Neighborhood Region Smoothing Regularization for Finding Flat Minima In Deep Neural Networks

Yang Zhao[1][0000−0001−5883−2799] and Hao Zhang[1]

Department of Electronic Engineering, Tsinghua University, Beijing, 100081, China
`zhao-yan18@mails.tsinghua.edu.cn`
`haozhang@tsinghua.edu.cn`

Abstract. Due to diverse architectures in deep neural networks (DNNs) with severe overparameterization, regularization techniques are critical for finding optimal solutions in the huge hypothesis space. In this paper, we propose an effective regularization technique, called Neighborhood Region Smoothing (NRS). NRS leverages the finding that models would benefit from converging to flat minima, and tries to regularize the neighborhood region in weight space to yield approximate outputs. Specifically, gap between outputs of models in the neighborhood region is gauged by a defined metric based on Kullback-Leibler divergence. This metric could provide insights in accordance with the minimum description length principle on interpreting flat minima. By minimizing both this divergence and empirical loss, NRS could explicitly drive the optimizer towards converging to flat minima, and meanwhile could be compatible with other common regularizations. We confirm the effectiveness of NRS by performing image classification tasks across a wide range of model architectures on commonly-used datasets such as CIFAR and ImageNet, where generalization ability could be universally improved. Also, we empirically show that the minima found by NRS would have relatively smaller Hessian eigenvalues compared to the conventional method, which is considered as the evidence of flat minima.[1]

Keywords: Deep Learning · Optimization · Flat Minima.

1 Introduction

Driven by the rapid development of computation hardwares, the scale of today's deep neural networks (DNNs) is increasing explosively, where the amount of parameters has significantly exceeded the sample size by even thousands of times [11,10,7]. These heavily overparameterized DNNs would induce huge hypothesis weight spaces. On the one hand, this provides the power to fit extremely complex or even arbitrary functions [13], on the other hand, it becomes much more challenging to seek optimal minima while resisting overfitting during training

[1] Code is available at `https://github.com/zhaoyang-0204/nrs`

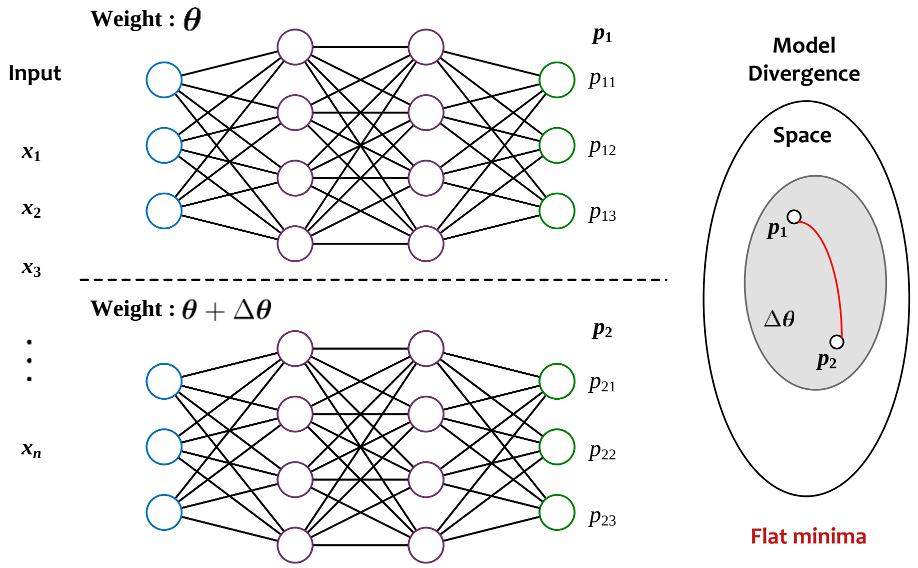

Fig. 1. Description of model divergence and flat minima. Given the same input, model divergence gauges the distance between models in the output space, and flat minima implies that models between these in the neighborhood region $\delta\boldsymbol{\theta}$ and the reference model $\boldsymbol{\theta}$ would be relatively lower compared to sharp minima. NRS would minimize the model divergence for models in this neighborhood to explicitly find flat minima during training.

[26]. Generally, minimizing only the empirical training loss that characterizes the difference between the true labels and the predicted labels (such as categorical cross-entropy loss) could not provide sufficient guarantee on acquiring minima with satisfactory model generalization [20]. Regularization techniques are in higher demand than ever for guiding the optimizers towards finding minima with better model generalization.

Researchers find that minima of well-generalized models are more likely to be located at the landscape where the geometry of its neighborhood region is flat [12,16]. In other words, models would benefit from biasing the convergence towards flat minima during training. A common simple idea is to minimize the empirical loss of models with random perturbations. Unfortunately, [9,30] empirically suggest that optimization only the empirical loss of these random perturbed models would not help to flatten the loss landscape.

