OPTIMIZATION OF RELU NEURAL NETWORKS IN QUOTIENT SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

It is well known that neural networks whose activation functions are rectified linear units (ReLU) are positively scale invariant. This will induce many classes of equivalent neural networks in the weight space. Without awareness of such equivalent classes, conventional algorithms like stochastic gradient descent may encounter serious issues when optimizing the neural networks. To tackle this problem, in this paper, we conduct a formal study on the positive scale equivalence of ReLU networks, in particular, regarding the quotient space induced by the positive scale equivalence, in which all equivalent ReLU networks correspond to the same vector. It is clear that the quotient space is more compact, and requires much fewer dimensions to represent a given ReLU network than the weight space. Furthermore, in the quotient space, the hyperspace composed by the critical points is also of lower dimensions, which will bring convenience to the optimization procedure. With these insights, we design a new algorithm to optimize the loss function directly in the quotient space, which we call *Quotient Stochastic Gradient Descent* (abbreviated as Quotient SGD). The new algorithm first identifies skeleton paths for a ReLU network, and then applies an inverse chain rule to compute the gradient in the quotient space. After that, a weight allocation method is employed to project the update in the quotient space back onto the weights of the original neural network. Our experiments show that Quotient SGD could significantly outperform the conventional SGD algorithm in optimizing ReLU networks, on a number of diverse benchmark datasets.

1 INTRODUCTION

Over the past ten years, neural networks with rectified linear hidden units (ReLU) (Hahnloser et al., 2000) as activation functions have demonstrated the power in many important applications, such as information system (Cheng et al., 2016; Wang et al., 2017), image classification (He et al., 2016a; Huang et al., 2017), text understanding (Vaswani et al., 2017), etc. These networks are usually trained with *Stochastic Gradient Descent* (SGD), where the gradient of loss function with respect to the weights can be efficiently computed via back propagation method (Rumelhart et al., 1986).

Recent studies (Neyshabur et al., 2015; LeCun et al., 2015) show that ReLU networks have positively scale-invariant property, i.e., 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. SGD with gradient in weight space does not consider the above positively scaling equivalence and may lead to unstable training and bad model performance, especially for an unbalanced network (Neyshabur et al., 2015).

In this paper, we ask two questions: 1) what will the space be if positively-scale-equivalent models are taken as one element; 2) can we benefit from optimizing ReLU networks in such a compact space.

To answer the above questions, we introduce a quotient space induced by the positively scaling equivalence, which we call it *quotient space* for simplicity [1] and design efficient optimization algorithms in the quotient space.

[1] Intuitively, the positively scale-equivalent ReLU networks are regarded as the same vector in quotient space.

More specifically, we prove that ReLU networks that are equivalent correspond to the same vector in the quotient space and the dimensionality of quotient space is smaller than that of weight space. The proof is composed of three steps: 1) We represent the model by paths (i.e., the products of all the weights and activation status on the paths) instead of weights, and prove that having the same values of paths and activation status is a sufficient and necessary condition that two ReLU networks are positively scale-equivalent; 2) Since there is inter-dependency among the paths, (see Figure 1), we use algebraic technique to find a group of independent paths, which we call them skeleton paths. Then we prove that to determine an equivalent class, there's no need to keep the values of all paths and activation status to be the same, but the values of skeleton paths and signs of some weights. Thus the vector in the quotient space is composed by skeleton paths and signs of some weights, which we call it *quotient vector*; 3) By analyzing the number of skeleton paths using algebraic technique, we prove that representing ReLU network in quotient space requires fewer dimensions than in weight space. To be specific, the dimensionality of representing ReLU network in quotient space is H less than the weight space, where H is the total number of hidden units in a multi-layer perceptron (MLP) or feature maps in a convolutional networks (CNN).

Based on the above theoretical result, a direct corollary is that the dimensionality of the hyperspace composed by critical points (such as saddle points and bad local minima) will be fewer, which will make optimization easier. Thus it provides support to optimization in quotient space instead of in weight space. Besides, based on the theoretical result, we define the equivalence ratio as $H/(\#params)$ to measure the level of positively scaling equivalence in weight space. Thus we can empirically test how severe the positively scaling equivalence will influence the optimization in weight 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). We first design *Skeleton Method* to efficiently find the skeleton paths for a ReLU network. Then we design inverse-chain rule method to compute the gradient w.r.t the quotient vector by leveraging the back propagation method. Finally we design weight-allocation method to project the update of quotient vector back to the network's weights. Please note that by using these techniques, there is almost no additional computation overhead of Quotient SGD in comparison with the standard SGD method.

We conduct experiments to show the effectiveness of Quotient SGD algorithm. We first investigate the influence of positively scaling equivalence to optimization by optimizing models with different equivalence ratio using both Quotient SGD and SGD. The experimental results show that the higher the equivalence ratio, the bigger the performance gap between Quotient SGD and SGD, which indicates the positively scaling equivalence in weight space will negatively influenced the optimization. Second, we implement Quotient SGD to deep convolutional networks on benchmark datasets, and demonstrate that Quotient SGD achieves clearly better performance than baseline optimization algorithms. The experimental results are consistent with our claim that optimizing ReLU networks in quotient space is a good choice.

2 BACKGROUNDS

2.1 RELATED WORKS

There have been some prior works that take the positively scaling equivalence 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). (Dinh et al., 2017) noticed that some measures such as sharpness are no longer suitable measurements to describe ReLU networks because of positively scaling equivalence property. 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) without optimizing the loss function in quotient space, they regularize the gradient in weight space by path norm; 2) without considering the dependency among all paths, it's hard to compute the exactly path-regularized gradients. Instead, Path-SGD approximates the gradients by updating each coordinate parameters independently.

Most recent, (Du et al., 2018) study the gradient flow in MLP or CNN models with linear, ReLU or Leaky ReLU activation, and prove the squared norms of gradient across different layers are automatically balanced and remained invariant in gradient descent with infinitesimal step size, which

indicates the necessary of a good initialization. In our work, we wouldn't care the models are balanced or not.

There are works that study the optimization or generalization of over-parameterized networks, where the number of network parameters exceeds the number of data samples. 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. (Brutzkus et al., 2018) give a complete analysis of SGD in a 2-layer over-parameterized network where the second layer does not undergo training and the data are linearly separable. In this paper, we study how to represent a given ReLU network in quotient space by fewer dimensions than weight space, with the same representation power. Please note, the dimensionality of our proposed quotient space isn't related to the amount of data samples, which is different from the over-parameterized networks.

2.2 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 $\mathcal{Y} = \mathbb{R}^K$. In the l-th layer ($l = 0, \cdots, L$), 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. We use $(i_0, \cdots, i_L)$ to denote the path starting from input feature node $O^0_{i_0}$ to output node $O^L_{i_L}$ passing hidden nodes $O^1_{i_1}, \cdots, O^{L-1}_{i_{L-1}}$ with $i_l = 1, \cdots, h_l$. We use $\mathcal{P}$ to denote the set composed by all the paths, i.e., $\mathcal{P} = \{(i_0, \cdots, i_L); i_0 \in [h_0], \cdots, i_L \in [h_L]\}$, and $\mathcal{P}_k = \{(i_0, \cdots, i_L); i_0 \in [h_0], \cdots, i_{L-1} \in [h_L - 1], i_L = k\}$. We can calculate the k-th output by paths (Balduzzi et al. (2017; 2015)), i.e.,

$$N^k_w(x) = \sum_{(i_0, \cdots, i_L) \in \mathcal{P}_k} \prod_{l=1}^{L} w^l(i_{l-1}, i_l) \cdot \prod_{l=1}^{L-1} \mathbb{I}(o^l_{i_l}(w, x) > 0) \cdot x_{i_0} \tag{1}$$

where $w_l(i_{l-1}, i_l)$ is the weight connecting nodes $O^{l-1}_{i_{l-1}}$ and $O^l_{i_l}$. [2] We denote the value of path $(i_0, \cdots, i_L)$ as $v_{(i_0, \cdots, i_L)} := \prod_{l=1}^{L} w^l(i_{l-1}, i_l)$ and the activation status as $a_{(i_0, \cdots, i_L)}(w, x) := \prod_{l=1}^{L-1} \mathbb{I}(o^l_{i_l}(w, x) > 0)$. Thus, the output can be computed as follows:

$$N^k_w(x) = \sum_{(i_0, \cdots, i_L) \in \mathcal{P}_k} v_{(i_0, \cdots, i_L)}(w) \cdot a_{(i_0, \cdots, i_L)}(w, x) \cdot x_{i_0}. \tag{2}$$

It is well known that ReLU neural networks are positively scale invariant (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.

3 Quotient Space of ReLU Networks

In this section, we define quotient space of ReLU networks, study the algebraic structure and the dimensionality of the quotient space.

3.1 Definition of the Quotient Space

We formally define the positively scaling operator. We first define a node positively scaling operator $g_{c,O}(w) : \mathcal{W} \to \mathcal{W}$ with constant $c > 0$ and one hidden node $O^l_{i_l}$ as

$$\tilde{w} = g_{c, O^l_{i_l}}(w),$$

where $\tilde{w}^l(i_{l-1}, i_l) = c \cdot w^l(i_{l-1}, i_l)$ for $i_{l-1} = 1, \cdots, h_{l-1}$; $\tilde{w}^{l+1}(i_l, i_{l+1}) = \frac{1}{c} \cdot w^{l+1}(i_l, i_{l+1})$ for $i_{l+1} = 1, \cdots, h_{l+1}$; and values of other elements of $\tilde{w}$ are the same with w.

