Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Continuously evolving dropout with multi-objective evolutionary optimisation

Pengcheng Jiang [a], Yu Xue [a,*], Ferrante Neri [b]

[a] School of Computer Science, Nanjing University of Information Science and Technology, Nanjing, 210044, China
[b] NICE group, Department of Computer Science, University of Surrey, Guildford, GU2 7XH, United Kingdom

A R T I C L E I N F O

Keywords:
Genetic algorithms
Multi-objective optimisation
Deep neural networks
Over-fitting
Dropout

A B S T R A C T

Dropout is an effective method of mitigating over-fitting while training deep neural networks (DNNs). This method consists of switching off (dropping) some of the neurons of the DNN and training it by keeping the remaining neurons active. This approach makes the DNN general and resilient to changes in its inputs. However, the probability of a neuron belonging to a layer to be dropped, the 'dropout rate', is a hard-to-tune parameter that affects the performance of the trained model. Moreover, there is no reason, besides being more practical during parameter tuning, why the dropout rate should be the same for all neurons across a layer. This paper proposes a novel method to guide the dropout rate based on an evolutionary algorithm. In contrast to previous studies, we associate a dropout with each individual neuron of the network, thus allowing more flexibility in the training phase. The vector encoding the dropouts for the entire network is interpreted as the candidate solution of a bi-objective optimisation problem, where the first objective is the error reduction due to a set of dropout rates for a given data batch, while the second objective is the distance of the used dropout rates from a pre-arranged constant. The second objective is used to control the dropout rates and prevent them from becoming too small, hence ineffective; or too large, thereby dropping a too-large portion of the network. Experimental results show that the proposed method, namely GADropout, produces DNNs that consistently outperform DNNs designed by other dropout methods, some of them being modern advanced dropout methods representing the state-of-the-art. GADroput has been tested on multiple datasets and network architectures.

1. Introduction

Deep learning constitutes an important branch of machine learning. It has developed rapidly in recent years and been widely applied to problems in many fields, such as image classification (Kokalis et al., 2020), object detection (Zhang et al., 2021) and time-series prediction (Xue et al., 2021b). The training stage is the most important process in applying a deep neural network to solve practical problems. To achieve the best overall performance, the parameters of the neural network need to be optimised with gradient-based back-propagation. At present, due to the lack of training data for increasingly complex tasks, the depth and width of neural networks have been greatly expanded, which leads to problems in the training process, causing over-fitting on the dataset (Xue et al., 2022b). Over-training on the training set is very susceptible to the problem of over-fitting, where reductions in training errors are accompanied by increases in generalisation errors, which can lead to the failure of the network at the test stage.

To avoid the over-fitting problem, researchers have developed many methods, which can be roughly divided into negative methods, semi-active methods and active methods. **(1) Negative methods** depend on large-scale searches on numerous models and hyper-parameters to find deep neural networks with good generalisation capacity. Negative methods include neural architecture search (Zhang et al., 2022; Lu et al., 2021) and neural network ensemble learning (Ma et al., 2022; Chandra and Yao, 2006). **(2) Semi-active methods** involve making small modifications to the model and fine-tuning later to avoid over-fitting. These include dynamic network structure and network pruning (Zhou et al., 2022, 2021). **(3) Active methods** prevent over-fitting through regularisation during training. The methods most frequently used are L_2, normalisation, data enhancement, weight decay (Krogh and Hertz, 1991; Loshchilov and Hutter, 2019) and dropout (Srivastava et al., 2014).

'Dropout' is to randomly inactivate a subset of neurons during training (Yin et al., 2021; Valcarce et al., 2019). The probability of inactivation is called the dropout rate. Hinton et al. (2012), Many studies have shown that the generalisation ability of the trained model can be greatly improved by using the dropout method. Many forms of dropout method have been proposed, including by Bernoulli (Ba and Frey, 2013) and Gaussian (Wang and Manning, 2013; Gal et al., 2017).

* Corresponding author.
E-mail addresses: pcjiang@nuist.edu.cn (P. Jiang), xueyu@nuist.edu.cn (Y. Xue), f.neri@surrey.ac.uk (F. Neri).

https://doi.org/10.1016/j.engappai.2023.106504
Received 14 July 2022; Received in revised form 24 April 2023; Accepted 19 May 2023
Available online xxxx

Most of these methods use different distribution functions based on stochastic progress to determine neuron inactivation. Some studies in recent years have attempted to add some trainable parameters to the dropout layers (Xie et al., 2022; Pham and Le, 2021). These parameters are changeable by back-propagation. However, this introduces the problem that the learning rate of these trainable parameters is very difficult to set for training.

1.1. Related work: Dropout in linear layers

Researchers have been working on various methods to inhibit over-fitting. At present, dropout and its variants are the most effective methods. The main idea is to set a probability of inactivation (*i.e.*, dropout rate p) for each neuron. Through partial activation during training, the neural network improves its generalisation ability in a way similar to network integration. On the basis of standard dropout, the dropout rate used by Ba and Frey (2013) in the Bernoulli mask is no longer set manually, but calculated by activation function value. Direct deleting of neurons requires using $\frac{1}{1-p}$ to adjust the output, such that the output of the training process has the same distribution as the output of the testing process, which will increase the computation during the training process. Therefore, some researchers use Gaussian dropout methods to enhance or weaken neurons, thus controlling the output distribution by changing mean value (Wang and Manning, 2013; Kingma et al., 2015; Gal et al., 2017). In the following research, Shen et al. (2018) successfully applied this idea to all basic modules. In contrast to most dropout methods, DropConnect does not inactivate neurons but removes connections. This is also called weight dropout (Wan et al., 2013). Maxout can be seen as another variant of dropout. The objective is to add another linear layer with K neurons after the original layer, and only the maximum value of the outputs generated from these K neurons is retained. Tseng et al. (2020) proposed the DropGrad method, which drops the gradient in back propagation process.

Since linear structures also exist in recurrent neural networks (RNNs), dropout can also be used. Some researchers believe that different parts of the RNN structure have different functions and therefore require different dropout strategies. RNNdrop is used to randomly discard the long-term memory to prevent over-fitting in the training process (Moon et al., 2015). In long short-term memory (LSTM), Semeniuta et al. (2016) the part that is added into the long-term memory in each block is randomly discarded. Gal and Ghahramani (2016) applied variational dropout on the input of the current block. Krueger et al. (2017) make applied blocks randomly activated instead of neurons. After generative adversarial networks (GANs) appeared, the adversarial dropout method was proposed (Saito et al., 2018; Park et al., 2018). This method guides the dropout method through the process of adversarial learning, which is also applied to RNNs (Park et al., 2019).

1.2. Related work: Dropout in convolutional layers

Convolutional neural networks (CNNs) have different properties to linear structures. The direct random inactivation of neurons in the convolution layers cannot effectively introduce noise in the training process. Therefore, most dropout methods in linear layers are not applicable to such structures. Wu and Gu (2015) used standard dropout before max-pooling, thereby partially eliminating the original maximum of each calculation. Park and Kwak (2017) combined this method with Gaussian dropout and completed Max-Drop using Gaussian gate. However, these two methods do not add enough noise to the training process. Tompson et al. (2015) randomly dropped the feature map in a more efficient way for the object localisation task. In the improvement of this method by Hou and Wang (2019), global pooling and random coefficient are used to control this process. Shi et al. (2020) focused on discarding the location with small self-information to make the neural network more attentive to shape bias.

