Neurocomputing 454 (2021) 189–200

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Active dropblock: Method to enhance deep model accuracy and robustness

Jie Yao [a], Weiwei Xing [a,*], Dongdong Wang [b], Jintao Xing [a], Liqiang Wang [b]

[a] School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China
[b] Department of Computer Science, University of Central Florida, 32816, USA

ARTICLE INFO

Article history:
Received 23 November 2020
Revised 22 April 2021
Accepted 25 April 2021
Available online 5 May 2021
Communicated by Zidong Wang

Keywords:
Deep neural network
Robustness
Dropblock
Active learning

ABSTRACT

In this study, we investigated a means to improve the robustness of deep network training on visual recognition tasks without sacrificing accuracy. The contribution of this work can reduce the dependence on model decay to gain a strong defense against malicious attacks, especially from adversarial samples. There are two major challenges in this study. First, the model defense capability should be strong and improved over the training stage. The other is that the degrading of the model performance must be minimized to ensure visual recognition performance. To tackle these challenges, we propose active dropblock (ActDB) by incorporating active learning into a dropblock. Dropblock effectively perturbs the feature maps, thus enhancing the invulnerability of gradient-based adversarial attacks. In addition, it selects an optimal perturbation solution to minimize the objective loss function, thereby reducing the model degradation. The proposed organic integration successfully solved the model robustness and accuracy simultaneously. We validated our approach using extensive experiments on various datasets. The results showed significant gains compared to state-of-the-art methods.

1. Introduction

Extracting correct features is critical to high-performing visual recognition model. However, this process is complex over deep model training and some incorrect focus on tiny details can cause high sensitivity and instability of deep learning classifiers. Due to this weakness, the obtained classifier may be easily fooled by some tiny imperceptible perturbation within query image.

Arising from this model fragility, lots of research starts to challenge the robustness of deep learning classifiers and proposed a series of model defense schemes. Most work on robustness in deep learning methods for vision has focused on the important challenges of robustness to adversarial examples[1,2]. However, building such robust models has proved to be quite challenging. Specifically, training a robust model may not only consume more resources, but also cause the standard accuracy to decrease[3].

In order to address this issue, we study how to develop a training scheme for visual recognition deep model with strong invulnerability and high accuracy. This work is mainly prepared in three aspects. First, we will incorporate perturbation into training process to reduce sensitivity to noise interference. In addition, we develop an active search scheme to optimize perturbation pattern and training stochasticity. Finally, we modified objective function to integrate optimized perturbation with training pipeline and ensure training efficiency over robustness improvement.

Dropout [4] was proposed to resolve overfitting problem. At the very beginning, this technique was used on fully connected layers. Later, it was used on convolutional layers. Recently, dropblock[5] was proposed to drop the units in a contiguous region of a feature map. This method is effective because features in convolutional layers are correlated, even with dropout, information about the input can still be sent to the next layer, which causes the networks to overfit. So dropping features in a structured block can better regularize convolutional networks.

Other than vanilla heuristic dropblock optimization in network, we employ active learning to improve the policy from dropblock selector. We first set a dropblock policy pool and use them to equip dropout layer for training the deep neural networks. We then set one iteration forward pass for each policy to identify the one with which the current network yields highest loss, or called uncertainty. Since only one forward pass is directed for each selection, the extra cost for active dropblock is relatively low. We then apply the loss from selected policy to conduct backpropagation and update network. We iterate this process consisting of forward pass for each dropout rate, comparison among dropblock policies, and

* Corresponding author.

https://doi.org/10.1016/j.neucom.2021.04.101
0925-2312/

network backpropagation with selected largest loss. It will not stop until the network training loss converges. (See Fig. 1).

To the best of our knowledge, we are the first to incorporate active learning into dropout to achieve the goal of enhancing model invulnerability as well as preserving performance accuracy. We empirically justify our approach by comparing it with PCL, which is the state-of-the-art method. The comparison demonstrates that this approach significantly outperforms PCL across all white-box attacks and this outperformance is kept for adversarial training. In addition, our experiments show that active dropblock achieved better classification performance than vanilla dropblock and is more stable on improving the robustness of deep neural networks.

2. Related work

2.1. Attacks and defenses

Convolutional neural networks can better simulate human vision system to detect physical objects, but the mechanism is complex and different from conventional human vision system and can cause high vulnerability problems[6]. In terms of this issue, a series of research is focused on challenging neural networks with a variety of methods, such as adversarial samples[7,1].

An adversarial image is a clean image perturbed by a small distortion carefully crafted to confuse a classifier. The classifier sometimes can be fooled by these distortions[8]. Adversarial distortion is regarded as the worst-case analysis type of network robustness[9]. Meanwhile, the challenge of robustness is usually accompanied by the attack and defense of deep neural network[10]. Numerous algorithms have been proposed to make the classifier more intelligent so that the models trained from these algorithms are more robust. [11] proposes local distributional smoothness (LDS) which can be used as a regularization term to promote the smoothness of the model distribution. The LDS of the model of the input data points is defined as the robustness of the model distribution relative to the local perturbations around the data points based on the KL-divergence. [2] finds that the use of different maximization techniques for misclassified examples has negligible impacts on the final robustness, and different minimization techniques are crucial. And based on this finding, [2] proposes a new algorithm called misclassification aware adversarial training (MART) which explicitly differentiates the misclassified and correctly classified examples during the training. [12]finds that the main reason for the existence of perturbations which can fool the deep neural network is the close proximity of different class samples in the learned feature space. And it proposes to class-wise disentangle the intermediate feature representations of deep networks.

2.2. Active learning

Active learning is a well-researched area[13–15]. There are two main strategies for active learning: representative sampling and uncertainty sampling[16].

Representative sampling algorithms select unlabeled examples representing batches of an unlabeled dataset to request labels. Intuitively, once the selected representative example set is labeled, it can serve as a replacement for the entire dataset. Therefore, minimizing the loss of agent execution is sufficient to ensure low errors relative to the entire dataset. Considering in practice collecting a large set of labeled images is very expensive, [17] selects representative examples based on core-set construction. Inspired by generative adversarial learning, [18] uses active learning as a binary classification task, and tries to select examples for labeling so that the labeled set and the unlabeled pool cannot be distinguished.

On the other hand, uncertainty sampling is based on a different principle-selecting new samples to minimize the algorithm's uncertainty on the target classifier. [19] proposes a new active learning strategy designed for deep neural networks by minimizing the number of data annotation queried from an oracle during

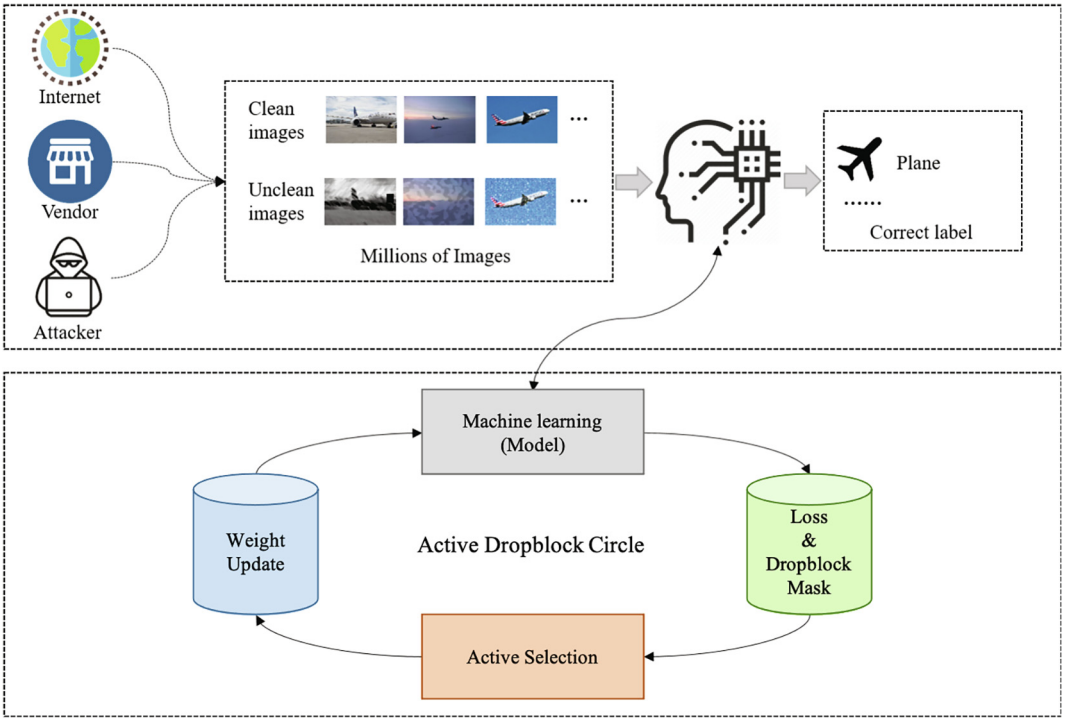


Fig. 1. The process flow of active dropblock.

training. [20] investigates some recently proposed methods for active learning with high-dimensional data and convolutional neural network classifiers. [21] proposes an approach that blends mixup and active learning. The former effectively augments the few unlabeled images by a big pool of synthetic images sampled from the convex hull of the original images, and the latter actively chooses from the pool hard examples for the student neural network and query their labels from the teacher model. And this work has inspired us to design the dropblock using the active learning.

