Optimization for Training Deep Models and Deep Learning Based Point Cloud Analysis and Image Classification

Yuanwei Wu

Submitted to the graduate degree program in Department of Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Date defended: December 9, 2019

Abstract

Deep learning (DL) has dramatically improved the state-of-the-art performances in broad applications of computer vision, such as image recognition, object detection, segmentation, and point cloud analysis. However, the reasons for such huge empirical success of DL still keep elusive theoretically. In this dissertation, to understand DL and improve its efficiency, robustness, and interpretability, we theoretically investigate optimization algorithms for training deep models and empirically explore deep learning based point cloud analysis and image classification. 1). **Optimization for Training Deep Models:** Neural network training is one of the most difficult optimization problems involved in DL. Recently, it has been attracting more and more attention to understand the global optimality in DL. However, we observe that conventional DL solvers have not been developed intentionally to seek for such global optimality. In this dissertation, we propose a novel approximation algorithm, *BPGrad*, towards optimizing deep models globally via branch and pruning. The proposed BPGrad algorithm is based on the assumption of Lipschitz continuity in DL, and as a result, it can adaptively determine the step size for the current gradient given the history of previous updates, wherein theoretically no smaller steps can achieve the global optimality. 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. 2). **Deep Learning Based Point Cloud Analysis and Image Classification:** The network architecture is of central importance for many visual recognition tasks. In this dissertation, we focus on the emerging field of point clouds analysis and image classification. **2.1) Point cloud analysis:** We observe that traditional 6D pose estimation approaches are not sufficient to address the problem where neither a CAD model of the object nor the ground-truth 6D poses of its instances are available during training. We propose a novel unsupervised approach to jointly learn the 3D object model and estimate the 6D poses of multiple instances of the same object in a single end-to-end deep neural network framework, with applications to depth-based instance segmentation. The inputs are depth images, and the learned object model is represented by a 3D point cloud. Specifically, our network produces a 3D object model and a list of rigid transformations on this model to generate instances, which when rendered must match the observed point cloud to minimizing the Chamfer distance. To render the set of instance point clouds with occlusions, the network automatically removes the occluded points in a given camera view. Extensive experiments evaluate our technique on several object models and a varying number of instances in 3D point clouds. Compared with popular baselines for instance segmentation, our model not only demonstrates competitive performance, but also learns a 3D object model that is represented as a 3D point cloud. **2.2) Low quality image classification:** We propose a simple while effective unsupervised deep feature transfer network to address the degrading problem of the state-of-the-art classification algorithms on low-quality images. No fine-tuning is required in our method. We use a pre-trained deep model to extract features for both high-resolution (HR) and low-resolution (LR) images, and feed them into a multilayer feature transfer network for knowledge transfer. An SVM classifier is learned directly using these transferred low-resolution features. Our network can be embedded into the state-of-the-art network models as a plug-in feature enhancement module. It preserves data structures in feature space for HR images, and transfers the distinguishing features from a well-structured source domain (HR features space) to a not well-organized target domain (LR features space). Extensive experiments show that the proposed transfer network achieves significant improvements over the baseline method.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the breakthrough on image classification achieved by AlexNet Krizhevsky et al. (2012) at 2012, deep learning has dramatically improved the state-of-the-art performance in many research areas and practical applications, such as in image classification Krizhevsky et al. (2012); He et al. (2016); Cen & Wang (2019a); Wu et al. (2019b); Xu et al. (2019), object detection Girshick et al. (2014); Ren et al. (2015); Ma et al. (2018); Zhu et al. (2018a); Liu et al. (2018); Li et al. (2019), visual tracking Zhu et al. (2018a); Bharati et al. (2016); Wu et al. (2017); Bharati et al. (2018), instance segmentation He et al. (2017); Gupta et al. (2019); Chen et al. (2019), 6D pose estimation Xiang et al. (2017); Wang et al. (2019a), point cloud analysis Qi et al. (2017a,b); Wu et al. (2019a), depth estimation He et al. (2018a,b), face recognition Taigman et al. (2014); Wang & Deng (2018); Cen & Wang (2019b), image-to-image translation Isola et al. (2017); Zhu et al. (2017), speech recognition Hinton et al. (2012); Xiong et al. (2016); Sotelo et al. (2017), and natural language processing Sutskever et al. (2014); Devlin et al. (2018). In this chapter, we first present an overview of deep learning. Then, we discuss the challenges and problems, followed by the contributions of our work in this dissertation. Finally, we briefly describe the main contents of each chapter in the roadmap of the dissertation.

1.1 Overview of Deep Learning

Deep learning is a powerful machine learning technique, which makes use of hierarchical neural networks. Convolutional Neural Networks (CNNs) LeCun et al. (1998) have been widely used for computer vision applications. The evolution of CNNs is deeply affected by the progress in

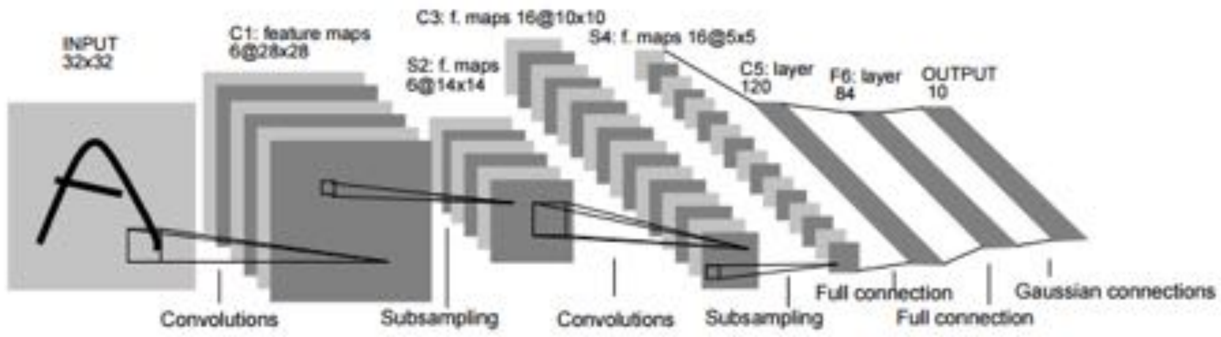


Figure 1.1: The architecture of LeNet-5 network (source: LeCun et al. (1998)).

visual perception mechanism of the living creatures decades ago Gu et al. (2015). Fukushima & Miyake (1982) introduced the concept of local receptive fields Hubel & Wiesel (1968) into their neocognitron. Inspired by this discovery, LeCun et al. (1989) proposed the modern framework of CNN and later it was imporved by LeCun et al. (1998). The so-called LeNet-5 is stacked with multiple layers and trained using Backpropagation LeCun et al. (1989). Take LetNet-5 as an example (see Figure 1.1), it consists of three types of layers, namely convolution, sub-sampling, and fully connected layers.

The convolutional layer is the core building block of a deep CNN. It consists of the weights and biases embedded in a set of learnable filters (aka "kernels"). In order to reduce the number of parameters, the convolutional layer shares parameters. As shown in Fig. 1.1, a typical filter size on a first layer of deep convolutional network is $5 \times 5 \times 1$ if the input is gray scale image ($5 \times 5 \times 3$ for color image). The dimension (width$\times$height) of the filter size is called the receptive field of the neuron, which is a hyper-parameter. The third dimension of each filter is called the channel of that filter. It depends on the input volume. During the forward propagation, each filter is convolving across the width and height of the input volume and compute the dot production at each sub-region. After convolution, each filter will produce a 2-dimensional feature map that gives the response of that filter at each spatial location. An element-wise nonlinear activation function is commonly used after the feature map.

The subsampling layer, also refers to pooling layer. This layer does not consist of learnable

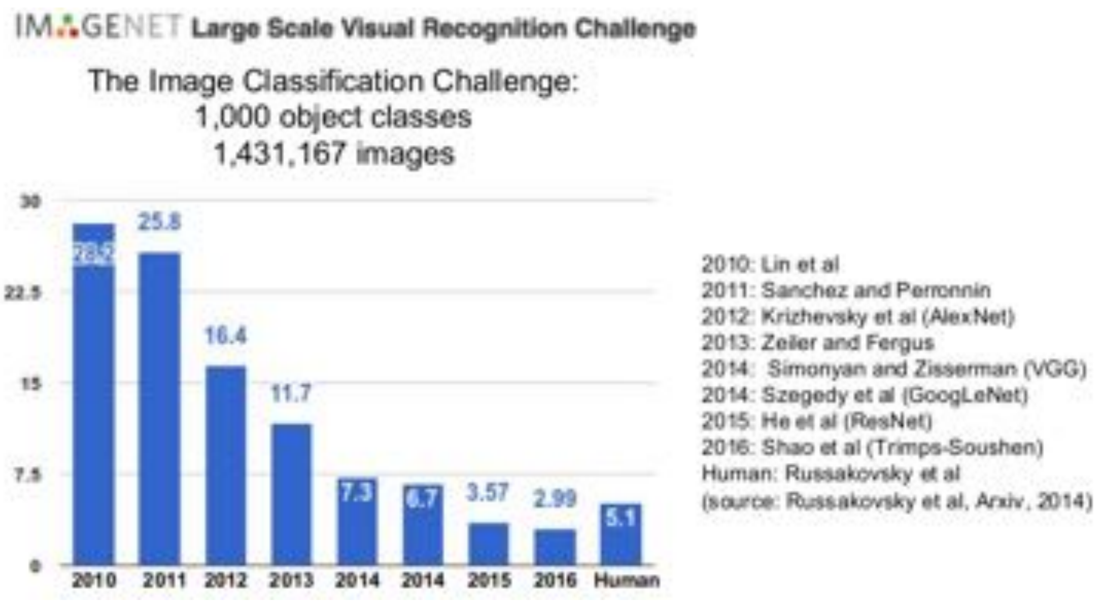


Figure 1.2: The winner results for the task of Image classification on ImageNet (source: Russakovsky et al. (2015)).

parameters. The basic idea is to convolve with the input volume on a subregion using a filter (normally with size of 2×2) to reduce the spatial size of input volume. The typical subsampling methods are max pooling or average pooling, which extracts the maximum value or the average value from the subregion. The intuitive idea behinds this layer is to drastically reduce the spatial dimension of the input volume layer. For example, it reduces the amount of parameters by 75% when using 2×2 filter, thus it reduces the computational load. The fully connected layer has dense connection with its input volume. As shown in Fig. 1.1, the $F6$ layer has 84 neurons, and the output layer has 10 neurons. The 84 neurons on $F6$ layer are densely connected with neurons on the output layer.

The number of categories is 1,000 for the Image Classification task on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Russakovsky et al. (2015). As we can see from Figure 1.2, from the year of 2010 to 2011, the error of Image Classification on ImageNet only decreased by 3%. At 2012, Krizhevsky et al. (2012) won the Image Classification challenge using deep CNN (refers to AlexNet), beated the second winner by about 10%. This success rekindled the widely use of deep learning in computer vision. In 2013, Zeiler & Fergus (2014) optimized

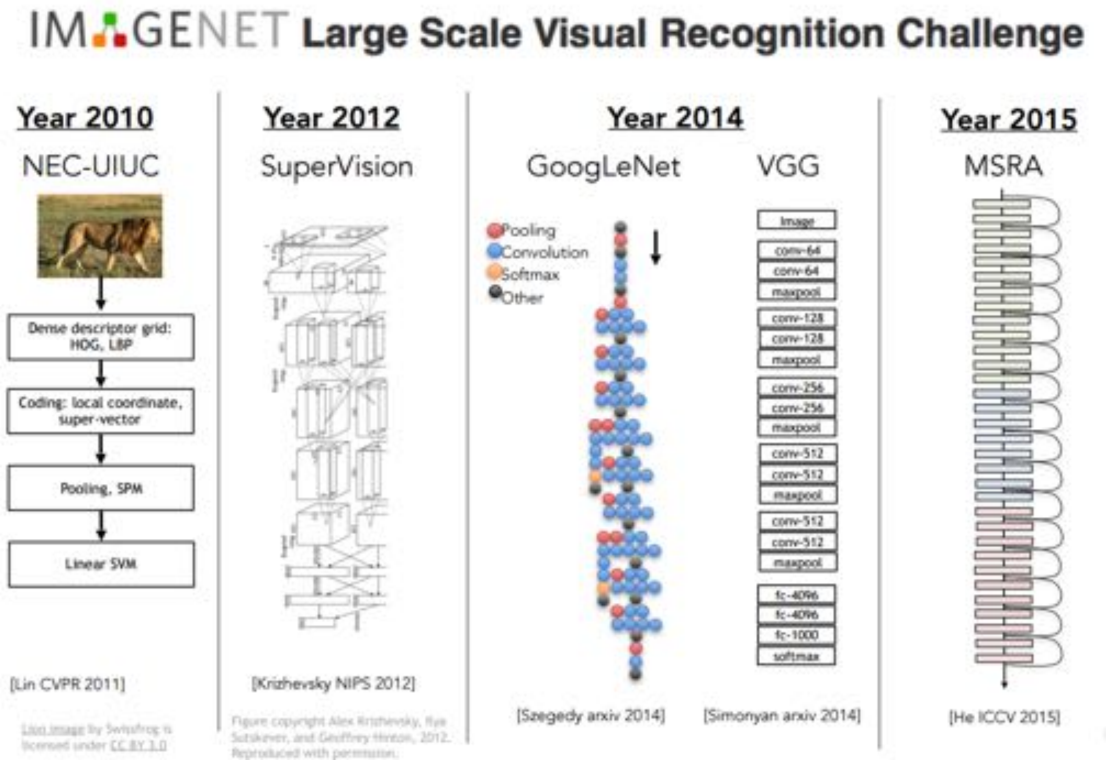


Figure 1.3: The state-of-the-art CNN architectures used in computer vision, 2010: SVM, 2012: AlexNet, 2014: GoogLeNet and VGGNet, 2015: ResNet (source: Stanford University CS231n, spring 2017).

the training procedures on top of the AlexNet architecture and won the Image Classification challenge with the top-5 error of 11.7%. In 2014, Szegedy et al. (2015) proposed GoogLeNet to win the Image Classification challenge with the top-5 error of 6.7%. Moreover, Szegedy et al. (2015) proposed inception module in their deep network, which makes the training more efficient. At the same year, Simonyan & Zisserman (2014) proposed VGGNet and won the second place with top-5 error of 7.3%. Moreover, Simonyan & Zisserman (2014) proposed to use kernel filters with smaller receptive field. They use 7×7, 5×5 and 3×3, compared with the larger values of 11×11 used in AlexNet. In 2015, the researchers in Microsoft Research Asia proposed ResNet, which is a much deeper architecture (with more than 100 layers) and pushed the classification top-5 error to 3.57%, which is even higher than the human's performance 5.1%. He et al. (2016)) proposed residual module which makes the training on deeper networks become more efficient. The architectures of the state-of-the-art deep CNNs are shown in Figure 1.3[1].

1.2 Problems and Challenges

It is well known that the empirical success of deep learning stems mainly from better network architectures He et al. (2016), the availability of massive dataset like ImageNet Deng et al. (2009), and increasing computation power of GPUs. However, the reasons for such huge empirical success of deep learning still keep elusive. How to improve the efficiency, interpretability and robustness of deep learning for different applications is still an open problem.

In this dissertation, we theoretically study the optimization techniques for training deep models, and empirically explore deep learning for tasks in point cloud analysis and low quality images classification, towards efficient and robust deep learning. We describe these problems, discuss existing challenges and present our approach to these problems below.

1.2.1 Optimization for Training Deep Models

Neural network training is one of the most difficult optimization problems involved in deep learning. It involves optimization in many contexts, such as how batch size affects training, the problems of local minima and saddle points, different training algorithm, importance of learning rate, and initialization policies and their implications. In this dissertation, we focus on one particular case of optimization, which is to find the parameters θ of a network that reduce a cost function $J(\theta)$ in neural network training.

1.2.1.1 Towards Global Optimality in Deep Learning

Global optimality is always desirable and preferred for optimization, which helps the generalization of learned models. In very recent years, researchers start to theoretically study the relations between global optimality and generalization in deep learning, such as Haeffele & Vidal (2017);

[1]source from `http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture1.pdf`

Yun et al. (2018); Haeffele & Vidal (2017); Yun et al. (2018); Lin & Jegelka (2018); Liang et al. (2018); Du et al. (2019); Zou et al. (2018); Zhu et al. (2018b,c); Zhou et al. (2019b). It has been proved that under certain (very restrictive) conditions, the critical points in deep learning can actually achieve global optimality, even though its objective is highly nonconvex. All these papers above also indicate that global optimality in deep learning improves the generalization. Such theoretical results may partially explain why such deep models work well in practical applications.

However, from the algorithmic perspective, it is extremely challenging to locate global optimality in deep learning due to its high dimensionality and non-convexity. To our best knowledge, current deep learning solvers, including stochastic gradient descent (SGD) Bottou et al. (2016), Adagrad Duchi et al. (2011), Adadelta Zeiler (2012), RMSProp Tieleman & Hinton (2012) and Adam Kingma & Ba (2014) are not intentionally developed for this purpose. Therefore, these solutions might not necessarily be the global optimum. How to locate or toward the global optimality in deep neural network training is still an open problem.

1.2.2 Deep Learning Based Point Cloud Analysis and Image Classification

The architecture of neural networks is of central importance for many visual recognition tasks. In this dissertation, we focus on the emerging field of unsupervised learning for point clouds and low quality image analysis.

1.2.2.1 Point Cloud Analysis

Estimating the 6D poses (3D rotation and 3D translation) of rigid objects in 3D point clouds is an active research area with many real-world applications, such as robotic grasping and manipulation, virtual-reality/augmented-reality, and human-robot interaction Brachmann et al. (2016); Xiang et al. (2017); Wang et al. (2019a). The prior methods proposed to solve this task typically make two strong assumptions (or simplifications) on the problem setup, such as (i) a 3D CAD model of the object is available, and (ii) a (sufficiently large) training set is available with annotated 6D poses of each object instance. However, traditional 6D pose estimation approaches are not sufficient to address this unsupervised problem, in which neither a CAD model of the object nor the ground-truth 6D poses of its instances are available during training. Therefore, it is very important to study this problem with deep learning framework and unsupervised learning techniques.

1.2.2.2 Low Quality Image Classification

The current state-of-the-art image recognition systems Krizhevsky et al. (2012); He et al. (2016) are typically learned using images that are of sufficiently high quality (*e.g.* 224×224 or higher). However, in many emerging applications such as robotics, autonomous driving, and surveillance videos analytics, the performances of such recognition system are largely jeopardized by low quality images/videos acquired from complex unconstrained environments, suffering from various types of degradations such as low resolution, noise, and motion blur. Therefore, how to address the performance degradation on low quality images is still an active research area.

1.3 Contributions

In this dissertation, to better understand deep learning towards its efficiency and robustness, we propose novel approximate algorithm *BPGrad* Wu et al. (2018) to theoretically study the optimization techniques for training the deep models, as well as empirically investigate the network architecture design for deep learning based *point cloud analysis* Wu et al. (2019a) and *low quality image classification* Wu et al. (2019b). Our contributions are summarized as follows.

- **Towards Global Optimality in Deep Learning:** In this work, 1) we propose a novel *approximation algorithm*, BPGrad, towards the global optimization in deep learning. To our best knowledge, our approach is the *first* algorithmic attempt to locate the global optimality for training deep models. 2) Theoretically we prove that BPGrad can converge to the global minima within finite iterations as the tractable lower and upper bounds allow it to prune the parameter space without missing. 3) Empirically we propose a novel and efficient deep learning solver based on BPGrad to reduce the requirement of computation as well as a footprint in memory with better performance in object recognition, detection, and segmentation. We provide both theoretical and empirical justifications of our solver for preserving the theoretical properties of BPGrad.

- **Point Cloud Analysis:** In this work, 1) we propose a novel task and generic unsupervised algorithm to jointly learn a complete 3D object model and estimate 6D poses of object instances in 3D point clouds. 2) We incorporate occlusion reasoning into our framework, which enables it to learn full 3D models from depth maps (whose point clouds do not show occluded sides or occluded portions of objects), and handle learning symmetric object models via modifications to our loss function involving multiple rotation channels. 3) We provide extensive experiments on unsupervised instance segmentation, demonstrating that in addition to learning a 3D object model, our method performs better than or comparably to standard baselines for instance segmentation.

- **Low Quality Image Analysis:** In this work, 1) We propose an unsupervised deep feature transfer network without using fine-tuning. The feature vectors of both the high resolution and low resolution images are extracted using pre-trained deep mode, and then are fed into a multilayer feature transfer network for knowledge transfer. A SVM classifier is learned directly using these transferred low resolution features. Our network can be embedded into the state-of-the-art CNNs as a plug-in feature enhancement module. 2) It preserves data structures in feature space for high resolution images, by transferring the discriminative features from a well-structured source domain (high resolution features space) to a not well-organized target domain (low resolution features space). 3) The proposed approach achieves better performance than the baseline method for low resolution image classification task.

1.4 Roadmap of Dissertation

The rest of the dissertation is organized as follows:

- **Chapter 2: Related Work**

This chapter reviews related works for deep learning solvers, global optimality in deep learning, point cloud analysis and low quality image classification.

- **Chapter 3: Towards Global Optimality in Deep Learning**

  This chapter describes the proposed novel approximation algorithm, *BPGrad*, which has the capability towards optimizing deep models globally via branch and pruning. The proposed BPGrad algorithm is based on the assumption of Lipschitz continuity of the objective function, 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. Finally, we propose an efficient BPGrad solver, and it outperforms conventional deep learning solvers such as Adagrad, Adadelta, RMSProp, and Adam in the tasks of object recognition, detection, and segmentation.

- **Chapter 4: Point Cloud Analysis**

  This chapter describes our novel unsupervised approach for joint 3D object model learning and 6D poses estimation of multiple instances of a same object, with applications to depth-based instance segmentation. The inputs are depth images, and the learned object model is represented by a 3D point cloud. To solve this problem, we propose to jointly optimize the model learning and pose estimation in an end-to-end deep learning framework. Specifically, our network produces a 3D object model and a list of rigid transformations on this model to generate instances, which when rendered must match the observed point cloud to minimizing the Chamfer distance. To render the set of instance point clouds with occlusions, the network automatically removes the occluded points in a given camera view. Extensive experiments evaluate our technique on several object models and varying number of instances in 3D point clouds. We demonstrate the application of our method to instance segmentation of depth images of small bins of industrial parts. Compared with popular baselines for instance segmentation, our model not only demonstrates competitive performance, but also learns a

3D object model that is represented as a 3D point cloud.

- **Chapter 5: Low Quality Image Analysis**

  This chapter presents our simple while effective unsupervised deep feature transfer algorithm for low resolution image classification. No fine-tuning on convenet filters is required in our method. We use pre-trained convenet to extract features for both high and low resolution images, and then feed them into a two-layer feature transfer network for knowledge transfer. A SVM classifier is learned directly using these transferred low resolution features. Our network can be embedded into the state-of-the-art deep neural networks as a plug-in feature enhancement module. Extensive experiments on VOC2007 test set show that the proposed method achieves significant improvements over the baseline of using feature extraction.

- **Chapter 6: Conclusions and Future Work**

  This chapter summarizes our contributions and discusses the strengths and limitations of each proposed method. Some open problems and future work are described at the end.

Chapter 2

Related Work

In this chapter, we review related areas of each of our problems below.

2.1 Towards Global Optimality in Deep Learning

We propose a novel approximate algorithm, *BPGrad*, towards optimizing deep models globally via branch and pruning. It is closely related to the global optimality of training deep neural networks, the conventional solvers, and the branch-bound-pruning algorithms. We review the related work as follows.

2.1.1 Global Optimality in Deep Learning

The empirical loss minimization problem in DL is high-dimensional and nonconvex with potentially numerous local minima and saddle points. Earlier work on training neural networks Blum & Rivest (1989) 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 challenges in training 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. For instance, Zhang et al. (2016) empirically demonstrated that sufficiently over-parametrized networks trained with stochastic gradient descent can reach global optimality. Lin & Jegelka (2018) showed that ResNet He et al. (2016) with one single neuron per hidden layer is able to generalize well and provide universal approximation as the depth goes to infinity. Liang et al. (2018) studied the landscape of neural networks and proved that by adding a special neuron and an associated regularizer, the new loss function has no spurious local minimum for binary classification tasks. Du et al. (2019) showed that randomly initialized gradient descent converges to a globally optimal solution at a linear convergence rate for the quadratic loss function on two-layer fully connected ReLU Nair & Hinton (2010) activated neural networks in over-parameterization regime. Zou et al. (2018) studied the binary classification problem using an over-parameterized deep ReLU network and proved that both gradient descent and stochastic gradient descent can find the global minima of the training loss with proper random weight initialization. Zhu et al. (2018b) extended the theoretical understanding on the generalization of over-parameterized two and three-layer neural networks. Zhu et al. (2018c) studied that SGD can find global minima in polynomial time on the training objective of over-parameterized networks. Brutzkus & Globerson (2017) 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 with a ReLU activation function. Nguyen & Hein (2017) 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. Yun et al. (2018) extended these results and propose several sufficient and necessary conditions for a critical point to be a global minimum. Haeffele & Vidal (2017) derived sufficient conditions to guarantee that the local minima for their proposed networks are globally optimal, and argued 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.

Several recent works have also studied on how to overcome poor local optima using the global loss structures. For example, Freeman & Bruna (2017) theoretically proved that local minima of a half-rectified network can be connected with a curve so that the loss is upper-bounded along the curve by a constant that depends on the model over-parameterization and the smoothness of the data. Kawaguchi (2016) showed that the error landscape does not have bad local minima in the optimization of linear deep neural networks. Choromanska et al. (2015) 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. Dauphin et al. (2014) argued that, based on the results from statistical physics and random matrix theory, it is the saddle points rather than local minima that causes the difficulty in deep learning. Lee et al. (2016) showed that, with random initialization, gradient descent almost surely converges to a local minimizer and not a saddle point. Hand & Voroninski (2017) provided theoretical properties for the problem of enforcing priors provided by generative deep neural networks via empirical risk minimization through establishing the favorable global geometry. To better understand the local structure of loss functions and the effect of loss landscapes on generalization, Li et al. (2018) proposed a visualization method for the loss surface near the minima found by SGD, and showed that the loss surfaces of modern residual networks seem to be smoother than those of VGG-like models.

Some other works explore the local structures of minima to study the differences between sharp and wide local minima during training found by SGD and its variants. For example, Hochreiter & Schmidhuber (1997) and Keskar et al. (2017) trained neural networks using small and large mini-batch sizes respectively, and found that flat minima deliver strong generalization, while sharp minima lead to pool performance on the test dataset. Keskar et al. (2017) claimed that SGD tends to converge to broad local optima, while batch gradient methods are more likely to converge to sharp optima. Dinh et al. (2017) argued that most notions of flatness are problematic for deep models and cannot be directly applied to explain generalization. Soudry & Carmon (2016) 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. Draxler et al. (2018) also provided evidence that local optima found by SGD are not isolated points in the parameter space, but can be connected by simple curves of near constant loss. Chaudhari et al. (2017) proposed Entropy-SGD that explicitly forces optimization towards wide valleys using the local geometry of the energy landscape, which leads to better generalization lying in large flat regions of the energy landscape, while avoiding the solutions with poor generalization located in the sharp valleys.

2.1.2 Deep Learning Solvers

SGD is one of the most widely used solvers for object recognition Krizhevsky et al. (2012); Simonyan & Zisserman (2014); He et al. (2016), object detection Girshick et al. (2014); Ren et al. (2015); He et al. (2017), and object segmentation Long et al. (2015a). 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 Momentum Qian (1999), Adagrad Duchi et al. (2011), Adadelta Zeiler (2012), RMSProp Tieleman & Hinton (2012) and Adam Kingma & Ba (2014). As stated in Ruder (2016), these solvers are able to escape the saddle points and often yield faster convergence empirically by integrating the advantages from both stochastic and batch methods where small mini-batches are used to adopt historical gradient information to automatically adjust the learning rate.

Adagrad is well suited to deal 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 the same based on the estimated mean and variance of the gradients. Mukkamala & Hein (2017) proposed variant solvers of RMSProp and Adagrad with logarithmic regret bounds. Readers may refer to Ruder (2016) for a comprehensive review on the gradient descent based optimization algorithms.

2.1.3 Branch, Bound and Pruning

Branch-and-bound (B&B) Land & Doig (2010) is one of the promising methods for global optimization in nonconvex problems. The basic idea of B&B is to recursively divide the feasible set of a problem into disjoint subsets (“branching"), where each node represents a subproblem that only conducts searches on the subset of that node. The key idea is to keep the track of bounds on the minimum, and use these bounds to “prune" the search space, removing candidate solutions that cannot be optimal provably. To our best knowledge, currently no DL solvers are developed based on B&B, while ours is.

Several approaches based on B&B have been proposed to solve the problems in combinatorial optimization, *e.g.*mixed integer programming (MIP) He et al. (2014), maximum a posteriori (MAP) inference Sun et al. (2012) and structured prediction Lehmann et al. (2011); Kokkinos (2011). Sun et al. (2012) solved the MAP inference problem in general Markov random fields by leveraging a data structure to reduce the time complexity using branch-and-bound technique. He et al. (2014) proposed an adaptive node order searching strategy learned by imitation learning on a branch-and-bound tree to specify the order of nodes and improve the chance of quickly finding a good incumbent solution for the MIP problem. Lehmann et al. (2011) addressed the task of object class detection using principled implicit shape model where branch-and-bound is used to search for objects. Qian & Liu (2013) proposed a decoding algorithm based on the branch-and-bound framework where non-local features are bounded by a linear combination of local features and the upper bound is searched by dynamic programming. Schwing & Urtasun (2012) presented a branch-and-bound approach for 3D indoor scene understanding to split the label space in terms of candidate sets of 3D layouts and bound the energy for entire sets by constructing upper-bounding contributions of each individual face. Kokkinos (2011) introduced a dual-tree branch-and-bound algorithm for part-based detection in which the upper bounds of the cost function of a part-based model are computed by dual trees. Ferrari et al. (2008) proposed a pruning approach to progressively reduce the search space for body parts for human pose estimation in TV and movie video shots. In object detection, Pedersoli et al. (2011) developed a multiple-resolutions hierarchical part based model and a coarse-to-fine inference procedure to recursively prune unpromising part placements from the search space.

