t+1}(\mathbf{x}_i) - \alpha_t o_t(\mathbf{x}_i)$
8:     Update cost function $[C_{t+1}(i)]_k \leftarrow \begin{cases} \exp\left([s_{t+1}(\mathbf{x}_i)]_k - [s_{t+1}(\mathbf{x}_i)]_{y_i}\right) & \text{if } k \neq y_i \\ -\sum_{k' \neq y_i} \exp\left([s_{t+1}(\mathbf{x}_i)]_{k'} - [s_{t+1}(\mathbf{x}_i)]_{y_i}\right) & \text{if } k = y_i \end{cases}, \forall i \in [n], k \in [K]$
9:     $t \leftarrow t+1$
10: end while
11: $T \leftarrow t-1$
12: Return: $W_{T+1}, \{f_t(\cdot), \forall t\}$
```

Algorithm 5 Training a ResNet module

```
1: Input: $g_t$
2: $(f_t, \alpha_{t+1}, W_{t+1}) \leftarrow \operatorname{argmin}_{f,\alpha,W} \sum_{i=1}^n \ell(\alpha W[f \circ g_t(\mathbf{x}_i) + g_t(\mathbf{x}_i)], y_i)$
3: $o_{t+1}(\mathbf{x}) = W_{t+1}[f_t \circ g_t(\mathbf{x}) + g_t(\mathbf{x})]$
4: Return: $f_t, \alpha_{t+1}, W_{t+1}, o_{t+1}$
```

C Proof of Proposition 4.1

Freund and Schapire [14] consider the problem of binary classification with $\mathcal{Y} = \{-1, +1\}$. Let $\mathcal{F}$ be a hypothesis space of weak classifiers mapping $\mathcal{X}$ to $\mathcal{Y}$. Freund and Schapire [14] consider the following weak learning condition. For any set of non-negative weights $\{w_i\}_{i=1}^n$ over points

$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ such that $\sum_i w_i = 1$, there is a classifier $f \in \mathcal{F}$ which achieves an error at most $\frac{1}{2} - \frac{\beta}{2}$, for some $\beta > 0$. That is, there exists $f \in \mathcal{F}$ such that

$$\sum_{i=1}^n w_i \mathbb{I}(y_i \neq f(\mathbf{x}_i)) \leqslant \frac{1}{2} - \frac{\beta}{2}.$$

This can equivalently be written as

$$\begin{aligned} \textstyle\sum_{i=1}^n w_i y_i f(\mathbf{x}_i) &= \textstyle\sum_{i:y_i=f(\mathbf{x}_i)} w_i y_i f(\mathbf{x}_i) - \sum_{i:y_i\neq f(\mathbf{x}_i)} w_i y_i f(\mathbf{x}_i) + 2\sum_{i:y_i\neq f(\mathbf{x}_i)} w_i y_i f(\mathbf{x}_i) \\ &= 1 + 2\textstyle\sum_{i:y_i\neq f(\mathbf{x}_i)} w_i y_i f(\mathbf{x}_i) \\ &\geqslant \beta \\ &= \beta \left(\textstyle\sum_{i=1}^n w_i\right) \end{aligned} \tag{7}$$

We now show that this condition implies Definition 4.1 in the label space. We first introduce the notion of inner product between functions mapping $\mathcal{X}$ to $\mathbb{R}$. For any f, g mapping $\mathcal{X}$ to $\mathbb{R}$, we define $\langle f, g\rangle_n$ as

$$\langle f, g\rangle_n = \frac{1}{n}\sum_{i=1}^n f(\mathbf{x}_i) g(\mathbf{x}_i).$$

Let the classification loss ℓ be such that $\ell(f(\mathbf{x}), y) = c(yf(\mathbf{x}))$ for some decreasing function $c : \mathbb{R} \to \mathbb{R}$. All the popular classification losses such as logistic, exponential, hinge losses satisfy this assumption. The functional gradient of $\widehat{R}_S$ w.r.t f in the above inner product space is defined as

$$\nabla_f \widehat{R}_S(f)(\mathbf{x}) = \begin{cases} y_i c'(y_i f(\mathbf{x}_i)), & \text{if } \mathbf{x} = \mathbf{x}_i \\ 0, & \text{otherwise} \end{cases},$$

where $c'(z)$ is the derivative of c at z. Note that since c is a decreasing function, $c'(z) < 0$ for any z. Using this notation, it is easy to see that any hypothesis class $\mathcal{F}$ satisfying Equation (7) satisfies the following condition for any function $h : \mathcal{X} \to \mathbb{R}$

$$\exists f \in \mathcal{F}, \quad \left\langle f, -\nabla_f \widehat{R}_S(h)\right\rangle_n \geqslant \beta \|\nabla_f \widehat{R}_S(h)\|_1 \geqslant \frac{\beta}{\sqrt{n}} \|\nabla_f \widehat{R}_S(h)\|_n,$$

where $\|\nabla_f \widehat{R}_S(h)\|_1 = n^{-1}\sum_{i=1}^n |\nabla_f \widehat{R}_S(h)(\mathbf{x}_i)|$. This can be shown by substituting w_i in Equation (7) with $-c'(y_i h(\mathbf{x}_i))$. This shows that the weak learning condition of Freund and Schapire [14] satisfies the weak learning condition in Definition 4.1, albeit in the label space.

D Discussion of Theorem 4.1

In this section, we discuss the results of Theorem 4.1.

Remark D.1 (Reference Classifier)**.** *The reference classifier (W^*, ϕ^*) in the bound in Theorem 4.1 can be any classifier, as long as $\|W^*\|_2 < \infty, \|\phi^*\|_{P^X} < \infty$. In particular, if there exists a Bayes optimal classifier satisfying this condition, then the above Theorem provides an excess risk bound w.r.t the Bayes optimal classifier.*

Remark D.2 (Breakdown of Rates)**.** *The $T^{-\alpha}$ term in the bound corresponds to the* optimization error. *The $\eta_t\epsilon_t$ term corresponds to the* approximation error *and the rest of the terms correspond to the* generalization error. *As T increases, the optimization error goes down, and as $\tilde{n}$ increases, the generalization error goes down. If there is no approximation error, that is $\epsilon_t = 0$ for all t, then the excess risk goes down to 0 as $\tilde{n}, T \to \infty$ at appropriate rate.*

Remark D.3 (Optimization Error)**.** *If $\beta = 1$, then for appropriate choice of step size the optimization error goes down as $O\left(T^{-1/3+\gamma}\right)$, for some arbitrarily small $\gamma > 0$. This rate is slower than the $O(T^{-1})$ rates for inexact gradient descent obtained by Devolder et al. [12], Schmidt et al. [33]. However, we note that unlike our work, these works assume that the level sets of the objective are bounded. Under the assumption that the level sets of population risk are bounded, the optimization error in Theorem 4.1 can be improved to $O(T^{-1})$. However, such a condition need not hold in the our setting.*

Remark D.4 (Lipschitzness of loss)**.** *The assumptions of smoothness and Lipschitzness on ℓ are satisfied by popular loss functions such as logistic loss,* softmax + cross entropy loss. *Consider logistic loss for binary classification $\ell(z, y) = \log(1 + e^{-yz})$. It is easy to verify that $\ell(z, y)$ is* 1*-Lipschitz and* 1*-smooth w.r.t. z. Similarly, the softmax + cross entropy loss, which is given by, $\ell(\mathbf{z}, y) = -\mathbf{z}[y] + \log\left(\sum_{k=1}^{K} e^{\mathbf{z}[k]}\right)$ is 1-Lipschitz and 1-smooth w.r.t.* $\mathbf{z}$.

Remark D.5 (Bounded Feature Transformers)**.** *The boundedness assumption on the functions in $\mathcal{G}_t$ is satisfied by neural networks made up of bounded activation functions such as* sigmoid, tanh.