Recently, [22] presents a new active learning strategy for semantic segmentation based on deep reinforcement learning and proposes a new modification of the deep Q-network (DQN) formulation for active learning, adapting it to the large-scale nature of semantic segmentation problems. [16] proposes a new algorithm named Batch Active learning by Diverse Gradient Embeddings (BADGE), which samples groups of points that are disparate when represented in a gradient space.

2.3. Dropout

Deep neural networks usually can easily suffer overfitting problems, which is one of numerical reasons for weak robustness. To address this issue, serveral approaches, such as weight decay and dropout[4], are proposed to alleviate overfitting by regularization technique. Dropout is rarely used in convultional layer design for lots of conventional architectures [23–27], but more popular in fully connected layer optimization[28,29]. Due to its easy implementation, lots of research is carried out to exploit the benefit from dropout such as DropConnect[30], DropPath[31], ScheduledDropPath[32], ShakeDrop regularization[33], and so forth. However, conventional dropout does not take spatial information into consideration, which limits its application within convolutional layers.To break this bottleneck, SpatialDropout[34] is proposed, where an entire channel is dropped from a feature map. Recently,inspired by Cutout[35], DropBlock[5] is proposed, which considers the spatial correlation within feature maps and zeroed out the units in a contiguous region of a feature map. This approach works effectively on defending adversarial attacks, but still requires lots of heuristic work on hyperparameter tuning and may suffer performance instability.

3. Approach

We present our approach of how to carry out the active dropout in this section. We will introduce the derivation of our approach in the following: 1) introducing the original dropout in deep neural networks, 2) constructing a big dropout selector pool which is far more complicated than the traditional way, 3) actively choosing the appropriate dropout selector to increase the uncertainty of the model and optimizing the parameters.

3.1. Dropblock in deep neural networks

In light of vanilla dropout[4], block based dropblock is developed as follows. Firstly, we can deal with the most popular case of deep feedforward networks with the activation function σ. The equation can be described as Eq. 1 where x_i^h is the output of unit i in layer h, the variable ω donates the weights and I donates the input vector.

$$x_i^h = \sigma(S_i^h) = \sigma(\sum_{l<h}\sum_{j}\omega_{ij}^{hl}x_j^l) \quad with \quad x_j^0 = I_j \tag{1}$$

Dropblock employs a mask $\mathbf{M}(\beta, B)$ to modify the feature maps, which can be described by Eq. 2. Here, $\mathbf{M}$ is the generated dropout mask which is determined by dropblock selector β and block size B. More details will be introduced in the next subsections.

$$x_i^h = \sigma(S_i^h) = \sigma(\mathbf{M}(\beta_j^l, B_j^l) \circ \sum_{l<h}\sum_{j}\omega_{ij}^{hl}x_j^l) \quad with \quad x_j^0 = I_j \tag{2}$$

This technique comes handy and effective in many works[30–32]. It can exponentially expand the size of image pool in an imagery way. For example, different strategy of β may generate different input vectors, which augments training data over epochs. The experiments justify that this training scheme can help to improve model robustness[5].

For Dropblock, there is a mask function $\mathbf{M}(\beta, B)$, which is developed by spatial sampling. Given delineated local regions, the centers of local masks $M_{i,j}$ are sampled given specific distribution. Then, the global mask $\mathbf{M}$ is obtained by the union of all local masks. Here, the local mask is generated by the rule as follows. For each zero position $M_{i,j} \in \mathbf{M}$, create a spatial square mask with the center being $M_{i,j}$, the width and height being block size B and set all the values of in the square to be zero[5].

3.2. Dropblock selector

The mask function $\mathbf{M}(\beta, B)$ is determined by dropout selector β, which follows the bernoulli distribution with the parameter of γ as shown in Eq. 3.

$$\beta \sim bernoulli(\gamma) \tag{3}$$

Here, the selection of the parameter γ is a policy decision. For example, [5] picks an unchanged policy, i.e., set the γ to a constant C_1 over the entire training stage.

$$\gamma \in \Omega \tag{4}$$

In our work, as shown in Eq. 4, we develop a finite set Ω to actively select γ. This number set consists of non-repeating and fractional numbers between 0 and 1. During each feedforward, one fraction number is actively selected and assigned to γ to generate the mask of $\mathbf{M}(\beta)$. After element-wise multiplication with this mask, feature maps are updated with dropblock scheme. With this policy setting, γ and the mask of $\mathbf{M}(\beta)$ become adaptive to training stages.

3.3. Active selection for policy

Let F denote loss function where x is the input data, y is the labels and θ is deep model parameters. We develop the following optimization functions to actively select policy. First, find the dropblock selector with largest loss after forward pass, which is described by local objective function in Eq. 5. Then, pick this selector as local optimal policy to update network by backpropagation. Globally, we still minimize the loss function over iterations until the loss converges as Eq. 6.

$$\mathscr{L}'(\theta) = \arg\max_{\delta} F(\theta, x+\delta, y) \tag{5}$$

where $\delta = x * (J - \mathbf{M}), J$ is all-ones matrix and $L'(\theta)$ is the local objective function with model parameter θ.

$$\theta = \arg\min_{\theta}[\mathscr{L}'(\theta)] \tag{6}$$

Given this selection scheme, the model will be optimized by perturbed features. The perturbations help model acquire more knowledge of potential adversarial samples, thus increasing invulnerability to attack. Also, model can benefit from data augmentation by Eq. 5.

3.4. Overall algorithm

Algorithm 1 presents the overall process of our approach **Active DropBlock**. In the beginning we have the activation maps A of previous layer, block size B, a finite policy set of dropout rate Ω, and training mode $mode \in \{Inference, training\}$. We firstly determine the training mode. If the training mode equals to *Inference* we will comply with the basic idea of dropout and return the A directly. Otherwise, we pick the drop rate γ from Ω, and we create a mask $\mathbb{M}$ which shares the same height and width with A. Then we apply the mask to A and normalize the result. Next, we will determine if A is greatly changed. If so, we will return the changed activation maps. Otherwise, we will pick another drop rate γ from Ω and redo from the fourth step.

Algorithm 1: Active DropBlock

INPUT: The activation maps of previous layer: A
INPUT: The size of active dropblock: B
INPUT: The finite policy set of dropout rate: Ω
INPUT: The training mode: $mode \in \{Training, Inference\}$
OUTPUT: The new activation maps of active dropblock: A^*
1. **If** $mode == Inference$ **then**
2. return $A^* = A$
3. **end if**
4. Pick one dropout rate γ from Ω
5. Generate the sample mask $\mathbf{M}(\beta, B)$ where $\beta \sim Bernoulli(\gamma)$
6. Apply the mask: $A = A \odot \mathbf{M}$
7. Normalize the features: $A^* = A \times count(\mathbf{M}) \div count\_ones(\mathbf{M})$
8. **If** A^* is greatly changed **then**
9. return A^*
10. **Else**
11. Start again from the fourth step

Here, the mask $\mathbb{M}$ is the union of local masks from a set of centers with value $M_{i,j} \in (0, 1)$. The values of these centers $M_{i,j}$ are sampled based on bernoulli distribution. In particular, the width and heights of these local masks are constrained by block size B.

4. Experiments

We design various experiments to test our approach, including both comparison experiments with state-of-the-art method [12] and ablation studies. Additionally, we also examine our approach when the available data is inadequate to the main task of interest.

