Received January 2, 2018, accepted February 19, 2018, date of publication March 5, 2018, date of current version April 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2810849

Improvement of Generalization Ability of Deep CNN via Implicit Regularization in Two-Stage Training Process

QINGHE ZHENG[1], MINGQIANG YANG[1], JIAJIE YANG[2], QINGRUI ZHANG[1], AND XINXIN ZHANG[1]
[1]School of Information Science and Engineering, Shandong University, Jinan 205100, China
[2]Department of Science, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada

Corresponding author: Mingqiang Yang (yangmq@sdu.edu.cn)

This work was supported in part by the Shandong Provincial Natural Science Foundation under Grant ZR2014FM030 and in part by the National Natural Science Foundation of China under Grant 61571275.

ABSTRACT Optimization of deep learning is no longer an imminent problem, due to various gradient descent methods and the improvements of network structure, including activation functions, the connectivity style, and so on. Then the actual application depends on the generalization ability, which determines whether a network is effective. Regularization is an efficient way to improve the generalization ability of deep CNN, because it makes it possible to train more complex models while maintaining a lower overfitting. In this paper, we propose to optimize the feature boundary of deep CNN through a two-stage training method (pre-training process and implicit regularization training process) to reduce the overfitting problem. In the pre-training stage, we train a network model to extract the image representation for anomaly detection. In the implicit regularization training stage, we re-train the network based on the anomaly detection results to regularize the feature boundary and make it converge in the proper position. Experimental results on five image classification benchmarks show that the two-stage training method achieves a state-of-the-art performance and that it, in conjunction with more complicated anomaly detection algorithm, obtains better results. Finally, we use a variety of strategies to explore and analyze how implicit regularization plays a role in the two-stage training process. Furthermore, we explain how implicit regularization can be interpreted as data augmentation and model ensemble.

INDEX TERMS Deep CNN, image classification, overfitting, generalization, anomaly detection, implicit regularization.

I. INTRODUCTION

Is data omnipotent in the era of Big Data? Or the success of AlphaGo Zero [1] means that the data is not as important as we thought?

The flourishing development of applications based on deep learning is inseparable from the massive data support, especially in image classification [2], behavior recognition [3], object detection [4], and so on. As a widely used regularization method, the data augmentation [2], [20] expands datasets in various ways to improve the generalization ability of deep CNN, such as horizontal flip, translation transformation, noise disturbance, etc. It is in line with people's intuition to approximate the real and natural sample distribution by increasing the amount of training data. But the overfitting problem of deep CNN still exists. The data-driven training method makes us start thinking about the impact of data distribution in high dimensional space on the training process. In other words, the necessary reduction of dataset under the trend of big data is more consistent with our intuition, *i.e.*, the effective expansion of the dataset is a more reasonable way to approximate the natural sample distribution, rather than a reckless expansion of the data.

Therefore, we plan to regularize the feature boundaries of deep CNN in a two-stage training process from the point of view of data punishment so as to improve the generalization ability of the network. The initial sample distribution is obtained in the pre-training stage, and the implicit regularization phase is used to optimize the feature boundary

TABLE 1. Comparison with different CNN regularization techniques.

Regularization methods	Working time	Object units
Weight decay [5]	training process	weights
DropConnect [9]	training process	weights
Dropout [10]	training process	hidden nodes
Maxout [13]	training and testing process	activation layers
Parametric ReLU [7]	training and testing process	activation layers
Switsh [8]	training and testing process	activation layers
Batch normalization [17]	training and testing process	feature maps
Stochastic pooling [12]	training and testing process	pooling layers
Data augmentation [20]	data pretreatment	training samples
DisturbLabel [16]	data pretreatment	loss layer
SoftLabel [15]	data pretreatment	loss layer
Transfer learning	pre-training process	weights
OrthoReg [19]	parameter initialization	weights
LSUV [18]	parameter initialization	weights
Xavier [65]	parameter initialization	weights
MSRA [7]	parameter initialization	weights
Our method	**training process**	**loss function**

by punishing samples. Regularization in the training process plays a role in the parameters adjustment of deep CNN, which is similar to some earlier regularization methods, such as weight decay, which can be interpreted as a way of constraining the parameters by l_2-regularization and has been widely adopted. But the l_2-regularization term plays a very different role in deep learning, and it seems to be more of a tuning parameter. As reported in [2], l_2-regularization (weight decay [5]) sometimes even helps optimization, illustrating its poorly understood nature.

Currently, avoiding overfitting problem is a significant challenge to train effective and robust deep CNNs. Early solutions include constraining the network complexity by using fewer weights or sharing weights [6], and using l_1-regularization method. Some new activation function such as Parametric ReLU [7] and Swish [8] achieve the same goal through sparse representation property. The sparse features are more likely to be linearly separable and have a smaller dependence on non-linear mapping mechanisms. There are also many regularization methods for network structure design that have been developed to prevent overfitting. Dropout [9] randomly discards the output of some hidden neurons in the training process and only updates the remaining weights in each mini-batch iteration. Similarly, DropConnect [10] just updates a randomly selected subset of the network parameters. Both of these two tricks introduce randomicity to the network and play a role of bagging by combining multiple models, avoiding the increase of computation time and storage costs. The auxiliary classification nodes in the GoogLeNet [11] can also play a role of the model ensemble. There are also some methods to avoid overfitting by introducing randomicity to the network. Stochastic Pooling [12] changes the deterministic pooling operation and randomly selects one input as the output in probability in the training process. Similarly, Probabilistic Maxout [13] turns the Maxout operation [14] to stochastic. SoftLabel [15] randomly adds noise to the sample label, and DisturbLabel [16] regularizes the loss layer of the network by randomly replacing the labels of some training samples. Although they are not directly presented as regularizers, there are other strategies to reduce the overfitting such as Batch Normalization [17], which decreases the overfitting by lowering the internal covariance shift.

Regarding model initialization, Mishkin and Matas [18] proposed that a proper initialization avoids the network converging to a poorer local minimum and can also bring the regularization effect. In the natural image classification task, the commonly used method is to build a good initialization model by pre-training on the ImageNet dataset. OrthoReg [19] eliminates interferences between negatively correlated features by imposing local constraints in feature decorrelation, allowing the regularizer to achieve higher decorrelation boundaries, and decreasing the overfitting problem more efficiently. In the experimental part, we introduce and compare different initialization methods, including LSUV initialization [18], orthogonal initialization, Xavier initialization [65] and other ways to observe the influence on the generalization ability.

A comparison of different regularization methods is summarized in Table 1. Instead of adding noise to input nodes, our work is to establish an optimized *feature boundary* by punishing the loss function of outliers that may be disturbed by the severe noise due to data transmission, strange collection or production methods, etc. The adjustment of *decision boundary* means that only the classifier is optimized and the sample distribution is fixed. While the adjustment of *feature boundary* means that both the sample distribution and the classifier are optimized. The rest of this paper is organized as follows. Section II briefly introduces related work. The two-stage training method is presented in Section III. Experimental settings and results are shown in Section IV. Discussions and conclusions are presented in Sections V and VI, respectively. The Appendix section shows some of the anomalous samples in MNIST, SVHN, USPS, and CIFAR10/100.

II. RELATED WORK

The extensive use of deep CNN in dealing with image classification problem has benefitted from and inspired a variety of research efforts, including designing deeper and wider network architectures [11], [21], studying more suitable non-linear activation functions [7], [14], exploring advanced pooling operations [12], [22], using better optimization methods [23], regularization techniques [16], [17] preventing deep CNN from overfitting problem, etc. In this section, we introduce the work related to the two-stage training process (pre-training and implicit regularization training).

A. PRE-TRAINING PROCESS

The first stage of the training process is the pre-training phase, which is used to obtain initial distribution of samples. At this stage, the deep convolution activation features (DeCAFs) [29] of samples are extracted by the trained model, which has been applied in a lot of work. Zhang and Zhang [24] discussed the classification results of various classifiers on the deep convolution activation features. Gehler and Nowozin [25] reported that the fusion of deep convolution activation features and artificial features could improve the classification accuracy. Gong *et al.* [26] constructed multi-scale features as the image representation by extracting the activations of different layers and obtained the state-of-the-art classification result with fisher coding. Shi *et al.* [27] used the deep convolution activation features for ground-based cloud classification, and it outperformed conventional artificial features considerably without using any tricks in the training process. Zhong *et al.* [28] improved classification results on the Caltech-UCSD birds dataset and SVHN dataset by applying PCA and stretching operation to change the dimensions of DeCAFs. A lot of work has shown that the deep convolution activation features based on deep CNN achieved excellent results in various fields. It is the foundation of our work.

B. IMPLICIT REGULARIZATION TRAINING PROCESS

The second stage of the training process is the implicit regularization. In this step, the primary task is to detect and punish anomalous samples to achieve the purpose of optimizing the feature boundaries. The sample punishment is equivalent to reducing the size of the dataset, which preserves the valid data to fit the real sample distribution. Detection of the anomalous sample is the basis of this step, which is very similar to the outlier detection task in data mining.

Outliers are samples that are significantly different from other data objects as if they were produced by different mechanisms. Therefore, automatically removing outliers can help us build large-scale datasets, or at least greatly reduce the required labeling costs. For example, as shown in Fig. 1a, the image retrieval results from a search engine which typically contains a number of anomalous samples that are irrelevant to the intent of the query. In our proposed regularization training process, our ideal outliers are the *unnatural natural* samples due to noise, different production methods and different production time. *Unnatural* means that the objects' pose or color in the image is not natural, resulting in an unnatural sample distribution. *Natural* represents a natural production method, compared to the adversarial samples. The outlier detection during implicit regularization training process is shown in Fig. 1b. The anomalous samples are strangely shaped and illegible dogs. For this problem, most methods [36]–[38] explicitly or implicitly assumed that the positive samples are in the dense area while the outliers are not, which have the same foundation with anomalous sample detection in the implicit regularization process.

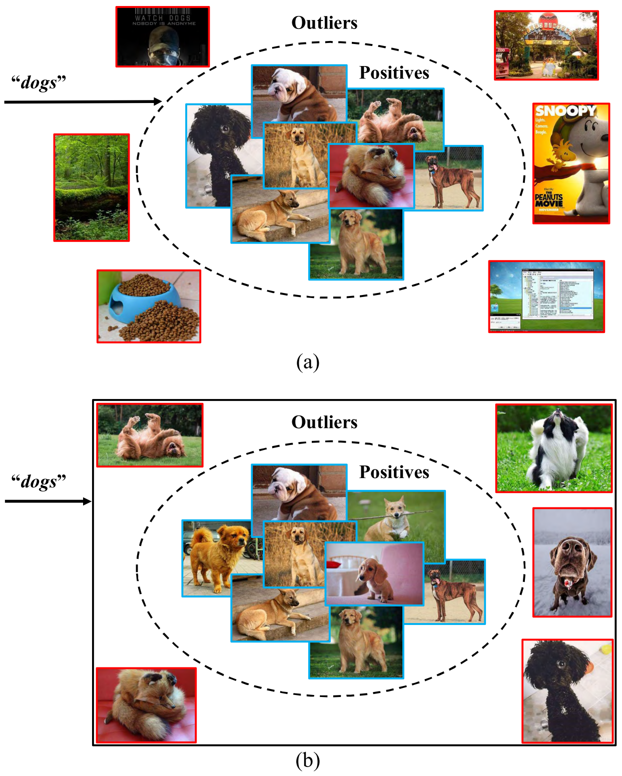


FIGURE 1. Example samples returned by an image search engine when querying "dog". The outliers shown in (a) are dog-independent samples while the outliers shown in (b) are dog-related but abnormal samples.

The reconstruction errors [31] of autoencoder are good indicator of outlier detection. When data are reconstructed from low-dimensional representations, the positives and the outliers can be well separated according to the reconstruction errors. Xia *et al.* [30] made the positive samples and outliers more separable by adding discriminative information in the training process of autoencoder. The method in [32] and [42] calculated the PCA projection of the data, and the sample with large projection variance is determined as the outlier. Ju *et al.* [33] designed an l_1-norm based probabilistic PCA model, in which the introduced hidden variables in the superposition can be used as a valid indicator for outlier detection. Rahmani and Atia [34] explored two randomized designs of robust PCA using low dimensional sample sketching. The statistical methods [35] fitted the parameter distribution over the training data, and the outliers are identified as those with low

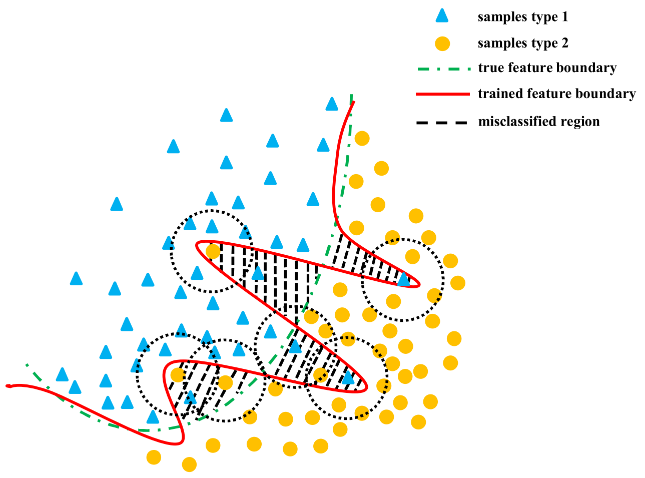


FIGURE 2. The decision boundary of a slightly overfitted deep CNN model and the sample distribution.

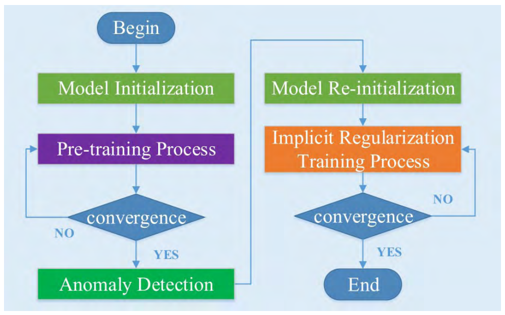


FIGURE 3. The two-stage training process of deep CNN.

Algorithm 1 Pre-Training Process

```
Input: $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$.
Initialization: network model $\mathbb{M} : \mathbf{f}(\mathbf{x}; \boldsymbol{\omega}) \in \mathbb{R}^C$,
    learning rate $\alpha_t$, momentum $\psi$, mini-batch size $M$.
    for each mini-batch $D_t = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^{M}$ do
        for each sample $(\mathbf{x}_m, \mathbf{y}_m)$ do
            forward-propagation:
            compute the output of layer $i$:
            $a^i = \sigma((\omega^i)^{\mathrm{T}} a^{i-1} + b^i)$;
            compute the loss function: $l(\mathbf{x}, \mathbf{y})$;
        end for
        back-propagation:
        $V_t = \psi V_{t-1} + (1-\psi) \cdot \frac{1}{|D_t|} \sum_{(\mathbf{x},\mathbf{y}) \in D_t} \nabla_{\boldsymbol{\omega}_t} [l(\mathbf{x}, \mathbf{y})]$;
        $\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \alpha_t V_t$;
    end for
Output: Updated model $\mathbb{M}' : \mathbf{f}(\mathbf{x}; \boldsymbol{\omega}') \in \mathbb{R}^C$.
```

probability under the learned distribution. Neighbor-based methods [36], [43] assumed that positive samples have close neighbors while outliers are far apart from each other. One-class SVM method [39] learned a maximum margin classifier to arbitrate outliers. Liu *et al.* [40] proposed an unsupervised one-class learning (UOCL) method that is superior to the above methods. With the inspiration of Gao and Li [41], we integrate the outlier detection into the image classification process instead of dealing with it independently. However, we do not need to make outlier judgment on the data in the testing set. We will further compare the influence of different outlier detection methods in the experimental section.

III. TWO-STAGE TRAINING METHOD

A. THE HYPOTHESIS AND IDEA

We first empirically describe the hypothesis and ideas of two-stage training method for improving generalization ability of deep CNN.

Hypothesis: a fully trained, slightly overfitted network model and sample distribution are shown in Fig. 2. The circle and triangle represent image representations of two kinds of samples, respectively. The dotted green line and the solid red line represent the real and the trained decision boundaries, respectively. The black shadow area is the misclassified area of the trained model on the testing set, which represents the generalization error. Theoretically, the larger the shadow area is, the higher the generalization error and the lower the classification accuracy of the testing set will be.

Idea: the feature boundary of the model can be adjusted by punishing the anomalous samples in the shadow area, which can regularize the network and reduce the overfitting. So the key solution is how to effectively detect the anomalous samples and establish an appropriate punishment mechanism. However, detection of anomalous samples depends on the image representation ability of the network, and the image representation ability of the network is adversely affected by anomalous samples. This looks like a chicken-and-egg problem. Actually, we can use a two-stage training process to detect anomalous samples and optimize the feature boundary.

Our proposed two-stage training method for decreasing overfitting problem of deep CNN consists of the following steps. The first stage is a pre-training process for determining sample distribution. The second stage is the implicit regularization process for optimizing the feature boundary. The procedure of two-stage training method is shown in Fig. 3. In this part, we only give the details of the training method; the specific network structure and hyper-parameters are given in the experimental setup section.

