Learning Deep ResNet Blocks Sequentially using Boosting Theory

Furong Huang [1] **Jordan T. Ash** [2] **John Langford** [3] **Robert E. Schapire** [3]

Abstract

We prove a *multi-channel telescoping sum boosting* theory for the ResNet architectures which simultaneously creates a new technique for boosting over features (in contrast to labels) and provides a new algorithm for ResNet-style architectures. Our proposed training algorithm, *BoostResNet*, is particularly suitable in non-differentiable architectures. Our method only requires the relatively inexpensive sequential training of T "shallow ResNets". We prove that the training error decays exponentially with the depth T if the weak module classifiers that we train perform slightly better than some weak baseline. In other words, we propose a weak learning condition and prove a boosting theory for ResNet under the weak learning condition. A generalization error bound based on margin theory is proved and suggests that ResNet could be resistant to overfitting using a network with l_1 norm bounded weights.

1. Introduction

Why do residual neural networks (ResNets) (He et al., 2016) and the related highway networks (Srivastava et al., 2015) work? And if we study closely why they work, can we come up with new understandings of how to train them and how to define working algorithms?

Deep neural networks have elicited breakthrough successes in machine learning, especially in image classification and object recognition (Krizhevsky et al., 2012; Sermanet et al., 2013; Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014) in recent years. As the number of layers increases, the nonlinear network becomes more powerful, deriving richer features from input data. Empirical studies suggest that challenging tasks in image classification (He et al., 2015; Ioffe & Szegedy, 2015; Simonyan & Zisserman, 2014; Szegedy et al., 2015) and object recognition (Girshick, 2015; Girshick et al., 2014; He et al., 2014; Long et al., 2015; Ren et al., 2015) often require "deep" networks, consisting of tens or hundreds of layers. Theoretical analyses have further justified the power of deep networks (Mhaskar & Poggio, 2016) compared to shallow networks.

However, deep neural networks are difficult to train despite their intrinsic representational power. Stochastic gradient descent with back-propagation (BP) (LeCun et al., 1989) and its variants are commonly used to solve the non-convex optimization problems. A major challenge that exists for training both shallow and deep networks is *vanishing* or *exploding gradients* (Bengio et al., 1994; Glorot & Bengio, 2010). Recent works have proposed normalization techniques (Glorot & Bengio, 2010; LeCun et al., 2012; Ioffe & Szegedy, 2015; Saxe et al., 2013) to effectively ease the problem and achieve convergence. In training deep networks, however, a surprising *training performance degradation* is observed (He & Sun, 2015; Srivastava et al., 2015; He et al., 2016): the training performance degrades rapidly with increased network depth after some saturation point. This training performance degradation is representationally surprising as one can easily construct a deep network identical to a shallow network by forcing any part of the deep network to be the same as the shallow network with the remaining layers functioning as identity maps. He et al. (He et al., 2016) presented a *residual network* (ResNet) learning framework to ease the training of networks that are substantially deeper than those used previously. And they explicitly reformulate the layers as learning residual functions with reference to the layer inputs by adding identity loops to the layers. It is shown in (Hardt & Ma, 2016) that identity loops ease the problem of spurious local optima in shallow networks. Srivastava et al. (Srivastava et al., 2015) introduce a novel architecture that enables the optimization of networks with virtually arbitrary depth through the use of a learned gating mechanism for regulating information flow.

Empirical evidence overwhelmingly shows that these deep residual networks are easier to optimize than non-residual ones. Can we develop a theoretical justification for this observation? And does that justification point us towards new algorithms with better characteristics?

[1]Department of Computer Science, University of Maryland; [2]Department of Computer Science, Princeton University; [3]Microsoft Research. Correspondence to: Furong Huang <furongh@cs.umd.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

1.1. Summary of Results

We propose a new framework, *multi-channel telescoping sum boosting* (defined in Section 4), to characterize a feed forward ResNet in Section 3. We show that the top level (final) output of a ResNet can be thought of as a layer-by-layer boosting method (defined in Section 2). Traditional boosting, which ensembles "estimated score functions" or "estimated labels" from weak learners, does not work in the ResNet setting because of two reasons: (1) ResNet is a telescoping sum boosting of weak learners, not a naive (weighted) ensemble; (2) ResNet boosts over "representations", not "estimated labels". We provide the first error bound for telescoping sum boosting over features. Boosting over features and boosting over labels are different. There is no existing work that proves a boosting theory (guaranteed 0 training error) for boosting features. Moreover, the special structure of a ResNet entails more complicated analysis: telescoping sum boosting, which has never been introduced before in the existing literature.

We introduce a learning algorithm (*BoostResNet*) guaranteed to reduce error exponentially as depth increases so long as a weak learning assumption is obeyed. BoostResNet adaptively selects training samples or changes the cost function (Section 4 Theorem 4.2). In Section 4.4, we analyze the generalization error of BoostResNet and provide advice to avoid overfitting. The procedure trains each residual block sequentially, only requiring that each provides a better-than-a-weak-baseline in predicting labels.

BoostResNet requires radically lower computational complexity for training than end-to-end back propagation (*e2eBP*). The number of gradient updates required by BoostResNet is much smaller than e2eBP as discussed in Section 4.3. Memorywise, *BoostResNet* requires only individual layers of the network to be in the graphics processing unit (GPU) while *e2eBP* inevitably keeps all layers in the GPU. For example, in a state-of-the-art deep ResNet, this might reduce the RAM requirements for GPU by a factor of the depth of the network. Similar improvements in computation are observed since each *e2eBP* step involves back propagating through the entire deep network.