In this paper, we show that by adding proper regularizations, such simple random perturbation could also lead to flat minima, where we propose a simple and effective regularization technique, called neighborhood region smoothing (NRS). NRS could regularize the models with random perturbation in the neighborhood region to yield the same outputs instead of their loss values. To accomplish such regularization, we firstly define the model divergence, a metric that gauges the divergence between models in the same weight space. We demonstrate that model divergence could provide interpretation of flat minima from the perspective of information theory, which shares similar core with the minimum description length principle. Since model divergence is differentiable, NRS regularization would be implemented in a straightforwards manner in practice, which would minimize the model divergence between models in the neighborhood region during optimizing for smoothing the surface around the target minima.

In our experiments, we show that NRS regularization could improve the generalization ability for various models on different datasets. We show that NRS regularization is compatible with other regularization techniques, which could improve the model performance moreover. Also, we confirm that for models with random perturbation in the neighborhood region, optimizing only the empirical loss could not lead to flat minima, just as [9,30] suggest. We empirically show that NRS regularization could reduce the largest eigenvalue of Hessian matrices, indicating that NRS regularization could indeed lead to flat minima.

2 Related Works

Training to make models generalize better is a fundamental topic in deep learning, where regularization is one of the most critical techniques that are commonly used during training.

Regularization is actually a rather broad concept involving techniques that may be beneficial to the training process in various ways. Generally, common regularization techniques would affect the model training from three aspects. Firstly, some regularization expect to reduce the searching hypothesis space, where the most conventional method is weight decay [17,19]. Secondly, some others try to complicate the target task so that the models could learn more "sufficiently". Typical methods include enlarging the input space such as cutout [5] and auto-augmentation [4], and interfering models during training like dropout [23], spatialdropout [24] regularization and stochastic depth regularization [14]. Thirdly, others implement normalizations, which includes batch normalization [15], layer normalization [1] and so on.

On the other hand, for a better understanding of generalization, researchers are also interested in studying the underlying factors that associate with the generalization ability of models. Several characteristics are demonstrated having impact on generalization including the flatness of minima, margins of classifiers [22,2], sparsity [18] and degree of feature extraction [27]. In particular, regarding the flatness of minima, despite still lack of theoretical justification and strict definition [20], empirical evidences have been found for such phenomenon. For example, [16] demonstrate that the reason large-batch training would lead to worse generalization than small-batch training is because large-batch training tends to stall at sharp minima. Moreover, by solving a minimax problem, [9,30,8] show that training could benefit from optimizing towards flat minima. Also, [28,29] explicitly regularize the gradient norm to bias training for the convergence of flat minima. All of these works have shown that model performance can be improved by adopting proper techniques that encourages flat minima.

3 Neighborhood Region Smoothing (NRS) Regularization

3.1 Basic Background

Consider inputs $\boldsymbol{x} \in \mathcal{X}$ and labels $\boldsymbol{y} \in \mathcal{Y}$ which conform the distribution $\mathscr{D}$ and a neural network model f parameterized by parameters $\boldsymbol{\theta}$ in weight space $\boldsymbol{\Theta}$ which maps the given input space $\mathcal{X}$ to the corresponding output space $\hat{\mathcal{Y}}$,

$$f(\cdot;\boldsymbol{\theta}) : \mathcal{X} \rightarrow \hat{\mathcal{Y}} \tag{1}$$

For classification tasks, $\hat{\boldsymbol{y}} \in \hat{\mathcal{Y}}$ could be the vector of the predicted probability distribution of each class.

Theoretically, given a loss function $L(\cdot)$, we expect to minimize the expected loss $L_e(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathscr{D}}[l(\hat{\boldsymbol{y}}, \boldsymbol{y}, \boldsymbol{\theta})]$. However, it is untractable in practice. We instead seek to acquire the model via minimizing the empirical loss $L_{\mathcal{S}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} l(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i, \boldsymbol{\theta})$, where the training set $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=0}^{N}$ is assumed to be drawn independently and identically from distribution $\mathscr{D}$. Gap between the expected loss and the empirical loss would directly lead to generalization errors of models.

For over-parametrized models, when minimizing the empirical loss on training set, the huge hypothesis weight space would be filled with numerous minima. These minima may have approximate empirical training loss but meanwhile provide diverse generalization ability. How to discriminate which minima are favorable to the model generalization is critical for getting better training performance.

In particular, minima of models with better generalization are supposed to locate at flatter surfaces, and these minima are often called flat minima. Although there may be different definitions for describing the flat minima [16,6], yet according to [12], the core interpretation behind conveys the similar idea that "a flat minimum is a large connected region in weight space where the error remains approximately constant". In other words, we expect that models parametrized by parameters in the neighborhood area of $\boldsymbol{\theta}$ could yield approximate outputs at flat minima.