2.2 Point Cloud Analysis

We propose an unsupervised algorithm to jointly learn 3D object model and estimate 6D pose for depth-based instance segmentation. The problems of 3D object model learning, 6D pose estimation, and 3D instance segmentation have been addressed by several prior publications. We review some of the most related work below.

2.2.1 3D Object Model Learning

Recovering 3D models of objects from images using deep neural networks has been gaining significant momentum in recent years Fan et al. (2017); Insafutdinov & Dosovitskiy (2018). Fan et al. (2017) address the problem of 3D reconstruction from a single image to generate 3D point clouds. Insafutdinov & Dosovitskiy (2018) address the problem of learning 3D shapes and camera poses from a collection of unlabeled category-specific images by assembling the pose estimators and then distilling to a single student model. However, these works focus on 3D reconstruction from images containing a single object instance, which do not consider occlusion and cluttering. Our work focuses on learning a 3D object model from depth images with multiple object instances, and successfully handling the occlusion issues.

With the assumption that no CAD object models are available (during either training or testing time), Wang et al. (2019b) uses a neural network to predict the 6D pose and size of previously unseen object instances. There are two key differences between our work and theirs: (i) their 6D pose and size estimation uses supervision of ground truth information for training data, which our system does not have. Second, their pose estimation is conditioned on region proposals and category prediction, which could make it difficult to estimate the pose of overlapped objects within the proposed bounding boxes. However, our 6D pose estimation does not depend on region proposals.

2.2.2 6D Pose Estimation

Deep neural networks have been used to perform pose estimation from images Do et al. (2018); Xiang et al. (2017) and point clouds Gao et al. (2018). Brachmann et al. (2016) present a generic 6D pose estimation system for both object instances and scenes which only needs a single RGB image as input. Tejani et al. (2014) propose Latent-Class Hough Forests for 3D object detection and pose estimation in heavily cluttered and occluded scenes. Wang et al. (2019a) present DenseFusion to fuse the color and depth data to extract pixel-wise dense feature embedding for estimating 6D pose of known objects from RGB-D images. Xiang et al. (2017) introduce a convolutional neural network, PoseCNN, for 6D object pose estimation from RGB images. Kehl et al. (2017) propose an SSD-style detector for 3D instance detection and full 6D pose estimation from RGB data in a single shot. Rad & Lepetit (2017) predict the 2D projections of the corners of an object's bounding box from color images, and compute the 3D pose from these 2D-3D correspondences with a PnP algorithm. Do et al. (2018) introduce an end-to-end deep learning framework, Deep-6D Pose, to jointly detect, segment, and recover 6D poses of object instances from a single RGB image. Gao et al. (2018) propose to directly regress a pose vector of a known rigid object from point cloud segments using a convolutional neural network. Sundermeyer et al. (2018) propose a self-supervised training strategy that enables robust 3D object orientation estimation using various RGB sensors while training only on synthetic views of a 3D object model. Sock et al. (2018) address recovering 6D poses of multiple instances in bin-picking scenarios using the depth modality for 2D detection, depth, and 3D pose estimation of individual objects and joint registration of multiple objects. Note that unlike our proposed method, all of these works require the object's 3D CAD model and use supervised learning from large datasets annotated with ground-truth 6D poses.

2.2.3 Instance Segmentation

Recent advances in instance segmentation on 2D images have achieved promising results. Many of those 2D instance segmentation approaches are based on segment proposals (He et al. (2017); Pinheiro et al. (2015); Dai et al. (2016)). DeepMask Pinheiro et al. (2015) learns to generate segment proposal candidates with a corresponding object score, and then classify using Fast R-CNN. Dai et al. (2016) proposes a multi-stage cascade to predict segment candidates from bounding box proposals. Mask R-CNN He et al. (2017) extends Faster R-CNN by adding a parallel branch to predict masks and class labels on top of RPN to produce object masks for instance segmentation. Inspired by these these pioneering 2D approaches, 3D instance segmentation (Wang et al. (2018); Yi et al. (2019); Pham et al. (2019)) on point clouds has also been attempted. Wang et al. (2018) propose a similarity group proposal network to predict point grouping proposals and a corresponding semantic class for each proposal for instance segmentation of point clouds. Pham et al. (2019) address the problems of semantic and instance segmentation of 3D point clouds using a multi-task point-wise network. Yi et al. (2019) propose a generative shape proposal network for instance segmentation. To the best of our knowledge, no previous work both learns a 3D object model and infers 6D pose from a 3D point cloud in an unsupervised fashion.

2.3 Low Quality Image Classification

We propose a simple yet effective deep feature transfer method for low resolution image classification. It is closely related to unsupervised learning of features and transfer learning. We review the related work as follows.

2.3.1 Unsupervised Learning of Features

Clustering has been widely used for image classification Caron et al. (2018); Yang et al. (2016); Ji et al. (2018). Caron et al. (2018) present a clustering method that jointly learns the parameters of a neural network and the cluster assignments of the resulting features. Yang et al. (2016) propose an approach to jointly learn the deep representations and image clusters by combining agglomerative clustering with CNNs and formulate them as a recurrent process. Ji et al. (2018) propose invariant information clustering relying on statistical learning by optimizing mutual information between related pairs for unsupervised image classification and segmentation.

2.3.2 Transfer Learning

It is commonly used in the scenario where the training and testing data distributions are different. Pan et al. (2008) learn a low-dimensional latent feature space to minimize the distance between distributions of the data in different domains. Glorot et al. (2011) study domain adaptation for sentiment classification using stacked denoising auto-encoders. Tommasi et al. (2010) present an SVM-based adaptation method to discriminatively select and weight the prior knowledge coming from different categories. Saenko et al. (2010) learn a regularized non-linear transformation in the context of object recognition to minimize the effect of domain-induced changes in the feature distribution. Chen et al. (2015) transfer knowledge stored in one previous network into each new deeper or wider network to accelerate the training of a significantly larger neural network. Yosinski et al. (2014) experimentally study the transferability of hierarchical features in deep neural networks. Azizpour et al. (2016) investigate the factors of transferability of a generic deep convolutional networks such as the network architecture, distribution of the training data, etc. Tzeng et al. (2015) learn a CNN architecture to optimize domain invariance and transfer information between tasks. Long et al. (2015b) propose a deep adaptation network architecture to match the mean embeddings of different domain distributions in a reproducing kernel Hilbert space. Guo et al. (2019) propose an adaptive fine-tuning approach to find the optimal fine-tuning strategy per instance for the target data. Readers can refer to Pan & Yang (2010) and the references therein for details about transfer learning.

Chapter 3

Towards Global Optimality in Deep Learning

3.1 Introduction

Global optimality is always desirable and preferred for optimization, which helps the generalization of learned models, in general. In fact very recently, there are substantial amount of work focusing on the theoretical analysis of the relations between global optimality and generalization in deep learning, such as Haeffele & Vidal (2017); Yun et al. (2018); Lin & Jegelka (2018); Liang et al. (2018); Du et al. (2019); Zou et al. (2018); Zhu et al. (2018b,c); Zhou et al. (2019b). All these papers above indicate that global optimality in deep learning improves the generalization.

From the algorithmic perspective, however, locating global optimality in DL is extremely challenging due to its high dimensionality and non-convexity. To our best knowledge, currently there are no DL solvers intentionally developed for this purpose, including stochastic gradient descent (SGD) Bottou et al. (2016), Adagrad Duchi et al. (2011), Adadelta Zeiler (2012), RMSProp Tieleman & Hinton (2012) and Adam Kingma & Ba (2014). Instead, regularization is often used to smooth the objective 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 Zhang et al. (2015); Chaudhari et al. (2017); Zhang & Brand (2017). These solutions, however, may not necessarily be the global optimum.

Inspired by the techniques of global optimization for nonconvex functions, we propose a novel approximation algorithm, *BPGrad*, which aims to locate the global optimality in DL via branch and pruning (BP) Sotiropoulos & Grapsa (2001). BP is a well-known algorithm developed to search for global solutions for nonconvex optimization problems. Its basic idea is to effectively and

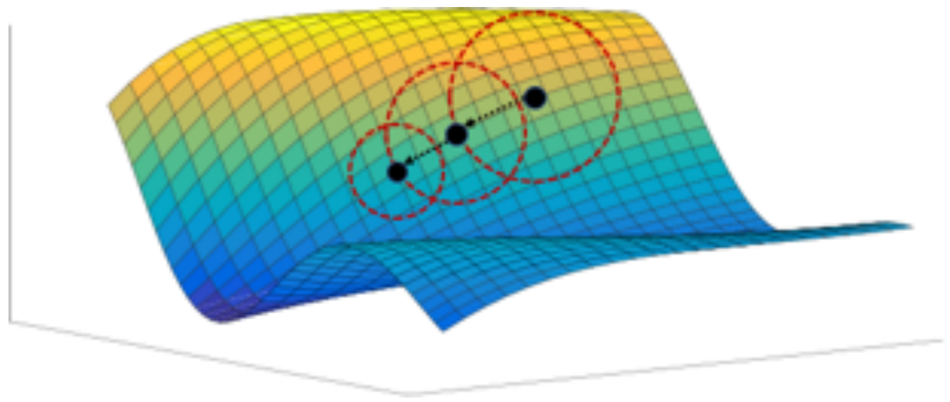


Figure 3.1: Illustration of the workflow of BPGrad, where each black dot denotes the solution at each iteration (*i.e.*branch), each directed dotted line denotes the current gradient, and each red dotted circle denotes the region wherein there should be no solutions achieving global optimality (*i.e.*pruning). BPGrad can automatically estimate the scales of these regions based on the function evaluation and the Lipschitz condition.

gradually shrink the gap between the lower and upper bounds of the global optimum by efficiently branching and pruning the parameter space. Fig. 3.1 illustrates the optimization procedure in BPGrad algorithm.

In order to branch and prune the parameter space, we assume that the objective functions in DL are Lipschitz continuous Eriksson et al. (2003) or can be approximated by Lipschitz functions, a fairly weak constraint as it always holds in DL Goodfellow et al. (2016). In fact, the Lipschitz condition provides us a natural way to estimate the lower bound in BP for locating the global optimum (see Sec. 3.2.3). It turns out as well that the Lipschitz condition can serve as regularization if needed, as illustrated in Fig. 3.2, to improve the generalization as demonstrated in Chaudhari et al. (2017). *In this sense, our BPGrad algorithm/solver essentially aims to locate global optimality in the smoothed objective functions for DL.*

From the optimization perspective, our algorithm shares similarities with the work Malherbe & Vayatis (2017) on global optimization of general Lipschitz functions (not specifically for DL). In Malherbe & Vayatis (2017) a uniform sampler is utilized to maximize the lower bound of the maximizer (equivalently minimizing the upper bound of the minimizer) subject to the Lipschitz condition. Convergence properties *w.h.p.* are derived. In contrast, our approach considers estimat-

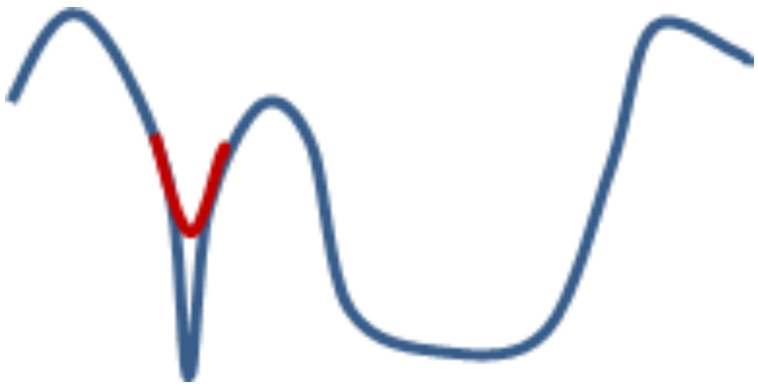


Figure 3.2: Illustration of Lipschitz continuity as regularization (red) to smooth a function (blue).

ing both the lower and upper bounds of the global optimum, and employs the gradients as guidance to effectively sample the parameter space for pruning. Theoretical analysis and experiments show that our algorithm can converge within finite iterations.

From the empirical solver perspective, our solver shares similarities with the work Koushik & Hayashi (2016) on improving SGD using the feedback from the objective. Specifically, Koushik & Hayashi (2016) tracks the relative changes in the objective 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 objective function to determine the learning rate adaptively but based on the rescaled distance between the feedback and the current lower bound estimation. Both theoretical and empirical justifications are established.

The main contributions of this paper are three-fold:

- We propose a novel *approximation algorithm*, BPGrad, towards the global optimization in DL. To our best knowledge, our approach is the *first* algorithmic attempt in this field.

- Theoretically we prove that BPGrad can converge to the global minima within finite iterations as the tractable lower and upper bounds allow it to prune the parameter space without missing.

- Empirically we propose a novel and efficient DL solver based on BPGrad to reduce the requirement of computation as well as a footprint in memory with better performance in object recognition, detection, and segmentation. We provide both theoretical and empirical justifications of our solver for preserving the theoretical properties of BPGrad.

This work substantially extends our conference publication Wu et al. (2018) from the following perspectives:

- A more comprehensive survey of the state-of-the-art is provided on the global optimality, popular first-order solvers in deep learning and branch-bound-pruning algorithms.
- We illustrate the effectiveness of the general BPGrad algorithm in one-dimensional space in Sec. 3.2.3.3. Moreover, to justify our method towards the global optimality, we apply the exact BPGrad algorithm and its efficient solver on one-dimensional functions and perform numerical analysis on a two-layer neural network optimization problem in Sec. 3.3.3.
- We provide more discussion and technical details on the robustness of parameter ρ and how the Lipschitz constant L could be estimated both theoretically and empirically in Sec. 3.4.1 and 3.4.2, respectively, which provides a useful insight on efficient hyper-parameter tuning.
- We conduct a comparative evaluation with SGD on object recognition, object detection, and object segmentation in Sec. 3.4.3, 3.4.4 and 3.4.5, respectively.

3.2 BPGrad Algorithm for Deep Learning

3.2.1 Notation

Let $\mathcal{X} \subseteq \mathbf{R}^d$ be the parameters space, $\mathbf{x} \in \mathcal{X}$ be the parameters of a given neural network, and $(\omega, y) \in \Omega \times \mathcal{Y}$ be a pair of a data sample ω and its associated label y. Let $\phi : \Omega \times \mathcal{X} \to \mathcal{Y}$ denote the nonconvex mapping function defined by the network, and f be the objective function with Lipschitz constant $L \geq 0$ to train the network. For all $\mathbf{x} = (x_1, \cdots, x_d) \in \mathbf{R}^d$, let $\|\mathbf{x}\|_2 = (\sum_{i=1}^{d} x_i^2)^{1/2}$ denote the standard ℓ_2-norm, ∇f be the gradient of f over parameters $\mathbf{x}$[1], $\nabla \tilde{f} = \frac{\nabla f}{\|\nabla f\|_2}$ be the

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

```
Algorithm 1: General BPGrad Algorithm
  Input  : objective function $f$ with Lipschitz constant $L \geq 0$, parameter $\rho \geq 0$, bounded
           feasible space $\mathscr{X}$
  Output: minimizer $\mathbf{x}^*$
1 Randomly initialize $\mathbf{x}_1 \in \mathscr{X}$;
2 while $\mathscr{X}_R \neq \mathscr{X}$ do
3     Draw sample $\mathbf{x}_{t+1} \sim \mathscr{X} \setminus \mathscr{X}_R(t)$;
4     $\mathscr{X}_R(t+1) \leftarrow \mathscr{X}_R(t) \bigcup \mathscr{B}(\mathbf{x}_{t+1}, r_{t+1})$;
5     $t \leftarrow t+1$;
6 end
7 return $\mathbf{x}^* = \mathbf{x}_{i^*}$ where $i^* \in \arg\min_{i=1,\cdots,t} f(\mathbf{x}_i)$;
```

normalized gradient (i.e., the direction of the gradient), and f^* be the global minimum.

3.2.2 Problem Statement

Given a deep network, the task of training process is to learn the parameters by minimizing the following objective function f:

$$\min_{\mathbf{x}\in\mathscr{X}} f(\mathbf{x}) \equiv \mathscr{E}_{(\omega\times y)\in\Omega\times\mathscr{Y}}\left[\mathscr{L}(y, \phi(\omega, \mathbf{x}))\right] + \mathscr{R}(\mathbf{x}), \tag{3.1}$$

where $\mathscr{E}$ is the expectation over data pairs, $\mathscr{L}$ is the loss function (*e.g.*cross entropy loss) to evaluate the differences between the ground-truth labels and the predicted labels of given data samples, and $\mathscr{R}$ is a form of regularization over parameters designed to prevent overfitting (*e.g.*weight decay via ℓ_2 regularization). We make assumptions throughout the paper as follows.

F1. f is lower-bounded by 0, *i.e.* $f(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathscr{X}$;

F2. f is differentiable for every $\mathbf{x} \in \mathscr{X}$;

F3. f is Lipschitz continuous, or can be approximated by Lipschitz functions, with constant $L \geq 0$.

3.2.3 Algorithm

Our BPGrad algorithm relies on the following assumption:

Definition 1 (Lipschitz Continuity Eriksson et al. (2003))**.** *A function $f : \mathscr{R}^m \to \mathscr{R}$ is* Lipschitz continuous *if there exists a Lipschitz constant $L \geq 0$ such that*

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|_2, \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathscr{X}. \tag{3.2}$$

3.2.3.1 Lower and Upper Bound Estimation

Consider the situation where samples $\mathbf{x}_1, \cdots, \mathbf{x}_t \in \mathscr{X}$ 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. (3.2) and simple algebra, we can obtain

$$\max_{i=1,\cdots,t} \left\{ f(\mathbf{x}_i) - L\|\mathbf{x}_i - \mathbf{x}^*\|_2 \right\} \leq f^* \leq \min_{i=1,\cdots,t} f(\mathbf{x}_i). \tag{3.3}$$

This provides us both the lower and upper bounds of the global minimum. The upper bound is tractable, however, the lower bound is *intractable*. The intractability comes from the fact that the optimal sample $\mathbf{x}^*$ is unknown, and thus makes the lower bound in Eq. (3.3) empirically unusable. To address this problem, we propose a novel tractable estimator, $\rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)$ $(0 \leq \rho < 1)$, for the lower bound. This estimator intentionally introduces a gap from the upper bound, which will be reduced by either decreasing the upper bound or increasing ρ. As proved in Thm. 1 (see Sec. 3.2.4), when the parameter space $\mathscr{X}$ 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.

3.2.3.2 Branch and Pruning

Based on our estimators, we propose a novel algorithm, called BPGrad, as illustrated in Alg. 1.

Branch: Alg. 1 conducts the branch operation to split the parameter space recursively by *sampling*. To achieve this goal, we need a mapping between the parameter space and the bounds. Considering the lower bound in Eq. (3.3), we propose sampling $\mathbf{x}_{t+1} \in \mathscr{X}$ based on the previous samples $\mathbf{x}_1, \cdots, \mathbf{x}_t \in \mathscr{X}$ so that it satisfies

$$\max_{i=1,\cdots,t} \left\{ f(\mathbf{x}_i) - L\|\mathbf{x}_i - \mathbf{x}_{t+1}\|_2 \right\} \leq \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i). \tag{3.4}$$

Note that an equivalent constraint has been used in Malherbe & Vayatis (2017).

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

$$\mathscr{X}_R(t) \overset{\text{def}}{=} \cup_{j=1,\cdots,t} \mathscr{B}\left(\mathbf{x}_j, r_j\right), \tag{3.5}$$

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

Pruning: 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. Parameter ρ controls such confidence or tolerance.

The implementation of Alg. 1 involves a sequential procedure of branching and pruning, which starts at an initial point x_1 by evaluating the function $f(x_1)$, calculating the radius $r_1 = \frac{1}{L}[f(x_1) - \rho \min_{i=1} f(x_1)]$, then at each step $t \geq 1$ to draw a new sample $\mathbf{x}_{t+1} \sim \mathscr{X} \setminus \mathscr{X}_R(t)$ which depends on the previous evaluations $\{(x_j, r_j, f(x_j))\}_{j=1,\cdots,t}$, and finally evaluate the objective function $f(x_{t+1})$ at this point. To illustrate its effectiveness in an interpretable domain, we have applied the Alg. 1 to solve a problem of controllable complexity in Sec. 3.2.3.3.

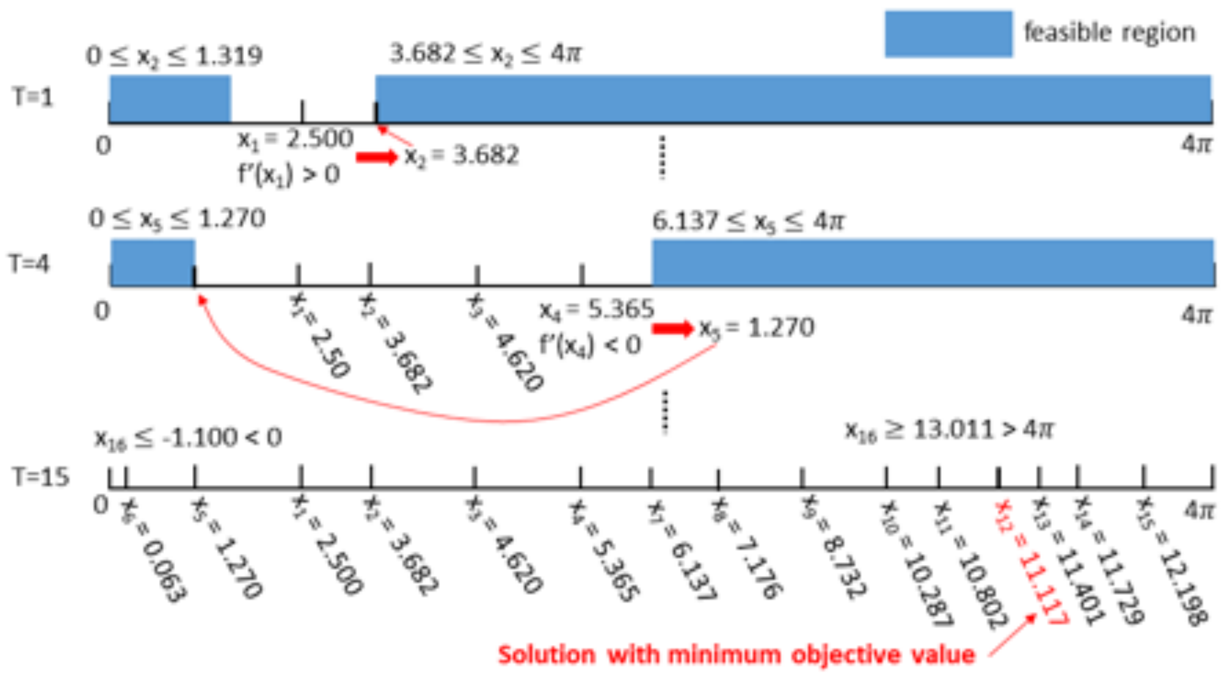


Figure 3.3: Illustration of BPGrad Alg. 1 in 1D space.

3.2.3.3 Illustration of Alg. 1 in One-Dimensional Space

In Fig. 3.3, we show an example of using Alg. 1 to solve the following nonconvex function:

$$f(x) = x\sin(x) + 15, \forall x \in [0, 4\pi]. \tag{3.6}$$

This function has a local and a global minimum at $x_{lmin} = 4.813$ and $x_{gmin} = 11.086$, respectively. The Lipschitz constant and initial point (at $T = 1$) are set to $L = 4\pi$ and $x_1 = 2.5$. Given x_1, using Alg. 1 we can obtain two feasible sets for x_2 as shown in Fig. 3.3. In branching, due to the gradient $f'(x_1) > 0$, we select the solution in the right set, leading to $x_2 = 3.682$. Then the infeasible set is pruned from the parameter space. Similarly, at iteration $T = 4$, since the gradient $f'(x_4) < 0$, the solution is in the left set with $x_5 = 1.27$. By alternating the procedures of branching and pruning, we eventually have searched all the parameter space within 16 iterations, and found the global minimum at $x_{12} = 11.117$. The error of this solution *w.r.t.*the ground-truth is 0.031.

Algorithm 2: BPGrad based Solver for Deep Learning

```
Input  : number of samples $T$, objective function $f$ with Lipschitz constant $L \geq 0$,
         momentum $0 \leq \mu \leq 1$, parameter $\rho \geq 0$
Output: minimizer $\mathbf{x}^*$
1 $\mathbf{v}_1 \leftarrow \mathbf{0}$, and randomly initialize $\mathbf{x}_1$;
2 for $t \leftarrow 1$ to $T-1$ do
3     $\mathbf{v}_{t+1} \leftarrow \mu \mathbf{v}_t - \frac{f(\mathbf{x}_t)-\rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)}{L} \cdot \frac{\nabla f(\mathbf{x}_t)}{\|\nabla f(\mathbf{x}_t)\|_2}$;
4     $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \mathbf{v}_{t+1}$;
5 end
6 return $\mathbf{x}^* = \mathbf{x}_T$;
```

3.2.4 Theoretical Analysis

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

$$\rho \min_{i=1,\cdots,t} f(\mathbf{x}_i) \leq f^* \leq \min_{i=1,\cdots,t} f(\mathbf{x}_i). \tag{3.7}$$

Proof. Since f^* is the global minimum, it always holds that $f^* \leq \min_{i=1,\cdots,t} f(\mathbf{x}_i)$. When $\mathscr{X}_R(t) \equiv \mathscr{X}$, 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 to the condition of $\mathscr{X}_R(t) \equiv \mathscr{X}$. We then complete the proof. □

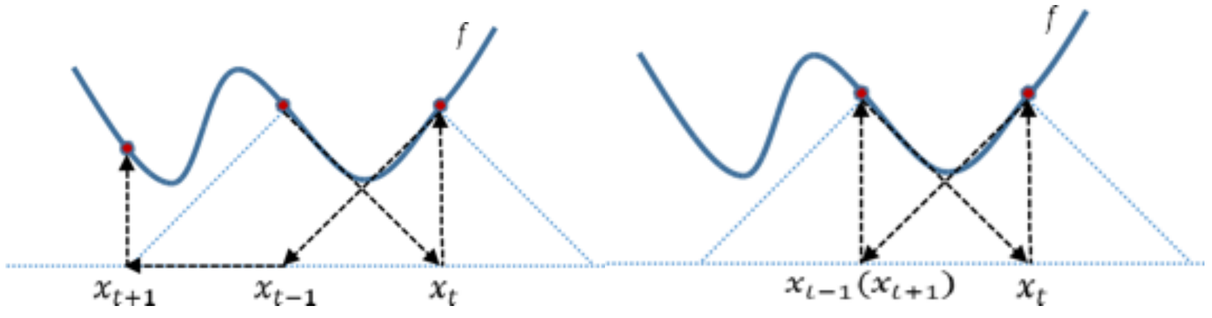
Corollary 1 (Approximation Error Bound)**.** *Given that both* $\min_{i=1,\cdots,t} f(\mathbf{x}_i) \leq \frac{\varepsilon}{1-\rho}$ *and* $\mathscr{X}_R(t) \equiv \mathscr{X}$ *hold, it is satisfied that*

$$\min_{i=1,\cdots,t} f(\mathbf{x}_i) - f^* \leq \varepsilon. \tag{3.8}$$

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

$$T \leq \left[\frac{2L}{(1-\rho) f_{\min}}\right]^d \cdot \frac{V_{\mathscr{X}}}{C}, \tag{3.9}$$

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



(a) Sampling using Alg. 1 (b) Sampling using Eq. (3.12)

Figure 3.4: 1D illustration of the difference in sampling between **(a)** using Alg. 1 and **(b)** using Eq. (3.12). Here the solid blue lines denote function f, the black dotted lines denote the sampling paths starting from $\mathbf{x}_{t-1} \to \mathbf{x}_t \to \mathbf{x}_{t+1}$, and each triangle surrounded by blue dotted lines denotes the RPS of each sample. It can be seen that (b) suffers from being stuck locally, while (a) can avoid the locality based on the RPS.

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

$$\|\mathbf{x}_{t+1} - \mathbf{x}_j\|_2 \geq \frac{1}{L}\left[f(\mathbf{x}_j) - \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)\right] \geq \frac{1-\rho}{L} \cdot \min_{i=1,\cdots,t} f(\mathbf{x}_i) \geq \frac{(1-\rho)f_{\min}}{L}. \tag{3.10}$$

This allows us to generate two balls $\mathscr{B}\left(\mathbf{x}_{t+1}, \frac{(1-\rho)f_{\min}}{2L}\right)$ and $\mathscr{B}\left(\mathbf{x}_j, \frac{(1-\rho)f_{\min}}{2L}\right)$ 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 $V_{\mathscr{X}}$, *i.e.*

$$V_{\mathscr{X}} \geq \sum_{t=1}^{T} V_{\mathscr{B}\left(\mathbf{x}_t, \frac{(1-\rho)f_{\min}}{2L}\right)} = C\left[\frac{(1-\rho)f_{\min}}{2L}\right]^d T. \tag{3.11}$$

Further using simple algebra we can complete the proof. □

3.3 Approximate DL Solver based on BPGrad

Although the BPGrad algorithm has nice theoretical properties for global optimization, we still need to solve the following problems in order to apply the Alg. 1 to deep learning applications.

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. The sampling involves the knowledge of previous samples, which incurs a significant amount of both computational and storage burden.

P3. Computing $f(\mathbf{x}_t)$ is time-consuming, especially for large-scale data.

To address *P1*, in practice we manually set the maximum numbers of iterations in Alg. 1. To address *P2* and simplify the sampling procedure, we propose to generate samples along the direction of the gradient with the following assumptions:

A1. $\mathscr{X}$ is sufficiently large where $\exists \eta_t \geq 0$ so that $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla \tilde{f}(\mathbf{x}_t) \in \mathscr{X} \setminus \mathscr{X}_R(t)$ always holds.

A2. $\eta_t \geq 0$ is always sufficiently small for local update.

A3. $\mathbf{x}_{t+1}$ can be sampled only based on $\mathbf{x}_t$ and $\nabla \tilde{f}(\mathbf{x}_t)$.

By imposing these assumptions upon the sampling procedure using Alg. 1, we define the step size as follows:

$$\eta_t = \frac{1}{L}\left[f(\mathbf{x}_t) - \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)\right]. \tag{3.12}$$

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

In summary, we present our BPGrad solver in Alg. 2 by modifying Alg. 1 for the sake of fast sampling as well as low memory footprint in DL, however, there is a risk of being stuck in local regions. Fig. 3.4 illustrates such scenarios using a 1D example. In (b) the sampling method falls into a loop because it does not consider the history of samples except for the 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.