The cutout method can be regarded as a variant of the dropout method on the input of networks, which randomly masks squares in the input images (DeVries and Taylor, 2017), but cannot be applied during the calculation of convolution. Ghiasi et al. (2018) solve this problem by dropping a randomly chosen area of the inputs of layers, improving the performance of the network in image classification and object detection. In addition, with the popularity of ResNet, Huang et al. (2016) and Hayou and Ayed (2021) randomly inactivated the residual block with random depth method, also known as DropPath. These two methods are both applicable to all multi-branch network structures. After analysing the residual block, Singh et al. (2016) combined identity mapping with zero setting operation and proposed Swapout, which makes each element randomly chosen from x, $f(x)$ and $x + f(x)$. Choe et al. (2021) combined the attention mechanism with the dropout method and automatically generated self-attention map, drop mask, and importance map as the basis of dropping features in the training process.

1.3. Related work: Adaptive dropout

The dropout rate plays an important role in the dropout implementation, as its incorrect setting may jeopardise the functioning of the neural network. In particular, Rennie et al. (2014) observed that large dropout rate values can lead to ineffective learning, especially in the late stages of learning. As a countermeasure, Rennie et al. (2014) suggested an annealing approach: that is, a reduction of the dropout rate over learning time according to a negative exponential function. However, Morerio et al. (2017) demonstrated that too-low dropout values are also detrimental to learning, as they cause an over-fitting of the data. These findings support two proposals: (1) the correct dropout rate should be close to neither 0 nor 1; (2) since learning is a dynamic process, successful learning might require a dynamic dropout rate.

Since it is difficult to state what dropout rate is more reasonable by proofs, some studies use methods to intervene in this stochastic process. Lakshminarayanan et al. (2017) used dropout based on deep integration. Achille and Soatto (2018) designed the regularisation according to the information bottleneck. Xie et al. (2022) used trainable parameters to adjust the random matrix that controls the dropout progress.

Some researchers add heuristic elements to this stochastic process. Pham and Le (2021) use the controller to adjust the dropout rate based on reinforcement learning. Salehinejad and Valaee (2022) use Gibbs distribution and energy function to adjust dropout and prune the network. Evolutionary algorithms are also used to adjust dropout. Chen et al. (2018) improve a framework that enables the genetic algorithm (GA), and differential evolution can be used to control the dropped neurons. This method is like ensemble learning, which can most often yield good results in sub-network training. However, there is a risk of further increasing over-fitting. Guo et al. (2022) design a new layer and use AR-MOEA to control the activated neurons. However, this method is specifically used to solve the surrogate problem in SAEA, but is rarely applied to other tasks. It also requires the use of dropout in the test process, which introduces more errors.

1.4. Motivation and proposal

Considering that it is difficult to find the best value of the dropout rate through traditional methods, some researchers use approaches with automatic adjustments, but introduce redundant computation, which can affect the speed of training (Pham and Le, 2021). Such metaheuristic approaches as evolutionary computation are effective for tuning the hyper-parameters in the dropout layers. Instead of requiring a deep understanding of the problem, they encode the solutions and approach the problem by searching and retaining solutions on the basis of their objective function values alone. In the case of evolutionary computation, this search heuristic is inspired by the idea of ‘evolution’.

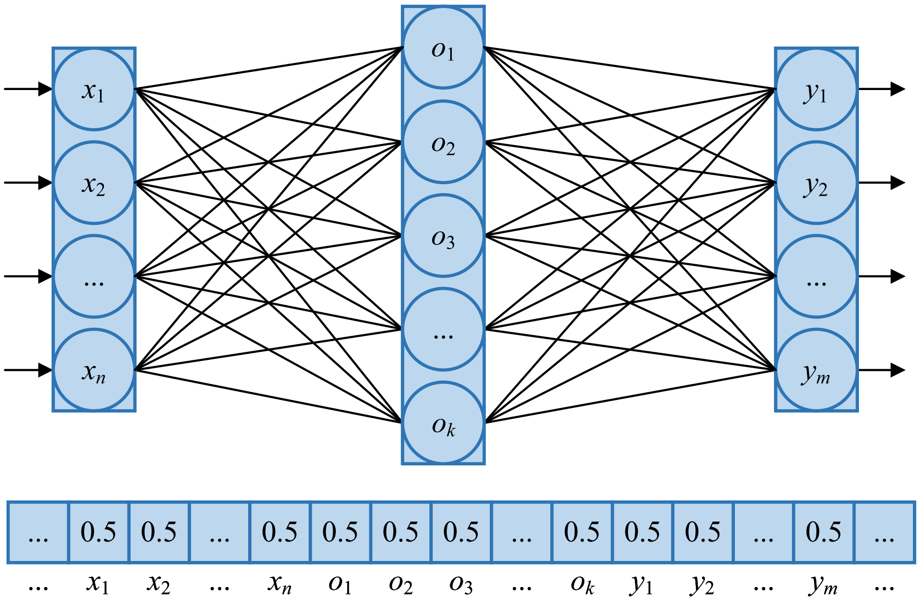

Fig. 1. Decoding strategy for dropout rates of each neuron.

In the context of neural network training and design, metaheuristics have been broadly used in various scenarios, such as Xue et al. (2021a,c, 2022b,c) and Xue et al. (2022a). In various contexts and for various purposes, metaheuristics and (especially) evolutionary methods have also been combined with dropout methods (Guo et al., 2022; Salehinejad and Valaee, 2022; Chen et al., 2018).

This paper proposes a metaheuristic approach to automating the dropout while enhancing its performance. More specifically, an adaptive dropout method based on Non-dominated Sorting Genetic Algorithm II (NSGA-II), called GADropout, is proposed to guide the dropout rate on each feature during training. The contributions of this paper are summarised as follows.

1. The evolutionary algorithm is fully introduced into the training process of the neural networks to dynamically adjust the probability of dropout.
2. The dropout rate of each feature in each layer can be flexibly varied, which greatly increases the flexibility of the algorithm.
3. The absolute value between the threshold and each individual is used as another objective to give the neural network the ability to automatically adapt to the data for different training phases.

2. GADropout

2.1. Encoding scheme and initialisation

In neural networks, the traditional dropout method adopts the same dropout rate, which needs to be set manually, for each neuron in the same layer. In addition, the dropout rate has been fixed during the whole training stage. In this paper, the dropout rates of each neuron are regarded as genes to form the individual, which are dynamically adjusted by an evolutionary algorithm. In this way, the most suitable dropout rates for each neuron in each layer can be used during the training process. We have proposed to encode the topology of a network as a vector of dropout rates. This vector is the candidate solution of the problem. As shown in Fig. 1, the dropout rate of each neuron corresponds to a position in the candidate solution. Each design variable (x_j, o_j, y_j in Fig. 1) is a decimal number between 0 and 1 representing the dropout rate of that neuron; that is, the probability that the neuron is inactivated during back propagation.

Let us consider that the proposed algorithm is population-based. With the purpose of introducing the notation, we indicate with $Ind_{i,j}$ the jth position of dropout rate within the ith candidate solution/individual (*i.e.*, $Ind_{i,*}$). The initialisation of the proposed GADropout algorithm consists of randomly generating a population of vectors whose values are between 0 and 1.

In the decoding stage, since dropout is only carried out during training, the scale coefficient should be used to scale the discarded value of neurons in each layer in order to maintain the same data distribution of neuron output during training and testing. Considering that most algorithms are updated based on batch stochastic gradient descent (Batch SGD), for each neuron $X_{*,j}$ that is processed under the corresponding dropout rate $Ind_{i,j}$, the mathematical expectation is as Eq. (1)

$$E(X_{*,j}) = X_{*,j} \times Ind_{i,j} \tag{1}$$

where X represents the neuron input, and k represents the index of the input among the current batch. The mathematical expectation data value of the current batch data after GADropout is calculated as Eq. (2).

$$E(X_{k,*}) = X_{k,*} \times Ind_{i,*} \tag{2}$$

To simplify and reduce the amount of calculation in the validating and testing process, the actual output through GADropout during training is calculated as Eq. (3)

$$Y = \frac{X \otimes M}{Ind_{i,*}} \tag{3a}$$

$$M_{*,j} \sim Bernoulli(k; Ind_{i,j}) \tag{3b}$$

where $\otimes$ is element-wise multiplication and M is a binary mask matrix in which each row $M_{*,j}$ is a vector of length k obeying a Bernoulli distribution with probability $Ind_{i,j}$. In the mask matrix (*i.e.*, M), each sub-mask $M_{*,j}$ is used for the inputs of a batch at the jth neuron. In the validating and testing process, the GADropout is skipped.

