IET Computer Vision

Research Article

IET Journals
The Institution of Engineering and Technology

Multi-sample inference network

ISSN 1751-9632
Received on 29th March 2018
Revised 29th October 2018
Accepted on 4th April 2019
E-First on 30th August 2019
doi: 10.1049/iet-cvi.2018.5126
www.ietdl.org

Daojun Liang[1], Feng Yang[1,2] ✉, Xiuping Wang[1], Xiaohui Ju[1]

[1]School of Information Science and Engineering, Shandong Normal University, Jinan 250014, Shandong Province, People's Republic of China
[2]Institute of Data Science and Technology, Shandong Normal University, Jinan 250014, Shandong Province, People's Republic of China
✉ *E-mail: yangfeng@sdnu.edu.cn*

Abstract: This study explores whether neural networks can classify multiple samples simultaneously in a forward process. Therefore, a multi-input multi-prediction network architecture has been proposed. The authors call this method a multi-sample inference network (MSIN). In addition to maximising the use of network shared parameters, the network can also use multiple samples for training. MSIN allows multiple samples to be randomly combined to act as data augmentation, and the random combination of corresponding labels can regularise the network as a loss regularisation, which makes MSIN have better generalisation performance. In contrast, category expansion is a problem that is difficult to solve because neural networks can only predict a fixed number of categories. The network proposed in this study can solve the category expansion problem by expanding the initial layers and the final layers. It is trained by using samples of multiple domains at the same time to ensure that the network has no significant decline in the predictive performance of the existing categories. The MSIN method can also be applied to the generative adversarial network to enable it to simultaneously generate samples of multiple sample domains.

1 Introduction

Can a neural network separate two categories at the same time in a forward process? If so, what characteristics should the network have? The traditional neural network can only predict one sample in one forward process. In this paper, we design a method that can classify two or more samples, which can greatly reuse the features between the network layers and reduce the running time. This paper explores the simultaneous classification of two samples of the same task or different tasks into corresponding classes while ensuring that the accuracy is the same as the single-sample classification.

The network architecture is shown in Fig. 1. The architecture is mainly divided into three modules. The first module is the initial block, which mainly adapts multiple samples to the same intermediate features. For example, images of different sizes use different convolution kernels to produce the same size of intermediate features. The middle feature module is a shared block so that the two samples share the computation of the forward and backward processes. The final block produces different high-level features, producing the corresponding categories for multiple samples.

There are many works that also use multiple samples to train the network, such as DisturbLabel [1], SamplePairing [2], and Mixup [3]. All these works have combined training samples and their labels to some extent. This makes training data not only for data augmentation but also for regularisation effect. As multi-sample inference network (MSIN) uses multi-sample training like these methods, MSIN can be combined with these methods to make neural networks produce better generalisation performance. Some works using deep learning to study sample similarity involves inputting two samples [4] or three samples [5] into the network separately or simultaneously inputting two samples [6] into the network to calculate similarities between samples. Different from these methods of training with multiple samples, the MSIN also uses multiple samples for inference, so that a single neural network can simultaneously predict multiple samples in one forward process. The neural network can infer about two mixed samples at the same time, which reveals a new feature of the neural network: the neural network can separate the mixed samples in the inference process.

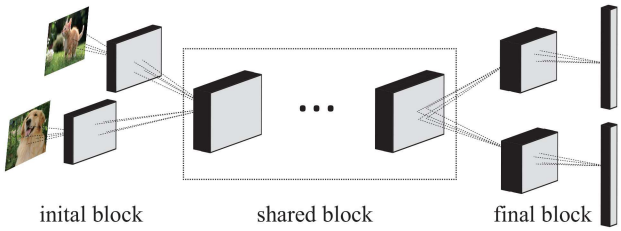

Fig. 1 *Multi-category predicted network architecture that shares the forward process. The neural network is divided into three modules, namely the initial block, the shared block, and the final block*

There are also many efforts to make neural networks have better generalisation capabilities by designing more efficient loss functions. FaceNet [7] uses a triple loss to make the intra-class distance larger than the inter-class distance and minimises the Euclidean distance of the positive pair to make it lower than the set margin. A centre loss [8] was proposed to improve the original softmax loss, which introduces cosine similarity to distinguish between different categories, making it more consistent with the sample distribution. Compared with the introduction of cosine similarity, the angle-based L-Softmax [9] loss can more accurately maximise the inter-class differences, so that a more robust model can be obtained. As the final layers of the MSIN are separated from each other, it can use a variety of loss functions. In the implementation of MSIN, softmax loss, L-Softmax [9] loss, and additional domain-based similarity loss are used as the loss functions.

The problem of category expansion has always been a problem in recognition problems. We can also use the properties of the MSIN to solve the category expansion problem of the network. As we all know, neural networks can only predict a fixed category of networks, which cannot be used to correctly predict categories outside its prediction range. Retraining the network will consume huge resources and time, and is not desirable for many situations. This paper proposes a channel combination method to extend the categories of neural networks. This not only enables neural networks to predict new categories but also does not have significant performance degradation in the prediction of existing categories.

Can the generative adversarial network (GAN) [10] generate samples from two different sample domains simultaneously? To achieve this, the MSIN method is extended to the generator and discriminator of GAN [10], enabling its generator and discriminator to simultaneously generate and discriminate two or more samples. We call the network that can generate multiple samples at the same time as multi-sample GAN (MSGAN), whose generator and discriminator are shared by multiple samples to the greatest extent, and it can generate more diverse patterns between different sample domains.

The main contributions of this paper include the following:

- A MSIN architecture has been proposed, which demonstrates that neural networks can predict multiple samples simultaneously in a forward process without confusion.
- MSIN can greatly reuse the features between the network layers and reduce the running time while ensuring that the accuracy is slightly lower than the single-sample classification.
- The properties of the MSIN can be used to solve the category expansion problem of the neural networks.
- The MSIN method can be extended to GAN [10] to enable its generator to produce a more diverse pattern.

The paper is organised as follows: first, related work will be introduced in Section 2. Then, the architecture of the MSIN will be detailed in Section 3, and the results of the experiment will be presented in Section 4. Next, the nature of the MSIN is discussed in Section 5. Finally, the paper is summarised in Section 6.

2 Related work

The neural network has a lot of hyperparameters and millions of training parameters so that it has a strong fitting ability and can even fit random noise [11]. Therefore, it is a challenging job to reduce the overfitting of neural networks. There are many factors that influence the generalisation ability of neural network from different aspects such as designing more effective network structure [12–18], exploring more effective activation functions [19–23], optimising network parameters [24–26], processing training data [15, 27], and reducing model complexity [28–32].

Recently, many works use multiple samples to train the network. In DisturbLabel [1], a small number of sample labels were randomly replaced with other labels. This is a regularisation method that is used at the loss layer to allow the neural network to avoid overfitting and enhance its generalisation performance. In SamplePairing [2] and Mixup [3], a method of training the neural network using two samples simultaneously is proposed. SamplePairing [2] randomly picks one sample in the training set to add to the original sample and uses the original sample's label to train the network. Mixup [3] uses a random value to weigh the two samples and their corresponding labels. All of the above methods have some effect of data augmentation and regularisation, and they can achieve better generalisation performance than empirical risk minimisation [33]. SamplePairing [2] and Mixup [3] are very similar to traditional data augmentation methods, but they are also different. SamplePairing [2] adds other samples from the training set as noise to regularise the neural network during training. Mixup [3] does not explicitly use the original sample training network, but the convex interpolation of the original sample. There are also some works that use neural networks to compare similarities between samples. The Siamese network [4] is mainly used for face verification, which learns the similarity measure between two samples by optimising a discriminative loss function. It outputs a similarity value to determine if two faces are from the same category. The authors in [5] proposed a triplet loss to train a network that formulates a continuous upper bound on empirical loss, with a new form of loss-augmented inference designed for efficient optimisation with the proposed loss on the Hamming space. The authors in [6] improve the original Siamese network in which two samples are simultaneous inputs to the network instead of being entered separately. Their main purpose is to learn a general similarity measure function to compare image patches. These networks are designed to measure the similarity of samples and cannot be used to predict the category of sample. Conversely, our network not only measures the sample similarity but also accurately predicts the category of sample and correctly distinguishes the domain of the sample.