3.3.1 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. (3.12). Then* $\mathbf{x}_{t+1}$ *satisfies Eq. (3.4) if it holds that*

$$\left\langle \mathbf{x}_i - \mathbf{x}_t, \nabla \tilde{f}(\mathbf{x}_t) \right\rangle \geq \frac{f(\mathbf{x}_i) - f(\mathbf{x}_t)}{L}, \forall i = 1, \cdots, t, \tag{3.13}$$

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

Proof. Based on Eq. (3.2), Eq. (3.12), and Eq. (3.13), we have

$$\|\mathbf{x}_i - \mathbf{x}_{t+1}\|_2 = \left(\|\mathbf{x}_i - \mathbf{x}_t\|_2^2 + \eta_t^2 + 2\eta_t \left\langle \mathbf{x}_i - \mathbf{x}_t, \nabla \tilde{f}(\mathbf{x}_t) \right\rangle\right)^{\frac{1}{2}} \geq \frac{1}{L}\left[f(\mathbf{x}_i) - \rho \min_{i=1,\cdots,t} f(\mathbf{x}_i)\right], \forall i = 1, \cdots, t, \tag{3.14}$$

which is essentially equivalent to Eq. (3.4) based on algebra. We then can complete the proof. □

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. (3.12). Then the condition in Eq. (3.13) can be rewritten as follows:*

$$\left\langle \mathbf{x}_i - \mathbf{x}_j, \nabla \tilde{f}(\mathbf{x}_j) \right\rangle \geq 0, \ 1 \leq \forall i < \forall j \leq t. \tag{3.15}$$

Discussion: Both Thm. 3 and Cor. 2 imply that, roughly speaking, our solver prefers sampling the parameter space along a path towards a single direction. However, the gradients in conventional backpropagation have little guarantee to satisfy Eq. (3.13) or Eq. (3.15) due to lack of such constraints in learning. On the other hand, momentum Sutskever et al. (2013) 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.

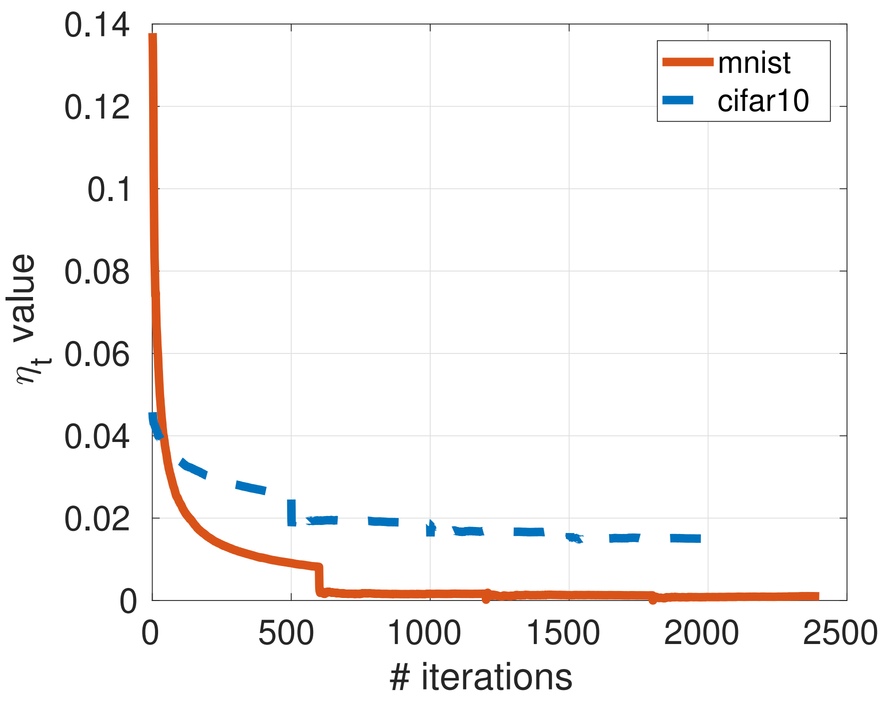


Figure 3.5: Plots of η_t on MNIST and CIFAR-10, respectively.

3.3.2 Empirical Justification

In this section, we discuss the feasibility of the assumptions *A1*-*A3* in reducing the computational and storage burden as well as preserving the properties towards global optimization in the applications of deep learning.

We utilize MatConvNet Vedaldi & Lenc (2015) as our testbed, and run our solver BPGrad in Alg. 2 to train the default networks in MatConvNet for MNIST LeCun (1998) and CIFAR-10 Krizhevsky et al. (2012), respectively, using the default parameters without explicit mention. Also we set $L = 15$ for MNIST and $L = 50$ for CIFAR-10 by default. For justification purpose we only run 4 epochs on each dataset, 600 and 500 iterations per epoch for MNIST and CIFAR-10, respectively. For more experimental details, please refer to Sec. 3.4.

Essentially assumption *A1* usually holds in deep learning due to its high dimensionality. Therefore, below we only focus on empirical justification of assumptions *A2* and *A3*.

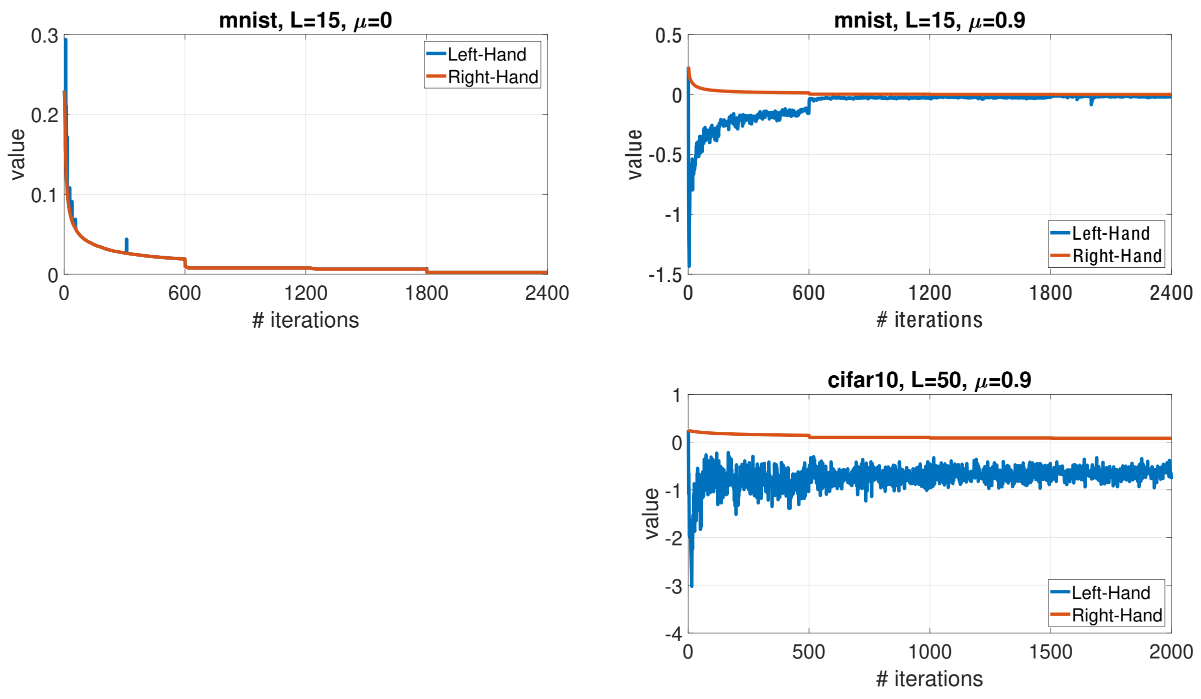


Figure 3.6: Comparison between LHS and RHS of Eq. (3.4) based on $\mathbf{x}_t$ returned by Alg. 2 using different values for momentum parameter μ.

3.3.2.1 Feasibility of A2

To justify this, we collect η_t's by running Alg. 2 on both datasets, and plot them in Fig. 3.5. In general, these numbers are indeed sufficiently small for local update based on gradients, and η_t decreases with the increase of iterations. This behavior is expected as the objective f is supposed to decrease as well *w.r.t.* the number of iterations. The value gap at the beginning on the two datasets is mainly induced by different L's.

3.3.2.2 Feasibility of A3

To justify this, we show some evidence in Fig. 3.6, where we plot the left-hand side (LHS) and right-hand side (RHS) of Eq. (3.4) based on $\mathbf{x}_t$ obtained from the 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. vanilla SGD update), the values on RHS are always no bigger than those on LHS correspondingly.

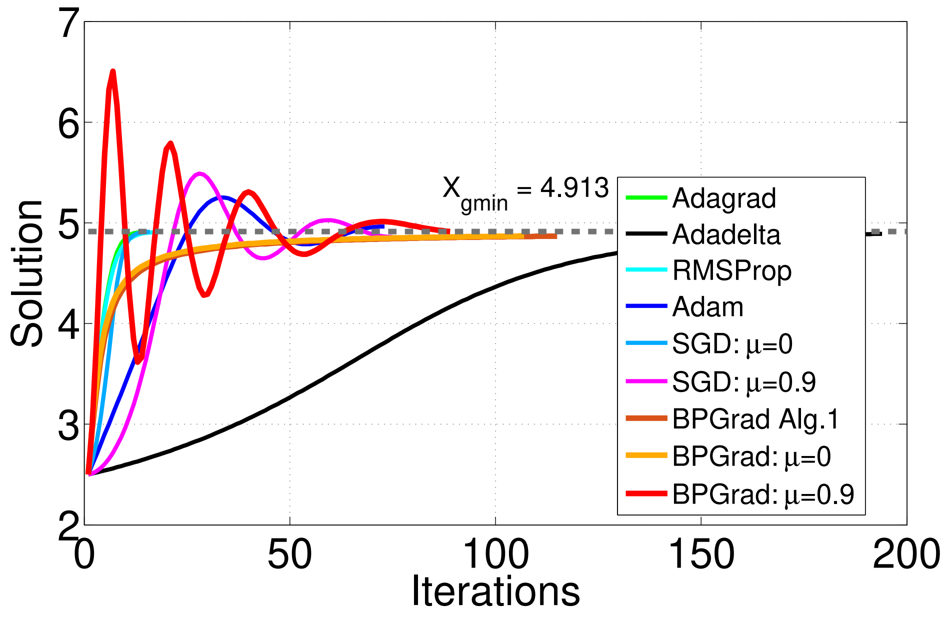


Figure 3.7: Trajectories of different solves on problems with known solutions (gray dashed line denotes the global solution for each function. The function $f_1 = x\sin(x) + 4.815, x \in [0, 8]$ has one global minimum at $X_{gmin} = 4.913$.

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 model updating, appear to be the only factor to make the samples of our solver satisfy Eq. (3.4). This also supports our claims in Thm. 3 and Cor. 2 about the relation between the model update and gradient in order to satisfy Eq. (3.4). More evidence have been provided by the experiments on MNIST and CIFAR-10 datasets in Sec. 3.4. Based on these evidence, it is safe to say that the assumption *A2* holds empirically when using sufficiently large values for momentum μ.

3.3.3 Convergence of BPGrad Algorithm and Solver

3.3.3.1 One-Dimensional Problems with Known Solutions

To explore the strength and weakness of the proposed approach in an interpretable domain, we first apply BPGrad in Alg. 1 and its approximate solver in Alg. 2 to nonconvex problems with limited

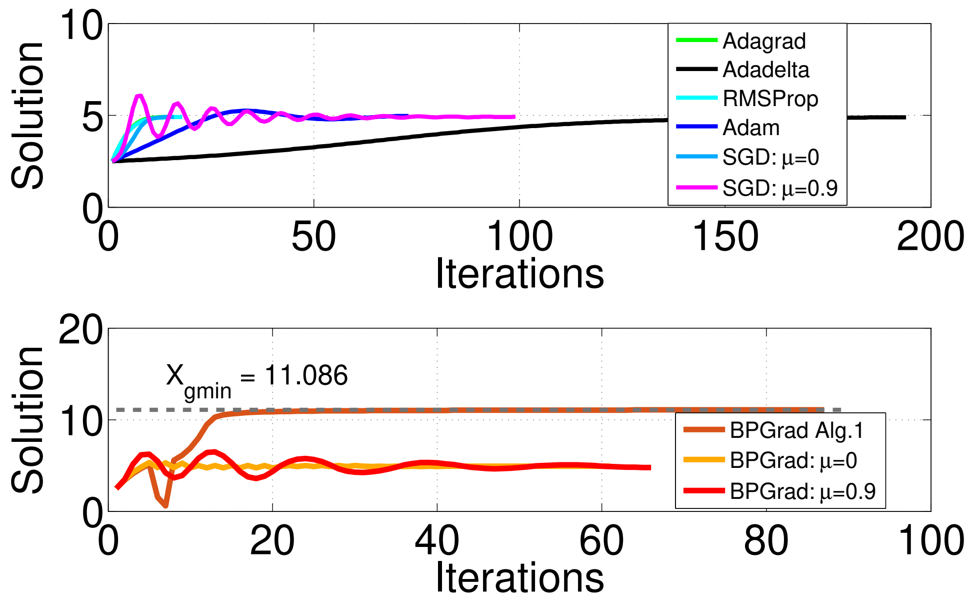


Figure 3.8: Trajectories of different solves on problems with known solutions (gray dashed line denotes the global solution for each function). The function $f_2 = x\sin(x) + 11.05, x \in [0, 4\pi]$ has a local minimum at $X_{gmin} = 4.913$, and a global minimum at $X_{gmin} = 11.086$.

complexity. The problem we consider is to search for the global minimum of the one-dimension sinusoidal function $f(x) = x\sin(x)$ with different constant offsets, which enables us to visualize the trajectories found by each solver. We perform a comparison with Adagrad, Adadelta, RMSProp, Adam and SGD.

The trajectories are shown in Fig. 3.7, Fig. 3.8, and Fig. 3.9. We use a grid-search to determine the best hyper-parameter setting for each solver (details can be found in the Appendix. A). We report the number of iterations that are needed to converge with a tolerance of $\varepsilon = 10^{-4}$ in terms of function values. We can observe that all the solvers find the global minimum of function f_1. However, only the BPGrad in Alg. 1 locates the global minimum of functions f_2 and f_3, respectively, while the other solvers are stuck at the local minimum. These observations empirically indicate that BPGrad in Alg. 1 is capable of reaching the global optimum.

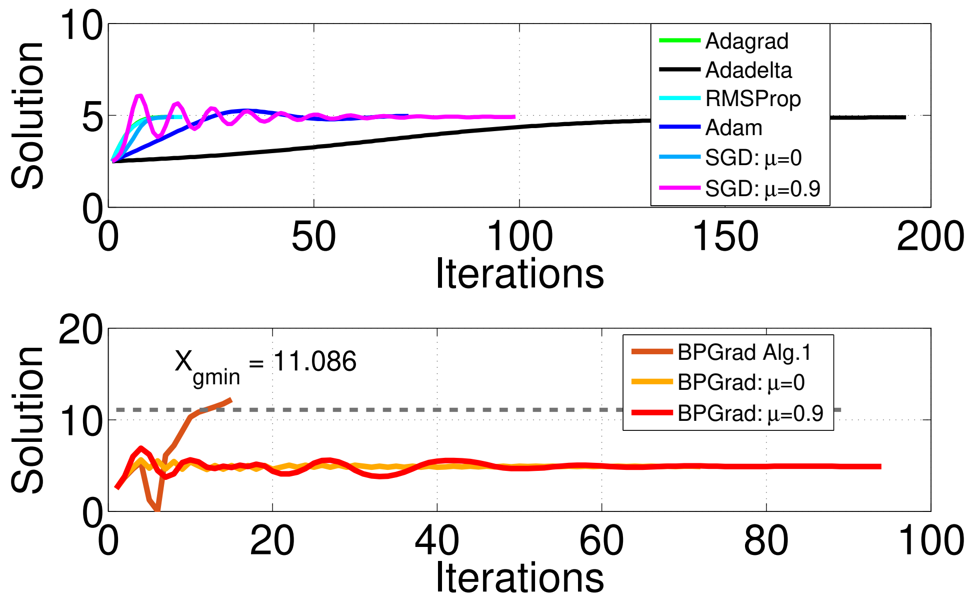


Figure 3.9: Trajectories of different solves on problems with known solutions (gray dashed line denotes the global solution for each function). The function $f_3 = x\sin(x) + 15, x \in [0, 4\pi]$ has the same local and global minimums as f_2, but it has a larger constant offset than f_2.

3.3.3.2 Rosenbrock Function Optimization

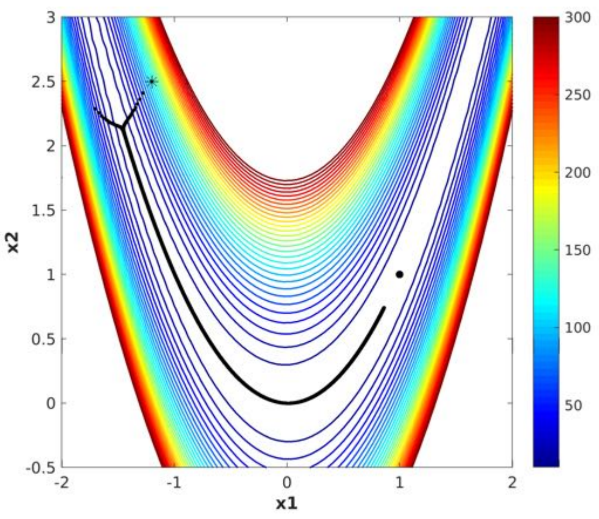
The famous Rosenbrock function is a popular test scheme for the gradient-based optimization algorithms. It is a unimodal function, and the global minimum lies in a narrow, parabolic valley. However, even though this valley is easy to find, it is difficult to converge to the minimum Picheny et al. (2013). To further investigate the generalization of our solver, we conduct numerical analysis on the Rosenbrock function and compare BPGrad against widely used DL optimizers, including SGD with and without momentum, Adam, RMSProp, Adadelta and Adagrad in MATLAB 2017a.

In this experiment, we consider the Rosenbrock function given below in two-dimensional form:

$$f(x_1, x_2) = (1 - x_1)^2 + 100 \cdot (x_2 - x_1^2)^2, \tag{3.16}$$

where its global minimum is $(x_1^*, x_2^*) = (1.0, 1.0)$.

We tune the parameters of each solver to achieve their best performance. Each optimizer starts


(a) SGD, $\mu = 0$

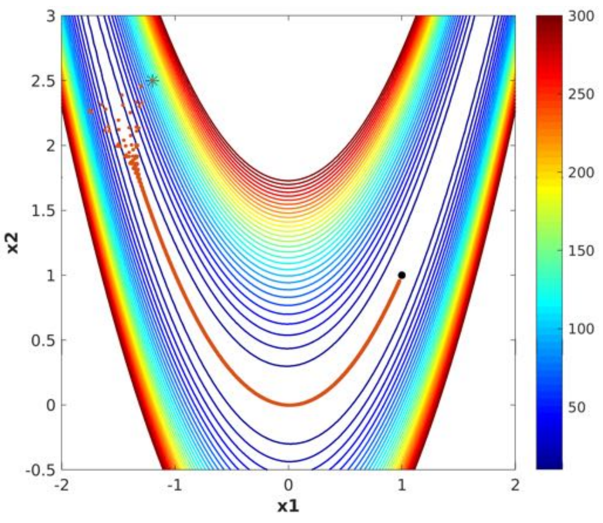
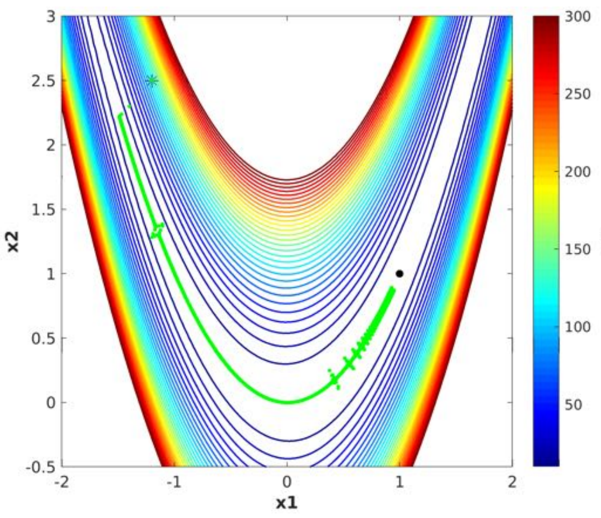

(b) SGD, $\mu = 0.9$

Figure 3.10: SGD on the Rosenbrock function in Eq. (3.16). The black asterisk and dot are the starting point and global optimum, respectively.

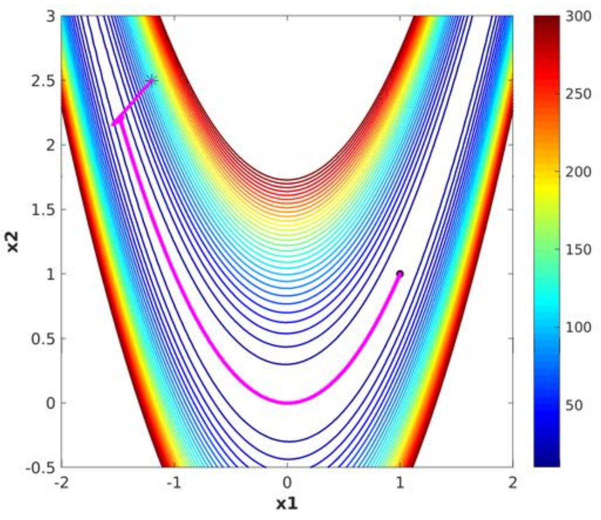
at the same initial point $(x_0, y_0) = (-1.2, 2.5)$, and is run for 4000 iterations with different learning rate and the best performance is plotted. For BPGrad, we set the Lipschitz constant $L = 500$ and with momentum $\mu = 0.9$ and no momentum $\mu = 0$. We use the same momentum for SGD, and set it to $\mu = 0.9$ with the global learning rate of 0.002, while for $\mu = 0$, the global learning rate is set to 0.001. For Adam and RMSProp, the global learning rate is 0.02, while for Adagrad it is 0.01. Adadelta does not require a global learning rate.

The convergence behaviors of all the solvers are shown in Fig. 3.10, Fig. 3.11 and Fig. 3.12. The convergence behaviors of our solver BPGrad is shown in Fig. 3.13. The convergent point for each solver is listed in Table 3.1. The results show that our solver BPGrad with momentum works better than the counterpart without momentum, achieving much tighter convergence. Further, our solver outperforms RMSProp, Adadelta, and SGD without using momentum ($\mu = 0$), and is comparable with Adam, Adagrad and SGD with momentum ($\mu = 0.9$). This demonstrates that our solver is generalized well, and it has the capability of locating global optimum for general optimization problem as well.

It is worthy of noticing that with $\mu = 0.9$ BPGrad moves much faster than all the other solvers



(a) RMSProp



(b) Adam

Figure 3.11: RMSProp and Adam on the Rosenbrock function in Eq. (3.16). The black asterisk and dot are the starting point and global optimum, respectively.

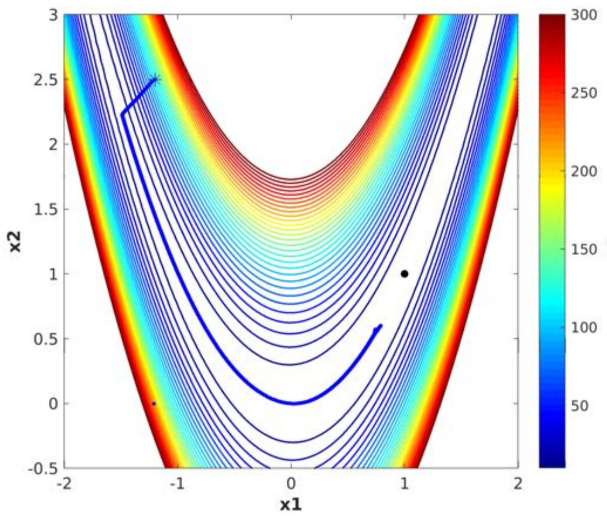
Table 3.1: Comparison of convergent points of our solver to the other conventional optimizers on the Rosenbrock function.

Solver / Param	SGD		RMSProp	Adam	Adadelta	Adagrad	BPGrad	
	$\mu = 0$	$\mu = 0.9$					$\mu = 0$	$\mu = 0.9$
x_1	0.86	0.98	0.95	1.0	0.78	0.99	0.77	0.97
x_2	0.74	0.96	0.87	0.99	0.57	0.98	0.60	0.95

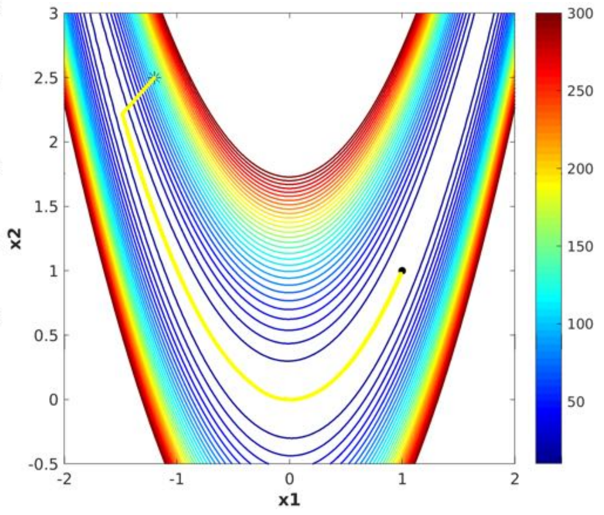
at the beginning towards the optimum, indicated by the discontinuity around the valley along the curve. This demonstrates that the branch-and-pruning procedure in our algorithm indeed is helpful to accelerate the convergence.

3.3.3.3 Two-layer Neural Network Optimization

Based on the empirical justification of convergence behavior of our solver in Sec. 3.3.2, in this section, we also consider a numerical demonstration of converging to the global optimum using our solver. Recently Li and Yuan in Li & Yuan (2017) proved theoretically that SGD can converge to the global minimum in polynomial time in two-layer neural networks with ReLU activation when the input data and network weight initialization follow Gaussian distributions.



(a) Adadelta



(b) Adagrad

Figure 3.12: Adadelta and Adagrad on the Rosenbrock function in Eq. (3.16). The black asterisk and dot are the starting point and global optimum, respectively.

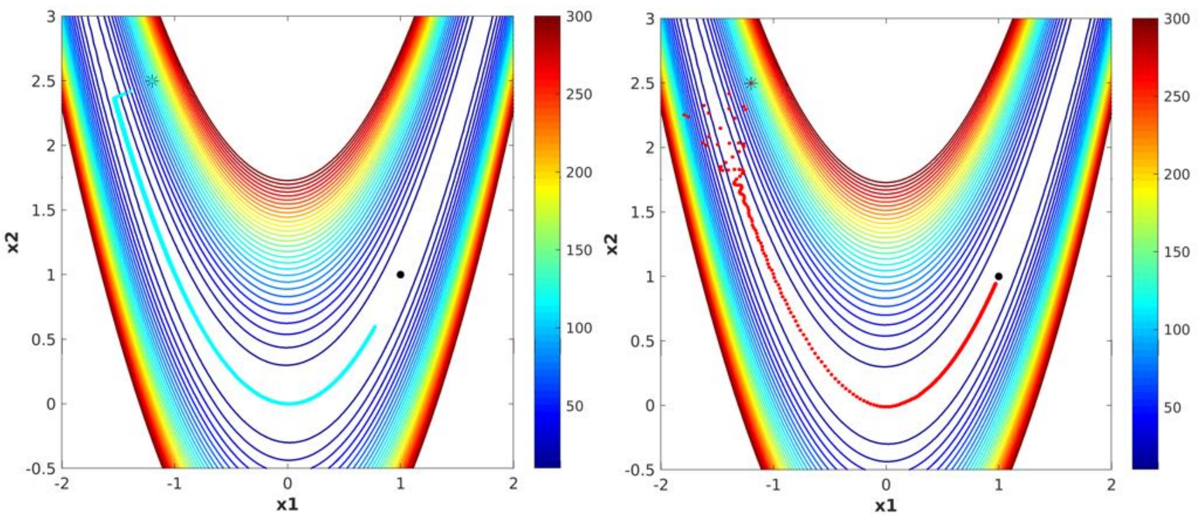
To demonstrate the convergence of our solver, we implement such a two-layer network in Li & Yuan (2017) as illustrated in Fig. 3.14 with $10,302$ parameters. We train the network using SGD and our BPGrad solver, respectively, with 20 epochs, batch size of 200 and momentum of 0.9.

We conduct grid search on learning rate (lr) and Lipschitz constant L for SGD and BPGrad, respectively. Then, we measure the Euclidean distance between the two learned network weights. We observe a marginal difference of 0.6 among the 10,302 dimensions. Numerically we can say that both the SGD and our solver converge to the same global minimum.

Given those strong evidences, we thus hypothesize that, with proper momentum, it is very likely that our solver in Alg. 2 will preserve the theoretical convergence properties of BPGrad algorithm in Alg. 1, and can converge to global optimum in DL.

3.4 Experiments

We utilize MatConvNet as our testbed, and employ its demo code as well as the default network architectures for different tasks. Since our solver can automatically determine the learning rates, we compare ours with SGD as well as another four widely used DL solvers with adaptive learning



(a) BPGrad, $\mu = 0$ (b) BPGrad, $\mu = 0.9$

Figure 3.13: BPGrad on the Rosenbrock function in Eq. (3.16). The black asterisk and dot are the starting point and global optimum, respectively.

rates, namely Adagrad, Adadelta, RMSProp, and Adam. We use grid-search to determine the best hyper-parameter settings (details can be found in Supplementary materials) for all the solvers and report their best performance.