2.2. Multi-objective optimisation

GADropout manipulates and evolves vectors of dropout rates on the basis of two objectives. We formulate the selection of dropout rates as a bi-objective optimisation problem. For each generic individual $Ind_{i,*}$, the first fitness value f_1 is calculated as follows. First, two data batches for training and validation, D_{train} and D_{val} respectively, are randomly selected from the data set. The validation error e_1 of DNN is calculated using D_{val}. Then, the DNN is trained with the dropout rates encoded in $Ind_{i,*}$. SGD is used for training. The validation error e_2 is then calculated again, using D_{val}. The objective function f_1 is the difference between the errors (Line 6 in Algorithm 1).

$$f_1 = e_2 - e_1. \tag{4}$$

The objective function f_1 represents the improvement, in terms of accuracy, due to the dropout strategy encoded in $Ind_{i,*}$. Since both e_1 and e_2 are positive definite, the sign of f_1 indicates whether this dropout strategy is beneficial or detrimental. In particular, the codomain of f_1 is $[-1, 1]$. If f_1 is negative, the dropout strategy is beneficial; if f_1 is positive, the dropout strategy is detrimental.

While f_1 assesses the quality inferred by the dropout strategy encoded in $Ind_{i,*}$, it does not take into consideration an important issue associated with the concept of dropout. Multiple studies show that the dropout rates should be neither too high nor too low (Garbin et al., 2020). A dropout rate ≈ 1 means that no dropout occurs, with the corresponding risk of over-fitting. Conversely, a dropout rate ≈ 0 means that the neuron has no outputs, thus risking that some features are excluded from training. Although several studies empirically suggest appropriate ranges of variation for dropout rates, there is no theoretical justification behind any specific setting for these bounds. On this premise, we propose to embed the effective ranges of dropout rates into the second objective function f_2. This objective function measures the average distance from a reference parameter t:

$$f_2 = \frac{1}{L}\sum_{j=0}^{L} |Ind_{i,j} - t| \tag{5}$$

where L is the length of the individual and t is a threshold value set in our experiments to 0.5, the middle of the range for dropout rates. The objective function f_2 thus has the role of guiding the search and preventing the dropout rates from approaching the undesired two extremes of their theoretical range (0 and 1). At the same time, this formulation of f_2 enables us to not have to set too-exact bounds for the ranges of dropout rates.

The details of the calculation of the two objective functions are shown in Algorithm 1.

2.3. Evolutionary framework

The population of dropout strategies $Ind_{i,*}$, with its corresponding objective functions, is processed by an evolutionary algorithm to detect promising dropout strategies. GADropout processes the DNN that the use intends to train and the corresponding dataset. First, the number of neurons L of the DNN is calculated. The initial population P_0 of dropout strategies $Ind_{i,*} \in [0, 1]^L$ is randomly generated. The individuals of P_0 are then evaluated, according to Algorithm 1, to generate a data structure F_0 containing the objective function values of each individual.

Algorithm 1 Calculation of the objective function values. **Evaluate**()

```
Input: Deep neural network DNN and individual (vector of dropout
    rates) $Ind_{i,*}$.
 1: $D_{train}$ ← Randomly sample a training batch.
 2: $D_{val}$ ← Randomly sample a validation batch.
 3: $e_1$ ← Validation error inferred by $D_{val}$ on DNN.
 4: Train DNN by using $D_{train}$ and the vector of dropout rates $Ind_i$.
 5: $e_2$ ← Validation error inferred by $D_{val}$ on DNN (after being trained
    with dropout).
 6: $f_1 = e_2 - e_1$.
 7: $f_2 = \sum_{j=0}^{L} |Ind_{i,j} - t|$.
 8: Append $f_1$ and $f_2$ to $Fitness$, i.e., $Fitness \leftarrow (f_1, f_2)$
 9: return $Fitness$
```

Algorithm 2 GADropout: Evolutionary framework.

```
Input: Deep neural network DNN, data set of size $N$.
Output: Trained deep neural network DNN.
 1: $L$ ← Calculate number of neurons in DNN.
 2: $P_0$ ← Initialise population.
 3: $F_0$ ← Evaluate($P_0$) with Algorithm 1
 4: for $g = 1$ to $G$ do
 5:     $Q_g$ ← Get new offspring with uniform crossover and polynomial
        mutation.
 6:     $F'_g$ ← Evaluate($Q_g$) ∪ $F_{g-1}$ (use Algorithm 1)
 7:     $P'_g \leftarrow Q_g \cup P_{g-1}$.
 8:     Non-dominated sorting.
 9:     $P_g$ ← Retain individuals in $P'_g$ with crowding rules.
10:     $F_g$ ← Delete the corresponding fitness value in $F'_g$.
11: end for
12: return The parameters of trained neural network DNN.
```

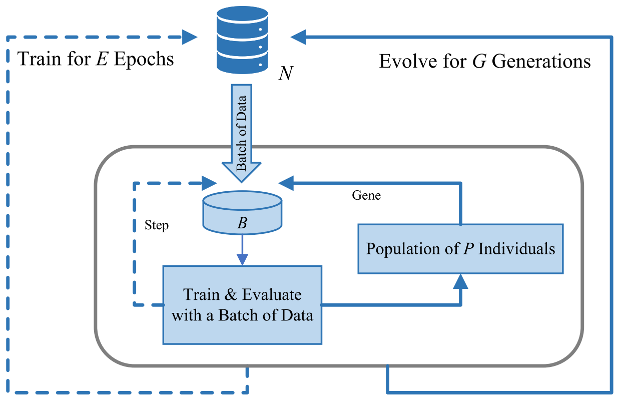

Fig. 2. The overall process of searching. The dashed lines signify the original training process, and the solid lines signify the iterative process of the population.

Then a loop of G generations is performed. At each generation g, from the population at the gth generation, P_g, the offspring population Q_g is produced by applying uniform crossover (Eiben and Smith, 2015) and polynomial mutation (Deb and Tiwari, 2008) to solutions selected by means of the binary tournament selection described in Deb et al. (2002). It is worth mentioning that we chose uniform crossover to limit its exploration, as we have observed that losing promising values of dropout rates tended to deteriorate the performance of DNN. Thus, we chose a crossover operator that preserves them. On the other hand, the polynomial mutation according to the implementation of Deb and Tiwari (2008) enables the generation of new dropout rates for a restricted number of design variables $Ind_{i,j}$ (dropout rates). The newly-generated individuals comprising the offspring population Q_g are then evaluated according to Algorithm 1, and both parent and offspring population are merged to produce the provisional population P'_g. The fast non-dominated sorting and crowding rules described in Deb et al. (2002) are then applied to select the population for the subsequent generation. Algorithm 2 outlines the steps of the evolutionary framework of GADropout.

The overarching scheme describing the evolutionary process of the solutions is depicted in Fig. 2. It shows how the population of P individuals $Ind_{i,*}$ interacts with the database and the training process. Note that two concurrent optimisation processes are integrated into the framework. The first is the evolution, over G generations, of the dropout strategies. The second is the training, over E epochs, of each DNN. Each training is performed on a data batch (D_{train} and D_{val} in Section 2.2) of size B, while N is the size of the entire dataset.