Remark D.6 (Modular Bounds)**.** *Note that the risk bounds are modular and only depend on the Rademacher complexity terms $\mathcal{R}(\mathcal{W}, \mathcal{G}_t), \mathcal{R}(\mathcal{G}_t)$ which capture the complexity of $\mathcal{G}_t$. To instantiate Theorem 4.1 for specific choices of $\mathcal{G}_t$, we need to bound these two complexity terms.*

Remark D.7 (Bounds on $0/1$ risk)**.** *Since $0/1$ loss is upper bounded by surrogate losses such as exponential, logistic loss, our Theorem also provides generalization bounds for $0/1$ loss.*

Remark D.8 (Sample Splitting)**.** *A natural question that might arise regarding sample splitting is: "does this make our approach similar to bagging and random forests (RFs)?". We would like to note that even with sample splitting, our approach is not similar to bagging and RFs. Bagging and RFs create ensembles by independently training each base learner. Whereas, in boosting, the base learners are fit greedily and are not independent of each other. Another important distinction between RFs and boosting is that RFs work with complex base classifiers with good predictive power and aim to reduce the variance of these classifiers by averaging the predictions of multiple independently trained base classifiers. Whereas in boosting, one works with base classifiers with very little predictive power (i.e., high bias) and combines multiple such base classifiers to create a strong classifier with good predictive power (i.e., low bias). Viewed this way, our approach is very similar to boosting than RFs.*

E Proof of Theorem 4.1

E.1 Intermediate Results

In this section we present some intermediate results which we use in the proof of Theorem 4.1. The proof of the Theorem can be found in Section E.2.

Lemma E.1. *Consider the setting of Theorem 4.1. Let (W_t, ϕ_t) be the t^{th} iterate generated by Algorithm 1 with Algorithm 3 as update routine. Then for any t, the following holds with probability at least $1 - \delta$ over datasets of size n*

$$R(W_t, \phi_t) \leqslant R(W_{t-1}, \phi_t) + 2\eta_t L \mathcal{R}(\mathcal{W}, \mathcal{G}_t) + \frac{4c\sigma_{max} B L t^{1-s}}{1-s}\left(\sqrt{\frac{\log 2/\delta}{\tilde{n}}} + \sqrt{\frac{K}{\tilde{n}}}\right),$$

where $\mathcal{R}(\mathcal{W}, \mathcal{G}_t)$ is the Rademacher complexity term, which is defined as

$$\mathcal{R}(\mathcal{W}, \mathcal{G}_t) = \mathbb{E}\left[\sup_{\substack{W\in\mathcal{W},\\ g\in\mathcal{G}_t}} \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\sum_{k=1}^{K} \rho_{ik}[Wg(\mathbf{x}_{t,i})]_k\right],$$

and the expectation is over the randomness from S_t, ρ's.

Proof. Throughout the proof, we condition on the past datasets $S_1, \ldots S_{t-1}$ and show that the Lemma holds for any choice of $S_1, \ldots S_{t-1}$. Consider the following upper bound for $R(W_t, \phi_t)$

$$\begin{aligned} R(W_t, \phi_t) &\leqslant \widehat{R}_{S_t}(W_t, \phi_t) + \sup_{W\in\mathcal{W}, g\in\mathcal{G}_t} |R(W, \phi_{t-1} + \eta_t g) - \widehat{R}_{S_t}(W, \phi_{t-1} + \eta_t g)| \\ &\overset{(a)}{\leqslant} \widehat{R}_{S_t}(W_{t-1}, \phi_t) + \sup_{W\in\mathcal{W}, g\in\mathcal{G}_t} |R(W, \phi_{t-1} + \eta_t g) - \widehat{R}_{S_t}(W, \phi_{t-1} + \eta_t g)| \\ &\leqslant R(W_{t-1}, \phi_t) + 2 \sup_{W\in\mathcal{W}, g\in\mathcal{G}_t} |R(W, \phi_{t-1} + \eta_t g) - \widehat{R}_{S_t}(W, \phi_{t-1} + \eta_t g)|, \end{aligned}$$

where (a) follows from the definition of W_t. We now rely on Rademacher complexity bounds in Theorem I.2 to bound the supremum in the RHS. To apply the bound, we first need to ensure

$\ell(W\phi_{t-1}(\mathbf{x}) + \eta_t Wg(\mathbf{x}), y)$ is bounded. Since $\sup_X \|g(X)\|_2 \leqslant B$ and $\lambda_{\max}\left(WW^T\right) \leqslant \sigma_{\max}^2$, it is easy to see that

$$\sup_X \|W\phi_{t-1}(\mathbf{x}) + \eta_t Wg(\mathbf{x})\|_2 \leqslant \sigma_{\max} B \sum_{i=1}^{t} \eta_i \leqslant \frac{c\sigma_{\max} B t^{1-s}}{1-s},$$

where the last inequality follows from the definition of η_t. Since ℓ is L-Lipschitz in its first argument, we can show that $\ell(W\phi_{t-1}(\mathbf{x}) + \eta_t Wg(\mathbf{x}), y)$ lies in an interval of width $\frac{2c\sigma_{\max} BLt^{1-s}}{1-s}$. Applying Theorem I.2, we get with probability at least $1-\delta$

$$\begin{aligned}
R(W_t, \phi_t) \leqslant& R(W_{t-1}, \phi_t) + 2\mathbb{E}\left[\sup_{W\in\mathcal{W}, g\in\mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_i \ell(W\phi_{t-1}(\mathbf{x}_{t,i}) + \eta_t Wg(\mathbf{x}_{t,i}), y_{t,i})\right] \\
&+ \frac{4c\sigma_{\max} BLt^{1-s}}{1-s}\sqrt{\frac{\log 2/\delta}{\tilde{n}}}.