An efficient loss function not only allows the model to have better generalisation performance but also makes it more robust. FaceNet [7] maximises the inter-class distance and minimises the intra-class distance of the samples by designing a triple loss. It keeps samples between different categories have a large margin, and samples between the same categories are as close as possible. The centre loss [8] involves simultaneous learning of a centre for deep features of each class and penalising the distances between the deep features and their corresponding class centres. L-Softmax [9] loss uses an angle-based similarity metric to explicitly encourage intra-class compactness and inter-class separability between learned features. Since MSIN can distinguish between samples in different domains, a domain-based similarity loss function is incorporated into the MSIN network, which can better describe the differences between samples.

Extending the data set and regularising the model can reduce the overfitting of the model from different aspects. Some data augmentation techniques directly distort data space, such as translation, rotation, flipping, cropping, adding noises etc. Other methods use slightly more sophisticated methods such as PatchShuffle [34] which divides the sample data into several different patches, sorting the pixels in each patch according to a certain rule, while keeping the overall structure of the image intact. The authors in [35] randomly select a rectangular area in the image, and fill the area with Gaussian noise, making the original information of the area as blocked, forcing the neural network to infer about the part of the information. SamplePairing [2] randomly picks a sample in the training set and adds it to the original sample to produce a new sample. These methods make the processed image slightly different from the original image, although the augmented data may not be in line with the original sample distribution or independent of the original sample. However, this can make the neural network learn other models related to the original sample, which can make the neural network have a more diversified representation ability and thus have better generalisation performance.

GAN is proposed in [10], which generates a sample through the generator, and then uses the discriminator to determine whether a sample is real or generated. The discriminator can only be fooled when the sample generated by the generator is sufficiently real, and the discriminator will try to find those samples that are generated by the generator. The generator and discriminator are doing a zero-sum game, and the final result is a Nash equilibrium between them. We apply the MSIN method to the GAN to enable it to simultaneously generate samples of multiple sample domains.

3 Method

3.1 MSIN architecture

How to classify two or more samples simultaneously in a forward process instead of using multiple forward processes to sequentially input samples into the neural network? A MSIN architecture is designed to predict multiple samples simultaneously. The architecture is able to accurately classify multiple samples while predicting the domains corresponding to them, which enables the same network to be used on different sample domains. Fig. 1 schematically depicts the MSIN architecture that can predict two samples simultaneously. The architecture is mainly divided into three parts: initial block, shared block, and final block.

3.1.1 Initial block: The initial block mainly produces the same size features map so that the features can be shared in the shared block. This block mainly includes one or two convolutional layers. When the feature size difference is large, we can add some convolutional layers with large stride and some pooling layers for downsampling it. According to the different situation of inputs, the initial layers can be divided into two types. First, each sample will be an independent input to the neural network, K samples will have K independent initial layers. Another situation is that the initial layers

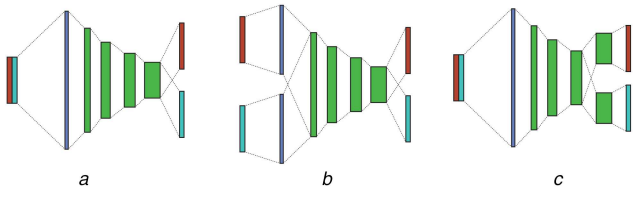

Fig. 2 *Architecture of the MSIN variants*
(a) MSIN, ***(b)*** MSIN-I, ***(c)*** MSIN-F

of the network have common low-level features, so the initial layers can combine these samples with a concatenate or add operations. These operations remove the initial block from the network, leaving only the shared block and the final block.

3.1.2 Shared block: The shared block is the main part of the network. It carries the main calculations and contains most of the convolutional layers of the network. These features will be shared by multiple simultaneously trained samples. The module can be implemented using a variety of network architectures, such as ResNet-like [36] or DenseNet-like [17], making the module more scalable and flexible. This module includes almost all of the MSIN convolution and pooling layers, so it takes up almost all of the computational overhead of the MSIN. Reducing the number of layers in the module can speed up the inference of the network, but it will reduce the network performance, so the module makes a compromise between the speed and performance of the network.

3.1.3 Final block: The final block has the highest level of abstraction and will produce the corresponding category for the sample. In different tasks, the final block can be used to train the task-related abstract features, which has higher performance for specific tasks. The final block may or may not contain a convolutional layer, which may result in a slight difference in performance. If the network does not contain the initial block and the final block do not contain the convolutional layers, the network is used as a benchmark for performance comparison (MSIN-B), which is the network with the most common features.

3.1.4 Variants of MSIN: The shared block is the key to the accuracy of the classification of the multiple categories. The number of the initial block and final block has a direct impact on the prediction accuracy, but it will make the network have more parameters and consume more computation. Different designs of the initial and final block of the basic network will result in variants of other MSINs. The final block of MSIN-B is set to two separate fully connected (FC) layers, using two concatenation samples as input layers. That is, except for the final FC layers, the remaining layers are shared layers. The network that uses a convolution layer as an independent structure in the initial block is denoted as MSIN-I1 (the number represents the number of layers), and the network that uses one block structure (two convolution layers) as an independent structure in the final block is denoted as MSIN-F2 (the last full connection layer is not included). Using different methods for multiple samples as input to the network will also result in very different classification performance. For example, taking two added samples or using two concatenated samples as inputs will yield different multi-class performance. We record the network structure of the two added samples as the input layer of the standard network as MSIN-P. Various MSIN variants are shown in Fig. 2, and their performance comparison is shown in Section 4.3.

3.2 Multiple categories prediction

In this section, we will explore to what extent the neural network confuses multiple samples. To further explore how neural networks can classify multiple categories simultaneously in a forward process while reducing computational time and computational resources. The neural network is designed with multiple inputs and multiple outputs. The middle block of the neural network is designed as a shared structure. The network architecture used in this section is given in Fig. 1. The ith classifier is denoted as C_i, and the total number of classifiers is denoted as K. We further denote the sample to be classified by the classifier C_i as x_i, its label as y_i, and its probability distribution as p_i. The difference with the traditional network is that the final layers of this network will output the prediction of multiple domains. If the samples are from different domains, we will record their corresponding domain as D_i.

