OPTIMIZATION OF RELU NEURAL NETWORKS USING QUOTIENT STOCHASTIC GRADIENT DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

It has been well known that neural networks with rectified linear hidden units (ReLU) as activation functions are positively scale invariant, which results in redundancy in their weight space (i.e., many ReLU networks with different weights are actually equivalent). In this paper, we formally characterize this redundancy/equivalence using the language of *quotient space* and discuss its negative impact on the optimization of ReLU neural networks. Specifically, we show that all equivalent ReLU networks correspond to the same vector in the quotient space, and each such vector can be characterized by the so-called skeleton paths in the ReLU networks. With this, we prove that the dimensionality of the quotient space is $\#\text{weight}-\#(\text{hidden nodes})$, indicating that the redundancy of the weight space is huge. In this paper, we propose to optimize ReLU neural networks directly in the quotient space, instead of the original weight space. We represent the loss function in the quotient space and design a new stochastic gradient descent algorithm to iteratively learn the model, which we call *Quotient stochastic gradient descent* (abbreviated as Quotient SGD). We also develop efficient techniques to ensure that the implementation of Quotient SGD almost requires no extra computations as compared to standard SGD. According to the experiments on benchmark datasets, our proposed Quotient SGD can significantly improve the accuracy of the learned model.

1 INTRODUCTION

Deep neural networks have achieved huge success in various AI tasks (Krizhevsky et al. (2012); LeCun et al. (2015); He et al. (2016); Xia et al. (2016); Goodfellow et al. (2014)). Since neural networks are highly non-convex, the optimization of them is challenging. In 1986, back propagation was experimented to efficiently compute the gradients of the weights in the neural networks (Rumelhart et al. (1986)). Afterwards, gradient methods (such as stochastic gradient descent and its variants like Adagrad, Adadelta, Adam) have become the main stream for neural network optimization (LeCun et al. (2015); Bottou (2010); Johnson & Zhang (2013); Duchi et al. (2011); Zeiler (2012); Kingma & Ba (2014)). At the same time, useful techniques, such as weight decay, skip connection, dropout, and batch normalization (Sutskever et al. (2013); Srivastava et al. (2014); Ioffe & Szegedy (2015)) were developed to address various problems encountered during the optimization of DNNs, such as overfitting and gradient vanishing.

This work is concerned with a specific yet important type of deep neural networks, i.e., neural networks whose activation functions are rectified linear hidden units (ReLU) (Hahnloser et al. (2000)). ReLU neural networks adopt ReLU activation functions and can avoid gradient vanishing to certain extent. Over the past ten years, ReLU networks have demonstrated their power in many important applications, such as image classification and text understanding (Krizhevsky et al. (2012); Ott et al. (2016)). However, it has been well known that ReLU networks are positively scale invariant (Neyshabur et al. (2015); Dinh et al. (2017)). That is, if the incoming weights of a hidden node with ReLU activation are multiplied by a positive constant c and the outgoing weights are divided by c, the neural network with the new weights will generate exactly the same output as the old one for an arbitrary input. We call such two neural networks *positively scale equivalent* for simplicity. [1] The

[1] Please note that, there are equivalent neural networks caused by other operators, such as permutation of hidden nodes. In this paper, we only focus on the positively scale equivalence.

existence of this type of equivalence implies that the weight space of ReLU networks is redundant and it will bring in negative impact on the optimization process sometimes (Neyshabur et al. (2015); Dinh et al. (2017)). Although such redundancy has been realized for a long time, almost all previous optimization algorithms blindly operate on the redundant weight space.

To solve the redundant problem and remove such kind of redundancy, we study the quotient space induced by the positively scale equivalence which we call it *quotient space* for simplicity [2] and design efficient optimization algorithms in quotient space.

First, we prove that two ReLU networks are positively scale equivalent if and only if all values and active status of their paths (i.e., the products of all the weights on the paths) are the same. Second, we find that the values of paths are not independent with each other. Directly update the values of paths and neglect the dependency is not reasonable. By noting that the relation between values of paths and weights follows a series generalized linear equations with a coefficient matrix related to the network structure (which we call it structure matrix), we define skeleton paths as paths whose value correspond to the maximal linearly independent column vectors of the structure matrix, and skeleton edges which are the free unknown variables of the series of linear equations. Moreover, we prove that all values and active status of paths can be calculated using values of skeleton paths together with the signs of skeleton edges. Thus, the quotient space is a vector space of the skeleton paths together with the signs of their skeleton weights. Then the Hausdofff dimension of the quotient space is as $\#$weights$-\#$(hidden nodes) for fully-connected neural networks (FNN) which is the same as the number of skeleton paths. Further analysis shows that the dimension of the manifold composed by the critical points of the loss in quotient space is smaller than that in the weight space, which indicates benefits to optimize neural networks in quotient space.

