Applied Intelligence (2023) 53:13852–13866
https://doi.org/10.1007/s10489-022-04208-6

Improved generalization performance of convolutional neural networks with LossDA

Juncheng Liu[1] · Yili Zhao[1]

Accepted: 25 September 2022 / Published online: 17 October 2022
© The Author(s) 2022

Abstract
In recent years, convolutional neural networks (CNNs) have been used in many fields. Nowadays, CNNs have a high learning capability, and this learning capability is accompanied by a more complex model architecture. Complex model architectures allow CNNs to learn more data features, but such a learning process tends to reduce the training model's ability to generalize to unknown data, and may be associated with problems of overfitting. Although many regularization methods have been proposed, such as data augmentation, batch normalization, and Dropout, research on improving generalization performance is still a common concern in the training process of robust CNNs. In this paper, we propose a dynamically controllable adjustment method, which we call LossDA, that embeds a disturbance variable in the fully-connected layer. The trend of this variable is kept consistent with the training loss, while the magnitude of the variable can be preset to adapt to the training process of different models. Through this dynamic adjustment, the training process of CNNs can be adaptively adjusted. The whole regularization process can improve the generalization performance of CNNs while helping to suppress overfitting. To evaluate this method, this paper conducts comparative experiments on MNIST, FashionMNIST, CIFAR-10, Cats_vs_Dogs, and miniImagenet datasets. The experimental results show that the method can improve the model performance of Light CNNs and Transfer CNNs (InceptionResNet, VGG19, ResNet50, and InceptionV3). The average maximum improvement in accuracy of Light CNNs is 4.62%, F1 is 3.99%, and Recall is 4.69%. The average maximum improvement accuracy of Transfer CNNs is 4.17%, F1 is 5.64%, and Recall is 4.05%.

Keywords Convolutional neural networks · Fully-connected layer · Dynamic adjustment · Generalization performance · Overfitting

1 Introduction

In recent years, convolutional neural networks (CNNs) have been applied in many areas of image processing, such as image classification, object recognition, and object detection [1–3]. Nowadays, CNNs have a high learning capability, and this learning capability is accompanied by a more complex model architecture. Although complex model architectures allow CNNs to learn more data features, such learning tends to reduce the model's ability to generalize to unknown data and may be associated with overfitting problems.

Numerous regularization methods have been proposed to improve generalization performance. However, almost all of these methods are applied at the input and intermediate layers of the model architecture, and each of them has some drawbacks. For example, data augmentation [4] is a regularization method applied to the input layer preprocessing and based on various feature-preserving transformations (e.g., image scaling, rotation, and random cropping) to generate additional training samples to enrich the training data. However, once the new training data is generated, it cannot be added or subtracted based on the training condition. A commonly used regularization method for convolutional layers is batch normalization(BN) [5], which optimizes the landscape based on normalizing the mean and variance of each small batch of features to make it smoother. This regularization method is usually

✉ Yili Zhao
ylzhao@swfu.edu.cn

Juncheng Liu
liujuncheng@swfu.edu.cn

1 College of Big Data and Intelligent Engineering, Southwest Forestry University, Bailong road, Kunming, 650224, Yunnan, China

fixed during the construction of the model architecture in terms of the position and number of features used, and the BN is hardly ever adjusted during the training process. Another commonly used regularization method is Dropout [6], where a subset of units is randomly dropped during each training iteration while the corresponding layer is regularized. Although this adaptation causes dynamic adjustment during training, the adjustment is done in a random manner, which means the effect of the adaptation may be subject to corresponding uncertainty. Similar to Dropout is the DropFilterR [7] regularization method. Unlike Dropout, DropFilterR introduces greater uncertainty by omitting various elements. These methods are almost never adjusted during the training of CNNs, and the intensity of the adjustment is not dynamic and controllable.

To counteract the drawbacks of the above methods, a dynamic and controllable regularization method based on fully-connected layers, called LossDA. is proposed in this paper. The dynamic adjustment changes with the training process and can be set in advance to produce the desired regularization. This regularization is not static and random, but a dynamic and controllable adjustment process. The regularization in this paper has two characteristics. First, the method is based on the fully-connected layer of CNNs, which does not colide with other methods in the input and intermediate layers, such as data augmentation and batch normalization. This can help the method work with other methods to further improve the performance of the model. Second, the method is regularized by embedding a disturbance variable in the fully-connected layer. The trend of this variable is consistent with the training loss. The magnitude of this variable can be adjusted in advance to match the training process of different models. This controlled adjustment process adaptively adjusts the fit of the CNNs to the training process.

The concrete implementation of the method is shown in Fig. 1. It consists of three parts, namely the LM module, the transfer operator, and the DA module. During the training process, the LM module automatically records the training loss generated for each new iteration and generates the disturbance variable values for the next iteration of training by the control parameter R in the module. The newly generated disturbance variables are entered into the DA module embedded in the fully-connected layer through the transfer operator to participate in the adjustment of the next training process. Throughout the process, the values of the disturbance variable change as the training loss changes. Moreover, the parameter R can be adjusted according to different models to achieve the best regularization effect from the training process. To evaluate this approach, we conducted tests comparing different CNNs on MNIST [8], FashionMNIST [9], CIFAR-10 [10], Cats_vs_Dogs [11], and miniImagenet [12] datasets, and used accuracy, F1, recall, and precision metrics data to evaluate the results. The experimental results show that the method can improve the model performance of commonly used Light CNNs and currently popular Transfer CNNs (InceptionResNet [13], VGG19 [14], ResNet50 [15], and InceptionV3 [16]). The adjustment strength of this method adaptively adjusts the training process, which improves the generalization of the model and helps to suppress overfitting.

The remainder of this paper is organized as follows: Section 2 begins with a look at the works associated with this method. Section 3 mainly presents the proposed method, LossDA. Section 4 describes the dataset and relevant details of the experiments. Section 5 contains the

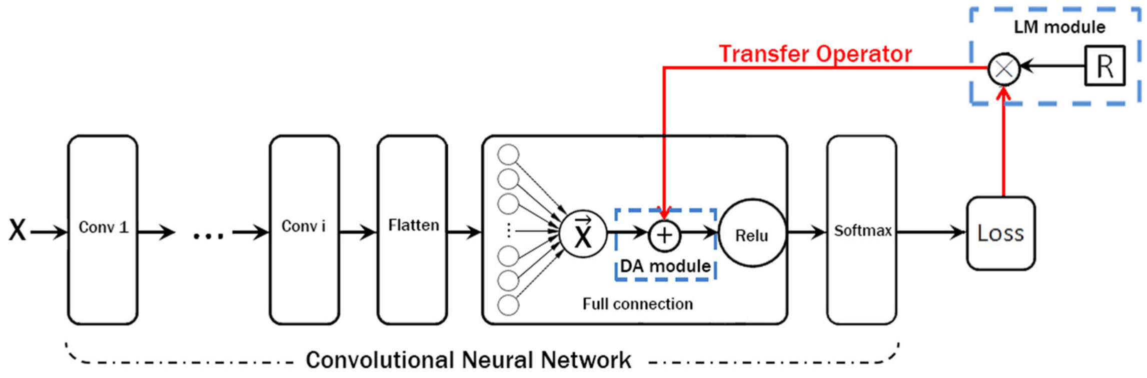


Fig. 1 The framework diagram of LossDA based on the fully-connected layer of the convolutional neural network (CNN). X is the input data of CNN. Conv 1 and Conv i denote the first convolutional layer and the i-th convolutional layer of the volume neural network, respectively, with a total of i convolutional layers, which also contain various operational layers, such as pooling layers. Flatten, Full connection, and Softmax denote the flat layer [18], the fully-connected layer, and the softmax function [45] of the CNN, respectively. In the fully-connected layer, $\vec{X}_i$ denotes the data input in the i-th iteration of the activation function of the fully-connected layer, and the activation function in the figure is taken as the Relu function [46]. In the transfer operator, the LM module and the DA module are included. The red solid line indicates that the training loss is passed through the LM module to generate the disturbance variables, which are then input to the DA module through the transfer operator to augment the training data in the fully-connected layer

results and discussions. Finally, Section 6 concludes with a summary of the paper.

2 Related works

This chapter deals with related work. It consists of three parts, namely applications and challenges of CNNs, fully-connected layers of CNNs, and common regularization methods.

