CamDrop: A New Explanation of Dropout and A Guided Regularization Method for Deep Neural Networks

Hongjun Wang
Sun Yat-sen University
wanghq8@mail2.sysu.edu.cn

Guangrun Wang*
Sun Yat-sen University
wanggrun@mail2.sysu.edu.cn

Guanbin Li
Sun Yat-sen University
liguanbin@mail.sysu.edu.cn

Liang Lin
Sun Yat-sen University
linliang@ieee.org

ABSTRACT

To force convolutional networks to explore more discriminative evidence throughout spatial regions, this paper presents a novel CamDrop to improve the conventional dropout in two aspects. First, by considering the intensity of class activation mapping (CAM) all around, CamDrop selectively abandons some specific spatial regions in predominating visual patterns at each iteration. In many classification tasks, CamDrop demonstrates its effectiveness and achieves considerable improvements on robust predictions for adversarial examples. Second, although dropout is a widely adopted technique that has been applied to regularize large models, the improvement in performance always attributes to better preventing DNN from overfitting. Here we give a new explanation of dropout from the perspective of optimization that it makes the upper bound of the magnitude of gradients much tighter, which leads to a more stable behavior of the gradients and effectively avoids neurons falling into the saturation region of the nonlinear activation, even when using high learning rates. Extensive experiments have been performed to prove the above two strengths of CamDrop.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Regularization**.

KEYWORDS

deep neural networks, regularization, dropout

ACM Reference Format:
Hongjun Wang, Guangrun Wang, Guanbin Li, and Liang Lin. 2019. CamDrop: A New Explanation of Dropout and A Guided Regularization Method for Deep Neural Networks. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357384.3357999

*Corresponding Author

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357999

1 INTRODUCTION

Recently, with massive data and increasing proliferation of machine learning, especially deep learning, remarkable performance has been achieved for solving a variety of visual recognition tasks[13–15]. While deep neural networks have brought superior performance, the lack of interpretability makes their predictions hard to be convinced, also leaving potential safety hazards. For example, an autonomous driving vision system can cause major accidents due to its poor generalization capabilities. From the detection of adversarial examples, it is crucial to building a more robust system to strengthen the representation power.

Actually, the underlying meaning of "robust" is that the representations extracted by DNN can represent more reasonable high-level semantics or detailed spatial information rather than only identifying the most discriminative part, and could be more transferable to a new domain. More specifically, we expect all contributing features for the target category in the image can be highlighted when applying existing interpretable visualization techniques, like class activation mapping (CAM)[48] and relevant works[27–29, 31]. This inspires us to design a mechanism that intentionally hides some specific spatial regions in dominant visual patterns at each iteration dependent on the CAM of inputs. To achieve this goal, we introduce a structured dropout *CamDrop*, which can be considered as a generalization of recent structured dropout formulations[9, 21, 38]. With the guidance of CAM, CamDrop can force the network to proactively explore other neglected parts autonomously instead of relying on external data.

After giving an intuitive explanation of the viability of this model, we focus our attention on other possibilities that can also make the CamDrop work. As a regularization method, dropout[35] is used as a practical tool to regularize deep neural networks in large vision models. The popular belief is that the improvement gains of dropout stem from preventing the models from overfitting during training or avoiding the so-called "co-adaptation" among the hidden nodes. However, it is strange that using dropout throughout training (including at very beginning) performs better than only applying at the end of training stage that is overfitting on the training set indeed. It also fails to explain why dropout even increases the correlation between nodes, as shown in [12].

Considering that dropout participating throughout the course of training is more robust, it is natural to doubt that such effects may benefit from better optimization. We find out that CamDrop makes the upper bound of the magnitude of gradients much tighter. This ensures that the magnitude of gradients is more controllable after

applying CamDrop and prevent neurons quickly falling into the saturation area of nonlinear activation even for the use of a larger range of learning rates. In fact, the degree of influence depends on how much such dropout variant decreases the L_1-norm of the partial derivative of the output probability vector with respect to the layer parameters. We provide both theoretical and experimental justification.

Our Contributions. (1) We propose a new type of dropout variant, called *CamDrop*, which forces models to seek multiple relevant parts for a specific category rather than the most discriminative one only. Our experiments demonstrate that the proposed CamDrop is superior to the dropout in different datasets in terms of both accuracy and visualization. Adding CamDrop to ResNet-50 architecture reduces image classification error rate from 23.41% to 22.93% on ImageNet and to ResNet-110 from 5.58% to 4.19% on CIFAR10.

(2) We find that dropout family makes the upper bound of the magnitude of gradients tighter, which benefits the optimization. This enables us to use a broader range of learning rates. In this regard, we give both theoretical and experimental validation.

(3) We also explore the feasibility of CamDrop for defending adversarial examples on ImageNet validation images and show that CamDrop reduces the attack success rate of PGD from 30.89% to 20.16%.

2 RELATED WORK

Visualization of CNNs To explicitly identify which parts convolutional neural networks play a role in classification prediction, many techniques[30, 32, 34, 45] have been proposed. They visualize the internal representation learned by deep models in an attempt to better understand their properties. One of the typical work is the Class Activation Mapping (CAM)[48], which can highlight the discriminative object parts detected by CNN.

Non-structured Dropout. Dropout[35] is an effective regularization to alleviate the overfitting of deep networks, which randomly sets input elements to zero in the training phase while making it inoperative during validation. Dropout is usually used for fully connected layers in neural networks. Most of the non-structured methods focus on the optimization based on Bayesian learning framework following its variational interpretation[6, 7, 17, 20, 25] or estimate hyperparameters from a prior specific distribution[18, 41, 46].

