BPGrad: Towards Global Optimality in Deep Learning via Branch and Pruning

Ziming Zhang* †
Mitsubishi Electric Research Laboratories
201 Broadway, Cambridge, MA 02139-1955
zzhang@merl.com

Yuanwei Wu,* Guanghui Wang
EECS, The University of Kansas
1450 Jayhawk Blvd., Lawrence, KS 66045
{y262w558, ghwang}@ku.edu

Abstract

Understanding the global optimality in deep learning (DL) has been attracting more and more attention recently. Conventional DL solvers, however, have not been developed intentionally to seek for such global optimality. In this paper we propose a novel approximation algorithm, BPGrad*, towards optimizing deep models globally via branch and pruning. Our BPGrad algorithm is based on the assumption of Lipschitz continuity in DL, and as a result it can adaptively determine the step size for current gradient given the history of previous updates, wherein theoretically no smaller steps can achieve the global optimality. We prove that, by repeating such branch-and-pruning procedure, we can locate the global optimality within finite iterations. Empirically an efficient solver based on BPGrad for DL is proposed as well, and it outperforms conventional DL solvers such as Adagrad, Adadelta, RMSProp, and Adam in the tasks of object recognition, detection, and segmentation.*

1. Introduction

Deep learning (DL) has been demonstrated successfully in many different research areas such as image classification [20], speech recognition [16] and natural language processing [32]. In general, its empirical success stems mainly from better network architectures [15], larger mount of training data [6], and better learning algorithms [12].

However, theoretical understanding of DL for its success still remains elusive. Very recently researchers start to understand DL from the perspective of optimization such as the optimality of learned models [13, 14, 36]. It has been proved that under certain (very restrictive) conditions the critical points learned for the deep models actually achieve global optimality, even though the optimization in deep learning is highly nonconvex. These theoretical results may partially explain why such deep models work well in practice.

*Joint first authors for the paper.
†Corresponding author.

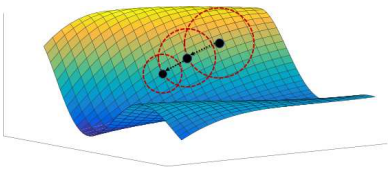


Figure 1. Illustration of how BPGrad works, where each black dot denotes the solution at each iterations (*i.e.* branch), directed dotted lines denote the current gradients, and red dotted circles denote the regions wherein there should be no solutions achieving global optimality (*i.e.* pruning). BPGrad can automatically estimate the scales of these regions based on the function evaluation of solutions and the Lipschitz continuity assumption.

Global optimality is always desirable and preferred in optimization. Locating global optimality in deep learning, however, is extremely challenging due to its high non-convexity, and thus no conventional DL solvers, *e.g.* stochastic gradient descent (SGD) [2], Adagrad [7], Adadelta [37], RMSProp [33] and Adam [18], is intentionally developed for this purpose, to our best knowledge. Alternatively different regularization techniques are applied to smooth the objective functions in DL so that the solvers can converge to some geometrically wider and flatter regions in the parameter space where good model solutions may exist [39, 4, 40]. But these solutions may not necessarily be the global optimum.

Inspired by the techniques in global optimization of nonconvex functions, we propose a novel approximation algorithm, *BPGrad*, which has the ability of locating global optimality in DL via branch and pruning (BP). BP [29] is a well-known algorithm developed for searching for global solutions for nonconvex optimization problems. Its basic idea is to effectively and gradually shrink the gap between the lower and upper bounds of global optimum by efficiently branching and pruning the parameter space. Fig. 1 illustrates the optimization procedure in BPGrad.

In order to branch and prune the space we assume that the objective functions in DL are Lipschitz continuous [8], or can be approximated by Lipschitz functions. This is motivated by the facts that (1) Lipschitz continuity provides a natural way to estimate the lower and upper bounds of the global optimum (see Sec. 2.3.1) used in BP, and (2) it can also serve as regularization, if needed, to smoothen the objective functions so that the returned solutions can generalize well.

In Fig. 2 we illustrate the functionality of Lipschitz continuity as regularization, where the noisy narrower but deeper valley is smoothed out, while the wider but shallower valley is preserved. Such regularization behavior can prevent algorithms from being stuck in bad local minima. Also this is advocated and demonstrated to be crucial in order to achieve good generalization of learned DL models in several recent works such as [4]. *In this sense, our BPGrad algorithm/solver essentially aims to locate global optimality in the smoothed objective functions for DL.*

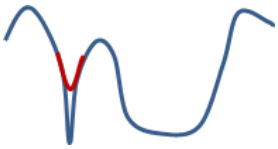


Figure 2. Illustration of Lipschitz continuity as regularization (red) to smoothen a function (blue).

Further BPGrad can generate solutions along the directions of gradients (*i.e*. branch) based on the estimated regions wherein no global optimum should exist theoretically (*i.e*. pruning), and by repeating such branch-and-pruning procedure BPGrad can locate global optimum. Empirically the high demand of computation as well as footprint in memory for running BPGrad inspires us to develop an efficient DL solver to approximate BPGrad towards global optimization.

Contributions: The main contributions of our work are:

C1. We propose a novel approximation algorithm, BPGrad, which is intent on locating global optimum in DL. To our best knowledge, our approach is the *first* algorithmic attempt towards global optimization in DL.

C2. Theoretically we prove that BPGrad can converge to global optimality within finite iterations.

C3. Empirically we propose a novel and efficient DL solver based on BPGrad to reduce the requirement of computation as well as footprint in memory. We provide both theoretical and empirical justification for our solver towards preserving the theoretical properties of BPGrad. We demonstrate that our solver outperforms conventional DL solvers in the applications of object recognition, detection, and segmentation.

1.1. Related Work

Global Optimality in DL: The empirical loss minimization problem in learning deep models is highly dimensional and nonconvex with potentially numerous local minima and saddle points. Blum and Rivest [1] showed that it is difficult to find the global optima because in the worst case even learning a simple 3-node neural network is NP-complete.