The architecture of the MSIN has been determined, and the most important thing is to design the loss function of the MSIN so that the loss function can ensure that the MSIN does not overfit a certain sample domain during training. Traditional single-sample inference networks generally use a softmax loss function for training. The formula for the softmax loss function is

$$\begin{aligned} L_{\text{soft}} &= \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{\mathrm{e}^{x[y]}}{\sum_j^Z \mathrm{e}^{x[j]}}\right) \\ &= \mathbb{E}_{y\sim p(y)}[\log \boldsymbol{C}(y)] \end{aligned} \tag{1}$$

where $x[y]$ denotes the yth element ($y \in [1, Z]$, Z is the number of classes) of the vector of class scores $\boldsymbol{C}$, and N is the number of training data. As MSIN has multiple output layers to predict multiple input samples, softmax loss is used directly for each output layer to reduce the cross-entropy of each sample domain. By adding the softmax loss of all sample domains directly, the loss function of MSIN which used to train multiple samples is reformulated as

$$\begin{aligned} L_{\text{entro}} &= \sum_{i=1}^{K} \lambda_i \mathbb{E}_{y_i\sim p_i(y_i)}[\log C_i(y_i)] \\ s.t. &\sum_{i=1}^{K} \lambda_i = 1 \end{aligned} \tag{2}$$

where λ controls the relative importance of the multiple objectives, and K is the number of the sample domains. It can take a fixed constant or take a random value in each min-batch, so that the final layers can be alternately trained. The effect of the value of λ on MSIN performance will be discussed in Section 5.2.

In order to explicitly encourage intra-class compactness and inter-class separability between learned features, MSIN requires a large margin loss function. The large margin loss is added to each classifier of the MSIN, which is formalised as

$$L_{\text{margin}} = \frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_{y_i\sim p_i(y_i)}[\max(0, M - \| y_i o_i \|_1)] \tag{3}$$

where o_i is the output of the classifier C_i, and M is the large margin value. If the training samples are from the same domain or similar domains, we hope that the MSIN classifiers will have smaller differences for similar samples and larger differences for different samples. Therefore, we need to regularise the output of each MSIN classifier. The formalisation of this regularisation loss function is

$$L_{\text{reg}} = \frac{2}{K(K-1)}\sum_{i\neq j, i<j}^{K} \mathbb{E}_{o_i\sim p_i(C_i(o_i))}\left[\| o_i - o_j \|_2\right] \tag{4}$$

The above is all the loss functions needed to train the MSIN. Add them together to get the total loss function

$$L_{\text{total}} = \beta_1 L_{\text{entro}} + \beta_2 L_{\text{margin}} + \beta_3 L_{\text{reg}} \tag{5}$$

where β_1, β_2, and β_3 are used to balance the corresponding loss function. The total loss is used for backward propagation. On the one hand, in the process of back propagation, the weight of each filter will be adjusted by multiple feedback signals, which regularises the neural network and can effectively alleviate the overfitting problem of neural networks. On the other hand, each sample will be somewhat confused with each other, which is

detrimental to their prediction accuracy. In Section 3.3 we will introduce a domain-based loss function to further expand and optimise L_{total}.

3.3 Sample domain prediction

If the K input samples of the MSIN are from the same domain, the number of mixed samples will be N^K. So many mixed samples will play a role in data augmentation, and the combined loss function of the corresponding labels will also have a regularisation effect. When using multiple samples to train MSIN, there will be some confusion between the samples, and we use this confusion to verify what exactly the MSIN does. If the MSIN can predict multiple categories of input samples at the same time, it indicates that the neural network using MSIN training can learn multiple samples at the same time and can avoid the confusion among multiple samples. This shows that MSIN is actually a regularisation method. If the MSIN can not predict more than one sample at the same time, it can only have good predictive ability for one type of sample, indicating that the synthetic sample is used as a new sample to be fitted by the neural network. In the inference phase, the test sample tends to be classified into the category of training samples that has the greatest similarity to it. This shows that MSIN plays a role of data augmentation. The experiments in Section 4.8 demonstrate that multiple samples used for MSIN training not only serve for regularisation but also serve for data augmentation.

If the input samples are from different domains, the initial block cannot determine the domain to which the sample corresponds, i.e. the MSIN cannot determine whether the sample x_1 is from the domain D_1 or from the domain D_2, and the same is true for the sample x_2. This causes the MSIN to confuse multiple samples of the input. For the MSIN to be able to predict different sample domains, an additional classifier D needs to be added to the MSIN to enable the MSIN to predict the domain D_i of the sample x_i. For K input samples, the classifier needs to predict K^K sample domains. So the cross-entropy loss and the additional large margin loss are used to train the domain-based classifier, and the loss function can be formalised as

$$\begin{aligned} L_{\text{domain}} &= \mathbb{E}_{y_i \sim p(y_i)}[\log D(y_i)] \\ &\quad + \gamma \mathbb{E}_{y_i \sim p(y_i)}[\max(0, M_{\text{d}} - \delta \| \, y_i o_{\text{d}} \, \|_1)] \\ \text{where} \quad \delta &= \begin{cases} 1, & o_{\text{d}} \in D_i \\ -1, & o_{\text{d}} \notin D_i \end{cases}. \end{aligned} \tag{6}$$

The o_{d} and M_{d} in (6) are the outputs and large margin value of the domain-based classifier D, and γ is used to balance the contribution of two loss functions.

Using N^K training data to generate K^K categories, the domain-based classifier will produce a large overfitting of these samples. L-Softmax [9] loss is used as an optional loss function to train each sample-based classifier C_i so that each class can be well separated in sample space. The formalisation of L-Softmax [9] loss function is

$$\begin{aligned} L_{\text{lsoft}} &= \sum_{i=1}^{K} \lambda_i \mathbb{E}_{y_i \sim p_i(y_i)}[\log L_i(y_i)] \\ \text{s.t.} \sum_{i=1}^{K} \lambda_i &= 1 \end{aligned} \tag{7}$$

where L_i is called the large margin classifier that uses the L-Softmax [9] loss. Equation (7) simply replaces the softmax function of (2) with the L-Softmax [9] function. Therefore, the total loss function of MSIN is

$$L_{\text{total}} = \beta_1 L_{\text{lsoft}} + \beta_2 L_{\text{margin}} + \beta_3 L_{\text{reg}} + \beta_4 L_{\text{domain}} \tag{8}$$

Comparing (8) with (5), it can be found that the original L_{soft} was replaced by L_{lsoft} and the L_{domain} loss was added. Using (8) as the loss function of MSIN, not only obtains better classification performance but also accurately predicts the domain corresponding to each sample.

3.4 Sample domain expansion

In this section, we consider using MSIN for sample domain expansion. We assume that x_1 is sampled from the domain D_1, and x_2 is sampled from the domain D_2. If traditional methods are used to predict two different domains, these two different domains will generally be combined into the same domain, so that the total number of categories of the combined domains is the sum of the categories of the two domains. Although using this method can also train two different domains of the sample, there are two disadvantages compared with MSIN: the method cannot distinguish the domain from which the training sample comes from, and it is difficult to expand the new category. The expansion of new categories will result in a sharp decline in the prediction accuracy of neural networks for existing categories. Using MSIN architecture can not only solve this problem well but also can also train and predict samples in two different domains at the same time. The MSIN allocates different initial layers and final layers for samples in different domains. When adding samples for new domains, it is only necessary to simply extend the initial layers and the final layers. Most importantly, neural networks need to be finetuned when expanding new categories. When the traditional method expands new categories, it will cause the neural network's prediction accuracy of existing categories to decrease significantly. However, the MSIN architecture can guarantee that when the new category is expanded, the prediction accuracy of the existing category is only slightly declined.