4.1. Comparison experiments

4.1.1. Experiment Setting

Our experiments are basically constructed on CIFAR-10 and Tiny-Imagenet. CIFAR-10 is one of the most widely used datasets for machine learning research. It contains 60,000 (50,000 for training and 10,000 for testing) 32×32 color images in 10 different classes. Tiny-Imagenet is a distinct subset of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) data set, which consists of 200 different categories. Each image label has 500 training images, 50 validation images. The original image resolution is 64×64 pixels and we resize them into 224×224 pixels.

Besides that we use some fine-grained visual classification (FGVC) datasets,which are the Stanford Cars[36] and the Aircraft [37], to examine the ability of our approach in data efficiency. The Stanford Cars dataset contains 8,144 training and 8,041 testing images across 196 car classes. The classes represent variations in car make, model, and year. The Aircraft dataset is a set of 10,000 images across 100 classes denoting a fine-grained set of airplanes of different varieties.

4.1.2. Evaluation metric.

We selected the classification accuracy after adversarial attack as one evaluation metric. This evaluation criterion is adopted by a majority of attack-defense approaches and easily compared with other conventional methods.

In particular, we also proposed accuracy degeneration percentage (ADP) to more clearly show model robustness across different methods. The calculation formula is developed as follows:

$$ADP = \frac{\text{Vanilla Accuracy} - \text{Attacked Accuracy}}{\text{Vanilla Accuracy}} \times 100\% \tag{7}$$

Here, larger ADP implies that model degeneration is more severe and invulnerability is weaker.

The selected attack benchmarks include one-step gradient method, like FGSM[38] and iterative methods, BIM[39], MIN[40], and Projected Gradient Descent (PGD)[41]. Here, FGSM only carries out one iteration optimization while others may conduct several iterations to obtain adversarial samples according to the experiment setting. Among these approaches, PGD is usually a stronger attack in virtue of the optimization from the first order information of the target model. It generates an adversarial sample by starting from a random position in the clean image neighborhood $\mathscr{U}(x, \epsilon)$. This method carries our for m iterations with a step size of α as:

$$x_m = x_{m-1} + \alpha \cdot sign(\nabla_{x_{m-1}} \mathscr{L}(x_{m-1}, y)) \tag{8}$$

$$x_m = clip(x_m, x_m - \epsilon, x_m + \epsilon) \tag{9}$$

Following this understanding we design our experiments in three aspects and all experiments are carried out based on CIFAR-10 dataset.

4.1.3. Competing method

We identify one state-of-the-art method as our competing method, which is the Prototype Conformity Loss (PCL) [12]. It forces the features for each class to lie inside a convex polytope that is maximally separated from the polytopes of other classes and achieve significant gains in comparison to state-of-the-art defenses. We incorporate our proposed method into PCL and equip this method to evaluate the performance variability.

4.2. Quantitative results

First, we train our model using PCL only and apply active dropblock on the model. Then, we use selected attack methods to attack the model and compared the performance change between PCL and PCL + active dropblock, the results of which are shown in Table 1. Our method can better maintain classification accuracy after adversarial attacks. For PCL, the ADPs for FGSM, BIM, MIM, and PGD are 25.15%, 63.96%, 63.29%, and 69.93%. For PCL + active dropblock, the ADPs for FGSM, BIM, MIM, and PGD are 22.68%, 36.96%, 34.92%, and 51.03%. Apparently, PCL + active dropblock yields significantly lower ADPs and model degeneration, and exhibits higher robustness. In particular, PCL + active dropblock shows higher classification accuracy under the setting of no attack. This observation implies that active dropblock can boost classification accuracy of PCL.

Secondly, we directed adversarial training based on FGSM attack to evaluate model robustness performance. Table 2 lists the results and shows that PCL + active dropblock still exhibits sig-

Table 1
Robustness of model without adversarial training under white-box attacks.

Training	PCL		PCL + Ours	
	Accuracy	ADP	Accuracy	ADP
No Attack	90.45	-	**93.12**	-
FGSM	67.7	25.15%	**72.0**	**22.68%**
BIM	32.6	63.96%	**58.7**	**36.96%**
MIN	33.2	63.29%	**60.6**	**34.92%**
PGD	27.2	69.93%	**45.6**	**51.03%**

Table 2
Robustness of model with adversarial training using FGSM attack.

Training	PCL_{FGSM}		PCL_{FGSM} + Ours	
	Accuracy	ADP	Accuracy	ADP
No Attack	91.28	-	**91.67**	-
FGSM	75.8	16.96%	**84.7**	**7.60%**

nificantly better performance than PCL. The ADP for vanilla PCL is 16.96%, but it is reduced to 7.60% for PCL + active dropblock. It justifies that active dropblock can boost defense performance of PCL from adversarial training. Also, when there is no attack but adversarial training, PCL + active dropblock can show better performance than vanilla PCL.

Next, we conduct adversarial training on vanilla PCL with PCL + active dropblock and compare their performances after iterative attack. During the training process, we set the iteration of PGD to 10 and after training, we use PGD, BIM and MIN attack methods to attack the model. The setting of each method follows the experiment setup in PCL. From the Table 3, PCL + active dropblock shows better performance across all experiments. For PGD, BIM, and MIN, the ADPs of PCL with adversarial training are 49.18%, 49.94%, and 46.35%, but PCL + active dropblock with adversarial training shows 44.66%, 32.94%, and 31.09%, respectively. These ADP reductions indicate that active dropblock can enhance model robustness of PCL after adversarial training compared to vanilla setting. Particularly, PCL + active dropblock can show better classification accuracy than vanilla PCL when there is adversarial training but no attack.

4.3. Ablation study

We investigate each component in our proposed method through ablation study and justify that they are effectively integrated and both imperative. For ablation study, we select both CIFAR-10 and Tiny-ImageNet for benchmarks which can better cover dataset scales. To ensure that the results are comparable, all configurations are set to the same for each classification task in following studies.

4.3.1. Active Dropblock vs Vanilla Dropblock

In this section, we compare active dropblock with vanilla dropblock extensively to justify our approach.

CIFAR-10 + VGG-16 We developed the first comparison study on CIFAR-10 task. We incorporate vanilla dropblocks and active dropblocks into VGG-16, separately, and examine their performance on a set of PGD attacks. The PDG attacks are developed with incremental steps. As shown in Fig. 2, when the step is set to 1, the classification accuracy of VGG-16, vanilla dropblock and active dropblock are almost the same. When we increase the step of PGD attack, the classification accuracy is decreasing as expected. However, both active dropblock and the vanilla dropblock are helpful for the robustness of the model. And when we set the step to 20, the accuracy of VGG-16 and vanilla dropblock is close to zero while it is obvious that the decline level of our method is much lower than the other cases.

CIFAR-10 + ResNet-18 Next, we investigate the performance difference on another well-used architecture, ResNets[23], which consists of two main components, including feature extraction layer group and the classification layer group. The feature extraction layer group always consist of four major layers and each of them is composed of different number of Basicblocks or Bottlenecks. We equip vanilla dropblock and active dropblock into ResNet-18 separately, and compare the performance gain from each equipment.

In order to get a comprehensive comparison between vanilla dropblock and active dropblock, we equip vanilla dropblock and active dropblock after each major layer of ResNet-18 separately. The batch size is set to 128 for each experiment. The total number of training epoch is 200. The learning rate is initialized to 0.01 and the drop interval are 80, 60, 40 and 20. The learning rate is divided by 10 when the training process reaches drop interval. And SGD is used to optimize the model. And all the results are averaged by three independent experiments with the same setting. The γ of vanilla dropblock is set to 0.2 and we set γ of active dropblock from 0 to 0.225 with 0.025 interval.

Firstly, we train the model without the adversarial training and the block size is set to 2 in consideration of consistency. The classification accuracy on different layers is shown in Fig. 3a. From the

Table 3
Robustness of model with adversarial training using PGD attack.