In spite of the difficulties in optimizing deep models, researchers have attempted to provide empirical as well as theoretical justification for the success of these models w.r.t. global optimality in learning. Zhang *et al*. [38] empirically demonstrated that sufficiently over-parametrized networks trained with stochastic gradient descent can reach global optimality. Choromanska *et al*. [5] studied the loss surface of multilayer networks using spin-glass model and showed that for many large-size decoupled networks, there exists a band with many local optima, whose objective values are small and close to that of a global optimum. Brutzkus and Globerson [3] showed that gradient descent converges to the global optimum in polynomial time on a shallow neural network with one hidden layer and a convolutional structure and a ReLU activation function. Kawaguchi [17] proved that the error landscape does not have bad local minima in the optimization of linear deep neural networks. Yun *et al*. [36] extended these results and proposed sufficient and necessary conditions for a critical point to be a global minimum. Haeffele and Vidal [13] suggested that it is critical to balance the degrees of positive homogeneity between the network mapping and the regularization function to prevent non-optimal local minima in the loss surface of neural networks. Nguyen and Hein [27] argued that almost all local minima are global optimal in fully connected wide neural networks, whose number of hidden neurons of one layer is larger than that of training points. Soudry and Carmon [30] employed smoothed analysis techniques to provide theoretical guarantee that the highly nonconvex loss functions in multilayer networks can be easily optimized using local gradient descent updates. Hand and Voroninski [14] provided theoretical properties for the problem of enforcing priors provided by generative deep neural networks via empirical risk minimization by establishing the favorable global geometry.

DL Solvers: SGD [2] is the most widely used DL solver due to its simplicity, whose learning rate (*i.e*., step size for gradient) is predefined. In general, SGD suffers from slow convergence, and thus its learning rate needs to be carefully tuned. To improve the efficiency of SGD, several DL solvers with adaptive learning rates have been proposed, including Adagrad [7], Adadelta [37], RMSProp [33] and Adam [18]. These solvers integrate the advantages from both stochastic and batch methods where small mini-batches are used to estimate diagonal second-order information heuristically. These solvers have the capability of escaping saddle points and often yield faster convergence empirically.

Specifically, Adagrad is well suited for dealing with sparse data, as it adapts the learning rate to the parameters, performing smaller updates on frequent parameters and larger updates on infrequent parameters. However, it suffers from shrinking on the learning rate, which motivates Adadelta, RMSProp and Adam. Adadelta accumulates squared gradients to be fixed values rather than over time in Adagrad, RMSProp updates the parameters based on the rescaled gradients, and Adam does so based on the estimated mean and variance of the gradients. Very recently, Mukkamala *et al*. [26] proposed variants of RMSProp and Adagrad with logarithmic regret bounds.

Convention *vs*. Ours: Though the properties of global optimality in DL are very attractive, as far as we know, however,

there is no solver developed intentionally to capture such global optimality so far. To fill this void, we propose our BPGrad algorithm towards global optimization in DL.

From the optimization perspective, our algorithm shares similarities with the recent work [25] on global optimization of general Lipschitz functions (not specifically for DL). In [25] a uniform sampler is utilized to maximize the lower bound of the maximizer (equivalently minimizing the upper bound of the minimizer) subject to Lipschitz conditions. Convergence properties *w.h.p.* are derived. In contrast, our approach considers estimating both lower and upper bounds of the global optimum, and employs the gradients as guidance to more effectively sample the parameter space for pruning. Convergence is proved to show that our algorithm will terminate within finite iterations.

From the empirical solver perspective, our solver shares similarities with the recent work [19] on improving SGD using the feedback from the objective function. Specifically [19] tracks the relative changes in the objective function with a running average, and uses it to adaptively tune the learning rate in SGD. No theoretical analysis, however, is provided for justification. In contrast, our solver does use the feedback from the object function to determine the learning rate adaptively but based on the rescaled distance between the feedback and the current lower bound estimation. Theoretical as well as empirical justifications are established.

2. BPGrad Algorithm for Deep Learning

2.1. Key Notation

We denote $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ as the parameters in the neural network, $(\omega, y) \in \Omega \times \mathcal{Y}$ as a pair of a data sample ω and its associated label y, $\phi : \Omega \times \mathcal{X} \to \mathcal{Y}$ as the nonconvex prediction function represented by the network, f as the objective function for training the network with Lipschitz constant $L \geq 0$, ∇f as the gradient of f over parameters $\mathbf{x}$[1], $\nabla \tilde{f} = \frac{\nabla f}{\|\nabla f\|_2}$ denotes the *normalized* gradient (*i.e.* direction of the gradient), f^* as the global minimum, and $\|\cdot\|_2$ as the ℓ_2-norm operator over vectors.

Definition 1 (Lipschitz Continuity [8])**.** *A function f is* Lipschitz continuous *with Lipschitz constant L on $\mathcal{X}$, if there is a (necessarily nonnegative) constant L such that*

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|_2, \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}. \quad (1)$$

2.2. Problem Setup

We would like to learn the parameters for a given network by minimizing the following objective function f:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \equiv \mathbb{E}_{(\omega\times y)\in\Omega\times\mathcal{Y}}\Big[\mathcal{L}(y, \phi(\omega, \mathbf{x}))\Big] + \mathcal{R}(\mathbf{x}), \quad (2)$$

[1]We assume $\nabla f \neq \mathbf{0}$ *w.l.o.g.* Empirically we can randomly sample a non-zero direction for update wherever $\nabla f = \mathbf{0}$.

where $\mathbb{E}$ denotes the expectation over data pairs, $\mathcal{L}$ denotes a loss function (*e.g.*, hinge loss) for measuring the difference between the ground-truth labels and the predicted labels given data samples, and $\mathcal{R}$ denotes a regularizer over parameters. Particularly we assume that:

F1. f is lower bounded by 0 and upper bounded as well, *i.e.* $0 \leq f(\mathbf{x}) < +\infty, \forall \mathbf{x} \in \mathcal{X}$;
F2. f is differentiable everywhere in the bounded space $\mathcal{X}$;
F3. f is Lipschitz continuous, or can be approximated by Lipschitz functions, with constant $L \geq 0$.