MSIN has two training methods. One is to train multiple prediction tasks synchronously (MSIN-S), and the other is to perform asynchronous training (MSIN-A) on each task. The reason why the two kinds of training methods are proposed is that the distribution of data is very different. For example, a small number of synthetic data sets cannot use MSIN to predict multiple categories simultaneously, such as SVHN [37] data sets. The vast majority of unsynthesised data can be trained using the MSIN-S. Using MSIN-S, both x_1 and x_2 samples can be trained simultaneously, allowing the neural network to simultaneously fit the joint distribution of the two domains. For tasks that cannot be trained using MSIN-S, MSIN-A is required to train the task. When training with MSIN-A, the data that is difficult to fit is trained first, and then the data that is relatively easy to fit is trained. For example, when we need to train with SVHN [37] and other data sets at the same time, we need to train the SVHN [37] data set first and then finetune MSIN on other data sets. Another situation is that when expanding the category of the network, it is necessary to maintain the generalisation ability of the existing category and also to train the data in another domain.

3.5 Multi-sample generative adversarial network

In this section, we will discuss the application of the MSIN network architecture to GAN, which we call the MSGAN. We want the generator to be able to simultaneously generate samples from two different fields in a forward process, and the corresponding discriminator can simultaneously determine whether the two samples are true or false. Extending the traditional GAN: the final layer of the generator and the initial layer of the discriminator are expanded to two sample domains. The loss function of MSGAN can be reformulated as

$$\begin{aligned} \min_G \max_D V(D,G) &= \sum_{i=1}^{K} \alpha_i \mathbb{E}_{x_i \sim p_{\text{data}_i}(x_i)}[\log(D(x))] \\ &\quad + \sum_{i=1}^{K} \kappa_i \mathbb{E}_{z_i \sim p_{z_i}(z_i)}[\log(D(1 - \eta_i G(z_i)))] \end{aligned} \tag{9}$$

where α is used to balance the discriminator's loss of real data in different sample domains, κ is used to balance the discriminator's loss of false data, and η is used to weigh the generator's loss

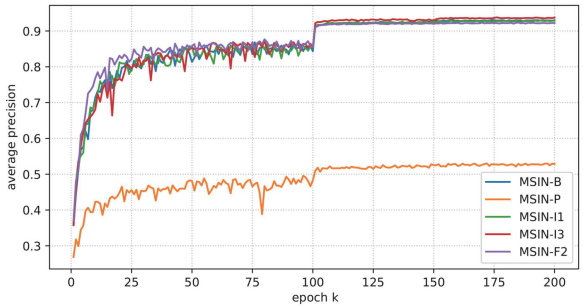

Fig. 3 *MSIN and its variants average test accuracy over multiple samples*

between each sample domain. It can be found in our experiments Section 4.9 that MSGAN can generate images in different sample domains.

4 Experiments

4.1 Data sets

We performed experiments on the MNIST [38], Fashion-MNIST [39], CIFAR10 [40], CIFAR100 [40], and SVHN [37] data sets, respectively. These data sets have approximately the same image size and the number of samples. We also apply standard augmentation [13, 15, 41–46]: horizontal flipping and translation by 4 pixels are adopted in our experiments.

4.1.1 MNIST: It contains 60,000 hand-written digits for training and 10,000 for testing. The images are collected from 250 writers and have shapes of $1 \times 28 \times 28$ where 1 denotes one channel.

4.1.2 Fashion-MNIST: It is a data set of images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 greyscale image, associated with a label from 10 classes. Fashion-MNIST [39] serves as a direct drop-in replacement for the original MNIST [38] data set for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

4.1.3 CIFAR: It consists of 60,000 32×32 coloured natural scene images, 10,000 of which are used for testing. Specifically, CIFAR-10 [40] contains 10 classes, with 5000 training images and 1000 testing images per class, while CIFAR-100 [40] has 100 classes, with 500 training images and 100 testing images for each class. Channel means are computed and subtracted in pre-processing.

4.1.4 SVHN: It is a real-world data set obtained from house numbers in Google Street View images. It consists of 10 classes, with 73,257 digits for training, 26,032 digits for testing, and 531,131 additional digits. All digits have been resized to 32-by-32 pixels. The task is to classify the central digit into a correct class. In our experiment, we did not apply additional data sets for training.

4.2 Training

For comparison purposes, we use the same network architecture and set the same hyperparameters for all data sets. We use PreAct ResNet-18 [36] and DenseNet-BC-70 [17] as the basic network architecture. Unless otherwise stated, PreAct ResNet-18 [36] will be set as the default basic network architecture. The network architecture is divided into four blocks, each of which is a residual block containing two convolutional layers. After each block, the feature size is halved and the number of channels is doubled. DenseNet-BC-70 [17] is a network structure that connects all the front layer information to the subsequent layer. It consists of three block structures, each of which is followed by a transfer structure with a pooled layer.

All the networks are trained on two NVIDIA Tesla k80 GPUs using stochastic gradient descent (SGD). We use a weight decay of 1×10^{-4} and a Nesterov momentum [47] of 0.9 without dampening. The batch size on each GPU is set to 128 for 200 epochs. The initial learning rate is set to 0.1 and is divided by 10 at 50 and 75% of the total number of training epochs. We use (5) as the default loss function when we do not perform sample domain prediction, otherwise we use (8) as the loss function. The β_i in (5) and (8) are set to the same value: $\beta_i = 1/K$, and the large margin values in (3) and (6) are set to 1.

4.3 Performance comparison of MSIN variations

We compared the performance of MSIN variants using different initial layers and different final layers on CIFAR-10 [40]. Fig. 3 shows the performance comparison of MSIN-P, MSIN-I1, MSIN-I3, MSIN-F2, and MSIN-B. It can be found that MSIN-I3 has higher performance than MSIN-I1, indicating that the initial layers separated from each other can retain more high-level features that will not be lost due to the competition of multiple classes. Compared with MSIN-I1, the performance of MSIN-B is slightly worse but almost the same, indicating that a few independent initial layers still maintain almost the same features, and as the number of layers increases, these features become more and more different. The MSIN-F2 with more parameters has a better performance in the initial training phase, but the final generalisation performance is slightly lower than the MSIN-B. This phenomenon indicates that the convolution layer is shared at the final layers, which is conducive to the regularisation of the synthesised label at the loss layer. It can be seen that the performance of MSIN-P is not normal, and the average performance is only about 54%. This is due to the addition of two samples at the input layer of the MSIN, which causes the two classifiers to confuse their respective samples. MSIN-B just changed the addition operation of the input layer to concatenation operation, which effectively avoids the occurrence of this abnormal phenomenon.

4.4 MSIN performance on same domain

To demonstrate that MSIN can separate multiple categories simultaneously in a forward process, we have designed experiments that can predict multiple categories simultaneously. We use MSIN-B to implement the experiments in this section. The training process of the network on the same data set is shown in Fig. 4. The figure shows the training accuracy of C_1 and C_2 on the same data set. The training and test accuracy of C_1 and C_2 are basically the same.

In CIFAR-10 [40] and MNIST [38], MSIN has a good performance both in training accuracy and testing accuracy. In general, two samples may have different levels of confusion, so the test performance of each category should be lower than the original single-category network. In MNIST [38], the original test error on PreAct ResNet-18 [36] was 0.8, and on MSIN-B it was 0.4. Classifying multiple samples at the same time not only does not degrade the test performance but also increases its performance. Note that MSIN-B only changes the input layer and output layer on PreAct ResNet-18 [36]. Their network architecture and parameters are basically the same. Therefore, this performance improvement is not a difference in network parameters or architecture. The reason is that MNIST [38] is a relatively simple task, there are more common patterns between the samples, thereby reducing the performance of PreAct ResNet-18 [36] underfitting the data set.