Training	PCL_{PGD}		PCL_{PGD} + Ours	
	Accuracy	ADP	Accuracy	ADP
No Attack	91.89	-	**92.15**	-
BIM	46.7	49.18%	**51.0**	**44.66%**
MIN	46.0	49.94%	**61.8**	**32.94%**
PGD	49.3	46.35%	**63.5**	**31.09%**

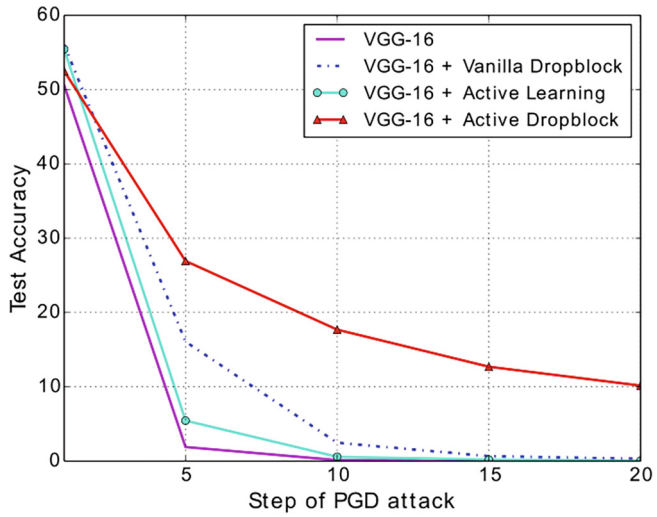


Fig. 2. Classification Accuracy (%) for CIFAR-10 under different steps of PGD attack.

experiment results we can find out that the vanilla ResNet-18 can consistently gain classification accuracy improvement after equipped with either components. Particularly, active dropblock has shown higher improvement than vanilla dropblock. The relative gain increases are 0.11%, 0.25%, 0.5% and 0.11% for major Layer 1, 2, 3 and 4, respectively. It worth mentioning that the highest relative gain occurs when we incorporate active dropblock into the third major Layer of ResNet-18. Then, we use PGD attack method with different steps to attack the model and the attack results are shown in Figures from 3b–3d. As we can see from the results, all of the classification accuracy results have been severely affected by the perturbations. However, both vanilla dropblock and active dropblock are useful in weakening the effect of PGD attack method on different layers. And our method is more effective than vanilla dropblock. It is worth mentioning that the relative increases of active dropblock are various on each layer. We envision the reason of this phenomenon could be the following: The sizes of the feature map on different layers are various and we use the same block size to carry out the experiments.

To further explore the impact of block size on active dropblock, we equipped vanilla dropblock and active dropblock on the third major layer of ResNet-18 with various block size to carry out the experiments. We set block size from 1 to 6 with 1 interval because the size of feature map generated by the third major layer of ResNet-18 is 8. As shown in Table 4, we use different steps of PGD attack to attack the model. What needs special explanation is that when we use 0 step to attack the model, we don't add any perturbation into the images, which means the classification accuracy in the third column corresponds to the normal classification accuracy just like the Fig. 3a.

As we can see in the third column of Table 4, the classification accuracy of vanilla ResNet-18 is 94.12. Both vanilla dropblock and active dropblock have shown their ability to increase the classification accuracy. Vanilla dropblock gains 0.56 improvement on average while our active dropblock can gain 0.99 improvement on average. And when we set the block size to 4, we get the highest classification accuracy on CIFAR-10 dataset. We envision the reason of this phenomenon could be the following: Both methods drop the features in structured block, which can enhance the response of the remaining features. So the model trained by either methods can have better ability to distinguish positive features from negative features in the testing dataset. However, when we use active learning to guide the training process, the value of loss is greatly changed than vanilla dropblock. Which can also improve the ability of enhancing the response of positive features.

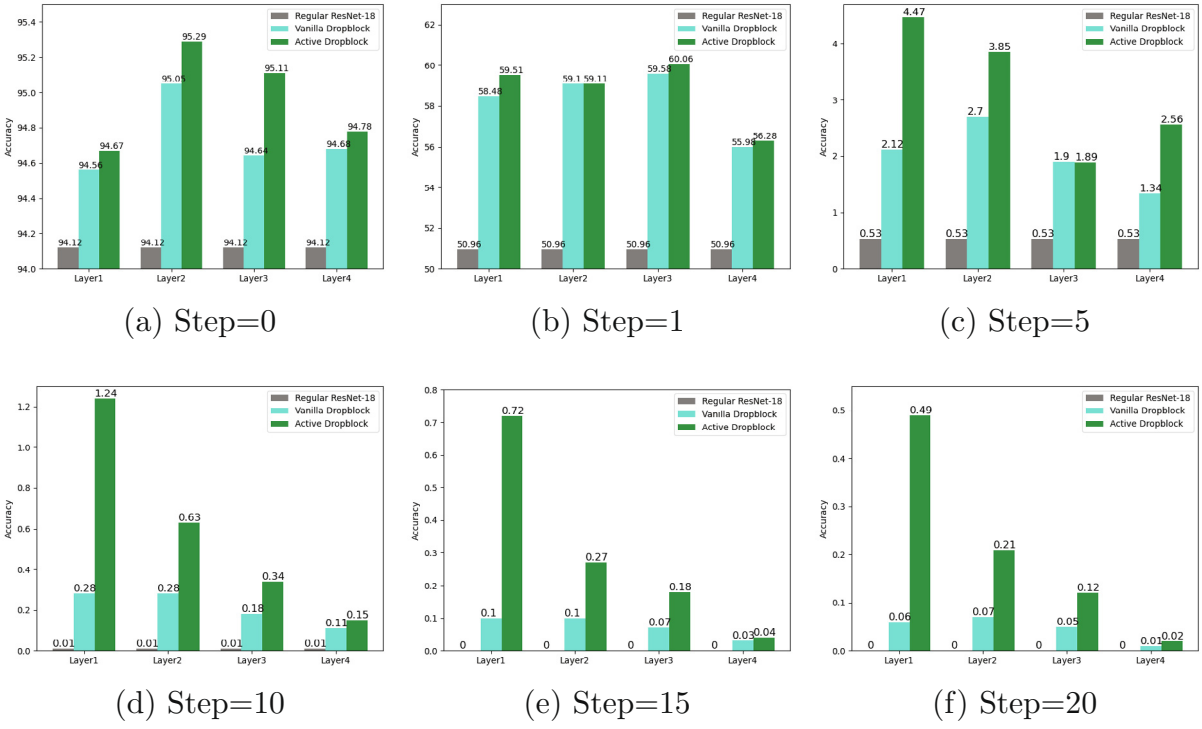


Fig. 3. Classification Accuracy (%) for CIFAR-10 on different layers of ResNet-18 combined with VanDB and ActDB. The models are trained without the adversarial training and the block size is set to 2. When the step is set to 0, we do not add any perturbation into images. So (a) is the standard classification accuracy. As the step increase, we start to use PGD to attack the model.

Table 4
Classification Accuracy (%) for CIFAR-10 under different steps of PGD attack. The dropblock is equipped on the third major layer of ResNet-18 with different block size. *DB* is short for Dropblock and *bs* is short fot block size in the light of table width.

Step		0	1	5	10	15	20
Regular		94.12	50.96	0.53	0.01	0	0
bs = 1	Vanilla *DB*	94.35	50.82	0.74	0.01	0	0
	Active *DB*	**94.81**	**57.06**	**1.62**	**0.07**	**0.03**	**0.01**
bs = 2	Vanilla *DB*	94.64	59.58	**1.9**	0.18	0.07	0.05
	Active *DB*	**95.11**	**60.06**	1.89	**0.34**	**0.18**	**0.12**
bs = 3	Vanilla *DB*	94.69	60.09	2.16	0.05	0.02	0
	Active *DB*	**95.01**	**60.7**	**3.2**	**0.31**	**0.11**	**0.05**
bs = 4	Vanilla *DB*	94.54	59.88	2.99	0.25	0.06	0.04
	Active *DB*	**95.34**	**62.53**	**5.01**	**0.44**	**0.1**	**0.05**
bs = 5	Vanilla *DB*	**95.24**	58.88	3.57	0.3	0.1	0.04
	Active *DB*	95.21	**59.25**	**5.55**	**0.53**	**0.12**	**0.05**
bs = 6	Vanilla *DB*	94.83	53.46	1.81	0.15	0	0
	Active *DB*	**95.2**	**57.97**	**3.97**	**0.5**	**0.13**	**0.05**

When we use PGD to attack the model, as shown in the rest five columns in Table 4, all of the classification accuracy results have been severely affected. Both vanilla dropblock and active dropblock have shown their ability to weaken the effect of PGD attack. However, when we set the block size to 1, which transforms dropblock into dropout, vanilla dropblock has shown slightly increment in protecting the model from PGD attack. Compared to other experiments whose block size is bigger than 1, the classification accuracy of vanilla dropblock is better than vanilla ResNet-18 after being attacked by PGD, which proves that dropping features in a structured way can improve the robustness of the model. It is worth mentioning that our acitve dropblock still shows its stable ability to improve the robustness of the model even if the block size is 1 and we have done various experiments to discuss about it in the next subsection.

