Functional Gradient Boosting based on Residual Network Perception

Atsushi Nitanda [1 2] **Taiji Suzuki** [1 2]

Abstract

Residual Networks (ResNets) have become state-of-the-art models in deep learning and several theoretical studies have been devoted to understanding why ResNet works so well. One attractive viewpoint on ResNet is that it is optimizing the risk in a functional space by combining an ensemble of effective features. In this paper, we adopt this viewpoint to construct a new *gradient boosting method*, which is known to be very powerful in data analysis. To do so, we formalize the gradient boosting perspective of ResNet mathematically using the notion of functional gradients and propose a new method called *ResFGB* for classification tasks by leveraging ResNet perception. Two types of generalization guarantees are provided from the optimization perspective: one is the margin bound and the other is the expected risk bound by the sample-splitting technique. Experimental results show superior performance of the proposed method over state-of-the-art methods such as LightGBM.

1. Introduction

Deep neural networks have achieved great success in classification tasks; in particular, residual network (ResNet) (He et al., 2016) and its variants such as wide-ResNet (Zagoruyko & Komodakis, 2016), ResNeXt (Xie et al., 2017), and DenseNet (Huang et al., 2017b) have become the most prominent architectures in computer vision. Thus, to reveal a factor in their success, several studies have explored the behavior of ResNets and some promising perceptions have been advocated. Concerning the behavior of ResNets, there are mainly two types of thoughts. One is the ensemble views, which were pointed out in Veit et al. (2016); Littwin & Wolf (2016). They presented that ResNets are ensemble of shallower models using an unraveled view of ResNets. Moreover, Veit et al. (2016) enhanced their claim by showing that dropping or shuffling residual blocks does not affect the performance of ResNets experimentally. The other is the optimization or ordinary differential equation views. In Jastrzebski et al. (2017), it was observed experimentally that ResNet layers iteratively move data representations along the negative gradient of the loss function with respect to hidden representations. Moreover, several studies (Weinan, 2017; Haber et al., 2017; Chang et al., 2017a;b; Lu et al., 2017) have pointed out that ResNet layers can be regarded as discretization steps of ordinary differential equations. Since optimization methods are constructed based on the discretization of gradient flows, these studies are closely related to each other.

On the other hand, gradient boosting (Mason et al., 1999; Friedman, 2001) is known to be a state-of-the-art method in data analysis; in particular, XGBoost (Chen & Guestrin, 2016) and LightGBM (Ke et al., 2017) are notable because of their superior performance. Although ResNets and gradient boosting are prominent methods in different domains, we notice an interesting similarity by recalling that gradient boosting is an ensemble method based on iterative refinement by functional gradients for optimizing predictors. However, there is a key difference between ResNets and gradient boosting methods. While gradient boosting directly updates the predictor, ResNets iteratively optimize the feature extraction by stacking ResNet layers rather than the predictor, according to the existing work.

In this paper, leveraging this observation, we propose a new gradient boosting method called *ResFGB* for classification tasks based on ResNet perception, that is, the feature extraction gradually grows by functional gradient methods in the space of feature extractions and the resulting predictor naturally forms a ResNet-type architecture. The expected benefit of the proposed method over usual gradient boosting methods is that functional gradients with respect to feature extraction can learn a deep model rather than a shallow model like usual gradient boosting. As a result, more efficient optimization is expected.

In the theoretical analysis of the proposed method, we first formalize the gradient boosting perspective of ResNet math-

[1]Graduate School of Information Science and Technology, The University of Tokyo [2]Center for Advanced Intelligence Project, RIKEN. Correspondence to: Atsushi Nitanda <atsushi_nitanda@mist.i.u-tokyo.ac.jp>, Taiji Suzuki <taiji@mist.i.u-tokyo.ac.jp>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

ematically using the notion of functional gradients in the space of feature extractions. That is, we explain that optimization in that space is achieved by stacking ResNet layers. We next show a good consistency property of the functional gradient, which motivates us to find feature extraction with small functional gradient norms for estimating the correct label of data. This fact is very helpful from the optimization perspective because minimizing the gradient norm is much easier than minimizing the objective function without strong convexity. Moreover, we show the margin maximization property of the proposed method and derive the margin bound by utilizing this formalization and the standard complexity analysis techniques developed in Koltchinskii & Panchenko (2002); Bartlett & Mendelson (2002), which guarantee the generalization ability of the method. As for another generalization guarantee, we also provide convergence analysis of the sample-splitting variant of the method for the expected risk minimization. We finally show superior performance, empirically, of the proposed method over state-of-the-art methods including LightGBM.

Related work. Several studies have attempted to grow neural networks sequentially based on the boosting theory. Bengio et al. (2006) introduced convex neural networks consisting of a single hidden layer, and proposed a gradient boosting-based method in which linear classifiers are incrementally added with their weights. However, the expressive power of the convex neural network is somewhat limited because the method cannot learn deep architectures. Moghimi et al. (2016) proposed boosted convolutional neural networks and showed superior empirical performance on fine-grained classification tasks, where convolutional neural networks are iteratively added, while our method constructs a deeper network by iteratively adding layers. Cortes et al. (2017) proposed AdaNet to adaptively learn both the structure of the network and its weight, and provided data-dependent generalization guarantees for an adaptively learned network; however, the learning strategy quite differs from our method and the convergence rate is unclear. The most related work is BoostResNet (Huang et al., 2017a), which constructs ResNet iteratively like our method; however, this method is based on an different theory rather than functional gradient boosting with a constant learning rate. This distinction makes the different optimization-generalization tradeoff. Indeed, our method exhibits a tradeoff with respect to the learning rate, which recalls perception of usual functional gradient boosting methods, namely a smaller learning rate leads to a good generalization performance.

2. Preliminary

In this section, we provide several notations and describe a problem setting of the classification. An important notion in this paper is the functional gradient, which is also introduced in this section.