\end{aligned}$$

We now focus on bounding the Rademacher complexity term appearing above. To this end, we rely on the composition property of Rademacher complexity. Since ℓ is L-Lipscthiz in the first argument, applying Theorem I.3 we get

$$\begin{aligned}
R(W_t, \phi_t) \leqslant& R(W_{t-1}, \phi_t) + 2L\mathbb{E}\left[\sup_{W\in\mathcal{W}, g\in\mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{k=1}^{K} \rho_{ik} [W\phi_{t-1}(\mathbf{x}_{t,i}) + \eta_t Wg(\mathbf{x}_{t,i})]_k\right] \\
&+ \frac{4c\sigma_{\max} BLt^{1-s}}{1-s}\sqrt{\frac{\log 2/\delta}{\tilde{n}}} \\
\leqslant& R(W_{t-1}, \phi_t) + 2\eta_t L\mathcal{R}(\mathcal{W}, \mathcal{G}_t) + 2L\underbrace{\mathbb{E}\left[\sup_{W\in\mathcal{W}} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{k=1}^{K} \rho_{ik} [W\phi_{t-1}(\mathbf{x}_{t,i})]_k\right]}_{T_1} \\
&+ \frac{4c\sigma_{\max} BLt^{1-s}}{1-s}\sqrt{\frac{\log 2/\delta}{\tilde{n}}}
\end{aligned}$$

T_1 can be bounded as follows. Let $\rho \in \mathbb{R}^{K\times\tilde{n}}$ be the matrix whose $(k,i)^{th}$ entry is given by ρ_{ik} and $\phi_{t-1}(S_t) \in \mathbb{R}^{D\times\tilde{n}}$ be the matrix whose $(j,i)^{th}$ entry is given by $[\phi_{t-1}(\mathbf{x}_{t,i})]_j$. T_1 can be rewritten in terms of $\rho, \phi_{t-1}(S_t)$ as

$$\begin{aligned}
T_1 &= \mathbb{E}\left[\sup_{W\in\mathcal{W}} \frac{1}{\tilde{n}} \left\langle \rho\phi_{t-1}(S_t)^T, W\right\rangle_F\right] \\
&\leqslant \left[\sup_{W\in\mathcal{W}} \|W\|_2\right] \mathbb{E}\left[\frac{1}{\tilde{n}} \|\rho\phi_{t-1}(S_t)^T\|_F\right] \\
&\leqslant \sigma_{\max} \mathbb{E}\left[\frac{1}{\tilde{n}} \|\rho\phi_{t-1}(S_t)^T\|_F\right] \\
&\leqslant \frac{\sigma_{\max}}{\tilde{n}} \sqrt{\mathbb{E}\left[\|\rho\phi_{t-1}(S_t)^T\|_F^2\right]} \\
&= \frac{\sigma_{\max}}{\tilde{n}} \sqrt{K\mathbb{E}\left[\|\phi_{t-1}(S_t)\|_F^2\right]} \leqslant \sigma_{\max} \sqrt{\frac{K}{\tilde{n}}} \mathbb{E}\left[\sup_X \|\phi_{t-1}(X)\|_2\right] \\
&\leqslant \frac{c\sigma_{\max} Bt^{1-s}}{1-s} \sqrt{\frac{K}{\tilde{n}}},
\end{aligned}$$

where the last inequality follows from our choice of step size η_t and our assumption on the boundedness of the outputs of functions in $\mathcal{G}_t$. Substituting this upper bound on T_1 in the previous inequality gives us the required bound on $R(W_t, \phi_t)$. $\square$

Lemma E.2. *Consider the setting of Theorem 4.1. Let (W_t, ϕ_t) be the t^{th} iterate generated by Algorithm 1 with Algorithm 3 as update routine. Then for any t, the following holds with probability at least $1-2\delta$ over datasets of size n*

$$\langle g_t, -\nabla_\phi R(W_{t-1}, \phi_{t-1})\rangle_P \geqslant \beta B \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P - \epsilon_t - 2\sigma_{max} L\mathcal{R}(\mathcal{G}_t) - 4\sigma_{max} BL\sqrt{\frac{\log 2/\delta}{\tilde{n}}}.$$

Proof. Let $\hat{P}_{\tilde{n},t}$ be the empirical distribution of dataset S_t. Since $\mathcal{G}_t$ satisfies the (β, ϵ_t)-weak learning condition w.r.t dataset S_t, we have

$$\left\langle g_t, -\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1}) \right\rangle_{P^X_{\tilde{n},t}} \geqslant \beta B \|\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1})\|_{P^X_{\tilde{n},t}} - \epsilon_t.$$

Consider the following lower bound for $\langle g_t, -\nabla_\phi R(W_{t-1}, \phi_{t-1})\rangle_P$

$$\begin{aligned}\langle g_t, -\nabla_\phi R(W_{t-1}, \phi_{t-1})\rangle_P \geqslant & \underbrace{\left\langle g_t, -\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1}) \right\rangle_{P^X_{\tilde{n},t}}}_{T_1} \\ & - \underbrace{\left| \left\langle g_t, -\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1}) \right\rangle_{P^X_{\tilde{n},t}} - \langle g_t, -\nabla_\phi R(W_{t-1}, \phi_{t-1})\rangle_P \right|}_{T_2}\end{aligned}$$

We now lower bound each of the terms appearing the RHS of the above inequality. Similar to the proof of Lemma E.1, throughout the proof we condition on the past datasets $S_1, \dots S_{t-1}$ and show that the Lemma holds for any choice of $S_1, \dots S_{t-1}$.

Bounding T_1. Using the weak learning condition, T_1 can be lower bounded as

$$T_1 \geqslant \beta B \|\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1})\|_{P^X_{\tilde{n},t}} - \epsilon_t.$$

Using triangle inequality, this can be further lower bounded as

$$T_1 \geqslant \beta B \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P - \beta B \left| \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P - \|\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1})\|_{P^X_{\tilde{n},t}} \right| - \epsilon_t.$$

We now bound the middle term in the RHS using standard concentration inequalities. Define random variable Z as

$$Z = W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(X), Y),$$

for $(X, Y) \sim P$ and define $\mathbf{z}_{t,i}$ as

$$\mathbf{z}_{t,i} = W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(\mathbf{x}_{t,i}), y_{t,i}),$$

where $\nabla \ell(u, y)$ denotes the gradient of ℓ w.r.t its first argument. Then from the definition of functional gradients $\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1}), \nabla_\phi R(W_{t-1}, \phi_{t-1})$, we have

$$\|\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1})\|^2_{P^X_{\tilde{n},t}} = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \|\mathbf{z}_{t,i}\|^2, \quad \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|^2_P = \mathbb{E}\left[\|Z\|^2\right].$$

Since ℓ is L-Lipschitz, it is easy to see that $\|Z\|$ is a bounded random variable and always lies in the interval $[0, \sigma_{\max} L]$. So using Chernoff bounds in Theorem I.1, we can show that the following holds with probability at least $1 - \delta$

$$\left| \sum_{i=1}^{\tilde{n}} \frac{1}{\tilde{n}} \|\mathbf{z}_{t,i}\|^2 - \mathbb{E}\left[\|Z\|^2\right] \right| \leqslant \sigma_{\max} L \sqrt{\frac{3\mathbb{E}\left[\|Z\|^2\right] \log 1/\delta}{\tilde{n}}}.$$

Now, consider the following

$$\begin{aligned}\left| \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P - \|\nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1})\|_{P^X_{\tilde{n},t}} \right| &= \left| \sqrt{\sum_{i=1}^{\tilde{n}} \frac{1}{\tilde{n}} \|\mathbf{z}_{t,i}\|^2} - \sqrt{\mathbb{E}\left[\|Z\|^2\right]} \right| \\ &\leqslant \frac{\left| \sum_{i=1}^{\tilde{n}} \frac{1}{\tilde{n}} \|\mathbf{z}_{t,i}\|^2 - \mathbb{E}\left[\|Z\|^2\right] \right|}{\sqrt{\mathbb{E}\left[\|Z\|^2\right]}},\end{aligned}$$

where the last inequality follows from the fact that $|\sqrt{a} - \sqrt{b}| = \frac{|a-b|}{\sqrt{a}+\sqrt{b}} \leqslant \frac{|a-b|}{\sqrt{b}}$. This shows that, with probability at least $1 - \delta$, T_1 can be lower bounded as

$$T_1 \geqslant \beta B \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P - \beta \sigma_{\max} B L \sqrt{\frac{3 \log 1/\delta}{\tilde{n}}} - \epsilon_t. \tag{8}$$

Bounding T_2. Using the definition of functional gradients, T_2 can be rewritten as follows

$$\begin{aligned}
T_2 &= \left| \mathbb{E}_X \left[\langle g_t(X), \nabla_\phi R(W_{t-1}, \phi_{t-1})(X) \rangle \right] - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \left\langle g_t(\mathbf{x}_{t,i}), \nabla_\phi \widehat{R}_{S_t}(W_{t-1}, \phi_{t-1})(\mathbf{x}_{t,i}) \right\rangle \right| \\
&= \left| \mathbb{E}_{X,Y} \left[\langle g_t(X), W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(X), Y) \rangle \right] - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \langle g_t(\mathbf{x}_{t,i}), W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(\mathbf{x}_{t,i}), y_{t,i}) \rangle \right| \\
&\leqslant \sup_{g \in \mathcal{G}_t} \left| \mathbb{E}_{X,Y} \left[\langle g(X), W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(X), Y) \rangle \right] - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \langle g(\mathbf{x}_{t,i}), W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(\mathbf{x}_{t,i}), y_{t,i}) \rangle \right|.