B. PRE-TRAINING PROCESS

The pre-training process is carried out on the whole training dataset, using mini-batch stochastic gradient descent (SGD) algorithm with momentum [44]. The training set is defined as $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$ where the sample is a D-dimensional vector $\mathbf{x}_n \in \mathbb{R}^D$, and its label is a C-dimensional vector $\mathbf{y}_n = (0, \cdots, 0, 1, 0, \cdots, 0)^T$ with the input of the correspond-ing category being 1 and the rest being 0. The goal is to train an initialized CNN model $\mathbb{M} : \mathbf{f}(\mathbf{x}; \boldsymbol{\omega}) \in \mathbb{R}^C$ to a convergent one $\mathbb{M}' : \mathbf{f}(\mathbf{x}; \boldsymbol{\omega}') \in \mathbb{R}^C$ for extracting the image representation, where $\boldsymbol{\omega}$ represents the weights. In order to observe the impact of different initialization on data distribution and network generalization ability, we compare the classification accuracy of different initialized network in experiments.

In the pre-training process, the t-th iteration of mini-batch SGD updates the current weighs $\boldsymbol{\omega}_t$ as:

$$V_t = \psi V_{t-1} + (1-\psi)\cdot \frac{1}{|D_t|}\sum_{(\mathbf{x},\mathbf{y})\in D_t} \nabla_{\boldsymbol{\omega}_t}\left[l(\mathbf{x},\mathbf{y})\right] \quad (1)$$

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \alpha_t V_t \quad (2)$$

where $l(\mathbf{x},\mathbf{y})$ represents the loss function, *e.g.*, cross-entropy function or log-likelihood function. $\nabla_{\boldsymbol{\omega}_t}\left[l(\mathbf{x},\mathbf{y})\right]$ is calculated by gradient back-propagation. D_t represents a mini-batch that randomly extracted from the whole training set D. α_t is the learning rate and ψ is the momentum. We give the pseudo codes of the pre-training process as shown in Algorithm 1 to facilitate the readers' understanding of the algorithm. Finally, we output the updated CNN model $\mathbb{M}' : \mathbf{f}(\mathbf{x};\boldsymbol{\omega}') \in \mathbb{R}^C$.

In order to make the pre-training network obtain accurate image representation ability as much as possible, we use data augmentation method in the image preprocessing stage, and we expand the training set by flipping, translating and scaling without using noise disturbance, color transform and contrast transform.

C. ANOMALY DETECTION

In this part, we introduce a simple density based anomaly detection method. The underlying assumption of density based anomaly detection method is that the density around positive samples is similar to the density of their neighbors, while the density around outliers is significantly different from them. In the big data environment, the dataset presents a more complex structure, and the sample may be regarded as outlier considering its local neighborhood rather than the whole dataset. Therefore, the density based anomaly detection is more appropriate than the distance based anomaly detection.

1) DENSITY BASED ANOMALY DETECTION ALGORITHM

For a subset $D' \in D$ under the same label, the k-distance of sample $\boldsymbol{x}$ is defined as $d_k(\boldsymbol{x})$, which is the distance $d(\boldsymbol{x},\boldsymbol{p})$ from the sample $\boldsymbol{x}$ and another sample $\boldsymbol{p}$. It satisfies:

- There are at least k samples $\boldsymbol{x}' \in D'$-$\{\boldsymbol{x}\}$ make $d(\boldsymbol{x},\boldsymbol{x}') \leq d(\boldsymbol{x},\boldsymbol{p})$.
- There are at most k-1 samples $\boldsymbol{x}'' \in D'$-$\{\boldsymbol{x}\}$ make $d(\boldsymbol{x},\boldsymbol{x}'') < d(\boldsymbol{x},\boldsymbol{p})$.

In this paper, we propose to use the cosine similarity between feature vectors to express the distance between them, as shown in equation 3.

$$d(\boldsymbol{x},\boldsymbol{p}) = \frac{\langle\boldsymbol{\theta}_1,\boldsymbol{\theta}_2\rangle}{||\boldsymbol{\theta}_1||\,||\boldsymbol{\theta}_2||} \quad (3)$$

where $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ represent the DeCAF feature vectors of samples $\boldsymbol{x}$ and $\boldsymbol{p}$, respectively. Note that the cosine similarity is equivalent to the Pearson correlation for mean-centered normalized vectors. The Euclidean distance between two samples in high-dimensional feature space can no longer reflect the actual relationship between them. As the feature dimension increases, the distance between samples can be severely affected by noise, so the data in high-dimensional spaces are often sparse. Empirically, principal components with lower variance in feature vectors are preferable; because in these dimensions, positive samples may be closer to each other, while outliers usually deviate from the most.

This means that $d_k(\boldsymbol{x})$ is the distance between sample $\boldsymbol{x}$ and its k-th nearest neighbor. Therefore, the k-distance neighbor of $\boldsymbol{x}$ contains all samples whose distance to $\boldsymbol{x}$ is no more than $d_k(\boldsymbol{x})$, which is denoted as

$$N_k(\boldsymbol{x}) = \left\{\boldsymbol{x}' | \boldsymbol{x}' \in D, d(\boldsymbol{x},\boldsymbol{x}') \leq d_k(\boldsymbol{x})\right\} \quad (4)$$

There may be more than k samples in $N_k(\boldsymbol{x})$, because there may be multiple samples at the same distance from $\boldsymbol{x}$.

For two samples $\boldsymbol{x}$ and $\boldsymbol{x}'$, if $d(\boldsymbol{x},\boldsymbol{x}') > d_k(\boldsymbol{x})$, then the reachable distance $rd_k(\boldsymbol{x} \leftarrow \boldsymbol{x}')$ from $\boldsymbol{x}'$ to $\boldsymbol{x}$ is $d(\boldsymbol{x},\boldsymbol{x}')$, otherwise it is $d_k(\boldsymbol{x})$. *i.e.*,

$$rd_k(\boldsymbol{x} \leftarrow \boldsymbol{x}') = \max\left\{d_k(\boldsymbol{x}), d(\boldsymbol{x},\boldsymbol{x}')\right\} + \epsilon \quad (5)$$

where k is a hyper-parameter that is used to control smoothness. ϵ is a very small constant (*e.g.*, $\epsilon = 0.001$) that used to avoid reachable distance being equal to zero. In general, the reachable distance is asymmetric, *i.e.*, $rd_k(\boldsymbol{x}\leftarrow\boldsymbol{x}') \neq rd_k(\boldsymbol{x}' \leftarrow\boldsymbol{x})$.

Then the locally reachable density $lrd_k(\boldsymbol{x})$ of the sample $\boldsymbol{x}$ can be defined as

$$lrd_k(\boldsymbol{x}) = \frac{||N_k(\boldsymbol{x})||}{\sum\limits_{\boldsymbol{x}'\in N_k(\boldsymbol{x})} rd_k(\boldsymbol{x}' \leftarrow \boldsymbol{x})} \quad (6)$$

So the anomalous degree $\mu_k(\boldsymbol{x})$ of the sample $\boldsymbol{x}$ is

$$\mu_k(\boldsymbol{x}) = \frac{\sum\limits_{\boldsymbol{x}'\in N_k(\boldsymbol{x})} \frac{lrd_k(\boldsymbol{x}')}{lrd_k(\boldsymbol{x})}}{||N_k(\boldsymbol{x})||} \quad (7)$$

Then we perform min-max normalization operation on the anomalous degree $\mu_k(\boldsymbol{x})$ of the sample $\boldsymbol{x}$ in each class as equation 8, which is linear transformations of initial data, and results are mapped to (0, 1).

$$\mu_k(\boldsymbol{x}) = \frac{1.05\mu_k(\boldsymbol{x}) - \mu_{\min}}{1.1\mu_{\max} - \mu_{\min}} \in (0,1) \quad (8)$$

where $\mu_{\min}$ and $\mu_{\max}$ are the smallest and largest anomalous degree, respectively. The anomalous degree reflects the average ratio of the reachable density of sample $\boldsymbol{x}$ to the reachable density of the k-nearest neighbor of $\boldsymbol{x}$. *i.e.*, it reflects the uniqueness of sample in the training set. The more unique a sample is, the higher the anomalous degree will be. The algorithm complexity is $O(dnk + nk\log(n))$, in which n represents the size of training set, d is the dimension of image representation, and k is the number of neighbors.

Anomaly detection mainly consists of four stages: data transmission, distance calculation, k-nearest neighbor search and calculation of anomalous degree. Most of the anomaly detection time is spent in the distance calculation and the k-nearest neighbor search stage, which accounts for more

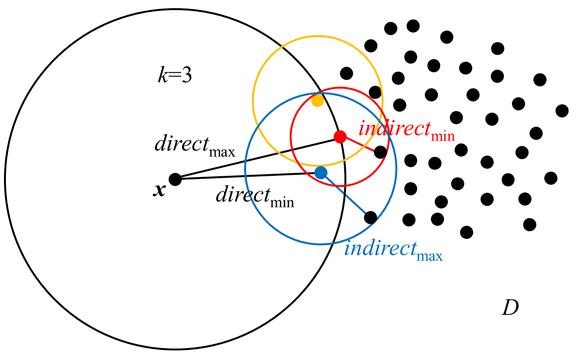


FIGURE 4. An example used to illustrate the property of density-based anomaly detection algorithm.

than 90% of the total execution time. So we calculate the distance in parallel using heterogeneous platforms based on CPU and GPU, and use cell method [45] to accelerate the k-nearest neighbor search process. In fact, we also use the block vector approximation algorithm [46] to test Euclidean distance based anomaly detection method, but the classification results are unsatisfactory.

Finally, we do not need to set the threshold to judge whether a sample is anomalous or not. We just need to output the anomalous degree of each sample for the implicit regularization training stage. *i.e.*, the dataset $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ is updated to $D = \{(\mathbf{x}_n, \mathbf{y}_n, \mu_n)\}_{n=1}^N$. In experiments, we will compare the impact of different outlier detection algorithms in the implicit regularization training process.

2) PROOF OF ALGORITHM PROPERTY

We use an example to analyze the property of density-based anomaly detection algorithm. For a sample $\boldsymbol{x}$ as shown in Fig. 4, let

$$direct_{\min}(\boldsymbol{x}) = \min\left\{rd_k(\boldsymbol{x}' \leftarrow \boldsymbol{x})|\boldsymbol{x}' \in N_k(\boldsymbol{x})\right\} \tag{9}$$

which represents the smallest reachable distance between sample $\boldsymbol{x}$ and its k-nearest neighbor. Similarly, let

$$direct_{\max}(\boldsymbol{x}) = \max\left\{rd_k(\boldsymbol{x}' \leftarrow \boldsymbol{x})|\boldsymbol{x}' \in N_k(\boldsymbol{x})\right\} \tag{10}$$

Then, consider the k-nearest neighbor of sample $\boldsymbol{x}$, let

$$\begin{aligned} &indirect_{\min}(\boldsymbol{x}) \\ &\quad = \min\left\{rd_k(\boldsymbol{x}' \leftarrow \boldsymbol{x})|\boldsymbol{x}' \in N_k(\boldsymbol{x}) \text{ and } \boldsymbol{x}'' \in N_k(\boldsymbol{x}')\right\} \quad (11) \\ &indirect_{\max}(\boldsymbol{x}) \\ &\quad = \max\left\{rd_k(\boldsymbol{x}' \leftarrow \boldsymbol{x})|\boldsymbol{x}' \in N_k(\boldsymbol{x}) \text{ and } \boldsymbol{x}'' \in N_k(\boldsymbol{x}')\right\} \quad (12) \end{aligned}$$

Then we can prove

$$\frac{direct_{\min}(\boldsymbol{x})}{indirect_{\max}(\boldsymbol{x})} \leq \mu(\boldsymbol{x}) \leq \frac{direct_{\max}(\boldsymbol{x})}{indirect_{\min}(\boldsymbol{x})} \tag{13}$$

This means that the anomalous degree $\mu(\boldsymbol{x})$ captures the relative density of sample $\boldsymbol{x}$. The lower the relative density is, the higher the anomalous degree will be.

Algorithm 2 Implicit Regularization Training Process

```
Input: $D = \{(\mathbf{x}_n, \mathbf{y}_n, \mu_n)\}_{n=1}^N$
Initialization: network model $\mathbb{M} : \mathbf{f}(\mathbf{x}; \boldsymbol{\omega}) \in \mathbb{R}^C$,
learning rate $\alpha_t$, momentum $\psi$, mini-batch size $M$.
  for each mini-batch $D_t = \{(\mathbf{x}_m, \mathbf{y}_m, \mu_m)\}_{m=1}^M$ do
    for each sample $(\mathbf{x}_m, \mathbf{y}_m, \mu_m)$ do
      forward-propagation:
      compute the output of layer $i$:
      $a^i = \sigma((\boldsymbol{\omega}^i)^{\mathrm{T}} a^{i-1} + b^i)$;
      compute the loss function :$l(\mathbf{x}, \mathbf{y})$;
      compute regularization factor:
      $\eta_n = \left[U\left(1 - \frac{\log \mu_n}{\beta \log U}\right)\right] * \varepsilon(\mu_n - U)$;
      update loss function:
      $l_n^\dagger = (1 - \eta_n) l_n$;
    end for
    back-propagation:
    $V_t = \psi V_{t-1} + (1 - \psi) \cdot \frac{1}{|D_t|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D_t} \nabla_{\boldsymbol{\omega}_t} \left[l^\dagger(\boldsymbol{x}, \boldsymbol{y})\right]$;
    $\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \alpha_t V_t$;
  end for
Output: Updated model $\mathbb{M}'' : \mathbf{f}(\mathbf{x}; \boldsymbol{\omega}'') \in \mathbb{R}^C$.
```

D. IMPLICIT REGULARIZATION PROCESS

Based on the idea of transfer learning, we re-initialize the parameters of fully connected layers and some high-level convolution layers and start a new training process. This process is considered as an implicit regularization work.

1) REGULARIZATION METHOD

We first set a threshold $U \in (0, 1)$ to prevent the case that the network cannot be effectively optimized due to the excessive punishment of samples. It determines whether a sample will be punished during the training process, and participates in controlling the intensity of punishment. Then the regularization factor can be defined as

$$\eta_n = \left[U\left(1 - \frac{\log \mu_n}{\beta \log U}\right)\right] * \varepsilon(\mu_n - U) \tag{14}$$

where ε and $*$ represent the step function and the convolution operation, respectively. β is a manually defined weighting factor that determines subjective regularization intensity. Experiments showed that the best effect was obtained when β belongs to [4, 7].

Then the updated loss function is

$$l_n^\dagger = (1 - \eta_n) l_n \tag{15}$$

where l_n is the original loss of sample $(\mathbf{x}_n, \mathbf{y}_n)$. In this step, the forward propagation of the implicit regularization training process is completed. Then we continue using the mini-batch SGD algorithm to train the network to converge. The pseudo code for this training stage is shown in Algorithm 2. The convergent model is used to complete the classification of the testing set and to observe the generalization performance.

Some studies [47] show that the mini-batch SGD algorithm itself has an implicit regularization effect on the network. We think that mini-batch SGD algorithm is a regularization method at the data level. Most normal data in a mini-batch decide the descent direction of gradient. Our method is to improve the regularization effect of mini-batch SGD training method through data intervention.

2) ANALYSIS OF ALGORITHM PROPERTY

We analyze the properties of regularization factor by mathematical derivation and function curves. The graph of regularization factor η is shown in Fig. 5a. The partial derivatives of regularization factor η to anomalous degree μ is

$$\frac{\partial \eta}{\partial \mu} = -\frac{1}{\beta \log U} \cdot \frac{1}{\mu} > 0 \tag{16}$$

The partial derivative of loss function $l_n^{\dagger}$ to anomalous degree μ is

$$\frac{\partial l_n^{\dagger}}{\partial \mu} = \frac{l_n}{\beta \log U} \cdot \frac{1}{\mu} < 0 \tag{17}$$

In each iteration, the expectation of regularization factor is

$$\begin{aligned} \mathrm{E}_{D_t}(\eta) &= \frac{\int_U^1 U\left(1 - \frac{\log \mu}{\beta \log U}\right) d\mu}{1-U} \\ &= U + \frac{U}{\beta \log U} + \frac{U^2}{\beta(1-U)} \end{aligned} \tag{18}$$

$$\frac{\partial \mathrm{E}_{D_t}(\eta)}{\partial U} = 1 + \frac{\log U - 1}{\beta \log^2 U} + \frac{U(2-U)}{\beta(1-U)^2} > 0 \tag{19}$$

The graph of equation 16 and 19 are shown in Fig. 5b and Fig. 5c, respectively. We can observe that the regularization factor has the following properties:

1.If keep U constant (-) and make μ increases ($\uparrow$), then we can get $\eta \uparrow$, $l_n^{\dagger} \downarrow$. *i.e.*, U-, $\mu \uparrow$, β- $\Rightarrow \eta \uparrow$, $l_n^{\dagger} \downarrow$.

2. μ-, $U \uparrow$, β- $\Rightarrow \mathrm{E}_{D_t}(\eta) \uparrow$, $\mathrm{E}_{D_t}(l^{\dagger}) \downarrow$.

3. μ-, U-, $\beta \uparrow \Rightarrow \eta \uparrow$, $l_n^{\dagger} \downarrow$.

