Can We Gain More from Orthogonality Regularizations in Training Deep CNNs?

2019 NeurIPS

Reviewed by Taehyeon

Summary

Intro

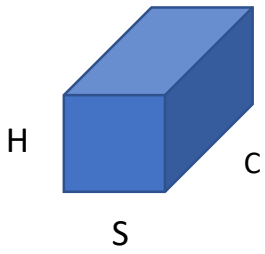
To do

Improve the performance by adding orthogonal regularization to the parameter by layer.

Settings

Before the orthogonal regularization, the conv layer has a 4-dimensional parameter, so we need a rule to change it first to Matrix. Rule is as follows.

Convolutional layer(filter)



Matrix

S: filter width
H: filter height
C: input channel number
M: output channel number

m’ = S x H x C
n’ = M

Summary

Baseline: Soft Orthogonality Regularization

Method

Regularization using Frobenius Norm

Drawback

The dimension of the column space and the row space is different, and the rank of the gram matrix(W^TW or WW^T) can not become the full rank, so the gram matrix can not become the identity matrix .

Weight decay

Weight decay and Orthogonal regularization must be done at the same time.

Previous works [14, 32, 33] proposed to require the Gram matrix of the weight matrix to be close to identity, which we term as Soft Orthogonality (SO) regularization:

$$\text{(SO)} \qquad \lambda||W^TW - I||_F^2, \tag{1}$$

Regularization formula

where λ is the regularization coefficient (the same hereinafter). It is a straightforward relaxation from the "hard orthogonality" assumption [12, 13, 15, 38] under the standard Frobenius norm, and can be viewed as a different weight decay term limiting the set of parameters close to a Stiefel manifold rather than inside a hypersphere. The gradient is given in an explicit form: $4\lambda W(W^TW - I)$, and can be directly appended to the original gradient w.r.t. the current weight W.

Regularization gradient formula

Summary

Method1: Double Soft Orthogonality Regularization

Method

A method to solve the rank problem of Soft Orthogonality. Gram matrix Regularization form by adding both.

Weight decay

Weight decay and Orthogonal regularization must be done at the same time.

The double soft orthogonality regularization extends SO in the following form:

$$\text{(DSO)} \quad \lambda(||W^T W - I||_F^2 + ||WW^T - I||_F^2). \tag{2}$$

Regularization formula

Note that an orthogonal W will satisfy $W^T W = WW^T = I$; an overcomplete W can be regularized to have small $||WW^T - I||_F^2$ but will likely have large residual $||W^T W - I||_F^2$, and vice versa for an under-complete W. DSO is thus designed to cover both over-complete and under-complete W cases; for either case, at least one term in (2) can be well suppressed, requiring either rows or columns of W to stay orthogonal. It is a straightforward extension from SO.

Another similar alternative to DSO is "selective" soft orthogonality regularization, defined as: $\lambda||W^T W - I||_F^2$, if $m > n$; $\lambda||WW^T - I||_F^2$ if $m \leq n$. Our experiments find that DSO always outperforms the selective regularization, therefore we only report DSO results.

Summary

Method2: Mutual Coherence Regularization

Method
The form of regularization based on the highest correlation between weights.

Weight decay
Weight decay and Orthogonal regularization must be done at the same time.

The mutual coherence [18] of W is defined as:

$$\mu_W = \max_{i \neq j} \frac{|\langle w_i, w_j \rangle|}{||w_i|| \cdot ||w_j||}, \quad (3)$$

→ Regularization formula

where w_i denotes the i-th column of W, $i = 1, 2, ..., n$. The mutual coherence (3) takes values between [0,1], and measures the highest correlation between any two columns of W. In order for W to have orthogonal or near-orthogonal columns, μ_W should be as low as possible (zero if $m \geq n$).