3.2 Model divergence

First, we would like to clarify a basic concept, the equivalence of two models,

Definition 1. *Two models $f(\cdot;\boldsymbol{\theta})$ and $f(\cdot;\boldsymbol{\theta}^{'})$ are called **observationally equivalent** if $f(\boldsymbol{x};\boldsymbol{\theta}) = f(\boldsymbol{x};\boldsymbol{\theta}^{'})$ for $\forall\ \boldsymbol{x} \in \mathcal{X}$.*

If two models are observationally equivalent, then they will definitely have the same outputs given any input in the input space, and meanwhile will lead to the same loss. But if the losses of two models are the same, it could not sufficiently ensure that they are observationally equivalent. In other words, loss is not a sufficient condition for the models to be observationally equivalent.

From previous demonstration, to find flat minima, we would expect that the reference model and the neighborhood models are as observationally equivalent as possible. So, it is natural to optimize to reduce the gap between these models, where we need to gauge such gap first. In general, this gap of outputs is usually assessed via the difference in empirical loss between these models when given the same inputs. But, this is apprarently inappropriate, because it is highly possible for models that are not observationally equivalent yield the same loss value, especially for categorical cross-entropy loss. Thus, minimize this loss gap could not fully guarantee that models will convergence to flat minima.

To this end, we would measure this gap between outputs by employing the Kullback-Leibler divergence,

Definition 2. *For model $f(\cdot;\boldsymbol{\theta})$ and model $f(\cdot;\boldsymbol{\theta}^{'})$ in the same weight space $\boldsymbol{\Theta}$, given $\boldsymbol{x} \in \mathcal{X}$, the gap of the two models in the output space could be gauged via the Kullback-Leibler divergence,*

$$d_p(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}) = \mathbb{E}_{\boldsymbol{x}}[D_{\mathrm{KL}}(f(\boldsymbol{x};\boldsymbol{\theta})||f(\boldsymbol{x};\boldsymbol{\theta}^{'}))] \tag{2}$$

where $D_{\mathrm{KL}}(\cdot||\cdot)$ denotes the KL divergence. We call $d_p(\boldsymbol{\theta}, \boldsymbol{\theta}^{'})$ the model divergence between $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{'}$.

Note that $d_p(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}) \geq 0$. Clearly, for $d_p(\boldsymbol{\theta}, \boldsymbol{\theta}^{'})$, the lower the value is, the more approximate the outputs that the two models could yield. Meanwhile, it is obvious that,

Lemma 1. *The two models $f(\cdot;\boldsymbol{\theta})$ and $f(\cdot;\boldsymbol{\theta}^{'})$ are observationally equivalent if and only if $d_p(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}) = 0$.*

Basically, this indicates that minimizing model divergence could in a sense serve as a sufficient condition for converging to flat minima in optimization.

Further, we would like to discuss the interpretation of flat minima from the perspective of model divergence. Based on information theory, KL divergence of the two output distributions $f(\boldsymbol{x};\boldsymbol{\theta})$ and $f(\boldsymbol{x};\boldsymbol{\theta}^{'})$ provides insights in regards to how many additional bits are required to approximate the true distribution $f(\boldsymbol{x};\boldsymbol{\theta})$ when using $f(\boldsymbol{x};\boldsymbol{\theta}^{'})$. Lower divergence indicates fewer information loss if using $f(\boldsymbol{x};\boldsymbol{\theta}^{'})$ to approximate $f(\boldsymbol{x};\boldsymbol{\theta})$.

So for flat minima, since the model is expected to have approximate outputs with models in its neighborhood region, they should have low model divergence. This indicates that when using models $f(\cdot;\boldsymbol{\theta}^{'})$ in the neighborhood to appropriate the true model $f(\cdot;\boldsymbol{\theta})$, only few extra information is required. In contrast, high model divergence would mean more extra information is required. Therefore, describing a flat minimum would require much fewer information than a sharp minimum. Remarkly, based on the minimum description length (MDL) principle, better models would benefit from simpler description. Accordingly, compared to a sharp minimum, a flat minimum will imply better model performance, since the required information description is shorter. This is actually in accordance with Occam's razor principle in deep learning.

3.3 NRS regularization

Generally, a flat minimum suggests that for $\forall\ \delta\boldsymbol{\theta} \in B(0, \epsilon)$ where $B(0, \epsilon)$ is the Euclidean ball centered at 0 with radius ϵ, the model $f(\cdot, \boldsymbol{\theta})$ and its neighborhood