Structured Dropout. Recently, to better integrate dropout into convolution layers, there have been growing interest in finding structured dropout, such as DropPath[21], SpatialDropout[38], DropbBlock[9], etc[8, 11, 16, 37, 40, 44, 49]. Instead of attaching dropout to the internal features, there are also some data augmentation methods which drop a fixed-size block from the raw images, like Cutout[5], [33] and RandomErasing[47]. Our method is inspired by [9] and [48], and it is different from DropPath which zeroes an entire layer out of training or SpatialDropout, which drops an entire channel from a feature map. CamDrop discards the contiguous regions only on the selected regions and channels. Our experiments show that CamDrop is more effective than any other dropout variants.

3 METHODOLOGY

In this section, we first give a general formulation of the dropout family, and then naturally introduce our CamDrop in this formulation. Then, we elaborate on how CamDrop zeroes out units in feature maps under the guidance of the salient image regions generated by CAM. Finally, we also reveal the internal mechanisms by which this model works in terms of optimization.

3.1 Notations

For the sake of understanding, we first provide some definitions of notations and necessary introductions. A typical deep neural network (DNN) classifier with N layers can be generally expressed as a mapping function $f(X, y) : \mathbb{R}^{C\times H\times W} \rightarrow \mathbb{R}^K$, where X and y denote the input of the network and ground-truth label with K classes respectively. Suppose the input of the l-th layer of DNN is X^{l-1} and X^l being its output, every basic layer of a DNN can be constructed as $X^l = \sigma(\mathbf{W}^l X^{l-1} + \mathbf{b}^l)$ with the parameter matrix $\mathbf{W}^l$, the bias vector $\mathbf{b}^l$ and the compound operators $\sigma(\cdot)$ (i.e. Pooling, Norm and ReLU). Let $Z \in \mathbb{R}^{C'\times H'\times W'}$ be the output feature map of the penultimate layer. Then, global average pooling is performed to reduce the spatial dimensions and the resulting vector $z \in \mathbb{R}^{C'\times 1}$ is fed into a fully-connected layer with weight matrix $\mathbf{A} = \left\{\alpha^1 \cdots, \alpha^K\right\} \in \mathbb{R}^{C'\times K}$ and softmax layer to predict the probabilities $y'_{k'}$ for class k', which can be expressed as $y'_{k'} = P(y = k'|z) = \exp\left(S^{k'}\right) / \sum_{k=1}^{K} \exp\left(S^k\right)$ (we omit bias term for simplicity). The unnormalized score $S^{k'}$ for class k' is:

$$S^{k'} = z^T \alpha^{k'} = \sum_{c=1}^{C'} \mathbf{A}_{c,k'} \overbrace{\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W}}^{\text{global average pooling}} Z_{c,h,w} \tag{1}$$

The above training procedure intends to minimize the objective function $\mathcal{L}$ (usually the cross-entropy loss) on training data by stochastic gradient descent (SGD) to obtain the optimal parameters.

3.2 Dropout variants

A General Form. Let $\mathbf{M}^l \in \{0, 1\}^Q$ be a binary mask tensor applied for X^l along each dimension, where Q is the product of the size of all axes. The general form of dropout methods can then be denoted as Eqn.(2) following the most popular setting[35], which scales X^l by $1/\gamma$ when training and remains unchanged during validation.

$$\begin{aligned} \widetilde{X}^l &= \mathbf{M}^l \odot X^l / \gamma \\ \mathbf{M}^l_{c,h,w} &\sim \text{Ber}(\gamma), \qquad \gamma \in [0, 1] \end{aligned} \tag{2}$$

where γ is the retaining rate and $\odot$ is the element-wise product of tensors (also known as the Hadamard product). More specifically, the resulting masked outputs $\widetilde{X}^l$ can be written as:

$$\widetilde{X}^l_{c,h,w} = \frac{1}{\gamma} \left\{\mathbf{M}^l_{c,h,w} X^l_{c,h,w} | (c, h, w) = (1, 1, 1), \ldots, (C, H, W)\right\} \tag{3}$$

In this subsection, let $\{\cdot\}$ be a bernoulli sampler yielding all selected locations within the range of input. Many types of structured dropout methods mainly differ in the relation of selected index (Figure 1), discussed as follows.