\end{aligned}$$

We now rely on uniform convergence bounds and bound the RHS in terms of the Rademacher complexity term $\mathcal{R}(\mathcal{G}_t)$. First note that the random variable $\langle g(X), W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(X), Y) \rangle$ is bounded and lies in the interval $[-\sigma_{\max} BL, \sigma_{\max} BL]$. This follows from the Lipschitz property of the loss ℓ and the boundedness of the functions in $\mathcal{G}_t$. Using Theorem I.2, we get the following upper bound for T_2, which holds with probability at least $1-\delta$

$$T_2 \leqslant 2\mathbb{E}\left[\sup_{g \in \mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_i \langle g(\mathbf{x}_{t,i}), W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(\mathbf{x}_{t,i}), y_{t,i}) \rangle \right] + 2\sigma_{\max} BL \sqrt{\frac{\log 2/\delta}{\tilde{n}}}.$$

We now focus on bounding the Rademacher complexity term in the above inequality. Define function $h_i : \mathbb{R}^D \to \mathbb{R}$ as follows

$$h_i(\mathbf{u}) = \langle \mathbf{u}, W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(\mathbf{x}_{t,i}), y_{t,i}) \rangle.$$

Note that, $h_i(\mathbf{u})$ is $\sigma_{\max} L$-Lipschitz in $\mathbf{u}$. The Rademacher complexity can be written in terms of h_i's as follows

$$\mathbb{E}\left[\sup_{g \in \mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_i \langle g(\mathbf{x}_{t,i}), W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(\mathbf{x}_{t,i}), y_{t,i}) \rangle \right] = \mathbb{E}_{S_t}\left[\mathbb{E}_\rho \left[\sup_{g \in \mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_i h_i(g(\mathbf{x}_{t,i})) \Big| S_t \right] \right].$$

Using the composition property of Rademacher complexities stated in Theorem I.3, we get

$$\begin{aligned}
\mathbb{E}\left[\sup_{g \in \mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_i \langle g(\mathbf{x}_{t,i}), W_{t-1}^T \nabla \ell(W_{t-1}\phi_{t-1}(\mathbf{x}_{t,i}), y_{t,i}) \rangle \right] &\leqslant \sigma_{\max} L \mathbb{E}_{S_t}\left[\mathbb{E}_\rho \left[\sup_{g \in \mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{j=1}^{D} \rho_{ij} [g(\mathbf{x}_{t,i})]_j \Big| S_t \right] \right] \\
&= \sigma_{\max} L \mathbb{E}\left[\sup_{g \in \mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{j=1}^{D} \rho_{ij} [g(\mathbf{x}_{t,i})]_j \right] \\
&= \sigma_{\max} L \mathcal{R}(\mathcal{G}_t).
\end{aligned}$$

So we have the following bound for T_2 which holds with probability at least $1-\delta$

$$T_2 \leqslant 2\sigma_{\max} L \mathcal{R}(\mathcal{G}_t) + 2\sigma_{\max} BL \sqrt{\frac{\log 2/\delta}{\tilde{n}}}. \tag{9}$$

Combining Equations (8), (9) gives us the required bound. □

E.2 Main Argument

Our analysis of inexact gradient descent uses similar arguments as in Temlyakov [34]. Let $\phi_t = \phi_{t-1} + \eta_t g_t$ be the t^{th} iterate generated by the algorithm. We first derive an upper bound for the reduction in population risk in the t^{th} iteration of the algorithm. From Lemma E.1 we know that with probability at least $1 - \delta/3T$

$$R(W_t, \phi_t) \leqslant R(W_{t-1}, \phi_t) + C_1(t), \tag{10}$$

where $C_1(t) = 2\eta_t L \mathcal{R}(\mathcal{W}, \mathcal{G}_t) + \frac{4c\sigma_{\max} BLt^{1-s}}{1-s} \left(\sqrt{\frac{\log 6T/\delta}{\tilde{n}}} + \sqrt{\frac{K}{\tilde{n}}} \right)$. Since ℓ is M smooth, the following holds for any two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^K$ and $y \in \mathcal{Y}$

$$\ell(\mathbf{u} + \mathbf{v}, y) \leqslant \ell(\mathbf{u}, y) + \langle \mathbf{v}, \nabla \ell(\mathbf{u}, y) \rangle + \frac{M\|\mathbf{v}\|_2^2}{2}.$$

Using this smoothness property, $R(W_{t-1}, \phi_t) = \mathbb{E}\left[\ell(W_{t-1}\phi_{t-1}(\mathbf{x}) + \eta_t W_{t-1} g_t, y)\right]$ can be upper bounded as

$$R(W_{t-1}, \phi_t) \leqslant R(W_{t-1}, \phi_{t-1}) + \eta_t \left\langle g_t, \nabla_\phi R(W_{t-1}, \phi_{t-1})\right\rangle_P + \frac{\eta_t^2 M \sigma_{\max}^2 \|g_t\|_P^2}{2}. \tag{11}$$

Combining Equations (10), (11), we get the following bound on $R(W_t, \phi_t)$ which holds with probability at least $1 - \delta/3T$

$$R(W_t, \phi_t) \leqslant R(W_{t-1}, \phi_{t-1}) + \eta_t \left\langle g_t, \nabla_\phi R(W_{t-1}, \phi_{t-1})\right\rangle_P + \frac{\eta_t^2 M \sigma_{\max}^2 B^2}{2} + C_1(t).$$

Next, from Lemma E.2 we know that the g_t chosen by the algorithm satisfies the following with probability at least $1 - 2\delta/3T$

$$\left\langle g_t, -\nabla_\phi R(W_{t-1}, \phi_{t-1})\right\rangle_P \geqslant \beta B \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P - \epsilon_t - C_2(t),$$

where $C_2(t) = 2\sigma_{\max} L \mathcal{R}(\mathcal{G}_t) + 4\sigma_{\max} B L \sqrt{\frac{\log 6T/\delta}{\tilde{n}}}$. Substituting this in the previous equation, we get the following bound on $R(W_t, \phi_t)$ which holds with probability at least $1 - \delta/T$

$$R(W_t, \phi_t) \leqslant R(W_{t-1}, \phi_{t-1}) - \eta_t \beta B \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P + \frac{c^2 M B^2 \sigma_{\max}^2}{2} t^{-2s} \tag{12}$$

$$+ \eta_t \epsilon_t + C_1(t) + \eta_t C_2(t). \tag{13}$$

Let $r_t = R(W_t, \phi_t) - R(W^*, \phi^*) - \sum_{i=1}^t (\eta_i \epsilon_i + C_1(t) + \eta_t C_2(t))$. Then the above equation implies the following recurrence on r_t

$$r_t \leqslant r_{t-1} + \frac{c^2 M B^2 \sigma_{\max}^2}{2} t^{-2s}. \tag{14}$$

We now try to tighten this recurrence. Let $W_{t-1}^\dagger$ be the pseudoinverse of W_{t-1}. From the convexity of ℓ we have

$$\begin{aligned} R(W_{t-1}, \phi_{t-1}) - R(W^*, \phi^*) &\overset{(a)}{=} R(W_{t-1}, \phi_{t-1}) - R(W_{t-1}, W_{t-1}^\dagger W^* \phi^*) \\ &\overset{(b)}{\leqslant} -\left\langle W_{t-1}^\dagger W^* \phi^* - \phi_{t-1}, \nabla_\phi R(W_{t-1}, \phi_{t-1})\right\rangle_P \\ &\leqslant \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P \left(\sigma_{\min}^{-1} \|W^*\|_2 \|\phi^*\|_P + \|\phi_{t-1}\|_P\right), \end{aligned}$$

where (a) follows from the definition of psuedoinverse and (b) follows from the convexity of ℓ. Letting $A_t = \sum_{i=1}^t \eta_i B$, we can lower bound $\|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|$ as

$$\|\nabla_\phi R(W_{t-1}, \phi_{t-1})\| \geqslant \frac{R(W_{t-1}, \phi_{t-1}) - R(W^*, \phi^*)}{\sigma_{\min}^{-1} \|W^*\|_2 \|\phi^*\|_P + A_{t-1}}.$$

Substituting this in Equation (12), we get

$$\begin{aligned} R(W_t, \phi_t) \leqslant & R(W_{t-1}, \phi_{t-1}) - \eta_t \beta B \left(\frac{R(W_{t-1}, \phi_{t-1}) - R(W^*, \phi^*)}{\sigma_{\min}^{-1} \|W^*\|_2 \|\phi^*\|_P + A_{t-1}}\right) + \frac{c^2 M B^2 \sigma_{\max}^2}{2} t^{-2s} \\ & + \eta_t \epsilon_t + C_1(t) + \eta_t C_2(t). \end{aligned}$$

Rewriting the above equation in terms of r_t, we get

$$\begin{aligned} r_t &\leqslant r_{t-1} - \eta_t \beta B \left(\frac{r_{t-1}}{\sigma_{\min}^{-1} \|W^*\|_2 \|\phi^*\|_P + A_{t-1}}\right) + \frac{c^2 M B^2 \sigma_{\max}^2}{2} t^{-2s} \\ &= \left(1 - \frac{\eta_t \beta B}{\sigma_{\min}^{-1} \|W^*\|_2 \|\phi^*\|_P + A_{t-1}}\right) r_{t-1} + \frac{c^2 M B^2 \sigma_{\max}^2}{2} t^{-2s}. \end{aligned} \tag{15}$$

In the rest of the proof, we try to solve the above recurrence relation on r_t to obtain the required excess risk bound. First note that there exists t_0 such that for all $t \geqslant t_0$ [2]

$$\frac{\eta_t \beta B}{\sigma_{\min}^{-1} \|W^*\|_2 \|\phi^*\|_P + A_{t-1}} \geqslant \frac{\alpha + 3\beta(1-s)}{4t}. \tag{16}$$

[2]To be precise, t_0 is such that $t_0^{1-s} = \frac{\alpha \sigma_{\min}^{-1} \|W^*\|_2 \|\phi^*\|_P}{c\beta B}$.