We wish to suppress μ_W as an alternative way to enforce orthogonality. Assume W has been first normalized to have unit-norm columns, $\langle w_i, w_j \rangle$ is essentially the (i, j)-the element of the Gram matrix $W^T W$, and $i \neq j$ requires us to consider off-diagonal elements only. Therefore, we propose the following mutual coherence (MC) regularization term inspired by (3):

$$\text{(MC)} \qquad \lambda ||W^T W - I||_\infty. \quad (4)$$

→

Regularization formula는 MC를 생각하고 준 regularization이 맞다. 하지만, (4)처럼 regularization을 하게 되면, gradient를 계산할 때, 연산량이 많아져 비효율적이게 된다. 따라서, 이렇게 하게 되었다.

Summary

Method3: Spectral Restricted Isometry Property Regularization

Method

A method that makes the spectral norm to go to 1

Weight decay

Weight decay and Orthogonal regularization must be done at the same time.

Assumption 1 *For all vectors $z \in \mathbb{R}^n$ that is k-sparse, there exists a small $\delta_W \in (0,1)$ s.t.* $(1-\delta_W) \leq \frac{||Wz||^2}{||z||^2} \leq (1+\delta_W)$.

We rewrite the special RIP condition with $k = n$ in the form below:

$$\left| \frac{||Wz||^2}{||z||^2} - 1 \right| \leq \delta_W, \forall z \in \mathbb{R}^n \tag{5}$$

Notice that $\sigma(W) = \sup_{z \in \mathbb{R}^n, z \neq 0} \frac{||Wz||}{||z||}$ is the spectral norm of W, i.e., the largest singular value of W. As a result, $\sigma(W^TW - I) = \sup_{z \in \mathbb{R}^n, z \neq 0} |\frac{||Wz||^2}{||z||^2} - 1|$. In order to enforce orthogonality to W from an RIP perspective, one may wish to minimize the RIP constant δ_W in the special case $k = n$, which according to the definition should be chosen as $\sup_{z \in \mathbb{R}^n, z \neq 0} |\frac{||Wz||^2}{||z||^2} - 1|$ as from (5). Therefore, we end up equivalently minimizing the spectral norm of $W^TW - I$:

$$\text{(SRIP)} \qquad \lambda \cdot \sigma(W^TW - I). \tag{6}$$

It is termed as the Spectral Restricted Isometry Property (SRIP) regularization.

We again refer to auto differentiation to compute the gradient of (6) for simplicity. However, even computing the objective value of (6) can invoke the computationally expensive EVD. To avoid that, we approximate the computation of spectral norm using the power iteration method. Starting with a randomly initialized $v \in \mathbb{R}^n$, we iteratively perform the following procedure a small number of times (2 times by default) :

$$u \leftarrow (W^TW - I)v, v \leftarrow (W^TW - I)u, \sigma(W^TW - I) \leftarrow \frac{||v||}{||u||}. \tag{7}$$

Singular value를 미분하는 것은 어렵다.따라서, 위의 방법(iterative)을 이용해 singular value를 미분한다.

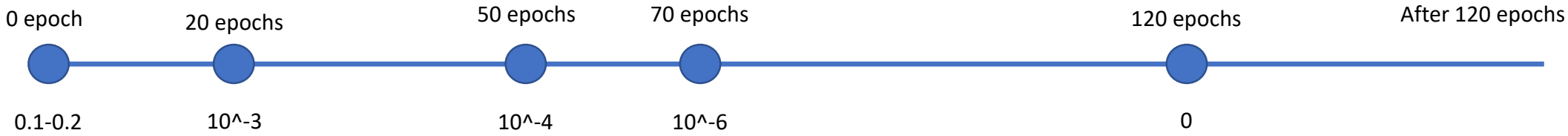
Experiment

Setting

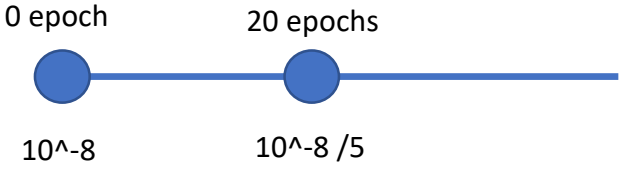
Scheme Change

How to schedule the weight decay and orthogonal regularizer parameters?