Algorithm 1 Neighborhood Region Smoothing (NRS) Regularization

```
Input: Training set $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=0}^N$; loss function $l(\cdot)$; batch size $B$; learning rate $\eta$;
total steps $K$; neighborhood region size $\epsilon$; model divergence penalty coefficient $\alpha$.
Parameter: Model parameters $\boldsymbol{\theta}$.
Output: Model with final optimized weight $\hat{\boldsymbol{\theta}}$.
1: Parameter initialization $\boldsymbol{\theta}_0$; get the number of devices $M$.
2: for step $k = 1$ to $K$ do
3:     Get sample batch $\mathcal{B} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=0}^B$.
4:     Shard $\mathcal{B}$ based on the number of devices $\mathcal{B} = \mathcal{B}_0 \cup \mathcal{B}_1 \cdots \cup \mathcal{B}_M$, where $|\mathcal{B}_0| = \cdots = |\mathcal{B}_M|$.
5:     Do in parallel across devices.
6:     Make a unique pseudo-random number generator $\kappa$.
7:     Generate random perturbation $\delta\boldsymbol{\theta}$ within area $B(0, \epsilon)$ based on $\kappa$.
8:     Create neighborhood model $f(\cdot, \boldsymbol{\theta} + \delta\boldsymbol{\theta})$.
9:     Compute gradient $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ of the final loss based on batch $\mathcal{B}_i$.
10:    Synchronize and collect the gradient $\boldsymbol{g}$.
11:    Update parameter $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \cdot \boldsymbol{g}$
12: end for
```

model $f(\cdot, \boldsymbol{\theta} + \delta\boldsymbol{\theta})$ are expected to have both low model divergence and low empirical training loss.

Therefore, we would manually add additional regularization in the loss during training for finding flat solutions,

$$\min_{\boldsymbol{\theta}} L_{\mathcal{S}}(\boldsymbol{\theta}) + \alpha \cdot d_p(\boldsymbol{\theta}, \boldsymbol{\theta} + \delta\boldsymbol{\theta}) + L_{\mathcal{S}}(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) \tag{3}$$

where α denotes the penalty coefficient of model divergence $d_p(\boldsymbol{\theta}, \boldsymbol{\theta} + \delta\boldsymbol{\theta})$. In Equation 3, the final training loss $L(\boldsymbol{\theta})$ contains three items:

- The first item is the conventional empirical training loss, denoting the gap between the true labels and the predicted labels. Optimize this item would drive the predicted labels towards the true labels.
- The second item is the model divergence regularization, denoting the divergence between outputs yielded separately from the the models with parameter $\boldsymbol{\theta}$ and the model $\boldsymbol{\theta} + \delta\boldsymbol{\theta}$ in the neighborhood. Optimize this item would explicitly drive the model to yield approximate outputs in the neighborhood.
- The third item is the empirical training loss of the neighborhood model, denoting the gap between true labels and the labels predicted by this neighborhood model. Optimize this item would explicitly force the neighborhood also learn to predict the true labels.

Compared to minimize only the empirical loss of perturbed models $L(\boldsymbol{\theta}+\delta\boldsymbol{\theta})$, Equation 3 imposes much more rigorous constraint for models in the neighborhood region. This also leads to allow us to be relaxing on model selection in the neighborhood, which enables to perform a simple random perturbation.

In order to provide sufficient $\delta\boldsymbol{\theta}$ samples in neighborhood $B(0, \epsilon)$, it is best to train each input sample with a distinct $\delta\boldsymbol{\theta}$. However, this would not fit in the general mini-batch parallel training paradigm well in practice because this would require extra computation graphs for each distinct $\delta\boldsymbol{\theta}$ when deploying using common deep learning framework like Tensorflow, Jax and Pytorch. For balancing the computation accuracy and efficiency, we would generate a unique $\delta\boldsymbol{\theta}$ for each training device instead of for each input sample. In this way, each device need to generate only one neighborhood model, indicating that they will use the same computation graph for samples loaded on this device. All the mini-batch samples would fully enjoy the parallel computing in each device, which would significantly decrease the computation budget compared to training each input sample with a distinct $\delta\boldsymbol{\theta}$.

Algorithm 1 concludes the pseudo-code of the full implementation of NRS regularization. In Algorithm 1, the default optimizer is stochastic gradient descent. We will show the effectiveness of NRS algorithm in the following experimental section.

4 Experimental Results

In this section, we are going to demonstrate the effectiveness of NRS regularization by studying the image classification performances on commonly-used datasets with extensive model architectures. In our experiments, the datasets include Cifar10, Cifar100 and ImageNet, and the model architectures include VGG [21], ResNet [11], WideResNet [25], PyramidNet [10] and Vision Transformer [7]. All the models are trained from scratch to convergence, which are implemented using Jax framework on the NVIDIA DGX Station A100 with four NVIDIA A100 GPUs.

