Beyond Dropout: Feature Map Distortion to Regularize Deep Neural Networks

Yehui Tang,[1*] **Yunhe Wang,**[2] **Yixing Xu,**[2] **Boxin Shi,**[4,5] **Chao Xu,**[1] **Chunjing Xu,**[2] **Chang Xu**[3]
[1]Key Lab of Machine Perception (MOE), CMIC, School of EECS, Peking University, China
[2]Huawei Noahs Ark Lab, [3]School of Computer Science, Faculty of Engineering, The University of Sydney, Australia
[4]National Engineering Laboratory for Video Technology, Peking University [5]Peng Cheng Laboratory
{yhtang, shiboxin}@pku.edu.cn, {yunhe.wang, xuyixing, xuchunjing}@huawei.com
chaoxu@cis.pku.edu.cn, c.xu@sydney.edu.au

Abstract

Deep neural networks often consist of a great number of trainable parameters for extracting powerful features from given datasets. One one hand, massive trainable parameters significantly enhance the performance of these deep networks. One the other hand, they bring the problem of over-fitting. To this end, dropout based methods disable some elements in the output feature maps during the training phase for reducing the co-adaptation of neurons. Although the generalization ability of the resulting models can be enhanced by these approaches, the conventional binary dropout is not the optimal solution. Therefore, we investigate the empirical Rademacher complexity related to intermediate layers of deep neural networks and propose a feature distortion method for addressing the aforementioned problem. In the training period, randomly selected elements in the feature maps will be replaced with specific values by exploiting the generalization error bound. The superiority of the proposed feature map distortion for producing deep neural network with higher testing performance is analyzed and demonstrated on several benchmark image datasets.

Introduction

The superiority of deep neural networks, especially convolutional neural networks (CNNs) has been well demonstrated in a large variety of computer vision tasks including image recognition (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016a; Wang et al. 2018a), object detection (Ren et al. 2015; Redmon et al. 2016), video analysis (Feichtenhofer, Pinz, and Zisserman 2016), Natural Language Processing (Wang, Li, and Smola 2019) *etc*. Actually, the huge success of deep CNNs should be attributed to the larger number of trainable parameters and available annotation data, *e.g.* the ImageNet (Deng et al. 2009) dataset with over 1 million images from 1000 different categories.

Since deep networks are often over parameterized for achieving higher performance on the training set, an important problem is to avoid over-fitting, *i.e.* the excellent performance achieved on the train set is expected to be repeated on the test set (Hinton et al. 2012; Wang et al. 2018b). In other words, the empirical risk should be closed to the expected risk. To this end, (Hinton et al. 2012) first proposed the conventional binary dropout approach, which reduces the co-adaptation of neurons by stochastically dropping part of them in the training phase. This operation can be either regarded as a model ensemble technique or a data augmentation method, which significantly enhances the performance of the resulting network on the test set.

To improve the performance of dropout implemented on deep neural networks, (Ba and Frey 2013) adaptively adjusted the dropout probability of each neuron by interleaving a binary belief network into the neural networks. Gaussian Dropout (Srivastava et al. 2014) multiplying the outputs of the neurons by Gaussian random noise is equal to the conventional binary dropout. It was further analyzed from the perspective of Bayesian regularization and the dropout probability can be optimized automatically (Kingma, Salimans, and Welling 2015). Instead of disabling the activation, DropConnect (Wan et al. 2013) randomly set a subset of network weights to zero. (Wan et al. 2013) derived a bound on the generalization performance for Dropout and DropConnect. (Zhai and Wang 2018) connected the bound with drop probability and optimized the dropout probability together with network parameters during the training. Focusing on the convolutional neural networks, (Ghiasi, Lin, and Le 2018) proposed to drop contiguous regions of a feature map to obstruct the information flow more radically.

Existing variants of dropout have made tremendous efforts for minimizing the gap between the expected risk and the empirical risk, but they all follow the general idea of disabling parts of the output of an arbitrary layer in the neural network. The essence of the success is to randomly obscure part of semantic information extracted by the deep neural network and avoid the massive parameters to over-fit the training set. Setting a certain number of the elements in the feature map to zero is a straightforward way to disturb the information propagation across layers in the neural network, but it is by no means the only way to accomplish this goal. Most importantly, such sort of hand-crafted operations are hardly to be the optimal ones in most cases.

In this work, we propose a novel approach for enhancing the generalization ability of deep neural networks by inves-

[*]This work was done while visiting Huawei Noahs Ark Lab.

tigating the distortion on the feature maps. The generalization error bound of the given deep neural network is established in terms of the Rademacher complexity of its intermediate layers. Distortion is introduced onto the feature maps to decrease the associated Rademacher complexity, which is then beneficial for improving the generalization ability of the neural network. Besides minimizing the general classification loss, the proposed distortion can simultaneously minimize the expected and empirical risks by adding distortions on feature maps. An extension to convolutional layers and corresponding optimization details are also provided. Experimental results on benchmark image datasets demonstrate that deep networks trained using the proposed feature distortion method perform better than those generated using state-of-the-art methods.

Preliminary

Dropout is a prevalent regularization technology to alleviate over-fitting of models and has achieved great success. It has been demonstrated dropout can improve the generalization ability of models both theoretically (Wan et al. 2013) and practically (Srivastava et al. 2014). In this section, we briefly introduce the generalization theory and dropout method.

Generalization Theory

Generalization theory focuses on the relation between the expected risk and the empirical risk. Considering an L-layer neural network $\boldsymbol{f}^L \in \mathcal{F}$, and a labeled dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ sampled from the ground-truth distribution $\mathcal{Q} \in \mathcal{X} \times \mathcal{Y}$, in which $\boldsymbol{x}_i \in \mathcal{X}$ and $\boldsymbol{y}_i \in \mathcal{Y}$. Denote the weight matrix as $\mathcal{K}^l \in \mathbb{R}^{d^l \times d^{l-1}}$ in which d^l is the dimension of the feature map of l-th layer, and the corresponding output features before and after activation functions ϕ of the l-th layer as $\boldsymbol{o}^l \in \mathbb{R}^{d^l}$ and $\boldsymbol{f}^l \in \mathbb{R}^{d^l}$, respectively. Omitting bias, we have $\boldsymbol{f}^{l+1}(\boldsymbol{x}_i) = \phi(\boldsymbol{o}^{l+1}(\boldsymbol{x}_i)) = \phi(\mathcal{K}^{l+1}\boldsymbol{f}^l(\boldsymbol{x}_i))$. For simplicity, we further refer $\mathcal{K}^{:l}$ as $\{\mathcal{K}^1, \cdots, \mathcal{K}^l\}$.