4. $U \approx 0 \Rightarrow \eta \approx 0 \Rightarrow l_n^{\dagger} \approx l_n$.

The first property ensures that the higher the anomalous degree of a sample is, the smaller role it will play in the training process. The second property allows us to control the number of punished samples by increasing the threshold. As the threshold U becomes higher, the number of anomalous samples will decrease; the average punishment on a subset D_t will be more severe; and remaining anomalous samples will play a smaller role in the training process. Anomalous degree μ and threshold U are a pair of trade-off parameters, which can avoid the adverse effect caused by improper setting of hyper-parameters. And the threshold prevents the network from overfitting to normal samples. The third property means that one can control the regularization intensity of training process by adjusting the value of β. The fourth property guarantees that even a minimal threshold does not cause the under-fitting problem. In this case, the loss function is almost unchanged. These properties guarantee the effectiveness of the implicit regularization training process.

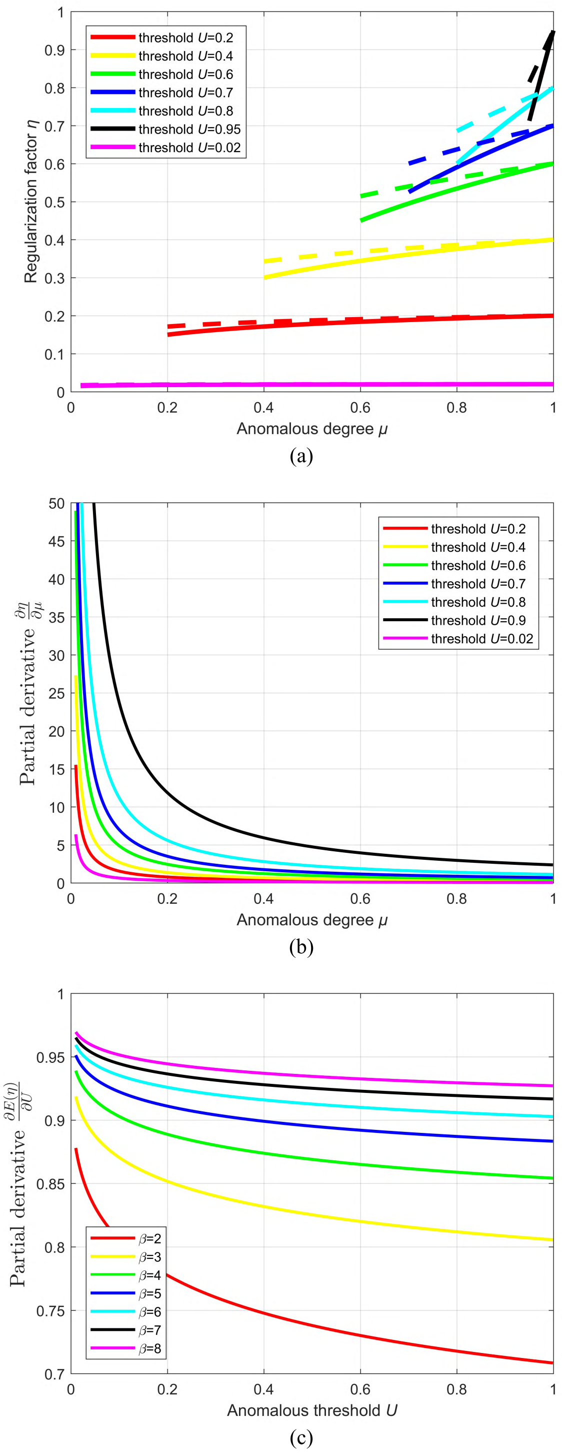


FIGURE 5. Graph (a) shows the relationship between regularization factor η, anomaly degree μ and threshold U under different β. The solid and the dotted lines represent the case of $\beta = 4$ and $\beta = 7$, respectively. (b) shows the partial derivative of loss function to anomalous degree μ under different U ($\beta = 4$). (c) shows the relationship between the expectation of anomalous degree with threshold U.

IV. EXPERIMENTAL SETTINGS AND RESULTS

The experimental process is displayed following the training steps and is organized as follows. In section *A*, we introduce the experimental datasets, the corresponding deep

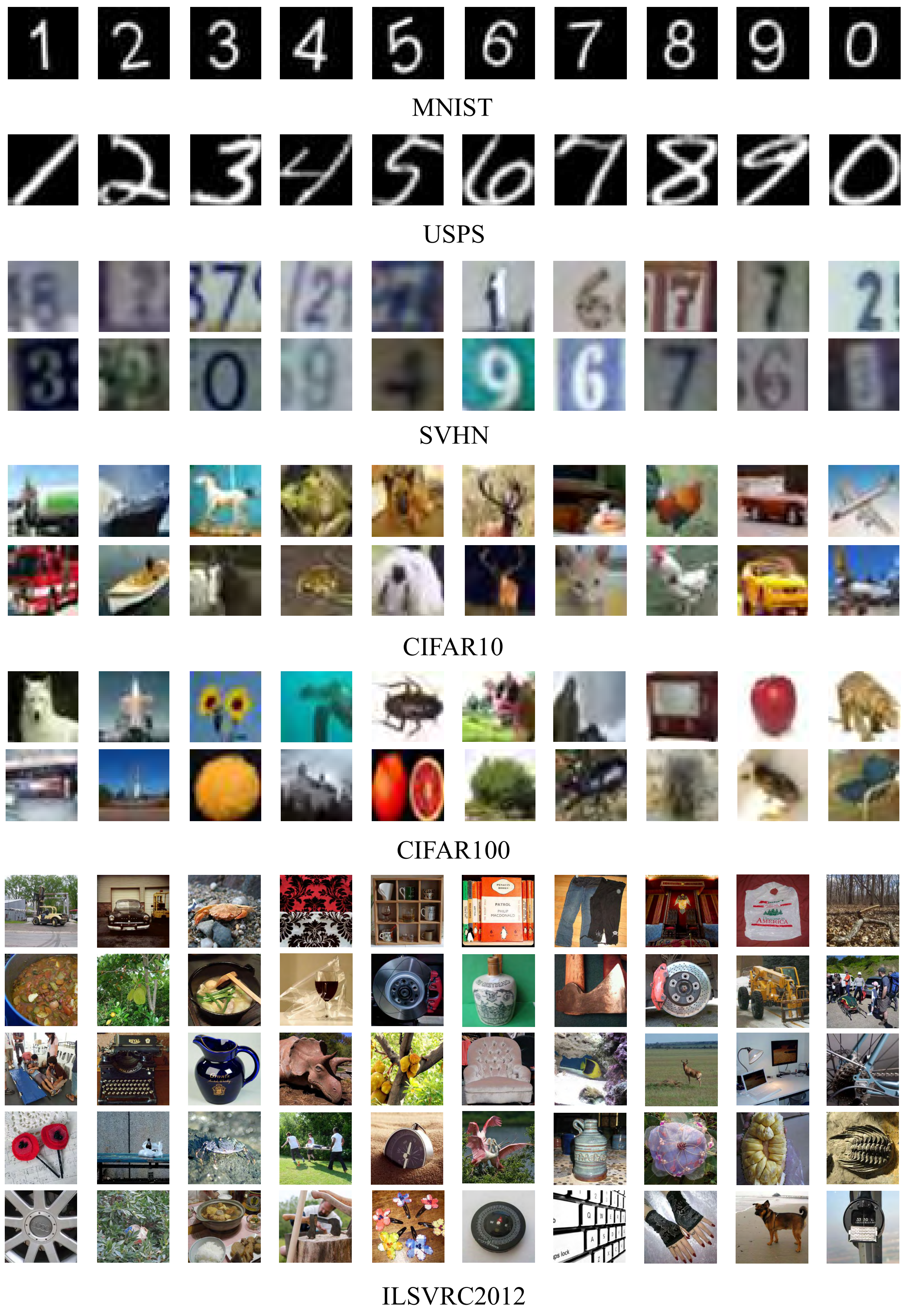


FIGURE 6. **Training samples of MNIST, USPS, SVHN, CIFAR10, CIFAR100, and ILSVRC2012.**

CNN structures and the parameters set. In section B, we compare the classification results of the testing sets of each dataset. Section C shows the influence of various outlier detection algorithms on samples and implicit regularization training process. Section D reports the effect of different initialization methods on the generalization of deep CNN during the two-stage training process. In section E, we verify the regularization effect of the proposed method by changing the structure of deep CNN. In section F, we use extensive experiments to test the sensitivity of the algorithm to

TABLE 2. The deep CNN architectures used in experiments.

Datasets	MNIST/ SVHN	USPS	CIFAR10/100	ILSVRC2012		
Architectures	FitNet-1 30K parameters	ALeNet 10K parameters	FitNet-4 2.5M parameters	Alexnet 60M parameters	VGG-16 130M parameters	NIN 7M parameters
	conv3-16 conv3-16 conv3-16 pool 2×2	conv5-32 pool 3×3	conv3-32 conv3-32 conv3-32 conv3-48 conv3-48 pool 2×2	conv11-96 pool 3×3 LRN 5×5	conv3-64 conv3-64 pool 2×2 conv3-128 conv3-128 pool 2×2	conv11-96 cccp1-96 cccp1-96 pool 3×3
	conv3-32 conv3-32 conv3-32 pool 2×2	conv5-32 pool 3×3	conv3-80 conv3-80 conv3-80 conv3-80 conv3-80 pool 2×2	conv5-256 pool 3×3 LRN 5×5	conv3-256 conv3-256 conv1-256 pool 2×2	conv5-256 cccp1-256 cccp1-256 pool 3×3
	conv3-48 conv3-48 conv3-64 pool 8×8 (global)	conv5-64 pool 3×3	conv3-128 conv3-128 conv3-128 conv3-128 conv3-128 pool 8×8 (global)	conv3-384 conv3-384 conv3-256 pool 3×3	conv3-512 conv3-512 conv1-512 pool 2×2	conv3-384 cccp1-384 cccp1-384 pool 3×3
	fc-500 softmax-10	fc-64 softmax-10	fc-500 softmax- 10(100)	fc-4096 fc-4096 softmax-1000	conv3-512 conv3-512 conv1-512 pool 2×2	conv3-1024 cccp1-1024 cccp1-1000 pool 6×6
					fc-4096 fc-4096 softmax-1000	softmax-1000

hyper-parameters. Finally, we visualize the implicit regularization training process on a toy dataset in section *G*.

A. DATASETS AND EXPERIMENT SETTINGS

We evaluate two-stage training method on six popular image classification datasets, *i.e.*, MNIST [6], SVHN [48] and US Postal Service (USPS) [49] for digit classification, CIFAR10 and CIFAR100 [5] for natural image classification, and ILSCRV2012 [50] for large-scale image classification.

MNIST is one of the most widely used datasets for digital classification, which contains 60000 training samples and 10000 testing samples, uniformly distributed on ten categories (0–9). All of them are 28×28 grayscale images. For data augmentation (DA), we randomly crop the training samples into 24×24 pixels.

SVHN is a larger collection of digital samples, *i.e.*, 73257 training images, 26032 testing images, and 531131 extra training images. All the samples are 32×32 RGB images. We preprocess The training data is preprocessed according to the method in [48], *i.e.*, 600 images per class from the training set and 400 images per class from the extra training set are taken out as validation set (including 10,000 samples), and the remaining 594388 images are used as training samples.

USPS dataset is one of the most popular datasets for handwritten digit classification and allows for rapid verification due to its small size. There are 7291 images for training, and 2007 images for testing. All the samples are 16×16 pixels and are scaled between 0 and 1. The dataset contains a large amount of image variability and is considered as a hard classification task. The difference in the size of each category is a bit notable. So we expand the training set to 21873 by using scale transformation.

CIFAR10 and *CIFAR100* are both subsets extracted from the 80-million tiny image dataset [51]. The datasets both consist of 50000 training samples and 10000 testing samples. And all the samples are 32×32 RGB images. CIFAR10 includes 10 basic classes and CIFAR100 divides each of them into a finer level (100 classes). In both datasets, training and testing samples are uniformly distributed across all classes.

ILSVRC2012 dataset [50] is used to evaluate the large-scale image classification performance of our method. It is a subset of the ImageNet dataset which includes 1000 classes. There are 1.3M images for training, 100K images for validation and 200K images for testing. We show some sample images of these six datasets in Fig. 6.

For the structures of deep CNN used in the experiments (see Table 2), we select the FitNet-1 architecture [52] for MNIST and SVHN dataset, and train it with the proposed two-stage method. Non-linearity in the network are set as follows: maxout with 2 linear units in convolution layers and maxout with 5 linear units in the fully connected layer. The initialization of network parameters is set by default (Hints).

TABLE 3. **Comparison with other methods on MNIST, SVHN and USPS.**

Number	MNIST	Classification accuracy (%) with DA/without DA	SVHN	Classification accuracy (%) with DA/without DA	USPS	Classification accuracy (%) with DA/without DA
1	Wan [10]	**99.79**/99.48	Wan	98.06/-	Wang [55]	-/97.96
2	RCNN [20]	-/99.69	RCNN	-/**98.23**	NMF[56]	-/98.56
3	NIN [21]	-/99.53	NIN	-/97.65	PCA[56]	-/98.46
4	Maxout [14]	-/99.55	Maxout	-/97.53	SELM-AE (sigmoid) [56]	-/98.77
5	HighWay-16 [54]	-/99.43	Wide ResNet [58]	**98.36**/- .	ELM-AE (linear) [56]	-/98.71
6	HighWay-32	-/99.55	CCANet [57]	98.21/-	CCANet	-/98.01
7	DSN-Softmax [62]	-/99.49	DSN-Softmax	-/97.87	PCANet[57]	-/98.11
8	DSN-SVM	-/99.61	DSN-SVM	-/98.08	Abe [59]	96.76/94.12
9	MIN [63]	-/**99.76**	Zeiler [12]	-/97.20	LMNN[60]	-/98.07
10	LeNet-Dropout	99.57/99.32	LeNet-Dropout	96.75/96.35	GB-LMNN[60]	-/97.69
11	LeNet-DisturbLabel	99.55/99.34	LeNet-DisturbLabel	96.73/96.31	LDS[61]	-/95.04
12	BigNet-Dropout [16]	99.71/99.64	BigNet-Dropout	97.92/97.77	BigNet-Dropout	95.40/95.32
13	BigNet-DisturbLabel	99.68/99.62	BigNet-DisturbLabel	97.79/97.72	BigNet-DisturbLabel	95.44/95.37
14	FitNet1	99.49/99.47	FitNet1	97.62/97.58	ALeNet	97.86/97.14
15	FitNet1-LSUV	-/99.52	FitNet1-LSUV	-/97.68	ALeNet-LSUV	98.17/97.26
16	FitNet1-LSUV-SVM	-/99.62	FitNet1-Teacher	-/97.62	ALeNet-LSUV-SVM	**98.94/98.85**
17	FitNet1-DisturbLabel	99.67/99.64	FitNet1-DisturbLabel	97.80/97.74	ALeNet-DisturbLabel	98.32/97.59
18	**FitNet1-Our method**	**99.80/99.78**	**FitNet1-Our method**	**98.39/98.28**	**ALeNet-Our method**	**98.96**/98.21
19	**GAN-FitNet1-Our method**	**99.82**/-	**GAN-FitNet1-Our method**	-/-	**GAN-ALeNet-Our method**	**98.99**/-

The initial learning rate is set to 0.001 and decreased by a factor of 10 after the 30th, 50th and 70th epoch. Another version of the LeNet (ALNet) [53] is used for the USPS dataset. The FitNet-4 network [52] that used for CIFAR10/100 is trained with the mini-batch SGD with momentum set to 0.6; the initial learning rate set to 0.01 and reduced by a factor of 10 after the 60th, 80th and 100th epoch, finishing at 120th epoch. Schmidhuber [54] trained the very deep CNN for 200 epochs. But too much training is more likely to lead to overfitting problem. For ILSVRC2012 classification task, we test the performance of the algorithm on three widely used network architectures: Alexnet, VGG-16 and Network in Network (NIN). The network parameters and the learning rate are set by default. The activation functions of all network use the Swish [8] due to its good experimental results and smoothness nature.

B. CLASSIFICATION RESULTS OF THE TESTING SET

1) NUMERICAL IMAGE CLASSIFICATION

The classification results of MNIST, SVHN and USPS datasets under different algorithms are shown in Table 3. In the following sections, to make the results easy to read, we boldface all results that are close to the best result on each dataset.

The two-stage training method achieved surprising results on FitNet1 and ALeNet. *i.e.*, 99.80% on MINST, 98.39% on SVHN and 98.96% on USPS with data augmentation, 99.78% on MINST, 98.28% on SVHN and 98.87% on USPS without data augmentation. Compared to the general training methods (Number 14 in Table 3), the classification accuracy of both three datasets has been improved. Compared with other regularization methods, the performance of implicit regularization training was able to outperform DisturbLabel (Number 17), and the classification accuracy of MNIST, SVHN and USPS increased by 0.13%, 0.59% and 0.64% (with data augmentation) and 0.14%, 0.54%, 0.62% (without data augmentation), respectively. Since our method does not conflict with dropout in the training process, we do not compare the classification results of these two regularization methods. Although the network adopts the default initial weights, the classification accuracy of the three datasets is still higher than that of the network with advanced LSUV initialization method (Number 15 and 16). Our performance even exceeds the classification results of some more advanced network structures, such as Highway network (Number 5 and 6), DSN (Number 7 and 8) and BigNet (Number 12 and 13).