It can be found that the training accuracy of the two samples are about 97% in CIFAR-100 [40] data set, indicating that there is still some confusion in the neural network. The first category of the average test accuracy is 71.9%, the second category of the average test accuracy is 71.2%. The classification accuracy is almost as good as the original single-category precision.

However, since SVHN [37] is a special task, each number is tiled into a picture of a 3×3 grid and the discrimination of some pictures is very low. Each picture in the SVHN [37] has 9 digits, which causes difficulties for MSIN training. It can be seen from Fig. 4*d* that only the training and testing process of C_2 is normal, while the training and testing accuracy of C_1 is maintained at about 20%. In the MSIN training process, there is competition between

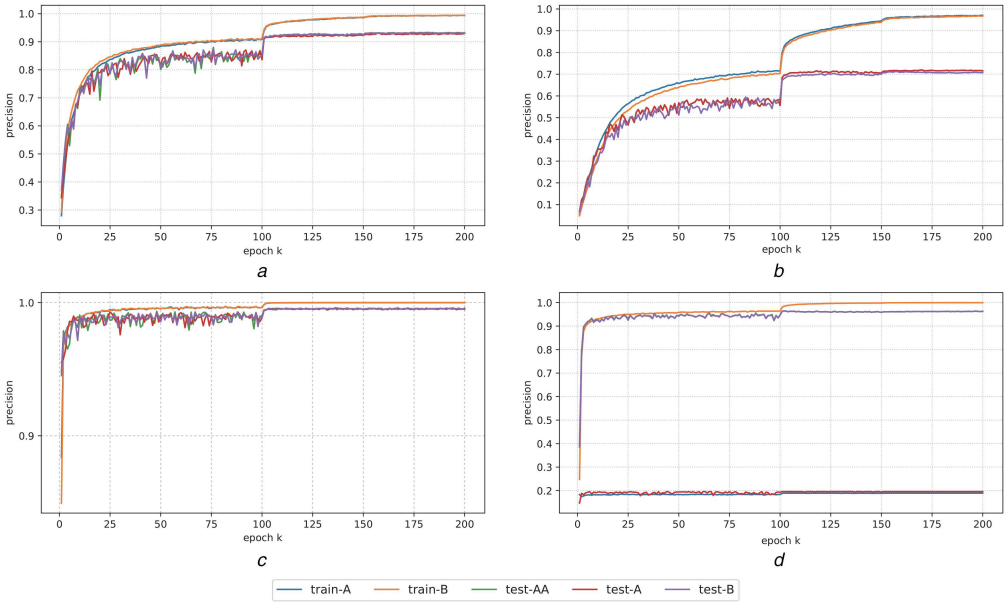

Fig. 4 *MSIN performance on data sets MNIST [38], CIFAR10 [40], CIFAR100 [40], and SVHN [37]. The suffix A in the legend represents the sample classified by the classifier C_1, and the suffix B represents the sample classified by the classifier C_2*

(a) CIFAR-10 (test A: 93.1%, test B: 92.9%), ***(b)*** CIFAR100 (test A: 71.9%, test B: 71.2%), ***(c)*** MNIST-10 (test A: 99.6%, test B: 99.6%), ***(d)*** SVHN (test A: 19.6%, test B: 96.4%)

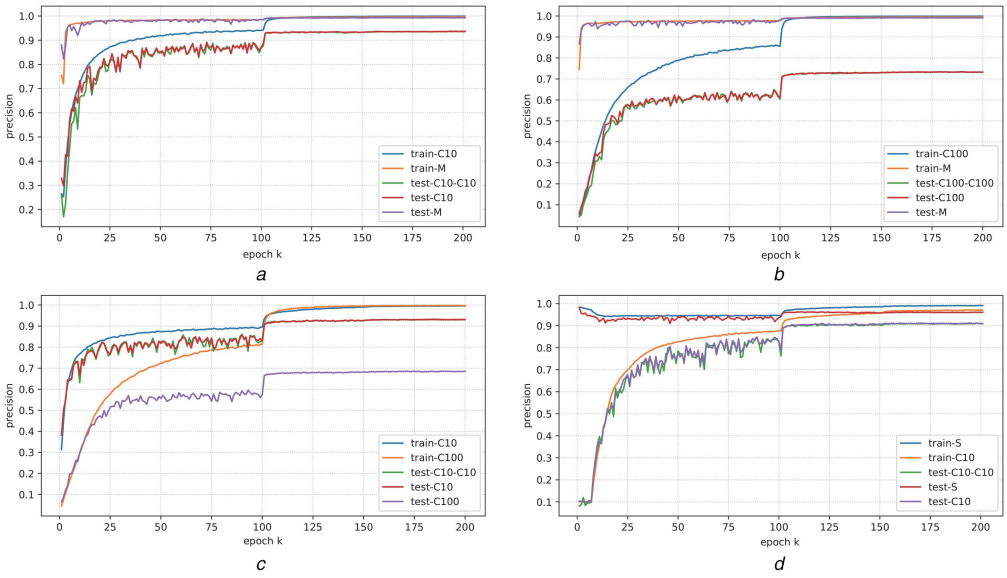

Fig. 5 *MSIN performance on different data sets. The two input samples of the MSIN come from different data sets. The suffix in the legend is a shorthand for a data set that represents the performance of MSIN on that data set*

(a) CIFAR-10 and MNIST, ***(b)*** CIFAR100 and MNIST, ***(c)*** CIFAR-10 and CIFAR100, ***(d)*** CIFAR-10 and SVHN

multiple numbers, often resulting in one sample occupying an absolute advantage, so that two categories cannot be distinguished at the same time.

4.5 MSIN performance on two domains

We use MSIN-B to train on different data sets to get its generalisation performance for simultaneous multi-task prediction. Its training process in different data sets is shown in Fig. 5. The names of various data sets are abbreviated. For example, CIFAR-10 [40] is abbreviated as C10. Note that each training task is trained simultaneously, except for the CIFAR-10 [40] and SVHN [37] data sets in Fig. 5*c*. Fig. 5 shows the training and the test accuracy of the MSIN-B network on various data sets. Each task is almost unaffected by other tasks during training, which is almost indistinguishable from training the task alone.

As described in Section 4.4, MSIN cannot train multiple samples simultaneously on the SVHN [37] data set, so we use MSIN-A for training. As can be seen in Fig. 5*d*, the MSIN is first trained on the SVHN [37] data set and then the CIFAR-10 [40] data set are added to train simultaneously. In this way, the network can maintain high test accuracy for SVHN [37] and avoid the MSIN to have good generalisation performance only for the samples on CIFAR-10 [40].

The original category represents the test error of the data set on PreAct ResNet-18 [36], and the multiple categories represent the test error of MSIN on different data sets at the same time. The single class indicates that the MSIN, after training on multiple data sets, is used to test for errors in one of the data sets.

Table 1 shows the average test error of the MSIN samples that simultaneously classify multiple different data sets (multiple category), the single-category test error with the same category as the input to the MSIN (single category), and the test error of each data set on PreAct ResNet-18 [36] (original category). It can be found that the multi-category test error is slightly higher than the original network test error but slightly lower than the single-category test error. These small differences are acceptable in the actual situation. In addition, we can use variants of MSIN to improve its performance if the computing resources allowed. By comparing the test errors of multiple categories and single

categories, it can be shown that the neural network will produce overfitting for different training processes, and it also shows that there is strong independence between the two classifiers.