2.1 Applications and challenges of CNNs

Convolutional neural networks (CNNs) have achieved brilliant success [17] and have become one of the most representative neural networks in the field of Deep Learning. In the era of Big Data, CNNs are able to achieve promising results using large amounts of data. As a result, many applications of CNNs have emerged. First, CNN applications for 1D scenarios usually use 1D convolutional kernels to process 1D data. CNNs are very effective in extracting features from a fixed-length segment of the entire dataset, e.g., time series prediction [18, 19] and signal recognition [20]. Second, CNNs are used in 2D image processing for four main types of tasks, namely image classification, object detection, image segmentation, and face recognition. Image classification [21, 22] involves classifying an image into a particular class, and CNNs represent a breakthrough in this area. Object detection [23] is an image classification based task where the system must not only recognize which class the input image belongs to, but also provide it with a bounding box. Image segmentation [24] involves segmenting an image into different regions, where the boundaries of the different semantic units in the image must be labeled. Face recognition is a biometric recognition technique based on facial features that is widely used in various domains [25, 26]. Finally, for multidimensional scenes, CNNs can be used for almost all multidimensional data. For example, in 3D scenes, CNNs are commonly used for human behavior recognition [27–29] or object recognition/detection [30]. In addition, 3D detection is widely used in medical imaging, e.g., X-ray images and CT [31–33].

In recent years, CNNs have received much attention due to their properties such as local connectivity, weight sharing, and downsampling for dimensionality reduction [17]. Local connectivity of CNNs is manifested in the fact that each neuron is no longer connected to all neurons of the previous layer, but only to a few neurons, which effectively reduces parameters and accelerates convergence. The sharing of weight is reflected in the fact that a group of connections can share the same weights, further reducing the parameters and decreasing the computational complexity of the network. In addition, the pooling layer uses the principle of local relevance of images to reduce the amount of data by downsampling for dimensionality reduction, while preserving useful information and improving the generalization capability of the network model. When sample data are fed directly into the network, CNNs can learn implicitly from training, and the whole process is automatic without human intervention. Although the advantages of the three characteristics have led to the widespread use of CNNs in various domains, existing networks still have shortcomings. In particular, how to effectively train deep network models is a problem that still needs to be investigated. Many regularization strategies and methods have been proposed, but the generalization performance of CNN networks still needs continuous improvement.

2.2 Fully-connected layers

Our approach is based on the fully-connected(FC) layer of CNNs, so we will introduce the applications related to the fully-connected layer.

A CNN architecture almost always has a fully-connected(FC) layer [2]. Typically, the FC layer is located at the end of each CNN architecture. Within this layer, each neuron is connected to all neurons in the previous layer, the so-called fully-connected approach. It is often used as a CNN classifier. It follows the basic approach of multilayer perceptual neural networks in that it is a type of feed-forward ANN. The input to the FC layer comes from the last pooling or convolutional layer. This input is in the form of a vector formed from the feature map after flattening [34]. The output of the FC layer represents the final output of the CNN. Many classical architectures based on CNNs have been developed, almost all of which use FC layers, e.g., use cases based on the GoogLeNet architecture [35, 36]. To achieve better model performance, many models built on CNNs even use multiple FC layers, e.g., [37, 38]. Moreover, other studies [39, 40] have shown the importance of fully-connected layers for CNNs. If a better regularization method for the FC layer is developed, then this method will help to be used in various CNN-based application domains.

2.3 Regularization methods

Here, the related methods focus on several common regularization techniques such as data augmentation [4], batch normalization [5], and Dropout [6].

A common regularization technique for the input layer of CNNs is data augmentation [4]. Data augmentation is often used as an effective regularization method to augment the training data of a convolutional neural network. Examples include horizontal flipping, color space augmentation, and random pruning. Later developments in data augmentation techniques also include: geometric transformations, color space transformations, kernel filters, mixed images, random deletion, feature space augmentation, adversarial training, GAN-based augmentation, neural style transfer, and meta-learning, among other augmentation techniques. In short, data augmentation is a regularization technique that enriches the training data.

A common regularization technique used in the middle layer of CNNs is batch normalization(BN) [5]. BN is a power guaranteed output activation that follows a Gaussian unit distribution. It is used to normalize the output of each layer by subtracting the mean and dividing by the standard deviation. BN is useful in two ways [2]. First, it prevents the emergence of gradient disappearance problems and controls the problem of poor initializations of weights. Second, it shortens the time required for the network to converge, especially for large-scale datasets. However, since the weights are continuously updated during training, it also introduces more internal covariance bias defined by changes in the activation distribution. This leads to a model that requires more time for convergence and for training. To solve this problem, layers representing batch normalization operations are used in the CNN architecture. This allows BN to be used not only as a preprocessor of the layers, but also as an integration and differentiation tool for the network.

Another regularization technique commonly used for intermediate layers of CNNs is Dropout [6]. The process of its implementation can be simply described as randomly removing neurons in each training epoch. Such random operations distribute the feature selection power evenly across the entire set of neurons while forcing the model to learn different independent features. Dropout is determined differently for the training process and the testing process [2]. Neurons that are dropped in the training process do not become part of the backpropagation or forwardpropagation. In the testing process, on the other hand, predictions are made based on the entire network. Similar approaches to Dropout include DropFilterR [7], Droppath [41], and Dropblock [42], as well as several variants based on the Dropout formulation, e.g., SpatialDropout [43] and Dropconnect [44].

In this paper, we propose a method for dynamical adjustment of the training process of CNNs. The method can be used not only in combination with other regularization methods, but also for optimal adjustments of the training process of CNNs. The regularization process of the method is dynamic and controllable. The main components of the method are described in the next sections.

3 The proposed method

In this section, we present our proposed dynamic adjustment method, whose general framework is shown in Fig. 1. It consists of three main parts, namely the loss memory module (the LM module), the transfer operator, and the dynamic adjustment module (the DA module). During the training process, the LM module automatically records the training loss generated for each new iteration and generates the disturbance variables for the next iteration of the training by the control parameter R in the module. The newly generated disturbance variables are input to the DA module embedded in the fully-connected layer through the transfer operator to participate in the adjustment of the next training process. Throughout the process, the disturbance variables change as the training loss changes. In addition, the parameter R can be adjusted based on different models to obtain the best training process setting. The individual components of the method are described in detail below.

3.1 The LM module

As mentioned above, the dynamic adjustment intensity of this method is consistent with the changing trend of training loss. To obtain this variation trend, the LM module that automatically records the loss generated in each new iteration is constructed in this method, and the adjustment intensity of the next iteration training is also generated. We assume that the convolutional neural network has trained epochs I in total ($I >= 1$), and I represents the i-th iteration of epochs I training. Accordingly, this paper assumes that the training loss of convolutional neural network training is $L \in \mathbb{R}^+$, then the training loss of the i-th iteration training can be expressed as l_i. In addition, we represent l_0 as training loss for initial training, and $l_0 = 0$.

After the first iteration of the convolutional neural network is trained, the corresponding training loss is calculated by the loss function and will be input to the LM module by the pass operator, e.g., the first epoch training loss value is l_1.

We use the function $M(.)$ to represent the calculation process of the LM module for the training loss, and use X_i^{loss} to represent the LM module in the i-th iteration of training to generate a disturbance variable, the value of which is the adjusted intensity value, to obtain the following functional representation as follows:

$$X_i^{Loss} = M(l_{i-1}) \tag{1}$$

where, X_i^{Loss} is the adjusted intensity value produced by the (i-1)-th iteration of training loss l_{i-1}. Equation (1) describes the current training loss memory process, which reflects the dynamic and adaptive nature of LossDA and is also a mathematical description of the LM module.

3.2 The R parameter

In practical use, the training process is different for each model, and in order to obtain the best disturbance variables, we need to adjust the output size in the LM module. In this paper, we use the R parameter to adjust the optimal value of the output disturbance variable in the LM module. In addition, the setting of the parameter R allows the disturbance variables to be adjusted only in the training phase of the CNNs, but not in the testing phase. Here, the parameter R is set so that the dynamic adjustment can be divided into two states.

The first one is a non-zero state, which is generally used in the training phase. As shown in Fig. 2a, when $R \neq 0$, the LM module output X_i^{Loss} is calculated as described before as follows:

$$X_i^{Loss} = M(l_{i-1}, R) \tag{2}$$

where l_{i-1} is the training loss of the (i-1)-th iteration, X_i^{Loss} denotes the disturbance variable of the i-th iteration training, and the range of R values is set to [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. When the CNNs start training, the disturbance variables are generated after the loss of each iteration of training by the LM module. And the individual process of disturbance by passing the operator into the DA module of the fully-connected layer is shown in Fig. 3. Equation (2) is the training loss the adjustment process according to the hyperparameter R. It shows the adjustability of LossDA and is a mathematical description of the LM module with the addition of the hyperparameter R.

The second one is the zero state, $R = 0$, which is generally used in the testing phase of CNNs, as shown in Fig. 2b. when the LM module outputs $X_i^{Loss} = 0$.