Then we introduce a **positively scaling operator** $g(\cdot) : \mathcal{W} \to \mathcal{W}$, which transforms a weight parameter to another one via series of node positively scaling operations (each operation corresponds

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

to one hidden node). For the ease of presentation, we use $\mathcal{O}$ to denote the set of all the hidden nodes in the network and index the element as O_γ for $\gamma = 1, 2, ..., H$, where H denotes the number of elements in $\mathcal{O}$. Then, $g(\cdot)$ can be written as

$$g(\cdot) := g_{c_1,O_1} \circ g_{c_2,O_2} \circ \cdots \circ g_{c_H,O_H}(\cdot), \tag{3}$$

where $\circ$ denotes function composition. We can see that $g(\cdot)$ is uniquely determined by a vector $(c_1, c_2, ..., c_H)$ and each c_γ is positively scale parameter at node O_γ where $c_\gamma > 0$ for all $\gamma = 1, ..., H$. We then collect all the $g(\cdot)$ together to form a set $\mathcal{G}$. It is easily to check that $\mathcal{G}$ forms a transformation group and we call the group action of $\mathcal{G}$ on $\mathcal{W}$ as $\mathcal{G}$-action (Please refer to Section 8 in Appendix.)

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 equivalent class which is related to positively scaling operators.

Definition 3.1 *($\mathcal{G}$-**equivalent class**) Consider two ReLU networks with weights $w, w' \in \mathcal{W}$ and the positively scaling transformation group $\mathcal{G}$. We say w and w' are positively scale-equivalent (abbreviated as $\mathcal{G}$-equivalent) if $\exists g \in \mathcal{G}$ such that $w = g(w')$, denote as $w \sim_\mathcal{G} w'$. The $\mathcal{G}$-equivalent class of ReLU network N_w is defined as a set $\mathcal{G}_w := \{w' : w \sim_\mathcal{G} w'\}$. We say a function $f : \mathcal{W} \to \mathbb{R}$ is positively scale-invariant (abbreviated as $\mathcal{G}$-invariant) if $f(w) = f(g(w)), \forall g \in \mathcal{G}$.*

Given $\mathcal{G}$-action on $\mathcal{W}$, the equivalence relation "$\sim_\mathcal{G}$" partitions $\mathcal{W}$ into $\mathcal{G}$-equivalent classes. In order to handle the positively scaling equivalence, we seek the invariant variables under $\mathcal{G}$-action, i.e., functions that are $\mathcal{G}$-invariant. The following theorem and corollary shows that the values and activation statuses of paths are invariant variables under transformation group $\mathcal{G}$.

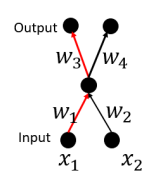
Theorem 3.2 *Consider two ReLU neural networks with weights $w, w' \in \mathcal{W}$. We have that $w \sim_\mathcal{G} w'$ iff for $\forall$ path $(i_0, \cdots, i_L) \in \mathcal{P}$ and any fixed input $x \in \mathcal{X}$, we have $v_{(i_0,\cdots,i_L)}(w) = v_{(i_0,\cdots,i_L)}(w')$ and $a_{(i_0,\cdots,i_L)}(w, x) = a_{(i_0,\cdots,i_L)}(w', x)$.*

Corollary 3.3 *$\forall g \in \mathcal{G}$, $\forall$ path $(i_0, \cdots, i_L) \in \mathcal{P}$ and any fixed input $x \in \mathcal{X}$, we have $v_{(i_0,\cdots,i_L)}(w) = v_{(i_0,\cdots,i_L)}(g(w))$ and $a_{(i_0,\cdots,i_L)}(w, x) = a_{(i_0,\cdots,i_L)}(g(w), x)$.*

Corollary 3.3 shows that values of paths have positively scale-invariant property of that weights don't have, i.e., $\exists g \in \mathcal{G}, s.t., w \neq g(w)$, we propose to optimize the values of paths instead of weights.

3.2 Independent Invariant Variables under $\mathcal{G}$-action

Although Theorem 3.2 shows the values of paths are invariant variables under $\mathcal{G}$-action, we found that the paths have inner-dependency and therefore their values and activation statuses are not independent (see Figure 1). In order to describe this dependency clearly, we seek a new representation of paths.



$$A = \begin{bmatrix} e & e & 1 & 1 \\ 1 & 1 & e & e \\ e & 1 & e & 1 \\ 1 & e & 1 & e \end{bmatrix}$$

Figure 1: This is a simple ReLU network with one hidden node. Suppose path values are $v_1(w) = w_1w_3, v_2(w) = w_1w_4, v_3(w) = w_2w_3, v_4(w) = w_2w_4$, we can see the inner-dependency between them, i.e., $v_4(w) = \frac{v_2(w)\cdot v_3(w)}{v_1(w)}$. A is the structure matrix of this example.

First, we define the addition operation "$\oplus$" and scalar-multiplication operation "$\odot$" in space $(\mathbb{R}/\{0\})^m$ as: $w \oplus w' = (w_1 \cdot w'_1, \cdots, w_m \cdot w'_m)$, $a \odot w = (sgn(w_1)|w_1|^{\ln a}, \cdots, sgn(w_m)|w_m|^{\ln a})$ where $w, w' \in \mathbb{R}/\{0\}^m$ It is easily to prove that $(\mathbb{R}/\{0\})^m$ equipped with addition operation "$\oplus$" and scalar-multiplication operation "$\odot$" is vector space.(Please refer to Section 8 in Appendix.)

For ease of reference, we omit the explicit notations $w^l(i_{l-1}, i_l)$ of one weight and denote the weight vector which orders the weights by layer as $w = (w^1_1, \cdots, w^1_{z_1}, \cdots, w^L_1, \cdots, w^L_{z_L})^T$, where $z_l = h_{l-1}h_l$ is the number of weights at layer l. Then in the vector space we defined above, each path can be represented as a m-dimensional vector as:

$$(\underbrace{1, 1, e, \cdots, 1}_{z_1}, \cdots, \underbrace{1, e, \cdots, 1}_{z_L})^T,$$

where e only exists at the position that the corresponding weight is contained in the path and other positions are formed by 1[3]. We use p_i for $i = 1, \cdots, n$ to denote the path vector where n is the total number of paths. Thus the value of path p_i can be calculated by the inner product of the weight vector and the path vector, i.e., $v_{p_i}(w) = w^T \odot p_i$.

We denote the matrix composed by all path vectors as $A = (p_1, \cdots, p_n)$ and call it *structure matrix* of ReLU networks. We observe that the columns in matrix A are not linearly independent, i.e.,some columns can be linearly represented by other columns. For example in Figure 1, $p_4 = p_2 \oplus p_3 \ominus p_1$ and the corresponding values of paths have the relationship: $v_4(w) = \frac{v_2(w) \cdot v_3(w)}{v_1(w)}$. In order to find the maximal linearly independent group of matrix A in this field, we study define the concept *skeleton path* and study the rank of matrix A.

Definition 3.4 ***(Skeleton path)*** *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 design an algorithm called *Skeleton Method* to identify skeleton paths efficiently, which will be introduced in Section 4. The next theorem shows the rank of structure matrix A.

Theorem 3.5 *If A is the structure matrix for a ReLU network, then we have $rank(A) = m - H$, where m is the dimension of weight vector w and H is the total number of hidden nodes for MLP (or feature maps for CNN models) with ReLU respectively.*

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 weights and skeleton paths.We design an efficient algorithm called *Skeleton Method* to construct skeleton weights and 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 MLP models with ReLU and we put the general case which is constructed in a recursive way in the Appendix.

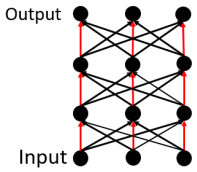


Figure 2: The weights with red color are skeleton weights.

1. Construct skeleton weights: for weight matrix $w^2, \cdots, w^{L-1}$, we select diagonal elements to be the *skeleton weights*. For weight matrix w^1, we select the element $w^1(i_1 \mod d, i_1)$ for column i_1 with $i_1 = 1, \cdots, h_1$ to be the *skeleton weights*. For weight matrix w^L, we select the element $w^L(i_{L-1}, i_{L-1} \mod K)$ for row i_{L-1} with $i_{L-1} = 1, \cdots, h_{L-1}$ to be the *skeleton weights*. The weights that are not be selected are called non-skeleton weights. Please refer to Figure 2 for illustration (red weights in the figure are skeleton weights).

2. Construct skeleton paths: A path which contains no more than one non-skeleton weights is a skeleton path. The proof of this statement could be found in supplementary. For example, in Figure 2, the paths in red color and the paths with only one black weight are all skeleton paths. All the other paths are non-skeleton paths.

According to Theorem **??**, the loss function can be represented using quotient vector and we denote the loss at training instance (x, y) as $l(\rho; x, y)$. After the construction of skeleton weights and skeleton paths, in each iteration, we need to calculate the gradient of ρ_i and update it by SGD, i.e.,

$$\rho_j^{t+1} = \rho_j^t - \eta_t \frac{\partial l(\rho; S^t)}{\partial \rho_j}\Big|_{\rho}, j = 1, \cdots, \tilde{n}, \tag{4}$$

[3] 1 and e are the additive identity and multiplicative identity in the field $\mathbb{R}/\{0\}$ equipped with "$\oplus$" and "$\odot$".

where S^t is the mini-batch training data in iteration t. For the sgn elements in ρ, we initialized the signs as $+1$ and keep them fixed in the training (Please refer to Section 11.3 in Appendix). In the next subsection, we design inverse-chain-rule method to compute the gradients of quotient vector ρ efficiently.

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 sign elements $\rho_{\tilde{n}+1}, \cdots, \rho_m$ in ρ, we have,

$$\frac{\partial l(w;x,y)}{\partial w_i} = \sum_{j=1}^{\tilde{n}} \frac{\partial l(\rho;x,y)}{\partial \rho_j} \cdot \frac{\partial \rho_j}{\partial w_i}, \quad \textit{for } i = 1, \cdots, m. \tag{5}$$

We first compute the gradients w.r.t. weights, i.e., $\frac{\partial l(w;x,y)}{\partial w_i}$ for $i = 1, \cdots, m$ though standard back propagation. Then we solve Eqn(5) to obtain the gradients w.r.t. skeleton paths, i.e., $\frac{\partial l(\rho;x,y)}{\partial \rho_j}$ for $j = 1, \cdots, \tilde{n}$. We call this process *Inverse-Chain-Rule Method*.