4.1 Cifar10 and Cifar100

We would start our investigation of NRS from evaluating its effect on the generalization ability of models on Cifar10 and Cifar100 dataset. Five network architectures would be trained from scratch, including both CNN models and ViT models, which are VGG16, ResNet18, WideResNet-28-10, PyramidNet-164 and Vision Transformer family[2]. For datasets, we would adopt several different augmentation strategies. One is the *Basic* strategy, which follows the conventional four-pixel extra padding, random cropping and horizontal random flipping. Meanwhile, other than the basic strategy, we would also perform more complex data augmentations to show that NRS would not conflict with other regularization techniques. Specifically, we would adopt the *Cutout* strategy when training CNN models, which would additionally perform cutout regularization [5]. But when training ViT models, we would instead adopt a *Heavy* strategy, which will additionally perform mixup regularization and train much longer compared to the *Basic* strategy.

[2] The names of all the mentioned model architectures is the same as them in their original papers.

CNN Models	Cifar10		Cifar100	
VGG16	Basic	Cutout	Basic	Cutout
Baseline	$93.12_{\pm 0.08}$	$93.95_{\pm 0.11}$	$72.28_{\pm 0.17}$	$73.34_{\pm 0.22}$
RPR	$93.14_{\pm 0.21}$	$93.91_{\pm 0.17}$	$72.31_{\pm 0.21}$	$73.29_{\pm 0.19}$
NRS	$\mathbf{93.79_{\pm 0.11}}$	$\mathbf{94.77_{\pm 0.17}}$	$\mathbf{73.61_{\pm 0.20}}$	$\mathbf{75.32_{\pm 0.19}}$
ResNet18	Basic	Cutout	Basic	Cutout
Baseline	$94.88_{\pm 0.12}$	$95.45_{\pm 0.16}$	$76.19_{\pm 0.21}$	$77.03_{\pm 0.11}$
RPR	$94.90_{\pm 0.24}$	$95.41_{\pm 0.23}$	$76.25_{\pm 0.31}$	$76.98_{\pm 0.18}$
NRS	$\mathbf{95.84_{\pm 0.15}}$	$\mathbf{96.47_{\pm 0.10}}$	$\mathbf{78.67_{\pm 0.17}}$	$\mathbf{79.88_{\pm 0.14}}$
WideResNet-28-10	Basic	Cutout	Basic	Cutout
Baseline	$96.17_{\pm 0.12}$	$97.09_{\pm 0.17}$	$80.91_{\pm 0.13}$	$82.25_{\pm 0.15}$
RPR	$96.14_{\pm 0.11}$	$97.02_{\pm 0.15}$	$80.94_{\pm 0.19}$	$82.19_{\pm 0.24}$
NRS	$\mathbf{96.94_{\pm 0.17}}$	$\mathbf{97.55_{\pm 0.13}}$	$\mathbf{82.77_{\pm 0.16}}$	$\mathbf{83.94_{\pm 0.16}}$
PyramidNet-164	Basic	Cutout	Basic	Cutout
Baseline	$96.32_{\pm 0.15}$	$97.11_{\pm 0.12}$	$82.33_{\pm 0.19}$	$83.50_{\pm 0.17}$
RPR	$96.31_{\pm 0.19}$	$97.09_{\pm 0.19}$	$82.25_{\pm 0.24}$	$83.58_{\pm 0.22}$
NRS	$\mathbf{97.23_{\pm 0.19}}$	$\mathbf{97.72_{\pm 0.11}}$	$\mathbf{84.61_{\pm 0.24}}$	$\mathbf{86.29_{\pm 0.18}}$
ViT Models	Cifar10		Cifar100	
ViT-TI16	Basic	Heavy	Basic	Heavy
Baseline	$83.07_{\pm 0.08}$	$89.22_{\pm 0.18}$	$60.12_{\pm 0.15}$	$65.44_{\pm 0.20}$
RPR	$83.02_{\pm 0.13}$	$89.28_{\pm 0.14}$	$60.18_{\pm 0.21}$	$65.21_{\pm 0.17}$
NRS	$\mathbf{84.17_{\pm 0.09}}$	$\mathbf{90.47_{\pm 0.14}}$	$\mathbf{61.29_{\pm 0.13}}$	$\mathbf{66.84_{\pm 0.22}}$
ViT-S16	Basic	Heavy	Basic	Heavy
Baseline	$85.14_{\pm 0.11}$	$92.49_{\pm 0.09}$	$61.83_{\pm 0.20}$	$72.40_{\pm 0.17}$
RPR	$85.17_{\pm 0.13}$	$92.12_{\pm 0.11}$	$61.92_{\pm 0.14}$	$72.33_{\pm 0.17}$
NRS	$\mathbf{85.86_{\pm 0.10}}$	$\mathbf{93.55_{\pm 0.16}}$	$\mathbf{62.59_{\pm 0.17}}$	$\mathbf{73.17_{\pm 0.14}}$
ViT-B16	Basic	Heavy	Basic	Heavy
Baseline	$88.42_{\pm 0.12}$	$91.54_{\pm 0.11}$	$63.49_{\pm 0.17}$	$72.32_{\pm 0.21}$
RPR	$88.39_{\pm 0.14}$	$91.57_{\pm 0.12}$	$63.38_{\pm 0.15}$	$72.35_{\pm 0.17}$
NRS	$\mathbf{89.09_{\pm 0.08}}$	$\mathbf{92.23_{\pm 0.14}}$	$\mathbf{64.46_{\pm 0.17}}$	$\mathbf{73.11_{\pm 0.13}}$

Table 1. Testing accuracy of various models on Cifar10 and Cifar100 when using the three training strategies.