3.3 The DA module

From Fig. 3, we can see that the process by which the LM module generates new disturbance variables at each iteration of training and inputs them to DA module by passing the operator is only the first step of our task. Next, we need to integrate the disturbance variables into the training data. In this section, we describe how the DA module embedded in the fully-connected layer is dynamically adjusted. Once the disturbance variables generated by the LM module are integrated into the training data, the new training data can contain two different kinds of data, namely the data of the training data itself and the data of the disturbance variables. If we use the function $F(.)$ to represent all of the convolutional neural network operations before the activation function in the fully-connected layer, then the training data on the fully-connected layer can be represented as follows:

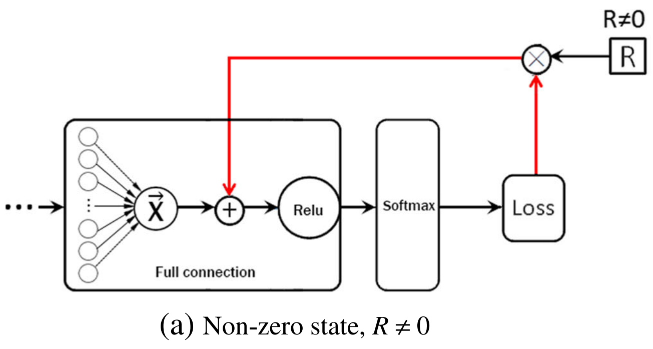
$$\vec{X}_i = F(X) \tag{3}$$

where X denotes the input data of the image, and $\vec{X}_i$ denotes the input data of the activation function of the fully-connected layer at the i-th iteration, which is equivalent to the training data of the fully-connected layer.

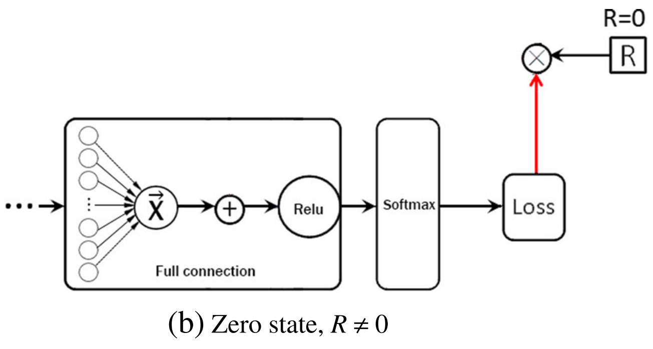
In particular, we set the function $A(.)$ to represent the operation process of the DA module, and the training data $\vec{X}$ here can be expressed after the adjustment of the DA module as follows:

$$\vec{X}_i^A = A(\vec{X}_i, X_i^{Loss}) \tag{4}$$

where $\vec{X}_i^A$ denotes the training data at the i-th iteration, after adjustment by the DA module. X_i^{Loss}, as described in the previous content above, is the disturbance variable generated by the LM module for the i-th iteration of training, which is input to the DA module through the transfer operator.



(a) Non-zero state, $R \neq 0$



(b) Zero state, $R \neq 0$

Fig. 2 The state of the LM module with different R parameters set during the training phase and the test phase

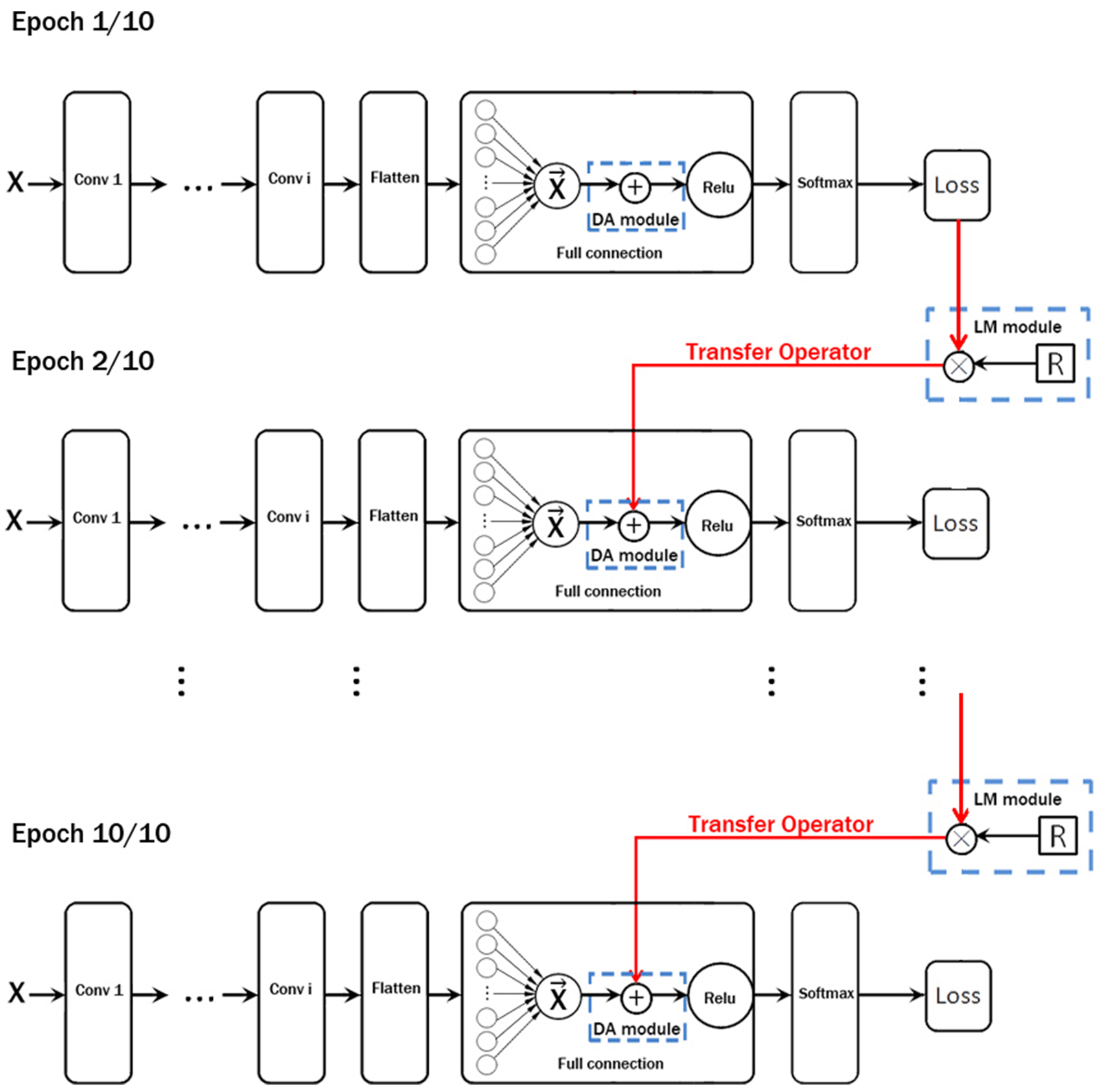


Fig. 3 The dynamic adjustment process of LossDA with 10 iterations of training. The transfer operator will iterate to generate the disturbance variables

As can be seen from (4), the function $A(.)$ contains two variables, $\vec{X}_i$ and X_i^{Loss}, representing the two inputs to the DA module, i.e., the training data and the disturbance variables, respectively. The new training data $\vec{X}_i^A$ is generated after calculation and input to the activation function for operation. Equation (4) is a description of the adjustment process of DA module to the internal structure of CNNs, i.e., a description of the adjustment process of DA module to the fully connected layer of CNNs.

Considering that the disturbance variables generated by the LM module are computed from the training loss of the previous iteration, the input data of the final activation function on the fully-connected layer is obtained by substituting (2) and (3) into (4) as follows:

$$\vec{X}_i^A = A(F(X), M(l_{i-1}, R)) \tag{5}$$

Equation (5) shows that the $\vec{X}_i^A$ integrated by the i-th iteration of the DA module training is determined primarily by X, $F(.)$, R, and l_{i-1}. By setting it up in this way, the training loss of the (i-1)-th iteration is associated with the data training of the i-th iteration accordingly. Equation (5) is a description of the adjustment process of the DA module with respect to the external structure of the CNNs, i.e., the mathematical description of the image X subjected to the adjustment of the DA module during the training process.

In the calculation of the DA module, this paper uses the addition operation as the integration rule of the DA module, and the corresponding $\vec{X}_i^A$ can be expressed as follows:

$$\vec{X}_i^A = F(X) + M(l_{i-1}, R) \tag{6}$$

As shown in (6), the disturbance variables specific to the input of the transfer operator become the new training data with the training data set in the fully-connected layer through the DA module. The disturbance variables change as the training loss changes and adaptively and dynamically adjust the input values of the activation function in the fully-connected layer. The transfer operator is run a total of $I - 1$ times in the entire convolutional neural network model.