Problem setting. Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y}$ be a feature space and a finite label set of cardinal c, respectively. We denote by ν a true Borel probability measure on $\mathcal{X} \times \mathcal{Y}$ and by ν_n an empirical probability measure of samples $(x_i, y_i)_{i=1}^n$ independently drawn from ν, i.e., $d\nu_n(X,Y) = \sum_{i=1}^n \delta_{(x_i,y_i)}(X,Y)dXdY/n$, where δ denotes the Dirac delta function. We denote by ν_X the marginal distribution on X and by $\nu(\cdot|X)$ the conditional distribution on Y. We also denote empirical variants of these distributions by $\nu_{n,X}$ and $\nu_n(\cdot|X)$. In general, for a probability measure μ, we denote by $\mathbb{E}_\mu$ the expectation with respect to a random variable according to μ, by $L_2(\mu)$ the space of square-integrable functions with respect to μ, and by $L_2^q(\mu)$ the product space of $L_2(\mu)$ equipped with $\langle \cdot, \cdot \rangle_{L_2^q(\mu)}$-inner product: for $\forall \xi, \forall \zeta \in L_2^q(\mu)$,

$$\langle \xi, \zeta \rangle_{L_2^q(\mu)} \overset{def}{=} \mathbb{E}_\mu[\xi(X)^\top \zeta(X)] = \mathbb{E}_\mu\left[\sum_{j=1}^q \xi_j(X)\zeta_j(X)\right].$$

We also use the following norm: for $\forall p \in (0,2]$ and $\forall \xi \in L_2^q(\mu)$, $\|\xi\|_{L_p^q(\mu)}^p \overset{def}{=} \mathbb{E}_\mu[\|\xi(X)\|_2^p] = \mathbb{E}_\mu\left[(\sum_{j=1}^q \xi_j^2(X))^{p/2}\right]$.

The ultimate goal in classification problems is to find a predictor $f \in L_2^c(\nu_X)$ such that $\arg\max_{y\in\mathcal{Y}} f_y(x)$ correctly classifies its label. The quality of the predictor is measured by a loss function $l(\zeta, y) \geq 0$. A typical choice of l in multiclass classification problems is $l(\zeta, y) = -\log(\exp(\zeta_y)/\sum_{\overline{y}\in\mathcal{Y}}\exp(\zeta_{\overline{y}}))$, which is used for the multiclass logistic regression. The goal of classification is achieved by solving the expected risk minimization problem:

$$\min_{f\in L_2^c(\nu_X)} \left\{ \mathcal{L}(f) \overset{def}{=} \mathbb{E}_\nu[l(f(X),Y)] \right\}. \tag{1}$$

However, the true probability measure ν is unknown, so we approximate $\mathcal{L}$ using the observed data probability measure ν_n and solve the empirical risk minimization problems:

$$\min_{f\in L_2^c(\nu_X)} \left\{ \mathcal{L}_n(f) \overset{def}{=} \mathbb{E}_{\nu_n}[l(f(X),Y)] \right\}. \tag{2}$$

In general, some regularization is needed for the problem (2) to guarantee generalization. In this paper, we rely on early stopping (Zhang et al., 2005) and some restriction on optimization methods for solving the problem.

Similar to neural networks, we split the predictor f into the feature extraction and linear predictor, that is, $f(x) = w^\top \phi(x)$, where $w \in \mathbb{R}^{d\times c}$ is a weight for the last layer and $\phi \in L_2^d(\nu_X)$ is a feature extraction from $\mathcal{X}$ to $\mathcal{X}$. For simplicity, we also denote $l(z, y, w) = l(w^\top z, y)$. Usually, ϕ is parameterized by a neural network and optimized using the stochastic gradient method. In this paper, we propose a way to optimize ϕ in $L_2^d(\nu_X)$ via the following problem:

$$\min_{\substack{w \in \mathbb{R}^{d\times c} \\ \phi \in L_2^d(\nu_X)}} \left\{ \mathcal{R}(\phi, w) \overset{def}{=} \mathbb{E}_\nu[l(\phi(X), Y, w)] + \frac{\lambda}{2}\|w\|_2^2 \right\} \quad (3)$$

where $\lambda > 0$ is a regularization parameter to stabilize the optimization procedure and $\|\cdot\|_2$ for w is a Euclidean norm. When we focus on the problem with respect to ϕ, we use the notation $\mathcal{R}(\phi) \overset{def}{=} \min_{w\in\mathbb{R}^{d\times c}} \mathcal{R}(\phi, w)$. We also denote by $\mathcal{R}_n(\phi, w)$ and $\mathcal{R}_n(\phi)$ empirical variants of $\mathcal{R}(\phi, w)$ and $\mathcal{R}(\phi)$, respectively, which are defined by replacing $\mathbb{E}_\nu$ by $\mathbb{E}_{\nu_n}$. In this paper, we denote by ∂ the partial derivative and its subscript indicates the direction.

Functional gradient. The key notion used for solving the problem is the functional gradient in function spaces. Since they are taken in some function spaces, we first introduce Fréchet differential in general Hilbert spaces.

Definition 1. *Let $\mathcal{H}$ be a Hilbert space and h be a function on $\mathcal{H}$. For $\xi \in \mathcal{H}$, we call that h is Fréchet differentiable at ξ in $\mathcal{H}$ when there exists an element $\nabla_\xi h(\xi) \in \mathcal{H}$ such that*

$$h(\zeta) = h(\xi) + \langle \nabla_\xi h(\xi), \zeta - \xi \rangle_{\mathcal{H}} + o(\|\xi - \zeta\|_{\mathcal{H}}).$$

Moreover, for simplicity, we call $\nabla_\xi h(\xi)$ Fréchet differential or functional gradient.

We here make an assumption to guarantee Fréchet differentiability of $\mathcal{R}, \mathcal{R}_n$, which is valid for multiclass logistic loss: $l(z, y, w) = -\log(\exp(w_y^\top z)/\sum_{\overline{y}\in\mathcal{Y}} \exp(w_{\overline{y}}^\top z))$.

Assumption 1. *The loss function $l(z, y, w) : \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^{d\times c} \to \mathbb{R}$ is a non-negative $\mathcal{C}^2$-function with respect to z and w, convex with respect to w, and satisfies the following smoothness: There exists a real number $A_r > 0$ depending on $r > 0$ such that*

$$\|\partial_z^2 l(z, y, w)\| \leq A_r \ for\ z \in \mathcal{X}, y \in \mathcal{Y}, w \in B_r(0),$$

where $\|\cdot\|$ used for a matrix $\partial_z^2 l$ is the spectral norm and $B_r(0) \subset \mathbb{R}^{d\times c}$ is a closed ball of center 0 and radius r.