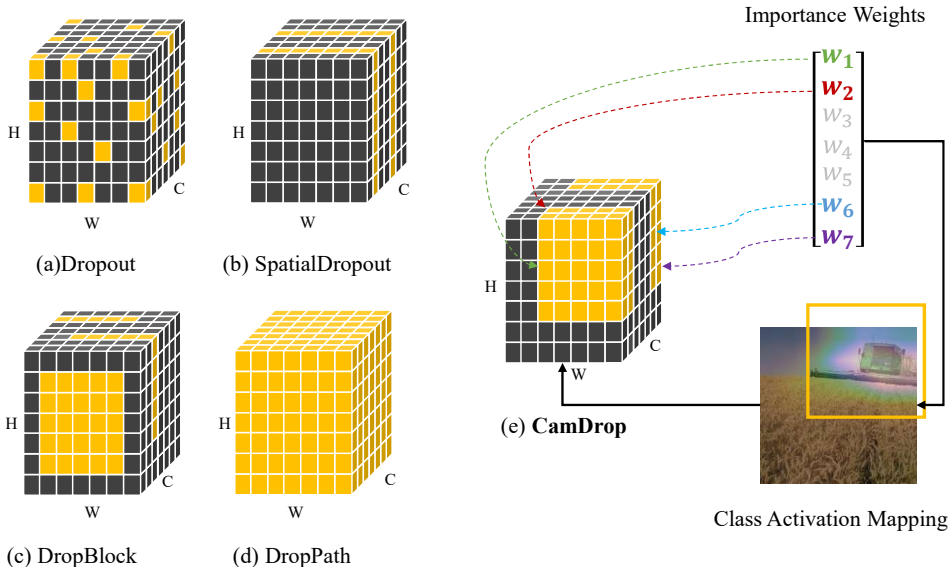


Figure 1: Several variants of structured dropout methods. Each subplot stands for a feature map tensor indexing by (C, H, W). The units in orange will be suppressed while the black will survive in this round. Our CamDrop is guided by the class activation mapping and its importance weights (bolded and colored) to selectively abandon the dominant regions in important channels.

- For **Dropout[35]**: $\mathbf{M}^l_{[\{C,H,W\}]} = 0$. This means that the units indexing by (c, h, w) are zeroed out individually.
- For **SpatialDropout[38]**: $\mathbf{M}^l_{[\{C\},:,:]} = 0$. This means that SpatialDropout drops several channels along the (H, W) axes together for each sample.
- For **DropbBlock[9]**: $\mathbf{M}^l_{[\{C\},\forall q \| q-\{H,W\}\|_1=r]} = 0$ with apothem r. This means that DropbBlock drops spatial continuous squares centered by $\{H, W\}$ across each feature channel together for each sample.
- For **DropPath[21]**: $\mathbf{M}^l_{[:,:,:]} = 0$. This means that DropPath drops the whole layer instead of a particular unit if such branch or input is not selected.

CamDrop. As for CamDrop, $\mathbf{M}^l_{[\{C\}^*,\forall q \| q-\{H,W\}^*\|_1=r]} = 0$, where $\{\cdot\}^*$ is denoted as a guided sampler. The main difference of CamDrop from the above four methods is that it connects the dependency of feature assignments to the input covariates rather than dropping out units randomly. The panorama of CamDrop can be described as following:

$$\mathbf{M}^l = \mathcal{F}(\mathcal{M}, r) \quad \mathcal{M} = \varphi(\boldsymbol{\psi}, \mathcal{T}(X^l, y)) \quad \boldsymbol{\psi} \sim \text{Ber}(\gamma) \tag{4}$$

with $\varphi(\cdot,\cdot)$ mapping ψ, X^l and y to a boolean 3D-tensor $\mathcal{M}$ and $\mathcal{F}$ expanding the specific units of resulting tensor by the size of r. In the above four random dropout methods, $\mathcal{T}$ and $\varphi(\cdot,\cdot)$ degenerate to zero mapping and identity mapping respectively. Next, we'll give more details about the affine function $\mathcal{T}$ and $\varphi(\cdot,\cdot)$.

3.3 Cam-Guided Mechanism

Intuitively, it is reasonable to expect a robust DNN to find as much correlated evidence on corresponding image regions as it can. To achieve this goal, DNN should be discouraged from 'outsmarting' itself by cutting corners, as clinging to the most obvious point may be one-sided or false positive. Therefore, we expect (1) more spatial elements would be activated to determine the final decision rather than the current dominant only. (2) more visual patterns will have a positive influence on the particular category. In fact, global average pooling acts as a bridge between channels in the last feature map and the final categories, for which the weight vector $\alpha^{k'} \in \mathbb{R}^{C'\times 1}$ indicates the importance of each channel for a specific class k' of y and the class activation maps $J^{k'} \in \mathbb{R}^{1\times H'\times W'}$ is the weighted sum of these channel patterns at different spatial locations:

$$J^{k'} = \sum_{c=1}^{C'} \alpha_c^{k'} Z_{c,:,:} \tag{5}$$

In other words, $J^{k'}$ can be considered as a set with $H' \times W'$ elements $\{j_{1,1}, \ldots, j_{H'\times W'}\}$, each of which implies the significance of units at spatial grid (h, w). Let T_n^s be the set of the n most important pixels up to the moment and its initial state $T_0 = J^{k'}$. Then T_n^s can be defined recurrently as $T_{n+1}^s = T_n^s \setminus \{j \in \mathbb{R} : -\max\{-T_n^s\}\}$, where $\setminus$ is used to denote the difference of set. Thus, we can create the *spatial-wise* binary mask $M^{(1)}$ as:

$$M^{(1)}_{h,w} = \begin{cases} 0, & J^{k'}_{h,w} > \inf\{T_n^s\} \\ 1, & J^{k'}_{h,w} < \inf\{T_n^s\} \end{cases} \tag{6}$$

Since the number of channels are doubled layer by layer, when it comes to the shallow layers, we compress the length of α to match and replace Eqn.(6) with Eqn.(7):

$$J^{k'} = \mathcal{T}(X^l, y) = \sum_{c=1}^{C''} \Big(\sum_{j=1}^{\left\lfloor \frac{C'}{C''} \right\rfloor} \alpha_j^{k'} \Big) X^l_{c,:,:} \tag{7}$$

where C'' stands for the number of channels in the shallow layers. Because the class activation map is always upsampled to the size of

the input image to visualize the regions of interest of a particular category, we adaptively select n for different layers according to their resolution by $n = (1 - \frac{1}{r^2}) \times H'W'$.