Therefore, we propose to optimize ReLU neural networks directly in the quotient space. We develop a *Quotient stochastic gradient descent* algorithm (abbreviated as Quotient SGD). Before the training process, we need to seek the skeleton paths and skeleton edges in the quotient space. Instead of directly deal with the structure matrix, we design *Skeleton Method* to construct the skeleton paths and skeleton edges layer by layer in a recursive manner. Next, we leverage the traditional back-propagation process to compute the gradients w.r.t. the quotient vector, according to a so-called inverse-chain-rule method; then it employs a so-called weight-allocation method to project the updates of quotient vector to the updates of weights, so as to implement the next round of back-propagation. Please note that by using these techniques, Quotient SGD almost has no additional computational costs as compared to the standard SGD method.

To verify the effectiveness of our proposed Quotient SGD algorithm, we conducted experiments for FNN, CNN, and ResNet on image classification tasks using benchmark datasets MNIST and CIFAR-10. The experimental results show that, Quotient SGD consistently converges faster than SGD with a better accuracy. This empirically indicates that optimizing ReLU neural networks in the quotient space is a better choice.

2 BACKGROUNDS

2.1 RELU NEURAL NETWORKS

Suppose $N_w(x) : \mathcal{X} \to \mathcal{Y}$ is a L-layer neural network with weight $w \in \mathcal{W} \subset \mathbb{R}^m$, where the input space is $\mathcal{X} \subset \mathbb{R}^d$ and the output space is $\mathcal{Y} = R^K$. In the l-th layer ($l = 0, \ldots, L$), there are h_l nodes. It is clear that, $h_0 = d, h_L = K$. We denote the nodes and their values as $\{O^l, o^l\}$. We use w_l to denote the weight matrix between layer $l-1$ and layer l. The values of the nodes are propagated as $o^l = \sigma(w_l^T o^{l-1})$, where $\sigma(\cdot) = max(\cdot, 0)$ is the ReLU activation function. The output of the neural network $N_w(x)$ is as $N_w(x) = w_L^T \sigma(w_{L-1}^T \sigma(\cdots \sigma(w_1^T x)))$. We use i_l to denote the index of nodes at layer l with $i_l = 1, \cdots, h_l$. We can also calculate the k-th output by paths (Balduzzi et al. (2017; 2015)), i.e.,

$$N_w^k(x) = \sum_{(i_0, \cdots, i_L = k)} \prod_{l=1}^{L} w_l(i_{l-1}, i_l) \cdot \prod_{l=1}^{L-1} \mathbb{I}(o_{i_l}^l(w, x) > 0) \cdot x_{p_0} \tag{1}$$

[2] Intuitively, the positively scale equivalent ReLU networks are regarded as the same point in quotient space.

where $(i_0, \cdots, i_L)$ is the path starting from input feature node $O^0_{i_0}$ to output node $O^L_{i_L}$ passing hidden nodes $O^1_{i_1}, \ldots, O^{L-1}_{i_{L-1}}$, and $w_l(i_{l-1}, i_l)$ is the weight of the edge connecting nodes $O^{l-1}_{i_{l-1}}$ and $O^l_{i_l}$. [3]

For ease of reference, we denote path $(i_0, \cdots, i_L)$ as p, whose value $v_p(w) = \prod_{l=1}^{L} w_l(i_{l-1}, i_l)$, active status $a_p(w, x) = \prod_{l=1}^{L-1} \mathbb{I}(o^l_{p_l}(w, x) > 0)$. Thus, the output can be computed as follows:

$$N^k_w(x) = \sum_{p:i_L=k} v_p(w) \cdot a_p(w, x) \cdot x_{i_0}. \tag{2}$$

It is well known that ReLU neural networks are positively scale invariant (LeCun et al. (2015)Neyshabur et al. (2015)). That is, if the incoming weights of one hidden node or a feature map are multiplied by a positive constant c and its outgoing weights are divided by the same constant, the network with the new weights will generate exactly the same output for any input as the original network.

2.2 Related Works

There have been some prior works that take the positively scale invariance of ReLU networks into consideration. For example, the gradients are normalized by path norm in (Neyshabur et al. (2015)), and by the norm of the weights in (Badrinarayanan et al. (2015)). In (Dinh et al. (2017)), researchers noticed that some measures such as sharpness are no longer suitable measurements to describe ReLU NNs because of positively scale invariance property. Motivated by these works, our goal is to seek a compact space and directly implement optimization in the new space instead of in the weight space, which optimize the value of paths directly no matter how the weights are distributed so that to remove entirely the problem of unbalanced weight norms along a path. The most related work is Path-SGD (Neyshabur et al. (2015)), which also considered the geometry inspired by path norm. This work has the following differences with ours: (1) they didn't consider the dependency between values of paths. It caused that the path-regularized gradient that Path-SGD used is an approximate solution of the steepest descent direction with respect to the path regularizer (See the sentence followed by Eq(6) in (Neyshabur et al. (2015))). (2) the theoretical background of path norm is based on the generalization theory. If the goal is improve generalization, we can also design regularizer in our new compact space. In this sense, it is not comparable to ours.