When block size is bigger than 1, it is clear that both methods have better performances than the base model. Specifically, vanilla dropblock improves the baseline by 7.2, 1.96, 0.18, 0.05 and 0.03 on average when the block size are 1, 5, 10, 15 and 20, respectively. But our method has further achieved another 1.72, 1.44, 0.23, 0.08 and 0.04 accuracy increases on average when comparing to the improvements of vanilla dropblock. And when block size is set to 4, our method achieved the best increases compared to vanilla dropblock, which are 2.65, 2.02, 0.19, 0.04 and 0.01 respectively. We envision the reason of this phenomenon could be the following: As we proved before, dropping features in a structured block is one of factors in improving the robustness of the model. Meanwhile, along with the model learning process, the features of images are abstracted more and more compendious. So in order to better serve the model training process, the features to be dropped cannot be either too much or too little. In other words, different layers may require different strategies of picking the value of block size. However, according to the experiments in the Table 5, our method can always achieve improvements regardless of block size.

Normal training vs Adversarial training An adversarial sample is a clean sample perturbed by a small distortion carefully crafted. Adversarial samples can always mislead the model and produce some unexpected behaviors. And adversarial training is a collection of techniques to train neural networks on how to spot intentionally misleading data or behaviors. This differs from the standard classification problem in machine learning, since the goal is not just to spot ”bad” inputs, but preemptively locate vulnerabilities and craft more flexible learning algorithms. Generally speaking, when we carry out adversarial training, we usually generate perturbations by some certain methods, like gradient-based methods, and add these perturbations onto the original images. And then we use these adversarial images to train the model. While when we train the model equipped with the dropblock, the features will be dropped in a structured block during the training process. Intuitively, this will neutralize the effect of adversarial training to some extend.

To find out this, we carry out various of experiments on CIFAR-10 dataset using adversarial training. We use PDG with 10 steps to generate the adversarial samples during the training process. And the rest configurations of experiments are the same as the previous part.

Firstly, we investigate the performance on image classification. We deploy vanilla dropblock and active dropblock on ResNet-18 respectively. As shown in Fig. 4a, both methods have better perfor-

Table 5
Classification Accuracy (%) for CIFAR-10 under different steps of PGD attack. The models are trained using the adversarial training with 10 steps and dropblock is equipped on the third major layer of ResNet-18 with different block size. *DB* is short for Dropblock and *bs* is short fot block size in the light of table width.

Step		0	1	5	10	15	20
Regular		90.52	80.09	35.88	28.73	26.45	25.37
bs = 1	Vanilla *DB*	90.7	80.8	37.6	30.09	27.74	26.61
	Active *DB*	**90.91**	**81.42**	**38.91**	**31.49**	**28.88**	**27.87**
bs = 2	Vanilla *DB*	**91.17**	81.54	39.11	32.06	29.26	27.85
	Active *DB*	91.16	**82.01**	**39.85**	**32.25**	**29.57**	**28.24**
bs = 3	Vanilla *DB*	**91.66**	81.67	39.33	31.74	29.17	27.87
	Active *DB*	91.57	**82.38**	**41.49**	**34.15**	**31.13**	**29.95**
bs = 4	Vanilla *DB*	91.11	80.46	37.66	30.34	27.76	26.57
	Active *DB*	**91.98**	**81.64**	**41.57**	**34.9**	**32.28**	**31.1**
bs = 5	Vanilla *DB*	91.07	79.98	36.32	29.26	26.87	25.77
	Active *DB*	**92.24**	**82.43**	**37.38**	**30.65**	**28.55**	**27.5**
bs = 6	Vanilla *DB*	**91.69**	81.33	37.6	30.61	28.28	26.98
	Active *DB*	91.53	**81.38**	**38.18**	**31.13**	**28.73**	**27.51**

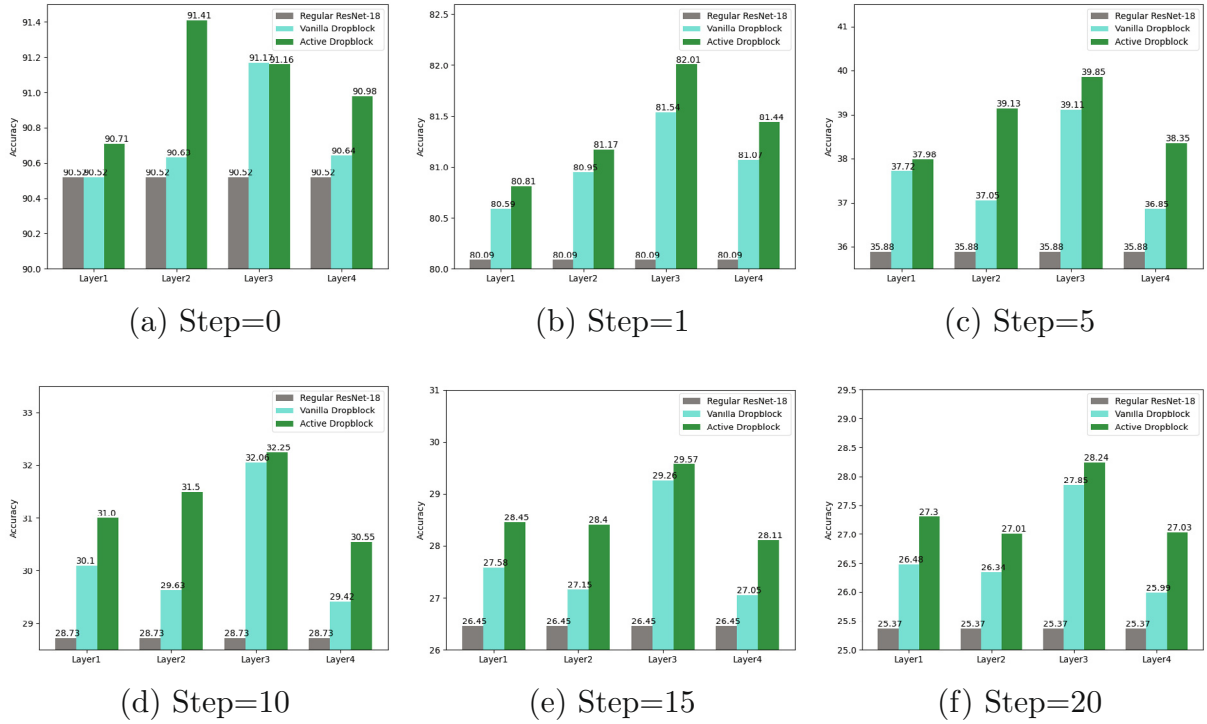


Fig. 4. Classification Accuracy (%) for CIFAR-10 on different layers of ResNet-18 combined with VanDB and ActDB. The models are trained using adversarial training with 10 steps. The block size is set to 2. As explained before, (a) stands the standard classification accuracy and the rest figures are classification results under different steps of PGD attack.

mances than the vanilla ResNet-18. But our method has further achieved better accuracy increases when comparing to the improvement of the other method in most cases.

Then, in order to find out the effect on robustness, we use PGD attack method with different steps to attack models and the results are shown in Figures from 4a–4f. As the same before, all of the classification accuracy results have been affected and both methods have shown better performances than the original ResNet-18.

To find out if there exist similar phenomenon when we change the value of block size, we carry our various experiments based on different block sizes. Firstly we train the models under the same configurations except block sizes, and both vanilla dropblock and active dropblock are equipped after the third major layer of ResNet-18. Then we use PDG with different steps to attack the models. When step equals to 0, it means that we don't add any perturbations onto the images. In other words, it represents the standard classification accuracy when the step is set to 0. As we can see from Table 5, both methods can gain accuracy increases compared to the vanilla ResNet-18. And it is worth mentioning that under adversarial training, the improvements are unstable. However, our method has further achieved another 0.33 accuracy increase on average when comparing to the improvement of the vanilla method. When we begin to increase the step of PGD attack method, the classification accuracy begin to be affected by perturbations. Both methods have shown better performances the vanilla ResNet-18. And it is obvious that our active dropblock has shown more superior ability of weakening the effect of perturbations than vanilla dropblock. Which is also worth to mention is that we still get the best performance when we set 4 to block size. Our method has further achieved another 1.18, 3.91, 4.56, 4.52 and 4.53 accuracy increase when comparing to the improvement of vanilla dropblock. From this part we can infer that the standard classification accuracy will be Influenced slightly when we use both adversarial training and dropblock. However, the robustness of model will get stronger.