For $\phi \in L_2^d(\nu_X)$, we set $w_\phi \overset{def}{=} \arg\min_{w\in\mathbb{R}^{d\times c}} \mathcal{R}(\phi, w)$ and $w_{n,\phi} \overset{def}{=} \arg\min_{w\in\mathbb{R}^{d\times c}} \mathcal{R}_n(\phi, w)$. Moreover, we define the following notations:

$$\nabla_\phi \mathcal{R}(\phi)(x) \overset{def}{=} \mathbb{E}_{\nu(Y|x)}[\partial_z l(\phi(x), Y, w_\phi)],$$

$$\nabla_\phi \mathcal{R}_n(\phi)(x) \overset{def}{=} \begin{cases} \partial_z l(\phi(x_i), y_i, w_{n,\phi}) & (x = x_i), \\ 0 & (\text{otherwise}). \end{cases}$$

We also similarly define functional gradients $\partial_\phi \mathcal{R}(\phi, w)$ and $\partial_\phi \mathcal{R}_n(\phi, w)$ for fixed w by replacing $w_\phi, w_{n,\phi}$ by w. It follows that

$$\nabla_\phi \mathcal{R}(\phi) = \partial_\phi \mathcal{R}(\phi, w_\phi), \nabla_\phi \mathcal{R}_n(\phi) = \partial_\phi \mathcal{R}_n(\phi, w_{n,\phi}).$$

The next proposition means that the above maps are functional gradients in $L_2^d(\nu_X)$ and $L_2^d(\nu_{n,X})$. We set $l_0 = \max_{y\in\mathcal{Y}} l(0, y)$.

Proposition 1. *Let Assumption 1 hold. Then, for $\forall\phi, \psi \in L_2^d(\nu_X)$, it follows that*

$$\mathcal{R}(\psi) = \mathcal{R}(\phi) + \langle \nabla_\phi \mathcal{R}(\phi), \psi - \phi \rangle_{L_2^d(\nu_X)} + H_\phi(\psi), \quad (4)$$

where $H_\phi(\psi) \leq \frac{A_{c_\lambda}}{2}\|\phi - \psi\|_{L_2^d(\nu_X)}^2$ ($c_\lambda = \sqrt{2l_0/\lambda}$). Furthermore, the corresponding statements hold for $\mathcal{R}(\cdot, w)$ ($\forall w \in \mathbb{R}^d$) by replacing $\mathcal{R}(\cdot)$ by $\mathcal{R}(\cdot, w)$ and for empirical variants by replacing ν_X by $\nu_{n,X}$.

We can also show differentiability of $\mathcal{L}(f)$ and $\mathcal{L}_n(f)$. Their functional gradients have the form $\nabla_f \mathcal{L}(f)(x) = \mathbb{E}_{\nu(Y|x)}[\partial_\zeta l(f(x), Y)]$ and $\nabla_f \mathcal{L}_n(f)(x_i) = \partial_\zeta l(f(x_i), y_i)$. In this paper, we derive functional gradient methods using $\nabla_\phi \mathcal{R}_n(\phi)$ rather than $\nabla_f \mathcal{L}_n(f)$ like usual gradient boosting (Mason et al., 1999; Friedman, 2001), and provide convergence analyses for problems (1) and (2). However, we cannot apply $\nabla_\phi \mathcal{R}_n(\phi)$ or $\partial_\phi \mathcal{R}_n(\phi, w)$ directly to the expected risk minimization problem because these functional gradients are zero outside the training data. Thus, we need a smoothing technique to propagate these to unseen data. The expected benefit of functional gradient methods using $\nabla_\phi \mathcal{R}_n(\phi)$ over usual gradient boosting is that the former can learn a deep model that is known to have high representational power. Before providing a concrete algorithm description, we first explain the basic property of functional gradients and functional gradient methods.

3. Basic Property of Functional Gradient

In this section, we explain the motivation for using functional gradients for solving classification problems. We first show the consistency of functional gradient norms, namely predicted probabilities by predictors with small norms converge to empirical/expected conditional probabilities. We next explain the superior performance of functional gradient methods intuitively, which motivate us to use it for finding predictors with small norms. Moreover, we explain that the optimization procedure of functional gradient methods can be realized by stacking ResNet layers iteratively on the top of feature extractions.

Consistency of functional gradient norm. We here provide upper bounds on the gaps between true empirical/expected conditional probabilities and predicted probabilities.

Proposition 2. *Let $l(\zeta, y)$ be the loss function for the multiclass logistic regression. Then,*

$$\|\nabla_f \mathcal{L}(f)\|_{L_1^c(\nu_X)} \geq \frac{1}{\sqrt{c}} \sum_{y\in\mathcal{Y}} \|\nu(y|\cdot) - p_f(y|\cdot)\|_{L_1(\nu_X)},$$

$$\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})} \geq \frac{1}{\sqrt{c}} \sum_{y\in\mathcal{Y}} \|\nu_n(y|\cdot) - p_f(y|\cdot)\|_{L_1(\nu_{n,X})},$$

where we denote by $p_f(y|x)$ the softmax function defined by the predictor f, i.e., $\exp(f_y(\cdot))/\sum_{\overline{y}\in\mathcal{Y}}\exp(f_{\overline{y}}(\cdot))$.

Many studies (Zhang, 2004; Steinwart, 2005; Bartlett et al., 2006) have exploited the consistency of convex loss functions for classification problems in terms of the classification error or conditional probability. Basically, these studies used the excess empirical/expected risk to estimate the excess classification error or the gap between the true conditional probability and the predicted probability. On the other hand, Proposition 2 argues that functional gradient norms give sufficient bounds on such gaps. This fact is very helpful from the optimization perspective for non-strongly convex smooth problems since the excess risk always bounds the functional gradient norm by the reasonable order, but the inverse relationship does not always hold. This means that finding a predictor with a small functional gradient is much easier than finding a small excess risk.

Note that the latter inequality in Proposition 2 provides the lower bound on empirical classification accuracy, which is confirmed by Markov inequality as follows.

$$\begin{aligned}\mathbb{P}_{\nu_n}[1 - p_f(Y|X) \geq 1/2] &\leq 2\mathbb{E}_{\nu_n}[1 - p_f(Y|X)] \\ &\leq 2\sqrt{c}\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.\end{aligned}$$

Generally, we can derive a bound on the empirical margin distribution (Koltchinskii & Panchenko, 2002) by using the functional gradient norm in a similar way, and can obtain a generalization bound using it, as shown later.