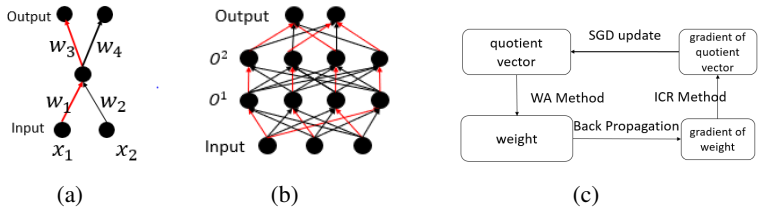
Another line of research studies the invariance property of neural networks induced by the permutation operations to hidden nodes within the same layer. For example, new initialization conditions or preconditioning methods were proposed based on this property in (Goodfellow et al. (2013)Jiang & Guo (2013)Ravanbakhsh et al. (2017)). Although this kind of invariance is out of the scale of our paper, we think our way of characterizing the positively scale invariance of ReLU networks may shed some insights on the way of handling the permutation invariance in a more direct way.

There are also works study the weight redundancy to do model compression (Arora et al. (2018); Cheng et al. (2017)). For example, (Arora et al. (2018)) develop new measures to describe the model capacity and prove that there exists smaller model have comparable generalization ability through the new measure. In this sense, they say the weights are redundant. However, the redundancy in their research is not directly caused by the positively scale invariance. The redundancy brought by positively scale invariance still exists after model compression as long as it uses ReLU to be the activation function in the compressed network.

3 Quotient Space of ReLU Networks

In this section, we seek a space which we can directly optimize the paths in it. First, we characterize the redundancy of a ReLU neural network by studying its quotient space about the equivalence induced by positive scaling operators. Second, we study the algebraic structure of the quotient space.

[3] The paths across the bias node can also be described in the same way. For simplicity, we omit the bias term.



(a) (b) (c)

Figure 1: (a) An example to motivate skeleton method; (b) General case to illustrate skeleton paths; (c) Flow chart of Quotient SGD.

3.1 Definition of the Quotient Space

Assume $g_{c,O}$ is a basic positive scaling operator with constant $c > 0$ on the weights of hidden node O, i.e.,

$$g_{c,O}(w) = \begin{cases} c \cdot w & w \in O_{in} \\ \frac{1}{c} \cdot w & w \in O_{out} \\ w & others \end{cases}$$

where O_{in} and O_{out} are the incoming and outgoing weights set of hidden node O. We consider the (general) positive scaling operators, denoted as below

$$\mathcal{G} = \left\{ g_{c^{L-1}_{h_{L-1}}, O^{L-1}_{h_{L-1}}} \circ \cdots \circ g_{c^l_{i_l}, O^l_{i_l}} \circ \cdots \circ g_{c^1_1, O^1_1}; c^l_{i_l} > 0, i_l \in [h_l], l = 1, \cdots, L-1 \right\},$$

where $\circ$ denotes function composition. [4] Clearly, if there exists an operator $g \in \mathcal{G}$ to make $w = g(w')$, ReLU networks N_w and $N_{w'}$ will generate the same output for any fixed input x. We define the following quotient space which is related to positive scale operators.

Definition 3.1 *Consider two ReLU networks with weights $w, w' \in \mathcal{W}$. If $\exists g \in \mathcal{G}$ such that $w = g(w')$, we say w and w' are positively scale equivalent and we denote $w \sim_{\mathcal{G}} w'$. The quotient space induced by positively scale equivalence is denoted as $\mathcal{W}/\sim_{\mathcal{G}} = \{\{w' \in \mathcal{W} : w \sim_{\mathcal{G}} w'\} : w \in \mathcal{W}\}$.*

Next, we investigate the properties of the quotient space $\mathcal{W}/\sim_{\mathcal{G}}$. The following theorem tells us $w \sim_{\mathcal{G}} w'$ if and only if the paths of N_w and $N_{w'}$ have the same values and active statuses (please refer to the Appdendix for the proof).

Theorem 3.2 *Consider two ReLU neural networks with weights $w, w' \in \mathcal{W}$. We have that $w \sim_{\mathcal{G}} w'$ iff for $\forall$ path p and $\forall$ input $x \in \mathcal{X}$, we have $v_p(w) = v_p(w')$ and $a_p(w, x) = a_p(w', x)$.*

Roughly speaking, Theorem 3.2 shows that if two ReLU neural networks N_w and $N_{w'}$ have the same value and active status of all paths for any fixed input, they are regarded as the same point in the quotient space, and vice versa. Thus, a directly idea is to implement the optimization algorithms to update the value of paths but not weights considering the redundancy of weight space. It motivates us to study the algebraic structure of the quotient space, i.e., the relation between values of paths, which will be discussed next.