Experimentally, we compare *BoostResNet* with *e2eBP* over two types of feed-forward ResNets, *multilayer perceptron residual network* (MLP-ResNet) and *convolutional neural network residual network* (CNN-ResNet), on multiple datasets. *BoostResNet* shows substantial computational performance improvements and accuracy improvement under the MLP-ResNet architecture. Under *CNN-ResNet*, a faster convergence for *BoostResNet* is observed.

One of the hallmarks of our approach is to make an explicit distinction between the classes of the multiclass learning problem and *channels* that are constructed by the learning procedure. A channel here is essentially a scalar value modified by the rounds of boosting so as to implicitly minimize the multiclass error rate. Our *multi-channel telescoping sum boosting* learning framework is not limited to ResNet and can be extended to other, even non-differentiable, nonlinear hypothesis units, such as decision trees or tensor decompositions. Our contribution does not limit to explaining ResNet in the boosting framework, we have also developed a new boosting framework for other relevant tasks that require multi-channel telescoping sum structure.

1.2. Related Works

Training deep neural networks has been an active research area in the past few years. The main optimization challenge lies in the highly non-convex nature of the loss function. There are two main ways to address this optimization problem: one is to select a loss function and network architecture that have better geometric properties (details refer to appendix A.1), and the other is to improve the network's learning procedure (details refer to appendix A.2).

Many authors have previously looked into neural networks and boosting, each in a different way. Bengio et al. (2006) introduce single hidden layer convex neural networks, and propose a gradient boosting algorithm to learn the weights of the linear classifier. The approach has not been generalized to deep networks with more than one hidden layer. Shalev-Shwartz (2014) proposes a selfieBoost algorithm which boosts the accuracy of an entire network. Our algorithm is different as we instead construct ensembles of classifiers. Veit et al. (2016) interpret residual networks as a collection of many paths of differing length. Their empirical study shows that residual networks avoid the vanishing gradient problem by introducing short paths which can carry gradient throughout the extent of very deep networks.

Comparison with AdaNet The authors of AdaNet (Cortes et al., 2016) consider ensembles of neural layers with a boosting-style algorithm and provide a method for structural learning of neural networks by optimizing over the generalization bound, which consists of the training error and the complexity of the AdaNet architecture. AdaNet uses the traditional boosting framework where weak classifiers are being boosted. Therefore, to obtain low training error guarantee, AdaNet maps the feature vectors (hidden layer representations) to a classifier space and boosts the weak classifiers. In AdaNet, features (representations) from each lower layer have to be fed into a classifier (in other words, be transferred to score function in the label space). This is because AdaNet uses traditional boosting, which ensembles score functions or labels. As a result, the top classifier in AdaNet has to be connected to all lower layers, making the structure bushy. Therefore AdaNet chooses its own structure during learning, and its boosting theory does not

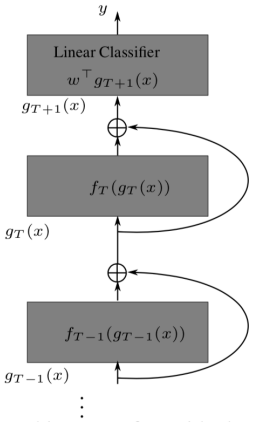


Figure 1: The architecture of a residual network (ResNet).

necessarily work for a ResNet structure. Our BoostResNet, instead, boosts features (representations) over multiple channels, and thus produces a less "bushy" architecture. We are able to boost features by developing this new "telescoping-sum boosting" framework, one of our main contributions. We come up with the new weak learning condition for the telescoping-sum boosting framework. The algorithm is also very different from AdaNet and is explained in details in section 3 and 4.

BoostResNet focuses on a ResNet architecture, provides a new training algorithm for ResNet, and proves a training error guarantee for deep ResNet architecture. A ResNet-style architecture is a special case of AdaNet, so AdaNet generalization guarantee applies here and our generalization analysis is built upon their work.

2. Preliminaries

A *residual neural network* (ResNet) is composed of stacked entities referred to as residual blocks. Each residual block consists of a neural network module and an identity loop (shortcut). Commonly used modules include MLP and CNN. Throughout this paper, we consider training and test examples generated i.i.d. from some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the label space. We denote by $S = ((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m))$ a training set of m examples drawn according to $\mathcal{D}^m$.

A Residual Block of ResNet ResNet consists of residual blocks. Each residual block contains a module and an identity loop. Let each module map its input $\tilde{x}$ to $f_t(\tilde{x})$ where t denotes the level of the modules. Each module f_t is a nonlinear unit with n channels, i.e., $f_t(\cdot) \in \mathbb{R}^n$. In *multilayer perceptron residual network* (MLP-ResNet), f_t is a shallow MLP, for instance, $f_t(\tilde{x}) = \tilde{V}_t^\top \sigma(\tilde{W}_t^\top \tilde{x})$ where $\tilde{W}_t \in \mathbb{R}^{n \times k}$, $\tilde{V}_t \in \mathbb{R}^{k \times n}$ and σ is a nonlinear operator such as sigmoidal function or relu function. Similarly, in *convolutional neural network residual network* (CNN-ResNet), $f_t(\cdot)$ represents the t-th convolutional module. Then the t-th residual block outputs $g_{t+1}(x)$

$$g_{t+1}(x) = f_t(g_t(x)) + g_t(x), \tag{1}$$

where x is the input fed to the ResNet. See Figure 1 for an illustration of a ResNet, which consists of stacked residual blocks (each residual block contains a nonlinear module and an identity loop).

