Linear Regularization: An easy approach to refine your model

Anonymous CVPR 2021 submission

Paper ID 7770

Abstract

In general computer vision classification tasks, neural networks suffer from overfitting on training data. We propose Linear Regularization (LinR) to prevent the network from overfitting. The method puts a linear constrain to endow the network with more linearity in the refining schedule. Extensive experiments on CIFAR-10/100, ImangeNet, PETA show superior performance improvements. We also achieve state-of-the-art mA score on PETA and PA-100k datasets. Specially, our method can refine the converged model in an unsupervised manner. Meanwhile, we give specific guidance about when should we do refine. Our method is also explained in the view of the loss function's landscape.

1. Introduction

Different approaches have been proposed for learning better representation and obtain better generalization without increasing the dataset scale. Data augmentation increases the network's generalization power by forcing the network prediction to be invariant with respect to augmented view of input image [6, 7, 34] or synthetic image [1]. Early stopping [9] prevent the neural network from overfitting on training dataset by stopping the training when the loss on hold-out validation dataset increase. L1 and L2 [28] regularization increase the neural network's generalization ability by constraining parameters with a zero value prior. Dropout [35] prevent the co-adaptation of neurons by randomly mask out neurons during training. Better architecture design like a skip or dense connection [15, 16] can simplify the loss landscape of neural network and increase the generalization power. Visualizing landscape of loss function [23] shows the shot cuts in network can promote convexity which is a good property for better generalization. Could we give the network more convexity by adding a regularization other than changing network's structures?

If we have a converged model with performance doesn't fit our expectations but we don't have extra data to train it further. We may fine tune it like [31] to train low-level layers sufficiently by rolling back the weights of high-level layers to their initial pre-trained weights. Or we can just fine tune the network with old data which is an easily overfitting method. But can we have a unsupervised method helpful both during and after training even without labels?

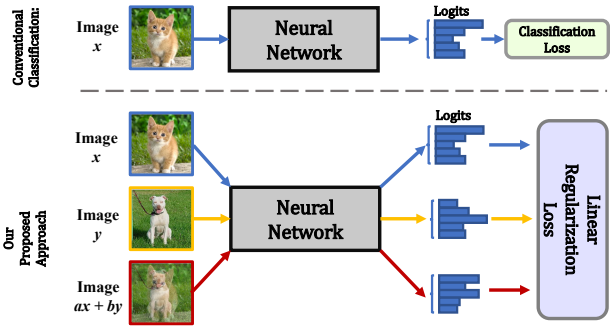


Figure 1. Overview of the comparison between conventional classification task and proposed Linear Regularization

Convolution Neural Networks (CNN) have been widely applied for solving challenging computer vision tasks like image classification [20, 15, 33], object detection [30, 4], and semantic segmentation [27, 25]. Huge quantities of human annotations are one of the core factors for CNN's success, without which the performance will decrease due to overfilling. In this paper, we aim to design simple and novel regularization methods that can increase CNN's generalization without increasing the dataset scale.

In this work, we propose an easy and efficient regularization approach to refine our model. In the view of loss landscape [23], ResNet[15] and DenseNet [16] can produce loss functions that are easy to train. When networks become deep enough, neural loss landscapes will change from nearly convex to badly chaotic. And the chaos is harmful to the network's generalization. But skip connections can prevent the transition to chaotic behavior. However, can we do it without short cut and put on some regularization on model parameters to restrain it and keep its convexity other than changing network structures? In specific, we propose a reg-

ularization to give the network more convexity by putting linear constrain on the network without changing the structure of the network like skip connections. Fig. 1 shows an overall structure of linear regularization (LinR). The proposed output process can be considered as a linearity also as additivity and homogeneity of degree 1 constrain on the macro model parameters other than L1/L2 regularization, which put constrain in each microcosmic parameter. The mix procedure is borrowed from mixup [38] which can be helpful to the regularization procedure.

Furthermore, we propose LinR with no label (LinRNL) that can improve the converged model without ground truth supervision. For large datasets with many output logits like ImageNet, we also have LinR confidence (LinRC) to filter out the logits with low confidence. As for the importance of the mix operation, we also have LinR with no mix version (LinRNM), which performs comparable well.

We carry out our experiments on one label and multi-label classification tasks on CIFAR-10/100, ImageNet, PETA, PA-100k to demonstrate our broad effectiveness. We also find an interesting phenomenon called "sweet point" that can provide more guidance about when to do our LinR refine. To visually understand how LinR works to provide good results. We also use [23] method to plot the loss function's landscape in a low dimension.

Our main contributions can be summarized as follows:

- The LinR on top of the existing regularization is newly proposed to improve neural network's generalization by putting linear regularization on refining. By LinR, we can refine the converged model to have better results even without ground truth supervision, which is like self-training. Meanwhile, we find there is a "sweet point" which can provide us guidance on when to put LinR in training schedule.

- LinR can work well on broad datasets with both one label classification and multi-label classification and network architectures. LinR can produce stable and more generalized parameters compared with fine tuning.

- Further more than mixup [38], we found LinR can work with no mix operation (LinRNM). Which means it is linearity helps the network better generalized. Also we hold explanation about how linearity improve results in the view of landscape of loss function.