3.2 Algebraic Structure of the Quotient Space

We can not directly update the values of paths because the paths have overlaps with each other and therefore their values and active statuses are not independent. In this section, we use linear algebra to describe the dependency among values of different paths. Suppose that the total number of paths is n and the weight w is a m-dimensional vector. For ease of reference, we index the paths as $p_1, p_2, \cdots, p_n$ and the elements of weight as $w = (w_1, \cdots, w_m)$ [5]. We consider the following generalized addition and scalar-multiplication operation in vector space $\mathbb{R}^m$ as: $w \oplus w' =$

[4] Please note that the positive scaling operators are defined for fixed indices of nodes, which means that permutations to nodes are not considered in our analysis.

[5] In the following context, w with a subscript is no longer denote the weight matrix of one layer.

$(w_1 \cdot w'_1, \cdots, w_m \cdot w'_m)$, $k \odot w = (w_1^{\ln k}, \cdots, w_m^{\ln k})$[6] where $w, w' \in \mathbb{R}^m, k \in \mathbb{R}^+$. The calculation of the values $v_{p_1}(w), v_{p_2}(w), \cdots, v_{p_n}(w)$ can be described as:

$$(v_{p_1}(w), v_{p_2}(w), \cdots, v_{p_n}(w)) = (w_1, w_2, \cdots, w_m) \odot A \tag{3}$$

$$= (\prod_{i=1}^{m} w_i^{\ln a(i,1)}, \prod_{i=1}^{m} w_i^{\ln a(i,2)}, \cdots, \prod_{i=1}^{m} w_i^{\ln a(i,m)}), \tag{4}$$

where $a(i,j)$ is the element at i-th row and j-th column of A, and the value of $a(i,j)$ is either e or 1 [7]. We call A the structure linear mapping matrix (abbreviated as structure matrix) of the ReLU NN, which determines a map from the weight space to vector space composed by values of paths. For example, consider a simple feedforward neural network (FNN) shown in Figure 1(a). Since path values $v_{p_1}(w) = w_1 w_3, v_{p_2}(w) = w_1 w_4, v_{p_3}(w) = w_2 w_3, v_{p_4}(w) = w_2 w_4$, we have

$$(v_{p_1}(w), v_{p_2}(w), v_{p_3}(w), v_{p_4}(w)) = (w_1, w_2, w_3, w_4) \odot \begin{bmatrix} e & e & 1 & 1 \\ 1 & 1 & e & e \\ e & 1 & e & 1 \\ 1 & e & 1 & e \end{bmatrix}. \tag{5}$$

Moreover, we can see the dependency among the values, i.e., $v_{p_4}(w) = \frac{v_{p_2}(w) \cdot v_{p_3}(w)}{v_{p_1}(w)}$. In order to decouple the values of paths, we need to get the maximal linearly independent column vectors of the structure matrix A [8]. We define the skeleton paths as follow.

Definition 3.3 *A group of paths are called skeleton paths if their corresponding columns in structure matrix A compose a maximal linearly independent column vectors of structure matrix A.*

We call other paths *non-skeleton path*. To get the maximal linearly independent column vectors of A is time cost if the size of the network is big. We design an efficient algorithm called *Skeleton Method* to construct skeleton paths, which will be introduced in Section 4. We can calculate the number of skeleton paths through the rank of A. Then we have the following theorem.

Theorem 3.4 *The number of skeleton paths is #(weights)-#(hidden nodes) or #(weights)-#(feature maps) for ReLU FNN and ReLU CNN respectively.*

For ReLU NN N_w, we use $q_i, i = 1, \cdots, \tilde{n}$ to denote the skeleton paths where $\tilde{n} = \#$ weights $-\#$ (hidden nodes).A directly fact following Definition 3.3 is that all the values of non-skeleton paths can be calculated by the values of skeleton paths through multiplication and division operators. If we have the values of skeleton paths and regard $w_1, \cdots, w_m$ are unknown variables, the number of free unknown variables is $m - \tilde{n}$ (Lay). Assume $w_{s,1}, \cdots, w_{s,m-\tilde{n}}$ are selected to be the free variables which we call them *skeleton weights*, we have the following lemma.

Lemma 3.5 *For the active status of any given path, it's value can be calculated using the values of skeleton paths and the signs of the skeleton weights.*

Based on the above lemma and Theorem 3.2, we can obtain the following characterization of the quotient space of ReLU networks.