As a reference, we compare the best classification results on three datasets (Number 1 and 9 in MNIST, Number 2 and 5

TABLE 4. Comparison with other methods on CIFAR10, CIFAR100 and ILSVRC2012.

Number	CIFAR10	Classification accuracy (%) with DA/without DA	CIFAR100	Classification accuracy (%) with DA/without DA	ILSVRC2012	Classification accuracy(%)
1	Wan	90.68/-	Srivastava [64]	-/63.15	ZFNet [2]	63.57
2	RCNN	92.91/91.31	RCNN	-/68.25	PReLU-net [7]	75.73
3	ALL-CNN [19]	92.75/-	ALL-CNN	66.29/-	BN-Inception	80.01
4	DSN	92.03/90.31	DSN	65.43/-	ResNet-152	82.44
5	NIN	91.19/89.59	NIN	64.32/-	Inception-V3	80.94
6	Maxout	90.62/88.32	maxout	65.46/61.43	GoogLeNet [11]	79.05
7	MIN	93.25/-	MIN	71.14/-	**Wide ResNet**	**82.96**
8	ELU [19]	93.45/-	ELU	75.72/-	Overfeat [2]	64.33
9	**Wide ResNet**	**95.63**/-	**Wide ResNet**	**76.60**/-	AlexNet [2]	61.18
10	LeNet-Dropout	85.76/80.58	LeNet-Dropout	58.72/50.92	AlexNet-DisturbLabel	62.17
11	LeNet-DisturbLabel	85.52/79.74	LeNet-DisturbLabel	58.16/48.17	**AlexNet-Our method**	**62.29**
12	BigNet-Dropout	92.92/88.77	BigNet-Dropout	72.95/66.70	VGGNet [21]	76.17
13	BigNet-DisturbLabel	92.07/90.18	BigNet-DisturbLabel	71.61/65.19	VGGNet-DisturbLabel	76.22
14	FitNet4	91.61/88.94	FitNet4	70.96/64.29	**VGGNet-Our method**	**76.35**
15	FitNet4-LSUV	93.94/-	FitNet4-LSUV	72.34/-	NIN	74.70
16	FitNet4-DisturbLabel	93.27/**91.61**	FitNet4-DisturbLabel	71.54/**68.96**	NIN-DisturbLabel	75.11
17	**FitNet4-Our method**	**95.69/93.28**	**FitNet4-Our method**	**76.64/71.88**	**NIN+Our method**	**75.32**

in SVHN and Number 16 in USPS) to illustrate the excellent performance of two-stage training method. For MNIST, the neural network using DropConnect method had the highest classification accuracy on the augmented dataset; MIN established a state-of-the-art on MNIST without data augmentation through a combination of maxout, batch normalization and NIN nonlinearities. We improved the classification results of the current state-of-the-art by 0.01% and 0.02%, respectively. Wide ResNet and RCNN achieved the best performance on SVHN dataset before and after data augmentation, respectively. We improved the classification results of the current state-of-the-art by 0.03% and 0.05%, respectively. For USPS dataset, features of images are extracted by the ALeNet structure with LSUV initialization method and are classified by SVM, and the best results are 98.94% (with DA) and 98.85% (without DA). Our classification results are also very competitive.

We even adopted an extreme data augmentation method that generated a number of digital samples on MNIST and USPS using a single Generative Adversarial Net (GAN). From the classification accuracy of the testing set (Number 19), the addition of generated samples further enhanced the generalization ability of the network though implicit regularization training process and increased the classification results from 99.80%/98.96% to 99.82%/98.99%. The experimental results show that the implicit regularization stage makes the data augmentation more efficient, in which the anomaly detection is crucial to the effective use of generated samples. In part 3, we will analyze the importance of anomaly detection.

2) NATURAL IMAGE CLASSIFICATION

The classification results of CIFAR10, CIFAR100 and ILSVRC2012 datasets under different algorithms are shown in Table 4.

In CIFAR10 and CIFAR100, the FitNet4 structure achieved a new state-of-the-art under two-stage training method. It improved the classification accuracy before and after data augmentation from 91.61%/95.63% and 68.96%/76.60% to 93.28%/95.69% and 71.88%/76.64%, respectively. Compared to the past training methods (Number 14 in Table 4), the classification accuracy under two-stage training method on both datasets was improved. Compared with other regularization methods, the performance of implicit regularization training was able to outperform DisturbLabel (Number 16), and the classification accuracy of CIFAR10 and CIFAR100 increased by 2.42%, 5.10% (with DA) and 1.67%, 2.92% (without DA), respectively. The performance of natural image datasets also exceeds some more advanced network structures, such as Wide ResNet (Number 5), maxout network (Number 6) and MIN (Number 7).

The classification results on ILSVRC2012 dataset are also quite competitive. The classification accuracy of Alexnet, VGG-16 and NIN are 62.29%, 76.35% and 75.32%, respectively. Compared with traditional training methods, their results were increased by 1.11%, 0.18% and 0.62%, respectively. However, we found that the classification accuracy of three networks on ILSVRC2012 dataset was only marginally improved compared with the improvement of classification results of CIFAR10 and CIFAR100. We believe that the large noisy area in the image and the complex shape

TABLE 5. **Classification results of different regularization methods with or without anomaly detection.**

Number	Regularization methods	Classification accuracy (%)					
		MNIST	SVHN	USPS	CIFAR10	CIFAR100	ILSVRC2012
1	RandomDisturbLabel	99.67	97.80	98.32	93.27	71.54	76.22
2	Anomaly detection+DisturbLabel	99.74	97.84	98.34	93.41	74.02	76.19
3	Random SoftLabel	99.62	97.72	98.34	93.19	71.22	76.20
4	Anomaly detection +SoftLabel	99.70	97.76	98.35	93.32	74.89	76.22
5	Random sample+ Implicit regularization	99.63	97.82	98.32	93.25	74.79	76.22
6	**Anomaly detection +Implicit regularization**	**99.80**	**98.39**	**98.96**	**95.69**	**76.64**	**76.23**
7	Anomaly detection +Sample dropout	98.07	95.12	96.81	89.52	67.96	70.04

of objects have a bad influence on the anomaly detection in high-dimensional space, which affects the optimization of feature boundary. Actually, the accuracy improvements on ILSVRC2012 are remarkable given that one year of architectural tuning and enlarging yielded 1.89% accuracy gain going from GoogLeNet (Number 6) to Inception-V3 (Number 5). It can be seen that under Alexnet (Number 10 and 11), VGG-16 (Number 13 and 14) and NIN (Number 16 and 17), the two-stage training method has a similar classification accuracy with the DisturbLabel method. This confirms our guess that when the anomaly detection is no longer valid, the implicit regularization training process is equivalent to punishing samples randomly, which is also equivalent to randomly adding noise to sample labels. Although our method did not achieve state-of-the-art on ILSVRC2012, better classification results can be obtained if we use the two-stage training method on a more advanced network structure.

For these three natural image datasets, the simple GAN is difficult to converge due to the complex background and object structure, so we did not test the influence of extreme data augmentation.

3) IMPACT OF ANOMALY DETECTION

The classification results of six datasets demonstrate the effectiveness of proposed two-stage training method for improving the generalization ability of deep CNNs, especially the implicit regularization training process. In this part, we report the contribution of anomaly detection to the implicit regularization process through experiments. In the implicit regularization training process, we adopt two ways of random selection and anomaly detection to select the punished samples and compare the three punishment methods, including DisturbLabel, SoftLabel and our proposed regularization factor.

For the DisturbLabel method, a disturbed label $\widetilde{\mathbf{y}} = (\widetilde{y}_1, \widetilde{y}_2, \cdots, \widetilde{y}_C)^{\boldsymbol{T}}$ is randomly generated for each sample $(\mathbf{x}, \mathbf{y})$ from a Categorical distribution (a genera-lization of the Bernoulli distribution) $P(\alpha)$:

$$\begin{cases} \widetilde{r} \sim P(\alpha), \\ \widetilde{y}_{\widetilde{r}} = 1, \\ \widetilde{y}_i = 0, \quad \forall i \neq \widetilde{r} \end{cases} \tag{20}$$

The Categorical distribution $P(\alpha)$ is defined as $P_r = 1 - \frac{C-1}{C} \cdot \alpha$ and $P_i = \frac{1}{C} \cdot \alpha$ for$i \neq r$, in which α represents the noise level and r represents the true label (*i.e.*, in the ground-truth label $\mathbf{y}$ of sample $\mathbf{x}$, $y_r = 1$). It is equivalent to disturb the label of each training sample with a probability α. In the case of random selection, the noise level is set to 20%. In the case of anomaly detection, each sample has a different noise level, which equals to their anomalous degree. In other words, samples with higher anomalous degree are given a disturbed label with higher probability.

For the SoftLabel method, a random label $\mathbf{z} = (z_1, z_2, \ldots, z_C)^{\mathrm{T}}$ is generated for each of the punished samples from a uniform distribution:

$$\begin{cases} z_i \sim U(0,1) \\ \mathbf{z} = (z_1, z_2, \ldots, z_C)^{\mathrm{T}} \end{cases} \tag{21}$$

Then the label is normalized as

$$\mathbf{z} = \frac{\mathbf{z}}{||\mathbf{z}||} \tag{22}$$

Finally, the soft label $\widetilde{\widetilde{\mathbf{y}}} = (\widetilde{\widetilde{y}}_1, \widetilde{\widetilde{y}}_2, \cdots, \widetilde{\widetilde{y}}_C)^{\boldsymbol{T}}$ of sample $(\mathbf{x}, \mathbf{y})$ is generated by

$$\widetilde{\widetilde{\mathbf{y}}} = (1-\alpha)\mathbf{y} + \alpha\mathbf{z} \tag{23}$$

In the case of random selection, the noise level is set to 20%. In the case of anomaly detection, each sample has a different noise level, which equals to their corresponding anomalous degree. In other words, samples with higher anomalous degree are disturbed by more severe noise.

In view of our proposed regularization factor, in the case of random selection, 20% samples are randomly selected to be outliers and the regularization factor is 0.2. In the case of anomaly detection, the processing is referred to the above description. Finally, we consider an extreme regularization operation that dropouts all the anomalous samples according to the threshold U, *i.e.*, they have no contribution to the training of network. This operation can be achieved by replacing the regularization factor by

$$\eta_n = \begin{cases} 1, & \mu_n \geq U \\ 0, & \mu_n < U \end{cases} \tag{24}$$

The classification results of five datasets are shown in Table 5. Comparing the experimental results of 1 and 2, 3 and 4, 5 and 6, we can see that the anomaly detection improves the classification accuracy of all the regularization methods on MNIST, SVHN, USPS, CIFAR10 and CIFAR100. This validates our hypothesis that anomaly

TABLE 6. Classification results of six datasets under different anomaly detection algorithms.

Methods	Classification accuracy (%)					
	MNIST	SVHN	USPS	CIFAR10	CIFAR100	ILSVRC2012
Ju [33]	99.63	97.90	98.57	93.29	73.53	75.51
Liu [40]	99.63	97.92	98.43	93.44	73.19	75.92
Rahmani [34]	99.60	98.04	98.52	93.78	74.25	76.39
Linear PCA [32]	99.68	98.02	98.60	94.44	75.86	76.11
Kernel PCA	99.66	98.17	98.69	94.41	76.01	76.14
Robust kernel PCA	99.71	98.23	98.74	95.53	76.22	76.20
Autoencoder [31]	99.77	98.37	98.92	95.72	76.63	76.94
Boost-autoencoder	99.74	**98.44**	98.95	95.74	77.11	**76.96**
OC-SVM [39]	99.68	**98.44**	98.94	95.68	77.07	-
UOCL [40]	99.69	98.42	98.90	**95.77**	**77.13**	-
Our Method	**99.80**	98.39	**98.96**	95.69	76.64	76.23

detection favors the optimization of feature boundary, *i.e.*, the sample distribution is adjusted and the decision boundary is optimized. Moreover, we find that the extreme regularization operation (Number 7), *i.e.*, sample dropout, has an adverse impact on the generalization ability of the network, resulting in a significant decrease in the classification accuracy on both six datasets. We think there are two possible reasons for the poor performance of sample dropout. The first is that the feature boundary is overfitted to normal samples, leading to a lack of generalization ability. In other words, the trained network is not robust enough. Another reason is that sample dropout makes the network underfit to the real sample distribution. Comparing all the results of the six datasets, the implicit training method based on anomaly detection outperforms DisturbLabel and SoftLabel method. The regularization way of our algorithm is similar to l_2-regularization, but the difference is tht the l_2-regularization aims at the network structure.

Experiments with ILSCVRC2012 are performed on the VGG-16 network. As mentioned above, the anomaly detection method is invalidated on ILSVRC2012 dataset, and the result of anomaly detection is equivalent to random sampling. The classification accuracy of ILSVRC2012 in each regularization method is similar, which confirms our inference.

C. COMPARISON OF DIFFERENT ANOMALY DETECTION ALGORITHMS

Aiming at the failure problem of anomaly detection in high dimensional space, we compare some more advanced and of course more complicated anomaly detection algorithms.

The first thing we need to do is to normalize the anomalous degree obtained by each anomaly detection algorithm for the implicit regularization process. Take autoencoder as an example, the anomalous degree is defined as the reconstruction error according to

$$\mu_n = ||x_n - f(x_n)||^2 \tag{25}$$

where x_n represent the DeCAF feature vector, $f(x_n)$ is the reconstructed copy. Then the anomalous degree is normalized by Equation 8, and the updated samples are then fed into the network for implicit regularization training.

The classification results based on the various anomaly detection algorithms are shown in Table 6. On SVHN and ILSVRC2012 datasets, the anomaly detection algorithm based on the discriminative autoencoder (Boost-autoencoder) achieves the classification accuracy of 98.44% and 76.96% respectively, which is 0.05% and 0.73% higher than our method. Compared to the general autoencoder, they further enhance this tool by learning more discriminative reconstructions. OC-SVM gets the same classification results on SVHN dataset. On MNIST and USPS datasets, our density based anomaly detection algorithm achieves the best performance. With no background noise interference, density based anomaly analysis is more effective than sample reconstruction. Without the interference of background noise, density based anomaly analysis can well describe the relationship between samples in high-dimensional space. On CIFAR10 and CIFAR100, the unsupervised one-class learning (UOCL) method achieves the best classification accuracy of 95.77% and 77.13%, respectively. In the UOCL method, an alternate optimization algorithm is designed to refine the classifiers and labels during the iteration. On ILSVRC2012, boost-autoencoder has the best performance because of the support of massive data. However, compared with the autoencoder requires an additional training of a neural network, the classification performance of density based anomaly detection algorithm is still competitive, while it has a lower algorithm complexity and needs less time.