Next, we show that Eqn(5) can be solved in a highly efficient manner. According to skeleton method, we clarify skeleton paths into two types: 1) *type I skeleton path* which contains one non-skeleton weight; 2) *type II skeleton path* which contains skeleton weights only. We denote type I skeleton path as p_{j_1} for $j_1 = 1, \cdots, z$ and type II skeleton path as p_{j_2} for $j_2 = z+1, \cdots, \tilde{n}$. We use $w^l_{\in p_j}$ to denote the weight that contained in skeleton path p_j at layer-l. Specially, we use $w^{ns}_{\in p_{j_1}}$ to denote the non-skeleton weight that contained in type I skeleton path p_{j_1}. We also assume that $l_m = argmax_l\{h_1, \cdots, h_l, \cdots, h_{L-1}\}$.

Given the following facts: (1) $\frac{\partial \rho_j}{\partial w_i} = \frac{\rho_j}{w_i}$ if w_i is contained in p_j; (2) the non-skeleton weight $w^{ns}_{\in p_{j_1}}$ is contained in only one skeleton path p_{j_1} and the skeleton weight $w^{l_m}_{\in p_{j_2}}$ is contained in only one type II skeleton path p_{j_2} and some type I skeleton paths p_{j_1}, Eqn (5) equals to the following series of equations:

$$\frac{\partial l(\rho;x,y)}{\partial w^{ns}_{\in p_{j_1}}} = \frac{\partial l(\rho;x,y)}{\partial \rho_{j_1}} \cdot \frac{\rho_{j_1}}{w^{ns}_{\in p_{j_1}}}; \quad \frac{\partial l(\rho;x,y)}{\partial w^{l_m}_{\in p_{j_2}}} = \frac{\partial l(\rho;x,y)}{\partial \rho_{j_2}} \cdot \frac{\rho_{j_2}}{w^{l_m}_{\in p_{j_2}}} + \sum_{j_1 : w^{l_m}_{\in p_{j_2}} \in p_{j_1}} \frac{\partial l(\rho;x,y)}{\partial \rho_{j_1}} \cdot \frac{\rho_{j_1}}{w^{l_m}_{\in p_{j_2}}}, \tag{6}$$

where the summation $\sum_{j_1 : w^{l_m}_{\in p_{j_2}} \in p_{j_1}}$ is over the type I skeleton paths with $w^{l_m}_{\in p_{j_2}}$ is contained in them.

Solving the above equations, we can obtain $\frac{\partial l(\rho;x,y)}{\partial \rho_{j_1}}$ and $\frac{\partial l(\rho;x,y)}{\partial \rho_{j_2}}$ easily. Please note that the calculation of $\frac{\partial l(\rho;x,y)}{\partial \rho_{j_2}}$ depends on $\frac{\partial l(\rho;x,y)}{\partial \rho_{j_1}}$, therefore we need to calculate $\frac{\partial l(\rho;x,y)}{\partial \rho_{j_1}}$ 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 $\rho_j^{t+1} = \rho_j^t - \eta_t \frac{\partial l(\rho;S^t)}{\partial \rho_j}\big|_{\rho=\rho^t}$ 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.

We define the *path-value-ratio* of ρ_j at iteration t as $R^t(\rho_j) := \rho_j^t / \rho_j^{t-1}$ and the *weight-ratio* of w_i at iteration t as $r^t(w_i) := w_i^t / w_i^{t-1}$. Assume that we have already obtained the path-value-ratio for all the skeleton paths $R^{t+1}(\rho_j)$ according to ICR method and the SGD update rule. Then we want to project the path-value-ratios onto the weight-ratios. Because we have $\rho_j = \prod_{l=1}^L w^l_{\in p_j}$, the weight-ratios $r^t(w^1_{\in p_j}), \cdots, r^t(w^L_{\in p_j})$ obtained after the projection should satisfy the following relationship: $R^t(\rho_j) = \prod_{l=1}^L r^t(w^l_{\in p_j})$. According to this relationship, we design *Weight-Allocation Method* as described below (we inherit the same notations from Sec 4.2, such as ρ_{j_1}, ρ_{j_2} and $w^{ns}_{\in p_{j_1}}, w^{l_m}_{\in p_{j_2}}$).

Assume that we have already obtained the path-value-ratio for all the skeleton paths $R^{t+1}(\rho_j)$ according to ICR method and the SGD update rule. Then we project the path-value-ratios onto the weight-ratios. The solution after projection must satisfy the relationship $R^t(\rho_j) = \prod_{l=1}^L r^t(w^l_{\in p_j})$.

Algorithm 1 Quotient SGD

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

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

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

2. Implement feed forward process and back propagation process to get $\frac{\partial l(w;S^t)}{\partial w_i}\big|_{w=w_t}$.
3. Calculate $\frac{\partial l(\rho;x,y)}{\partial \rho_{j_2}}$ and $\frac{\partial l(\rho;x,y)}{\partial \rho_{j_1}}$ according to Eqn.6.
4. Calculate weight-ratio:
$r^t(w^{l_m}) = r^t((w^s)^{l_m}) \times G_{l_m} + r^t((w^{ns})^{l_m}) \times (1 - G_{l_m})$. For $l \neq l_m$, $r^t(w^l) = r^t((w^{ns})^l) \times (1 - G_l)$
(*Here, "$\times$" means element-wise product.)
5. Update the weights as $w^{t+1} = w^t \cdot r^{t+1}(w)$.

end for

Ensure: w^T.

First, for type II skeleton path p_{j_2}, we allocate its path-value-ratio $R^{t+1}(\rho_{j_2})$ to the weight-ratio $r^{t+1}(w^{l_m}_{\in p_{j_2}})$ for skeleton weight $w^{l_m}_{\in p_{j_2}}$, and set the weight-ratios of other skeleton weights contained in p_{j_2} as 1. Then for type I skeleton path p_{j_1}, before the allocation, it needs to divide the ratios of the skeleton weights contained in it and allocate the remaining value to non-skeleton weight $w_{\in p_{j_1}}$. In conclude, we have

$$r^{t+1}(w^{l_m}_{\in p_{j_2}}) = R^{t+1}(\rho_{j_2}); j_2 = z+1, \cdots, \tilde{n} \tag{7}$$

$$r^{t+1}(w^{l}_{\in p_{j_2}}) = 1; l \neq l_m, j_2 = z+1, \cdots, \tilde{n} \tag{8}$$

$$r^{t+1}(w^{ns}_{\in p_{j_1}}) = \frac{R^{t+1}(\rho_{j_1})}{\prod_{i:w_i \in p_{j_1}, w_i \neq w^{ns}_{\in p_{j_1}}} r^{t+1}(w_i)}; j_1 = 1, \cdots, z, \tag{9}$$

where the product $\prod_{i:w_i \in p_{j_1}, w_i \neq w^{ns}_{\in p_{j_1}}}$ is over weights that are contained in p_{j_1} except $w^{ns}_{\in p_{j_1}}$. Please note that the calculation of the denominator at the right side of Eq. (10) will be used Eq.(8) and Eq.(9), therefore we need to calculate $r^{t+1}(w^{l_m}_{\in p_{j_2}})$ first. It is easily to obtain that after weight-allocation method, for each skeleton path p_j, the relationship $R^t(\rho_j) = \prod_{l=1}^{L} r^t(w^l_{\in p_j})$. is satisfied. Finally, we can update the weights using the projected ratios as $w_i^{t+1} = w_i^t \cdot r^t(w_i)$ for $i = 1, \cdots, m$.

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.

5 Experiments

In this section, we first investigate the influence of equivalence to the optimization in weight space with different level of equivalence, and examine whether it achieves performance gain by optimization in quotient space. Then we evaluate Quotient SGD on deep convolutional network models with the aim of verifying that does our proposed optimization algorithm outperforms other baseline optimization methods, and whether it achieves better performance on benchmark datasets. Finally we extend the experiments of Path-SGD in (Neyshabur et al., 2015) to Quotient SGD, and analysis the necessary of consideration with the dependency among paths. All experiments are averaged over 5 independent trials if without explicit note.

5.1 The Influence of Equivalence

As discussed in previous section, optimization in weight space would face the positively scaling equivalence and the linearly dependence equivalence. In this section, we study the influence of equivalence to optimization for ReLU Networks, in comparison with optimization in quotient space without equivalence. As proof in Section 3, the dimensionality of weight space is H more than quotient space, where the H is the total number of hidden units in a MLP or feature maps in a CNN.

We define the equivalence ratio as $H/\#params$, where the #params is the dimensionality of weight space. We train several 2-hidden-layer MLP models on Fasion-MNIST (Xiao et al., 2017) with different number of hidden units in each layer, and study the performance gap between the models optimized by Quotient SGD and SGD. The detailed training strategies and network structures could be found in supplementary.

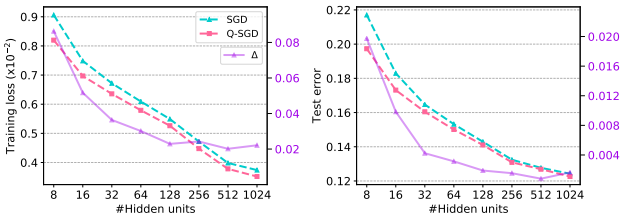


Figure 3: Training loss and test error on MLPs with ReLU. We train several 2-hidden-layer MLP models with different number of hidden units in each layer. The equivalence ratio (i.e. $H/\#params$) decreases along with the number of hidden units increasing. The number of equivalence ratio is shown in 5 in supplementary.

The experimental result is shown in Figure 3. The influence of equivalence to the optimization wouldn't show up without in comparison with the optimization in quotient space, hence we focus on the performance improvement achieved by Quotient SGD comparing with SGD. As we can see, 1). with each number of H, Quotient SGD clearly outperforms SGD on both training loss and test error, which verifies our claim of optimization loss function in quotient space is a better choice; 2). as the H increasing while the equivalence ratio decreasing, the performance gap gradually descreases as well, which indicates that the equivalence in weight space indeed improperly influenced the optimization of the models.