4.6 MSIN performance on multiple domains

To test the performance of the MSIN on multiple sample domains, we used four data sets (MNIST [38], Fashion-MNIST [39], CIFAR10 [40], and CIFAF100 [40]) to train the MSIN. In this section, DenseNet-BC-70 [17] is used as the basic network architecture, (5) is used as the loss function. β_1 and β_2 in (5) is set to 1, and β_3 is set to 0.1.

Fig. 6 shows the performance of the MSIN on a multi-sample domain. It can be found that the MSIN can separate all the samples on the four different domains. The performance of the MSIN is slightly lower when predicting the four sample domains than when predicting the three sample domains. Compared with the single-sample inference network, the performance of the MSIN is slightly declined, but the availability of the MSIN is basically guaranteed.

4.7 Train multi-channel mixed MSINs

The MSIN-B is used to train multi-channel samples from the CIFAR-10 [40] data set. We split each sample from CIFAR-10 [40] into three channels and randomly combine these three channels into a new sample in each min-batch. We use (5) as the loss function for this training. The MSIN will be divided into three classifiers, each of which is responsible for one channel. The entire training and testing process is shown in Fig. 7.

In Fig. 7, it can be seen that multiple different channels have a normal training process, indicating that the MSIN can be successfully extended to multiple categories. As there are different degrees of competition between the various channels, the training error does not decrease to zero. Another reason is that the random combination of individual samples makes the distribution range of the data greatly increased. Moreover, there is a large correlation between the channels of the same data set, the weights are adjusted by multiple channels, resulting in the network not being overfitted to any one of the channels. Another conclusion of this experiment is that the different channels of the sample correspond to different training and test performances. The second channel of the sample in CIFAR-10 [40] contributes more to its classification accuracy, while the first channel contributes less.

4.8 Sample domain prediction using MSIN

When predicting the sample domain, it is necessary to give each domain a label corresponding to each sample, and K sample domains correspond to K^2 labels. We want each batch to contain an equal number of samples from different domains, so after assigning labels to the sample domains, we need to randomise the order of the samples. This section has the same hyperparameter settings for MSIN as in Section 4.6. The difference is that the batch size needs to be set to 32, and the training epochs are changed to 100 because the order of the samples is randomised so that the batch size becomes the original K^2 times.

Fig. 8 shows the sample domain prediction accuracy of the MSIN on four different data sets. It can be found that on the MNIST [38] and Fashion-MNIST [39] data sets, the MSIN can predict the corresponding sample domain well, and the prediction accuracy is 100%, while the prediction accuracy on CIFAR-10 [40] and CIFAR-100 [40] is relatively low, about 90%. The reason for the analysis may be that CIFAR-10 [40] and CIFAR-100 [40] have different categories and cannot be constrained by the L_{margin} loss in (8). Better tuning of the hyperparameters of MSIN and its loss function will result in better performance.

4.9 Multi-sample generation using MSGAN

We extended the GAN [10] architecture to MSGAN to verify its ability to generate different samples. The architecture of MSGAN is shown in Table 2. Each linear layer of MSGAN is followed by a LeakRelu whose negative slope is set to 0.2. The dimension of the latent variable z is set to 200, and the vector after the generator is reshaped into a 28×28 image. Note that the generator generates samples of two different domains, respectively. The generated image is reshaped into a one-dimensional vector as the input to the discriminator, and its output layer uses two adversarial losses to determine whether the generated samples are true or false.

We use the Adam [26] method to train MSGAN, its learning rate is set to 2×10^{-4}, and the decay of first-order momentums of its gradient is set to 0.5 and 0.999, respectively. All of the balance factors in (9) are set to 1, and the entire training process trains 200 epochs on 2 NVIDIA Titan Xp GPUs with a batch size of 64.

Fig. 9 shows the image generated by MSGAN. It can be found that MSGAN can simultaneously generate samples from different sample domains, and handwritten numbers from the MNIST data set produce different patterns. For example, the handwritten number 3 and 9 in the figure. Since we are using a one-dimensional non-convolution neural network, the quality of the image needs to be further improved. We believe that using a deeper convolutional neural network and better adjusting the balance factor in (9) can further improve the quality of image generation, and we will further explore MSGAN in future work.

5 Discussion

5.1 Training MSIN with random label

In order to study whether the MSIN network learns the joint or independent distribution of multiple samples, we have set a classifier in MSIN-B to be a random label. This indicates that the MSIN classifier has some confusion about its input samples. Fig. 10 shows the training of the MSIN network with a random label on CIFAR-10 [40]. It can be found that the training and test accuracy of the classifier C_2 with a random label is all about 10%, i.e. the classifier is in a state of random guessing. The classifier C_1 training process is relatively normal. The accuracy of the single category is about 3.5% which is lower than multiple categories, indicating that the classifiers in the MSIN are not completely independent of each other, but still have higher independence. There is greater independence between MSIN's classifiers, which explains to a certain extent why MSIN is effective when classifying multiple samples.

5.2 Effect of the value of λ on MSIN performance

We change the value of λ in (5) to a random value in each min-batch to test whether the MSIN training process is stable, or whether alternate training can be used on the MSIN. The training process for this network is shown in Fig. 11. Compared with MSIN, both training and test accuracy have declined. It can be seen that the training error of MSIN is not close to 0, the accuracy of multi-category test has a slight decrease, and the accuracy of single category is seriously reduced. The single-category test accuracy has a large jitter and is even more unstable than the MSIN with a random label. When the value of a is close to zero, the feedback signal of the corresponding classifier approaches zero, causing the classifier to lose competitiveness. Therefore, using alternate training in MSIN will not be conducive to performance improvement.

6 Conclusions

The MSIN proved the interesting fact that the neural network can correctly predict multiple samples at the same time, which is worthy of theoretical research. MSIN and its variants can classify multiple samples simultaneously in one forward process. Our experiments show that this method can effectively separate multiple categories while avoiding the confusion of multiple samples. Since the MSIN can predict multiple samples without adding parameters, this can significantly reduce the forward process of the neural network, thereby reducing inference time and hardware consumption. The properties of MSIN can be used to solve the category expansion problem. It can not only make the extended network have better generalisation ability for new

Table 1 MSIN test error on different data sets

	C10	MNIST	C100	MNIST	C10	C100	C10	SVHN
original category	5.8	0.8	25.8	0.8	5.8	25.8	5.8	3.7
multiple category	6.2	0.8	26.5	0.8	6.8	31.4	8.8	3.9
single category	6.3	0.8	26.7	0.8	7	32.1	9.1	3.9

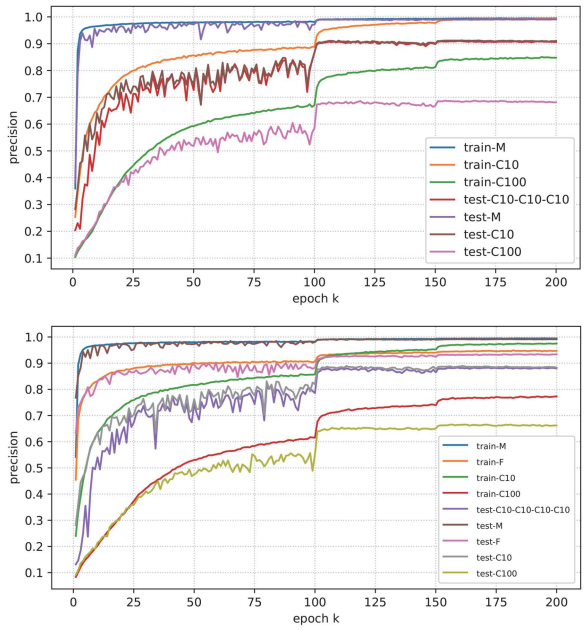

Fig. 6 *Performance of the MSIN on MNIST (M), Fashion-MNIST (F), CIFAR-10 (C10) and CIFAR-100 (C100) data sets. The suffix in the legend is shorthand for a data set that represents the performance of MSIN on that data set*