3.4.1 Estimation of Lipschitz Constant L

We take the experiments on MNIST and CIFAR-10 datasets as examples to show the possibility of automatically tuning or reducing the searching space for manually tuning the parameter. For MNIST dataset, we take LeNet-5 as the network in our experiments. We randomly initialize the parameters (including weights and biases), and randomly feed a mini-batch into the network (*i.e.* feed-forward) to compute the objective value. Specifically, the filter weights are initialized with random numbers following a Gaussian distribution and the biases are initialized to be zero. The training samples are randomly shuffled and a mini-batch of 100 samples are selected from this shuffled training pool. We repeat this procedure for 600 times in one epoch on MNIST dataset, leading to 600 copies of initial network parameters as well as 600 objectives. Similarly we repeat it for 500 times with mini-batch size of 100 on CIFAR-10 dataset with similar network, leading to

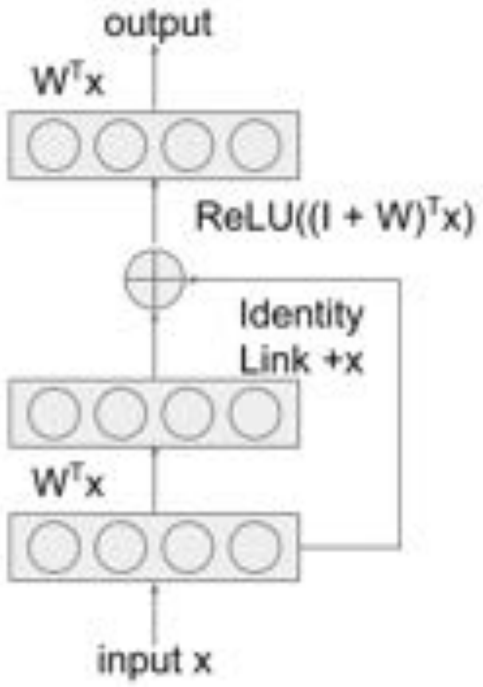


Figure 3.14: Illustration of two-layer networks.

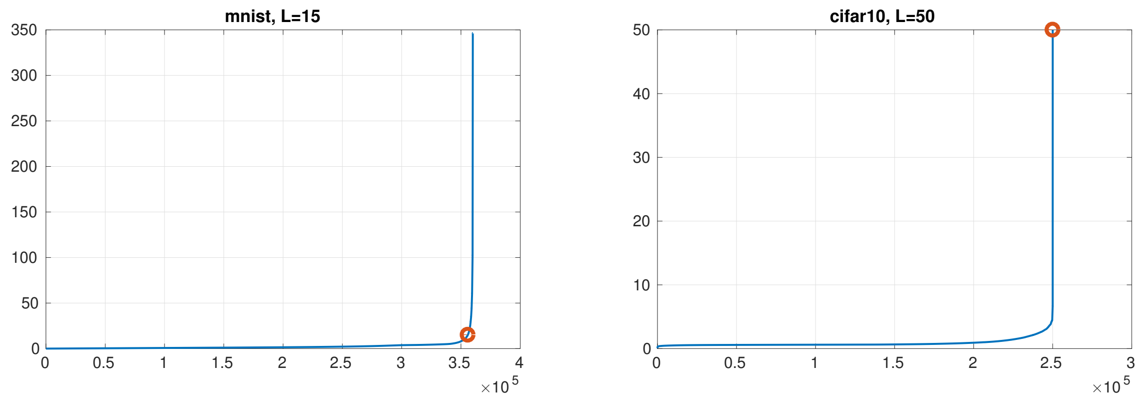
the same amount of initial network parameters and objectives.

Based on the definition of Lipschitz continuity in Eq. (1), we can compute L as follows:

$$L = \frac{|f(\mathbf{x}_1) - f(\mathbf{x}_2)|}{\|\mathbf{x}_1 - \mathbf{x}_2\|_2}, \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathscr{X}, \tag{3.17}$$

where $\mathbf{x_i}$ denotes one copy of initial network parameters, and $f(\mathbf{x_i})$ denotes the corresponding objective value.

By utilizing all the initial parameters as well as the objectives, we compute L based on Eq. (3.17) and plot the distributions of these L's in Fig. 3.15. It is evident that only a tiny portion of computed L's have relatively large values on both datasets. This behavior indicates that the surfaces induced by the objective functions in DL are in general quite smooth (*i.e.*without large jumps in the surface), and only in some regions the curvatures are high, where good models may exist. Therefore, this behavior verifies that our Lipschitz continuity assumption in deep learning can be satisfied *w.h.p.* empirically. In practice, we select the value as initial L that surpasses a threshold on the



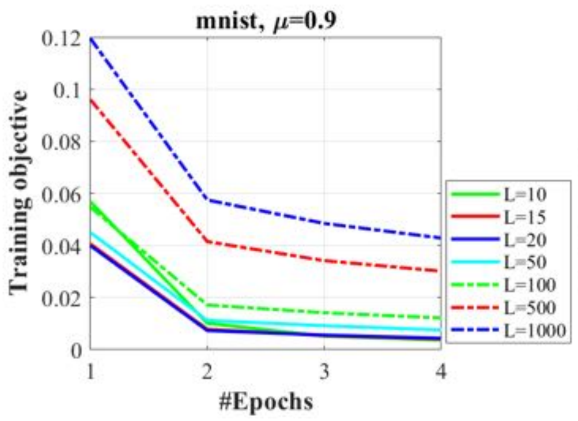
(a) MNIST (b) CIFAR-10

Figure 3.15: Distributions of computed Lipschitz constant L's in ascending order (y-axis denotes the value of L). Here the red dots denote the L values that are used in our experiments, (a) $L = 15$ on MNIST and (b) $L = 50$ on CIFAR-10.

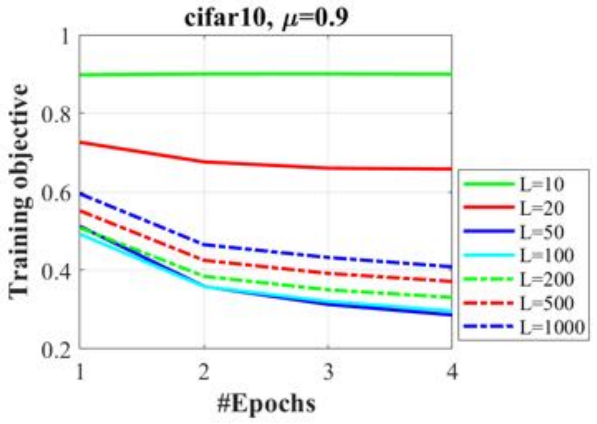
distribution of computed L's.

In addition, we observe in the experiments that the parameter L as Lipschitz constant is quite robust *w.r.t.*its performance, indicating that heavily tuning this parameter is unnecessary in practice. To demonstrate the robustness of Lipschitz constant L in our solver, we compare the training objectives of our solver by varying L in Fig. 3.16 (top row). To highlight the differences, we only crop and show the results in the first four epochs, and the remaining results have similar behavior. As we can see on the MNIST dataset, when L varies from 10 to 100, the corresponding curves are clustered. Similar clustering is observed as well on the CIFAR-10 dataset for L varies from 50 to 1000.

Moreover, we notice that the best L in Fig. 3.16, *i.e.* $L = 20$ on MNIST and $L = 50$ on CIFAR-10, respectively, tends to be within the range of the computed L's on each dataset shown in Fig. 3.15. This observation can be used to facilitate the parameter tuning procedure, as we do not necessarily consider any parameter that is far beyond the range, *e.g.* $L = 500$ or $L = 1000$ on MNIST. Therefore, we set $L = 15$ for MNIST and $L = 50$ for CIFAR-10, respectively, in our solver (see Sec. 3.4.3.1 and Sec. 3.4.3.2).

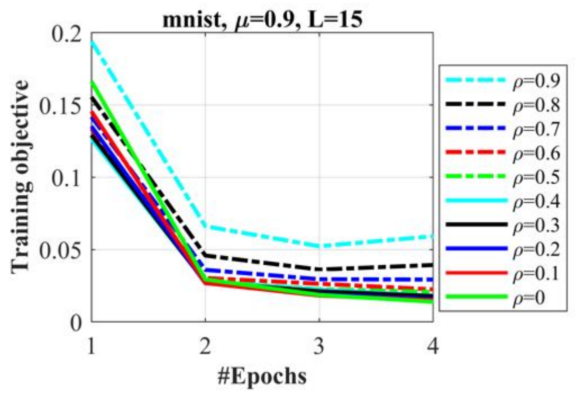


(a) MNIST

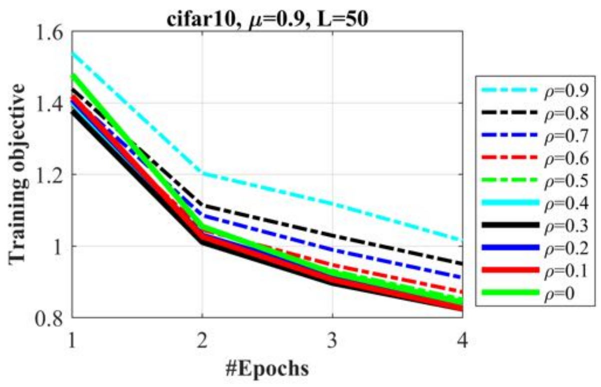


(b) CIFAR-10

Figure 3.16: Illustration of robustness of Lipschitz constant L in our solver.



(a) MNIST



(b) CIFAR-10

Figure 3.17: Illustration of robustness of parameter ρ in our solver.

3.4.2 Effect of ρ on performance

Similar robustness is also observed in the experiments for the parameter ρ related to the lower bound estimator in our solver. We compare the training objectives of our solver by varying ρ in a similar setting as L in Fig. 3.17. For the MNIST dataset, we set $L = 15$ and vary ρ from 0 to 0.9. As we can see from the result, when ρ varies from 0 to 0.5, the corresponding curves are clustered. Similar result is obtained on CIFAR-10 for ρ varies from 0 to 0.5. Therefore, in the following experiments, we set the initial value of $\rho = 0.1$ to make a trade-off between the estimation value

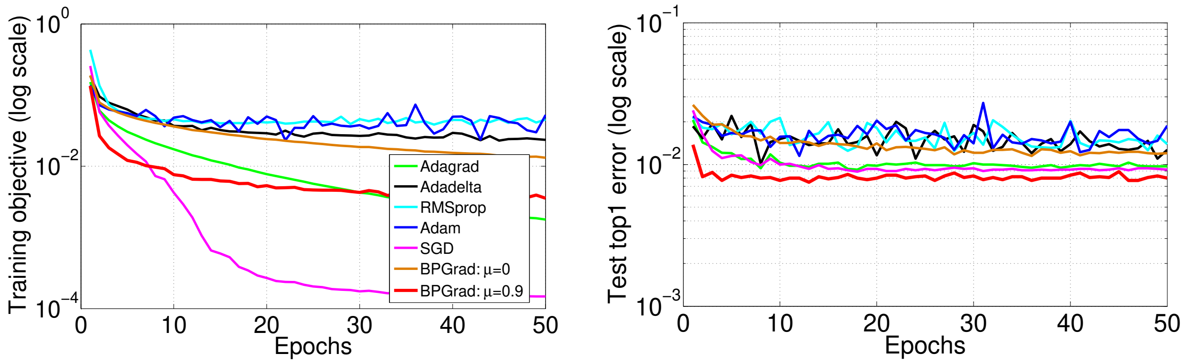


Figure 3.18: Comparison on **(left)** training objectives and **(right)** test top-1 errors for object recognition using LeNet-5 on MNIST.

of the lower bound and size of the removable parameter space for our BPGrad solver.

3.4.3 Object Recognition

In this section, we explore the use of BPGrad solver in object recognition with different CNNs on four benchmark datasets: MNIST LeCun (1998), distorted versions of MNIST Jaderberg et al. (2015), CIFAR-10 Krizhevsky et al. (2012) and ImageNet Russakovsky et al. (2015). For all the datasets, we follow the default implementation to train the individual CNN model on each dataset.

3.4.3.1 MNIST

The MNIST dataset consists of handwriting digits 0 to 9 which are gray images with a resolution of 28×28 pixels. There are $60,000$ training images and $10,000$ testing images in total in 10 classes labeled from 0 to 9. For this dataset, we train an individual LeNet-5 LeCun et al. (1998) model using each solver. For the details of network architectures please refer to the demo code. Specifically, for all the solvers, we train the network for 50 epochs with a mini-batch size 100, weight decay 0.0005, and momentum 0.9. In addition, we fix the initial weights for all solvers and the feeding order of mini-batches for a fair comparison. The global learning rate is set to 0.001 on MNIST for Adagrad, RMSProp, Adam, and SGD. Adadelta does not require a global learning rate.

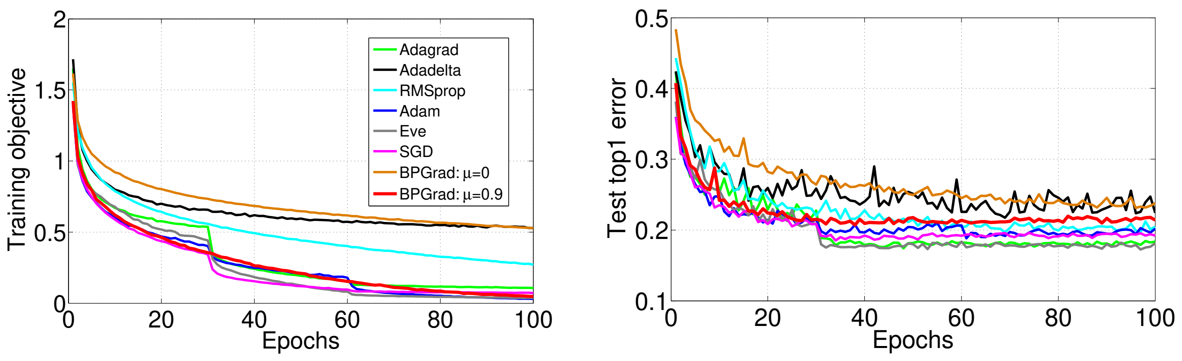


Figure 3.19: Comparison on **(left)** training objectives and **(right)** test top-1 errors for object recognition using network similar to LeNet on CIFAR-10.

The results are shown in Fig. 3.18. 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. It is clear that our solver with $\mu = 0.9$ works much better than its counterparts, achieving lower training objectives as well as a lower top-1 error at test time. This again provides evidence to support the importance of satisfying Eq. (3.4) in our solver to search for optimal solutions toward global minima.

The proposed Lipschitz continuity assumption can serve as regularization in deep learning. As can be seen in Fig. 3.15 on MNIST, the $L = 15$ used in Fig. 3.18 is much smaller than the maximumly computed L, making the surface of the approximate function much smoother. This eventually leads to a higher objective than SGD and Adagrad in Fig. 3.18. The test error of the BPGrad solver, however, is lower than SGD and Adagrad, owing to the functionality of regularization.

3.4.3.2 CIFAR-10

The CIFAR-10 dataset consists of 10 object classes of natural images with $50,000$ training images and $10,000$ test images, where the color image resolution is 32×32 pixels.

Similar to LeNet LeCun et al. (1998), for each solver in this experiment, we train an individual model for 100 epochs on this dataset, with a mini-batch size 100, weight decay 0.0005, and

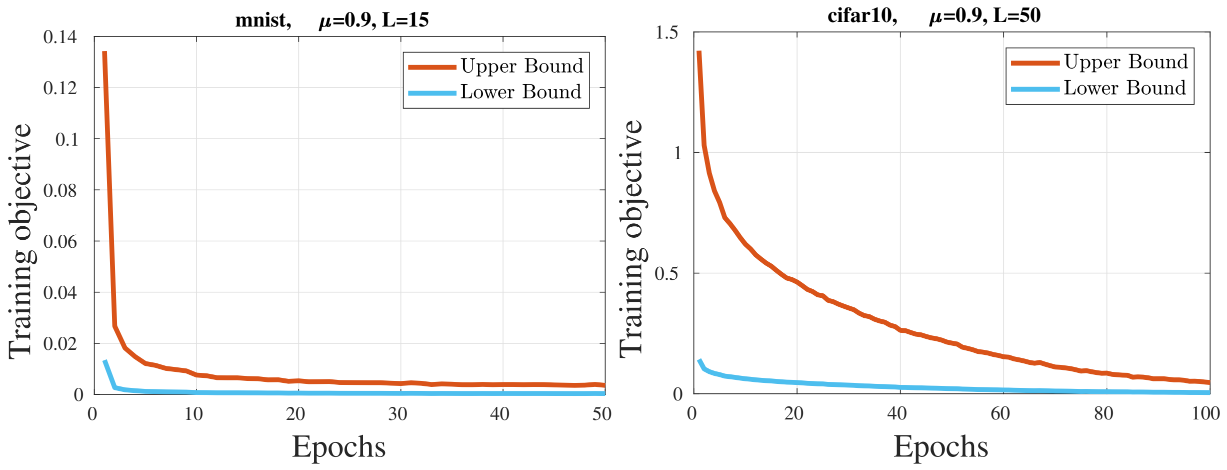


Figure 3.20: Plots of lower and upper bounds for (**left**) MNIST and (**right**) CIFAR-10 of our solver using LetNet.

momentum 0.9. In addition, we fix the initial weights for this network and the feeding order of mini-batches for a fair comparison. The global learning rate is set to 0.001 for RMSProp; but to 0.01 for Adagrad, Adam and Eve Koushik & Hayashi (2016), and it is reduced to 0.005 and 0.001 at the 31-st and 61-st epochs. The initial learning rate for SGD is 0.05, and it is multiplied by 0.1 at the 31-st and 61-st epochs. The Lipschitz constant L for our solver is set to 50 for this network.

The results are shown in Fig. 3.19. Our solver achieves the best performance in terms of training objective, but leading to a slightly inferior top-1 error at test time using LeNet. 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. In this experiment, we also compare with an extra solver, Eve, which was proposed in related work Koushik & Hayashi (2016) that empirically improves Adam with the feedbacks from the objective function. We can observe that our BPGrad solver achieves very competitive performance compared with Eve. Moreover, as reported in recent work Berrada et al. (2018), BPGrad outperforms Adagrad and Adam on CIFAR-10 dataset using both wide residual networks Zagoruyko & Komodakis (2016) and densely connected convolutional networks Huang et al. (2017), which further provides

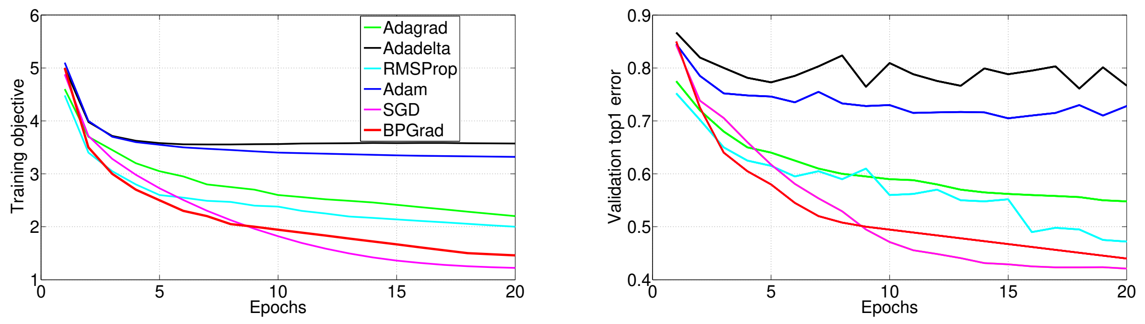


Figure 3.21: Comparison on **(left)** training objectives and **(right)** validation top-1 errors for object recognition using ImageNet ILSVRC2012.

Table 3.2: Top-1 recognition error (%) on ImageNet ILSVRC2012 dataset.

	Adagrad	Adadelta	RMSProp	Adam	SGD	BPGrad
training	49.0	71.6	46.0	70.0	**28.6**	<u>33.0</u>
validation	54.8	76.7	47.2	72.8	**42.1**	<u>44.0</u>

a solid evidence to support the advantage and robustness of our approach.

Finally, we provide empirical evidence on the convergence of lower and upper bounds estimations on the MNIST and CIFAR-10 datasets. As shown in Fig. 3.20, the global optimum is tightly bounded by our solver in finite number of iterations during training. The difference between the training objectives of lower and upper bounds is computed to evaluate the convergence in our solver. For MNIST, the gap between the lower and upper bounds is reduced from 0.121 to 0.003 in 50 epochs. Similarly, on CIFAR-10, the gap shrinks from 1.280 to 0.042. This provides an insight on how our solver is able to find the global solution using branch and pruning strategy. It samples (“branch") a candidate solution along the direction of the local gradient, then checks this branch against the estimated upper and lower bounds for the optimal solution, and removes (“pruning") the candidates that cannot produce a better solution than the best one found so far.

3.4.3.3 ImageNet ILSVRC2012

The ImageNet Russakovsky et al. (2015) dataset contains about 1.28M training images and 50K validation images among 1000 object classes. In this experiment, we employ the numbers of epochs in the demo files, since those values have been fine-tuned for different solvers. Following the demo code, we train the same AlexNet Krizhevsky et al. (2012) network on the ImageNet dataset 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 $\mu = 0.9$.

The results are shown in Fig. 3.21. We can observe that BPGrad converges faster than SGD before the 10-th Epoch at both training and test time, and achieves slightly inferior performance than SGD. However, we observe that BPGrad converges faster than all the other four competitors to achieve the lowest objective as well as the lowest top-1 error on the validation dataset. Specifically, our result is 3.2% lower than the second best adaptive solver, RMSProp, at the 20-th epoch, as listed in Table 3.2. The state-of-the-art top-1 error using AlexNet on ILSVRC2012 validation data is 42.6%[2], while our solver achieves **42.2%** top-1 error in 50 epochs, which is 0.4% lower than the state-of-the-art performance.

All the above experiments demonstrate the capability of the proposed solver BPGrad in training deep models for large scale object recognition.

3.4.4 Object Detection

Using the framework and source code of Fast RCNN Girshick (2015), we compare different solvers on the PASCAL VOC2007 dataset Everingham et al. (2007) with 20 object classes. The default object proposal approach is selective search Uijlings et al. (2013). 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, respectively, and use the default learning rates for the competitors. For our solver, we set $L = 100$.

[2] `http://www.vlfeat.org/matconvnet/pretrained/`

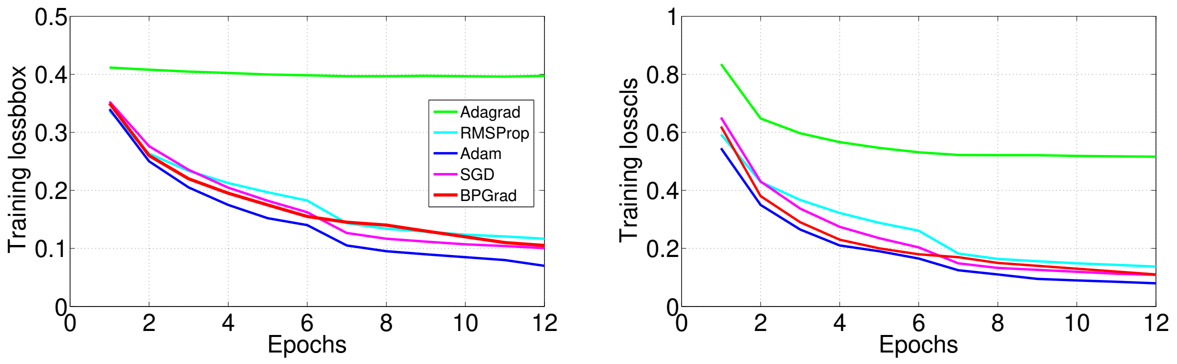


Figure 3.22: Loss comparison on VOC2007 trainval dataset for object detection, including **(left)** the regression loss using bounding boxes and **(right)** the classification loss.

Table 3.3: Average precision (AP, %) of object detection on VOC2007 test dataset.

	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
SGD	**72.0**	77.8	65.7	50.9	**40.0**	**78.1**	**78.2**	80.5	**41.3**	73.2	66.8	**78.5**	81.8	73.6	**66.8**	29.5	65.7	**69.4**	**75.0**	61.9	**66.3**
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

The training loss and average precision at the test time are shown in Fig. 3.22 and Table 3.3, respectively. Ours achieves slightly higher values in terms of training loss than Adam, however, our solver achieves on par performance with Adam, and SGD at test time on average.

3.4.5 Object Segmentation

Following the work Long et al. (2015a) 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 0.9 and weight decay 0.0005 on PASCAL VOC2011 Everingham et al. (2010b) segmentation set. For Adagrad, RMSProp, Adam and SGD, we find that the default parameters are able to achieve the best performance. For Adadelta, we tune its parameters with $\varepsilon = 10^{-9}$. The global learning rate for RMSProp is set to 10^{-5} and 10^{-4} for Adagrad and Adam, respectively. Adadelta does not require the global learning

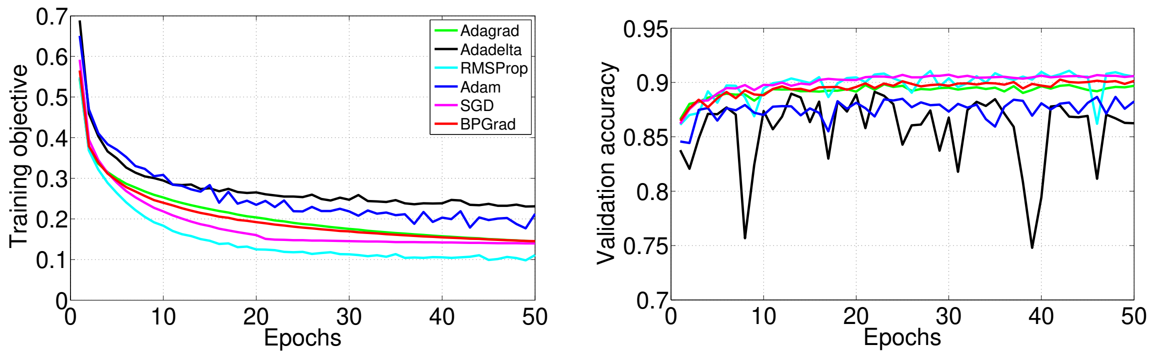


Figure 3.23: Segmentation performance comparison using FCN-32s model on (**left**) VOC2011 training and (**right**) validation datasets.

rate. For our solver, we set $L = 500$.

Fig. 3.23 shows the learning curve on the training and validation datasets, and Table 3.4 lists the test-time comparison results. In this experiment, our solver has very similar learning behavior as Adagrad, however, it achieves better performance at test time. The results demonstrate that our solver has the ability to learn robust and deep models for object segmentation. We can also observe from Fig. 3.23 that our solver is reliable as it exhibits smaller fluctuation over epochs in comparison with the competitors. 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 to learn robust and deep models for object segmentation.

In summary, we can observe that our BPGrad solver can not only achieve on par performance with SGD, but also eliminating the manual tuning of learning rate.

3.5 Conclusion

In this paper, we have proposed a novel approximation algorithm, *BPGrad*, towards searching for the global optimality in DL via branch and pruning based on the 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). The Lipschitz continuity not only provides us a way

Table 3.4: Numerical comparison on semantic segmentation performance (%) using VOC2011 test dataset at the 50-th epoch.

	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	<u>90.2</u>	71.0	73.9
Adam	50.9	87.2	66.4	68.2
SGD	**63.3**	**90.4**	<u>78.7</u>	**77.5**
BPGrad	<u>62.4</u>	89.8	**79.6**	<u>77.3</u>

to estimate the lower and upper bounds of the global optimality, it also serves as the regularization to further smooth the objective functions in DL.

We have theoretically proved that under some conditions our BPGrad algorithm can converge to the 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. A justification on preserving the properties of BPGrad is provided both theoretically and empirically. We have demonstrated the superiority of our BPGrad solver to several popular DL solvers for vision applications of object recognition, detection, and segmentation.

Based on the empirical analysis in the paper, we can see that our solver is capable of finding the solutions close to the global minima. However, with millions of parameters in deep neural networks, it is still an open problem to visualize empirically and analyze theoretically how an algorithm converges to the global or local minima. Moreover, It is an exciting direction of our future work continuing the adaptive solvers to train deep neural networks.

Chapter 4

Point Cloud Analysis

4.1 Introduction

Estimating the 6D poses (3D rotation and 3D translation) of rigid objects in 3D point clouds is an active research area with myriad real-world applications. Examples of such applications include, but are not limited to, robotic grasping and manipulation, virtual reality, augmented reality, and human-robot interaction Brachmann et al. (2016); Xiang et al. (2017); Sundermeyer et al. (2018); Kehl et al. (2017); Rad & Lepetit (2017); Tejani et al. (2014); Wang et al. (2019a). Methods proposed to solve this task typically make two strong assumptions (or simplifications) on the problem setup, such as (i) a 3D CAD model of the object is available, and (ii) a (sufficiently large) training set is available with annotated 6D poses of each object instance. In this paper, we address an extremely challenging variation of this task in which neither the CAD model nor the object poses are available during training or testing. Specifically, we aim to learn a high-fidelity 3D object models solely from depth images, without any ground-truth information about the object or the poses of the observed instances, while also estimating 6D poses of the object instances and their segmentations, all within a *single* deep learning framework.