We would focus our investigations on the comparisons between three training schemes. The first one would train with the standard categorical cross-entropy loss, which is $\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$. This is our baseline. The second training scheme is to optimize the same cross-entropy loss of the models with random perturbation (RPR) in the neighborhood region instead of the true model, which is $\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}+\delta\boldsymbol{\theta})$. This one is for confirming that minimizing the loss of simple random perturbations could not be helpful to the generalization ability of models, just as papers [9,30] suggest. The last training scheme would be training with our NRS regularization. It should be noted that we would keep any other deployment the same for the three schemes during training except for the techniques mentioned in specific scheme.

For the common training hyperparameters, we would perform a grid search for acquiring the best performance for CNN models. Specifically, the base learning rate is searched over $\{0.01, 0.05, 0.1, 0.2\}$, the weight decay coefficient is searched over $\{0.0005, 0.001\}$ and the batch size is searched over $\{128, 256\}$. Also, we would adopt cosine learning rate schedule and SGD optimizer with 0.9 momentum during training. As for ViT models, we use fixed hyperparameters and use different hyperparameter deployments for the two data augmentation strategies. For *Basic* strategy, learning rate is 0.001, weight decay is 0.3, training epoch is 300, batch size is 256 and the patch size is 4×4 while learning rate is $2e-4$, weight decay is 0.03 and training epoch is 1200 for *Heavy* strategy. Meanwhile, we would adopt the Adam optimizer during training. Additionally, for all the CNN and ViT models, we would use three different random seeds and report the average mean and variance across the testing accuracies of the three seeds.

For RPR scheme, it involves one extra hyperparameter, which represents the radius of the neighborhood region ϵ. So similarly, we would perform a grid search over $\{0.05, 0.1, 0.5\}$ as well. As for NRS strategy, it involves two extra hyperparameters, the radius of the neighborhood region ϵ and the penalty coefficient of model divergence α. We would adopt the same search for ϵ, and α is searched over $\{0.5, 1.0, 2.0\}$. Notably, grid search of ϵ and α in both RPR and NRS would be performed on the basis of hyperparameters of the best model acquired by the first training strategy.

Table 1 shows the corresponding testing accuracies of Cifar10 and Cifar100, where all the reported results are the best results during the grid search of hyperparameters. We could see that in the table, all the testing accuracies have been improved by NRS regularization to some extent compared to the baseline, which confirms its benefit for model training. We also try NRS regularization on the recent Vision Transformer model. We could find that when performing *Basic* augmentation, the testing accuracy of ViT models would be much lower than that of CNN models. This is because training ViT models generally requires plenty of input samples. In this case, NRS could also improve the generalization ability of the such models. And when performing *Heavy* augmentation, training performance would be improved significantly, where again, performance could be improved further when training with the NRS scheme.

From the results, we could find that NRS regularization would not conflict with optimizers and current regularizations like dropout (in VGG16) and batch normalization, which is important for practical implementation. Additionally, we also confirm that simply optimizing the models with neighborhood random perturbation like RPR could indeed have no effect on improving the generalization ability of models.

	ImageNet	
VGG16	Top-1 Accuray	Top-5 Accuray
Baseline	$73.11_{\pm 0.08}$	$91.12_{\pm 0.08}$
NRS	$\mathbf{73.52}_{\pm \mathbf{0.09}}$	$\mathbf{91.39}_{\pm \mathbf{0.09}}$
ResNet50	Top-1 Accuray	Top-5 Accuray
Baseline	$75.45_{\pm 0.15}$	$93.04_{\pm 0.07}$
NRS	$\mathbf{76.29}_{\pm \mathbf{0.13}}$	$\mathbf{93.53}_{\pm \mathbf{0.10}}$
ResNet101	Top-1 Accuray	Top-5 Accuray
Baseline	$77.15_{\pm 0.11}$	$93.91_{\pm 0.07}$
NRS	$\mathbf{78.02}_{\pm \mathbf{0.10}}$	$\mathbf{94.44}_{\pm \mathbf{0.08}}$

Table 2. Testing accuracy of various models on ImageNet dataset when using standard training strategy (Baseline) and NRS regularization strategy.