2.3. Algorithm

2.3.1 Lower & Upper Bound Estimation

Consider the situation where samples $\mathbf{x}_1, \cdots, \mathbf{x}_t \in \mathcal{X}$ exist for evaluation by function f with Lipschitz constant L, whose global minimum f^* is reached by the sample $\mathbf{x}^*$. Then based on Eq. 1 and simple algebra, we can obtain

$$\max_{i=1,\cdots,t}\Big\{f(\mathbf{x}_i) - L\|\mathbf{x}_i - \mathbf{x}^*\|_2\Big\} \leq f^* \leq \min_{i=1,\cdots,t} f(\mathbf{x}_i). \quad (3)$$

This provides us a tractable upper bound and an *intractable* lower bound, unfortunately, of the global minimum. The intractability comes from the fact that $\mathbf{x}^*$ is unknown, and thus makes the lower bound in Eq. 3 unusable empirically.

To address this problem, we propose a novel tractable estimator, $\rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)$ $(0 \leq \rho < 1)$. This estimator intentionally introduces a gap from the upper bound, which will be shrunk by either decreasing the upper bound or increasing ρ. As proved in Thm. 1 (see Sec. 2.4), when the parameter space $\mathcal{X}$ is fully covered by the samples $\{\mathbf{x}_i\}$, this estimator will become the lower bound of f^*.

In summary, we define our lower and upper bound estimators for the global minimum as $\rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)$ and $\min_{i=1,\cdots,t} f(\mathbf{x}_i)$, respectively.

2.3.2 Branch & Pruning

Based on our estimators, we propose a novel approximation algorithm, BPGrad, towards global optimization in DL via branch and pruning. We show it in Alg. 1 where the predefined constant $\epsilon \geq 0$ controls the precision of the solution.
Branch: The *inner* loop in Alg. 1 conducts the branch operation to split the parameter space recursively by *sampling*. Towards this goal, we need a mapping between the parameter space and the bounds. Considering the lower bound in Eq. 3, we propose sampling $\mathbf{x}_{t+1} \in \mathcal{X}$ based on the previous samples $\mathbf{x}_1, \cdots, \mathbf{x}_t \in \mathcal{X}$ so that it satisfies

$$\max_{i=1,\cdots,t}\Big\{f(\mathbf{x}_i) - L\|\mathbf{x}_i - \mathbf{x}_{t+1}\|_2\Big\} \leq \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i). \quad (4)$$

Note that an equivalent constraint has been used in [25].

Algorithm 1 BPGrad Algorithm for Deep Learning

Input : objective function f with Lipschitz constant $L \geq 0$, precision $\epsilon \geq 0$
Output : minimizer $\mathbf{x}^*$

Randomly initialize $\mathbf{x}_1$, $t \leftarrow 1$, $\rho \leftarrow 0$;
while $\min_{i=1,\cdots,t} f(\mathbf{x}_i) \leq \frac{\epsilon}{1-\rho}$ **do**
  while $\exists \mathbf{x}_{t+1} \in \mathcal{X}$ *satisfies Eq. 4* **do**
    Compute $\mathbf{x}_{t+1}$ by solving Eq. 5;
    $t \leftarrow t+1$;
  end
  Increase ρ such that $0 \leq \rho < 1$ still holds;
end
return $\mathbf{x}^* = \mathbf{x}_{i^*}$ where $i^* \in \arg\min_{i=1,\cdots,t} f(\mathbf{x}_i)$;

To improve sampling efficiency for decreasing the objective, we propose sampling along the directions of (stochastic) gradients with small distortion. Though gradients only encode local structures of (nonconvex) functions in a high dimensional space, they are good indicators for locating local minima [23, 28]. Specifically, we propose a minimization problem for generating samples:

$$\min_{\mathbf{x}_{t+1}\in\mathcal{X},\eta_t\geq 0} \left\|\mathbf{x}_{t+1} - \left(\mathbf{x}_t - \eta_t \nabla \tilde{f}(\mathbf{x}_t)\right)\right\|_2^2 + \gamma\eta_t^2, \quad (5)$$
$$\text{s.t.} \max_{i=1,\cdots,t} \left\{f(\mathbf{x}_i) - L\|\mathbf{x}_i - \mathbf{x}_{t+1}\|_2\right\} \leq \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i),$$

where $\gamma \geq 0$ is a predefine constant controlling the trade-off between the distortion and the step size $\eta_t \geq 0$. That is, under the condition in Eq. 4, the objective in Eq. 5 aims to generate a sample that has small distortion from an anchor point, whose step size is small as well due to the locality property of gradients, along the direction of the gradient.

Note that other reasonable objective functions may also be utilized here for sampling purpose as long as the condition in Eq. 4 is satisfied. More efficient sampling objectives will be investigated in our future work.

Pruning: In fact Eq. 4 specifies that new samples should be generated outside the union of a set of balls defined by previous samples. To precisely describe this requirement, we introduce a new concept of removable solution space in our work as follows:

Definition 2 (Removable Parameter Space (RPS))**.** *We define the RPS, denoted as $\mathcal{X}_R$, as*

$$\mathcal{X}_R(t) \overset{\text{def}}{=} \cup_{j=1,\cdots,t} \mathcal{B}\left(\mathbf{x}_j, r_j\right), \quad (6)$$

where $\mathcal{B}(\mathbf{x}_j, r_j) = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_j\|_2 < r_j, \mathbf{x} \in \mathcal{X}\}, \forall j$ defines a ball centered at sample $\mathbf{x}_j \in \mathcal{X}$ with radius $r_j = \frac{1}{L}\left[f(\mathbf{x}_j) - \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)\right], \forall j$.