When the new training data is subjected to the activation function operation, here, let the output be Z_i, which can be expressed as:

$$Z_i = Relu(F(X) + M(l_{i-1}, R)) \tag{7}$$

where the activation function at this point is the Relu function [46].

To better demonstrate the role of the DA module in convolutional neural networks, the new training loss, l_i, with softmax function [45] and loss function calculation, can be expressed as:

$$l_i = Loss(softmax(relu(F(X) + M(l_{i-1}, R)))) \tag{8}$$

From (8), it can be seen that l_i is influenced by the image data X and l_{i-1}, and if (2) and (3) are substituted into (8), we get the following:

$$l_i = Loss(softmax(relu(\vec{X}_i + X_i^{Loss}))) \tag{9}$$

From (8) and (9), it can be seen that when the image data X is trained for the i-th iteration, the training data $\vec{X}_i$ generated in the fully-connected layer is adjusted by the disturbance variables generated by the LM module for the $i - 1$ iteration, and then the new training loss l_i is generated after the activation function, softmax function, and loss function are calculated. In the above training process, the training data of all iterations is adjusted in the DA module, except for the training data of the first iteration, which is not adjusted by the DA module. This is the whole dynamic adjustment process of the method in this paper and its corresponding programming content, as shown in Algorithm 1.

```
1: Input: data $X$, number of epoch $I$ and parameter $R$ of LM module
2: Output: Dynamic adjusting the CNN training process
3: Initialize $l_0 = 0$, $R = [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]$, in the LM module
4: for $i \in 0, 1, \ldots, I$ do
5:     Get CNN Loss function value $l_{i-1}$;
6:     Calculate disturbance variable $X_i^{Loss}$ via (2);
7:     Generate the i-th iteration data $\vec{X}_i$ at the fully-connected layer via (3);
8:     Calculate the new input data $\vec{X}_i^A$ by $X_i^{Loss}$ and $\vec{X}_i$ via (6);
9:     Generate the new function value $l_i$ via (9);
10: end for
11: $R = 0$
```

Algorithm 1 Training process of LossDA.

The content of the pseudo-code describes the entire process of dynamic adjustment of LossDA. The entire content of the pseudo-code consists mainly two parts. The first part is the definition of the inputs and outputs and the initializations of the relevant parameters. This part corresponds to lines 1-3 of the pseudo-code. The second part is the For loop, which describes the arithmetic process of the code from the previous loss value l_{i-1} to the current loss value l_i, where each line of code specifies the corresponding mathematical formula. This paper focuses on the For loop, i.e., lines 5–9 of the pseudo-code. Line 5 of the pseudo-code is used to determined the loss value of the previous epoch l_{i-1}. Line 6 of the pseudo-code is based on the disturbance variable X_i^{Loss} generated by (2), where the input to (2) is l_{i-1}. Line-7 of the pseudo-code is based on (3) to generate the i-th iteration data $\vec{X}_i$ in the fully-connected layer, which is also the training data to be dynamically adjusted by the DA module. Line-8 of pseudo-code is based on (6) to calculate the new data $\vec{X}_i^A$, which is the data to be dynamically perturbed by the DA module. Line-9 of the pseudo-code is based on (9) to calculate the new loss value l_i for the current iteration. Line-10 of the pseudo-code represents the end of training, while in line 11 the pseudo-code assigns the value 0 to the hyperparameter R, i.e., the zero state of the hyperparameter R in the test phase. The arithmetic process of the above pseudo-code can be useful for understanding the dynamic adjustment process of the method proposed in this paper.

4 Experiments

This section presents the datasets and experimental details.

4.1 Datasets

To evaluate the method in this paper, the datasets were used for the experiments: MNIST [8], FashionMNIST [9], CIFAR10 [10], Cats_vs_Dogs [11], and miniImagenet [12]. MNIST is a 0–9 digital image dataset consisting of 60,000 training samples and 10,000 test samples, each of which is a handwritten 28×28 grayscale digital image. FashionMNIST is an image dataset provided by Zalando's research department, comprising a total of 70,000 frontal images of different items in 10 categories, with 60,000/10,000 training and test data and 28 × 28 grayscale images. CIFAR10 is a small universal objects recognizing dataset that includes 60,000 images in 10 categories, 50,000/10,000 training and test data, and 32 × 32 colour images. Cats_vs_Dogs is a dataset of images containing dogs and cats. It includes 25,000 images of cats and dogs, 17500/7500 training and test data, and 84 × 84 colour images. The miniImagenet dataset is selected from the ImageNet dataset [47] and

Table 1 The statistics of the datasets

Datasets	Type	Size
MNIST	Image	28 × 28 Gray
FashionMNIST	Image	28 × 28 Gray
CIFAR10	Image	32 × 32 Color
Cats_vs_Dogs	Image	84 × 84 Color
miniImagenet	Image	84 × 84 Color

contains 100 classes with 60,000 colour images, each class including 600 samples, 42,000/18,000 training and test data, and 84 × 84 color images (Table 1).

4.2 Experiments details

The experimental details mainly present the hyperparameter settings of the experiments, the CNN architectures of the training datasets, and the evaluation metrics of the test results.

A. Hyperparameters Settings

For all CNN architectures of the five datasets, three experiments with different numbers of epochs (5, 10, and 15) were performed, and each experiment was repeated 15 times. In these experiments, the initial value of the learning rate was set to 10^{-4}, the batch size of each input was set to 64, and the activation functions were all used with the Relu function [46]. For the CNNs with LossDA, the values of the parameter R ranged from [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5] during the training phase to $R = 0$ at the end of training. In addition, for the CNNs with the transfer learning model, the weight parameters are also set to the untrained mode [48] using ImageNet [49].

B. CNN Architectures

The experiments were conducted on five datasets with different CNNs, respectively, and the specific information is shown in Table 2.

For the MNIST, FashionMNIST, and CIFAR10 datasets, three Light CNNs were used, named CNN-1, CNN-2, and CNN-3, respectively. CNN-1 uses three convolutional layers (output dimensions 32,64,128), two pooling layers (pooling size of from top[2, 2]), and one fully-connected layer. CNN-2 adds 2 BN to CNN-1, but uses only 1 pooling layer. CNN-3 is based on CNN-2, increasing the number of convolutional layers from 3 to 6, while increasing the number of pooling layers from 1 to 4.

For the Cats_vs_Dogs and miniImagenet datasets, we used a total of four Transfer CNNs (InceptionResNetV2 [13], VGG19 [14], ResNet50 [15], and InceptionV3 [16]), referred to as CNN-InceptionResNet, CNN-VGG19, CNN-ResNet50 and CNN-InceptionV3, respectively. For the Cats_vs_Dogs dataset, CNN-InceptionResNet, CNN-VGG19, and CNN-ResNet50 were used, respectively, with a fully-connected layer and an output dimension of 512. In the miniImagenet dataset, CNN-InceptionResNet, CNN-VGG19, and CNN-InceptionV3 were used respectively, with the output dimension of their fully-connected layers being 512. The difference between the architectures used in the two datasets is that Cats_vs_Dogs uses CNN-ResNet50, while miniImagenet uses CNN-InceptionV3. This is because miniImagenet has a smaller number of single-class images, and after testing, it was found to be unsuitable for using CNN-ResNet50, so it was switched to CNN-InceptionV3. CNN-InceptionV3 was used instead. The above 4 Transfer CNNs are implemented using TensorFlow framework [50].

C. Evaluation Metrics

To better evaluate the model performance of CNNs, four common evaluation metrics are used for all test results in this paper, which are accuracy (ACC), F1-score (F1), recall (R), and precision (P). In addition, the values of the evaluation indices are based on the average of the results of 15 repeated experiments. Below is the formula for calculating the four metrics:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

Table 2 CNNs of the datasets

Datasets	CNNs	Model Architectures
MNIST	CNN-1	3Conv+2MaxPool+1FC
FashionMNIST	CNN-2	3Conv+1BN+1MaxPool+1BN+1FC
CIFAR10	CNN-3	6Conv+1BN+4MaxPool+1BN+1FC
Cats_vs_Dogs	CNN-InceptionResNet	InceptionResNetV2+1FC
Cats_vs_Dogs	CNN-VGG19	VGG19+1FC
Cats_vs_Dogs	CNN-ResNet50	ResNet50+1FC
miniImagenet	CNN-InceptionResNet	InceptionResNetV2+1FC
miniImagenet	CNN-VGG19	VGG19+1FC
miniImagenet	CNN-InceptionV3	InceptionV3+1FC

$$recall = \frac{TP}{TP + FN} \quad (11)$$

$$precision = \frac{TP}{TP + FP} \quad (12)$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \quad (13)$$

where TP and TN are the positive and negative samples that are correctly predicted, and FP and FN are the positive and negative samples that are incorrectly predicted, respectively.