Note that DNN training by means of the GADropout outlined in Algorithm 2 requires time comparable to that of a standard DNN training. Specifically, the time complexity of the algorithm general remains essentially unchanged, as long as Eq. (6) is verified.

$$E \times \left\lceil \frac{N}{B} \right\rceil \approx G \times P \tag{6}$$

In other words, the training times are comparable as long as GADropout and traditional training process the same amount of data. GADropout may introduce some overhead due to algorithmic steps like the non-dominated sorting, but this is only minor.

To share the implementation details of our method with the reader, we have published our code at https://github.com/pcjiang1998/GADropout.

3. Experiments and analysis

3.1. Datasets

To demonstrate the effectiveness of the proposed method in preventing over-fitting during neural network training, sufficient experiments on multiple datasets were carried out. Some small and medium-sized datasets on multi-layer perception were collected from UCI Machine Learning Repository, including 'Wine',[1] 'Letter Recognition',[2] 'dorothea',[3] 'Dry Bean Dataset'[4] and 'Shill Bidding Dataset'.[5] In addition, several large datasets were collected, including 'Swarm Behaviour Data'[6] and 'Winnipeg Dataset'.[7] The smallest dataset had about 200 samples, while the largest had over 320,000 samples. In addition, the 'MNIST'[8] and 'Fashion MNIST'[9] datasets were flattened and used as one-dimensional data. The MNIST and Fashion MNIST datasets both have 10 categories, composed of 60,000 grey-scale pictures of size 28 × 28 for training and 10,000 pictures for testing, including, respectively, pictures of handwritten digits and pictures of grey-scale clothing.

To test the availability of the proposed method on classifiers in convolutional neural networks, experiments were performed on various scale datasets. Lenet was selected as the neural network model on MNIST and Fashion MNIST datasets. CIFAR dataset[10] is a medium-sized dataset commonly used in image classification tasks, which has two versions. CIFAR-10 has 10 categories, and CIFAR-100 has 100 categories. They are both composed of 60,000 pictures of size 32 × 32 × 3, with 50,000 images for training and 10,000 images for testing. Another dataset, the street view house numbers (SVHN) dataset,[11] contains 99,289 colour pictures of house numbers: 73,257 images for training and 26,032 images for testing. It contains a large number of house numbers of different sizes, mixed with a lot of irrelevant noise, so it is more practical. In addition, down-sampled Imagenet dataset[12] was also selected for experimental comparison. Imagenet is a huge dataset, containing millions of images in 1000 categories, posing a huge challenge to the classification tasks. To demonstrate the capacity of the proposed method to solve the over-fitting problem, the sub-dataset, which was down-sampled, was used for the experiments, *i.e.*, mini-Imagenet and Imagenet-32 × 32 were selected.

[1] https://archive.ics.uci.edu/ml/datasets/Wine.
[2] https://archive.ics.uci.edu/ml/datasets/Letter+Recognition.
[3] https://archive.ics.uci.edu/ml/datasets/dorothea.
[4] https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset.
[5] https://archive.ics.uci.edu/ml/datasets/Shill+Bidding+Dataset.
[6] https://archive.ics.uci.edu/ml/datasets/Swarm+Behaviour.
[7] https://archive.ics.uci.edu/ml/datasets/Crop+mapping+using+fused+optical-radar+data+set.
[8] http://yann.lecun.com/exdb/mnist/.
[9] https://github.com/zalandoresearch/fashion-mnist.
[10] https://www.cs.toronto.edu/~kriz/cifar.html.
[11] http://ufldl.stanford.edu/housenumbers/.
[12] https://image-net.org/.

3.2. Parameter setting

For the above datasets, networks of different sizes were used to prove the effectiveness of the algorithm. To demonstrate the effectiveness of the proposed method and to ensure a fair comparison, each network was trained for the same number of epochs on the same data. From all datasets that were not classified as test sets, 20% was taken from the training dataset for testing. From all datasets that were not divided into validation datasets, 20% was extracted from the remaining training datasets for validation. After the partition, the rest of the training datasets were used as the actual training datasets. Specifically, for multi-layer perception, each set of experiments was conducted on the complete training dataset for 200 epochs, and the learning rate was set to 0.01. Three classic convolutional neural networks, Lenet, VGG16 and ResNet18, were used for comparison. Each set of experiments trained 300 epochs on the complete training dataset. During the training process, the learning rate decreased to one-tenth at 150th epoch and 225th epoch. During the training process in all experiments, cross-entropy loss was used for back propagation, and stochastic gradient descent (SGD) optimiser was used to optimise the parameters, in which weight decay was set to 5e–4 and momentum was set to 0.9. Since the process of GADropout is an iterative process based on population, the population size and number of search generations needed to be set. Assuming that the number of training epochs originally required is E, the total amount of training data is N, and the size of each batch obtained by SGD optimiser during training is B, the population size P is set to $\lceil N/B \rceil$ and the number of search epochs is set to E. Therefore, the total number of evaluations is $\lceil N/B \rceil \times E$. In all experiments using the proposed method, the threshold in the optimisation objective f_2 was set to 0.5 by experience.

To further illustrate the ability of our method to consistently outperform the other algorithms, we repeated the experiments on each dataset multiple times. For the datasets from UCI, MNIST and FashionMNIST, we ran each algorithm 10 times. For the SVHN and CIFAR datasets, we ran each set of experiments for 5 times as each run requires a large computational time. For each problem instance we show average value, standard deviation and corresponding p-value.

3.3. Performances on UCI datasets

First, the datasets from UCI were used in the experiment. On the Winnipeg dataset, the batch size was set to 128. On all other datasets, the batch size was set to 32. Thirteen comparison algorithms were selected for algorithm comparison (see Table 1). All deep neural networks contain two hidden layers, both with 1024 neurons. To prevent the neural networks from being difficult to train due to large values in features, all features in the datasets were normalised before training. For non-numeric data in features, one-hot encoding was used. The final results are shown in Table 1, and GADropout achieves good results in various comparative experiments. To illustrate the convergence of GADropout, we plot the population values in the objective space on three datasets, which are 'Dry Bean Dataset', 'Shill Bidding Dataset' and 'Letter Recognition'. Their hyper-volume is 1.22, 1.11 and 2.72, separately. Each figure shows the population of the 50th, 100th, 150th and 200th generations. From Fig. 3, we see that GADropout can continuously search in smaller directions for both objectives.

3.4. Performances on MNIST and fashion MNIST

In this experiment, some comparisons were made on MNIST and Fashion MNIST datasets. First, the deep neural network was used to test the effectiveness of the proposed method, using the same structure as the experiments in Section 3.3. Second, the comparison experiment was performed again, using variant structures based on Lenet, whose first layer is filled with 2 pixels. In the experiments on these two structures, batch size was set to 256. As mentioned in Section 3.2, in

Table 1
Classification accuracy (%) of experiments conducted on UCI datasets using multi-layer perceptrons with two hidden layers. (The last line represents the total numbers of evaluations during evolution.)