Although $J^{k'}$ can identify regions where DNN is considered important, the weighted sum loses the intensity information of every present visual pattern. For example, suppose the highest response region of $J^{k'}$ centers around (h, w), but meanwhile it may have another dominant region distant from (h, w) in channel c', which not displayed in $J^{k'}$ for being counteracted by another high activation with negative influence at the same location in channel c''. This shows that α is also important for the final decision. Analogously, we can give the *depth-wise* binary mask $M^{(2)}$ as:

$$M_c^{(2)} = \begin{cases} 0, & \alpha_c^{k'} > \inf\left\{T_{n'}^d\right\} \\ 1, & \alpha_c^{k'} < \inf\left\{T_{n'}^d\right\} \end{cases} \tag{8}$$

where n' is a hyperparameter and $\inf\{\cdot\}$ is the infimum of a set. The definition of $T_{n'}^d$ is similar to T_n^s but performed on $\alpha^{k'}$. Combined with the valid seed region $M^{(3)}$ whose expanded mask with the size of s fully contained inside the feature map, the tensor of sampled center pixels can be given as:

$$\mathcal{M} = \neg\boldsymbol{\psi} \wedge (\bigwedge_i \neg M^{(i)}) \tag{9}$$

where $\neg$ and $\wedge$ ($\bigwedge$) are "logical not" and "logical and" (logical conjunction). Finally, we sweep over $\mathcal{M}$ and set all pixels as zero if and only if the pixels belong to $\{\forall q \mid \|q - u\|_1 = r\}$ where u is the zero entries in $\mathcal{M}$, and then normalize the mask to get the final mask tensor $\mathbf{M}^l$:

$$\mathbf{M}^l = \frac{CHW}{\sum_{c=1,h=1,w=1}^{C,H,W} \mathcal{M}_{c,h,w}} \tag{10}$$

3.4 CamDrop for Optimization

From the above analysis, we have already given an intuitive description that CamDrop brings more opportunities for those disadvantaged areas in the current iteration. In this section, we attempt to provide a new perspective for CamDrop from optimization. The new understanding suggests that CamDrop increase the amount of gradient information flowing through layers since it makes every update of parameters smoother.

In retrospect, SGD decreases parameters in the direction of the negative gradient of loss with a small step size. At the t-th update iteration, the weights and biases in the l-th layer will be updated as:

$$\begin{aligned} \mathbf{W}_t^l &:= \mathbf{W}_{t-1}^l - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{t-1}^l} \\ \mathbf{b}_t^l &:= \mathbf{b}_{t-1}^l - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}_{t-1}^l} \end{aligned} \tag{11}$$

The derivative terms in Eqn.(11) (disregarding η) can be factorized as:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{t-1}^l} &= (\mathbf{W}^{l+1} \odot \mathbf{M})^T \frac{\partial \mathcal{L}}{\partial \boldsymbol{S}} \odot \sigma'\left(\mathbf{W}^l X^{l-1} + \mathbf{b}^l\right) {X^{l-1}}^T \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}_{t-1}^l} &= (\mathbf{W}^{l+1} \odot \mathbf{M})^T \frac{\partial \mathcal{L}}{\partial \boldsymbol{S}} \odot \sigma'\left(\mathbf{W}^l X^{l-1} + \mathbf{b}^l\right) \mathbf{1}^T \end{aligned} \tag{12}$$

where $\partial\mathcal{L}/\partial\boldsymbol{S} = \text{softmax}(\boldsymbol{S}) - \boldsymbol{y}$. Here we see the gradients after the traditional dropout operation flow through the neurons that are not killed off during the forward pass and the remaining are unchanged. The number of dropout neurons determines how far the direction of descent will be deviated from the true, which is the primary reason for bringing predictive uncertainy in its output. But on the other hand, it coincidentally prevents units from falling into saturation areas when $\mathbf{M}$ decreases $\partial\mathcal{L}/\partial\boldsymbol{b}$. For the ease of deduction, we simplify Eqn.(12) according to the chain rules as:

$$\frac{\partial \mathcal{L}}{\partial b_{t-1,i}^l} = \frac{\partial \mathcal{L}}{\partial \mathrm{S}} \frac{\partial \mathrm{S}}{\partial b_{t-1,i}^l} \tag{13}$$

Actually, the upper bounds of the loss with respect to the layer biases can be given by the Hölder inequality:

$$\left|\frac{\partial \mathcal{L}}{\partial b_{t-1,i}^l}\right| \le \max_{k'} \left|\frac{\partial \mathcal{L}}{\partial S_{k'}}\right| \left\|\frac{\partial \mathrm{S}}{\partial b_{t-1,i}^l}\right\|_1 \tag{14}$$

We can further expand the term $\left|\frac{\partial \mathcal{L}}{\partial y_{k'}}\right|$ as:

Since the absolute value of every element in $\partial\mathcal{L}/\partial\boldsymbol{S} = \text{softmax}(\boldsymbol{S}) - \boldsymbol{y} = \boldsymbol{y'} - \boldsymbol{y}$ cannot exceed 1 for any $y_{k'}, k' \in \{1 \cdots K\}$, the inequality in Eqn.(14) can be reduced to:

$$\left|\frac{\partial \mathcal{L}}{\partial b_{t-1,i}}\right| \le \left\|\frac{\partial \mathrm{S}}{\partial b_{t-1,i}}\right\|_1 \tag{15}$$