Taking the image classification task as an example, the expected risk $R(\boldsymbol{f}^L)$ over the population and the empirical risk $\hat{R}(\boldsymbol{f}^L)$ on the training set can be formulated as:

$$R(\boldsymbol{f}^L) = \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{Q}} [\![\ell(\boldsymbol{f}^L(\boldsymbol{x}, \mathcal{K}^{:L}), \boldsymbol{y})]\!], \tag{1}$$

$$\hat{R}(\boldsymbol{f}^L) = \frac{1}{N} \sum_{(\boldsymbol{x}_i,\boldsymbol{y}_i)\in\mathcal{D}} \ell(\boldsymbol{f}^L(\boldsymbol{x}_i, \mathcal{K}^{:L}), \boldsymbol{y}_i), \tag{2}$$

where $\ell(\cdot)$ denotes 0-1 loss. Various techniques have been developed to quantify the gap between the expected risk and the empirical risk, such as PAC learning (Hanneke 2016) , VC dimension (Sontag 1998) and Rademacher complexity (Koltchinskii, Panchenko, and others 2002). Wherein, the empirical Rademacher complexity (ERC) has been widely used as it often leads to a much tighter generalization error bound. The formal definition of ERC is given as follows:

Definition 1 *For a given training dataset with N instances $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ generated by the distribution $\mathcal{Q}$, the empirical Rademacher complexity of the function class of the network $\boldsymbol{f}^L$ is defined as:*

$$\tilde{R}_D(\boldsymbol{f}^L) = \frac{1}{N} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left| \sup_{k,\mathcal{K}^{:L}} \sum_{i=1}^{N} \sigma_i \boldsymbol{f}^L(\boldsymbol{x}_i, \mathcal{K}^{:L})[k] \right|, \tag{3}$$

where Rademacher variables $\boldsymbol{\sigma} = \{\sigma_1, \cdots, \sigma_N\}$, σ_i's are independent uniform random variables in {-1,+1} and $\boldsymbol{f}^L(\boldsymbol{x}_i, \mathcal{K}^{:L})[k]$ is the k-th element in $\boldsymbol{f}^L(\boldsymbol{x}_i, \mathcal{K}^{:L})$.

Using empirical Rademacher complexity and MaDiarmid's inequality, the upper bound of the expected risk $R(\boldsymbol{f}^L)$ can be derived by Theorem 1 (Koltchinskii, Panchenko, and others 2002).

Theorem 1 *Given a fixed $\rho > 0$, for any $\delta > 0$, with probability at least $1 - \delta$, for all $\boldsymbol{f}^L \in \mathcal{F}$*

$$\begin{aligned} R(\boldsymbol{f}^L) \leq \hat{R}(\boldsymbol{f}^L) + \frac{2(d^L)^2}{\rho} \tilde{R}_D(\boldsymbol{f}^L) \\ + \left(1 + \frac{2(d^L)^2}{\rho}\right) \sqrt{\frac{\ln \frac{1}{\delta}}{2N}}, \end{aligned} \tag{4}$$

where d^L denotes the output dimension of the network.

According to Theorem 1 we can find that the gap between expected and empirical risks can be bounded with the help of the empirical Rademacher complexity $\tilde{R}_D(\boldsymbol{f})$ over the specific neural network and dataset. Directly calculating the ERC is vary hard (Kawaguchi, Kaelbling, and Bengio 2017), and thus the upper bound or approximate values of the ERC are usually used in the training phase for obtaining models with better generalization (Kawaguchi, Kaelbling, and Bengio 2017; Zhai and Wang 2018). (Kawaguchi, Kaelbling, and Bengio 2017) obtained models with better generalization by decreasing a regularization term related to the ERC. The effectiveness of decreasing ERC in previous works inspires us to leverage ERC to refine the conventional dropout methods.

Dropout

Dropout is a classical and effective regularization technology to improve the generalization capability of models. There are many variants of dropout,*e.g.* variational dropout and (Kingma, Salimans, and Welling 2015) DropBlock (Ghiasi, Lin, and Le 2018)). Most of them follows the technology of disabling part elements of the feature maps. In general, these methods can be formulated as:

$$\hat{\boldsymbol{f}}^l(\boldsymbol{x}_i) = \boldsymbol{f}^l(\boldsymbol{x}_i) - \boldsymbol{m}_i^l \circ \boldsymbol{f}^l(\boldsymbol{x}_i), \tag{5}$$

where $\circ$ denotes the element-wise product, $\boldsymbol{f}^l(\boldsymbol{x}_i)$ [1] and $\hat{\boldsymbol{f}}^l(\boldsymbol{x}_i)$ are the original feature and distorted features, respectively. In addition, $\boldsymbol{m}_i^l \in \{0,1\}^{d^l}$ is the binary mask applied on feature map $f^l(\boldsymbol{x}_i)$, and each element in $\boldsymbol{m}_i^l$ is draw from Bernoulli distribution, *i.e.* set to 1 with the dropping probability p. Admittedly, implementing dropout on the features in the training phase will force the given network

[1] Without ambiguity, $\boldsymbol{f}^l(\boldsymbol{x}_i, \mathcal{K}^{:l})$ is denoted as $\boldsymbol{f}^l(\boldsymbol{x}_i)$ for simplicity.

paying more attentions on those non-zero regions, and partially solve the "over-fitting". However, disabling the original feature is a heuristic approach and may not always leads to the optimal solution for addressing the aforementioned over-fitting problem in deep neural networks.

Approach

Instead of fixing the value of perturbation, we aim to learn the distortion of the feature map by reducing the ERC of the network. Generally, the disturbing operation employed on the output feature $\boldsymbol{f}^l(\boldsymbol{x}_i)$ of the l-th layer with input data $\boldsymbol{x}_i$ can be formulated as:

$$\hat{\boldsymbol{f}}^l(\boldsymbol{x}_i) = \boldsymbol{f}^l(\boldsymbol{x}_i) - \boldsymbol{m}_i^l \circ \boldsymbol{\varepsilon}_i^l, \tag{6}$$

where $\boldsymbol{\varepsilon}_i^l \in \mathbb{R}^{d^l}$ is the distortion applied the on feature map $\boldsymbol{f}^l(\boldsymbol{x}_i)$. Compared to the dropout method (Eq. (5)) which manually set the distortion as $\varepsilon_i^l = \boldsymbol{f}^l(\boldsymbol{x}_i)$, Eq. (6) automatically learns the form of distortion in the guide of ERC. Directly using $\tilde{R}_D(\boldsymbol{f}^L)$ which is the ERC of the network to guide the distortion $\boldsymbol{\varepsilon}_i^l$ is very hard. Since $\tilde{R}_D(\boldsymbol{f}^L)$ is calculated on the final layer *w.r.t.* the output of the neural network, and it is difficult to trace the intermediate feature maps of the neural network during the training phase. Hence, we reformulate $\tilde{R}_D(\boldsymbol{f}^L)$ by considering the output feature of an arbitrary layer, and obtain the following theorem based on (Wan et al. 2013).