Theorem 3.6 *Consider two ReLU neural networks with weights $w, w' \in \mathcal{W}$, we have $w \sim_{\mathcal{G}} w'$ iff $v_{q_i}(w) = v_{q_i}(w'), \forall i = 1, \cdots, \tilde{n}$ and $sgn(w_{s,j}) = sgn(w'_{s,j}), \forall j = 1, \cdots, \tilde{m}$, where $\tilde{n}$ is the number of skeleton paths, and $\tilde{m}$ is the number of hidden nodes.*

Therefore, the quotient space $\mathcal{W}/\sim_{\mathcal{G}}$ of ReLU network can be represented as

$$\mathcal{W}/\sim_{\mathcal{G}} = \{\rho(w) \triangleq (v_{q_1}(w), \cdots, v_{q_{\tilde{n}}}(w), sgn(w_{s,1}), \cdots, sgn(w_{s,\tilde{m}})), w \in \mathcal{W}\} \subset \mathbb{R}^{\tilde{n}} \times \{-1, 1\}^{\tilde{m}}.$$

[6] If $w_i < 0$, $w_i^{\ln k} := sgn(w_i)|w_i|^{\ln k}$.

[7] Please note that e and 1 are the zero element and identity element in the generalized linear space.

[8] Please note that the linearly independent is about the generalized addition and multiplication operators. For convenience, we can take log transformation of elements in matrix A, then we use the traditional techniques to get the maximal linearly independent group of column vectors.

Because the number of skeleton paths is smaller than the number of weights and the signs of the outgoing skeleton weights can only be $+1$ or -1, we can see that the Hausdorff dimension of the quotient space is $\tilde{n}$, which is smaller than that of the weight space.

As aforementioned, the outputs of neural networks can be calculated according to paths (in Eqn (1)), and the information on paths (i.e., the values and the activation statuses) are determined by the quotient vectors, i.e.,

$$N_\rho^k(x) = \sum_{p:p_L=k} v_p(\rho) \cdot a_p(x,\rho) \cdot x_{p_0},$$

where $\rho \in \mathbb{R}^{\tilde{n}} \times \{-1,1\}^{\tilde{m}}$ is a vector in the quotient space. Therefore the loss function can be represented as $l(N_\rho^k(x), y)$. We prove the following theorem showing that the redundancy of the weight space bring in high-dimensional manifold of critical points to the loss surface.

Theorem 3.7 *Denote $C(\mathcal{W})$ and $C(\mathcal{W}/\sim_{\mathcal{G}})$ as the sets with the critical points of the loss surface in the weight space and quotient space respectively. Then the dimensionality of $C(\mathcal{W}/\sim_{\mathcal{G}})$ is smaller than that of $C(\mathcal{W})$.*

According to the above theoretical results, we only need to update the value of the quotient vector (instead of all paths) in quotient space and Theorem 3.7 shows the benefit to do this.

4 Algorithm: Quotient SGD

In this section, we will introduce the *Quotient stochastic gradient descent* algorithm (abbreviated as Quotient SGD) that optimizes ReLU neural network models in the quotient space. This new algorithm makes use of three procedures, named Skeleton Method, Inverse-Chain Rule (ICR) and Weight-Allocation (WA), respectively, to ensure efficient computations (with almost the same complexity of forward and backward propagations in the weight space).

4.1 Skeleton Method

Before the training process, we construct the skeleton edges (the weight on whom is skeleton weight) and skeleton paths according to the Skeleton Method. According to the definition of skeleton paths, they are related to the maximal linearly independent vectors of A. To get the maximal linearly independent vectors is time cost if the size of the network is big. We design an efficient algorithm called *Skeleton Method* to construct skeleton paths. Due to space limitation, we only show skeleton method for the case that the number of hidden nodes at each hidden layer is equal for ReLU FNN and we put the general case in the Appendix.

1. Construct skeleton edges: for weight matrix $w_2, \cdots, w_{L-1}$, we select diagonal elements to be the *skeleton edges*. For weight matrix w_1, we select the element $w_1(j \mod d, j)$ for column j, $j = 1, \cdots, h_1$ to be the *skeleton edges*. For weight matrix w_L, we select the element $w_L(j, j \mod K)$ for row j, $j = 1, \cdots, h_{L-1}$ to be the *skeleton edges*. The edges are not be selected are called non-skeleton edges. Figure 1(b) for a vivid illustration (red edges in the figure are skeleton edges).

2. Construct skeleton paths: A path which contains no more than one non-skeleton edges is a skeleton path. [9] For example, in Figure 1(b), the paths in red color and the paths with only one black edge are all skeleton paths. All the other paths are non-skeleton paths.