In order to observe the effect of anomaly detection on the training process, we collect anomalous degrees of all the original training samples in five datasets (MINST, SVHN, USPS, CIFAR10 and CIFAR100), as shown in Fig. 7. In order to intuitively compare the regularization effect of different algorithms, the anomalous degrees are drawn in ascending order. Due to properties of regularization factor, only samples above the horizontal line $\eta = U$ are penalized, which we show with dotted lines. It can be seen that only a small percentage of samples in MNIST and USPS datasets is penalized, while a larger percentage of samples is punished in SVHN, CIFAR10 and CIFAR100 datasets. An interesting phenomenon is that UOCL algorithm punishes more samples than Boost-autoencoder in the natural image datasets

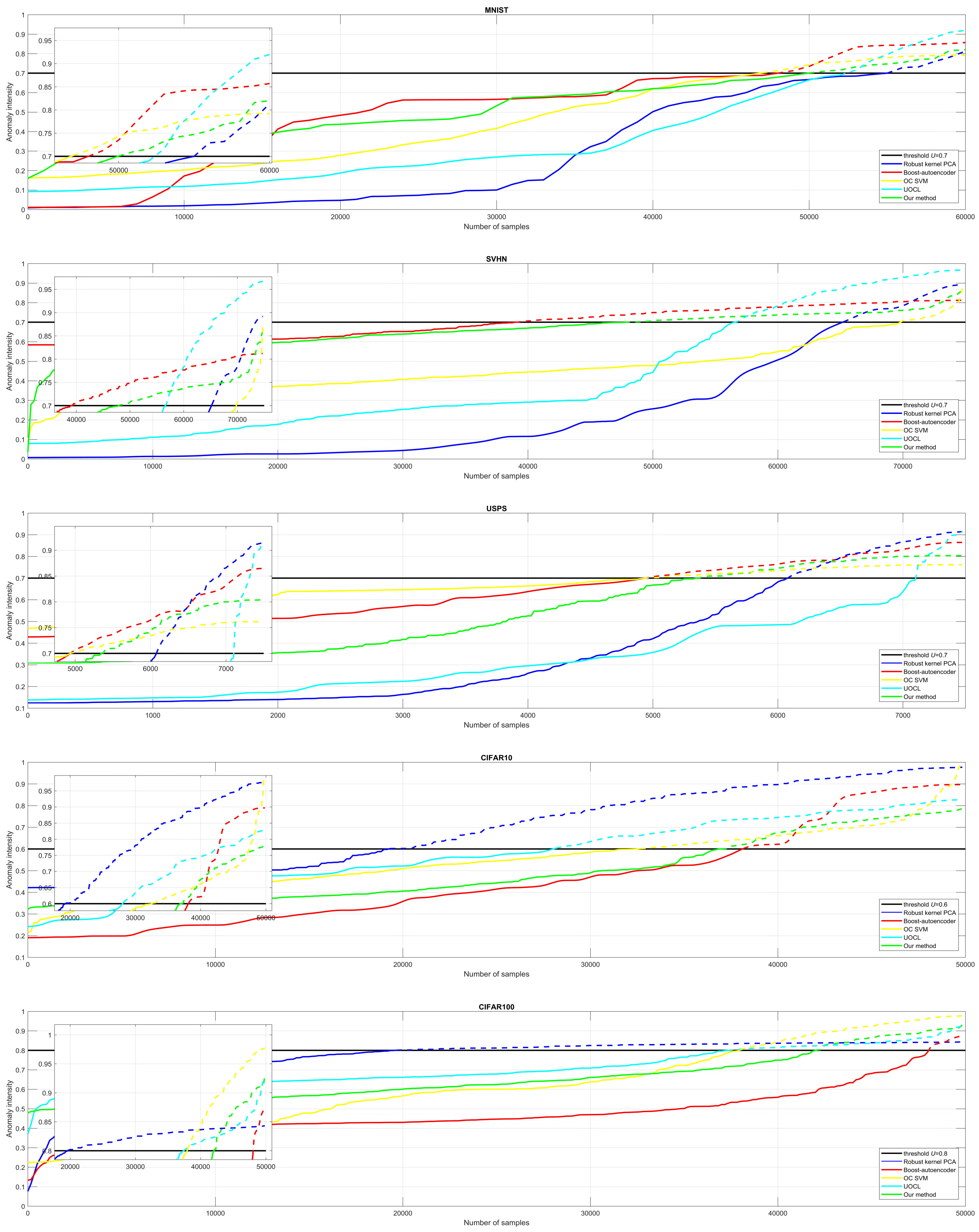


FIGURE 7. Anomaly analysis results of MNIST, SVHN, USPS, CIFAR10 and CIFAR100 datasets.

TABLE 7. Classification results of different initialization methods on MNIST, CIFAR10 and CIFAR100.

MNIST	Classification accuracy(%) without DA		CIFAR10	Classification accuracy(%) with DA		CIFAR100	Classification accuracy(%) with DA	
	General training method	Implicit regularization method		General training method	Implicit regularization method		General training method	Implicit regularization method
Fitnet1-LSUV	99.52	99.78	Fitnet4-LSUV	93.94	95.64	Fitnet4-LSUV	72.34	76.66
Fitnet1-Ortho	99.52	99.75	Fitnet4-Ortho	93.78	95.60	Fitnet4-Ortho	70.44	76.60
Fitnet1-Gauss	99.37	99.73	Fitnet4-Gauss	90.59	94.95	Fitnet4-Gauss	65.58	75.82
Fitnet1-Xavier	99.40	99.79	Fitnet4- Xavier	92.48	95.66	Fitnet4- Xavier	69.32	76.58
Fitnet1-MSRA	99.42	99.79	Fitnet4- MSRA	92.49	95.62	Fitnet4- MSRA	69.54	76.61
Fitnet1-Sparse	99.43	99.76	Fitnet4- sparse	92.46	95.62	Fitnet4- sparse	68.09	75.92
Fitnet1-Hints	99.49	99.80	Fitnet4-Hints	91.61	95.69	Fitnet4-Hints	70.96	76.64

CIFAR10 and CIFAR100, but punishes fewer samples in the digit datasets MNIST, SVHN and USPS. The anomaly detection algorithm based on robust PCA reconstruction is almost invalid in CIFAR10 and CIFAR100. It almost uniformly punishes more than half of the samples and does not play the role of regularization. Meanwhile, it hinders the optimization of deep CNN. It can be seen from the experimental results that an accurate and efficient anomaly detection algorithm is crucial for the implicit regularization training process.

In the Appendix, we show some image samples with high anomalous degrees in five datasets to observe the property of sample distribution and the difference between implicit regularization training process and general training process.

D. INFLUENCE OF DIFFERENT INITIALIZATION

Mishkin and Matas [18] considered that a proper initialization strategy can bring the same regularization effect; it can avoid the network converging to the poor local extreme and achieve a good generalization ability. In this part, we compare the convergence and generalization ability of the deep CNN under different initialization methods and analyze the influence of different initialization on implicit regularization training process.

Orthogonal initialization makes the weight matrix orthogonal to each other by using singular value decomposition. Layer-sequential unit-variance (LSUV) initialization [18] consists of two steps. It pre-initializes weights of each convolution layer with orthonormal matrices first and then normalizes the variance of the output of each layer to 1. The initialization method based on Gaussian function sets the weight matrix to Gaussian filter. Xavier Initialization [65] further makes the variance of input and output consistent. The mean value is 0, and the standard deviation is generally set to

$$stddev = \sqrt{\frac{2}{m+q}} \tag{26}$$

where m and q are the number of inputs and outputs, respectively. MSRA initialization [7] is a variant of the Xavier, and the standard deviation is generally set to

$$stddev = \sqrt{\frac{2}{m}} \tag{27}$$

The main idea of Sparse initialization is to initialize each layer exactly with k non-zero weights, which makes the total number of neurons in this layer independent of the number of inputs. Hints [52] is the default initialization of the FitNet structure, and it is also the default initialization method in this paper.

We conduct two sets of experiments on MNIST, CIFAR10 and CIFAR100 datasets. In the first group of experiments, all the networks are trained by traditional training methods. In the second group of experiments, we train all the different initialized networks by using the two-stage training method. The final classification results of different initialized networks are shown in Table 7. Compare the results of three datasets under two training methods, the classification accuracy of the two-stage training method is higher than those of the general training method under almost all initialization. Comparing the results of all initialization methods under every dataset, an interesting phenomenon is that the classification accuracy of networks with different initialization are very close after two-stage training process. The difference between the classification accuracy of different initialization methods on MNIST is not more than 0.1%, and the difference between the classification results of CIFAR10 and CIFAR100 is not more than 1%. Empirically, we think that the implicit regularization training process based on anomaly detection changes the sample distribution in high-dimensional space and the form of loss function; different initialization methods eventually converge to the approximate local optimum.

The convergence curves of the implicit regularization training process of three datasets under different initialization conditions are shown in Fig. 8. It can be seen that the classification error rates of training set in the final convergence of different initialized networks are very close; the dominant difference is the convergence speed. LSUV has the fastest convergence rate, which is about 8 epochs, 12 epochs and 11 epochs ahead of orthogonal initialization on MNIST, CIFAR10 and CIFAR100, respectively. Gaussian kernel initialization is generally very slow on three datasets, and there are almost no differences in convergence rate between MSRA initialization and Hints initialization. If the training efficiency is not considered, the classification results of deep CNNs

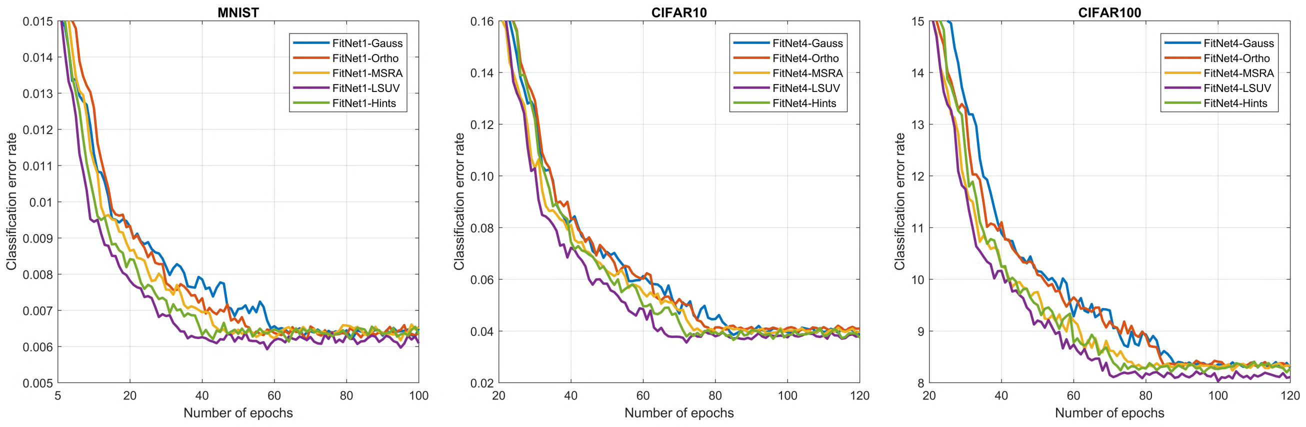


FIGURE 8. Convergence curves of each initialized network on MNIST, CIFAR10 and CIFAR100.

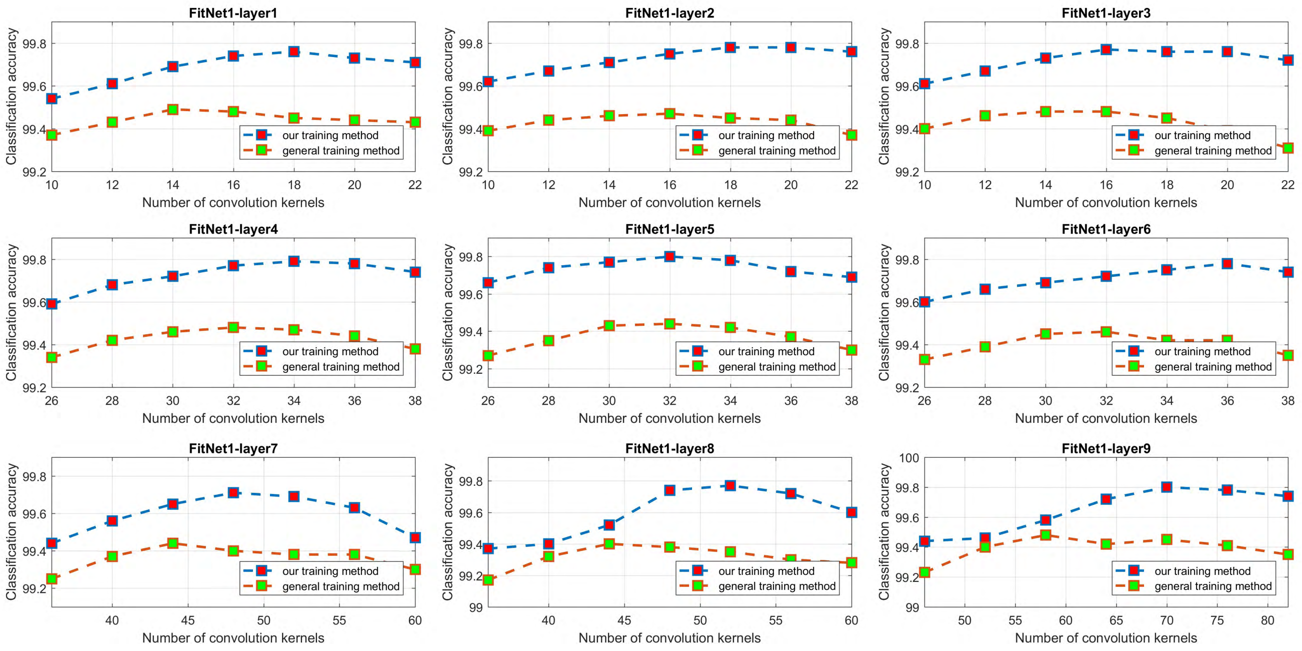


FIGURE 9. Relationship between the number of convolution kernels per layer and the classification accuracy of FitNet1.

based on the two-stage training method are almost unaffected by the initialization, and of course, some extreme initialization conditions are excluded.

E. CHALLENGING THE IMPLICIT REGULARIZATION HYPOTHESIS

A prominent characteristic of regularizer [66] is that the effectiveness of regularization increases as the network capacity (*e.g.*, the number of convolution kernels and hidden nodes) increases, effectively converting the restriction on the network complexity to another. So we explore the relationship between the number of convolution kernels of each layer and the generalization performance of deep CNN under two training methods. The hypothesis that sample punishment acts as an implicit regularizer would suggest that we could see an increasing trend of the classification accuracy on the testing set as the number of convolution kernels of each layer is increased.

We do extensive systematic experiments on the digital image dataset and the natural image dataset to verify the hypothesis. The relationships between the number of convolution kernels per layer and the classification accuracy of MNIST and CIFAR10 are shown in Fig. 9 and Fig. 10, respectively. With the increase of the number of convolution kernels per layer, the generalization error of the network is gradually reduced. When the number of filters exceeds a threshold, the generalization error of the network begins to increase. There are individual oscillations throughout the

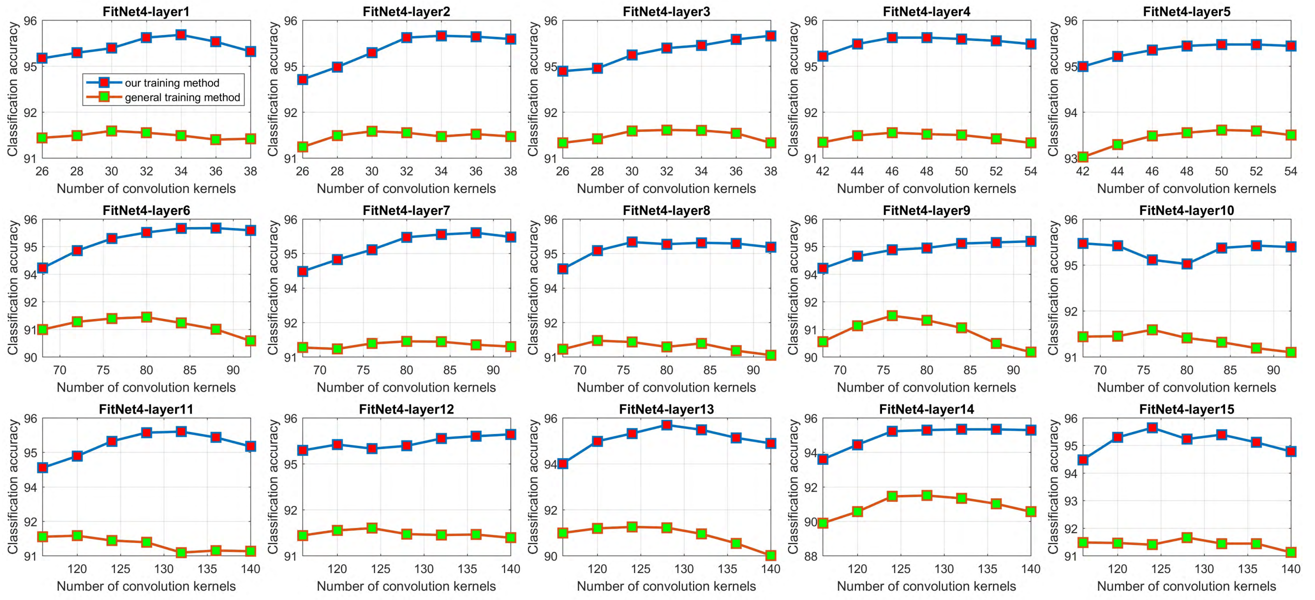


FIGURE 10. Relationship between the number of convolution kernels per layer and the classification accuracy of FitNet4.

process (*e.g.*, FitNet4-layer10 and FitNet4-layer15), which we think that they may be related to the order of samples entering the network. Compared with the implicit regularization training method, the classification accuracy of deep CNN under general training method drops earlier. The experimental results re-verify our implicit regularization hypothesis. As the number of convolution kernels per layer increases, the generalization ability of FitNet4 increases more than that of FitNet1. Small networks have a limited capacity already so that further restricting (or introducing an additional bias) can be harmful to generalization. Such a result seems incompatible with a pure optimization effect. This is also a more systematic result we observed: the two-stage training process is more helpful for deep CNN (like FitNet4), but it is easy to cause damage to the shallow network (like FitNet1).

F. HYPER-PARAMETERS SENSITIVITY

In order to test the practicality of the algorithm, we observe the influence of hyper-parameters on the classification ability. The two-stage training process includes three additional hyper-parameters that need to be artificially set: the number of neighbors k in the anomaly detection process, the weight β in regularization factor, and the threshold U. The classification results of six datasets under different hyper-parameters are shown in Fig.11, Fig.12 and Fig.13.

The classification accuracy of six datasets under different k is shown in Fig. 11. It can be seen that the difference in the classification accuracy of different k is not obvious. Especially in the ILSVRC2012, the classification accuracy is basically unrelated to the choice of k, and it oscillates with the change of k. In CIFAR10 and CIFAR100, the classification accuracy increases as the value of k increases and eventually tends to be stable. In MNIST, USPS and SVHN, the extremely small and excessively large value of k lead to a decline in classification accuracy. Overall, the fluctuation of the classification accuracy is not significant. The results show that the algorithm is robust to the choice of hyper-parameter k, and the classification performance of the model does not depend too much on the value of k. Moreover, we observed that in MNIST, SVHN, USPS, CIFAR10 and CIFAR100, the network achieved the highest classification accuracy at $k = 10$, 15, 10 (or 15), 20 and 90, respectively. It seems that the appropriate value of k is related to the number of dataset categories, which can serve as a guide to the choice of hyper-parameters.