5.2 Deep Convolutional Network

In this section, we apply our Quotient SGD to image classification tasks and conduct experimental studies based on CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009). ResNet and its variants (He et al., 2016a;b; Zagoruyko & Komodakis, 2016; Xie et al., 2017) are popular deep convolutional ReLU models with huge successes in many image related tasks. In this section, we employ the original ResNet architecture described in (He et al., 2016a) in our experiments. Specifically, we apply skeleton method within each residual block since the residual connections break down the structure matrix described in Section 3.2. For better comparison, we also conduct our studies on a stacked deep CNN described in (He et al., 2016a) (refer to PlainNet), which suffers serious equivalence problem among the paths. We train 34 layers ResNet and PlainNet models on the datasets following the training strategies in the original paper, and investigate the optimization performance of Quotient SGD[4] comparing to the baseline SGD algorithm. The detailed training strategies could be found in supplementary. Empirically the optimization algorithms achieve better generalization performance on test dataset by combining with proper regularization methods. In this section, we focus on the performance of different optimization algorithms, and will discuss the combination of Quotient SGD and regularization in supplementary.

Table 1: Classification error rate (%) on image classification task.

		C10	C100
Plain-34	SGD	7.76 (±0.17)	36.41(±0.54)
	Q-SGD	**7.00** (±0.10)	**30.74** (±0.29)
ResNet-34	SGD	7.13 (±0.22)	28.60(±0.26)
	Q-SGD	**6.66** (±0.13)	**27.74** (±0.06)

[4]Batch normalization is widely used in modern CNN models. Please refer to supplementary for the combination of Quotient SGD and batch normalization.

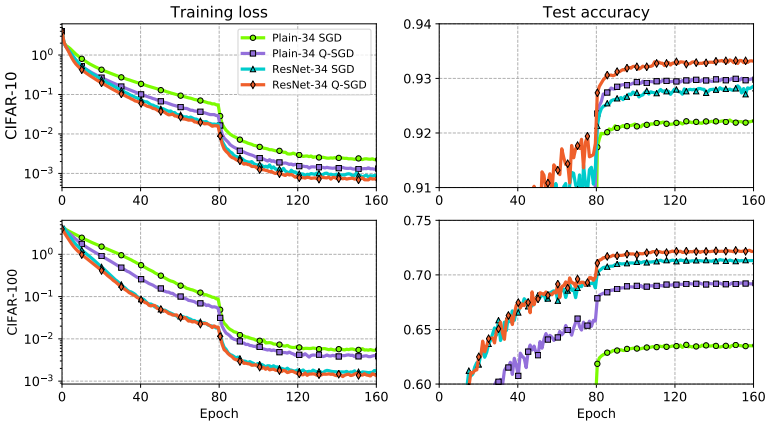


Figure 4: Training loss and test accuracy w.r.t. the number of effective passes on PlainNet and ResNet.

As shown in Figure 4 and Table 1, our Quotient SGD clearly outperforms SGD on each network and each dataset. To be specific, 1). the lowest training loss and best test accuracy are both achieved by ResNet-34 with Quotient SGD on all datasets, which indicates that Quotient SGD indeed helps the optimization of ResNet model, even the residual connection has already significantly mitigate the equivalence problem; 2). by eliminating the vast equivalence in the weight space of PlainNet model, Quotient SGD surprisingly achieves about 0.8% and 5.7% improvement respectively on both dataset, which is consistent with our theoretical analysis in previous section, and verifies the benefit of optimization in quotient space. More specifically, after applying Quotient SGD to eliminate the equivalence, Plain-34 achieves even better accuracy comparing with ResNet-34 trained by SGD, which demonstrates the influence of equivalence to optimization in weight space as well.

5.3 Comparison with Path-SGD

We further empirically investigate the performance of Path-SGD and Quotient SGD. As described in Section 2.1, Path-SGD also consider the positively scaling equivalence, but 1). instead of optimize the loss function in quotient space, Path-SGD regularizes path norm during optimization; 2). Path-SGD ignores the dependency among the paths. We extend the experiments in (Neyshabur et al., 2015) to Quotient SGD without unbalance initialization, and conduct our studies on MNIST and CIFAR-10 datasets. The detailed training strategies and description of network structure could be found in supplementary.

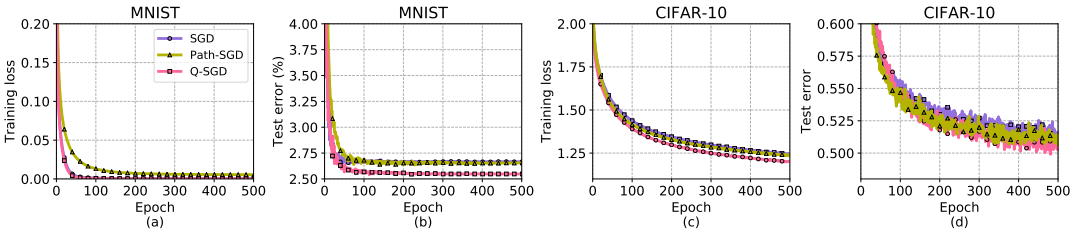


Figure 5: Experiments on CNN and ResNet

As shown in Figure 5, Path-SGD indeed shows good generalization on both dataset, and better training loss on CIFAR-10 comparing with SGD. However, with considering the dependency among the paths and optimization in quotient space, Quotient SGD achieves clearly better performance than Path-SGD, which is consistent with our theoretical analysis in the previous sections.

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.

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.

Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. SGD learns over-parameterized networks that provably generalize on linearly separable data. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJ33wwxRb`.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 7–10. ACM, 2016.

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

Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in Neural Information Processing Systems*, 2018.

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. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

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.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, pp. 3, 2017.

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

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

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.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pp. 12. ACM, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 5987–5995. IEEE, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Shuxin Zheng, Qi Meng, Huishuai Zhang, Wei Chen, Nenghai Yu, and Tie-Yan Liu. Capacity control of relu neural networks by basis-path norm. *arXiv preprint arXiv:1809.07122*, 2018.

APPENDIX: OPTIMIZATION OF RELU NEURAL NETWORKS IN QUOTIENT SPACE

The supplementary document is composed of examples of skeleton weights and skeleton paths for different MLP structures, proofs of propositions, lemmas and theorems and the additional information about the experiments in the paper Optimization of ReLU Neural Networks using Quotient Stochastic Gradient Descent .

7 NOTATIONS

Table 2: Notations

Notations	Object
m	dimension of weight space
H	total number of hidden nodes or feature maps
n	total number of paths
$\tilde{n} = m - H$	total number of skeleton paths and dimension of quotient space
$\mathcal{W} \subset \mathbb{R}^m$	weight vector space
$w = (w_1, \cdots, w_m)$	weight vector with $m = \sum_{l=1}^{L} h_{l-1} h_l$ for MLP
w^l	weight matrix at layer l with size $h_{l-1} \times h_l$ for MLP
$w^l(i_{l-1}, i_l)$	weight element in matrix w^l at position (i_{l-1}, i_l)
$O_{i_l}^l$	the i_l-th hidden node at layer l

Table 3: Index

Index	Range	Object
l	$\{1, \cdots, L\}$	index of layer
i_l	$\{1, \cdots, h_l\}$	index of hidden nodes at l-layer
$(i_L, i_{L-1}, \cdots, i_0)$	$i_l \in [h_l], l \in [L]$	explicit index of path
p	$\{1, \cdots, n\}$	index of path
p_i	$\{p_1, \cdots, p_{\tilde{n}}\} \subset \{1, \cdots, n\}$	index of skeleton path
s_j	$\{s_1, \cdots, s_{m-\tilde{n}}\} \subset \{1, \cdots, m\}$	index of skeleton weight

Table 4: Mathematical Notations

Notation	Meaning
$\#$	the number of
$/$	division or set minus
$\circ$	function composition

8 SOME CONCEPTS IN ABSTRACT ALGEBRA

Definition 8.1 ***(Transformation group)*** *Suppose that $\mathcal{G}$ is a set of transformations, and $\circ$ is an operation defined between the elements of $\mathcal{G}$. If $\mathcal{G}$ satisfies the following conditions: 1) (operational closure) for any two elements $g_1, g_2 \in \mathcal{G}$, it has $g_1 \circ g_2 \in \mathcal{G}$; 2) (associativity) for any three elements $g_1, g_2, g_3 \in \mathcal{G}$, it has $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$; 3) (unit element) there exists unit element $e \in \mathcal{G}$, so that for any element $g \in \mathcal{G}$, there is $g \circ e = g$; 4) (inverse element) for any element $g \in \mathcal{G}$, there exists an inverse element $g^{-1} \in \mathcal{G}$ of g such that $g \circ g^{-1} = e$. Then, $\mathcal{G}$ together with the operation "$\circ$" is called a transformation group.*

Definition 8.2 ***(Group action)*** *If $\mathcal{G}$ is a group and $\mathcal{W}$ is a set, then a (left) group action $\phi_{\mathcal{G},\mathcal{W}}$ of $\mathcal{G}$ on $\mathcal{W}$ is a function $\phi_{\mathcal{G},\mathcal{W}} : \mathcal{G} \times \mathcal{W} \to \mathcal{W}$ that satisfies the following two axioms (where we denote $\phi(g, w)$ as $g \cdot w$): 1) (identity) $e \cdot w = w$; 2) (compatibility) $(g \circ h) \cdot w = g \circ (h \cdot w)$ for all $g, h \in \mathcal{G}$ and all $w \in \mathcal{W}$.*

Definition 8.3 ***(Field)*** *A field is a set $\mathcal{F}$ together with two operations, called addition and multiplication that satisfy the following conditions: $\forall a, b, c \in \mathcal{F}$, 1) Associativity of addition and multiplication: $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$. 2) Commutativity of addition and multiplication: $a + b = b + a$ and $a \cdot b = b \cdot a$. 3) Additive and multiplicative identity: there exist two different elements 0 and 1 in $\mathcal{F}$ such that $a + 0 = a$ and $a \cdot 1 = a$. 4) Additive inverses: for every a in $\mathcal{F}$, there exists an element in $\mathcal{F}$, denoted $-a$, called additive inverse of a, such that $a + (-a) = 0$. 5) Multiplicative inverses: for every $a \neq 0$ in $\mathcal{F}$, there exists an element in $\mathcal{F}$, denoted by a^{-1} , called the multiplicative inverse of a, such that $a \cdot a^{-1} = 1$. 6) Distributivity of multiplication over addition: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.*

9 General Skeleton Methods and Examples of MLP Models

Algorithm 2 Skeleton Method

Require: $w_1, \cdots, w_L$.