After the construction of skeleton edges and skeleton paths, in each iteration, we need to calculate the gradient of ρ_i and update it by SGD, i.e.,

$$\rho_i^{t+1} = \rho_i^t - \eta_t \frac{\partial l(N_\rho^k(x); S^t)}{\partial \rho_i}\Big|_\rho, i = 1, \cdots, \tilde{n}, \tag{6}$$

where S^t is the mini-batch training data in iteration t. For the sgn elements in ρ, randomly initialized the signs and keep them fixed in the training. In the next subsection, we design inverse-chain-rule method to compute the gradients of quotient vector ρ efficiently.

[9] We put the proof of this statement in the Appendix.

Algorithm 1 Quotient SGD

Require: initialization w^0, learning rate η_t, training data set D.

1. Construct skeleton edges using Skeleton Method and use mask matrix $G_1, \cdots, G_L$ with elements $0, 1$ to distinguish them with non-skeleton edges.

for $t = 1, \cdots, T$ **do**

2. Implement feed forward process and back propagation process to get δw^t.
3. Calculate $R^t(\rho_j : w_i) = 1 - \eta_t \frac{w_j^t \cdot \delta_{w_j}^t - w_j^t \cdot \sum_{\rho_i : w_j} w_i^t \cdot \delta_{w_i}}{(\rho_j^t)^2}$, for $i = 1, \cdots, z$.
4. Update the weights as

$w_j^{t+1} = w_j^t \cdot R^t(\rho_j : w_i)$ and $w_i^{t+1} = \frac{w_i^t - \eta_t \cdot \frac{\delta_{w_i}^t \cdot (w_i^t)^2}{(\rho_i^t)^2}}{R^t(\rho_j : w_i)}$.

end for

Ensure: w^T.

4.2 Inverse-Chain-Rule (ICR) Method

The basic idea of the ICR method is to connect the gradients w.r.t. weights and those w.r.t. to the quotient vectors by exploring the chain rules in both directions. That is, given sgn elements in ρ, we have,

$$\frac{\partial l(w; x, y)}{\partial w_j} = \sum_{i=1}^{\tilde{n}} \frac{\partial l(\rho; x, y)}{\partial \rho_i} \cdot \frac{\partial \rho_i}{\partial w_j}, j = 1, \cdots, m. \tag{7}$$

For ease of reference, we denote $\frac{\partial l(w;x,y)}{\partial w}$ and $\frac{\partial \rho_i}{\partial w}$ as δ_w and δ_{ρ_i} respectively. We first compute the gradients w.r.t. weights δ_w though standard back propagation. Then we solve Eqn(7) to obtain the gradients w.r.t. skeleton paths δ_{ρ_i}. We call this process *Inverse-Chain-Rule Method*.

Next, we show that Eqn(7) can be solved in a highly efficient manner. Suppose that $q_i (i = 1, \cdots, z)$ is the skeleton path containing one non-skeleton edge (denoted as w_i), and $q_j (j = z + 1, \cdots, \tilde{n})$ is the skeleton path containing skeleton edges only, and w_j is its skeleton edge at layer 1.

Given the following facts: (1) $\frac{\partial \rho_i}{\partial w} = \frac{\rho_i}{w} = \frac{v_{q_i}(w)}{w}$ if w is contained in q_i; (2) w_i is contained in only one skeleton path ρ_i and w_j is contained in only one ρ_j and some ρ_i, it is easy to solve Eqn (7) and obtain the following results: [10]

$$\delta_{\rho_i} = \frac{\delta_{w_i} \cdot w_i}{\rho_i} \quad \textit{and} \quad \delta_{\rho_j} = \frac{w_j \cdot (\delta_{w_j} - \sum_{\rho_i : w_j} \delta_{\rho_i} \cdot \frac{\rho_i}{w_j})}{\rho_j}, \tag{8}$$

where $\rho_i : w_j$ denotes the ρ_i which contains w_j.

Please note that the calculation of δ_{ρ_j} depends on δ_{ρ_i}, therefore we need to calculate δ_{ρ_i} first. Based on the above deductions, we eventually have the gradients of the quotient vector. Then we can update the quotient vector using stochastic gradient $\rho^{t+1} = \rho^t - \eta_t \delta_\rho^t$ at the $(t + 1)$-th iteration.

4.3 Weight-Allocation (WA) Method

After the quotient vector is updated by SGD, in a new iteration, we employ ICR again by leveraging BP with the new weight. Thus, we need to project the updates on paths back to the updates of weights.

Please note that the value of the skeleton path is the product of all weights on it. Assume $\rho_i = \prod_{l=1}^{L} w_{i_l}$, then the update ratio on skeleton path q_i (denoted as $R^{t+1}(\rho_i)$) and the update ratio on weight w_{i_l} (denoted as $r^{t+1}(w_{i_l})$) at iteration $t + 1$ have the following relationship:

$$R^{t+1}(\rho_i) \triangleq \frac{\rho_i^{t+1}}{\rho_i^t} = \prod_{l=1}^{L} \frac{w_{i_l}^{t+1}}{w_{i_l}^t} = \prod_{l=1}^{L} r^{t+1}(w_{i_l}). \tag{9}$$

Assume that we have already obtained the update ratio for all the skeleton paths $R^{t+1}(\rho_i)$ according to ICR method and the SGD update rule. Then by solving Eqn (9) with unknowns $r^{t+1}(w_{i_l})$, we can

[10] These results are established for the case where $h_1 = \max\{h_1, \cdots, h_{L-1}\}$. If it is not the case, we still need to solve a small number of equations.