	Wine	Letter Recognition	Swarm Behaviour	Winnipeg	Dorothea	Dry Bean	Shill Bidding
No dropout	85.36 ± 2.50 1.028e−10	96.01 ± 0.10 7.624e−16	67.79 ± 0.06 6.698e−21	99.40 ± 0.15 6.353e−02	96.09 ± 0.92 2.363e−07	90.93 ± 0.15 7.020e−17	97.59 ± 0.30 2.869e−14
Standard dropout (Hinton et al., 2012)	93.21 ± 1.07 5.117e−07	96.10 ± 0.08 7.266e−16	67.95 ± 0.05 9.558e−21	99.44 ± 0.16 2.433e−01	96.40 ± 1.03 2.617e−06	91.50 ± 0.15 2.316e−14	97.95 ± 0.31 7.968e−13
Gaussian dropout (Srivastava et al., 2014)	94.64 ± 1.79 2.946e−04	96.19 ± 0.09 8.196e−15	67.98 ± 0.07 2.505e−19	99.48 ± 0.15 6.545e−01	96.11 ± 0.63 1.653e−08	91.79 ± 0.15 1.289e−12	98.37 ± 0.27 9.635e−12
Variational dropout (Kingma et al., 2015)	87.50 ± 1.79 8.209e−11	96.28 ± 0.07 4.781e−15	68.03 ± 0.09 7.735e−18	99.42 ± 0.14 9.866e−02	97.23 ± 0.92 8.522e−05	91.15 ± 0.14 3.010e−16	98.06 ± 0.30 1.675e−12
DropConnect (Wan et al., 2013)	88.21 ± 1.64 1.285e−10	96.42 ± 0.11 1.964e−12	68.33 ± 0.07 1.021e−16	99.38 ± 0.16 3.860e−02	97.37 ± 0.99 2.958e−04	91.41 ± 0.12 1.970e−15	98.15 ± 0.36 3.409e−11
Concrete dropout (Gal et al., 2017)	92.50 ± 1.92 1.937e−06	96.96 ± 0.09 2.140e−06	68.42 ± 0.09 8.519e−15	99.44 ± 0.11 1.064e−01	97.46 ± 1.11 9.666e−04	92.74 ± 0.12 1.252e−03	98.19 ± 0.33 1.829e−11
Fraternal dropout (Zolna et al., 2018)	94.29 ± 1.75 9.923e−05	96.32 ± 0.11 3.159e−13	68.46 ± 0.08 7.905e−15	99.38 ± 0.07 1.600e−03	97.63 ± 0.65 1.689e−04	91.73 ± 0.12 7.009e−14	98.28 ± 0.27 2.610e−12
Gradient dropout (Tseng et al., 2020)	92.14 ± 3.11 4.388e−05	96.92 ± 0.10 9.144e−07	68.58 ± 0.07 7.594e−14	99.42 ± 0.09 4.391e−02	97.54 ± 0.65 9.282e−05	92.61 ± 0.14 3.291e−05	99.60 ± 0.29 1.237e−02
Meta dropout (Lee et al., 2020)	92.50 ± 2.97 5.798e−05	97.08 ± 0.10 1.138e−03	68.62 ± 0.09 1.386e−12	99.41 ± 0.07 1.037e−02	97.88 ± 1.00 5.461e−03	91.89 ± 0.10 3.204e−13	99.54 ± 0.21 5.286e−04
Auto dropout (Pham and Le, 2021)	95.71 ± 2.14 1.100e−02	97.18 ± 0.09 6.174e−02	68.80 ± 0.08 1.302e−10	99.49 ± 0.08 5.305e−01	98.14 ± 0.75 8.453e−03	92.88 ± 0.13 1.306e−01	99.72 ± 0.26 9.163e−02
Advance dropout (Xie et al., 2022)	97.14 ± 2.67 3.058e−01	97.44 ± 0.08 1.193e−03	68.79 ± 0.09 6.230e−10	99.41 ± 0.09 2.252e−02	98.66 ± 0.80 1.832e−01	92.87 ± 0.14 1.064e−01	99.84 ± 0.25 6.016e−01
GA-dropout (Chen et al., 2018)	94.64 ± 2.40 1.372e−03	96.31 ± 0.13 1.229e−12	68.05 ± 0.07 1.059e−18	99.40 ± 0.15 6.824e−02	97.86 ± 0.91 3.216e−03	92.09 ± 0.63 3.806e−04	97.96 ± 0.31 8.022e−13
DE-dropout (Chen et al., 2018)	92.86 ± 3.19 2.085e−04	96.71 ± 0.32 6.623e−05	68.20 ± 0.15 2.010e−13	99.41 ± 0.07 1.097e−02	97.86 ± 0.88 2.734e−03	92.21 ± 0.50 2.133e−04	97.96 ± 0.31 1.037e−12
GADropout	98.21 ± 1.79 (8×10^2)	97.27 ± 0.11 (8×10^4)	69.20 ± 0.06 (9.62×10^4)	99.51 ± 0.09 (3.26×10^5)	99.15 ± 0.78 (4×10^3)	92.98 ± 0.15 (5.46×10^4)	99.89 ± 0.16 (2.54×10^4)

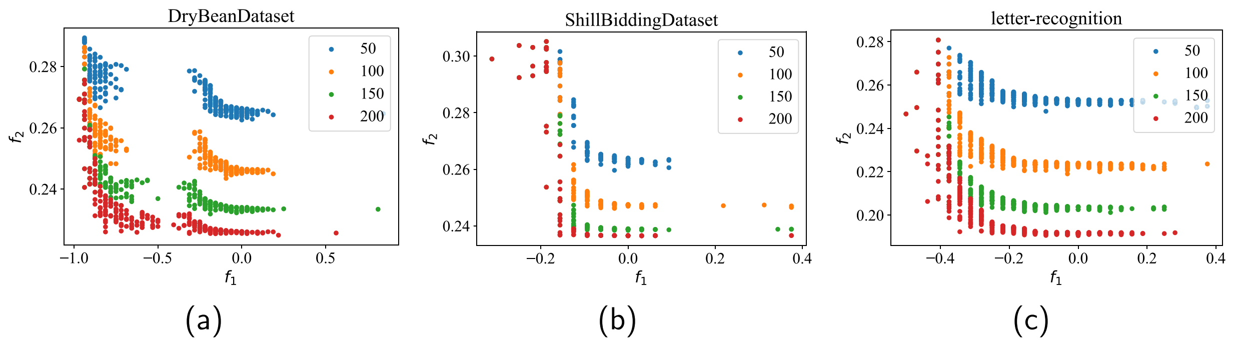

Fig. 3. The 50th, 100th, 150th and 200th populations.

the experiment on deep neural network, all the experiments trained 200 epochs with a learning rate of 0.01, and all the experiments on Lenet train 300 Epochs with the learning rate of 0.1 and scaled down 0.1 times at 150th and 225th. Experimental results are shown in Table 2. As shown, the GADropout method achieves good results on small-scale data being applied to both MLP and convolutional neural networks.

3.5. Performances on SVHN & CIFAR

In this section, GADropout was applied to the relevant experiments of two classical convolutional neural networks. VGG16 and ResNet18 were selected for comparison of algorithm effectiveness. These two networks are currently the most widely-used convolutional neural networks, which may to some extent explain the effectiveness of GADropout method.

First, SVHN datasets were used, and six algorithms were selected for comparison. For all methods, the batch size was set to 256 during training. As shown in Table 3, GADropout performs only 0.03% less than the best method on ResNet18, but 0.06% better than the best method on VGG16.

In addition, experiments were conducted on the CIFAR-10 and CIFAR-100 datasets, and twelve methods were used to compare with GADropout. In this experiment, the batch size was set to 128. As shown in Table 4, after the training process with ResNet18, GADropout improved 0.11% and 0.12% over the best method on CIFAR-10 and CIFAR-100 datasets, respectively. Compared with the best method, VGG16 with GADropout improved 0.37% and 0.18% on CIFAR-10 and CIFAR-100 datasets, respectively.

3.6. Performances on mini-Imagenet & Imagenet-32 × 32

To verify the effectiveness of the GADropout method on large datasets, two subsets of the Imagenet dataset were selected. First, on the mini-Imagenet dataset, the batch size was set to 256, and twelve

Table 2
Classification accuracy (%) on MNIST & Fashion MNIST. The total number of evaluation for all of the comparison is 5.64×10^4.