The classification accuracy of six datasets under different β is shown in Fig. 12. β is the weight of regularization factor, which is used to manually adjust the regularization intensity. It can be seen that at the beginning, with the increase of β, the classification accuracy increases gradually. But a too large β leads to an under-fitting problem. Compared with CIFAR10 and CIFAR100, the network trained on ILSVRC2012 is more robust and the classification accuracy is almost unaffected by implicit regularization due to the mass data (even when β is increased from 8 to 13, the classification accuracy is reduced by about 0.2%).

The classification accuracy of six datasets under different U is shown in Fig. 13. U represents the outlier threshold and is responsible for controlling the number of punished samples. The generalization performance of deep CNN is not sensitive to the value of U, which is related to the property of the regularization factor (introduced in section 3). An appropriate U can improve the generalization ability of the network, too large or too small values do not have a terrible effect. A large U makes few samples punished, and a small U makes most

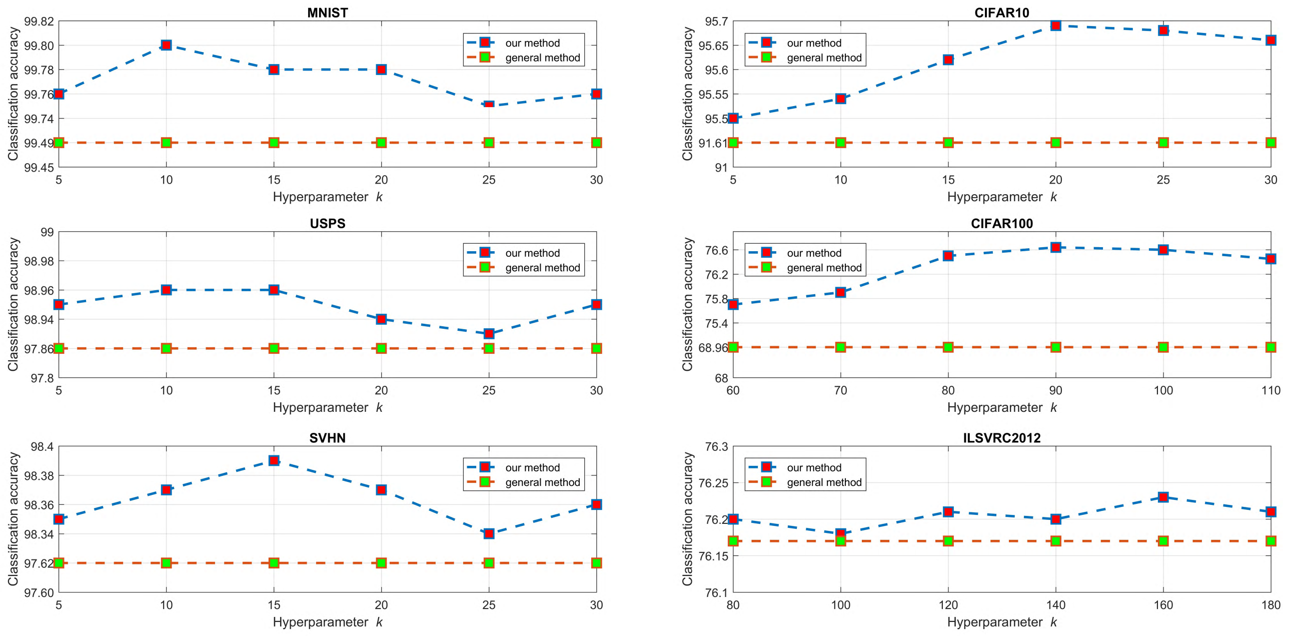


FIGURE 11. Relationship between the classification accuracy and the neighbour *k*.

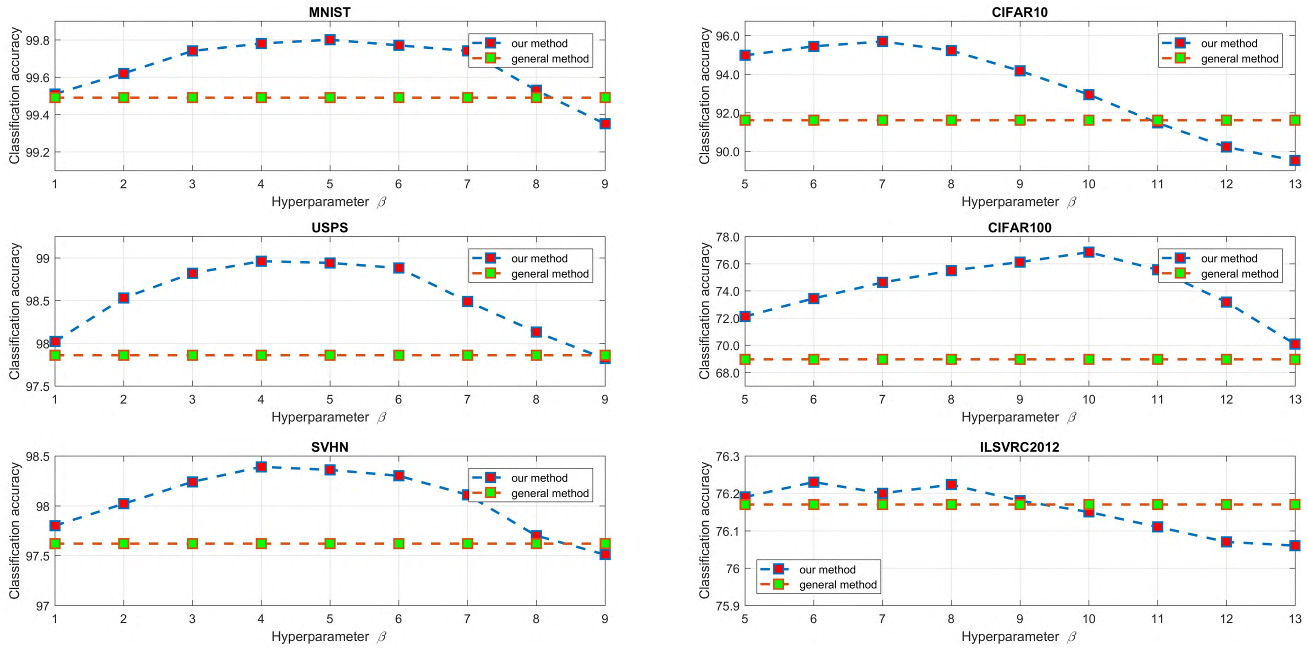


FIGURE 12. Relationship between the classification accuracy and the weight β.

of the samples punished with a minimal regularization factor (close to 0). However, a small U $(0.1 \leq \mathrm{U} \leq 0.4)$ means that there are a large number of samples are punished, which influences the network optimization. Experiments show that the network has an excellent generalization performance when U belongs to (0.4, 0.8).

In this section, experiments show that these three extra hyper-parameters, unlike the learning rate, have little influence on the optimization process and classification ability of deep CNN. And experiments also provide some guidance for the choice of three hyper-parameters.

G. VISUALIZATION ANALYSIS

In order to observe the influence of implicit regularization on decision boundary, we build a two-dimensional toy dataset $D_{\text{toy}} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{171}$ containing three categories to observe

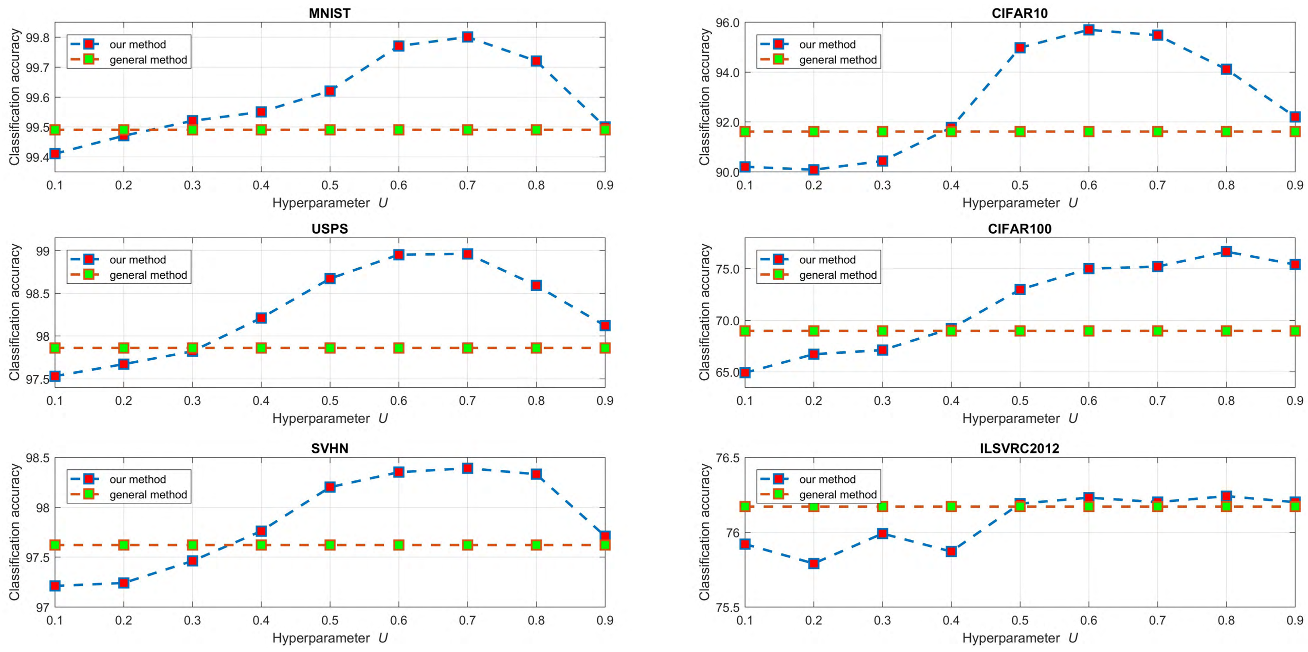


FIGURE 13. **Relationship between the classification accuracy and the threshold *U*.**

the change of decision boundary. The input $\mathbf{x}_n = (x_{n1}, x_{n2})$ is the coordinate of each point, in which $x_{n1}, x_{n2} \in (0, 600)$. Then we use the general training method and the two-stage training method to train a small neural network separately, and draw the decision area of the trained network model, as shown in Fig. 14. In the coordinate system, $\circ$, $+$ and $*$ represent the respective samples of the three categories in the toy dataset. The red, green and blue areas represent decision areas of the three types of samples respectively. Fig. 14a shows the classification result of the model obtained by general training method. Fig. 14b illustrates the anomaly detection results. We use the circumcircle of each sample reflects the anomalous degree. The larger the circle area is, the higher the anomalous degree will be. The Fig. (c), Fig. (d), Fig. (e) and Fig. (f) are the classification results of the network model based on implicit regularization training, in which $\beta = 4$, 5, 6, and 7. Comparing the decision boundaries of (a), (c), (d), (e) and (f), it can be seen that the two-stage training method plays a regularized role in the convergence of the model and can prevent the network from overfitting to outliers. As β increases, the regularization effect becomes stronger and the decision boundary becomes more smooth and natural, which is consistent with human cognition.

V. DISCUSSIONS

A. DIFFERENCE FROM L2-REGULARIZATION

The l_2-regularization is equivalent to weight decay in deep learning, which is an effective regularization method. From the learning theory perspective, it reduces the network overfitting by constraining model complexity. From the optimization or numerical calculation perspective, l_2-norm helps to deal with the inversion problem of ill-conditioned matrix. However, as the number of data increases, the effect of standard l_2-regularization diminishes. It is also a regularization way at the loss function level. For the dataset $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$, the loss function of network with l_2-regularization is updated as

$$l_n^{\dagger} = l_n + \frac{\lambda}{2}||\boldsymbol{\omega}||_2^2 \tag{28}$$

where λ is the regularization factor that determines the regularization intensity. The implicit regularization method based on anomaly detection updates the loss function to

$$l_n^{\dagger} = (1 - \eta_n)l_n = l_n - \eta_n l_n \tag{29}$$

Both of these two methods regularize the network at the loss layer, but the regularization objects are different. The l_2-regularization aims at the network structure, while implicit regularization process aims at training samples. From the optimization perspective, unlike the damping effect of l_2-regularization, our proposed method has an additional momentum.

B. INTERPRETATION AS DATA AUGMENTATION

The real goal of parameter updating during training process is to minimize the expected loss, but in fact, we just update the parameters with a minimization of the empirical loss (the average loss for all training samples). If the sample distribution of training set is not reasonable, the empirical loss and expected loss are quite different. The description of the probability space may be incomplete due to lack of samples, so it is easy to lead to overfitting in the training set. The expansion of training set can help the sample distribution

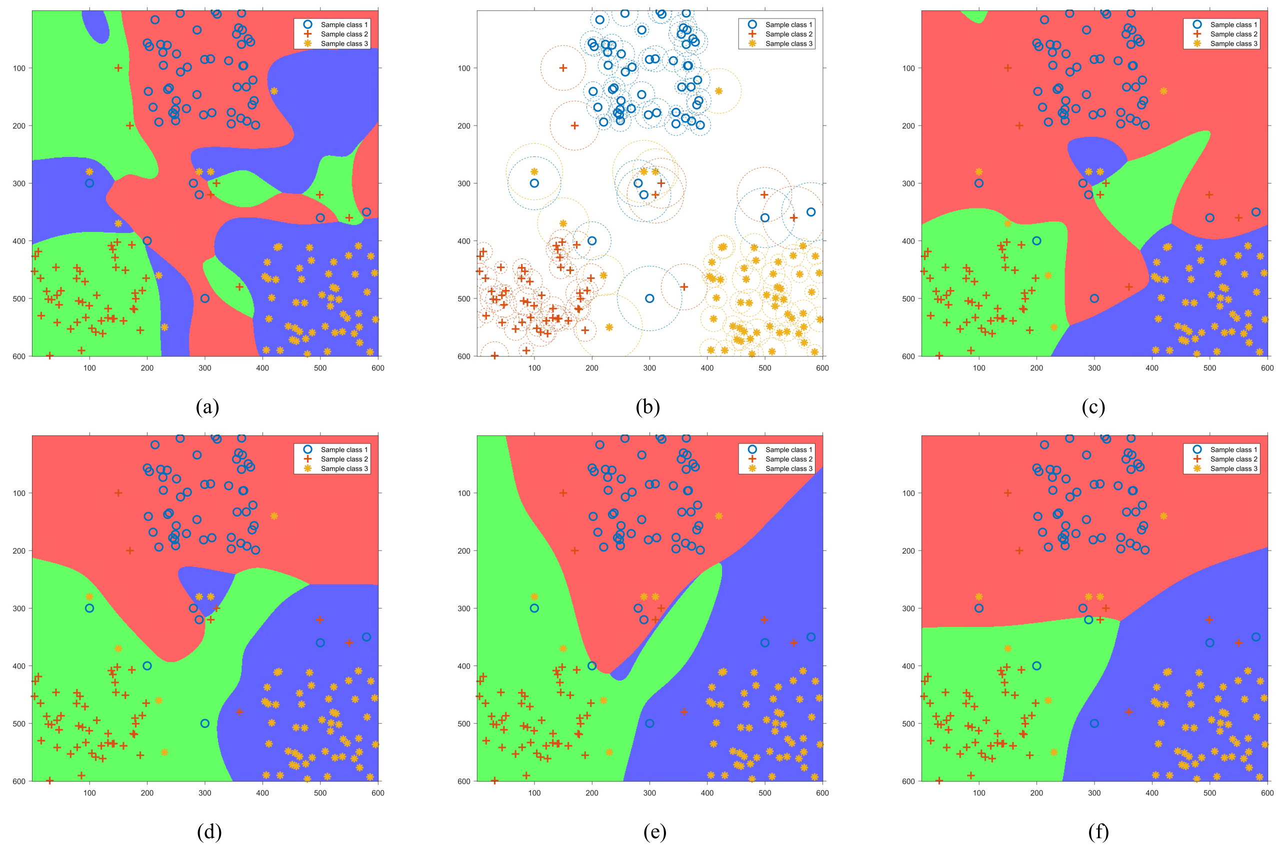


FIGURE 14. **The decision areas of neural network on the toy datasets. (a) shows the network model obtained by traditional training method. (b) shows the anomaly detection results. (c), (d), (e) and (f) show the network models obtained by implicit regularization training method with different β.**

to approximate the real distribution, and it is a manual way to guide the establishment of decision boundary; this is an effective but unnatural way.