Low resolution vs High resolution Then, we investigate the performance difference on larger scale dataset. We construct our experiments based on the Tiny-ImageNet. Tiny-ImageNet is the subset of the ImageNet dataset. Although the size of this dataset is relatively smaller but it covers a good variety of classes with higher resolution. Adequate class coverage ensures its moderate task complexity and ability to evaluate model power. Each class of Tiny-ImageNet only contains 500 training images while there are 200 classes in Tiny-ImageNet so it is relatively difficult for the model to learn all effective features in limited epochs. In other words it is easier to attack the model training based on this dataset successfully.

For network configuration, we select ResNets[23] as well. The batch size of these experiments is set to 64. SGD is used as the optimizer of the model. We use 0.01 to initialize the learning rate and every 30 epochs the learning rate is divided by 10. The total epoch of these experiments is 120.

As shown in Table 6, the attack step equals to 0 means the model doesn't get attacked. The characteristic of normal classification accuracy stays almost the same as what we have shown in former section, which is that by applying our active dropblock can obtain performance increasing. When the model begins to get attacked, all of the test accuracy drops are very serious. And the

Table 6
Classification Accuracy (%) for Tiny-Imagenet under different steps of PGD attack.

Model	ResNet-18		
Step	Regular	Vanilla Dropblock	Active Dropblock
0	59.84	60.80	**61.96**
1	11.40	11.82	**12.78**
5	2.72	2.8	**3.28**
10	1.94	2.16	**2.38**
15	1.82	2.08	**2.28**
20	1.82	2.10	**2.24**

vanilla dropblock is not helpful for the robustness. But our method has shown more stable effect. We envision the reason of this phenomenon could be the following: Both methods drop the features in structured block, which can enhance the response of the remaining features. This will lead to the classification accuracy increasing as the first line of Table 7. Meanwhile, this can improve the robustness of the model when ratio of training set and test set is adequate. However, when the size of training data is small, this effect will decrease. So training the model in our active way is the key to improve the robustness of the model.

ResNet-18 vs ResNet-34 We further examine the impact from deeper network structure. We incorporate vanilla dropblock and active dropblock into different layers of ResNet-18 and ResNet-34, respectively. We set the ResNet-18 to base model architecture and the experimental results are listed in Fig. 5. The vanilla ResNet-18 can consistently gain classification accuracy improvement after equipped with either components. Particularly, active dropblock has shown higher improvement than the vanilla dropblock. The relative gain increases are 1.41%, 2.48%, 1.56%, and 1.91%, for Layer 1, 2, 3 and 4, respectively. It is worth mentioning that the highest gain occurs when we incorporate active dropblock into the last layer, i.e., layer 4, of the ResNet-18. (See Fig. 6).

We also investigate model performance under ResNet-34 with more complex and deeper architecture. As shown in Fig. 5, our active dropblock consistently outperform vanilla dropblock. The relative gain increases are 1.68%, 0.91%, 2.09%, and 1.29%. In particular, the performance gain from vanilla dropblock is significantly reduced. The accuracy even becomes worse after vanilla dropblock is equipped in layer 1 and 2. By contrast, ResNet-34 still consistently benefits from active dropblock across all layers. This observation indicates that our approach shows good and stable performance in deeper network. In addition, higher performance gain occurs in later layers, which suggests that more benefits could be gained when active dropblock is equipped in later layers.

4.3.2. Active Dropblock vs Vanilla Active Dropout

We also compared active dropblock with vanilla active dropout in model robustness. The difference between two approaches is dropout strategy. Vanilla active dropout consists of pixel level (i.e., drop block size = 1) dropout and active learning. The experiments are developed with CIFAR-10 and VGG-16. The selected attack method is PGD and incremental step attack is chosen to examine model robustness. The results are illustrated in Fig. 2. Apparently, active dropblock significantly outperforms vanilla active dropout, while vanilla active dropout only shows slightly better performance than vanilla VGG-16. Meanwhile, we also carry out various experiments with CIFAR-10 and ResNet-18 as shown in Table 8. As we have seen in previous parts, active dropout have better performance than vanilla dropout while both method outperform the base model. This indicates that the benefit from active learning is significantly boosted when drop out pattern is spatially block-based. It implies that active learning and dropblock are both imperative to best performance gain of active dropout.

Table 7
Classification Accuracy (%) for Tiny-Imagenet under different steps of PGD attack.

Model	ResNet-34		
Step	Regular	Vanilla Dropblock	Active Dropblock
0	62.14	62.22	**63.52**
1	12.76	11.96	**14.56**
5	2.96	2.9	**3.6**
10	2.14	2.06	**2.68**
15	2.1	1.98	**2.54**
20	2.1	1.94	**2.54**

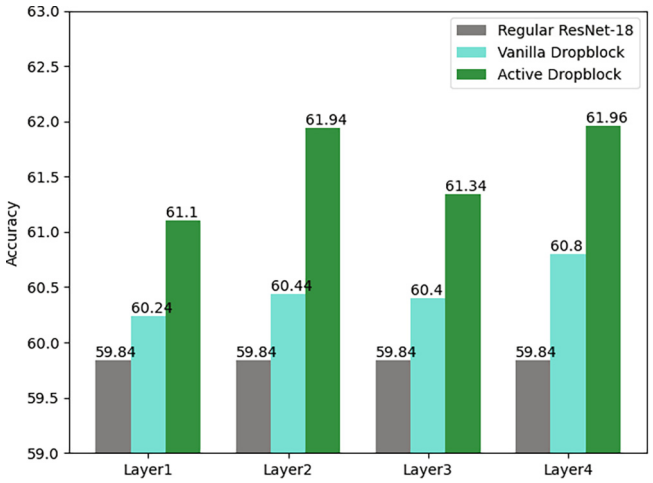


Fig. 5. Classification Accuracy (%) for Tiny-ImageNet on different layers of ResNet-18 combined with Vanilla Dropblock and Active Dropblock.

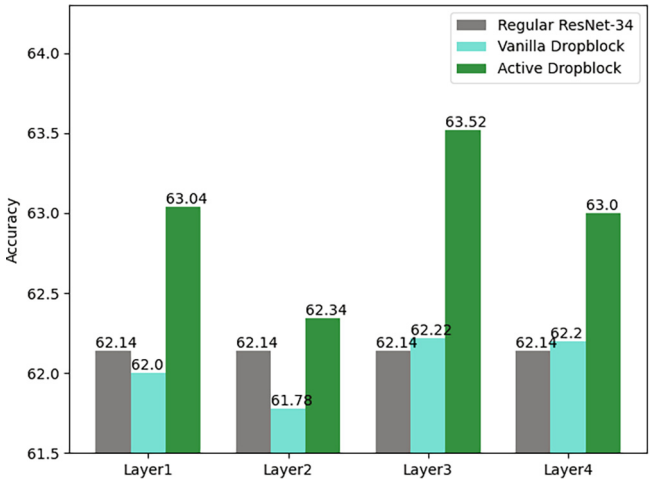


Fig. 6. Classification Accuracy (%) for Tiny-ImageNet on different layers of ResNet-34 combined with Vanilla Dropblock and Active Dropblock.

4.4. Data efficiency of active dropblock

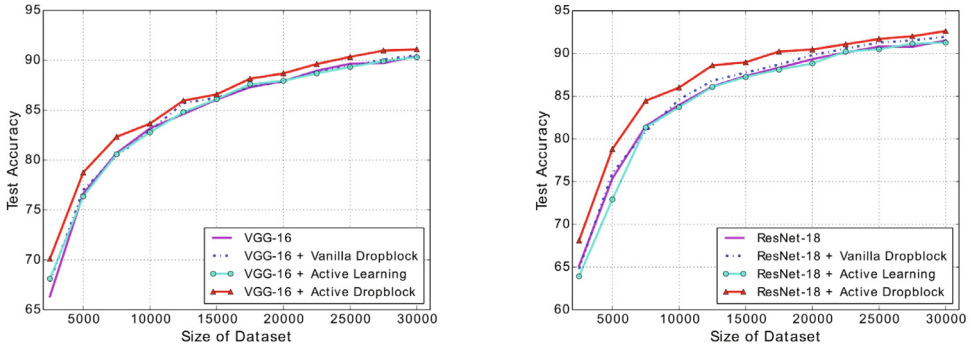
In addition to better model robustness and accuracy, our active dropout can exhibit higher data efficiency over learning tasks since it is able to augment features and exploit data information. We justifies this data efficiency with image classification and fine-grained visual classification.