Powerful optimization ability and connection to residual networks. In the above discussion, we have seen that the classification problem can be reinterpreted as finding a predictor with small functional gradient norms, which may lead to reasonable convergence compared to minimizing the excess risk. However, finding such a good predictor is still difficult because a function space is quite comprehensive, and thus, a superior optimization method is required to achieve this goal. We explain that functional gradient methods exhibit an excellent performance by using the simplified problem. Namely, we apply the functional gradient method to the following problem:

$$\min_{\phi\in L_2^d(\nu_X)} \left\{ \mathcal{R}'(\phi) \overset{def}{=} \mathbb{E}_{\nu_X}[r(\phi(X))] \right\}, \tag{5}$$

where r is a sufficiently smooth function. Note that the main problem (3) is not interpreted as this simplified problem correctly, but is useful in explaining a property and an advantage of the method and leads to a deeper understanding.

If $\mathcal{R}'$ is Fréchet differentiable, the functional gradient is represented as $\nabla_\phi \mathcal{R}'(\phi)(\cdot) = \nabla_z r(\phi(\cdot))$, where z indicates the input to r. Therefore, the negative functional gradient indicates the direction of decreasing the objective r at each point $\phi(x)$. An iteration of the functional gradient method with a learning rate η is described as

$$\phi_{t+1} \leftarrow \phi_t - \eta\nabla_z r \circ \phi_t = (id - \eta\nabla_z r)\circ\phi_t.$$

We can immediately notice that this iterate makes ϕ_t one level deeper by stacking a residual network-type layer $id - \eta\nabla_z r$ (He et al., 2016), and data representation is refined through this layer. Starting from a simple feature extraction ϕ_0 and running the functional gradient method for T-iterations, we finally obtain a residual network:

$$\phi_T = (id - \eta\nabla_z r)\circ\cdots\circ(id - \eta\nabla_z r)\circ\phi_0.$$

Therefore, feature extraction ϕ gradually grows by using the functional gradient method to optimize $\mathcal{R}'$. Indeed, we can show the convergence of ϕ_T to a stationary point of $\mathcal{R}'$ in $L_2^d(\nu_X)$ under smoothness and boundedness assumptions by analogy with a finite-dimensional optimization method. This is a huge advantage of the functional gradient method because stationary points in $L_2^d(\nu_X)$ are potentially significant better than those of finite-dimensional spaces. Note that this formulation explains the optimization view (Jastrzebski et al., 2017) of ResNet mathematically.

We now briefly explain how powerful the functional gradient method is compared to the gradient method in a finite-dimensional space, for optimizing $\mathcal{R}'$. Let us consider any parameterization of $\phi_t \in L_2^d(\nu_X)$. That is, we assume that ϕ_t is contained in a family of parameterized feature extractions $\mathcal{M} = \{g_\theta : \mathcal{X} \to \mathcal{X} \mid \theta \in \Theta \subset \mathbb{R}^m\} \subset L_2^d(\nu_X)$, i.e., $\exists\theta' \in \Theta\ s.t.\ \phi_t = g_{\theta'}$. Typically, the family $\mathcal{M}$ is given by neural networks. If $\mathcal{R}'(g_\theta)$ is differentiable at θ', we get $\nabla_\theta \mathcal{R}'(g_\theta)|_{\theta=\theta'} = \langle \nabla_\phi \mathcal{R}'(\phi_t), \nabla_\theta g|_{\theta=\theta'}\rangle_{L_2^d(\nu_X)}$ according to the chain rule of derivatives. Note that $\nabla_\phi \mathcal{R}'(\phi_t)$ dominates the norm of gradients. Namely, if ϕ_t is a stationary point in $L_2^d(\nu_X)$, ϕ_t is also a stationary point in $\mathcal{M}$ and there is no room for improvement using gradient-based methods. This result holds for any family $\mathcal{M}$, but the inverse relation does not always hold. This means that gradient-based methods may fail to optimize $\mathcal{R}'$ in the function space, while the functional gradient method exceeds such a limit by making a feature extraction ϕ_t deeper. For detailed descriptions and related work in this line, we refer to Ambrosio et al. (2008); Daneri & Savaré (2010); Sonoda & Murata (2017); Nitanda & Suzuki (2017; 2018).

4. Algorithm Description

In this section, we provide concrete description of the proposed method. Let $\phi_t \in L_2^d(\nu_X)$ and w_t denote t-th iterates of ϕ and w. As mentioned above, since functional gradients $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$ for the empirical risk vanish outside the training data, we need a smoothing technique to propagate these to unseen data. Hence, we use the convolution $T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$ of the functional gradient by using an adaptively chosen kernel function k_t on $\mathcal{X}$. The convolution is applied element-wise as follows.

$$\begin{aligned} T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1}) &\overset{def}{=} \mathbb{E}_{\nu_{n,X}}[\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(X)k_t(X,\cdot)] \\ &= \frac{1}{n}\sum_{i=1}^{n} \partial_z l(\phi_t(x_i), y_i, w_{t+1})k_t(x_i,\cdot). \end{aligned}$$

Namely, this quantity is a weighted sum of $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(x_i)$ by $k_t(x_i,\cdot)$, which we also call a functional gradient. In particular, we restrict the form of a kernel k_t to the inner-product of a non-linear feature embedding to a finite-dimensional space by $\iota_t : \mathbb{R}^d \to \mathbb{R}^D$, that is, $k_t(x,x') = \iota_t(\phi_t(x))^\top \iota_t(\phi_t(x))$. The requirements on the choice of ι_t to guarantee the convergence are the uniform boundedness and sufficiently preserving the magnitude of the functional gradient $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$. Let $\mathcal{F}$ be a given restricted class of bounded embeddings. We pick up ι_t from this class $\mathcal{F}$ by approximately solving the following problem to acquire magnitude preservation:

$$\max_{\iota_t \in \mathcal{F}} \|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2. \tag{6}$$

where we define $\|T_{k_t,n}\xi\|_{k_t}^2 = \langle \xi, T_{k_t,n}\xi\rangle_{L_2^d(\nu_{n,X})}$ for a vector function ξ. Detailed conditions on ι_t and an alternative problem to guarantee the convergence will be discussed later. Note that due to the restriction on the form of k_t, the computation of the functional gradient is compressed to the matrix-vector product. Namely,

$$\begin{aligned} &A_t \overset{def}{=} \frac{1}{n}\sum_{i=1}^{n} \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(x_i)\iota_t(\phi_t(x_i))^\top, \\ &T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1}) = A_t\iota_t(\phi_t(\cdot)). \end{aligned}$$

Therefore, the functional gradient method $\phi_{t+1} \leftarrow \phi_t - \eta T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$ can be recognized as the procedure of successively stacking layers $id - \eta A_t\iota_t(\phi_t(\cdot))$ ($t \in \{0,\ldots,T-1\}$) and obtaining a residual network. The entire algorithm is described in Algorithm 1. Note that because a loss function l is chosen typically to be convex with respect to w, a procedure in Algorithm 1 to obtain w_{n,ϕ_t} is easily achieved by running an efficient method for convex minimization problems. The notation T_0 is the stopping time of iterates with respect to w. That is, functional gradients $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})$ are computed at $w_{t+1} = w_{n,\phi_t}$ and correspond to $\nabla_\phi \mathcal{R}_n(\phi_t)$ when $t < T_0$ and computed at an older point of w when $t \geq T_0$, rather than $\nabla_\phi \mathcal{R}_n(\phi_t)$.