1. **Skeleton incoming weight:** For matrix w_1, get the elements $w_1((j \mod h_0), j), \forall j \in [h_1]$.
2. **Skeleton outgoing weight:** For matrix w_2, get the elements $w_2(j, (j \mod h_2)), \forall j \in [h_1]$.
3. **Basis path set:** Construct set
   $S^1 = \cup_{j=1}^{h_1} \{\{(i_0, j, (j \mod h_2)) | i_0 \in [h_0]\} \cup \{(j \mod h_0), j, i_2) | i_2 \in [h_2]\}\}$
4. Partition set S into $S_j^1 = \{(i_0, i_1, j) | i_0 \in [h_0], i_1 \in [h_1]\}, j \in [h_2]$ according to the third element in the triple.

for $l = 2, ..., L-1$ **do**

5. **Skeleton incoming path:** For matrix w_l, get elements $w_l((j \mod h_{l-1}), j)$. Select $p_{l,j}^s$ which is one of the path passed O_j^l and composed by skeleton weights.
6. **Skeleton outgoing weight:** For matrix w_{l+1}, get the elements $w_{l+1}(j, (j \mod h_{l+1})), \forall j \in [h_l]$.
7. **Basis path set:** Construct set
   $S^l = \cup_{j=1}^{h_l} \{(s, (j \mod h_{l+1}))) | s \in S_j^l\} \cup \{(p_{l,j}^s, i_{l+1}) | i_{l+1} \in [h_{l+1}]\}\}$
8. Partition set S^l into $S_j^l = \{(i_0, \cdots i_l, j) | i_0 \in [h_0], \cdots, i_l \in [h_l], \}, j \in [h_2]$ according to the last element (indexed by j) in the array.

end for

Ensure: Set S^{L-1}.

In this section, we will introduce the skeleton method in Algorithm 2 in a recursive way for general case and use some examples to explain the types of skeleton weights, skeleton paths constructed by Skeleton Method in the main paper. We call the skeleton path which only contains skeleton weights all-skeleton paths and skeleton path which contains one non-skeleton weight skip-skeleton paths.

First, we consider MLP models with the same number of hidden nodes of each layer Hinton et al. (2012). Fig.6(a) shows an example of MLP model. It only displays the skeleton weights. We can see that the number of skeleton paths is three which equals the number of hidden nodes of one layer. If the number of hidden nodes is $h_1 = \cdots = h_{L-1} = h$, by using Skeleton Method in Algorithm 1, the number of skeleton weights is $H = hL$ and the number of all-skeleton paths equals the number of hidden nodes of one layer, which is h. Thus we have

$$\#\textit{(skeleton paths)} \tag{10}$$

$$= \#\textit{(skip-skeleton paths)} + \#\textit{(all-skeleton paths)} \tag{11}$$

$$= \#\textit{(non-skeleton weights)} + \#\textit{(all-skeleton paths)} \tag{12}$$

$$= \#\textit{weights} - \#\textit{(skeleton weights)} + \#\textit{(hidden nodes of one layer)} \tag{13}$$

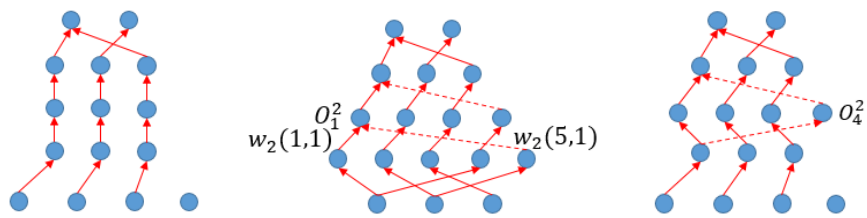
$$= \sum_{l=1}^{L} h_{l-1} h_l - hL + h \tag{14}$$

$$= h_0 h + (L-2)h^2 + h h_L - (L-1)h. \tag{15}$$

Second, we consider MLP models with decreasing number of hidden nodes, i.e., $h_1 \geq h_2 \geq \cdots \geq h_{L-1}$. Fig.6(b) shows an example. We can see that the number of skeleton paths is five which equals the largest number of hidden nodes,i.e., h_1. We can see that because the $h_l > h_{l-1}$, some nodes (e.g., O_1^2) will have multiple incoming weights (e.g., $w_2(1,1), w_2(5,1)$) which are skeleton outgoing

weight for the front hidden nodes. For these nodes, they select one of skeleton weights to be the skeleton incoming weights, which is displayed using red full line (e.g.,$w_2(1,1)$). Others are displayed using red dotted line (e.g., $w_2(5,1)$). Thus the path which contains only skeleton weights and less than one dotted skeleton weight is a all-skeleton path. We have

$$\begin{aligned}
&\#\textit{(skeleton paths)} && (16)\\
=&\#\textit{(skip-skeleton paths)} + \#\textit{(all-skeleton paths)} && (17)\\
=&\#\textit{(non-skeleton weights)} + \#\textit{(all-skeleton paths)} && (18)\\
=&\#\textit{weights} - \#\textit{(skeleton weights)} + \#\textit{(hidden nodes of between the first and second layer)} && (19)\\
=&\sum_{l=1}^{L} h_{l-1}h_l - (\sum_{l=1}^{L} h_l + h_1) + h_1 && (20)\\
=&\sum_{l=1}^{L} h_{l-1}h_l - \sum_{l=1}^{L-1} h_l. && (21)
\end{aligned}$$



(a) (b) (c)

Figure 6: Examples

Third, we consider MLP models with unbalanced number of hidden nodes of each layer. If $h_{l-1} < h_l$ and $h_l > h_{l+1}$ for one layer l with $l \geq 2$, there exists hidden nodes whose the skeleton incoming weight and skeleton outgoing weight are both dotted,e.g., O_4^2 in Fig.6(b). In this case, the number of all-skeleton paths is $h_1 + \sum_{l=1}^{L-2}(\max\{h_l, h_{l+1}\} - h_l)$. It is because that each node must be passed by at least one all-skeleton paths. Thus for the example showed in Fig.6(b), although both the incoming skeleton weight and the outgoing skeleton weight of O_4^2 are dotted, the path $(1,1,4,1,1)$ passed O_4^2 is all-skeleton path. In this case, the number of all-skeleton paths is $h_1 + \sum_{l=1}^{L-2}(\max\{h_l, h_{l+1}\} - h_l)$.

$$\begin{aligned}
&\#\textit{(skeleton paths)} && (22)\\
=&\#\textit{(skip-skeleton paths)} + \#\textit{(all-skeleton paths)} && (23)\\
=&\#\textit{(non-skeleton weights)} + \#\textit{(all-skeleton paths)} && (24)\\
=&\#\textit{weights} - \#\textit{(skeleton weights)} + \#\textit{(all-skeleton paths)} && (25)\\
=&\sum_{l=1}^{L} h_{l-1}h_l - (h_1 + \sum_{l=1}^{L-2} \max\{h_l, h_{l+1}\} + h_{L-1}) + (h_1 + \sum_{l=1}^{L-2}(\max\{h_l, h_{l+1}\} - h_l)) && (26)\\
=&\sum_{l=1}^{L} h_{l-1}h_l - \sum_{l=1}^{L-1} h_l. && (27)
\end{aligned}$$

10 Proofs of Theoretical Results in Section 3

In this section, we will provide proofs of the lemma and theorems in Section 3 of the main paper.

10.1 Proof of Theorem 3.2

Theorem 3.2: *Consider two ReLU neural networks with weights $w, w' \in \mathcal{W}$. We have that $w \sim w'$ iff for $\forall$ path p and $\forall$ input $x \in \mathcal{X}$, we have $v_w(p) = v_{w'}(p)$ and $a_w(p,x) = a_{w'}(p,x)$.*

Proof: The sufficiency is trivial according to the representation of $N_w^k(x)$, which is shown in Eqn(2) in the main paper.

For the necessity, if $w \sim w'$, then there exist a positive rescaling operator $g_{N_{w'},C}$ to make $g_{N_{w'},C}(w') = w$. We use i_l to denote the node index of nodes in layer-l and $i_l \in [h_l]$ (which is denoted by p_l in the main paper). Then we have $w_l(i_{l-1}, i_l) = \frac{1}{c_{i_{l-1}}^{l-1}} \cdot c_{i_l}^l \cdot w_l'(i_{l-1}, i_l)$ for $l = 2, \cdots, L-1$, because each weight may be modified by the operators of its connected two nodes $g_{c_{i_{l-1}}^{l-1}, O_{i_{l-1}}^{l-1}}$ and $g_{c_{i_l}^l, O_{i_l}^l}$. Thus $v_w(p) = v_{w'}(p)$ is satisfied because

$$v_{w'}(p) = \prod_{l=1}^{L} w_l'(i_{l-1}, i_l) = c_{i_1}^1 w_1'(i_0, i_1) \cdot \left(\prod_{l=2}^{L-1} \frac{1}{c_{i_{l-1}}^{l-1}} \cdot c_{i_l}^l w_l'(i_{l-1}, i_l) \cdot \right) \cdot \frac{1}{c_{i_{L-1}}^{L-1}} w_L'(i_{L-1}, i_L) \tag{28}$$

$$= \prod_{l=1}^{L} w_l(i_{l-1}, i_l) = v_w(p). \tag{29}$$

Next we need to prove that $a_w(p, x) = a_{w'}(p, x)$ is also satisfied. Because the value of the activation is determined by the sign of $o^l, l = \{1, \cdots, L-1\}$, we just need to prove that

$$[o_{w,1}^l(x), \cdots, o_{w,h_l}^l(x)] = [c_1^l \cdot o_{w',1}^l(x), \cdots, c_{h_l}^l \cdot o_{w',h_l}^l(x)],$$

where $c_{i_l}^l, i_l \in [h_l], l \in [L-1]$ are positive numbers. We prove it by induction.