Output of ResNet Due to the recursive relation specified in Equation (1), the output of the T-th residual block is equal to the summation over lower module outputs, i.e., $g_{T+1}(x) = \sum_{t=0}^{T} f_t(g_t(x))$, where $g_0(x) = 0$ and $f_0(g_0(x)) = x$. For binary classification tasks, the final output of a ResNet given input x is rendered after a linear classifier $\mathbf{w} \in \mathbb{R}^n$ on representation $g_{T+1}(x)$ (In the multiclass setting, let C be the number of classes; the linear classifier $W \in \mathbb{R}^{n \times C}$ is a matrix instead of a vector.):

$$\widehat{y} = \tilde{\sigma}\left(F(x)\right) = \tilde{\sigma}(\mathbf{w}^\top g_{T+1}(x)) = \tilde{\sigma}\left(\mathbf{w}^\top \sum_{t=0}^{T} f_t(g_t(x))\right) \tag{2}$$

where $F(x) = \mathbf{w}^\top g_{T+1}(x)$ and $\tilde{\sigma}(\cdot)$ denotes a map from classifier outputs (scores) to labels. For instance $\tilde{\sigma}(z) = \text{sign}(z)$ for binary classification ($\tilde{\sigma}(z) = \arg\max_i z_i$ for multiclass classification). The parameters of a depth-T ResNet are $\{\mathbf{w}, \{f_t(\cdot), \forall t \in T\}\}$. A ResNet training involves training the classifier $\mathbf{w}$ and the weights of modules $f_t(\cdot)$ $\forall t \in [T]$ when training examples $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ are available.

Boosting Boosting (Freund & Schapire, 1995) assumes the availability of *a weak learning algorithm* which, given labeled training examples, produces a *weak classifier* (a.k.a. *base classifier*). The goal of boosting is to improve the performance of the weak learning algorithm. The key idea behind boosting is to choose training sets for the weak classifier in such a fashion as to force it to infer something new about the data each time it is called. The weak learning algorithm will finally combine many weak classifiers into a single *strong classifier* whose prediction power is strong.

From empirical experience, ResNet remedies the problem of training error degradation (instability of solving non-convex optimization problem using SGD) in deeper neural networks. We are curious about whether there is a theoretical justification that identity loops help in training. More importantly, we are interested in proposing a new algorithm that avoids end-to-end back-propagation (*e2eBP*) through the deep network and thus is immune to the instability of SGD for non-convex optimization of deep neural networks.

3. ResNet in Telescoping Sum Boosting Framework

As we recall from Equation (2), ResNet indeed has a similar form as the strong classifier in boosting. The key difference is that boosting is an ensemble of estimated hypotheses whereas ResNet is an ensemble of estimated feature representations $\sum_{t=0}^{T} f_t(g_t(x))$. To solve this problem, we introduce an auxiliary linear classifier $\mathbf{w}_t$ on top of each residual block to construct a *hypothesis module*. Formally, a *hypothesis module* is defined as

$$o_t(x) \stackrel{\text{def}}{=} \mathbf{w}_t^\top g_t(x) \in \mathbb{R} \tag{3}$$

in the binary classification setting. Therefore $o_{t+1}(x) = \mathbf{w}_{t+1}^\top [f_t(g_t(x)) + g_t(x)]$ as $g_{t+1}(x) = f_t(g_t(x)) + g_t(x)$. We emphasize that given $g_t(x)$, we only need to train f_t and $\mathbf{w}_{t+1}$ to train $o_{t+1}(x)$. In other words, we feed the output of previous residual block ($g_t(x)$) to the current module and train the weights of current module $f_t(\cdot)$ and the auxiliary classifier $\mathbf{w}_{t+1}$.

Now the input, $g_{t+1}(x)$, of the $t+1$-th residual block is the output, $f_t(g_t(x)) + g_t(x)$, of the t-th residual block. As a result, $o_t(x) = \sum_{t'=0}^{t-1} \mathbf{w}_t^\top f_{t'}(g_{t'}(x))$. In other words, the auxiliary linear classifier is common for all modules underneath. It would not be realistic to assume a common auxiliary linear classifier, as such an assumption prevents us from training the T hypothesis module sequentially. We design a ***weak module classifier*** using the idea of telescoping sum as follows.

Definition 3.1. *A weak module classifier is defined as*

$$h_t(x) \stackrel{\text{def}}{=} \alpha_{t+1} o_{t+1}(x) - \alpha_t o_t(x) \tag{4}$$

where $o_t(x) \stackrel{\text{def}}{=} \mathbf{w}_t^\top g_t(x)$ is a hypothesis module, and α_t is a scalar. We call it a "telescoping sum boosting" framework if the weak learners are restricted to the form of the weak module classifier.

ResNet: Ensemble of Weak Module Classifiers Recall that the T-th residual block of a ResNet outputs $g_{T+1}(x)$, which is fed to the top/final linear classifier for the final classification. We show that an ensemble of the weak module classifiers is equivalent to a ResNet's final output. We state it formally in Lemma 3.2. For purposes of exposition, we will call $F(x)$ the output of ResNet although a $\tilde{\sigma}$ function is applied on top of $F(x)$, mapping the output to the label space $\mathcal{Y}$.