Theorem 2 *Let $\mathcal{K}^l[k,:]$ denotes the k-th row of the weight matrix $\mathcal{K}^l$ and $\|\cdot\|_p$ is the p-norm of vector. Assume that $\|\mathcal{K}^l[k,:]\|_p \leq B^l$, and then the ERC of output can be bounded by the ERC of intermediate feature:*

$$\begin{aligned}&\tilde{R}_D(\boldsymbol{f}^L) \leq 2\tilde{R}_D(\boldsymbol{o}^L) \leq 2B^L\tilde{R}_D(\boldsymbol{f}^{L-1}) \leq \cdots \\ &\leq 2^{L-t}\tilde{R}_D(\boldsymbol{f}^t)\prod_{l=t+1}^{L} B^l \leq 2^{L-t+1}\tilde{R}_D(\boldsymbol{o}^t)\prod_{l=t+1}^{L} B^l,\end{aligned} \tag{7}$$

where $\boldsymbol{o}^l$ and $\boldsymbol{f}^l$ are the feature maps before and after activation function respectively.

The above theorem shows that the ERC of the network $\tilde{R}_D(\boldsymbol{f}^L)$ is upper bounded by the ERC of output feature $\tilde{R}_D(\boldsymbol{f}^t)$ or $\tilde{R}_D(\boldsymbol{o}^t)$ of t-th layer [2]. Thus, decreasing $\tilde{R}_D(\boldsymbol{f}^t)$ or $\tilde{R}_D(\boldsymbol{o}^t)$ can heuristically decrease $\tilde{R}_D(\boldsymbol{f}^L)$. Note that $\boldsymbol{f}^t$ is the feature map of arbitrary intermediate layer t of the network, and the distortion is also applied on intermediate features. Thus, $\tilde{R}_D(\boldsymbol{f}^t)$ or $\tilde{R}_D(\boldsymbol{o}^t)$ is used to guide the distortion in the following.

Feature Map Distortion

In this section, we will illustrate the way of decreasing ERC by applying the distortion $\boldsymbol{\varepsilon}^l$ on the feature map of l-th layer $\boldsymbol{f}^l(\boldsymbol{x}_i)$. By doing so, all the ERCs in the subsequent layers will be affected, and $\tilde{R}_D(\boldsymbol{o}^t)$ satisfying $l < t \leq L$ can guide the distortion $\boldsymbol{\varepsilon}^l$ of l-th layer. Recall that in theorem 2, the closer a layer is to the output layer, the tighter the upper bound of the ERC of the whole network is, and may reduce $\tilde{R}_D(\boldsymbol{f}^L)$ more effectively. However, if $t \gg l$, the relationship between $\tilde{R}_D(\boldsymbol{o}^t)$ and $\boldsymbol{\varepsilon}^l$ becomes complex and it is difficult to guide $\boldsymbol{\varepsilon}^l$ with $\tilde{R}_D(\boldsymbol{o}^t)$. Thus, we use the ERC of $(l+1)$-th layer $\tilde{R}_D(\boldsymbol{o}^{l+1})$ to guide the distortion $\boldsymbol{\varepsilon}^l$ in l-th layer. Specifically, we reduce $\tilde{R}_D(\boldsymbol{o}^{l+1})$ by optimizing $\boldsymbol{\varepsilon}^l$. Denoting

$$\boldsymbol{g}^l(\boldsymbol{x}) = \sum_{i=1}^{N} \sigma_i \hat{\boldsymbol{f}}^l(\boldsymbol{x}_i), \tag{8}$$

for simplicity, $\boldsymbol{g}^l(\boldsymbol{x}) \in \mathbb{R}^{d^l}$ has the same dimension as feature map $\boldsymbol{f}^l(\boldsymbol{x}_i)$. And then, $\tilde{R}_D(\boldsymbol{o}^{l+1})$ is calculated as:

$$\tilde{R}_D(\boldsymbol{o}^{l+1}) = \frac{1}{N}\mathbb{E}_{\boldsymbol{\sigma}} \sup_{k,\mathcal{K}^{:l+1}} \left|\left\langle \mathcal{K}^{l+1}[k,:]^T, \boldsymbol{g}^l(\boldsymbol{x})\right\rangle\right|, \tag{9}$$

where $\mathcal{K}^{l+1}[k,:] \in \mathbb{R}^{1\times d^l}$ denotes the k-th row of the weight matrix $\mathcal{K}^{l+1}$ and $\mathcal{K}^{:l+1} = \{\mathcal{K}^1, \mathcal{K}^2, \cdots, \mathcal{K}^{l+1}\}$. An ideal $\boldsymbol{\varepsilon}^l$ will reduce the ERC of the next layer $\tilde{R}_D(\boldsymbol{o}^{l+1})$ while preserving the representation power.