The proposed task is very challenging for two key reasons. First, it is a "chicken-and-egg problem". On the one hand, it is difficult to estimate the correct pose if the object model is poor, and on the other hand, an accurate pose estimation is necessary to learn the object model (see the Ablation studies in Sec. 4.3.5). Second, we do not make any assumptions on the object shapes; as a result, the objects may contain symmetries, which can introduce suboptimal local solutions into the optimization landscape. Our general problem setup also brings in additional challenges such

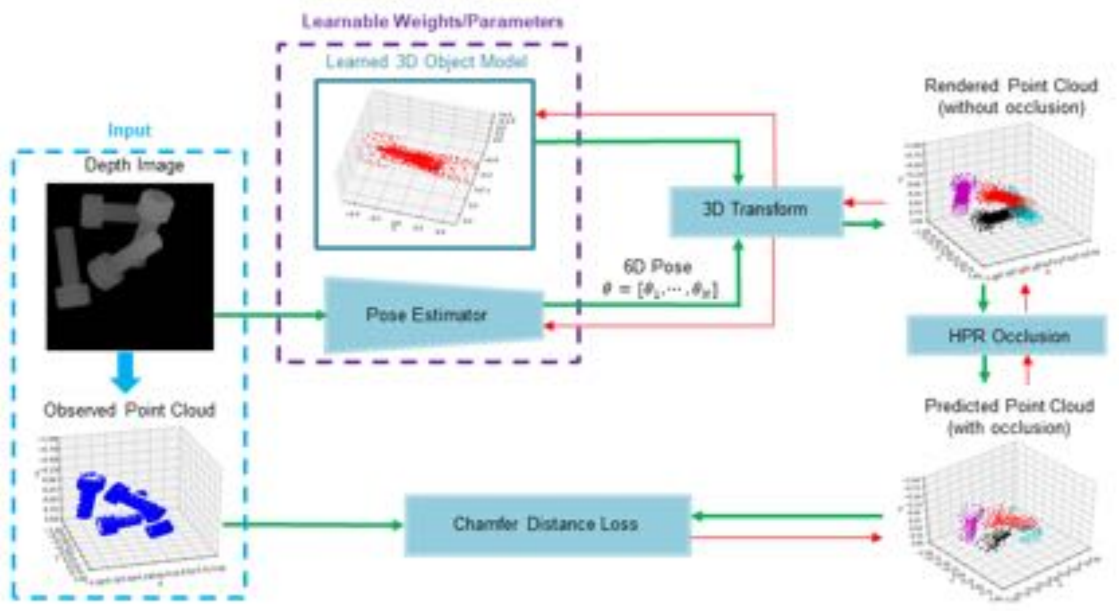

Learnable Weights/Parameters
Learned 3D Object Model
Input
Depth Image
Observed Point Cloud
Pose Estimator
6D Pose
$\theta = [\theta_1, \cdots, \theta_N]$
3D Transform
Rendered Point Cloud (without occlusion)
HPR Occlusion
Predicted Point Cloud (with occlusion)
Chamfer Distance Loss

Figure 4.1: Overview of our unsupervised approach of 3D object model learning and 6D pose estimation. The pose estimator takes as input a depth image, then regresses a 6D pose θ_i for each object instance i. The learned 3D object model is rendered using the pose. The predicted point cloud is obtained by removing the occluded points from the rendered point cloud using HPR occlusion. The chamfer distance between the predicted and the observed point cloud is used to drive the learning process. We denote forward propagation through the model using green arrows, and backpropagation using red. (Best viewed in color.)

as handling object self-occlusion and object-object occlusions.

To address this problem, we propose a novel deep learning framework, summarized in Fig. 4.1. Our proposed method takes as input a depth map image of a scene (a bin containing multiple instances of an object), and produces as outputs a 3D point-cloud model of the object and the 6D pose of every instance of the object. Our network consists of three key modules: (i) a 3D object model, initialized as a random point cloud, which when learned should capture the 3D structure of the underlying object; (ii) a pose estimator neural network that takes as input the depth map and produces a list of rigid transformations (3D rotations and translations), one for each instance of the object; (iii) a renderer that applies each rigid transformation to the 3D object model, combines the instances, and performs hidden point removal (HPR) Katz et al. (2007) to obtain a predicted point cloud in which occluded points have been removed; and (iv) a loss function that compares the predicted point cloud to the observed point cloud (obtained from the depth image) using chamfer distance as in Fan et al. (2017); Yang et al. (2018). Using point clouds allows us to learn high-fidelity object models, in contrast to voxels Choy et al. (2016); Tatarchenko et al. (2017); Tulsiani et al. (2017) or meshes Kanazawa et al. (2018); Kato et al. (2018) used in previous works. Our network is trained end-to-end from scratch, and all of the learned weights (the pose estimator network weights and the 3D object model) are optimized simultaneously via backpropagation.

We evaluate the proposed algorithm qualitatively on the task of 3D object model reconstruction for the single-instance case. We evaluate it quantitatively for multiple-instance cases on the task of instance segmentation from depth images of an unknown object (in which only the number of instances is known). The results demonstrate that in addition to performing better than or comparably to baseline methods on instance segmentation, our model also produces reasonable 3D point-cloud models of a variety of industrial objects.

To summarize, the main contributions of this work are three-fold as follows:

- We propose a novel task and generic unsupervised algorithm to jointly learn a complete 3D object model and estimate 6D poses of object instances in 3D point clouds.

- We incorporate occlusion reasoning into our framework, which enables it to learn full 3D models from depth maps (whose point clouds do not show occluded sides or occluded portions of objects), and handle learning symmetric object models via modifications to our loss function involving multiple rotation channels.

- We provide extensive experiments on unsupervised instance segmentation, demonstrating that in addition to learning a 3D object model, our method performs better than or comparably to standard baselines for instance segmentation.

4.2 Proposed Approach

In this section, we present our unsupervised approach for model learning and 6D pose estimation, and describe its application to instance segmentation in 3D point clouds. In Sec. 4.2.1, we describe our network architecture. Sec. 4.2.2 focuses on the core of our approach, the 3D object model and pose estimator network that are jointly learned through backpropagation. Sec. 4.2.3 describes how occlusion modeling is utilized to facilitate the learning of full 3D object model from point clouds obtained from depth images. Sec. 4.2.4 describes the unsupervised loss function. Finally, Sec. 4.2.5 explains how the predicted point cloud is used for instance segmentation in 3D point clouds.

4.2.1 Architecture Overview

Suppose the number of instances N is known, and that all instances are rigid 6D transformations of a single 3D shape. Without loss of generality, we define the complete 3D object model X using the form of a point cloud representation that describes the object surface. We denote $X = \{x_i | 1 \leq i \leq m\} \in \mathscr{R}^3$ as a set of m points with 3D positions. We denote 6D poses $\theta \in SE(3)$ as a homogeneous transformation matrix. In other words, a 6D pose $\theta = [R|t]$ consists of a 3D rotation $R \in SO(3)$ and a 3D translation $t \in \mathscr{R}^3$. We denote a viewpoint (camera center) as C. Since we estimate the 6D poses of objects at a given camera view, the poses are defined *w.r.t.* the camera coordinate frame.

Fig. 4.1 illustrates the overall proposed architecture. The learned portion of the system consists of a learned 3D object model and a pose estimator network. The learned object model is a point cloud in which the 3D point locations are learnable parameters. Given a depth image with N instances as input, the pose estimator outputs an estimated a 6D pose θ_i for each instance $i \in [1, \cdots, N]$. Next, the system generates a rendered point cloud by transforming N instances of the learned object model using the estimated 6D poses. The occlusion module then uses the HPR operator to remove points in the rendered point cloud that would not be visible from the given camera view. This occlusion modeling facilitates the learning of a complete 3D object model, because only unoccluded points will be compared with the observed point cloud. Finally, chamfer distance, an unsupervised loss, is used to evaluate the difference between the predicted point cloud and observed point cloud. The object model and the pose estimator are jointly updated by backpropagating the chamfer distance from the loss layer. We now discuss each module in detail.

4.2.2 Learning

3D Object Model Learning: We use a 3D point cloud $X \in \mathscr{R}^{m \times 3}$ to represent a learnable 3D object model. It is randomly and uniformly initialized within a cube of size ρ centered at the origin of the coordinate system, and is then updated during backpropagation. The estimated pose for each instance is applied to render the 3D object model. After rendering, we remove the hidden points using HPR occlusion, which is discussed in Sec. 4.2.3. We performed experiments with multi-layer perceptrons to generate a more structured point-cloud object model, but they did not improve the performance.

Pose Estimator: For the pose estimator, we use a ResNet-18 He et al. (2016) network as the backbone to input a depth image and estimate a 6D pose θ_i for each instance i. To represent 3D rotations, since the Euler-angle representation can result in gimbal lock[1] Zhou et al. (2019a), we instead use quaternions to represent rotations. The pose estimator regresses the 3D rotation and translation parameters for each instance. In total, we have a vector $\theta_i \in \mathscr{R}^{7 \times 1}$ to learn rotation and translation for each instance, where the first four parameters encode the quaternion representation of the rotation, and the remaining three parameters indicate the 3D translation. If each input depth map contains N instances of an unknown object, then the output of the pose estimator module is a $7N \times 1$ vector containing an estimated pose for each instance.

3D Transformation: From the estimated pose θ_i for instance i, we compute the rotation matrix R_i and the translation vector t_i. Then, we directly apply the 3D transformation to the object model X. We refer to this a 3D transformation with one rotation channel. However, during the experiments we found that for some elongated objects, the learned 3D object models are (incorrectly) symmetric across a plane perpendicular to the long axis (*e.g.* Bolt, Obj14, and Obj24 shown in Fig. 4.3(b)). We argue that it is because an incorrect estimated pose that is a $180°$ rotation (in a plane containing the long axis) away from the true pose would be a local minimum of the optimization. This would in turn cause the model learning to converge to a shape that is invariant with respect to such a $180°$ rotation. In order to prevent the model from getting stuck in this incorrect

[1]Certain configurations result in a loss of degree of freedom for the rotation.

local minimum of optimization, we propose to use two rotation channels in the 3D transformation module. It has benefited the model learning of several objects, as shown in Fig. 4.3(c). We now describe the details of using one vs. two rotation channels.

One-rotation-channel setting: For each instance i, we apply the rotation matrix R_i and translation vector t_i to the object model X to render the transformed point cloud using $X_{T_i} = R_i X + t_i$, where $i \in [1, \cdots, N]$.

Two-rotation-channels setting: For each instance i, in addition to using the rotation matrix R_i obtained from θ_i as the first channel's rotation matrix, $R_i^1 = R_i$, we also use R_i to obtain a second channel's rotation matrix: $R_i^2 = -R_i$. The intuition is that if one channel's rotation is near the incorrect local minimum described above, the other channel's rotation will be near the correct global minimum. The transformed point cloud of the first and second rotation channel is $X_{T_i}^1 = R_i X + t_i$ and $X_{T_i}^2 = -R_i X + t_i$, respectively. Note that although $-R_i$ is an orthogonal matrix, it is not a rotation matrix because it has determinant -1. Thus for an object X with no plane of symmetry, such as the camera head in Fig. 4.3(a), $X_{T_i}^2$ will not be a physically realizable rigid transformation of X. However, as long as an object X has a plane of symmetry, then $X_{T_i}^2$ will be equivalent to a rigid transformation of X. This is probably why using two rotation channels $\big($in Fig. 4.3(c)$\big)$ does not improve the learned object model for the camera head, but it does improve the learned model of the other objects (each of which has symmetry across a plane). For objects with planar symmetry, using two vs. one rotation channels is a trade-off between a high-fidelity object model and training time, because given N instances, there are 2^N possible combinations of rotation, which require longer training time.

4.2.3 HPR Occlusion

Since the observed point cloud is sampled from a given camera view C, some parts of objects are occluded (not visible). However, the rendered point cloud is generated using a complete 3D object model. To match the predicted point cloud with the observed point cloud, we need to remove from the rendered point cloud those points that would be occluded from the given camera view C. Thus, we integrate the HPR operator, which computes occlusions directly from point clouds, in our HPR occlusion module.

The occlusion module consists of two steps: inversion and convex hull construction. Given the transformed point cloud X_T and camera view C placed at the origin, we can create a D-dimensional sphere with radius R, centered at the origin C, and all the points in X_T are included in the sphere. Here, we perform inversion using spherical flipping, which is to reflect every point x_i inside the sphere along the ray from C to x_i to its image outside the sphere. After the construction of convex hull of the reflected points and camera view C, the point x_i is marked visible from C if its image point resides on this convex hull.

After removing the hidden points from the rendered point cloud, we obtain the predicted point cloud, which is used to calculate the chamfer distance from the observed point cloud. The HPR occlusion is able to improve the quality of the learned object model and facilitate the learning of a complete 3D object model, as discussed in Sec. 4.3.5.

4.2.4 Unsupervised Loss

The difference between the predicted point cloud and observed point cloud is calculated using chamfer distance. We assume the observed point cloud has n points. The predicted point cloud has m points. Let S denote the observed point cloud, and $\tilde{S}$ the predicted point cloud. For our loss function, we compute the reconstruction error for $\tilde{S}$ using the chamfer distance as in Yang et al. (2018):

$$\mathscr{L}(S,\tilde{S}) = \max\left\{ \frac{1}{|S|}\sum_{x\in S}\min_{\tilde{x}\in\tilde{S}}||x-\tilde{x}||_2, \frac{1}{|\tilde{S}|}\sum_{\tilde{x}\in\tilde{S}}\min_{x\in S}||\tilde{x}-x||_2 \right\}. \quad (4.1)$$

The term $\min_{\tilde{x}\in\tilde{S}}||x-\tilde{x}||_2$ enforces that every 3D point x in the observed point cloud has a matching 3D point $\tilde{x}$ in the predicted point cloud, and vice versa for the term $\min_{x\in S}||\tilde{x}-x||_2$. The max operation enforces that both directional distances between S and $\tilde{S}$ must be small. For the one-rotation-channel setting, we calculate the loss directly using Eq. (4.1).

Loss for two rotation channels: For two rotation channels, there are 2^N combinations of rotations given N instances, requiring a modified loss function:

$$\mathscr{L} = \sum_{i=1}^{2^N} w_i \cdot \mathscr{L}_i, \ where\ \ w_i = \frac{\exp\left(-\gamma \frac{\mathscr{L}_i}{\mathscr{L}_{sum}}\right)}{\sum_{j=1}^{2^N} \exp\left(-\gamma \frac{\mathscr{L}_j}{\mathscr{L}_{sum}}\right)}, \tag{4.2}$$

where $\mathscr{L}_{sum}$ is the summation of loss across 2^N channel combinations, $\mathscr{L}_i$ is the loss calculated using Eq. (4.1) for each channel combination i, w_i is softmin weight for the loss $\mathscr{L}_i$ of each combination, and γ is a variable that starts at 0 and is increased (with step size 0.002) at each iteration of learning. The goal of the softmin-weighted loss is to initially weight both rotation channels equally, but to gradually force the learning process to select a single winning rotation channel for each instance as training progresses.

4.2.5 Instance Segmentation in 3D Point Clouds

The learned object model and estimated 6D poses can be used for instance segmentation in 3D point clouds. For each point x_i in the observed point cloud S, we find its nearest neighbor in the predicted point cloud, and assign x_i the instance label of that nearest neighbor. In this way, we can segment all the instances in the observed point cloud, as shown in Fig. 4.2.

The labeling of different instances has to be invariant to permutations, *i.e.* it does not matter which specific label an instance is assigned, as long as the label is not the same as the other object instance labels. Following the evaluation metric based on counting pairs in clustering Ramage et al. (2009), we propose to evaluate the performance of instance segmentation using pairwise F1 score, as shown in Fig. 4.2, which is described in detail as follows. Let S_{gt} and S_{pred} denote the observed point cloud with ground-truth instance labels and with predicted instance labels, respectively. For a pair of points (a,b) in the observed point cloud, let (a_g,b_g) denote their instance labels from S_{gt}, and let (a_d,b_d) denote their instance labels from S_{pred}. We define the pairwise labels for S_{gt} as

$$L(a_g,b_g) = \begin{cases} 1, & if a_g = b_g. \\ 0, & otherwise. \end{cases} \tag{4.3}$$

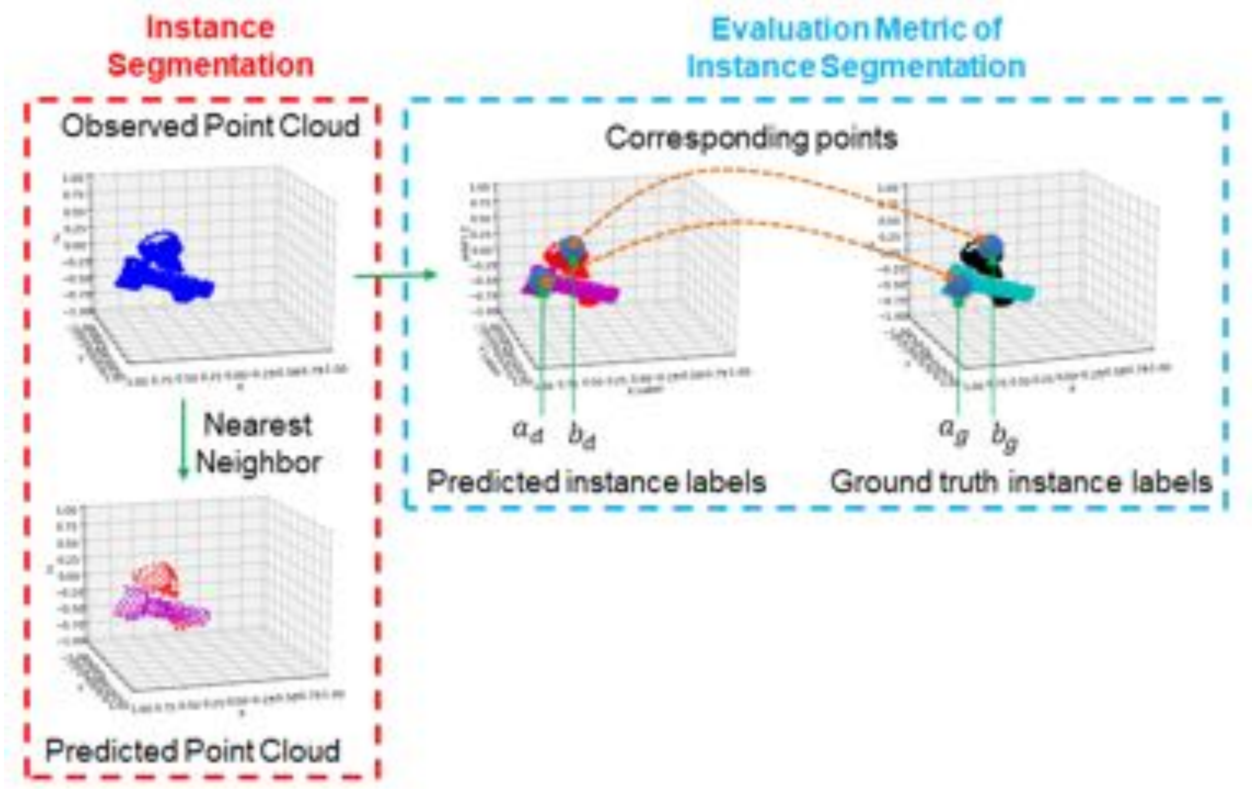


Figure 4.2: The pipeline of instance segmentation in 3D point clouds, and the evaluation metric of instance segmentation. For every point in the observed point cloud S, we assign to the point the instance label of its nearest neighbor in the predicted point cloud. The performance of instance segmentation is evaluated by calculating the pairwise F1 score.

Similarly, we define the pairwise labels $L(a_d, b_d)$ for S_{pred}. As shown in Table 4.1, for every possible pair of points (a, b), we compare the ground-truth pairwise label $L(a_g, b_g)$ to the predicted pairwise label $L(a_d, b_d)$, and consider the predicted label for the pair correct if $L(a_d, b_d) = L(a_g, b_g)$. We use this to compute the precision ($P = \frac{TP}{TP+FP}$) and recall ($R = \frac{TP}{TP+FN}$), and compute pairwise F1 score using $F1 = \frac{2P \cdot R}{P+R}$. We report the F1 score for instance segmentation (see Table 4.2).

Table 4.1: The definition of pair-wise labels.

	$L(a_g, b_g) = 1$	$L(a_g, b_g) = 0$
$L(a_d, b_d) = 1$	True Positive (TP)	False Positive (FP)
$L(a_d, b_d) = 0$	False Negative (FN)	True Negative (TN)

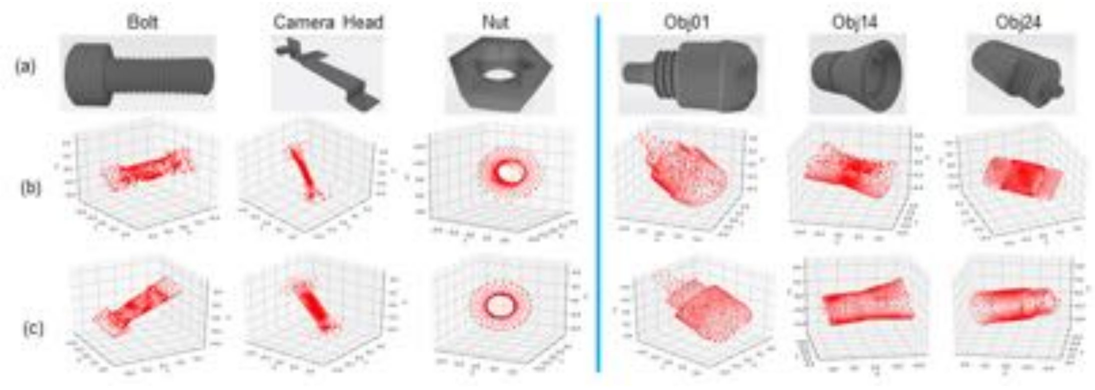


Figure 4.3: 3D object models that our method learned for six industrial objects from depth images containing one instance ($N = 1$). (a) True object CAD models, compared with the 3D object models learned using the setting of (b) one rotation channel or (c) two rotation channels.

4.3 Experiments

We first describe the datasets with synthetic depth images and the implementation details. We then present the learned 3D object models. We next evaluate the performance of instance segmentation. We finally discuss the ablation studies.

4.3.1 Datasets

We generate synthetic depth images by feeding the 3D CAD model of different objects to our customized simulator. The simulator uses PhysX-3.3.0 Nvidia (2019) to model the physics of dropping N identical objects one at a time into a bin, then renders the object to generate a depth image. We then extract the 3D point cloud directly from each depth image. To illustrate the robustness and generalization of our algorithm, we select six industrial objects: a bolt, camera head, and nut from our own library, plus three representative industrial objects (Obj01, Obj14, and Obj24) from the public T-LESS dataset Hodan et al. (2017). The 3D CAD models of the six objects are shown in Fig. 4.3(a).

For each object, we generated 5,000 depth images (and corresponding 3D point clouds) as our training dataset for each number of instances $N = 1, 2, 3, 4$ of each of the six objects. (In the scenario of bin picking, each bin contains multiple instances of a single object.) We select

500 images from the training dataset as our evaluation dataset. (When evaluating unsupervised learning, it is common for the evaluation set to be a subset of the training set.)

We qualitatively evaluate the reconstructed 3D object model and numerically evaluate the performance of instance segmentation on the evaluation dataset. The ground-truth instance labels of the depth images and point clouds are only used during *evaluation* of instance segmentation, not during training, because the training is unsupervised. We evaluate the performance of instance segmentation using the pairwise F1 score (see Section 4.2.5).

4.3.2 Implementation Details

We use PyTorch Paszke et al. (2017) as the framework to implement our algorithm. The pose estimator consists of a ResNet-18 He et al. (2016) to regress the 3D transformation parameters θ_i, for $i = 1, \cdots, N$. The resolution of depth images is 224×224. We replicate the depth image into three channels as input to the ResNet-18. No data augmentation is used during training. We select Adam Kingma & Ba (2014) as the optimizer with default parameters. We choose initial learning rate $l_m = 1e^{-3}$ to learn the 3D object model, and $l_r = 1e^{-4}$ for the ResNet-18 pose estimator. The learning rate is decreased by a factor of 0.31 after every 50 epochs. For each object, a separate model is trained for each of $N = 1, 2, 3$, and 4 instances. Each model is trained for 250 epochs, with batch size 250.

4.3.3 Learned 3D Object Model

Fig. 4.3(b) and (c) show the 3D object models learned from single instance ($N = 1$) data using the one-rotation-channel setting and the two-rotation-channel setting, respectively, for six industrial objects. Our method is able to learn reasonable high-fidelity 3D object models using one rotation channel for the Nut and Obj01. However, it learned (incorrect) symmetric object models for the Bolt, Obj14, and Obj24. Using two rotation channels enables the method to reconstruct high-fidelity 3D object models for these objects that more closely match the object CAD models shown in Fig. 4.3(a). As explained in Section 4.2.2, the Camera Head does not have a plane of symmetry,

Table 4.2: Instance segmentation comparison of baseline clustering methods with our method. F1 scores of instance segmentation on the validation dataset are shown. The complexity of instance segmentation increases with the number of instances N in each input depth image. Ours (v1) is one-rotation-channel setting, and Ours (v2) is two-rotation-channels setting.

# Instances	Methods	Bolt	Camera Head	Nut	Obj01	Obj14	Obj24	Ave. on 6 objects
N=2	K-means	0.89	0.89	0.99	0.97	0.98	0.98	0.95
	Spectral Clustering	0.96	0.91	0.90	0.91	0.87	0.94	0.90
	Ours (v1)	**0.99**	**0.94**	**1.0**	**1.0**	**1.0**	**1.0**	**0.98**
	Ours (v2)	0.96	0.93	0.99	0.97	**1.0**	**1.0**	**0.98**
N=3	K-means	0.79	0.74	**0.98**	**0.93**	0.96	0.96	0.89
	Spectral Clustering	0.90	0.83	0.96	0.91	0.81	0.89	0.88
	Ours (v1)	0.98	**0.93**	**0.98**	**0.93**	**0.98**	**0.99**	**0.97**
	Ours (v2)	**0.99**	0.84	**0.98**	**0.93**	0.96	**0.99**	0.95
N=4	K-means	0.71	0.65	**0.97**	**0.90**	**0.92**	**0.94**	0.85
	Spectral Clustering	0.85	0.76	0.94	0.86	0.77	0.85	0.84
	Ours (v1)	0.81	**0.85**	0.96	**0.90**	0.87	0.86	0.88
	Ours (v2)	**0.88**	0.84	0.96	0.89	0.91	0.85	**0.89**

which may explain why our two-rotation-channel setting did not improve the learned model for that object. On the whole, the learned 3D object models in Fig. 4.3 demonstrate the capability of our algorithm to reconstruct a complete 3D object model without any supervision from an object CAD model or ground truth pose.

4.3.4 Instance Segmentation Results

As the task we are solving is new, we could not find proper baselines based on deep neural networks. Therefore, we quantitatively compare the performance of instance segmentation of our unsupervised algorithm with two non-deep clustering baselines: K-means Pedregosa et al. (2011) and Spectral Clustering Pedregosa et al. (2011). As explained in Sec. 4.2.5, we use the predicted point cloud generated by our method to perform instance segmentation on each input 3D point cloud.

We report the pairwise F1 scores in Table 4.2. As we can see, with $N = 2$ and 3 instances, our algorithm significantly outperforms the two baselines across all six objects. K-means is slightly better than ours for $N = 4$ instances on the Nut, Obj01, Obj14 and Obj24. However, our method outperforms K-means on the Bolt and Camera Head by about 10% and 20%, respectively. We

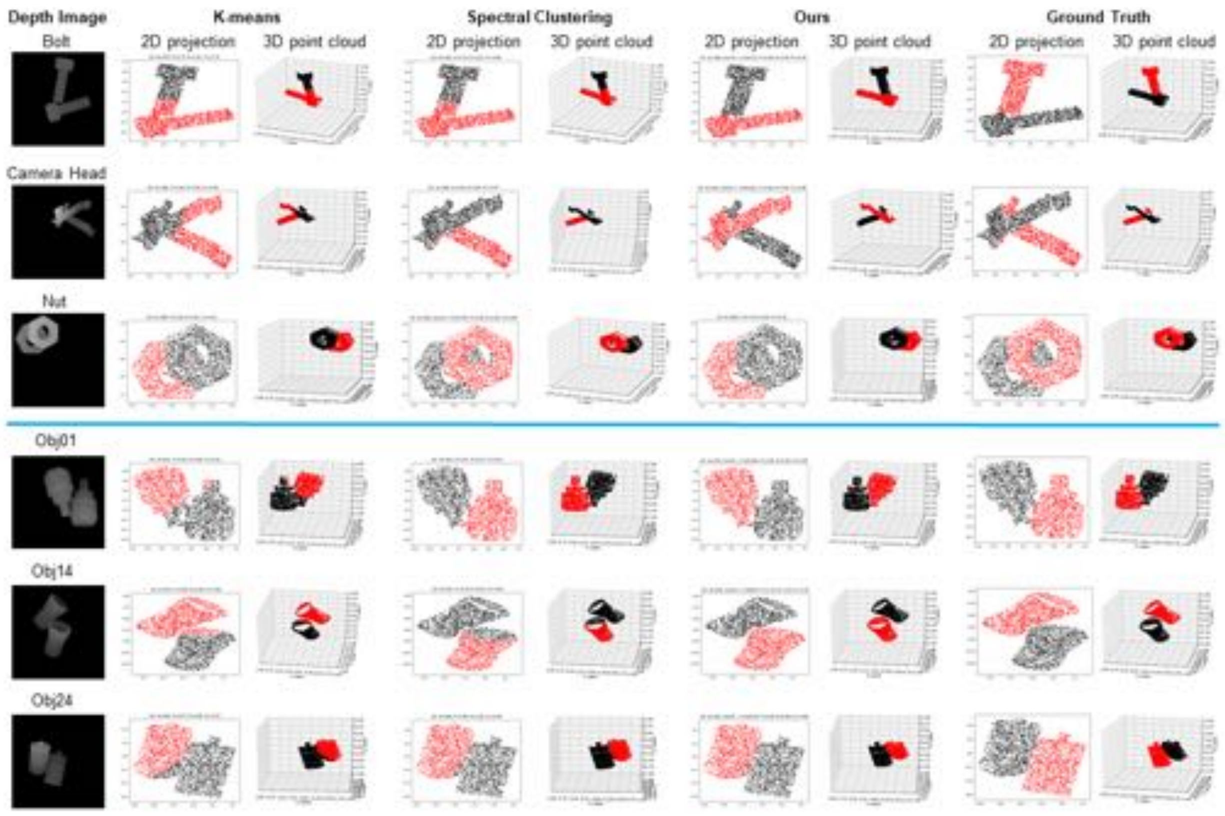


Figure 4.4: The qualitative results of instance segmentation of 2D projection and 3D point clouds with $N = 2$ instances.

believe that this is because K-means is good at clustering the objects with roughly spherical shapes but not more elongated objects. In contrast, our algorithm is able to handle both types of object shapes. Fig. 4.4, Fig. 4.5 and Fig. 4.6 show qualitative results of instance segmentation in 2D projection and 3D point clouds for $N = 2, 3$ and 4 instances, respectively, obtained using the single-rotation-channel setting.

Note that our algorithm is addressing a much more difficult task than simply instance segmentation. Unlike the two baseline algorithms, our algorithm does not just segment the point clouds into instances, but also simultaneously learns a high-fidelity 3D object model and estimates the 6D pose of objects in an unsupervised manner.