Lemma 3.2. *Let the input $g_t(x)$ of the t-th module be the output of the previous module, i.e., $g_{t+1}(x) = f_t(g_t(x)) + g_t(x)$. Then the summation of T weak module classifiers divided by α_{T+1} is identical to the output, $F(x)$, of the depth-T ResNet,*

$$F(x) = \mathbf{w}^\top g_{T+1}(x) \equiv \frac{1}{\alpha_{T+1}} \sum_{t=0}^{T} h_t(x), \tag{5}$$

where the weak module classifier $h_t(x)$ is defined in Equation (4)*.*

See Appendix B for the proof. Overall, our proposed ensemble of weak module classifiers is a new framework that allows for sequential training of ResNet. Note that traditional boosting algorithm results do not apply here. We now analyze our telescoping sum boosting framework in Section 4. Our analysis applies to both binary and multiclass, but we will focus on the binary class for simplicity in the main text and defer the multiclass analysis to the Appendix F.

4. Telescoping Sum Boosting for Binary Classification

Below, we propose a learning algorithm whose training error decays exponentially with the number of weak module classifiers T under a weak learning condition. We restrict to bounded hypothesis modules, i.e., $|o_t(x)| \leq 1$.

4.1. Weak Learning Condition

The weak module classifier involves the difference between (scaled version of) $o_{t+1}(x)$ and $o_t(x)$. Let $\tilde{\gamma}_t \stackrel{\text{def}}{=} \mathbb{E}_{i \sim D_{t-1}}[y_i o_t(x_i)] > 0$ be the *edge* of the hypothesis module $o_t(x)$, where D_{t-1} is the weight of the examples. As the hypothesis module $o_t(x)$ is bounded by 1, we obtain $|\tilde{\gamma}_t| \leq 1$. So $\tilde{\gamma}_t$ characterizes the performance of the hypothesis module $o_t(x)$. A natural requirement would be that $o_{t+1}(x)$ improves slightly upon $o_t(x)$, and thus $\tilde{\gamma}_{t+1} - \tilde{\gamma}_t \geq \gamma' > 0$ could serve as a weak learning condition. However this weak learning condition is too strong: even when current hypothesis module is performing almost ideally ($\tilde{\gamma}_t$ is close to 1), we still seek a hypothesis module which performs consistently better than the previous one by γ'. Instead, we consider a much weaker learning condition, inspired by training error analysis, as follows.

Definition 4.1 (γ-Weak Learning Condition)**.** *A weak module classifier $h_t(x) = \alpha_{t+1} o_{t+1} - \alpha_t o_t$ satisfies the γ-weak learning condition if $\frac{\tilde{\gamma}_{t+1}^2 - \tilde{\gamma}_t^2}{1 - \tilde{\gamma}_t^2} \geq \gamma^2 > 0$ and the covariance between $\exp(-y o_{t+1}(x))$ and $\exp(y o_t(x))$ is non-positive.*

The weak learning condition is motivated by the learning theory and it is met in practice (refer to Figure 4).

Interpretation of weak learning condition For each weak module classifier $h_t(x)$, $\gamma_t \stackrel{\text{def}}{=} \sqrt{\frac{\tilde{\gamma}_{t+1}^2 - \tilde{\gamma}_t^2}{1 - \tilde{\gamma}_t^2}}$ characterizes the normalized improvement of the correlation between the true

Algorithm 1 BoostResNet: telescoping sum boosting for binary-class classification

Input: m labeled samples $[(x_i, y_i)]_m$ where $y_i \in \{-1, +1\}$ and a threshold γ
Output: $\{f_t(\cdot), \forall t\}$ and $\mathbf{w}_{T+1}$ ▷ Discard $\mathbf{w}_{t+1}, \forall t \neq T$

1: Initialize $t \leftarrow 0$, $\tilde{\gamma}_0 \leftarrow 0$, $\alpha_0 \leftarrow 0$, $o_0(x) \leftarrow 0$
2: Initialize sample weights at round 0: $D_0(i) \leftarrow 1/m, \forall i \in [m]$
3: **while** $\gamma_t > \gamma$ **do**
4: $f_t(\cdot), \alpha_{t+1}, \mathbf{w}_{t+1}, o_{t+1}(x) \leftarrow$ Algorithm 2$(g_t(x), D_t, o_t(x), \alpha_t)$
5: Compute $\gamma_t \leftarrow \sqrt{\frac{\tilde{\gamma}_{t+1}^2 - \tilde{\gamma}_t^2}{1 - \tilde{\gamma}_t^2}}$ ▷ where $\tilde{\gamma}_{t+1} \leftarrow \mathbb{E}_{i \sim D_t}[y_i o_{t+1}(x_i)]$
6: Update $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-y_i h_t(x_i))}{\sum_{i=1}^{m} D_t(i) \exp[-y_i h_t(x_i)]}$ ▷ where $h_t(x) = \alpha_{t+1} o_{t+1}(x) - \alpha_t o_t(x)$
7: $t \leftarrow t + 1$
8: **end while**
9: $T \leftarrow t - 1$

Algorithm 2 BoostResNet: oracle implementation for training a ResNet block

Input: $g_t(x)$,D_t,$o_t(x)$ and α_t
Output: $f_t(\cdot)$, α_{t+1}, $\mathbf{w}_{t+1}$ and $o_{t+1}(x)$