	MNIST		Fashion MNIST	
	MLP	Lenet	MLP	Lenet
No dropout	98.25 ± 0.09 5.067e−14	98.84 ± 0.08 8.019e−13	89.12 ± 0.07 5.175e−19	89.58 ± 0.14 1.382e−14
Standard dropout (Hinton et al., 2012)	98.49 ± 0.06 9.381e−13	98.76 ± 0.14 2.301e−10	89.37 ± 0.14 6.116e−12	89.75 ± 0.23 8.123e−10
Gaussian dropout (Srivastava et al., 2014)	98.47 ± 0.08 7.477e−12	98.82 ± 0.18 5.047e−08	89.38 ± 0.24 3.429e−08	89.73 ± 0.11 4.777e−15
Variational dropout (Kingma et al., 2015)	98.39 ± 0.15 1.615e−09	98.69 ± 0.11 8.976e−13	89.57 ± 0.10 7.874e−12	89.76 ± 0.18 2.077e−11
DropConnect (Wan et al., 2013)	98.48 ± 0.05 1.579e−13	98.86 ± 0.03 6.502e−17	89.57 ± 0.14 1.388e−09	89.78 ± 0.13 1.785e−13
Concrete dropout (Gal et al., 2017)	98.43 ± 0.12 2.635e−10	98.75 ± 0.24 4.457e−07	89.63 ± 0.15 2.684e−08	89.77 ± 0.22 5.937e−10
Fraternal dropout (Zolna et al., 2018)	98.37 ± 0.12 4.543e−11	98.78 ± 0.06 2.711e−15	89.44 ± 0.18 1.775e−09	89.61 ± 0.22 3.439e−11
Gradient dropout (Tseng et al., 2020)	98.55 ± 0.20 7.363e−06	98.91 ± 0.04 3.942e−15	89.82 ± 0.17 1.555e−04	89.87 ± 0.10 2.295e−14
Meta dropout (Lee et al., 2020)	98.69 ± 0.20 6.888e−04	98.82 ± 0.10 9.584e−12	89.71 ± 0.05 9.099e−14	89.94 ± 0.09 2.727e−14
Auto dropout (Pham and Le, 2021)	98.87 ± 0.05 1.856e−03	99.01 ± 0.03 4.609e−14	89.95 ± 0.13 6.437e−03	90.35 ± 0.02 4.100e−13
Advance dropout (Xie et al., 2022)	98.89 ± 0.08 4.001e−02	99.02 ± 0.03 7.848e−14	89.85 ± 0.11 5.234e−06	90.44 ± 0.07 2.955e−06
GA-dropout (Chen et al., 2018)	98.57 ± 0.12 3.211e−08	98.73 ± 0.13 3.281e−11	89.59 ± 0.09 3.075e−12	90.01 ± 0.13 4.315e−11
DE-dropout (Chen et al., 2018)	98.64 ± 0.09 2.466e−08	98.63 ± 0.07 2.979e−16	89.79 ± 0.06 6.025e−11	90.05 ± 0.08 1.147e−13
GADropout	98.96 ± 0.06	99.34 ± 0.04	90.08 ± 0.03	90.61 ± 0.04

Table 3
Classification accuracy (%) on SVHN dataset. The total number of evaluation for all of the comparison is 6.87×10^4.

	ResNet18	VGG16
No dropout	97.71 ± 0.21 2.280e−03	97.92 ± 0.19 1.620e−03
Standard dropout (Hinton et al., 2012)	97.89 ± 0.16 1.130e−02	98.02 ± 0.23 1.630e−02
Gaussian dropout (Srivastava et al., 2014)	97.87 ± 0.11 1.320e−03	98.05 ± 0.15 4.260e−03
Uniform dropout (Shen et al., 2018)	98.08 ± 0.03 8.050e−02	98.10 ± 0.16 1.410e−02
β-Dropout (Liu et al., 2019)	98.04 ± 0.09 7.250e−02	98.11 ± 0.09 2.120e−03
Advanced dropout (Xie et al., 2022)	98.17 ± 0.08 5.210e−01	98.29 ± 0.06 2.170e−01
GADropout	98.14 ± 0.06	98.35 ± 0.08

methods were used for comparison in VGG16 and ResNet18. On the mini-Imagenet dataset, all individuals in history were evaluated a total of 4.5×10^4 times. As illustrated in Table 5, GADropout improves accuracy by 0.67% and Top-5 accuracy by 0.45% for ResNet18. On VGG16, the proposed method improves accuracy by 0.61% and Top-5 accuracy by 0.45%.

Finally, GADropout was used in the relevant experiments of Imagenet-32 × 32 dataset, and the same comparison algorithm was adopted. After 3.003×10^5 evaluations for all historical individuals, the training stage was finally completed on ResNet18 and VGG16. As shown in Table 6, GADropout achieves good results on both classical neural networks. In the comparison experiment of ResNet18, compared with the existing best method, the accuracy of the proposed method is improved by 1.13%, and the accuracy of Top-5 is improved by 0.27%. In the comparison experiment of VGG16, the accuracy and Top-5 accuracy of the proposed algorithm on the dataset are better than the existing best method by 0.69% and 0.2% respectively.

On the Imagenet-32 × 32 dataset, the training time of each epoch was compared. As shown in Table 6, GADropout needs to perform verification on the verification set before and after training each epoch, so the training time is affected by calculation efficiency and data reading efficiency, which leads to the algorithm spending an additional 10%–20% of the original training time.

4. Conclusion

Deep neural network performs very well in classification tasks. However, the phenomenon of over-fitting is very likely to appear in the training process. The existing dropout method cannot automatically adjust the deactivation probability of each neuron. Therefore, we proposed an improved dropout technique that uses an evolutionary approach to guide the process of random neuron inactivation. In the process of population iteration, we used a multi-objective optimisation strategy to control and select the searched probabilities. In addition to a genetic algorithm, this method integrates the training stage and evaluation process, thus ensuring no increase in time complexity.

The proposed method achieves good results on popular classification datasets and various sizes of data. Compared to existing dropout methods, GADropout can significantly improve the accuracy of the network for classification tasks. Our results demonstrate that the dropout probability has a variation pattern in the process of population evolution. In future work, understanding and proving the basis of this phenomenon will be very meaningful in better explaining dropout's effectiveness. In our experiments, we used generic popular architectures for computer vision. Future work will include an extension of GADroput to larger modern structures, including, for example, attention networks and vision transform networks. In addition, we plan to further improve the modelling of the optimisation problem by defining more meaningful objectives.

Table 4
Classification accuracy (%) on CIFAR-10 & CIFAR-100. The total number of evaluation for all of the comparison is 9.39×10^4.