During the training phase, considering a mini-batch $\bar{\boldsymbol{x}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_{\bar{N}}\}$ with $\bar{N}$ samples, the mask and distortion of the l-th layer are $\boldsymbol{m}^l = \{\boldsymbol{m}_1^l, \boldsymbol{m}_2^l, \cdots, \boldsymbol{m}_{\bar{N}}^l\}$ and $\boldsymbol{\varepsilon}^l = \{\boldsymbol{\varepsilon}_1^l, \boldsymbol{\varepsilon}_2^l, \cdots, \boldsymbol{\varepsilon}_{\bar{N}}^l\}$, respectively. Taking the classification problem as an example, the weights of the network are updated via minimizing the cross-entropy loss. Based on the current updated weights $\mathcal{K}^l$ and Rademacher variables $\bar{\boldsymbol{\sigma}} = \{\sigma_1, \sigma_2, \cdots, \sigma_{\bar{N}}\}$, the optimized disturbance $\hat{\boldsymbol{\varepsilon}}^l$ is obtained by solving the optimization problem:

$$\hat{\boldsymbol{\varepsilon}}^l = \arg\min_{\boldsymbol{\varepsilon}^l} \mathcal{T}(\bar{\boldsymbol{x}}, \boldsymbol{\varepsilon}^l), \;\; l = 1, 2, \cdots, L \tag{10}$$

where

$$\begin{aligned}&\mathcal{T}(\bar{\boldsymbol{x}}, \boldsymbol{\varepsilon}^l) \\ &= \frac{1}{\bar{N}}\left[\sup_k \left|\left\langle \mathcal{K}^{l+1}[k,:]^T, \boldsymbol{g}^l(\bar{\boldsymbol{x}})\right\rangle\right| + \frac{\lambda}{2}\sum_{i=1}^{\bar{N}} \|\boldsymbol{\varepsilon}_i^l\|_2^2\right],\end{aligned} \tag{11}$$

in which $\|\cdot\|_2$ denotes the l_2-norm of the vector and λ is a hyper-parameter balancing the objective function and the intensity of distortion. Intuitively, a violent distortion will destroy the original feature and reduce the representation power.

Optimization of the Distortion

Our goal is to reduce the first term in Eq. (11) related to ERC while constraining the intensity of distortion $\boldsymbol{\varepsilon}_i^l$. Note that the conventional dropout which sets $\boldsymbol{\varepsilon}_i^l = \boldsymbol{f}^l(\boldsymbol{x}_i)$ also achieves the similar goal in a special situation. When the drop probability $p = 1$ and all the elements in mask $\boldsymbol{m}_i^l$ are set to 1, the distortion $\boldsymbol{\varepsilon}_i^l = \boldsymbol{f}^l(\boldsymbol{x}_i)$ makes $\boldsymbol{g}^l(\bar{\boldsymbol{x}}) = 0$ and thus the first term in Eq. (11) is zero, showing that the

[2] The definition of $\tilde{R}_D(\boldsymbol{f}^t)$ and $\tilde{R}_D(\boldsymbol{o}^t)$ in t-th layer has the same form as Definition 1, *i.e.* $\tilde{R}_D(\boldsymbol{f}^t) = \frac{1}{N}\mathbb{E}_{\boldsymbol{\sigma}}\left|\sup_{k,\mathcal{K}^{:t}} \sum_{i=1}^{N} \sigma_i \boldsymbol{f}^t(\boldsymbol{x}_i, \mathcal{K}^{:t})[k]\right|$ and $\tilde{R}_D(\boldsymbol{o}^t) = \frac{1}{N}\mathbb{E}_{\boldsymbol{\sigma}}\left|\sup_{k,\mathcal{K}^{:t}} \sum_{i=1}^{N} \sigma_i \boldsymbol{o}^t(\boldsymbol{x}_i, \mathcal{K}^{:t})[k]\right|$

dropout also has the potential to reduce ERC. However, the semantic information is also dropped away and the network will make random guess. In the general case where $p < 1$, the conventional dropout disables part of the feature maps, which may decrease the value of $\mathcal{T}(\bar{\boldsymbol{x}}, \boldsymbol{\varepsilon}^l)$, but there is no explicit interaction with the empirical Rademacher complexity. We choose $\boldsymbol{f}^l(\boldsymbol{x}_i)$ as the initial value of $\boldsymbol{\varepsilon}_i^l$ and optimize Eq. (10) with gradient descent. The partial derivative of $\mathcal{T}(\bar{\boldsymbol{x}}, \boldsymbol{\varepsilon}^l)$ *w.r.t.* $\boldsymbol{\varepsilon}_i^l$ is calculated as:

$$\frac{\partial \mathcal{T}(\bar{\boldsymbol{x}}, \boldsymbol{\varepsilon}^l)}{\partial \boldsymbol{\varepsilon}_i^l} = -\frac{1}{\bar{N}} \sigma_i s_{\hat{k}} \mathcal{K}^{l+l}[\hat{k}, :]^T \circ \boldsymbol{m}_i^l + \frac{\lambda}{\bar{N}} \boldsymbol{\varepsilon}_i^l, \quad (12)$$

where

$$\hat{k} = \arg\max_k \left| \left\langle \mathcal{K}^{l+1}[k, :]^T, \boldsymbol{g}^l(\bar{\boldsymbol{x}}) \right\rangle \right|, \quad (13)$$

$$s_{\hat{k}} = \operatorname{sign} \left\langle \mathcal{K}^{l+1}[\hat{k}, :]^T, \boldsymbol{g}^l(\bar{\boldsymbol{x}}) \right\rangle. \quad (14)$$

Eq. (13) chooses the row of weight matrix to obtain the maximum inner product $\left\langle \mathcal{K}^{l+1}[k, :]^T, \boldsymbol{g}^l(\bar{\boldsymbol{x}}) \right\rangle$ and Eq. (14) calculates the sign of the inner product. The equations above show that the optimization of distortion $\boldsymbol{\varepsilon}^l$ is related to the feature $\boldsymbol{f}^l(\boldsymbol{x}_i)$ and the weight $\mathcal{K}^{l+1}$ in the following layer. Note that precisely calculating the gradient $\frac{\partial \mathcal{T}(\boldsymbol{x}, \boldsymbol{\varepsilon}^l)}{\partial \boldsymbol{\varepsilon}_i^l}$ is time-consuming and not necessary, and it can be appropriately estimated without much influence of the performance. Rademacher variable σ_i is randomly sampled from $\{\pm 1\}$ with equal probability (Definition 1), and thus the impact of $s_{\hat{k}}$ can be neglected. Selecting the row index k of $\mathcal{K}^{l+1}$ is also related to the random variable σ_i, and hence we leverage the random variables to approximate the process. Denote $\mathcal{K}_M^{l+1} = [\max(\mathcal{K}^{l+1}[:, 1]), \max(\mathcal{K}^{l+1}[:, 2]), \cdots, \max(\mathcal{K}^{l+1}[:, d^l])]^T$ in which the j-th element is the maximum value of the j-th column of weight matrix $\mathcal{K}^{l+1}$. Then the gradient $\frac{\partial \mathcal{T}(\boldsymbol{x}, \boldsymbol{\varepsilon}^l)}{\partial \boldsymbol{\varepsilon}_i^l}$ is approximated as:

$$\frac{\partial T^{l+1}}{\partial \boldsymbol{\varepsilon}_i^l} \approx -\frac{1}{\bar{N}} \sigma_i \boldsymbol{u} \circ \mathcal{K}_M^{l+1} \circ \boldsymbol{m}_i^l + \frac{\lambda}{\bar{N}} \boldsymbol{\varepsilon}_i^l, \quad (15)$$

where $\boldsymbol{u} \in d^l$ is a random variable whose elements are sampled from standard normal distribution $\mathcal{N}(0, 1)$ with zero mean and standard deviation. $\boldsymbol{u} \circ \mathcal{K}_M^{l+1}$ is to approximate the process of selecting the row of weight $\mathcal{K}^{l+1}$. Denote γ as the step length and we can update $\boldsymbol{\varepsilon}_i^l$ along the negative gradient direction:

$$\boldsymbol{\varepsilon}_i^l \leftarrow \boldsymbol{\varepsilon}_i^l - \gamma \frac{\partial T^{l+1}}{\partial \boldsymbol{\varepsilon}_i^l}. \quad (16)$$

To train an optimal neural network, we tend to simultaneously reduce the empirical risk on the training dataset (*e.g.* minimizing the cross entropy) and the Rademacher complexity. There is thus a balance between the ordinary loss and the reduction of Rademacher complexity. This can be realized by alternatively optimizing between the ordinary loss *w.r.t.* weights of the network and Rademacher complexity *w.r.t.* the distortion $\boldsymbol{\varepsilon}^l$. After obtaining the updated weights of the network, the distortion $\boldsymbol{\varepsilon}_i^l$ is optimized to decrease the objective $\mathcal{T}(\boldsymbol{x}, \boldsymbol{\varepsilon}^l)$. After each update of weights of the network, the $\boldsymbol{\varepsilon}_i^l$ can be updated for several times, which is usually adopted in practice for training efficiency (Goodfellow et al. 2014). Using the case that applying distortion on feature maps of all the layers as an example, the training procedure of the network is summarized in Algorithm 1. Following dropout(Srivastava et al. 2014), the feature map is rescaled by a factor of p at testing stage, which is equally implemented as dividing p in the training phase in practice(Srivastava et al. 2014).

Algorithm 1 Feature map distortion for training networks.

Input: Training data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$, The weights of the network $\mathcal{K}^{:L} = \{\mathcal{K}^1, \mathcal{K}^2, \cdots, \mathcal{K}^L\}$

repeat
  for l in $1, \cdots L$ **do**
    Calculate the feature map $\boldsymbol{f}^l(\boldsymbol{x}_i)$ of the l-th layer;
    Generate the distortion $\boldsymbol{\varepsilon}_i^l$ and the corresponding sample mask $\boldsymbol{m}_i^l$;
    Obtain distorted feature $\hat{\boldsymbol{f}}^l(\boldsymbol{x}_i)$ (Eq. (6));
    Feed-forward the network using $\hat{\boldsymbol{f}}^l(\boldsymbol{x}_i)$;
  end for
  Backward and update weights $\mathcal{K}^{:L}$ in the network;
until Convergence;

Output: The resulting deep neural network.

Extension to Convolutional Layers

Convolutional layer can be seen as a special full-connected layer with sparse connection and shared weights. Hence, the distortion $\boldsymbol{\varepsilon}^l$ can be learned in the same way as that in the FC layer. In the following, we focus on distorting the feature maps to reduce the empirical Rademacher complexity in convolutional layers, considering the particularity of convolution operations.

The convolutional kernel of l-th layer is denoted as $\mathcal{K}^l \in \mathbb{R}^{d_c^l \times d_c^{l-1} \times d_h^{l-1} \times d_w^{l-1}}$, and the corresponding output feature maps before and after activation function ϕ are denoted as $O^l(\boldsymbol{x}_i) \in \mathbb{R}^{d_c^l \times d_{h'}^l \times d_{w'}^l}$ and $F^l(\boldsymbol{x}_i) \in \mathbb{R}^{d_c^l \times d_{h'}^l \times d_{w'}^l}$, respectively. d_h^l and d_w^l are the height and width of convolutional kernels while $d_{h'}^l$ and $d_{w'}^l$ are those of the feature map. The mask $M_i^l \in \mathbb{R}^{d_c^l \times d_{h'}^l \times d_{w'}^l}$ and distortion $\boldsymbol{\varepsilon}_i^l \in \mathbb{R}^{d_c^l \times d_{h'}^l \times d_{w'}^l}$ of the l-th layer have the same dimension as feature map $F^l(\boldsymbol{x}_i)$ and is applied to $F^l(\boldsymbol{x}_i)$ to get the disturbed feature map $\hat{F}^l(\boldsymbol{x}_i)$, *i.e.*

$$\hat{F}^l(\boldsymbol{x}_i) = F^l(\boldsymbol{x}_i) - M_i^l \circ \boldsymbol{\varepsilon}_i^l. \quad (17)$$

Similar to the fully-connected layer, the ERC $\tilde{R}_D(O^{l+1})$ in the $(l+1)$-th layer is used to guide the optimization of distortion $\boldsymbol{\varepsilon}^l$ in layer l. Given a mini-batch $\bar{\boldsymbol{x}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_{\bar{N}}\}$ together with mask $M^l = \{M_1^l, M_2^l, \cdots, M_{\bar{N}}^l\}$ and distortion $\boldsymbol{\varepsilon}^l = \{\boldsymbol{\varepsilon}_1^l, \boldsymbol{\varepsilon}_2^l, \cdots, \boldsymbol{\varepsilon}_{\bar{N}}^l\}$, and two symbols $G^l(\bar{\boldsymbol{x}})$ and

Table 1: Accuracies of conventional CNNs on CIFAR-10 and CIFAR-100 datasets.