5 Results and discussion

In this section, we demonstrate and discuss the effect of LossDA regularization using various results. These results include optimal comparison ones for different datasets on different architectures, comparison ones for different training epochs, test accuracy for different *R* parameters, and the training process for different datasets.

5.1 Comparison of testing results

The statistical results in Table 3 are based on comparative experiments using the LossDA method on seven CNNs for each of the five datasets. The seven architectures are divided into three Light CNNs (CNN-1, 2, and 3) and four Transfer CNNs (CNN-InceptionResNet, VGG19, ResNet50, and InceptionV3). CNNs marked with "with LossDA" represent CNN architectures that use the LossDA method. Test results were determined using four metrics (ACC, F1, R, and P). Among the three lightweight architectures, the CNN-3 architecture using CIFAR10 dataset showed the highest improvement. The results (%) without LossDA were 71.31, 71.91, 71.31, and 74.14. The results (%) with LossDA were 75.93, 75.9, 76, and 76.08. The metric values (%) improved by 4.62, 3.99, 4.69, and 1.94, respectively. In addition, among the four transfer architectures, the CNN-ResNet50 architecture using the Cats_vs_Dogs dataset showed the highest improvement. The results (%) without LossDA were 67.39, 65.83, 67.49, and 71.48, respectively. The results (%) with LossDA were 71.56, 71.47, 71.54, and 71.79, respectively. The metric values (%) improved by 4.17, 5.64, 4.05, and 0.31, respectively.

The CNN architectures for the five dataset are Light CNNs and Transfer CNNs. The Light CNNs (CNN-1,2,3) mainly consist of convolutional layers, pooling layers and fully-connected layers, with the addition of BN to the CNN-2,3 models, which are among the most commonly used light models. The basic models of Transfer CNNs used for the experiments are InceptionResNet, VGG19, ResNet50 and InceptionV3, all with ImageNet weights [47], which are among the better known transfer models. In addition, Cats_vs_Dogs and miniImagenet which were used for Transfer CNNs are both representative datasets. The former is a dataset with few classifications but many single-class images (2 classes, 12,500 images/class) and the latter is a dataset with many classifications but few single-class images (100 classes, 600 images/class), that have opposite characteristics. From these analyses, it appears that the compared experiments are diverse and representative, which helps to comprehensively evaluate the effectiveness of LossDA applications.

Table 3 Comparison test results of five datasets on each of the seven CNN architectures using the LossDA method

Datasets	CNNs	ACC[%]	F1[%]	R[%]	P[%]
MNIST	CNN-1 with LossDA	**99.36**	**99.36**	**99.36**	**99.36**
MNIST	CNN-1	97.90	97.88	97.88	97.97
FashionMNIST	CNN-1 with LossDA	**92.42**	**92.40**	**92.39**	**92.51**
FashionMNIST	CNN-2	91.09	91.03	91.09	91.19
CIFAR10	CNN-2 with LossDA	**75.93**	**75.90**	**76.00**	**76.08**
CIFAR10	CNN-2	71.31	71.91	71.31	74.14
Cats_vs_Dogs	InceptionResNet with LossDA	**87.04***	**87.04***	**87.05***	**87.06***
Cats_vs_Dogs	InceptionResNet	85.47	85.53	85.56	85.71
Cats_vs_Dogs	VGG19 with LossDA	84.00*	84.04*	84.04*	84.17*
miniImagenet	VGG19	**83.40**	**83.38**	**83.40**	**83.46**
miniImagenet	ResNet50 with LossDA	71.56*	71.47*	71.54*	71.79*
miniImagenet	ResNet50	67.39	65.83	67.49	71.48

The evaluation metrics for this result include: accuracy, F1, recall, and precision, abbreviated as ACC, F1, R, and P. The higher the metric value, the better. Bold numbers indicate the best value on the same dataset; asterisks are used to compare multiple models on the same dataset; and numbers with asterisks indicate the best value among multiple models. The statistical values in the table are the average of the results of 15 tests, where the number of training epochs for each test was 15

The comparison of the test results confirms the original intention of using the LossDA method on multiple CNN architectures, and LossDA can be used not only on Light CNNs but also on the currently popular Transfer CNNs. All the comparison results demonstrate that the method can improve the model performance of the CNN architectures used for the experiments. It is believed that such a regularization method can be used for other CNN architectures. This is because our method is based on the application of a fully-connected layer, which most CNN architectures have. The LossDA method, on the other hand, does not change the architecture of the model, and the iterative loss it uses is also generated during the training process. LossDA is simply embedded in the fully-connected layer and dynamically adjusts the CNN training process. This is like adding a regulatory device to the CNN architecture, that is dynamically adjustable, non-random, and controllable.

5.2 Comparison of different epochs

To further verify the adjustment effect of LossDA, this paper compares the trend of the test scores in different epochs, as shown in Fig. 4. The trend of the curves in the subplots shows that the red line is higher than the yellow line in all subplots, and both the red and yellow lines show an increasing trend in most subplots. The only difference is that the yellow line in subplot "(a) MNIST" shows a decreasing trend when the number of training epochs is 15, while the red line continues to increase at a steady rate.

The trend of the curves shows the regularization effect of the LossDA method at different epochs. On the whole, the red line with LossDA is better than the yellow line without LossDA in terms of model performance. On the other hand, the red and yellow lines of most subplots show an increasing trend, indicating that the training process is not overfitting. Moreover, the trends of the red and yellow lines in each subplot of (a) MNIST are different. This shows that the generalization performance of the model with LossDA is much higher than that of the model without LossDA when the number of training epochs is 15.

The trend in Fig. 4 again proves the regularization effect of LossDA. The comparison results show that the dynamic adjustment method proposed in this paper is an adaptive adjustment process. This is because the adjustment of LossDA is based on the number of training epochs, the number of dynamic adaptations increases with the number of epochs. Meanwhile, the adjustment intensity changes with the change in training loss, thus adapting to the overall training process. Moreover, the anomaly in "(a) MNIST" is due to the fact that the model with the yellow line has an overfitting problem, while the model with the red line has no overfitting due to the use of LossDA, which fully demonstrates the suppression of overfitting by the method in this paper.

5.3 Analysis of the R parameter

In this section, experiments related to the parameter R are presented. The experiments were mainly performed with the parameters $R = [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]$, and the corresponding statistical analysis was performed as shown in Fig. 5. The test accuracy of $R = 0$ is set as the benchmark accuracy, and when the test accuracy is higher than the benchmark accuracy, the corresponding non-zero R values are called available R values.

The results in Fig. 5 show that the most available R values are for the MNIST and FashionMNIST datasets, with $R = [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]$, followed by the CIFAR10 and Cats_vs_Dogs datasets, with R values of 0.5, 1, 2, and 1.5, 2, 2.5, respectively. The miniImagenet datasets with $R = 0.5$ have the least available.

These results show that the larger the range of available R values, the better the regularization effect of selecting the best value. In the five datasets, MNIST and FashionMNIST with $R = [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]$, almost all values are available with higher than the baseline value ($R = 0$), indicateing that a wide range of R values is available in these two datasets. In contrast, for the CIFAR10 and Cats_vs_Dogs datasets, only some of their parameters R are available ($R = [0.5, 1, 2, 1.5, 2, 2.5]$, respectively), and the rest are below the corresponding benchmark values, indicating that there is a limit to the range of available R values. The lowest is the miniImagenet dataset, where the available R-value is only 0.5 and all other R-values are below the benchmark, indicating that the available R-values in this dataset are concentrated in a smaller range and need more tests are required to find them.

The presence of available R-values again indicates the original purpose of this method. In the experiments, available R values were found for all datasets, which shows that LossDA can be used to improve the performance metrics of the model. Moreover, we can select the ideal value from these available R values to maximize the improvement of model performance. On the other hand, some values of the parameter R are not available in the experiments, which may be due to factors such as the image size or the number of single-class images in different datasets that make the test accuracy lower than the benchmark accuracy, such as the number of single-class images (600 images/class) in the miniImagenet dataset, which is relatively less than other datasets. The experimental results show that the available R values can be found for different datasets, but only the range of useful values varies.