Our implicit regularization training process can also be interpreted as a method of data augmentation. Consider a sample $(\mathbf{x}, \mathbf{y}, \mu)$, the log-likelihood loss function in the pre-training process is

$$l(\mathbf{x}, \mathbf{y}) = -\sum_{k=1}^{C} \mathbf{y}_k \log[f_k(\mathbf{x})] \tag{30}$$

where $f(\mathbf{x})$ is the output of deep CNN. And the updated loss function in the implicit regularization training process is

$$\begin{aligned} l^{\dagger}(\mathbf{x}, \mathbf{y}) &= (1-\eta) l(\mathbf{x}, \mathbf{y}) \\ &= -\sum_{k=1}^{C} \mathbf{y}_k \log[f_k(\mathbf{x})]^{(1-\eta)} \\ &= -\sum_{k=1}^{C} \mathbf{y}_k \log[\widetilde{f}_k(\mathbf{x})] \end{aligned} \tag{31}$$

where $\widetilde{f}_k(\mathbf{x}) = f_k(\mathbf{x})^{(1-\eta)}$ can be seen as a noisy component. The noisy output vector is $\widetilde{f}(\mathbf{x}) = [\mathbf{f}_1(\mathbf{x}), \widetilde{f}_2(\mathbf{x}), \ldots, \widetilde{f}_C(\mathbf{x})]^T$. The noise intensity is related to the characteristics of sample and the sample distribution of training set. In fact, the noisy output $\widetilde{f}(\mathbf{x})$ can be projected back into the sample space by minimizing the squared error $||\widetilde{f}(\mathbf{x}) - f(\widetilde{\mathbf{x}})||_2^2$ [67], in which $\widetilde{\mathbf{x}}$ represent the augmented sample. In other words, the anomalous sample $(\mathbf{x}, \mathbf{y}, \mu)$ can be seen as the augmented sample $(\widetilde{\mathbf{x}}, \mathbf{y})$ with noise disturbance.

C. INTERPRETATION AS MODEL ENSEMBLE

In this part, we show that the implicit regularization process can be explained as implicit model ensemble. Hinton *et al.* [68] reports that the combination of deep CNNs trained on different noisy datasets is helpful to improve the classification performance. However, it is prohibitively expensive to train each neural network alone, considering that it requires a large number of noisy datasets.

Consider a dataset after an anomaly detection $D = \{(\mathbf{x}_n, \mathbf{y}_n, \mu_n)\}_{n=1}^{N}$, which is equivalent to a noisy dataset $\widetilde{D} = \{(\widetilde{\mathbf{x}}_n, \mathbf{y}_n)\}_{n=1}^{N}$. Each iteration in the implicit regularization training process is similar to the network training on different noisy set $\widetilde{D}_t$, which is a mini-batch of the noisy dataset $\widetilde{D}$. Therefore, the implicit regularization training process can be viewed as training many networks that have substantial sharing weights but different training samples. In fact, the dropout can be explained as an efficient way to approximately combine numerous CNNs with different architecture that trained on the same data, while the implicit regularization training process can be interpreted as an efficient method to approximately combine a large number of identical CNN structures but trained on different noisy datasets.

D. IMPACT ON THE OPTIMIZATION PROCESS

In this part, we analyze and illustrate the impact of implicit regularization on the network optimization process. The updated loss function $l^\dagger : \mathbb{R}^C \to \mathbb{R}$ is continuously differentiable, so the gradient of $l^\dagger$, *i.e.*, $\nabla l^\dagger : \mathbb{R}^C \to \mathbb{R}^C$ is Lipschitz continuous with Lipschitz constant $L > 0$, namely,

$$||\nabla l^\dagger(\boldsymbol{\omega}) - \nabla l^\dagger(\overline{\boldsymbol{\omega}})||_2 \leq L||\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}||_2 \text{ for all } \{\boldsymbol{\omega},\overline{\boldsymbol{\omega}}\} \subset \mathbb{R}^C \tag{32}$$

Then we can prove

$$\begin{aligned}
l^\dagger(\boldsymbol{\omega}) &= l^\dagger(\overline{\boldsymbol{\omega}}) + \int_0^1 \frac{\partial l^\dagger(\boldsymbol{\omega} + t(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}))}{\partial t} dt \\
&= l^\dagger(\overline{\boldsymbol{\omega}}) + \int_0^1 \nabla l^\dagger(\boldsymbol{\omega} + t(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}))^T(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})dt \\
&= l^\dagger(\overline{\boldsymbol{\omega}}) + \nabla l^\dagger(\boldsymbol{\omega})^T(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}) \\
&\quad + \int_0^1 [\nabla l^\dagger(\boldsymbol{\omega} + t(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})) - \nabla l^\dagger(\overline{\boldsymbol{\omega}})]^T(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})dt \\
&\leq l^\dagger(\overline{\boldsymbol{\omega}}) + \nabla l^\dagger(\boldsymbol{\omega})^T(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}) \\
&\quad + \int_0^1 L||t(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}})||_2||\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}||_2 dt \\
&= l^\dagger(\overline{\boldsymbol{\omega}}) + \nabla l^\dagger(\boldsymbol{\omega})^T(\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}) + \frac{1}{2}L||\boldsymbol{\omega} - \overline{\boldsymbol{\omega}}||_2^2
\end{aligned} \tag{33}$$

Then,

$$\begin{aligned}
&l^\dagger(\boldsymbol{\omega}_{t+1}) - l^\dagger(\boldsymbol{\omega}_t) \\
&\leq \nabla l^\dagger(\boldsymbol{\omega}_t)^T(\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t) + \frac{1}{2}L||\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t||_2^2 \\
&= -\alpha_t(1-\psi)\psi^t[\frac{1}{|D_t|}\sum_{j\in D_t}(1-\eta_j)]g(\boldsymbol{\omega}_t)\sum_{i=1}^{t}\psi^{-i}g(\boldsymbol{\omega}_i) \\
&\quad + \frac{1}{2}\alpha_t^2 L(1-\psi)^2\psi^{2t}[\frac{1}{|D_t|}\sum_{j\in D_t}(1-\eta_j)]^2[\sum_{i=1}^{t}\psi^{-i}g(\boldsymbol{\omega}_i)]^2 \\
&\approx -\alpha_t(1-\psi)\psi^t[\frac{1}{|D_t|}\sum_{j\in D_t}(1-\eta_j)]g(\boldsymbol{\omega}_t)\sum_{i=1}^{t}\psi^{-i}g(\boldsymbol{\omega}_i)
\end{aligned} \tag{34}$$

where

$$g(\boldsymbol{\omega}_t) = \frac{1}{|D_t|}\sum_{(\mathbf{x},\mathbf{y})\in D_t} \nabla_{\boldsymbol{\omega}_t}[l(\mathbf{x},\mathbf{y})] \tag{35}$$

Finally, we can get

$$\begin{aligned}
&\mathrm{E}_{D_t}[l^\dagger(\boldsymbol{\omega}_{t+1})] - \mathrm{E}_{D_t}[l^\dagger(\boldsymbol{\omega}_t)] \\
&\leq \mathrm{E}_{D_t}[\nabla l^\dagger(\boldsymbol{\omega}_t)^T(\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t)] + \mathrm{E}_{D_t}(\frac{1}{2}L||\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t||_2^2) \\
&\approx -\alpha_t(1-\psi)\psi^t\mathrm{E}_{D_t}[\frac{1}{|D_t|}\sum_{j\in D_t}(1-\eta_j)] \\
&\quad \cdot \mathrm{E}_{D_t}[g(\boldsymbol{\omega}_t)\sum_{i=1}^{t}\psi^{-i}g(\boldsymbol{\omega}_i)] \\
&= \begin{cases} -\alpha_t\mathrm{E}_{D_t}[\frac{1}{|D_t|}\sum_{j\in D_t}(1-\eta_j)]\mathrm{E}_{D_t}[g^2(\boldsymbol{\omega}_t)], & \text{if } \psi = 0 \\ 0 \ (V_0 = 0), & \text{if} \psi = 1 \end{cases} \\
&= Boundary_t
\end{aligned} \tag{36}$$

where $Boundary_t$ is the upper bound of the decrease of loss function after t^{th} iteration. This shows that the optimization process with momentum is continues in a Non-Markovian manner in the sense that $\boldsymbol{\omega}_{t+1}$ is a random variable related to all $\boldsymbol{\omega}_i$ in the past t iterations. While the optimization process without momentum (*i.e.*, $\psi = 0$) is continues in a Markovian manner in the sense that $\boldsymbol{\omega}_{t+1}$ depends only on the iterate $\boldsymbol{\omega}_t$ and not on any past iterates. The expected decrease of loss function yielded by the t^{th} iteration is bounded above by a deterministic quantity involving: the expected directional derivative of $l(\mathbf{x}, \mathbf{y})$ at $\boldsymbol{\omega}$ in the past t iterations, the learning rate α, the momentum ψ, and the regularization factor η. The regularization factor has a linear influence on the optimization of deep CNN. Similarly,

$$\begin{aligned}
&\mathrm{E}_{D_{t-1}}[l^\dagger(\boldsymbol{\omega}_t)] - \mathrm{E}_{D_{t-1}}[l^\dagger(\boldsymbol{\omega}_{t-1})] \\
&\leq \mathrm{E}_{D_{t-1}}[\nabla l^\dagger(\boldsymbol{\omega}_{t-1})^T(\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1})] \\
&\quad + \mathrm{E}_{D_{t-1}}(\frac{1}{2}L||\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1}||_2^2) \\
&\approx -\alpha_{t-1}(1-\psi)\psi^{t-1}\mathrm{E}_{D_{t-1}}[\frac{1}{|D_{t-1}|}\sum_{j\in D_{t-1}}(1-\eta_j)] \\
&\quad \cdot \mathrm{E}_{D_{t-1}}[g(\boldsymbol{\omega}_{t-1})\sum_{i=1}^{t-1}\psi^{-i}g(\boldsymbol{\omega}_i)] \\
&= Boundary_{t-1}
\end{aligned} \tag{37}$$

Then we can prove

$$\begin{aligned}
&\frac{Boundary_t}{Boundary_{t-1}} \\
&= \frac{\psi^t \cdot \mathrm{E}_{D_t}[\frac{1}{|D_t|}\sum_{j\in D_t}(1-\eta_j)] \cdot \mathrm{E}_{D_t}[g(\boldsymbol{\omega}_t)\sum_{i=1}^{t}\psi^{-i}g(\boldsymbol{\omega}_i)]}{\psi^{t-1} \cdot \mathrm{E}_{D_{t-1}}[\frac{1}{|D_{t-1}|}\sum_{j\in D_{t-1}}(1-\eta_j)] \cdot \mathrm{E}_{D_{t-1}}[g(\boldsymbol{\omega}_{t-1})\sum_{i=1}^{t-1}\psi^{-i}g(\boldsymbol{\omega}_i)]} \\
&= \psi \cdot \frac{\alpha_t}{\alpha_{t-1}} \cdot \frac{\mathrm{E}_{D_t}[g(\boldsymbol{\omega}_t)\sum_{i=1}^{t}\psi^{-i}g(\boldsymbol{\omega}_i)]}{\mathrm{E}_{D_{t-1}}[g(\boldsymbol{\omega}_{t-1})\sum_{i=1}^{t-1}\psi^{-i}g(\boldsymbol{\omega}_i)]} \\
&= \frac{\mathrm{E}_{D_t}[g(\boldsymbol{\omega}_t)\sum_{i=1}^{t}\psi^{1-i}g(\boldsymbol{\omega}_i)]}{\mathrm{E}_{D_{t-1}}[g(\boldsymbol{\omega}_{t-1})\sum_{i=1}^{t-1}\psi^{-i}g(\boldsymbol{\omega}_i)]}, \quad \text{if } \alpha_{t-1} = \alpha_t
\end{aligned} \tag{38}$$

This illustrates that the changing rate of the upper bound of decreasing loss has nothing to do with the regularization factor. In other words, the implicit regularization factor has

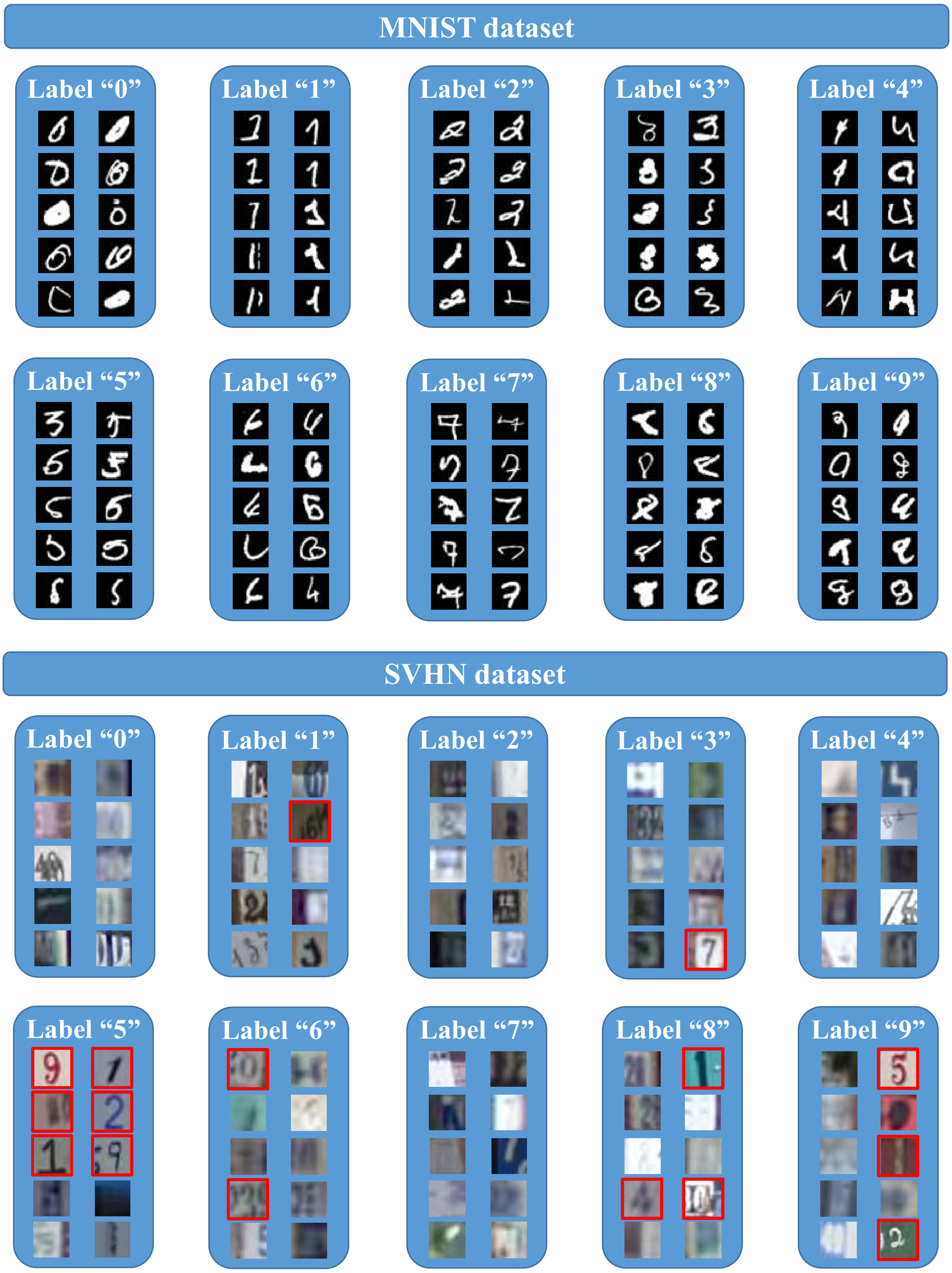


FIGURE 15. Sample images with high anomalous degrees in MNIST and SVHN.