This is an upper bound of the update without dropout techniques (without $\mathbf{M}$ in Eqn.(12)). And based on the above filtering rules in Eqn.(6) and Eqn.(8), CamDrop always masks out the several notable neurons after rectified linear unit, which translate the bounds to a more favorable worst-case bound on the landscape with respect to layer biases:

$$\left|\frac{\partial \mathcal{L}}{\partial b_{t-1,i}^l}\right| \le \left\|\frac{\partial \mathrm{S}}{\partial b_{t-1,i}^l}\right\|_1^{(c)} \le \left\|\frac{\partial \mathrm{S}}{\partial b_{t-1,i}^l}\right\|_1^{(t)} \le \left\|\frac{\partial \mathrm{S}}{\partial b_{t-1,i}^l}\right\|_1 \tag{16}$$

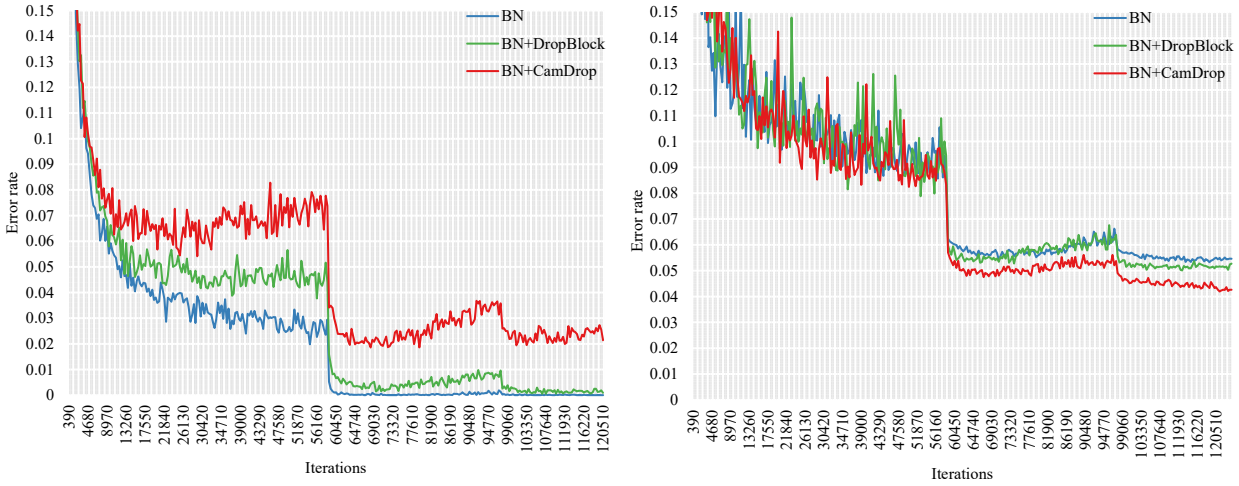
where $\|\cdot\|^{(c)}$ and $\|\cdot\|^{(t)}$ stand for the L_1-norm of derivative with Cam/traditional dropout mask respectively. Eqn.(16) reveals that CamDrop further gives a tighter upper bound of the update of biases $\partial\mathcal{L}/\partial\boldsymbol{b}$ at each iteration, which smoothes the updating procedure of biases term and would benefit more from using high learning rate. The upper bounds of layer weights $\mathbf{W}$ can be similarly derived as in Eqn.(13)- Eqn.(16). These help DNN avoid learning large negative biases term of parameters even in the early phases of training, which is different from traditional ideas that dropout is just a tool intended for solving overfitting.

4 EXPERIMENTS

In the following sections, we investigate the effectiveness of CamDrop mainly for image classification and verify the correctness of our theorem in Sec. 3.4. In addition to performance gains, we also demonstrate the robustness of the proposed CamDrop to resisting against adversarial examples.

4.1 Classification on CIFAR10

As it is extremely time-consuming to train the ImageNet[4] from scratch, we apply CamDrop to ResNet-110[13] with extensive experiments on CIFAR10[19]. CIFAR10 dataset consists of 60k RGB



(a) train error. (b) val error.

Figure 2: Comparison of the validation error curves on CIFAR10. We show the error rate *vs.* the numbers of training iterations. The baseline model is ResNet-110. For better visualization, we only select the curves of baseline and the best method to compare. Our CamDrop achieves much lower top-1 validation error than both baseline and model with DropBlock.

images across ten classes. The following experiments are trained on the training set with 50k images and evaluated on the testing set with 10k images. We fix $\gamma = 0.9$ and $r = 8$ since the size of last feature maps is 8. The learning rate is decayed by the factor of 0.1 at 150, 250, and 350. Other settings are the same as in [42]. Figure 2 shows the error curves and Table 1 shows the final results.

Model	Top-1 Error (%)
ResNet-110	5.58
ResNet-110+dropout[35]	5.47
ResNet-110+Droppath[21]	5.39
ResNet-110+SpatialDropout[38]	5.28
ResNet-110+DropBlock[9]	5.23
ResNet-110+CamDrop	**4.19**

Table 1: Comparison of error rates (%) of ResNet-110 in the ImageNet validation set. The error curves are in Figure 2. We report the median error rate of the best 5 results.

Figure 2 demonstrates that CamDrop has higher training error (left) but lower validation error (right) than DropBlock, indicating that CamDrop makes DNN more robust. To further illustrate the effectiveness of CamDrop, we perform more ablation studies to investigate different components of CamDrop.

The effectiveness of CAM. An evaluation of our proposed CamDrop is shown in Table 2. With a regular setting, ResNet-110 trained with dropout guided by CAM enjoys performance gains when compared to simply average maps across (C, H, W) dimensions of feature maps. Actually, DropBlock is comparable to the method adding simply average maps, which indicates that not all guided modes are effective for dropout. We also show the class activations of original ResNet-50 and the ones trained with DropBlock/CamDrop on ImageNet in Figure 3. The results demonstrate that CamDrop can catch the more semantic meaning of images than any others.