	CIFAR-10		CIFAR-100	
	ResNet18	VGG16	ResNet18	VGG16
No dropout	94.83 ± 0.21 3.390e−05	93.86 ± 0.13 1.740e−06	76.44 ± 0.23 7.620e−07	73.62 ± 0.24 9.120e−07
Standard dropout (Hinton et al., 2012)	94.84 ± 0.23 6.890e−05	93.81 ± 0.08 6.670e−08	76.67 ± 0.04 2.450e−10	73.77 ± 0.16 1.280e−07
Gaussian dropout (Srivastava et al., 2014)	95.02 ± 0.12 5.920e−06	93.83 ± 0.10 2.670e−07	76.62 ± 0.24 2.830e−06	73.78 ± 0.22 1.180e−06
Uniform dropout (Shen et al., 2018)	94.86 ± 0.16 6.950e−06	93.82 ± 0.11 4.250e−07	76.76 ± 0.07 3.160e−09	73.76 ± 0.19 3.820e−07
Concrete dropout (Gal et al., 2017)	94.99 ± 0.21 1.640e−04	93.79 ± 0.11 3.230e−07	76.48 ± 0.20 3.470e−07	73.67 ± 0.06 4.680e−10
Variational dropout (Kingma et al., 2015)	95.05 ± 0.20 2.350e−04	93.81 ± 0.08 6.670e−08	76.76 ± 0.20 1.880e−06	73.98 ± 0.24 7.340e−06
β-Dropout (Liu et al., 2019)	95.07 ± 0.06 2.300e−07	93.95 ± 0.15 1.070e−05	76.79 ± 0.24 8.280e−06	74.03 ± 0.05 2.600e−09
Continuous dropout (Shen et al., 2018)	94.92 ± 0.24 1.930e−04	93.86 ± 0.13 1.740e−06	76.90 ± 0.12 1.700e−07	73.85 ± 0.12 3.430e−08
Information dropout (Achille and Soatto, 2018)	94.97 ± 0.15 1.420e−05	93.88 ± 0.19 2.500e−05	76.47 ± 0.12 1.020e−08	73.70 ± 0.04 1.900e−10
Gaussian Soft dropout (Xie et al., 2019)	95.09 ± 0.19 2.750e−04	93.97 ± 0.12 3.310e−06	77.22 ± 0.10 1.140e−06	74.07 ± 0.07 1.060e−08
Laplace Soft dropout (Xie et al., 2019)	95.03 ± 0.10 2.140e−06	93.95 ± 0.09 5.090e−07	77.13 ± 0.16 8.020e−06	74.05 ± 0.24 1.180e−05
Advanced dropout (Xie et al., 2022)	95.52 ± 0.03 2.920e−03	94.28 ± 0.03 1.740e−06	77.78 ± 0.04 5.860e−03	74.94 ± 0.03 7.420e−04
GADropout	95.63 ± 0.05	94.65 ± 0.06	77.90 ± 0.06	75.12 ± 0.07

Table 5
Classification accuracy (%) on mini-Imagenet dataset. The total number of evaluation for all of the comparison is 4.5×10^4.

	ResNet18		VGG16	
	Top-1	Top-5	Top-1	Top-5
No dropout	71.80	91.16	76.35	93.05
Standard dropout (Hinton et al., 2012)	72.02	92.38	76.21	92.86
Gaussian dropout (Srivastava et al., 2014)	71.98	91.75	76.14	92.89
Uniform dropout (Shen et al., 2018)	72.07	91.99	76.84	93.39
Concrete dropout (Gal et al., 2017)	71.56	91.67	76.35	93.07
Variational dropout (Kingma et al., 2015)	72.06	92.04	76.56	93.32
β-Dropout (Liu et al., 2019)	72.24	92.89	77.13	93.98
Continuous dropout (Shen et al., 2018)	72.33	93.47	76.71	93.67
Information dropout (Achille and Soatto, 2018)	71.90	92.01	76.44	93.12
Gaussian Soft dropout (Xie et al., 2019)	71.74	91.88	76.56	93.23
Laplace Soft dropout (Xie et al., 2019)	71.55	91.63	76.61	93.76
Advanced dropout (Xie et al., 2022)	72.89	93.56	77.35	94.11
GADropout	73.56	94.01	77.96	94.56

Table 6
Classification accuracy (%) on Imagenet-32 × 32 dataset. The total number of evaluation for all of the comparison is 3.003×10^5. Time is the training seconds during each epoch.

	ResNet18			VGG16		
	Top-1	Top-5	Time	Top-1	Top-5	Time
No dropout	45.46	70.47	556	40.58	64.50	407
Standard dropout (Hinton et al., 2012)	45.72	70.73	572	41.26	64.14	412
Gaussian dropout (Srivastava et al., 2014)	45.00	69.74	580	41.33	64.22	438
Uniform dropout (Shen et al., 2018)	45.26	69.62	573	41.36	64.88	416
Concrete dropout (Gal et al., 2017)	45.75	70.55	565	41.37	64.33	426
Variational dropout (Kingma et al., 2015)	45.53	69.63	582	41.45	64.35	447
β-Dropout (Liu et al., 2019)	45.09	69.73	697	41.51	64.27	606
Continuous dropout (Shen et al., 2018)	44.60	69.39	560	41.23	64.07	421
Information dropout (Achille and Soatto, 2018)	45.55	70.31	575	41.35	64.12	413
Gaussian Soft dropout (Xie et al., 2019)	46.33	71.04	587	41.69	64.59	419
Laplace Soft dropout (Xie et al., 2019)	46.31	71.14	583	41.57	65.45	426
Advanced dropout (Xie et al., 2022)	47.03	71.75	582	42.65	66.23	420
GADropout	48.16	72.02	642	43.34	66.43	483

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (61876089, 61876185, 61902281), the Natural Science Foundation of Jiangsu Province, China (BK20141005) and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (14KJB520025).

References

Achille, A., Soatto, S., 2018. Information dropout: Learning optimal representations through noisy computation. IEEE Trans. Pattern Anal. Mach. Intell. 40 (12), 2897–2905.

Ba, J., Frey, B., 2013. Adaptive dropout for training deep neural networks. In: Advances in Neural Information Processing Systems, Vol. 26.

Chandra, A., Yao, X., 2006. Evolving hybrid ensembles of learning machines for better generalisation. Neurocomputing 69 (7), 686–700.

Chen, T., Jia, W., Sun, Y., 2018. The improvement of dropout strategy based on two evolutionary algorithms. In: 2018 IEEE International Conference on Robotics and Biomimetics. pp. 814–819.

Choe, J., Lee, S., Shim, H., 2021. Attention-based dropout layer for weakly supervised single object localization and semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 43 (12), 4256–4271.

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197.

Deb, K., Tiwari, S., 2008. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. European J. Oper. Res. 185 (3), 1062–1087.

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

Eiben, A.E., Smith, J.E., 2015. Introduction to Evolutionary Computing, second ed. In: Natural Computing Series, Springer.

Gal, Y., Ghahramani, Z., 2016. A theoretically grounded application of dropout in recurrent neural networks. In: Advances in Neural Information Processing Systems, Vol. 29.

Gal, Y., Hron, J., Kendall, A., 2017. Concrete dropout. In: Advances in Neural Information Processing Systems, Vol. 30.

Garbin, C., Zhu, X., Marques, O., 2020. Dropout vs. batch normalization: an empirical study of their impact to deep learning. Multimedia Tools Appl. 79 (19–20), 12777–12815.

Ghiasi, G., Lin, T.-Y., Le, Q.V., 2018. DropBlock: A regularization method for convolutional networks. In: Advances in Neural Information Processing Systems, Vol. 31.

Guo, D., Wang, X., Gao, K., Jin, Y., Ding, J., Chai, T., 2022. Evolutionary optimization of high-dimensional multiobjective and many-objective expensive problems assisted by a dropout neural network. IEEE Trans. Syst. Man Cybern. Syst. 52 (4), 2084–2097.

Hayou, S., Ayed, F., 2021. Regularization in ResNet with stochastic depth. In: Advances in Neural Information Processing Systems, Vol. 34. pp. 15464–15474.

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

Hou, S., Wang, Z., 2019. Weighted channel dropout for regularization of deep convolutional neural network. Proc. AAAI Conf. Artif. Intell. 33 (01), 8425–8432.

Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q., 2016. Deep networks with stochastic depth. In: European Conference on Computer Vision. pp. 646–661.

Kingma, D.P., Salimans, T., Welling, M., 2015. Variational dropout and the local reparameterization trick. In: Advances in Neural Information Processing Systems, Vol. 28.

Kokalis, C.-C.A., Tasakos, T., Kontargyri, V.T., Siolas, G., Gonos, I.F., 2020. Hydrophobicity classification of composite insulators based on convolutional neural networks. Eng. Appl. Artif. Intell. 91, 103613.

Krogh, A., Hertz, J., 1991. A simple weight decay can improve generalization. In: Advances in Neural Information Processing Systems, Vol. 4.