This follows from the observation that $\eta_t = ct^{-s}$ and $A_{t-1} \leqslant \frac{cBt^{1-s}}{1-s}$. We now make use of Theorem I.5 for solving the recurrence in Equation (15). We first show that r_t satisfies the conditions for Theorem I.5 with $a = \alpha, b = (\alpha + \beta(1-s))/2$ and $D = t_0$ and for some A which we specify later. From Equation (14) we have

$$r_{t+1} \leqslant r_t + \frac{c^2 MB^2\sigma_{\max}^2}{2} t^{-2s} \leqslant r_t + A(t-1)^{-\alpha},$$

where the last inequality holds for any $A \geqslant \frac{c^2MB^2\sigma_{\max}^2}{2}$ and for our choice of α, s specified in the theorem statement. This shows that the first condition of Theorem I.5 is satisfied by r_t. Next, suppose $r_t \geqslant At^{-\alpha}$, for some $t \geqslant t_0$. Then using Equations (15) and (16), r_{t+1} can be bounded as follows

$$\begin{aligned} r_{t+1} &\leqslant \left(1 - \frac{\alpha + 3\beta(1-s)}{4t}\right) r_t + \frac{c^2MB^2\sigma_{\max}^2}{2} t^{-2s} \\ &= \left(1 - \frac{\alpha + \beta(1-s)}{2t}\right) r_t \underbrace{- \left(\frac{\beta(1-s) - \alpha}{4t}\right) r_t + \frac{c^2MB^2\sigma_{\max}^2}{2} t^{-2s}}_{T_1}. \end{aligned}$$

Following our choices for α, s and using the fact that $r_t \geqslant At^{-\alpha}$, it is easy to verify that $T_1 \leqslant 0$ for sufficiently large A. This shows that for appropriately chosen A, we have

$$r_{t+1} \leqslant \left(1 - \frac{\alpha + \beta(1-s)}{2t}\right) r_t.$$

Since the conditions for Theorem I.5 are satisfied, using it to solve our recurrence gives us the following bound on r_t which holds with probability at least $1-\delta$

$$r_T \leqslant O\left(\frac{1}{T^\alpha}\right).$$

This finishes the proof of the Theorem.

F Proof of Corollary 4.1

A simple intuition for why the exact greedy approach satisfies similar risk bounds as gradient greedy approach is that in exact greedy approach one solves the greedy step in Equation (2) exactly. Whereas, in gradient greedy approach, the greedy step is only solved approximately and so one would expect the objective value of exact greedy approach to be smaller than gradient greedy approach. We formalize this intuition in the proof. Let (W_t, ϕ_t), where $\phi_t = \phi_{t-1} + \eta_t g_t$, be the t^{th} iterate generated by the exact greedy algorithm. And let $(\tilde{W}, \tilde{\phi}_t)$, where $\tilde{\phi}_t = \phi_{t-1} + \eta_t \tilde{g}_t$, be the iterate obtained by running gradient greedy update in the t^{th} iteration of the algorithm. We now bound $R(W_t, \phi_t)$ in terms of $R(\tilde{W}, \tilde{\phi}_t)$

$$\begin{aligned} R(W_t, \phi_t) &\leqslant \widehat{R}_{S_t}(W_t, \phi_t) + \sup_{W \in \mathcal{W}, g \in \mathcal{G}_t} |R(W, \phi_{t-1} + \eta_t g) - \widehat{R}_{S_t}(W, \phi_{t-1} + \eta_t g)| \\ &\overset{(a)}{\leqslant} \widehat{R}_{S_t}(\tilde{W}_t, \tilde{\phi}_t) + \sup_{W \in \mathcal{W}, g \in \mathcal{G}_t} |R(W, \phi_{t-1} + \eta_t g) - \widehat{R}_{S_t}(W, \phi_{t-1} + \eta_t g)| \\ &\leqslant R((\tilde{W}_t, \tilde{\phi}_t)) + 2 \sup_{W \in \mathcal{W}, g \in \mathcal{G}_t} |R(W, \phi_{t-1} + \eta_t g) - \widehat{R}_{S_t}(W, \phi_{t-1} + \eta_t g)|, \end{aligned}$$

where (a) follows from the definition of W_t, ϕ_t which are obtained by minimizing Equation (2). Note that the supremum in the RHS above can be bounded using Lemma E.1.

From the proof of Theorem 4.1, we know that $R((\tilde{W}_t, \tilde{\phi}_t))$ can be upper bounded in terms of $R((W_{t-1}, \phi_{t-1}))$. To be precise, from Equation (12) in the proof of Theorem 4.1, we know that with probability at least $1 - 2\delta/T$

$$\begin{aligned} R(\tilde{W}_t, \tilde{\phi}_t) \leqslant & R(W_{t-1}, \phi_{t-1}) - \eta_t \beta B \|\nabla_\phi R(W_{t-1}, \phi_{t-1})\|_P + \frac{c^2MB^2\sigma_{\max}^2}{2} t^{-2s} \\ & + \eta_t \epsilon_t + C_1(t) + \eta_t C_2(t). \end{aligned}$$

This shows that

$$\begin{aligned}R(W_t,\phi_t) \leqslant& R(W_{t-1},\phi_{t-1}) - \eta_t\beta B\|\nabla_\phi R(W_{t-1},\phi_{t-1})\|_P + \frac{c^2MB^2\sigma_{\max}^2}{2}t^{-2s}\\ &+ \eta_t\epsilon_t + C_1(t) + \eta_t C_2(t).\end{aligned}$$

Using the exact same techniques as in the proof of Theorem 4.1, we get the required risk bound on $R(W_T,\phi_T)$.

G Proof of Corollary 4.2

The major part of the proof involves bounding the Rademacher complexity terms appearing in the risk bound of Theorem 4.1. We first bound $\mathcal{R}(\mathcal{G})$.