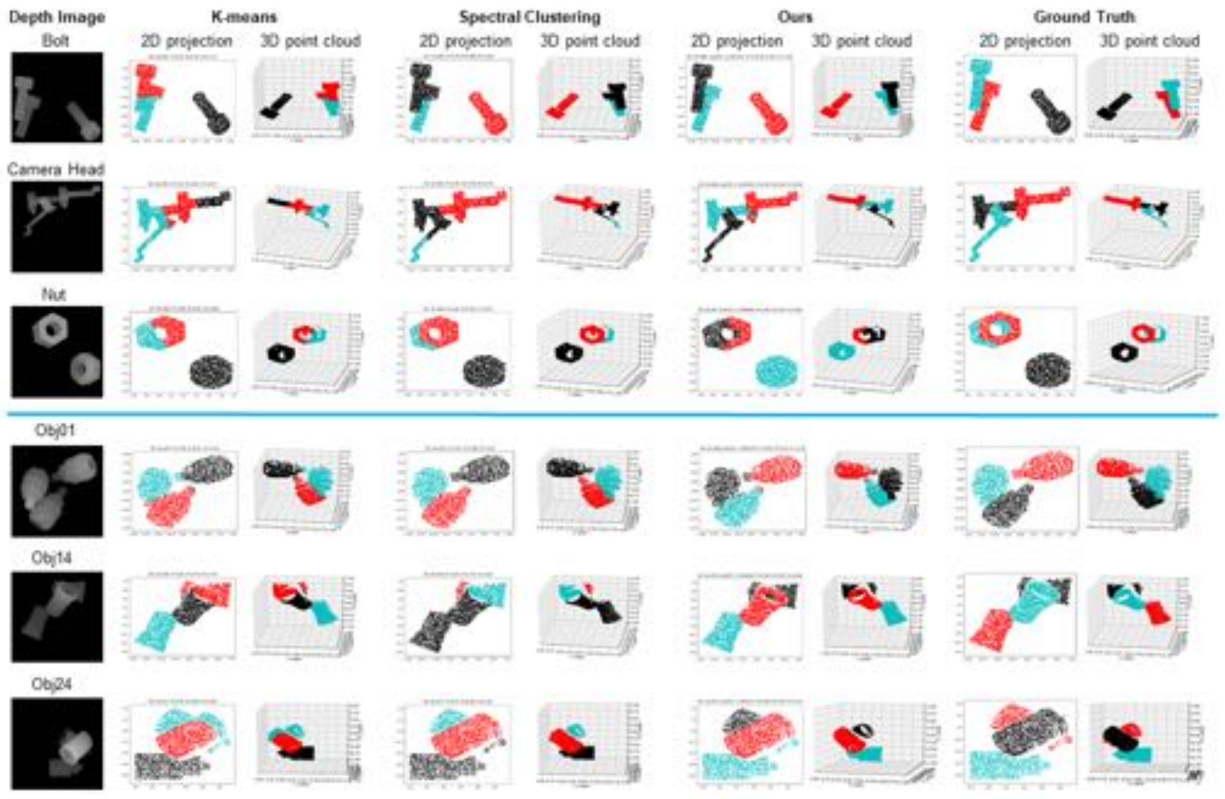


Figure 4.5: The qualitative results of instance segmentation of 2D projection and 3D point clouds with $N = 3$ instances.

4.3.5 Ablation Experiments

We run a number of ablation experiments to analyze the importance of various aspects of our algorithm. The complete numerical results of all ablation experiments are provided in the Appendix B.

Different rotation representations. We compare the reconstruction quality of object model and the performance of instance segmentation using three different rotation representations: axis-angle, ortho6D Zhou et al. (2019a), and quaternion. We show the learned 3D models of the Nut for $N = 1$ and 2 in Fig. 4.7. It is apparent from the figure that the quaternion representation of 3D rotaitons yields a slightly better 3D model. This mirrors the numerical results, reported in the Appendix B, which show that the quaternion representation is able to achieve a smaller average chamfer distance (smaller loss) than the other two representations.

Varying the radius r in HPR occlusion. In HPR occlusion, the value of the radius parameter

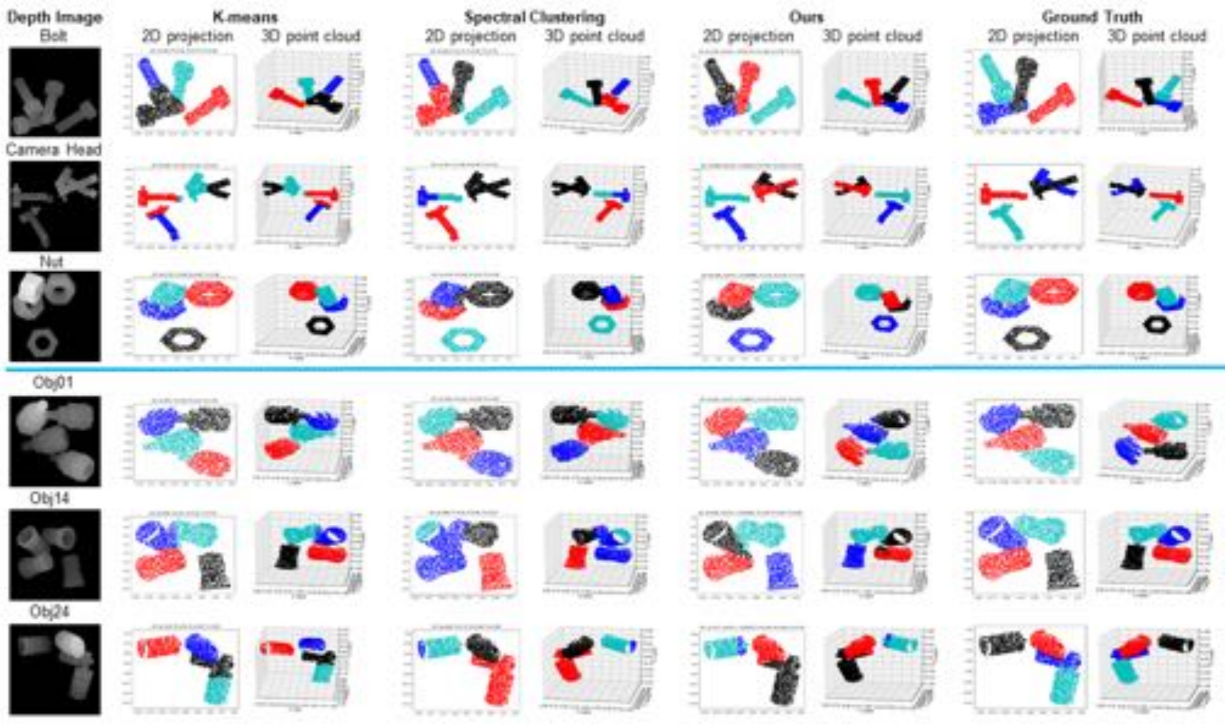


Figure 4.6: The qualitative results of instance segmentation of 2D projection and 3D point clouds with $N = 4$ instances.

r affects the number of visible points in the predicted point cloud. In Fig. 4.8, we show the effect of varying the HPR radius value on the 3D object model learned by our method. For a single instance ($N = 1$) of Bolt, it can learn a reasonable object model for $r = 2.0$ and $r = 2.9$. However, with $r = 3.14$ the 3D object model more resembles a cylinder. For two instances ($N = 2$), it learns a 3D object model similar to a dumbbell with $r = 2.0$, while it learns reasonable 3D object models with $r = 2.9$ and $r = 3.14$. Without using HPR occlusion (right), it learns a flat 3D object model for both $N = 1$ and $N = 2$. This demonstrates that the HPR occlusion module is critical for learning complete high-fidelity 3D object models.

Simplified two-rotation-channel setting. To simplify the training process for the two-rotation-channel setting, we can calculate a modified loss using $\mathscr{L} = \min(\mathscr{L}_i)$, for $i = 1, \cdots, 2^N$, instead of using the softmin-weighted loss. The learned 3D object models are shown in Fig. 4.9. Using one rotation channel, it dumbell-shaped (two-headed) 3D object model of the Bolt. However, our method is able to learn high-fidelity object model of the Bolt using the simplified two-rotation-

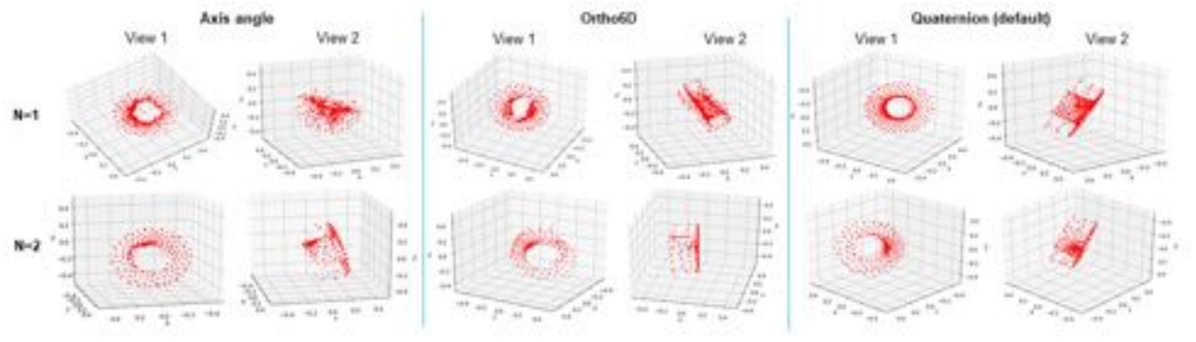


Figure 4.7: Effect of the algorithm's rotation representation on the learned 3D object model. The learned 3D object model is shown from two different views for $N = 1$ (*top*) and $N = 2$ (*bottom*) instances of the Nut.

channel setting. The higher fidelity object model also results in improved instance segmentation: For $N = 4$, the F1 scores for the one-rotation-channel, two-rotation-channel, and simplified two-rotation-channel settings are 0.81, 0.88, and 0.90, respectively.

4.4 Conclusion

We introduce a novel method to a novel task: the joint unsupervised learning of a 3D object model and estimation of 6D pose from depth images of multiple instances of an object. We also apply the method to instance segmentation in 3D point clouds. Unlike the traditional task of 6D pose estimation, our problem considers those cases in which neither the CAD model nor the ground-truth 6D pose are available during training or testing. To handle pose ambiguity of an unknown object, we jointly optimize the model learning and pose estimation in end-to-end deep learning framework. To handle occlusions, we incorporate an occlusion model into our architecture, which enables the model-generated point cloud to match the point cloud that was obtained from the input depth map. The task, and our proposed approach, have important applications in numerous areas including augmented reality, robotics, and 3D scene understanding.

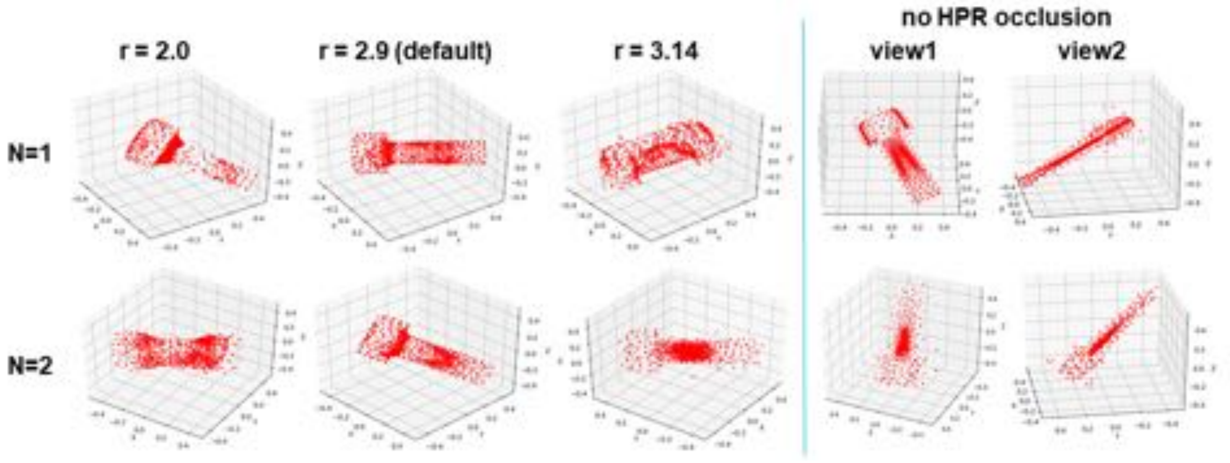


Figure 4.8: The learned 3D object models using the simplified two-rotation-channel setting with different values of radius r in HPR occlusion with $N = 1$ (*top*) and $N = 2$ (*bottom*) instances of the Bolt. At right, we plot two views of the 3D object models learned *without* HPR occlusion.

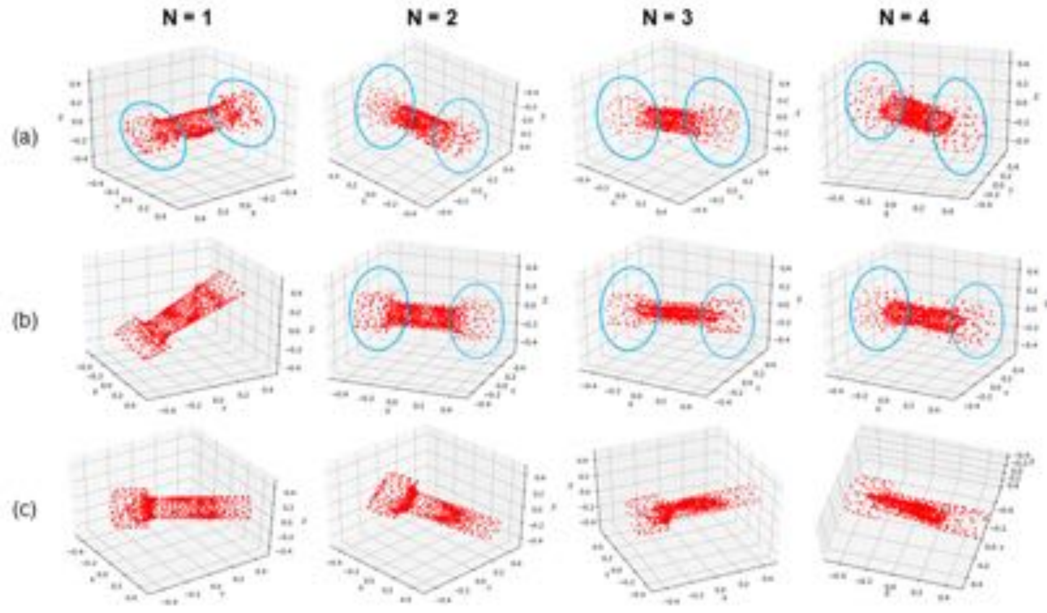


Figure 4.9: The learned complete 3D object models of Bolt using the (a) one-rotation-channel, (b) two-rotation-channel, and (c) simplified two-rotation-channel setting, for different numbers of instances N.

Chapter 5

Low Quality Image Classification

5.1 Introduction

Recently, deep Convolutional Neural Networks (CNNs) have demonstrated impressive improvements in image classification Krizhevsky et al. (2012); He et al. (2016), object detection Girshick et al. (2014); Ren et al. (2015); Ma et al. (2018), and instance segmentation He et al. (2017). The success of CNNs has become possible mostly due to the large amount of labeled dataset ImageNet Deng et al. (2009), as well as advances in better learning algorithms Goyal et al. (2017); Wu et al. (2018) and computing resources (*i.e.* GPUs). All those work typically assume that images are of sufficiently high resolution (*e.g.* 224×224 or higher).

However, the training of deep CNNs requires large amount of labeled dataset. In order to overcome this limitation, researchers have introduced transfer learning techniques. A common way to make use of transfer learning in the computer vision applications is to start from a pre-trained model in a similar task or domain, and then fine-tune the parameters to the new task. For example, the pre-trained model on ImageNet dataset for image classification can be fine-tuned for object detection on Pascal VOC dataset Girshick et al. (2014); Ren et al. (2015).

In this paper, we focus on low resolution (*e.g.* 32×32 or less) image classification as for privacy purpose, it is common to use low resolution images in real-world applications, such as face recognition in surveillance videos Zou & Yuen (2011). Without additional information, learning from low resolution images always reduces to an ill-posed optimization problem, and achieves a much degraded performance Pinheiro et al. (2015).

As shown in Fig. 5.1, the deep feature of high resolution images extracted from the pre-trained

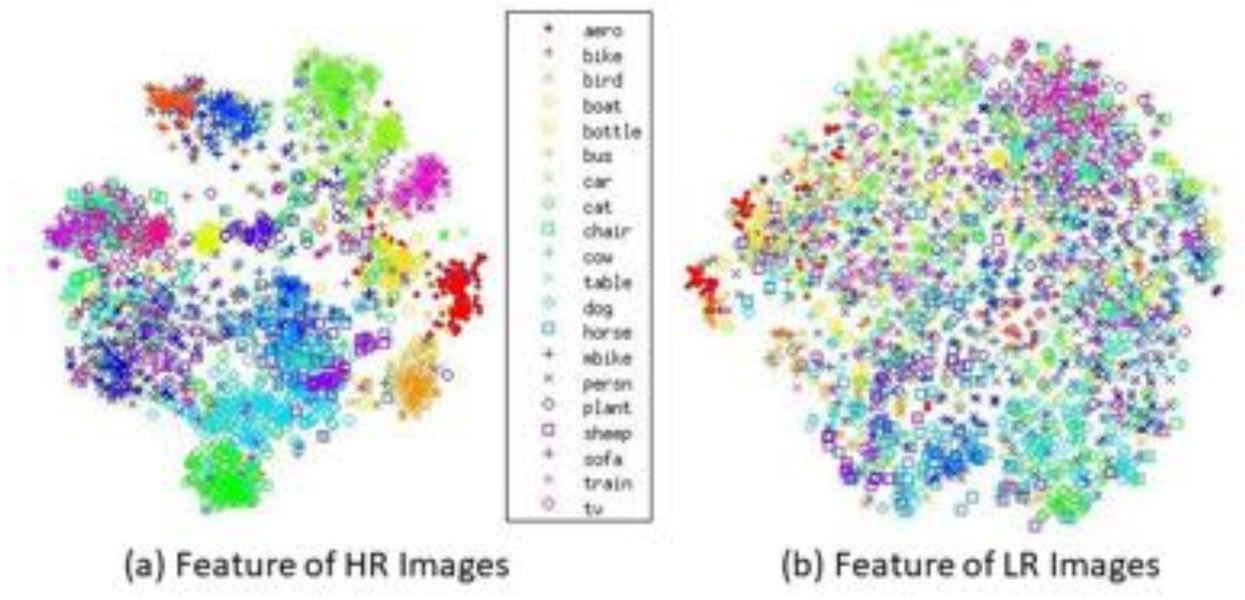


Figure 5.1: The tSNE Maaten & Hinton (2008) of deep features (2048-D) of VOC2007 train set extracted from pool5 layer of pre-trained ResNet-101 He et al. (2016). (a) Feature of High Resolution (HR) images, and (b) feature of Low Resolution (LR) images. The HR features are well separated, however, the LR features are mixed together.

convenet has already learned discriminative per-class feature representation. Therefore, it is able to be well separated in the tSNE visualization. However, the extracted feature of low resolution images is mixed together. A possible solution is to exploit the transfer learning, leveraging the discriminative feature representation from high resolution images to low resolution images.

In this paper, we propose a simple while effective unsupervised deep feature transfer approach that boosts classification performance in low resolution images. We assume that we have access to the labeled high resolution images during training, but at test we only have the low resolution images. Most existing datasets are of high resolution. Moreover, it is much easier to label subcategories of the high resolution images. Therefore, we believe it is a reasonable assumption. We aim to transfer knowledge from such high resolution images to real world scenarios that only have low resolution images. The basic intuition behind our approach, is to guide the feature learning for the target low resolution domain using the high quality discriminative representations.

We have summarized the contributions of our work as follows.

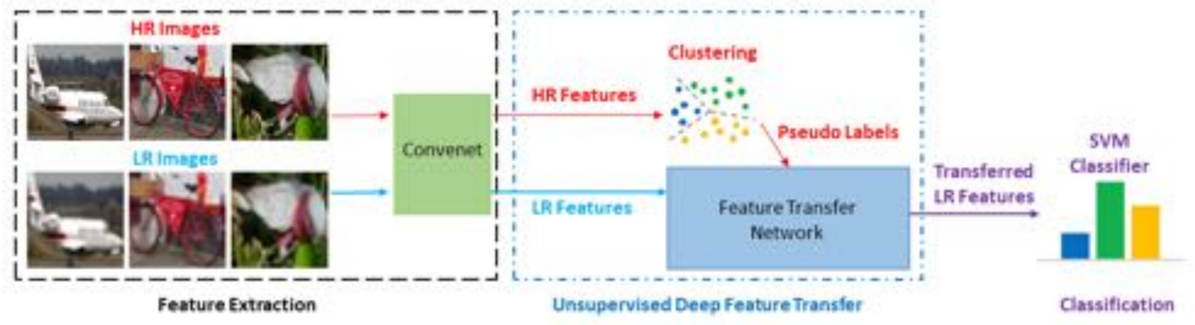


Figure 5.2: The overview of proposed unsupervised deep feature transfer algorithm. It consists of three modules. In the feature extraction module, a pre-trained deep convenet is used as feature extractor to obtain HR and LR features from HR and LR images, respectively. Then, we cluster the HR features to obtain pseudo-labels, which are used to guide the feature transfer learning of LR features in the feature transfer network. Finally, a SVM classifier is trained on the transferred LR features.

- No fine-tuning on convenet filters is required in our method. We use the pre-trained convenet to extract features for both high resolution and low resolution images, and then feed them into a two-layer feature transfer network for knowledge transfer. A SVM classifier is learned directly using these transferred low resolution features. Our network can be embedded into the state-of-the-art CNNs as a plug-in feature enhancement module.

- It preserves data structures in feature space for high resolution images, by transferring the discriminative features from a well-structured source domain (high resolution features space) to a not well-organized target domain (low resolution features space).

- Our performance is better than that of baseline using feature extraction approach for low resolution image classification task.

5.2 Proposed Approach

In this section, we describe our unsupervised deep feature transfer approach.

5.2.1 Preliminary

With the recent success of deep learning in computer vision, the deep convnets have become a popular choice for representation learning, to map raw images to an embedding vector space of fixed dimensionality. In the context of supervised learning, they could achieve better performance than human-beings on standard classification benchmarks He et al. (2015); Krizhevsky et al. (2012) when trained with large amount of labeled data.

Let f_θ denote the convenet mapping function, where θ is the corresponding learnable parameters. We refer to the vector obtained by applying this mapping to an image as feature or features. Given a training set $X = \{x_1, \cdots, x_N\}$ of N images, and the corresponding ground truth labels $Y = \{y_1, \cdots, y_N\}$, we want to find an optimal parameter θ^* such that the mapping f_θ^* predicts good general features. Each image x_i associates with a class label y_i in $\{0,1\}^k$. Let g_w denote a classifier with parameter ω. The classifier would predict the labels on top of the features $f_\theta(x_i)$. The parameter θ of the mapping function and the parameter ω of the classifier are then learned jointly by optimizing the following objective function:

$$\min_{\theta,\omega} \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}(g_w(f_\theta(x_i), y_i)), \tag{5.1}$$

where $\mathscr{L}$ is the multinominal logistic loss for measuring the difference between the predicted labels and ground-truth labels given training data samples.

5.2.2 Unsupervised Deep Feature Transfer

The idea of this work is to boost the feature learning for low resolution images by exploiting the capability of unsupervised deep feature transfer from the discriminative high resolution feature. The overview of proposed approach is shown in Fig. 5.2. It consists of three modules: feature extraction, unsupervised deep feature transfer, and classification, discussed below.

Feature extraction. We observe that the deep features extracted from convenet could generate well separated clusters as shown in Fig. 5.1. Therefore, we introduce the transfer learning to boost

Table 5.1: We use grid search to find the optimal combination of N_1 and N_2 for the two-layer feature transfer network by calculating the mean average precision (mAP) on VOC2007 validation set.

N_2 \ N_1	256	512	1024	2048	4096
20	0.704	0.741	**0.771**	0.786	**0.800**
100	0.718	**0.752**	0.768	**0.789**	**0.800**
200	**0.727**	0.746	**0.771**	0.788	**0.800**
500	0.717	0.743	0.766	0.784	0.795
1000	0.713	0.743	0.762	0.783	0.793
2048	0.718	0.739	0.765	0.783	0.794

the low resolution features learning via the supervision from high resolution features. Then, we extract the features (N-Dimensional) of both high and low resolution images from a pre-trained deep convenet. More details are described in Sec. 5.3.2.

Unsupervised deep feature transfer. We propose a feature transfer network to boost the low resolution features learning. However, in our assumption, the ground truth labels for low resolution images are absent. Therefore, we need to make use of the information from high resolution features. In order to do this, we propose to cluster the high resolution features and use the subsequent cluster assignments as "pseudo-label" to guide the learning of feature transfer network with low resolution features as input. Without loss of generality, we use a standard clustering algorithm, k-means. The k-means takes a high resolution feature as input, in our case the feature $f_\theta(x_i)$ extracted from the convenet, and clusters them into k distinct groups based on a geometric criterion. Then, the pseudo-label of low resolution features are assigned by finding its nearest neighbor to the k centroids of high resolution features. Finally, the parameter of the feature transfer network is updated by optimizing Eq. (5.1) with mini-batch stochastic gradient descent.

Classification. The final step is to train a commonly used classifier such as Support Vector Machine (SVM) using the transferred low resolution features. In testing, given only the low resolution images, first, our algorithm extracts the features. Then feeds them to the learned feature transfer network to obtain the transferred low resolution features. Finally, we run SVM to get the classification results directly.

5.3 Experiments

In this section we first describe the dataset, then discuss the implementation details, and finally we report experimental results.

5.3.1 Dataset

We conduct the low resolution classification on the PASCAL VOC2007 dataset Everingham et al. (2015) with 20 object classes. There are $5,000$ images in VOC2007 trainval set and $4,952$ images in VOC2007 test set. However, the images in the dataset are of high resolution. We follow Lin et al. (2014) to generate the low resolution images. In this work, we generate high resolution images by resizing the original images to 224×224 using bicubic interpolation. We generate the low resolution images by down-sampling the original images to 32×32, and then up-sampling to 224×224.

5.3.2 Implementation Details

We conduct our experiment using Caffe Jia et al. (2014). We use the ResNet-101 He et al. (2016) pre-trained on ILSVRC2012[1] Russakovsky et al. (2015) as the backbone convenet to extract the features from high and low resolution images. We extract the features from the pool5 layer, which gives a feature vector with dimension of $N = 2048$.

The feature transfer network is a two-layer fully connected network. We conduct grid search to find the optimal design for the network architecture, see Sec. 5.3.3. It is initialized using MSRA initialization He et al. (2015). We train the feature transfer network using stochastic gradient descent with weight decay 0.0005, momentum 0.9, batch size $1,000$, epoch $1,000$, total iteration $31,561$. The initial learning rate is 0.01, and is decreased by 10 after every $15,000$ iterations.

[1]We download the Caffe Model from `https://github.com/BVLC/caffe/wiki/Model-Zoo`

Table 5.2: Per-class average precision (%) for object classification on the VOC2007 test set.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
HR	97.6	92.7	89.2	85.8	90.6	87.5	96.2	94.3	81.4	83.3	80.0	86.9	84.2	90.0	95.4	95.0	88.3	71.6	96.0	95.9	89.1
LR	87.5	84.8	77.5	77.4	80.4	76.5	90.6	72.1	75.1	72.9	69.5	65.0	71.7	73.9	92.8	90.8	78.3	48.6	83.3	92.3	78.1
Ours	89.1	86.5	80.1	78.1	79.6	77.4	92.4	75.4	79.4	73.2	72.5	68.5	74.0	77.1	95.0	91.9	77.6	53.4	86.1	92.5	80.0

5.3.3 Feature Transfer Network

The feature transfer network is shallow, with two fully connected layers. Let N_1 and N_2 denote the neurons of the first and second fully connected layers, respectively. We conduct grid search to find the optimal combination for N_1 and N_2, as shown in Table 5.1. The number N_2 is determined by the number of clusters k for the pseudo labels in k-means.

As we can see, when the neurons of N_2 is fixed, the mAP increases as the neurons of N_1 increases. This is because the capacity of the two-layers feature transfer network increases as the neurons increases in N_1. However, given a fixed number of neurons of N_1, the value of mAP would increase first, and then decrease when the value of neurons in N_2 is larger enough, maybe 200 is a threshold value in our two-layer network as shown in the table. We observe that the hyperparameters with $N_2 = 100$ and $N_1 = 4096$ for the neurons give us the best performance. We use the same values in our experiment.

5.3.4 Low Resolution Image Classification

We evaluate the performance of image classification in the context of binary classification task on the VOC2007 test set using SVM Chang & Lin (2011) classifier in matlab. We have compared our algorithm with two baselines: HR and LR, discussed below. HR is to use the extracted high resolution features (2048-D) of VOC2007 trainval set to train the SVM and report the classification performance on VOC2007 test set. It is similar for LR, but with the extracted low resolution features (2048-D). Our method transfers the low resolution feature from 2048-D to 100-D. Therefore, we train the SVM using the 100-D features for each class. We show the comparison in Table 5.2.

The HR is the upper bound of our method, while the LR is the lower bound. As we can see from

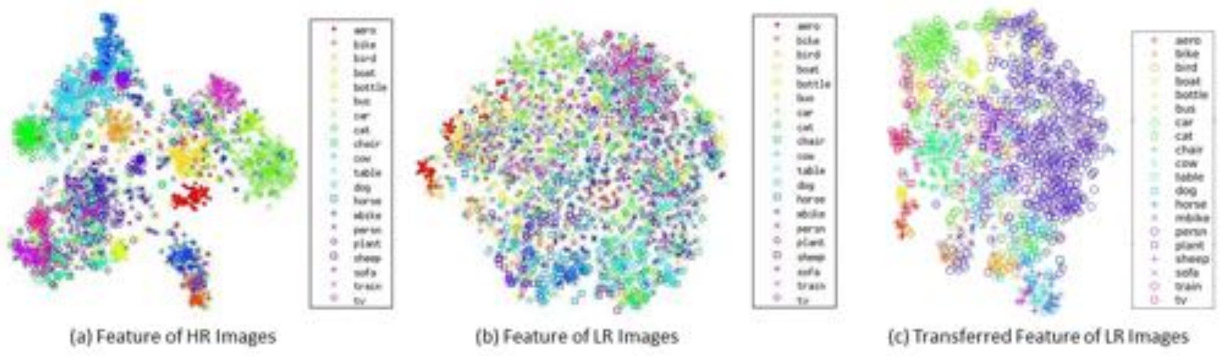


Figure 5.3: The tSNE of features on VOC2007 test set. (a) Feature (2048-D) of High Resolution (HR) images, (b) feature (2048-D) of Low Resolution (LR) images, (c) transferred feature (100-D) of LR images.

the Table 5.2, the proposed unsupervised deep feature transfer is able to boost the low resolution image classification by about 2%. Except for the classes of "bottle" and "sheep", our method outperforms the LR. As shown in Fig. 5.3, we find the transferred low resolution features are separated much better than the extracted low resolution features. Those indicate that the proposed unsupervised deep feature transfer algorithm does help transfer more discriminative representations from high resolution features. Therefore, it boost the mAP performance on low resolution images classification task. The feature transfer network could also be embedded into the state-of-the-art deep neural networks as a plug-in module to enhance the learned features.