RPS specifies a region wherein the function evaluations of all the points cannot be smaller than the lower bound estimator conditioning on the Lipschitz continuity assumption. Therefore, when the lower bound estimator is higher than the global minimum f^*, we can safely remove all the points in RPS without evaluation. However, when it becomes smaller than f^*, we risk missing the global solutions.

To address this issue, we propose the *outer* loop in Alg. 1 to increase the lower bound for drawing more samples which may further decrease the upper bound later.

2.4. Theoretical Analysis

Theorem 1 (Lower & Upper Bounds)**.** *Whenever $\mathcal{X}_R(t) \equiv \mathcal{X}$ holds, the samples generated by Alg. 1 satisfies*

$$\rho \min_{i=1,\cdots,t} f(\mathbf{x}_i) \leq f^* \leq \min_{i=1,\cdots,t} f(\mathbf{x}_i). \quad (7)$$

Proof. Since f^* is the global minimum, it always holds that $f^* \leq \min_{i=1,\cdots,t} f(\mathbf{x}_i)$. Now when $\mathcal{X}_R(t) \equiv \mathcal{X}$, suppose $\rho \min_{i=1,\cdots,T} f(\mathbf{x}_i) > f^*$ holds, then there would exist at least one point (*i.e.* global minimum) left for sampling, contradicting the condition of $\mathcal{X}_R(t) \equiv \mathcal{X}$. We then complete the proof. □

Corollary 1 (Approximation Error Bound)**.** *Whenever both $\min_{i=1,\cdots,t} f(\mathbf{x}_i) \leq \frac{\epsilon}{1-\rho}$ and $\mathcal{X}_R(t) \equiv \mathcal{X}$ hold, it is satisfied that*

$$\min_{i=1,\cdots,t} f(\mathbf{x}_i) - f^* \leq \epsilon. \quad (8)$$

Theorem 2 (Convergence within Finite Samples)**.** *The total number of samples, T, in Alg. 1 is upper bounded by:*

$$T \leq \left[\frac{2L}{(1-\rho)f_{\min}}\right]^d \cdot \frac{V_{\mathcal{X}}}{C}, \quad (9)$$

where $V_{\mathcal{X}}$ denotes the volume of the space $\mathcal{X}$, $C = \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}+1\right)}$ denotes a constant, and $f_{\min} = \min_{i=1,\cdots,T} f(\mathbf{x}_i)$ denotes the minimum evaluation.

Proof. Given $\forall j, \forall t$ such that $1 \leq j \leq t \leq T-1$, we have

$$\begin{aligned}\|\mathbf{x}_{t+1} - \mathbf{x}_j\|_2 &\geq \frac{1}{L}\left[f(\mathbf{x}_j) - \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)\right] \quad (10)\\ &\geq \frac{1-\rho}{L} \cdot \min_{i=1,\cdots,t} f(\mathbf{x}_i) \geq \frac{(1-\rho)f_{\min}}{L}.\end{aligned}$$

This allows us to generate two balls $\mathcal{B}\left(\mathbf{x}_{t+1}, \frac{(1-\rho)f_{\min}}{2L}\right)$ and $\mathcal{B}\left(\mathbf{x}_j, \frac{(1-\rho)f_{\min}}{2L}\right)$ so that they have no overlap with each other. As a result we can generate T balls with radius of $\frac{(1-\rho)f_{\min}}{2L}$ and no overlaps, and their accumulated volume should be no bigger than $V_{\mathcal{X}}$. That is,

$$V_{\mathcal{X}} \geq \sum_{t=1}^{T} V_{\mathcal{B}\left(\mathbf{x}_t, \frac{(1-\rho)f_{\min}}{2L}\right)} = C\left[\frac{(1-\rho)f_{\min}}{2L}\right]^d T. \quad (11)$$

Further using simple algebra we can complete the proof. □

Algorithm 2 BPGrad based Solver for Deep Learning

Input : number of evaluations n repeating N times at most, objective function f with Lipschitz constant $L \geq 0$, momentum $0 \leq \mu \leq 1$

Output : minimizer $\mathbf{x}^*$

$t \leftarrow 1, \mathbf{v}_1 \leftarrow \mathbf{0}$, and randomly initialize $\mathbf{x}_1$;
for $m \leftarrow 1$ **to** N **do**
  $\rho \leftarrow 1 - \frac{1}{m}$;
  while $t < mn$ **do**
    $\mathbf{v}_{t+1} \leftarrow \mu \mathbf{v}_t - \frac{f(\mathbf{x}_t) - \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)}{L} \cdot \frac{\nabla f(\mathbf{x}_t)}{\|\nabla f(\mathbf{x}_t)\|_2}$;
    $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \mathbf{v}_{t+1}$;
    $t \leftarrow t + 1$;
  end
  if $\min_{i=1,\cdots,t} f(\mathbf{x}_i) \leq \frac{\epsilon}{1-\rho}$ *holds* **then** Break ;
end
return $\mathbf{x}^* = \mathbf{x}_{i^*}$ where $i^* \in \arg\min_{i=1,\cdots,n} f(\mathbf{x}_i)$;

3. Approximate DL Solver based on BPGrad

Though the BPGrad algorithm has nice theoretical properties for global optimization, directly applying Alg. 1 to deep learning will incur the following problems that limit its empirical usage:

P1. From Thm. 2 we can see that due to the high dimensionality of the parameter space in DL it is impractical to draw sufficient samples to cover the entire space.
P2. Solving Eq. 5 involves the knowledge of previous samples, which incurs significant amount of both computational and storage burden for deep learning.
P3. Computing $f(\mathbf{x}_t)$ and $\nabla \tilde{f}(\mathbf{x}_t), \forall \mathbf{x}_t \in \mathcal{X}$ is time-consuming, especially for large-scale data.

To address problem *P1*, in practice we manually set the maximum iterations for both inner and outer loops in Alg. 1.