$$\begin{aligned}\mathcal{R}\left(\mathcal{G}\right) &= \mathbb{E}\left[\sup_{g\in\mathcal{G}}\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\sum_{j=1}^{D}\rho_{ij}[g(\mathbf{x}_{t,i})]_j\right]\\ &= \mathbb{E}\left[\sup_{C:\max_j\|C_{j,*}\|_1\leqslant\Lambda}\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\sum_{j=1}^{D}\rho_{ij}\sigma(\langle C_{j,*},\mathbf{x}_{t,i}\rangle)\right]\\ &\leqslant \sum_{j=1}^{D}\mathbb{E}\left[\sup_{\|C_{j,*}\|_1\leqslant\Lambda}\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\rho_{ij}\sigma(\langle C_{j,*},\mathbf{x}_{t,i}\rangle)\right]\\ &\overset{(a)}{\leqslant} \sum_{j=1}^{D}\mathbb{E}\left[\sup_{\|C_{j,*}\|_1\leqslant\Lambda}\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\rho_{ij}\left\langle C_{j,*},\mathbf{x}_{t,i}\right\rangle\right]\\ &\leqslant \sum_{j=1}^{D}\Lambda\mathbb{E}\left[\frac{1}{\tilde{n}}\left\|\sum_{i=1}^{\tilde{n}}\rho_{ij}\mathbf{x}_{t,i}\right\|_\infty\right]\\ &= \frac{D\Lambda}{\tilde{n}}\mathbb{E}\left[\left\|\sum_{i=1}^{\tilde{n}}\rho_{i1}\mathbf{x}_{t,i}\right\|_\infty\right]\\ &\overset{(b)}{\leqslant} 2D\Lambda\sqrt{\frac{\log d}{\tilde{n}}},\end{aligned}$$

where (a) follows from the Lipschitzness of sigmoid activation function and composition property of Rademacher complexities(see Theorem I.3) and (b) follows from the following well known property of sub-Gaussian random variables. Let $Z_1,\dots Z_n$ be n random variables, not necessarily independent. Moreover, lets suppose each Z_i is sub-Gaussian with parameter σ. Then $\mathbb{E}\left[\max_i Z_i\right] \leqslant \sqrt{2\sigma^2\log n}$. Since $\mathcal{X} \subseteq [0,1]^d$, it is easy to see that conditioned on data S_t, each co-ordinate of $\sum_{i=1}^{\tilde{n}}\rho_{i1}\mathbf{x}_{t,i}$ is a sub-Gaussian random variable with parameter $\sqrt{\tilde{n}}$. So using the above stated property of sub-Gaussian random variables, we get

$$\begin{aligned}\mathbb{E}_\rho\left[\left\|\sum_{i=1}^{\tilde{n}}\rho_{i1}\mathbf{x}_{t,i}\right\|_\infty\right] &= \mathbb{E}_\rho\left[\max_{j\in[d]}\max\left\{\sum_{i=1}^{\tilde{n}}\rho_{i1}[\mathbf{x}_{t,i}]_j, -\sum_{i=1}^{\tilde{n}}\rho_{i1}[\mathbf{x}_{t,i}]_j\right\}\right]\\ &\leqslant \sqrt{2\tilde{n}\log 2d}.\end{aligned}$$

Next, we bound $\mathcal{R}(\mathcal{W}, \mathcal{G})$

$$\begin{aligned}
\mathcal{R}(\mathcal{W}, \mathcal{G}) &= \mathbb{E}\left[\sup_{\substack{W \in \mathcal{W}, \\ g \in \mathcal{G}}} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{k=1}^{K} \rho_{ik} [W g(\mathbf{x}_{t,i})]_k\right] \\
&\leqslant \sum_{k=1}^{K} \mathbb{E}\left[\sup_{\substack{W \in \mathcal{W}, \\ g \in \mathcal{G}}} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_{ik} \left\langle W_{k,*}, g(\mathbf{x}_{t,i})\right\rangle\right] \\
&\overset{(a)}{\leqslant} 2\sigma_{\max} K \sum_{j=1}^{D} \mathbb{E}\left[\sup_{\|C_{j,*}\|_1 \leqslant \Lambda} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_i \left\langle C_{j,*}, \mathbf{x}_{t,i}\right\rangle\right] + O\left(\frac{\sigma_{\max} K \sqrt{D} \log D}{\sqrt{\tilde{n}}}\right) \\
&\overset{(b)}{\leqslant} 4\sigma_{\max} K D \Lambda \sqrt{\frac{\log d}{\tilde{n}}} + O\left(\frac{\sigma_{\max} K \sqrt{D} \log D}{\sqrt{\tilde{n}}}\right) \\
&\leqslant O\left(\frac{\sigma_{\max} K D \Lambda \log(dD)}{\sqrt{\tilde{n}}}\right).
\end{aligned}$$

where (a) follows from the property of Rademacher complexity stated in Theorem I.4 and (b) uses the arguments used to bound $\mathcal{R}(\mathcal{G})$ above. Substituting the above bounds for $\mathcal{R}(\mathcal{G})$ and $\mathcal{R}(\mathcal{W}, \mathcal{G})$ in Theorem 4.1 and using the fact that $\sup_X \|g(X)\|_2 \leqslant \sqrt{D}$, for all $g \in \mathcal{G}$, we get the required risk bound.

H Proof of Corollary 4.3

Similar to the proof of Corollary 4.2, we focus on bounding the Radmacher complexity terms $\mathcal{R}(\mathcal{G}_t)$ and $\mathcal{R}(\mathcal{W}, \mathcal{G}_t)$. To bound $\mathcal{R}(\mathcal{G}_t)$, we use the same argument we used to bound $\mathcal{R}(\mathcal{G})$ in Corollary 4.2.

$$\begin{aligned}
\mathcal{R}(\mathcal{G}_t) &= \mathbb{E}\left[\sup_{g \in \mathcal{G}_t} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{j=1}^{D} \rho_{ij} [g(\mathbf{x}_{t,i})]_j\right] \\
&= \mathbb{E}\left[\sup_{C : \max_j \|C_{j,*}\|_1 \leqslant \Lambda} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \sum_{j=1}^{D} \rho_{ij} \sigma(\langle C_{j,*}, \phi_{t-1}(\mathbf{x}_{t,i})\rangle)\right] \\
&\leqslant \sum_{j=1}^{D} \mathbb{E}\left[\sup_{\|C_{j,*}\|_1 \leqslant \Lambda} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_{ij} \sigma(\langle C_{j,*}, \phi_{t-1}(\mathbf{x}_{t,i})\rangle)\right] \\
&\leqslant \sum_{j=1}^{D} \mathbb{E}\left[\sup_{\|C_{j,*}\|_1 \leqslant \Lambda} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \rho_{ij} \left\langle C_{j,*}, \phi_{t-1}(\mathbf{x}_{t,i})\right\rangle\right] \\
&\leqslant \sum_{j=1}^{D} \Lambda \mathbb{E}\left[\frac{1}{\tilde{n}} \left\|\sum_{i=1}^{\tilde{n}} \rho_{ij} \phi_{t-1}(\mathbf{x}_{t,i})\right\|_\infty\right] \\
&= \frac{D\Lambda}{\tilde{n}} \mathbb{E}\left[\left\|\sum_{i=1}^{\tilde{n}} \rho_i \phi_{t-1}(\mathbf{x}_{t,i})\right\|_\infty\right] \\
&\overset{(a)}{\leqslant} \frac{2cD\Lambda t^{1-s}}{1-s} \sqrt{\frac{\log d}{\tilde{n}}},
\end{aligned}$$

where (a) uses similar arguments as in the proof of Corollary 4.2 and relies on the fact that $\|\phi_{t-1}(\mathbf{x})\|_\infty \leqslant \sum_{i=1}^{t-1}\eta_i \leqslant \frac{ct^{1-s}}{1-s}$. Next, we bound $\mathcal{R}(\mathcal{W}, \mathcal{G}_t)$

$$\begin{aligned}
\mathcal{R}(\mathcal{W}, \mathcal{G}_t) &= \mathbb{E}\left[\sup_{\substack{W\in\mathcal{W},\\ g\in\mathcal{G}_t}} \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\sum_{k=1}^{K}\rho_{ik}[Wg(\mathbf{x}_{t,i})]_k\right]\\
&\leqslant \sum_{k=1}^{K}\mathbb{E}\left[\sup_{\substack{W\in\mathcal{W},\\ g\in\mathcal{G}_t}} \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\rho_{ik}\left\langle W_{k,*}, g(\mathbf{x}_{t,i})\right\rangle\right]\\
&\overset{(a)}{\leqslant} 2\sigma_{\max}K\sum_{j=1}^{D}\mathbb{E}\left[\sup_{\|C_{j,*}\|_1\leqslant\Lambda}\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\rho_i\left\langle C_{j,*}, \phi_{t-1}(\mathbf{x}_{t,i})\right\rangle\right] + O\left(\frac{\sigma_{\max}K\sqrt{D}\log D}{\sqrt{\tilde{n}}}\right)\\
&\overset{(b)}{\leqslant} \frac{4\sigma_{\max}cDK\Lambda t^{1-s}}{1-s}\sqrt{\frac{\log d}{\tilde{n}}} + O\left(\frac{\sigma_{\max}K\sqrt{D}\log D}{\sqrt{\tilde{n}}}\right)\\
&\leqslant O\left(t^{1-s}\frac{\sigma_{\max}KD\Lambda\log dD}{\sqrt{\tilde{n}}}\right),
\end{aligned}$$

where (a) follows from the property of Rademacher complexity stated in Theorem I.4 and (b) relies on arguments used to bound $\mathcal{R}(\mathcal{G}_t)$. Substituting the above bounds for $\mathcal{R}(\mathcal{G}_t)$ and $\mathcal{R}(\mathcal{W}, \mathcal{G}_t)$ in Theorem 4.1, we get the required risk bound.