2. Related Works

Regularization in training neural network Reducing generalized error in training neural network is vital. The commonly used method to reduce overfitting is dropout [35] which implemented by randomly removing connections between neurons in a neural network to prevent complex co-adaptations on training data. L2 regularization [28] uses squared magnitude of coefficient as penalty term to the loss function. L1 regularization [28] is similar to L2 regularization, but it will produce more sparse parameters. To overcome the phenomenon of internal covariate shift [17], batch normalization solve the problem through a normalization step that fixes the means and variances of each layer's inputs. Also there are many normalization [2, 37]. L1/L2 regularization put regularization on microcosmic view. But LinR do regularization on macroscopic manner by minimizing a loss function only concerned with input and outputs other than every parameters. Also, we don't have to change any architecture like the normalization methods.

Linearity in network architecture ResNet and DenseNet are representative networks that has more Linearity compare with the networks without shortcuts. As [23] describes, shotcuts can provide more smooth loss surface and linearity in network. Other than changing connection in network to gain more smooth loss surface, LinR can also make the loss surface more smooth with help of a simple loss function.

Mixup Mixup's [38] operation for processing image is a method that LinR borrows to construct the mix operation: $ax + by$. In a sense, mixup can be treated as a special case of LinR. Mixup also regularizes the neural network to promote linear behavior in training. The difference between mixup and LinR can be: Mixup needs label and the mix operation which can boost linearity's additivity property Eq. (1). But LinR can promote linearity without mix operation which can be treated as only promoting Homogeneity of degree 1 Eq. (2) . Also, our results can improve further on well mixup trained baselines. Further experiments will demonstrate that.

Self-supervised learning SSL [5, 14, 10] empowers us to exploit a variety of labels that come with the data for free. For producing a dataset with clean labels is expensive. SSL want to make full use of the unlabeled data with aid of self-supervised tasks, also named pretext task. However SSL usually don't concern about the final performance of the specific downstream tasks. So SSL can't refine a task-specific converged model to have further better results. In the view of different stage to join in the model fine tuning procedure, LinR can be a remedy to make full use of data in during training and after training stage in an unsupervised way.

Moreover, BYOL [11] and LinR both output two tensors and compute the loss between them. We both use stop-gradient to keep one output tensor unchanged and push another tensor closer to it. We carry on experiment on changing it to have gradient for comparison shown as table 6.

3. Linear Regularization (LinR)

In this section, we explain our proposed Linear Regularization (LinR) in detail. We start from the property of lin-

ear function in calculus, and then a general LinR algorithm is introduced and illustrated. Furthermore, we extend the LinR to different versions for tackling different scenarios including large number of categories and refinement with no label.

3.1. Linear Regularization (LinR) in Neural Network

In calculus, a linear function $F(x)$ is a function that satisfies additivity and homogeneity of degree 1:

Additivity:

$$F(x+y) = F(x) + F(y) \tag{1}$$

Homogeneity of degree 1:

$$F(\gamma x) = \gamma F(x) \tag{2}$$

From above two equations, we can deduce:

$$F(ax+by) = aF(x) + bF(y) \tag{3}$$

Generally, if we treat a neural network as a function, then $f(\theta)$ represents the feedforward network with parameter θ, which is composed of both linear and nonlinear components such as convolutional layers and activation functions. x and y represent two input images, then ax means x multiplied by a scalar a. An ideal linear network should have the property of Eq. (3), such as one single fully connected layer. But such a network only has very pool representation capability and it cannot fit the data distribution. As illustrated in [23], the loss surface of deep networks tend to have significant non-convexity and less non-convexity would bring better generalization ability. Considering a linear function is always convex, we suspect that if a network is compatibly linear without losing its expression ability, it may gain a more smooth landscape [23] and generalize better to unseen test distribution. So we propose a regularization to endow a neural network with more linearity to improve generalization, and at the same time keep its representation capability. Then the network should benefit from both of them. That is to say we expect a trade off between linearity and strong expression in our model.

To this end, We introduce a regularization function $loss_{LinR}$ to put constrain on parameters θ. Given two different images $(x,\ y) \sim E$, we define the loss function of proposed LinR, i.e. $loss_{LinR}$, as follows:

$$loss_{LinR} = Dis(P1(x,y|\theta), P2(x,y|\theta)) \tag{4}$$

$$P1 = f(ax+by|\theta) \tag{5}$$

$$P2 = af(x|\theta) + bf(y|\theta) \tag{6}$$

$$Dis(P,Q) = D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \tag{7}$$

Algorithm 1 LinR method

Input:
Training data sets, converged model with parameter θ^p, N_{cls}, N_{LinR}, T. N is number of training data. B_{cls}, B_{LinR} are batch size for computing $loss_{cls}$ and $loss_{LinR}$.
Output: Refined network with parameter θ^{LinR}

```
initialize the model parameters θ with θ^p.
t ← 0
while t < T do
    t ← t + 1.
    update θ with classification loss for N_cls iterations by Eq. (8).
    update θ with loss_LinR for N_LinR iterations by Eq. (7).
end while
return θ^LinR ← θ
```

$f(\theta)$ here denotes the neural network. Dis stands for the distance between $P1$ and $P2$, where we apply KL-divergence to compute the distance. a and b are two parameters and they satisfy a constraint that $a+b=1$. In our setting, a is sampled from $Beta(\alpha,\alpha)$ where $\alpha = 0.75$, and b is changed according to above constraint. For numerical stability, we stop the gradient from $P2$ in the computing graph, which is easily implemented by detach function in Pytorch code.