Method	CIFAR-10 (%)	CIFAR-100 (%)
CNN	81.99	49.72
CNN + Dropout (Srivastava et al. 2014)	82.95	54.19
CNN + Vardrop (Kingma, Salimans, and Welling 2015)	83.15	54.53
CNN + Sparse Vardrop (Molchanov, Ashukha, and Vetrov 2017)	82.13	54.26
CNN + RDdrop (Zhai and Wang 2018)	83.11	54.65
CNN + Feature Map Distortion	**85.24 $\pm$ 0.08**	**56.23$\pm$ 0.12**

$Q^{l+1}(\bar{\boldsymbol{x}})$ are defined for notion simplicity:

$$G^l(\bar{\boldsymbol{x}}) = \sum_{i=1}^{N} \sigma_i \hat{F}^l(\bar{\boldsymbol{x}}_i), \tag{18}$$

$$Q^{l+1}(\bar{\boldsymbol{x}})[k,:,:] = \sum_{c=1}^{d_c^l} \mathcal{K}^{l+1}[k,c,:,:] * G^l[c,:,:], \tag{19}$$

where $*$ denotes convolutional operation. $G^l(\bar{\boldsymbol{x}})$ is related to the distorted feature and the Rademacher variable in the l-th layer, and Eq. (19) applies the convolutional operation on $G^l(\bar{\boldsymbol{x}})$. Given the notation mentioned above, $\boldsymbol{\varepsilon}^l$ can be derived by minimizing the following objective function:

$$\hat{\boldsymbol{\varepsilon}}^l = \arg\min_{\boldsymbol{\varepsilon}^l} \mathcal{T}(\bar{\boldsymbol{x}}, \boldsymbol{\varepsilon}^l), \tag{20}$$

where

$$\begin{aligned}\mathcal{T}(\bar{\boldsymbol{x}}, \boldsymbol{\varepsilon}^l) = &\frac{1}{\bar{N} d_h^{l+1} d_w^{l+1}} \sup_k \sum_{h'=1}^{d_h^{l+1}} \sum_{w'=1}^{d_{w'}^{l+1}} \left| Q^{l+1}(\bar{\boldsymbol{x}})[k,h',w'] \right| \\ &+ \frac{\lambda}{2\bar{N}} \sum_{i=1}^{\bar{N}} \|\boldsymbol{\varepsilon}_i^l\|_2^2.\end{aligned} \tag{21}$$

$\mathcal{T}(\bar{\boldsymbol{x}}, \boldsymbol{\varepsilon}^l)$ comes from the simplified implementation $\tilde{R}_D(O^{l+1})$ which is the ERC in a mini-batch. As Eq. (21) calculates average over the spatial dimension of $Q^{l+1}(\bar{\boldsymbol{x}})$, elements in different spatial locations of $\boldsymbol{\varepsilon}_i^l$ has equal contribution to $Q^{l+1}(\bar{\boldsymbol{x}})$. Thus, the partial derivative of $Q^{l+1}(\bar{\boldsymbol{x}})$ *w.r.t.* $\boldsymbol{\varepsilon}_i^l$ is:

$$\begin{aligned}\frac{\partial \mathcal{T}}{\partial \boldsymbol{\varepsilon}_i^l[c,h',w']} = &-\frac{1}{\bar{N} d_h^{l+1} d_w^{l+1}} \sigma_i \sum_{h=1}^{d_h^{l+1}} \sum_{w=1}^{d_w^{l+1}} \mathcal{K}^{l+1}[\hat{k},c,h,w] S[\hat{k},h,w] \\ &+ \frac{\lambda}{\bar{N}} \boldsymbol{\varepsilon}_i^l, h' \in \{1,2\cdots,d_{h'}^l\},\ \ w' \in \{1,2\cdots,d_{w'}^l\},\end{aligned} \tag{22}$$

where

$$\hat{k} = \arg\max_k \sum_{h=1}^{d_{h'}^{l+1}} \sum_{w=1}^{d_{w'}^{l+1}} \left| Q^{l+1}(\bar{\boldsymbol{x}})[k,h',w'] \right|, \tag{23}$$

$$S = \operatorname{sign}\left(Q^{l+1}(\bar{\boldsymbol{x}})\right) \tag{24}$$

in which $S \in \{\pm 1\}^{d_c^l \times d_{h'}^l \times d_{w'}^l}$ is the sign of each element in $Q^{l+1}(\bar{\boldsymbol{x}})$. Considering the impact of Rademacher variable σ_i and similar to the method in FC layer, random variables $S' \in \{\pm 1\}^{d_h^l \times d_w^l}$ and $U \in \mathbb{R}^{d_c^l \times d_{h'}^l \times d_{w'}^l}$ are introduced to simply Eq. (22), which are used to approximate S and the channel selection process of $\mathcal{K}^{l+1}$ respectively. Each element in S' is ± 1 with equal probability and each element in U follows the standard normal distribution $\mathcal{N}(0,1)$. Given the gradient, the distortion $\boldsymbol{\varepsilon}_i^l$ is updated in a similar way as FC layer. The algorithm of the feature distortion on the convolutional layers is similar to Algorithm 1.

Different from the method applied on FC layers where each element of the binary mask M^l is sampled independently, we draw lessons from DropBlock (Ghiasi, Lin, and Le 2018) where elements in a contiguous square block with given size $block\_size$ of the feature map is distorted simultaneously. We denote the extension of the proposed method to convolutional layers as "block feature map distortion".

Experiments

In this section, we conduct experiments on several benchmark datasets to validate the effectiveness of the proposed feature map distortion method. The method is implemented on both FC layers and convolutional layers, which are validated with conventional CNNs and modern CNNs (*e.g.* ResNet) respectively. In order to set unified hyperparameters γ for different layers, we multiply γ by the standard deviation of the feature maps in each layer, and alternately update the distortion and weight one step for efficiency. The distortion probability (dropping probability for dropout and dropblock) increases linearly from 0 to the appointed distortion probability p following (Ghiasi, Lin, and Le 2018).

Experiments on Fully Connected Layers

To validate the effect of the proposed feature map distortion method implemented on the FC layers, we conduct experiments on a conventional CNN on CIFAR-10 and CIFAR-100 dataset. The proposed method is compared with multiple state-of-the-art variants of dropout.

Dataset. CIFAR-10 and CIFAR-100 dataset both contain 60000 natural images with size 32×32. 50000 images are used for training and 10000 for testing. The images are divided into 10 categories and 100 categories, respectively. 20% of the training data are regarded as validation sets. Data augmentation method is not used for fair comparison.

Implementation details. The conventional CNN has three convolutional layers with 96, 128 and 256 filters, respectively. Each layer consists of a 5×5 convolutional operation with stride 1 followed by a 3×3 max-pooling operation with stride 2. Then the features are sent to two fully-connected layers with 2048 hidden units each. We imple-

Table 2: Accuracies of ResNet-56 on CIFAR10 and CIFAR-100 dataset.