5.4 Conclusion

In this paper, we propose a unsupervised deep feature transfer algorithm for low resolution image classification. The proposed two-layer feature transfer network is able to boost the classification by 2% on mAP. It can be embedded into the state-of-the-art deep neural networks as a plug-in feature enhancement module. While our current experiments focus on generic classification, we expect our feature enhancement module could be very useful in detection, retrieval, and category discovery settings as well in the future.

Chapter 6

Conclusions and Future Work

In this chapter, we first summarize our main contributions on the problems we have studied towards efficient and robust deep learning in optimization for training deep models, point cloud analysis and low quality image classification. Our contributions include novel approximate algorithm, *BPGrad* solver technique for the training of deep models. In addition, we also propose a novel and single end-to-end deep network framework for 3D object model learning and 6D pose estimation in point cloud. Furthermore, we propose a novel unsupervised deep feature transfer network for low quality image classification. Finally, we discuss some interesting directions the future work.

6.1 Main Contributions

Towards Global Optimality in Deep Learning: In this work, 1) we have proposed a novel approximation algorithm, *BPGrad*, towards searching for the global optimality in DL via branch and pruning based on the Lipschitz continuity assumption. The Lipschitz continuity not only provides us a way to estimate the lower and upper bounds of the global optimality, it also serves as the regularization to further smooth the objective functions in deep learning. 2) We have theoretically proved that under some conditions our BPGrad algorithm can converge to the global optimality within finite iterations. 3) Empirically in order to avoid the high demand of computation as well as storage for BPGrad in deep learning, we propose a new efficient solver. A justification on preserving the properties of BPGrad is provided both theoretically and empirically. We have demonstrated the superiority of our BPGrad solver to several popular DL solvers for vision applications of object recognition, detection, and segmentation.

Point Cloud Analysis: In this work, 1) we introduce a novel method to a novel task: the jointly 3D object model learning and 6D pose estimation from depth images with multiple instances of single type object with application to instance segmentation in 3D point clouds. Unlike the traditional task of 6D pose estimation, our problem considers those cases in which the CAD model and the ground-truth 6D pose are not available during training or testing. 2) To handle pose ambiguity of an unknown object, we jointly optimize the model learning and pose estimation using an end-to-end deep neural network. To handle occlusions, we incorporate an occlusion model into our architecture, which enables the model-generated point cloud to match the point cloud that was obtained from the input depth map. 3) We provide extensive experiments on unsupervised instance segmentation, demonstrating that in addition to learning a 3D object model, our method performs better than or comparably to standard baselines for instance segmentation.

Low Quality Image Classification: In this work, 1) we propose an unsupervised deep feature transfer network for low quality image classification. 2) The proposed multilayer feature transfer network preserves data structures in feature space of HR images, by transferring the discriminative features from a well-structured source domain (HR features space) to a not well-organized target domain (LR features space). 3) The proposed feature transfer network is a plug-in feature enhancement module, which could be embedded into the state-of-the-art deep neural networks. Our feature transfer network could boost the classification performance by 2% on mAP from the baseline method on low quality images.

6.2 Future Work

We discuss below a few research directions that would extend our current works.

Optimization for Training Deep Models: Based on the theoretical analysis in our current work, we claim that our solver is able to locate the solutions close to the global minima. However, with millions of parameters in deep neural networks, it is still an open problem to empirically visualize and and theoretically analyze how an algorithm converges step by step. Li et al. (2018) propose a simple “filter normalization" method that could visualize loss function curvature and make meaningful side-by-side comparisons between loss functions. One possible approach is to apply a variety of visualization methods to explore the effect of training parameter, solvers and network architectures on the loss function landscape and the shape of minimizers (global optimality). Another promising direction towards efficient and robust deep learning is to keep improving the variants of SGD solvers Reddi et al. (2018); Zhang et al. (2019); Liu et al. (2019).

Low Quality Image Recognition and Detection: There have been tremendous progress on images and videos recognition using deep learning techniques in recent years. However, most of prior deep models are trained, applied, and evaluated on high quality benchmark datasets, such as PASCAL VOC Everingham et al. (2010a), ImageNet Deng et al. (2009); Russakovsky et al. (2015) and MSCOCO Lin et al. (2014). However, in many emerging applications such as autonomous driving, mobile devices and robotics, the captured visual images and videos are of low quality from complex and unconstrained environments. Therefore, the performances would be largely degraded when applied to low quality images and videos. Moreover, the low quality images and videos suffer from various types of degradations, such as noise, motion blur, illumination, low resolution, contrast, out-of-focus, brightness, and sharpness. The prior visual recognition models would fail when the level of degradations passes some empirical threshold. Our current work focuses on deep feature transfer learning for low resolution image classification. One possible direction is to explore an universal feature transfer network architecture in order to handle the above mentioned various types of degradations. We would extend our current deep feature enhancement module to object detection, retrieval, and category discovery settings as well in the future.

Point Cloud Analysis: Our current work presents an approach to jointly learn the 3D object model and estimate the 6D poses of multiple instances of the same object. Specifically, knowing the number of instances in the scene, the single deep network learns a point cloud describing the model's shape and produces a list of rigid transformations, *i.e.*, one for each instance. Given the predicted transformations and the model, a 3D point cloud is generated and compared to the corresponding ground-truth point cloud. This allows for unsupervised training where neither ground-truth poses nor CAD models are needed. The task, and our proposed approach, have important applications in numerous areas including augmented reality, robotics, and 3D scene understanding. One possible direction is to perform evaluation on sequences coming from real depth sensors of robotic arms. Another promising direction is to propose occlusion-aware loss function instead of handling occlusion using external operations.

References

Azizpour, H., Razavian, A. S., Sullivan, J., Maki, A., & Carlsson, S. (2016). Factors of transferability for a generic convnet representation. *IEEE TPAMI*, 38(9), 1790–1802.

Berrada, L., Zisserman, A., & Kumar, M. P. (2018). Deep frank-wolfe for neural network optimization. *arXiv preprint arXiv:1811.07591*.

Bharati, S. P., Nandi, S., Wu, Y., Sui, Y., & Wang, G. (2016). Fast and robust object tracking with adaptive detection. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on* (pp. 706–713).: IEEE.

Bharati, S. P., Wu, Y., Sui, Y., Padgett, C., & Wang, G. (2018). Real-time obstacle detection and tracking for sense-and-avoid mechanism in uavs. *IEEE Transactions on Intelligent Vehicles*, 3(2), 185–197.

Blum, A. & Rivest, R. L. (1989). Training a 3-node neural network is np-complete. In *Neurips* (pp. 494–501).

Bottou, L., Curtis, F. E., & Nocedal, J. (2016). Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*.

Brachmann, E., Michel, F., Krull, A., Ying Yang, M., Gumhold, S., et al. (2016). Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *CVPR* (pp. 3364–3372).

Brutzkus, A. & Globerson, A. (2017). Globally optimal gradient descent for a convnet with gaussian inputs. *arXiv preprint arXiv:1702.07966*.

Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *ECCV* (pp. 132–149).

Cen, F. & Wang, G. (2019a). Boosting occluded image classification via subspace decomposition-based estimation of deep features. *IEEE transactions on cybernetics*.

Cen, F. & Wang, G. (2019b). Dictionary representation of deep features for occlusion-robust face recognition. *IEEE Access*, 7, 26595–26605.

Chang, C.-C. & Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM TIST*, 2(3), 27.

Chaudhari, P., Choromanska, A., Soatto, S., & LeCun, Y. (2017). Entropy-sgd: Biasing gradient descent into wide valleys. *ICLR*.

Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C. C., & Lin, D. (2019). Hybrid task cascade for instance segmentation. In *CVPR*.

Chen, T., Goodfellow, I., & Shlens, J. (2015). Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. In *AISTATS* (pp. 192–204).

Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV* (pp. 628–644).: Springer.

Dai, J., He, K., & Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *CVPR* (pp. 3150–3158).

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Neurips* (pp. 2933–2941).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR* (pp. 248–255).: Ieee.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dinh, L., Pascanu, R., Bengio, S., & Bengio, Y. (2017). Sharp minima can generalize for deep nets. In *ICML*, volume 70 (pp. 1019–1028).

Do, T.-T., Cai, M., Pham, T., & Reid, I. (2018). Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv preprint arXiv:1802.10367*.

Draxler, F., Veschgini, K., Salmhofer, M., & Hamprecht, F. A. (2018). Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*.

Du, S. S., Zhai, X., Poczos, B., & Singh, A. (2019). Gradient descent provably optimizes over-parameterized neural networks. In *ICLR*.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(Jul), 2121–2159.

Eriksson, K., Estep, D., & Johnson, C. (2003). *Applied Mathematics Body and Soul: Vol I-III*. Springer-Verlag Publishing.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1), 98–136.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010a). The pascal visual object classes (voc) challenge. *IJCV*, 88(2), 303–338.

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

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010b). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2), 303–338.

Fan, H., Su, H., & Guibas, L. J. (2017). A point set generation network for 3d object reconstruction from a single image. In *CVPR*.

Ferrari, V., Marin-Jimenez, M., & Zisserman, A. (2008). Progressive search space reduction for human pose estimation. In *CVPR* (pp. 1–8).: IEEE.

Freeman, C. D. & Bruna, J. (2017). Topology and geometry of half-rectified network optimization. *ICLR*.

Fukushima, K. & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets* (pp. 267–285). Springer.

Gao, G., Lauri, M., Zhang, J., & Frintrop, S. (2018). Occlusion resistant object rotation regression from point cloud segments. In *ECCV* (pp. 0–0).

Girshick, R. (2015). Fast r-cnn. In *CVPR* (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR* (pp. 580–587).

Glorot, X., Bordes, A., & Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML* (pp. 513–520).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

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

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., & Wang, G. (2015). Recent advances in convolutional neural networks. *arXiv preprint arXiv:1512.07108*.

Guo, Y., Shi, H., Kumar, A., Grauman, K., Rosing, T., & Feris, R. (2019). Spottune: transfer learning through adaptive fine-tuning. In *IEEE CVPR* (pp. 4805–4814).

Gupta, A., Dollar, P., & Girshick, R. (2019). Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*.

Haeffele, B. D. & Vidal, R. (2017). Global optimality in neural network training. In *CVPR* (pp. 7331–7339).

Hand, P. & Voroninski, V. (2017). Global guarantees for enforcing deep generative priors by empirical risk. *arXiv preprint arXiv:1705.07576*.

He, H., Daume III, H., & Eisner, J. M. (2014). Learning to search in branch and bound algorithms. In *Neurips* (pp. 3293–3301).

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *ICCV* (pp. 2961–2969).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE ICCV* (pp. 1026–1034).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, L., Wang, G., & Hu, Z. (2018a). Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 27(9), 4676–4689.

He, L., Yu, M., & Wang, G. (2018b). Spindle-net: Cnns for monocular depth inference with dilation kernel method. In *2018 ICPR* (pp. 2504–2509).: IEEE.

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

Hochreiter, S. & Schmidhuber, J. (1997). Flat minima. *Neural Computation*, 9(1), 1–42.

Hodan, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., & Zabulis, X. (2017). T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *WACV* (pp. 880–888).: IEEE.

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *CVPR*.

Hubel, D. H. & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1), 215–243.

Insafutdinov, E. & Dosovitskiy, A. (2018). Unsupervised learning of shape and pose with differentiable point clouds. In *Neurips* (pp. 2802–2812).

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *The IEEE CVPR*.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Neurips* (pp. 2017–2025).

Ji, X., Henriques, J. F., & Vedaldi, A. (2018). Invariant information distillation for unsupervised image segmentation and clustering. *arXiv preprint arXiv:1807.06653*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia* (pp. 675–678).: ACM.

Kanazawa, A., Tulsiani, S., Efros, A. A., & Malik, J. (2018). Learning category-specific mesh reconstruction from image collections. In *ECCV* (pp. 371–386).

Kato, H., Ushiku, Y., & Harada, T. (2018). Neural 3d mesh renderer. In *CVPR* (pp. 3907–3916).

Katz, S., Tal, A., & Basri, R. (2007). Direct visibility of point sets. In *ACM Transactions on Graphics (TOG)*, volume 26 (pp.$\tilde{2}$4).: ACM.

Kawaguchi, K. (2016). Deep learning without poor local minima. In *Neurips* (pp. 586–594).

Kehl, W., Manhardt, F., Tombari, F., Ilic, S., & Navab, N. (2017). Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *ICCV* (pp. 1521–1529).

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. *ICLR*.

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

Kokkinos, I. (2011). Rapid deformable object detection using dual-tree branch-and-bound. In *Neurips* (pp. 2681–2689).

Koushik, J. & Hayashi, H. (2016). Improving stochastic gradient descent with feedback. *arXiv preprint arXiv:1611.01505*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Neurips* (pp. 1097–1105).

Land, A. H. & Doig, A. G. (2010). An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008* (pp. 105–132). Springer.

LeCun, Y. (1998). The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

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

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

Lee, J. D., Simchowitz, M., Jordan, M. I., & Recht, B. (2016). Gradient descent only converges to minimizers. In *COLT* (pp. 1246–1257).

Lehmann, A., Leibe, B., & Van Gool, L. (2011). Fast prism: Branch and bound hough transform for object class detection. *IJCV*, 94(2), 175–197.

Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems* (pp. 6389–6399).

Li, K., Ma, W., Sajid, U., Wu, Y., & Wang, G. (2019). Object detection with convolutional neural networks. *arXiv preprint arXiv:1912.01844*.

Li, Y. & Yuan, Y. (2017). Convergence analysis of two-layer neural networks with relu activation. In *Neurips* (pp. 597–607).

Liang, S., Sun, R., Lee, J. D., & Srikant, R. (2018). Adding one neuron can eliminate all bad local minima. In *Neurips* (pp. 4350–4360).

Lin, H. & Jegelka, S. (2018). Resnet with one-neuron hidden layers is a universal approximator. In *Advances in Neural Information Processing Systems* (pp. 6169–6178).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European Conference on Computer Vision* (pp. 740–755).: Springer.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Han, J. (2019). On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2018). Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*.

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

Long, M., Cao, Y., Wang, J., & Jordan, M. I. (2015b). Learning transferable features with deep adaptation networks. In *ICML* (pp. 97–105).

Ma, W., Wu, Y., Wang, Z., & Wang, G. (2018). Mdcn: Multi-scale, deep inception convolutional neural networks for efficient object detection. In *ICPR* (pp. 2510–2515).: IEEE.

Maaten, L. v. d. & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov), 2579–2605.

Malherbe, C. & Vayatis, N. (2017). Global optimization of lipschitz functions. In *ICML*.

Mukkamala, M. C. & Hein, M. (2017). Variants of rmsprop and adagrad with logarithmic regret bounds. *arXiv preprint arXiv:1706.05507*.

Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML* (pp. 807–814).

Nguyen, Q. & Hein, M. (2017). The loss surface of deep and wide neural networks. *arXiv preprint arXiv:1704.08045*.

Nvidia (2019). PhysX-3.3.0. `https://www.nvidia.com/object/physx-9.19.0218-driver.html`.

Pan, S. J., Kwok, J. T., & Yang, Q. (2008). Transfer learning via dimensionality reduction. In *AAAI*, volume 8 (pp. 677–682).

Pan, S. J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10), 1345–1359.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. In *Neurips Autodiff Workshop*.

Pedersoli, M., Vedaldi, A., & Gonzalez, J. (2011). A coarse-to-fine approach for fast deformable object detection. In *CVPR* (pp. 1353–1360).: IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2825–2830.

Pham, Q.-H., Nguyen, T., Hua, B.-S., Roig, G., & Yeung, S.-K. (2019). Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In *CVPR* (pp. 8827–8836).

Picheny, V., Wagner, T., & Ginsbourger, D. (2013). A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3), 607–626.

Pinheiro, P. O., Collobert, R., & Dollár, P. (2015). Learning to segment object candidates. In *Neurips* (pp. 1990–1998).

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems* (pp. 5099–5108).

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1), 145–151.

Qian, X. & Liu, Y. (2013). Branch and bound algorithm for dependency parsing with non-local features. *Transactions of the Association for Computational Linguistics*, 1, 37–48.

Rad, M. & Lepetit, V. (2017). Bb8: a scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *ICCV* (pp. 3828–3836).

Ramage, D., Heymann, P., Manning, C. D., & Garcia-Molina, H. (2009). Clustering the tagged web. In *Second ACM International Conference on Web Search and Data Mining (WSDM 2009)*.

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Neurips* (pp. 91–99).

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *IJCV*, 115(3), 211–252.

Saenko, K., Kulis, B., Fritz, M., & Darrell, T. (2010). Adapting visual category models to new domains. *ECCV*, (pp. 213–226).

Schwing, A. G. & Urtasun, R. (2012). Efficient exact inference for 3d indoor scene understanding. In *ECCV* (pp. 299–313).: Springer.

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

Sock, J., Kim, K. I., Sahin, C., & Kim, T.-K. (2018). Multi-task deep networks for depth-based 6d object pose and joint registration in crowd scenarios. *arXiv preprint arXiv:1806.03891*.

Sotelo, J., Mehri, S., Kumar, K., Santos, J. F., Kastner, K., Courville, A., & Bengio, Y. (2017). Char2wav: End-to-end speech synthesis.

Sotiropoulos, D. G. & Grapsa, T. N. (2001). A branch-and-prune method for global optimization. In *Scientific Computing, Validated Numerics, Interval Methods* (pp. 215–226). Springer.

Soudry, D. & Carmon, Y. (2016). No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*.

Sun, M., Telaprolu, M., Lee, H., & Savarese, S. (2012). Efficient and exact map-mrf inference using branch and bound. In *Artificial Intelligence and Statistics* (pp. 1134–1142).

Sundermeyer, M., Marton, Z.-C., Durner, M., Brucker, M., & Triebel, R. (2018). Implicit 3d orientation learning for 6d object detection from rgb images. In *ECCV* (pp. 699–715).

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *ICML* (pp. 1139–1147).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

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

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *CVPR* (pp. 1701–1708).

Tatarchenko, M., Dosovitskiy, A., & Brox, T. (2017). Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *ICCV* (pp. 2088–2096).

Tejani, A., Tang, D., Kouskouridas, R., & Kim, T.-K. (2014). Latent-class hough forests for 3d object detection and pose estimation. In *ECCV* (pp. 462–477).: Springer.

Tieleman, T. & Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Tommasi, T., Orabona, F., & Caputo, B. (2010). Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *IEEE CVPR* (pp. 3081–3088).: IEEE.

Tulsiani, S., Zhou, T., Efros, A. A., & Malik, J. (2017). Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR* (pp. 2626–2634).

Tzeng, E., Hoffman, J., Darrell, T., & Saenko, K. (2015). Simultaneous deep transfer across domains and tasks. In *IEEE ICCV* (pp. 4068–4076).

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2), 154–171.

Vedaldi, A. & Lenc, K. (2015). Matconvnet: Convolutional neural networks for matlab. In *ACM Multimedia* (pp. 689–692).

Wang, C., Xu, D., Zhu, Y., Martin-Martin, R., Lu, C., Fei-Fei, L., & Savarese, S. (2019a). Densefusion: 6d object pose estimation by iterative dense fusion. In *CVPR*.

Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., & Guibas, L. J. (2019b). Normalized object coordinate space for category-level 6d object pose and size estimation. In *CVPR*.

Wang, M. & Deng, W. (2018). Deep face recognition: A survey. *arXiv preprint arXiv:1804.06655*.

Wang, W., Yu, R., Huang, Q., & Neumann, U. (2018). Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *CVPR* (pp. 2569–2578).

Wu, Y., Marks, T. K., Cherian, A., Chen, S., Feng, C., Wang, G., & Alan, S. (2019a). Unsupervised deep feature transfer for low resolution image classification. In *ICCV 2019 5th International Workshop on Recovering 6D Object Pose*.

Wu, Y., Sui, Y., & Wang, G. (2017). Vision-based real-time aerial object localization and tracking for uav sensing system. *IEEE Access*, 5, 23969–23978.

Wu, Y., Zhang, Z., & Wang, G. (2018). Bpgrad: Towards global optimality in deep learning via branch and pruning. In *IEEE CVPR*.

Wu, Y., Zhang, Z., & Wang, G. (2019b). Unsupervised deep feature transfer for low resolution image classification. In *ICCV 2019 Workshop and Challenge on Real-World Recognition from Low-Quality Images and Videos*.

Xiang, Y., Schmidt, T., Narayanan, V., & Fox, D. (2017). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., & Zweig, G. (2016). Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*.

Xu, W., Wu, Y., Ma, W., & Wang, G. (2019). Adaptively denoising proposal collection for weakly supervised object localization. *Neural Processing Letters*, (pp. 1–14).

Yang, J., Parikh, D., & Batra, D. (2016). Joint unsupervised learning of deep representations and image clusters. In *IEEE CVPR* (pp. 5147–5156).

Yang, Y., Feng, C., Shen, Y., & Tian, D. (2018). Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR* (pp. 206–215).

Yi, L., Zhao, W., Wang, H., Sung, M., & Guibas, L. J. (2019). Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *CVPR*.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *NeurIPS* (pp. 3320–3328).

Yun, C., Sra, S., & Jadbabaie, A. (2018). Global optimality conditions for deep neural networks. In *ICLR*.

Zagoruyko, S. & Komodakis, N. (2016). Wide residual networks. In *BMVC*.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zeiler, M. D. & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV* (pp. 818–833).: Springer.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, M. R., Lucas, J., Hinton, G., & Ba, J. (2019). Lookahead optimizer: k steps forward, 1 step back. *arXiv preprint arXiv:1907.08610*.

Zhang, S., Choromanska, A. E., & LeCun, Y. (2015). Deep learning with elastic averaging sgd. In *Neurips* (pp. 685–693).

Zhang, Z. & Brand, M. (2017). Convergent block coordinate descent for training tikhonov regularized deep neural networks. In *Neurips*.

Zhou, Y., Barnes, C., Lu, J., Yang, J., & Li, H. (2019a). On the continuity of rotation representations in neural networks. In *CVPR*.

Zhou, Y., Yang, J., Zhang, H., Liang, Y., & Tarokh, V. (2019b). SGD converges to global minimum in deep learning via star-convex path. In *ICLR*.

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE ICCV*.

Zhu, P., Wen, L., Du, D., Bian, X., Ling, H., Hu, Q., Nie, Q., Cheng, H., Liu, C., Liu, X., et al. (2018a). Visdrone-det2018: The vision meets drone object detection in image challenge results. In *ECCV*.

Zhu, Z., Li, Y., & Liang, Y. (2018b). Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*.

Zhu, Z., Li, Y., & Song, Z. (2018c). A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*.

Zou, D., Cao, Y., Zhou, D., & Gu, Q. (2018). Stochastic gradient descent optimizes over-parameterized deep relu networks. *arXiv preprint arXiv:1811.08888*.

Zou, W. W. & Yuen, P. C. (2011). Very low resolution face recognition problem. *IEEE TIP*, 21(1), 327–340.

Appendix A

Additional Experiments of Chapter 3

A.1 Parameters tuning on NNIST and CIFAR-10

We perform grid search of parameters on MNIST for Adagrad, Adadelta, RMSProp and Adam, as shown in Fig. A.1, Fig. A.2, Fig. A.3 and Fig. A.4, respectively.

We perform grid search of parameters on CIFAR-10 for Adagrad, Adadelta, RMSProp and Adam, as shown in Fig. A.5, Fig. A.6, Fig. A.7 and Fig. A.8, respectively.

We summarize the parameters as follows.

- Adagrad: $\varepsilon = 10^{-10}$,

  $\rho = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\ (\textbf{default})]$.

- Adadelta: $\varepsilon = 10^{-6}$,

  $\rho = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\ (\textbf{default})]$.

- RMSProp: $\varepsilon = 1e^{-8}$,

  $\rho = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\ (\textbf{default})]$.

- Adam: $\varepsilon = 10^{-8}$,

  $\beta_1 = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\ (\textbf{default}), 0.99]$,

  $\beta_2 = [0.99, 0.999\ (\textbf{default}), 0.9999]$.

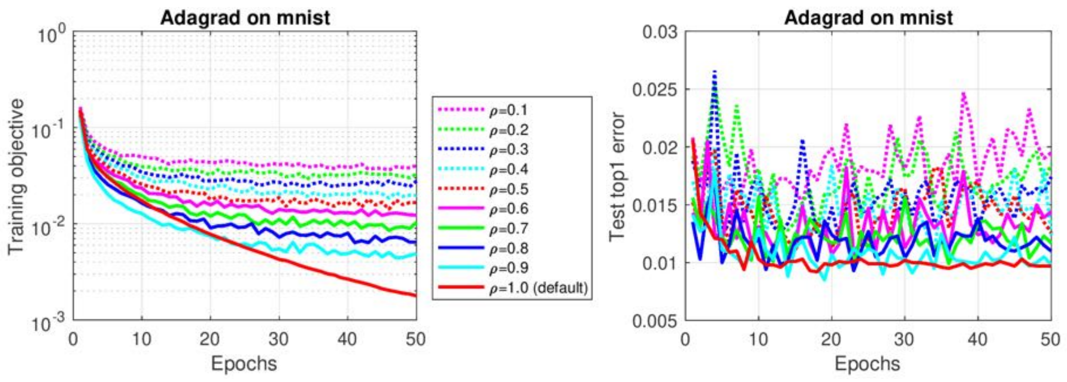


Figure A.1: Parameters search on MNIST **(left)** training objectives and **(right)** test top-1 errors for Adagrad.

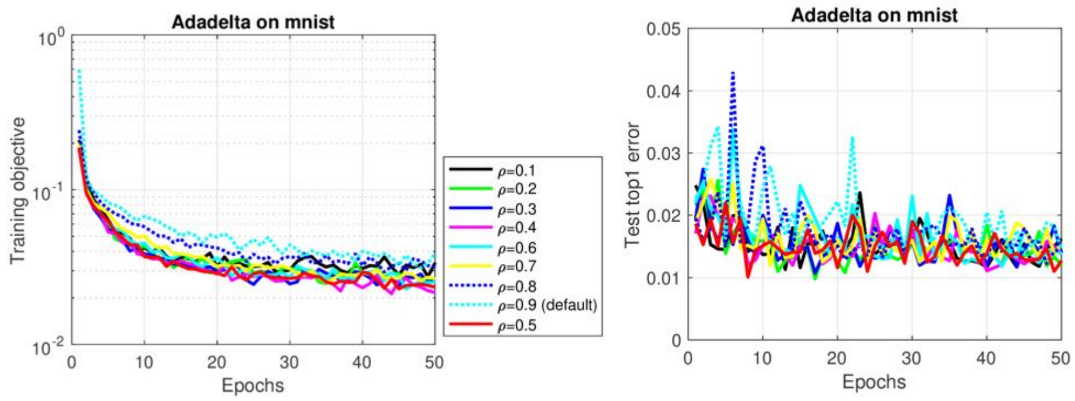


Figure A.2: Parameters search on MNIST **(left)** training objectives and **(right)** test top-1 errors for Adadelta.

A.2 Parameters tuning for Adam

We perform experiments using Adam with $\beta_1 = 0.9, \beta_2 = 0.999$ $(default)$ and $\beta_1 = 0.99, \beta_2 = 0.999$ on ImageNet for recognition and on VOC2011 for segmentation.

Fig. A.9 shows the comparison results on ImageNet. We can observe there is little difference between the two parameter settings. It exhibits similar trend for segmentation as shown Fig. A.10.

A.3 Parameters of solvers on 1-D functions

1. For function $f_1 = x\sin(x) + 4.815, x \in [0, 8]$:

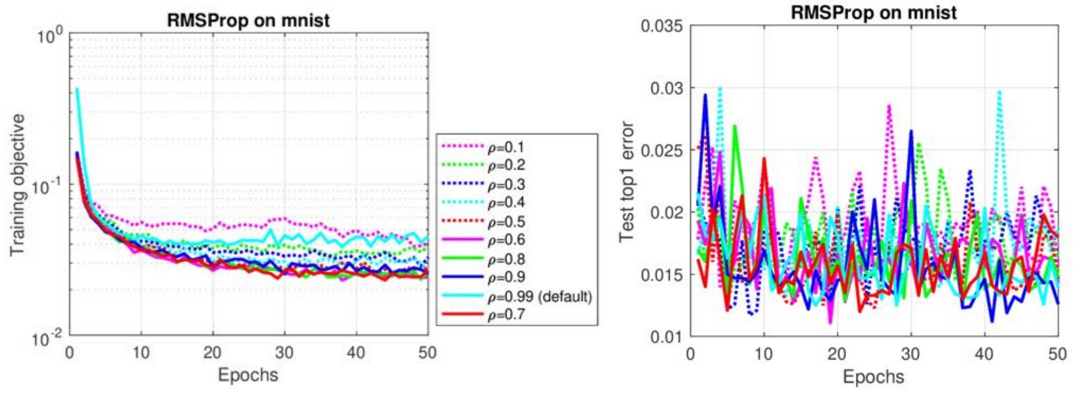


Figure A.3: Parameters search on MNIST **(left)** training objectives and **(right)** test top-1 errors for RMSProp.