3.2. Refine Model with LinR

With the loss function of our proposed Linear Regularization (LinR), the next step is how we can properly apply it to improve our model. Considering the trade off between representation capacity and linearity, We propose to take a converged model and utilize our LinR loss to refine the model. More specifically, the network is updated iteratively by LinR loss and traditional classification loss. The diagram and algorithm in detail are shown in Fig. 2 and Algorithm 3.2.

As in Algorithm 3.2, we first take a converged model and perform our refinement for t iterations. In each iteration, we first update the parameter with conventional classification loss $loss_{cls}$ for N_{cls} iterations and then LinR loss $loss_{LinR}$ for another N_{LinR} iterations. Classification loss function can be written as Eq. (8).

$$H(p,q) = -\sum_{x\in\mathcal{X}} p(x)\log q(x) \tag{8}$$

To be specific, N_{cls} and N_{LinR} are often set as N/B_{cls} and N/B_{LinR}, which means we go through the data of entire dataset for each loss and then switch. But for larger data set like ImageNet [8], if we set N_{LinR} and N/B_{cls} as the needed iterations to go through the whole dataset, the network may already forget what it learns from last loss when

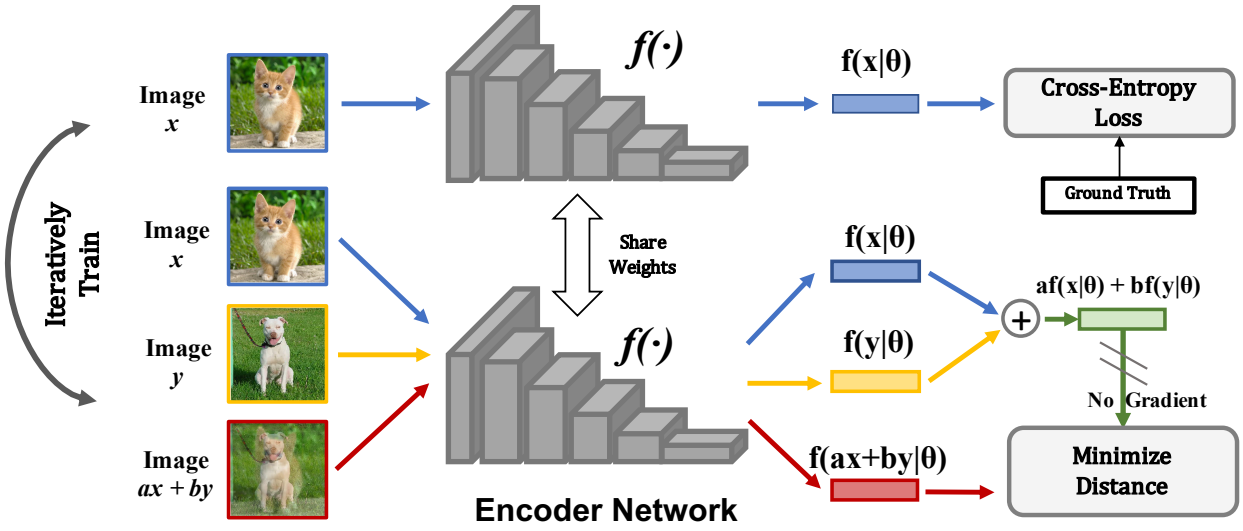


Figure 2. Diagram of propose LinR refinement. In Linear Regularization iteration, a KL-divergence loss between $f(ax + by|\theta)$ and $af(x) + bf(y)$ is minimized, where θ are the trained weights. In Classification iteration, cross-entropy loss is minimized. The weights θ are shared. And we refine our model by iteratively training with those two losses.

finishing iterations of current loss, which makes the iterative training meaningless. To this end, we set the N_{cls} and N_{LinR} to 500 in ImageNet dataset.

Since we initialize the parameters with converged model, there could be some concern about the degree of convergence. From empirical result in Sec. 4.8, we find our refinement is actually quite robust to that and we can boost the performance without needing a fully converged model.

3.3. Linear Regularization Confidence (LinRC) and Linear Regularization with No Label (LinRNL)

In the experiment session, we empirically find the category of output logits (number of classes, NC for short) affects the refining behavior. We observe that Algorithm 3.2 cannot work well with ImageNet which contain 1000 categories while CIFAR-10 only contain 10 categories. $P2$ can be treated as a pseudo label under the setting of self-training algorithm. The precision of pseudo label is vital to the success of self-training algorithm. This motivate us to mask out low confidence regions which provide meaningless information other than noise. As shown in the follow equation:

$$\begin{aligned} mask &= P2 > \frac{1}{NC} \\ P1 &= P1[mask]. \\ P2 &= P2[mask]. \end{aligned} \tag{9}$$

We will generate a mask which set True for category larger than a threshold vice versa. After acquiring mask, we apply mask over P1 and P2 and only calculate loss over the selected logits. We name our algorithm Linear Regularization Confidence (LinRC).