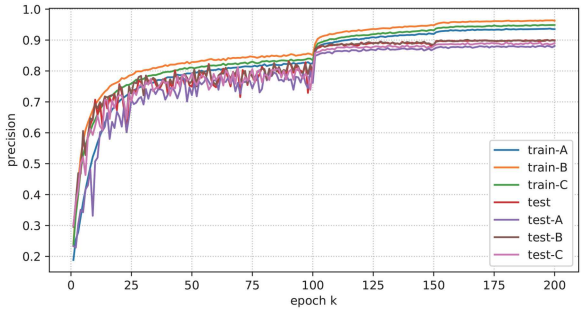

Fig. 7 *MSIN is used to train multi-channel samples from the CIFAR-10 data set. The suffixes A, B, and C in the legend represent the first, second, and third channels from different samples. The word 'test' without suffixes represents a sample from the CIFAR-10 data set*

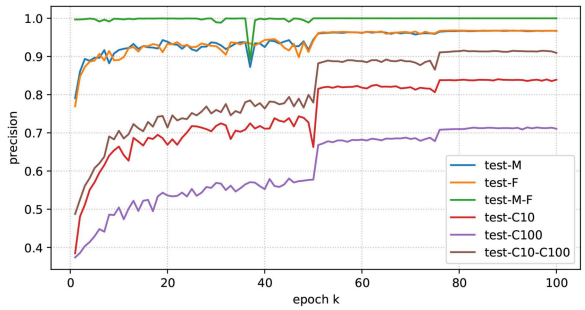

Fig. 8 *Performance of the MSIN on sample domain prediction. The suffix in the legend is a shorthand for a data set that represents the performance of MSIN on that data set*

categories but also can maintain the prediction performance of existing categories.

Table 2 MSGAN architecture

Model	Generator	Discriminator
Shared	Liner (z, 128)	Liner(28 × 28 × 2512)
	Liner (128, 256)	Liner(512, 256)
	Liner (256, 512)	
	Liner (512, 1024)	
Final	{Liner (1024, 28 × 28)} × 2	{Liner(256, 1), sigmoid()} × 2

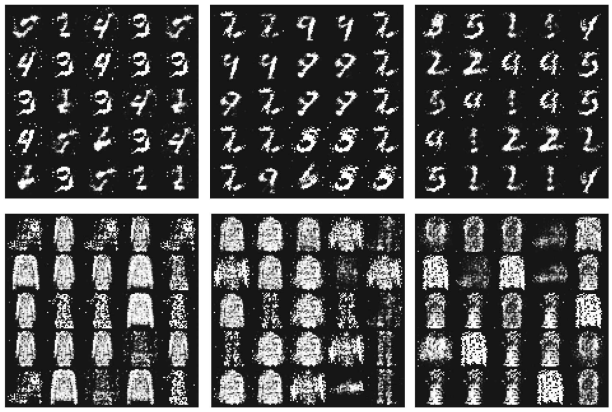

Fig. 9 *MSGAN is used to generate samples from the MNIST (first row) and Fashion-MNIST (second row) data sets, respectively*

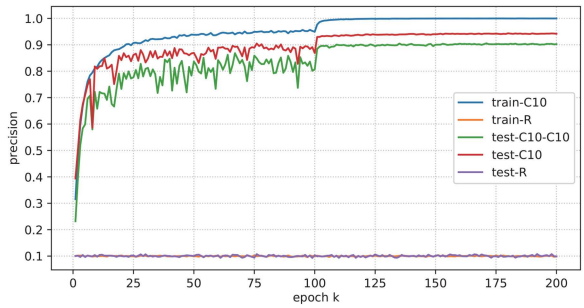

Fig. 10 *MSIN performance on CIFAR-10 with a classifier set to random labels. The suffix R in the legend represents the samples that is set as random labels, and C10 represents the normal samples*

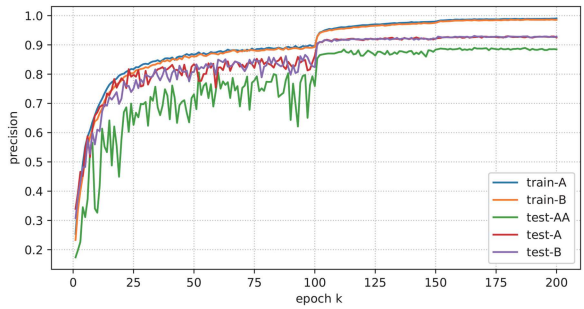

Fig. 11 *Effect of the value of λ on MSIN performance. The suffix A in the legend represents the sample classified by the classifier C_1, and the suffix B represents the sample classified by the classifier C_2*

On the basis of the MSIN architecture, there are lots of work to do: we can try to maximise the diversity of the final block to improve the prediction accuracy of the multiple categories, such as the use of different network structures or different blocks. The

ensemble training approach can also be used that allows for the addition of sample channels at the initial block. For example, use an odd number of final layers to vote on the final prediction. These are some of the most interesting things to do, including using this architecture in a MSGAN. The generator can be made to generate different patterns of samples in a forward process, and we will leave this work for the future.

7 Acknowledgments

This work is partially supported by the Natural Science Foundation of Shandong China (No.61373081) and the Taishan Scholar Project of Shandong, China.

8 References

[1] Xie, L., Wang, J., Wei, Z., *et al.*: 'DisturbLabel: regularizing CNN on the loss layer'. 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, June 2016, pp. 4753–4762

[2] Inoue, H.: 'Data augmentation by pairing samples for images classification'. CoRR, abs/1801.02929, 2018

[3] Zhang, H., Cissé, M., Dauphin, Y.N., *et al.*: 'Mixup: beyond empirical risk minimization'. ICLR, 2017

[4] Chopra, S., Hadsell, R., LeCun, Y.: 'Learning a similarity metric discriminatively, with application to face verification'. 2005 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'05), San Diego, USA, June 2005, vol. 1, pp. 539–546

[5] Norouzi, M., Fleet, D.J., Salakhutdinov, R.R.: 'Hamming distance metric learning', in Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.): '*Advances in neural information processing systems 25*' (Curran Associates Inc., Harrahs and Harveys, Lake Tahoe, USA, 2012), pp. 1061–1069

[6] Zagoruyko, S., Komodakis, N.: 'Learning to compare image patches via convolutional neural networks'. The IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Boston, USA, June 2015, pp. 4353–4361

[7] Schroff, F., Kalenichenko, D., Philbin, J.: 'FaceNet: A unified embedding for face recognition and clustering'. The IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Boston, USA, June 2015, pp. 815–823

[8] Wen, Y., Zhang, K., Li, Z., *et al.*: 'A discriminative feature learning approach for deep face recognition', in Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.): '*Computer vision – ECCV 2016*' (Springer International Publishing, Cham, 2016), pp. 499–515