1: $(f_t, \alpha_{t+1}, \mathbf{w}_{t+1}) \leftarrow \arg\min_{(f,\alpha,\mathbf{v})} \sum_{i=1}^{m} D_t(i) \exp\left(-y_i \alpha \mathbf{v}^\top [f(g_t(x_i)) + g_t(x_i)] + y_i \alpha_t o_t(x_i)\right)$
2: $o_{t+1}(x) \leftarrow \mathbf{w}_{t+1}^\top [f_t(g_t(x)) + g_t(x)]$

labels y and the hypothesis modules $o_{t+1}(x)$ over the correlation between the true labels y and the hypothesis modules $o_t(x)$. The condition specified in Definition 4.1 is mild as it requires the hypothesis module $o_{t+1}(x)$ to perform only slightly better than the previous hypothesis module $o_t(x)$. In residual network, since $o_{t+1}(x)$ represents a depth-$(t+1)$ residual network which is a deeper counterpart of the depth-t residual network $o_t(x)$, it is natural to assume that the deeper residual network improves slightly upon the shallower residual network. When $\tilde{\gamma}_t$ is close to 1, $\tilde{\gamma}_{t+1}^2$ only needs to be slightly better than $\tilde{\gamma}_t^2$ as the denominator $1 - \tilde{\gamma}_t^2$ is small. The assumption of the covariance between $\exp(-y o_{t+1}(x))$ and $\exp(y o_t(x))$ being non-positive is suggesting that the weak module classifiers should not be adversarial, which may be a reasonable assumption for ResNet.

4.2. BoostResNet

We now propose a novel training algorithm for telescoping sum boosting under binary-class classification as in Algorithm 1. In particular, we introduce a training procedure for deep ResNet in Algorithm 1 & 2, **BoostResNet**, which only requires sequential training of shallow ResNets.

The training algorithm is a *module-by-module* procedure following a *bottom-up* fashion as the outputs of the t-th module $g_{t+1}(x)$ are fed as the training examples to the next $t+1$-th module. Each of the shallow ResNet $f_t(g_t(x)) + g_t(x)$ is combined with an auxiliary linear classifier $\mathbf{w}_{t+1}$ to form a hypothesis module $o_{t+1}(x)$. The weights of the ResNet are trained on these shallow ResNets. The telescoping sum construction is the key for successful interpretation of ResNet as ensembles of weak module classifiers. The innovative introduction of the auxiliary linear classifiers ($\mathbf{w}_{t+1}$) is the key solution for successful multi-channel representation boosting with theoretical guarantees. Auxiliary linear classifiers are only used to guide training, and they are not included in the model (proved in Lemma 3.2). This is the fundamental difference between BoostResNet and AdaNet. AdaNet (Cortes et al., 2016) maps the feature vectors (hidden layer representations) to a classifier space and boosts the weak classifiers. Our framework is a multi-channel representation (or information) boosting rather than a traditional classifier boosting. Traditional boosting theory does not apply in our setting.

Theorem 4.2. [***Training error bound***] *The training error of a T-module telescoping sum boosting framework using Algorithms 1 and 2 decays exponentially with the number of modules T,*

$$\Pr_{i \sim S}\left(\tilde{\sigma}\left(\sum_t h_t(x_i)\right) \neq y_i\right) \leq e^{-\frac{1}{2}T\gamma^2}$$

if $\forall t \in [T]$ the weak module classifier $h_t(x)$ satisfies the γ-weak learning condition defined in Definition 4.1.

The training error of Algorithms 1 and 2 is guaranteed to decay exponentially with the ResNet depth even when each hypothesis module $o_{t+1}(x)$ performs slightly better than its previous hypothesis module $o_t(x)$ (i.e., $\gamma > 0$). Refer to Appendix F for the algorithm and theoretical guarantees for multiclass classification.

4.3. Oracle Implementation for ResNet

In Algorithm 2, the implementation of the oracle at line 1 is equivalent to

$$(f_t, \alpha_{t+1}, \mathbf{w}_{t+1}) = \arg\min_{(f,\alpha,\mathbf{v})} \frac{1}{m}\sum_{i=1}^{m} \exp\left(-y_i \alpha \mathbf{v}^\top [f(g_t(x_i)) + g_t(x_i)]\right) \quad (6)$$

The minimization problem over f corresponds to finding the weights of the t-th nonlinear module of the residual network. Auxiliary classifier $\mathbf{w}_{t+1}$ is used to help solve this minimization problem with the guidance of training labels y_i. However, the final neural network model includes none of the auxiliary classifiers, and still follows a standard ResNet structure (proved in Lemma 3.2). In practice, there are various ways to implement Equation (6). For instance, Janzamin et. al. (Janzamin et al., 2015) propose a tensor decomposition technique which decomposes a tensor formed by some transformation of the features x combined with labels y and recovers the weights of a one-hidden layer neural network with guarantees. One can also use back-propagation as numerous works have shown that gradient based training are relatively stable on shallow networks with identity loops (Hardt & Ma, 2016; He et al., 2016).