Also, in the refining procedure, to better understanding LinR, we have the LinR with no label version (LinRNL): Remove line 5 in Algorithm 3.2. In this way, we don't have labels as supervision to do refinement. We may wonder if LinRNL can improve the model's performance by only bringing more linearity to converged models.

To make it clear, in following experiments, we let LinR refer to the original algorithm in Sec. 3.2, LinRC and LinRNL respectively refer to two proposed extensions.

4. Experiments

In this section, we evaluate the effectiveness of the $f(\theta^{LinR})$ refined by LinR, LinRNL, LinRC. We first test our method on ImageNet with LinRC. We conduct experiment on CIFAR 100 with LinR on both strong and week baselines respectively with AlexNet and VGG-19 [33]. We do refinement with no ground truth labels (LinRNL) on CIFAR-10 to demonstrate the availability of LinRNL. PETA and PA-100k are typical multi-label datasets with many attributes. In the view of output logits, multi-label classification uses sigmoid other than softmax to operate with output logits. So the logits distribution between one label classification and multi-label classification is quite different. Will the output distribution impact algorithm's performance? We carry on experiments on PETA and PA-100k with LinR to verify it. Experiments show our method LinR can perform well on both one label classification and multi-label classification problems.

We also compare mixup and LinR on refining model for they both have mix operation. We carry on 2 experiments:

1. Can LinR improve further on converged model trained with mixup? 2. Can a Linear regularization do without mix operation by only utilizing homogeneity of degree 1 Eq. (2). The first Experiment shows LinR can improve further with mixup. The second experimetnt shows LinR without mix operation can also work comparable well.

4.1. Evaluation Metrics

For CIFAR-10/100, we use accuracy as metric. For ImageNet, we use top 1/5 accuracy ks metrics. For multi-label datasets (PETA, PA-100K) we have mA, F1-score, accuracy, precision, recall as metrics. Among the 5 metrics, mA and F1-score are the most convincing metrics.

4.2. Data and Implementation Details

ImageNet consists of 1.2 million natural images for training and 50000 images for validation in 1000 classes. The CIFAR-10 [19] dataset consists of 60000 images in 10 classes. The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes. CIFAR-100 is a bit harder than CIFAR-10 since the baseline results of CIFAR-10 are far lower than CIFAR-100. So we use CIFAR-10, 100 for different experiments. We use ResNet-18 as backbone on ImageNet, for it is lightweight and fast. We use VGG-19 and AlexNet on CIFAR-100, for their baseline accuracy has a big gap. We use VGG-19 and DenseNet-121 respectively on CIFAR-10, for VGG-19 and DenseNet's model architectures are quite different (DenseNet has many shortcuts.). The PETA dataset consists of 19,000 images with 61 binary attributes and 4 multi-class attributes. The PA-100k dataset is by far the largest dataset for pedestrian attributes recognition, which includes 100,000 pedestrian images in total collected from outdoor surveillance cameras.

The experiment code is implemented with Pytorch. In LinR refining schedule, the optimizer for classification loss is SGD with learning rate 3e-4, momentum 0.9, and weight decay 5e-4. The optimizer for $loss_{LinR}$ is SGD with momentum 0.9, learning rate 1e-2 or 1e-3 with no weight decay. Each experiment runs with $T = 50$. N_{LinR} is set as needed iterations to go through an epoch for CIFAR-10/100, PETA and PA-100k. N_{LinR} of ImageNet is 500.

4.3. Performance on One Label Datasets Classification

4.3.1 Experiments on ImageNet and CIFAR-100

For heavy computation, we carry on ImageNet on 8 GPUS with ResNet-18 with batch size 32. We use LinRC to filter the logits with low value as Eq. (9). On CIFAR-100, we apply LinR with AlexNet and VGG-19. The baseline weights for ImageNet and CIFAR-100 are from [3].

In Table 1, ImageNet improves 0.402 % and 0.153 % on top1/top5 accuracy compared with the baseline. In Table 2, we can see LinR can improve accuracy of VGG-19 and AlexNet for 0.73% and 2.18 % compare with their baselines.

Table 1. The results of ImageNet on ResNet-18, the baseline result is from well trained model. We can improve the results in few refining epochs.

	Top1 accuracy	Top5 accuracy
baseline	69.758	89.0769
baseline + LinRC	**70.16**	**89.23**

Table 2. The results of CIFAR-100 on VGG-19 and AlexNet. The accuracy of CIFAR-100 with the two models are quite different. LinR can improve baselines with both high and low results.

	VGG-19	AlexNet
baseline	72.59	43.87
baseline + LinR	**73.32**	**46.05**

Table 3. Accuracy of CIFAR-10 on baseline and refining. The baseline is converged. But we can refine baseline to have even better results than baselines with LinRNL. We test the method on VGG-19(bn) and DenseNet-121.