Orthogonal Regularizer



Weight decay(only SO/DSO)



This is because others are insensitive to the choice of weight decay parameter, potentially due to their stronger effects in enforcing $W^T W$ close to I; we thus stick to the initial parameter through out training for them. (MC/SRIP)

Why weight decay?

From experiments, they observe that fully replacing l2 weight decay with orthogonal regularizers will accelerate and stabilize training at the beginning of training, but will negatively affect the final accuracies achievable.

Experiment

Setting

Scheme Change

How to schedule the weight decay and orthogonal regularizer parameters?

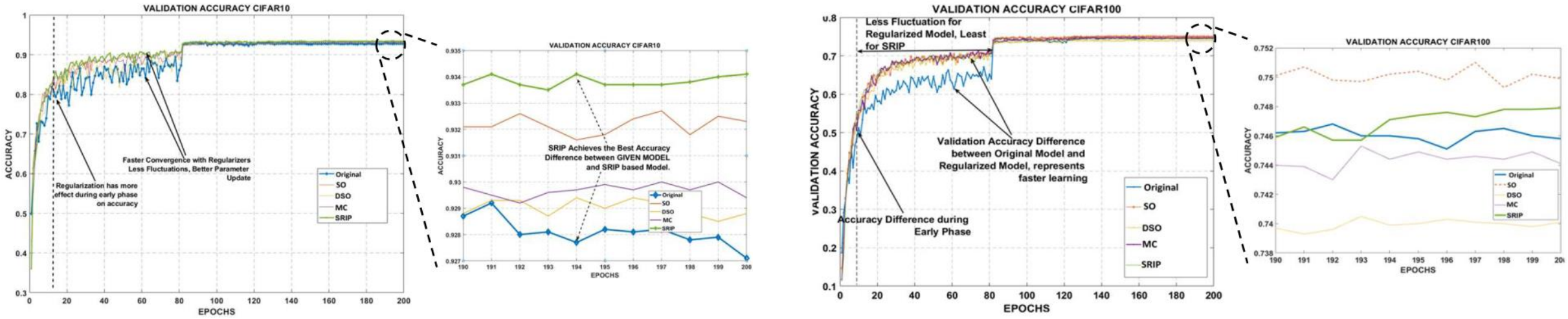


Figure 1: Validation curves during training for ResNet-110. Top: CIFAR-10; Bottom: CIFAR-100;

Experiment

Architecture

Architecture

ResNext110, Wide ResNet 28-10, ResNext 29-8-64

Data

CIFAR-10, CIFAR-100, ImageNet, SVHN

ResNet 110 Model [6] The 110-layer ResNet Model [6] is a very strong and popular ResNet version. It uses Bottleneck Residual Units, with a formula setting given by $p = 9n + 2$, where n denotes the total number of convolutional blocks used and p the total depth. We use the Adam optimizer to train the model for 200 epochs, with learning rate starting with 1e-2, and then subsequently decreasing to 10^{-3}, 10^{-5} and 10^{-6}, after 80, 120 and 160 epochs, respectively.

Wide ResNet 28-10 Model [21] For the Wide ResNet model [21], we use depth 28 and k (width) 10 here, as this configuration gives the best accuracies for both CIFAR-10 and CIFAR-100, and is (relatively) computationally efficient. The model uses a Basic Block B(3,3), as defined in ResNet [6]. We use the SGD optimizer with a Nesterov Momentum of 0.9 to train the model for 200 epochs. The learning rate starts at 0.1, and is then decreased by a factor of 5, after 60, 120 and 160 epochs, respectively. We have followed all other settings of [21] identically.

ResNext 29-8-64 Model [20] For ResNext Model [20], we consider the 29-layer architecture with a cardinality of 8 and widening factor as 4, which reported the best state-of-the-art CIFAR-10/CIFAR-100 results compared to other contemporary models with similar amounts of trainable parameters. We use the SGD optimizer with a Nesterov Momentum of 0.9 to train the model for 300 epochs. The learning starts from 0.1, and decays by a factor of 10 after 150 and 225 epochs, respectively.

Table 1: Top-1 error rate comparison by ResNet 110, Wide ResNet 28-10 and ResNext 29-8-64 on CIFAR-10 and CIFAR-100. * indicates results by us running the provided original model.