	Top-1 Error (%)
Avg guided	5.07
CAM guided	**4.19**

Table 2: Comparison of two mechanisms on CIFAR10 classification dataset, better guidance brings expected gains.

Across all channels vs. Dominant channels only To verify the necessity of the guidance of important weights, we make a comparison with abandoning dominant regions of CAM across all channels or just performing on the dominant channels. Table 3 shows that CamDrop guided by both CAM and its important weights works better, which stresses the differences in visual patterns.

	Top-1 Error (%)
All	4.51
Dominants	**4.19**

Table 3: Across all channels *vs.* dominant channels only. Importance weights improve results as they catch the position of every visual pattern.

The proportion of dominant visual patterns. Although the number of dropout points along spatial dimensions can be adaptively calculated, it remains to be determined how many visual patterns should take into consideration as discussed in Sec. 3.3. Thus far all reported results are trained with $n' = \frac{C}{8}$. As shown in Table 4, CamDrop performs reasonably better than DropBlock

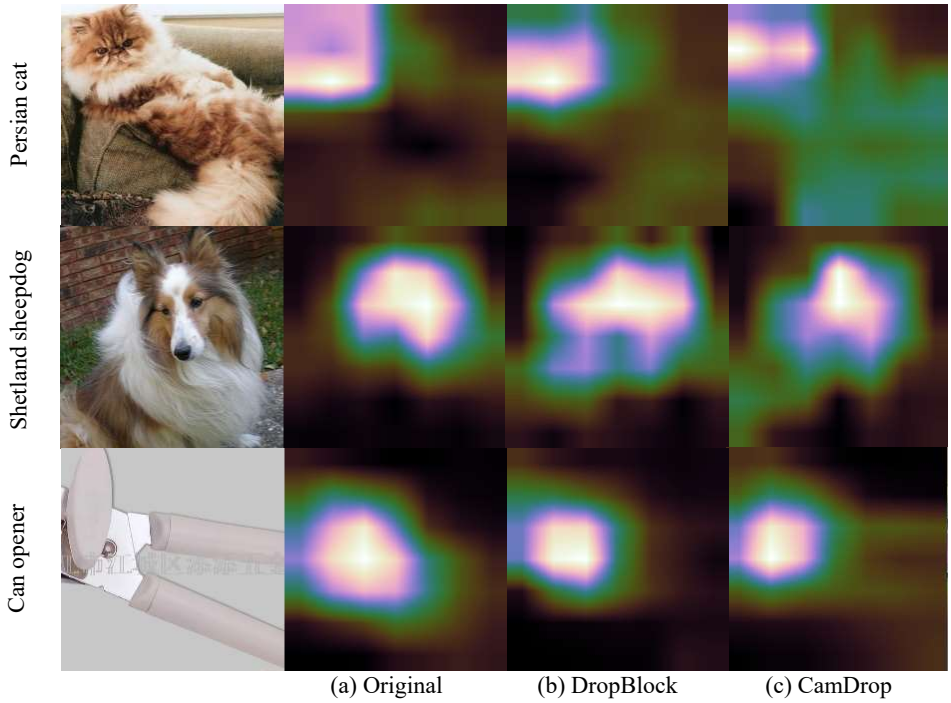


Figure 3: Visualization of the class activation map of ResNet-50 model trained with CamDrop/DropBlock and without any dropout method on ImageNet.

for all values of n' we studied. Actually, CamDrop is equivalent to DropBlock when n and n' are equal to their spatial resolution and the number of channels. Note that the accuracy decreases with the increment of proportion on the whole because the meaning of 'dominant' disappears in the extreme case of $n' = C$. This indirectly implies the effectiveness of selecting the most dominant visual patterns of CAM.

	Top-1 Error (%)
$n' = C/16$	4.39
$n' = C/8$	**4.19**
$n' = C/4$	4.75
$n' = C/2$	4.78

Table 4: Comparison of different number of dominant visual patterns. The best accuracy is achieved by using $n' = C/8$.

More than overfitting. As a warming up of the next section, we set the learning rate 20 times larger than the default at every stage. As a result, using CamDrop can still achieve considerable improvements (1st/2nd rows) without a carefully designed schedule. More *importantly*, when inversing the linear scheme in [49] by gradually increasing γ over time from the target value to 1, we find that the validation errors of models trained with CamDrop are close even though the inversed one is overfitting at the final phase (2nd/3rd rows, Table 5). But in sharp contrast, there is a large gap between the two overfittings with and without CamDrop (1st/3rd rows). All of these suggest that overfitting reduction is *not* the main reason for the accuracy improvements of CamDrop.

Model	Val Error (%)	Train Error (%)
ResNet-110, $lr = 2.0$	7.16	2.22e-4
ResNet-110, $lr = 2.0$ with CamDrop	5.94	8.53e-2
ResNet-110, $lr = 2.0$ with CamDrop, inversely	6.29	1.66e-4

Table 5: Comparison of the top-1 error on both training and validation set of CIFAR10 at high learning rate. Note that the model trained with CamDrop performs better than the baseline model even if it is overfitting as well.

4.2 Analysis on MNIST

To verify our theorem in Sec. 3.4, we investigate the connection among the learning rate lr, the retaining rate γ and accuracy on MNIST. MNIST[22] is a collection of handwritten digits with a training set of 60k images and a test set of 10k image. Specifically, we consider training a ResNet-8 architecture on MNIST by sweeping over several learning rates and remaining rates, and visualize all 64 results in Figure 4 and have several observations.