	VGG-19	DenseNet-121
baseline	91.08	92.94
baseline + LinR	**93.37**	**95.27**

4.3.2 LinRNL Results on CIFAR-10

CIFAR-10 is an easier dataset compared with CIFAR-100, so repeat the same refining schedule as LinR is unchallenged. So we try an aggressive way to do refine without labels. We think $P2$ can provide enough supervised information. We exclude line 5 in Algorithm 3.2 when carrying experiment on CIFAR-10. In this way, LinRNL is like doing self-training in a mix way. We first train from scratch to have a comparable results which achieve accuracy 91.08% on VGG-19 and 92.94% on DenseNet-121 as baselines. Then we refine the baseline model with LinRNL. After a few epochs of refining, we find the baseline after refining can be better even without supervision. Results are as Table 3. LinR improve the accuracy on VGG-19 and DenseNet-121 for 2.29% and 2.23% respectively. Interestingly, refining can produce better results than baselines with no ground truth supervision.

4.4. Performance on Multi-label Datasets

We refine on PETA and PA-100K datasets with LinR and compare it with the SOTAs. We carry on experiments on the baseline. We only run it with $T = 3$. On PETA and PA-100k, our method achieves best in mA, F1 and recall scores. There is a trade-off between precision and recall,

Table 4. Results of PETA with ResNet-50 backbone, all the compared models' backbone are ResNet-50. Strong baseline is the SOTA among all earlier results, and we achieve new SOTA by putting LinR method on the strong basline. Here we call strong baseline SB for short.

	mA	Accu	Prec	Recall	F1
MsVAA [32]	84.35	78.69	87.27	85.51	86.09
VAC [12]	83.63	78.94	**87.63**	85.45	86.23
ALM [36]	84.24	77.84	85.79	85.60	85.41
SB [18]	85.30	79.00	86.62	86.52	86.30
SB + finetune	85.24	**79.14**	86.84	86.45	86.38
SB + LinR	**85.92**	79.07	84.63	**89.05**	**86.47**

Table 5. Table shows results of PA-100K of ResNet 50 and comparison between several SOTAs including more other backbones. PA-100K with fine tune can't make improvement on mA or F1 score. With LinR, we improve mA by 0.77% and have comparable F1% score, also with 0.91% on recall.

model	mA	Accu	Prec	Recall	F1
DeepMAR [21])	72.70	70.39	82.24	80.42	81.32
HPNet [26]	74.21	72.19	82.97	82.09	82.53
LGNet [26]	76.96	75.55	86.99	83.17	85.04
PGDM [22]	74.95	73.08	84.36	82.24	83.29
VSGR [24]	79.52	**80.58**	89.40	87.15	**88.26**
VRKD [29]	77.87	78.49	88.42	86.08	87.24
AAP [13]	80.56	78.30	**89.49**	84.36	86.85
VAC [12]	79.16	79.44	88.97	86.26	87.59
ALM [36]	80.68	77.08	84.21	**88.84**	86.46
SB	80.55	79.06	87.51	87.13	86.93
SB + finetune	80.55	78.69	87.17	86.95	86.66
SB + LinR	**81.32**	78.96	86.61	88.04	86.90

so mA and F1 score are more reliable and convinced. Our algorithm can acquire the best results on mA on the two datasets. We also have ablation experiments to verify the improvement is from the refinement other than fine tuning (line 6 in Algorithm 3.2.).

PETA is a relatively small data set, lots of benchmarks are carried on ResNet-50, so we set ResNet-50's strong baseline as our baseline and use it to refine. PA-100k is relatively larger and we compare it with many SOTAs. On PETA, Table 4 shows our results achieve the highest mA, F1 and recall score. LinR improves mA score for 0.68% and F1 score for 0.09%. Besides, we have the highest recall score 89.05% where the before works' recall have never surpass 87%. On PA-100K, Table 5 achieves best in mA and recall score. It improves mA for 0.77% and comparable F1 86.90% compared with the baseline's recall 86.93%.

On both PETA and PA-100k, fine tuning won't improve mA or F1 scores. It shows that our method's superiority compared with fine tuning.

4.5. Refining and Fine Tuning

We find LinR can reduce overfitting compared with fine tuning. First we have a baseline model trained on CIFAR-10 with accuracy 91.08%. We respectively fine tune it with labels and refine it without label by LinRNL. When we do fine tuning, we use SGD with learning rate 0.001 and momentum 0.9 with weight decay 5e-4. We fine tune and refine it both for 200 epochs, and plot the figure where y axis represents test accuracy and x axis represents train accuracy. In Fig. 3, We can see the refined model achieves the highest results. Also, for given train accuracy, LinRNL can have better accuracy, which means LinR can reduce overfitting and produce better results on test data.

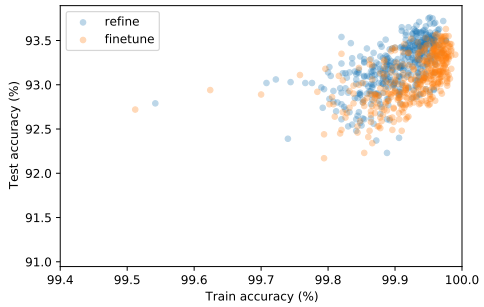


Figure 3. On CIFAR-10's converged model with accuracy 91.08%. Blue dots represent the train/test accuracy on LinRNL. Orange dot represents the train/test accuracy on fine tuning. A better model with a upper left dot is less overfitting and better generalized.