Computational & Memory Efficiency *BoostResNet* training is memory efficient as the training process only requires parameters of two consecutive residual blocks to be in memory. Given that the limited GPU memory being one of the main bottlenecks for computational efficiency, *BoostResNet* requires significantly less training time than *e2eBP* in deep networks as a result of reduced communication overhead and the speed-up in shallow gradient forwarding and back-propagation. Let M_1 be the memory required for one module, and M_2 be the memory required for one linear classifier, the memory consumption is $M_1 + M_2$ by *BoostResNet* and $M_1 T + M_2$ by *e2eBP*. Let the flops needed for gradient update over one module and one linear classifier be C_1 and C_2 respectively, the computation cost is $C_1 + C_2$ by *BoostResNet* and $C_1 T + C_2$ by *e2eBP*.

4.4. Generalization Error Analysis

In this section, we analyze the generalization error to understand the possibility of overfitting under Algorithm 1. The strong classifier or the ResNet is $F(x) = \frac{\sum_t h_t(x)}{\alpha_{T+1}}$. Now we define the *margin* for example (x, y) as $yF(x)$. For simplicity, we consider MLP-ResNet with n multiple channels and assume that the weight vector connecting a neuron at layer t with its preceding layer neurons is l_1 norm bounded by $\Lambda_{t,t-1}$. Recall that there exists a linear classifier $\mathbf{w}$ on top, and we restrict to l_1 norm bounded classifiers, i.e., $\|\mathbf{w}\|_1 \leq C_0 < \infty$. The expected training examples are l_∞ norm bounded $r_\infty \stackrel{\text{def}}{=} \mathbb{E}_{S\sim\mathcal{D}}\left[\max_{i\in[m]} \|x_i\|_\infty\right] < \infty$. We introduce Corollary 4.3 which follows directly from Lemma 2 of (Cortes et al., 2016).

Corollary 4.3. *(Cortes et al., 2016) Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ and $\mathcal{S}$ be a sample of m examples chosen independently at random according to $\mathcal{D}$. With probability at least $1-\delta$, for $\theta > 0$, the strong classifier $F(x)$ (ResNet) satisfies that*

$$\Pr_{\mathcal{D}}(yF(x) \leq 0) \leq \Pr_{S}(yF(x) \leq \theta) + \frac{4C_0 r_\infty}{\theta}\sqrt{\frac{\log(2n)}{2m}}\sum_{t=0}^{T}\Lambda_t + \frac{2}{\theta}\sqrt{\frac{\log T}{m}} + \beta(\theta, m, T, \delta) \quad (7)$$

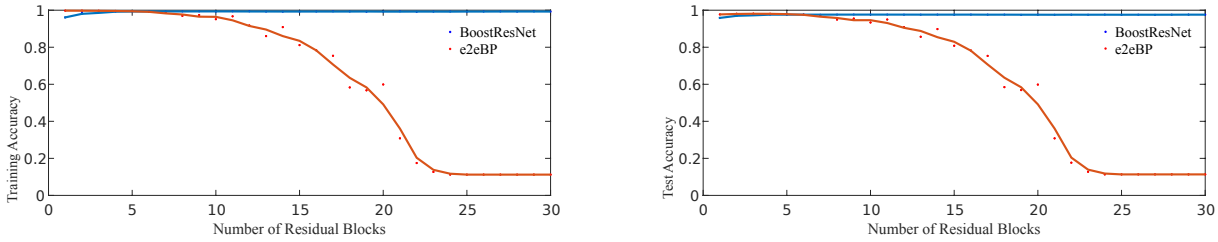
where $\Lambda_t \stackrel{\text{def}}{=} \prod_{t'=0}^{t} 2\Lambda_{t',t'-1}$ *and* $\beta(\theta, m, T, \delta) \stackrel{\text{def}}{=} \sqrt{\left\lceil \frac{4}{\theta^2}\log\left(\frac{\theta^2 m}{\log T}\right)\right\rceil \frac{\log T}{m} + \frac{\log\frac{2}{\delta}}{2m}}$.

From Corollary 4.3, we obtain a generalization error bound in terms of margin bound $\Pr_S(yF(x) \leq \theta)$ and network complexity $\frac{4C_0 r_\infty}{\theta}\sqrt{\frac{\log(2n)}{2m}}\sum_{t=0}^{T}\Lambda_t + \frac{2}{\theta}\sqrt{\frac{\log T}{m}} + \beta(\theta, m, T, \delta)$. Larger margin bound (larger θ) contributes positively to generalization accuracy, and l_1 norm bounded weights (smaller $\sum_{t=0}^{T}\Lambda_t$) are beneficial to control network complexity and to avoid overfitting. The dominant term in the network complexity is $\frac{4C_0 r_\infty}{\theta}\sqrt{\frac{\log(2n)}{2m}}\sum_{t=0}^{T}\Lambda_t$ which scales as least linearly with the depth T. See Appendix D for the proof.

This corollary suggests that stronger weak module classifiers which produce higher accuracy predictions and larger edges, will yield larger margins and suffer less from overfitting. The larger the value of θ, the smaller the term $\frac{4C_0 r_\infty}{\theta}\sqrt{\frac{\log(2n)}{2m}}\sum_{t=0}^{T}\Lambda_t + \frac{2}{\theta}\sqrt{\frac{\log T}{m}} + \beta(\theta, m, T, \delta)$ is. With larger edges on the training set and when $\tilde{\gamma}_{T+1} < 1$, we are able to choose larger values of θ while keeping the error term zero or close to zero.