[9] Liu, W., Wen, Y., Yu, Z., *et al.*: 'Large-margin softmax loss for convolutional neural networks'. Proc. 33rd Int. Conf. on Machine Learning, volume 48 of Proc. Machine Learning Research, New York, New York, USA, 20–22 June 2016, pp. 507–516, PMLR

[10] Goodfellow, I., Pouget-Abadie, J., Mirza, M., *et al.*: 'Generative adversarial nets', in Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (Eds.): '*Advances in neural information processing systems 27*' (Curran Associates Inc., Montreal, Quebec, Canada, 2014), pp. 2672–2680

[11] Zhang, C., Bengio, S., Hardt, M., *et al.*: 'Understanding deep learning requires rethinking generalization'. CoRR, abs/1611.03530, 2016

[12] Russakovsky, O., Deng, J., Su, H., *et al.*: 'Imagenet large scale visual recognition challenge', *Int. J. Comput. Vis.*, 2014, **115**, (3), pp. 211–252

[13] Lee, C.Y., Xie, S., Gallagher, P., *et al.*: 'Deeply-supervised nets'. Eprint Arxiv, 2014, pp. 562–570

[14] Zagoruyko, S., Komodakis, N.: 'Wide residual networks', arXiv preprint arXiv:1605.07146, 2016

[15] He, K., Zhang, X., Ren, S., *et al.*: 'Deep residual learning for image recognition'. 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, June 2016, vol. 00, pp. 770–778

[16] Xie, S., Girshick, R., Dollar, P., *et al.*: 'Aggregated residual transformations for deep neural networks'. 2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Honolulu, Hawaii, USA, July 2017, vol. 00, pp. 5987–5995

[17] Huang, G., Liu, Z., Weinberger, K.Q.: 'Densely connected convolutional networks'. CVPR, 2016, pp. 2261–2269

[18] Liang, D., Yang, F., Zhang, T., *et al.*: 'Wpnets and pwnets: from the perspective of channel fusion', *IEEE. Access.*, 2018, **6**, pp. 1–11

[19] Nair, V., Hinton, G.E.: 'Rectified linear units improve restricted Boltzmann machines'. Int. Conf. Machine Learning, Haifa, Israel, 2010, pp. 807–814

[20] He, K., Zhang, X., Ren, S., *et al.*: 'Delving deep into rectifiers: surpassing human-level performance on imagenet classification'. 2015 IEEE Int. Conf. Computer Vision (ICCV), Santiago, Chile, December 2015, pp. 1026–1034

[21] Jarrett, K., Kavukcuoglu, K., Ranzato, M., *et al.*: 'What is the best multi-stage architecture for object recognition?' 2009 IEEE 12th Int. Conf. Computer Vision, Kyoto, Japan, September 2009, pp. 2146–2153

[22] Xu, B., Wang, N., Chen, T., *et al.*: 'Empirical evaluation of rectified activations in convolutional network'. CoRR, abs/1505.00853, 2015

[23] Shang, W., Sohn, K., Almeida, D., *et al.*: 'Understanding and improving convolutional neural networks via concatenated rectified linear units'. CoRR, abs/1603.05201, 2016

[24] Duchi, J., Hazan, E., Singer, Y.: 'Adaptive subgradient methods for online learning and stochastic optimization', *J. Mach. Learn. Res.*, 2011, **12**, (7), pp. 257–269

[25] Bergstra, J., Yamins, D.L.K., Cox, D.D.: 'Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures', *JMLR*, 2013, **28**, pp. 115–123

[26] Kingma, D.P., Ba, J.: 'Adam: a method for stochastic optimization'. ICLR, 2014

[27] Krizhevsky, A., Sutskever, I., Hinton, G.E.: 'Imagenet classification with deep convolutional neural networks'. Int. Conf. Neural Information Processing Systems, Harrahs and Harveys, Lake Tahoe, USA, 2012, pp. 1097–1105

[28] Srivastava, N., Hinton, G., Krizhevsky, A., *et al.*: 'Dropout: a simple way to prevent neural networks from overfitting', *J. Mach. Learn. Res.*, 2014, **15**, (1), pp. 1929–1958

[29] Ba, J., Frey, B.: 'Adaptive dropout for training deep neural networks', in Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (Eds.): '*Advances in neural information processing systems 26*' (Curran Associates Inc., Harrahs and Harveys, Lake Tahoe, USA, 2013), pp. 3084–3092

[30] Ioffe, S., Szegedy, C.: 'Batch normalization: accelerating deep network training by reducing internal covariate shift'. Proc. 32nd Int. Conf. on Machine Learning, Lille, France, 2015

[31] Wan, L., Zeiler, M., Zhang, S., *et al.*: 'Regularization of neural networks using dropconnect'. Proc. 30th Int. Conf. on Machine Learning, volume 28 of Proc. of Machine Learning Research (PMLR), Atlanta, Georgia, USA, 17–19 June 2013, pp. 1058–1066.

[32] Zeiler, M.D., Fergus, R.: 'Stochastic pooling for regularization of deep convolutional neural networks'. CoRR, 2013

[33] Vapnik, V.N.: '*Statistical learning theory*' (John wiley, 1998)

[34] Kang, G., Dong, X., Zheng, L., *et al.*: 'Patchshuffle regularization'. CoRR, abs/1707.07103, 2017

[35] Zhong, Z., Zheng, L., Kang, G., *et al.*: 'Random erasing data augmentation'. CoRR, abs/1708.04896, 2017

[36] He, K., Zhang, X., Ren, S., *et al.*: 'Identity mappings in deep residual networks', in Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.): '*Computer vision – ECCV 2016*' (Springer International Publishing, Cham, 2016), pp. 630–645

[37] Netzer, Y., Wang, T., Coates, A., *et al.*: 'Reading digits in natural images with unsupervised feature learning'. Nips Workshop on Deep Learning & Unsupervised Feature Learning, Granada Spain, 2011

[38] Lecun, Y.: 'The MNIST database of handwritten digits'. http://yann.lecun.com/exdb/mnist/, 1998

[39] Xiao, H., Rasul, K., Vollgraf, R.: 'Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms'. arXiv preprint arXiv:1708.07747, 2017

[40] Krizhevsky, A.: 'Learning multiple layers of features from tiny images'. Tech Report, 05 2012, University of Toronto, Toronto, Ontario, Canada

[41] Lin, M., Chen, Q., Yan, S.: 'Network in network'. ICLR, 2013

[42] Romero, A., Ballas, N., Kahou, S.E., *et al.*: 'Fitnets: hints for thin deep nets'. ICLR, 2014

[43] Springenberg, J.T., Dosovitskiy, A., Brox, T., *et al.*: 'Striving for simplicity: the all convolutional net'. Eprint Arxiv, 2014

[44] Srivastava, R.K., Greff, K., Schmidhuber, J.: 'Training very deep networks', in Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (Eds.): '*Advances in neural information processing systems 28*' (Curran Associates Inc., Montreal, Quebec, Canada, 2015), pp. 2377–2385

[45] Huang, G., Sun, Y., Liu, Z., *et al.*: 'Deep networks with stochastic depth', in Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.): '*Computer vision – ECCV 2016*' (Springer International Publishing, Cham, 2016), pp. 646–661

[46] Larsson, G., Maire, M., Shakhnarovich, G.: 'Fractalnet: ultradeep neural networks without residuals'. CoRR, abs/1605.07648, 2016

[47] Sutskever, I., Martens, J., Dahl, G., *et al.*: 'On the importance of initialization and momentum in deep learning'. Int. Conf. Machine Learning, Atlanta, USA, 2013, pp. III–1139