(1) For o^1 of a L-layer MLP ($L > 2$): Suppose that $\sigma(\cdot)$ is a ReLU activation function. For the i_1-th hidden node, we have

$$o_{w,i_1}^1 = \sigma\left(\sum_{i_0=1}^{h_0} w_1(i_0, i_1) x_{i_0}\right) = \sigma\left(\sum_{i_0=1}^{h_0} c_{i_1}^1 \cdot w_1'(i_0, i_1) x_{i_0}\right) = c_{i_1}^1 \cdot \sigma\left(\sum_{i_0=1}^{h_0} w_1'(i_0, i_1) x_{i_0}\right) = c_{i_1}^1 \cdot o_{w',i_1}^1. \tag{30}$$

(2) For o^l of the L-layer MLP ($l > 2$): Suppose that

$$[o_{w,1}^j(x), \cdots, o_{w,h_j}^j(x)] = [c_1^j \cdot o_{w',1}^j(x), \cdots, c_{h_j}^j \cdot o_{w',h_j}^j(x)], j = \{1, \cdots, l-1\}.$$

Then we have

$$o_{w,i_l}^l = \sigma\left(\sum_{i_{l-1}=1}^{h_{l-1}} w_l(i_{l-1}, i_l) o_{w,i_{l-1}}^{l-1}(x)\right) = \sigma\left(\sum_{i_{l-1}=1}^{h_{l-1}} \frac{1}{c_{i_{l-1}}^{l-1}} \cdot c_{i_l}^l \cdot w_l'(i_{l-1}, i_l) \cdot c_{i_{l-1}}^{l-1} \cdot o_{w',i_{l-1}}^{l-1}(x)\right) = c_{i_l}^l \cdot o_{w',i_l}^l. \tag{31}$$

Thus we finish the proof. □

10.2 Proof of Lemma 3.3

Lemma 3.3: *The value of any path p can be calculated as $v_w(p) = \prod_{i=1}^{z_1} v(p_s)^{\alpha_i}, \alpha_i \in \{-L+1, \cdots, 0, 1\}$ and $a_w(a, x)$ can be calculated in a recursive way.*

Proof sketch: For the first statement, we prove it by induction.

(a) For a 2-layer MLP, it is similar as the example shown in Figure 2(a) in the main paper.

(b) For a L-layer MLP, suppose that all the paths which are not skeleton path can be represented by the skeleton paths from input layer to layer $L-1$ as following:

$$w_{L-1}(i_{L-2}, i_{L-1}) \cdots w_1(i_0, i_1) = \frac{\prod_{l=1}^{L-1} p_l^s(w_l(i_{l-1}, i_l))}{\prod_j p_j^a},$$

where $p_l^s(w_l(i_{l-1}, i_l))$ (abbreviated as p_l^s) is the skeleton path which contains the weight $w_l(i_{l-1}, i_l)$, and p_j^a denotes a skeleton path which contains only skeleton weights. Then for L-layer MLPs

$(L > 2)$, we have

$$
\begin{aligned}
& w_L(i_{L-1}, i_L) w_{L-1}(i_{L-2}, i_{L-1}) \cdots w_1(i_0, i_1) \\
=& w_L(i_{L-1}, i_L) \cdot \frac{\prod_{l=1}^{L-1} p_l^s(w_l(i_{l-1}, i_l))}{\prod_j p_j^a} \\
=& p_{L-1}^s(w_{L-1}(i_{L-1}, i_{L-2})) \cdot w_L(i_{L-1}, i_L) \cdot \frac{\prod_{l=1}^{L-2}(p_l^s(i_{l-1}, w_l(i_l)) \cdot w_{O_{p_l^s}^L}^{\rightarrow})}{\prod_j (p_j^a \cdot w_{O_{p_j^a}^L}^{\rightarrow})} \\
=& \frac{(p_{O_{i_{L-1}}^{L-1}}^a \cdot w_L(i_{L-1}, i_L)) \cdot (p_{L-1}^s(w_{L-1}(i_{L-1}, i_{L-2})) \cdot w_{L, p_{L-1}^s}^{\rightarrow})}{p_{O_{i_{L-1}}^{L-1}}^a \cdot w_{L, p_{L-1}^s}^{\rightarrow}} \cdot \frac{\prod_{l=1}^{L-2}(p_l^s(w_l(i_{l-1}, i_l)) \cdot w_{O_{p_l^s}^L}^{\rightarrow})}{\prod_j (p_j^a \cdot w_{O_{p_j^a}^L}^{\rightarrow})} \\
=& \frac{\prod_{l=1}^{L} p_l^s(w_{i_{l-1}, i_l})}{\prod_j p_j^a},
\end{aligned}
$$

where $w_{O_{p_j^a}^L}^{\rightarrow}$ is the skeleton weight at $L-$layer which connects p_l^s and $p_{O_{i_{L-1}}^{L-1}}^a$ is the skeleton incoming path of the hidden node $O_{i_{L-1}}^{L-1}$. The third equation is established by the a path which contains less than one non-skeleton weight is a skeleton path.

For the second statement, because the activation status $a_w((i_L, \cdots, i_0), x)$ is determined by all the $o_{i_l}^l(x)$, we just need to prove the sign of $o_{i_l}^l(x)$ is determined by the skeleton paths p_s. For each hidden node $O_{i_l}^l$, there exist only one skeleton path which passes it and only contain skeleton weights (We call the skeleton path which contains only skeleton weights *all-skeleton path*). We denote the all-skeleton path which passes $O_{i_l}^l$ as $p_a(O_{i_l}^l) = w_1(O_{i_l}^l) \cdot \prod_{j=2}^{L} w_j(O_{i_l}^l)$, where $w_j(O_{i_l}^l)$ denotes the skeleton weight of $p_a(O_{i_l}^l)$ at the j-th layer which is also an skeleton outgoing weight for one hidden node. We will prove $o_{i_l}^l(x)$ can be calculated as

$$
o_{i_l}^l(x) = \frac{1}{\prod_{j=l+1}^{L} w_j(O_{i_l}^l)} \cdot F_{i_l}^l(p_s; x), \tag{32}
$$

where $F_{i_l}^l(p_s; x)$ is a function which is determined by the quotient vector p_s and the input x. If Eqn(32) is satisfied, the sign of $o_{i_l}^l(x)$ can be determined as following:

$$
sgn(o_{i_l}^l(x)) = sgn(w_{l+1}(O_{i_l}^l)) \cdots sgn(w_L(O_{i_l}^l)) \cdot sgn(F_{i_l}^l(p_s; x)). \tag{33}
$$

Next we prove Eqn(32) by induction.

(1) For $l = 1$,

$$
o_{i_1}^1(x) = \sum_{i_0=1}^{h_0} w_1(i_0, i_1)) \cdot x_{i_0} = \sum_{i_0=1}^{h_0} \frac{p_a(w_1(i_0, i_1))}{\prod_{j=2}^{L} w_j(O_{i_1}^1)} \cdot x_{i_0} = \frac{1}{\prod_{j=2}^{L} w_j(O_{i_1}^1)} \sum_{i_0=1}^{h_0} p_a(w_1(i_0, i_1))) \cdot x_{i_0}, \tag{34}
$$

where $p_a(w_1(i_0, i_1)))$ is the skeleton path which contains $w_1(i_0, i_1)$. It means that Eqn(32) is satisfied with $F_{i_1}^1(p_s; x) = \sum_{i_0=1}^{h_0} p_a(w_1(i_0, i_1))) \cdot x_{i_0}$.

(2) For $l > 1$, suppose that $o^{l-1}_{i_{l-1}}(x) = \frac{1}{\prod_{j=l}^{L} w_j(O^{l-1}_{i_{l-1}})} \cdot F^{l-1}_{i_{l-1}}(p_s; x)$,

$$o^l_{i_l}(x) = \sum_{i_{l-1}=1}^{h_{l-1}} w_l(i_{l-1}, i_l) \cdot o^{l-1}_{i_{l-1}} \tag{35}$$

$$= \sum_{i_{l-1}=1}^{h_{l-1}} \frac{p_a(w_l(i_{l-1}, i_l))}{\prod_{j=l+1} w_j(O^l_{i_l})} \cdot o^{l-1}_{i_{l-1}} \tag{36}$$

$$= \sum_{i_{l-1}=1}^{h_{l-1}} \frac{p_a(w_l(i_{l-1}, i_l))}{\prod_{j=l+1}^{L} w_j(O^l_{i_l}) \cdot w_1(O^{l-1}_{i_{l-1}}) \cdot \prod_{j=1}^{l-1} w_j(O^{l-1}_{i_{l-1}})} \cdot \frac{1}{\prod_{j=l}^{L} w_j(O^{l-1}_{i_{l-1}})} \cdot F^{l-1}_{i_{l-1}}(p_s; x) \tag{37}$$

$$= \frac{1}{\prod_{j=l+1}^{L} w_j(O^l_{i_l})} \sum_{i_{l-1}=1}^{h_{l-1}} \frac{p_a(w_l(i_{l-1}, i_l))}{w_1(O^{l-1}_{i_{l-1}}) \cdot \prod_{j=1}^{l-1} w_j(O^{l-1}_{i_{l-1}})} \cdot \frac{1}{\prod_{j=l}^{L} w_j(O^{l-1}_{i_{l-1}})} \cdot F^{l-1}_{i_{l-1}}(p_s; x) \tag{38}$$

$$= \frac{1}{\prod_{j=l+1}^{L} w_j(O^l_{i_l})} \sum_{i_{l-1}=1}^{h_{l-1}} \frac{p_a(w_l(i_{l-1}, i_l))}{p_a(O^{l-1}_{i_{l-1}})} \cdot F^{l-1}_{i_{l-1}}(p_s; x) \tag{39}$$

$$= \frac{1}{\prod_{j=l+1}^{L} w_j(O^l_{i_l})} \cdot F^l_{i_l}(p_s; x). \tag{40}$$

Thus we have finished the proof the second statement [5].