4.6. About Specific Settings of LinR

To demonstrate the robustness of LinR on choosing α, we set α equals 0.1, 1, and 6 respectively on the experiment on CIFAR-10 with LinRNL. About the Dis, we also change it to mean square error (MSE) compared with KL-divergence. As to the setting of stopping gradient of $P2$, we also change it to have gradient. The results are shown as Table 6. It shows that LinR is robust to α's value. But LinR is sensitive to different Dis. We suppose MSE's regularization may be too strict. We can treat $P2$ as pseudo label. If we enable $P2$ with gradient, the pseudo label may change and lose its information.

4.7. LinR and Mixup

Mixup is a similar method to our method LinR. We first train a baseline with mixup on CIFAR-10 with DenseNet-121. Compared with the baseline with accuracy 91.08%. The baseline well trained with mixup has accuracy 93.54%. On the stronger baseline with mixup, we use LinRNL and achieve accuracy 95.9%. It means that our method LinR can improve results further on a model well trained with mixup. The results are shown as Table 7.

Table 6. The default setting is KL divergence as Dis, α equals 0.75, $P2$ are with no gradient. The baseline line accuracy is 91.08%. Each experiment changes a factor to different way and other factors keep unchanged. MSE change KL-divergece as $loss_{LinR}$. $P2$ gradient enable $P2$ with gradient. We can see α's value is arbitrary to choose. But change Dis to MSE or enable $P2$ with gradient harms LinR's performance.

setting	accuracy(%)	setting	accuracy (%)
$\alpha = 0.1$	95.56	$\alpha = 6.0$	95.32
MSE	93.37	$P2$ gradient	93.27

Table 7. We put on LinR respectively on well trained baselines train with and without mixup, the baseline with mixup performs better than baseline without mixup. And the baseline + LinR can improve further on the mixup baseline. It shows our method is non-overlap with mixup.

	mixup trained	no mixup trained
baseline	93.54	92.94
baseline + LinR	**95.9**	**95.27**
baseline + LinRNM	-	95.13

In another aspect, we can have a version of LinR without mixed operation. We call it as LinRNM (Linear regularization with no mix). It can be written as Eq. (10):

$$loss_{LinRNM} = Dis(f(ax|\theta), af(x|\theta)) \tag{10}$$

Dis represents the KL-divergence as Eq. (7) and a is a scalar range from 0 to 1. It is a special case of LinR without mix operation. It can be easy to derive from Eq. (2).

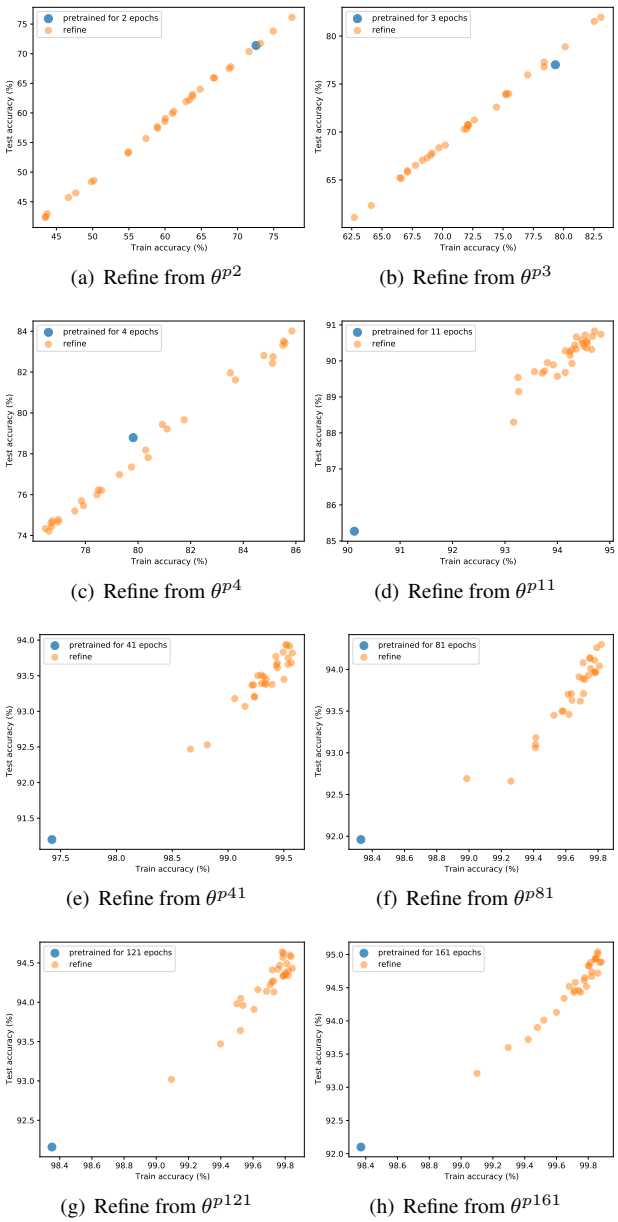
In Table 7, The baseline (without mixup) + LinRNM can show the LinRNM can also have similar accuracy equals 95.13% close to baseline (without mixup) + LinR 95.27%.