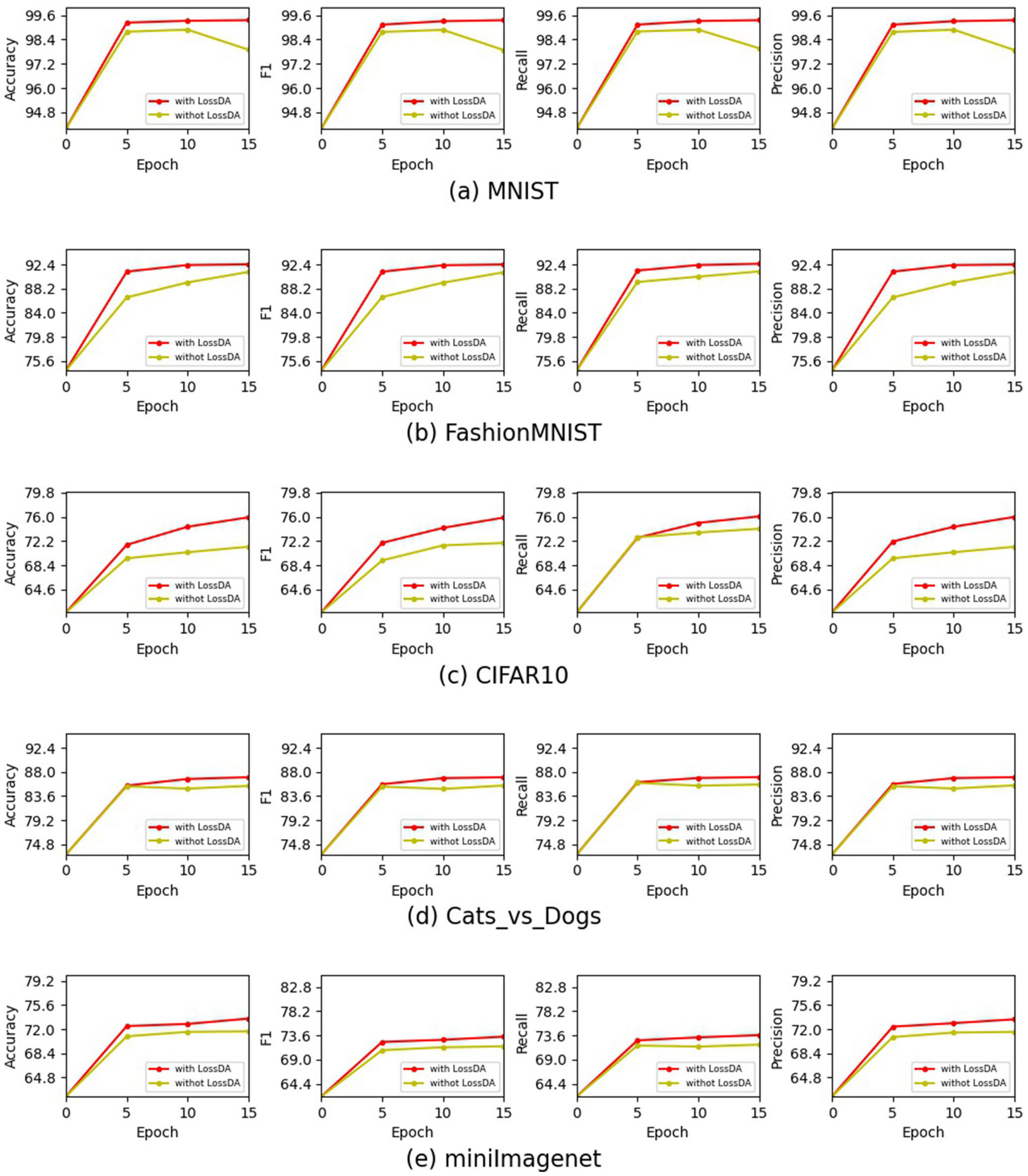


Fig. 4 Trends of the tested metrics over different epochs. The comparison experiment was performed on five datasets: (a) MNIST, (b) FashionMNIST, (c) CIFAR10, (d) Cats_vs_Dogs, and (e) miniImagenet. The horizontal coordinates of each subplot are the number of training epochs, denoted by 0, 5, 10, and 15, respectively. The vertical coordinates of the subplots are the four metrics (accuracy, F1, recall, and precision). The CNN architectures for the MNIST, FashionMNIST, and CIFAR10 datasets are CNN-1,2, and 3, respectively, and the CNN architectures for the Cats_vs_Dogs and miniImagenet datasets are CNN-InceptionResNet

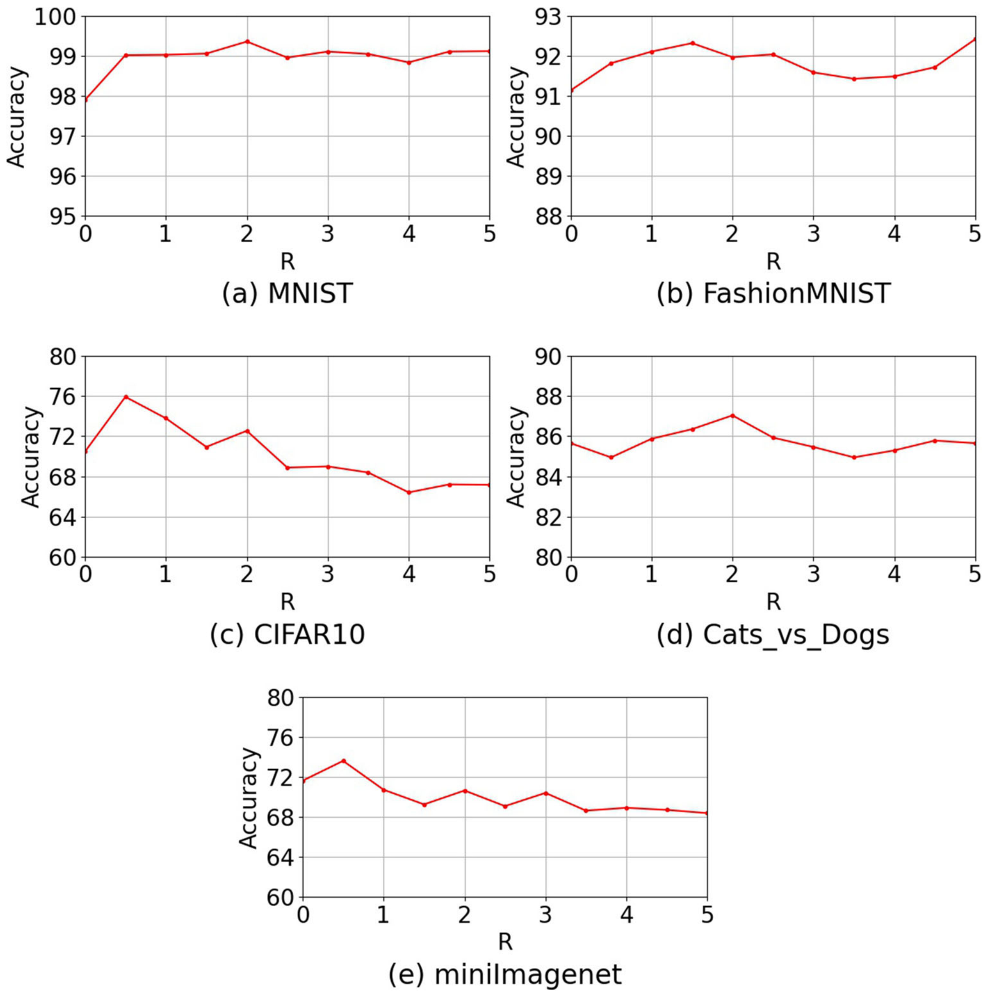


Fig. 5 Test accuracy for different R parameters. The five datasets of the experiment are numbered as (a) MNIST, (b) FashionMNIST, (c) CIFAR10, (e) Cats_vs_Dogs, and (e) miniImagenet. The horizontal coordinates of each subplot are the R parameters. The scale of the coordinate values is 0,1,2,3,4,5, and the vertical coordinates are the test accuracy values, where $R = 0$ represents the test accuracy when the model does not use the LossDA method and is set as the baseline accuracy. The CNN models used for the five data sets are the same as those used for the model in Section 5.2

5.4 Comparison of training process

In this section, the regularization effect of LossDA during training is presented, as shown in Fig. 6. In the figure, the red line is higher than the yellow line in all subplots. Among them, (b) FashionMNIST and (c) CIFAR10 subplots have the most red lines higher than the yellow line, and the red line shows an upward trend, followed by (d) Cats_vs_Dogs and (e) miniImagenet subplots, both of which have red lines higher than the yellow line, and the least is (a) MNIST subplot, which has a slightly higher red line than the yellow line.