Model	CIFAR-10 (%)	CIFAR-100 (%)
Resnet-56	93.95 ± 0.09	71.81 ± 0.21
Resnet-56 + DropBlock (Ghiasi, Lin, and Le 2018)	94.18 ± 0.14	73.08 ± 0.23
Resnet-56 + Block Feature Map Distortion	$\mathbf{94.50 \pm 0.15}$	$\mathbf{73.71 \pm 0.20}$

ment the distortion method on each FC layer. Distortion probability p is selected from $\{0{,}4,\ 0.5,\ 0.6\}$ and the step length γ is set to 5. The model is trained for 500 epoch with batchsize 128. The learning rate is initialized with 0.01, and decayed by a factor of 10 at 200, 300 and 400 epochs. We run our method 5 times with different random seeds and report the average accuracy with standard deviation.

Compared methods. The CNN model trained without extra regularization tricks is used as the baseline model. Further more, we compare our method with the widely used dropout method (Hinton et al. 2012) and several state-of-the-art variants, including Vardrop (Kingma, Salimans, and Welling 2015), Sparse Vardrop (Molchanov, Ashukha, and Vetrov 2017) and RDdrop (Zhai and Wang 2018).

Results. The test accuracies on both CIFAR-10 and CIFAR-100 are summarized in Table 1. The proposed feature map distortion method is superior to the compared methods by a large margin on both two datasets. CNN trained with the help of the proposed method achieves an accuracy of 85.24%, which improves the performance of the state-of-the-art RDdrop method with 2.13% and 1.58% on CIFAR-10 and CIFAR-100 dataset, respectively. It shows that the proposed feature map distortion method can reduce the empirical Rademacher complexity effectively while preserve the representation power of the model, resulting in a better test performance.

Experiments on Convolutional Layers

It is much important to apply the proposed method to convolutional layer since modern CNN such as ResNet mostly consist of convolutional layers. In this section, we apply the proposed method on convolutional layers and conduct several experiments on both CIFAR-10 and CIFAR-100 dataset.

Implementation details. The widely-used ResNet-56 (He et al. 2016b) which contains three groups of blocks is used as the baseline model. DropBlock method (Ghiasi, Lin, and Le 2018) is used as the peer competitor. Both the proposed block feature map distortion method and DropBlock method are implemented after each convolution layers in the last group with *block_size=6*, and the distortion probability (dropping probability for DropBlock) p is selected from $\{0.01, 0.02, \cdots, 0.1\}$. The step length γ is set to 30 empirically. Standard data augmentation including random cropping, horizontal flipping and rotation(within ± 15 degrees) are conducted during training. The networks are trained for 200 epochs, batchsize is set to 128 and weight decay is set to 5e-4. The initial learning rate is set to 0.1 and is decayed by a factor of 5 at 60, 120 and 160 epochs. All the methods are repeated 5 times with different random seeds and the average accuracies with standard deviations are reported.

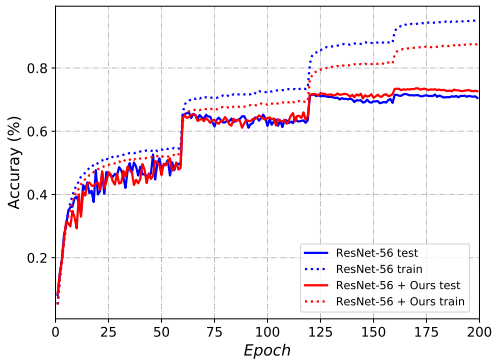


Figure 1: Training curves on the CIFAR-100 dataset.

Results. The results on both CIFAR-10 and CIFAR-100 dataset are shown in Table 2. The proposed method is superior to DropBlock method and improves the performance with 0.32% and 0.63%, respectively. It shows that the proposed feature map distortion methods suits for convolutional layers and can improves the performance of modern network structures.

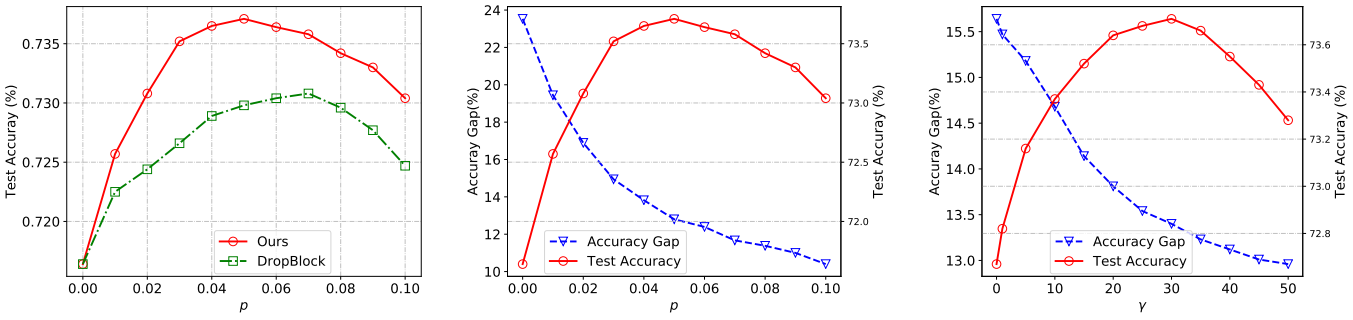
Training curve. The training curves on CIFAR-100 dataset are shown in Figure 1. The solid line and dotted line denote the test stage and the training stage respectively, while the red line and blue line denote the proposed feature map distortion method and the baseline model. When training converges, the baseline ResNet-56 traps in overfitting problem and achieves a higher training accuracy but lower test accuracy, while the proposed feature map distortion method overcome this problem and achieves a higher test accuracy, which shows the improvement of model generalization ability.

Feature map distortion *v.s.* DropBlock. The test accuracy of our method (red) and the Dropblock method (green) with various distortion probability (dropping probability) p on CIFAR-100 dataset are shown in Figure 2(a). Increasing the drop probability p enhances the effect of regularization, and the test accuracy can be improved when setting p in an appropriate range. Note that our method achieves a better performance than DropBlock with p in a larger range, which demonstrate the superior of feature map distortion.

Test accuracy *v.s.* accuracy gap. Figure 2(b) and (c) show how test accuracy (red) and the accuracy gap between training and testing accuracies (blue) vary when setting different distortion probability p and length step γ. Larger p implies that more locations of the feature maps are distorted while γ controls the intensity of disturbing in each location. Increasing either p or γ bring stronger regularization, resulting in smaller gap between the training and testing

Table 3: Accuracies of ResNet-50 on ImageNet dataset.