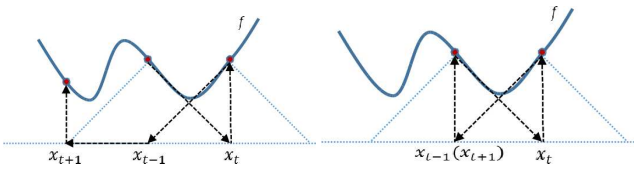
To address problem *P2*, we further make some extra assumptions to simplify the branching/sampling procedure based on Eq. 5 as follows:

A1. Minimizing distortion is much important than minimizing step sizes, *i.e.* $\gamma \ll 1$;
A2. $\mathcal{X}$ is sufficiently large where $\exists \eta_t \geq 0$ so that $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla \tilde{f}(\mathbf{x}_t) \in \mathcal{X} \setminus \mathcal{X}_R(t)$ always holds;
A3. $\eta_t \geq 0$ is always sufficiently small for local update.
A4. $\mathbf{x}_{t+1}$ can be sampled only based on $\mathbf{x}_t$ and $\nabla \tilde{f}(\mathbf{x}_t)$.

By imposing these assumptions upon Eq. 5, we can directly compute the solution as follows:

$$\eta_t = \frac{1}{L}\left[f(\mathbf{x}_t) - \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)\right]. \tag{12}$$

To address problem *P3*, we utilize mini-batches to estimate $f(\mathbf{x}_t)$ and $\nabla \tilde{f}(\mathbf{x}_t)$ efficiently in each iteration.



(a) Sampling using Eq. 5 (b) Sampling using Eq. 12

Figure 3. 1D illustration of difference in sampling between **(a)** using Eq. 5 and **(b)** using Eq. 12. Here the solid blue lines denote function f, the black dotted lines denote the sampling paths starting from $\mathbf{x}_{t-1} \rightarrow \mathbf{x}_t \rightarrow \mathbf{x}_{t+1}$, and each big triangle surrounded by blue dotted lines denotes the RPS of each sample. As we see, (b) suffers from being stuck locally, while (a) can avoid the locality based on the RPS.

In summary, we list our BPGrad solver in Alg. 2 by modifying Alg. 1 for the sake of fast sampling as well as low memory footprint in DL, but at the risk of being stuck in local regions. Fig. 3 illustrates such scenarios in a 1D example. In (b) the sampling method falls into a loop because it does not consider the history of samples but only current one. In contrast, the sampling method in (a) is able to keep generating new samples by avoiding the RPS of previous samples with more computation and storage, as expected.

3.1. Theoretical Analysis

Theorem 3 (Global Property Preservation). *Let* $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla \tilde{f}(\mathbf{x}_t)$ *where* η_t *is computed using Eq. 12. Then* $\mathbf{x}_{t+1}$ *satisfies Eq. 4 if it holds that*

$$\left\langle \mathbf{x}_i - \mathbf{x}_t, \nabla \tilde{f}(\mathbf{x}_t) \right\rangle \geq \frac{f(\mathbf{x}_i) - f(\mathbf{x}_t)}{L}, \forall i = 1, \cdots, t, \tag{13}$$

where $\langle \cdot, \cdot \rangle$ *denotes the inner product between two vectors.*

Corollary 2. *Suppose that a monotonically decreasing sequence* $\{f(\mathbf{x}_i)\}_{i=1,\cdots,t}$ *is generated to minimize function* f *by sampling using Eq. 12. Then the condition in Eq. 13 can be rewritten as follows:*

$$\left\langle \mathbf{x}_i - \mathbf{x}_j, \nabla \tilde{f}(\mathbf{x}_j) \right\rangle \geq 0, \ 1 \leq \forall i < \forall j \leq t. \tag{14}$$

Discussion: Both Thm. 3 and Cor. 2 imply that our solver prefers sampling the parameter space along a path towards a single direction, roughly speaking. However, the gradients in conventional backpropagation have little guarantee to satisfy Eq. 13 or Eq. 14 due to lack of such constraints in learning. On the other hand, momentum [31] is a well-known technique in deep learning to dampen oscillations in gradients and accelerate directions of low curvature. Therefore, our solver in Alg. 2 involves momentum to compensate such drawbacks in backpropagation for better approximation of Alg. 1.

3.2. Empirical Justification

In this section we discuss the feasibility of the assumptions *A1-A4* for reducing computation and storage as well

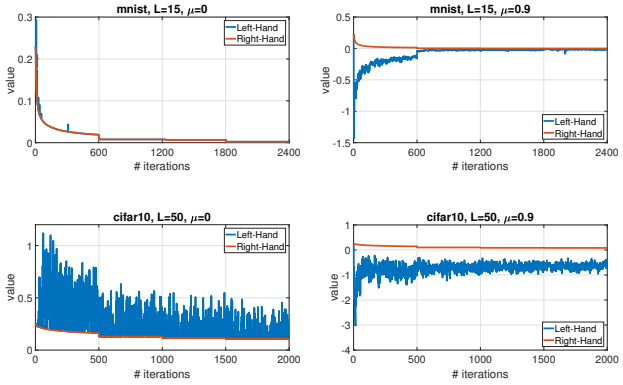


Figure 5. Comparison between LHS and RHS of Eq. 4 based on $\mathbf{x}_t$ returned by Alg. 2 using different values for momentum parameter μ.

as preserving the properties towards global optimization in deep learning. We utilize MatConvNet [35] as our testbed, and run our solver in Alg. 2 to train the default networks in MatConvNet for MNIST [21] and CIFAR10 [20], respectively, using the default parameters without explicit mention. Also we set $N = 1, \mu = 0, L = 15$ for MNIST and $L = 50$ for CIFAR10 by default. For justification purpose we only run 4 epochs on each dataset, 600 and 500 iterations per epoch for MNIST and CIFAR10, respectively. For more experimental details, please refer to Sec. 4.

Essentially assumption *A1* is made to support the other three to simplify the objective in Eq. 5, and assumption *A2* usually holds in deep learning due to its high dimensionality. Therefore, below we only focus on empirical justification of assumptions *A3* and *A4*.