First, as shown in Figure 4(b), it is clear to see that ResNet-8 can be easily trained with and without CamDrop when $lr \leq 1.0$, but none of them can give a tolerable result when $lr \geq 10.0$ since the update of parameters in Eqn.11 acutely jitters at every iteration.

Second, when compared with ResNet-8 trained with and without CamDrop ($\gamma = 1$), Figure4(a) shows a significant difference in such performance change between $\gamma = 1$ and $\gamma = \{0.9, 0.8, 0.7, 0.6, 0.5\}$, especially when $3.0 \leq lr \leq 10.0$. As expected, the experimental phenomena is in line with our theoretical analyses in Sec. 3.4.

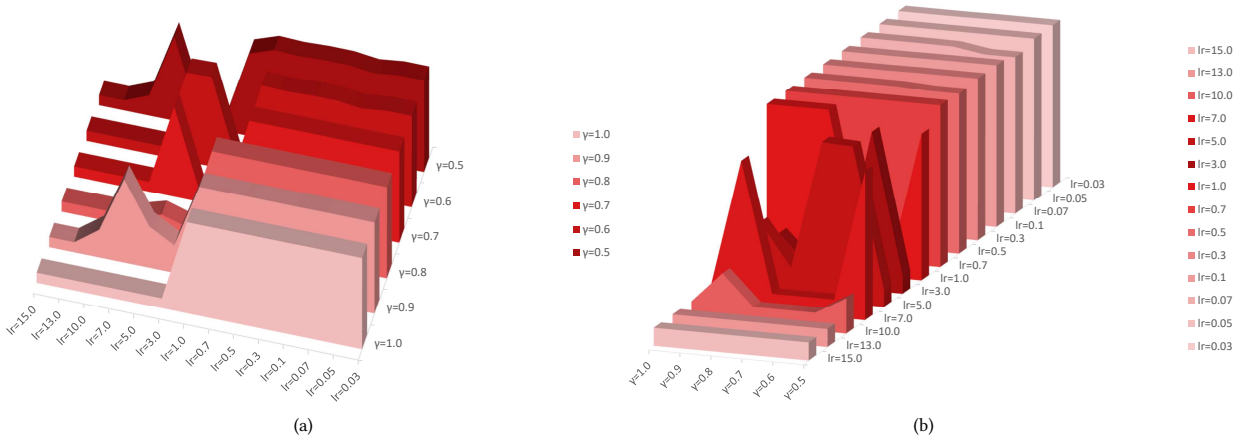


Figure 4: To clearly show the relationships among the learning rate (lr), the remaining rate (γ) and top1-accuracy (z-axis), we set (a) the learning rate and (b) the remaining rate as the major axis by turns. The remaining rate ranges from 0.5 to 1.0, while the learning rate covers several values between 15.0 and 0.03. The z-axis is used to plot the value of accuracy. The results show that CamDrop saves some cases which cannot be trained at high learning.

Moreover, the front rows in Figure4(b) demonstrate that the side effect of high learning rate cannot be handled by a lower remaining rate. The update of gradient and dropout are contested with each other in optimizing. On the one hand, dropout decreases L_1-norm of the gradient to smooth the updating procedure by suppressing units. On the other hand, the direction of the gradient is directly affected by every anticipated unit. However, like generative adversarial networks (GAN)[10], it is *not* a zero-sum game based on the experience and suggested that dropout with more effective guidance will further improve results.

4.3 ImageNet Classification and Adversarial Defense

The ILSVRC 2012 classification dataset[4] contains ~1.28M training images and 50k validation images labeled with 1,000 categories. Following the common practice[15, 36], we perform horizontal flip, scale, and aspect ratio augmentation for training images and apply a single center crop of 224×224 pixels during evaluation. We refer to [42] for other implementation details. We evaluate the top-1 and top-5 classification error on the validation set.

Implementation Details. We adopt ResNet-50 as implemented by [42] with default hyperparameters settings, which is a strong baseline to compare with. The implementation of DropBlock is based on the official release code. Note that different from [9], we curtail the number of training epochs from 270 to conventional 105 and perform all experiments with a regular batch size of 256 on 4 GPUs rather than 1024 on TPUs, which is more friendly to those lack of resources. The learning rate is decayed by the factor of 0.1 at 50, 80, 100, 105 epochs. We report the median error rate of the final 5 epochs in Table 6.

Comparison with DropBlock. Results in Table 6 show the effectiveness of CamDrop. Actually, we also see that the error rate of ResNet-50 with/without DropBlock is roughly the same after reducing the epochs of training while our CamDrop can adapt to such settings. This suggests that CamDrop may perform much better under the same configuration as DropBlock claimed in [9].

Model	Top-1 Error (%)	Top-5 Error (%)
ResNet-50	23.61	6.93
ResNet-50+dropout[35]	23.63	6.89
ResNet-50+Droppath[21]	23.57	6.85
ResNet-50+SpatialDropout[38]	23.51	6.83
ResNet-50+DropBlock[9]	23.41	6.82
ResNet-50+CamDrop	**22.93**	**6.67**

Table 6: Comparisons of top-1 and top-5 error rate on the validation set of ImageNet by using ResNet50. We apply CamDrop and DropBlock after both convolution layers and skip connections with $r = 7$ and $\gamma = 0.9$. We report average over last 5 runs.