Model	Top-1 Accuracy (%)	Top-5 Accuracy (%)
ResNet-50	76.51 ± 0.07	93.20 ± 0.05
ResNet-50 + Dropout (Srivastava et al. 2014)	76.80 ± 0.04	93.41 ± 0.04
ResNet-50 + DropPath (Larsson, Maire, and Shakhnarovich 2016)	77.10 ± 0.08	93.50 ± 0.05
ResNet-50 + SpatialDropout (Tompson et al. 2015)	77.41 ± 0.04	93.74 ± 0.02
ResNet-50 + Cutout (DeVries and Taylor 2017)	76.52 ± 0.07	93.21 ± 0.04
ResNet-50 + AutoAugment (Cubuk et al. 2018)	77.63	93.82
ResNet-50 + Label Smoothing (Szegedy et al. 2016)	77.17 ± 0.05	93.45 ± 0.03
ResNet-50 + DropBlock (Ghiasi, Lin, and Le 2018)	78.13 ± 0.05	94.02 ± 0.02
ResNet-50 + Feature Map Distortion	77.71 ± 0.05	93.89 ± 0.04
ResNet-50 + Block Feature Map Distortion	**78.76 ± 0.05**	**94.33 ± 0.03**



(a) (b) (c)

Figure 2: The impact of distortion probability p and step length γ on CIFAR-100 dataset. Test accuracies *w.r.t.* distortion probability p for feature map distortion and dropblock are shown in (a). Test accuracies and accuracy gaps *w.r.t.* distortion probability p and step length γ are shown in (b) and (c).

accuracies, which means a stronger generalization ability. However, disturbing too many locations or disturbing a location with too much intensity may destroy the representation power and having negative impact on the final testing accuracy. Instead of using fixed intensity in conventional dropout and DropBlock method, out method applies proper intensity distortion on proper locations and results in better performance.

Experiments on ImageNet Dataset

In this section, we conducts experiments on large-scale ImageNet dataset and implement the feature map distortion method with conventional dropout and the recent DropBlock method, namely "Feature Map Distortion" and "Block Feature Map Distortion", respective.

Dataset. ImageNet dataset contains 1.2M training images and 50000 validation images, consisting of 1000 categories. Standard data augmentation methods including random cropping and horizontally flipping is conducted on training data.

Implementation details. We follow the experimental settings in (Ghiasi, Lin, and Le 2018) for fair comparison. The prevalent ResNet-50 is used as the baseline model. The distortions are applied on the feature maps after both convolutional layers and skip connections in the last two groups. The step length is set to 5. For feature map distortion implemented based on conventional dropout, distortion probability p (dropping probability) is set to 0.5 as suggested by (Srivastava et al. 2014). For Block feature map distortion, the *block_size* and p (dropping probability) are set to 6 and 0.05 following (Ghiasi, Lin, and Le 2018). We report the single-crop top-1 and top-5 accuracies on the validation set and repeat the methods three time with different random seeds.

Compared method. Multiple state-of-the-art regularization methods are compared, including dropout based methods, data augmentation and label smoothing. DropPath(Larsson, Maire, and Shakhnarovich 2016), SpatialDropout(Tompson et al. 2015) and Dropblock (Ghiasi, Lin, and Le 2018) are the state-of-the-art variants of dropout. Data augmentation including Cutout (DeVries and Taylor 2017) and AutoAugment (Cubuk et al. 2018)), and label smoothing (Szegedy et al. 2016) are prevalent regularization techniques to alleviate over-fitting.

Results. In Table 3, the proposed feature distortion method can not only increase the performance of deep neural networks using conventional dropout method, but also enhance the peformance of the recent Dropblock method, since our method is also suitable and well adapted to convolutional layers. As a result, the feature map distortion improve the accuracy from 76.80% to 77.71% compared to the conventional dropout method . The block feature map dis-

tortion method achieves top-1 accuracy 78.76%, which surpass other state-of-the art methods from a large margin. The results demonstrate that our method can simultaneously increase the generalization ability and preserving the useful information of original features.

Conclusion

Dropout based methods have been successfully used for enhancing the generalization ability of deep neural networks. However, eliminating some of units in neural networks can be seen as a heuristic approach for minimizing the gap between expected and empirical risks of the resulting network, which is not the optimal one in practice. Here we propose to embed distortions onto feature maps of the given deep neural network by exploiting the Rademacher complexity. We further extend the proposed method to convolutional layers and explore the detailed feed-forward and back-propagation procedures. Thus, we can employ the proposed method into any off-the-shelf deep neural architectures. Extensive experimental results show that the feature distortion technique can be easily embedded into mainstream deep networks to achieve better performance on benchmark datasets over conventional approaches.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant No. 61876007, 61872012 and Australian Research Council under Project DE-180101438.

References

Ba, J., and Frey, B. 2013. Adaptive dropout for training deep neural networks. In *Advances in Neural Information Processing Systems*, 3084–3092.

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

DeVries, T., and Taylor, G. W. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Feichtenhofer, C.; Pinz, A.; and Zisserman, A. 2016. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1933–1941.

Ghiasi, G.; Lin, T.-Y.; and Le, Q. V. 2018. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, 10727–10737.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Hanneke, S. 2016. The optimal sample complexity of pac learning. *The Journal of Machine Learning Research* 17(1):1319–1333.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Kawaguchi, K.; Kaelbling, L. P.; and Bengio, Y. 2017. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, 2575–2583.

Koltchinskii, V.; Panchenko, D.; et al. 2002. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics* 30(1):1–50.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Larsson, G.; Maire, M.; and Shakhnarovich, G. 2016. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*.

Molchanov, D.; Ashukha, A.; and Vetrov, D. 2017. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2498–2507. JMLR. org.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Sontag, E. D. 1998. Vc dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences* 168:69–96.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1):1929–1958.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Tompson, J.; Goroshin, R.; Jain, A.; LeCun, Y.; and Bregler, C. 2015. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 648–656.

Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *International conference on machine learning*, 1058–1066.

Wang, Y.; Xu, C.; Chunjing, X.; Xu, C.; and Tao, D. 2018a. Learning versatile filters for efficient convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1608–1618.

Wang, Y.; Xu, C.; Xu, C.; and Tao, D. 2018b. Packing convolutional neural networks in the frequency domain. *IEEE transactions on pattern analysis and machine intelligence*.

Wang, C.; Li, M.; and Smola, A. J. 2019. Language models with transformers. *CoRR* abs/1904.09408.

Zhai, K., and Wang, H. 2018. Adaptive dropout with rademacher complexity regularization.