4.8. The Sweet Point

All the above experiments carry on LinR and its varieties are on the converged models. The models is converged and have good enough performance. How about our method's performance on a not converged model?

We first train a DenseNet-121 on CIFAR-10 for 200 epochs with learning rate 0.01 and save the model every epoch for a checkpoint. A model with θ^{pe} is the e th epoch trained parameter. Based on these models, we do refinement on the models with LinRNL for T = 30 with learning rate 0.001. We have the Fig. 4 about train accuracy versus test accuracy scatter on the refined models.

In Fig. 4, across figures from (a) to (h), there are always refined results (orange points) higher than the baselines (blue points). It means LinR can improve the model's performance no matter the baseline is low or high. From (a) to (d), more and more refined results are better than the baselines. It shows that a higher baseline will have more stable and even higher results. Also, there is a "sweet point"



(a) Refine from θ^{p2} (b) Refine from θ^{p3}

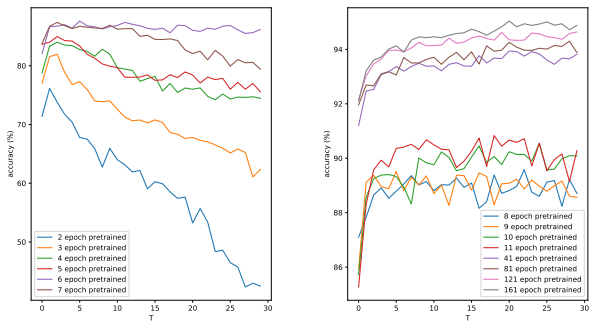
(c) Refine from θ^{p4} (d) Refine from θ^{p11}

(e) Refine from θ^{p41} (f) Refine from θ^{p81}

(g) Refine from θ^{p121} (h) Refine from θ^{p161}

Figure 4. Plots of various refine results based on different trained DenseNet-12 on CIFAR-10. From (a) to (h), models have a better starting point with more accuracy for they have trained for more epochs. The blue points are the accuracy on train/test dataset on trained model. The orange points indicate the performance of refined model in t within T times.

between (a) and (d)'s epochs. We define "sweet point" as the dividing line epoch: Before the "sweet point" epoch, the refine results will fluctuate at baselines. But after the "sweet point" epoch, all the refine results will surpass the baselines. The "sweet point" can give us guidance for when to put LinR in training schedule. Also, we can plot the results in view of refining duration as Fig. 5. In figure (a),



(a) Before sweet point (b) After sweet point

Figure 5. Plots of refining results with different converged model. (a) include models trained from 2 epochs to 7 epochs. These refine results only improve at the first refining epoch, but then the results will go down. (b) shows the refining results after 8 epoch will be stable and can improve a lot.

as we refine, the accuracy will only increase at the first epoch. Then it will go down and fluctuate. But with baseline trained more than 7 epochs's refinement results are stable and always better than the baselines. This procedure is like reviewing knowledge after school without teacher's supervision, but you can learn from your homework or notes. LinR refining is like doing homework to review and learn better.

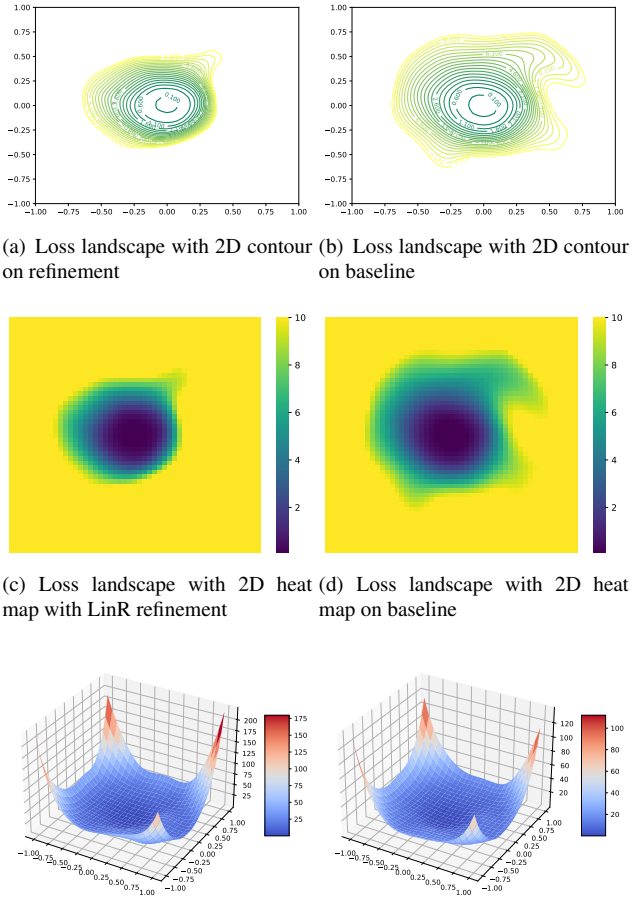
4.9. Understanding LinR in Viewing Landscape

Here we use [23] to visualize the loss landscape of the network in low dimension. We train a VGG-9 on CIFAR-10 and then refine it with LinRNL. The baseline's accuracy is 88.93% and the refined accuracy is 90.48%.