4.2 ImageNet

Next, we would check the effectiveness of NRS regularization on a large-scale dataset, ImageNet. Here, we would take VGG16, ResNet50 and ResNet101 model architectures as our experimental targets. For datasets, all the images would be resized and cropped to 224×224, and then randomly flipped in the horizontal direction. For common hyperparameters, instead of performing a grid search, we would fix the batch size to 512, the base learning rate to 0.2, the weight decay coefficient to 0.0001. During training, we would smooth the label with 0.1. All models would be trained for a total of 100 epochs with three different seeds.

Our investigation would focus on the comparisons between two training strategies. One is the standard training with categorical cross-entropy loss, which is our baseline. The other one is trained with NRS regularization. For hyperparameters in NRS regularization, we would fix the ϵ to 0.1 and α to 1.0 according to previous tuning experience. Table 2 shows the corresponding results.

As we could see in Table 2, the testing accuracy could be improved again to some extent when using NRS regularization. This further confirms that NRS regularization could be beneficial to the generalization ability of models.

5 Further Studies of NRS

5.1 Parameter Selection in NRS

In this section, we would investigate the influence of the two parameters ϵ and α in NRS regularization on the results. The investigation is conducted on Cifar10 and Cifar100 using WideResNet-28-10. We train the models from scratch using NSR with the same common hyperparameters of the best models acquired by the baseline strategy. And then, we would perform the grid search for the two parameters using the same scheme as in previous section.

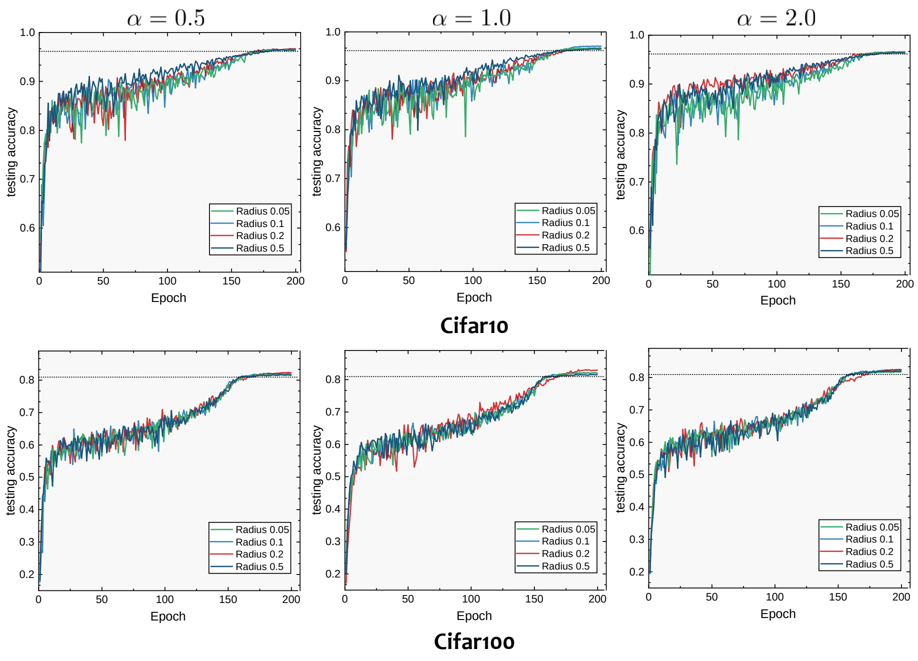

Fig. 2. Evolutions of testing accuracy on Cifar10 and Cifar100 during training when trained with different parameters in NRS. The black dash lines refer to the reference lines which are the best testing accuracy trained using standard strategy.

Figure 2 shows the evolution of testing accuracy during training when using the corresponding different parameters. We could see that actually for all the deployed ϵ and α, the generalization ability could be somewhat improved on both Cifar10 and Cifar100 datasets compared to the baseline (the black reference line in the figure). This again demonstrates the effectiveness of NRS regularization. From the figure, we could find that the model could achieve the best performance when $\epsilon = 0.1$ and $\alpha = 1.0$ for Cifar10 dataset and $\epsilon = 0.2$ and $\alpha = 1.0$ for Cifar100 dataset. Also, we could find that the generalization improvement on Cifar100 would generally be larger than that on Cifar10.

5.2 Eigenvalues of Hessian Matrix

In this section, we would investigate the intrinsic change of models when using the NSR regularization. In general, the eigenvalues of Hessian matrix are considered to have connections with the flatness of minima. Specifically, the largest eigenvalue of Hessian matrix of flat minima could be larger than that of sharp minima.

Since the dimension of the weight space is so huge, solving the Hessian matrix in a direct manner is nearly impossible. Therefore, we would employ the approximate method introduced in [3]. It calculates the diagonal block of Hessian matrix recursively from the deep layers to the shallower layers.

We would still use WideResNet-28-10 model and Cifar10 as our investigation target. We would use the best models acquired by the three training strategies in the previous section. Also, we would calculate the eigenvalues of Hessian matrix for the last layer in each model. Table 3 reports the results.