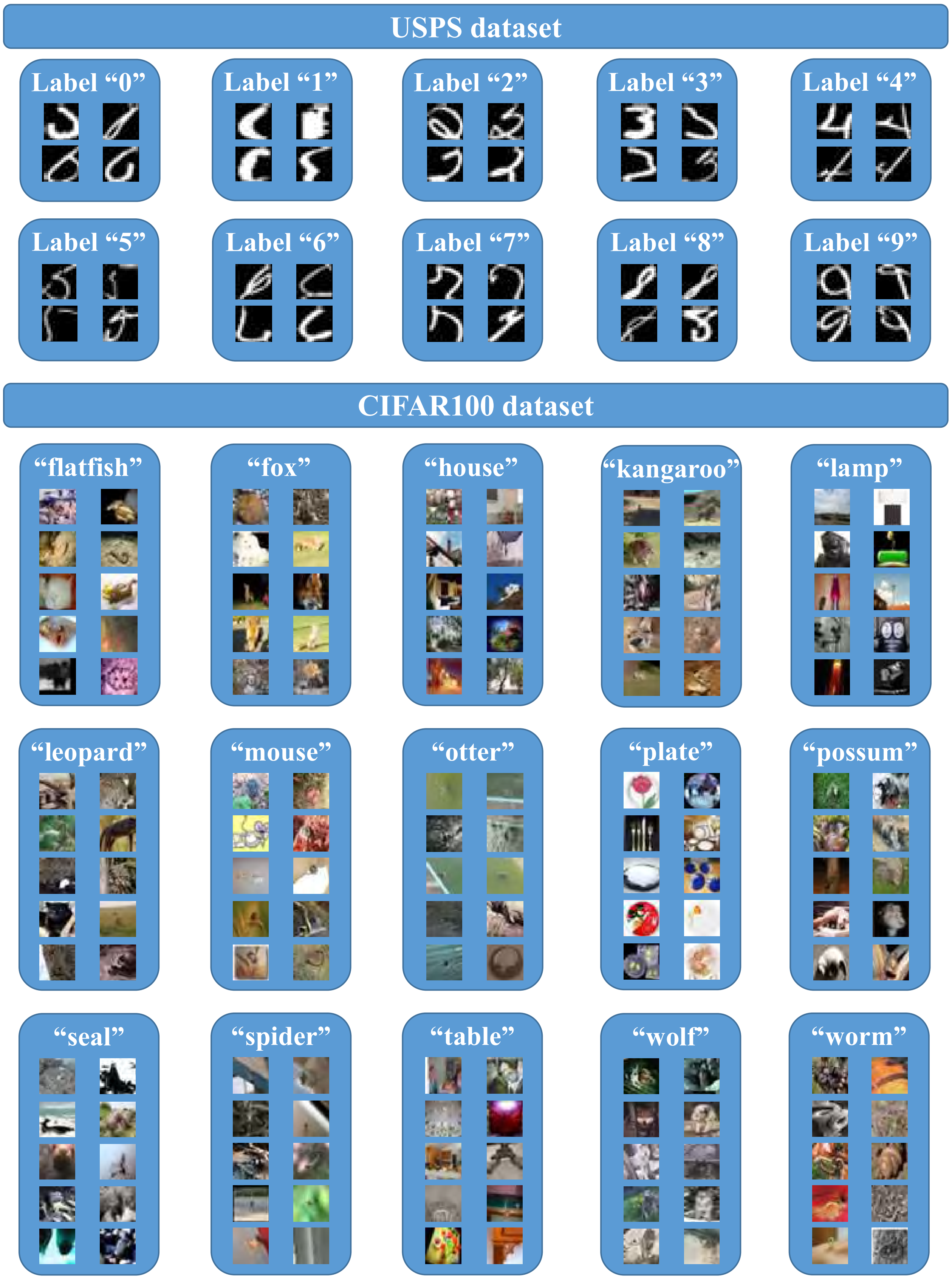


FIGURE 16. **Sample images with high anomalous degrees in USPS and CIFAR10/100.**

no influence on the change of convergence rate. Moreover, the intermediate process of equation 38 also explains why a decreasing learning rate in the iterations can help with the network optimization.

VI. CONCLUSION

In this paper, we present the two-stage training method, a novel algorithm that first regularizes deep CNNs from the data decay perspective. The two-stage training method includes three parts: pre-training stage, anomaly detection and implicit regularization training stage. It first completes the anomaly detection based on the pre-training model and then sets a regularization factor for the implicit regularization training process. To verify the algorithm, we do extensive experiments to show that the algorithm improves the generalization ability of deep CNN by optimizing the feature boundary, and is robust to the choice of hyper-parameters. Moreover, the classification ability of the convergent network model is hardly affected by initialization. The classification results show that our method achieves the best performances in MNIST, SVHN and USPS, CIFAR10/100. Even in ILSVRC2012, we also achieve a fairly competitive performance. And the two-stage training method consistently outperforms other regularization methods on deep CNNs, such as weight decay, SoftLabel and DisturbLabel. In theory, the two-stage training method based on more advanced network models can perform better classification results. And we believe that the implicit regularization approach at the data level provides a new way for AI learning, which is applicable to all machine learning algorithms (*e.g.*, deep forests and support vector machine).

At the same time, the experience points out a few lessons and future directions, which we summarize as follows:

- Density-based anomaly detection algorithm is not reliable in high dimensional space due to the sparse sample distribution and the influence of noise.
- The two-stage training method is time-consuming. How to integrate the two stages into an end-to-end pattern remains to be studied.
- Whether the regularization method at the data level is valid for other machine learning algorithms needs to be verified.

These questions will be further explored in future research.

APPENDIX

In this appendix, we show some sample images with high anomalous degrees in MNIST, SVHN, USPS, and CIFAR10/100 datasets, as shown in Fig. 15 and Fig. 16. In MNIST and USPS, most of the outliers have odd shapes, which are very different from samples of the same category. In SVHN, we even detected many mislabeled training samples, *i.e.*, the samples in red borders. Their sample number are 5739, 61494, 6292, 9303, 8120, 20091, 40130, 63554, 71383, 25491, 12807, 9328, 37661, 6291, 33793, 66859 in sequence, which may be caused by the carelessness during the crowdsourcing annotation process. In CIFAR10/100, most anomalous samples are images with too small objects, lying on similar and complex backgrounds. These outliers have same characteristics that the object features are difficult to extract and the sample distribution is unnatural.

ACKNOWLEDGMENT

The authors appreciate the MNIST, SVHN, USPS, CIFAR10/100 and ILSVRC2012 datasets. And the authors wish to gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017, doi: 10.1038/nature24270.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 2, pp. 84–90, Jun. 2012, doi: 10.1145/3065386.
[3] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE ICCV*, Santiago, Chile, Dec. 2015, pp. 4489–4497, doi: 10.1109/ICCV.2015.510.
[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, Montreal, QC, Canada, 2015, pp. 91–99.
[5] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009, vol. 1, no. 4, p. 7.
[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE ICCV*, Santiago, Chile, Dec. 2015, pp. 1026–1034, doi: 10.1109/ICCV.2015.123.
[8] P. Ramachandran, B. Zoph, and Q. V. Le. (Oct. 2017). "Swish: A self-gated activation function." [Online]. Available: https://arxiv.org/abs/1710.05941
[9] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. (Jul. 2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580
[10] L. Wan, M. Zeiler, S. Zhang, Y. Cun, and R. Fergus, "Regularization of neural networks using DropConnect," in *Proc. ICML*, Atlanta, GA, USA, 2013, pp. 1058–1066.
[11] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE CVPR*, Boston, MA, USA, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.
[12] M. Zeiler and R. Fergus. (Jan. 2013). "Stochastic pooling for regularization of deep convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1301.3557
[13] J. Springenberg and M. Riedmiller. (Dec. 2013). "Improving deep neural networks with probabilistic maxout units." [Online]. Available: https://arxiv.org/abs/1312.6116
[14] I. Goodfellow, D. W. Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. ICML*, Atlanta, GA, USA, 2013, pp. 1319–1327.
[15] H. Proenca, J. Neves, S. Barra, T. Marques, and J. C. Moreno, "Joint head pose/soft label estimation for human recognition in-the-wild," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 12, pp. 2444–2456, Jan. 2016, doi: 10.1109/TPAMI.2016.2522441.
[16] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, "DisturbLabel: Regularizing CNN on the loss layer," in *Proc. IEEE CVPR*, Las Vegas, NV, USA, Jun. 2016, pp. 4753–4762, doi: 10.1109/CVPR.2016.514.
[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, Lille, France, 2015, pp. 448–456.
[18] D. Mishkin and J. Matas, "All you need is a good init," in *Proc. IEEE ICLR*, San Juan, Puerto Rico, Nov. 2016, pp. 3013–3018.

[19] P. Rodríguez, J. Gonzàlez, G. Cucurull, J. M. Gonfaus, and X. Roca. (Apr. 2017). "Regularizing CNNs with locally constrained decorrelations." [Online]. Available: https://arxiv.org/abs/1611.01967

[20] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, 2010.

[21] K. Simonyan and A. Zisserman. (May 2015). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[22] C. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *Proc. ICAIS*, San Diego, CA, USA, 2015, pp. 464–472.

[23] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. ICAIS*, San Diego, CA, USA, 2015, pp. 562–570.

[24] L. Zhang and D. Zhang. (Oct. 2015). "SVM and ELM: Who wins? Object recognition with deep convolutional features from ImageNet." [Online]. Available: https://arxiv.org/abs/1506.02509

[25] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. IEEE ICCV*, Kyoto, Japan, Sep. 2009, pp. 221–228.

[26] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. ECCV*, Zurich, Switzerland, 2014, pp. 392–407.

[27] C. Shi, C. Wang, Y. Wang, and B. Xiao, "Deep convolutional activations-based features for ground-based cloud classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 6, pp. 816–820, Jun. 2017, doi: 10.1109/LGRS.2017.2681658.

[28] G. Zhong, S. Yan, K. Huang, Y. Cai, and J. Dong, "Reducing and stretching deep convolutional activation features for accurate image classification," *Cognit. Comput.*, vol. 42, no. 10, pp. 1–8, Oct. 2017, doi: 10.1007/s12559-017-9515-z.

[29] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. ICML*, Beijing, China, 2014, pp. 647–655.

[30] Y. Xia, X. Cao, F. Wen, G. Hua, and J. Sun, "Learning discriminative reconstructions for unsupervised outlier removal," in *Proc. IEEE ICCV*, Santiago, Chile, Dec. 2015, pp. 1511–1519.

[31] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. ACM MLSDA*, 2014, pp. 4–11, doi: 10.1145/2689746.2689747.

[32] M. L. Shyu, S. C. Chen, and K. Sarinnapakorn, "A novel anomaly detection scheme based on principal component classifier," in *Proc. IEEE Found. Directions DataMining Workshop*, Jan. 2003, pp. 172–179.

[33] F. Ju, Y. Sun, J. Gao, Y. Hu, and B. Yin, "Image outlier detection and feature extraction via L1-norm-based 2D probabilistic PCA," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4834–4846, Dec. 2015, doi: 10.1109/TIP.2015.2469136.

[34] M. Rahmani and G. K. Atia, "Randomized robust subspace recovery and outlier detection for high dimensional data matrices," *IEEE Trans. Signal Process.*, vol. 65, no. 6, pp. 1580–1594, Dec. 2017, doi: 10.1109/TSP.2016.2645515.

[35] J. Kim and C. D. Scott, "Robust kernel density estimation," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 2529–2565, Jan. 2012.

[36] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of Data Mining in Computer Security*, vol. 6. New York, NY, USA: Springer, 2002, pp. 77–101, doi: 10.1007/978-1-4615-0953-0_4.

[37] Y. Xia, X. Cao, F. Wen, and J. Sun, "Well begun is half done: Generating high-quality seeds for automatic image dataset construction from Web," in *Proc. ECCV*, Zurich, Switzerland, 2014, pp. 387–400.

[38] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[39] L. M. Manevitz and M. Yousef, "One-class document classification via neural networks," *Neurocomputing*, vol. 70, nos. 7–9, pp. 1466–1481, Mar. 2007, doi: 10.1016/j.neucom.2006.05.013.

[40] W. Liu, G. Hua, and J. R. Smith, "Unsupervised one-class learning for automatic outlier removal," in *Proc. IEEE CVPR*, Columbus, OH, USA, Jun. 2014, pp. 3826–3833.

[41] Y. Gao and Y. Li, "Improving gaussian process classification with outlier detection: With applications in image classification," in *Proc. ACCV*, Queenstown, New Zealand, 2010, pp. 153–164.

[42] H. Xu, C. Caramanis, and S. Mannor, "Outlier-robust PCA: The high-dimensional case," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 546–572, Jan. 2013, doi: 10.1109/TIT.2012.2212415.

[43] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proc. ACM SIGMOD ICMD*, Dallas, TX, USA, 2000, pp. 427–438.

[44] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. ICCS*, Paris, France, 2010, pp. 177–186.

[45] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: Algorithms and applications," *Very Large Data Bases J.*, vol. 8, nos. 3–4, pp. 237–253, 2000.

[46] T. Bottesch, T. Buhler, and M. Kachele, "Speeding up k-means by approximating Euclidean distances via block vectors," in *Proc. ICML*, New York, NY, USA, 2016, pp. 2578–2586.

[47] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. ICLR*, Toulon, France, 2017, pp. 1–14.

[48] Y. Netzer *et al.*, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, Granada, Spain, 2011, pp. 1–9.

[49] M. Markou and S. Singh, "Novelty detection: A review-part 1: Statistical approaches," *Signal Process.*, vol. 83, no. 12, pp. 2481–2497, 2003, doi: 10.1016/j.sigpro.2003.07.018.

[50] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.

[51] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008, doi: 10.1109/TPAMI.2008.128.

[52] A. Romero *et al.* (May 2015). "FitNets: Hints for thin deep nets." [Online]. Available: https://arxiv.org/abs/1412.6550

[53] S. Lin, L. Cai, and R. Ji, "Masked face detection via a modified LeNet," *Neurocomputing*, vol. 218, pp. 197–202, Dec. 2016, doi: 10.1016/j.neucom.2016.08.056.

[54] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. NIPS*, Montreal, QC, Canada, 2015, pp. 2377–2385.

[55] J. Wang, "Semi-supervised learning using multiple one-dimensional embedding based adaptive interpolation," *Int. J. Wavelets Multiresolut Inf. Process.*, vol. 14, no. 2, pp. 37–47, 2016, doi: 10.1142/S0219691316400026.

[56] C. L. Lekamalage *et al.*, "Dimension reduction with extreme learning machine," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3906–3918, Aug. 2016, doi: 10.1109/TIP.2016.2570569.

[57] X. Yang, W. Liu, D. Tao, and J. Cheng, "Canonical correlation analysis networks for two-view image recognition," *Inf. Sci.*, vol. 385, no. 3, pp. 338–352, Apr. 2017, doi: 10.1016/j.ins.2017.01.011.

[58] X. Zhang, N. Vishwamitra, H. Hu, and F. Luo. (Oct. 2017). "CrescendoNet: A simple deep convolutional neural network with ensemble behavior." [Online]. Available: https://arxiv.org/abs/1710.11176.

[59] S. Abe, "Fusing sequential minimal optimization and Newton's method for support vector training," *Int. J. Mach. Learn.*, vol. 7, no. 3, pp. 345–364, 2016, doi: 10.1007/s13042-014-0265-x.

[60] D. Johnson, C. Xiong, and J. Corso, "Semi-supervised nonlinear distance metric learning via forests of max-margin cluster hierarchies," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 1035–1046, Apr. 2016, doi: 10.1109/TKDE.2015.2507130.

[61] C. Gong, T. Liu, D. Tao, K. Fu, E. Tu, and J. Yang, "Deformed graph Laplacian for semisupervised learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2261–2274, Oct. 2015, doi: 10.1109/TNNLS.2014.2376936.

[62] C. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. (Sep. 2014). "Deeply-supervised nets." [Online]. Available: https://arxiv.org/abs/1409.5185

[63] J. Chang and Y. Chen. (Nov. 2015). "Batch-normalized maxout network in network." [Online]. Available: https://arxiv.org/abs/1511.02583

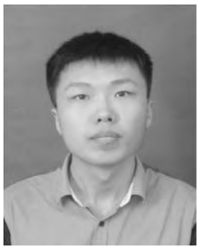
[64] N. Srivastava and R. Salakhutdinov, "Discriminative transfer learning with tree-based priors," in *Proc. ICML Workshop*, Atlanta, GA, USA, 2013, pp. 2094–2102.

[65] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.

[66] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Feb. 2010.

[67] X. Bouthillier, K. Konda, P. Vincent, and R. Memisevic. (Jun. 2015). "Dropout as data augmentation." [Online]. Available: https://arxiv.org/abs/1506.08700

[68] G. Hinton, O. Vinyals, and J. Dean. (Mar. 2015). "Distilling the knowledge in a neural network." [Online]. Available: https://arxiv.org/abs/1503.02531



QINGHE ZHENG was born in Jining, China, in 1993. He received the B.S. degree in communication engineering from the Xi'an University of Posts and Telecommunications, Xi'an, China, in 2014. He is currently pursuing the Ph.D. degree in information and communication engineering with Shandong University, Jinan, China. He is a member of the Intelligent Visual Laboratory. He has authored several conference and journal papers on his research topic. His research interests include computer vision and machine learning.

He is a member of International Association of Engineers. He is serving as a Reviewer for *Neurocomputing*, *IET Electronics Letters*, and the *International Journal of Computer Science*.

MINGQIANG YANG was born in Jinan, China, in 1969. He received the B.S. degree in radio technology from the Shandong University of Technology, Jinan, in 1992, the M.S. degree in communication and information processing from Shandong University, Jinan, in 2000, and the Ph.D. degree in signal and image processing from the Institut National des Sciences Appliquées de Rennes, France, in 2008. He is currently the Leader of the Intelligent Visual Laboratory and also the Deputy Director of the Institute for Information Processing and Applied Technology.

From 2001 to 2004, he was a Research Assistant with Shandong University. Since 2009, he has been an Associate Professor with the School of Information Science and Engineering, Shandong University. He has authored two books, over 40 scientific peer-reviewed articles, and over 10 inventions. His research interests include image processing and feature extraction. He has presided over seven national, provincial, and school scientific research projects.

JIAJIE YANG is currently a Professor with the Faculty of Science, The University of British Columbia.

QINGRUI ZHANG was born in Jinan, China, in 1993. He received the B.S. degree in electronic information science and technology from the Ocean University of China in 2015. He is currently pursuing the M.S. degree in information and communication engineering with Shandong University, Jinan. He is a member of the Intelligent Visual Laboratory. His research interests include image segmentation and classification.

XINXIN ZHANG was born in Heilongjiang, China, in 1991. She received the B.S. degree in signal processing from Nanchang University in 2015. She is currently pursuing the M.S. degree in electronic and communication engineering with Shandong University, Jinan, China. She is a member of the Intelligent Visual Laboratory. Her research interests include image segmentation and edge detection.