Feasibility of *A3*: To justify this, we collect η_t's by running Alg. 2 on both datasets, and plot them in Fig. 4. Overall these numbers are indeed sufficiently small for local update based on gradients, and η_t decreases with the increase of iterations, in general. This behavior is expected as the objective f is supposed to decrease as well w.r.t. iterations. The value gap at the beginning on the two datasets is induced mainly by different L's.

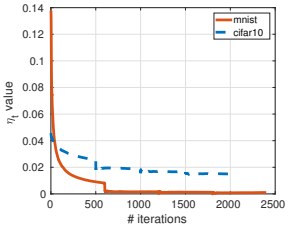


Figure 4. Plots of η_t on MNIST and CIFAR10, respectively.

Feasibility of *A4*: To justify this, we show some evidences in Fig. 5, where we plot the left-hand side (LHS) and right-hand side (RHS) of Eq. 4 based on $\mathbf{x}_t$ returned by Alg. 2. As we see in all the subfigures on the right with $\mu = 0.9$ the values on RHS are always no smaller than those on LHS correspondingly. In contrast, in the remaining subfigures on the left with $\mu = 0$ (*i.e.* conventional SGD update) the values on RHS are always no bigger than those on LHS correspondingly. These observations appear to be robust across different datasets, and irrelevant to parameter L which determines the radius of balls, *i.e.* step sizes for gradients. The momentum parameter μ, which is related to the directions of gradients for updating models, appear to be the only factor to make the samples of our solver satisfy Eq. 4. This also supports our claims in Thm. 3 and Cor. 2 about the relation between model update and gradient in order to satisfy Eq. 4. More evidences have been provided in Sec. 4.1.1. Given these evidences we hypothesize that assumption *A4* may hold empirically when using sufficiently large values for μ.

4. Experiments

To demonstrate the generalization of our BPGrad solver, we test it in the applications of object recognition, detection, and segmentation by training deep convolutional neural networks (CNNs). We utilize MatConvNet as our testbed, and employ its demo code as well as default network architectures for different tasks. Since our solver has the ability of determining learning rates adaptively, we compare ours with another four widely used DL solvers with adaptive learning rates, namely Adagrad, Adadelta, RMSProp, and Adam. We tune the parameters in these solvers to achieve their best performance as we can.

4.1. Object Recognition

4.1.1 MNIST & CIFAR10

The MNIST digital dataset consists of a training set of $60K$ images and a test set of $10K$ images in 10 classes labeled from 0 to 9, where all images have the resolution of 28×28 pixels. The CIFAR-10 dataset consists of a training set of $50K$ images and a test set of $10K$ images in 10 object classes, where the image resolution is 32×32 pixels.

We follow the default implementation to train an individual CNN similar to LeNet-5 [22] on each dataset. For the details of network architectures please refer to the demo code. Specifically for all the solvers, we train the networks for 50 and 100 epochs on MNIST and CIFAR10, respectively, with a mini-batch size 100, weight decay 0.0005, and momentum 0.9. In addition, we fix the initial weights for two networks and the feeding order of mini-batches for fair comparison. The global learning rate is set to 0.001 on MNIST for Adagrad, RMSProp and Adam. On CIFAR10, the global learning rate is set to 0.001 for RMSProp, but to 0.01 for Adagrad, Adam and Eve [19], and it is reduced to 0.005 and 0.001 at the 31-st and 61-st epoch. Adadelta does not require the global learning rate.

For our solver, the parameters n and N typically depend on the numbers of mini-batches and epochs, respectively. Empirically we find that $N = 1$ seems to work well, and thus we use it by default for all the experiments. Accordingly by default n will be set to the product of the numbers of mini-batches and epochs.

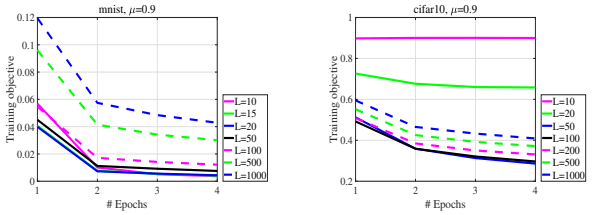


Figure 6. Illustration of robustness of Lipschitz constant L in our solver.

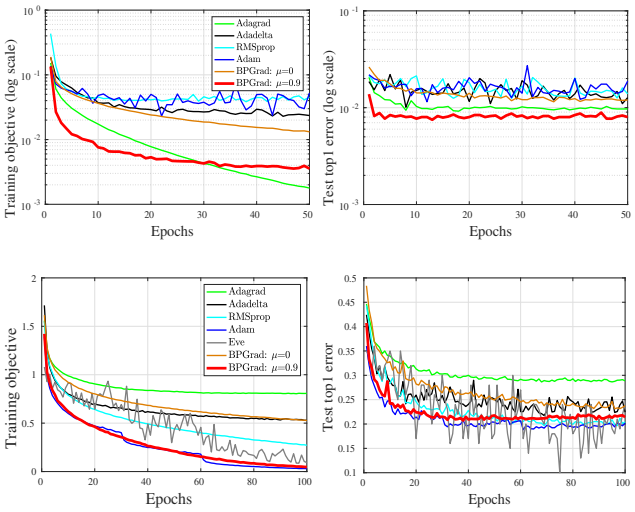


Figure 7. Comparison on **(left)** training objectives and **(right)** test top-1 errors for object recognition using **(top)** MNIST and **(bottom)** CIFAR10.

Also we find that the parameter L as Lipschitz constant is quite robust w.r.t. performance, indicating that heavily tuning this parameter is unnecessary in practice. To demonstrate this, we compare the training objectives of our solver by varying L in Fig. 6. To highlight the differences, here we crop and show the results in the first four epochs, but note that the remaining results have similar behavior. As we can see on MNIST when L varies from 10 to 100, the corresponding curves are clustered, similarly on CIFAR10 for L from 50 to 1000. We decide to set $L = 15$ for MNIST and $L = 50$ for CIFAR10, respectively, in our solver.