WideResNet-28-10	λ_{max}
Baseline	49.15
RPR	51.79
NRS	**12.42**

Table 3. The largest eigenvalue of Hessian matrix of models trained with three different strategies.

As can be seen from the table, using NRS regularization can significantly reduce the largest eigenvalue of the Hessian matrix compared to using the other two strategies. Also, using RPR strategy indeed could not lead to flat minima. This again verify that using NRS regularization would find flat minima during training.

6 Conclusion

In this paper, we propose a simple yet effective regularization technique, called Neighborhood Region Smoothing, for finding flat minima during training. The key idea of NRS is regularizing the neighborhood region of models to yield approximate outputs. Using outputs in NRS could give stronger regularization than using loss values, so simple random perturbation in the neighborhood region would be effective. We define model divergence to gauge the gap between outputs of models in the neighborhood region. In this way, NRS regularization is achieved by explicitly minimizing both the empirical loss and the model divergence. In our experiments, we show that using NRS regularization could improve the generalization ability of a wide range of models on diverse datasets compared to two other training strategies. We also investigate to give the best hyperparameters in NRS on Cifar10 and Cifar100 dataset. Finally, smaller eigenvalue of Hessian matrix confirms that NRS regularization could indeed to flat minima.

Acknowledgements We would like to thank all the reviewers and the meta-reviewer for their helpful comments and kindly advices. We would like to thank Jean Kaddour from University College London for the kindly discussions.

References

1. Ba, L.J., Kiros, J.R., Hinton, G.E.: Layer normalization. arXivPreprint **abs/1607.06450** (2016)
2. Bartlett, P.L., Foster, D.J., Telgarsky, M.: Spectrally-normalized margin bounds for neural networks. In: Advances in Neural Information Processing Systems. pp. 6240–6249 (2017)
3. Botev, A., Ritter, H., Barber, D.: Practical gauss-newton optimisation for deep learning. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017. vol. 70, pp. 557–565 (2017)
4. Cubuk, E.D., Zoph, B., Mané, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. arXivPreprint **abs/1805.09501** (2018)
5. Devries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXivPreprint **abs/1708.04552** (2017)
6. Dinh, L., Pascanu, R., Bengio, S., Bengio, Y.: Sharp minima can generalize for deep nets. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017. vol. 70, pp. 1019–1028 (2017)
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: 9th International Conference on Learning Representations, ICLR 2021 (2021)
8. Du, J., Yan, H., Feng, J., Zhou, J.T., Zhen, L., Goh, R.S.M., Tan, V.Y.F.: Efficient sharpness-aware minimization for improved training of neural networks. arXivPreprint **abs/2110.03141** (2021)
9. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: 9th International Conference on Learning Representations, ICLR 2021 (2021)
10. Han, D., Kim, J., Kim, J.: Deep pyramidal residual networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. pp. 6307–6315 (2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016. pp. 770–778 (2016)
12. Hochreiter, S., Schmidhuber, J.: Flat minima. Neural Comput. **9**(1), 1–42 (1997)
13. Hornik, K., Stinchcombe, M.B., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**(5), 359–366 (1989)
14. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Computer Vision - ECCV 2016 - 14th European Conference. pp. 646–661 (2016)
15. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015. vol. 37, pp. 448–456 (2015)
16. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: 5th International Conference on Learning Representations, ICLR 2017 (2017)
17. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: Advances in Neural Information Processing Systems,. pp. 950–957 (1991)
18. Liu, S.: Learning sparse neural networks for better generalization. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. pp. 5190–5191. ijcai.org (2020)

19. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: 7th International Conference on Learning Representations, ICLR 2019 (2019)
20. Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N.: Exploring generalization in deep learning. In: Advances in Neural Information Processing Systems. pp. 5947–5956 (2017)
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015 (2015)
22. Sokolic, J., Giryes, R., Sapiro, G., Rodrigues, M.R.D.: Robust large margin deep neural networks. IEEE Trans. Signal Process. **65**(16), 4265–4280 (2017)
23. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
24. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C.: Efficient object localization using convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015. pp. 648–656 (2015)
25. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: Proceedings of the British Machine Vision Conference 2016, BMVC 2016 (2016)
26. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: 5th International Conference on Learning Representations, ICLR 2017 (2017)
27. Zhao, Y., Zhang, H.: Quantitative performance assessment of CNN units via topological entropy calculation. In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022 (2022)
28. Zhao, Y., Zhang, H., Hu, X.: Penalizing gradient norm for efficiently improving generalization in deep learning. In: International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (2022)
29. Zhao, Y., Zhang, H., Hu, X.: Randomized sharpness-aware training for boosting computational efficiency in deep learning. arXivPreprint **abs/2203.09962** (2022)
30. Zheng, Y., Zhang, R., Mao, Y.: Regularizing neural networks via adversarial model perturbation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021. pp. 8156–8165 (2021)