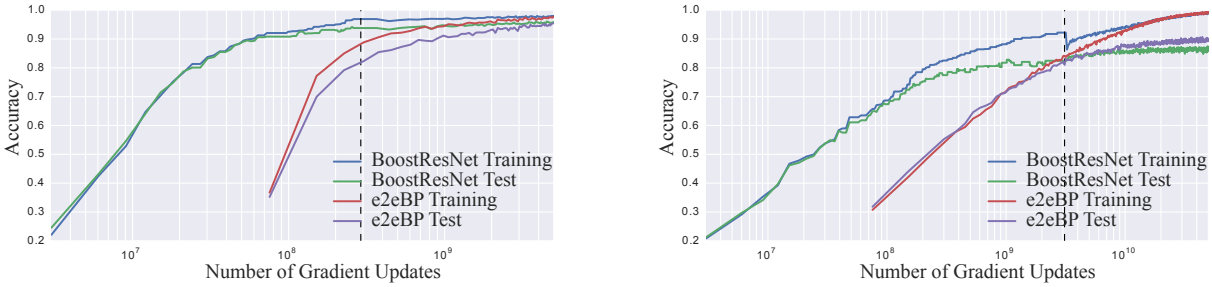
5. Experiments

We compare our proposed *BoostResNet* algorithm with *e2eBP* training a ResNet on the MNIST (LeCun et al., 1998), street view house numbers (SVHN) (Netzer et al., 2011), and CIFAR-10 (Krizhevsky & Hinton, 2009) benchmark datasets. Two different types of architectures are tested: a ResNet where each module is a fully-connected multi-layer perceptron (MLP-ResNet) and a more common, convolutional neural network residual network (CNN-ResNet). In each experiment the architecture of both algorithms is identical, and they are both initialized with the same random seed. As a baseline, we also experiment with standard boosting (AdaBoost.MM (Mukherjee & Schapire, 2013)) of convolutional modules for SVHN and CIFAR-10 datasets. Our experiments are programmed in the Torch deep learning framework for Lua and executed on NVIDIA Tesla P100 GPUs. All models are trained using the Adam variant of SGD (Kingma & Ba, 2014).



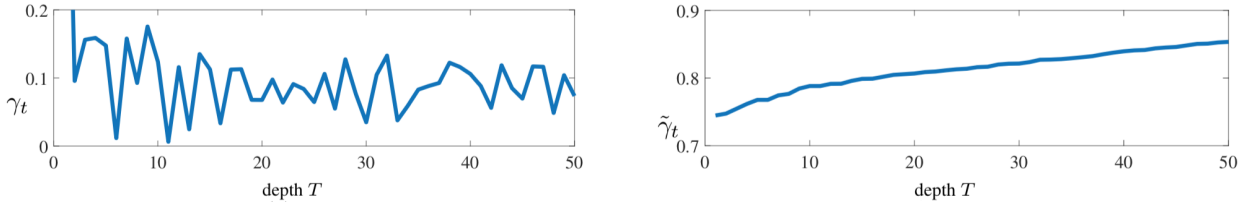
(a) depth vs training accuracy **(b)** depth vs test accuracy

Figure 2: Comparison of *BoostResNet* (ours, blue) and *e2eBP* (baseline, red) on **multilayer perceptron residual network** on MNIST dataset.



(a) SVHN **(b)** CIFAR-10

Figure 3: Convergence performance comparison between *e2eBP* and *BoostResNet* on **convolutional neural network residual network** on the SVHN and CIFAR-10 dataset. The vertical dotted line shows when BoostResNet training stopped, and we began refining the network with standard e2eBP training.



(a) γ_t **(b)** $\tilde{\gamma}_t$

Figure 4: Visualization of edge γ_t and edge for each residual block $\tilde{\gamma}_t$. The x-axis represents depth, and the y-axis represents γ_t or $\tilde{\gamma}_t$ values. The plots are for a convolutional network composed of 50 residual blocks and trained on the SVHN dataset.

Hyperparameters are selected via random search for highest accuracy on a validation set. They are specified in Appendix H. In BoostResNet, the most important hyperparameters, according to our experiments, are those that govern when the algorithm stops training the current module and begins training its successor.

MLP-ResNet on MNIST The MNIST database (LeCun et al., 1998) of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The data contains ten classes. We test the performance of *BoostResNet* on MLP-ResNet using MNIST dataset, and compare it with *e2eBP* baseline. Each residual block is composed of an MLP with a single, 1024-dimensional hidden layer. The training and test error between *BoostResNet* and *e2eBP* is in Figure 2 as a function of depth. Surprisingly, we find that training error degrades for *e2eBP*, although the ResNet's identity loop is supposed to alleviate this problem. Our proposed sequential training procedure, *BoostResNet*, relieves gradient instability issues, and continues to perform well as depth increases.

CNN-ResNet on SVHN SVHN (Netzer et al., 2011) is a real-world image dataset, obtained from house numbers in Google Street View images. The dataset contains over 600,000 training images, and about 20,000 test images. We fit a 50-layer, 25-residual-block CNN-ResNet using both *BoostResNet* and *e2eBP* (figure 3a). Each residual block is composed of a CNN using 15 3 $\times$ 3 filters. We refine the result of *BoostResNet* by initializing the weights using the result of *BoostResNet* and run end-to-end back propagation (e2eBP). From figure 3a, our *BoostResNet* converges much faster (requires much fewer gradient updates) than *e2eBP*. The test accuracy of *BoostResNet* is comparable with *e2eBP*.