4.4.1. Image Classification

We developed the experiments on image classification task with dataset of CIFAR-10. The investigated network architectures are VGG-16 and ResNet-18. To show data-efficiency, we reduce the size of images to 2500 randomly, and conduct network training with the obtained reduced dataset. The results are shown in Fig. 7. It is found that our active dropblock can yield significant accuracy improvement compared to active dropout, vanilla dropblock, and corresponding vanilla network backbone. Then, we increase the data size to further examine the consistency of this advantage and the results are included in Fig. 7. As expected, active dropblock consistently outperforms the others, while the advantage becomes smaller when data size is larger than 32500, which data size is already adequately large. This finding indicates that active dropout can augment data and improvement learning efficiency from limited data.

Table 8
Classification Accuracy (%) for CIFAR-10 on different layers of ResNet-18 combined with Vanilla Dropout and Active Dropout. *DO* is short for Dropout in the light of table width.

Model	ResNet-18 Without Adversarial Training								
Step	Regular	Layer1		Layer2		Layer3		Layer4	
		Vanilla *DO*	Active *DO*	Vanilla *DO*	Active *DO*	Vanilla *DO*	Active *DO*	Vanilla *DO*	Active *DO*
0	94.12	94.46	**94.49**	94.22	**95.01**	94.35	**94.81**	94.17	**94.65**
1	50.96	52.22	**57.14**	51.9	**58.11**	50.82	**57.06**	48.94	**56.58**
5	0.53	1.22	**2.38**	0.64	**2.74**	0.74	**1.62**	0.94	**1.51**
10	0.01	0.08	**0.22**	0.01	**0.32**	0.01	**0.07**	0.02	**0.04**
15	0	0.01	**0.08**	0	**0.15**	0	**0.03**	0	**0.01**
20	0	0	**0.03**	0	**0.08**	0	**0.01**	0	**0.01**
Model	ResNet-18 With Adversarial Training(Training step: 10)								
Step	Regular	Layer1		Layer2		Layer3		Layer4	
		Vanilla *DO*	Active *DO*	Vanilla *DO*	Active *DO*	Vanilla *DO*	Active *DO*	Vanilla *DO*	Active *DO*
0	90.52	**90.78**	90.51	**90.61**	90.44	90.7	**90.91**	90.54	**90.99**
1	80.09	80	**80.91**	79.81	**80.7**	80.8	**81.42**	80.34	**80.68**
5	35.88	35.7	**37.38**	36.12	**37.34**	37.6	**38.91**	37.09	**37.77**
10	28.73	28.44	**30.1**	28.59	**29.86**	30.09	**31.49**	29.52	**30.62**
15	26.45	25.94	**27.49**	25.81	**27.39**	27.74	**28.88**	26.9	**27.86**
20	25.37	24.8	**26.51**	24.72	**26.27**	26.61	**27.87**	25.81	**26.76**



(a) VGG-16 (b) ResNet-18

Fig. 7. Classification accuracy on CIFAR-10 with different size of training dataset and the whole testing dataset. (a) is trained by VGG-16 and (b) is trained by ResNet-18. The ActDB is applied on the posterior layers of these models.

4.4.2. Fine-grained visual classification

In addition to some kind of low resolution dataset, we also perform some experiments on some high resolution dataset. In this section we developed the experiments on fine-grained visual classification task with datasets of Stanford Cars and Aircrafts. The investigated network architectures are ResNet-18, ResNet-34, and DRN-C-26. We resize each dataset into 448 × 448 and the batch size of each experiment is set to 64. We use SGD as the training optimizer and use 0.001 to initialize the learning rate. Meanwhile, the learning rate is divided by 10 every 60 epochs and we use 300 epochs to carry out these experiments.

The comparison results are reported in Table 9. It shows that active dropout also obtain consistently higher performance than vanilla network backbones. These observations justify that the feature augmentation from active dropblock can improve training efficiency on large-scale dataset.

Furthermore, we provide some qualitative results to illustrate the feature maps from active dropblock equipped DRN-C-26. From Fig. 8, active dropblocks generate more spotty features but the activation areas better cover the entire objects. This implies that the active dropblock can enlarge the receptive field and render network detect more distinct features between objects, which can help to increase classification accuracy. Also, the spotty and random features show higher complexity of activation process, which can increase attack optimization cost.

5. Discussion and conclusions

Dropblock is an effective approach to improve the robustness of deep neural network. It proposes to drop features in a structured block. The feature extraction ability can be improved in this way and there are a lot of works have used it to improve their performance[42,43]. However, most of them are rely on some empirical

Table 9
Classification Accuracy (%) for Fine-Grained Visual Classification on different databases with input size of 448.

Task	Stanford Cars		Aircrafts	
Backbone	Regular	Active	Regular	Active
ResNet-18	84.72	**87.56**	81.49	**82.84**
ResNet-34	89.13	**89.64**	86.11	**86.23**
DRN-C-26	90.30	**90.74**	86.80	**87.91**

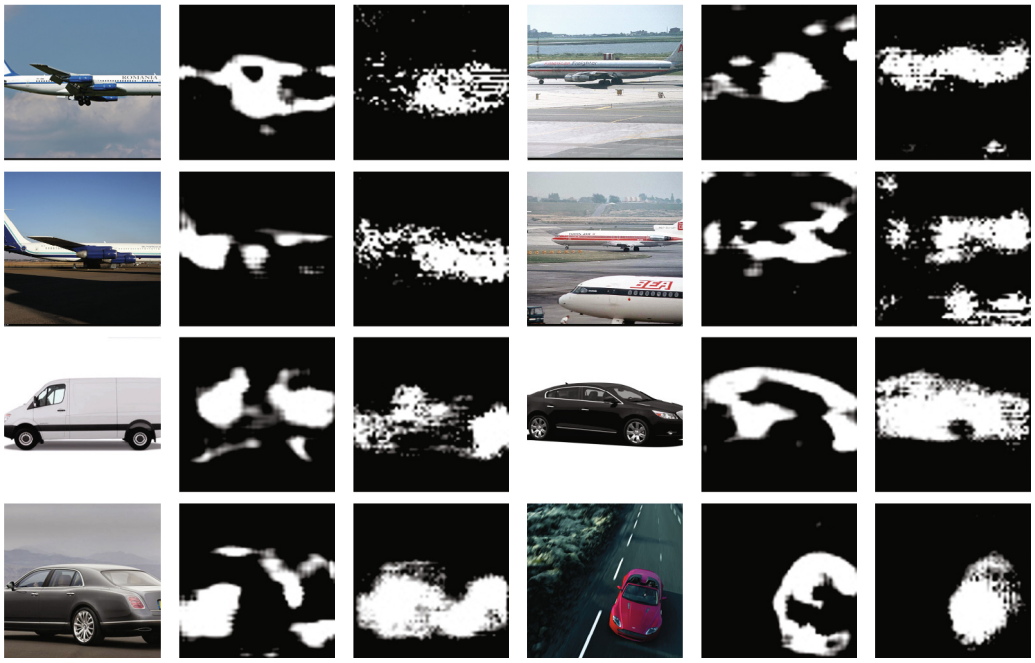


Fig. 8. Activation maps extracted from DRN-C-26 while we use test dataset to generate these activation maps. (a) and (d) are the training images from dataset. (b) and (e) are the activation maps based on the vanilla dropblock. (c) and (f) are the activation maps based on our method.

information to choose the hyperparameter like *block_size* or γ, which usually takes a lot of time on experiments. On the other aspect, data augmentation is a critical component of training deep neural network. One of the latest work on data augmentation[44] investigates how learned, specialized data augmentation policies improve generalization performance. However, this kind of methods usually focus on input level or low level of model, which cannot guarantee the stability of augmenting.