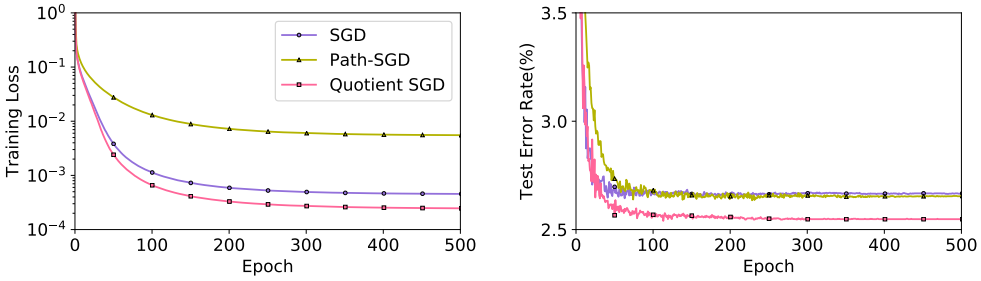


Figure 2: Experiments on FNN

project the updates to the weight space, and obtain the new weights accordingly. Considering the number of equations (equal to the number of skeleton paths) is less than the number of unknown variables ($\#weight$), we have many different feasible solutions. For sake of efficiency, we choose one specific solution as the allocation rule, which we call the *Weight-Allocation Method*, as described below (we inherit the same notations from Section 4.2, such as q_i, w_i and q_j, w_j).

First, for ρ_j, we allocate its update ratio $R^{t+1}(\rho_j)$ to the weight w_j and set the ratios of other skeleton weights contained in ρ_j as 1. Then for ρ_i, before the allocation, it needs to divide the ratios of the skeleton weights contained in it and allocate the remaining value to w_i, i.e.,

$$r^{t+1}(w_j) = R^{t+1}(\rho_j); j = z+1, \cdots, \tilde{n} \tag{10}$$

$$r^{t+1}(w_i) = \frac{R^{t+1}(\rho_i)}{\prod_{w_j \in \rho_i} r^{t+1}(w_j)}; i = 1, \cdots, z \tag{11}$$

$$r^{t+1}(w_s) = 1; s = \tilde{n}+1, \cdots, m, \tag{12}$$

where $w \in \rho_i$ means that w is contained in ρ_i. Finally, we can update the weights using the projected ratios as $w^{t+1} = w^t \cdot r^t(w)$.

Please note by combining the ICR and WA methods, we can obtain the explicit update rule for Quotient SGD, which is concluded in Algorithm 1. In this way, we obtain the correct quotient gradients. The extra computational complexity of the ICR and WA methods are far lower than that of forward and backward propagation, and can therefore be neglected in practice. For ease of reference, we illustrate the flow chart of Quotient SGD in Figure 1(c).

5 Experiments

In this section, we report our experimental results to demonstrate the efficiency of Quotient SGD. In our experiments, we tested fully connected network (FNN), convolutional networks (CNN), and ResNet, all with ReLU activation functions. We used two benchmark datasets: MNIST (LeCun (1998)) and CIFAR-10 (Krizhevsky & Hinton (2009)) [11].

We studied FNN on the MNIST dataset and take vanilla SGD and Path-SGD as baselines. The structure of the FNN model contains five fully-connected layers with 64 hidden units along with ReLU activations. The learning rates follow an exponential decay strategy as $\eta_t = e^{-\alpha t}$(Ruder (2016); Hinton et al. (2012)). We searched for the best α from $[1, 0.5, 0.1, \cdots, 10^{-6}]$ for SGD and used the same α for Quotient SGD and Path-SGD. We studied CNN/ResNet on the CIFAR-10 dataset and take vanilla SGD as the baseline [12]. The CNN model we used is a 34-layer PlainNet with batch normalization. The ResNet model also contains 34 layers, and contains batch normalization. The experimental settings such as learning rates were determined according to (He et al. (2016)).

In all of the experiments, the training minibatch size is set as 64 for each task. All the algorithms use similar random initialization strategies, except that the weights on skeleton edges (except skeleton

[11] We put the details of Skeleton Methods and ICR methods for CNN and ResNet in the Appendix.

[12] Path-SGD was not implemented on CNN in their original paper, so we do not have a good reference for the experiments.