10.3 Proof of Theorem 3.4

Theorem 3.4: *Consider two ReLU neural networks with weights $w, w' \in \mathcal{W}$, we have $w \sim w'$ iff $v_w(p^i_s) = v_{w'}(p^i_s), \forall i = 1, \cdots, z_1$, and $sgn(w^j_s) = sgn(w'^j_s), \forall j = 1, \cdots, z_2$, where z_1 is the number of skeleton paths, and z_2 is the number of hidden nodes. Therefore, the quotient space $\mathcal{W}/\sim$ of ReLU network is isomorphic to the following space:*

$$Q = \{\rho(w) \triangleq (v_w(p_s), sgn(w_s)), w \in \mathcal{W}\} \tag{41}$$

$$= \{(v_w(p^1_s), \cdots, v_w(p^{z_1}_s), sgn(w_1), \cdots, sgn(w_{z_2}))\} \subset R^{z_1} \times \{-1, 1\}^{z_2}. \tag{42}$$

For MLP models with ReLU, $z_1 = \#$ weights $-\#$ (hidden nodes) and $z_2 = \#$(hidden nodes); for CNN models with ReLU, $z_1 = \#$ weights $-\#$ (feature maps) and $z_2 = \#$(feature maps).

Proof: The first part of the theorem is established directly by Lemma 3.3 and Lemma 3.3'. For the second part, $z_2 = \#$(hidden nodes) because each hidden node has only one skeleton outgoing weight according to Skeleton Method. Next, we count the number of skeleton paths. The number of skeleton paths which is contained only skeleton weights are h_1 because $h_1 = \max\{h_1, \cdots, h_{L-1}\}$. The number of skeleton weights (both skeleton incoming weights and skeleton outgoing weights) is $\#$(hidden nodes)$+h_1$ because we should count the skeleton outgoing weights of all the hidden nodes and the skeleton incoming weights at layer-1. Then we have $z_1 = \#$ weights $-\#$ (hidden nodes)$-h_1 + h_1 = \#$ weights $-\#$ (hidden nodes). □

11 Theoretical Properties of the Loss in Equivalent Class Space

In this section, we show some the theoretical properties of the loss which is represented in the quotient space.

11.1 Gradient Calculation and Properties

Theorem 11.1 *Then the gradient of the loss function with respect to ρ can be calculated as below, $\forall i = 1, \cdots, z_1$*

$$\frac{\partial N^k_\rho(x)}{\partial \rho_i}\Big|_\rho = \sum_{p: p_L = k} \frac{\partial v(p; \rho)}{\partial \rho_i}\Big|_\rho \cdot a(p; \rho, x) \cdot x_{p_0}. \tag{43}$$

[5] Please note that for the simplicity, we omit the notation $v(p)$ and just use p to denote the path value.

Proof sketch: By chain rule, we have

$$\frac{\partial N_\rho^k(x)}{\partial \rho_i}\Big|_\rho = \sum_{p:p_L=k}\left(\frac{\partial v(p;\rho)}{\partial \rho_i}\Big|_\rho \cdot a(p;\rho,x)\cdot x_{p_0} + \frac{\partial a(p;\rho,x)}{\partial \rho_i}\Big|_\rho \cdot v(p;\rho)\cdot x_{p_0}\right).$$

Thus we need to calculate $\frac{\partial v(p;\rho)}{\partial \rho_i}$ and $\frac{\partial a(p;\rho,x)}{\partial \rho_i}$. Since $v(p;\rho)$ is a continuous function about ρ_i according to Lemma 3.3, it is easy to get its gradient. For $\frac{\partial a(p;\rho,x)}{\partial \rho_i}$, we have

$$\frac{\partial a(p;\rho,x)}{\partial \rho_i} = \mathbb{I}(o_{p_{L-1}}^{L-1}(x)>0)\cdots\mathbb{I}(o_{p_1}^{1}(x)>0), \tag{44}$$

The indicator function $\mathbb{I}(z)$ is a hard function about z, whose gradient is 0 at $z\in(-\epsilon,\epsilon)$ with $\epsilon>0$. The range of its subgradient at $z=0$ is $[0,\infty]$. We define $0\cdot\infty=0$ Lay (1997), then we can calculate $\frac{\partial a(p;\rho,x)}{\partial \rho_i}$ as following:

$$\frac{\partial a(p;\rho,x)}{\partial \rho_i} = \sum_{j=1}^{L-1}\mathbb{I}(o_{p_{L-1}}^{L-1}>0)\cdots\frac{\partial \mathbb{I}(o_{p_j}^{j}>0)}{\partial \tilde{p}_j}\cdots\mathbb{I}(o_{p_1}^{1}>0). \tag{45}$$

(1) If $a(p;\rho,x)=1$, then $\frac{\partial a(p;\rho,x)}{\partial \rho_i}=0$.

(2) If $a(p;\rho,x)=0$, there is at least an $\mathbb{I}(o_{p_j}^{j}(x)>0)=0$. If all $o_{p_j}^{j}(x)\neq 0$, then $\frac{\partial a(p;\rho,x)}{\partial \rho_i}=0$.

(3) If there is an $o_{p_j}^{j}(x)=0$, then

$$\begin{aligned}
\frac{\partial N_\rho^k(x)}{\partial \rho_i}\Big|_\rho &= \sum_{p:p_L=k}\left(\frac{\partial a(p;\rho,x)}{\partial \rho_i}\Big|_\rho \cdot p(p;\rho)\cdot x_{p_0}\right)\\
&= \sum_{p:p_L=k} v(p,\rho)\cdot x_{p_0}\cdot\sum_{j=1}^{L-1}\mathbb{I}(o_{p_{L-1}}^{L-1}(x)>0)\cdots\frac{\partial \mathbb{I}(o_{p_j}^{j}(x)>0)}{\partial \rho_i}\cdots\mathbb{I}(o_{p_1}^{1}(x)>0)\\
&= \frac{\partial \mathbb{I}(o_{p_j}^{j}(x)>0)}{\partial o_{p_j}^{j}}\cdot\frac{\partial o_{p_j}^{j}}{\partial \rho_i}\cdot\sum_{p:p_L=k} v(p(p_j),O_{p_j}^{j})\cdot x_{p_0}\\
&= \frac{\partial \mathbb{I}(o_{p_j}^{j}>0)}{\partial o_{p_j}^{j}}\cdot\frac{\partial o_{p_j}^{j}}{\partial p_j}\cdot o_{p_j}^{j}\cdot\sum_{p:p_L=k}\prod_{l=j+1}^{L} w_l(p_{l-1},p_l)\cdot x_{p_0}\\
&= \infty\cdot 0 = 0,
\end{aligned}$$

where $v(p(p_j),O_{p_j}^{j})$ denotes the value of the path which passes the hidden node $O_{p_j}^{j}$. Thus we have finished the proof. □

We can get the following corollary directly according to Theorem 4.1.

Corollary 11.2 *The gradients of ρ for equivalent ReLU neural networks with different weights take exactly the same value.*

11.2 Loss Surface with Fewer Critical Points

Theorem 11.3 *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 $C(\mathcal{W})$.*

Proof: If $\rho(w_0)\in C(\mathcal{W}/\sim_{\mathcal{G}})$, then we have $\nabla_{\rho_i(w_0)}L(\rho(w_0))=0, i=1,\cdots,z_1$. By chain rule, we have

$$\frac{\partial L(w)}{\partial w_l(i_{l-1},i_l)}\Big|_{w_0} = \left(\sum_{j=1}^{z_1}\frac{\partial L(\rho)}{\partial \rho_i}\cdot\frac{\partial \rho_i}{\partial w_l(i_{l-1},i_l)}\right)\Big|_{w_0} = 0, \tag{46}$$

because $\frac{\partial L(\rho)}{\partial \rho_i}\Big|_{\rho(w_0)}=0, \forall i\in[z_1]$. Thus we can see that if the set composed by critical points in quotient space is a m-dimensional manifold, the dimension of the set composed by the corresponding critical points in weight space is larger than m. It is because the vector $\rho(w_0)$ corresponds to an equivalence class with Hausdorff dimension #(hidden nodes) in the weight space.

12 SUPPLEMENTARY INFORMATION OF THE EXPERIMENTS

12.1 UPDATE RULE OF QUOTIENT SGD

First, according to the ICR Method, we can get the update rule of the quotient as below,

$$\rho_i^{t+1} = \rho_i^t - \eta_t \frac{\delta_{w_{n,i}}^t \cdot w_{n,i}^t}{\rho_i^t}$$
$$\rho_j^{t+1} = \rho_j^t - \eta_t \frac{w_{s,j} \cdot (\delta_{w_{s,j}}^t - \sum_{\rho_i : w_{s,j}} \delta_{\rho_i}^t \cdot \frac{\rho_i^t}{w_{s,j}^t})}{\rho_j}$$

Combined with the weight allocation method, we can get the update rule of quotient SGD as follows:

$$w_{s,j}^{t+1} = w_{s,j}^t - \eta_t \cdot \frac{\delta_{w_{s,j}}^t \cdot (w_{s,j}^t)^2 - w_{s,j}^t \cdot \sum_{\rho_i : w_{s,j}} \delta_{w_{n,i}^t} \cdot w_{n,i}^t}{(\rho_j^t)^2}$$
$$w_{n,i}^{t+1} = \frac{w_{j_1}^t - \eta_t \cdot \frac{\delta_{w_{j_1}}^t \cdot (w_{n,i}^t)^2}{(\rho_i^t)^2}}{R^t(\rho_j : w_{n,i})},$$
$$w^{t+1} = w^t, w \neq w_{n,i}, w_{s,j}$$

where $R^t(\rho_j : w_{n,i})$ is ratio of the skeleton path p_j which contains non-skeleton weight $w_{n,i}$.