The trend of each subplot curve in Fig. 6 shows that LossDA has the best regularization effect for the FashionMNIST and CIFAR10 datasets, indicating that the accuracy of the model with LossDA is much higher than that of the model without LossDA. First, when the number of iterations increases, the training accuracy also improve, and the corresponding test results also improve. This is because LossDA is an adaptive dynamic adjustment method, and the results of the FashionMNIST and CIFAR10 datasets in Section 5.2 at 10 and 15 epochs of testing exactly confirm this assumption. Second, the red lines of the subplots are higher than the yellow lines in both the Cats_vs_Dogs and miniImagenet datasets, but not as good as the plot of the previous two datasets, because the two datasets use the transfer model and the weights are in untrained mode, which greatly reduces the role of the method in this paper to adjust the training process. However, as for the overall training process of the transfer model, the accuracy of the model with LossDA is still higher than that of the model without LossDA, indicating that LossDA is still effective for the training process of the transfer model. In addition, the training accuracy on the MNIST dataset improves the least, which may be due to the upper bound of training accuracy. The later training accuracy has reached the upper limit of 99-100%. Nevertheless, the red line is still slightly higher than the yellow line on the MNIST subplot in (a). The above experimental results show that LossDA not only improves the accuracy values of the training process, but also has better adaptability to different training models.

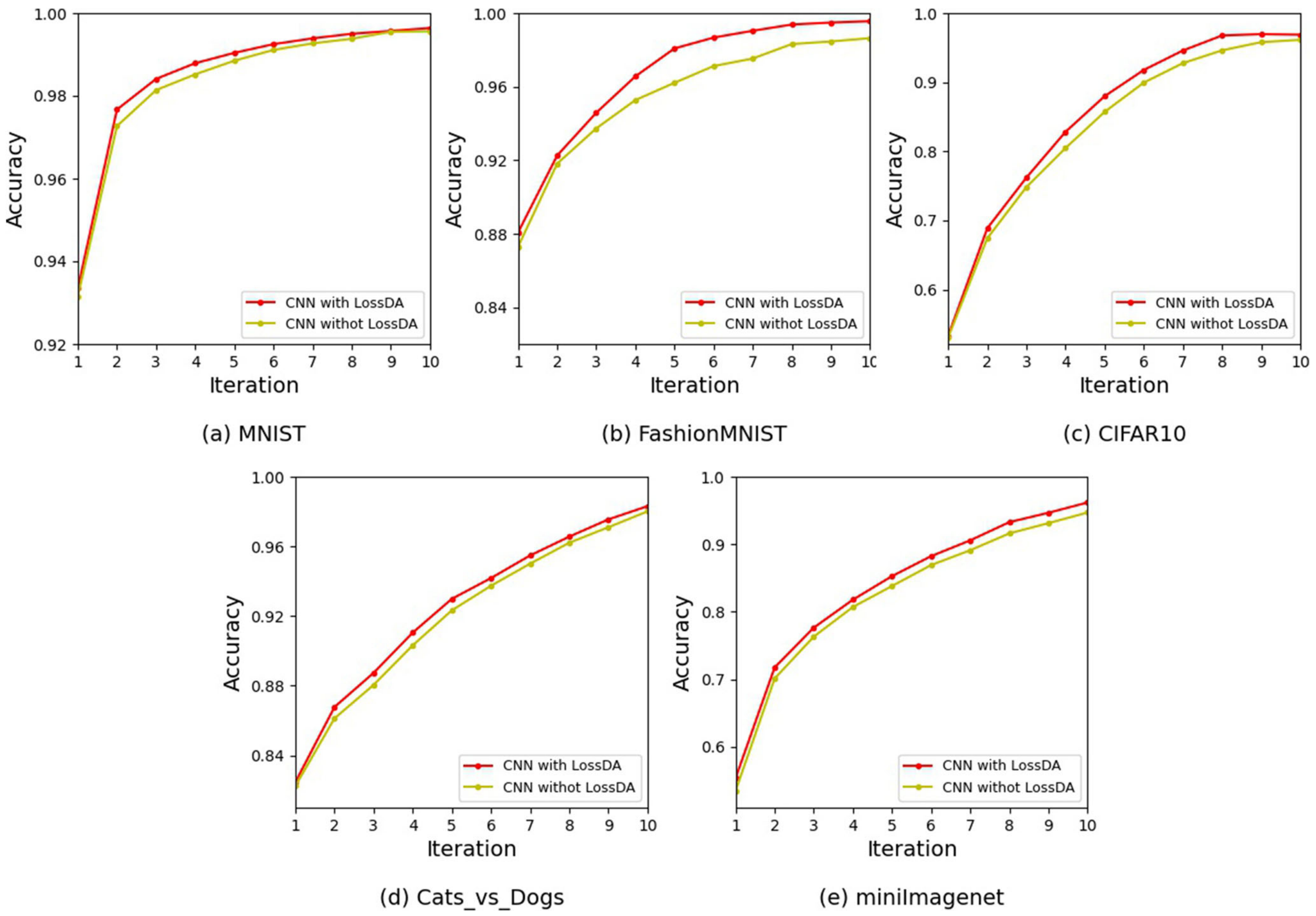


Fig. 6 Adjustment Effect of LossDA in the Training Process. The figure shows five datasets numbered (a) MNIST, (b) FashionMNIST, (c) CIFAR10, (e) Cats_vs_Dogs, and (e) miniImagenet. The horizontal coordinates of each subplot are the number of training iterations (1–10), and the vertical coordinates are the training accuracy (%). The plot shows two types of curves. The red line means that the model uses the LossDA method, while the yellow line means that the model does not use the LossDA method. The CNN models used here for the five datasets are consistent with the models in Section 5.2

6 Conclusion

In this paper, we propose a dynamically controllable adjustment method called LossDA. The method is embedded in the fully-connected layer to adaptively adjust the training process of the CNN model. Its adjustment intensity can be varied with the training loss and the magnitude of the adjustment can be preset to achieve the best adjustment effect on the training process. The method improves the generalization performance of the model and can help to suppress overfitting during training. To evaluate the method, this paper conducts extensive comparison experiments with different CNN architectures, including the commonly used Light CNNs and the currently popular Transfer CNNs (InceptionResNet, VGG19, ResNet50 and InceptionV3). The experimental results show that this dynamically controllable approach not only improves the model performance of various CNN architectures (highest improvement in accuracy of 4.62%, F1 of 5.64%, and recall of 4.69%), but also effectively suppresses the overfitting problem. LossDA can be applied to almost any CNN architecture with a fully-connected layer. However, it is also limited in that it needs to be embedded in a fully connected layer to work. In future work, we hope to break this limitation so that this dynamically controllable tuning method can be widely applied to a wider range of network architectures.

Acknowledgements The authors gratefully acknowledge the financial supports by the National Science Foundation of China under Grant number 61662072, and Yunnan Agricultural Basic Research Joint Special Project number 202101BD070001-058.

Author Contributions Juncheng Liu conceived the study, collected the datasets and drafted the manuscript. Yili Zhao provided some ideas and tool support, and supervised the study and provide article guidance. All authors read and approved the final manuscript.

Funding This work was supported by the National Natural Science Foundation of China (Grant No.61662072), and Yunnan Agricultural Basic Research Joint Special Project (Grant No.202101BD070001-058).

Code Availability The codes used during the study are available from the corresponding author by request.

Availability of data and materials Data related to the current study are free access from Internet.

Declarations

Conflict of Interests The authors declare no conflict of interest.