Algorithm 1 ResFGB

```
Input: $S = (x_i, y_i)_{i=1}^n$, initial points $\phi_0,\ w_0$, the number
of iterations $T$ of $\phi$, the number of iterations $T_0$ of $w$,
embedding class $\mathcal{F}$, and learning rate $\eta$
for $t = 0$ to $T-1$ do
  if $t < T_0$ then
    $w_{t+1} \leftarrow w_{n,\phi_t} = \arg\min_{w\in\mathbb{R}^{d\times c}} \mathcal{R}_n(\phi_t, w)$
  else
    $w_{t+1} \leftarrow w_t$
  end if
  Get $\iota_t$ by approximately solving (6) on $S$
  $A_t \leftarrow \frac{1}{n}\sum_{i=1}^n \partial_z l(\phi_t(x_i), y_i, w_{t+1})\iota_t(\phi_t(x_i))^\top$
  $\phi_{t+1} \leftarrow \phi_t - \eta A_t\iota_t(\phi_t(\cdot))$
end for
Return $\phi_{T-1}$ and $w_T$
```

Choice of embedding. We here provide policies for the choice of ι_t. A sufficient condition for ι_t to achieve good convergence is to maintain the functional gradient norm, which is summarized below.

Assumption 2. *For positive values γ, ϵ, $p \leq 2$, q, and K, a function $k_t(x,x') = \iota_t(\phi_t(x))^\top \iota_t(\phi_t(x))$ satisfies $\|\iota_t(x)\|_2 \leq \sqrt{K}$ on $\mathcal{X}$, and $\gamma\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_p^d(\nu_{n,X})}^q - \gamma\epsilon \leq \|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2$.*

This assumption is a counterpart of that imposed in Mason et al. (1999). The existence of ι_t, not necessarily included in $\mathcal{F}$, satisfying this assumption is confirmed as follows. We here assume that ϕ_t is a bijection that is a realistic assumption when learning rates are sufficiently small because of the inverse mapping theorem. Then, since $\nu_n(\cdot|X) = \nu_n(\cdot|\phi_t(X))$, functional gradients $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(x)$ become the map of $\phi_t(x)$, so we can choose ι_t such that

$$\iota_t(\phi_t(\cdot)) = \partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)/\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)\|_2.$$

By simple computation, we find that $k_t(x,x') \leq 1$ and $\|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2$ are lower-bounded by $\frac{1}{d}\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{L_1^d(\nu_{n,X})}^2$. A detailed derivation is provided in Appendix. Thus, Assumption 2 may be satisfied if an embedding class $\mathcal{F}$ is sufficiently large, but we note that too large $\mathcal{F}$ leads to overfitting. Therefore, one way of choosing ι_t is to approximate $\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)/\|\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})(\cdot)\|_2$ rather than maximizing (6) directly, and indeed, this procedure has been adopted in experiments.

5. Convergence Analysis

In this section, we provide a convergence analysis for the proposed method. All proofs are included in Appendix. For the empirical risk minimization problem, we first show the global convergence rate, which also provides the generalization bound by combining the standard complexity analyses. Next, for the expected risk minimization problem, we describe how the size of $\mathcal{F}$ and the learning rate control the tradeoff between optimization speed and generalization by using the sample-splitting variant of Algorithm 1, whose detailed description will be provided later.

Empirical risk minimization. Using Proposition 1, Assumption 2, and an additional assumption on w_t, we can show the global convergence of Algorithm 1. The following inequality shows how functional gradients decrease the objective function, which is a direct consequence of Proposition 1. When $\eta \leq \frac{1}{A_{c_\lambda} K}$, we have

$$\mathcal{R}_n(\phi_{t+1}, w_{t+2}) \leq \mathcal{R}_n(\phi_t, w_{t+1}) - \frac{\eta}{2}\|T_{k_t,n}\partial_\phi \mathcal{R}_n(\phi_t, w_{t+1})\|_{k_t}^2.$$

Therefore, Algorithm 1 provides a certain decrease in the objective function; moreover, we can conclude a stronger result.

Theorem 1. *Let Assumptions 1 and 2 hold. Consider running Algorithm 1 with a learning rate $\eta \leq \frac{1}{A_{c_\lambda} K}$. If $p \geq 1$ and the minimum eigenvalues of $(w_t{}^\top w_t)_{t=0}^{T_0}$ have a uniform lower bound $\sigma^2 > 0$, then we get*

$$\min_{t\in[T]} \|\nabla_f \mathcal{L}_n(f_t)\|_{L_1^c(\nu_{n,X})} \leq \left(\frac{2\mathcal{R}_n(\phi_0, w_1)}{\eta\gamma\sigma^q T} + \frac{\epsilon}{\sigma^q}\right)^{\frac{1}{q}} \tag{7}$$

where we denote $f_t = w_{t+1}^\top \phi_t$ and $[T] = \{0, \dots, T-1\}$.

Remark. (i) This theorem states the convergence of the minimum functional gradient norm obtained by running Algorithm 1, but returning the last iterate or the best iterate on the validation set is practically better. (ii) Although a larger value of T_0 may affect the bound in Theorem 1 because of dependency on the minimum eigenvalue of $(w_t^\top w_t)_{t=0}^{T_0}$, optimizing w at each iteration facilitates the convergence speed empirically.