I Some Useful Results

Theorem I.1 (Chernoff Bounds). *Let $X = \sum_{i=1}^n X_i$, where X_i's are independently distributed in $[0,1]$. Then, for $\epsilon \in (0,1)$*

$$\mathbb{P}\left(X > (1+\epsilon)\mathbb{E}[X]\right) \leqslant \exp\left(-\frac{\epsilon^2}{3}\mathbb{E}[X]\right), \quad \mathbb{P}\left(X < (1-\epsilon)\mathbb{E}[X]\right) \leqslant \exp\left(-\frac{\epsilon^2}{2}\mathbb{E}[X]\right).$$

Theorem I.2 (Bartlett and Mendelson [4]). *Let $\mathcal{F}$ be a class of functions mapping $\mathcal{X}$ to $[a,b]$ and let $\{X_i\}_{i=1}^n$ be independently selected according to the probability measure P. Then for any integer n and any $0 < \delta < 1$, with probability at least $1-\delta$ over samples of length n, every f in $\mathcal{F}$ satisfies*

$$\left|\frac{1}{n}\sum_{i=1}^n f(X_i) - \mathbb{E}[f(X)]\right| \leqslant 2\mathcal{R}(\mathcal{F}) + (b-a)\sqrt{\frac{\log 2/\delta}{n}},$$

where $\mathcal{R}(\mathcal{F})$ is the Rademacher complexity of $\mathcal{F}$ which is defined as

$$\mathcal{R}(\mathcal{F}) = \mathbb{E}\left[\sup_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^n \rho_i f(X_i)\right],$$

where the expectation is taken w.r.t the Rademacher random variables ρ's and data $\{X_i\}_{i=1}^n$.

We next present an important result on the composition property of Rademacher complexities.

Theorem I.3 (Maurer [27]). *Let $\mathcal{F}$ be a class of functions mapping $\mathcal{X}$ to $\mathbb{R}^d$ and let $\{h_i\}_{i=1}^n$ be L-Lipschitz functions from $\mathbb{R}^d$ to $\mathbb{R}$. Then*

$$\mathbb{E}_\rho\left[\sup_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^n\rho_i h_i(f(X_i))\right] \leqslant L\mathbb{E}_\rho\left[\sup_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^n\sum_{j=1}^d\rho_{ij}[f(X_i)]_j\right].$$

Theorem I.4 (Proposition A.12 of Allen-Zhu et al. [1]). *Let $u : \mathbb{R} \to \mathbb{R}$ be a fixed 1-Lipschitz function. Given $\mathcal{F}_1 \ldots \mathcal{F}_m$ classes of functions $\mathcal{X} \to \mathbb{R}$ and suppose for each $j \in [m]$ there exists a function $f_j^{(0)} \in \mathcal{F}_j$ satisfying $\sup_{\mathbf{x}\in\mathcal{X}}|u(f_j^{(0)}(\mathbf{x}))| \leqslant A$, then*

$$\mathcal{F}' = \left\{\mathbf{x} \to \sum_{j=1}^m v_j u(f_j(\mathbf{x}))\middle| f_j \in \mathcal{F}_j \wedge \|v\|_1 \leqslant B \wedge \|v\|_\infty \leqslant D\right\}$$

satisfies

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}'}\sum_{i=1}^{n}\frac{1}{n}\rho_i\sum_{j=1}^{m}v_j u(f_j(\mathbf{x}_i))\right] \leqslant 2D\sum_{j=1}^{m}\mathbb{E}\left[\sup_{f\in\mathcal{F}_j}\sum_{i=1}^{n}\frac{1}{n}\rho_i f(\mathbf{x}_i)\right] + O\left(\frac{AB\log m}{\sqrt{n}}\right).$$

Theorem I.5 (Temlyakov [34])**.** *Let four positive numbers $a < b \leqslant 1$, $A, D \in \mathbb{N}$ be given and let a sequence $\{r_t\}_{t=1}^{\infty}$ have the following properties: $r_1 \leqslant A$ and for any $t \geqslant 2$*

$$r_t \leqslant r_{t-1} + A(t-1)^{-a}.$$

Moreover, suppose the sequence is such that if $r_t \geqslant At^{-a}$ for some $t \geqslant D$, then $r_{t+1} \leqslant r_t(1 - b/t)$. Then there exists a constant C such that for all $t \in \mathbb{N}$ we have

$$r_t \leqslant Ct^{-a}.$$

J Experiments

This section provides experimental details, including the datasets, hyperparameter settings, and additional experimental evidence not presented in the main paper.

We first note that in our experiments for DenseCompBoost, we use a slight variant of $\mathcal{G}_t$ defined in Equation (3)

$$\mathcal{G}_t = \left\{ h + g \circ \left(\sum_{i=0}^{t-1}\alpha_i\phi_i\right), \text{ for } h \in \mathcal{H}, g \in \mathcal{G}, \alpha_i \in \mathbb{R} \right\},$$

where $\mathcal{H}, \mathcal{G}$ are weak feature transformer classes. We use this variant because the dimensions of the input feature space and the representation space need not be the same, and as a consequence $\mathcal{G}_t$ in Equation (3) can not always be used. Similar to StdCompBoost, we consider two choices for $\mathcal{H}, \mathcal{G}$: one based on fully connected blocks and the other based on convolution blocks.

J.1 Drawbacks of Layer-by-Layer fitting

In this section, we provide empirical evidence highlighting drawbacks of layer-by-layer fitting and how our proposed techniques address these drawbacks. Similar to Section 5, we use StdCompBoost to denote standard layer-by-layer fitting.

DenseCompBoost can recover from mistakes. We mentioned earlier that compared to StdCompBoost, one advantage of DenseCompBoost is that the dense connections allow it to more easily recover from mistakes made in earlier layers. We now provide empirical evidence to support this claim. We introduce mistakes in the weights of the first layers learned using StdCompBoost and DenseCompBoost. To be precise, we fix the weights of the first layer of both StdCompBoost and DenseCompBoost to (a) the same random matrix, (b) an all-0 matrix, and then continue the training of the later layers. Table 3 shows the results: while StdCompBoost suffers a significant performance drop (from 82.49% when every layer is greedily trained, to 72.99% with a random first layer), the performance of dense greedy is barely affected (from 95.70% when every layer is trained, to 95.0% with a random first layer). Similar trend occurs when setting the first layer to 0: dense greedy still achieves a 93.69% test accuracy, while standard greedy would fail to train at all since any signal in the data has been cut off.