References

1. Mittal S, Srivastava S, Jayanth JP (2022) A survey of deep learning techniques for underwater image classification. IEEE Trans Neural Netw Learn Syst
2. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L (2021) Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. J Big Data 8(1):1–74
3. Ji Y, Zhang H, Zhang Z, Liu M (2021) Cnn-based encoder-decoder networks for salient object detection: a comprehensive review and recent advances. Inf Sci 546:835–857
4. Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):1–48
5. Li Y, Wang N, Shi J, Hou X, Liu J (2018) Adaptive batch normalization for practical domain adaptation. Pattern Recogn 80:109–117
6. Achille A, Soatto S (2018) Information dropout: Learning optimal representations through noisy computation. IEEE Trans Pattern Anal Machine Intell 40(12):2897–2905
7. Pan H, Niu X, Li R, Shen S, Dou Y (2020) Dropfilterr: a novel regularization method for learning convolutional neural networks. Neural Process Lett 51(2):1285–1298
8. Castro DC, Tan J, Kainz B, Konukoglu E, Glocker B (2019) Morpho-mnist: quantitative assessment and diagnostics for representation learning. J Mach Learn Res 20(178):1–29
9. Yu S, Wickstrøm K, Jenssen R, Príncipe JC (2020) Understanding convolutional neural networks with information theory: An initial exploration. IEEE Trans Neural Netw Learn Syst 32(1):435–442
10. Li G, Zhang M, Li J, Lv F, Tong G (2021) Efficient densely connected convolutional neural networks. Pattern Recogn 109:107610
11. Rey-Area M, Guirado E, Tabik S, Ruiz-Hidalgo J (2020) Fucitnet: Improving the generalization of deep learning networks by the fusion of learned class-inherent transformations. Information Fusion 63:188–195
12. Wu Z, Zhao H (2022) Hierarchical few-shot learning based on coarse-and fine-grained relation network. Artif Intell Rev 1–20
13. Peng S, Huang H, Chen W, Zhang L, Fang W (2020) More trainable inception-resnet for face recognition. Neurocomputing 411:9–19
14. Whitney HM, Li H, Ji Y, Liu P, Giger ML (2019) Comparison of breast mri tumor classification using human-engineered radiomics, transfer learning from deep convolutional neural networks, and fusion methods. Proc IEEE 108(1):163–177
15. Wen L, Li X, Gao L (2020) A transfer convolutional neural network for fault diagnosis based on resnet-50. Neural Comput Applic 32(10):6111–6124
16. Liu Z, Yang C, Huang J, Liu S, Zhuo Y, Lu X (2021) Deep learning framework based on integration of s-mask r-cnn and inception-v3 for ultrasound image-aided diagnosis of prostate cancer. Futur Gener Comput Syst 114:358–367
17. Li Z, Liu F, Yang W, Peng S, Zhou J (2021) A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans Neural Netw Learn Syst
18. Erdenebayar U, Kim H, Park J-U, Kang D, Lee K-J (2019) Automatic prediction of atrial fibrillation based on convolutional neural network using a short-term normal electrocardiogram signal. J Korean Med Sci 34(7)
19. Harbola S, Coors V (2019) One dimensional convolutional neural network architectures for wind prediction. Energy Convers Manag 195:70–75
20. Abdeljaber O, Sassi S, Avci O, Kiranyaz S, Ibrahim AA, Gabbouj M (2018) Fault detection and severity identification of ball bearings by online condition monitoring. IEEE Trans Ind Electron 66(10):8136–8147
21. Jiang Y, Chen L, Zhang H, Xiao X (2019) Breast cancer histopathological image classification using convolutional neural networks with small se-resnet module. PloS One 14(3):0214587
22. Zhang M, Li W, Du Q (2018) Diverse region-based cnn for hyperspectral image classification. IEEE Trans Image Process 27(6):2623–2634
23. Ji Y, Zhang H, Zhang Z, Liu M (2021) Cnn-based encoder-decoder networks for salient object detection: a comprehensive review and recent advances. Inf Sci 546:835–857
24. Minaee S, Boykov YY, Porikli F, Plaza AJ, Kehtarnavaz N, Terzopoulos D (2021) Image segmentation using deep learning: a survey. IEEE Trans Pattern Anal Machine Intell
25. Jeevan G, Zacharias GC, Nair MS, Rajan J (2022) An empirical study of the impact of masks on face recognition. Pattern Recogn 122:108308
26. Li C, Huang Y, Huang W, Qin F (2021) Learning features from covariance matrix of gabor wavelet for face recognition under adverse conditions. Pattern Recogn 119:108085
27. Huang J, Zhou W, Li H, Li W (2018) Attention-based 3d-cnns for large-vocabulary sign language recognition. IEEE Trans Circuits Syst Video Technol 29(9):2822–2832
28. Bi Z, Huang W (2021) Human action identification by a quality-guided fusion of multi-model feature. Futur Gener Comput Syst 116:13–21
29. Islam MS, Bakhat K, Khan R, Iqbal M, Islam MM, Ye Z (2021) Action recognition using interrelationships of 3d joints and frames based on angle sine relation and distance features using interrelationships. Appl Intell 51(8):6001–6013
30. Huang J, Yan W, Li G, Li T, Liu S (2021) Learning disentangled representation for multi-view 3d object recognition. IEEE Trans Circuits Syst Video Technol 32(2):646–659
31. Karthik R, Menaka R, Hariharan M, Won D (2022) Contour-enhanced attention cnn for ct-based covid-19 segmentation. Pattern Recogn 125:108538
32. Castiglione A, Vijayakumar P, Nappi M, Sadiq S, Umer M (2021) Covid-19: automatic detection of the novel coronavirus disease from ct images using an optimized convolutional neural network. IEEE Trans Industrial Inform 17(9):6480–6488

33. Ahuja S, Panigrahi BK, Dey N, Rajinikanth V, Gandhi TK (2021) Deep transfer learning-based automated detection of covid-19 from lung ct scan slices. Appl Intell 51(1):571–585
34. La Grassa R, Gallo I, Landro N (2021) Learn class hierarchy using convolutional neural networks. Appl Intell 51(10):6622–6632
35. Yuesheng F, Jian S, Fuxiang X, Yang B, Xiang Z, Peng G, Zhengtao W, Shengqiao X (2021) Circular fruit and vegetable classification based on optimized googlenet. IEEE Access 9:113599–113611
36. Deepak S, Ameer P (2020) Retrieval of brain mri with tumor using contrastive loss based similarity on googlenet encodings. Comput Biol Med 125:103993
37. Yin X, Liu Q, Huang X, Pan Y (2021) Real-time prediction of rockburst intensity using an integrated cnn-adam-bo algorithm based on microseismic data and its engineering application. Tunn Undergr Space Technol 117:104133
38. Whitney HM, Li H, Ji Y, Liu P, Giger ML (2019) Comparison of breast mri tumor classification using human-engineered radiomics, transfer learning from deep convolutional neural networks, and fusion methods. Proc IEEE 108(1):163–177
39. Basha SS, Dubey SR, Pulabaigari V, Mukherjee S (2020) Impact of fully connected layers on performance of convolutional neural networks for image classification. Neurocomputing 378:112–119
40. Li J, Li B, Xu J, Xiong R, Gao W (2018) Fully connected network-based intra prediction for image coding. IEEE Trans Image Process 27(7):3236–3247
41. Moradi R, Berangi R, Minaei B (2019) Sparsemaps: convolutional networks with sparse feature maps for tiny image classification. Expert Syst Appl 119:142–154
42. Wang J, Gao F, Dong J, Du Q (2020) Adaptive dropblock-enhanced generative adversarial networks for hyperspectral image classification. IEEE Trans Geosci Remote Sens 59(6):5040–5053
43. Li X, Kong X, Liu Z, Hu Z, Shi C (2021) A novel framework for early pitting fault diagnosis of rotating machinery based on dilated cnn combined with spatial dropout. IEEE Access 9:29243–29252
44. Mobiny A, Yuan P, Moulik SK, Garg N, Wu CC, Van Nguyen H (2021) Dropconnect is effective in modeling uncertainty of bayesian deep networks. Scientif Rep 11(1):1–14
45. Maharjan S, Alsadoon A, Prasad P, Al-Dalain T, Alsadoon OH (2020) A novel enhanced softmax loss function for brain tumour detection using deep learning. J Neurosci Methods 330:108520
46. Alhassan AM, Zainon WMNW (2021) Brain tumor classification in magnetic resonance image using hard swish-based relu activation function-convolutional neural network. Neural Comput Applic 33(15):9075–9087
47. Dif N, Attaoui MO, Elberrichi Z, Lebbah M, Azzag H (2022) Transfer learning from synthetic labels for histopathological images classification. Appl Intell 52(1):358–377
48. Karthik R, Menaka R, Hariharan M (2021) Learning distinctive filters for covid-19 detection from chest x-ray using shuffled residual cnn. Appl Soft Comput 99:106744
49. Dasari CM, Bhukya R (2022) Explainable deep neural networks for novel viral genome prediction. Appl Intell 52(3):3002–3017
50. Xie Y, He M, Ma T, Tian W (2022) Optimal distributed parallel algorithms for deep learning framework tensorflow. Appl Intell 52(4):3880–3900

Juncheng Liu is a MSc student in the College of Big Data and Intelligent Engineering, Southwest Forestry University. His research interests are in neural networks and deep learning, focusing on applying semi-supervised learning techniques to computer vision problems, especially medical image segmentation.

Yili Zhao is an Associate Professor in the College of Big Data and Intelligent Engineering, Southwest Forestry University. He obtained his Ph.D. from Yunnan University in 2018. His research interests are in neural networks and deep learning, focusing on applying deep learning techniques to computer vision problems, especially image segmentation.