Generalization bound. Here, we derive a generalization bound using the margin bound developed by Koltchinskii & Panchenko (2002), which is composed of the sum of the empirical margin distribution and Rademacher complexity of predictors. The margin and the empirical margin distribution for multiclass classification are defined as $m_f(x,y) \overset{def}{=} f_y(x) - \max_{y'\neq y} f_{y'}(x)$ and $\mathbb{P}_{\nu_n}[m_f(x,y) \leq \delta]$ ($\delta > 0$), respectively. When l is the multiclass logistic loss, using Markov inequality and Proposition 2, we can obtain an upper bound on the margin distribution:

$$\mathbb{P}_{\nu_n}[m_f(x,y) \leq \delta] \leq \left(1 + \frac{1}{\exp(-\delta)}\right)\sqrt{c}\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.$$

Since the convergence of functional gradient norms has been shown in Theorem 1, the resulting problem to derive a generalization bound is to estimate Rademacher complexity, which can be achieved using standard techniques developed by Bartlett & Mendelson (2002); Koltchinskii & Panchenko (2002). Thus, we specify here the architecture of predictors. In the theoretical analysis, we suppose $\mathcal{F}$ is the set of shallow neural networks $B\sigma(Cx)$ for simplicity, where B, C are weight matrices and σ is an element-wise activation function. Then, the t-th layer is represented as

$$\phi_{t+1}(x) = \phi_t(x) - \eta D_t \sigma(C_t \phi_t(x)),$$

where $D_t = A_t B_t$, and a predictor is $f_{T-1}(x) = w_T^\top \phi_{T-1}(x)$. Bounding norms of these weights by controlling the size of $\mathcal{F}$ and λ, we can restrict the Rademacher complexity of a set of predictors and obtain a generalization bound. We denote by $\mathcal{G}_{T-1}$ the set of predictors under constraints on weight matrices where L_1-norms of each row of $w_T^\top$, C_t, and D_t are bounded by Λ_w, Λ, and Λ'.

Theorem 2. *Let l be the multiclass logistic regression loss. Fix $\delta > 0$. Suppose σ is L_σ-Lipschitz continuous and $\|x\|_2 \leq \Lambda_\infty$ on $\mathcal{X}$. Then, for $\forall\rho > 0$, with probability at least $1-\rho$ over the random choice of S from ν^n, we have $\forall f \in \mathcal{G}_{T-1}$,*

$$\mathbb{P}_\nu[m_f(X,Y) \leq 0] \leq \frac{2c^3\Lambda_\infty\Lambda_w}{\delta\sqrt{n}}(1+\eta\Lambda\Lambda' L_\sigma)^{T-1} + \sqrt{\frac{\log(1/\rho)}{2n}} + \left(1 + \frac{1}{\exp(-\delta)}\right)\sqrt{c}\|\nabla_f \mathcal{L}_n(f)\|_{L_1^c(\nu_{n,X})}.$$

Combining Theorems 1 and 2, we observe that the learning rate η, the number of iterations T, and the size of $\mathcal{F}$ have an impact on the optimization-generalization tradeoff, that is, larger values of these quantities facilitate the convergence on training data while the generalization bound becomes gradually loose. Especially, this bound has an exponential dependence on depth T, which is known to be unavoidable (Neyshabur et al., 2015) in the worst case for some networks with L_1 or the group norm constraints, but this bound is useful when an initial objective is small and required T is also small sufficiently.

We note another type of bound can be derived by utilizing VC-dimension or pseudo-dimension (Vapnik & Chervonenkis, 1971). When the activation function is piece-wise linear, such as Relu function $\sigma(x) = \max\{0, x\}$, reasonable bounds on these quantities are given by Bartlett et al. (1998; 2017). Thus, for that case, we can obtain better bounds with respect to T by combining our analysis and the VC bound, but we omit the precise description for simplicity. We next show the other generalization guarantee from the optimization perspective by using the modified algorithm, which may slow down the optimization speed but alleviates the exponential dependence on T in the generalization bound.

Sample-splitting technique. To remedy the exponential dependence on T of the generalization bound, we introduce the sample-splitting technique which has been used recently to provide statistical guarantee of expectation-maximization algorithms (Balakrishnan et al., 2017; Wang et al., 2015). That is, instead of Algorithm 1, we analyze its sample-splitting variant. Although Algorithm 1 exhibits good empirical performance, the sample-splitting variant is useful for analyzing the behavior of the expected risk. In this variant, the entire dataset is split into T pieces, where T is the number of iterations, and each iteration uses a fresh batch of samples. The key benefit of the sample-splitting method is that it allows us to use concentration inequalities independently at each iterate ϕ_t rather than using the complexity measure of the entire model. As a result, sample-splitting alleviates the exponential dependence on T presented in Theorem 2. We now present the details in Algorithm 2. For simplicity, we assume $T_0 = 0$, namely the weight vector w_t is fixed to the initial weight w_0.

Algorithm 2 Sample-splitting ResFGB

```
  Input: $S = (x_i, y_i)_{i=1}^n$, initial points $\phi_0,\ w_0$, the number of iterations $T$, embedding class $\mathcal{F}$, and learning rates $\eta$
  Split $S$ into $T$ disjoint subsets $S_1, \ldots, S_T$ of size $\lfloor n/T \rfloor$
  for $t = 0$ to $T-1$ do
    Define $\mathcal{R}_{\lfloor n/T \rfloor}(\phi_t, w)$ using $S_t$
    Get $\iota_t$ by approximately solving (6) on $S_t$
    $A_t \leftarrow \lfloor \frac{T}{n} \rfloor \sum_{i=1}^{\lfloor n/T \rfloor} \partial_z l(\phi_t(x_i), y_i, w_0)\iota_t(\phi_t(x_i))^\top$
    $\phi_{t+1} \leftarrow \phi_t - \eta A_t \iota_t(\phi_t(\cdot))$
  end for
  Return $\phi_{T-1}$ and $w_0$
```

Our proof mainly relies on bounding a statistical error of the functional gradient at each iteration in Algorithm 2. Because the population version of Algorithm 1 strictly decreases the value of $\mathcal{R}$ due to its smoothness, we can show that Algorithm 2 also decreases it with high probability when the norm of a functional gradient is larger than a statistical error bound. Thus, we make here an additional assumption on the loss function to bound the statistical error, which is satisfied for a multiclass logistic loss function.

Assumption 3. *For the differentiable loss function $l(z, y, w)$ with respect to z, w, there exists a positive real number β_r depending on $r > 0$ such that $\|\partial_z l(z, y, w)\|_2 \leq \beta_r$ for $z \in \mathcal{X}, y \in \mathcal{Y}, w \in B_r(0)$.*

We here introduce the notation required to describe the statement. We let $\mathcal{F}^j$ be a collection of j-th elements of functions in $\mathcal{F}$. For a positive value M, we set

$$\epsilon(m, \rho) \overset{def}{=} \beta_{\|w_0\|_2} \sqrt{\frac{KdD}{m}} \left(2M + \sqrt{2K \log \frac{2dD}{\rho}} \right).$$

The following proposition is a key result to bound a statistical error as mentioned above.