CNN-ResNet on CIFAR-10 The CIFAR-10 dataset is a benchmark dataset composed of 10 classes of small images,

TRAINING:	BOOSTRESNET	E2EBP	BOOSTRESNET+E2EBP	E2EBP	ADABOOST
NGU	3×10^8	3×10^8	2×10^{10}	2×10^{10}	1.5×10^9
TRAIN	96.9%	85%	98.8%	98.8%	95.6%
TEST	93.8%	83%	96.8%	96.8%	92.3%

Table 1: Accuracies of SVHN task. NGU is the number of gradient updates taken by the algorithm in training.

TRAINING:	BOOSTRESNET	E2EBP	BOOSTRESNET+E2EBP	E2EBP	ADABOOST
NGU	3×10^9	3×10^9	1×10^{11}	1×10^{11}	1.5×10^{10}
TRAIN	92.1%	82%	99.6%	99.7%	91.3%
TEST	82.1%	80%	88.1%	90.0%	87.1%

Table 2: Accuracies of CIFAR-10 task. NGU is the number of gradient updates taken by the algorithm in training.

such as animals and vehicles. It consists of 50,000 training images and 10,000 test images. We again fit a 50-layer, 25-residual-block CNN-ResNet using both *BoostResNet* and *e2eBP* (figure 3b). *BoostResNet* training converges to the optimal solution faster than *e2eBP*. Unlike in the previous two datasets, the efficiency of BoostResNet comes at a cost when training with CIFAR-10. We find that the test accuracy of the *e2eBP* refined *BoostResNet* to be slightly lower than that produced by *e2eBP*.

Weak Learning Condition Check The weak learning condition (Definition 4.1) inspired by learning theory is checked in Figure 4. The required better than random guessing edge γ_t is depicted in Figure 4a, it is always greater than 0 and our weak learning condition is thus non-vacuous. In Figure 4b, the representations we learned using *BoostResNet* is increasingly better (for this classification task) as the depth increases.

Comparison of BoostResNet, e2eBP and AdaBoost Besides e2eBP, we also experiment with standard boosting (AdaBoost.MM (Mukherjee & Schapire, 2013)), as another baseline, of convolutional modules. In this experiment, each weak learner is a residual block of the ResNet, paired with a classification layer. We do 25 rounds of AdaBoost.MM and train each weak learner to convergence. Table 1 and table 2 exhibit a comparison of BoostResNet, e2eBP and AdaBoost performance on SVHN and CIFAR-10 dataset respectively.

On SVHN dataset, the advantage of BoostResNet over e2eBP is obvious. Using 3×10^8 number of gradient updates, BoostResNet achieves 93.8% test accuracy whereas e2eBP obtains a test accuracy of 83%. The training and test accuracies of SVHN are listed in Table 1. BoostResNet training allows the model to train much faster than end-to-end training, and still achieves the same test accuracy when refined with *e2eBP*. To list the hyperparameters we use in our BoostResNet training after searching over candidate hyperparamters, we optimize learning rate to be 0.004 with a 9×10^{-5} learning rate decay. The gamma threshold is optimized to be 0.001 and the initial gamma value on SVHN is 0.75. On CIFAR-10 dataset, the main advantage of BoostResNet over e2eBP is the speed of training. BoostResNet refined with e2eBP obtains comparable results with e2eBP. This is because we are using a suboptimal architecture of ResNet which overfits the CIFAR-10 dataset. AdaBoost, on the other hand, is known to be resistant to overfitting. In BoostResNet training, we optimize learning rate to be 0.014 with a 3.46×10^{-5} learning rate decay. The gamma threshold is optimized to be 0.007 and the initial gamma value on CIFAR-10 is 0.93. We find that a standard ResNet, to its credit, is quite robust to hyperparameters, namely learning rate and learning rate decay, provided that we use an optimization procedure that automatically modulates these values.

6. Conclusions and Future Works

Our proposed BoostResNet algorithm achieves exponentially decaying (with the depth T) training error under the weak learning condition. BoostResNet is much more computationally efficient compared to end-to-end back-propagation in deep ResNet. More importantly, the memory required by BoostResNet is trivial compared to end-to-end back-propagation. It is particularly beneficial given the limited GPU memory and large network depth. Our learning framework is natural for non-differentiable data. For instance, our learning framework is amenable to take weak learning oracles using tensor decomposition techniques. Tensor decomposition, a spectral learning framework with theoretical guarantees, is applied to learning one layer MLP in (Janzamin et al., 2015). We plan to extend our learning framework to non-differentiable data using general weak learning oracles.

References

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Bengio, Y., Le Roux, N., Vincent, P., Delalleau, O., and Marcotte, P. Convex neural networks. *Advances in neural information processing systems*, 18:123, 2006.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. Adanet: Adaptive structural learning of artificial neural networks. *arXiv preprint arXiv:1607.01097*, 2016.

Freund, Y. and Schapire, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pp. 23–37. Springer, 1995.

Girshick, R. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pp. 249–256, 2010.

Hardt, M. and Ma, T. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

He, K. and Sun, J. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5353–5360, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pp. 346–361. Springer, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Janzamin, M., Sedghi, H., and Anandkumar, A. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*, 2015.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

Mhaskar, H. N. and Poggio, T. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

Mukherjee, I. and Schapire, R. E. A theory of multiclass boosting. *Journal of Machine Learning Research*, 14 (Feb):437–497, 2013.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 5, 2011.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

Shalev-Shwartz, S. Selfieboost: A boosting algorithm for deep learning. *arXiv preprint arXiv:1411.3436*, 2014.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

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

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.