From Fig. 6, we can see the landscape of refining is more smooth on the surrounding area and quite similar around the minimal which is the dark green area in (c) and (d). By putting on linear regularization, the landscape can be more smooth. And this character of smoothness can release overfitting and improve generalization.

5. Conclusion

There has been rapid recent progress in training networks to have better results. Unfortunately, we don't have a regularization that operates in view of the whole network's linearity. We propose Linear regularization (LinR) and its variants to efficiently improve the results on wide datasets and model structures. We also develop LinRNL to refine with no ground truth and find the phenomenon called "sweet point" that can give us instruction on when to put LinR in the refining schedule. To demonstrate the difference between LinR and mixup, We find model trained with mixup



(a) Loss landscape with 2D contour on refinement (b) Loss landscape with 2D contour on baseline

(c) Loss landscape with 2D heat map with LinR refinement (d) Loss landscape with 2D heat map on baseline

(e) Loss landscape with 3D surface with LinR refinement (f) Loss landscape with 3D surface on baseline

Figure 6. The plots shows the loss landscape on VGG-9 baseline (b, d, f) trained on CIFAR-10 and its refined landscape (a, c, e) . From the 2D contour and 2D heat map landscape figure, we can see the refined landscape is similar in the middle where the loss is minimal, but they are different on the surrounding area of the landscape. The refined landscape is smooth and looks like a circle, where the baseline landscape has many branches. On the 3D surface, we can see they are general structural similar which indicates LinR don't change the landscape too much.

can still improve further with LinR. Further, we propose LinR without mix operation (LinRNM) and it also works well. We also change the default setting to explain the rationality of our algorithm. To explain the reason why our method works, LinR is explained in the view of loss function's landscape.

References

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017. 1

[2] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. 2

[3] bearpaw. pytorch-classification. https://github.com/bearpaw/pytorch-classification. 5

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 1

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 1

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 1

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[9] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995. 1

[10] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Pires, Zhaohan Guo, Mohammad Azar, Bilal Piot, Koray Kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 06 2020. 2

[12] Hao Guo, Kang Zheng, Xiaochuan Fan, Hongkai Yu, and Song Wang. Visual attention consistency under image transforms for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 729–739, 2019. 6

[13] Kai Han, Yunhe Wang, Han Shu, Chuanjian Liu, Chunjing Xu, and Chang Xu. Attribute aware pooling for pedestrian attribute recognition. *arXiv preprint arXiv:1907.11837*, 2019. 6

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 2

[15] 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*, pages 770–778, 2016. 1

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015. 2

[18] Jian Jia, Houjing Huang, Wenjie Yang, Xiaotang Chen, and Kaiqi Huang. Rethinking of pedestrian attribute recognition: Realistic datasets with efficient method. *arXiv preprint arXiv:2005.11909*, 2020. 6

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. 1

[21] Dangwei Li, Xiaotang Chen, and Kaiqi Huang. Multi-attribute learning for pedestrian attribute recognition in surveillance scenarios. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 111–115. IEEE, 2015. 6

[22] Dangwei Li, Xiaotang Chen, Zhang Zhang, and Kaiqi Huang. Pose guided deep model for pedestrian attribute recognition in surveillance scenarios. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2018. 6

[23] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018. 1, 2, 3, 8

[24] Qiaozhe Li, Xin Zhao, Ran He, and Kaiqi Huang. Visual-semantic graph reasoning for pedestrian attribute recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8634–8641, 2019. 6

[25] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 82–92, 2019. 1

[26] Xihui Liu, Haiyu Zhao, Maoqing Tian, Lu Sheng, Jing Shao, Shuai Yi, Junjie Yan, and Xiaogang Wang. Hydraplus-net: Attentive deep features for pedestrian analysis. In *Proceedings of the IEEE international conference on computer vision*, pages 350–359, 2017. 6

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[28] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-*

first international conference on Machine learning, page 78, 2004. 1, 2

[29] Li QZ, Huang KQ, et al. Pedestrian attribute recognition by joint visual-semantic reasoning and knowledge distillation. 2019. 6

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[31] Youngmin Ro, Jongwon Choi, Dae Ung Jo, Byeongho Heo, Jongin Lim, and Jin Young Choi. Backbone cannot be trained at once: Rolling back to pre-trained network for person re-identification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8859–8867, 2019. 1

[32] Nikolaos Sarafianos, Xiang Xu, and Ioannis A Kakadiaris. Deep imbalanced attribute classification using visual attention aggregation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 680–697, 2018. 6

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 4

[34] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *2017 IEEE international conference on computer vision (ICCV)*, pages 3544–3553. IEEE, 2017. 1

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 1, 2

[36] Chufeng Tang, Lu Sheng, Zhaoxiang Zhang, and Xiaolin Hu. Improving pedestrian attribute recognition with weakly-supervised multi-scale attribute-specific localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4997–5006, 2019. 6

[37] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017. 2

[38] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2