Proposition 3. *Let Assumption 3 hold and each $\mathcal{F}^j$ be the VC-class (for the definition see van der Vaart & Wellner (1996)). For $\iota \in \mathcal{F}$, we assume $\|\iota(x)\|_2 \leq \sqrt{K}$ on $\mathcal{X}$. We set μ to be ν_X or $\nu_{m,X}$ and $k(x, x')$ to be $\iota(\phi(x))^\top \iota(\phi(x'))$. Then, there exists a positive value M depending on $\mathcal{F}$ and it follows that with probability at least $1-\rho$ over the choice of the sample of size m, $\epsilon(m, \rho)$ upper-bounds the following.*

$$\sup_{\iota \in \mathcal{F}} \|T_k \partial_\phi \mathcal{R}(\phi, w_0) - T_{k,m} \partial_\phi \mathcal{R}_m(\phi, w_0)\|_{L_2^d(\mu)}.$$

Since each iterate in Algorithm 2 is computed on a fresh batch not depending on previous batches, Proposition 3 can be applied to all iterates with $m \leftarrow \lfloor n/T \rfloor$ and $\rho \leftarrow \delta/T$ for $\delta \in (0, 1)$. Thus, when $\lfloor n/T \rfloor$ is large and η is small sufficiently, functional gradients used in Algorithm 2 become good approximation to the population variant, and we find that the expected risk function is likely to decrease from Proposition 1. Moreover, we note that statistical errors are accumulated additively rather than the exponential growth. Concretely, we obtain the following generalization guarantee.

Theorem 3. *Let Assumptions 1, 2, and 3 and the same assumption in Proposition 3 hold. Consider running Algorithm 2. If $p \geq 1$, $\|\partial_\zeta l(\zeta, y)\|_2 \leq B$, and the minimum eigenvalue of ${w_0}^\top w_0$ is lower-bounded by $\sigma^2 > 0$, then we get with probability at least $1 - \rho$,*

$$\|\nabla_f \mathcal{L}(w_0^\top \phi_{t_*})\|_{L_1^c(\nu_X)} \leq B \left(\frac{2T}{n} \log \frac{T}{\rho} \right)^{\frac{1}{4}} + \sqrt{\frac{B}{\gamma^{\frac{1}{q}} \sigma}}$$
$$\cdot \left\{ \frac{\mathcal{R}_0}{\eta T} + \beta_{\|w_0\|_2} \epsilon \left(\frac{n}{T}, \frac{\rho}{T} \right) + \frac{\eta}{2} A_{\|w_0\|_2} K^2 \beta_{\|w_0\|_2}^2 + \gamma\epsilon \right\}^{\frac{1}{2q}}$$

where $\mathcal{R}_0 = \mathcal{R}(w_0, \phi_0)$ and t_* is the index giving the minimum value of $\|\nabla_f \mathcal{L}_{\lfloor n/T \rfloor}(w_0^\top \phi_t)\|_{L_p^c(\nu_{\lfloor n T \rfloor, X})}$.

6. Experiments

In this section, we present experimental results of the binary and multiclass classification tasks. We run Algorithm 1 and compare it with support vector machine, random forest, multilayer perceptron, and gradient boosting methods. We here introduce settings used for Algorithm 1. As for the loss function, we test both multiclass logistic loss and smooth hinge loss, and as for the embedding class $\mathcal{F}$, we use two or three hidden-layer neural networks. The number of hidden units in each layer is set to 100 or 1000. Linear classifiers and embeddings are trained by Nesterov's momentum method. The learning rate is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$. These parameters and the number of iterations T are tuned based on the performance on the validation set.

Table 1. Test classification accuracy on binary and multiclass classification.

METHOD	LETTER	USPS	IJCNN1	MNIST	COVTYPE	SUSY
RESFGB (LOGISTIC)	**0.976** **(0.0019)**	**0.953** **(0.0007)**	**0.989** **(0.0004)**	0.986 (0.0007)	0.966 (0.0004)	**0.804** **(0.0000)**
RESFGB (SMOOTH HINGE)	0.975 (0.0014)	0.952 (0.0023)	**0.989** **(0.0005)**	**0.987** **(0.0010)**	0.965 (0.0005)	**0.804** **(0.0004)**
MULTILAYER PERCEPTRON	0.971 (0.0059)	0.948 (0.0045)	0.988 (0.0010)	0.986 (0.0010)	0.965 (0.0015)	**0.804** **(0.0004)**
SUPPORT VECTOR MACHINE	0.959 (0.0062)	0.948 (0.0023)	0.977 (0.0015)	0.969 (0.0041)	0.824 (0.0059)	0.754 (0.0534)
RANDOM FOREST	0.964 (0.0012)	0.939 (0.0018)	0.980 (0.0005)	0.972 (0.0005)	0.948 (0.0005)	0.802 (0.0004)
GRADIENT BOOSTING	0.964 (0.0011)	0.938 (0.0039)	0.982 (0.0010)	0.981 (0.0004)	**0.972** **(0.0005)**	**0.804** **(0.0005)**

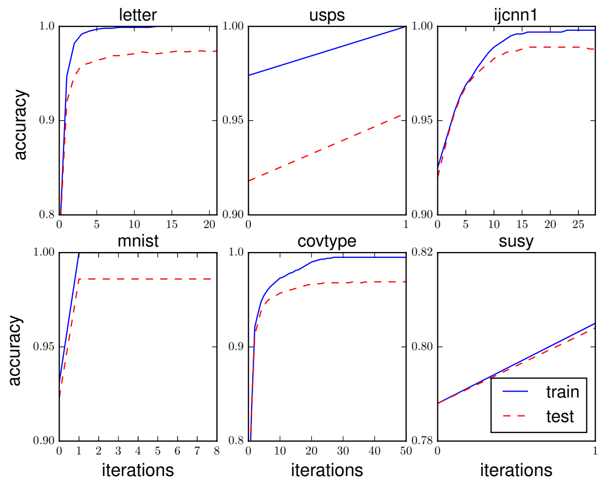

Figure 1. Learning curves for Algorithm 1 with multiclass logistic loss on libsvm datasets showing classification accuracy on training and test sets versus the number of iterations.