Model	Regularizer	CIFAR-10	CIFAR-100
ResNet-110 [6]	None	7.04*	25.42*
	SO	6.78	**25.01**
	DSO	7.04	25.83
	MC	6.97	25.43
	SRIP	**6.55**	25.14
Wide ResNet 28-10 [21]	None	4.16*	20.50*
	SO	3.76	18.56
	DSO	3.86	18.21
	MC	3.68	18.90
	SRIP	**3.60**	**18.19**
ResNext 29-8-64 [20]	None	3.70*	18.53*
	SO	3.58	17.59
	DSO	3.85	19.78
	MC	3.65	17.62
	SRIP	**3.48**	**16.99**

Experiment

Architecture

Architecture

ResNext110, Wide ResNet 28-10, ResNext 29-8-64

Data

CIFAR-10, CIFAR-100, ImageNet, SVHN

ResNet 110 Model [6] The 110-layer ResNet Model [6] is a very strong and popular ResNet version. It uses Bottleneck Residual Units, with a formula setting given by $p = 9n + 2$, where n denotes the total number of convolutional blocks used and p the total depth. We use the Adam optimizer to train the model for 200 epochs, with learning rate starting with 1e-2, and then subsequently decreasing to 10^{-3}, 10^{-5} and 10^{-6}, after 80, 120 and 160 epochs, respectively.

Wide ResNet 28-10 Model [21] For the Wide ResNet model [21], we use depth 28 and k (width) 10 here, as this configuration gives the best accuracies for both CIFAR-10 and CIFAR-100, and is (relatively) computationally efficient. The model uses a Basic Block B(3,3), as defined in ResNet [6]. We use the SGD optimizer with a Nesterov Momentum of 0.9 to train the model for 200 epochs. The learning rate starts at 0.1, and is then decreased by a factor of 5, after 60, 120 and 160 epochs, respectively. We have followed all other settings of [21] identically.

ResNext 29-8-64 Model [20] For ResNext Model [20], we consider the 29-layer architecture with a cardinality of 8 and widening factor as 4, which reported the best state-of-the-art CIFAR-10/CIFAR-100 results compared to other contemporary models with similar amounts of trainable parameters. We use the SGD optimizer with a Nesterov Momentum of 0.9 to train the model for 300 epochs. The learning starts from 0.1, and decays by a factor of 10 after 150 and 225 epochs, respectively.

Experiments on ImageNet We train ResNet 34, Pre-ResNet 34 and ResNet 50 [40] on the ImageNet dataset with and without SRIP regularizer, respectively. The training hyperparameters settings are consistent with the original models. The initial learning rate is set to 0.1, and decreases at epoch 30, 60, 90 and 120 by a factor of 10. The top-5 error rates are then reported on the ILSVRC-2012 val set, with single model and single-crop. [15] also reported their top-5 error rates with both ResNet 34 and Pre-ResNet 34 on ImageNet. As seen in Table 2, SRIP clearly outperforms the best for all three models.

Table 2: Top-5 error rate comparison on ImageNet.

Model	Regularizer	ImageNet
ResNet 34 [6]	None	9.84
	OMDSM [15]	9.68
	SRIP	**8.32**
Pre-Resnet 34 [40]	None	9.79
	OMDSM [15]	9.45
	SRIP	**8.79**
ResNet 50 [6]	None	7.02
	SRIP	**6.87**

Experiments on SVHN On the SVHN dataset, we train the original Wide ResNet 16-8 model, following its original implementation in [21] with initial learning 0.01 which decays at epoch 60,120 and 160 all by a factor of 5. We then train the SRIP-regularized version with no change made other than adding the regularizer. While the original Wide ResNet 16-8 gives rise to an error rate of **1.63%**, SRIP reduces it to **1.56%**.

Summary: It makes the performance of model better.

Contribution

Strong point & Weak point

Strong point

Simple, acceleration of training in early stage, outstanding performance improvements

Weak point

Incremental, no distinct theoretical analysis, No comparison with other regularizations

Summary: It makes the performance of model better.