12.2 SKELETON METHOD FOR RESNET AND ICR METHOD FOR BATCH NORMALIZATION

For ResNet, we implement the Skeleton Method to construct skeleton weights and skeleton paths in each residual block. Because of the skip-connection, there is an identity weight which doesn't change during the optimization. Thus, if the skip-connected weight connects node O, there isn't a positively scaling operator $g_{O,c} \in \mathcal{G}$ to make $w \sim g_{O,c}(w)$. So we can't construct skeleton paths for the whole network. The redundancy of ResNet only exists in each residual block. In this sense, the redundancy of ResNet is less severe than other neural network structure.

Because of the existence of the batch-normalization layers, the output of neural network with BN is $\hat{o_{ij}}^l = \frac{o_{ij}^l - \mu}{\sqrt{\sigma^2 + \epsilon}}$, where o_{ij}^l means the i-th output in layer-l which is calculated using the i-th sample, $\mu = \frac{1}{b}\sum_{i_0=1}^b o_{ij}^l$ is the expectation of $o_{ij}, i = 1, \cdots, b$ and $\sigma^2 = \frac{1}{b}\sum_{i=1}^b (o_{ij}^l - \mu)^2$ is the variance. Assume that $o_{ij}^l = w_l o_i^{l-1}$ and the inputs o_i^{l-1} has expectation 0 and variance 1 (It can be roughly satisfied for neural networks with BN.), we have $\sigma^2 \approx \|w_l\|$. Thus the loss function of the NN with BN layer can be approximately represented as $l(\rho, \|w_{O_{in}}\|; x, y) = \frac{f(\rho; x, y)}{\prod_O \|w_{O_{in}}\|}$ where $w_{O_{in}}$ denotes the incoming weights of node O and $f(\rho; x, y)$ denotes a function which is only related to ρ.

Thus inverse-chain-rule for NN with BN layer can be approximated by the following equations. If w is an incoming weight of node O, we have

$$\frac{\partial l(w; x, y)}{\partial w} \approx \sum_{i=1}^{z_1} \frac{\partial l(\rho; x, y)}{\partial \rho_i} \cdot \frac{\partial \rho_i}{\partial w} \cdot \frac{1}{\|w_{O_{in}}\|}, \tag{47}$$

which results in

$$\frac{\partial l(w; x, y)}{\partial w} \cdot \|w_{O_{in}}\| \approx \sum_{i=1}^{z_1} \frac{\partial l(\rho; x, y)}{\partial \rho_i} \cdot \frac{\partial \rho_i}{\partial w}. \tag{48}$$

Then we use $\frac{\partial l(w; x, y)}{\partial w} \cdot \|w_{O_{in}}\|$ to δ_w in the ICR methods.

12.3 INITIALIZATION METHOD OF SKELETON WEIGHTS

According to our analysis in the main paper, only the signs of skeleton weights matter the optimization. Thus we need to determine the signs of skeleton weights before training process. For the absolute value of skeleton weights, we can see from section 10.1 that different absolute value of skeleton

weights well determine different scale of learning rate. Although our theoretical results show that the absolute value of skeleton weights can be randomly set, we choose them to be 1 for easier learning rate tuning and robustness.

In order to verify how signs of skeleton weights influence the performance. We test the performance for various combination of signs for them on image classification task (see section 5.2). Results shows that there are no differences for them. A intuitive explanation is that the selected network model is over-parameterized and the approximation ability will not be influenced by signs of skeleton weights. For simplicity, we initialize the value of skeleton weights as 1.

12.4 Detailed Training Strategies in Section 5.1

In this section, our aim is to verify the influence of equivalence to optimization in weight space. We train several 2-hidden-layer MLP models with different equivalence ratio (i.e. $H/\#params$) on Fasion dataset. The original size of input image is 28×28. We normliazed the input data, and to reduce the dimensions of input feature, we downsample the image to 7×7 by using average pooling. The network structue is followed by [49:h:h:10] where h is the number of hidden units in each layer. The detailed model properties are shown in table 5. All models are initialized by (He et al., 2015) except the skeleton weights which is mentioned in Section 4 without explicit note. We use the learning rate of 0.01 and mini-batch size of 64 for vanilla SGD and Quotient SGD, and train each model for 100 epochs.

Table 5: Network information in Section 5.1.

#hidden units	8	16	32	64	128	256	512	1024
#params	536	1200	2912	7872	23936	80640	292352	1108992
H	16	32	64	128	256	512	1024	2048
equivalence ratio	1.49×10^{-2}	1.33×10^{-2}	1.10×10^{-2}	8.13×10^{-3}	5.35×10^{-3}	3.17×10^{-3}	1.75×10^{-3}	9.23×10^{-4}

12.5 Detailed Training Strategies in Section 5.2

In this section, we extend our Quotient SGD to deep convolutional networks. CIFAR-10 and CIFAR-100 have been used in the experiment. We apply random crop to the input image by size of 32 with padding 4, and normalize each pixel value to [0,1]. We then apply random horizontal flipping to the image. The mini-batch size of 128 is used in this experiment. The training is conducted for 64k iterations. We follow the learning rate schedule strategy in the original paper (He et al., 2016a), specifically, the initial learning rates of vanilla SGD and Quotient SGD are set to 1.0 and then divided by 10 after 32k and 48k iterations. The ResNet implementation can be found in https://github.com/pytorch/vision/ and the models are initialized by the default methods in PyTorch.

12.6 The Combination of Quotient SGD and Regularization

The optimization algorithms achieve better generalization performance on test dataset by combining with proper regularization methods. In the previous experiments, we focus on the difference performance of optimization algorithms. In this section, we conduct the experiments to investigate the combination of Quotient SGD and regularization. In weight space, weight norm is widely used as regularization for ReLU networks (He et al., 2016a; Huang et al., 2017). Recently, (Zheng et al., 2018) propose the basis path norm in quotient space. In this section, we reproduce the experiments in (He et al., 2016a; Zheng et al., 2018) on SGD regularized by weight norm (SGD+WD) and Quotient SGD regularized by basis path norm (Q-SGD+BPR), and extend them on CIFAR-100 dataset. The learning rate of 1.0 is widely used to train ResNet model and its variants on CIFAR dataset, hence we employ it in our experiment as well. We do a wide range grid search for the hyper-parameter λ for weight decay and basis path regularization from $\{0.1, 0.2, 0.5\} \times 10^{-\alpha}$, where $\alpha \in \{3, 4, 5, 6\}$, and report the best performance based on the CIFAR-10 validation set. We use the same hyper-parameter on CIFAR-100 dataset.

As shown in Figure 7 and 8, the Quotient SGD achieves clearly better performance than SGD with combining regularization method on all models and all datasets. To be specific, the test accuracy of

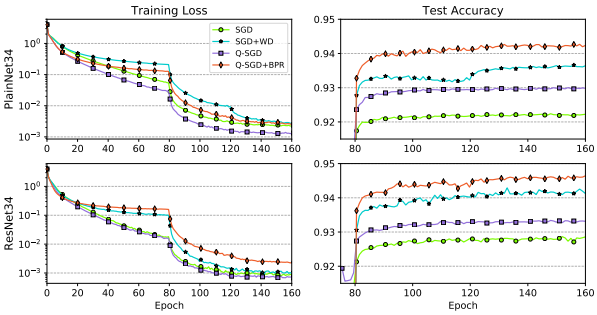


Figure 7: Training loss and test accuracy w.r.t. the number of effective passes on CIFAR-10 dataset.

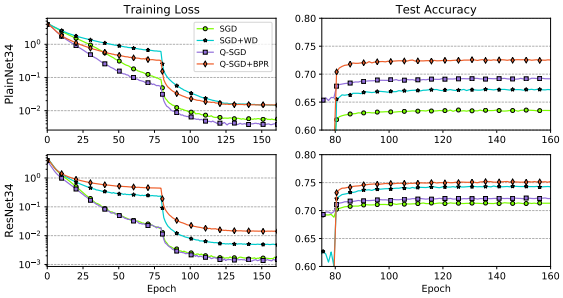


Figure 8: Training loss and test accuracy w.r.t. the number of effective passes on CIFAR-100 dataset.

94.29% is gained by ResNet-34 trained by SGD and weight decay on CIFAR-10 dataset (the number reported in (He et al., 2016a) is 91.25%), while Quotient SGD with basis path regularization improves the test accuracy to 94.67%. On CIFAR-100 dataset, the test accuracy of ResNet-34 trained by SGD and weight decay is 74.39% (the ResNet-34 result on CIFAR-100 hasn't been reported in (He et al., 2016a), a result of ResNet-110 with similar training strategy on this dataset is reported in (Zagoruyko & Komodakis, 2016) which is 74.84%), while Quotient SGD with basis path regularization improves the test accuracy to 75.20%. The experimental results verify our analysis again that optimization in quotient space is a better choice.

12.7 Detailed Training Strategies in Section 5.3

Path-SGD (Neyshabur et al., 2015) also notice the positively scaling equivalence in neural networks with linear or ReLU activation. Instead of optimizing the loss function in quotient space, they use path norm as regularizer to the gradient in weight space. Meanwhile, the dependency among all paths hasn't been noted, which leads to the computation overhead of the gradient of path norm is very high. In section 5.3, we extend the experiments in (Neyshabur et al., 2015) to Quotient SGD on MNIST and CIFAR-10 dataset. A 5-hidden-layer MLP model is used in this experiment with 64 units in each layer. We do grid search for the learning rate of each algorithm from $1.0 \times 10^{-\alpha}$, where α is an integer between 0 to 10. We report the best result for each algorithm. The mini-batch size of 64 is used, and the input images of gray scale are normalized to $[0, 1]$.