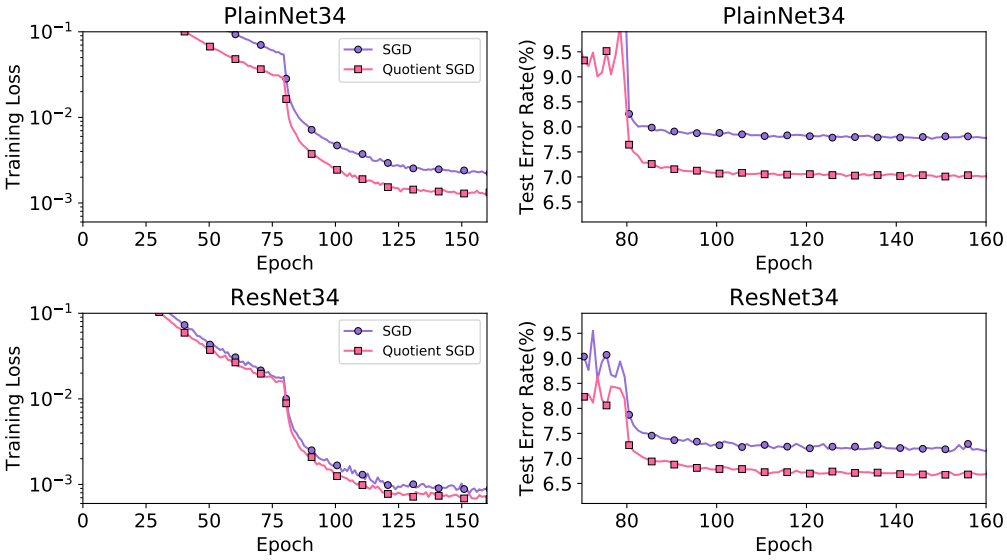


Figure 3: Experiments on CNN and ResNet

edges in layer 1) are randomly initialized as 1 or -1 in Quotient SGD. For each task, we report the training loss curvature and the test accuracy curvature with respect to the number of passed epochs. Each figure shows the average training loss of ten experiments with random initializations under the same setting.

Figure 2 shows the results of training loss and test accuracy for Quotient SGD, SGD and Path-SGD on the MNIST dataset, while Figure 3 shows the results for Quotient SGD and SGD on the CIFAR-10 dataset. From these figures, we can see that on average, Quotient SGD achieves both lower training loss and better test accuracy than the baselines that perform optimizations in the weight space. This verifies our claim that optimizing the loss function in the quotient space is a better choice. Furthermore, the improvement of the training loss on ResNet is not as significant as those for FNN and PlainNet. Our explanation is that the redundancy of ResNet is smaller because of the skip-connection and batch normalization. [13]

6 Conclusion

In this paper, we study the quotient space for ReLU neural networks and proposed a novel optimization algorithm called Quotient SGD. Specifically, we characterize the quotient space by using skeleton paths of the ReLU networks, and demonstrate that the dimensionality of the quotient space is significantly lower than the original weight space, which indicates the redundancy of the weight space. In order to remove the challenges caused by the redundancy, we represent the loss function in the quotient space and then propose inverse-chain-rule (ICR) and weight allocation (WA) methods to enable efficient stochastic gradient descent in the quotient space. We have conducted extensive experiments to verify the empirical effectiveness of our proposed approach. In the future, we will examine the performance of Quotient SGD on more large-scale tasks and investigate the quotient space for other invariant operators and other neural network models.

References

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

[13] Please note that we have not used any other tricks like momentum and weight decay in our implementations.

Vijay Badrinarayanan, Bamdev Mishra, and Roberto Cipolla. Symmetry-invariant optimization in deep networks. *arXiv preprint arXiv:1511.01754*, 2015.

David Balduzzi, Hastagiri Vanchinathan, and Joachim M Buhmann. Kickback cuts backprop's red-tape: Biologically plausible credit assignment in neural networks. In *AAAI*, pp. 485–491, 2015.

David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? *arXiv preprint arXiv:1702.08591*, 2017.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

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

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Bin Jiang and Hua Guo. Permutation invariant polynomial neural network approach to fitting potential energy surfaces. *The Journal of chemical physics*, 139(5):054112, 2013.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

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

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

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

David C Lay. *Linear Algebra and its applications, 1997*. Addison Wesley Longman, Inc. ISBN 0-201-76717-1.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Behnam Neyshabur, Ruslan R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2422–2430, 2015.

Joachim Ott, Zhouhan Lin, Ying Zhang, Shih-Chii Liu, and Yoshua Bengio. Recurrent neural networks with limited numerical precision. *arXiv preprint arXiv:1608.06902*, 2016.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. *arXiv preprint arXiv:1702.08389*, 2017.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. *arXiv preprint arXiv:1611.00179*, 2016.

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