We use the following benchmark datasets: letter, usps, ijcnn1, mnist, covtype, and susy. We now explain the experimental procedure. For datasets not providing a fixed test set, we first divide each dataset randomly into two parts: 80% for training and the rest for test. We next divide each training set randomly and use 80% for training and the rest for validation. We perform each method on the training dataset with several hyperparameter settings and choose the best setting on the validation dataset. Finally, we train each model on the entire training dataset using this setting and evaluate it on the test dataset. This procedure is run 5 times.

The mean classification accuracy and the standard deviation are listed in Table 1. The support vector machine is performed using a random Fourier feature (Rahimi & Recht, 2007) with an embedding dimension of 10^3 or 10^4. For multilayer perceptron, we use three, four, or five hidden layers and rectified linear unit as the activation function. The number of hidden units in each layer is set to 100 or 1000. As for random forest, the number of trees is set to 100, 500, or 1000 and the maximum depth is set to 10, 20, or 30. Gradient boosting in Table 1 indicates LightGBM (Ke et al., 2017) with the hyperparameter settings: the maximum number of estimators is 1000, the learning rate is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$, and number of leaves in one tree is chosen from $\{16, 32, \ldots, 1024\}$.

As seen in Table 1, our method shows superior performance over the competitors except for covtype. However, the method that achieves higher accuracy than our method is only LightGBM on covtype. We plot learning curves for one run of Algorithm 1 with logistic loss, which depicts classification accuracies on training and test sets. Note that the number of iterations are determined by classification results on validation sets. This figure shows the efficiency of the proposed method.

7. Conclusion

We have formalized the gradient boosting perspective of ResNet and have proposed new gradient boosting method by leveraging this viewpoint. We have shown two types of generalization bounds: one is by the margin bound and the other is by the sample-splitting technique. These bounds clarify the optimization-generalization tradeoff of the proposed method. Impressive empirical performance of the method has been confirmed on several benchmark datasets. We note that our method can take in convolutional neural networks as feature extractions, but additional efforts will be required to achieve high performance on image datasets. This is one of important topics left for future work.

Acknowledgements

This work was partially supported by MEXT KAKENHI (25730013, 25120012, 26280009, and 15H05707), JST-PRESTO (JPMJPR14E4), and JST-CREST (JPMJCR14D7, JPMJCR1304).

References

Ambrosio, L., Gigli, N., and Savaré, G. *Gradient Flows in Metric Spaces and in the Space of Probability Measures*. Lectures in Mathematics. ETH Zürich. Birkhäuser Basel, 2008.

Balakrishnan, S., Wainwright, M. J., Yu, B., et al. Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1): 77–120, 2017.

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

Bartlett, P. L., Maiorov, V., and Meir, R. Almost linear VC dimension bounds for piecewise polynomial networks. In *Advances in Neural Information Processing Systems 11*, pp. 190–196, 1998.

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *arXiv preprint arXiv:1703.02930*, 2017.

Bengio, Y., Roux, N. L., Vincent, P., Delalleau, O., and Marcotte, P. Convex neural networks. In *Advances in Neural Information Processing Systems 19*, pp. 123–130, 2006.

Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., and Holtham, E. Reversible architectures for arbitrarily deep residual neural networks. *arXiv preprint arXiv:1709.03698*, 2017a.

Chang, B., Meng, L., Haber, E., Tung, F., and Begert, D. Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*, 2017b.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. Adanet: Adaptive structural learning of artificial neural networks. In *International Conference on Machine Learning 34*, pp. 874–883, 2017.

Daneri, S. and Savaré, G. Lecture notes on gradient flows and optimal transport. *arXiv preprint arXiv:1009.3737*, 2010.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, pp. 1189–1232, 2001.

Haber, E., Ruthotto, L., and Holtham, E. Learning across scales-a multiscale method for convolution neural networks. *arXiv preprint arXiv:1703.02009*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Huang, F., Ash, J., Langford, J., and Schapire, R. Learning deep resnet blocks sequentially using boosting theory. *arXiv preprint arXiv:1706.04964*, 2017a.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700—4708, 2017b.

Jastrzebski, S., Arpit, D., Ballas, N., Verma, V., Che, T., and Bengio, Y. Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*, 2017.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, pp. 3149–3157, 2017.

Koltchinskii, V. and Panchenko, D. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1):1–50, 2002.

Littwin, E. and Wolf, L. The loss surface of residual networks: Ensembles and the role of batch normalization. *arXiv preprint arXiv:1611.02525*, 2016.

Lu, Y., Zhong, A., Li, Q., and Dong, B. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*, 2017.

Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, pp. 512–518, 1999.

Moghimi, M., Belongie, S. J., Saberian, M. J., Yang, J., Vasconcelos, N., and Li, L.-J. Boosted convolutional neural networks. In *Proceedings of the British Machine Vision Conference*, pp. 24.1–24.13, 2016.

Neyshabur, B., Tomioka, R., and Srebro, N. Norm-based capacity control in neural networks. In *Proceedings of Conference on Learning Theory 28*, pp. 1376–1401, 2015.

Nitanda, A. and Suzuki, T. Stochastic particle gradient descent for infinite ensembles. *arXiv preprint arXiv:1712.05438*, 2017.

Nitanda, A. and Suzuki, T. Gradient layer: Enhancing the convergence of adversarial training for generative models. *arXiv preprint arXiv:1801.02227*, 2018.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pp. 1177–1184, 2007.

Sonoda, S. and Murata, N. Double continuum limit of deep neural networks. In *ICML Workshop Principled Approaches to Deep Learning*, 2017.

Steinwart, I. Consistency of support vector machines and other regularized kernel classifiers. *IEEE Transactions on Information Theory*, 51(1):128–142, 2005.

van der Vaart, A. and Wellner, J. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer, 1996.

Vapnik, V. and Chervonenkis, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2): 264, 1971.

Veit, A., Wilber, M. J., and Belongie, S. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems 29*, pp. 550–558, 2016.

Wang, Z., Gu, Q., Ning, Y., and Liu, H. High dimensional em algorithm: Statistical optimization and asymptotic normality. In *Advances in Neural Information Processing Systems 28*, pp. 2521–2529, 2015.

Weinan, E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pp. 5987–5995, 2017.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, pp. 87.1—87.12, 2016.

Zhang, T. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, pp. 56–85, 2004.

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