Next we show the solver comparison results in Fig. 7. To illustrate the effect of momentum in our solver in terms of performance, here we plot two variants of our solver with $\mu = 0$ and $\mu = 0.9$, respectively. As we see our solver with $\mu = 0.9$ works much better than the counterpart, achieving lower training objectives as well as lower top-1 error at test time. This again provides evidence to support the importance of satisfying Eq. 4 in our solver to search for good solutions toward global optimality.

Overall, our solver performs best on MNIST and slightly inferior on CIFAR10 at test time, although in terms of training objective it achieves competitive performance on MNIST and the best on CIFAR10. We hypothesize that this behavior comes from the effect of regularization on Lipschitz continuity. However, our solver can decrease the objectives much faster than all the competitors in the first few epochs. This observation reflects the superior ability of our solver in determining adaptive learning rates for gradients. Especially on CIFAR10 we also compare an extra solver Eve based on our implementation. Eve was proposed in recent related work [19] that improves Adam with the feedbacks from the objective function, and tested on CIFAR10 as well. As we can see, our solver is much more reliable, performing consistently over epochs.

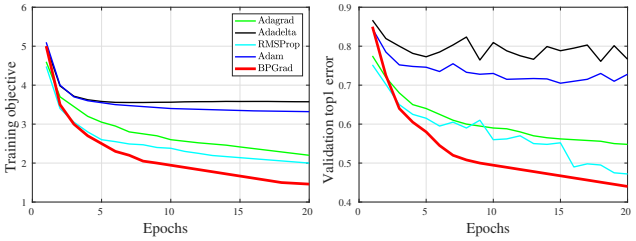


Figure 8. Comparison on **(left)** training objectives and **(right)** validation top-1 errors for object recognition using ImageNet ILSVRC2012.

	Adagrad	Adadelta	RMSProp	Adam	BPGrad
training	49.0	71.6	46.0	70.0	**33.0**
validation	54.8	76.7	47.2	72.8	**44.0**

Table 1. Top-1 recognition error (%) on ImageNet ILSVRC2012 dataset.

4.1.2 ImageNet ILSVRC2012 [20]

This dataset contains about $1.28M$ training images and $50K$ validation images among 1000 object classes. Following the demo code, we train the same AlexNet [20] on it from the scratch using different solvers. We perform training for 20 epochs, with a mini-batch size 256, weight decay 0.0005, momentum 0.9, and default learning rates for the competitors. For our solver we set $L = 100$ and $N = 12$.

We show the comparison results in Fig. 8. It is evident that our solver works the best at both training and test time. Namely, it converges faster to achieve lower objective as well as lower top-1 error on validation dataset. In terms of numbers, ours is 3.2% lower than the second best, RMSProp, at the 20-th epoch as listed in Table 1.

Based on all the experiments above we conclude that our solver is suitable to train deep models for object recognition.

4.2. Object Detection

Following Fast RCNN [11] in the demo code, we conduct the solver comparison on the PASCAL VOC2007 dataset [9] with 20 object classes using selective search [34] as default object proposal approach. For all solvers, we train the network for 12 epochs using the $5K$ images in VOC2007 trainval set and test it using $4.9K$ images in VOC2007 test set. We set the weight decay and momentum to 0.0005 and 0.9,

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
Adagrad	67.5	71.5	60.7	47.1	28.3	72.7	76.7	77.0	34.3	70.2	64.0	72.0	74.2	69.5	64.9	28.8	57.4	60.5	73.1	61.1	61.7
RMSProp	69.1	75.8	61.5	47.9	30.2	74.7	77.1	79.4	33.2	71.1	66.3	74.4	76.3	69.9	65.1	28.9	62.9	62.5	73.2	60.8	63.0
Adam	68.9	**79.9**	64.1	**56.6**	37.0	**77.4**	**77.7**	82.5	38.2	71.5	64.7	**77.6**	77.7	**75.0**	**66.8**	30.6	**65.9**	65.1	**74.4**	**67.9**	**66.0**
BPGrad	**69.4**	77.7	**66.4**	55.1	**37.2**	76.1	**77.7**	**83.6**	**38.6**	**73.8**	**67.4**	76.0	**81.9**	72.7	66.3	**31.0**	64.2	**66.2**	73.8	64.9	**66.0**

Table 2. Average precision (AP, %) of object detection on VOC2007 test dataset.

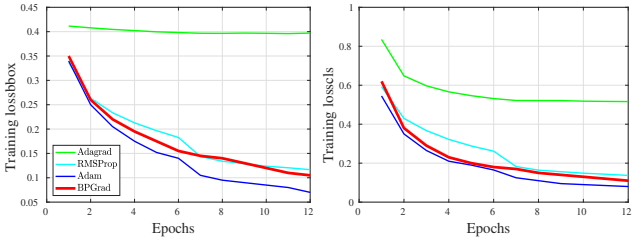


Figure 9. Loss comparison on VOC2007 trainval dataset, including **(left)** the regression loss using bounding boxes and **(right)** the classification loss.

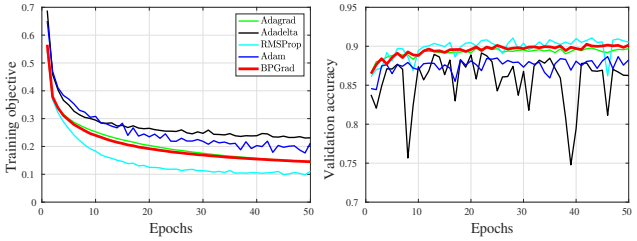


Figure 10. Segmentation performance comparison using FCN-32s model on VOC2011 training and validation datasets.

respectively, and use default learning rates for the competitors. We do not compare with Adadelta because we cannot obtain reasonable performance after heavy parameter tuning. For our solver we set $L = 100$.

We show the training comparison in Fig. 9, and test results in Table 2. Though our training losses are inferior to those of Adam in this case, our solver works as well as Adam at test time on average, achieving best AP on 11 out of 20 classes. This demonstrates the suitability of our solver in training deep models for object detection.