Defense against Adversarial Examples. To verify the robustness of DNN trained with CamDrop, we choose the Projected Gradient Descent (PGD)[26] as our white-box attacker, which projects the perturbation on an L_p-ball of specified radius and clips the values of the adversarial sample into the permitted data range. We perform experiments with fixed L_∞ norm of 5 and use the ResNet-50 as our baseline. The remaining hyper-parameters of the PGD attacker are: the iteration of attack= 2, the step size of attack= 1. Since it is less meaningful to apply untargeted attacks on ImageNet for its numerous closely related classes, all accuracies we report on ImageNet are for targeted attacks. As [1, 43] recommend, we construct targeted adversarial examples with target classes selected uniformly at random. A targeted attack is considered successful if the image is classified to the target label. We present both the error rate (%) of our model and the attack success rate (%) of attack on ImageNet also in Tables 7. Moreover, results in Tables 7 show that using CamDrop can obtain more robust predictions for adversarial examples.

Model	Error Rate (%)	Success Rate (%)
PGD	82.21	30.89
PGD/CamDrop	**65.73**	**20.16**

Table 7: The model of target networks is Resnet-50. We provide both the error rate (%) of our model and the attack success rate (%) of adversarial examples by PGD attack.

5 DISCUSSION

We present a novel dropout method called CamDrop to improve the robustness of DNN models by masking the dominant regions with the guidance of class activation mapping. Our method performs well on several classification datasets and has a positive influence on defending against adversarial examples. Moreover, we give a new explanation of dropout from the view of optimization, showing that dropout techniques actually make the upper bound of the magnitude of gradients much tighter to a certain extent.

One potential issue with hiding spatial regions dependent on the CAM is that the input data for the algorithm must be image-like. Otherwise there is no CAM for them. Thus, CamDrop cannot be *directly* applied to other tasks (e.g., text, audio). However, lots of data with non-Euclidean structure (such as social network and knowledge management) can be converted into graphs with corresponding relationships between vertices and edges. Therefore, graph convolutional network (GCN) [2, 3] and its variants[23, 24, 39] can be used to establish graph and then extract spatial features to solve the above problem.

6 ACKNOWLEDGEMENTS

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2016YFB1001004, in part by National Natural Science Foundation of China (NSFC) under Grant No. 61836012, and in part by the Natural Science Foundation of Guangdong Province under Grant No. 2017A030312006.

REFERENCES

[1] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420* (2018).
[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).
[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.
[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
[5] Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (2017).
[6] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. 1050–1059.
[7] Yarin Gal, Jiri Hron, and Alex Kendall. 2017. Concrete dropout. In *Advances in Neural Information Processing Systems*. 3581–3590.
[8] Xavier Gastaldi. 2017. Shake-shake regularization. *arXiv preprint arXiv:1705.07485* (2017).
[9] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. 2018. DropBlock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*. 10727–10737.
[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
[11] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389* (2013).
[12] Sangchul Hahn and Heeyoul Choi. 2018. Understanding Dropout as an Optimization Trick. arXiv:arXiv:1806.09783
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[14] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.
[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.
[16] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. 2016. Deep networks with stochastic depth. In *European conference on computer vision*. Springer, 646–661.
[17] Durk P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*. 2575–2583.
[18] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in neural information processing systems*. 971–980.
[19] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Citeseer.
[20] John Lambert, Ozan Sener, and Silvio Savarese. 2018. Deep learning under privileged information using heteroscedastic dropout. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8886–8895.
[21] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648* (2016).
[22] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[23] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive graph convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[24] Renjie Liao, Marc Brockschmidt, Daniel Tarlow, Alexander L Gaunt, Raquel Urtasun, and Richard Zemel. 2018. Graph partition neural networks for semi-supervised classification. *arXiv preprint arXiv:1803.06272* (2018).
[25] Yuhang Liu, Wenyong Dong, Lei Zhang, Dong Gong, and Qinfeng Shi. 2019. Variational Bayesian Dropout With a Hierarchical Prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7124–7133.
[26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
[27] Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. 2019. CNN fixations: an unraveling approach to visualize the discriminative image regions. *IEEE Transactions on Image Processing* 28, 5 (2019), 2116–2125.
[28] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. 2015. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 685–694.
[29] Pedro O Pinheiro and Ronan Collobert. 2015. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1713–1721.
[30] Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *Advances in neural information processing systems*. 2953–2961.
[31] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*. 618–626.
[32] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).
[33] Krishna Kumar Singh and Yong Jae Lee. 2017. 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)*. IEEE, 3544–3553.
[34] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).
[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015.

Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[37] Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195* (2019).

[38] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 648–656.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[40] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International conference on machine learning*. 1058–1066.

[41] Haotian Wang, Wenjing Yang, Zhenyu Zhao, Tingjin Luo, Ji Wang, and Yuhua Tang. 2019. Rademacher dropout: An adaptive dropout for deep neural network via optimizing generalization gap. *Neurocomputing* 357 (2019), 177–187.

[42] Yuxin Wu et al. 2016. Tensorpack. https://github.com/tensorpack/.

[43] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. 2018. Feature denoising for improving adversarial robustness. *arXiv preprint arXiv:1812.03411* (2018).

[44] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. 2018. ShakeDrop Regularization for Deep Residual Learning. *arXiv preprint arXiv:1802.02375* (2018).

[45] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

[46] Ke Zhai and Huan Wang. 2018. Adaptive dropout with rademacher complexity regularization. (2018).

[47] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2017. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896* (2017).

[48] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2921–2929.

[49] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8697–8710.