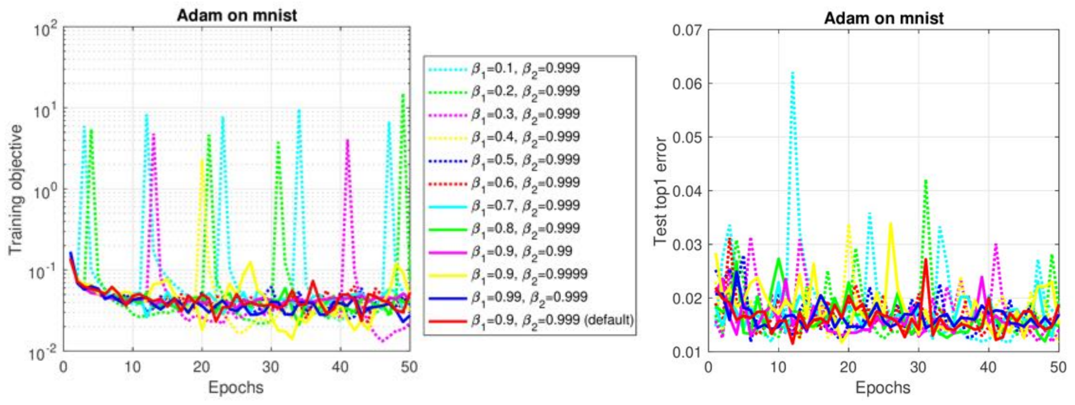


Figure A.4: Parameters search on MNIST **(left)** training objectives and **(right)** test top-1 errors for Adam.

- Adagrad: $\varepsilon = 10^{-10}, \rho = 0.9$, learning rate (lr)=0.5,
- Adadelta: $\varepsilon = 10^{-6}, \rho = 0.99$,
- RMSProp: $\varepsilon = 10^{-6}, \rho = 0.99, lr = 0.05$,
- Adam: $\varepsilon = 10^{-6}, \beta_1 = 0.9, \beta_2 = 0.99, lr = 0.1$,
- SGD: $\mu = 0, lr = 0.1$,
- SGD: $\mu = 0.9, lr = 0.008$,
- BPGrad Alg. 1: $\rho = 0.1, L = 4\pi$,
- BPGrad: $\rho = 0.1, L = 4\pi, \mu = 0$,

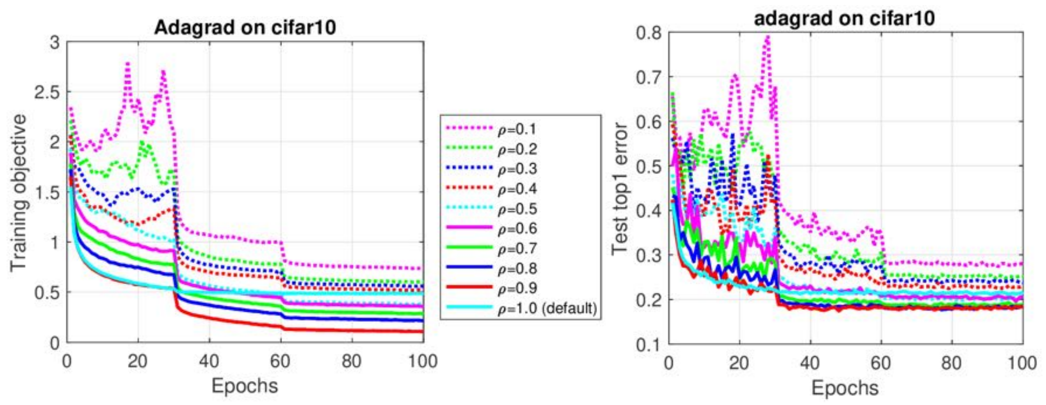


Figure A.5: Parameters search on CIFAR-10 **(left)** training objectives and **(right)** test top-1 errors for Adagrad.

- BPGrad: $\rho = 0.1, L = 4\pi, \mu = 0.9$.

2. For functions $f_2 = x\sin(x) + 11.05, x \in [0, 4\pi]$ and $f_3 = x\sin(x) + 15, x \in [0, 4\pi]$:

- Adagrad: $\varepsilon = 10^{-10}, \rho = 0.9$, learning rate (lr)=0.5,
- Adadelta: $\varepsilon = 10^{-6}, \rho = 0.99$,
- RMSProp: $\varepsilon = 10^{-6}, \rho = 0.99, lr = 0.05$,
- Adam: $\varepsilon = 10^{-6}, \beta_1 = 0.9, \beta_2 = 0.99, lr = 0.1$,
- SGD: $\mu = 0, lr = 0.1$,
- SGD: $\mu = 0.9, lr = 0.1$,
- BPGrad Alg. 1: $\rho = 0.1, L = 4\pi$,
- BPGrad: $\rho = 0.1, L = 4\pi, \mu = 0$, L is multiplied by 2 every 10 iterations (similar to regularization using weight decay).
- BPGrad: $\rho = 0.1, L = 4\pi, \mu = 0.9$, L is multiplied by 2 every 10 iterations.

A.4 Parameters of solvers on Rosenbrock function optimization

All the solvers start at the same initial point $(x_0, y_0) = (-1.2, 2.5)$.

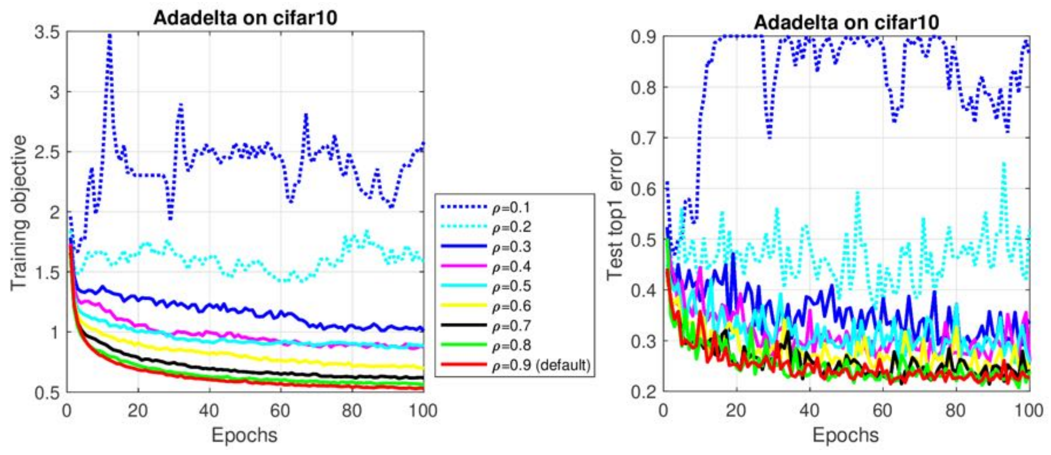


Figure A.6: Parameters search on CIFAR-10 **(left)** training objectives and **(right)** test top-1 errors for Adadelta.

- Adagrad: $\varepsilon = 10^{-10}, \rho = 0.9, lr = 0.01$,
- Adadelta: $\varepsilon = 10^{-6}, \rho = 0.99$,
- RMSProp: $\varepsilon = 10^{-6}, \rho = 0.99, lr = 0.02$,
- Adam: $\varepsilon = 10^{-6}, \beta_1 = 0.9, \beta_2 = 0.999$ **(default)**, $lr = 0.02$,
- SGD with momentum: $lr = 0.002, \mu = 0.9$,
- SGD no momentum: $lr = 0.002, \mu = 0$,
- BPGrad: $\rho = 0.1, L = 500, \mu = 0$,
- BPGrad: $\rho = 0.1, L = 500, \mu = 0.9$.

A.5 Parameters of solvers on MNIST using LeNet-5

All the solvers use weight decay=0.0005 and momentum=0.9, unless stated otherwise.

- Adagrad: $\varepsilon = 10^{-10}, \rho = 1.0$ **(default)**,
- Adadelta: $\varepsilon = 10^{-6}, \rho = 0.5$,

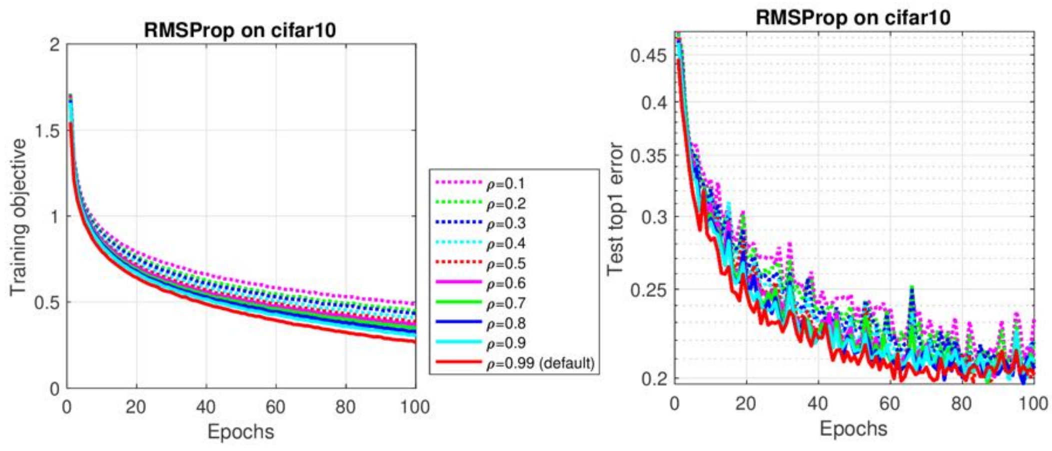


Figure A.7: Parameters search on CIFAR-10 **(left)** training objectives and **(right)** test top-1 errors for RMSProp.

- RMSProp: $\varepsilon = 10^{-8}, \rho = 0.7$,
- Adam: $\varepsilon = 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$ **(default)**,
- SGD: $lr = [0.001 * ones(1,100)]$,
- BPGrad: $\rho = 0.1, L = 15, \mu = 0$,
- BPGrad: $\rho = 0.1, L = 15, \mu = 0.9$.

A.6 Parameters of solvers on CIFAR-10 using LeNet

All the solvers use weight decay=0.0005 and momentum=0.9, unless stated otherwise.

- Adagrad: $\varepsilon = 10^{-10}, \rho = 0.9$,
- Adadelta: $\varepsilon = 10^{-6}, \rho = 0.9$ **(default)**,
- RMSProp: $\varepsilon = 10^{-8}, \rho = 0.99$ **(default)**,
- Adam: $\varepsilon = 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$ **(default)**,
- Eve: $lr = [0.01 * ones(1,30), 0.005 * ones(1,30), 0.0001 * ones(1,40)]$,

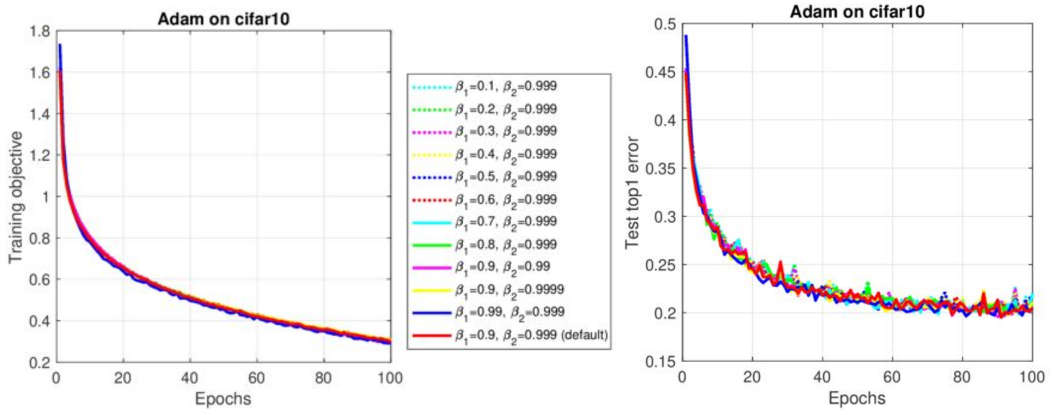


Figure A.8: Parameters search on CIFAR-10 **(left)** training objectives and **(right)** test top-1 errors for Adam.

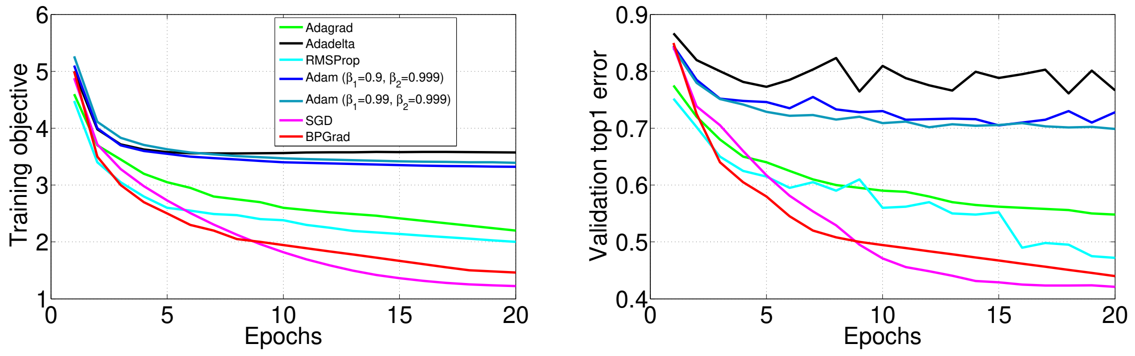


Figure A.9: Comparison on **(left)** training objectives and **(right)** validation top-1 errors for object recognition using ImageNet ILSVRC2012.

- SGD: $lr = [0.05 * ones(1,30), 0.005 * ones(1,30), 0.0005 * ones(1,40)]$,
- BPGrad: $\rho = 0.1, L = 50, \mu = 0$,
- BPGrad: $\rho = 0.1, L = 50, \mu = 0.9$.

A.7 Parameters of solvers on ImageNet

All the solvers use weight decay=0.0005 and momentum=0.9, unless stated otherwise.

- Adagrad: $\varepsilon = 10^{-10}, \rho = 1.0$ **(default)**,

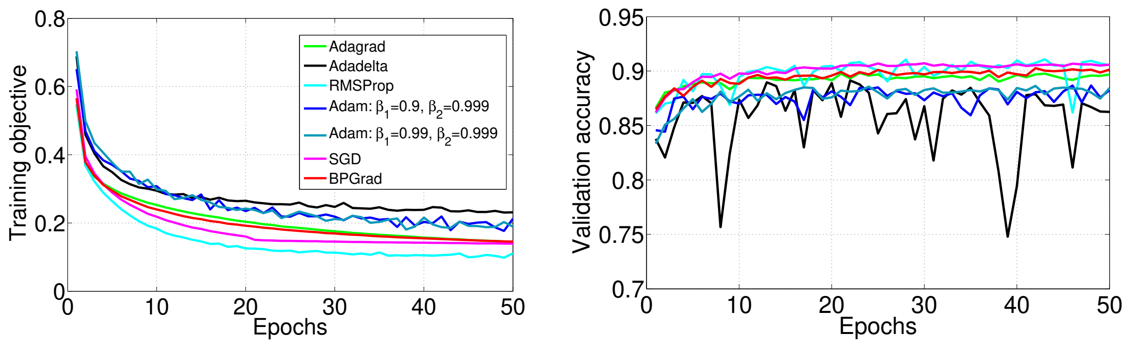


Figure A.10: Segmentation performance comparison using FCN-32s model on VOC2011 (**left**) training and (**right**) validation datasets.

- Adadelta: $\varepsilon = 10^{-6}, \rho = 0.9$ **(default)**,
- RMSProp: $\varepsilon = 10^{-8}, \rho = 0.99$ **(default)**,
- Adam: $\varepsilon = 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$ **(default)**,
- SGD: $lr = [\log(-1,-4,20)]$,
- BPGrad: $\rho = 0.1, L = 100, \mu = 0.9$.

A.8 Parameters of solvers for object detection

- Adagrad: $\varepsilon = 10^{-10}, \rho = 1.0$ **(default)**,
- RMSProp: $\varepsilon = 10^{-8}, \rho = 0.99$ **(default)**,
- Adam: $\varepsilon = 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$ **(default)**,
- SGD: $lr = [0.001 * ones(1,6), 0.0001 * ones(1,6)]$,
- BPGrad: $\rho = 0.1, L = 100, \mu = 0.9$.

A.9 Parameters of solvers for object segmentation

- Adagrad: $\varepsilon = 10^{-10}, \rho = 1.0$ **(default)**,
- Adadelta: $\varepsilon = 10^{-6}, \rho = 0.9$ **(default)**,
- RMSProp: $\varepsilon = 10^{-8}, \rho = 0.99$ **(default)**,
- Adam: $\varepsilon = 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$ **(default)**,
- SGD: $lr = [0.001 * ones(1,20), 0.0001 * ones(1,30)]$,
- BPGrad: $\rho = 0.1, L = 500, \mu = 0.9$.

Appendix B

Additional Experiments and Ablation Studies of Chapter 4

In this appendix material, we provide additional numerical results to experiments discussed in Chapter 4, as well as reporting on additional experiments and ablation studies.

B.1 Additional results for Instance Segmentation

Table 4.2 in Chapter 4 showed pairwise F1 scores for instance segmentation experiments. In Table B.1, we complete the table of results by also reporting the precision and recall scores that were used to calculate the pairwise F1 scores.

Table B.1: Instance segmentation comparison of baseline clustering methods with our method. F1 scores, precision (P) and recall (R) of instance segmentation on the validation dataset are shown. The complexity of instance segmentation increases with the number of instances *N* in each input depth image. SC is Spectral Clustering. Ours-v1 is one-rotation-channel setting. Ours-v2 is two-rotation-channel setting.

		Bolt			Camera Head			Nut			Obj01			Obj14			Obj24			Ave. on 6 objects		
		F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R
N=2	Kmeans	0.89	0.88	0.90	0.89	0.88	0.90	0.99	0.99	0.99	0.97	0.97	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.95	0.95	0.95
	SC	0.96	0.95	0.98	0.91	0.87	**0.96**	0.90	0.85	0.99	0.91	0.87	0.96	0.87	0.82	0.98	0.94	0.92	0.99	0.90	0.88	0.98
	Ours-v1	**0.99**	**0.99**	**0.99**	**0.94**	**0.94**	0.94	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.98**	**0.98**	**0.98**
	Ours-v2	0.96	0.96	0.96	0.93	0.93	0.93	0.99	0.99	**1.0**	0.97	0.96	0.97	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.98**	0.97	**0.98**
N=3	Kmeans	0.79	0.78	0.80	0.74	0.73	0.75	**0.98**	**0.99**	**0.98**	**0.93**	**0.93**	0.92	0.96	0.95	0.95	0.96	0.96	0.96	0.89	0.89	0.89
	SC	0.90	0.85	0.96	0.83	0.77	0.91	0.96	0.95	0.97	0.91	0.86	**0.98**	0.81	0.72	0.97	0.89	0.84	0.97	0.88	0.83	0.96
	Ours-v1	0.98	0.98	0.98	**0.93**	**0.93**	**0.93**	**0.98**	0.98	**0.98**	**0.93**	**0.93**	0.94	**0.98**	**0.98**	**0.98**	**0.99**	**0.99**	**0.99**	**0.97**	**0.97**	**0.97**
	Ours-v2	**0.99**	**0.99**	**0.99**	0.84	0.83	0.86	**0.98**	0.98	**0.98**	**0.93**	0.93	0.93	0.96	0.96	0.96	**0.99**	**0.99**	**0.99**	0.95	0.95	0.95
N=4	Kmeans	0.71	0.70	0.72	0.65	0.65	0.67	**0.97**	**0.98**	**0.97**	**0.90**	**0.91**	0.89	**0.92**	**0.92**	0.91	**0.94**	**0.95**	0.94	0.85	**0.90**	0.85
	SC	0.85	0.78	**0.95**	0.76	0.69	**0.86**	0.94	0.92	0.96	0.86	0.78	**0.97**	0.77	0.67	**0.95**	0.85	0.79	**0.96**	0.84	0.77	**0.94**
	Ours-v1	0.81	0.80	0.82	**0.85**	**0.85**	**0.86**	0.96	0.96	**0.97**	**0.90**	0.90	0.90	0.87	0.87	0.88	0.86	0.84	0.89	0.88	0.87	0.89
	Ours-v2	**0.88**	**0.88**	0.89	0.84	0.83	0.84	0.96	0.96	**0.97**	0.89	0.88	0.89	0.91	0.91	0.91	0.85	0.83	0.88	**0.89**	0.88	0.90

B.2 Ablation Studies

B.2.1 Different Rotation Channels Setting

The F1 score, precision (P), recall (R) and average chamfer distance (CD) is shown in Table B.2.

Table B.2: Ablation study by varying the number of rotation channels on **Bolt** with multiple instances (CD: the average of chamfer distance over the evaluation dataset, F1: F1 score, P: precision, R: recall). Ours-v1 is one-rotation-channel setting. Ours-v2 is two-rotation-channel setting. Ours-v3 is simplified two-rotation-channel setting.

	N=1				N=2				N=3				N=4			
	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R
Ours-v1	0.0007	–	–	–	0.0019	**0.99**	**0.99**	**0.99**	0.0016	0.98	0.98	0.98	0.0029	0.81	0.80	0.82
Ours-v2	0.0009	–	–	–	0.0013	0.96	0.96	0.96	0.0015	0.99	0.99	0.99	0.0023	0.88	0.88	0.89
Ours-v3	**0.0005**	–	–	–	**0.0011**	0.96	0.96	0.96	**0.0014**	**0.99**	**0.99**	**0.99**	**0.0022**	**0.90**	**0.90**	**0.91**

B.2.2 Vary HPR Radius Values

The results of Nut are shown in Table B.3.

Table B.3: Ablation study of HPR parameter in occlusion model for 3D object model reconstruction and instance segmentation on **Nut** and **Bolt** (– indicates not applicable).

		HPR=2				HPR=2.9 (default)				HPR=3.14				no HPR			
		CD	F1	P	R	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R
Nut	N=1	0.0059	–	–	–	0.0011	–	–	–	0.0010	–	–	–	0.0011	–	–	–
	N=2	0.0074	0.99	0.99	0.99	0.0014	1.0	1.0	1.0	0.0020	1.0	1.0	1.0	0.0030	0.95	0.96	0.95
Bolt	N=1	0.0015	–	–	–	0.0010	–	–	–	0.0018	–	–	–	0.0006	–	–	–
	N=2	0.0032	0.98	0.98	0.99	0.0014	0.99	0.99	0.99	0.0011	0.99	0.99	0.99	0.0011	0.99	0.99	0.99

B.2.3 Different Rotation Representations

As shown in Table B.4, we find that it achieves best performance using quaternion as rotation representation.

Table B.4: Ablation study of rotation representations for 3D object model reconstruction and instance segmentation on **Nut** (– indicates not applicable).

	Quaternion (default)				Ortho6D Zhou et al. (2019a)				Axis Angle			
	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R
N=1	0.0011	–	–	–	0.0018	–	–	–	0.0042	–	–	–
N=2	0.0014	1.0	1.0	1.0	0.0014	1.0	1.0	1.0	0.0022	1.0	1.0	1.0

B.2.4 Vary the Number of Points in the Learned Object Model

We vary the number of points in the learned object model from $N = 512, 1024$, to 2048. The results are shown in Table. B.5. As we can see, it can achieve slightly better object model and instance segmentation performance.

Table B.5: Ablation study of varying the number of points in learnable 3D object model for 3D object model reconstruction and instance segmentation on **Nut** (– indicates not applicable).

	NP=512				NP=1024 (default)				NP=2048			
	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R
N=1	0.0011	–	–	–	0.0011	–	–	–	0.0012	–	–	–
N=2	0.0038	1.0	1.0	1.0	0.0014	1.0	1.0	1.0	0.0060	0.99	0.99	0.99

B.2.5 Vary the Range in the Learned Object Model

We vary the range in the randomly initialized object model from $0.1, 0.25, 0.5$, to 1.0 as shown in Table B.6. We can see with range of 0.5 it can achieve better performance.

Table B.6: Ablation study of varying the range in learnable 3D object model for 3D object model reconstruction and instance segmentation on **Nut** (– indicates not applicable).

	Range ρ=0.1				Range ρ=0.25				Range ρ=0.5 (default)				Range ρ=1.0			
	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R
N=1	0.0011	–	–	–	0.0010	–	–	–	0.0012	–	–	–	0.0025	–	–	–
N=2	0.0016	0.99	0.99	0.99	0.0017	1.0	1.0	1.0	0.0014	1.0	1.0	1.0	0.0016	0.95	0.96	0.95

B.2.6 Vary the Number of Training Samples

We vary the number of training samples from $M = 500, 1K, 2K, 3K$, to $5K$. The results are shown in Table B.7. As we can see, it can achieve better object model and instance segmentation performance when using $5,000$ training samples.

Table B.7: Ablation study by varying the number of training images on **Nut** and **Bolt** with multiple instances. (CD: average chamfer distance over the evaluation set, smaller is better. F1: F1 score, P: precision, R: recall, larger is better).

		M=500				M=1K				M=2K				M=3K				M=5K (default)			
		CD	F1	P	R	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R	CD	F1	P	R
Nut	N=1	0.0017	–	–	–	0.0044	–	–	–	0.0011	–	–	–	**0.0010**	–	–	–	0.0011	–	–	–
	N=2	**0.0015**	**1.0**	**1.0**	**1.0**	0.0034	0.99	0.99	0.99	0.0025	**1.0**	**1.0**	**1.0**	0.0033	0.99	0.99	0.99	0.0021	0.99	0.99	0.99
	N=3	0.0061	0.97	0.97	0.97	**0.0052**	**0.98**	**0.98**	**0.98**	0.0058	**0.98**	**0.98**	**0.98**	0.0058	**0.98**	**0.98**	**0.98**	0.0064	**0.98**	**0.98**	**0.98**
	N=4	**0.0058**	**0.96**	**0.96**	**0.97**	**0.0058**	**0.96**	**0.96**	**0.97**	0.0060	0.95	0.95	0.96	0.0065	0.95	0.94	0.96	0.0068	**0.96**	**0.96**	**0.97**
Bolt	N=1	0.0034	–	–	–	0.0052	–	–	–	0.0029	–	–	–	0.0006	–	–	–	**0.0005**	–	–	–
	N=2	0.0010	0.97	0.97	0.97	0.0010	0.96	0.96	0.96	**0.0009**	0.99	0.99	0.99	**0.0009**	**1.0**	**1.0**	**1.0**	0.0011	0.96	0.96	0.96

Appendix C

List of Publications

Publications at top tier conferences and journals: ♠

Indications of impact (> 20 citations, awards): red color

Code: [B] Book; [C] Conference; [J] Journal; [W] Workshop; [S] Submitted

- Published Book Chapter

  1. *[B]* Kaidong Li, Wenchi Ma, Usman Sajid, **Yuanwei Wu** and Guanghui Wang, (2020). Object Detection with Convolutional Neural Networks. Deep Learning in Computer Vision: Principles and Applications, ISBN 9781138544420, 2020.

- Published Conference/Workshop/Journal Papers:

  1. *[W]* **Yuanwei Wu**, Tim K. Marks, Anoop Cherian, Siheng Chen, Chen Feng, Guanghui Wang and Alan Sullivan, "Unsupervised Joint 3D Object Model Learning and 6D Pose Estimation for Depth-Based Instance Segmentation", *Oral*, In Proc. of the IEEE International Conference on Computer Vision (ICCV) 5th International Workshop on Recovering 6D Object Pose, Seoul, Korea, Oct. 2019.

  2. *[W]* **Yuanwei Wu**, Ziming Zhang and Guanghui Wang, "Unsupervised Deep Feature Transfer for Low Resolution Image Classification", *Poster*, in Proc. of the IEEE International Conference on Computer Vision (ICCV) Workshop on Real-world Recognition from Low-quality Images and Videos, Seoul, Korea, Oct. 2019.

  3. *[C]* **Yuanwei Wu***, Ziming Zhang*, and Guanghui Wang, "Bpgrad: Towards global optimality in deep learning via branch and pruning", *Poster*, In Proc. of the IEEE Con-

ference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, Utah, USA, June 2018. (*: denotes co-first authors, acceptance rate=979/3300=29.6%) ♠

4. *[C]* Ziming Zhang, **Yuanwei Wu**, Wenchi Ma, Guanghui Wang and Alan Sullivan, "Self-Orthogonality Module: A Network Architecture Plug-in for Learning Orthogonal Filters", IEEE Winter Conference on Applications of Computer Vision (WACV), Mar. 2020.

5. *[C]* Wenchi Ma, **Yuanwei Wu**, Zongbo Wang and Guanghui Wang, "MDCN: Multi-Scale, Deep Inception Convolutional Neural Networks for Efficient Object Detection", *Poster*, International Conference on Pattern Recognition (ICPR), Beijing, China, Aug. 2018.

6. *[W]* Pengfei Zhu, Longyin Wen, $\cdots$, **Yuanwei Wu**, et al. "Visdrone-det2018: The vision meets drone object detection in image challenge results." In Proceedings of the European Conference on Computer Vision (ECCV), pp. 437-468, 2018.

7. *[C]* Sushil Bharati, Soumyaroop Nandi, **Yuanwei Wu**, Yao Sui and Guanghui Wang, "Fast and Robust Object Tracking with Adaptive Detection", *Oral*, IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), San Jose, CA, USA, Nov. 2016.

8. *[J]* **Yuanwei Wu**, Yao Sui and Guanghui Wang, "Vision-based Real-Time Object Localization and Tracking for UAV sensing system", IEEE Access, Vol. 5, 23969-23978, 2017.

   Citations: 33

9. *[J]* Wenchi Ma, **Yuanwei Wu**, Feng Cen and Guanghui Wang, "MDFN: Multi-Scale Deep Feature Learning Network for Object Detection", accept, Pattern Recognition, Dec. 2019.

10. *[J]* Wenju Xu, **Yuanwei Wu**, Wenchi Ma, and Guanghui Wang, "Adaptively Denoising Proposal Collection for Weakly Supervised Object Localization." Neural Processing

Letters (2019): 1-14.

11. *[J]* Bharati Sushil, **Yuanwei Wu**, Yao Sui and Guanghui Wang, “Real-time obstacle detection and tracking for sense-and-avoid mechanism in UAVs”, IEEE Trans. on Intelligent Vehicles 3, no. 2 (2018): 185-197.

- Submitted Conference/Journal Papers:

1. *[J]* **Yuanwei Wu**, Ziming Zhang and Guanghui Wang, “BPGrad: Towards Global Optimality in Deep Learning via Branch and Pruning”, submitted to IEEE Trans. Neural Netw. Learn. Syst., Jun. 2019. ♠