Krueger, D., Maharaj, T., Kramár, J., Pezeshki, M., Ballas, N., Ke, N.R., Goyal, A., Bengio, Y., Courville, A.C., Pal, C.J., 2017. Zoneout: Regularizing RNNs by randomly preserving hidden activations. In: International Conference on Learning Representations.

Lakshminarayanan, B., Pritzel, A., Blundell, C., 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In: Advances in Neural Information Processing Systems, Vol. 30.

Lee, H.B., Nam, T., Yang, E., Hwang, S.J., 2020. Meta dropout: Learning to perturb latent features for generalization. In: International Conference on Learning Representations.

Liu, L., Luo, Y., Shen, X., Sun, M., Li, B., 2019. β -Dropout: A unified dropout. IEEE Access 7, 36140–36153.

Loshchilov, I., Hutter, F., 2019. Decoupled weight decay regularization. In: International Conference on Learning Representations.

Lu, Z., Whalen, I., Dhebar, Y., Deb, K., Goodman, E.D., Banzhaf, W., Boddeti, V.N., 2021. Multiobjective evolutionary design of deep convolutional neural networks for image classification. IEEE Trans. Evol. Comput. 25 (2), 277–291.

Ma, L., Ma, Y., Lin, Q., Ji, J., Coello, C.A.C., Gong, M., 2022. SNEGAN: Signed network embedding by using generative adversarial nets. IEEE Trans. Emerg. Top. Comput. Intell. 6 (1), 136–149.

Moon, T., Choi, H., Lee, H., Song, I., 2015. Rnndrop: A novel dropout for RNNs in ASR. In: 2015 IEEE Workshop on Automatic Speech Recognition and Understanding. pp. 65–70.

Morerio, P., Cavazza, J., Volpi, R., Vidal, R., Murino, V., 2017. Curriculum dropout. In: 2017 IEEE International Conference on Computer Vision. pp. 3564–3572.

Park, S., Kwak, N., 2017. Analysis on the dropout effect in convolutional neural networks. In: Proceedings of the Asian Conference on Computer Vision. pp. 189–204.

Park, S., Park, J., Shin, S.-J., Moon, I.-C., 2018. Adversarial dropout for supervised and semi-supervised learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32.

Park, S., Song, K., Ji, M., Lee, W., Moon, I.-C., 2019. Adversarial dropout for recurrent neural networks. Proc. AAAI Conf. Artif. Intell. 33 (01), 4699–4706.

Pham, H., Le, Q.V., 2021. AutoDropout: Learning dropout patterns to regularize deep networks. Proc. AAAI Conf. Artif. Intell. 35 (11), 9351–9359.

Rennie, S.J., Goel, V., Thomas, S., 2014. Annealed dropout training of deep networks. In: 2014 IEEE Spoken Language Technology Workshop. pp. 159–164.

Saito, K., Ushiku, Y., Harada, T., Saenko, K., 2018. Adversarial dropout regularization. In: International Conference on Learning Representations.

Salehinejad, H., Valaee, S., 2022. EDropout: Energy-based dropout and pruning of deep neural networks. IEEE Trans. Neural Netw. Learn. Syst. 33 (10), 5279–5292.

Semeniuta, S., Severyn, A., Barth, E., 2016. Recurrent dropout without memory loss. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. pp. 1757–1766.

Shen, X., Tian, X., Liu, T., Xu, F., Tao, D., 2018. Continuous dropout. IEEE Trans. Neural Netw. Learn. Syst. 29 (9), 3926–3937.

Shi, B., Zhang, D., Dai, Q., Zhu, Z., Mu, Y., Wang, J., 2020. Informative dropout for robust representation learning: A shape-bias perspective. In: Proceedings of the 37th International Conference on Machine Learning, Vol. 119. pp. 8828–8839.

Singh, S., Hoiem, D., Forsyth, D., 2016. Swapout: Learning an ensemble of deep architectures. In: Advances in Neural Information Processing Systems, Vol. 29.

Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1929–1958.

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

Tseng, H.-Y., Chen, Y.-W., Tsai, Y.-H., Liu, S., Lin, Y.-Y., Yang, M.-H., 2020. Regularizing meta-learning via gradient dropout. In: Proceedings of the Asian Conference on Computer Vision.

Valcarce, D., Landin, A., Parapar, J., Barreiro, A., 2019. Collaborative filtering embeddings for memory-based recommender systems. Eng. Appl. Artif. Intell. 85, 347–356.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R., 2013. Regularization of neural networks using DropConnect. In: Proceedings of the 30th International Conference on Machine Learning, Vol. 28. pp. 1058–1066.

Wang, S.I., Manning, C.D., 2013. Fast dropout training. In: International Conference on Machine Learning. pp. 118–126.

Wu, H., Gu, X., 2015. Towards dropout training for convolutional neural networks. Neural Netw. 71, 1–10.

Xie, J., Ma, Z., Lei, J., Zhang, G., Xue, J.-H., Tan, Z.-H., Guo, J., 2022. Advanced dropout: A model-free methodology for Bayesian dropout optimization. IEEE Trans. Pattern Anal. Mach. Intell. 44 (9), 4605–4625.

Xie, J., Ma, Z., Zhang, G., Xue, J.-H., Tan, Z.-H., Guo, J., 2019. Soft dropout and its variational Bayes approximation. In: 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing. pp. 1–6.

Xue, Y., Cai, X., Neri, F., 2022a. A multi-objective evolutionary algorithm with interval based initialization and self-adaptive crossover operator for large-scale feature selection in classification. Appl. Soft Comput. 127, 109420.

Xue, Y., Jiang, P., Neri, F., Liang, J., 2021a. A multi-objective evolutionary approach based on Graph-in-Graph for neural architecture search of convolutional neural networks. Int. J. Neural Syst. 31 (09), 2150035.

Xue, Y., Tang, Y., Xu, X., Liang, J., Neri, F., 2022b. Multi-objective feature selection with missing data in classification. IEEE Trans. Emerg. Top. Comput. Intell. 6 (2), 355–364.

Xue, Y., Tong, Y., Neri, F., 2022c. An ensemble of differential evolution and adam for training feed-forward neural networks. Inform. Sci. 608, 453–471.

Xue, Y., Zhang, Q., Neri, F., 2021b. Self-adaptive particle swarm optimization-based echo state network for time series prediction. Int. J. Neural Syst. 31 (12), 2150057.

Xue, Y., Zhu, H., Neri, F., 2021c. A self-adaptive multi-objective feature selection approach for classification problems. Integr. Comput.-Aided Eng. 29 (1), 3–21.

Yin, L., Chen, L., Liu, D., Huang, X., Gao, F., 2021. Quantum deep reinforcement learning for rotor side converter control of double-fed induction generator-based wind turbines. Eng. Appl. Artif. Intell. 106, 104451.

Zhang, H., Jin, Y., Hao, K., 2022. Evolutionary search for complete neural network architectures with partial weight sharing. IEEE Trans. Evol. Comput. 26 (5), 1072–1086.

Zhang, K., Wang, W., Lv, Z., Fan, Y., Song, Y., 2021. Computer vision detection of foreign objects in coal processing using attention CNN. Eng. Appl. Artif. Intell. 102, 104242.

Zhou, Y., Yen, G.G., Yi, Z., 2021. A knee-guided evolutionary algorithm for compressing deep neural networks. IEEE Trans. Cybern. 51 (3), 1626–1638.

Zhou, Y., Yen, G.G., Yi, Z., 2022. Evolutionary shallowing deep neural networks at block levels. IEEE Trans. Neural Netw. Learn. Syst. 33 (9), 4635–4647.

Zolna, K., Arpit, D., Suhubdy, D., Bengio, Y., 2018. Fraternal dropout. In: International Conference on Learning Representations.