		layer 1	layer 2	layer 3	layer 4	layer 5
StdCompBoost	Random	49.71	50.25	52.51	69.70	72.99
DenseCompBoost	Random	49.71	50.86	70.07	92.31	95.00
	Zero	50.06	61.76	89.19	93.17	93.69

Table 3: Test accuracy at each layer, with the first layer being set to a random value or the all-0 matrix. Compared to the performance without corrupted first layer, StdCompBoost suffers a performance drop, while DenseCompBoost is almost unaffected, demonstrating its ability to recover from mistakes made in early layers.

Narrow-to-Wide architecture of CmplxCompBoost. Note that in CmplxCompBoost, we increase the widths of layers over iterations. We now justify this choice of architecture. There are two possible ways to vary the complexity of the $\tilde{\mathcal{G}}_t$, increasing or decreasing. We tested both approaches on one tabular dataset CovType, and one image dataset SVHN. On CovType, we started with a layer width of 4096, then increase or decrease the width of subsequent layers by 512 at each layer. On SVHN, the starting layer width is 128, followed by 4 additional layers, each increasing or decreasing the width by 16. As can be seen in table 4, increasing complexity gives slightly better results for both the datasets, therefore we choose to increase the width for CmplxCompBoost in all other experiments.

	Decreasing width	Increasing width
CovType	95.58 $\pm$ 0.04	**95.64** $\pm$ 0.16
SVHN	88.30 $\pm$ 0.28	**89.05** $\pm$ 0.01

Table 4: Test accuracy using CmplxCompBoost with decreasing or increasing layer widths.

J.2 Datasets and Hyperparameters

In this section, we present the details of datasets used in our experiments and describe our process for hyperparameter selection.

Simulated Datasets. We generated 3 synthetic binary classification datasets in $\mathbb{R}^{32}$. Simulation 1 is a concentric ellipsoids dataset, where a point $\mathbf{x}$ is classified based on $\mathbf{x}^T A\mathbf{x}$, for some randomly generated positive semi-definite matrix A. Simulations 2 and 3 are datasets whose classification boundaries are polynomials of degrees 8 and 9 respectively. For each of these datasets, we generated 10^6 samples for training and testing.

Hyper-parameters. We used hold-out set validation to pick the best hyper-parameters for all the methods. We used 20% of the training data as validation data and picked the best parameters using grid search, based on validation accuracy. After picking the best parameters, we train on the entire training data and report performance on the test data. For all the greedy techniques based on neural networks, we used fully connected blocks and tuned the following parameters: weight decay, width of weak feature transformers, number of iterations T. For CmplxCompBoost, we set $\Delta = D_0/5$. For end-to-end training, we tuned weight decay, width of layers, depth. We used SGD for optimization of all these techniques. The number of epochs and step size schedule of SGD are chosen to ensure convergence. For XGBoost, we tuned the number of trees, depth of each tree, learning rate.

Benchmark Datasets. We consider the following image datasets: CIFAR10, MNIST, FashionMNIST [35], MNIST-rot-back-image [24], convex [35], SVHN [28], and the following tabular datasets from UCI repository [7]: letter recognition [17], forest cover type (covtype), connect4. The convex dataset involves classifying shapes in images as either convex or non-convex. MNIST-rot-back-image is generated from MNIST by rotating the images and adding random images in the background.

Hyper-parameters. For covtype dataset, which doesn't come with a test set, we randomly sample 20% of the original data and use it as the test set. We use a similar hyper-parameter selection technique as above and tune the same set of hyper-parameters as described above. We use convolution blocks for CIFAR10, SVHN, FashionMNIST, convex, MNIST-rot-back-image and fully connected blocks for the rest. We limit the width of fully connected blocks to 4096, and the number of output channels in convolution blocks to 128 while tuning the hyper-parameters for the composition boosting techniques and end-to-end training. For AdaBoost and additive representation boosting, we set these limits to 16000 and 350 respectively. For CmplxCompBoost with convolution blocks, we set $\Delta = D_0/8$. We *do not* use data augmentation in our experiments.

J.3 Further Experimental Details

Tables 5, 6 list the statistics of datasets used in our experiments. We now list the hyper-parameters tuned for each dataset and learning algorithm. Table 7 presents the list of hyper-parameters tuned for XGBoost. All the other techniques we use in our experiments rely on neural networks. We use SGD with momentum to learn these models. In all our experiments, we set the initial learning rate of SGD to 0.01, momentum to 0.9, batch size to 64 and tune the following weight decay values: $\{0.0001, 0.0005, 0.001, 0.005, 0.01\}$. The number of epochs we used for SGD varied with the dataset and is chosen to be large enough to ensure convergence. Over the course of the SGD optimization, we reduce the learning rate by a factor of 0.5, if the training loss doesn't decrease for certain number of SGD iterations (we rely on scheduler-tolerance option in PyTorch to implement this). We run all the greedy techniques (*AdaBoost, additive feature boosting, StdCompBoost, DenseCompBoost, CmplxCompBoost*) for 10 iterations and use validation dataset to decide the best early stopping rule. For End-2-End training, we tune two values of depth: $5, 10$. Tables 8, 9 presents the list of all the other hyper-parameters tuned.

Table 5: Details of simulated datasets used in our experiments. We use 20% of the training data as validation set for picking the best hyper-parameter

Dataset	Simulation 1	Simulation 2	Simulation 3
# Train samples	1000000	1000000	1000000
# Test samples	500000	500000	500000
# Classes	2	2	2

Table 6: Details of benchmark datasets used in our experiments. We use 20% of the training data as validation set for picking the best hyper-parameter

Details	**Image Datasets**				
	SVHN	FashionMNIST	CIFAR10	Convex	MNIST-rot-back-image
# Train samples	73257	60000	50000	8000	12000
# Test samples	26032	10000	10000	50000	50000
# Classes	10	10	10	2	10

Details	**Tabular Datasets**			
	MNIST	Letter	CovType	Connect4
# Train samples	60000	15000	464809	54045
# Test samples	10000	5000	116203	13512
# Classes	10	26	7	3

Table 7: List of hyper-parameters tuned for XGBoost, on all the datasets used in our experiments.

Parameter	Values Tuned
Tree Depth	$\{10, 15, 20\}$
Learning Rate	$\{0.1, 0.2\}$
Number of Trees	$\{400, 800, 1600\}$

Table 8: List of hyper-parameters tuned for various compositional boosting techniques and end-2-end training.

Dataset	Hyper-parameters tuned
Simulation-1	width:$\{32, 64, 128\}$
Simulation-2	width:$\{64, 128, 256\}$
Simulation-3	width:$\{256, 512, 1024\}$
SVHN	output channels:$\{32, 64, 128\}$
FashionMNIST	output channels:$\{32, 64, 128\}$
Convex	output channels:$\{32, 64, 128\}$
MNIST-rot-back-image	output channels:$\{32, 64, 128\}$
CIFAR10	output channels:$\{32, 64, 128\}$
MNIST	width:$\{256, 512, 1024\}$
LETTER	width:$\{256, 512, 1024\}$
Covtype	width:$\{1024, 2048, 4096\}$
Connect4	width:$\{256, 512, 1024\}$

Table 9: List of hyper-parameters tuned for AdaBoost and additive feature boosting. To be fair for additive boosting techniques, we considered wider weak learners than the ones used for compositional boosting and end-2-end training.

Dataset	Hyper-parameters tuned
Simulation-1	width:$\{256, 512, 1024\}$
Simulation-2	width:$\{256, 512, 1024\}$
Simulation-3	width:$\{4096, 8192, 16384\}$
SVHN	output channels:$\{128, 256, 350, 512\}$
FashionMNIST	output channels:$\{128, 256, 350, 512\}$
Convex	output channels:$\{128, 256, 350, 512\}$
MNIST-rot-back-image	output channels:$\{128, 256, 350, 512\}$
CIFAR10	output channels:$\{128, 256, 350, 512\}$
MNIST	width:$\{256, 512, 1024\}$
LETTER	width:$\{256, 512, 1024\}$
Covtype	width:$\{4096, 8192, 16384\}$
Connect4	width:$\{256, 512, 1024\}$