To resolve this dilemma, we use the active learning to enhance the dropblock and proposed the active dropblock. We force the selection of dropout selector to disturb the input vector to a great extent and keep the basic optimizing principle of deep neural network. By doing this way, the upper boundary of dataset can be increased and the model trained from this way can predict correctly some input data which may not be collected in the dataset. According to our experiments, our method can improve the data effectiveness when the dataset is fragmentary. When we use the whole dataset or a small-scale dataset to train the model, we can get a higher result than the original dropblock. Meanwhile, our method can improve the robustness of deep neural network and achieve better results compared to the state-of-the-art work.

To the best of our knowledge, this is the first work applying the active learning to the dropblock and focusing on the robustness of the deep neural network. This work can greatly improve the feature extraction ability of model and our method can apply to many other works. We believe that more research work in this direction are valuable, like improve the training effectiveness of model. Some of these idea will be considered in future work.

CRediT authorship contribution statement

Jie Yao: Conceptualization, Methodology, Software, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Visualization, Project administration. **Weiwei Xing:** Validation, Formal analysis, Supervision, Funding acquisition. **Dongdong Wang:** Methodology, Writing - original draft, Visualization. **Jintao Xing:** Resources, Visualization. **Liqiang Wang:** Validation, Formal analysis.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

Funding: This work was supported by the National Natural Science Foundation of China [grant number 61876018].

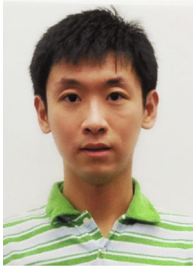
References

[1] N. Carlini, D. Wagner, Adversarial examples are not easily detected: Bypassing ten detection methods, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, 2017, pp. 3–14.
[2] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, Q. Gu, Improving adversarial robustness requires revisiting misclassified examples, in: International Conference on Learning Representations, 2020.
[3] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, A. Madry, Robustness may be at odds with accuracy, arXiv preprint arXiv:1805.12152.
[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[5] G. Ghiasi, T.-Y. Lin, Q.V. Le, Dropblock: A regularization method for convolutional networks, Adv. Neural Inform. Processing Syst. (2018) 10727–10737.
[6] A. Azulay, Y. Weiss, Why do deep convolutional networks generalize so poorly to small image transformations?, J Mach. Learn. Res. 20 (184) (2019) 1–25.
[7] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572.
[8] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, arXiv preprint arXiv:1611.01236.
[9] D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, arXiv preprint arXiv:1903.12261.
[10] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, Ensemble adversarial training: Attacks and defenses, arXiv preprint arXiv:1705.07204.
[11] T. Miyato, S.-I. Maeda, M. Koyama, K. Nakae, S. Ishii, Distributional smoothing with virtual adversarial training, arXiv preprint arXiv:1507.00677.

[12] A. Mustafa, S. Khan, M. Hayat, R. Goecke, J. Shen, L. Shao, Adversarial defense by restricting the hidden space of deep neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 3385–3394.
[13] B. Settles, Active learning literature survey, Tech. rep., University of Wisconsin-Madison Department of Computer Sciences (2009).
[14] S. Hanneke et al., Theory of disagreement-based active learning, Foundations and Trends, Mach. Learn. 7 (2–3) (2014) 131–309.
[15] S. Dasgupta, Two faces of active learning, Theor. Computer Sci. 412 (19) (2011) 1767–1781.
[16] J.T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, A. Agarwal, Deep batch active learning by diverse, uncertain gradient lower bounds, arXiv preprint arXiv:1906.03671.
[17] O. Sener, S. Savarese, Active learning for convolutional neural networks: A core-set approach, arXiv preprint arXiv:1708.00489.
[18] D. Gissin, S. Shalev-Shwartz, Discriminative active learning, arXiv preprint arXiv:1907.06347.
[19] M. Ducoffe, F. Precioso, Adversarial active learning for deep networks: a margin based approach, arXiv preprint arXiv:1802.09841.
[20] W.H. Beluch, T. Genewein, A. Nürnberger, J.M. Köhler, The power of ensembles for active learning in image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9368–9377.
[21] D. Wang, Y. Li, L. Wang, B. Gong, Neural networks are more productive teachers than human raters: Active mixup for data-efficient knowledge distillation from a blackbox model, arXiv preprint arXiv:2003.13960.
[22] A. Casanova, P.O. Pinheiro, N. Rostamzadeh, C.J. Pal, Reinforced active learning for image segmentation, arXiv preprint arXiv:2002.06583.
[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[24] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500.
[25] F. Yu, V. Koltun, T. Funkhouser, Dilated residual networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 472–480.
[26] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-first AAAI conference on artificial intelligence, 2017.
[27] D. Han, J. Kim, J. Kim, Deep pyramidal residual networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5927–5935.
[28] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.
[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
[30] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus, Regularization of neural networks using dropconnect, in: International conference on machine learning, 2013, pp. 1058–1066.
[31] G. Larsson, M. Maire, G. Shakhnarovich, Fractalnet: Ultra-deep neural networks without residuals, arXiv preprint arXiv:1605.07648.
[32] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.
[33] Y. Yamada, M. Iwamura, T. Akiba, K. Kise, Shakedrop regularization for deep residual learning, arXiv preprint arXiv:1802.02375.
[34] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, C. Bregler, Efficient object localization using convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 648–656.
[35] T. DeVries, G.W. Taylor, Improved regularization of convolutional neural networks with cutout, arXiv preprint arXiv:1708.04552.
[36] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3d object representations for fine-grained categorization, in: Proceedings of the IEEE international conference on computer vision workshops, 2013, pp. 554–561.
[37] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, A. Vedaldi, Fine-grained visual classification of aircraft, arXiv preprint arXiv:1306.5151.
[38] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572.
[39] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, arXiv preprint arXiv:1607.02533.
[40] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 9185–9193.
[41] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, arXiv preprint arXiv:1706.06083.
[42] K. Lee, S. Maji, A. Ravichandran, S. Soatto, Meta-learning with differentiable convex optimization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 10657–10665.
[43] G. Ghiasi, T.-Y. Lin, Q.V. Le, Nas-fpn: Learning scalable feature pyramid architecture for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7036–7045.
[44] B. Zoph, E.D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, Q.V. Le, Learning data augmentation strategies for object detection, arXiv preprint arXiv:1906.11172.

Jie Yao received the B.S. degree in Software Engineer from Beijing Jiaotong University, China, in 2016. During 2019–2020, he was a visiting student at University of Central Florida. Currently, he is a Ph.D. candidate of School of Software Engineering at Beijing Jiaotong University, China. His research interests include image processing, computer vision and deep model robustness.

Weiwei Xing received her B.S. degree in Computer Science and Technology and Ph.D. degree in Signal and Information Processing from Beijing Jiaotong University, in 2001 and 2006 respectively. During 2011–2012, she was a visiting scholar at University of Pennsylvania. Currently, she is an associate professor at School of Software Engineering, Beijing Jiaotong University. Her research interests mainly include intelligent information processing and artificial intelligence.



Dongdong Wang is a Ph.D. candidate in computer science at University of Central Florida. He earned his Master of Science in Environmental Science from Duke University in 2017. His research explores the approaches for knowledge distillation and deep neural network optimization and the application field is focused on computer vision. His work is developed upon numerical analysis and optimization. He has published several peer-reviewed papers in leading conferences such as CVPR and AAAI.

Jintao Xing received his B.S. degree in Software Engineering from Beijing Jiaotong University, China, in 2017. Currently he is a Ph.D. student at degree at School of Software Engineering, Beijing Jiaotong University. His main research interests include image processing, computer vision, transportation planning, and intelligent transportation systems.

Liqiang Wang is an associate professor in the Department of Computer Science at the University of Central Florida. He is the director of Big Data Lab. He was a faculty member (2006–2015) in the Department of Computer Science at the University of Wyoming. He received Ph.D. in Computer Science from Stony Brook University in 2006. He was a visiting Research Scientist in IBM T.J. Watson Research Center during 2012–2013. His research focuses on big data techniques, which include the following aspects: (1) improving the accuracy, security, privacy, and fairness of big data analytics; (2) optimizing performance, scalability and resilience of big data processing, especially on Cloud and GPU platforms; (3) using program analysis and deep learning techniques to detect and avoid programming errors, execution anomaly, as well as performance defects in big data programs. He received NSF CAREER Award in 2011 and Castagne Faculty Fellowship (2013-2015).