4.3. Object Segmentation

Following the work [24] for semantic segmentation based on fully convolutional networks (FCN), we train FCN-32s with per-pixel multinomial logistic loss and validate it with the standard metric of mean pixel intersection over union (IU), pixel accuracy, and mean accuracy. For all the solvers, we conduct training for 50 epochs with momentum 0 and weight decay 0.0005 on PASCAL VOC2011 [10] segmentation set. For Adagrad, RMSProp and Adam, we find that the default parameters are able to achieve the best performance. For Adadelta, we tune its parameters with $\epsilon = 10^{-9}$. The global learning rate for RMSProp is set to 10^{-5} and 10^{-4} for both Adagrad and Adam. Adadelta does not require the global learning rate. For our solver, we set $L = 500$.

We show the learning curves on training and validation datasets in Fig. 10, and list the test-time comparison results in Table 3. In this case our solver has very similar learning behavior as Adagrad, but achieves the best performance at test time. The smaller fluctuation over epochs on the validation dataset demonstrates again the superior reliability of our solver, compared with the competitors. Taking these observations into account, we believe that our solver has the ability of learning robust deep models for object segmentation.

	mean IU	pixel accuracy	mean accuracy	average
Adagrad	60.8	89.5	77.4	75.9
Adadelta	46.6	86.0	54.4	62.3
RMSProp	60.5	**90.2**	71.0	73.9
Adam	50.9	87.2	66.4	68.2
BPGrad	**62.4**	89.8	**79.6**	**77.3**

Table 3. Numerical comparison on semantic segmentation performance (%) using VOC2011 test dataset at the 50-th epoch.

5. Conclusion

In this paper we propose a novel approximation algorithm, namely BPGrad, towards searching for global optimality in DL via branch and pruning based on Lipschitz continuity assumption. Our basic idea is to keep generating new samples from the parameter space (*i.e.* branch) outside the removable parameter space (*i.e.* pruning). Lipschitz continuity not only provides us a way to estimate the lower and upper bounds of global optimality, but also serves as regularization to further smooth the objective functions in DL. Theoretically we prove that under some conditions our BPGrad algorithm can converge to global optimality within finite iterations. Empirically in order to avoid the high demand of computation as well as storage for BPGrad in DL, we propose a new efficient solver. Theoretical and empirical justification on preserving the properties of BPGrad is provided. We demonstrate the superiority of our solver to several conventional DL solvers in object recognition, detection, and segmentation.

Acknowledgement

Dr. Zhang was supported by MERL. Mr. Wu and Prof. Wang were supported in part by the Kansas NASA EPSCoR Program under Grant KNEP-PDG-10-2017-KU, the United States Department of Agriculture (USDA) under Grant USDA 2017-67007-26153, and Nvidia GPU grant.

References

[1] A. Blum and R. L. Rivest. Training a 3-node neural network is np-complete. In *NIPS*, pages 494–501, 1989. 2

[2] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016. 1, 2

[3] A. Brutzkus and A. Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. *arXiv preprint arXiv:1702.07966*, 2017. 2

[4] P. Chaudhari, A. Choromanska, S. Soatto, and Y. LeCun. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016. 1, 2

[5] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *AISTATS*, pages 192–204, 2015. 2

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 1

[7] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(Jul):2121–2159, 2011. 1, 2

[8] K. Eriksson, D. Estep, and C. Johnson. *Applied Mathematics Body and Soul: Vol I-III*. Springer-Verlag Publishing, 2003. 1, 3

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 7

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html. 8

[11] R. Girshick. Fast r-cnn. In *CVPR*, pages 1440–1448, 2015. 7

[12] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 1

[13] B. D. Haeffele and R. Vidal. Global optimality in neural network training. In *CVPR*, pages 7331–7339, 2017. 1, 2

[14] P. Hand and V. Voroninski. Global guarantees for enforcing deep generative priors by empirical risk. *arXiv preprint arXiv:1705.07576*, 2017. 1, 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[16] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. 1

[17] K. Kawaguchi. Deep learning without poor local minima. In *NIPS*, pages 586–594, 2016. 2

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 2

[19] J. Koushik and H. Hayashi. Improving stochastic gradient descent with feedback. *arXiv preprint arXiv:1611.01505*, 2016. 3, 6, 7

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1, 6, 7

[21] Y. LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998. 6

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[23] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent only converges to minimizers. In *COLT*, pages 1246–1257, 2016. 4

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 8

[25] C. Malherbe and N. Vayatis. Global optimization of lipschitz functions. In *ICML*, 2017. 3

[26] M. C. Mukkamala and M. Hein. Variants of rmsprop and adagrad with logarithmic regret bounds. *arXiv preprint arXiv:1706.05507*, 2017. 2

[27] Q. Nguyen and M. Hein. The loss surface of deep and wide neural networks. *arXiv preprint arXiv:1704.08045*, 2017. 2

[28] I. Panageas and G. Piliouras. Gradient descent only converges to minimizers: Non-isolated critical points and invariant regions. *arXiv preprint arXiv:1605.00405*, 2016. 4

[29] D. G. Sotiropoulos and T. N. Grapsa. A branch-and-prune method for global optimization. In *Scientific Computing, Validated Numerics, Interval Methods*, pages 215–226. Springer, 2001. 1

[30] D. Soudry and Y. Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016. 2

[31] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, pages 1139–1147, 2013. 5

[32] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014. 1

[33] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 1, 2

[34] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. 7

[35] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACM Multimedia*, pages 689–692, 2015. 6

[36] C. Yun, S. Sra, and A. Jadbabaie. Global optimality conditions for deep neural networks. *arXiv preprint arXiv:1707.02444*, 2017. 1, 2

[37] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 1, 2

[38] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. 2

[39] S. Zhang, A. E. Choromanska, and Y. LeCun. Deep learning with elastic averaging sgd. In *NIPS*, pages 685–693, 2015. 1

[40] Z. Zhang and M. Brand. Convergent block coordinate descent for training tikhonov regularized deep neural networks. In *NIPS*, 2017. 1